UMEÅ UNIVERSITET **27 mars 2023**

RadioInfo Version 2.0

Applikationsutveckling i Java - 7,5hp

Namn och email:
Gustaf Söderlund - et14gsd@cs.umu.se

Innehåll

1 Introduktion **1**

2 Ändringar **1**

3 Systembeskrivning **2**
3.1 Trådsäkerhet 2
3.2 Designmönster 3
3.3 View 4
3.4 Controller 7
3.5 Model 9
3.5.1 ProgramInfo 9
3.5.2 ChannelInfo 9
3.5.3 ChannelParsing 9
3.5.4 ProgramParsing 9
3.5.5 Model 10

4 Användarhandledning **11**

1 Introduktion

Programmet RadioInfo läser information från kanaler och tablåer från Sveriges Radio. Användaren presenteras med ett program som representerar en programtablå där man kan läsa information om olika program som sänds på de kanaler Sveriges Radio tillhandahåller.

I rapporten hänvisas fet stil på text till metoder och kursiv stil på text till klasser.

Ändringar från tidigare version finns beskrivet i avsnittet "ändringar".

2 Ändringar

Från tidigare inlämning av laborationsuppgiften behövdes följande åtgärdas:

1. "view.getComboBox().getSelectedItem(); Ej trådsäkert anropa från annan tråd än edt
2. uppdatering manuell och automatisk uppdaterar ej datat från apit: visar bara upp gammalt nedladdat data igen

För att åtgärda punkt 1 har följande anrop *view.getComboBox().getSelectedItem();* förflyttats så att det utförs på EDT. Det är gjort genom att istället anropa metoden i en DoInBackground-tråd anropar man metoden precis innan man skapar en ny tråd.

För att lösa problemet på punkt 2 har en ny actionlistener skapats i Controller klassen. Den här nya actionlistener som skapats använder en metod *parser()* från **model** som hämtar nytt data från apit. Den här actionlistenern anropas genom att antingen klicka på uppdatera-knappen, men också på en timer som kör en gång i timmen.

3 Systembeskrivning

3.1 Trådsäkerhet

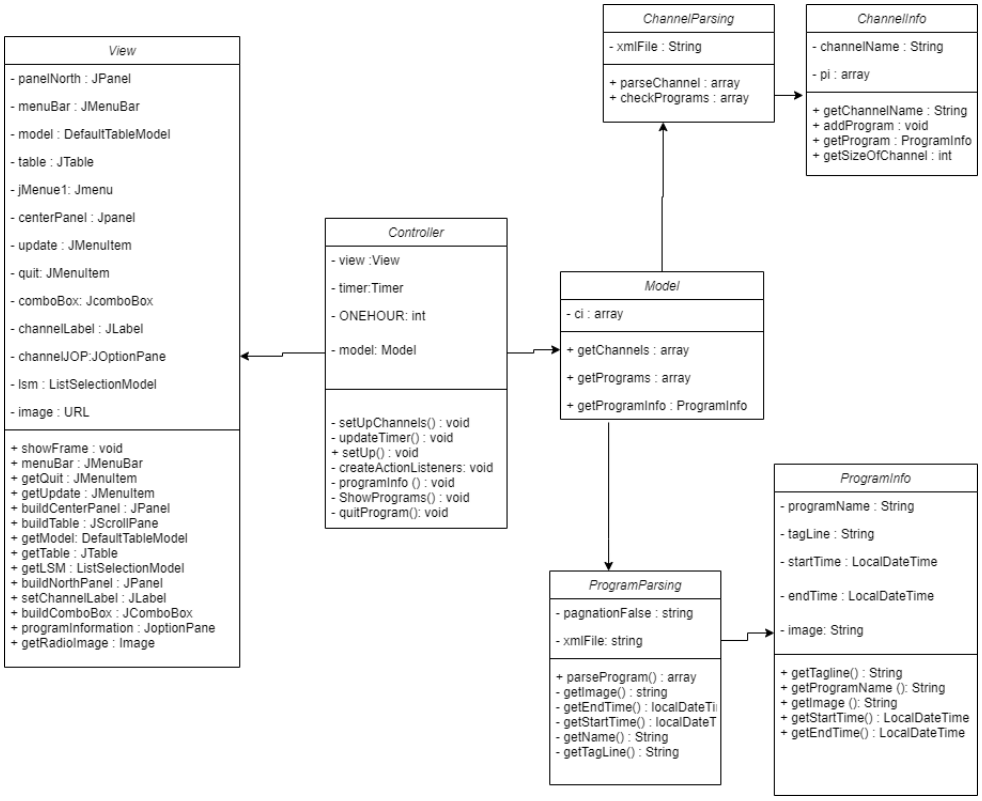
Programmet startar med en Main-tråd som i sin tur startar EDT(Event Dispatching Thread). EDT hanterar uppbyggnaden av programmets utseende och hur hanteringen av knapptryck i programmet sker. I system ska användargränssnittet endast ändras via EDT. Systemet ska inte göra långa metoder på EDT. För att hantera detta används Swingworkers *doInBackground* och *Done* metoder. När användaren interagerar med gränssnittet t.ex. väljer en kanal skapas en *doInBackground* tråd via swingworker som hanterar interaktionen och hämtar t.ex. en lista med program och tider via **Model**. När *doInBackground*-tråden är färdig körs *Done*-metoden på EDT som skickar denna lista till **View** och användaren ser resultatet i programmet.

För att lösa problemet med att flera trådar accessar och ändrar på samma data används keywordet *synchronized* på de metoder som finns i model. Med *synchronized* undviker man att flera trådar kör den metod som använder keywordet *synchronized* samtidigt. Det vill säga om man klickar på uppdatera-knappen i programmet flera gånger kommer inte alla trådar som skapats att köra den funktionen samtidigt och skapa problem. Det som händer istället är att flera trådar skapas och en efter en kör de metoden allteftersom.

3.2 Designmönster

Systemet är uppbyggt med hjälp av MVC-modellen, Model-View-Controller. View i MVC-modellen representerar det grafiska användargränssnittet och är själva programmet som användaren presenteras. Model i MVC-modellen gör uppgifter och arbetet som ska hända i användargränssnittet. Controller i MVC-modellen lyssnar på interaktioner från användargränssnittet och kommunicerar mellan Model och View.

Ett UML-Diagram konstruerades till systemet vilket presenteras i figur 1.

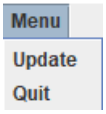


Figur 1: UML-diagram över systemet

3.3 View

View-klassen ansvarar för att skapa ett grafiskt användargränssnitt åt användaren och representerar View från MVC-modellen. Då **View** extendar JFrame konstrueras ett JFrame-objekt i **View**-klassens konstruktor. I JFrame placeras det två stycken JPanels och en JMenuBar.

Metoden *menuBar* skapar ett JMenuBar objekt som placeras i norra blocket av JFramen. Användaren kan se detta som ett menyfönster längst upp i programmet. JMenubar-objektet innehåller ett JMenu-objekt som i sin tur innehåller två olika JMenuItem-Objekt *Update* och *Quit* som användaren kan interagera med. Varje JMenuItem har en separat Actionlistener via **Controller**-klassen.



Figur 2: Menyn som användaren kan se längst upp i fönstret.

Metoden *buildCenterPanel* ansvarar för att konstruera ett JPanel-objekt i det centrala blocket av JFrame. JPanel-objektet får bakgrundsfärgen vit och en tabell placeras i panelen med hjälp av metoden *buildTable*.

Metoden *buildTable* skapar en JTable-objekt. JTable har tre kolummner(*Program, start* och *end*) och antalet rader beror på vilken kanal som valts och hur många program som sänds. JTable-komponenten används som input till skapandet av ett JScrollPane-objekt vilket möjliggör för användaren att scrolla i tabellen om det finns för många rader. Den Centrala JPanelen med tabell åskådliggörs i figur 3.

Program	start	end
Vaken	Sat Aug 21 03:02:00 CEST 2021	Sat Aug 21 04:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 04:00:00 CEST 2021	Sat Aug 21 04:02:00 CEST 2021
Vaken	Sat Aug 21 04:02:00 CEST 2021	Sat Aug 21 05:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 05:00:00 CEST 2021	Sat Aug 21 05:03:00 CEST 2021
P4 Retro	Sat Aug 21 05:03:00 CEST 2021	Sat Aug 21 06:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 06:00:00 CEST 2021	Sat Aug 21 06:03:00 CEST 2021
Ring så spelar vi	Sat Aug 21 06:03:00 CEST 2021	Sat Aug 21 07:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 07:00:00 CEST 2021	Sat Aug 21 07:05:00 CEST 2021
Ring så spelar vi	Sat Aug 21 07:05:00 CEST 2021	Sat Aug 21 07:30:00 CEST 2021
Dalanytt	Sat Aug 21 07:30:00 CEST 2021	Sat Aug 21 07:33:00 CEST 2021
Ring så spelar vi	Sat Aug 21 07:33:00 CEST 2021	Sat Aug 21 08:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 08:00:00 CEST 2021	Sat Aug 21 08:03:00 CEST 2021
Ring så spelar vi	Sat Aug 21 08:03:00 CEST 2021	Sat Aug 21 08:30:00 CEST 2021
Dalanytt	Sat Aug 21 08:30:00 CEST 2021	Sat Aug 21 08:33:00 CEST 2021
Ring så spelar vi	Sat Aug 21 08:33:00 CEST 2021	Sat Aug 21 08:50:00 CEST 2021
Stora händelser i Barnradion	Sat Aug 21 08:50:00 CEST 2021	Sat Aug 21 09:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 09:00:00 CEST 2021	Sat Aug 21 09:04:00 CEST 2021
Melodikrysset	Sat Aug 21 09:04:00 CEST 2021	Sat Aug 21 09:30:00 CEST 2021
Dalanytt	Sat Aug 21 09:30:00 CEST 2021	Sat Aug 21 09:33:00 CEST 2021
Melodikrysset	Sat Aug 21 09:33:00 CEST 2021	Sat Aug 21 10:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 10:00:00 CEST 2021	Sat Aug 21 10:03:00 CEST 2021
Fejk - UR	Sat Aug 21 10:03:00 CEST 2021	Sat Aug 21 10:30:00 CEST 2021
Dalanytt	Sat Aug 21 10:30:00 CEST 2021	Sat Aug 21 10:33:00 CEST 2021
Fejk - UR	Sat Aug 21 10:33:00 CEST 2021	Sat Aug 21 11:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 11:00:00 CEST 2021	Sat Aug 21 11:03:00 CEST 2021
P3 Historia	Sat Aug 21 11:03:00 CEST 2021	Sat Aug 21 11:30:00 CEST 2021
Dalanytt	Sat Aug 21 11:30:00 CEST 2021	Sat Aug 21 11:33:00 CEST 2021
P3 Historia	Sat Aug 21 11:33:00 CEST 2021	Sat Aug 21 12:00:00 CEST 2021
Nyheter från Ekot	Sat Aug 21 12:00:00 CEST 2021	Sat Aug 21 12:03:00 CEST 2021
P4 Extra	Sat Aug 21 12:03:00 CEST 2021	Sat Aug 21 12:30:00 CEST 2021
Dalanytt	Sat Aug 21 12:30:00 CEST 2021	Sat Aug 21 12:33:00 CEST 2021
P4 Extra	Sat Aug 21 12:33:00 CEST 2021	Sat Aug 21 13:00:00 CEST 2021

Figur 3: Centrala delen i användargränssnittet

Metoden *buildNorthPanel* skapar ett JPanel-objekt som placeras i det norra blocket av JFrame. JPanelen använder layoutmanagern Boxlayout och i panelen kommer det att lagras två komponenter en JComboBox samt en JLabel som användaren kan interagera med.

Metoden *buildComboBox* skapar ett JComboBox-objekt vars anvsar är att hålla i alla kanaler som laddas in och presentera dem för användaren i form av en lista som öppnas när man klickar på objektet med musenpekaren.

Metoden *setChannelLabel* skapar ett JLabel-objekt med en skriven text Select radio station:". Denna JLabel placeras bredvid den skapta JComboBox-objektet och ansvarar för att hänvisa användaren att det är via JComboBoxen som man väljer en kanal.

JPanelen som placerats i det norra blocket av JFrame tillsammans med JLable- och ett JComboBox-objekt kan åskådliggöras i figur 4.

Figur 4: Norra delen i användargränssnittet

Metoden *programInformation* skapar ett JOptionPane-objekt när användaren interagerat med tabellen, dvs när användaren har klickat på ett program som verkar intressant. Användaren presenteras då med ett fönster där man kan läsa information om det valda programmet. I fönstret som användaren presenteras kommer det även att finnas en bild för det valda programmet. Bilden konverteras från en URL sträng till ett Image objekt med hjälp av metoden *getRadioImage*.

Om det skulle vara så att det valda programmet ej har någon tillhörande bild så kommer den ej att presenteras och enbart text information visas för användaren. Fönstret som användaren kommer att presenteras åskådliggörs i figur 5.

Figur 5: Programinformation

Följande 11 metoder används för att hämta ett attribut:

getQuit returnerar en JMenuItem.

getUpdate returnerar en JMenuItem.

getmodel returnerar en DefaultTableModel.

getTable returnerar en JTable.

getLsm returnerar en ListSelectionModel.

getComboBox returnerar en JComboBox.

getChannelInfo returnerar en JTextArea.

3.4 Controller

Controller-klassen ansvarar för kommunikationen mellan **View** och **Model**. **Controller**-klassen representerar Controller i MVC-Modellen. I **Controller** initialiseras det ett fler tal actionlisteners till de objekt som användaren kan ineragera med i programmet. När användaren interagerar med ett objekt sker ett event på en actionlistener och beroende på event anropas en metod. I en metod används swingworkers *doInBackground*-metod. I *doInBackground*-metoden kan programmet arbeta med **Model** och när färdig avslutas tråden och swingworkers *done*-metod körs. I *done*-metoden kan programmet arbeta med det som ska presenteras för användaren i **View** och det sker på EDT.

Metoden *setUpChannels* hämtar alla kanaler via **model** och lägger dem i dropdown-menyn.

I metoden *createActionListeners* skapas alla actionlisteners som används i programmet.

Metoden *setUp* sätter upp hela programmet genom att skapa actionlisteners, hämta kanaler, visar GUI med hjälp av *showFrame* metoden. här startar också timern som ska uppdatera programmet varje timme.

I **Controller**-klassen används också en timer som skapas med hjälp av *updateTimer* metoden. Syftet med metoden är att anropa *updateProgram*-metoden en gång i timmen.

Metoden *showPrograms* anropas då ett event har inträffat i en actionlistener som är kopplad till Drop-down menyn som finns i norra blocket av JFrame i **View** eller när en uppdatering sker varje timme.

showPrograms's *doInBackground* metod kommer att hämta den valda kanalen från menyn och därefter hämta alla program som assoccieras med den kanalen. Kanalerna ligger i en lista som håller ProgramInfo-objekt.

I showPrograms's *done* metod kommer den lista som returnerats från *doInBackground* att hämtas och placeras i tabellen som skapats i det centrala blocket av JFrame i **View**. Användaren kommer då att se olika program som sänds på varje rad.

Metoden *quitProgram* anropas då ett event har inträffat i en actionlistener som är kopplad till "Quit"knappen från menyn i **View**.
Ingen Swingworker *doInBackground* tråd krävs i detta fall då programmet redan är på EDT och metoden behöver inte använda **Model**.

Metoden *programInfo* anropas då ett event har inträffat i en actionlistener som är

kopplad till tabellen från **View**.

I programInfo's *doInBackground* metod kontrolleras först vilken rad som valts i tabellen från **View**, kanalen och programmet som användaren klickat på sparas till två variabler. Därefter hämtar man ut den information som finns tillgänglig till programmet med hjälp av metoden *getProgramInfo* som tar in en kanal och ett program och returnerar ett ProgramInfo Objekt.

I programInfo's *done* metod hämtas ProgramInfo Objekt som returnerades från *doInBackground* och presenterar det för användaren i ett nytt fönster som dyker upp på skärmen.

Metoden *updateProgram* anropas på ett event har inträffat på en actionlistener som är kopplad till update knappen i menyn. Men metoden anropas också då timern från *updateTimer*-metoden har nått en timme.

Metodens syfte är att uppdatera alla kanaler som finns lagrade genom att skriva över alla gamla kanaler med de nya som hämtas med en metod från model *parser()*.

Uppdateringen av kanaler görs ifrån en *doInBackground* för att undvika att programmet fastnar"under tiden den hämtar nya kanaler.

I *done* tar man bort gammalt data och ersätter med det nya som man hämtat.

3.5 Model

Klasserna **Model**, **ChannelInfo**, **ChannelParsing**, **ProgramInfo** och **ProgramParsing** representerar tillsammans Model i MVC modellen. **Model** delen av programmet bär som ansvar att läsa av XML-filen från Sveriges Radio's API och plocka ut nödvändig information som kanaler, program, tider, bilder och programinformation.

3.5.1 ProgramInfo

Klassen **ProgramInfo** är en klass som skapar ett program-objekt. Här har Programnamn, beskrivning, start- och sluttid och en bild till programmet sparats som ett eget objekt.

3.5.2 ChannelInfo

Klassen **ChannelInfo** har en liknande uppgift som **ProgramInfo**, men istället för program skapar man ett kanal-objekt där nödvändig information om en kanal sparas. I det här programmet lagras kanalnamnet och en lista med alla program som tillhör den kanalen i ett kanal-objekt.

3.5.3 ChannelParsing

Klassen **ChannelParsing** bär ansvaret att läsa in alla kanaler och dess program. Programmen som läses in måste vara program som ligger inom +/- 12 timmar från den nuvarande tiden.

Metoden *parseChannel* läser in alla kanaler från sveries radio och skapar kanal-objekt i en lista som sen returneras.

Metoden *checkPrograms* används i *parseChannel* och hämtar programmen för en kanal här kontrolleras det också om programmet sändes igår eller om det sänds imorgon från nuvarande tid.

3.5.4 ProgramParsing

Klassen **ProgramParsing** bär ansvaret att läsa in alla program som finns tillgänglig i Sveriges radio från en given kanal.

Metoden *parseProgram* läser in programmen och skapar ProgramInfo-objekt för varje program.

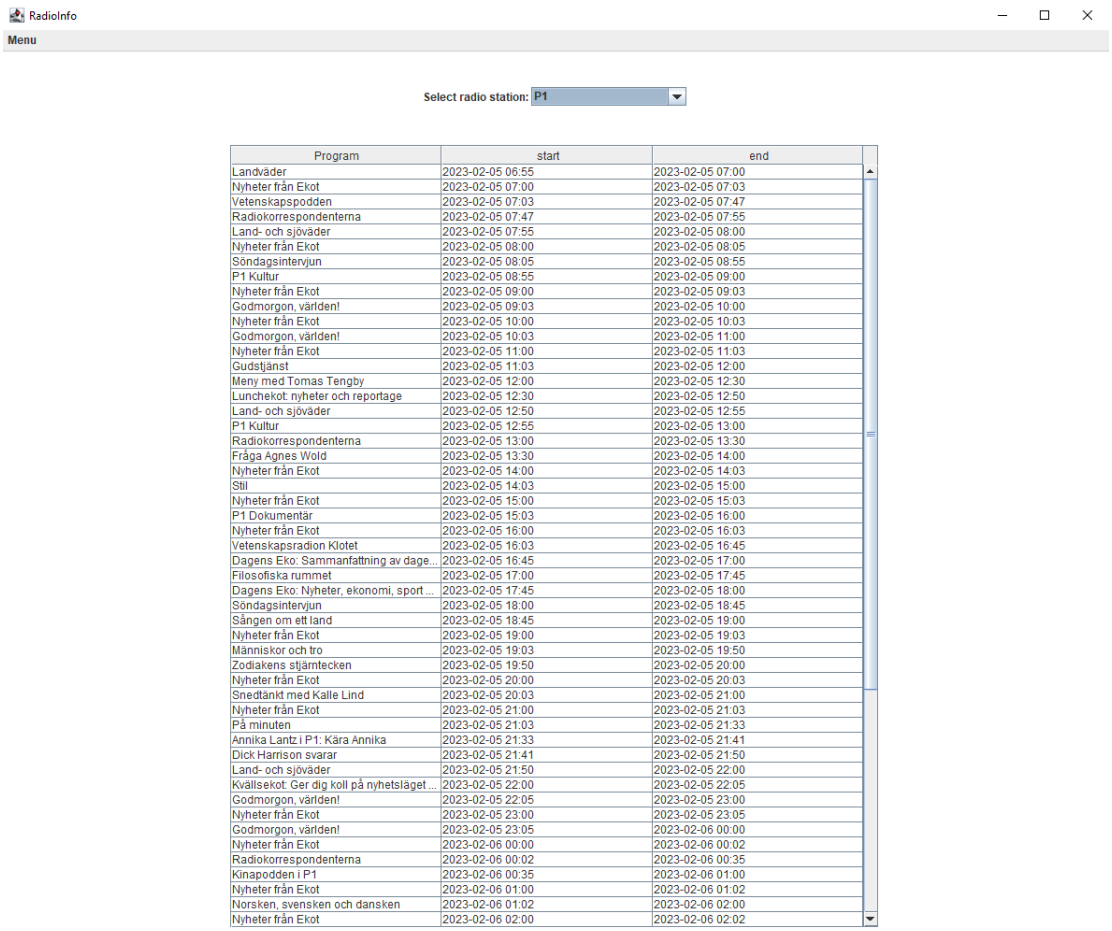
3.5.5 Model

I klassen **Model** finns de metoder som Controllern använder sig av. Här finns metoder som *getChannels* som hämtar alla namn på kanalerna. *getPrograms* som hämtar alla program för en given kanal. *getProgramInfo* hämtar ett specifikt programInfo-objekt. *parser* metoden hämtar och ersätter gamla kanaler från Sverigesradio-api.

Alla metoder i model använder keyword synchronized för att undvika racecondition när de anropas från Controllern.

4 Användarhandledning

Programmet RadioInfo går att starta på två sätt. Antingen startar man JAR-filen eller genom kompliera klassen **RadioInfo**. När programmet startar kommer användaren att mötas av följande vy, se figur 6.

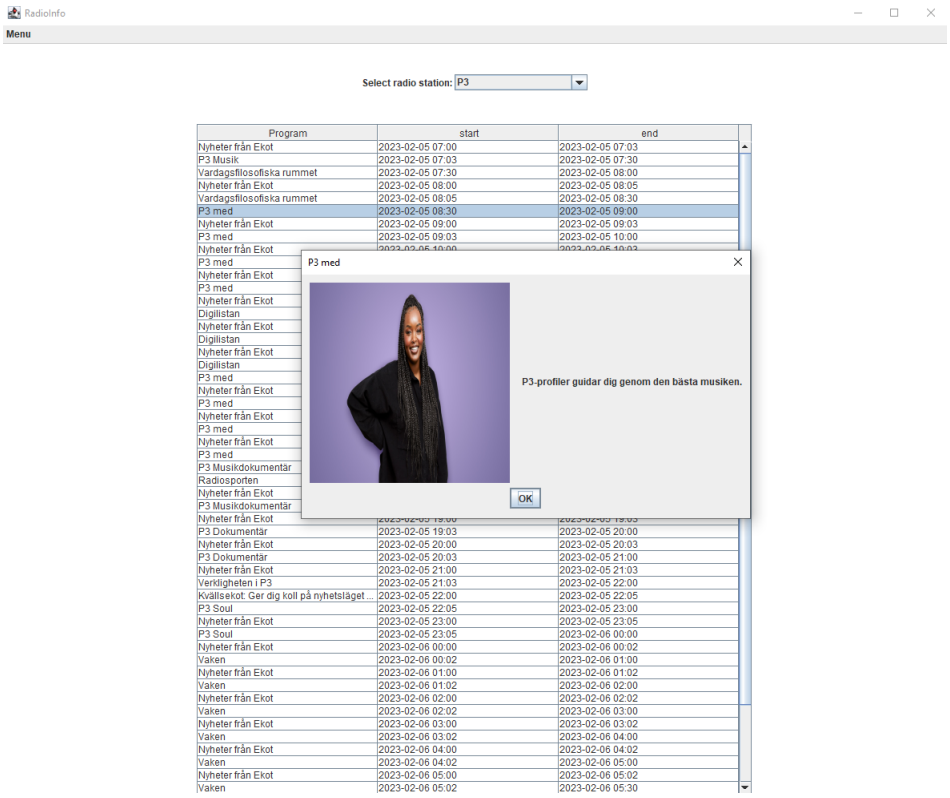


Figur 6: Vy över start av program

I start vyn möts användaren av ett fönster som är uppdelat i två sektioner. I den övre sektionen kommer användaren att kunna välja en specifik kanal via en drop-down meny. I mitten av programmet ser användaren en tabell där schemat för programmen kommer att fyllas i när en kanal har blivit vald.

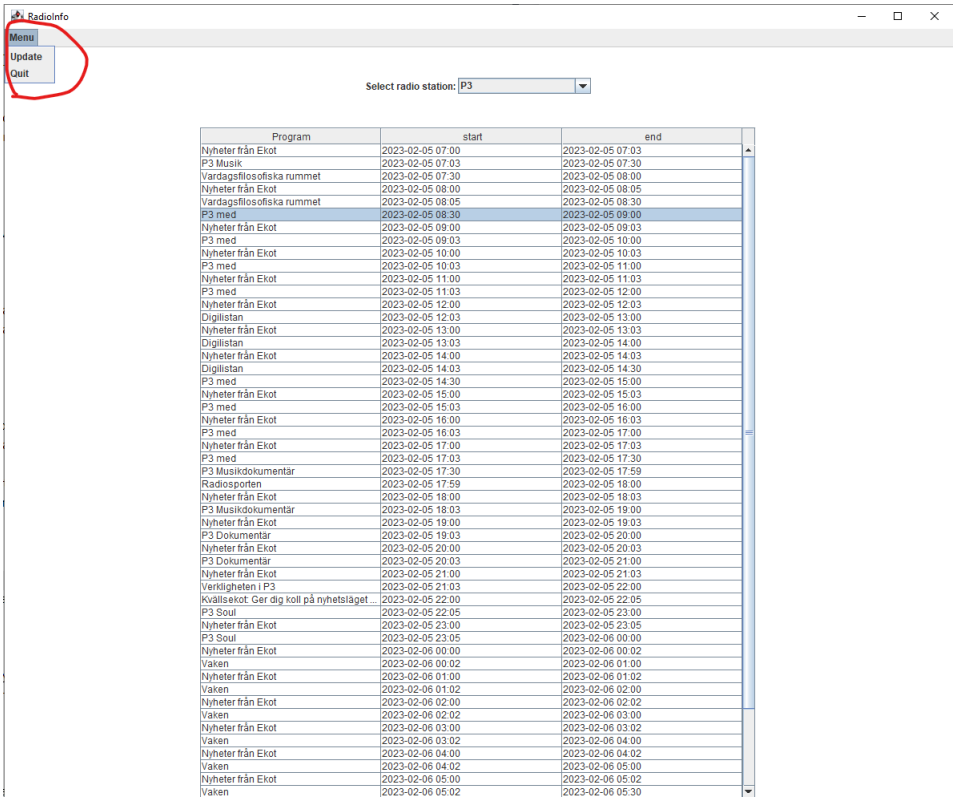
När en kanal är vald som t.ex. P3. I tabellen kan användaren se vilka program som sänds, starttid och sluttid. Tiderna speglar 12 timmar bakåt och 12 timmar fram från nuvarande tid.

När användaren klickar på ett program kommer ett fönster att dyka på skärmen som beskriver kort med text och bild vad programavsnittet handlar om vilket åskådliggörs i figur 7.



Figur 7: Vy efter användare klickat på ett program

Om användaren klickar på menyn som finns tillgänglig längst upp i fönstret av programmet möts man av olika alternativ att interagera med vilket åskådliggörs i figur 8.



Figur 8: Vy efter användare klickat på menyn

Update-knappen kommer att låta användaren förnya földet i tabellen.

Quit-knappen avslutar programmet.