GUIs in Java

Melvyn Ian Drag

November 26, 2019

Abstract

So far all of our projects have only done input/output on the command line. Today we will add a GUI to a project.

1 Introduction

A GUI is a **Graphical User Interface**. The term refers to a program with a pretty graphical front end that features things like buttons and sliders, will perhaps show images and text in beautiful ways, and more. So far we've only done commandline-driven programs, but you a probably more familiar with GUIs in the form of various desktop applications. People who use Linux, BSD, and (to a lesser extent) Macs are often comfortable with and happy to use command line programs - the rest of the world, as a generalization, wants a GUI.

2 Goal of the Night

Tonight's class is something of a guided research project. We'll start with a simple bit of code and then add some features to it.

3 How to Make a GUI

Two popular java libraries for making GUIs are *swing* and *awt*. So far as I can tell, *swing* is widely used and *awt* is considered old and only really used for teaching because it is very simple. That's probably an unfair generalization, but that's what sticks in my memory after a few months of reading about Java.

3.1 A question!

In our code today we will be importing stuff from both *awt* and *swing*.

```
1 import javax.swing.JFrame;
2 import javax.swing.JButton;
3 import javax.swing.JOptionPane;
4 import javax.swing.SwingUtilities;
5 import java.awt.event.ActionListener;
6 import java.awt.event.ActionEvent;
```

The *ActionEvent* and *ActionListener* are both from *awt*, and these are the tools we use to program what our code does when a button is clicked. Why are we mixing libraries here? Does *swing* not have it's own dedicated action handlers? This code we're using today I got randomly off the internet on some Java blog.

See: **Insert link here, when found, can't remember where I got the code and I'm offline right now**

4 Extra Credit Today

4.1 Objective

I want you to create little application has 4 buttons. When clicked, 1 of them will print out the utf16 hex representation of an emoji. The other will print the utf16 binary representation. The other will print the utf8 hex. The last the utf8 binary.

As you can see by the figure 1 this is just a little thing I dreamt up and scribbled on the back of a piece of paper. But I've played with the ideas involved and it is a doable assignment if you know a little bit of Java!

Also, look at 2. I've given you code that creates this program. The code is very short and sweet, and does what we need! It has a couple of buttons that, when clicked, throw up a little window with a string in it. We just need to figure out what strings we want to show. Let's do that together.

4.2 Team Work - Let's Work out the Hex and Binary

As a team, figure out what the UTF-8 and UTF-16 representations are for a smiley emoji in binary and in hex. Write it on the board. These are strings to be used in your programs.

4.3 The Template

I've worked a bit on some code template I found on the internet. This is what you will use for the assignment. This code below when you compile and run it, then click a button should do what is shown in figure 2

Below is the source code for the template. As always, the code can be found in the class repo. The code for this week is in *Week13_Swing/Code/Swing/Sample.java*

```
1  import javax.swing.JFrame;
2  import javax.swing.JButton;
3  import javax.swing.JOptionPane;
4  import javax.swing.SwingUtilities;
5  import java.awt.event.ActionListener;
6  import java.awt.event.ActionEvent;
7
8  public class Sample extends JFrame {
9    public Sample() {
10     super();
11     this.setTitle("HelloApp");
12     this.getContentPane().setLayout(null);
13     this.setBounds(100, 100, 180, 140);
14     this.add(makeButton());
```

```
15      this.add(makeButton2());
16      this.setVisible(true);
17      this.setResizable(false);
18      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
19    }
20
21    private JButton makeButton() {
22      final JButton b = new JButton();
23      b.setText("Click me!");
24      b.setBounds(40, 80, 200, 30);
25      b.addActionListener(new ActionListener() {
26        public void actionPerformed(ActionEvent e) {
27          JOptionPane.showMessageDialog(b, "Hello World!");
28        }
29      });
30      return b;
31    }
32
33    private JButton makeButton2() {
34      final JButton b2 = new JButton();
35      b2.setText("Click me!");
36      b2.setBounds(40, 40, 100, 30);
37      b2.addActionListener(new ActionListener() {
38        public void actionPerformed(ActionEvent e) {
39          JOptionPane.showMessageDialog(b2, "Hello World!");
40        }
41      });
42      return b2;
43    }
44
45    public static void main(String[] args) {
46      // Swing calls must be run by the event dispatching thread.
47      try{
48        SwingUtilities.invokeAndWait(() -> new Sample());
49      }
50      catch( Exception e ){
51        System.out.println("error");
52      }
53    }
54  }
```

Notice that there are calls to the following functions:

1. *setBounds()*
2. *setTitle()*
3. *setResizeable()*

4. *addActionListener()*

5. *showMessageDialog()*

Let's take a minute to play with these functions and see what they do (make sure to change the showMessageDialog() call in addActionListener(), but don't acturally change action listener. That can get tricky.)

4.4 Extra Credit

See if you can accomplish the task and I'll give you an extra credit! Ready? Set.... Go!

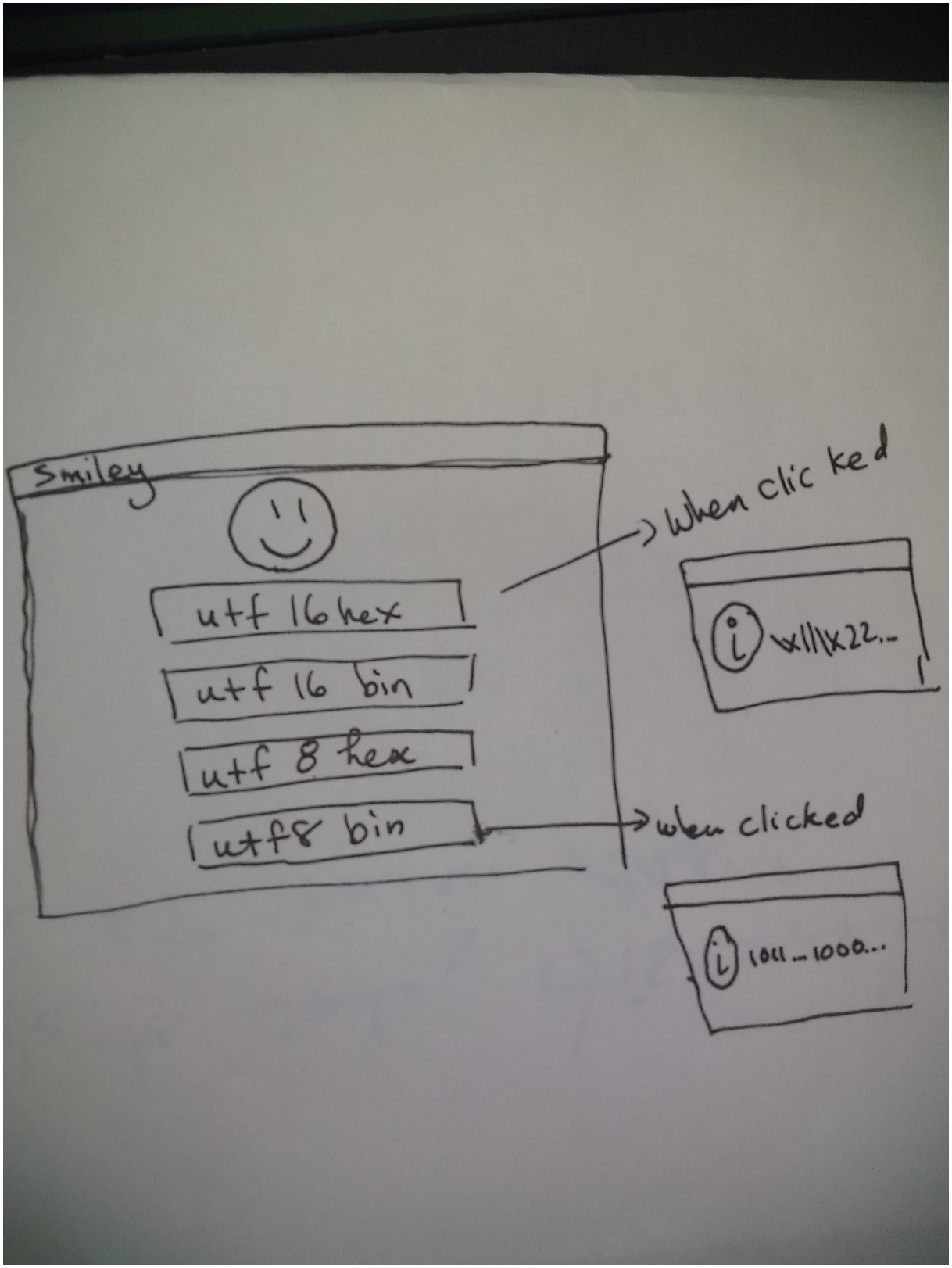

Figure 1: Your goal for today.

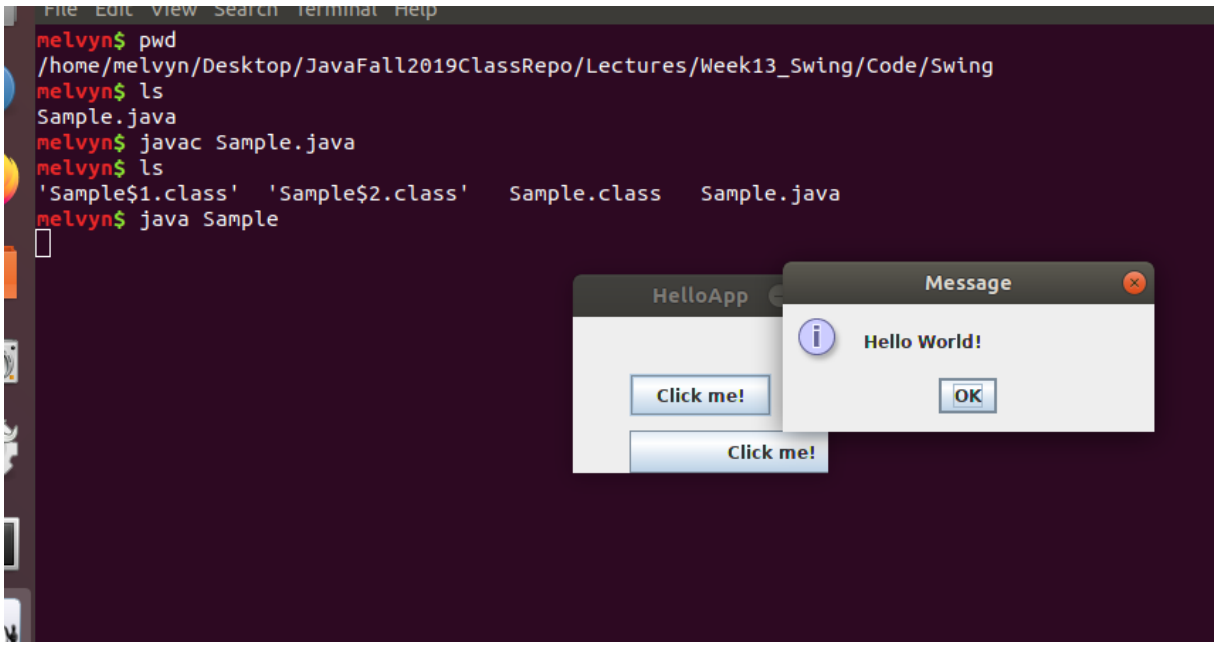

Figure 2: What happens when you run the template code.