Java Swing for cool GUIs

Module 1

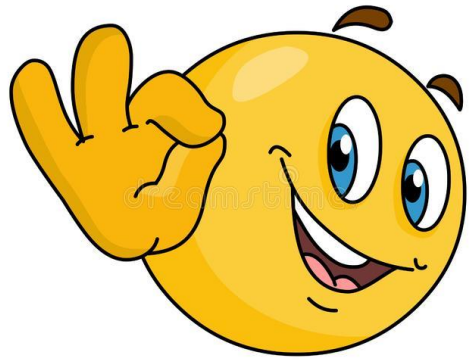
Patrick Agergaard, paag@kea.dk

DISCLAIMER

- The course is basic and practical.
- No threading.
- No MVC, design patterns or GRASP.
- The course is only on Swing - it's up to you to utilize it ☺

Learning goals

- Background: Why GUI and Swing?
- Hello World (minimum JFrame program)
- Centering frame on screen
- pack()
- JTextLabel
- JTextField




• Semester
• Class
• Fun fact ☺

Background: Why GUI and Swing?


AWT
Java
ZIUBY


JAVA
SWING


JavaFx

Hello World
(minimum JFrame program)

Hello, world
(Minimum JFrame program)

```
import javax.swing.JFrame;

public class HelloWorld {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Hello, world");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

Hello, world - centered

```
import javax.swing.JFrame;

public class HelloWorldCentered {
    public static void main(String[] args) {
      JFrame frame = new JFrame("Hello, world");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setLocationRelativeTo(null); // Center frame on screen
      frame.setVisible(true);
    }
}
```

Hello, world - with component

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

public class HelloWorldWithComponent {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Hello, world");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Hello, world - with component!");
    frame.add(label, BorderLayout.CENTER);
    frame.setLocationRelativeTo(null); // center on screen
    frame.pack();
    frame.setVisible(true);
  }
}
```

Use a FlowLayoutManager (+ another JLabel)

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.FlowLayout;

public class HelloWorldWithComponent {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Hello, world");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new FlowLayout());
    JLabel label = new JLabel("Hello, world - with component!");
    JLabel label2 = new JLabel("Label 2");
    frame.add(label);
    frame.add(label2);
    frame.setLocationRelativeTo(null); // center on screen
    frame.pack();
    frame.setVisible(true);
  }
}
```

Use a JPanel (by default, do this!)

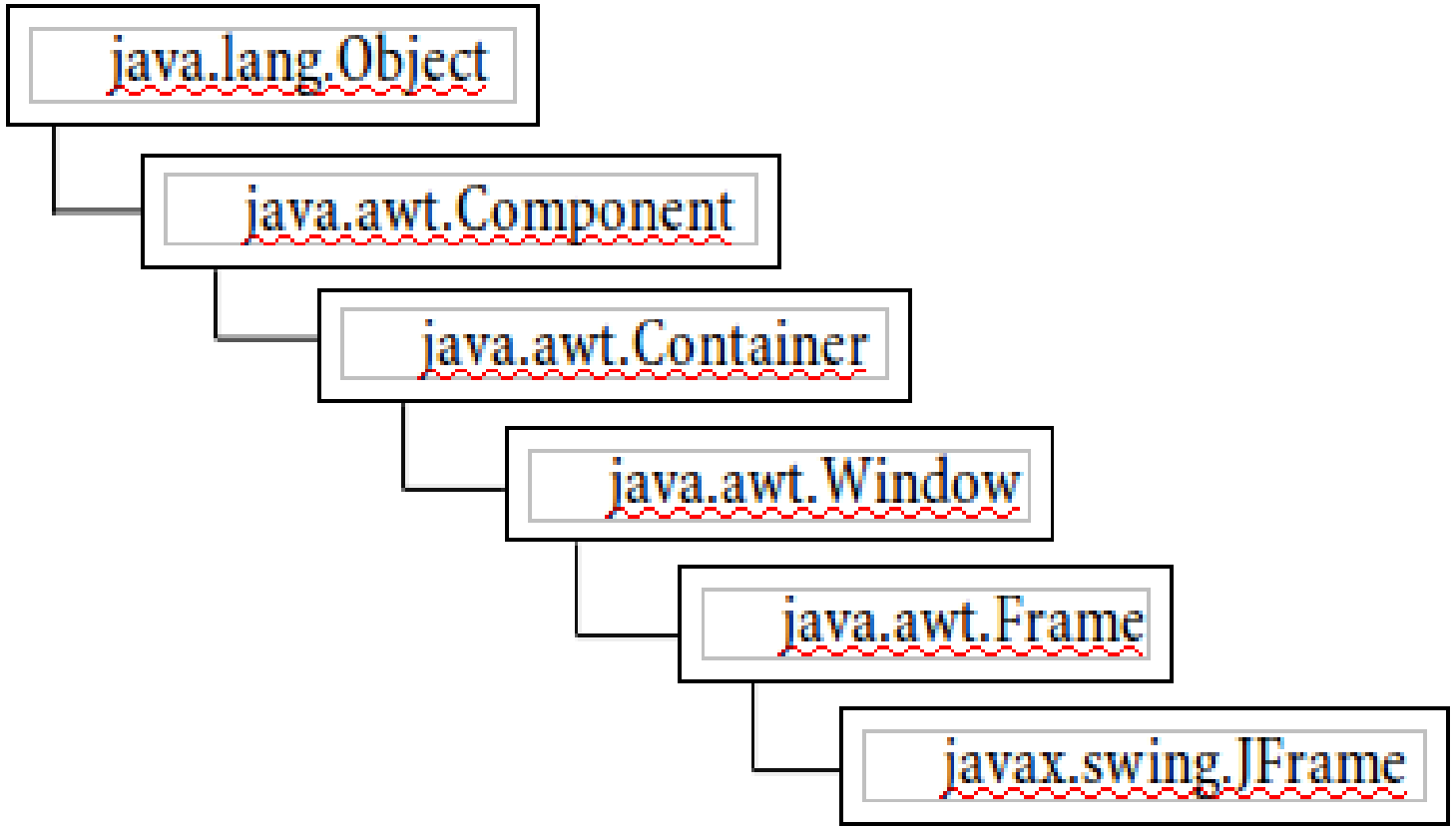
```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.FlowLayout;
import javax.swing.JPanel;

public class UseAJPanel {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Use a JPanel");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel panel = new JPanel();
    // panel.setLayout(new FlowLayout());
    frame.add(panel);
    JLabel label = new JLabel("Hello, world - with component!");
    JLabel label2 = new JLabel("Label 2");
    panel.add(label);
    panel.add(label2);
    frame.setLocationRelativeTo(null); // center on screen
    frame.pack();
    frame.setVisible(true);
  }
}
```

YOU WOULDN'T LET THIS
HAPPEN TO YOUR PHONE.
DON'T LET IT HAPPEN TO YOU EITHER.
SELF-CARE IS A PRIORITY
NOT A LUXURY.


```
class NeverExtendJFrame extends JFrame { }
```

```
class AlwaysInstantiateJFrame {
  private JFrame frame;
}
```


java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Window
java.awt.Frame
javax.swing.JFrame

Why bother using JPanel?

- To group components together.
- To better organize your components.
- To enable us to use ***multiple layouts*** and combine their effects. (For example a GridLayout for number pad, a CardLayout for display panel where you can switch drawings).
- To break down a large problem into sub-problems. So instead of implementing everything in one big frame, you can concentrate implementing the properties of each individual panels (such as layout, dimension, background image, background color..etc)
- For reusability if you have customized panels. A common panel can be used by different classes.
- For easy debugging, since you can test each panel individually.
- For scalability.
- For maintainability.

https://stackoverflow.com/questions/36101720/why-i-should-use-a-jpanel

Some JPanel options

Set background color:

```
panel.setBackground(Color.CYAN);
```

Set transparent background:

```
panel.setOpaque(false); // make transparent background (default: opaque)
```

Set a raised bevel border:

```
panel.setBorder(BorderFactory.createBevelBorder(BevelBorder.RAISED));
```

Set an etched border:

```
panel.setBorder(BorderFactory.createEtchedBorder());
```

Set a line border:

```
panel.setBorder(BorderFactory.createLineBorder(Color.RED));
```

Set a titled and etched border:

```
panel.setBorder(BorderFactory.createTitledBorder(
   BorderFactory.createEtchedBorder(), "Login Panel"));
```

Other components we can use

https://www.studytonight.com/java/java-swing-components.php

https://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

Window closing behaviors

Which action to execute when the user clicks on the frame's close button:

Do nothing (default):

frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

Hide the frame:

frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

In this case, the frame becomes invisible. To show it again, call:

frame.setVisible(true);

Dispose the frame:

frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

In this case, the frame is removed and any resources used by the frame are freed. If the frame is the last displayable window on screen, the JVM may terminate.

Exit the program (JVM terminates):

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

https://www.codejava.net/java-se/swing/jframe-basic-tutorial-and-examples

Menu bar

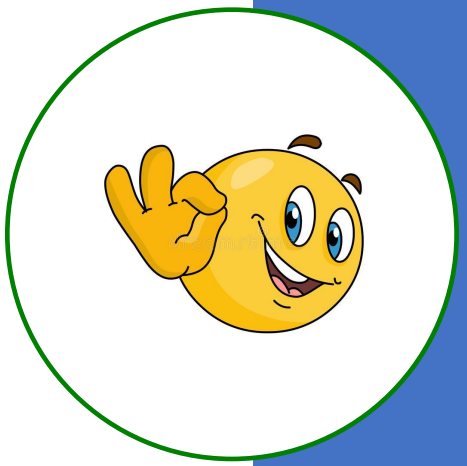
Adding a menu bar:

```
1   import javax.swing.JFrame;
2   import javax.swing.JMenuBar;
3   import javax.swing.JMenu;
4   import javax.swing.JMenuItem;
5
6   public class MenuBar {
7     public static void main(String[] args) {
8       JFrame frame = new JFrame( title: "Menu bar");
9       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10
11      // Menu:
12      JMenuBar menuBar = new JMenuBar();
13      JMenu menuFile = new JMenu( s: "File");
14      JMenuItem menuItemExit = new JMenuItem( text: "Exit");
15      menuFile.add(menuItemExit);
16      menuBar.add(menuFile); // adds menu bar to the frame
17      frame.setJMenuBar(menuBar);
18
19      frame.setVisible(true);
20    }
21  }
```

https://www.codejava.net/java-se/swing/jframe-basic-tutorial-and-examples

What was most important today?

- Understand the Hello, world program
- Adding JComponents
- We know that FlowLayout is default
- Understand why we use JPanels



Next module: Buttons and actions - yay!

QUESTIONS

- Background: Why GUI and Swing?
- Hello World (minimum JFrame program)
- Centering frame on screen
- pack()
- JTextLabel
- JTextField