

UMEÅ UNIVERSITET
Institutionen för Datavetenskap

10 januari 2019

5DV135: Applikationsutveckling i Java
Hösten 2018, 7.5p

Rapport för Laboration 2 - RadioInfo

Name Buster Hultgren Wörn
E-mail dv17bhn@cs.umu.se

Kursansvarig
Johan Eliasson
Handledare

Klas af Geijerstam, Jakob Lindqvist, Elias Åström, William Viktorsson

Innehåll

1	Introduktion	1
1.1	Filer och körning	1
1.2	Användarhandledning	2
2	Systembeskrivning	5
2.1	Paket och klasser	5
2.2	Model-View-Controller	7
2.3	Flödesschema och trådar	7
2.4	Trådsäkerhet	9
3	Diskussion	10

1 Introduktion

I denna laboration ska vi bygga programmet RadioInfo. Detta är ett program som hämtar information från Sveriges Radios API och visa dess kanalers tablå. Programmet ska implementeras med Model-View-Controller (MVC) stil, och modellen ska köras på andra trådar än Event Dispatch Thread (EDT) tråden. Programmet ska vara trådsäkert.

1.1 Filer och körning

Programmet använder inga resurser, är beroende på en .jar fil (jfxrt.jar). Källkoden av 16 olika källfiler, där RadioInfo är main-filen:

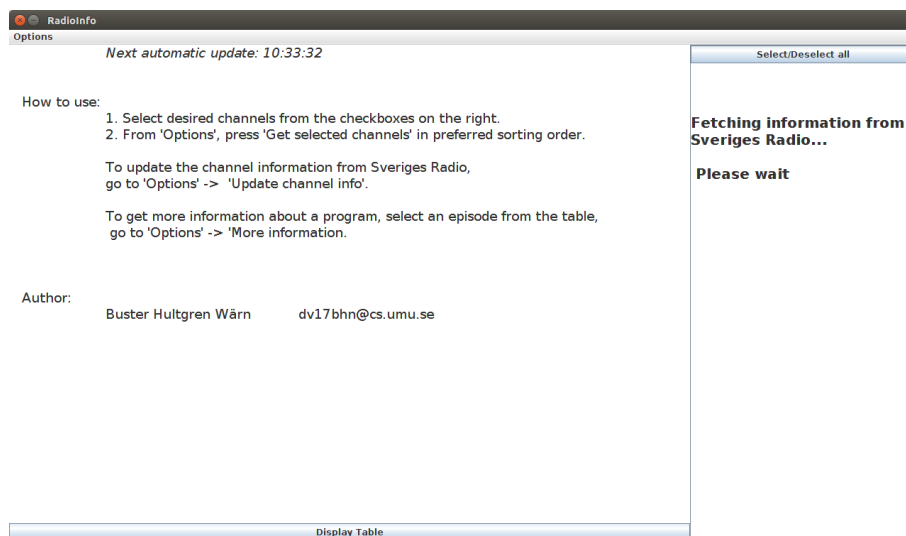
1. RadioInfo.java
2. Controller
3. UpdateRadioInfoWorker
4. GetRadioInfoWorker
5. GetDescriptionWorker
6. Winodw
7. ChannelTableModel
8. EpisodePanel
9. MultipleCheckBox
10. Model
11. APIChannelParser
12. Channel
13. ChannelEpisode
14. ThreadSafeList
15. CopySafe
16. EpisodeSorter

En .jar fil följer även med - RadioInfo.jar - och det är via denna som programmet startas. Från samma mapp där .jar filen ligger. RadioInfo tar inte emot några in-parametrar. avnänd följande kommando i en terminal för att starta programmet:

\$ java -jar RadioInfo.jar

1.2 Användarhandledning

När programmet startar möts användaren av vyn i figur 1. Fönstret är uppdelat i två olika delar, till vänster är huvudsidan - den del som har programmets egenliga innehåll (hjälp sida, tabellsida eller en mer detaljerad sida om ett viss avsnitt). Till höger finns en selektionssida med olika JCheckBox:ar. Där väljs vilka tablåer som ska visas i tabellen. I figur 1 så hämtas har informationen från Sveriges Radio inte hämtats en och ett meddelande visas istället.



Figur 1: Första panelen (help) - information har inte laddats

När informationen väl har visats så dycker alla tablåer som modellen hittar hos Sveriges radio upp, visat i igr 2.. Här kan användaren själv välja alla kanaler som ska visas i tabellen. Efter att ha valt en eller flera, så hämtas kanalerna genom Öptionsmenyn. Där finns ett antal val:

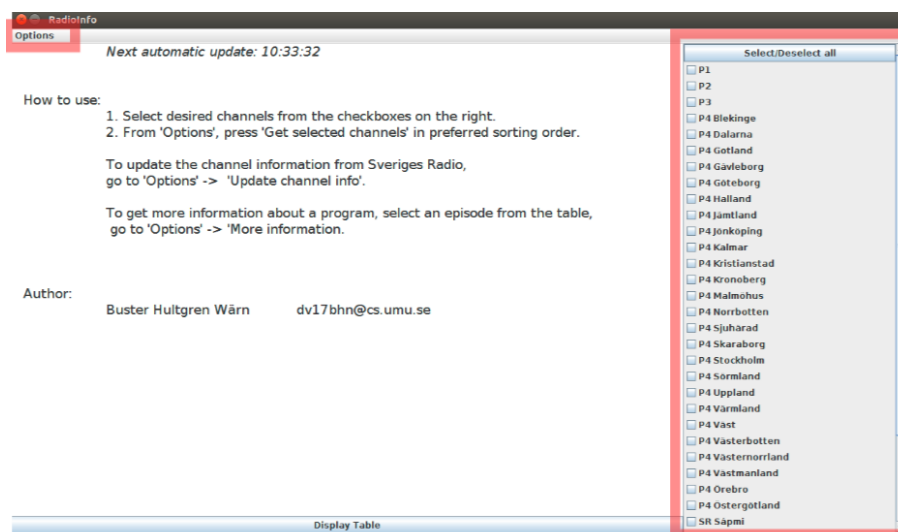
- Get selected channels - hämtar
 - Sort by channels ascending
 - Sort by channels descending
 - Sort by episodes ascending
 - Sort by episodes descending
 - Sort by date/time ascending
 - Sort by date/time descending
- Update channel information
- More information
- Help

"Get selected channels"hämtar de valda kanalerna. Därefter finns sex olika sätt att sortera dessa på, som visas i en undermeny.

"Update channel information"kallar på en uppdatering ifrån modellen. Då hämtas ny information ifrån Sveriges Radio.

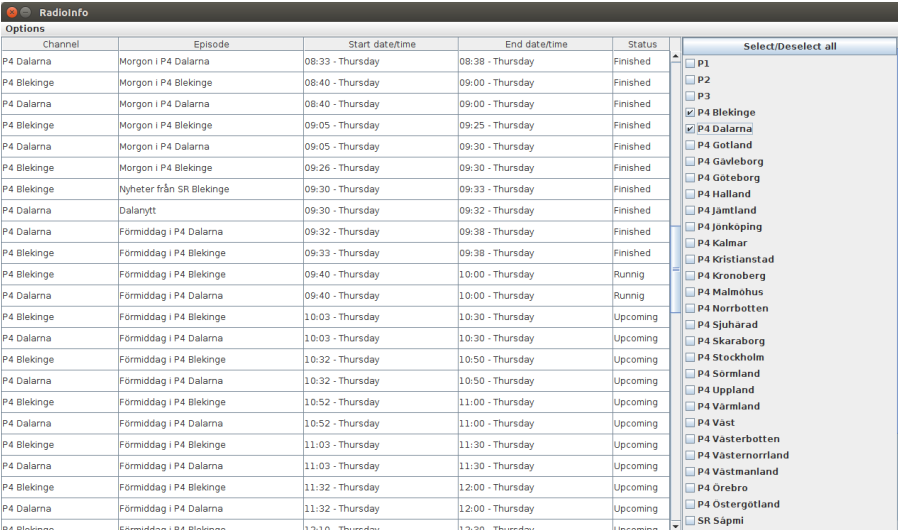
"More information"kan tryckas på ifall ett specifikt avsnitt i tabellen trycks på. Då byts huvudsidan till att visa mer information kring just det avsnittet.

"Helptar"en till nuvarande huvudsida, som visar lite information kring hur programmet används samt nästa automatiska uppdateringstid.



Figur 2: Första panelen (help) - information har laddats

Användaren kan välja mellan att hämta en eller flera kanaler. I figur 3 så har P4 Blekinge och P4 Dalarna hämtats.



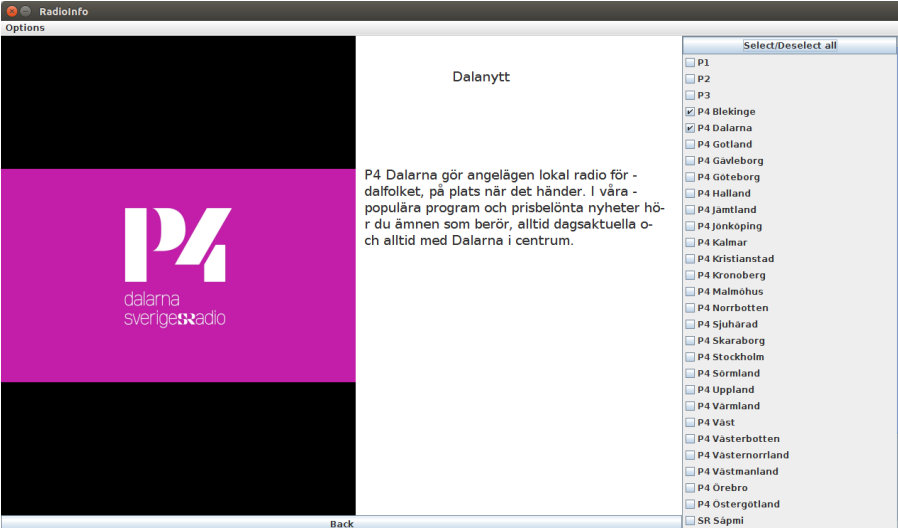
Channel	Episode	Start date/time	End date/time	Status
P4 Dalarna	Morgon i P4 Dalarna	08:33 - Thursday	08:38 - Thursday	Finished
P4 Blekinge	Morgon i P4 Blekinge	08:40 - Thursday	09:00 - Thursday	Finished
P4 Dalarna	Morgon i P4 Dalarna	08:40 - Thursday	09:00 - Thursday	Finished
P4 Blekinge	Morgon i P4 Blekinge	09:05 - Thursday	09:25 - Thursday	Finished
P4 Dalarna	Morgon i P4 Dalarna	09:05 - Thursday	09:30 - Thursday	Finished
P4 Blekinge	Morgon i P4 Blekinge	09:26 - Thursday	09:30 - Thursday	Finished
P4 Blekinge	Nyheter från SR Blekinge	09:30 - Thursday	09:33 - Thursday	Finished
P4 Dalarna	Dalanytt	09:30 - Thursday	09:32 - Thursday	Finished
P4 Dalarna	Förmiddag i P4 Dalarna	09:32 - Thursday	09:38 - Thursday	Finished
P4 Blekinge	Förmiddag i P4 Blekinge	09:33 - Thursday	09:38 - Thursday	Finished
P4 Blekinge	Förmiddag i P4 Blekinge	09:40 - Thursday	10:00 - Thursday	Runnig
P4 Dalarna	Förmiddag i P4 Dalarna	09:40 - Thursday	10:00 - Thursday	Runnig
P4 Blekinge	Förmiddag i P4 Blekinge	10:03 - Thursday	10:30 - Thursday	Upcoming
P4 Dalarna	Förmiddag i P4 Dalarna	10:03 - Thursday	10:30 - Thursday	Upcoming
P4 Blekinge	Förmiddag i P4 Blekinge	10:32 - Thursday	10:50 - Thursday	Upcoming
P4 Dalarna	Förmiddag i P4 Dalarna	10:32 - Thursday	10:50 - Thursday	Upcoming
P4 Blekinge	Förmiddag i P4 Blekinge	10:52 - Thursday	11:00 - Thursday	Upcoming
P4 Dalarna	Förmiddag i P4 Dalarna	10:52 - Thursday	11:00 - Thursday	Upcoming
P4 Blekinge	Förmiddag i P4 Blekinge	11:03 - Thursday	11:30 - Thursday	Upcoming
P4 Dalarna	Förmiddag i P4 Dalarna	11:03 - Thursday	11:30 - Thursday	Upcoming
P4 Blekinge	Förmiddag i P4 Blekinge	11:32 - Thursday	12:00 - Thursday	Upcoming
P4 Dalarna	Förmiddag i P4 Dalarna	11:32 - Thursday	12:00 - Thursday	Upcoming
P4 Blekinge	Förmiddag i P4 Blekinge	12:10 - Thursday	12:30 - Thursday	Upcoming

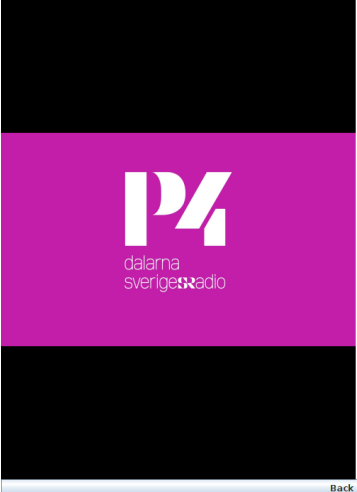
Select/Deselect all

- ☐ P1
- ☐ P2
- ☐ P3
- ☒ P4 Blekinge
- ☒ P4 Dalarna
- ☐ P4 Gotland
- ☐ P4 Gävleborg
- ☐ P4 Göteborg
- ☐ P4 Halland
- ☐ P4 Jämtland
- ☐ P4 Jönköping
- ☐ P4 Kalmar
- ☐ P4 Kristianstad
- ☐ P4 Kronoberg
- ☐ P4 Malmöhus
- ☐ P4 Norrbotten
- ☐ P4 Sjuhärad
- ☐ P4 Skaraborg
- ☐ P4 Stockholm
- ☐ P4 Sörmland
- ☐ P4 Uppland
- ☐ P4 Värmland
- ☐ P4 Väst
- ☐ P4 Vasterbotten
- ☐ P4 Vasternorrland
- ☐ P4 Vastmanland
- ☐ P4 Örebro
- ☐ P4 Östergötland
- ☐ SR Sjöpmi

Figur 3: Tabell med episoder

Om användaren vill se mer information kring ett visst avsnitt så ändras till så ändras huvudsida till panelen som visas i figur 4. Här visas en bild, avsnittets titel och en beskrivning kring avsnittet.





Dalanytt

P4 Dalarna gör angelägen lokal radio för - dalfolket, på plats när det händer. I våra - populära program och prisbelönte nyheter hör du ämnen som berör, alltid dagsaktuella och alltid med Dalarna i centrum.

Select/Deselect all

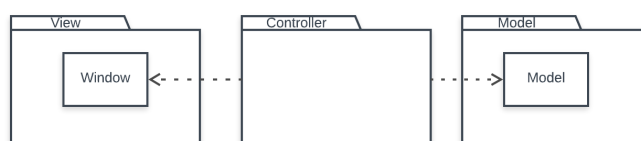
- ☐ P1
- ☐ P2
- ☐ P3
- ☒ P4 Blekinge
- ☒ P4 Dalarna
- ☐ P4 Gotland
- ☐ P4 Gävleborg
- ☐ P4 Göteborg
- ☐ P4 Halland
- ☐ P4 Jämtland
- ☐ P4 Jönköping
- ☐ P4 Kalmar
- ☐ P4 Kristianstad
- ☐ P4 Kronoberg
- ☐ P4 Malmöhus
- ☐ P4 Norrbotten
- ☐ P4 Sjuhärad
- ☐ P4 Skaraborg
- ☐ P4 Stockholm
- ☐ P4 Sörmland
- ☐ P4 Uppland
- ☐ P4 Värmland
- ☐ P4 Väst
- ☐ P4 Vasterbotten
- ☐ P4 Vasternorrland
- ☐ P4 Vastmanland
- ☐ P4 Örebro
- ☐ P4 Östergötland
- ☐ SR Sjöpmi

Figur 4: Mer detaljerad information för ett avsnitt

2 Systembeskrivning

2.1 Paket och klasser

RadioInfo är uppdelad i tre olika paket: Model, View och Controller. Anledningen till denna struktur beskrivs mer i kapitel 2.2. Både modellen och vyn har varsin klass som ger en gränsyta till utanför paketet. Alla klasser i Kontrollern använder sig av denna gränsyta. Figur 5 är inte ett fullständigt paket-diagram, men den beskriver ändå relationen mellan paketen. Paketet Controller är beroende av både Model och Wiew, vars gränsytor som klasser är Model och Window.



Figur 5: (Ofullständigt) paket diagram av RadioInfo.

Controller paketet består utav fyra klasser, varav 3 är förläggningar (extensions) av SwingWorkers vilka styrs av Controller (klassen). Controller sköter det huvudsakliga flödet, och skapar upp en instans av någon utav dessa SwingWorkers då en specific uppgift behöver utföras.

1. Controller
2. UpdateRadioInfoWorker
3. GetRadioInfoWorker
4. GetDescriptionWorker

View paketet består också av fyra klasser, där tre är kompositioner av paketets gränsyta Window. ChannelTableModel är en förlängning av DefaultTableModel, vilket är ett sätt att representera data i en tabell. ChanelTableModel används för att representera de olika avsnitt som laddas in av Sveriges Radio. EpisodePanel är panelen som ansvarar över att visa upp tydligare information över ett avsnitt. MultipleCheckBox visar upp alla kanaler som finns, och användaren kan bocka i de kanaler som de vill visa.

1. Winodw
2. ChannelTableModel
3. EpisodePanel
4. MultipleCheckBox

Model är det paket med mest klasser. APICchannelParser är klassen som laddar ner och går igenom Sveriges Radio API. I listan ThreadSafeList lägger APICchannelParser in alla kanaler som ett

Channel objekt, och varje Channel innehåller en till ThreadSafeList lista med avsnitt, som finns under den kanalen. Avsnitt representeras via ChannelEpisode klassen.

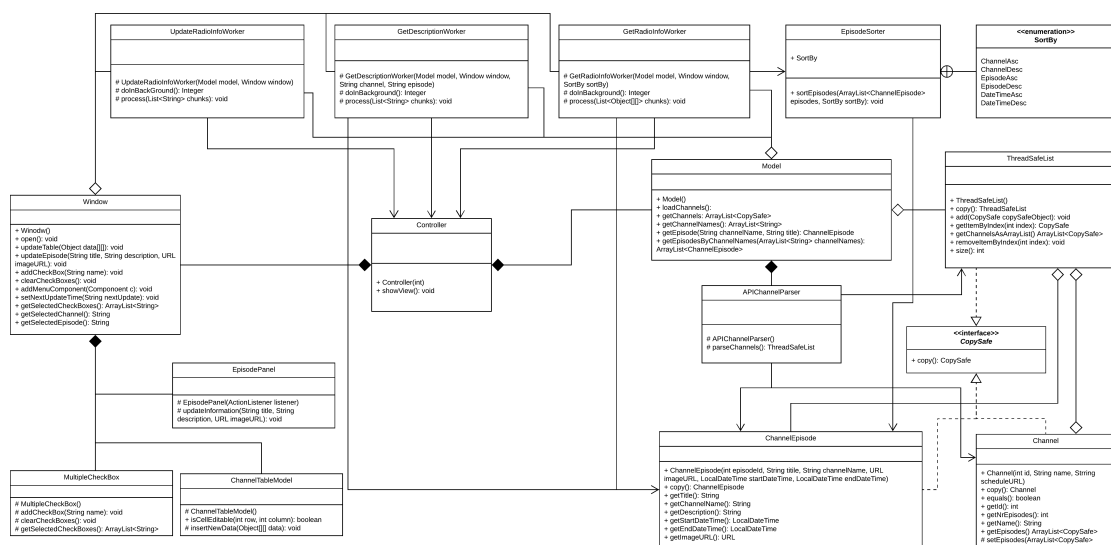
CopyFace är ett interface, som har en public metod copy() i sig. Copy() ska enbart göra en deepcopy av sig själv. ThreadSafeList, Channel och ChannelEpisode implementerar alla CopySafe, vilket betyder att alla tre av dessa klasser kan kopieras.

EpisodeSorter är en klass vars enda ansvar är att ta in en ArrayList med ChannelEpisode objekt och sortera dessa. Med en privat konstruktor undviker EpisodeSorter att det finns instanser av sig; endast dess publika metoder ska användas. I EpisodeSorter ligger en nästad enumeration SortBy, som är ett inställning på hur listan med episoder ska sorteras.

Model klassen är som tidigare beskrivit gränsytan till hela Model paketet. Endast EpisodeSorter kan nå utanför Model klassen.

1. Model
2. APIChannelParser
3. Channel
4. ChannelEpisode
5. ThreadSafeList
6. CopySafe
7. EpisodeSorter

I figur 6 visas UML-diagrammet för RadioInfos alla klasser. Det är dock lite svårt att få med den hela detaljerade bilden som ett PDF, därför finns även bilden uppladdad på [git](#).



Figur 6: Klass UML över RadioInfo.

2.2 Model-View-Controller

Programmet följer Model-View-Controller (MVC) design mönstret, och är uppdelad i paket utifrån detta. En huvudsaklig klass finns i varje paket som visar den öppna gränsytan som är tänkt att användas.

Model-delen (modellen) är den del som sköter hand om programmets huvudsakliga logik och ttnygreberäkningar. I detta fall så handlar sköts Sveriges Radio API härifrån. Först så hämtas det som ett XML format och konverteras till mer lätthanterliga datatyper.

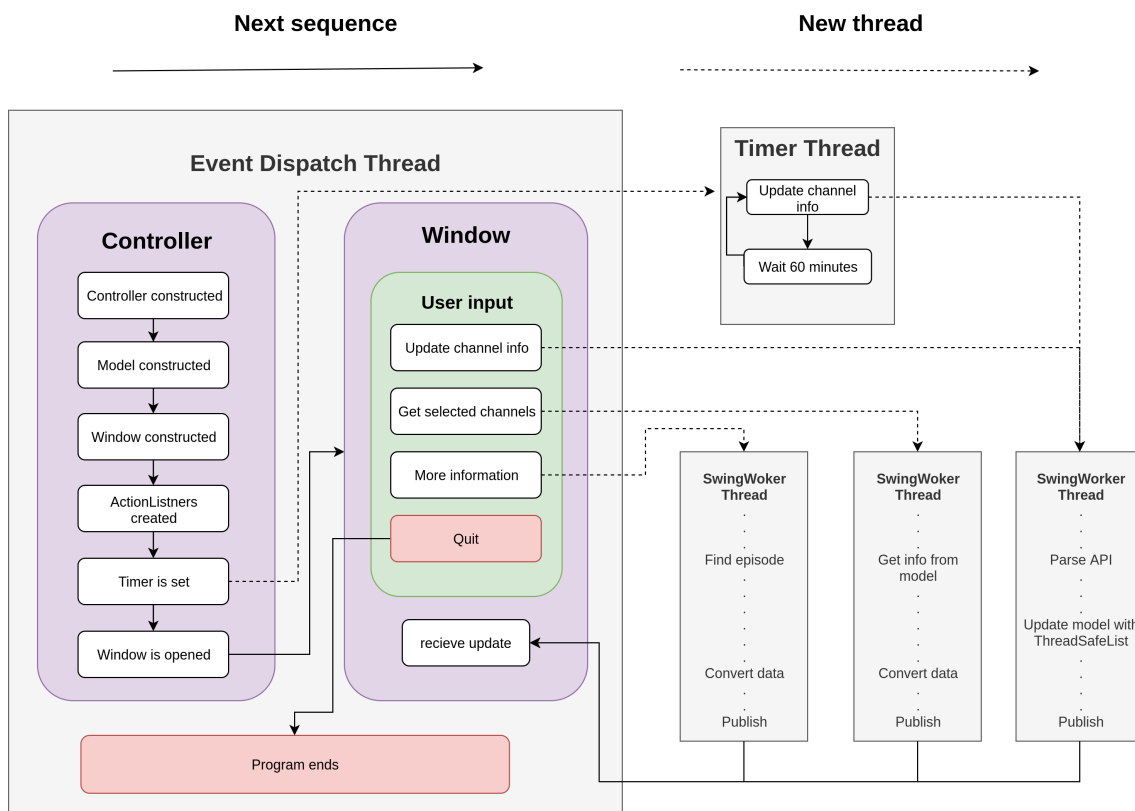
View-delen (vyn) är det grafiska gränssnittet - även känt som Graphical User Interface (GUI) - till programmet. Här ger användaren input till programmet, för att bestämma vad som ska ske. Förutom några små logiska flöden, så bygger koden upp ett grafiskt gränssnitt till användaren såsom knappar, fönster, text och liknande. I gränssnittet läggs lyssnare in i diverse knappar och menyer för att få information ifrån vad användaren gör.

Controller-delen (kontrollern) är den del som får modellen och vyn att kunna arbeta tillsammans. kontrollern har instanser av både vyn och modellen, och sköter hand om det huvudsakliga flödet, till exempel om en knapp trycks eller när modellen har hämtat data. Kontrollern bär också ansvar över att skapa trådar.

Implementationen av Model-View-Controller leder till att modellen och vyn inte är beroende av varandra. Vyn skulle kunna bytas ut mot en annan utan att modellen påverkas och vice-versa. Kontrollern är byggd för att se till att modellen och vyn kan kommunicera, och är därför direkt beroende av båda. Ändras gränsytan till modellen eller vyn så kommer kontrollern att behöva anpassas.

2.3 Flödesschema och trådar

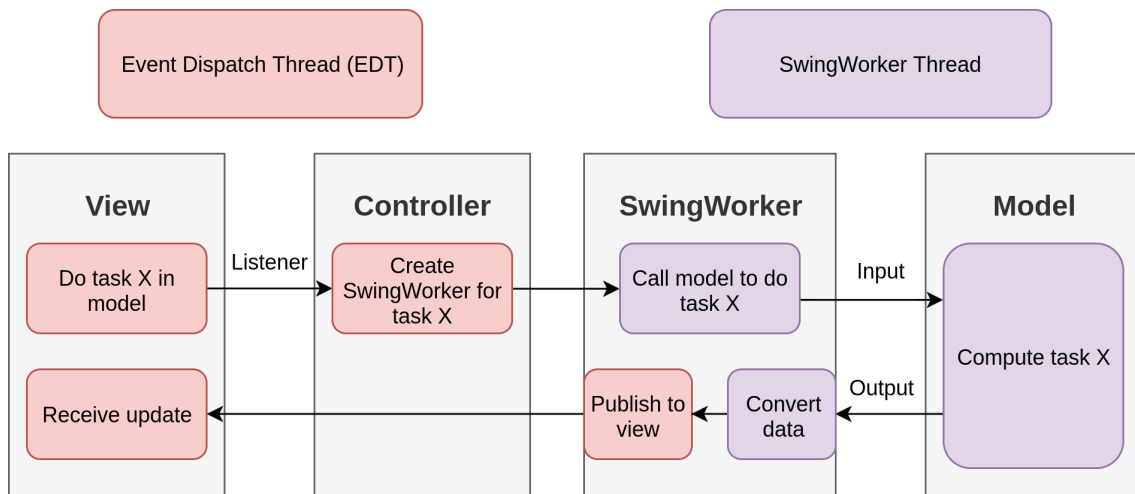
RadioInfo kör kontinuerligt, och har inget fast flödesschema, utan varierar utifrån vad användaren väljer att göra. Den har dock ett par fasta punkter. Från mainfilen skapas kontrollern, vilket skapar instanser av modellen och vyn. Olika lyssnare skapas därefter och sätts in i vyn. Därefter skapas den första tråden, en timer, och efter detta så öppnas fönstret. Figur 7 visar översiktligt hur programmet kör.



Figur 7: Flödesschema för RadioInfo

Timerns uppgift är att kalla på en uppdatering av modellens information. Den gör detta genom att skapa en ny SwingWorker tråd, som sköter hand om uppdateringen. När timern har kallat på uppdateringen, så väntar/sover tråden som timern kör på i 60 minuter. När tråden har sovit/väntat i denna tid så kallar den på en ny uppdatering. Detta upprepas under hela programmets körtid.

När fönstret öppnas så finns det fyra större saker som användaren kan göra. Användaren kan kalla på en uppdatering, hämta valda kanaler, hämta mer detaljerad information kring en viss kanal och avsluta programmet. De tre första valen är alla kallelser på modellen. Lyssnare ifrån kontrollern plockar upp vilket av dessa val som görs och skapar upp en SwingWorker med uppgift att utföra det val som gjordes. SwingWorkern kör modellen på sin egen tråd som hämtar den data den behöver och ger tillbaka denna till SwingWorkern. Några sista konverteringar av data görs på tråden för att sedan publiceras till vyn. Figur 8 visar hur detta flöde sker.



Figur 8: Flödet mellan View, Controller, SwingWorker och Model. Rött representerar EDT:n, medans lila representerar SwingWorkerns tråd.

2.4 Trådsäkerhet

Då olika trådar används ökar effektiviteten hos ett program, men det blir även svårare att utveckla. Då flera trådar delar på någon typ av resurs så måste det finnas någon implementation som ser till att trådarna inte arbetar med samma resurs samtidigt för att undvika allt ifrån små logiska problem som är svåra att debugga till runtime errors.

I detta program körs sammanlagt sex olika trådar - Main tråden, Timer tråden, EDT:n och sen tre olika SwingWorker trådar. Det är SwingWorker trådarna som använder delade resurser ifrån modellen, och det är här olika saker kan gå fel om flera trådar kör samtidigt.

Den första SwingTråden vill uppdatera information ifrån Sveriges Radio och de andra två vill hämta denna information, därför använder trådarna en implementerad klass ThreadSafeList. ThreadSafeList är, som namnet tyder på, en trådsäker lista. Den använder sig av en ArrayList för att lagra information om ett objekt. Varje metod i ThreadSafeList är synchronized så att två olika trådar inte kan kalla på dessa metoder samtidigt. Om ett objekt hämtas i ThreadSafeList så hämtas inte det faktiska objektet, utan en deepcopy görs.

ThreadSafeList lagrar endast objekt som implementerar CopySafe. CopySafe är ett interface med en metod - copy(). Allt copy() ska göra är att göra en deepcopy på sig själv. På detta sätt kan ThreadSafeList skicka kopior av sina objekt och inte referenser till samma objekt. Och ThreadSafeList skickade faktiska referenser så skulle objekten i ThreadSafeList kunna manipuleras, via referenserna, utanför ThreadSafeList och klassen skulle därmed inte vara trådsäker.

Så alla resurser som delas mellan de olika trådarna ligger i ThreadSafeList, som är trådsäker. Det finns dock ett annat problem, och det är att EDT tråden inte är trådsäker. Risken finns att flera SwingWorker trådar försöker uppdatera vyn samtidigt. Detta undviks, genom metoderna process() och publish() i SwingWorker. De separata SwingWorker trådarna kallar på publish() som tar emot en viss indata. Denna indata skickas till process() som kör på EDT:n. Detta är implementationer i SwingWorker, och de är trådsäkra. All faktisk uppdatering av vyn sker i process(), vilket görs

på EDT tråden.

3 Diskussion

Av de tidigare kurser jag har läst inom datavetenskap när det kommer till programmering så de teoretiska och tekniska delarna de svåraste. Att bygga ett huffmankomprimeringsprogram i kursen DV2, eller att bygga en terminal i Systemnära programmerings kursen var både väldigt tekniska, mänd det behövdes inte allt för mycket kod.

I detta projekt så är mindre individuella delar, eller algoritmer aldrig en utmaning. Det finns två saker som för mig har varit extra utmanande under detta arbete.

Att hela tiden arbeta med olika bibliotek som jag tidigare inte stött på är svårt. Nästan varje gång jag behövt implementera en ny funktionalitet så har det tagit stopp på grund av någonting med en gränsyta som jag inte förstått mig på. Java är väl dokumenterat på internet, men det brukar ändå ta en längre tid av läsande innan jag får någon typ av uppfattning kring hur gränsytan bör användas, och jag blir aldrig någon expert på detta.

Den stora omfattningen av projektet är också utmanande. För att lyckas få ihop sitt program så att det blir så generellt som möjligt, samtidigt som det följer MVC modellen krävs en hel del planering innan. Även med denna planering missade jag mycket, som gav mig problem längre fram i implementationen.

Jag, och många runt om mig, tycker att denna kurs är lite hård och att arbetena är lite för stora. Det krävs mycket tid för att klara dem och ett starkt argument kan göras till att kursen går mer än på halvfart. Detta har dock sin fördel, jag har lärt mig väldigt mycket av detta projekt. Precis när jag börjar tro att jag är duktig på programmering så kommer ett sånt här projekt som visar motsatsen. Och efteråt så blir jag mycket bättre. Detta är jag extremt nöjd med, och efteråt kan jag vara säker att jag fick lära mig mycket.