Umeå University
Department of Computing Science

Application Development in Java 7.5 p
5DV135

RadioInfo

Submitted 2017-01-11
Author: Lorenz Gerber (dv15lgr@cs.umu.se lozger03@student.umu.se)
Instructor: Johan Eliasson / Jan Erik Moström / Alexander Sutherland / Filip Allberg / Adam Dahlgren Lindström

Contents

1 Introduction

The aim of this laboration was to develop an application that uses the open web API from 'Sveriges Radi' (*http://sverigesradio.se/api/documentation/v2/index.html*) to present the current radio program in a graphical user interface.

2 Usage

2.1 Compiling from command line

The source code for the Java application 'RadioInfo' was divided into the packages 'model', 'view', 'controller' and 'data_io'. The class containing the 'main' method was contained in the base directory (default package). The follwing steps were conducted to build a jar application file:

```
mkdir biuld
javac -d ./build *java model/*.java view/*.java controller/*.java ./
mode_io/*.java
cd build
jar cvfe RadioInfo.jar RadioInfoMain *.class model/ view/ controller/ ./
data_io/
```

The '.jar' file is named 'RadioInfo.jar' while the 'main' containing class is name 'RadioInfoMain'. The source code is provided in a separate tar.gz file.

2.2 Program Usage

On some operating systems, double click on 'RadioInfo.jar' will start the applicatoin. From the command line on OSX and -nix systems, it can be invoked by:

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

The application will start and directly show a three split window with the program table on the left, textual program description om the middle and program image on the right. Radio programs that started before the current time are shown with white font on black blackground while programs to be broadcasted are shown with black font on white background. To get more information about a specific program, the user can click in the table on a program. If available, the respective description and image will be shown in the middle and right side panel. The 'RadioInfo' application has a menu bar with two drop-down menus: File and Channels. In the file menu, 'Reload' allows to refresh the program table by reloading it from the web, while 'quit' will terminate the background update service and exit the application. In the 'channels' menu, all available channels are shown and can be selected. After selection, the respective program is loaded from the net and visualized in the leftmost panel. There is a background process running that will update the program table once every hour.

3 System Description

The whole application is structured according to the Model-View-Controller pattern. This mantra was also followed for packaging the source files. Besides the model, view and

controller package, there is also a ‘data_io’ package that contains the XML classes for interacting with the web API. The ‘RadioInfoMain’ class can be found in the base directory.

3.1 General Structure

In figure 1, the basic structure of the application is shown. The ‘main’ method in *RadioInfoMain Class* starts a new Swing thread, by defining an anonym *runnable* method. In this method first the main data containers ‘channels’ (*ChannelListModel Class*) and ‘programs’ (*ProgramListModel Class*) are set up. Next the gui instance is started and used as method argument to the ‘main’ (*MainController*). ‘main’ then bootstraps the whole application.

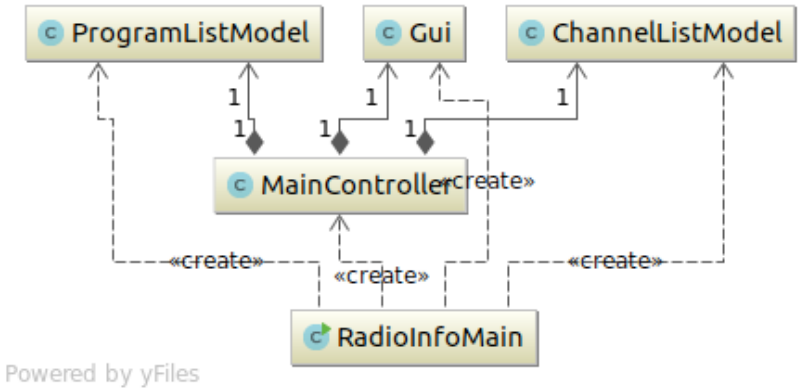

Figure 1: *This figure shows the basic structure of the application: The main method is in the RadioInfoMain.class. It defines and starts a new thread. In this thread the two main data containers, channels (ChannelListModel class) and programs (ProgramListModel Class) are set up. Then it instantiates the gui and the MainController. The main controller takes a reference to the gui instance as construction argument. The main controller bootstraps the whole application then.*

3.2 Data and Data Handling

test test

3.3 The Graphical User Interface

test testo

3.4 Threads and Thread Safety

The application uses separate threads for updating the data: SwingWorkers are invoked to not block the UI while pulling new data from the web API. As these operations could be potentially slow, they could be invoked again while the old one is still running. More over, user interactions such as clicking in the table potentially access the same data containers: ‘programs’ and ‘channels’ and ‘current channel’. To make the application thread safe, synchronized methods where used. As all datacontainers get accessed through the MainController, the getters and setters in the MainController for the respective data containers were synchronized.

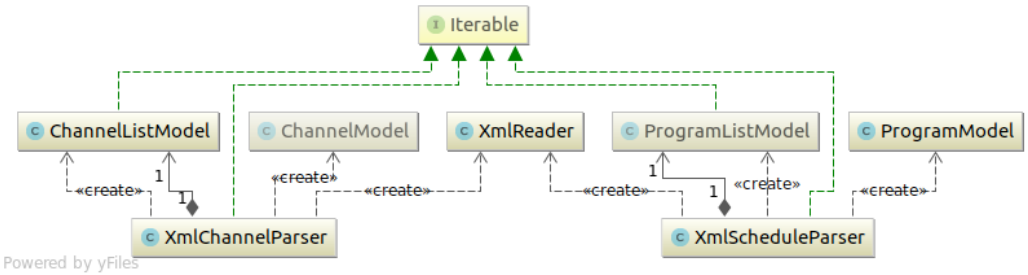

Figure 2: *This figure shows the data models used in RadioInfo Application*

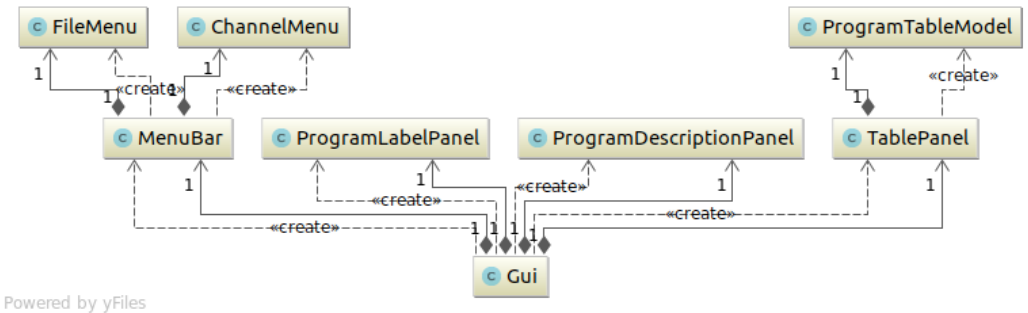

Figure 3: *This figure shows the simple structure of the graphcial user interface. TablePanel, ProgramDescriptionPanel and ProgramLablePanel are all sub-classed from JPanel. They are organized in a JFrame using a GridLayout layout manager that specifies three columns and one row. The MenuBar (subclassed from JMenuBar) has two sub menus (subclassed from JMenu): FileMenu and ChannelMenu.*

3.5 Summary of Design Patterns

Observer-Observable Model-View-Controller Iterator

4 Limitations

5 Testing