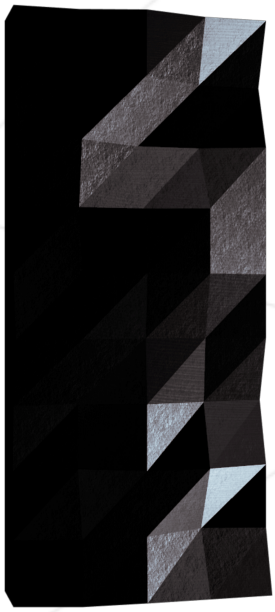

Swingy

GUI programming in Java

Alex alex@academyplus.ro
42 Staff pedago@42.fr

Summary: This is the second project from the **Java** *world at 42. You will learn to develop GUI applications with the SWING framework*

Version: 1

Contents

Chapter I

Forewords

In England 932 A.D., King Arthur, along with his squire, Patsy, recruits his Knights of the Round Table: Sir Bedevere the Wise, Sir Lancelot the Brave, Sir Galahad the Pure, Sir Robin the Not-Quite-So-Brave-As-Sir-Lancelot, and the aptly named Sir Not-appearing-in-this-film. On the way, Arthur battles the Black Knight, who continues to fight despite having his arms and legs cut off. (King Arthur declares at the end of the fight, "All right, we'll call it a draw.") The knights reach Camelot, but following a song-and-dance cutaway, Arthur decides not to enter, because "'tis a silly place". The group is soon encountered by God (shown as a cutout animation of British cricketer W.G. Grace), who instructs them to seek the Holy Grail.

Chapter II

Introduction

This is the second project in a series of 4 `Java` projects, produced by Academy+Plus. This time the focus will be on graphical user interfaces and related software design patterns.

The interaction of computer software with humans, raises many problems and requires specific solutions. Object Oriented programming is a paradigm that suits well this kind of problems, but also needs to introduce new ideeas, like event-driven programming.

You will have to implement a minimalistic text-based RPG game and apply the best practices suited for this problem. At the end of the project you will understand how to abstract the user interface or the view from other parts of the application.

Chapter III

Goals

A group of hipster entrepreneurs wants to launch a new text based rpg. Since they don't believe beautiful graphics is important and think that the game play is the most important part of making a successful game, they decided to invest their money in other areas and make the game text based. They even want to release it in 2 phases:

- First phase for hardcore hipsters that will be console based
- Second phase for regular hipsters, that will also have a simple GUI for taking user input.

They come to your Java shop because they know they will get the best products if they work with you. Altough they mostly care about the features, they have some programming requirements to be implremented in the project:

- Respect the `Model-View-Controller` design pattern.
- Automated build with `Maven`.
- Annotation based user input validation.

Only a good and clear implementation will be accepted. For this to happen, it will have a clean design, will be easy to read and understand by your peers and will be easy to change in case the requirements are modified.

`Builder pattern`

`Hibernate Validator`

`The major IDEs have good support for support for Maven. You will need to learn about the Maven plugin system in order to generate a runnable jar file.`

Chapter IV

General instructions

- You are allowed to use language features up to the latest Java LTS Version included.
- You are are allowed to use any external libraries, build tools or code generators.
- Do not use the default package.
- Create your own relevant packages following the `Java` package naming conventions.
- Java is compiled into an intermediate language. This will generate some .class files. Do not commit them on your repository!
- Make sure you have javac and java available as commands in your terminal.
- Make sure you have the mvn command line tool available, or use one bundled in your IDE.
- Build the project running the command bellow in the root of your project folder. This needs to generate a runnable .jar file that can launch the game

```
$mvn clean package
```

Chapter V

Mandatory part

You need to implement a text-based RPG based on the gameplay and conditions described below. The program needs to follow the Model-View-Controller architecture and allow switching between the console view and GUI view.

V.1 Gameplay

A player can have multiple heroes of different types. We leave it at you to name the hero types and fine tune the different starting stats between them, When the player starts the game he has 2 options:

- Create a hero
- Select a previously created hero.

In either case, the player can see the hero stats:

- Hero name
- Hero class
- Level
- Experience
- Attack
- Defense
- Hit Points

Hero stats are affected by the hero level and artifacts. There are 3 types of artefacs:

- Weapon - increases the attack
- Armor - increases defense
- Helm - increases hit points

After choosing a hero the actual game begins. The hero needs to navigate a square map with the size calculated by the formula (level-1)*5+10 -(levela hero of level 7 will be placed on a 39X39 map.

The initial position of the hero is in the center of the map. He wins the game if he reaches on of the borders of the map. Each turn he can move one position in one of the 4 four directions:

- North
- East
- South
- West

When a map is generated, villains of varying power will be spread randomly over the map. When a hero moves to a position occupied by a villain, the hero has 2 options:

- Fight, which engages him in a battle with the villain
- Run, which gives him a 50If the odds aren't on his side, he must fight the villain.

You will need to simulate the battle between the hero and monster and present the user the outcome of the battle. We leave it at you to find a nice simulation algorythm that decides based on the hero and monster stats, who will win. You can include a small "luck", component in the algo in order to make the game more entertaining.

If a hero looses a battle, he dies and also looses the mission.

If a hero wins a battle, he gains:

- Experience points, based on the villain power. Of course, he will level up if he reaches the next level experience.
- An artifact, which he can keep or leave. Of course, winning a battle doesn't guarantee that an artefact will be droped and the quality of the artefact also varies depending on the villain's strenght.

Leveling up is based on the following formula `level*1000+(level-1)^2 * 450`. So the necessary experience to level up will follow this pattern:

- Level 1 - 1000 XP
- Level 2 - 2450 XP
- Level 3 - 4800 XP
- Level 4 - 8050 XP
- Level 5 - 12200 XP

V.2 Features

The game can be launched in 2 modes as described below.

```
$java -jar swingy.jar console
```

```
$java -jar swingy.jar gui
```

A user's heroes and their state will be preserved, when the user exits the game, in a text file. When starting the game, your program will load the heroes from this file.

V.3 Validation

You will need to integrate a third party library in your project in order to provide annotation based validation. We highly recommend that you use a library that implements the javax.validation specification.

You will not allow any abnormal user input to disrupt the game behaviour. Validation failure will be highlighted to the user.

Chapter VI

Bonus part

Bonus points will be given if:

- You persist the user's heroes in a relational database, instead of a text file.
- You can switch between console view and GUI view at runtime, without closing the game.

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.