Department of Informatics

Professor: R. Ingold
Teaching Assistants: M. Alberti, L. Vögtlin
Assistant: R. Geissbühler

Assignment 06

1 Theory

In this exercise you have to answer the following theoretical questions. Keep your answers short and precise.

(a) What are the different form of interactions with an application? When are they appropriate?

(b) What is the main difference between a GUI and a CLI in terms of user interaction? *Hint: think about who decides the flow of the program.*

(c) Which are the main libraries for GUI development in Java? Which one is the best and why?

(d) What is the main advantage of the MVC design pattern?

(e) Mention 5 *common* graphic components available in Swing and AWT.

(f) Mention 3 *common* interactive components available in Swing.

(g) What is a `java.util.EventListener` ? How does it work?

(h) How can we specify the behaviour of a button?

(i) What is an internal class? Can it be static ?

(j) What is an anonymous class? Why would I use one?

2 Implementation

For this exercise you are required to submit one file: ButtonCreator.java, implemented as described below.

(a) Implement a `ButtonCreator` class which implements `Runnable`. This class has to show the user two different windows:

1. A Selector window like shown in Figure 1
2. A ButtonColor window like shown in Figure 2 and 3

So first you show to the user a Selector window (see Figure 1) where the user can chose the amount of buttons for the ButtonColor window. When the user entered a number n and pushed the "OK" button you have to hide the Selector window (setVisible(false)) and create a new ButtonColor window (see Figure 2 and 3).

Hint: This window consists out of a JLabel (Amount of Buttons), a JTextField [1] and a JButton.

In the new ButtonColor window you show now n buttons. Each of that buttons has a random color name and when the button got pressed it changes the color of the background to this color.

Hint: You can define a fixed array of colors and just randomly chose from it.

As a starting point you can take the ButtonColor window presented in the lecture.

[1] https://docs.oracle.com/javase/8/docs/api/javax/swing/JTextField.html

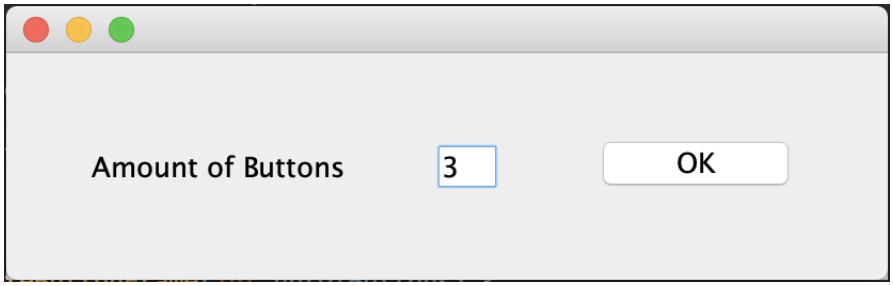

Figure 1: Example of the GUI for the implementation exercise to select the amount of buttons.

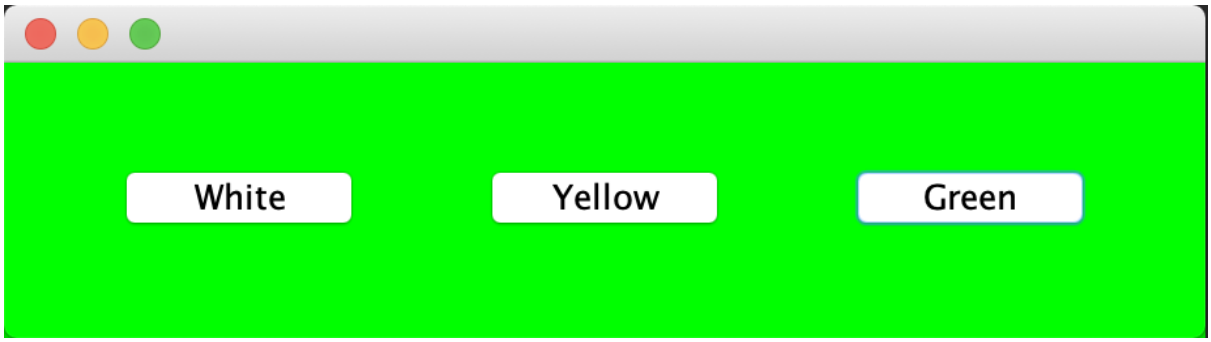

Figure 2: Example of the GUI for the implementation exercise after "Green" button have been pressed.

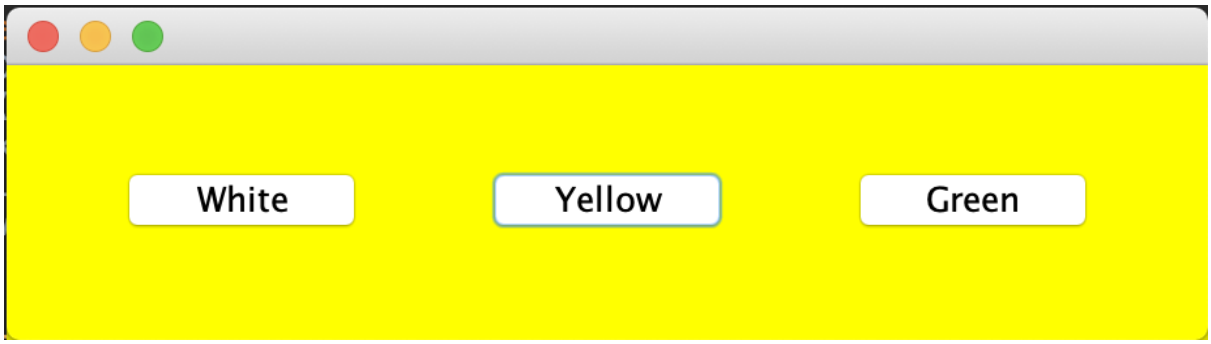

Figure 3: Example of the GUI for the implementation exercise after "Yellow" button have been pressed.

3 Debugging

In this exercise we want to practice the use of the debugger and familiarize with the error messages of the compiler. Refer to the files `Main.java` attached to this series for this exercise.

(a) Run the `Main` file.

(b) Notice that everything works without errors, against expectations for the debugging exercise. Does this mean this code runs perfectly fine? If you think it does, read the course slides over and over again until you find out why the correct answer is that it doesn't. What is the problem with this code? Why does this happen?

(c) Fix the problem outlined in previous point by modifying the code accordingly.

(d) What is going on at line 23? Where is the definition of the `ActionListener`? What does the symbol $->$ mean and what is that?

For reference (you don't have to rewrite them yourself, we provide the java files attached to the series):

```
1  package core;
2
3  import javax.swing.*;
4
5  public class Main{
6
7      public static void main(String[] args) {
8          init();
9      }
10
11     private static void init() {
12         JFrame frame = new JFrame();
13         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14         frame.add(content());
15         frame.pack();
16         frame.setVisible(true);
17     }
18
19     private static JPanel content() {
20         JPanel panel = new JPanel();
21         JLabel label = new JLabel("Hello World ! This is my first
               Swing application");
22         JButton button = new JButton("Exit");
23         button.addActionListener(e -> System.exit(0));
24         panel.add(label);
25         panel.add(button);
26         return panel;
27     }
28 }
```

4 Bonus exercise

For this exercise you are required to submit two files: ClickCounter.java and ClickCounterWithAnonymClass.java, implemented as described below.

(a) Implement a `ClickCounter` class which implements `Runnable`. This class has to show the user a GUI *similar* to the one shown in Figure 4 where a counter shows how many time a user has clicked the button. Use an inner class for implementing `ActionListener` of the button.

(b) Implement a `ClickCounterWithAnonymClass` class as an exact copy of `ClickCounter` with the difference that this makes use of an anonymous class for the `ActionListener` on the button instead of a inner class.

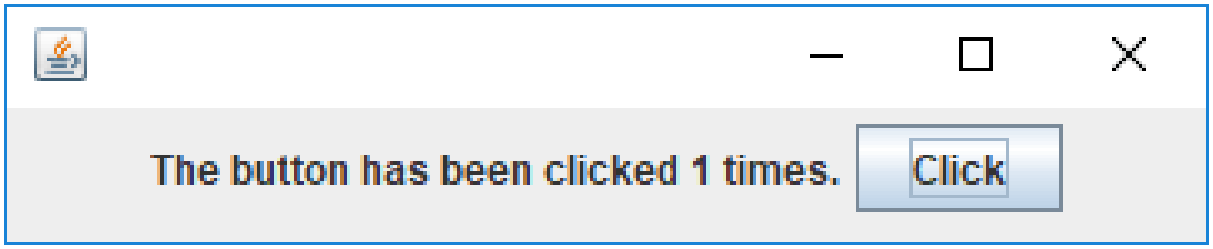

Figure 4: Example of the GUI for the bonus exercise after the button have been pressed once.

5 Project

For this step of the project, you have to plan and start developing the *Graphical User Interface (GUI)* for the Space Invaders game. Also you have to implement the shooting for the `Player` and the `Enemies` as well as the movement of the `Player` with the help of `EventListeners` to give commands.

5.1 Planning the GUI

As a first step you need to draw on paper or with the help your favourite drawing tool your GUI. The GUI need at least a main window where the game takes place. All other components you can choose freely.

5.2 Implementing the Main Window

As you finish planing your GUI you have to implement it. For that you have to create a class `Board` which will represent your GUI.
Your class `Board` needs to extends the class `Display`, which will make it possible to draw images onto your `Board`. The class `Board` needs to have the following properties:

- height
- width
- enemies
- shots
- player

These parameters are initialized within the constructor. Choose the scope of the variables and the access though getters and setters as needed.
To store the `Enemies` and the `Shots` use the lists you implemented in Series 3. You are allowed to change the lists or create new ones as needed. Do NOT use another type of collection to store the shots or the enemies.
You also have to implement the paint method of our new class `Board`. Do not forget: Every `Sprite` can already draw itself onto a `Painting`.
Optional: You can also implement methods which displays the victory and the defeat message.

5.3 Create a Main Class and Add KeyListeners

To start the whole game you need to create a Main class which also acts as the controller with the name `SpaceInvaders`. As a template for this class you can take any `Tester` class you wrote in the past weeks. Just take care that it extends the class `Animation` and have a private empty constructor.
The class needs to have the following properties:

- board_with
- board_height
- the ShotList
- the EnemyList
- the player
- the board

Do not forget to override the `init()` method. In this method you should create the `Board` and add a `KeyListerner` to it. For that you can use the class `KeyAdapter` in a anonymous way or implement it as an inner class. Override in this class the method `keyPressed()` and add the different functionality for the different keys. If you want to control your `Player` with the arrow keys and the space bar you can use the constants `KeyEvent.VK_LEFT`, `KeyEvent.VK_RIGHT` and `KeyEvent.VK_SPACE`. Do not forget to set the `Board` as the display.
To start your GUI create a main method and create a new instance of your `SpaceInvaders` class and call the `launch()` method with the automatic parameter set to `true`.

5.4 Attacking for the Player and the Enemies

Now we want that our `Player` and the `Enemies` can attack and so can fire shots. Implement your attack/shot method for the `Player` and the `Enemies`. Take into account the following restrictions:

Player Should just be allowed to fire a shot every 500ms

Enemies Fires at each frame a shot with a 0.5% probability

You do not have to implement the health of the Game Objects nor the collision.