Javarobot för att identifiera HTML-länkar utan referenser.

(HS-IDA-EA-97-114)

Christer Lundberg (a94chrlu@ida.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det datavetenskapliga programmet under vårterminen 1997.

Handledare: Tom Ziemke

Javarobot för att identifiera HTML-länkar utan referenser.

Examensrapport inlämnad av Christer Lundberg till Högskolan i Skövde, för Kandidatexamen (BSc) vid Institutionen för Datavetenskap.

970812

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Javarobot för att identifiera HTML-länkar utan referenser.

Christer Lundberg (a94chrlu@ida.his.se)

Key words: Software Agents, Internet, World-Wide Web, WWW, HTML, Java, Links

Abstract

This work is about a software-agent which localises HTML-links on the World Wide Web that have no correct references. The report contains a short background on the Internet, World Wide Web, software-agents and Java. The problem can be solved by using a couple of strategies. Three of them are deepth-first, breadth-first and a multiple software-agent. The choise fell at the breadth-first strategy The software-agent that was implemented did not reached all the goals that was set up at the beginning of the project. It does not meets the goal to be fully recursive at one choosen domain. It only seeks for two levels at the domain. Nevertheless it seems perfectly possible to implement this software-agent in such a way that it should reach the goals that was set up.

Innehållsförteckning

Figurförteckning

Sammanfattning

World-Wide Web (WWW) skapades 1989 av Tim Berners-Lee vid CERN. Detta utgör en del av Internet som funnits sedan långt tillbaka. Här kan privatpersoner, skolor, företag, myndigheter och många andra lägga upp information i form av hemsidor. Dessa hemsidor kan innehålla allt från bara namn och adress till avancerade försäljningssidor. Dessa sidor kan sedan nås av de som har tillgång till Internet eller WWW.

Principen för publicering av information på Internet är att den skall vara fri och tillgänglig för alla. Detta har lett till att det finns miljontals sidor på detta Internet som innehåller mer eller mindre viktig information och som ofta länkar till andra sidor på nätet.

Många av sidorna ligger ofta på samma plats under lång tid och orsakar inte några större problem för de som letar information på Internet. Men det är även många sidor som flyttar mellan olika adresser och en del sidor läggs helt enkelt ned. Detta ger som resultat att de länkar (referenser) som går till den sida som har flyttat eller inte finns kvar alls längre blir ogiltiga. Den informationssökande får då ett meddelande om att denna sida inte finns tillgänglig. Eftersom det är många sidor som flyttar fram och tillbaka eller helt enkelt läggs ned händer detta relativt ofta. Detta leder till irritation för den som söker information på Internet.

Det sätt som finns för att komma undan detta problem är att varje person som är ansvarig för en sida håller sina länkar uppdaterade. Detta kan göras manuellt men kan vara väldigt tidskrävande, även för en ganska liten sida. Istället skulle man kunna tänka sig att en mjukvarurobot kontrollerar vilka länkar som är giltiga och meddelar den ansvarige för respektive sida om vilka länkar som inte är giltiga. Detta skulle underlätta underhållsarbetet för dessa sidor avsevärt och även minska irritation hos de besökare som kommer till en sida för att söka efter information.

Det finns olika strategier för hur en sådan mjukvarurobot skulle kunna arbeta. Den skulle kunna arbeta efter en djupet först eller bredden först strategi. Djupet först innebär att mjukvaruroboten skulle ta den första länken och följa den till nästa sida och sedan första länk på den nya sedan och följa den tills man har nått ett stoppkriterium. När den nått detta stoppkriterium vänder den upp igen och tar nästa länk och följer denna nedåt. Bredden först innebär att mjukvaruroboten följer den första länken ett steg och sedan tillbaka för att ta nästa länk som den följer ett steg. När alla länkar på en sida har följts upp följer man de länkar som finns på de sidor som de länkar man följde förut på samma sätt.

Då sökdjupet, den rymd som går att nå från alla kontrollerade länkar, kan bli näst intill oändligt måste man ha ett kriterium för hur långt sökningen skall få fortsätta. Ett förslag är att man enbart följer länkar som finns inom en viss domän, tex högskolans sidor (http://www.his.se/).

Den sökstrategi som valdes blev bredden först på grund av uteslutningsmetoden då de andra strategierna föll ifrån.

De komponenter som behövdes för att implementera en mjukvarurobot av detta slag var en mekanism som drar ut alla HTML-länkar ur ett hypertextdokument, något som kontrollerar länkarnas status och en uppsättning listor för att hålla reda på vilka sidor

som skall kontrolleras, vilka som är kontrollerade samt vilka länkar som var bra respektive dåliga. Dessa komponenter tillverkades och sattes sedan ihop till en helhet.

Den ihopsatta mjukvaruroboten visade sig dock inte fungera fullt ut. Det fanns begränsningar i det antal nivåer som mjukvaruroboten söker på. Dock hittar den majoriteten av länkarna på de sidor som besöks och av de länkar som hittas kontrolleras alla samt får en svarskod och ett svarsmeddelande.

Det är dock möjligt att modifiera mjukvaruroboten så att den skulle kunna fungera på det sätt som var tänkt från början.

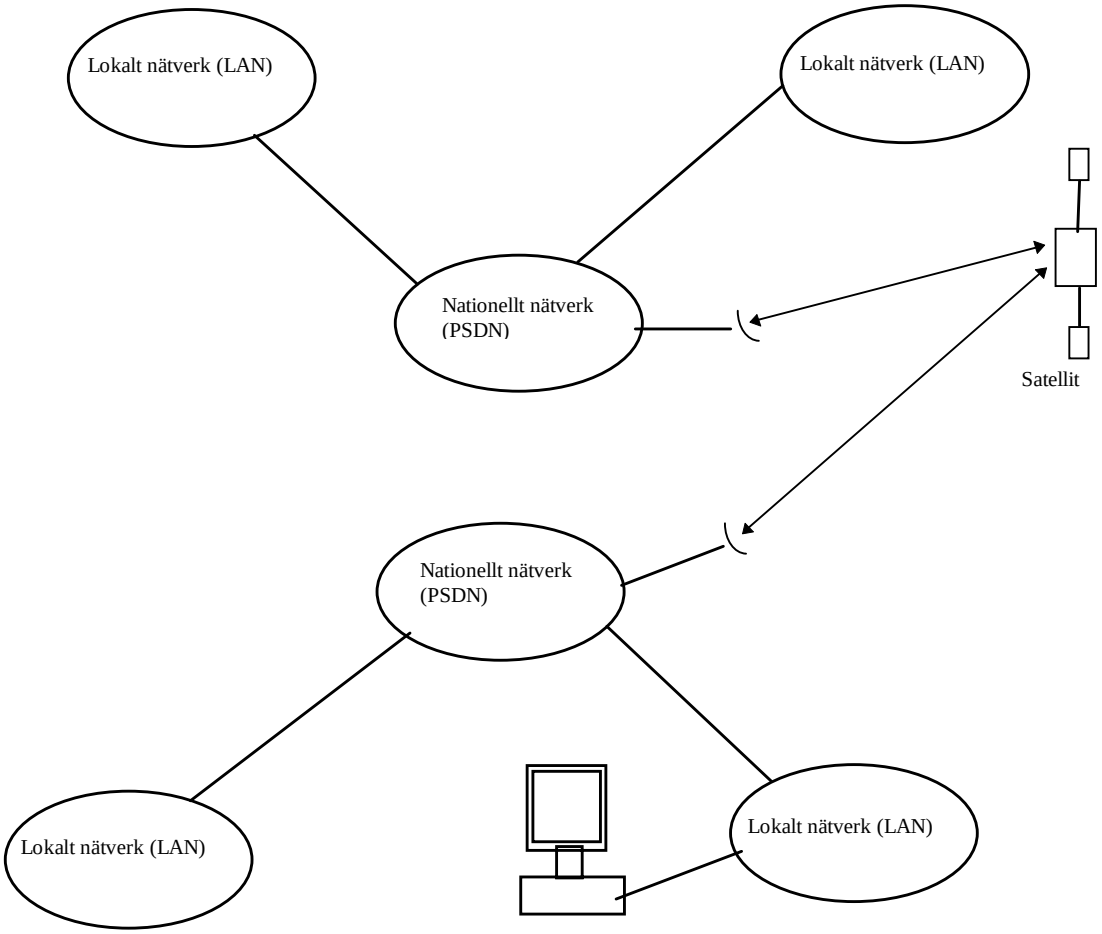
1 Bakgrund

För att få en ökad förståelse av problemet kommer här en bakgrundsbeskrivning av de olika huvuddelarna som berörs i detta arbete.

I bakgrundsbeskrivningen kommer en beskrivning och en kort historik om vad Internet, World-Wide Web, Mjukvarurobotar och Java är, i nämnd ordning.

1.1 Internet

Internet är ett världsomspännande nätverk (se Figur 1) av andra nätverk och enskilda datorer [Hal94]. Nätverken kan i sin tur bestå av andra nätverk. Nätverken finns i alla former från små lokala nätverk till större mer komplicerade nätverk. De som administrerar dessa är allt ifrån militära- och statliga myndigheter, skolor, kommersiella företag och privatpersoner.



Figur 1 Principskiss av Internet

Internet var ursprungligen ett amerikanskt experiment, från den militära försvarsdepartementet, designat med redundans för att kunna klara stora påfrestningar i händelse av krig utan att fallera, tex bombning av städer. Det kallades då för

ARPAnet. För att klara kommunikationen använde man Internet Protocol (IP). Detta kommunikationsprotokoll blev populärt över hela världen tack vare att det både är robust och enkelt.

Senare behövde man fler protokoll och dessa samlades i en svit i vad som dagligt tal kallas Transfer Control Protocol/Internet Protocol (TCP/IP). TCP/IP associeras idag med ett flertal protokoll för olika typer av dataöverföring [Hal94], [Dec94]. Dessa är:

- File Transfer Protocol (FTP)
- Remote Terminal Protocol (TELNET)
- Simple Mail Transfer Protocol (SMTP)
- Name Server Protocol (NSP)
- Simple Network Management Protocol (SNMP)
- Transfer Control Protocol (TCP)
- User Data Protocol (UDP)
- Internet Protocol (IP)

Under 1970- och 80-talet växte behovet utanför försvarsmaktens väggar på grund av att UNIX baserade arbetsstationer blev allt vanligare. De flesta använde en Berkleybaserad version av UNIX som har IP nätverksprotokoll som en del av operativsystemet. Universitet och forskningscentra utvecklade sådana system och ville ansluta sig till ARPAnet för att underlätta kommunikationen med andra universitet och forskningscentra.

En av dessa organisationer var National Science Foundation (NSF). De ville få sina fem statligt finansierade datacenters så åtkomliga som möjligt för den akademiska världen. Det visade sig att ARPAnet inte riktigt fyllde de krav man ställde så NSF bildade ett eget nätverk där dessa fem datacenters bands ihop med varandra. Detta nätverk bands sedan ihop med regionala universitet som i sin tur var bundna till andra universitet i deras regioner. På detta sätt kunde NSF binda ihop ett stort antal universitet i ett nätverk utan att behöva ansluta sig direkt till var och ett av dem.

Under slutet av 1980-talet blev nätet på NSFnet så överbelastat på grund av trafiken att man behövde uppgradera sitt nätverk. Man fick ett kontrakt med Merit Network Incorporation i samarbete med IBM och MCI. Nätverkets kapacitet växte med en faktor 20. Nu när man fått större bandvidd uppmuntrade NSF Universiteten att ge alla sina studenter tillgång till nätverket. Samtidigt ökade den internationella användningen av Internet snabbt eftersom att fler och fler Universitet världen över anslöt sig till nätet.

Under några år var Internet endast öppet för Universitet, högskolor och andra icke kommersiella organisationer världen över, men nyligen tillät man även kommersiella intressen på Internet. Detta ledde till att Internet expanderade explosionsartat eftersom att det nu blev möjligt för företag och privatpersoner att på ett enkelt och relativt billigt sätt annonsera och sälja sina varor och tjänster.

1.2 World-Wide Web (WWW)

World-Wide Web är en del av Internet och kan beskrivas som ett gigantisk bibliotek eller ett stort köpcentra. Här kan människor som har tillgång till Internet och någon

form av webläsare söka och ta del av information och även kommunicera med andra människor som finns anslutna. WWW är det grafiska gränssnittet mot Internet. Här finns så kallade hemsidor skrivna med hjälp av HyperText Markup Language (HTML) och de överförs med hjälp av HyperText Transmission Protocol (HTTP).

Under 1989 började Tim Berners-Lee vid CERN att ta fram det som vi idag kallar World-Wide Web. Forskarna vid CERN ville hitta ett enkelt och effektivt sätt att nå ut med information till sina kollegor både inom och utom organisationen. Det system för informationsförmedling som skulle byggas skulle använda hypertext (se Figur 2). HyperText är ett sätt att publicera information på ett nätverk på så sätt att alla som är anslutna till nätet kan ta del av denna. Hypertextsidorna skapades i HTML. Vid slutet av 1990 hade de en textbaserad prototyp färdig och i början av 1993 fanns den första grafiska webläsaren ute. Från början var det tänkt att endast hypertext skulle förmedlas över det nya systemet, men idag presenteras även hypermedia, såsom grafik, bilder, ljud, video och annat via WWW.

De huvudmål man satte upp för projektet var [Dec94]:

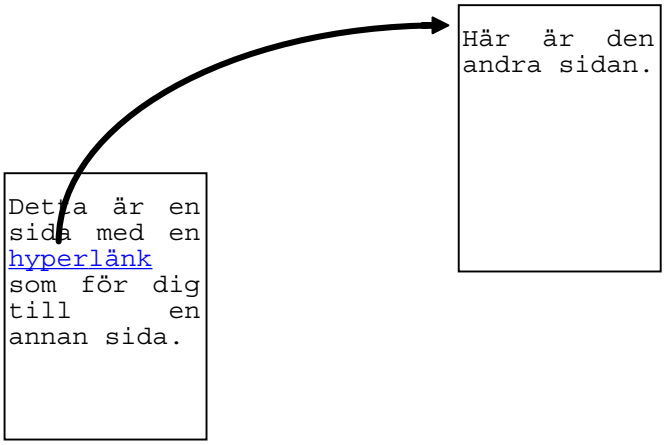
- Tillhandahålla ett gemensamt och enkelt protokoll för att begära information (läsbart av människor) som finns lagrat på ett fjärrsystem med hjälp av ett nätverk.
- Tillhandahålla ett protokoll för att automatiskt utbyta information i ett format som är gemensamt för både leverantör och mottagare.
- Tillhandahålla en metod som åtminstone gör det möjligt att läsa text på datorskärmen.
- Tillhandahålla och underhålla minst en dokumentsamling där användaren kan lägga upp sina egna dokument.
- Tillhandahålla någon form av sökmöjlighet på nyckelord. Resultatet presenteras i ett hypertextdokument med hypertextlänkar till de refererade sidorna..
- Tillåta att privata individuellt underhållna dokumentsamlingar länkas (refereras med hypertextlänkar) till andra dokumentsamlingar.
- Att använda public domain program där det är möjligt.
- Tillhandahålla programvara för att klara detta utan kostnad.

Tittar man närmare på dessa mål som var grundläggande i det första läget inser man snart att de flesta fortfarande gäller i allra högsta grad.

Hypertextsidor (se Figur 2) är sidor publicerade på WWW med hypertextlänkar som för besökaren till andra hypertextsidor på WWW. Adressen för dessa sidor kallas för Uniform Resource Locator (URL). URLen består av tre delar:

1. Vilket protokoll som skall användas.
2. Vilken adress (domän) som avses.
3. Sökvägen till objektet på aktuell adress.

Tex **http://[1]www.his.se[2]/ida/a94chrlu[3]**



Figur 2 Principskiss på hypertextsida med länk

Dessa hypertextsidor är skrivna i HTML (se Figur 3) som är en förenklad form av SGML (Standard Generalized Markup Language). SGML används för att göra dokument läsbara från många olika plattformar. Idag är man framme vid HTML 3.2 som har nästan samma möjligheter att presentera information som i ett vanligt ordbehandlingsprogram som Microsoft Word™. HTML fungerar på det sättet att man placerar ut en starttag och en sluttag som talar om för webläsaren hur det som finns mellan tagarna ska presenteras. Ett dokument börjar alltid med en HTML-tag som talar om att det är ett HTML dokument, sedan kommer en HEAD-tag där man kan placera titeln på dokumentet innanför en TITLE-tag. Därefter kommer en BODY-tag där man först kan bestämma vilken bakgrund som skall finnas på sidan samt vilken färg text och länkar skall ha. Därefter placerar man in den text man vill ha i dokumentet och om man vill ha bilder, tabeller och länkar. Detta formaterar man med olika taggar som talar om hur det ska presenteras. På exemplet i Figur 3 finns en rubriktag som gör en rubrik och en länktag som utgör en hypertextlänk. Resultatet kan ses i Figur 4.

```
<HTML>

<HEAD>

<TITLE>Exempel på HTML</TITLE>

</HEAD>

<BODY Background="" BGColor=#ffffff Text=#400040 Link=#0000ff VLink=#800080
ALink=#ff0000>

<H1>Detta är en rubrik</H1>

Detta är ett mycket litet exempel på hur ett HTML-dokument kan se ut.<BR>

<A href="http://www.his.se/" Alt="Detta är en förklaring">Detta är en
hypertextlänk till Högskolan i Skövde</A>

</BODY>

</HTML>
```

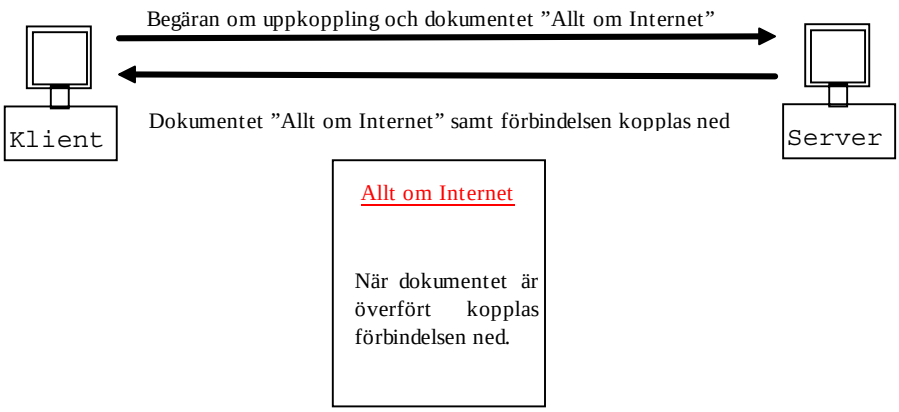
Figur 3 Exempel på HTML-kod för ett hypertextdokument

Figur 4 Hypertextdokument enligt det exempel på HTML-kod i föregående Figur 3.

För att komma åt dessa hypertextsidor som finns på WWW använder man olika webläsare så som Netscape, HotJava, Microsoft Internet Explorer, Mosaic, Lynx. Eller något av alla de andra verktyg som finns på marknaden. Dessa visar hypertextsidorna genom ett grafiskt gränssnitt (gäller inte Lynx som är textbaserad). För att överföra sidornas information mellan klient (webläsare) och server använder man HTTP (HyperText Transfer Protocol). Överföringen av sidorna mellan webläsare och server sker i fyra steg (se även Figur 5).

1. Uppkoppling
2. Begäran
3. Svar
4. Nedkoppling.

Först försöker klienten att koppla upp sig mot servern. Går inte detta avbryts normalt det hela av en time-out. När servern har svarat och klienten och servern har upprättat en kontakt begär klienten det valda objektet av servern. Begäran innehåller information om vilket protokoll som används, vilket objekt klienten vill ha och hur klienten vill att servern ska svara. Det vanligaste sättet för servern att svara på är GET och innebär helt enkelt att servern överför objektet till klienten. Servern svarar sedan på det sätt den har blivit ombedd på. Vanligtvis innebär det att servern börjar överföra det efterfrågade objektet till klienten. Nu presenterar webläsaren det efterfrågade objektet på det sätt som är lämpligt beroende på vilken form objektet har (normalt ett HTML dokument, gif- eller jpeg bild, postscript fil eller textdokument) och kopplar ned förbindelsen med servern (se Figur 5).



Figur 5 Principskiss över överföring av ett dokument på WWW

Förutom WWW arean har man därifrån även tillgång till FTP, TELNET, GOPHER, NEWS och MAIL areorna av Internet.

1.3 Mjukvarurobotar

En mjukvarurobot kan sägas vara ett program som utför en uppgift mer eller mindre självständigt. Dessa uppgifter kan variera kraftigt, allt från att utföra en specifik uppgift till kommunicera med andra mjukvarurobotar eller människor. Man talar ofta om Software-agents i dessa sammanhang.

Det finns många definitioner på vad en mjukvarurobot är, men man skulle kunna säga att en mjukvarurobot är ett system som finns i och är en del av en omgivning som den känner av och påverkar över tiden efter ett eget mönster och påverkar det som mjukvaruroboten känner av i framtiden. I princip kan man säga att en mjukvarurobot utför något som du själv skulle kunna utföra om du hade tid [Fra96].

Man kan dela in mjukvarurobotarna i olika klasser efter hur de arbetar eller vilken uppgift de har [Fra96].

- Reaktiv, överför indata till utdata i den miljö mjukvaruroboten arbetar.
- Autonom, bestämmer själv över vilka åtgärder den skall vidtaga och hur den ska lösa ett problem.
- Målorienterad, utför en specifik uppgift.
- Kontinuerlig, är en kontinuerligt rullande process.
- Kommunikativ, kommunicerar med andra mjukvarurobotar eller människor.
- Lärande, ändrar beteende beroende på den erfarenhet den har samlat in sedan tidigare.
- Mobil, kan flytta sig mellan olika maskiner.
- Flexibel, beteendet är inte bestämt på förhand.
- Karaktär, kan inta personlighetsdrag och känslomässiga tillstånd.

Några typiska arbetsområden för mjukvarurobotar skulle kunna vara:

- Söka efter information på WWW eller Internet.
- Sköta någons e-post automatiskt.
- Arrangera möten mellan personer med hjälp av en planeringskalender.
- Leta upp hypertextlänkar som saknar referenser.

Några områden som redan nu eller snart kommer att hanteras av mjukvarurobotar är:

- Underhåll och drift av system och nätverk.
- Mobil access och underhåll av information.
- Hantering av e-post.
- Samarbete som att ordna möten mellan människor när dessa har tid.
- Sköta arbetsscheman och administrativa sysslor.
- Elektronisk försäljning.
- Adaptiva användargränsnitt.

Det språk de flesta mjukvarurobotar använder för att kommunicera med varandra är Agent Communication Language (ACL) som i sin tur använder sig av Knowledge Interchange Format (KIF) och Knowledge Query and Manipulation Language (KQML). KIF och KQML används för att ställa frågor mot andra mjukvarurobotar och utbyta information mjukvarurobotar emellan [Www20], [Www21]. Ett programspråk som kan stödja detta är Java.

1.4 Java

Java är ett relativt nytt objektorienterat programmeringsspråk som till stora delar bygger på och påminner om C++. Java är plattformsoberoende och kan därmed köras på vilken dator som helst. Fördelen med detta är att användaren inte behöver bry sig om hans plattform stödjer Java och i så fall vilken version av Java som understöds.

Det hela började 1990 då Patrick Naughton som var programmerare vid Sun Microsystems tröttnade på den uppsjö av operativsystem som användes inom Sun [Www03], [Www04], [Www05]. Han hade fått ett erbjudande om jobb vid NeXT som han hade beslutat sig för att anta. Han berättade detta för sin chef Scott McNealy som då bad Naughton att göra en lista över vad han ansåg var fel på Sun.

När Naughton levererat sin lista till Sun visade det sig att det var många på Sun som höll med honom. Sun beslutade då att ge Naughton och några andra programmerare från Sun ett erbjudande att göra precis vad de ville med enda kravet att det skulle vara något häftigt. En av de andra var James Gosling.

Gruppen antog erbjudandet och antog namnet Green. De låste in sig i ett rum och diskuterade vad som de tyckte om och vad som de inte tyckte om med de elektroniska varor och prylar som fanns på marknaden. Sedan satte de upp ett stort antal elektriska apparater såsom TV-apparater, Nintendo Game Boys och fjärrkontroller för att se om de kunde hitta ett språk så att apparaterna kunde kommunicera med varandra.

De upptäckte att alla apparater hade olika processorer vilket ledde till att om en tillverkare ville lägga till funktioner i tex en TV var de beroende av vad hårdvaran och de inbyggda programmen tillät. Eftersom att de flesta apparater har ett mycket begränsat programmeringsutrymme fick de en idé om att skapa ett nytt

objektorienterat programspråk som skulle kunna ge nya dimensioner åt hårdvaruprogrammering. Detta nya språk döpte Gosling till Oak efter en ek som stod utanför hans fönster.

Språket var baserat på C++ , men det var bantat till ett minimum för att få plats i de begränsade chipsutrymmen som finns tillgängligt i små apparater. Detta tillät programmerarna att på ett enklare och på ett kraftfullt sätt förbättra och förändra hårdvaran.

Under de följande 18 månaderna, undersökte gruppen varför folk gillade vissa datorspel, och hur de reagerade på olika sorters elektronisk utrustning Efter att ha samlat ihop alla data från undersökningen, konstruerade de något som liknade en fjärrkontroll med en liten skärm. På skärmen visades en liten animerad figur som kallades Duke och som guidade användaren runt bland fjärrkontrollens funktioner. Gruppen ansåg att det viktigaste var att det de skapade skulle vara engagerande och roligt att använda.

I november 1992 bytte gruppen namn från Green till First Person och blev ett dotterbolag till Sun. Man bestämde sig för att inrikta sig på interaktiv television. Efter att ha fört förhandlingar med Time-Warner och 3DO och misslyckats med att komma fram till någon överenskommelse lade man ned det projektet.

I april 1993 släppte the National Center for Supercomputing Applications (NCSA) den första grafiska webläsaren för WWW och Internet. First Person beslutade sig då för att satsa på Internet istället. De beslutade sig för att Oak istället skulle bli ett språk för WWW och Internet.

När Oak var färdigutvecklat och redo för att släppas, beslutade sig Sun för att döpa om Oak till Java och att det skulle vara fritt. I maj 1995 på Sun World 95 släppte Sun Java och även webläsaren HotJava.

Java är relativt enkelt till sin natur och ligger ganska nära Pascal vad det gäller enkelheten i att lära och använda. Java är dock i dagsläget inte lika kraftfullt som C++ men kan komma att närma sig i framtiden.

Några nackdelar med Java jämfört med C++ är:

- Java kan inte överlagra operatorer.
- Java kan inte utnyttja multipelt arv utan kan endast ärva en klass.
- Java stödjer inte mallar.

Javakoden bygger på klasser som innehåller all information och funktionalitet.

Några nyheter som finns i Java är att varje källkodsfil innehåller endast en klass och källkodsfilen får samma namn som klassen. Filen skall också ha tilläget .java. Behovet av inkluderingsfiler är också borta eftersom att man kan anropa funktioner och referera variabler som definieras senare i koden. Pekarna har också försvunnit liksom strukturvariabler och objekt. Java använder garbage collection istället för att frigöra minne efter att man använt det som i C++. Java går istället igenom alla minnesobjekt och tar bort de som inte längre refereras.

Java definierar en binär standard som är processoroberoende och detta innebär att en kompilerad Javafil inte blir något exekverbart program utan måste tolkas av Java Virtual Machine (JVM) som talar om hur filen skall exekveras. JVM är dock beroende av vilken processor som används och måste därför finnas till alla

operativsystem. Arbetet med att ta fram JVM för alla operativsystem pågår för fullt och inom en snar framtid kommer det troligen att finnas en JVM för varje operativsystem. Nackdelen med JVM är att programmen utförs långsammare än de skulle gjorts annars men fördelen är att programmet kan köras överallt där de finns en JVM.

Java är på grund av sin portabilitet, multitrådning och objektorienterade karaktär lämpligt för att implementera mjukvarurobotar.

2 Problembeskrivning

2.1 Kort beskrivning av problemet

Principen för publicering av information på Internet är att den skall vara fri och tillgänglig för alla. Detta har lett till att det finns miljontals sidor på detta Internet som innehåller mer eller mindre viktig information och som länkar till andra sidor på nätet.

Många av sidorna ligger ofta på samma plats under långtid och orsakar inte några större problem för de som letar information på Internet. Men det är även många sidor som flyttar mellan olika adresser och en del sidor läggs helt enkelt ned. Detta ger som resultat att de länkar (referenser) som går till den sida som har flyttat eller inte finns kvar alls längre blir ogiltiga. Den informationssökande får då ett meddelande om att denna sida inte finns tillgänglig. Eftersom att det är många sidor som flyttar fram och tillbaka eller helt enkelt läggs ned händer detta relativt ofta. Detta leder till irritation för den som söker information på Internet.

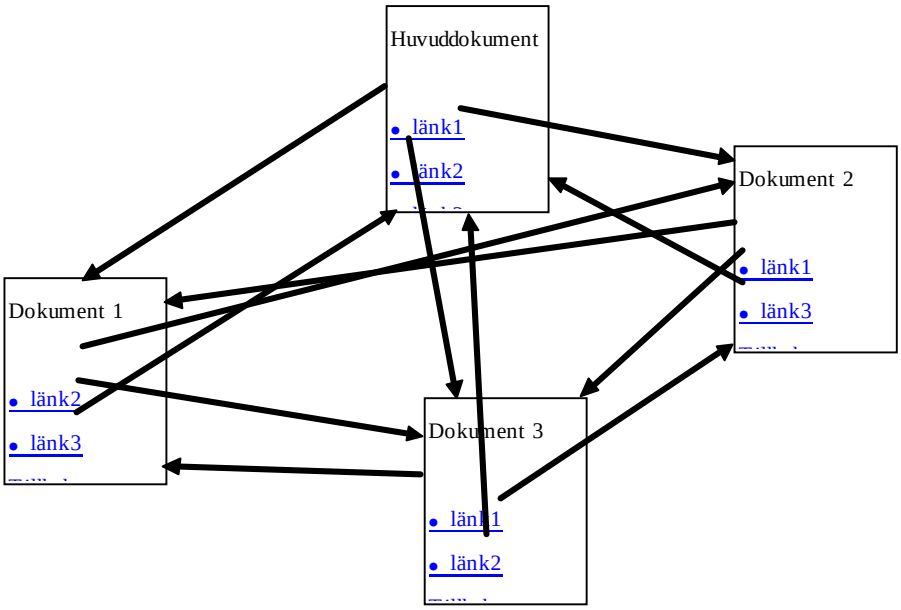
Det sätt som finns för att komma undan detta problem är att varje person som är ansvarig för en sida håller sina länkar uppdaterade. Detta kan göras manuellt men kan vara väldigt tidskrävande, även för en ganska lite sida. Istället skulle man kunna tänka sig att en mjukvarurobot sköter om att kontrollera vilka länkar som är giltiga och meddelar den ansvarige för respektive sida om vilka länkar som inte är giltiga. Detta skulle underlätta underhållsarbetet för dessa sidor avsevärt och även minska irritation hos de besökare som kommer till en sida för att söka efter information.

Det finns olika strategier för hur en sådan mjukvarurobot skulle kunna arbeta. Den skulle kunna arbeta efter en djupet först eller bredden först strategi. Djupet först innebär att mjukvaruroboten skulle ta den första länken och följa den till nästa sida och sedan första länk på den nya sedan och följa den tills man har nått ett stoppkriterium. När den nått detta stoppkriterium vänder den upp igen och tar nästa länk och följer denna nedåt. Bredden först innebär att mjukvaruroboten följer den första länken ett steg och sedan tillbaka för att ta nästa länk som den följer ett steg. När alla länkar på en sida har följts upp följer man de länkar som finns på de sidor som de länkar man följde förut på samma sätt.

Då sökdjupet, den rymd som går att nå från alla kontrollerade länkar, kan bli näst intill oändligt måste man ha ett kriterium för hur långt sökningen skall få fortsätta. Ett förslag är att, åtminstone i ett inledande skede, att man enbart följer länkar som finns inom en viss domän, tex högskolans sidor (http://www.his.se/).

2.2 Lösningsalternativ

Då hypertextsidorna refererar varandra med hjälp av hypertextlänkar kan det uppstå ganska röriga strukturer för länkarna (se Figur 6). Detta beror på att det anses som god sed att ha hypertextlänkar som refererar till huvudsidan i alla underliggande hypertextdokument. Dessutom finns det även hypertextlänkar som går till samma hypertextdokument på ett flertal platser. Detta är svårt att undvika och skall heller inte undvikas. Detta skapar dock problem om man på ett automatiskt sätt vill spåra alla hypertextlänkar som inte har någon giltig referens eftersom att detta skapar cykliska grafer i sökrymden.



Figur 6 Principskiss över länkstruktur

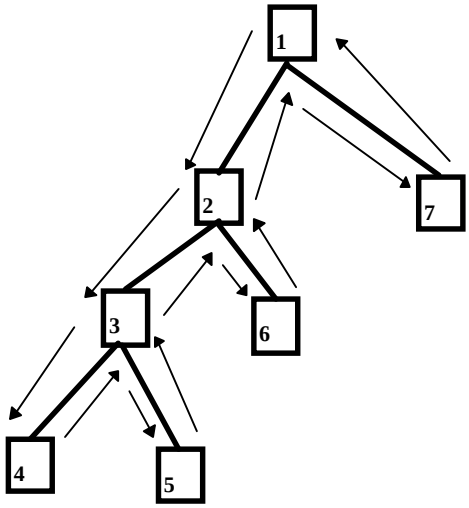
Som synes i Figur 6 ovan så blir det snabbt cykliska strukturer vilket kan skapa problem för en mjukvarurobot som skall kontrollera all hypertextlänkar. Därför måste man ha någon form av stoppkriterium eller en form av databas som håller reda på var mjukvaruroboten har varit.

De lösningsalternativ som man skulle kunna tänka sig är att mjukvaruroboten söker efter djupet först, bredden först eller att mjukvaruroboten helt enkelt delar upp sig i fler mjukvarurobotar och fortsätter enligt någon av de tidigare strategierna.

2.2.1 Djupet först

Om man låter mjukvaruroboten söka efter djupet först har man den fördelen att det är ganska lätt att implementera i jämförelse med en bredden först strategi. Djupet först fungerar genom att mjukvaruroboten följer den första länken på en sida och sedan följer första länken på nästa sida. När mjukvaruroboten har nått botten, om den inte har gått in i en cykel, vänder den uppåt igen för att ta nästa länk på sidan ovanför och följer denna nedåt tills det är dags att vända igen (se Figur 7).

Problemet med den här strategin är att den väldigt lätt skulle kunna hamna i ett cykliskt beteende som mjukvaruroboten inte skulle ta sig ur. Detta skulle kunna lösas genom att mjukvaruroboten får komma ihåg var den har varit så att den kan undvika att följa länkar som den redan har kontrollerat. Dessutom tar det väldigt lång tid att kontrollera att alla länkar på en sida är korrekta, eftersom att den hela tiden går nedåt i sökrymden. Detta innebär att de hypertextsidor som återfinns längst ned i sökrymden blir färdiga först.

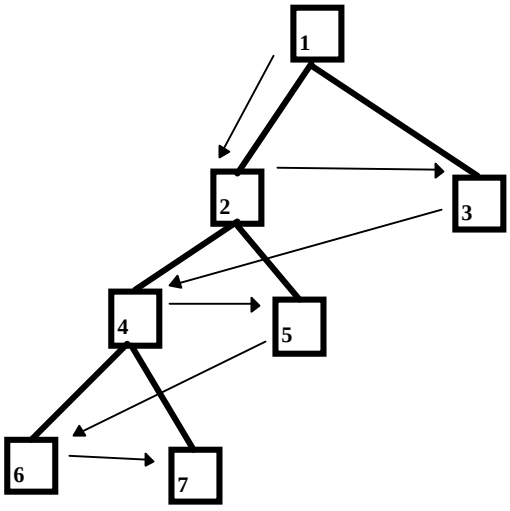


Figur 7 Principskiss för en djupet först strategi

2.2.2 Bredden först

Använder man en bredden först strategi är fördelen den att mjukvaruroboten kontrollerar alla länkar på den första sidan först för att sedan gå ett steg nedåt i sökrymden hela tiden (se Figur 8).

Fördelen med detta är att man får de översta sidorna i sökrymden kontrollerade först men nackdelen är att mjukvaruroboten måste komma ihåg många fler länkar.

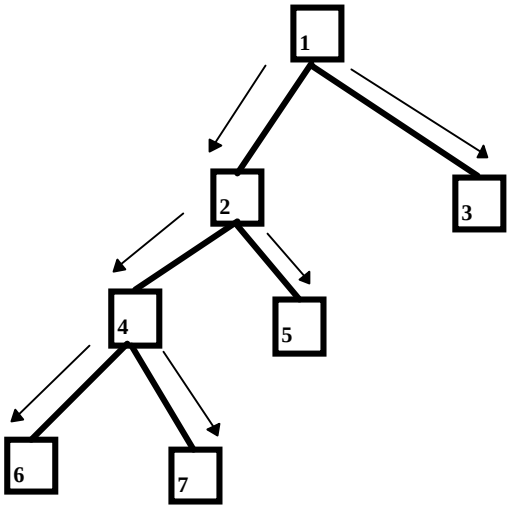


Figur 8 Principskiss för en bredden först strategi

Problemet med cykler finns fortfarande kvar i den här lösningen men problemet är inte lika allvarligt här. Dock bör mjukvaruroboten även här komma ihåg vilka länkar den har besökt förut.

2.2.3 Multipel mjukvarurobot

Problemet skulle även kunna lösas genom att mjukvaruroboten delar upp sig i flera mjukvarurobotar som sedan fortsätter på egen hand igenom sökrymden (se Figur 9). Mjukvarurobotarna skulle då kunna arbeta efter någon av de två strategierna eller en kombination av de bägge. För att inte få mjukvarurobotar som delar sig i all oändlighet måste mjukvarurobotarna även här hålla reda på var de har varit och dessutom skulle man kunna sätta ett stoppkriterium för hur många gånger en mjukvarurobot får dela sig. Detta kan dock innebära ett problem eftersom att man då inte kan garantera att alla refererade sidor nås.



Figur 9 Principskiss över en multipel mjukvarurobot

Mjukvaruroboten skulle kunna dela upp sig så att varje mjukvarurobot tilldelas en länk som den följer och sedan sker samma sak på nästa nivå och på näst nivå. Detta skulle kunna snabba upp kontrollen väsentligt. Det är även möjligt att implementera en sådan mjukvarurobot med hjälp av Java eftersom att Java stödjer multitrådning.

2.3 Val av lösningsalternativ och avgränsning

Den lösning som på det effektivaste och mest smakfulla sättet skulle lösa problemet är alternativet med multipel mjukvarurobot. Detta alternativ är det bästa eftersom att problem med cykler minimeras samt att kontrollen av länkar skulle gå mycket fortare.

Lösningen bör även spara undan de länkar som saknar referenser i en särskild fil eller i ett HTML-dokument så att det blir enkelt att följa upp. Orsaken till varför länken sparades ska även den sparas undan. Detta för att kunna se om det var en länk utan referens, en time-out, om mjukvaruroboten inte kunde hitta servern eller någon annan orsak.

Eftersom att WWW är stort och i princip oändligt svårt att genomsöka föreslås mjukvaruroboten att endast söka inom en given domän, i det här fallet Högskolan i Skövdes domän (http://www.his.se/). Mjukvaruroboten skall dock kontrollera alla länkar som leder ut från skolans domän men ej följa dessa vidare.

3 Metod

För att genomföra studien måste man använda sig av någon form av metod. I det här kapitlet kommer ett par olika metoder att presenteras och slutligen kommer en av dessa att väljas.

Arbetet kan genomföras som en deduktiv eller induktiv studie. Då det redan på förhand är beslutat att mjukvaruroboten skall implementeras i Java kommer inte några alternativa programmeringsspråk för att implementera den med att tas upp. Dock finns det som nämndes i föregående kapitel ett antal strategier för hur mjukvaruroboten skulle kunna implementeras. Strategierna är djupet först, bredden först samt en multipel mjukvarurobot.

3.1 Typ av studie

Två typer av studier som finns är deduktiv- och induktiv studie. En deduktiv studie innebär att litteratur studeras och att man ur ett givet material drar slutsatser medan en induktiv studie innebär att arbetet genomförs genom att pröva sig fram med olika metoder och bygga upp resultatet successivt till en helhet där man sedan kan dra slutsatser.

3.1.1 Deduktiv studie

När en deduktiv studie genomförs läser man olika litteratur källor och gör en analys av dessa. Analysen av litteraturen jämförs sedan med de teser man ställt upp och man drar sedan slutsatser utifrån detta.

Fördelen med en deduktiv studie är att man redan har ett färdigt material att jobba efter. Detta material är ofta redan vedertaget och accepterat av stora kretsar. Dock får man inte glömma bort att kritiskt granska det man kommer fram till utifrån sin studie eftersom att författaren kan ha fel, eller att dennes teorier är föråldrade eller så nya att de inte kan anses verifierade ännu.

Nackdelen med en teoretisk studie som denna är att den inte alltid kan användas fullt ut för att göra bedömningar om mer praktiska problem inom nya områden. Trots detta kan det dock vara till stor hjälp att läsa och analysera litteratur även vid mer försöksartade undersökningar. Detta för att få en bakgrundsbild som sedan kan hjälpa till att öka förståelsen för problemet och hur detta skulle kunna lösas.

3.1.2 Induktiv studie

Vid en induktiv studie arbetar man mer försöksmässigt och gör undersökningar som sedan analyseras och får ligga till grund för nästa steg i processen. När ett steg är avslutat fortsätter man med nästa och bygger på så sätt upp en helhet som sedan analyseras och får ligga till grund för de slutsatser man sedan drar av arbetet.

Fördelen med induktiva studier är att man kan experimentera sig fram till en lösning genom att analysera varje steg och sedan bygga på detta för att komma framåt. På detta vis bygger man så småningom upp en helhet av de delar man har kommit fram till under sitt tidigare arbete.

Nackdelen med denna typ av studie är att man kan köra in på fel spår och inte upptäcka detta förrän i slutet av sitt arbete. Därför är det viktigt att man under tiden gör en analys av hur helheten förväntas bli och försöker att styra mot denna

förväntning så att man inte riskerar att komma in på ett stickspår som i slutänden visar sig vara felaktigt.

3.1.3 Val av studietyp

Med hänsyn till arbetets art och de för och nackdelar som presenterats i föregående kapitel kommer arbetet att genomföras som en induktiv studie. Denna studie kommer att grunda sig på litterära källor och andras arbete för att få en bakgrundsbild som kan hjälpa till för att hålla arbetet på rätt spår. Det material som har tjänat som bakgrundsfakta för mitt arbete finns presenterat i referenslistan utan någon inbördes rangordning. Arbetet kommer sedan att utföras genom att bygga moduler av Javakod för att kontrollera om dessa fungerar tillfredsställande för att sedan successivt sätta ihop dessa till en helhet. Modulerna och helheten kommer sedan att testköras och resultaten från dessa testkörningar samt en analys av arbetet i övrigt kommer att ligga till grund för mina slutsatser.

3.2 Strategier

Som redan nämnts finns ett antal strategier att arbeta efter. De har alla olika förtjänster och sina egna brister. Strategierna gicks igenom i kapitel 2 men kommer att beröras här igen. Strategierna är djupet först, bredden först samt multipel mjukvarurobot. Dessa strategier kommer återigen att gås igenom för att slutligen välja ut en av dessa strategier för att arbeta efter.

3.2.1 Gemensamma egenskaper för modellen

Grundidén för mjukvaruroboten är att den skall kontrollera alla sidor under en viss domän. Det är även meningen att den adress som anges som start adress skall vara vägledande för hur sökningen skall uppföra sig. Som exempel kan ges att om adressen är: http://www.his.se/ skall den kontrollera alla länkar som kan nås från högskolans indexsida. Är adressen däremot på följande form: http://www.his.se/ida/~a94chrlu/ skall endast de sidor som kan nås från a94chrlu’s sida kontrolleras. Vidare skall även en enskild sida kunna kontrolleras. Detta sker genom att man anger adressen för sidan samt anger sidans filnamn: http://www.his.se/ida/~a94chrlu/Welcome.html.

Arbetsgången går i princip i fyra steg:

- Hämta ett dokument från WWW
- Plocka ut länkar från sidan
- Testa om länkarna är giltiga eller inte
- Skriv ut resultatet

Första länken/sidan fås genom det argument som skickas med i kommandoraden när man startar mjukvaruroboten. Denna sida skall sedan fungera som referens vad gäller den sida som skall vara basdomän, dvs den begränsande delen för vilka sidor som skall kontrolleras. Länkar som ligger utanför denna basdomän skall endast kontrolleras om de är giltiga eller inte. Sidor som påträffas inom denna domän skall läsas in och kontrolleras i sin helhet.

Nästa sida som kontrolleras fås genom att undersöka om den hör till den domän som angavs som basdomän eller inte. Hörde den till basdomänen skall samtliga länkar på

den aktuella sidan kontrolleras annars kontrolleras bara om den är giltig eller inte. När det inte finns fler sidor att kontrollera är sökning avklarad och programmet avslutas.

För att testa om en länk är giltig eller inte kontrollerar man den svarskod som sänds tillbaka av den WWW-server som är värd för den aktuella sidan. Tillsammans med svarskoden hämtas även eventuella svarsmeddelanden hem. Även dessa kommer från sidans WWW-server. Svarskoderna består av ett tresiffrigt heltal. Börjar svarskoden på en tvåa (2**) är sidan godkänd, börjar den däremot på en trea eller högre (3**, 4**, osv) kan sidan inte nås av någon anledning. Därför kontrolleras om svarskoden är lägre än 300. Om koden är lägre adderas sidan till godkända länkar annars adderas den till länkar som blivit underkända. Några vanliga svarskoder med tillhörande svarsmeddelande är:

- 200 Document follows
- 403 Forbidden
- 404 Not found

Resultatet skrivs sedan ut och sökningen fortsätter. Utskriften av resultatet sker på så sätt att först skrivs en rubrik som tala om om det är bra eller dåliga länkar. Sedan kommer en rad som talar om på vilken sida länken fanns. Därefter kommer några rader som talar om vilken länk det var, vilken svarskod och vilket svarsmeddelande som returnerades. Ett exempel på hur resultatet kan se ut finns i Figur 10 nedan.

```
Bra länkar:
Sida: http://hem1.passagen.se/qaz1/
Länk: http://hem1.passagen.se/qaz1/index2.html
Response-code: 200
Response-message: OK
Dåliga Länkar:
…
Bra länkar:
…
Sida: http://hem1.passagen.se/qaz1/index2.html
Länk: http://www.arachnoid.com/careware/
Response-code: 200
Response-message: Document follows
…
Dåliga länkar:
Sida: http://hem1.passagen.se/qaz1/index2.html
Länk: http://den.har.lanken/finns/inte.html
Response-code: 404
Response-message: Document not found
```

Figur 10 Exempel på resultatutskrift

3.2.2 Djupet först

En djupet först strategi arbetar efter principen att följa första bästa väg tills det inte finns fler vägar att följa, för att sedan vända tillbaka upp igen. Därefter tar den nästa möjliga väg och följer den tills denna tar slut. Så fortsätter sökningen tills alla sidor och länkar är kontrollerade (se Figur 7).

Denna strategi skulle kunna implementeras på sätt att när en länk påträffas på en sida kontrolleras denna. Om länken är giltig och om den tillhör basdomänen sparas den undan i en lista. Sedan följer man länken och kontrollerar nästa sida på samma sätt. Samma procedur upprepas tills det inte finns fler länkar att följa. Nu tar man den senast tillagda länken i listan och följer denna enligt samma procedur som tidigare. Detta fortsätter tills listan är tom och inga fler länkar finns att kontrollera. De besökta länkarna sparas sedan undan i en separat lista som kontrolleras så att ingen länk besöks mer än en gång. Detta förhindrar cykliska beteenden hos mjukvaruroboten.

Fördelarna med en djupet först strategi är att den är relativt enkel att implementera. Den kräver inte häller lika mycket resurser av det system som det installeras på eftersom att enkelheten i implementationen använder så lite resurser som möjligt.

Nackdelarna är dock att om mjukvaruroboten skulle missa att en länk är besökt och färdigkontrollerad, på grund av en implementeringsmiss, så skulle den förmodligen ganska snart gå in i ett cykliskt sökbeteende. Detta på grund av de länkstrukturer som har nämnts i tidigare kapitel (se Figur 6). En annan nackdel är att sidorna i botten på den länkstruktur som skall kontrolleras blir färdiga först och toppsidorna blir färdiga sist. Detta kan uppfattas som ett problem om man av någon anledning skulle vilja eller vara tvungen att avbryta sökningen.

3.2.3 Bredden först

En bredden först strategi arbetar på så sätt att den kontrollerar alla länkar på en sida först för att sedan kontrollera alla länkar på nästa sida. På detta sätt fortsätter den ned genom länkhierarkin tills alla sidor och länkar är kontrollerade (se Figur 8).

Denna strategi kräver en något annorlunda implementation än den djupet först strategin. När en sida kontrolleras letar den upp samtliga länkar på en sida och kontrollerar dessa. Om en länk tillhör basdomänen och inte finns i en lista för kontrollerade länkar sparas den undan i en lista för sidor att besöka. Den sparar även undan den genomsökta sidan i en lista för kontrollerade länkar. Sedan tar den första länken i listan för sidor att besöka och upprepar proceduren tills alla sidor och länkar är kontrollerade.

Fördelarna med denna strategi är att toppsidorna i en domän blir kontrollerade först om man av någon anledning skulle vilja eller vara tvungen att avbryta sökningen, samt att den är mindre känslig för cykliska beteenden än djupet först strategin.

3.2.4 Multipel mjukvarurobot

Denna strategi går ut på att mjukvaruroboten arbetar efter någon av de tidigare nämnda strategierna och delar upp sig i en ny mjukvarurobot när en länk till en sida som skall besökas påträffas. På detta sätt fortsätter sökningen tills alla sidor i basdomänen är kontrollerade (se Figur 9).

Strategin skulle kunna implementeras genom någon av de tidigare strategierna. Strategin kräver dessutom en gemensam uppsättning av listor för sidor som skall besökas samt för sidor som är besökta. När en länk till en sida som tillhör basadressen påträffas och denna inte finns i listan för sidor att besöka eller listan för besökta sidor startas en tråd upp som fungerar enligt samma princip som huvudroboten. Sedan besöker denna tråden den aktuella sidan och kontrollerar den samma. Skulle nya länkar till sidor som tillhör basdomänen dyka upp startas ytterligare en tråd. Detta innebär att vi får flera trådar som parallellt besöker de olika sidorna och kontrollerar

dessa. När en tråd kommer till ett läge där den inte har några nya sidor att besöka och att den aktuella sidan är kontrollerad avslutas denna. När en tråds samtliga undertrådar är avslutade avslutas denna. När huvudroboten inte längre har några aktiva trådar kvar och basadressen är kontrollerad avslutas körningen.

Fördelarna med denna strategi är att toppsidorna i länkhierarkin blir kontrollerade först samt att sökningen kan gå snabbare än vid de tidigare nämnda strategierna. Detta beror på att om en tråd blir uppehållen med att invänta ett svar från en länk kan de övriga trådarna fortsätta ostört att kontrollera övriga delar av länkhierarkin. Detta innebär att det kan gå snabbare eftersom att de andra strategierna är sekventiella i sitt beteende och måste vänta på svar innan kontrollen kan gå vidare.

Nackdelarna är att denna strategi tar mer systemresurser i anspråk och att ett synkroniseringsproblem uppstår. Detta eftersom att de olika trådarna måste komma överens om vem som skall ha tillgång till de gemensamma listorna. Detta problem måste lösas så att inga konflikter så som deadlock eller starvation kan inträffa.

3.2.5 Val av strategi

Vid val av strategi måste de olika metoderna ställas mot varandra för att på så sätt kunna välja den strategi som kan anses lämpligast för uppgiften. Då detta är en första version av en Java-baserad mjukvarurobot för att identifiera länkar som saknar referenser måste detta också vägas in i bedömningen.

Då en djupet först strategi har allt för stora nackdelar på grund av att den gör klart bottensidorna först och är mest känslig för cykler vid en implementeringsmiss väljs denna bort. Kvar återstår då bredden först och en multipel mjukvarurobot. Den strategi som är mest tilltalande tack vare effektivitet med avseende på söktid och beteende i allmänhet är den multipla mjukvaruroboten. Dock kräver denna en synkroniseringsmekanism som kan komplicera implementeringen och vid en implementeringsmiss vålla problem vad gäller förutseende av beteende. Då det endast finns en begränsad tid till förfogande för att lösa uppgiften och att det är en första prototyp skjuts denna strategi på framtiden och kan eventuellt ge upphov till framtida arbete vad gäller examensarbeten vid Högskolan i Skövde. Kvar blir då bredden först strategin som väljs på grund av att den har ett någorlunda smakfullt beteende och att den ligger inom rimliga gränser för att implementera då det inte finns någon grund att bygga på.

4 Implementation

När vi kommer till implementation av mjukvaruroboten kommer arbetsmomenten att beskrivas var för sig och i en helhet. Implementationen har genomförts efter studier av Suns Java Tuturial [Www25] och de exempel som finns där samt studier av liknande mjukvarurobotar implementerade i andra programmeringsspråk. Som exempel kan nämnas LinkScan från Electronic Software Publishing Corporation (Elsop) [Www09] som är en mjukvarurobot som kontrollerar länkar på en viss domän. Dessutom hämtades idéer och inspiration från ett flertal andra platser där jag i princip kan hänvisa till hela referenslistan.

I nästa steg kommer en serie tester för att kontrollera att mjukvaruroboten fungerar som den ska. Bland annat skall det testas om alla länkar blir identifierade på en sida och sedan om den behandlar de olika svarskoderna för de kontrollerade länkarna på rätt sätt. Vidare skall det även kontrolleras att den valda bredden först strategin följs.

4.1 Komponenterna

Det första som behövde göras var att gå igenom vilka komponenter som skulle komma att behövas för implementationen och att besluta sig för hur arbetet skulle läggas upp. Ett beslut togs om att bygga de olika komponenterna var för sig i moduler som sedan skulle förenas i en enda stor slut slutprodukt. Dessa moduler var de funktioner som skulle finnas i mjukvaruroboten.

De komponenter som behövdes var:

- Något som hämtar hem ett hypertextdokument från WWW
- Något som identifierar om en länk är giltig eller inte
- Något som plockar ut de länkar som finns i ett hypertextdokument
- En uppsättning listor som håller reda på vilka sidor som skall besökas, vilka sidor som är besökta, vilka länkar som var giltiga och slutligen en lista som höll reda på vilka länkar som saknade referens

4.1.1 Hämta en sida

Att hämta hem en sida från WWW var den första uppgiften som måste lösas. För att lösa detta behövdes en inventering av vilka paket i Java som behövde användas. Dessa var java.io för input/output operationer samt java.net [Www26] för kontakten över Internet och för att kunna hantera läsning av hypertextdokument placerade på WWW.

Problemet löstes genom att en klass som kallades för hamta skapades (se figur 11). Denna fungerade som så att den tog argumentet från kommandoraden (i ett vanligt UNIX-shell) som adress och sedan skapade ett URL objekt som innebär att en förbindelse öppnas mot ett hypertextdokument. Sedan skapades en inputstream för att läsa rader från hypertextdokumentet. Därefter sattes denna in i en while loop som fortsatte tills att det inte fans fler rader att läsa. I denna loop skrevs sedan respektive inläst rad ut. Eventuella exceptions, dvs om något fel eller undantag inträffar, måste tas om hand. Gör man inte detta går det helt enkelt inte att kompilera koden. Användningen av programmet sker enligt nedan:

java hamta <http://adress.till.den/sida/som/skall/läsas.html>

```
import java.net.*;
import java.io.*;
public class hamta {
   public static void main (String args[]) {
      int i;
      String rad;
      URL u;
      if  (args.length > 0) {
        //Öppna URL för läsning
        try {
           u = new URL(args[0]);
           try {
              BufferedReader sida = new BufferedReader(new
                           InputStreamReader(u.openStream()));
              try {
                 while ((rad = sida.readLine()) != null) {
                    System.out.println(rad);
                 }// while
              }  // try
              catch (Exception e) {
                 System.err.println(e);
              } // catch
           } // try
           catch (Exception e) {
              System.err.println(e);
           } // catch
        } // try
        catch (MalformedURLException e) {
           System.err.println(args[0] + " är en ogiltig URL");
           System.err.println(e);
        } // catch
      } // if
   } // main
}  // hamta
```

Figur 11 Källkod för hamta.java

Denna klass fungerade bra men visade sig inte behövas senare under arbetet. Dock gav den nyttiga lärdomar inför det fortsatta arbetet. Utskrifterna från klassen blev rena kopior av den HTML-kod som hypertextsidorna var skrivna med.

4.1.2 Giltiga länkar

Nästa steg var att kontrollera om en länk var giltig eller inte. Efter noggranna studier av Javas API [Www26] insågs att en uppgradering till JDK1.1.1 [Www26] behövdes istället för JDK1.0.2 som var aktuell när arbetet påbörjades. Det som saknades i den tidigare versionen var en funktion för att hämta hem den svarskod och det svarsmeddelande som den WWW-servern som är värd för den aktuella hypertextsidan. Det hade kommit ett tillägg i paketet java.net som kallades java.net.HttpURLConnection som innehöll just dessa funktioner.

Problemet löstes genom att skapa klassen respons som har till uppgift att kontrollera svarskod och svarsmeddelande för ett hypertextdokument utan att behöva hämta hem hela dokumentet (se Figur 12). Respons fungerar på samma sätt som hamta för vad som gäller för adress via kommandoraden. Först skapas ett URL objekt som har till uppgift att öppna kommunikation med det angivna hypertextdokumentet. Sedan skapas ett HttpURLConnection objekt för att kunna avläsa svarskod och svarsmeddelande. Detta objekt tilldelas sedan URL objektet för att kunna kommunicera med Hypertextdokumentet. Sedan skrivs svarskoden och svarsmeddelandet ut.

```
HttpURLConnection pp = (HttpURLConnection) p.openConnection();
System.out.println("Response-code: " + pp.getResponseCode());
System.out.println("Response-message: " + pp.getResponseMessage());
```

Figur 12 Hämta hem svarskod och svarsmeddelande

Denna funktion fungerade utmärkt och levererade de önskade värdena på skärmen.

4.1.3 Identifiera länkar

Nästa steg var att identifiera de länkar som finns på ett aktuellt hypertextdokument. Detta visade sig vara lite svårare än förväntat. I Suns API i för JDK1.1.1 [Www26] fanns inga färdiga funktioner för detta utan en egen funktion för att lösa detta skapades. Eftersom det finns ett stort antal sätt att skriva giltiga länkar i HTML bara de börjar med <a och slutar med och att HTML accepterar både gemener och versaler i blandad ordning så länge adressen har rätt utformning måste man tänka till. Vad är det som identifierar en länk? Hur ska funktionen kunna ta ut så många länkar som möjligt ur en sida och ändå hålla det på en rimlig nivå för arbetsinsatsen. Kan det göras generellt så att den hittar alla länkar? Svaren på dessa frågor var inte helt uppenbara.

Idén för att plocka ut länkar är att man identifierar att det är en länk med hjälp av den tag som inleder den och fortsätter tills inga fler länkar kan identifieras på den aktuella sidan.

För att kunna plocka ut de länkar som finns på en sida behöver dessa identifieras. Ett stort problem i sammanhanget är att dessa länkar/referenser kan se väldigt olika ut, men ändå vara en giltigt länk/referens.

Ovanstående problem har vållat stora problem då ingen självklar lösning förelåg. Från början verkade det enkelt, det var ju bara att läsa ett tecken åt gången och kontrollera när ett ”<” dyker upp. Så enkelt var det dock inte eftersom att länkarna kan skrivas på många olika sätt och ändå vara giltiga. Dessutom finns ett flertal så kallade tags som dessutom börjar på a.

Principen för lösningen ser ut som följer:

1. Om vi läser in ett < fortsätter vi läsa tills vi hittar motsvarande >
2. Spara tagen
3. Konvertera tagen till stora bokstäver
4. Kontrollera om tagen börjar med <A HREF
5. Om så är fallet spara detta som en position i tagen annars läs vidare
6. Läs tills vi hittar ett = spara som nästa position
7. Läs tills vi hittar ett ” spara som nästa position
8. Läs tills vi hittar ytterligare ett ” spara som sista position
9. URLen finns nu mellan position tre och fyra
10. Läs om URLen från den ursprungliga tagen mellan position tre och fyra
11. Om vi läser ett : är det en absolut adress om inte är det en relativ adress
12. Om det var en relativ adress gör om det till en absolut adress

Sedan är länken identifierad i dokumentet och kan sedan användas för kontroll. Detta resulterade i klassen DraLank som även kom att utgöra stommen i mjukvaruroboten (se Figur 13).

```
…
try {
     DataInputStream theHTML = new DataInputStream(theURL.openStream());
     while (true) {
```

```
        thisChar = (char) theHTML.readByte();
        if (thisChar == '<') {
          theTag = readTag(theHTML);
          theTag = convertTag(theTag);
       } // if
      } // while
…
public static String readTag(DataInputStream is) {
    StringBuffer theTag = new StringBuffer("<");
    char theChar = '<';
    try {
       while (theChar != '>') {
         theChar = (char) is.readByte();
         theTag.append(theChar);
       } // while
     }  // try
…
} // readTag
…
public String convertTag(String tag) {
    int p1, p2, p3, p4;
    try {
      String s1 = tag.toUpperCase();
      if (s1.startsWith("<A HREF") || s1.startsWith("<A \nHREF") ||
s1.startsWith("<A\nHREF")) {
        p1 = s1.indexOf("HREF");
      }
      else {
        return tag;
      } // if
      p2 = s1.indexOf ("=", p1);
      if (p2 == -1) return tag;
      p3 = p2+1;
      while (Character.isWhitespace(s1.charAt(p3))) {
        p3++;
      } // while
      if (s1.charAt(p3) == '"') p3++;
      if (s1.substring(p3,s1.length()).startsWith("MAILTO:")) return tag;
      p4 = p3+1;
      while (!Character.isWhitespace(s1.charAt(p4)) &&
       s1.charAt(p4) != '"') {
        p4++;
      } // while
      String link = tag.substring(p3, p4);
      String nlink = tag.substring(p3, p4);
      if (link.indexOf(":")  == -1) {
           URL newURL = new URL(theURL, link);
           tag = s1.substring(0,p3) + newURL + s1.substring(p4,s1.length());
           nlink = newURL.toString();
        }  // if
    System.out.println(nlink);
    }  // try
…
    return tag;
  } // convertTag
} // DraLank
```

Figur 13 Centrala delar ur DraLank.java

DraLank.java inspirerades av och byggdes till stora delar på Pagesaver.java [Www27]. Dock finns det vissa problem med DraLank.java. Den klarar inte av att hitta alla formuleringar av länkar trots konvertering av dem. Detta beror på hur start tagen för länkarna ser ut. Trots detta så hittar DraLank de allra flesta länkar tack vare villkoret i Figur 14.

```
if (s1.startsWith("<A HREF") || s1.startsWith("<A \nHREF") ||
     s1.startsWith("<A\nHREF"))
```

Figur 14 Villkor för att identifiera att det är en länk

Orsaken till varför villkoret är utformat som det är beror på att dessa är de vanligaste sätten att börja en länktag på. Dock fångar villkoret inte upp länktagar på den form som kan ses i Figur 15, och ej heller en mängd andra konstiga sätt att skriva länktagar på.

```
<
A
HREF
```

Figur 15 Exempel på länktag som inte fångas upp

Ett annat problem är att många hypertextdokument har börjat använda Java-script och Java-applets för att presenter länkar i dokumenten. Eftersom att de länkar som finns i scripts eller applets inte börjar med en normal länktag hittas inte dessa vid en sökning.

4.1.4 Listor

Det var nu dags att hitta en lämplig form för de listor som skulle användas för att hålla reda på de olika länkarna. För att hitta rätt typ av lista kontrollerades återigen vilken information som skulle sparas i listorna. Denna undersökning ledde fram till att det vore enklast om man lade ihop all information som skulle lagras om respektive länk. Detta skulle göra det enkelt att skapa ett objekt för den undersökta länken. Dessa objekt skulle sedan sparas i någon form av lista. Dessa listor skulle kunna ha formen av en fil, array eller stack.

Eftersom att det vore ineffektivt att kontinuerligt läsa och skriva information från och till filer under arbetets gång uteslöts filalternativet som tillfälligt lagringsmedium. Stackhantering skulle kunna vara en lösning men skulle vålla problem med utskriftsordningen när det var dags för att skriva ut informationen och skulle innebära en onödig extra hantering för att vända utskriftsordning på informationen. Därför valdes arrayer som lagringsmedium eftersom att det går att enkelt att läsa från och skriva till en array från en godtycklig position. Detta faktum blev huvudorsaken till varför någon typ av array skulle användas.

Eftersom att det var objekt som skulle lagras i arrayen vore det lämpligt med en array som hanterade just sådana. Nu var problemet hur att hantera att antalet objekt i arrayen kunde variera. Detta skulle kunna lösas genom att man dynamiskt allokerar upp utrymme i respektive array. Efter att ha studerat JDK1.1.1s API [Www26] ytterligare en gång kom jag fram till att det fanns en arraytyp i paketet java.util som hette java.util.Vector som automatiskt anpassade sin storlek beroende på om man lade till eller tog bort information från vektorn. Dessutom kan vektorn skapas med en ursprunglig kapacitet innan den behöver allokeras om och dessutom kan det anges hur stor en kapacitetsökning skall vara.

Nu var det dags att skapa de objekt som skulle användas för lagring om de olika länkarnas status. Dessa objekt skulle innehålla vilken sida länken fanns på, länkens namn, svarskod och svarsmeddelande från den aktuella sidans WWW-server. Objektet kom att se ut som i Figur 16.

```
class Sidblock {
            String sida = null;
            String lank = null;
            String kod = null;
            String med = null;
          public void skriv(){
            try {
               System.out.println(sida + "\n" + lank + "\n" +
                                  kod + "\n" + med + "\n");
            } catch(Exception e) {}
          }// skriv
        }// Sidblock
```

Figur 16 Klass för objektet SidBlock

Objektet utrustades dessutom med en utskriftsfunktion för att underlätta hanteringen av informationen.

De vektorer som slutligen kom att användas var ‘attScanna’, ‘scannade’, ‘bra’ och ‘daliga’. De typer av information som skulle sparas i respektive vektor var:

- attScanna - adressen till de sidor som skall kontrolleras i form av en String
- scannade - adressen till de sidor som är kontrollerade i form av en String
- bra - objektet Sidblock för de länkar som är giltiga
- daliga - objektet Sidblock för de länkar som är ogiltiga

Eftersom att vektorn lagra all information som objekt måste man göra en cast innan vektorn tilldelas ett objekt så att vektorn vet vilken typ av objekt som skall lagras. Hur vektorerna skapas med en initial kapacitet och med vilken kapacitet de ska utökas med efter behov kan ses i Figur 17.

```
Vector attScanna = new Vector(100, 50); // Adresser som skall scannas
Vector scannade = new Vector(100, 50); // Adresser som är scannade
Vector bra = new Vector(100, 100); // Adresser som är OK
Vector daliga = new Vector(50, 10); // Adresser som inte var OK
```

Figur 17 Skapa vektorer med ursprunglig kapacitet.

Eftersom att sidorna som skall kontrolleras förmodas vara färre än de länkar som leder ut från basdomänen utökas dessa med 50 extraplatser vid behov. Dessutom antas antalet länkar som är giltiga vara fler än de som saknar referenser, därav att vektorn för dåliga referenser får så pass liten ursprunglig kapacitet och så liten kapacitetsökning.

4.2 Sammansättning av komponenter

När komponenterna är klara är det dags att sätta ihop dessa till en fungerande enhet. Denna enhet kom att heta Barbapappa.java (se Appendix B) eftersom att det verkade bli ett konststycke i sig att få enheterna att fungera tillsammans.

DraLank.java fick tjäna som stomme för Barbapappa eftersom att DraLank skulle komma att utföra det tyngsta och mest centrala arbetet. De övriga komponenterna infogades som funktioner i Barbapappa.

Barbapappa är beroende av tre paket i JDK1.1.1 [Www26], nämligen java.net, java.io och java.util. Barbapappa börjar med att definiera en URL som senare kommer att användas som grund för vilken sida som för stunden kontrolleras. Efter detta definieras även de vektorer som kommer att användas.

Basdomänen definieras sedan genom att man anger den vid kommandoraden vid uppstarten av Barbapappa (se Figur 18). Ett URL objekt (root) skapas med hjälp den adress som angavs vid uppstarten för att öppna en kommunikationskanal mot sidan. Denna adress kommer även att tjäna som basadress för Barbapappa.

```
java Barbapappa <http://den.adress.som/skall/kontrolleras/<sida.html>>
```

Figur 18 Uppstart av Barbapappa

Ett Barbappapa objekt (dl) skapas med den URL som angavs vid uppstarten samt tilldelar den först definierade URLen (theURL) . Sedan körs funktionen draLankarna (se Figur 19) på detta objekt (dl).

```
public void draLankarna() {
…
```

```
try {
     DataInputStream theHTML = new DataInputStream(theURL.openStream());
     while (true) {
       thisChar = (char) theHTML.readByte();
       if (thisChar == '<') {
         theTag = readTag(theHTML);
         theTag = convertTag(theTag);
      } // if
     } // while
…
  }  //  draLankarna
```

Figur 19 Huvuddelarna av draLankarna

draLankarna öppnar en Datainputstream för URLen (theHTML) för att kunna läsa den aktuella sidan. Sedan går draLankarna in i en while loop som itererar tills ett end of file undantag inträffar. I loopen läser draLankarna tecken för tecken. Om funktionen läser in ett < så påbörjas funktionen readTag (se Figur 20) som läser in den aktuella tagen tecken för tecken och utökar theTag så länge nästa tecken inte är ett >. readTag returnerar sedan theTag till draLankarna.

```
…
    try {
       while (theChar != '>') {
         theChar = (char) is.readByte();
         theTag.append(theChar);
       } // while
     }  // try
…
return theTag.toString();
  } // readTag
```

Figur 20 Huvuddelarna av readTag

draLankarna anropar då funktionen convertTag (se Figur 21) som börjar med att skapa fyra positioner (position1 - 4) som initieras till 0. Sedan skapas en String (s1) av den medskickade tagen och omvandlar denna till versaler. convertTag kontrollerar nu om det är en länktag, dvs om den börjar med <A HREF. Om den inte gör det returneras tagen utan åtgärd. Om s1 började på <A HREF eller likvärdigt tilldelas position1 positionen för HREF. position2 tilldelas nu positionen för =. Om det inte fanns något = returneras tagen utan åtgärd. position3 tilldelas nu position2 + 1 samt ökas på så länge nästa tecken är ett blanksteg. Om position3 nu är ett = ökas position3 med ett ytterligare en gång. Om delsträngen mellan position3 och strängens slut börjar med MAILTO: returneras tagen eftersom att detta inte är en tag som skall kontrolleras. position4 tilldelas nu position3 + 1. Så länge tecknet på position4 inte är blanksteg och inte är ett ” ökas position4 med ett. Adressen som refererades av länken finns nu mellan position3 och 4 i s1. Nu skapas två Stringar (link och nlink) som tilldelas delsträngen mellan position3 och position4 som baseras på den ursprungliga oförändrade tagen. Detta eftersom att adresserna är känsliga för gemener och versaler och att s1 var omvandlad till enbart versaler. Om link inte innehåller ett : skapas en ny URL (newURL) som består av basdomänen (theURL) följt av link. Detta för att skapa en absolut adress så att denna kan kontrolleras senare. Den ursprungliga tagen tilldelas nu den ursprungliga tagen med undantag av att den nu består av versaler förutom adressen som nu även blivit absolut. nlink tilldelas nu newURL som String, dvs den absoluta adressen. convertTag anropar nu funktionen respons med nlink som argument.

```
public String convertTag(String tag) {
    int position1 = 0, position2 = 0, position3 = 0, position4 = 0;
    try {
      String s1 = tag.toUpperCase();
      if (s1.startsWith("<A HREF") || s1.startsWith("<A \nHREF") ||
          s1.startsWith("<A\nHREF")) {
         position1 = s1.indexOf("HREF");
```

```
            }
            else {
              return tag;
            } // if
            position2 = s1.indexOf ("=", p1);
            if (position2 == -1) return tag;
            position3 = position2+1;
            while (Character.isWhitespace(s1.charAt(position3))) {
              position3++;
            } // while
            if (s1.charAt(position3) == '"') position3++;
            if (s1.substring(position3,s1.length()).startsWith("MAILTO:")) return
                  tag;
            position4 = position3+1;
            while (!Character.isWhitespace(s1.charAt(position4)) &&
             s1.charAt(position4) != '"') {
              position4++;
            } // while
            String link = tag.substring(position3, position4);
            String nlink = tag.substring(position3, position4);
            if (link.indexOf(":")  == -1) {
                 URL newURL = new URL(theURL, link);
                 tag = s1.substring(0, position3) + newURL +
                       s1.substring(position4,s1.length());
                 nlink = newURL.toString();
              }  // if
            respons(nlink);
          }  // try
      …
      return tag;
  } // convertTag
```

Figur 21 Huvuddelar av convertTag

Funktionen respons (se Figur 22) skapar nu en ny URL (p) av den medskickade adressen (nlink -> adr). Ett HttpURLConnection objekt (pp) skapas av p för att kunna avläsa svarskod och svarsmeddelande från den WWW-server som agerar värd åt p. Ett Sidblock objekt (s) skapas. De olika komponenterna av s tilldelas respektive värden. Om svarskoden för pp > 300 läggs s till listan för ogiltiga länkar annars läggs den till listan för länkar som har giltig referens. respons returnerar adr till convertTag som i sin tur returnerar tag till draLankarna.

```
      public String respons(String adr) {
              try {
                 URL p = new URL(adr);
                 HttpURLConnection pp = (HttpURLConnection) p.openConnection();
                 String sida, lank, kod, med;
                 Sidblock s = new Sidblock();
                 s.sida = "Sida: " + theURL;
                 s.lank = "Länk: " + adr;
                 s.kod = "Response-code: " + pp.getResponseCode();
                 s.med = "Response-message: " + pp.getResponseMessage();
                 if (pp.getResponseCode() > 300) {
                     daliga.addElement(s);
                 }
                 else {
                     bra.addElement(s);
                 }
                 if (adr.startsWith(theURL.toString()) &&
                     !scannade.contains(adr)){
                     attScanna.addElement(adr);
                 }
                  if (adr.startsWith(theURL.toString())){
                     scannade.addElement(adr);
                 }

                 adr = pp.getResponseMessage();
              }  // try
              …
                 return adr;
      } // respons
```

Figur 22 Huvuddelar av respons

draLankarna i sin tur tilldelar theTag värdet av funktionen convertTag och avslutas sedan och återvänder till main. main skriver sedan ut resultatet av de godkända och de

underkända länkarna. En ny URL c skapas av det första elementet i listan för länkar att kontrollera (attScanna). Elementet som stod först i listan attScanna tas bort och skulle ha överförts till listan för kontrollerade länkar (scannade). Ett nytt Barbapappa objekt (ndl) skapas av c och funktionen draLankarna utförs på ndl. Processen börjar om på samma sätt som för dl och fortsätter tills samtliga funna sidor som tillhör basdomänen är kontrollerade.

På grund av en designmiss som ej har kunnat rättas till så söker Barbapappa endast en nivå ned i basdomänen. Detta beror på att för mycket funktionalitet lades i funktioner som anropades från andra funktioner. Detta borde ha lösts genom att göra moduler av varje funktionalitet och anropa dessa från main istället. Detta skulle ha gjort det lättare att uppnå önskad funktionalitet. Som exempel kan anges att funktionen respons anropas från funktionen convertTag. ConvertTag borde ha returnerat sitt värde till main och därifrån skulle sedan respons ha anropats. Dessutom borde vektorerna för länkobjekten varit globala istället för, som nu, lokala.

4.3 Körinstruktion

Mjukvaruroboten Barbapappa kan kontrollera en domän eller ett enstaka hypertextdokument för att identifiera länkar som saknar referenser.

För att kontrollera en hel domän startas Barbapappa med nedanstående kommando:

```
java Barbapappa <http://den.domän.som/skall/kontrolleras/> > resultat.txt
```

För att kontrollera en enstaka sida startas Barbapappa med nedanstående kommando:

```
java Barbapappa <http://den.sida.som/skall/kontrolleras.html> > resultat.txt
```

Vid användnig på annan server än den som är belägen på högskolan i Skövde bör

följande rader ändras:

```
Properties prop = System.getProperties();
prop.put("proxySet", "true");
prop.put("proxyHost", "wwwproxy.his.se");
prop.put("proxyPort", "80");
```

Dessa återfinns i main på Barbapappa.java.

- `prop.put("proxySet", "true");` anger att proxy skall användas och sätts till `false` om så inte är fallet.
- `prop.put("proxyHost", "wwwproxy.his.se");` anger vilken proxy som används. `wwwproxy.his.se` byts ut mot namnet på den proxy som skall användas istället.
- `prop.put("proxyPort", "80");` anger vilken port som används. `80` byts ut mot den port som skall användas (normalt är dock 80 rätt port).

5 Tester

Vid testningen av Barbapappa kontrolleras om mjukvaruroboten:

- lyckas hitta samtliga länkar som följer det angivna villkoret för uppfångst på de hypertextdokument som sorterar under basdomänen
- lyckas kontrollera alla funna länkar som finns på de sidor som skall kontrolleras på ett tillfredsställande sätt
- lyckas hitta alla dokument som sorterar under huvuddokumentet
- lyckas följa den valda bredden först strategin

För att få ett opartiskt urval av sidor att testköra på skapades en slumpgenerator (Se Appendix C, sida.p). Denna slumpar fram en sida ur Svenska Akademins Ordlista, därefter slumpas ett uppslagsord. De sidor som tas fram är de med uppslagsord. Uppslagsordet räknas från första ordet på den slumpade sidan och det antal icke upprepade ord framåt som talet anger. Därefter används AltaVista™ för att ta fram aktuell sida att kontrollera. Den första träffen med dokument på svenska som matchar det framslumpade ordet tas som referens om det är tillgängligt. Skulle dokumentet vara utdaterat tas nästa träff i listan. Om träffen är en enskild sida, dvs den består av en basadress och avslutas med ett enskiltdokument, kontrolleras både den enskilda sidan samt närmsta nod bakåt i adressen. De framslumpade orden och adresserna återfinns i Appendix C, uppslag.txt.

5.1 Test, hitta alla länkar på ett hypertextdokument

För att testa denna funktion görs provkörningar på ett antal olika domäner och enstaka sidor. Resultaten jämförs sedan mot HTML-koden för de olika dokumenten varvid det kontrolleras att samtliga länkar som uppfylls av uppfångstvillkoret identifieras.

För att få ett godkänt resultat på testet krävs att 100% av länkarna som uppfyllde uppfångstvillkoret identifierades på varje enskilt dokument samt totalt sett över hela sökdomänen vid varje testkörning.

5.2 Test, svarskod och svarsmeddelande

För att testa denna funktion görs provkörningar på ett antal olika domäner och enstaka sidor. Resultaten kontrolleras mot de länkar som blev funna vid test 4.4.1.

För godkänt resultat krävs att 100% av de länkar som refererade ett HTML-dokument fick en svarskod och registrerades under rätt kategori. Länkar som refererar dokument som ej är HTML-dokument räknas bort vid sammanställning av resultatet.

5.3 Test, hitta alla hypertextdokument

För att testa denna funktion görs provkörningar på ett antal olika domäner och enstaka sidor. Resultaten jämförs sedan med den faktiska länkstruktur som finns under den aktuella basdomänen samt de länkar som blev funna vid test 4.4.1.

För godkänt resultat krävs att 100% av de HTML-dokument som sorterar under basdomänen återfanns. Sidor som ej blev funna i test 4.4.1 räknas bort vid sammanställning av resultatet.

5.4 Test, strategi

För att testa denna funktion görs provkörningar på ett antal olika domäner. Resultaten jämförs sedan mot strategin.

För godkänt resultat krävs att 100% av de funna länkarna blev besökta i den ordning som de registrerades.

5.5 Test, sammanställning

Sammanställningen av testerna har av utrymmesskäl tvingats kortas ned. Urdrag av testresultaten kommer att visas här samt i Appendix C. Övriga testkörningar medsändes i digitalt lagrad form i filen testres.tar.gz.

5.5.1 Hitta alla länkar, sammanställning

För att kontrollera att alla länkar som uppfyllde uppfångstvillkoret hittades jämfördes källkoden för de kontrollerade dokumenten mot de testresultat som producerades när Barbapappa kördes. Källkoderna återfinns i begränsad mån i Appendix C och på fullständig form i filen testres.tar.gz.

http://strindberg.ling.uu.se/~sarag/puh.html

```
<HTML>
<HEAD>
<TITLE>Puh
…
<A HREF="Bilder/pigletbi.gif">
…
<A HREF="Bilder/pooh3.gif">
…
<A HREF="http://www.harju.ee/pooh/part2/pooh2_0.html">Inledning
…
<A HREF="http://www.harju.ee:80/pooh/part1/pooh1_0.html">
…
<A HREF="http://infoweb.magi.com/~datakes/winniepo.html>Länk till länkar</A>
…
<A HREF="http://strindberg.ling.uu.se/~sarag/">Hem till Sara</A>
</HTML>
```

Resultat puh.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running              /usr/local/apps/JDK1.1.1/bin/java              Barbapappa
http://strindberg.ling.uu.se/~sarag/puh.html

Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/pigletbi.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/pooh3.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/
Response-code: 200
Response-message: Document follows


Dåliga länkar:

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://www.harju.ee/pooh/part2/pooh2_0.html
Response-code: 404
Response-message: Not found

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
```

```
Länk: http://www.harju.ee:80/pooh/part1/pooh1_0.html
Response-code: 404
Response-message: Not found

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: <A HREF="http://infoweb.magi.com/~datakes/winniepo.html>
Response-code: 999
Response-message: java.lang.StringIndexOutOfBoundsException: String index out
of range: 56
Kontrollera detta manuellt!
        Förmodligen är det inte ett HTML-dokument  eller så är det inte ett
HTTP-dokument eller liknande!


Länkar att scanna:
```

Figur 23 Jämförelse av länkar

Övriga dokument följer detta mönster. Även vid kontroll av domäner följs detta mönster med det undantaget att Barbapappa endast söker sig en nivå nedåt samt kotrollerar länkarna på de sidor som påträffas där. Både länkar inom respektive utanför den egna domänen kontrolleras.

Barbapappa hittar alla länkar som följer uppfångstvillkoret, se exemplet i figur 14, i övrigt hänvisas till Appendix C samt filen testres.tar.gz. Dock kan det inte anses tillfredställande att eventuella länkar på lägre nivåer inte kontrolleras.

5.5.2 Svarskod/svarsmeddelande, sammanställning

Vid en undersökning av resultaten, se Appendix C samt filen testres.tar.gz, framkommer det att Barbappa levererar en svarskod och ett svarsmeddelande för varje funnen och kontrollerad länk. Detta uppnåddes genom att låta undantagshanteringen generera en svarskod och ett svarsmeddelande då detta inte alltid kunde ske under normala förhållanden. Vid de tillfällen då normala svarskoder inte kan fås genereras koden 999, vilket innebär att det är en felkod som fick genereras av Barbapappa. Vid dessa tillfällen levererar även Barbapappa ett eget svarsmeddelande. För exempel på detta och även de normala svarskoderna och svarsmeddelandena se figur 23.

Resultatet av detta test anses tillfredsställande då inga funna och kontrollerade länkar lämnas utan svarskod eller svarsmeddelande.

5.5.3 Hitta alla hypertextdokument, sammanställning

Som redan nämnts i 5.5.1 så finner Barbapappa alla hypertextdokument som uppfyller uppfångstvillkoret i figur 14. Dock hittar och scannar Barbapappa endast de dokument som ligger en nivå under basdomänen, dvs vi får en kontroll på länkar ned till två nivåer, men endast till en nivå när det gäller att kontrollera hela sidor.

Detta kan inte anses tillfredsställande och beror på att Barbapappa inte byggdes i tillräckligt många och små moduler. Barbapappa byggdes istället som så att funktioner anropade funktioner istället för att låta funktioner rapportera resultat upp till main som sedan i sin tur anroper nästa funktion.

5.5.4 Strategi, sammanställning

Vid en undersökning av resultaten visa det sig att Barbapappa i ett första steg följer den angivna strategin (bredden först). I nästa steg fortsätter Barbapappa att följa strategin. Det skulle kunna tyda på att strategin fungerar. Referenser för testerna återfinns i Appendix C samt i filen testres.tar.gz.

Det är dock otillfredsställande att Barbapappa inte söker mer en nivå nedåt och kontrollerar två nivåer nedåt. Detta innebär att användaren skulle tvingas att köra Barbapappa upprepade gånger för att kontrollera en domän som består av mer än två nivåer under basdomänen, dvs en sida som refererar en sida som refererar en sida.

Detta problem skulle kunna ha lösts genom att ha fler små moduler i Barbapappa istället för några stora som anropar funktioner i sin tur. Som exempel kan nämnas convertTag som anropar respons. Det bästa hade varit om convertTag returnerat resultatet till main som sedan i sin tur fått anropa respons för att få statusen på länkarna. Vidare borde även de vektorer som använts varit globala istället för lokala.

6 Slutsats

Den slutsats man kan dra efter implementeringsarbete och testning är att det är fullt möjligt att implementera en mjukvarurobot som identifiera HTML-länkar som saknar referenser. Detta antyds redan i bakgrundsbeskrivningen av mjukvarurobotar och Java i kapitel 1.3 respektive 1.4. För att lyckas med detta behövs (Se kapitel 4.1)

- något som identifierar länkar ur ett hypertextdokument
- något som kontrollerar länkarnas status
- listor som talar om
    1. vilka sidor som skall scannas
    2. vilka sidor som är scannade
    3. vilka länkar som hade giltig referens
    4. vilka länkar som saknade giltig referens

Dessa villkor uppfylls av

1. Funktionen DraLank som finns beskriven i kapitel 4.1.3
2. Funktionen respons från kapitel 4.1.2
3. Klassen Sidblock för punkt 1 och 2 under punkten för listor samt arraytypen Vector för att lagra information om punkt 1 till 4 (Se kapitel 4.1.4).

Förutom detta behöver man ha någon form av strategi som skall bestämma sökordningen på dokumenten (Se hela kapitel 3.2).

När detta sedan skall sättas ihop skall detta ske på ett sådant sätt att varje enskild funktionalitet byggs in i egen modul som anropas från main i programmet. Dessa moduler levererar sedan sitt resultat till main som i sin tur anropar nästa modul med det senaste svaret som parameter. De vektorer eller listor som man använder bör vara globala så att man inte får ett flertal olika listor med samma namn men i olika objekt.

Följer man dessa mått och steg anser jag att det är/hade varit möjligt att implementera en mjukvarurobot med hjälp av Java för att identifiera HTML-länkar som saknar referenser.

7 Framtida arbete

Som förslag på framtida arbete med utgångspunkt från *Javarobot för att identifiera HTML-länkar utan referenser* finns:

- modifiera Barbapappa så att den blir rekursiv. Se kommentarer om detta i kapitel 4, 5 och 6.
- modifiera Barbapappa så att den klarar av att hantera fler dokumenttyper än HTML, samt klara av andra protokoll än HTTP
- effektivisera Barbapappa med avseende på söktider. Finns det bättre sökalgoritmer? Finns det effektivare/snabbare funktioner i Java? Detta kan bland annat göras genom att rationalisera bort resultaten från länkar som har godkända referenser.
- implementera Barbapappa som en multipel mjukvarurobot
- undersöka om det är möjligt att implementera Barbapappa som en applet
- undersöka om, och om det är möjligt implementera Barbapappa så att den även identifierar länkar som göms i Java-scripts och i Java-applets.

Referenser

[Bra94] Bratko, I. (1994), *Prolog programming for artificial intelligence*, second edition, Addison Wesley Publishing Company

[Cha85] Chartrand, G. (1985), *Introductory Graph Theory*, Dover Publications, Mineola

[Dec94] December, J., Randall N. (1994), *The World Wide Web Unleashed*, first edition, SAMS Publishing, Indianapolis

[Eag94] Eager, B. (1994), *Using the World Wide Web*, Que Corporation, Indianapolis

[Fra96] Franklin, S., Graesser, A. (1996), *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, Institute for Intelligent Systems, University of Memphis

[Hal94] Halsall, F. (1994), *Data communications, computer networks, and open systems*, third edition, Addison Wesley Publishing Company

[Lan94] Langton, C. (1994), *Modeling Adaptive Autonomous Agents*, Artificial Life Journal, Vol. 1, No. 1 & 2. MIT Press

[Shi87] Shiftlet, A. (1987), *Discrete mathematics for computer science*, West Publishing Company, St. Paul

[Www01] *Java-To-Go*, http://ptolemy/eecs.berkley.edu/dgm/javatools/java-to-go/, 17 mars 1997

[Www02] *UMBC Agent Web*, http://www.cs.umbc.edu/agents/, 17 mars 1997

[Www03] *What is Java?*, http://www.javasoft.com/nav/whatis/, 17 mars 1997

[Www04] *Java: The inside story*, http://www.sun.com/sunworldonlime/swol-07-1995/swol-07-java.html, 17 mars 1997

[Www05] *The Java Saga*, http://www.hotwired.com/wired/3.12/features/java.saga.html, 17 mars 1997

[Www06] *The Java Language Environment, A White Paper*, http://www.javasoft.com/doc/language_environment, 17 mars 1997

[Www07] *Att publicera och underhålla information i World Wide Web*, http://www.dsv.su.se/~fk/htmlbok/toc.html, 17 mars 1997

[Www08] *W3C, Mobile Code*, http://www.w3.org/pub/WWW/MobileCode/, 17 mars 1997

[Www09] *LinkScan Home Page*, http://www.elsop.com/linkscan/, 17 mars 1997

[Www10] *Software Agents for Cooperative Environment Administration*, http://arti.vub.ac.be/www/milieu/milieu-memo/milieu-memo.html, 17 mars 1997

[Www11] *Software Agents (Chap 5 of my SM Thesis)*, http://www.ai.mit.edu/people/mhcoen/agents/chapter2_1.html, 17 mars 1997

Referenser

[Www12] *Guidelines for Robot Writers,* http://info.webcrawler.com/mak/projects/robots/guidelines.html, 17 mars 1997

[Www13] *Database of Web Robots, Overview,* http://info.webcrawler.com/mak/projects/robots/active/html/index.html, 17 mars 1997

[Www14] *ACME Java - Software,* http://www.acme.com/java/software/, 17 mars 1997

[Www15] *Guide to Cyberspace 6.1: Contents,* http://www.telstra.com.au/docs/www.guide/, 17 mars 1997

[Www16] *Non-Technical Overview of the Internet, Web, Web Marketing, Web Sites and more,* http://www.resource.net/basics.html, 17 mars 1997

[Www17] *Understanding the Internet,* http://www.blaze.net.au/whatis/intro.html, 17 mars 1997

[Www18] *Entering the World-Wide Web: A Guide to Cyberspace,* http://www.hcc.hawaii.edu/guide/www.guide.html, 17 mars 1997

[Www19] *JATLite,* http://java.stanford.edu/java_agent/html/, 17 mars 1997

[Www20] *KQML Software Design Document,* ftp://gopher.cs.umbc.edu/pub/ARPA/kqml/software/kats/kqml-sdd_ToC.html, 17 mars 1997

[Www21] *KQML Software User's Manual,* ftp://gopher.cs.umbc.edu/pub/ARPA/kqml/software/kats/kqml-sum_ToC.html, 17 mars 1997

[Www22] *Programming Mobile Agents in Java™,* http://www.trl.ibm.co.jp/aglets/aglet-book/index.html, 17 mars 1997

[Www23] *Java..projektarbete SMRO33*, http://jotamac.sm.luth.se/a96c.html/, 17 mars 1997

[Www24] *W3C - The World Wide Web Consortium*, http://www.w3.org/pub/WWW/, 17 mars 1997

[Www25] *The Java Tutorial,* http://www.javasoft.com/docs/books/tutorial/index.html, 17 mars 1997

[Www26] *The Java Development Kit*, http://java.sun.com/products/jdk/1.1/docs/, 17 mars 1997

[Www27] *Examples from Java Network Programming,* ftp://sunsite.unc.edu/pub/languages/java/javafaq/, 17 mars 1997

Index

Index

—T—

—U—

—V—

Appendix A, WWW-källor

Www03, WHAT IS JAVA?

Try the Java applet version

WHAT IS JAVA?
Intro FAQ

by Jason English

It all started with a blunt letter.

In 1990, Sun Microsystems software engineer Patrick Naughton was fed up with trying to support the hundreds of different combinations of software APIs used within the company. When he told CEO and friend Scott McNealy of his plans to accept a job offer from NeXT, McNealy didn't take the news sitting down. He asked Naughton to create a list of his complaints and to suggest a solution "as if you were God."

When Naughton created his list, he didn't pull any punches about Sun's shortcomings. Naughton said the NeWS software architecture the company was working on should be scrapped, and he suggested that of the more than one hundred people working in the Window Systems Group at that time, most of them wouldn't be needed if Sun straightened out the technical mess. After Naughton sent the letter to McNealy, he figured it would be ignored. "Why should I care?" he asked himself. "I'm leaving anyway."

Much to Naughton's surprise, the letter did make a difference. Quietly, it was e-mailed to many of Sun's top software engineers. Naughton's e-mail box was flooded with messages from colleagues who agreed with his assessment of the company's situation. Bill Joy, a Sun founder, and James Gosling, Naughton's mentor at Sun, supported his views and added fuel to the fire by raising many of the same concerns to other senior executives.

The day Naughton was to leave for NeXT, Sun made him a counter offer. The company would create a team of top software developers and free them to do whatever they wanted. The only expected deliverable: make something cool.

The team of six, codenamed Green, went into a self-imposed exile, very much like the scientists on the Manhattan Project. The team stocked the refrigerator with Cokes and Dove bars; discussed what they liked and didn't like about the technologies that were out on the market; and took apart countless electronic devices, such as Nintendo Game Boys, TV set-top boxes and remote controls.

The reason for this free-form exploration of Nintendos and other consumer electronic devices was to find a way for the appliances to talk to each other. The team discovered early on that electronic devices such as VCR's, laser disc players, and stereos were all made with different CPU's. Thus if a manufacturer wanted to add functions or features to a TV or VCR, they were stuck because they were limited by what the hardware and its wired-in programming would allow them to do. This, coupled with the fact that the chips used by many of these devices were limited in program space, suggested a fresh approach to software programming that might be a key to enabling innovation in this product space.

The team's efforts kicked off the development of a new object-oriented programming language that Gosling called Oak, after the tree outside his window. Loosely based on C++, the language was stripped down to a bare minimum in order to be compatible with the limited space the chips in handheld devices would offer, and was designed to allow programmers to more easily support dynamic, changeable hardware.

[Duke] As work on Oak continued, the Green team conducted extensive research into how and why people were attracted to certain video games and how they interacted with various kinds of electronic equipment. After collecting their research data, the team developed a handheld, remote-control-like device with a tiny visual interface. The device, dubbed "*7", featured an animated character named "Duke" who helped guide users through the easy-to-use, image rich, graphical interface remote control. Central to the design of the *7 was the conviction that the interface must be engaging and fun to use, and that the device itself must be a small, personal artifact. "Duke," created by Joe Palrang, would go on to become Java's mascot.

Sun turned the Green team into a wholly owned company called First Person. The new Operating Company had an interesting concept but still no idea what to do with it. After struggling to come up with a marketable idea, the company decided to pursue the interactive television market that seemed to be emerging.

A deal with Time-Warner to create set-top boxes fell through at the last minute. Another potential deal with 3DO was scrapped when that company's chief executive wanted exclusive rights to the technology. Thus First Person's foray into creating set-top boxes for video-on-demand fizzled.

The company's fortune's changed in 1993 when the National Center for Supercomputing Applications introduced Mosaic, and the World Wide Web was born. More web technology soon followed, and the Internet, formerly a home only to computer scientists and educators, began to bustle with traffic.

In early 1994 the First Person team recommended focusing its limited resources on a software system for online multimedia. Bill Joy took that initiative further by positioning Oak as a "language based operating system" and took up Naughton's suggestion to give it away in source form on the Internet. The Oak language itself became the product, instead of part of a device. Arthur van Hoff wrote an Oak compiler entirely in Oak instead of in C. Naughton and Jonathan Payne built an Oak-ready browser called "WebRunner." The first applet -- Duke waving back at his parents over the Internet -- was born.

Sun backed the decision to give the language away, but not before renaming it Java. Much has been made of the now famous name but consider the fact that it could have been called Neon, Lyric, Pepper or Silk.

With Java in the hands of the Internet community at large, all that was needed was a way to run Java applets. "WebRunner" was renamed the HotJava browser because of a trademark conflict. Then, Netscape began supporting Java. Now millions are Java-ready, and Duke has never looked back.

So what exactly is Java?

It is commonly thought of as a way to make Web pages sexy -- incorporating stock tickers, sound or video into Web pages. It has evolved into much more. It is becoming known as a computing platform -- the base upon which software developers can build applications. Developers can build a variety of applications using Java -- traditional spreadsheets and word processors in addition to mission critical applications used by the biggest companies: accounting, asset management, databases, human resources and sales.

Java applications, or applets, are different from ordinary applications in that they reside on the network in centralized servers. The network delivers the applet to your system when you request them. For example, let's say that you want to check your personal financial portfolio. You'd dial in to your financial institution and use your Web browser to log into the bank's system. The portfolio data will be shipped to you along with the applet needed to view it. Let's assume that you're considering moving your money from one account to another. No need to perform a series of cut-and-paste exercises. The system will also send you an applet that will allow you to change the rate of interest and length of investment to perform a series of "what-if" scenarios.

From the corporations' point-of-view, Java will simplify the creation and deployment of applications thus saving money. Applications created in Java can be deployed without modification to any computing platform, thus saving the costs associated with developing software for multiple platforms. And because the applications are stored on centralized servers, there is no longer a need to have people insert disks or ship CD's to update software.

So what will the future hold for companies and their use of Java? Only time will tell, but one thing is certain -- it's unlikely that letters such as the one written by Patrick Naughton complaining about multiple and incompatible software APIs will ever need to be sent again.

For a more detailed history, read Michael O'Connell's SunWorld Magazine story Java: The Inside Story, and David Bank's HotWired account, The Java Saga.

Fortune Magazine has an excellent introduction to the swirling business currents surrounding Sun and Java in Sun's Java: The Threat To Microsoft Is Real by Brent Schlender.

[Image]

Www04, Java: The inside story

Java: The inside story

We interview Java's creators to find what they had in mind.

Abstract

Poised to fill World Wide Web browsers everywhere with animation, audio, and real-time interactivity, Sun's Java language has survived an odyssey through consumer electronics, PDAs, set-top boxes, and CD-ROMs. While some of these areas may yet be exploited

by the language formerly known as Oak, the Internet is Java's launch pad. How'd it get there, and what's its destiny? Will it successfully cross over into the (gasp!) non-Unix marketplace?

(How bad is your Java habit? Check out our Java survey and tell us what you are doing with Java.)

Note: This article was published in July 1995

One would think that a key component in any business's current strategy for success would have been deliberately created based upon a clearly defined mission. But in the case of Sun Microsystems Inc.'s Java programming language -- the environment that turns static Web pages into interactive, dynamic, animated documents bolstered by distributed, platform-independent applications -- it seems the solution preceded the problem.

As Java creator James Gosling explained in a recent interview with SunWorld Online, the genesis of Sun's Web-enhancing technology can be traced to early 1991, when a small group of Sun engineers formed to explore opportunities in the consumer electronics market. At the time, the World Wide Web was still in the drawing rooms.

Consumers vs. engineers
"We were trying to build a distributed system that would make sense as a business [product] ... to sell modern software technology to consumer electronics" manufacturers, Gosling says.

Gosling, 40, joined Sun in 1984 (coming from IBM's research division) and soon afterward began work on the technically impressive but commercially unsuccessful NeWS windowing system. He also wrote GOSMACS, the first EMACS text editor implementation in C.

During this consumer electronics effort, eventually referred to as the "Green" project, Gosling and fellow project engineers learned a great deal about the value of qualities such as reliability, cost, standards, and simplicity -- top priorities in the consumer marketplace. (See the timeline sidebar for additional details.) In contrast to workstation users, who typically want lots of power and will tolerate (and sometimes seemingly demand) high prices, steep learning curves and various bugs in exchange, consumers demand low-cost, bug-free and relatively simple, easy-to-use products.

"Consumers don't care which CPU is inside," says Gosling. They don't appreciate big or powerful RISC-based processors, which are "expensive and proprietary." To compete in the consumer electronics market, companies "treat CPUs as commodities" that can be swapped for lower-cost alternatives nearly instantaneously, and invest resources into backward compatibility and adherence to established standards in long- lived machines, such as toasters and televisions.

Gosling notes that just as modern toasters with embedded electronics employ the same basic user interface as his mother's 42 year-old toaster (which "still works just fine"), so must other consumer electronics products. Television followed a similar path of backward- compatibility when color broadcasts began (today's TV signals can be viewed on 1950s-era black-and-white sets), and faces its next challenge in making the move from analog to digital signals.

Starting with C++
To make development a more platform-neutral process (and thus accommodate the consumer market's demand for CPU flexibility), Gosling began by extending the C++ compiler. Eventually, however, he realized that even with lots of extras, C++ would not suffice. Thus, Oak was conceived in mid-1991. (The name came to Gosling when, while creating a directory for the new language, he glanced out his window, and spotted a tree. But the name didn't survive a trademark search, and was dropped in favor of Java.)

"All along, the language was a tool, not the end," Gosling says. "This was nice in a number of ways. The goal was never 'Let's take on C++,' [but] to build a system that would let us do a large, distributed, heterogeneous network of consumer electronic devices all talking to each other."

In the fall of 1992, after what then-project engineer Patrick Naughton characterizes as "massive amounts of hacking on Oak, the Green OS, the UI, [and the] hardware," among other things, the Green project delivered "*7" (as in the star on a telephone keypad), the PDA-like device that Gosling calls a "handheld remote control."

"In 18 months, we did the equivalent of what 75-people organizations at Sun took three years to do," boasts Naughton -- "an operating system, a language, a toolkit, an interface, a new hardware platform, three custom chips...using new risky technology at every turn. We pulled out all our teeth and put them in each others' mouths."

Naughton, 30, was the project lead on Sun's OpenWindows user environment before joining the secret Green team.

The *7's small form factor helped emphasize the small size and efficiency of the code, which was the core technology. The product was demonstrated around Sun and impressed important people like Scott McNealy and Bill Joy, but the next step was uncertain.

Request for proposal ("Instead of writing a few papers and moving on with life...")
While the team was working on Oak and *7, team members Ed Frank (hardware/technology) and Mike Sheridan (business planning) wrote business and technology road maps for a company in the mold of Dolby Labs that would create and license technology and get its logo alongside Dolby's on consumer electronics products. They had finished several versions of the plans by the time of the *7 demo. But in early 1993, as Sun weighed Java's options, the Green team (now incorporated as FirstPerson Inc.) got wind of a request for proposal from Time-Warner for a set-top box operating system and video-on-demand technology. "A perfect fit," recalls Gosling.

FirstPerson quickly zeroed in on the set-top box OS market, and placed a bid with Time-Warner. But despite having been told that they had the best technology, Sun did not win the bid, due to what Gosling and Naughton characterize as wholly non-technical reasons, such as business politics. "[SGI's Jim] Clark sold his sword to get the deal," Naughton says.

FirstPerson kept trying to pursue set-top boxes until early 1994, when it concluded that "the market wasn't real," Gosling says. "A lot of people hyped things beyond reason." Apparently, the interactive TV market still isn't ripe. Two recent examples: An interactive cable TV trial of 50,000 homes in Omaha, NE put together by U.S. West (network), 3DO (set-top boxes), and DEC (video servers) was cancelled this spring after two years; and Viacom just stopped work on its full-scale test in the San Francisco bay area -- to refocus on applying its miles of cable TV lines to a tried and true market: telephone service.

Naughton says he waged an eventually successful campaign to stop pursuing set-top boxes and instead focus on online services, CD-ROMs, and desktop platforms. FirstPerson dissolved, and about half of its staff moved to Sun Interactive to develop digital video data servers. But a few people still pursued applying Java's technology to network-based desktop systems.

Spinning into the Web
By mid-1994, the World Wide Web was big. "We realized we could build a really cool browser," says Gosling. "It was one of the few things in the client/server mainstream that needed some of the weird things we'd done:

architecture-neutral, real-time, reliable, secure -- issues that weren't terribly important in the workstation world. So we built a browser."

By early fall, Naughton and fellow Sun engineer Jonathan Payne finished writing WebRunner, a Web browser written using the Java language. This early incarnation of HotJava showed off Java in a new light, and a demo impressed SunLabs director Bert Sutherland and Eric Schmidt, Sun chief technology officer -- no doubt in part because they could envision Sun reaping rewards.

What it all means
"The browser equals something that creates a market" for tools, servers, development environments, Gosling says. And Java plays a key role in those tools. "In the pre-Java world, you look at the WWW world, and a page is essentially a piece of paper. In the Java world, a browser becomes a framework. Content providers are empowered to describe behavior and data formats and everything."

More generically, Gosling envisions Java will make people rethink what client/server computing is about. "In the standard model, you have some databases, write a bunch of clients that interact with the database, and build some front end." In this model it can be difficult to organize multiple systems and maintain upgrades, especially when they come from different places, Gosling says.

With Java and Web tools, in contrast, you have inherent organization, Gosling says. "If you build the client side of an application in Java, then launching a client app becomes just switching to a page. Installing is trivial -- just put it on a Web server. And there are no ports, just one version of the application." Already, Gosling says, lots of companies organize databases as Web pages using the Common Gateway Interface (CGI) -- the specified standard for running external programs under an HTTP server.

Looking ahead
At this point, Gosling says the Java language is fairly solid, and he doesn't see any major changes, just some fleshing out. It's the browser that needs work, and that's where the team's efforts are largely concentrated. Gosling says it should be complete by the end of summer. What then? "We've been building a tool, now we'd like to use it," says Gosling -- to build some commercial products, including Web authoring tools.

And none too soon. After the browser, "Sun's biggest challenge is authoring tools" to help develop content, says Dwain Aidala, VP and general manager of Mitsubishi Electronics' North American Multimedia Business Center.

Aidala, whose company has been working with the Java technology in embedded systems for the last two years, says Sun also should make Java a bit lighter and expand its potential applications beyond the Internet and the Web. "Java is limited primarily by how small they can make the interpreter," Aidala says. "It has the potential to go anywhere there's networked processors. ... Telco, CATV, closed systems ... all in the future."

Indeed, Sun acknowledges its efforts to employ Java technology in interactive televisions/set-top boxes, handheld devices, PDAs, telephones, VCRs -- even light switches. "The Internet is the first platform, the perfect way" to introduce Java, says Kim Polese, senior product manager of Java and HotJava.

Still, Java has a ways to go. Of the 713 people who filled out SunWorld Online surveys at its June prototype Web site, only 13 people (1.8 percent) use HotJava -- currently the only choice for taking advantage of Java applets -- as their primary browser. Netscape Communications' agreement to license Java technology means a Java- aware version of the Netscape Navigator browser could appear around the end of the year; with the number-one browser up to speed, Java will become a common component in the broader Web community.

Competition? What competition?
Sun insists it has no direct competition. Although nobody is advertising an equal technology, languages such as Kaleida Labs' ScriptX and Lingo (Macromedia Director's animation scripting language) have come up in the same breath as Java. And General Magic offers somewhat similar technology, its Telescript messaging and agent-based system. But as Polese explains, Telescript differs greatly from Java.

Instead of the server sending information based on client requests, the servers sends things when the server deems appropriate, so code comes to clients without clients asking for it. This raises security questions, Polese says. Since Telescript is aimed at phone networks and PDAs, not desktops, the security issues may not apply.

When one considers the applications of the technology, ever-present Microsoft enters the ring. Beyond the typical browser companies or service providers, and beyond Microsoft's monolithic force and looming Microsoft Network, Microsoft has Visual Basic. While this language lacks the portability of Java, it can run on any Windows-based PC. And while VB applications may never be as light as the heaviest Java applet, they are downloadable. "VBA [Visual Basic for Applications, a built-in programming language for MS applications that replaces older macro languages and provides a unified programming interface to the outside world] is interpreted, extensible and downloadable," says HotJava co-author Patrick Naughton.

"Visual Basic can and will do what Java does," says Naughton, who is now the vice president of technology at Starwave Corp. (Starwave is Microsoft co-founder Paul Allen's Seattle-based interactive consumer products and services company and maker of the popular ESPNET SportsZone Web site, among others.) "Visual Basic already has 30 million users." Who cares about cross-platform issues when you own the PC market?

"Visual Basic makes up for its inferiority as a programming language for serious object oriented design by having a easy to use, visual application construction tool which is already in use by a massive developer community."

"The only thing it doesn't have that matters is security. That's its Achilles' heel -- it's gonna kill 'em." Still, Naughton adds, "PC viruses have been around for 15 years" and haven't prevented the dominance of Windows and DOS. "The issue is that today's viruses have been spread by the rather slow method of floppy disks and downloads from BBSes. Once MSN [the Microsoft Network, due to launch August 24] has people downloading VBA chunks of code as a matter of course, the virus threat is more tangible." In an environment that downloads things for a living, security takes on paramount importance.

In addition to Visual Basic, Microsoft hopes to deliver a tight integration of online network-based services with CD-ROMs. This would allow things like Encarta CD-based encyclopedias to be kept current via supplements seamlessly added via the online network.

"I think this is what Microsoft Network is going to be," says Mark Winther, VP of worldwide telecommunication at IDC's Link Resources Corp., a New York City-based market research and analysis company focusing on the consumer information market. "It doesn't give you live, dynamic, 3-D home pages, but is very powerful and close to what the mass market wants."

Despite all its strengths, Microsoft's concern about Java indicates Sun has the upper hand at this point. "Bill [Gates] knows about and asks about it," says Naughton. "Microsoft sent two guys to pick my brain."

Marketing strategy
Rather than hoarding the technology, Sun has realized the importance of generating broad product interest and acceptance, and therefore freely offers the binaries -- and even the source code -- of key Java components via the Internet.

Sun plans to license Java technology widely to companies such as Netscape that offer Web browsers, online service providers, and software OEMs to "make the long term more solid," Polese says.

"The basic strategy is to license Java to people who have a need for network-centric applications," says Eric Schmidt, Sun's chief technology officer. "The first and obvious target is the browser world. Nothing in the design of Java limits it to Unix or any other operating system. ... It needs to be on all [major] platforms to be successful, and we are going to make sure that happens."

Secondly, Sun is working with third parties to build development tools and object libraries, Polese says, noting that some of these tools are designed for nonprogrammers and offer "completely WYSIWYG" interfaces that let Web

creators do things such as drag and drop images and other objects into pages.

"There's no reason why layers can't be built on top" of Java, ranging from scripting tools to more sophisticated tools such as RAD Technologies' PowerMedia authoring tool or Dimension X's animation tools, Polese says.

"What we would really like to see are new types of applications developed using this technology," Schmidt says. "We are trying to avoid the fate of NeWS [Sun's proprietary windowing environment that lost a standards battle with the X Window System] by working more aggressively with everyone in the industry. I think the terms are low enough and the value high enough that we have a good chance of having most of the movers and shakers sign up."

Perhaps these combined efforts will cause tomorrow's Internet -- as well as other networked computer environments -- to overflow with ubiquitous Java objects.

"The people [Sun] needs to market Java to are not those reading [programming-related newsgroups]," says Naughton, but less sophisticated users. "If average consumer can see these things and use them, they'll demand it. Naughton says Sun needs to land deals with major commercial players such as America Online and CompuServe.

"The key is not Sun, but how many service providers and publishers use Java on their servers," says IDC/Link Resources' Winther. "I see no reason why an increasing number of new sites won't employ it. ... It's real powerful."

About the author
Since writing this cover story, Michael O'Connell has launched JavaWorld (http://www.javaworld.com), IDG's monthly Web-only magazine for Java developers and professionals, where he is editor-in-chief. Previously, he was an editor with SunWorld and a reviews editor with Advanced Systems magazine.

Don't forget to check out our Java survey.

What did you think of this article:
-Excellent! -Okay -Poor [-----] -Too long -Just right -Too short

---

URLs included in this article:

* Dimension X http://www.dnx.com
* General Magic's Magic Cap and Telescript
  http://www.genmagic.com/MagicCapDocs/Concepts/Telescript.html
* Java applet programming contest
  http://java.sun.com/contest/external.html
* Java applets http://java.sun.com/applets/
* Java/HotJava mailing lists http://java.sun.com/mail.html
* Java Web Server http://java.sun.com/
* Kaleida http://www.kaleida.com/
* Macromedia http://www.macromedia.com/Tools/DMS/index.html
* RAD Technologies http://www.batnet.com/RAD/
* Starwave Online Services http://www.starwave.com/

[Feedback Form]
[Table of contents] [Sun's homepage] [Next story]

If you have problems with this magazine, contact webmaster@sunworld.com

---

[Lessons Sun learned from the consumer market] [Back to story]

Because of the consumer electronics industry's utter lack of commitment to particular processors, Gosling realized early on that "apps couldn't be compiled to a specific CPU." He and fellow Sun engineers learned a few other key lessons as well. In the consumer market, product developers must:

* Allow interfaces to evolve gracefully.
* Tackle security and reliability problems.
* Recognize that multitasking helps a lot; existing standards ("which come from the teletype of 25 years ago") don't.
* Consider alternatives to C and C++, whose features encourage creation of unreliable, fragile code.

[Lessons Sun learned from the consumer market] [Back to story]

[Java's evolution: a timeline] [Back to story]

Java's development as outlined by Patrick Naughton, co-author of the HotJava browser and current VP of technology at Starwave Corp.

December 5, 1990
Naughton turns down offer to join NeXT, starts work on what becomes the "Green" consumer environment project at Sun.

January 15, 1991
"Stealth Project" (as named by Scott McNealy) brainstorming meeting in Aspen with Bill Joy, Andy Bechtolsheim, Wayne Rosing, Mike Sheridan, James Gosling and Patrick Naughton.

February 1, 1991
Gosling, Sheridan, and Naughton begin work in earnest. Naughton focuses on "Aspen" graphics system, Gosling on programming language ideas, Sheridan on business development.

April 8, 1991
Move off-site to 2180 Sand Hill Road; break direct LAN connection (and most communication) to Sun; project settles upon the name "Green."

April 15, 1991
Ed Frank (a SPARCstation 10 architect), Craig Forrest (SS10 chip designer), and Chris Warth (NeWS software developer) join Green.

May 1991
Ed Frank coins *7 (pronounced "star-seven") as the hardware prototype's name. (*7 was was the code you pressed in the Sand Hill office to answer any ringing phone from any telephone.)

June 1991
Gosling starts working on the "Oak" interpreter, which, several years later (following a trademark search), is renamed "Java."

August 1, 1991
Oak and Aspen merge and run their first real program.

August 19, 1991
Green team demonstrates basic user interface ideas and graphics system to Sun co-founders Scott McNealy and Bill Joy.

October 17, 1991
Sheridan and Naughton coin "1st Person" as the name for the team's design philosophy, and eventually the name of the company.

November 17, 1991
Green reconnects to Sun main network with 56K line.

March 1, 1992
Jonathan Payne, who later co-wrote HotJava, joins Green.

Summer 1992
Massive amounts of hacking on Oak, the Green OS, the UI, the Star7 hardware, and related components.

September 4, 1992
Star 7 device is complete, and demonstrated to Joy and McNealy.

October 1, 1992
Wayne Rosing joins from SunLabs (which had formed in July 1990) and

assumes management of the team.

November 1, 1992
FirstPerson is incorporated.

January 15, 1992
The team moves to 100 Hamilton Avenue in Palo Alto. Poetically, this is the former DEC Western Research Lab, where the original "Hamilton Group", (aka OSF) was founded.

March 15, 1993
FirstPerson focuses on interactive television after learning about Time Warner's RFP for its interactive cable TV trial in Orlando, FL.

April, 1993
NCSA Mosaic 1.0, the first graphical browser for the Internet, is released.

June 14, 1993
Time Warner goes with SGI for its interactive cable TV trial, despite acknowledged superiority of Sun technology and assurances in mid-April that Sun won the deal.

Summer, 1993
Naughton flies 300,000 miles selling Oak to anyone involved in consumer electronics and interactive television; meanwhile, the rate at which people are gaining access to the Internet reaches breakneck speed.

August, 1993
After months of promising negotiations with 3DO to provide set-top box OS, 3DO president Trip Hawkins offers to buy technology outright. McNealy refuses, and deal falls through.

September, 1993
Arthur Van Hoff joins team, originally to do application development environment aimed at interactive television; ends up doing mostly language design.

December 7, 1993
A high-level operations review of FirstPerson finds the group has no real partners or marketing strategy and a looming launch date.

February 8, 1994
FirstPerson's planned public launch at Richard Saul Wurman's Technology, Entertainment and Design Conference, "TED5," is canceled.

February 17, 1994
Alternative FirstPerson business plan for doing CD-ROM/online multimedia platform based on Oak presented to Sun executives to very mixed reviews.

April 25, 1994
Sun Interactive created, half of FirstPerson employees leave to join it.

June, 1994
"Liveoak" project started. Designed by Bill Joy to use Oak for a big small operating system project.

July, 1994
Naughton reduces the "Liveoak" project's scope to simply retargeting Oak at the Internet after writing a throwaway implementation of a Web browser in a long weekend hack.

September 16, 1994
Payne and Naughton start writing "WebRunner," a Mosaic-like browser later renamed "HotJava"

September 29, 1994
HotJava prototype is first demonstrated to Sun executives.

October 11, 1994
Naughton leaves for Starwave.

Autumn, 1994
Van Hoff implements Java compiler in Java. (Gosling had previously implemented it in C.)

May 23, 1995
Sun formally announces Java and HotJava at SunWorld '95.

[Java's evolution: a timeline] [Back to story]

[Feedback Form]
[Table of contents] [Sun's homepage] [Next story]

If you have problems with this magazine, contact webmaster@sunworld.com
URL: http://www.sun.com/sunworldonline/swol-07-1995/swol-07-java.html
Last updated: 1 July 1995

Www05, The Java Saga

Join the HotWired Network, it's free. Members log in.

The Java Saga

by David Bank

With three minutes to go before the midnight deadline in August 1995, Sun Microsystems engineer Arthur van Hoff took one last look at Java and HotJava, the company's new software for the World Wide Web, and pondered what his colleagues call Arthur's Law: Do it right, or don't do it. Satisfied, the Dutch programming wizard encrypted the files containing the software's source code, moved them to an Internet site, and e-mailed the key to Netscape Communications Corporation, Java's first commercial customer. Five years after the project was launched, Java was done Ð with a minute to spare.

As he sat at his workstation ready to push the button, van Hoff had good reason to hesitate. Since early versions of the software were released in December 1994, Java has unleashed stratospheric expectations. While today's Web is mostly a static brew Ð a grand collection of electronically linked brochures Ð Java holds the promise of caffeinating the Web, supercharging it with interactive games and animation and thousands of application programs nobody's even thought of. At the same time, Java offers Sun and other Microsoft foes renewed hope that Bill Gates's iron grip on the software business can be pried loose. Microsoft rules the desktop, but as networking expands its role, says van Hoff, Java could turn out to be "the DOS of the Internet." Indeed, Sun is rushing to make Java a de facto standard on the burgeoning Web. If Sun succeeds, even Microsoft will have a hard time muscling in.

Software developers are busy shaping Java into applications that will add new life to Web browsers like Netscape and Mosaic, producing programs that combine real-time interactivity with multimedia features that have been available only on CD-ROM. (Java is a programming language, HotJava an "interpreter" installed onto a browser, enabling Java programs delivered over the Web to run on the desktop.) What's a Java application? Point to the Ford Motor website, for instance, and all you'll get are words and pictures of the latest cars and trucks. Using Java, however, Ford's server could relay a small application (called an applet) to a customer's computer.

From there, the client could customize options on an F-series pickup while calculating the monthly tab on various loan rates offered by a finance company or local bank.

Add animation to these applications and the possibilities are endless. Hollywood and Madison Avenue are salivating. "Java allows us to do the things that advertisers and studios are asking us to do," says Karl Jacob, CEO and chief technologist at Dimension X Inc., a San Francisco company creating 3-D websites using Java. "Until now, everything on the Web was fizzling, not sizzling."

Even if Java turns piping hot, how might it lift profits at Sun, which turns out Unix-based workstations and servers for its bread and butter? It's rumored that Netscape paid a paltry US$750,000 to license HotJava (escaping any per-copy charges), a figure that Sun, whose annual revenues will top $6 billion this year, does not dispute. Sun is giving away Java and HotJava free for noncommercial use, in a fast-track attempt to make them the standard before Microsoft begins shipping a similar product, codenamed Blackbird, in early 1996.

Java is unlikely ever to become a major profit center at Sun, though any increase in Web traffic is bound to increase sales of Sun's workstations and servers. But in this case, emotion may be at least as important as profit. Sun chief Scott McNealy is a fierce competitor, and his blood lust for Bill Gates has fueled the Java project from the beginning. McNealy is especially excited about Java's ability to run on any computer, using Windows, Mac OS, Unix, or any other operating system Ð posing a threat to Microsoft hegemony. Spinning into the future, McNealy even sees the day when disposable word processors and spreadsheets will be delivered over the Web via Java, priced per use. "This blows up Gates's lock and destroys his model of a shrink-wrapped software that runs only on his platform," effuses McNealy.

Maybe he's dreaming. But Java's progression thus far is a lesson in what can happen when a major company loosens the reins on some of its most precocious talent. The story of Java also highlights the sometimes serendipitous nature of technological development in the face of vague and fast-changing markets.

The origins of Java go back to 1990, when the World Wide Web was barely a glimmer in a British programmer's eye. The personal computer was in its ascendancy, and many inside and outside Sun thought the company had missed major opportunities in the desktop market. Its high-end workstation and server markets were rolling along fine, but as PC use spread across the landscape, the company faced being stranded in a narrowing slice of the computer market. Sun machines had a reputation for being too complicated, too ugly, and too nerdy for mass consumption.

Thus, McNealy was more than ready to listen when a well-regarded 25-year-old programmer with only three years at the company told him he was quitting. Patrick Naughton played on McNealy's ice hockey team. Over beers, Naughton told McNealy that he was quitting to join NeXT Computer Inc., where, he said, "they're doing it right." McNealy paused for a second then shrewdly asked Naughton a favor. "Before you go, write up what you think Sun is doing wrong. Don't just lay out the problem. Give me a solution. Tell me what you would do if you were God."

The following morning, Naughton threw his heart and soul into the challenge.

He typed out a list of Sun's short comings along with his own glowing appraisal of NeXT's critically acclaimed NeXTstep operating system. Twelve screens later, he e-mailed his report to McNealy, who forwarded it to the entire management chain.

A firestorm was ignited. Among Naughton's suggestions: hire an artist to pretty up Sun's uninspired interfaces; pick a single programming tool kit; focus on a single windows technology, not several; and, finally, lay off just about everybody in the existing windows group. (Naughton figured they wouldn't be needed if the previous suggestions were taken.)

Naughton held off NeXT while he awaited the response. The following morning, his e-mail box was bursting. Hundreds of CC'd readers had read his recipe for what ailed Sun and had agreed in a resounding chorus. A typical reaction: "Patrick wrote down everything I say to myself in the morning but have been afraid to admit." Another voice was that of James Gosling, a remarkable programmer whose opinions carried great weight higher up. Naughton was "brutally right," Gosling e-mailed. "Somewhere along the line, we've lost touch with what it means to produce a quality product."

Naughton joined what one participant called a "bitchfest" attended by a number of high-level engineers. It was John Gage, Sun's science office director, says Naughton, who really dug in, asking, "What is it you really want to do?" The group blue-sky'd until 4:30 the next morning. During those wee hours, they came up with some core principles for a new project: consumers are where it's at; build a small environment created by a small team Ð small enough to fit around a table at a Chinese restaurant; and make the environment, whatever it may become, include a new generation of

machines that are personal and simple to use Ð computers for normal people.

If still vague, these principles were enough to get Gage's executive juices flowing. With his support, Naughton pitched the high concept to Wayne Rosing, then president of Sun Laboratories Inc. and onetime vice president of engineering at Apple. Naughton laid down key demands he'd scribbled on the back of a restaurant place mat: the project would be located offsite, away from corporate "antibodies" well known for attacking innovative ideas; the project's mission would be kept a secret from all but the top executives at Sun; the software and hardware designs would not have to be compatible with Sun's existing products; and for the first year, the team would be given a million bucks to spend.

Rosing took the idea to McNealy over dinner, and afterward called his assistant by car phone. Via e-mail, the assistant relayed Rosing's pledge to Naughton: "I expect to make 100 percent delivery on what we discussed." Within a day, two of Sun's top dogs, Bill Joy and Andy Bechtolshiem, were throwing in their approvals. That same day, Naughton got his wish. Naughton, Gosling, and Mike Sheridan, who had come to Sun after it bought out his start-up, would be given carte blanche to pursue new projects.

But what projects? At this point the team, codenamed Green, had only a vague notion of what it would do. Competing head-on with Microsoft was out: the Goliath had already won the battle for the mass-market desktop. Instead, the team resolved to bypass Microsoft and the PC market altogether by designing a software system that could run anywhere, even on devices that people did not yet think of as computers. That meant the system had to be compact and simple Ð the complete opposite of Sun's existing offerings. "We wanted computers to go away, to instead become an everyday thing," Naughton said. "We thought the third wave of computing would be driven by consumer electronics. The hardware would come from Circuit City, and the software would come from Tower Records."

But Green was still a solution in search of a problem. The shared epiphany that set the team's early direction came in the spring of '91 in a hot tub near Lake Tahoe, where Sun's high-level staff had gathered for an annual retreat. Gosling, Sheridan, and Naughton, now joined by Ed Frank, one of Sun's top hardware engineers, soaked and drank beer. Gosling made the observation that computer chips were appearing in toasters, VCRs, and many other household appliances, even in the doorknobs of their Squaw Valley ski-lodge rooms. "That's getting pretty ubiquitous when it's in the bloody doorknob," he said. Yet three remote-control devices were needed just to get a television, a VCR, and a living-room sound system to work. Needless to say, most people still couldn't program any of them. The wonder wasn't that chips were everywhere but that they were being used so badly.

"With a little computer science, all of these things could be made to work together," Gosling insisted. A light switch with a liquid crystal display and a touch pad could play little movies to demonstrate what it controlled and how, he brainstormed. Any control device could do multimedia, and multimedia could help people do real, useful things. There in the hot tub, Gosling recalls, the Green team decided to build a prototype of a device that could control everyday consumer appliances.

Thus began Green's glory days. In April 1991, the team moved from Sun's main campus to office space above a branch of the Bank of America on Menlo Park's Sand Hill Road, cutting itself off from Sun's internal computer network. The programmers disconnected culturally as well. "We thought if we stayed over there, we would end up with just another workstation," Sheridan says. "I was obnoxious about keeping it secret."

They cleared the center of the large room for lab benches and couches, stocked the refrigerator with Dove bars and Cokes, and spent hours playing Nintendo games Ð the better to understand hypnotically engaging user interfaces. Business issues were put on the back burner to give the technical ideas room to roam. Their mission statement was laid down in a business plan they called Behind the Green Door: "To develop and license an operating environment for consumer devices that enables services and information to be persuasively presented via the emerging digital infrastructure."

The key insights into the software that would run such devices came to Gosling at a Doobie Brothers concert at the Shoreline Amphitheater in Mountain View, California. As he sat slouched in front-row seats letting the

music wash over him, Gosling looked up at wiring and speakers and semirobotic lights that seemed to dance to the music. "I kept seeing imaginary packets flowing down the wires making everything happen," he recalls. "I'd been thinking a lot about making behavior flow through networks in a fairly narrow way. During the concert, I broke through on a pile of technical issues. I got a deep feeling about how far this could all go: weaving networks and computers into even fine details of everyday life."

Gosling quickly concluded that existing languages weren't up to the job. C++ had become a near-standard for programmers building specialized applications where speed is everything Ð computer-aided design for instance, where success is measured by the number of polygons generated per second. But C++ wasn't reliable enough for what Gosling had in mind. It was fast, but its interfaces were inconsistent, and programs kept on breaking. However, in consumer electronics, reliability is more important than speed. Software interfaces had to be as dependable as a two-pronged plug fitting into an electrical wall socket. "I came to the conclusion that I needed a new programming language," Gosling says.

As it happened, Gosling, who wrote his first computer language at 14, had been working on a C++ replacement at home. "From the initial 'Oh, fuck' to getting it to a reasonable state" took only a few months, he says. Naughton, meanwhile, had been working on graphic animations, which would serve as the device's interface. By August 1991, Gosling had the graphics running in his new language, which he called Oak (named for the tree outside his office window); this was the progenitor of Java.

By now, the Green team's ambition was to build a device that would work as an interface to cyberspace. Its aim was to create a visual interface to a virtual world. If you wanted to record a TV program while you were away from home, you could control your video by working a virtual video in the virtual world Naughton was designing. The virtual world Ð in color and in 3-D Ð was written in Oak language and spruced up by graphic artists.

All the new programs needed now was a device to run them on. The team wanted a working box, small enough to hold, with batteries included. To build one, the members trotted out what they call "hammer technology"; as Naughton describes it, this involved finding "something that has a real cool 'mumble' (a neat piece of hardware). Then you hit it with a hammer, take the mumble off, and use it. We got a consumer-grade Sharp minitelevision, hit it with a hammer, and got an active-matrix color LCD. We put a resistive touch screen on the front, making sure there'd be no moving parts on the system, no buttons, no power switches, nothing," Naughton explains. The team then wanted to add stereo speakers inside, but couldn't find any to fit the case. "We went to Fry's and bought a dozen Game Boys, played like mad for about three hours, then broke them open Ð that's where the speakers came from."

The guts of the device were even more remarkable: one of Sun's high-end Sparc workstations stuffed into a dark green aluminum case barely bigger than a softball. Frank's hardware team built three custom chips and designed a motherboard that folded in on itself to save space. The team hacked furiously all through the summer of 1992. "It was a blood bath," Naughton says. "We bit off way more than any seven people should chew. We were arrogant sons of bitches to think we could pull it off. We had so many free variables that we had nothing to come back to. We didn't have anything we knew would work."

The demo was shown to McNealy in August 1992. McNealy saw a hand-held contraption with a small screen and no buttons. When you touched the screen, it turned on. Cool! It opened to a cartoon world Ð no menus! A character named Duke, a molar-shaped imp with a big red nose, guided the user through the rooms of a cartoon house. You steered with your finger Ð no mouse! Sliding your finger across the screen, you picked up a virtual TV guide on the sofa, selected a movie, dragged the movie to the cartoon image of a VCR, and programmed the VCR to record the show. This was even more elegant than it first sounded. Everything was done without a keyboard, simply by ripping objects with your finger and dropping them with a "ka-ching" sound.

Sun's boss was ecstatic. Nothing like this smooth, natural interface existed at the time (no Bob, no Magic Cap, no eWorld). And nothing like Oak. A natural cheerleader, McNealy dashed off a hyperactive e-mail. "This is real breakthrough stuff. Don't fail me now. We need to sell this puppy hard, and there's tons of work to make it real. You deliver, it will win.

I can sell this stuff. Charge! Kill H-P, IBM, MSFT, and Apple all at once." Rosing added his own exudations: "Bill Gates is gonna weep."

The prototype was not the only thing the Green team brought to the table. Oak was more than cartoons. Oak was to be an industrial-strength object-oriented language that would work over networks in a very distributed manner. Little packets of code (objects) would scoot around the Net, functioning independently of the devices they were in (computer, telephone, or toaster). Entire applications, like an e-mailer, say, could be built by stitching together objects in a modular way Ð and the objects wouldn't all have to live in the same place. Plus, Oak dealt with a chief concern of distributed computing: it encased security, encryption, and authentication procedures into its core so security was essentially invisible to users.

At demos, Naughton would go to the white board to show the scope of Oak, filling the blankness with lines crisscrossing from home computers, to cars, to TVs, to phones, to banks, to Ð well, to everything. Oak was to be the mother tongue of the network of all digital things.

The Green team had been talking to Mitsubishi Electric about using Oak-based interfaces in cellular phones, televisions, and home and industrial automation systems. France Telecom, which was looking to upgrade its Minitel system, was also interested, Sheridan says. He followed up with a detailed business plan dubbed Beyond the Green Door, proposing that Sun establish a separate subsidiary to push the Oak technology into the consumer marketplace.

Two months after the demo, Sun set up the team as FirstPerson Inc., a wholly owned subsidiary. Rosing moved over from Sun Labs to head the project. Meanwhile, Sheridan, passed over and feeling disrespected, cleaned out his desk and went home. "I thought that having brought it so far, I should be running it," says Sheridan, who now heads a technology consulting group in Virginia. Gosling and Naughton stayed. "There was a changing of the guard, and it wasn't exactly smooth," Gosling remembers.

As FirstPerson grew from 14 employees to more than 60 and moved from its one-room hideout to palatial offices in downtown Palo Alto, things began to drift.

Mitsubishi and France Telecom weren't interested after all, nor was anyone else. Sun figured the cost of the chip, memory, and display device needed for Oak could be squeezed to about $50, but consumer electronics makers were used to paying nominal bucks for chips that made their products easier to use.

In the meantime, the notion of the information superhighway had taken over rational minds and declared itself to be arriving Real Soon Now. Most people understood the I-way to be interactive TV. So when Time Warner circulated proposals in March 1993 to begin interactive television trials in Orlando, FirstPerson jumped at the chance to provide the set-top boxes. As far as Sun was concerned, interactive TV was the technology of the moment, and the company was desperate to use Oak as its entry. Naughton, Gosling, and Joy, who had taken a sporadic interest in Oak, trekked many times to Time Warner's technology center in Denver. There they labored over prototype designs and hashed out cost estimates for a set-top box that would link a television to the information superhighway, using Oak to coordinate a vast complex of images, data, and money securely over a distributed network.

But the deal went to Silicon Graphics Inc., Sun's cross-town rival in the high-end workstation market. Around Sun, McNealy took heat for not pushing as hard with Time Warner as Jim Clark, SGI's founder and then-chair. "All that mattered to Time Warner was who was going to commit to delivering a $300 box on top of the TV," McNealy grouses. "Nobody knows how to do a $300 set-top box that does what they want it spec'd to do." SGI then delivered a box that cost almost 10 times that amount in the summer of '93.

In hindsight, losing the deal was a stroke of good luck, given the overruns, glitches, and weak consumer interest that made the Orlando project a disaster for both Time Warner and SGI. (Clark later left SGI, abandoning interactive TV in favor of the emerging World Wide Web, and founded Netscape Ð whose browser would later incorporate Java, the successor to Oak. But more of that in a moment.)

A few months later, FirstPerson got tantalizingly close to a deal with 3DO, a company having trouble selling an expensive CD-ROM game machine and thus trying to double the product as a set-top box. It took just 10 days to get Oak running on one of 3DO's game boxes but three months to negotiate a commercial agreement. Finally, the paperwork was ready, and Trip Hawkins, the founder of 3DO, weighed in demanding exclusive rights to the technology. McNealy refused. Given 3DO's precarious state Ð its expensive game machines had languished on store shelves for two years Ð that loss turned into a blessing, too.

There were few new avenues left for FirstPerson to try. "What was really dumb for us was focusing on set-top boxes and putting on blinders," Gosling said. "Interactive TV was a mistake. There was so much enthusiasm about it we didn't understand the unreality of that universe."

Those blinders kept Sun Ð along with many others Ð from grasping the significance of another emerging phenomenon. In June 1993, Marc Andreessen and Eric Bina at the National Center for Supercomputing Applications at the University of Illinois had released the first version of the Mosaic browser, and the formerly obscure World Wide Web began to take off. But it was at least three more months before Eric Schmidt, Sun's chief technology officer, saw the software for the first time. How had Sun, so long a supplier of well over 50 percent of all host computers to the Internet, missed the Web's mass appeal? "We just took our eye off the ball," Schmidt admits.

FirstPerson was in disarray. Nerves were frayed. Marching orders came down from Sun management: Find something to produce profits. Now! A new business plan was drawn up early in 1994, which unceremoniously dumped the speculative markets FirstPerson had pursued and began focusing on personal computers Ð the `technology the project was supposed to leapfrog in the first place. The new plan was to create a corps of CD-ROM developers who would write in Oak and, ideally, stick with it as their platform language while moving applications to the commercial online services. Eventually, by the turn of the century, the team thought, broadband networks for interactive TV would finally be ready, and Oak would be established as the programming language of choice. The plan, remarkably, contained no mention of Mosaic or the Web.

For different reasons, the plan met with little enthusiasm among Sun's top executives. They wanted a strategy that would drive demand for Sun's core hardware products, something a program for CD-ROMs would not do. With no profitable strategy in sight, FirstPerson was scrapped in the spring of 1994, the set-top box project shelved, and the interactive TV crew, now renamed Sun Interactive, collapsed into an alliance with Thomson Consumer Electronics to develop scaled-down box and video servers Ð without Oak for now. It took Bill Joy, who again turned his attention to the project, to rescue Oak.

Joy, a Sun co-founder, is a rare hybrid: a legendary programmer who understands every line of code and carries enormous clout with execs. He is described lovingly by colleagues as a brilliant wild man whose ideas are at the lunatic fringe. Four years ago, annoyed by Silicon Valley area traffic after the Loma Prieta earthquake and disillusioned about the prospect of doing anything interesting until the Microsoft juggernaut had run its course, Joy established a small Sun research lab at the foot of the ski mountain in Aspen, Colorado. (The town also had a good bookstore, he says, explaining his choice of locale.) Tired of disputes within Sun, he had mostly disengaged, though he dropped in occasionally on the FirstPerson project.

The Web's sudden emergence changed all that. For Joy, the prospect of bringing Oak to the Internet recalled his days at the University of California, Berkeley, nearly two decades earlier, when he'd developed the Berkeley flavor of the Unix operating system out of the original code from Bell Labs and pushed it into widespread use through the Net. That had laid the foundation for Sun. Because of Joy, Sun finally saw that the Internet could become Oak's redemption. Joy's support was critical in what became known as the Internet Play, the "profitless" approach to building market share Ð a ploy Netscape had made famous by giving away its browser. "There was a point at which I said, 'Just screw it, let's give it away. Let's create a franchise,'" Joy says.

Joy and Schmidt wrote yet another plan for Oak and sent Gosling and Naughton back to work adapting Oak for the Internet. Gosling, whom Joy calls "the

world's greatest programmer," worked on the Oak code, while Naughton set out to develop a true "killer app."

In January 1995, Gosling's version of Oak was renamed the more marketable Java. Naughton's killer app was an interpreter for a Web browser, later named HotJava. He wrote the bones of it in a single weekend. Following Joy's dictum, they intended to make it available free on the Web.

But Naughton and Gosling didn't entirely trust Sun to turn Java over to the Internet. Sun has always been a proponent of open standards for software interfaces that allow anyone to build his or her own compatible applications. But this strategy was going further to include the free release of a software implementation. Kim Polese, Java's senior product manager, wrote in big letters on her office white board, "Open means...." and kept adding items to the list. It was one thing for college hackers to release university-grade software like Mosaic for free, or even for a start-up like Netscape to offer its browser for free noncommercial use. It was something else for a major company such as Sun to give away its technological crown jewels, the source code for some of its most valuable technology.

Even Schmidt had doubts about whether he'd be able to live up to his promise to protect the team from the pressure at Sun. "The conversation that never took place, but that I could feel all around me, was, 'Eric, you are violating every principle in the company,'" Schmidt says. "'You are taking our technology and giving it away to Microsoft and every one of our competitors. How are you going to make money?' At the time, I didn't have an answer. I would make something up. I would lie. What I really believed was that Java could create an architectural franchise. The quickest way was through volume and the quickest way to volume was through the Internet."

In December 1994, Java and HotJava (at this stage still called Oak) were posted in a secret file deep in the Net; only a select few were given pointers and invited to check it out. Three months later, Marc Andreessen, who had gone on to start Netscape with Jim Clark, was given a copy. Andreessen gushed to the San Jose Mercury News: "What these guys are doing is undeniably, absolutely new. It's great stuff." That was how the Java team knew it was going to finally make it. "That quote was a blessing from the god of the Internet," Polese says.

Now that HotJava is being given away, Sun has to make sure it cements Java as a standard Ð and then figure out how Java can make money. The Netscape deal Ð incorporating Java into its browser Ð helps establish a population of Java users. But Sun has to go much further to make it easy enough for anyone other than hard-core nerds to populate the Web with applets.

Sun has promised, but not delivered, tool kits that will allow artists, writers, and other would-be Web authors to speak Java fluently. That's what's needed to increase the supply of enticing applications and to spur users to demand that software suppliers include the technology in their offerings. And Sun has to accelerate that circle into a dizzying spin before competing technologies come along to challenge Java's position. "Sun's window is six to twelve months," says Sheridan. "They need to move quickly because Microsoft will respond in a way that freezes development."

The Java applets are the key. Here's why: for a program to run on a computer, it must first be translated from a language like Basic or C into the machine's native tongue. Because this translation process is incredibly time-consuming, most software comes already translated. But that means different versions have to be created for different computers. Java gets around this problem by using an intermediate language Ð a sort of Esperanto that is not machine specific but that can quickly be interpreted by any computer.

The result is that small programs Ð applets Ð can fly around the Net without regard to what kind of hardware they end up on. If you need to watch an animation that requires a particular fancy doodad to run it, but you don't have that doodad, your machine will pick up the Java-coded applet along with the animation file and run both. Who cares where the software lives? Who cares what kind of machine you have? Who cares about Microsoft?

Microsoft's response will be Blackbird, a package due for initial release in January 1996 that will contain a Web application programming language and an interpreter of its own, at first based on C++ and later on Microsoft's own

Visual Basic. Unlike Java, Blackbird's language will work only on the Windows platform at first, but that may not be such a problem given Windows's 80 percent share of the PC market. Anyway, Microsoft is planning a Mac version. And it also claims that Blackbird will be easier to use than professional-programmer-oriented Java.

Sun is pursuing licensing deals as fast as it can to set Java as the standard before Blackbird flutters in. It recently penned an agreement with Toshiba to use Java on a wireless Internet device and claims to have more than 25 potential Java agreements in the pipeline. Though the company says the low-price licensing deal with Netscape is a onetime exception, Sun is explicit about making its technology cheap. The published rates for licensing Java's source code for commercial use include a $125,000 upfront fee plus $2 a copy. "It's priced below our cost," Schmidt says. "This loses money in the licensing business for the foreseeable future. It's a strategic investment in market share."

Another way to avoid that fate would be to license Java to Microsoft and thus complete the penetration of the entire market. Schmidt says he's willing, but Bill Gates hasn't called. Others doubt whether McNealy could bring himself to consort with the enemy even if Gates showed up at the door. They believe that Sun's desire to beat Microsoft may be even stronger than its desire to see Java succeed. "There are cheaper ways to say 'Fuck you' to Bill Gates," says Naughton, who left Sun last year after he became, as he puts it, "damaged goods" during FirstPerson's nadir. After Sun showed him the indignity of a 2 percent raise, he says, Naughton joined Starwave Corp. in Seattle, where he is using Java to create online services.

Sun is racing to stay ahead of the accelerating wave. The day after that midnight deadline for sending the finished code to Netscape, Joy was already at work pushing the limits of what Java could do. His team was aiming a videocamera at a computer-controlled water fountain in Aspen and putting its image on the Web. The idea was to let anybody with a HotJava interpreter anywhere on the Internet control the spray and interact with kids playing in the water. "I've got 15 patents I could file as soon as I type them," Joy says. "I figure I've got five years. It's like we've got a blank sheet and it says 'Internet.' Normally, the best products don't win. The Internet is an opportunity for the best products to win. Java is great technically and people want it. I'm happy to get that once in my life Ð or maybe twice."

Often, great technologies are born into the world without one of three essential factors for success: a committed champion, a willing marketplace, and a workable business model. Clearly, Java had its champions Ð believers like James Gosling, Arthur van Hoff, Bill Joy, Patrick Naughton, and the many who fell by the wayside during Java's long, twisting history. Tweaked, renamed, and repositioned, this time the idea has found a willing marketplace; in time, a distributed object-oriented language like Java will probably establish itself as the foundation of the Net. But whether the standard will be Java depends on whether Sun finds a business model to keep it alive.

Www27, PageSaver.java

```
import java.net.*;
import java.io.*;

public class PageSaver {

  URL theURL;

  public static void main (String args[]) {

    // Loop through the command line arguments
    for  (int i = 0; i < args.length; i++) {

      //Open the URL for reading
      try {
        URL root = new URL(args[0]);
        PageSaver ps = new PageSaver(root);
        ps.saveThePage();
      }
      catch (MalformedURLException e) {
        System.err.println(args[0] + " is not a parseable URL");
        System.err.println(e);
      }
    } //  end for
```

```
} // end main


public PageSaver(URL u) {

  theURL = u;

}

// saveThePage opens a DataInputStream from the URL,
// opens a PrintStream onto a file for the output,
// and then copies one to the other while rewriting tags
public void saveThePage() {

 char thisChar;
 String theTag;
 PrintStream  p = null;

 try {
   DataInputStream theHTML = new DataInputStream(theURL.openStream());
   p = makeOutputFile();

   while (true) {
     thisChar = (char) theHTML.readByte();
     if (thisChar == '<') {
       theTag = readTag(theHTML);
       theTag = convertTag(theTag);
       p.print(theTag);
     }
     else {
       p.print(thisChar);
     }
   } // end while
 }  // end try
 catch (EOFException e) {  // This page is done
 }
 catch (Exception e) {
    System.err.println(e);
 }
 finally {
   p.close();
 }

}  // end SaveThePage


// We need open a file on the local file system
// with the same name as the remote file;
// then chain a PrintStream to the file
public PrintStream makeOutputFile() throws IOException {

  FileOutputStream fout;

  String theFile = theURL.getFile();

  // the getFile method returns the filename prefixed with a slash,
  // e.g. /index.html instead of index.html. That slash needs to be removed.
  theFile = theFile.substring(1);
  System.err.println("\n\n\n" + theFile + "\n\n\n");
  if (theFile.equals("")) theFile = "index.html";

  // At this point you should check to see whether
  // the file already exists and, if it does,
  // ask the user if they wish to overwrite it

  fout = new FileOutputStream(theFile);

  return new PrintStream(fout);

}

// The readTag method is called when a < is encountered
// in the input stream.  This method is responsible
// for reading the remainder of the tag.
// Note that when this method has been called the <
// has been read from the input stream but has not yet been sent
// to the output stream.
// This method has trouble (as do most web browsers)
// if it encounters a raw < sign in the Stream. Technically
// raw < signs should be encoded as &lt; in the original HTML.
public static String readTag(DataInputStream is) {

  StringBuffer theTag = new StringBuffer("<");
```

```
  char theChar = '<';

  try {
     while (theChar != '>') {
       theChar = (char) is.readByte();
       theTag.append(theChar);
     } // end while
   }  // end try
   catch (EOFException e) {
     // Done with the Stream
   }
   catch (Exception e) {
      System.err.println(e);
   }

   return theTag.toString();

}

// The convertTag method takes a complete tag as
// a String and, if it's a relative link, converts it
// to an absolute link.  The converted tag is returned.
public String convertTag(String tag) {

  // temporary position variables
  int p1, p2, p3, p4;

  try {
    // HTML tags are cases insensitive so converting
    // it to upper case makes the problem slightly easier
    String s1 = tag.toUpperCase();
    // Find the beginning and the end of the URL
    //
    if (s1.startsWith("<A HREF")) {
      p1 = s1.indexOf("HREF");
    }
    else if (s1.startsWith("<IMG ")) {
      p1 = s1.indexOf("SRC");
    }
    else if (s1.startsWith("<APPLET ")) {
      p1 = s1.indexOf("CODEBASE");
    }
    else { // this is not a link based tag
      return tag;
    }
    // find the =
    p2 = s1.indexOf ("=", p1);
    if (p2 == -1) return tag;
    // Ideally the = sign is immediately followed by
    // a " mark followed by the URL which is closed by a ".
    // However since a lot of HTML is non-conforming we
    // need to be a little sneakier. In this case we read
    // characters in the URL until an character which is not
    // whitespace is encountered.
    p3 = p2+1;
    while (Character.isSpace(s1.charAt(p3))) {
      p3++;
    }
    if (s1.charAt(p3) == '"') p3++;

    // p3 now points to the beginning of the URL
    // The URL is read until a closing " or whitespace is seen
    p4 = p3+1;
    while (!Character.isSpace(s1.charAt(p4)) &&
     s1.charAt(p4) != '"') {
      p4++;
    }

    // The URL is the text between p3 and p4
    // URL's are in general NOT case insensitive so the URL
    // must be read from the original tag and not from s1
    // which was uppercased
    String link = tag.substring(p3, p4);

    // Is it a relative URL?  Relative URLs
    // don't contain colons.
    if (link.indexOf(":")  == -1) {
         // build an absolute URL from the relative URL
         URL newURL = new URL(theURL, link);
         // replace the old URL with the new URL
         tag = s1.substring(0,p3) + newURL + s1.substring(p4,s1.length());
      }  // end if
```

```
    }  // end try
    catch (StringIndexOutOfBoundsException e) {
      // Most of the time a StringIndexOutOfBoundsException here means
      // the tag was not standard conforming so
      // the algorithm for finding the URL crapped out.
      // If that's the case, the original tag is returned.
    }
    catch (Exception e) {
      System.err.println(e);
    }


    return tag;

  }

}
```

Appendix B, källkod

```
/*****************************************************************************
** Java-program för att identifiera länkar som saknar referenser i HTML-dokument. **
** Utvecklad med JDK1.0.2 samt JDK1.1.1.                                      **
**                                                                           **
** Användning:                                                               **
**   java Barbapappa <http://den.adress.som/skall/kontrolleras/<sida.html>>  **
**                                                                           **
** (c) Christer Lundberg, Högskolan Skövde                                   **
*****************************************************************************/

import java.net.*;
import java.io.*;
import java.util.*;

/*******************************************************
 * Huvudklass för Barbapappa                           *
 * Definierar theURL till en URL                       *
 * Hamta argument från kommando                        *
 * Öppna URL för läsning                               *
 * Skapar en root för den URL som skickades med vid start *
 * Skapar ett Barbapappa objekt från root              *
 * Anropar dra.Lankarna för ovanstående objekt.        *
 *******************************************************/

public class Barbapappa {
      URL theURL;
      Vector attScanna = new Vector(100, 50); // Adresser som skall scannas
      Vector scannade = new Vector(100, 50); // Adresser som är scannade
      Vector bra = new Vector(100, 100); // Adresser som är OK
      Vector daliga = new Vector(50, 10); // Adresser som inte var OK

   public static void main (String args[]) {
      for  (int i = 0; i < args.length; i++) {
        try {
           Properties prop = System.getProperties();
           prop.put("proxySet", "true");
           prop.put("proxyHost", "wwwproxy.his.se");
           prop.put("proxyPort", "80");

           URL root = new URL(args[0]);
           Barbapappa dl = new Barbapappa(root);
           dl.draLankarna();
           System.out.println("\nBra länkar:\n");
           while (!dl.bra.isEmpty()){
              Sidblock a = (Sidblock)dl.bra.firstElement();
              a.skriv();
              dl.bra.removeElement(a);
           }
           System.out.println("\nDåliga länkar:\n");
           while (!dl.daliga.isEmpty()){
              Sidblock a = (Sidblock)dl.daliga.firstElement();
              a.skriv();
              dl.daliga.removeElement(a);
           }
           System.out.println("\nLänkar att scanna:\n");
           while (!dl.attScanna.isEmpty()){
              String b = (String)dl.attScanna.firstElement();
              URL c = new URL(b);
              Barbapappa ndl = new Barbapappa(c);
              dl.attScanna.removeElement(b);
              ndl.draLankarna();
              System.out.println("\nBra länkar:\n");

              while (!ndl.bra.isEmpty()){
                Sidblock a = (Sidblock)ndl.bra.firstElement();
                a.skriv();
                ndl.bra.removeElement(a);
              }
              System.out.println("\nDåliga länkar:\n");
              while (!ndl.daliga.isEmpty()){
                Sidblock a = (Sidblock)ndl.daliga.firstElement();
                a.skriv();
                ndl.daliga.removeElement(a);
              }
              System.out.println("\nLänkar att scanna:\n");
              while (!ndl.attScanna.isEmpty()){
                String a = (String)ndl.attScanna.firstElement();
                System.out.println(a);
                ndl.attScanna.removeElement(a);
```

```
            }
         }


        } // try
        catch (MalformedURLException e) {
           System.err.println(args[0] + " Denna URL kan ej parsas");
           System.err.println(e);
           System.out.println("Sida: " + args[0]);
           System.out.println("Länk: " + args[0]);
           System.out.println("Response-code: 999");
              System.out.println("Response-message: " + e + "\n Kontrollera detta
manuellt! \nFörmodligen angav du en icke giltig startadress! \nAnvändning: java
Barbapappa <http://den.adress.som/skall/kontrolleras/<sida.html>> ");
        } // catch
      } // for
   } // main

/************************************************
 * Konstruktor för Barbapappa                   *
 * theURL tilldelas den URL som skickas med anropet *
 ************************************************/

   public Barbapappa(URL u) {
      theURL = u;
   } // Barbapappa konstruktor


/************************************************
 * Funktion för att dra länkar ur ett dokument  *
 * draLankarna öppnar en DataInputStream från URLen,  *
 * samt ÷ppnar en PrintStream f÷r utmatning till fil. *
 * thisChar = aktuell char                      *
 * theTag = aktuell tag                         *
 * link = den del av URLen som innehåller adress      *
 * Skapar DataInputStream för theURL            *
 * Läser byte från theHTML som kommer från theURL     *
 * om vi hittar ett "<" har vi en tag           *
 * läs tagen och tilldela theTag resultatet     *
 * konvertera tagen                             *
 ************************************************/

   public void draLankarna() {
      char thisChar;
      String theTag = null;
      String link = null;
      try {
        DataInputStream theHTML = new DataInputStream(theURL.openStream());
        while (true) {
           thisChar = (char) theHTML.readByte();
           if (thisChar == '<') {
              theTag = readTag(theHTML);
              theTag = convertTag(theTag);
           } // if
        } // while
      }  // try
      catch (EOFException e) {  // sidan är läst
      }   // nop
      catch (Exception e) {
        System.err.println(e);
        String sida, lank, kod, med;
        Sidblock s = new Sidblock();
        s.sida = "Sida: " + theURL;
        s.lank = "Länk: " + theTag;
        s.kod = "Response-code: 999";
         s.med = "Response-message: " + e + "\n Kontrollera detta manuellt! \nEn
länktag kan förmodligen inte läsas!";
        daliga.addElement(s);
      } // fel
      finally {
      } // final
   }  //  draLankarna



/*****************************************************************
 * Funktion för att läsa en tag                                  *
 * readTag anropas när vi stöter på ett "<" i den inlästa strängen. *
 * Det kan uppstå problem om ett "<" upptäcks i en textmassa,    *
 * dessa skall normalt skrivas "&lt;" i HTML-koden istället.     *
 * Skapa StringBuffer f÷r theTag                                 *
 * Definierar theChar till char och tilldelar den "<"            *
 * Läs hela tagen, dvs så länge vi inte hittar ett ">"           *
```

```
 * Tilldela theChar nästa byte                                          *
 * Lägg till till theTag, dvs utöka tagen med ett tecken                *
 * Returnera theTag, dvs tagen, som sträng                              *
 ************************************************************************/

   public static String readTag(DataInputStream is) {
      StringBuffer theTag = new StringBuffer("<");
      char theChar = '<';
      try {
        while (theChar != '>') {
           theChar = (char) is.readByte();
           theTag.append(theChar);
        } // while
      }  // try
      catch (EOFException e) { // klar med Stream
      } // catch EOF
      catch (Exception e) {
        System.err.println(e); // fel
        System.out.println("Sida: " + theTag);
        System.out.println("Länk: " + theTag);
        System.out.println("Response-code: 999");
        System.out.println("Response-message: " + e + "\n Kontrollera detta manuellt!
\nFörmodligen kan inte en tag läsas!");
      } // catch
      return theTag.toString();
   } // readTag

/************************************************************************
 * Funktion för att konvertera en tag convertTag tar en tag som         *
 *  sträng och omvandlar till en absolut adress om nödvändigt.          *
 *  Returnerar den omvandlade tagen.                                    *
 * position1 till position4 används som tillfälliga positioner i tagen  *
 * Då HTML tags ej bryr sig om små eller stora bokstäver                *
 * omvandlas denna till stora bokstäver.                                *
 * Skapa s1 som String samt tilldela den tagen stora bokstäver          *
 * Hitta början på URLen, dvs "<A HREF" och sätt position 1,            *
 * annars returnera tagen.                                              *
 * Hitta "=" och sätt som position 2, om inget "=" returnera tagen.     *
 * Normalt följs "=" omedelbart av ett " följt av URLen,                *
 * dock tillåter HTML att så inte är fallet. Därför läser vi tills      *
 * att vi hittar ett tecken som inte är blankt, medans tecken är blankt *
 * öka position 3. Om strängen är en MAILTO returnera tagen.            *
 * Position 3 pekar nu på URLens början, läs URL tills vi hittar ett "  *
 * eller ett blankt tecken och sätt position 4. Medans URLen ej är slut *
 * öka position 4. URLen finns nu mellan position 3 och 4.              *
 * Eftersom att URLer är känsliga för stora och små tecken              *
 * läser vi om URLen från den ursprungliga tagen.                       *
 * Definiera link som String och tilldela den URLen.                    *
 * Definiera nlink som String och tilldela den URLen.                   *
 * Kontrollera om det är en absolut eller relativ URL, relativa URLer   *
 * innehåller inte ":". Om en relativ URL gör vi en absolut URL,        *
 * newURL definieras som URL och tilldela den link. Tilldela tagen      *
 * den nya URLen, tilldela nlink den nya URLen (endast URLen).          *
 * Om tagen inte följde den standard vi läste från returneras den       *
 * ursprungliga tagen, slutligen returneras tagen.                      *
 ************************************************************************/

   public String convertTag(String tag) {
      int position1 = 0, position2 = 0, position3 = 0, position4 = 0;
      try {
        String s1 = tag.toUpperCase();
          if    (s1.startsWith("<A    HREF")    ||    s1.startsWith("<A    \nHREF")    ||
s1.startsWith("<A\nHREF")) {
           position1 = s1.indexOf("HREF");
        }
        else {
           return tag;
        } // if
        position2 = s1.indexOf ("=", position1);
        if (position2 == -1) return tag;
        position3 = position2+1;
        while (Character.isWhitespace(s1.charAt(position3))) {
           position3++;
        } // while
        if (s1.charAt(position3) == '"') position3++;
        if (s1.substring(position3,s1.length()).startsWith("MAILTO:")) return tag;
        position4 = position3+1;
        while (!Character.isWhitespace(s1.charAt(position4)) &&
              s1.charAt(position4) != '"') {
           position4++;
        } // while
        String link = tag.substring(position3, position4);
        String nlink = tag.substring(position3, position4);
```

```
         if (link.indexOf(":")  == -1) {
            URL newURL = new URL(theURL, link);
                           tag     =     s1.substring(0,position3)     +     newURL     +
s1.substring(position4,s1.length());
            nlink = newURL.toString();
         }  // if
         respons(nlink);
       }  // try
       catch (StringIndexOutOfBoundsException e) {
         String sida, lank, kod, med;
         Sidblock s = new Sidblock();
         s.sida = "Sida: " + theURL;
         s.lank = "Länk: " + tag;
         s.kod = "Response-code: 999";
          s.med  =  "Response-message:  "  +  e  +  "\nKontrollera  detta  manuellt!
\nFörmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!";
         daliga.addElement(s);

       } // catch OutOfBounds
       catch (Exception e) {
         System.err.println(e);  // fel
         String sida, lank, kod, med;
         Sidblock s = new Sidblock();
         s.sida = "Sida: " + theURL;
         s.lank = "Länk: " + tag;
         s.kod = "Response-code: 999";
          s.med  =  "Response-message:  "  +  e  +  "\nKontrollera  detta  manuellt!
\nFörmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!";
         daliga.addElement(s);

       } // catch
       return tag;
    } // convertTag

/**************************************************************************
 * Funktion för att kontrollera status på länkarna                        *
 *                                                                        *
 **************************************************************************/

    public String respons(String adr) {
       try {
         URL p = new URL(adr);
         HttpURLConnection pp = (HttpURLConnection) p.openConnection();
         String sida, lank, kod, med;
         Sidblock s = new Sidblock();
         s.sida = "Sida: " + theURL;
         s.lank = "Länk: " + adr;
         s.kod = "Response-code: " + pp.getResponseCode();
         s.med = "Response-message: " + pp.getResponseMessage();

         if (pp.getResponseCode() > 300) {
            daliga.addElement(s);
         }
         else {
            bra.addElement(s);
         }
         if (adr.startsWith(theURL.toString()) && !scannade.contains(adr)){
            System.out.println("LÄGG TILL " + adr);
            attScanna.addElement(adr);
         }
         adr = pp.getResponseMessage();
         if (adr.startsWith(theURL.toString())){
            System.out.println("SCANNAD " + adr);
            scannade.addElement(adr);
         }

       }  // try
       catch (MalformedURLException e) {
         System.err.println(e + "\"" + adr + "\"" + " \n The structure of the URL seems
strange! Please check!");
         String sida, lank, kod, med;
         Sidblock s = new Sidblock();
         s.sida = "Sida: " + theURL;
         s.lank = "Länk " + adr;
         s.kod = "Response-code: 999";
          s.med  =  "Response-message:  "  +  e  +  "\nKontrollera  detta  manuellt!
\nFörmodligen är det något fel på URLens struktur!";
         daliga.addElement(s);

       } // catch MUE
       catch (IOException e) {
```

```
         System.err.println("\"" + e + "\"" + "\n Something is wrong with this URL!
Please check that " + adr + " is a valid URL!");
         String sida, lank, kod, med;
         Sidblock s = new Sidblock();
         s.sida = "Sida: " + theURL;
         s.lank = "Länk " + adr;
         s.kod = "Response-code: 999";
          s.med  =  "Response-message:  "  +  e  +  "\nKontrollera  detta  manuellt!
\nFörmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!";
         daliga.addElement(s);

      } // catch IOE
//       System.out.println();
      return adr;
   } // respons

      class Sidblock {
           String sida = null;
           String lank = null;
           String kod = null;
           String med = null;

        public void skriv(){
           try {
              System.out.println(sida + "\n" + lank + "\n" +
                                 kod + "\n" + med + "\n");
           } catch(Exception e) {
           } // catch Exception
        }
      }

} // Barbapappa
```

Appendix C, testkörningar

Fullständiga testkörningar återfinns I separat appendix.

Sida.p

```
program sida(input,output);

var r,s,o,p : real;
   i,j      : integer;
begin
   for i := 1 to 10 do
   begin
      s := seed(wallclock);
      r := random(wallclock)*675;
      o := random(wallclock)*60+1;
      while r < 19.0 do begin
        r := random(wallclock)*675;
      end;
      writeln('Sida: ', r:1:0);
      writeln('Ord: ', o:1:0);
      for j := 1 to trunc(random(wallclock)*100000000) do begin end;
   end;
end.
```

Uppslag.txt

```
Sida: 312
Ord: 5
=>landa
http://www.algonet.se/~hosarvid/gen/landa.htm
http://www.algonet.se/~hosarvid/gen/
Sida: 350

Ord: 10
=>milliard
http://www.medfak.lu.se/MVA/

Sida: 40
Ord: 44
=>arrivist -> arriärgarde -> arrogans
http://www.helsingborg.se/newspaper/06/HD18.HTM
http://www.helsingborg.se/newspaper/06/ (TOM)

Sida: 406
Ord: 18
=>palma
http://www.svkyrkan.se/skut/verk95/mallorca.htm
http://www.svkyrkan.se/skut/verk95/

Sida: 443
Ord: 23
=>puh
http://strindberg.ling.uu.se/~sarag/puh.html
http://strindberg.ling.uu.se/~sarag/

Sida: 481
Ord: 28
=>sabbat
http://www.l.kth.se/www/club/lsqling/texter1.htm
http://www.l.kth.se/www/club/lsqling/

Sida: 171
Ord: 1
=>förlösa
http://www.sundsvall.se/vartweb/nov96/07.html
http://www.sundsvall.se/vartweb/nov96/ (HTTP/1.0 403 Access Forbidden)

Sida: 209
Ord: 6
=>helikopter
http://www.utsidan.se/outside/1996/0237.html
http://www.utsidan.se/outside/1996/

Sida: 593
Ord: 42
=>tomat
http://www.arosnet.se/werbeka/krog/tomsards.htm
```

```
http://www.arosnet.se/werbeka/krog/ (HTTP/1.0 403 Access Forbidden)

Sida: 631
Ord: 47
=>vaccin
http://www.ktv.fi/kovi1596.htm
http://www.ktv.fi/
```

http://strindberg.ling.uu.se/~sarag/puh.html

```
<HTML>
<HEAD>
<TITLE>Puh
</TITLE>
</HEAD>
<BODY BACKGROUND="Bilder/page09.gif" text="#990000">
<Font size=+4>
<TD VALIGN=top ALIGN=right>
<center>Puhsida på gång</center>
</Font>
<A HREF="Bilder/pigletbi.gif">
<IMG SRC="Bilder/pigletsm.gif"></A>
<br>
<br>
<br>
<Font size=+3>
Läs The House at Pooh Corner</Font>
<A HREF="Bilder/pooh3.gif">
<IMG SRC="Bilder/pooh3sm.gif"></A>
<br>
<A HREF="http://www.harju.ee/pooh/part2/pooh2_0.html">Inledning
</A> med länkar till de olika kapitlen längst ner på sidan.

<br>
<br>
<Font size=+3>
Läs Winnie-the-Pooh</Font>
<br>
<A HREF="http://www.harju.ee:80/pooh/part1/pooh1_0.html">
Inledning</A> Vidarelänkar till bokens olika kapitel längst ner.
<br>

<br>
<br>
<Font size=+4>
Länkar</Font>
<br>
<br>
Förlåt. De länkar jag hade här tidigare ledde inte längre någonstans, så jag tog bort
dem. Jag
ska leta fram några nya och lägga dit, det kan ta ett par dar.
<br>
<br>
<A HREF="http://infoweb.magi.com/~datakes/winniepo.html>Länk till länkar</A>
<br>
<br>
<br>
<br>
<br>
<Font size=+3>
<A HREF="http://strindberg.ling.uu.se/~sarag/">Hem till Sara</A>
</Font>


</BODY>
</HTML>
```

Resultat puh.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running                  /usr/local/apps/JDK1.1.1/bin/java                  Barbapappa
http://strindberg.ling.uu.se/~sarag/puh.html

Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/pigletbi.gif
Response-code: 200
Response-message: Document follows
```

```
Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/pooh3.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://strindberg.ling.uu.se/~sarag/
Response-code: 200
Response-message: Document follows


Dåliga länkar:

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://www.harju.ee/pooh/part2/pooh2_0.html
Response-code: 404
Response-message: Not found

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: http://www.harju.ee:80/pooh/part1/pooh1_0.html
Response-code: 404
Response-message: Not found

Sida: http://strindberg.ling.uu.se/~sarag/puh.html
Länk: <A HREF="http://infoweb.magi.com/~datakes/winniepo.html>
Response-code: 999
Response-message:  java.lang.StringIndexOutOfBoundsException:  String  index  out  of
range: 56
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!


Länkar att scanna:
```

http://strindberg.ling.uu.se/~sarag/

```
<HTML>
<HEAD>
<TITLE>Hos Sara
</TITLE>
</HEAD>


<BODY  BACKGROUND="Bilder/beerback.gif"  text="#FFFFFF"  link="#FFFF66"  alink="#FFFF33"
vlink="#FF9900">

<IMG SRC="Bilder/kisi_snyst.gif">
<center>
<IMG SRC="Bilder/star.gif">
<Font size=+5>TRISTESS HOS SARA</Font>
<IMG SRC="Bilder/star.gif">
</center>
<br>
<br>
<Font size=+3>Hej Du.</Font>
<br>
<br>
<Font      size=+3>Du      är      <B>inte</B>      besökare      nummer      </Font><IMG
SRC="http://strindberg.ling.uu.se/
cgi-bin/count?width=4&link=~sarag/welcome.html"><Font size=+3> </Font>
<BR>
<BR>Glad vår!<br>
<A HREF="Bilder/Gonzo.GIF">Hoppas att du mår bra!</A>

<br>
<br>
<IMG SRC="Bilder/yl_star.gif">
<Font size=+4><A HREF="jag.html">Om mig</Font></A>
<br>
<br>
<IMG SRC="Bilder/yl_star.gif">
<Font size=+4>
<A HREF="hueremere.html">Hueremere</A></Font>
<br>
<br>
<IMG SRC="Bilder/yl_star.gif">
Ta en titt på min förvirrade
<Font size=+3>
<A HREF="Blandat">Allt Möjligt</A> sida
```

```
</Font>
<br>
<br>
<IMG SRC="Bilder/yl_star.gif"><Font size=+4>
Rimma             med             mitt             och             Marias             <A
HREF="http://strindberg.ling.uu.se/~mariae/Rimlex/rimlex.html">rimlexikon</A>
</Font>
<br>
<br>
<A HREF="http://strindberg.ling.uu.se/~mariae/Asterix/asterixsa.html">
<IMG border=0 SRC="Bilder/asterix1.gif">
</A>Ännu                                    ett                                    <A
HREF="http://strindberg.ling.uu.se/~mariae/Asterix/asterixsa.html">lexikon</A> som jag
och Maria gjort.
<br><br>
<IMG SRC="Bilder/yl_star.gif">
Var   med   om   <A   HREF="http://www.ferndale.com/universal/msc/index.html">Interaktiv
såpopera!</A> Eller låt bli.
<br>
<br>


<IMG SRC="Bilder/yl_star.gif">
Gör ett besök hos
<A HREF="http://strindberg.ling.uu.se/~mariae">Maria!</A>
  eller hos
<A HREF="http://strindberg.ling.uu.se/~bodil">Bodil</A>


<br>
<IMG SRC="Bilder/yl_star.gif">
<Font size+4>
Hitta rätt med
<A HREF="http://altavista.digital.com/">Altavista</A></Font>
<br>
<br>
<IMG SRC="Bilder/yl_star.gif">
Fyll i min
<Font size=+3>
<A HREF="sarag">gästbok</A></Font>
 eller kolla vilka som <A HREF="http://strindberg.ling.uu.se/guestbook/sarag">
fyllt i den tidigare</A>
<br>
<br>
<IMG SRC="Bilder/yl_star.gif">
Till
<A HREF="http://strindberg.ling.uu.se/">Språkteknologerna</A>
<br>
<br>
<Font size+5>
Nu ska jag gå och fika</Font><IMG SRC="Bilder/coffee9bkg.jpeg"><br>
<br>
<BR>
<br>
<center>
<FONT SIZE=+2>
<IMG SRC="Balls/envelope.gif">
<A HREF="mailto:sarag@strindberg.ling.uu.se">Skicka mig gärna ett brev</A>
</FONT>
</center>
</BODY>
</HTML>
```

Resultat puhH.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running                  /usr/local/apps/JDK1.1.1/bin/java                  Barbapappa
http://strindberg.ling.uu.se/~sarag/
LÄGG TILL http://strindberg.ling.uu.se/~sarag/Bilder/Gonzo.GIF
LÄGG TILL http://strindberg.ling.uu.se/~sarag/jag.html
LÄGG TILL http://strindberg.ling.uu.se/~sarag/hueremere.html
LÄGG TILL http://strindberg.ling.uu.se/~sarag/Blandat
LÄGG TILL http://strindberg.ling.uu.se/~sarag/sarag

Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/Gonzo.GIF
Response-code: 200
Response-message: Document follows
```

Appendix C, testkörningar

```
Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~sarag/jag.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~sarag/hueremere.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~sarag/Blandat
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~mariae/Rimlex/rimlex.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~mariae/Asterix/asterixsa.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~mariae/Asterix/asterixsa.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~mariae
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~bodil
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://altavista.digital.com/
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/~sarag/sarag
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/guestbook/sarag
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://strindberg.ling.uu.se/
Response-code: 200
Response-message: Document follows


Dåliga länkar:

Sida: http://strindberg.ling.uu.se/~sarag/
Länk: http://www.ferndale.com/universal/msc/index.html
Response-code: 500
Response-message: Error from proxy


Länkar att scanna:


Bra länkar:


Dåliga länkar:


Länkar att scanna:


Bra länkar:
```

Appendix C, testkörningar

```
Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://www.uu.se/
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://strindberg.ling.uu.se/~fredriko/ling.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/slottfi.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/mariafi.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/gatafi.gif
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://strindberg.ling.uu.se/~sarag/
Response-code: 200
Response-message: Document follows


Dåliga länkar:

Sida: http://strindberg.ling.uu.se/~sarag/jag.html
Länk: http://sunsite.kth.se/DDS/bib/ostergotland_lan/finspa/fina.htm
Response-code: 404
Response-message: Not found


Länkar att scanna:


Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/Huere/Sese.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/Huere/Lidde.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/Huere/huere.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/Huere/hu.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/Huere/utlat.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag/hubild/allfrien.jpg
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://www.ljusdal.se/~cnil/Bomb/index.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/hueremere.html
Länk: http://strindberg.ling.uu.se/~sarag
Response-code: 200
Response-message: Document follows
```

Appendix C, testkörningar

```
Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/ide.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.webcom.com/eha/idiom.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.inch.com/~jeffz/icon.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.ljusdal.se/~cnil/Bomb/index.htm
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.csd.uu.se/~mrytther/NN
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://seds.lpl.arizona.edu/nineplanets/nineplanets/nineplanets.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://seds.lpl.arizona.edu/nineplanets/nineplanets/nineplanets.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.phoenix.net/~jacobson/rgb.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/sarag
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/puh.html
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.cybercomm.nl/~stp/h_index.htm
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.ncl.ac.uk/~n1zai/pingu.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/Bilder/sesamest3.jpg
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.iuma.com/IUMA-2.0/pages/home_page/homepage.html
Response-code: 200
Response-message: OK

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/fiskar.html
Response-code: 200
```

```
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://strindberg.ling.uu.se/~sarag/
Response-code: 200
Response-message: Document follows


Dåliga länkar:

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.hyperreality.com/a/
Response-code: 500
Response-message: Error from proxy

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.interlog.com/~maikin/aikin.html
Response-code: 500
Response-message: Error from proxy

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: <A HREF="ftp://ftp.luth.se/pub/sounds/songs/">
Response-code: 999
Response-message:                                          java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://strindberg.ling.uu.se/~sarag/Blandat
Länk: http://www.cyberiacafe.net/cyberia/guide/ccafe.htm#working_europe
Response-code: 404
Response-message: Object Not Found


Länkar att scanna:


Bra länkar:

Sida: http://strindberg.ling.uu.se/~sarag/sarag
Länk: http://www.let.rug.nl/~welling/
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/sarag
Länk: http://www.let.rug.nl/ahc/
Response-code: 200
Response-message: Document follows

Sida: http://strindberg.ling.uu.se/~sarag/sarag
Länk: http://www.let.rug.nl/~welling/usa/revolution.html
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:
```

http://www.sundsvall.se/vartweb/nov96/07.html

```
<HTML>
<HEAD>
  <META NAME="GENERATOR" CONTENT="Adobe PageMill 2.0 Mac">
  <TITLE>V&aring;rt Sundsvall - Hon vill f&ouml;rl&ouml;sa kulturen i stan</TITLE>
</HEAD>
<BODY BACKGROUND="../GIFs/bakgrunda.gif">

<P><A NAME="anchor289183"></A></P>

<P><IMG SRC="../head.gif" WIDTH="522" HEIGHT="87" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3" ALT="Logotype"></P>

<P><TABLE WIDTH="521" BORDER="0" CELLSPACING="0" CELLPADDING="0" HEIGHT=
"41">
<TR>
<TD    WIDTH="105"    VALIGN="TOP"><TABLE    WIDTH="98"    BORDER="0"    CELLSPACING="0"
CELLPADDING="0">
<TR>
<TD WIDTH="100%" VALIGN="TOP"><B><FONT SIZE=+2>Nummer 1<BR>
```

```
</FONT>November 1996</B></TD></TR>
</TABLE>
<TABLE WIDTH="102" BORDER="0" CELLSPACING="0" CELLPADDING="0" HEIGHT="373">
<TR>
<TD WIDTH="100%" VALIGN="TOP"><P><HR><B>Redakt&ouml;r:<BR>
</B>Bo Sj&ouml;gren</P>

<P><B>Telefon </B>(vxl):<BR>
060-19 10 00</P>

<P><B>Fax:</B><BR>
060-12 81 91</P>

<P><B>Postadress:<BR>
</B><A HREF="mailto:sundsvalls.kommun@sundsvall.se">Sundsvalls<BR>
kommun</A><BR>
851 85 Sundsvall</P>

<P><B>Bes&ouml;ksadress:</B><BR>
Norrmalmsgatan 4</P>

<P><B>Ansv. Utgivare:<BR>
</B><A HREF="mailto:gunnar.eklow@sundsvall.se">Gunnar Ekl&ouml;w</A></P>

<P><HR></P>

<P ALIGN=CENTER><MAP NAME="home">
    <AREA SHAPE="rect" COORDS="2,1,99,34" HREF="http://www.sundsvall.se/">
</MAP><IMG SRC="../GIFs/home.gif" WIDTH="101" HEIGHT="36" ALIGN="BOTTOM"
NATURALSIZEFLAG="3" ISMAP ALT="hem" BORDER="0" USEMAP="#home"><BR>
<FONT SIZE=-1><A HREF="http://www.sundsvall.se/">Kommunens hemsida</A></FONT></P>

<P ALIGN=CENTER><HR> <TABLE WIDTH="100%" BORDER="3" CELLSPACING="0" CELLPADDING=
"2">
<TR>
<TD WIDTH="100%" BGCOLOR="#f58a00" ALIGN="CENTER"> <FONT SIZE=-1>Sidorna &auml;r
designade f&ouml;r Netscape 3.0 Gold<BR>
Ladda ner version (alt+klicka)<BR>
</FONT><IMG SRC="../GIFs/net.gif" WIDTH="88" HEIGHT="31" ALIGN="BOTTOM"
NATURALSIZEFLAG="3" ALT="Netscapelogga"><BR>
<FONT                                                                    SIZE=-2><A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">Mac</A>
|                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">Win95</A>
|                                                                            <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">Win
3.1</A></FONT></TD></TR>
</TABLE>
</TD></TR>
</TABLE>
</TD>
<TD WIDTH="353" VALIGN="TOP" ALIGN="RIGHT"><HR WIDTH="400"><BR>
<TABLE WIDTH="398" BORDER="0" CELLSPACING="0" CELLPADDING="7">
<TR>
<TD    WIDTH="82%"    VALIGN="TOP"    HEIGHT="126"><TABLE    WIDTH="100%"    BORDER="0"
CELLSPACING="0" CELLPADDING="0">
<TR>
<TD WIDTH="100%" VALIGN="TOP"><B><FONT SIZE=+4>Hon vill f&ouml;rl&ouml;sa <BR>
kulturen i stan</FONT></B></TD></TR>
</TABLE>
 <TABLE WIDTH="96%" BORDER="0" CELLSPACING="0" CELLPADDING="4" HEIGHT="110">
<TR>
<TD  WIDTH="26%"  VALIGN="BOTTOM"><IMG  SRC="../GIFs/Suzz.gif"  WIDTH="113"  HEIGHT="147"
ALIGN="BOTTOM" NATURALSIZEFLAG=
"3" ALT="Suzanne Askelöf"><BR>
<B><FONT SIZE=-1>Foto: Marcus Engstr&ouml;m</FONT></B></TD>
<TD   WIDTH="56%"   VALIGN="BOTTOM"   ALIGN="RIGHT"><P   ALIGN=LEFT><TABLE   WIDTH="100%"
BORDER="1" CELLSPACING="0" CELLPADDING=
"4">
<TR>
<TD WIDTH="100%" BGCOLOR="#e6e6e6">ARKIV<BR>
<B>Suzanne Askel&ouml;f</B></TD></TR>
<TR>
<TD WIDTH="100%" BGCOLOR="#e6e6e6"><P><B>&Aring;lder: </B>52 &aring;r</P>

<P><B>Familj: </B>Gift, tre vuxna s&ouml;ner</P>

<P><B>Arbete: </B>Tj&auml;nsteman med ansvar f&ouml;r kultur, fritids- och
flyktingfr&aring;gor i Sundsvalls kommun</P>
```

Appendix C, testkörningar

```
<P><B>Intressen: </B>L&auml;ser mycket, seglar. &quot;Jag skulle vilja f&aring;
Tall Ships Race till Sundsvall&quot;</TD></TR>
</TABLE>
</TD></TR>
</TABLE>
</TD>
<TD    WIDTH="17%"    BGCOLOR="#ffff00"    VALIGN="TOP"><P><B><FONT    COLOR="#BF0000"
SIZE=+1>G&aring; till</FONT></B></P>

<P><A HREF="../vart.html">Inneh&aring;ll</A></P>

<P><A HREF="08.html">N&auml;sta sida</A></P>

<P><A HREF="06b.html">F&ouml;reg&aring;ende sida</A></TD></TR>
<TR>
<TD VALIGN="TOP" WIDTH="82%"><P><B>Att g&aring; mot nya stord&aring;d anses vara en
f&ouml;rvaltnings-
chefs uppgift, s&auml;ger Suzanne Askel&ouml;f. Den 15 augusti tilltr&auml;dde
hon jobbet som Sundsvalls nya kultur- och fritidschef. Humanistisk bildning
&auml;r en viktig del av hennes kultursyn och hon vill skapa ett intensivt
och brusande kulturliv i Sundsvall.<BR>
</B><IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3"><B>- Likgiltig verksamhet &auml;r inget att l&auml;gga samh&auml;llspengar
p&aring;, s&auml;ger hon.</B></P>

<P>Hennes bakgrund &auml;r n&auml;stan f&ouml;r bra. Akademisk examen med
en kombination av litteratur, historia och nordiska spr&aring;k. Hon har
arbetat som kultursekreterare, studiekonsulent och p&aring; bibliotek, har
bred erfarenhet av arbete med internationella fr&aring;gor centralt p&aring;
Kommunf&ouml;rbundet och &auml;r dessutom sportintresserad. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Men nu vill Suzanne Askel&ouml;f tillbaka till verkligheten. Luften
blir till slut lite tunn d&auml;r uppe p&aring; toppen, s&auml;ger hon.
N&auml;r hon sedan p&aring;st&aring;r att hennes yrkesstolthet best&aring;r
i att utveckla verksamheter trots knappa resurser, torde ansvariga politiker
i Sundsvall jubla &ouml;ver sitt kap. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Hon beskriver sig som en kr&auml;vande chef som vill att m&auml;nniskor
ska ha roligt p&aring; jobbet. Och &auml;r arbetsuppgiften lite st&ouml;rre
&auml;n man egentligen f&ouml;rm&aring;r blir resultatet n&auml;stan alltid
intressant.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">- De som har arbetat med mig genom &aring;ren brukar lite fl&auml;mtande
s&auml;ga att det b&ouml;rjade i h&ouml;g hastighet, sen blev det bara v&auml;rre,
s&auml;ger hon med ett skratt. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Sj&auml;lv har hon &auml;nnu en tid av relativ ansvarsfrihet. Nu g&auml;ller
det att l&auml;ra k&auml;nna nya staden, inv&aring;narna och medarbetarna.
Hon saknar inte id&eacute;er och &aring;sikter &auml;ven om hon vill vara
lite &aring;terh&aring;llsam.</P>

<P><B>Kultur och fritid samtidig</B>t<BR>
Kultur och fritid har klara ber&ouml;ringspunkter, h&auml;vdar hon. I verksamhet
f&ouml;r exempelvis barn och ungdom k&auml;nns det mycket egendomligt att
spalta upp omr&aring;dena. En annan kombination kan vara kultur och utbildning.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">- I ekonomiskt tr&auml;ngda l&auml;gen m&aring;ste alla kommuner hitta
nya s&auml;tt att arbeta. Det finns ingen facit, bara olika s&auml;tt att
se vad som &auml;r f&ouml;rnuftigt, s&auml;ger Suzanne Askel&ouml;f.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">En kulturbyr&aring;krats uppgift &auml;r att se till att kulturlivet
p&aring; orten blir intensivt och sp&auml;nnande.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">&Auml;r man en intressant person kan man vara delaktig i det kulturella
livet. Alla kulturbyr&aring;krater &auml;r inte intressanta men kan i geng&auml;ld
vara skickliga p&aring; att f&aring; saker att h&auml;nda. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">- Det betyder inte att jag kommer att rycka och dra i organisationen,
s&auml;ger hon best&auml;mt.</P>

<P><B><I>Vad ska man d&aring; sk&auml;ra i n&auml;r pengarna tryter?</I></B><BR>
- I det som &auml;r on&ouml;digt, s&auml;ger hon utan att n&auml;rmare precisera.
Min yrkesstolthet ligger inte i att bara ha en massa bra id&eacute;er utan
att fullf&ouml;lja ett uppdrag s&aring; bra som m&ouml;jligt inom givna
ramar. Och det &auml;r inte alltid en nackdel med knappa resurser. Inte
ens basverksamheter kan i dagsl&auml;get bara rulla p&aring; och syssla
med finslipning av sina rutiner. Det &auml;r den krassa verkligheten.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Hon s&auml;ger sig veta vad hennes f&ouml;retr&auml;dare g&aring;tt
igenom n&auml;r hon fick det n&aring;got otacksamma uppdraget att sl&aring;
samman tre f&ouml;rvaltningar till en. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">- Jag har ocks&aring; gjort sv&aring;ra saker och sedan g&aring;tt.
```

Appendix C, testkörningar

```
Jag &auml;r nog en f&ouml;r&auml;ndrare, mer &auml;n en f&ouml;rvaltare,
s&auml;ger Suzanne Askel&ouml;f.</P>

<P><B>Kunniga men inte bildade</B><BR>
Suzanne Askel&ouml;f l&auml;ser f&ouml;r att leva och &ouml;verleva. Hon
&auml;r all&auml;tare men med smak som f&ouml;r&auml;ndrats n&aring;got
med &aring;ren. F&ouml;rutom sk&ouml;nlitteratur blir det allt mer biografier
och faktab&ouml;cker.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Hon s&auml;ger att vi borde l&auml;gga st&ouml;rre vikt vid humanistisk
bildning, och l&auml;ra oss v&aring;rt kulturarv. Svenskarna &auml;r ett
av v&auml;rldens mest massmediekonsumerande folk, vi &auml;r kunniga men
vi inte bildade, s&auml;ger hon. En akademisk examen kan vara en v&auml;g
att skaffa sig en grund, men m&aring;nga akademiker &auml;r inte s&auml;rskilt
bildade.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Inom arbetarr&ouml;relsen finns en stolt och intressant bildningstradition.
Den ska man v&auml;rna om, anser Askel&ouml;f. M&auml;nniskan v&auml;xer
n&auml;r hon g&ouml;r kunskapen till sin egen. Om svensken s&auml;ger hon
att han &auml;r f&ouml;r rakt p&aring; sak, utomlands anses det faktiskt
lite buffligt. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">- Det skulle inte skada om vi blir lite trevligare, lite mer spirituella,
s&auml;ger hon med aningen av glimt i &ouml;gat.</P>

<P><B><I>Hur ska du d&aring; konkretisera dina id&eacute;er?</I></B><BR>
- Det f&aring;r n&auml;stan visa sig, s&auml;ger Suzanne Askel&ouml;f. Det
p&aring;g&aring;r en verksamhet som jag ska ta pulsen p&aring;, jag vill
ha bredd s&aring;v&auml;l som djup. Jag skulle vilja fungera som en barnmorska
f&ouml;r f&ouml;reningsm&auml;nniskor, kulturarbetare, intresserade privatpersoner,
l&auml;rare, journalister, bibliotekarier.<BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Alla f&ouml;rvaltningschefer idag har dessutom ansvar f&ouml;r att deras
fr&aring;gor har en internationell sida. Hon rekryterades p&aring; internationella
meriter, hur de erfarenheterna ska anv&auml;ndas p&aring; lokalplanet &auml;r
inte helt klart. Klart &auml;r i alla fall att hon vill placera Sundsvall
p&aring; Europakartan. <BR>
<IMG SRC="../GIFs/indra.gif" WIDTH="7" HEIGHT="7" ALIGN="BOTTOM" NATURALSIZEFLAG=
"3">Och om Sundsvall &auml;r sn&auml;llt mot henne stannar hon ett tag.
Ett och annat stord&aring;d f&ouml;rv&auml;ntar vi oss nog, vare sig vi
f&ouml;redrar idrott eller kultur eller &auml;r engagerade i flyktingfr&aring;gor.
</P>

<P ALIGN=RIGHT><B>Ann Lundberg</B></TD>
<TD  WIDTH="17%"  BGCOLOR="#ffff00"  VALIGN="BOTTOM"><TABLE  WIDTH="100%"  BORDER="0"
CELLSPACING="0" CELLPADDING="0">
<TR>
<TD   WIDTH="100%"   VALIGN="BOTTOM"><P><B><FONT   COLOR="#BF0000"   SIZE=+1>G&aring;
till</FONT></B></P>

<P><A HREF="../vart.html">Inneh&aring;ll</A></P>

<P><A HREF="08.html">N&auml;sta sida</A></P>

<P><A HREF="06b.html">F&ouml;reg&aring;ende sida</A></P>

<P><A HREF="#anchor289183">Toppen <BR>
p&aring; sidan</A></TD></TR>
</TABLE>
</TD></TR>
</TABLE>
<BR>
<TABLE WIDTH="392" BORDER="0" CELLSPACING="0" CELLPADDING="0">
<TR>
<TD WIDTH="100%"><TABLE WIDTH="98%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
<TR>
<TD  WIDTH="100%"><HR  NOSHADE  WIDTH="102%"><FONT  SIZE=-1>Inneh&aring;llet  p&aring;
V&aring;rt
Sundsvalls sidor &auml;r upphovsr&auml;ttsligt skyddat&copy; och tillh&ouml;r
EkonoMedia     Aff&auml;rspress     AB.     F&ouml;r     synpunkter     kontakta     <A
HREF="mailto:vartsundsvall%3Cwebmaster@sundsvall.se%3E">webmaster</A>.
Sidorna &auml;r designade f&ouml;r att visas med Java-st&ouml;dd Netscape
3.0      eller      senare.      H&auml;mta      version:      (alt+klicka)      <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">Mac
OS</A>                                    |                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">Windows
95</A>                                    |                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">Windows
3.1 </A></FONT></TD></TR>
</TABLE>
```

```
</TD></TR>
</TABLE>
</TD></TR>
</TABLE>
</BODY>
</HTML>


<APPLET code="WRQPMCounter.class"
codebase="/WRQPMCtrl" width=2
       height=0
archive="WRQJavaPM.zip">
<param name="cabbase" value="WRQJavaPM.cab">
<param name="InitCount" value="0">
<param name="CookieIDhigh" value="0">
<param name="CookieIDlow" value="5259">
<param name="Velocity" value="0">
<param name="szURL" value="http://194.71.24.11:80/vartweb/nov96/07.html">
<param name="ActiveUsers" value="0">
<param name="ClientPullTime" value="30000">
<param name="Digits" value="1">
<param name="Order" value="1">
<param name="Form" value="Hidden">
</APPLET>
```

Resultat forlosa.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running                   /usr/local/apps/JDK1.1.1/bin/java                    Barbapappa
http://www.sundsvall.se/vartweb/nov96/07.html
LÄGG TILL http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183

Bra länkar:

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/vart.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/nov96/08.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/nov96/06b.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/vart.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/nov96/08.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/nov96/06b.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Response-code: 200
Response-message: OK


Dåliga länkar:

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                          <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">
Response-code: 999
```

```
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html
Länk:                                                                              <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!


Länkar att scanna:

LÄGG TILL http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183

Bra länkar:

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/vart.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/nov96/08.html
Response-code: 200
Response-message: OK
```

Appendix C, testkörningar

```
Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/nov96/06b.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/vart.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/nov96/08.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/nov96/06b.html
Response-code: 200
Response-message: OK

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Response-code: 200
Response-message: OK


Dåliga länkar:

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/mac/netsca
pe3.0Gold.hqx">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!

Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                    <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g3
230.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!
```

```
Sida: http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
Länk:                                                                                <A
HREF="ftp://ftp.luth.se/pub/infosystems/www/netscape/pub/navigator/gold/3.0/windows/g1
630.exe">
Response-code: 999
Response-message:                                        java.lang.ClassCastException:
sun.net.www.protocol.ftp.FtpURLConnection
Kontrollera detta manuellt!
Förmodligen är det inte ett HTML-dokument  eller så är det inte ett HTTP-dokument
eller liknande!


Länkar att scanna:

http://www.sundsvall.se/vartweb/nov96/07.html#anchor289183
```

http://www.arosnet.se/werbeka/krog/tomsards.htm

```
<html>
<head><title>TOMAT SARDA</title></head>
<body bgcolor="#ffcdcd">
<h3>International Netrestaurant</h3>
<h1>Tomat Sarda</h1>
<hr>
Du beh&ouml;ver: (f&ouml;r 4 personer)<br>
4 stora (8 sm&aring;) tomater<br>
4 sardiner i olja<br>
2 &auml;gg, h&aring;rdkokta<br>
50 g sm&ouml;r<br>
citronsaft<br>
peppar, salt, persilja<p>
Sk&auml;r med en spetsig kniv fr&aring;n tomaternas ovansida ett konformat h&aring;l
och ta ur k&auml;rnorna med en tesked. Hacka &auml;ggen s&aring; fint som m&ouml;jligt
och blanda dem med sardinerna och sm&ouml;ret. Tills&auml;tt en matsked citronsaft,
peppar och salt. R&ouml;r igenom en g&aring;ng till och fyll tomaterna med smeten.
Stoppa i en liten kvist persilja &ouml;verst.
<hr>
Tala om det f&ouml;r dina v&auml;nner, om det smakade bra.<br>
Tillbaka till <a href="krog.htm">Menyn</a><hr>
8.10.1995 by the "chef"<br>
<a href="mailto:werbeka@arosnet.se">werbeka@arosnet.se</a>
<img src="/cgi-bin/counter.exe?-stransp+-t000000+wbkkro0045"></img><br>
</body>
</html>
```

Resultat tomat.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running                  /usr/local/apps/JDK1.1.1/bin/java                   Barbapappa
http://www.arosnet.se/werbeka/krog/tomsards.htm

Bra länkar:

Sida: http://www.arosnet.se/werbeka/krog/tomsards.htm
Länk: http://www.arosnet.se/werbeka/krog/krog.htm
Response-code: 200
Response-message: OK


Dåliga länkar:


Länkar att scanna:
```

http://www.ktv.fi/kovi1596.htm

```
<html>
<head>
<title>Kommunen och Vi</title>
</head>
<body background="tausta.gif">
<TABLE WIDTH=100%>
<TR><TD WIDTH=175 VALIGN=TOP><img src="logo2.gif" ALT="KTV"></TD><TD>


<center>
<table border=0>
<tr>
<td><a href="kovi.htm"><img src="e_.gif"></a></td>
<td><a href="index.html"><img src="k_.gif"></a></td>
</tr>
```

```
</table>
</center>

<ul>
<h2>Kommunen och Vi 15/96</h2>
<b>


Ledaren 15/96: <A HREF="#1">Oktober är nära</A> <br>
<A HREF="#2">Egenkontroll vaccin mot matförgiftningar</A> <br>
<A HREF="#3">Egenkontroll ger kvalité</A> <br>
<A HREF="#4">Styrelsebeslut: Fortsatt förbundsförsäkring</A> <br>
<A HREF="#5">Studiedagar i höst</A> <br>



<p>
<hr>
<br><h2>Ledaren 15/96:<br>
<A NAME=1>Oktober är nära</h2></A><BR>
<p>

Två viktiga val förrättas i Finland i slutet av oktober. Valet av kommunfullmäktige är
redan en tradition som bör observeras av varje KAT-medlem. Det är inte en småsak vilka
fullmäktigeledamöter beslutar om kommunernas frågor. Eller hurudan ideologi de står
för. Finns där en samling av privatiseringsfanatiker eller försvarare av trygga och
kvalitativa kommunala tjänster. Valet i oktober görs för fyra år framåt. Med andra ord
till utgången av år 2000.
<p>

Det lönar sig inte att stå och vänta med händerna i fickorna och se vad som komma
skall, utan istället bör man själv välja en lämplig och bra kandidat.
<p>

Samtidigt väljs också de första parlamentsledamöterna till Europaparlamentet från vårt
land. De nu sittande parlamentarikerna valdes undantagsvis av riksdagen.
<p>

Europaparlamentet kan kännas avlägset, men så är det nödvändigtvis inte. I ett
integrerat Europa inverkar allt på allting och det är viktigt att också de rätta
människorna väljs till parlamentet som ska fatta beslut och leda utvecklingen.
<p>

I dessa val kan det vara skäl att fundera två gånger, om man ska ge sin röst till en
kändis, som inte nödvändigtvis har en blekaste blå aning om olika saker, eller om man
ger sin röst till en person som är väl insatt i olika frågor. Det bestämmer Du.

<p>
<hr>
<p>
<A NAME=2><h2>Egenkontroll vaccin mot matförgiftningar</h2></A><BR>
<p>
En matförgiftning är alla köksanställdas mardröm. Egenkontroll är ett effektivt vaccin
mot matförgiftningar och i kommunala storkök i Finland följer man hela tiden med att
maten är av toppkvalité.
<p>

I sommarhettan och i rötmånadens tider får vi läsa nyheter om matförgiftningar runtom
i världen. I Japan har till och med dödsfall inträffat och cirka 8000 människor har
insjuknat. Smittokällan var länge okänd och skolluncherna undersöktes noggrant,
eftersom skolbarn drabbades mest. Rädisor visade sig sedan vara roten till det onda i
Japan.
<p>

Hur ser det då ut i våra skolkök, där tusentals skolluncher lagas? Kommunen och Vi har
besökt ett centralkök i Lovisa där ungefär tusen portioner bereds dagligen.
<p>

I Lovisa finska skolans centralkök och distributionskök står ångan nästan i taket. Det
är kokhett både ute och inne. Sensommarens värmebölja sätter personalen och deras
yrkeskunnighet på prov.
<p>

Första lunchen serveras vid elva, men före det ska allting vara klart och maten till
tre daghem förs från köket redan klockan tio. Förberedelserna för dagens rätt -
köttfärs - har redan börjat dagen innan. Köttfärslimporna har stekts i ugnar,
potatismos och riven morot tillagas på förmiddagen. Då uppvärms också köttfärsen.
<p>

Från storköket serveras maten till lågstadiet och från distributionsköket delas maten
ut till elever och lärare på högstadiet och gymnasiet. I centralköket i
lågstadiebyggnaden arbetar ansvariga kokerska Rea Rissanen, ansvariga kokerska Raili
Harju och köksbiträdena Irene Weppling-Airikka samt Kyllikki Miettinen, som varit med
```

om skolans nästan hela kökshistoria i och med de 23 år hon arbetat i skolköket. I distributionsköket jobbar köksbiträdena Terttu Husu, Airi Salonen och Eeva Vihreäluoto.
<p>
<h2>Hygienkontroll</h2>
<p>

Egenkontroll betyder i deras arbete en ständig uppföljning av att maten har rätt temperatur och att hygienen är så bra som möjlig i köket och matsalarna. I och för sig är inte egenkontroll något nytt inom köksbranschen, där man alltid har varit noggrann med hygienen. Egenkontroll betyder att vi skriver upp till exempel temperaturen på maten. Förr hade vi inte heller så ofta desinfiering av t.ex. bord och nu gör vi det varannan vecka, förklarar köksbiträde Terttu Husu.
<p>

Ansvariga kokerska Rea Rissanen visar digitalmätaren som används vid mätning av värmen på maten. Det är inte så svårt att hålla maten varm, för vi har dessa värmeskåp och maten ska vara minst 60 grader. Vi har mera problem med att snabbt avkyla maten, för vi har inte några sådana avkylningsskåp och i sommarhettan kan det vara svårt att få maten snabbt att svalna.
<p>

<h2>Kontrollkedja</h2>
<p>

Egenkontroll i storköken betyder inte bara mätning av värme, utan i mångt mycket är det fråga om hygien och kontrollen kan kallas för en kedja som börjar redan hos livsmedelstillverkaren. Färska livsmedel är ett måste.
<p>

Terttu Husu och Airi Salonen berättar hur de jobbar turvis och parvis med olika uppgifter i denna egenkontrollkedja. Vi har rivit morötter idag innan vi kom hit till distributionsköket. En noggrann tvättning och genomgång av rotfrukter och sallader är mycket viktig, om man vill förhindra matförgiftningar. Dessutom är det självklart att vi tvättar händerna ofta. Det är en viktig del av egenkontrollen, understryker Terttu Husu och Airi Salonen.
<p>

Strax innan maten ska serveras förs den i värmeskåp till distributionsköket över skolgården.
<p>

Nu är det inte så jobbigt, men i snö och slask är det mindre roligt att skuffa värmeskåpet hit till distributionsköket, säger Eeva Vihreäluoto.
<p>

I matsalen finns tre utdelningsbord och strax före elva är de första eleverna bakom dörren. Maten delas ut i en otroligt snabb takt. Eleverna sätter sig vid borden och när de första ätit, får de sista sin mat. Det är liv och rörelse i den låga matsalen, men kökspersonalen säger att oljudet inte stör dem. Efter första matpasset följer en snabb torkning av borden och diskning.
<p>
Vi har fått nyligen mera tallrikar, så nu behöver vi inte diska alla tallrikarna mellan dessa två matutdelningar. Då var det nog ganska stressande att hinna med allt, berättar Eeva Vihreäluoto.
<p>

Köksbiträdena vet vad de ska göra. Terttu Husu torkar bord och för kärlen till disken. Eeva Vihreäluoto fördiskar och Airi Salonen står vid diskmaskinen och matar den än med brickor än med glas och bestick. Här går det undan.
<p>

Ungefär 45 minuter senare kommer följande elevrusch. Personalen arbetar målmedvetet och i distributionsköket - som inte är särdeles stort - fylls vagnarna på nytt med rena kärl. Snabbt skuffas vagnarna ut i matsalen. Köksbiträdena ställer rena kärl, drycker och köttfärsen, potatismosen och rivna moroten på utdelningssborden. Effektivt och klart för nästa elevgrupper.
<p>

När de sista eleverna och lärarna lämnat matsalen kan kökspersonalen "dra andan" för första gången sedan de började på morgonen. Efterstädningen och kontrollarbetet kan nu få vänta några minuter. Egenkontrollanterna är värda en paus.
<p>

BENITA KOPRA

<p>
<hr>
<p>
<A NAME=3><h2>Egenkontroll ger kvalité</h2></A>
<p>

Appendix C, testkörningar

Inom måltidsservicen används egenkontroll som garanterar att livsmedlen och måltiderna har god kvalité. Det betyder att storkökspersonalen ser till att de måltider som serveras fyller alla de krav som ställts på egenkontroll.
<p>

Egenkontroll garanterar att man kan äta maten tryggt och man uppnår den goda kvalitén genom att iaktta de bestämmelser som finns om hantering och tillverkning av livsmedel. Kökspersonalen har en mycket viktig roll i denna egenkontroll och det betyder att man måste hela tiden kontrollera och uppföljа hela matlagningsprocessen från råvara till matbord och även därefter, dvs städning, diskning och avfall.
<p>

Måltidsservicens egenkontroll kan i korthet beskrivas som hygien, temperaturmätning, städning och rengöring, avfallshantering, näringsrekommendationer och kundernas reaktioner.
<p>

Kvalitéten på maten och livsmedlen har traditionellt varit mycket bra i Finland och storkökspersonalen har burit det största ansvaret för den goda kvalitén. Egenkontroll medför ytterligare ansvar och uppföljning av den goda och trygga kvalitén på målitiderna i våra storkök.
<p>

En regelbunden egenkontroll kunde vara ett kriterium när man värderar arbetskraven i köksarbetet, eftersom egenkontroll kräver av personalen större kunnande och ständig uppföljning.

<p>
<hr>
<p>
<A NAME=4><h2>Styrelsebeslut: Fortsatt förbundsförsäkring</h2></A>
<p>

Den nya förbundsstyrelsen för Kommunsektorns fackförbund KAT har sammanträtt för första gången i slutet av augusti. Det första mötet präglades av rutinfrågor. Några beslut betyder att KAT-föreningarna bör när de planerar verksamheten för nästa år ta i betraktande vissa evenemang under år 1997. Förbundsförsäkringen kommer dessutom att fortsätta enligt styrelsens beslut.
<p>

Under föreningsåret 1997 ordnas flera tillställningar och jippon för KAT-föreningarnas medlemmar.
<p>

<h2>Sommarjippo på Särkänniemi</h2>
<p>

Det traditionella sommarjippot arrangeras i Särkänniemi nöjespark i Tammerfors 8 - 9.6.1997. KATs sommarjippo är avsett för hela familjen.
<p>

<h2>Storträff i Valkeakoski</h2>
<p>

KATarna samlas till en storträff i samband med Työväen Musiikkitapahtuma i Valkeakoski 25 - 27.7.1997. Denna träff är en av de många tillställningar som ordnas under föreningsåret.
Närmare information om både sommarjippot och storträffen ges till föreningarna via cirkulär och Kommunen och Vi -tidningen.
<p>

<h2>Förbundsförsäkringen fortsätter i Pohjola</h2>
<p>

Den nya förbundsstyrelsen beslöt också att KATs medlemsförsäkring i Pohjola - föräkringsbolag fortsätter även i nästa år. Medlemskortet är också medlemmens försäkringskort, så som under tidigare år. Noggrann information om medlemskortet och dess postning ges senare bl.a. via Kommunen och Vi -tidningen.

<p>
<hr>
<p>
<A NAME=5><h2>Studiedagar i höst</h2></A>
<p>

I höst ordnas två studiedagar för olika yrkesgrupper: för dagvårdare och familjedagvårdare samt för sjukhus-, vård- och anstaltbiträden.
<p>

Den första för dagvårdare och familjedagvårdare är 26-27.10. i Karis. Ansökningar till dessa studiedagar görs på KATs kursblankett och de bör vara på KATs utbildningsavdelning senast 9.9.1996.

```
<p>

Också studiedagar för sjukhus-, vård- och anstaltsbiträden arrangeras i Karis 9-10.11.
och sista ansökningsdagen är 23.9.
<p>

KATs svenskspråkiga kurser och studiedagar på Kommunsektorns skola i Karis i höst
finns presenterade i KATs kursbroschyr. Föreningarnas studiesekreterare har av dessa
kursbroschyrer.
<p>

Ytterligare information om kursinnehåll, avgifter, reseersättningar m.m. ges av
ombudsmännen på distriktsbyråerna samt av KATs utbildningsavdelning.
<p>

KATs utbildningsbroschyr för år 1997 har utkommit. I broschyren presenteras all den
service som förbundet bjuder på i nästa år. Föreningarnas studiesekreterare har av
dessa broschyrer.
<p>
</ul>
</TD>

<TR><TD WIDTH=175 VALIGN=TOP><img align=left src="merkki.gif" ALT=" "></TD>
</TR>
</TABLE>
</body>
</html>
```

Resultat vaccin.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running /usr/local/apps/JDK1.1.1/bin/java Barbapappa http://www.ktv.fi/kovi1596.htm
LÄGG TILL http://www.ktv.fi/kovi1596.htm#1
LÄGG TILL http://www.ktv.fi/kovi1596.htm#2
LÄGG TILL http://www.ktv.fi/kovi1596.htm#3
LÄGG TILL http://www.ktv.fi/kovi1596.htm#4
LÄGG TILL http://www.ktv.fi/kovi1596.htm#5

Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

LÄGG TILL http://www.ktv.fi/kovi1596.htm#1
```

Appendix C, testkörningar

```
Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#1
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/kovi1596.htm#1
LÄGG TILL http://www.ktv.fi/kovi1596.htm#2

Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#2
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows
```

Appendix C, testkörningar

```
Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/kovi1596.htm#2
LÄGG TILL http://www.ktv.fi/kovi1596.htm#3

Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#3
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/kovi1596.htm#3
LÄGG TILL http://www.ktv.fi/kovi1596.htm#4

Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows
```

```
Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#4
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/kovi1596.htm#4
LÄGG TILL http://www.ktv.fi/kovi1596.htm#5

Bra länkar:

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi1596.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi1596.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi1596.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi1596.htm#4
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi1596.htm#5
Länk: http://www.ktv.fi/kovi1596.htm#5
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/kovi1596.htm#5
```

http://www.ktv.fi/

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>Kunta-alan ammattiliitto KTV</title>
<META NAME="GENERATOR" CONTENT="Mozilla/3.01Gold (Win95; I) [Netscape]">
</head>
<body background="tausta.gif">
<TABLE WIDTH=100%>
<TR><TD WIDTH=175 VALIGN=TOP><img src="logo2.gif" ALT="KTV"></TD><TD>


<h2>Kunta-alan ammattiliitto KTV ry</h2>
Kunta-alan ammattiliitto KTV on Suomen suurin ammattiliitto, joka edustaa 230 000
kuntien, kuntayhtymien, seurakuntien ja kuntalaisille palveluja tuottavien yritysten
työntekijää.
```

```
<p>
<img align=bottom src="whitebal.gif" ALT="*">perustettu vuonna 1931 nimellä Suomen
kunnantyöntekijäin liitto
<br><img align=bottom src="whitebal.gif" ALT="*">KTV:läiset tuottavat työllään
suurimman osan peruspalveluista Suomessa
<br><img align=bottom src="whitebal.gif" ALT="*">jäsenkunta tekee kunnissa töitä yli
2000 <a href="sivu2.htm">ammattinimikkeellä</a>
<br><img align=bottom src="whitebal.gif" ALT="*">jäsenistä enemmistö naisia
<br><img align=bottom src="whitebal.gif" ALT="*"><a href="sivu3.htm">jäsenmäärä</a>
vaihtelee kuntien tilanteen mukaan
<br><img align=bottom src="whitebal.gif" ALT="*">KTV on <a
href="http://www.sak.fi/">Suomen Ammattiliittojen Keskusjärjestön SAK:n</a> suurin
jäsenliitto
<p>
 <a href="sivu4.htm"></a>
<br>

<table border="4">
<td><small>
<a href="sivu5.htm">PALVELUKSESSANNE</a>
<br>
 <a href="sivu10.htm">KTV TODAY</a>
<br>
 <a href="me.htm">Kunta ja Me<br> (4/97)</a>
<br>
 <a href="kovi.htm">Kommunen och Vi (4/97)</a>
<br></td>

<td valign=top><small><a href="opj1.htm">OPISKELIJAN OMAT SIVUT</a>
<br>
<a href="yhdvuosi.htm">YHDISTYSVUOSI 1997</a>
<br></td>

<td valign=top><small><a href="yhdistys.htm">YHDISTYSTEN KOTISIVUT</a>
</td>
</table>
<br>
<p>


</TD>

<TR><TD WIDTH=175 VALIGN=TOP><img align=left src="merkki.gif" ALT=" "></TD>
<TD><H6 align=center><a href="mailto:Eija.Kayhko@ktv.fi">Ylläpito: KTV-
Webmaster</a></H6></TD></TR>
</TABLE>


</body>
</html>
```

Resultat vaccinH.txt

```
Exec= java
Test av java1.1.1. Meddela ev. skumma saker till admin@ida.his.se
Running /usr/local/apps/JDK1.1.1/bin/java Barbapappa http://www.ktv.fi/
LÄGG TILL http://www.ktv.fi/sivu2.htm
LÄGG TILL http://www.ktv.fi/sivu3.htm
LÄGG TILL http://www.ktv.fi/sivu4.htm
LÄGG TILL http://www.ktv.fi/sivu5.htm
LÄGG TILL http://www.ktv.fi/sivu10.htm
LÄGG TILL http://www.ktv.fi/me.htm
LÄGG TILL http://www.ktv.fi/kovi.htm
LÄGG TILL http://www.ktv.fi/opj1.htm
LÄGG TILL http://www.ktv.fi/yhdvuosi.htm
LÄGG TILL http://www.ktv.fi/yhdistys.htm

Bra länkar:

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/sivu2.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/sivu3.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.sak.fi/
Response-code: 200
Response-message: OK
```

Appendix C, testkörningar

```
Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/sivu4.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/sivu5.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/sivu10.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/me.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/opj1.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/yhdvuosi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/
Länk: http://www.ktv.fi/yhdistys.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/sivu2.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu2.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/sivu3.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu3.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows


Dåliga länkar:
```

Appendix C, testkörningar

```
Länkar att scanna:

LÄGG TILL http://www.ktv.fi/sivu4.htm#1
LÄGG TILL http://www.ktv.fi/sivu4.htm#2
LÄGG TILL http://www.ktv.fi/sivu4.htm#3

Bra länkar:

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/sivu4.htm#1
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/sivu4.htm#2
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/sivu4.htm#3
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/me.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/kovi.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu4.htm
Länk: http://www.ktv.fi/me1696.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:

http://www.ktv.fi/sivu4.htm#1
http://www.ktv.fi/sivu4.htm#2
http://www.ktv.fi/sivu4.htm#3

Bra länkar:

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/sivu6.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/sivu11.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/sivu7.htm
Response-code: 200
```

```
Response-message: Document follows

Sida: http://www.ktv.fi/sivu5.htm
Länk: http://www.ktv.fi/sivu8.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/sivu10.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/sivu10.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me497.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me397.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me297.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me197.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1896.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1796.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1696.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1596.htm
Response-code: 200
```

```
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1496.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/me.htm
Länk: http://www.ktv.fi/me1396.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/kovi497.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/kovi1796.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/kovi1696.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/kovi.htm
Länk: http://www.ktv.fi/kovi1596.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/opj4.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/opj2.htm
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/opj5.htm
Response-code: 200
```

```
Response-message: Document follows

Sida: http://www.ktv.fi/opj1.htm
Länk: http://www.ktv.fi/opj3.htm
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/yhdvuosi.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/yhdvuosi.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:


Bra länkar:

Sida: http://www.ktv.fi/yhdistys.htm
Länk: http://www.ktv.fi/index.html
Response-code: 200
Response-message: Document follows

Sida: http://www.ktv.fi/yhdistys.htm
Länk: http://www.ktv.fi/yhd215/
Response-code: 200
Response-message: Document follows


Dåliga länkar:


Länkar att scanna:
```

Appendix D, innehållsförteckning för “testres.tar.gz”

Filen testres.tar.gz innehåller de fullständiga testsviterna oredigerade.

Öppnas genom att tex i ett UNIX-shell skriva:

```
prompt> gunzip testres.tar.gz
prompt> tar -xvf testres.tar
```

testres.tar.gz

```
-rw-------   1 dv4       407621 Aug 12 19:59 arrogans.txt
-rw-------   1 dv4         7703 Aug 12 19:59 förlösa.txt
-rw-------   1 dv4          221 Aug 12 20:00 förlösaH.txt
-rw-------   1 dv4         1679 Aug 12 20:01 helikopter.txt
-rw-------   1 dv4      1015801 Aug 12 20:47 helikopterH.txt
-rw-------   1 dv4       394079 Aug 12 19:55 landa.txt
-rw-------   1 dv4       306640 Aug 12 19:53 landaH.txt
-rw-------   1 dv4         8619 Aug 12 19:53 milliard.txt
-rw-------   1 dv4          782 Aug 12 19:55 palma.txt
-rw-------   1 dv4        38827 Aug 12 19:56 palmaH.txt
-rw-------   1 dv4         1362 Aug 12 19:56 puh.txt
-rw-------   1 dv4         8895 Aug 12 20:07 puhH.txt
-rw-------   1 dv4      4705706 Aug 12 21:26 sabbat.txt
-rw-------   1 dv4         1098 Aug 12 21:25 sabbatH.txt
-rw-------   1 dv4            0 Oct  1 18:40 testresinnehall.txt
-rw-------   1 dv4          375 Aug 12 20:03 tomat.txt
-rw-------   1 dv4          218 Aug 12 20:03 tomatH.txt
-rw-------   1 dv4         6580 Aug 12 20:06 vaccin.txt
-rw-------   1 dv4         8450 Aug 12 20:06 vaccinH.txt
```