Application development in Java – Assignment 2

Threads and XML - RadioInfo
Version 1.0

Sofia Leksell
sole0037@ad.umu.se, id20sll@cs.umu.se

2022-01-13

Table of Contents

1 Introduction

This assignment is covering the ability to use threads and parse an XML document. The program that is created should be implemented with regards of thread safety. Great emphasis is also placed on design pattern. The MVC(Model-view-controller) design pattern can be seen in the classes that have been constructed. All these components is needed to create the program RadioInfo which is letting the user to see the Swedish Radio's(SR) channels' program schedules. This information is presented by the implementation of a GUI and use of SwingWorker.

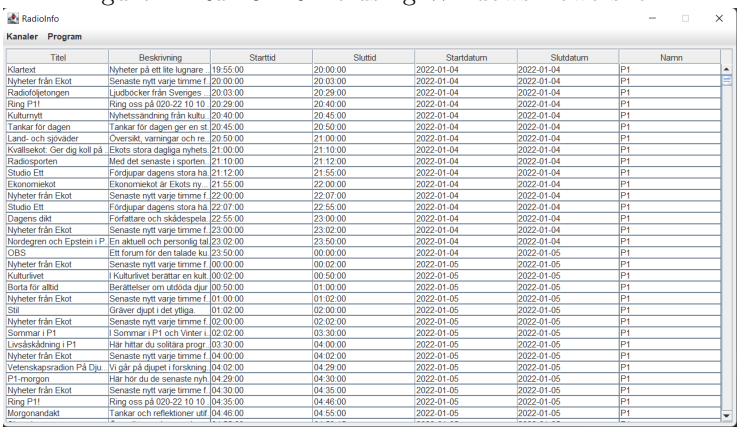
2 User manual

When the all the classes were compiled and ready, the classes were packaged into a JAR file. The JAR file was built through the built in tool in the IDE IntelliJ. The program RadioInfo is invoked by the command line

```
java -jar RadioInfo.jar
```

Windows Powershell was used as a commando line interface. When the program is invoked, a window pops up where the user can see an overview of all channels' program schedule. The figure beneath shows how the program looks when started by the command line.

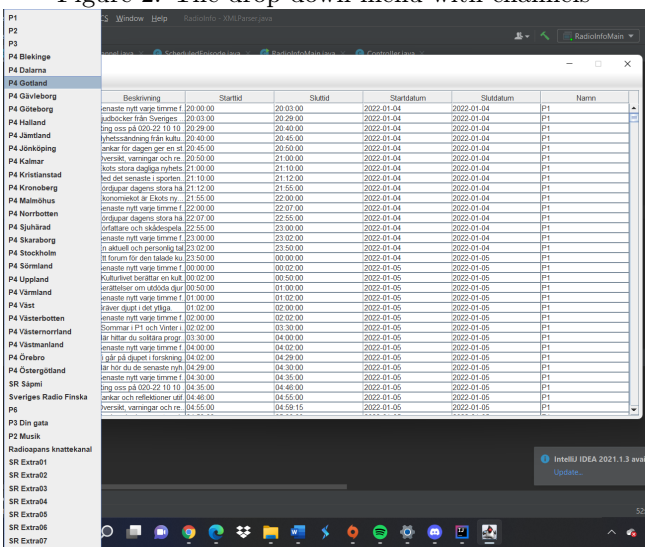
Figure 1: Run JAR file using Windows Powershell



Titel	Beskrivning	Starttid	Sluttid	Startdatum	Slutdatum	Namn
Klartext	Nyheter på ett lite lugnare	19:55:00	20:00:00	2022-01-04	2022-01-04	P1
Nyheter från Ekot	Senaste nytt varje timme f	20:00:00	20:03:00	2022-01-04	2022-01-04	P1
Radioföljetongen	Ljudböcker från Sveriges	20:03:00	20:29:00	2022-01-04	2022-01-04	P1
Ring P1!	Ring oss på 020-22 10 10	20:29:00	20:40:00	2022-01-04	2022-01-04	P1
Kulturnytt	Nyhetssändning från kultu	20:40:00	20:45:00	2022-01-04	2022-01-04	P1
Tankar för dagen	Tankar för dagen ger en st	20:45:00	20:50:00	2022-01-04	2022-01-04	P1
Land- och sjöväder	Översikt, varningar och re	20:50:00	21:00:00	2022-01-04	2022-01-04	P1
Kvällsekot. Ger dig koll på	Ekots stora dagliga nyhets	21:00:00	21:10:00	2022-01-04	2022-01-04	P1
Radiosporten	Med det senaste i sporten	21:10:00	21:12:00	2022-01-04	2022-01-04	P1
Studio Ett	Fördjupar dagens stora hä	21:12:00	21:55:00	2022-01-04	2022-01-04	P1
Ekonomiekot	Ekonomiekot är Ekots ny	21:55:00	22:00:00	2022-01-04	2022-01-04	P1
Nyheter från Ekot	Senaste nytt varje timme f	22:00:00	22:07:00	2022-01-04	2022-01-04	P1
Studio Ett	Fördjupar dagens stora hä	22:07:00	22:55:00	2022-01-04	2022-01-04	P1
Dagens dikt	Författare och skådespela	22:55:00	23:00:00	2022-01-04	2022-01-04	P1
Nyheter från Ekot	Senaste nytt varje timme f	23:00:00	23:02:00	2022-01-04	2022-01-04	P1
Nordegren och Epstein i P	En aktuell och personlig tal	23:02:00	23:50:00	2022-01-04	2022-01-04	P1
OBS	Ett forum för den talade ku	23:50:00	00:00:00	2022-01-04	2022-01-05	P1
Nyheter från Ekot	Senaste nytt varje timme f	00:00:00	00:02:00	2022-01-05	2022-01-05	P1
Kulturlivet	I Kulturlivet berättar en kult	00:02:00	00:50:00	2022-01-05	2022-01-05	P1
Borta för alltid	Berättelser om utdöda djur	00:50:00	01:00:00	2022-01-05	2022-01-05	P1
Nyheter från Ekot	Senaste nytt varje timme f	01:00:00	01:02:00	2022-01-05	2022-01-05	P1
Stil	Gräver djupt i det ytliga	01:02:00	02:00:00	2022-01-05	2022-01-05	P1
Nyheter från Ekot	Senaste nytt varje timme f	02:00:00	02:02:00	2022-01-05	2022-01-05	P1
Sommar i P1	I Sommar i P1 och Vinter i	02:02:00	03:30:00	2022-01-05	2022-01-05	P1
Livsåskådning i P1	Här hittar du solitära progr	03:30:00	04:00:00	2022-01-05	2022-01-05	P1
Nyheter från Ekot	Senaste nytt varje timme f	04:00:00	04:02:00	2022-01-05	2022-01-05	P1
Vetenskapsradion På Dju	Vi går på djupet i forskning	04:02:00	04:29:00	2022-01-05	2022-01-05	P1
P1-morgon	Här hör du de senaste nyh	04:29:00	04:30:00	2022-01-05	2022-01-05	P1
Nyheter från Ekot	Senaste nytt varje timme f	04:30:00	04:35:00	2022-01-05	2022-01-05	P1
Ring P1!	Ring oss på 020-22 10 10	04:35:00	04:46:00	2022-01-05	2022-01-05	P1
Morgonandakt	Tankar och reflektioner utif	04:46:00	04:55:00	2022-01-05	2022-01-05	P1

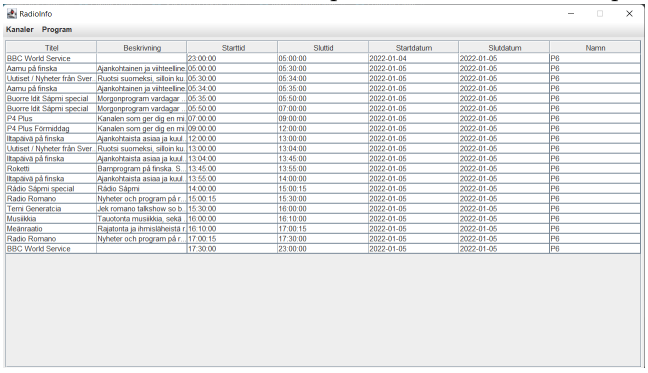
The user have multiple choices for what he wants to do. The user can click on the icon "Kanaler" of the menu bar and choose one of the menu items.

Figure 2: The drop down menu with channels



When clicking on one menu item, for example "P1", the table will only show episodes broadcasted by that channel. If the channel does not broadcast any episodes in a range of -12 and +12 hours, the gui will show an empty table. If the user clicks on any row on the table, a picture and a description will pop up giving more information about that episode.

Figure 3: The view when the user has pressed "P6" on the drop down menu



If the user has clicked on a channel to be viewed in the table and afterwards clicks on "Uppdatera" under the menu "Program", the gui is updated to the start view. Under the menu "Program", there is a menu item "Info" containing

information regarding the program as well as a exit button "Avsluta". The gui will also be updated automatically every hour.

3 System description

The system is constructed by the three packages forming the MVC pattern, that is XMLParser.java, Channel.java, ScheduledEpisode.java(Model package), GUI.java(View package) and Controller.java(Controller package). The main class is run in RadioInfoMain.java.

3.1 XMLParser

A class XMLParser.java has been created to read and write the XML files available in the open API that SR has. To be able to read these files, the Java DOM Parser was used. This allows the programmer to reach the information of the file by the nodes that have been created from the components of the XML file.

The XMLParser is built up by various methods that parses the url address connected to the channels aswell as for the scheduled episodes. To be able to save all the componentes connected to the channels and episodes, two arraylist are initialized in the scope. These arraylist can be reached by calling the methods getListOfChannels() and getListOfEpisodes().

When parsing the channels, the url address must match which channel id to be parsed. If the user does not want to see any specific channel but rather just an overview, the input to the method parseChannel() should be "". If the user presses the menu item "P1", the argument will be 132 which is the channel id for P1. The url address will in that case end with that channel id. An additional end to the url address will be added because all the available pages must be parsed. The method parseAmountOfPages() is therefore invoked and the amount of pages in the document is returned. If there is only one page to be parsed, 1 is returned. To get all the elements in the document, the method getElementsforChannel() is invoked. This method is setting the elements(using the Channel class) needed to present necessary information in the gui, and adding the elements in the arraylist listOfChannels.

The schedule url is set for every channel and is parsed using the method parseEpisodes(). To be able to get all the information from the arraylist, every Channel object of the list is looped through. The method getThreeDays() is taking in a date as a parameter and setting it to the date of today. Two other dates is also set and are placed in an arraylist of object LocalDate. These dates are then put to the end of the url address to parse for every date. This is necessary because the 12 hour range can occur either the day before today or after today. The amountOfPages() method is also called for to be able to parse the episodes on every page.

When the url address for the scheduled episode is set and parsed, the document's nodes are looped through to get all necessary elements. A local variable timecheck is created to check whether or not the episode is broadcasted in the

range of -12 and 12 hours from current time. If this is correct, the variable timecheck is set to true, otherwise the next episode will be looped through. The scheduled episodes that are in the correct range are placed in the arraylist listOfEpisodes containing ScheduledEpisode objects.

3.2 Channel and ScheduledEpisode

The class Channel.java sets and gets the information regarding the channel that has been parsed. These variables' values are put in an arraylist of Channel objects. This arraylist is then used to present the information in the GUI. The variables that are used in the class are id, name and scheduleURL. Unnessecary information from the document were not set and therefore not presented in the gui.

The class ScheduledEpisode.java sets and gets the information regarding the channels' program schedules that has been parsed. These variables' values are put in an arraylist of ScheduledEpisode objects. This arraylist is then used to present the information in the GUI. The variables that are used in the class are title, descritption, imageurl, name, startDate, startTime, endTime and endDate. Unnessecary information from the document were not set and therefore not presented in the gui.

3.3 GUI

The class Gui.java creates the user interface. Every component creating the gui is placed and packed inside a JFrame. At the top of the gui there is a menubar containing two menus with menu items. These are created in the method createMenuBar() where a XMLParser object is taken in as an argument to be able to reach the arraylist of Channel objects. The arraylist will be looped through to create a menu item for each channel name.

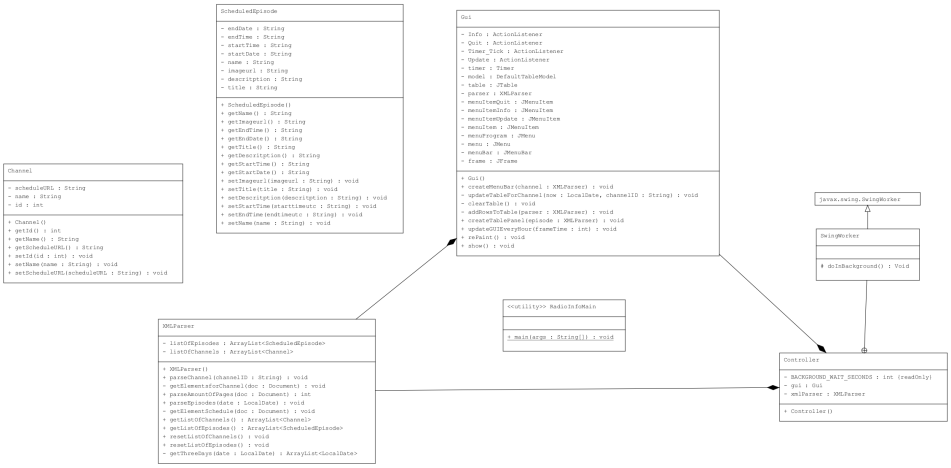
An action listener is added to every channel name in the menu. When clicking on a channel name in the menu, the table will be updated with the episodes that the channel is broadcasting. A menu "Program" is created with action listeners connected to every menu item. When clicking on "Uppdatera", the methods updateTableForChannel() is invoked. When clicking "Info" a JDialog is created with information regarding the program RadioInfo. When clicking "Avsluta", the entire window will be closed and the program will stop running.

The table is created by a DefaultTableModel and JTable. The columns are set and the rows are created by invoking addRowsToTable() which loops through the arraylist of ScheduledEpisode objects. Every row in the table is connected to a mouse listener. If the user clicks on a row in the table, a picture and description pops up using JDialog. When the table is updated by the method updateTableForChannel, the table is either updated to another channel or to the starting view. The GUI is also updated every hour by invoking the method updateGUIEveryHour() which using a Timer object to keep track of time. The timer is started when the program starts.

3.4 Controller and RadioInfoMain

The program will start by running RadioInfoMain.java, were the main class is situated. A Controller object will be created and the GUI will be updated using SwingWorker. The GUI is viewed on the event dispatch thread and the background tasks executes on the worker thread(by method doinbackground()). The worker thread will be executed when the GUI object has been created correctly. The updates for the gui is then executed on the background thread by firstly parsing the channel and the scheduled episodes. There is not a specific channel that is being parsed when starting the program, hence the argument "" in parseChannel(). The methods createMenuBar(xmlParser), createTablePanel(xmlParser), updateGUIEveryHour(an hour in milliseconds), gui.rePaint() are invoked to update the gui and start the timer.

Figure 4: UML Diagram representing the program RadioInfo



4 Thread safety

The Swing library is not in itself thread safe which means that the program must be built to be thread safe. When a Java program starts up, one thread begins running immediately. This is called the main thread main thread because it is the one that is executed when our program begins. If other threads are executed, these threads will be spawned from the main thread. To implement a thread safe program, one should consider to handle data that needs to be computed, while the user interface must still be interactive to user input. For this reason, the program is divided so that the graphical user interface and updates runs only on the EDT thread that Swing provides, while time consuming updates

such as loading the XML file and displaying that information are done in the background with SwingWorkers, on the worker thread.

5 Design Patterns

This program is mainly built up from a design pattern called MVC. In the model classes, there is only data saved to be presented in the gui and handled in the controller class. The view, only takes care of presenting the data. It has access to the model's data but does not manipulate the data. The controller class exists between the model and view and works as an updater to the gui. In the case of my program, a bit of the controller's responibilities has been put to the gui class. This worked better for my code and the updates are still run on the worker thread.

6 Further development

The program that was built has several areas of improvement. The graphical user interface can be developed to interact better with the user. To this day, the program has a pretty simple design and because of time limitation this could not be developed in the amount that was desired. Therefore, things like adding more information regarding channels and episode, add pictures and colors could be beneficial.

When the user starts the program, the table shows the scheduled episodes in the order of channels. The earliest broadcasted episodes are put to the top of that channel's episodes. It would probably be better if the table was sorted in a way were the episodes broadcasted in the nearest future are put first, regardless of channel.

If the user has decided to look at only one channel's episodes and presses the update button in the drop down menu, the user will be thrown back to the start view. It could be benificial if the actual view showing where to be updated instead.

When starting the program, it takes a while for the program to load the data for the gui. To make the interaction with the user better, a bit of the arraylist of data could be updated to the table before the entire data set has been parsed. Because of lack of time, this was not implemented.