ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA

Rock Paper Scissors Lizard Spock

Carlo Antenucci, Alessandro Fedi, Leonardo Iannacone

Contents

1. Introduction

Rock Paper Scissors Lizard Spock (RPSLS) is an expansion of the classic selection method game rock-paper-scissors. It operates on the same basic principle, but includes two additional weapons: the lizard (formed by the hand as a sock-puppet-like mouth) and Spock (formed by the Star Trek Vulcan salute). The game was invented by Sam Kass with Karen Bryla, as "Rock Paper Scissors Spock Lizard"[1].

This software version of the game is developed at University of Bologna and has three main nodes:

SIB Server: provides the users interacions and stores the games informations (status, players, scores, etc.)

Client GUI: represents the playing node, which allows the users to play with each other

Admin GUI: supply to the SIB Server Administrator a graphical interface useful to manage the game ontology and the game's entries.

This is the **Developer Manual**. Code for this project is available at https://github.com/LeoIannacone/rpsls

2. Requirements

2.1. Download code

You can get code by typing this in your command line:

```
git clone https://github.com/LeoIannacone/rpsls
```

If you do not have **git** installed on your machine, you can get the latest version of **RPSLS** at this address: https://github.com/LeoIannacone/rpsls/archive/master.zip

2.2. Jena Libraries

You need to get **Jena** libraries and put them in the right place. You can use to script **download_jena.sh** provided along with RPSLS code.

Enter the folder with RPSLS code and type this in a command line:

```
bash libs/download-jena.sh
```

2.3. Eclipse configuration

RPSLS has been developed in **Eclipse IDE**. If you want use this IDE you have to configure a variable called "**CALCOLATORI_LIBS**" and point it to **RPSLS/libs** folder.

Open Eclipse and set **RPSLS/workspace** as current workspace.

Go to ***Window → Preferences → Java → Build Path → Classpath Variables*** and click on "**New...**"

In **Name** field enter "**CALCOLATORI_LIBS**"

In **Path** enter the Folder where you have downloaded RPSLS and then choose /**libs** subdirectory.

Confirm by clicking on **Ok** buttons and then restart Eclipse.

3. The Java Projects

We have splitted code into different projects and packages. Here some explanation.

3.1. The library sofia-kp

The project *sofia-kp* is the Java-KPI interface made by University of Bologna. For more information, please visit the project homepage at: http://sourceforge.net/projects/smartm3-javakpi

3.2. The main projects

3.2.1. it.unibo.games.rpsls.interfaces

This project contains the main interfaces on which we built RPSLS. Here you can find three packages:

— **it.unibo.games.rpsls.interfaces** contains base interfaces which describe some Game entitiy and some basic connector (SIB) entity

[1] http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock

— **it.unibo.games.rpsls.interfaces.admin** contains the interfaces for admin
— **it.unibo.games.rpsls.interfaces.client** contains the interfaces for client
Whenever you want to extent RPSLS you may start analizing problem and add new interfaces here.

3.2.2. it.unibo.games.rpsls

In this project you can find the main entities which describe the Game itself and some Utils we used. Packages involved are:
— **it.unibo.games.rpsls.game** contains the entities involved in a simple Rock Paper Scissors Lizard Spock instance
— **it.unibo.games.rpsls.utils** contains some common utils used in other projects and packages

3.3. Connector project

The *it.unibo.games.rpsls.connector* projects contains the classes able to communicate with SIB. Here you can find following packages.

3.3.1. it.unibo.games.rpsls.connector

The **it.unibo.games.rpsls.connector** package contains all basic information and basic classes about SIB communication. Here we are:
— **Config.java** contains the default SIB configuration such as host and port
— **SIBFactory.java** is KPi able to retreive information from SIB and build Java classes as defined in the pacakge **it.unibo.games.rpsls.game**
— **SIBSubscription.java** is a basic class to make faster and easier subscription into SIB
— **Utils.java** a simple utils used in this project

3.3.2. it.unibo.games.rpsls.connector.admin*

The packages **it.unibo.games.rpsls.connector.admin** contains all methods and classes needed by the Admin GUI to interact with SIB.

In the subpackage **it.unibo.games.rpsls.connector.admin.resources** you can find the Ontology we develop to describes this game.

3.3.3. it.unibo.games.rpsls.connector.client

The packages **it.unibo.games.rpsls.connector.client** contains all methods and classes needed by the Client GUI to interact with SIB.

3.4. Client GUI project

The project *it.unibo.games.rpsls.client* contains the package **it.unibo.games.rpsls.client.gui** where the Client GUI has been developed.

The classes in this package are structured as follow:
— **ClientMainWindow.java** is the main class which creates the window and an instance of **SIBClient** and implements the ClientObserver interface able to handle with subscriptions events
— **ViewDefault.java** is an abstract class used to subclass all other Views
— **View*.java** are all the views for the GUI, in details:
  — **ViewConfigConnetor.java** is the first view showed, which asks to configure the SIBClient connector with an Host and a Port
  — **ViewWelcome.java** is the view showed at second step. Here user inserts a username and choose to Start a new game or Join an existing one
  — **ViewJoinGame.java** if user choose to join a game, a new instance of this class is created. This is the class in chair to handle with events come from SIBSubscriptionWaitingGames.java
  — **ViewMatch.java** is the view where a game or match takes place
  — **ViewWin.java** is the last view showed when a game ends. It reports to users the result of the match and/or if the opponent player has left game before games ended
— **Panel*.java** are common classes used to build up the views. For more details please read the code.
The other subpackages **it.unibo.games.rpsls.client.gui.data.*** contains instead all data (most of them are images) used in the client GUI.

3.5. Admin GUI project

The project *it.unibo.games.rpsls.admin* contains the package **it.unibo.games.rpsls.admin.gui** which, similar for the client GUI, has all the classes used to develop the Admin GUI.

Classes in this package are structured as follow:

— **AdminMainWindow.java** is the main class which creates the window and an instance of **SIBAdmin** and implements the AdminObserver interface able to handle with subscriptions events

— **ViewDefault.java** is an abstract class used to subclass all other Views

— **ViewConfigConnetor.java** is the first view showed, which asks to configure the SIBAdmin connector with an Host and a Port

— **ViewMain.java** is the main view for the GUI. Here user can choose what to do on the SIB by admin side such as Reset, Init, Clean, Delete Games.

3.6. Other projects

3.6.1. it.unibo.games.rpsls.test

This project contains some test we made during develop. Tests are really important for software developing, when you are writing new code, you can use tests to validate your code works as expected or if you are going to refactor or modify old code, you can use tests to ensure your changes haven't affected your application's behavior unexpectedly.

Please use tests to validate your extensions.

3.6.2. it.unibo.games.rpsls.prototypes

Here you can find some prototype we made during develop. Prototype are piece of software which respect the requirements (in our case, interfaces) and help developers to focus on another part while someone else is writing the real code which implements interfaces.

If you are in a team, you may want to use some prototype.

4. License