

Applikationsutveckling i Java

7.5p 5DV135

RADIOINFO

VEDAD CORIC

Handledare: Johan Eliasson, (johane@cs.umu.se)
Elias Åström, (eam@cs.umu.se)
Joel Sandman, (sandman@cs.umu.se)
Klas Af Geijerstam Unger, (klasa@cs.umu.se)
Anders Broberg

Innehåll

1. Introduktion.....	2
2. Användning.....	2
2.1 Program användning	2
2.2 Kompilering och körning via terminal	3
3. System beskrivning.....	4
3.1 Huvudstrukturen	4
3.2 Data Hantering och Data Modell.....	4
3.2.1 Parsing av Schemalagd data	5
3.3 GUI.....	5
3.3.1 Användarinteraktioner	6
3.3 Trådar och Trådsäkerhet	6
4. Slutsatser och Begränsningar	7
5. Referenser	8

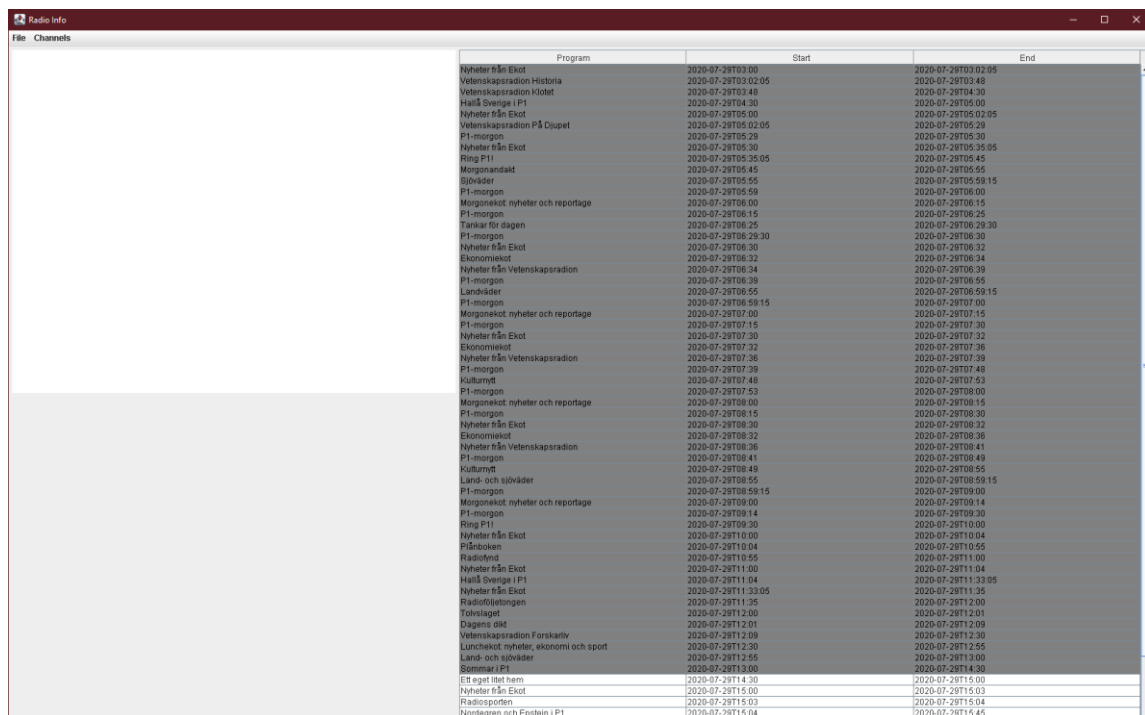
1. Introduktion

Poängen med detta projekt är att skapa en applikation som använder sig av det öppna "web API" från 'Sveriges Radio' (<http://sverigesradio.se/api/documentation/v2/index.html>) som presenterar en tablå av nuvarande radio program via ett GUI (*graphical user interface*).

2. Användning

2.1 Program användning

För dem flesta operativ system som finns i dagens marknad så räcker det med ett klassiskt dubbel klick på 'RadiInfo.jar' filen. Detta kommer då att starta programmet och slänga in användaren till start fönstret.



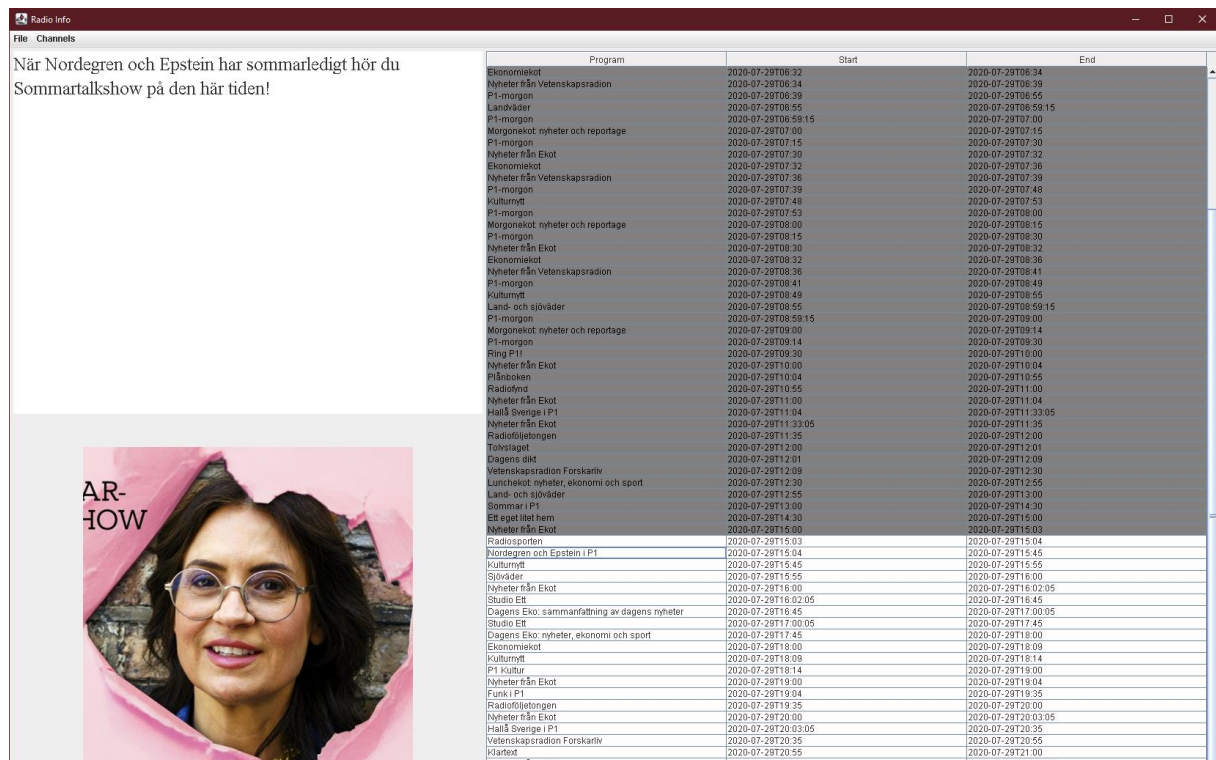
The screenshot shows the 'RadiInfo' application window with a 'Channels' menu. The main area displays a table of radio programs with columns for Program, Start, and End. The table lists various programs from Sveriges Radio, including 'Nyheter från Ekot', 'Veetenskapradion Historia', 'Veetenskapradion Klotet', 'Hälsö Sverige i P1', 'Veetenskapradion På Djupet', 'P1-morgon', 'Ring P11', 'Morgonandakt', 'Spöväder', 'P1-morgon', 'Morgonveetekt nyheter och reportage', 'P1-morgon', 'Tankar för dagen', 'P1-morgon', 'Nyheter från Ekot', 'Ekonomisveet', 'Nyheter från Veetenskapradion', 'P1-morgon', 'Landväder', 'P1-morgon', 'Morgonveetekt nyheter och reportage', 'P1-morgon', 'Nyheter från Ekot', 'Ekonomisveet', 'Nyheter från Veetenskapradion', 'P1-morgon', 'Kulturmötet', 'P1-morgon', 'Morgonveetekt nyheter och reportage', 'P1-morgon', 'Nyheter från Ekot', 'Ekonomisveet', 'Nyheter från Veetenskapradion', 'P1-morgon', 'Kulturmötet', 'Land- och sölväder', 'P1-morgon', 'Morgonveetekt nyheter och reportage', 'P1-morgon', 'Ring P11', 'Nyheter från Ekot', 'P1-morgon', 'Radiosporten', 'Nyheter från Ekot', 'Radiosporten', 'Tidningslaget', 'Dagens ditt', 'Veetenskapradion Forskarliv', 'Land- och sölväder', 'Sammar i P1', 'Et eget till hem', 'Nyheter från Ekot', 'Radiosporten', 'Radiosporten', and 'Nordgren och Epstein i P1'. The table is organized into three panes: a list of programs on the left, a detailed description of the selected program in the middle, and a timeline of the program on the right.

Program	Start	End
Nyheter från Ekot	2020-07-29T03:00	2020-07-29T03:02:05
Veetenskapradion Historia	2020-07-29T03:02:05	2020-07-29T03:48
Veetenskapradion Klotet	2020-07-29T03:48	2020-07-29T04:30
Hälsö Sverige i P1	2020-07-29T04:30	2020-07-29T05:00
Nyheter från Ekot	2020-07-29T05:00	2020-07-29T05:02:05
Veetenskapradion På Djupet	2020-07-29T05:02:05	2020-07-29T05:29
P1-morgon	2020-07-29T05:29	2020-07-29T05:30
Nyheter från Ekot	2020-07-29T05:30	2020-07-29T05:35:05
Ring P11	2020-07-29T05:35:05	2020-07-29T05:45
Morgonandakt	2020-07-29T05:45	2020-07-29T05:55
Spöväder	2020-07-29T05:55	2020-07-29T05:59:15
P1-morgon	2020-07-29T05:59	2020-07-29T06:00
Morgonveetekt nyheter och reportage	2020-07-29T06:00	2020-07-29T06:15
P1-morgon	2020-07-29T06:15	2020-07-29T06:25
Tankar för dagen	2020-07-29T06:25	2020-07-29T06:29:30
P1-morgon	2020-07-29T06:29:30	2020-07-29T06:30
Nyheter från Ekot	2020-07-29T06:30	2020-07-29T06:32
Ekonomisveet	2020-07-29T06:32	2020-07-29T06:34
Nyheter från Veetenskapradion	2020-07-29T06:34	2020-07-29T06:39
P1-morgon	2020-07-29T06:39	2020-07-29T06:55
Landväder	2020-07-29T06:55	2020-07-29T06:59:15
P1-morgon	2020-07-29T06:59:15	2020-07-29T07:00
Morgonveetekt nyheter och reportage	2020-07-29T07:00	2020-07-29T07:15
P1-morgon	2020-07-29T07:15	2020-07-29T07:30
Nyheter från Ekot	2020-07-29T07:30	2020-07-29T07:32
Ekonomisveet	2020-07-29T07:32	2020-07-29T07:36
Nyheter från Veetenskapradion	2020-07-29T07:36	2020-07-29T07:39
P1-morgon	2020-07-29T07:39	2020-07-29T07:48
Kulturmötet	2020-07-29T07:48	2020-07-29T07:53
P1-morgon	2020-07-29T07:53	2020-07-29T08:00
Morgonveetekt nyheter och reportage	2020-07-29T08:00	2020-07-29T08:15
P1-morgon	2020-07-29T08:15	2020-07-29T08:30
Nyheter från Ekot	2020-07-29T08:30	2020-07-29T08:32
Ekonomisveet	2020-07-29T08:32	2020-07-29T08:36
Nyheter från Veetenskapradion	2020-07-29T08:36	2020-07-29T08:41
P1-morgon	2020-07-29T08:41	2020-07-29T08:49
Kulturmötet	2020-07-29T08:49	2020-07-29T08:55
Land- och sölväder	2020-07-29T08:55	2020-07-29T08:59:15
P1-morgon	2020-07-29T08:59:15	2020-07-29T09:00
Morgonveetekt nyheter och reportage	2020-07-29T09:00	2020-07-29T09:14
P1-morgon	2020-07-29T09:14	2020-07-29T09:30
Ring P11	2020-07-29T09:30	2020-07-29T10:00
Nyheter från Ekot	2020-07-29T10:00	2020-07-29T10:04
P1-morgon	2020-07-29T10:04	2020-07-29T10:55
Radiosporten	2020-07-29T10:55	2020-07-29T11:00
Nyheter från Ekot	2020-07-29T11:00	2020-07-29T11:04
Hälsö Sverige i P1	2020-07-29T11:04	2020-07-29T11:33:05
Nyheter från Ekot	2020-07-29T11:33:05	2020-07-29T11:35
Radiosporten	2020-07-29T11:35	2020-07-29T12:00
Tidningslaget	2020-07-29T12:00	2020-07-29T12:01
Dagens ditt	2020-07-29T12:01	2020-07-29T12:09
Veetenskapradion Forskarliv	2020-07-29T12:09	2020-07-29T12:30
Land- och sölväder	2020-07-29T12:30	2020-07-29T12:55
Sammar i P1	2020-07-29T12:55	2020-07-29T13:00
Et eget till hem	2020-07-29T13:00	2020-07-29T14:30
Nyheter från Ekot	2020-07-29T14:30	2020-07-29T15:00
Radiosporten	2020-07-29T15:00	2020-07-29T15:03
Radiosporten	2020-07-29T15:03	2020-07-29T15:04
Nordgren och Epstein i P1	2020-07-29T15:04	2020-07-29T15:45

Figur 1: Start skärm

Start fönstret visar användaren 3 rutor, program tabellen ligger längs till höger och program beskrivning ligger längst till vänster där program bilderna genereras under program beskrivningen. Radio program som startade innan den nuvarande klockslaget markeras grå medans dem som börjar efter nuvarande klockslag markeras ej. Detta görs så att det blir lättare för användaren att se när vilka program körs och vilka program har redan kört. För att få mer information om ett radio program så

går det att bara trycka på radio programmet i tabellen. Detta leder till att en program beskrivning samt möjligen en bild som representerar programmet visas på vänster sida av programmet.



Figur 2: När en användare väljer ett program så kommer en möjlig bild upp och en beskrivning

Om det visar sig vara så att det inte finns någon bild som är associerad till radio programmet så kommer det komma upp en "default" bild som representerar att det inte finns någon bild associerad till just detta program.

Det finns också en 'drop-down' meny högst upp. Denna meny tillåter användaren att välja mellan flera olika kanaler över hela Sverige som P1, P2 osv, samt möjligheten att manuellt uppdatera tabellen, hitta lite info om programmet samt avsluta programmet via en dedikerad 'quit' knapp. Programmet kommer under körning att auto uppdatera tabellen varje timme.

2.2 Kompilering och körning via terminal

För att kompilera och köra programmet så kan man antingen som nämnt innan trycka på applikationen som vilken applikation som helst på datorn, annars kan man via CMD i windows terminalen skriva

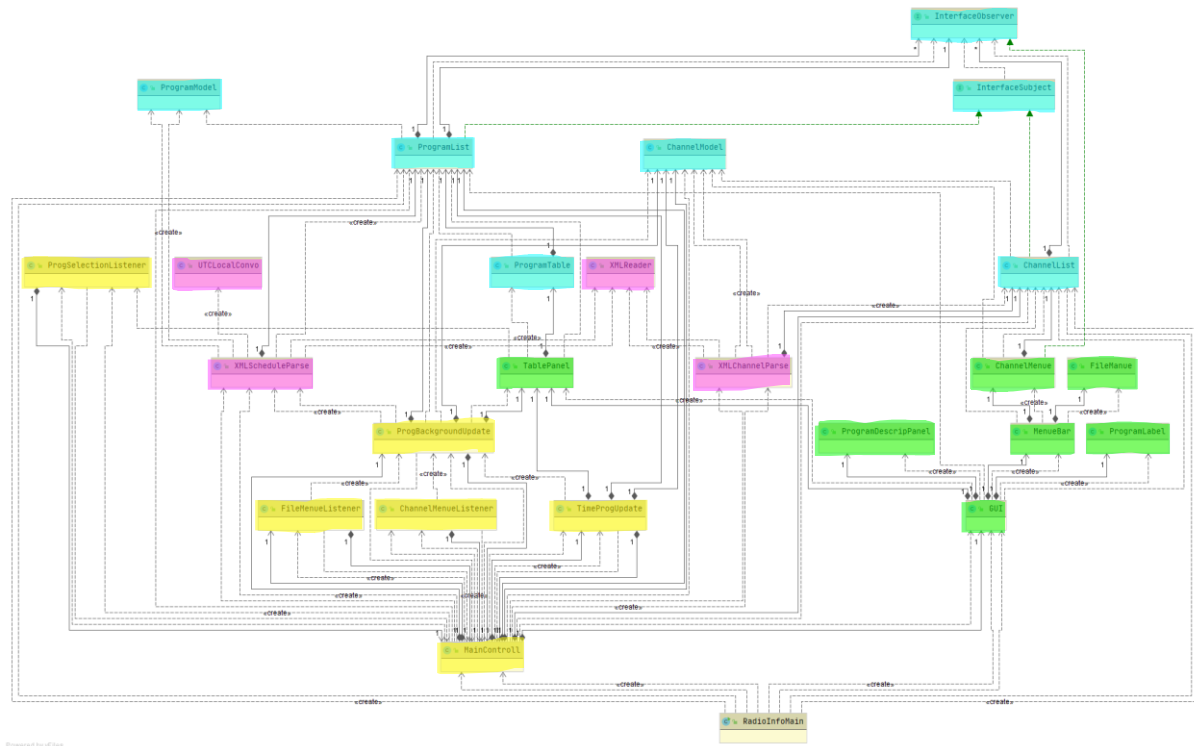
```
java -jar C:\yourPath\yourProgram.jar
```

för att kompilera programmet så kan man antingen köra via IntelliJ och skapa en .jar fil genom att trycka på 'artifact' sen trycka på 'build jar file'. Ett annat sätt är att skapa jar är genom CMD (windows terminal).

```
C:\>cd \mywork
C:\mywork> path c:\JDKPath\;%path%
C:\mywork> javac *.java
C:\mywork> echo Main-Class: jarFile >manifest.txt
C:\mywork> jar cvfm jarFile.jar manifest.txt *.class OR
C:\mywork> ajr cvfe jarFile.jar *.class
```

3. System beskrivning

Hela systemet är strukturerat efter "Model-View-Controller" modellen genom att paketera varje klass i paket. Förutom paketen "Model", "View" och "Controller" så finns det ett fjärde paket som hanterar all XML parning i programmet samt web API hanteringen. Detta paket har namnet "XML"



Figur 3: En förnkad version av UML (fullständigt UML ligger i referenser)

som vi kan se i figure 3 så är varje paket färg markerat, gul för *controller*, blå för *Model*, grön för *View* och rosa för *XML* paketet.

3.1 Huvudstrukturen

När denna applikation kör *main* klassen så startas en ny "Swing thread" (enligt definitionen en "anonymous runnable" metod). Inom denna metod så skapas först *data containers* för kanalerna och programmen. Sen initierar *main* metoden en instans av GUI. I vårt fall så hanterar *Main* klassen all initiering av applikationen.

3.2 Data Hantering och Data Modell

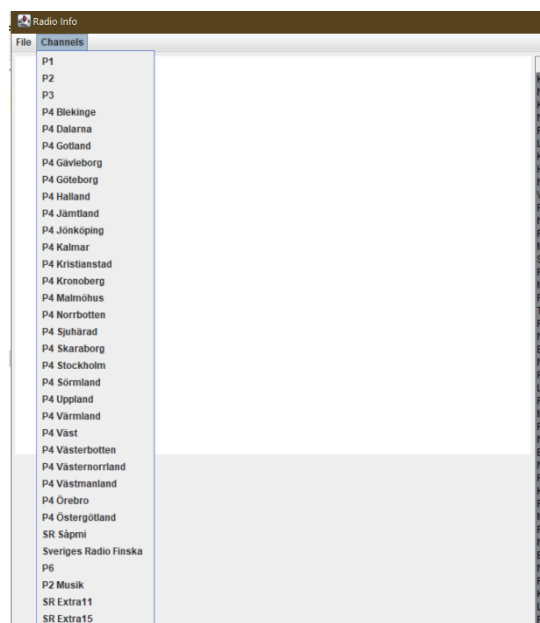
All relevant data som applikationen använder sig av är uppdelat till två delar, kanal och program som hanteras i klasserna *ChannelList* och *ProgramList*. Dessa klasser har inte så mycket annan funktionalitet än att bara spara informationen på ett vettigt sätt. Informationen visas i två typer av *containers* som hanteras av klasserna *ChannelModel* och *ProgramModel*. Dessa *containers* kallas sen av GUI för att visas för användaren. *ChannelModel* visar sin information i form av en "drop-down" meny som ger användaren möjlighet att välja bland olika kanaler. *ProgramModel* klassen används för att fylla på tabellen som ligger i GUI. Vi kan också se i figure 3 att i paketet XML så kan klasserna XMLChannelParse och XMLScheduleParse konstruera respektive *data container* klasserna. XML parsingen använder sig av XMLReader klassen som i sin tur använder sig av DOM parsing.

3.2.1 Parsing av Schemalagd data

Det givna API för att läsa av det schemalagda data har ett antal filtrerings alternativ. Det är dock inte möjligt att filtrera inom en specifik tids områden. Enligt specifikationen så ska tids området vara på +/- 12 timmar från nuvarande tid. Tids informationen som vi får från API:n är i UTC och därför behövs konverteras i lokal tid. Detta hanteras av en av klasserna som ligger i XML paketet, *UTCLocalConvo*. För att ProgramModel klassen skall vara generisk över hela applikationen så känner klassen inte till XML konverterings klassen. Hur det fungerar istället är att XML parsingen fyller på "*ProgramModel data structure*" intern till en egen '*ProgramModel*' objekt. Detta objekt kopieras sedan till GUI. Men problemet kvarstår, hur ska vi få en tabell under just den dagen. Det problemet är inte så super komplicerat när man tänker på det. Det schemalagda datan har ett par dagar redan för att fylla på tidtabellen under en längre tid. Det enda vi behöver göra är att ta reda på innan programmet startar vilken dag det är för användaren. Med hjälp av en funktion så kan vi parsas program från dagen innan och dagen efter, alltså 3 dagar. Datan sparas i form av en ArrayList som vi använder sen för att få fram relevant data från "*web API:n*" för en specifik datum. '*ProgramModel*' klassen har en metod implementerat som hjälper oss lite till, denna metoden filtrerar alla program och kanaler inom +/- 12 timmar tidsområdet.

3.3 GUI

GUI:t är designad med tanke på att det ska vara så simpelt som möjligt. För många så skulle *BoarderLayout* vara en lättare och snabbare sätt att bygga upp GUI:t men jag bestämde mig att implementera *GridLayout*. Designen är uppbyggd av 3 rutor som implementerar *JPanel*. En för en navigerbar tabell som implementerar *JTable* och dem två resterande för programbeskrivning och möjlig programbild (figure 1 och figure 2 visar en fullständig bild på hur systemet ser ut. Det finns också en top meny som är byggd på Java klassen *JMenuBar* som har 2 *JMenu*'s. En av dem två menyerna ger användaren möjlighet att byta kanal som vi kan se på figure 4. Denna meny använder sig av *ChannelList* för att få den nödvändiga datan för dem olika kanalerna. Tabellen i applikationen har 3 spalter, program namn, start tid och slut tid. Tabell designen använder sig av *AbstractTabelModel* som en '*baseline*' vilket sen modifieras till det vi har i applikationen. Denna tabell modell implementerar *ProgramTable* klassen. Detta gjordes för att underlätta initieringen under körning.



Figur 4: "Drop-Down" meny med kanaler som användaren kan välja mellan

3.3.1 Användarinteraktioner

En uppsättning av '*observable/observable*' interfaces användes för både *ChannelList* och *ProgramList* som '*subjects*' och *channelMenu* som '*observable*'. Annars så används *ActionEventListener* för resten. Alla *ActionEventListeners* är initierad från main kontroll klassen.

3.3 Trådar och Trådsäkerhet

Denna applikation använder sig av olika trådar för att uppdatera dan relevanta datan. *SwingWorkers* är kallade för att GUIt ska inte blockas medans systemet uppdaterar eller drar ner ny data från web APllet. På grund av att dessa operationer kan vara långsamma så kan dessa *SwingWorkers* kallas igen medans den gamla versionen fortfarande kör. En annan del som vi måste tackla är användarens roll i systemet. När användaren trycker på tabellen eller väljer en ny kanal, Detta kan väldigt lätt få systemet att bli icke trådsäkert. När användaren trycker på tabellen eller nå liknande så finns det en risk att GUIt låser sig och systemet kraschar. För att få hela applikationen trådsäker använder vi oss av synkroniserade metoder. Alla '*data-containers*' koms åt via *MainControll* klassen, alla *getter* och *setter* funktioner i *MainControll* klassen för all respektive data är synkroniserade.

4. Slutsatser och Begränsningar

Vi börjar med begränsningarna i systemet. GUIt är byggt via Java Swing vilket i sig själv ganska "sloppy" sätt att bygga en GUI på, den fungerar utan några problem och är samt väldigt lätta att skapa något som fungerar. Dock så är Swing väldigt begränsat som den är. Några begränsningar som finns i GUIt är att den är inte optimerad för *window re-sizing* osv. En annan begränsningar som finns är i tabellen, när du startar programmet så börjar du högs upp på tabellen och man måste scrolla ner till nuvarande tid.

Koden som är skriven för denna applikation skrevs om från början efter den förra inlämningen för att det fanns för många grundläggande problem som jag ville ändra. Under omskrivningen så simplifierades koden så mycket som möjligt. Såklart så kunde jag implementera den förra kanal listan jag byggde under förra inlämningen, men då skulle det krävas en annan typ av parsing och en annan, mer komplicerad trådsäkrings metod, vilket i helhet skulle bara riskera att koden inte skulle fungera igen.

[1] UML, fullständig UML diagram: