

5DV135 HT18  
Applikationsutveckling i java  
OU2 - XML och Trådar

Typ	Namn	Email
Lärare	Johan Eliasson	johane@cs.umu.se
Handledare	Anders Broberg	bopspe@cs.umu.se
Handledare	Elias Åström	eam@cs.umu.se
Handledare	Jakob Lindqvist	jakobl@cs.umu.se
Handledare	Klas af Geijerstam Unger	klasa@cs.umu.se
Handledare	William Viktorsson	williamv@cs.umu.se

# Innehåll

<b>1</b>	<b>Introduktion</b>	<b>1</b>
<b>2</b>	<b>Användarhandledning</b>	<b>1</b>
2.1	Beskrivning . . . . .	1
2.2	Handledning i programmet . . . . .	1
2.3	Kompilering och körning . . . . .	2
2.4	Körning . . . . .	2
<b>3</b>	<b>Systembeskrivning</b>	<b>2</b>
3.1	MVC . . . . .	2
3.1.1	Kontroller . . . . .	3
3.1.2	Vy . . . . .	3
3.1.3	Modell . . . . .	3
3.1.4	Workers . . . . .	3
3.2	Flödesbeskrivning . . . . .	3
3.3	XML-parsing . . . . .	4
3.4	Menyn . . . . .	4
3.5	Vidareutveckling . . . . .	4
3.6	Trådsäkerhet . . . . .	4
3.7	UML-diagram . . . . .	5

## 1 Introduktion

Denna rapport beskriver utförandet av den andra obligatoriska uppgiften i kursen Applikationsutveckling i java (HT-18) vid Umeå Universitet. I denna uppgift skrivs en grafisk applikation i java med gränssnittsbiblioteket Swing. Applikationen är en radiotablå applikation för Sveriges Radio (SR). Applikationen hämtar tablåer med hjälp av HTTP get requests till SR:s API och presenterar kanalerna och deras tablå för användaren. Då det är en java applikation kan de köras på alla operativsystem som implementerat en relativt modern version av Java Virtual Machine (JVM).

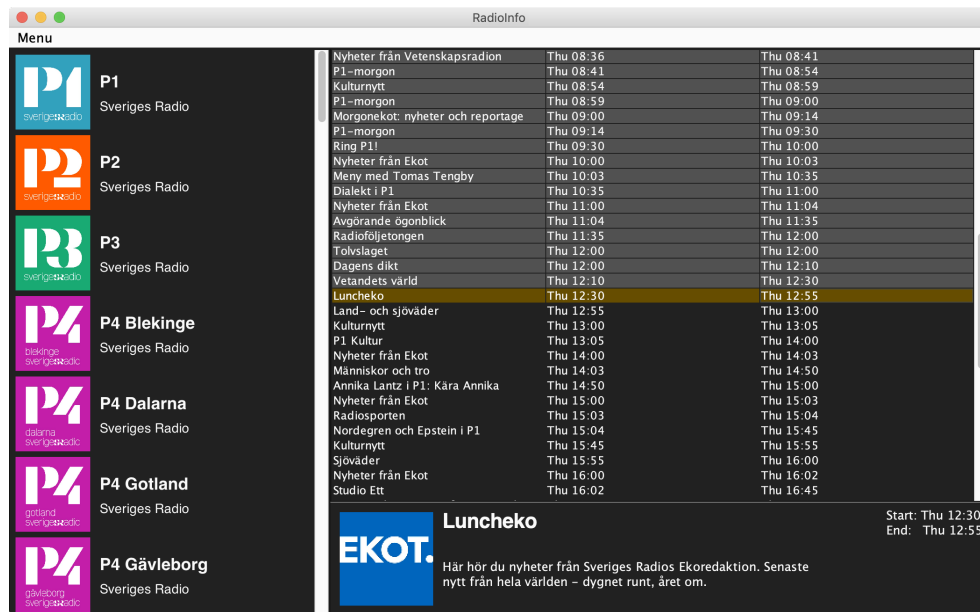
## 2 Användarhandledning

### 2.1 Beskrivning

Programmet består av två paneler. En panel till vänster där alla kanaler listas upp och en panel till höger där Tablån listas upp. Nedanför den panelen visas en beskrövning av ett valt program ur tablån. I tablån visas redan gågna program i grått, valt program i guld och kommande program i den mörkgråa bakgrundsfärgen.

### 2.2 Handledning i programmet

För att välja en ny kanal klickar man bara på en av dem i vänsta spalten och då visas dess tablå upp. Kanalerna kan uppdateras genom att klicka i menyn och välja menyvalet "Update Channels" (CMD/CTRL + C) eller vänta då de uppdateras varje timme som programmet körs. Radioprogrammen cacheas så de behöver inte hämtas flera gånger. De behöver dock hämtas om varje timme då kanalerna uppdateras så att de alltid är aktuella. Om de vill hämtas om tidigare kan menyvalet "Update Shows" (CMD/CTRL + S) väljas och då tvingar man en ny hämtning av tablåerna nästa gång en kanal väljs. Om ingen tablå finns att hämta presenteras ingen data. Om applikationen inte kan ansluta till SR:s servrar presenteras en informationsruta som talar om felet.



The screenshot shows a Java application window titled "RadioInfo". On the left is a "Menu" with logos and names for various radio stations: P1, P2, P3, P4 Blekinge, P4 Dalarna, P4 Gotland, and P4 Gävleborg. The main area displays a program schedule for Thursday. The schedule is a table with three columns: program name, start time, and end time. The "Lunchkeo" program is highlighted in yellow. Below the schedule, there is a section for "EKOT." with a description and the current time range.

Program	Start	End
Nyheter från Vetenskapsradion	Thu 08:36	Thu 08:41
P1-morgon	Thu 08:41	Thu 08:54
Kulturmytt	Thu 08:54	Thu 08:59
P1-morgon	Thu 08:59	Thu 09:00
Morgonekot: nyheter och reportage	Thu 09:00	Thu 09:14
P1-morgon	Thu 09:14	Thu 09:30
Ring P1!	Thu 09:30	Thu 10:00
Nyheter från Ekot	Thu 10:00	Thu 10:03
Meny med Tomas Tengby	Thu 10:03	Thu 10:35
Dialekt i P1	Thu 10:35	Thu 11:00
Nyheter från Ekot	Thu 11:00	Thu 11:04
Avgränsande ögonblick	Thu 11:04	Thu 11:35
Radioföljetongen	Thu 11:35	Thu 12:00
Tolvslaget	Thu 12:00	Thu 12:00
Dagens dikt	Thu 12:00	Thu 12:10
Vetandets värld	Thu 12:10	Thu 12:30
<b>Lunchkeo</b>	<b>Thu 12:30</b>	<b>Thu 12:55</b>
Land- och sjöväder	Thu 12:55	Thu 13:00
Kulturmytt	Thu 13:00	Thu 13:05
P1 Kultur	Thu 13:05	Thu 14:00
Nyheter från Ekot	Thu 14:00	Thu 14:03
Människor och tro	Thu 14:03	Thu 14:50
Annika Lantz i P1: Kara Annika	Thu 14:50	Thu 15:00
Nyheter från Ekot	Thu 15:00	Thu 15:03
Radiosporten	Thu 15:03	Thu 15:04
Nordegren och Epstein i P1	Thu 15:04	Thu 15:45
Kulturmytt	Thu 15:45	Thu 15:55
Sjöväder	Thu 15:55	Thu 16:00
Nyheter från Ekot	Thu 16:00	Thu 16:02
Studio Ett	Thu 16:02	Thu 16:45

**Lunchkeo** Start: Thu 12:30 End: Thu 12:55  
Här hör du nyheter från Sveriges Radios Ekoredaktion. Senaste nytt från hela världen - dygnet runt, året om.

Figur 1 – Exempely över programmet

## 2.3 Kompilering och körning

För att bygga en .jar fil av detta program kan man göra på många olika byggsystem såsom Maven eller Ant. Jag har använt mig av IntelliJ som i sin tur använt sig av Ant. För att bygga programmet gör man följande:

Klicka på File, Project Structure. Där man skapar en artifact av en .jar fil där main klassen väljs. Sedan klickar man på Build, Build Artifacts. Sedan kan programmet köras från terminalen.

## 2.4 Körning

Programmet är färdigkompilerad java8 kod och komprimerat till en .jar fil. För att köra RadioInfo körs kommandot:

```
java -jar /YOUR_PATH/RadioInfo.jar
```

## 3 Systembeskrivning

### 3.1 MVC

Koden använder tre huvudmoduler, vy, en kontroller och en modell där vyn och modellen kommunicerar med varandra genom kontrollern. Vyn och modellen är i mitt programinterfases som implementationerna måste implementera för att kunna användas i kontrollern. Modellen sköter i detta fall hanteringen av kanaler och deras tablåer.

### 3.1.1 Kontroller

Kontrollern har en väldigt enkel design och i min implementation skapar och initialiserar den endast vyn och sätter två action listener på meny knapparna. Action listenerarna är båda en instans av en `SwingWorker` som vet om testmodellen och vyn. Detta sätts via konstruktorn hos kontrollerklassen. Om ett fel uppstår så meddelas användaren via en notifikationsruta.

### 3.1.2 Vy

Vyn skapas genom att skapa en klass som implementerar view interfacet. Sedan körs vyns `initialize()` metod för konfigurering av vyn och `display()` metoden för att visa fönstret. All kommunikation med vyn sker genom dess publika metoder som finns i `View` interfacet. Min vy ärver från en `JFrame` för enkel användning då man inte behöver referera till en `JFrame` hela tiden utan bara kan köra metoden. Metoden `initialize` körs i min implementation på huvudtråden medans resten av vyns metoder körs på EDT:n. För att lägga till kanaler till vyn används metoden `addChannel()` som presenterar dem i vyn. För att rensa alla från vyn körs `clearChannelPanel()` metoden. Om man vill tvinga fram en uppdatering av GUI:t kan metoden `update()` köras som updaterar ramen. För att meddela användaren om något kan metoden `notifyUser()` köras med ett meddelande som sedan visas för användaren. För att skapa en tabell används metoden `createTable()` som visare en tabell för användaren.

### 3.1.3 Modell

Modellen implementerar `Model` interfacet och har tre publika metoder en som hämtar kanalerna vid namn `getChannels()`. En som uppdaterar kanalerna med hjälp av en HTTP request till SR:s API vid namn `updateChannels()`. Denna metod tar ett tag att köra då den ska skicka en request och sedan parsea XML svaret. Därav används den så sällan som möjligt i min implementation. Sedan finns metoden `clearSavedShows()` som tar bort alla sparade shows från alla kanaler vilket tvingar att de hämtas på nytt nästa gång de vill nås.

### 3.1.4 Workers

Min implementation använder sig av 3 trådade tjänster. Två av typen `SwingWorker` och en `ScheduledExecutorService`. En `SwingWorker` vid namn `ChannelUpdater` uppdaterar kanalerna i vyn. Den hämtar kanalerna och publiserar EDT:n som publiserar dem till vyn. Om den körningen misslyckas meddelas användaren. Annars körs sedan den andra `SwingWorker` vid namn `ShowGetter` som hämtar tablån för en kanal och publiserar den till EDT:n som visar upp den i vyn. Om denna misslyckas så visar den det för användaren i ett felmeddelande. Den sista trådade tjänsten schemalägger att en uppdatering av kanalerna ska ske varje timme så att informationen alltid är ny och aktuell.

## 3.2 Flödesbeskrivning

När användaren startar programmet skapas för en modell och en vyn som initialiseras. Sedan startas en `ChannelUpdater` som uppdaterar alla kanaler och publiserar dem till vyn. Efter den har kört startas en `ShowGetter` som hämtar tablån för den första kanalen och publiserar den till vyn. När sedan användaren trycker på en kanalknapp körs en `ShowGetter` som kollar om den kanalens tablå redan hämtats. Om den har det så publiserar den dem till vyn, annars hämtar den tablån och publiserar den till vyn, någonting som tar avsevärt mycket längre tid.

### 3.3 XML-parsing

All hämtad data från SR:s API kommer i XML format och måste därav parsas av en XML parser. `DocumentBuilder` som jag använder har flera olika valideringsfunktioner för att validera XML-datan. Då denna data är så otroligt stor och XML-parsing tar ett tag att utföra valde jag att stänga av dem då hastigheten på parsingen ökade märkbart. Detta för att användaren inte ska behöva vänta allt för länge på sin radiotablå.

### 3.4 Menyn

Menyn i mitt program har två alternativ, "Update Channelssom kan klickas på eller nås med kommandot CMD/CTRL + C som kör `UpdateChannels` och "Update Showssom kan klickas på eller nås med kommandot CMD/CTRL + S som rensar alla shows från alla kanaler vilket tvingar en ny hämtning nästa gång de vill nås.

### 3.5 Vidareutveckling

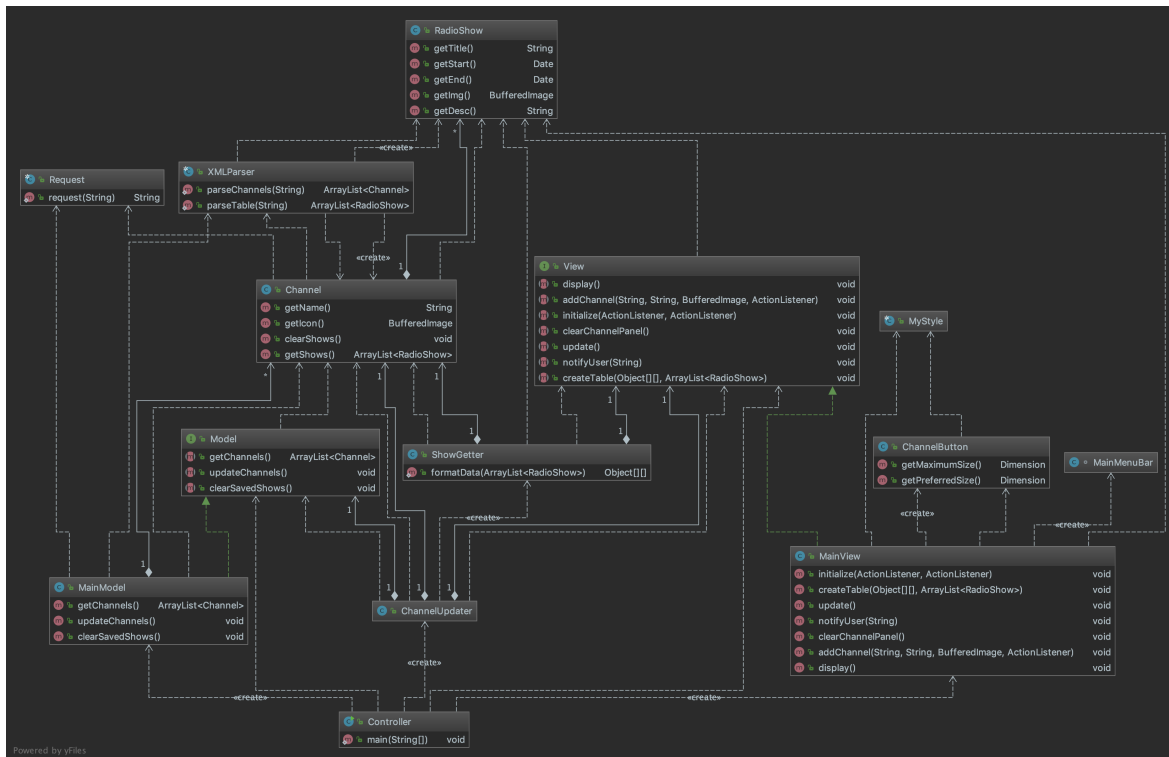
Då jag använt Interfaces för modellen och vyn är vidareutveckling av detta program ganska enkelt. Så länge som man använder uppfyller dess kriterier av metoder så går en egen vy eller modell att användas. Om mer tid funnits hade jag gärna lagt in en laddningssymbol under tiden som kanalerna hämtas eller under tiden som en tablå hämtas. Jag hade även lagt mer tid på GUI:t och korrigerat lite placering och färgval som jag gjort. Färgvalen är lätta att ändra då de ligger i en statisk `MyStyle` klass och används för att rita stylea GUI:t.

### 3.6 Trådsäkerhet

Då jag valt att använda så få trådar som möjligt och ge tydliga ansvarsområden om funktionaliteten till olika klasser där mycket nås statiskt som per definition är trådsäkert är trådsäkerheten hög i min kod. Jag har stresstestat där jag tryckt på många menyval, bytt kanal, klickat i tablån, uppdaterat kanalerna samtidigt men koden håller sig stabil då saker inte ändras ofta eller i onödan. Datan som alla klasser processerar är kanalerna och deras tablåer. Därför är alla metoder som hanterar dem `synchronized` för att undvika att de nås samtidigt. Så att ingen kan skriva över dem när de används utan endast en klass kan arbeta med dem i taget.

### 3.7 UML-diagram

Nedan visas ett UML-diagram över programmet. Controller är huvudprogrammet där main-funktionen finns. Den skapar GUI:t på en tråd och kör sedan alla actionlisteners på nya trådar som bara finns så länge som de exekverar sin funktion. Detta för att GUI:t inte ska frysa när en knapp klickas.



Figur 2 – UML-diagram över Programmet