Exercises

Java Swing for cool GUIs, module 2

Exercise 2-1: Refactoring

Usually, we don't copy and paste because we want to do the hard work to build competencies 😉 But for this exercise, copy the following code into your IDE, and refactor it so you have a main() method that instantiates the class and calls a run() method. Let the constructor set up the GUI work, and the run() method wait for 1,000 milliseconds and change the text of the label.

```
import javax.swing.JFrame;
import javax.swing.JLabel;

public class HelloWorldNotRefactored {

  public static void main(String[] args) {
    // 1. Move JFrame and JLabel to attributes right under class declaration
    JFrame frame;
    JLabel label;

    // 2. Move this segment to construtor:
    frame = new JFrame("HelloWorldRefactored");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.add(label = new JLabel("The label"));
    frame.setSize(300, 200);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);

    // 3. Move this segment to run() method
    try {
      Thread.sleep(1000);
    } catch (Exception e) { }
    label.setText("Label's new text");
  }

  // 4. Finally, let main() method create an instance of class and call run()
method
}
```

Exercise 2-2: ValidateAndRepaint

All you need to do here is to call validate() and repaint() on the JFrame instance. Run this code, then find the commented line below and call the methods. Then, run the code again.

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;

public class ValidateAndRepaint {
  private JFrame frame;
  private JPanel panel;
  private JLabel label;

  ValidateAndRepaint() {
    frame = new JFrame("ValidateAndRepaint");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.add(panel = new JPanel());
    panel.add(label = new JLabel("The label"));
    frame.setSize(300, 250);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
  }

  void run() {
    JButton button = new JButton("Click me");
    panel.add(button);
    // call frame.validate() and frame.repaint() here
  }

  public static void main(String[] args) {
    new ValidateAndRepaint().run();
  }
}
```

Exercise 2-3: Add a JButton - or two

Run the following code. Then, add a JButton to the panel and run the code again.

For extra fun, add an extra button. Do you see both buttons, or do you need to revisit your code?

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class HelloButton {
  private JFrame frame;
  private JPanel panel;
  private JButton button;

  HelloButton() {
    frame = new JFrame("HelloButton");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);
    frame.setLocationRelativeTo(null);
    frame.add(panel = new JPanel());
    // Add JButton here
    frame.setVisible(true);
  }

  void run() { }

  public static void main(String[] args) {
    new HelloButton().run();
  }
}
```

Exercise 2-4: Add an ActionListener

Add an ActionListener that changes the text of the button to "Thanks for clicking!".

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class HelloActionListener {
  private JFrame frame;
  private JPanel panel;
  private JButton button;

  HelloActionListener() {
    frame = new JFrame("HelloActionListener");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);
    frame.setLocationRelativeTo(null);
    frame.add(panel = new JPanel());
    panel.add(button = new JButton("Click me!"));
    frame.setVisible(true);

      // Add ActionListener here
      // Then add the ActionListener to the button

  }

  public static void main(String[] args) {
    new HelloActionListener();
  }
}
```

Exercise 2-5: Add an anonymous ActionListener

Add an anonymous ActionListener that changes the text of the button to "Thanks for clicking!".

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class HelloAnonymousActionListener {
  private JFrame frame;
  private JPanel panel;
  private JButton button;

  HelloAnonymousActionListener() {
    frame = new JFrame("HelloActionListener");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);
    frame.setLocationRelativeTo(null);
    frame.add(panel = new JPanel());
    panel.add(button = new JButton("Click me!"));
    frame.setVisible(true);

    // Add anonymous ActionListener to button here


  }

  public static void main(String[] args) {
    new HelloAnonymousActionListener();
  }
}
```

Exercise 2-6: Add an ActionListener from a separate class

Use the following code. Then, create a class that implements the ActionListener interface and overrides the actionPerformed(ActionEvent ae) method. Let this method change the text of the button to "Thanks for clicking!".

Now add an instance of that class as an ActionListener to the button.

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class HelloActionListenerInterface {
  private JFrame frame;
  private JPanel panel;
  private JButton button;

  HelloActionListenerInterface() {
    frame = new JFrame("HelloActionListener");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 250);
    frame.setLocationRelativeTo(null);
    frame.add(panel = new JPanel());
    panel.add(button = new JButton("Click me!"));
    frame.setVisible(true);
    // Add the instance of the ActionListener implementer here
  }

  public static void main(String[] args) {
    new HelloActionListenerInterface();
  }
}

// Create your own class that implements ActionListener here
```

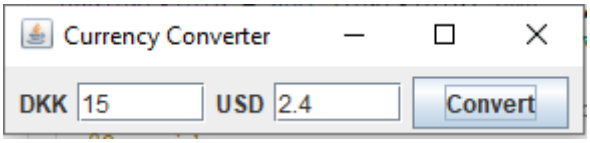
Exercise 2-7: Currency converter

Create a program that can convert currencies from Danish Kroner (DKK) to US Dollars (USD). You do not have to validate user input, etc. - just focus on creating the GUI part.

Conversion rule:

1 DKK = 0,16 USD

Your GUI could look something like this:



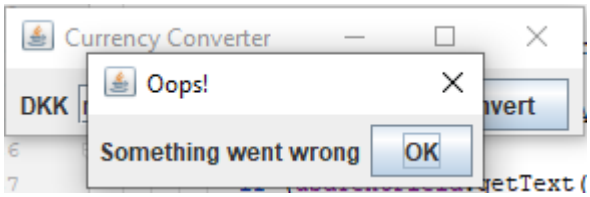
If you have time, you can improve the program so that it detects whether one of the fields is empty, and then converts the opposite value - using the same button. So, if you user types a number in DKK, the program converts to USD - and vice versa. Again, there is no need to validate user input, etc.

For this part, you will also need to know:

1 USD = 1.19 DKK

Advanced (we'll cover this next time)

Make sure that if something goes wrong, your program doesn't crash - it just reacts like this:



When the user clicks OK, they get a new chance in the main window.