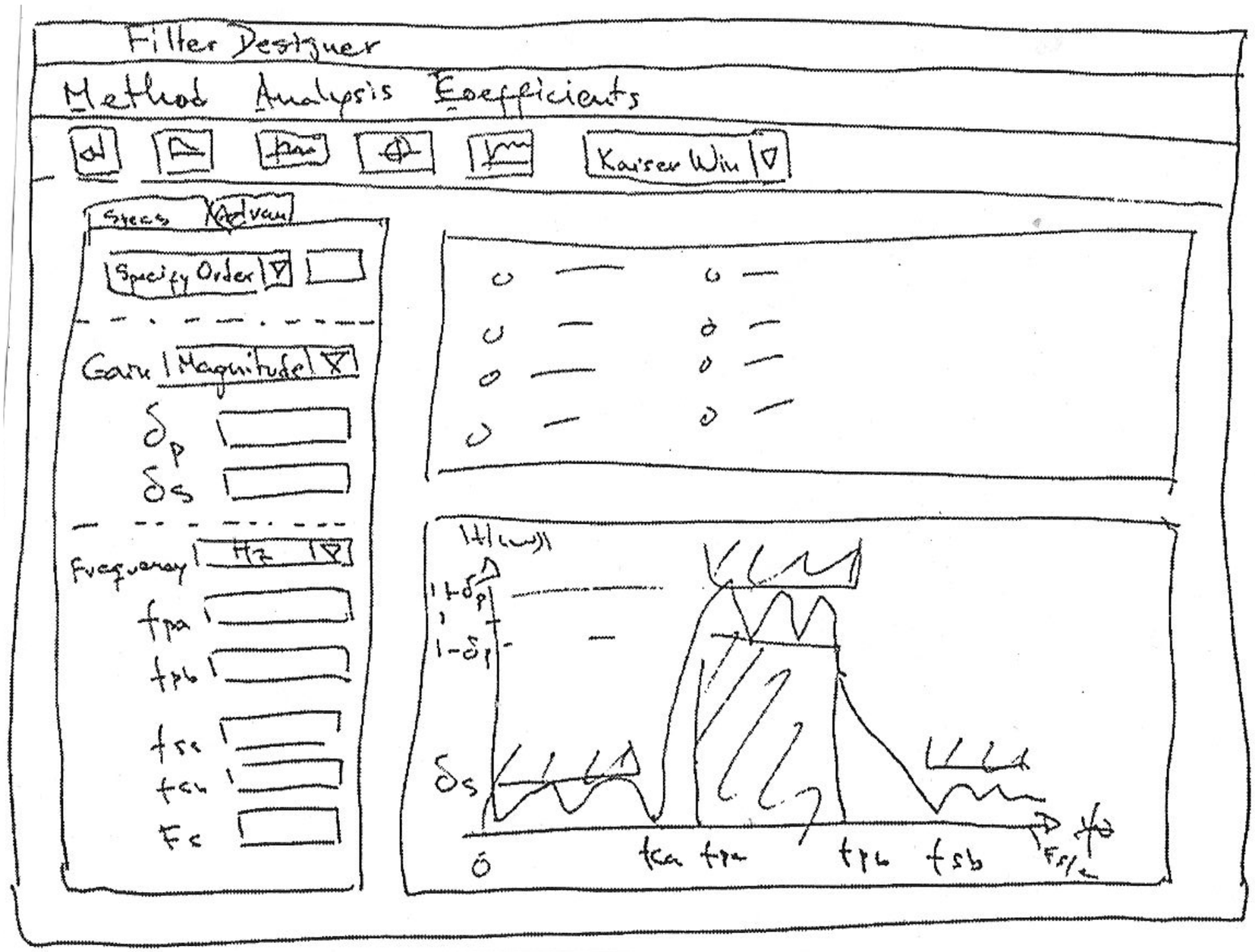
GUI Entwicklung

Sascha Just
Softwarepraktikum 2014

Paper Prototyping

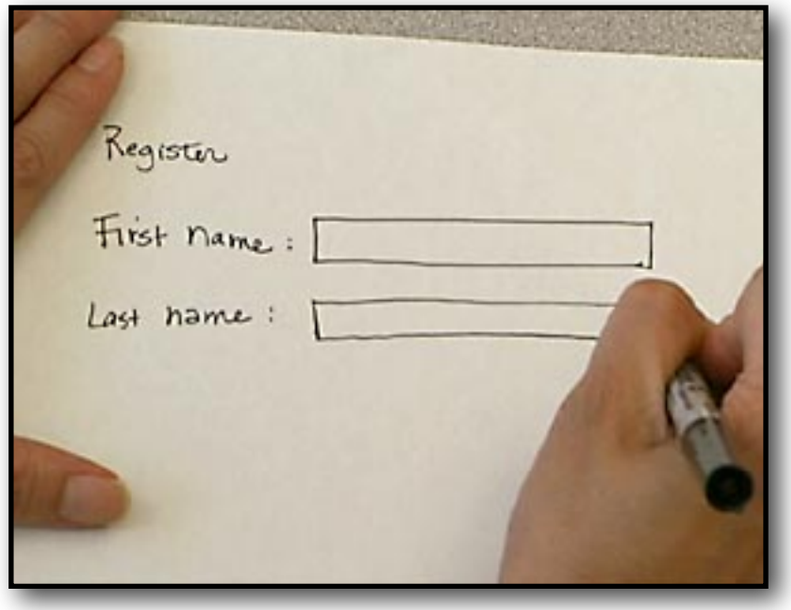
Paper Prototyping



Paper Prototyping

Credits: Nielsen Norman Group

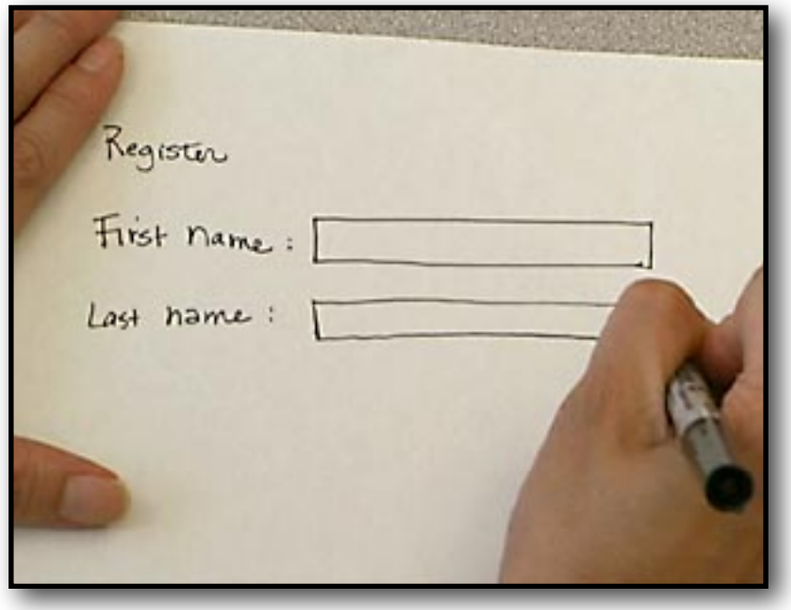
1.



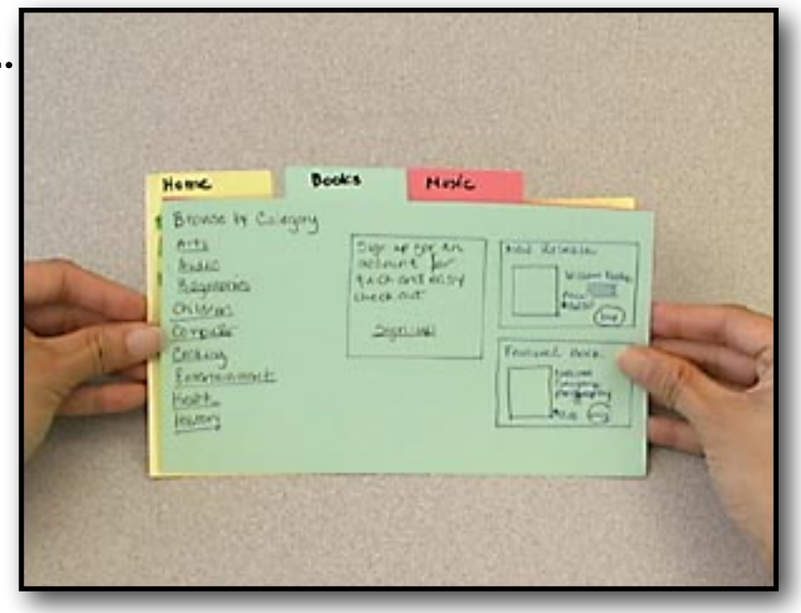
Paper Prototyping

Credits: Nielsen Norman Group

1.



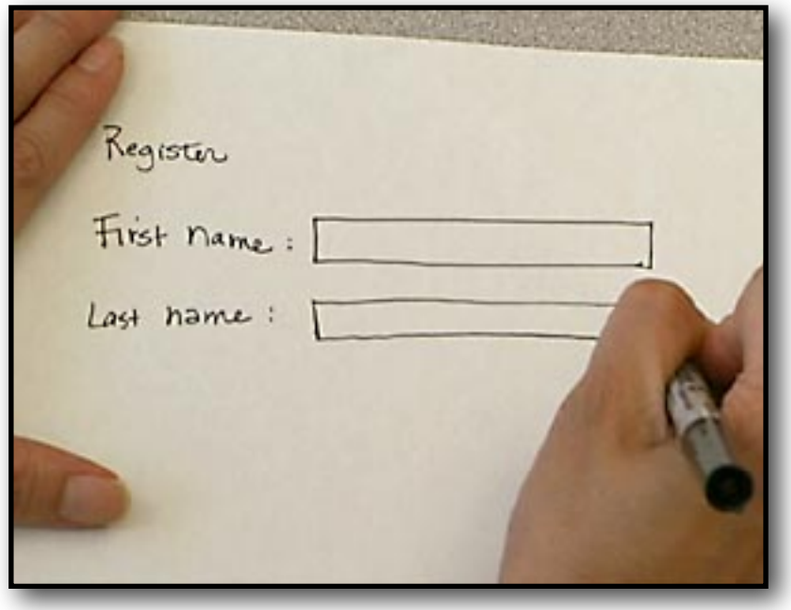
2.



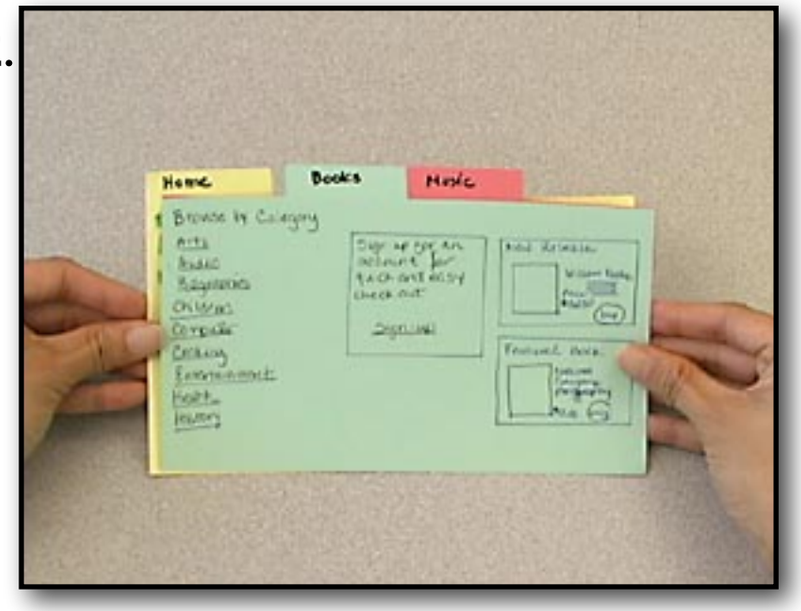
Paper Prototyping

Credits: Nielsen Norman Group

1.



2.

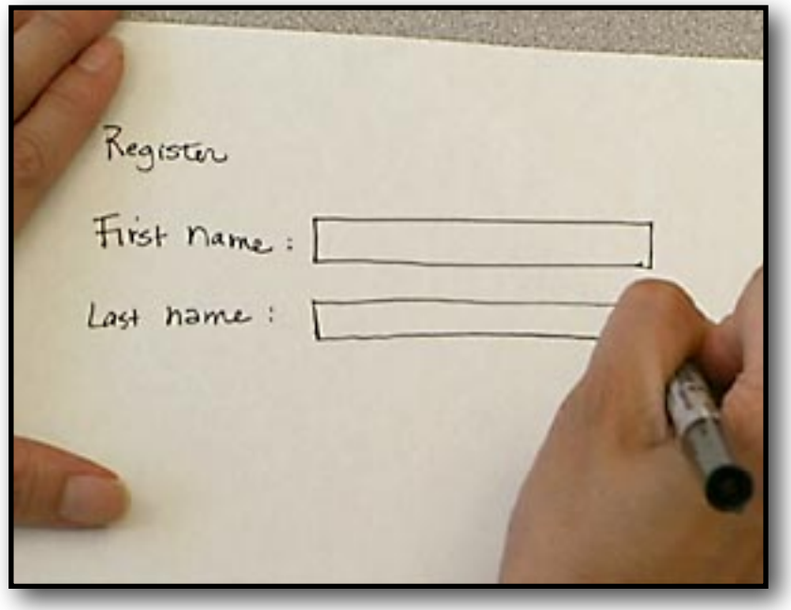


3.

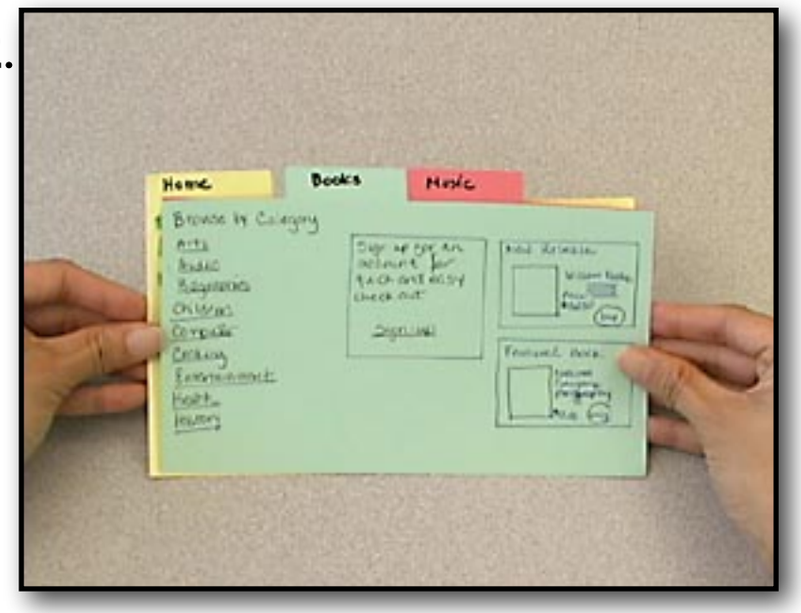
Paper Prototyping

Credits: Nielsen Norman Group

1.

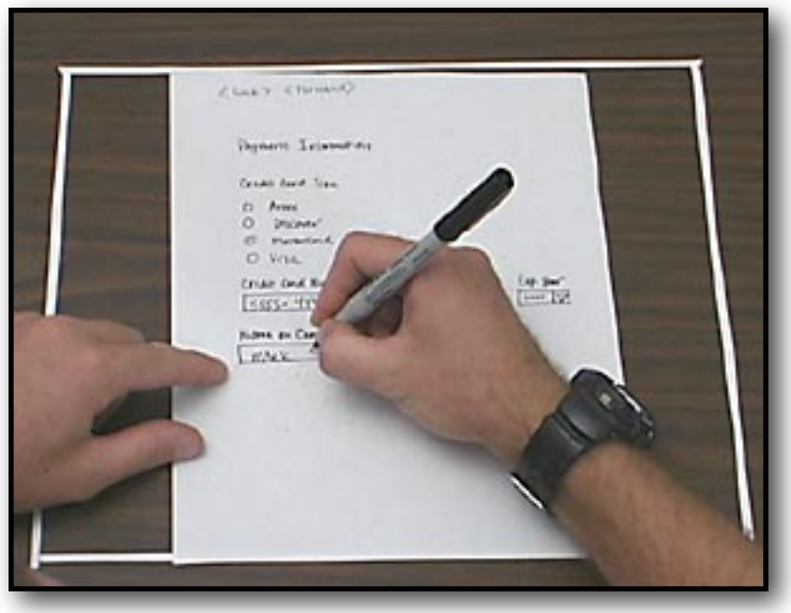


2.



3.

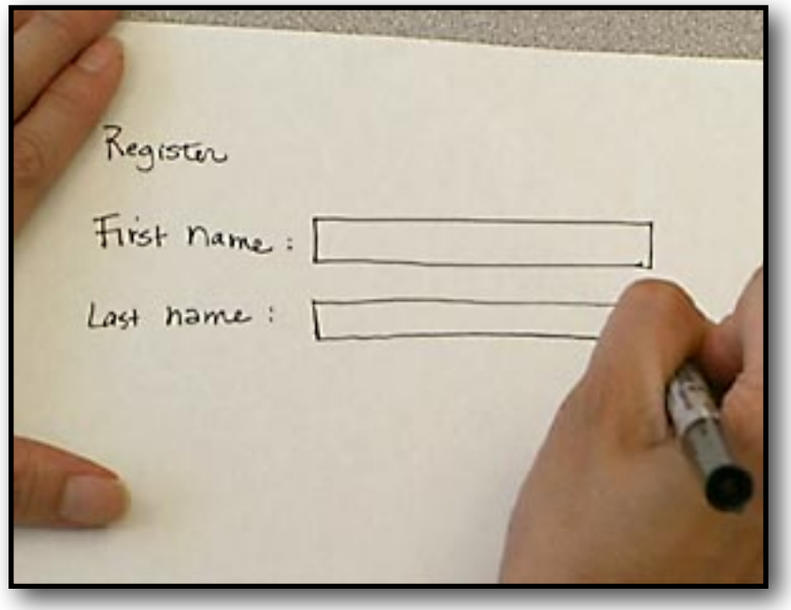
4.



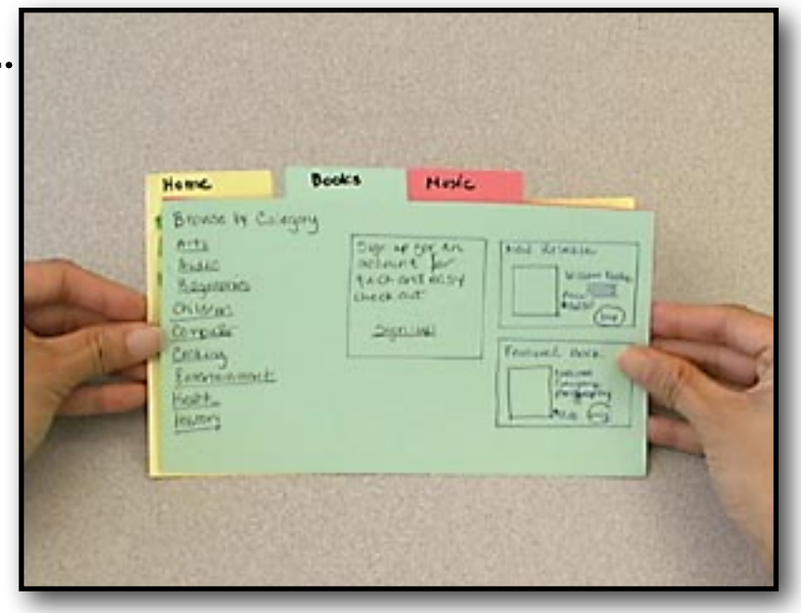
Paper Prototyping

Credits: Nielsen Norman Group

1.

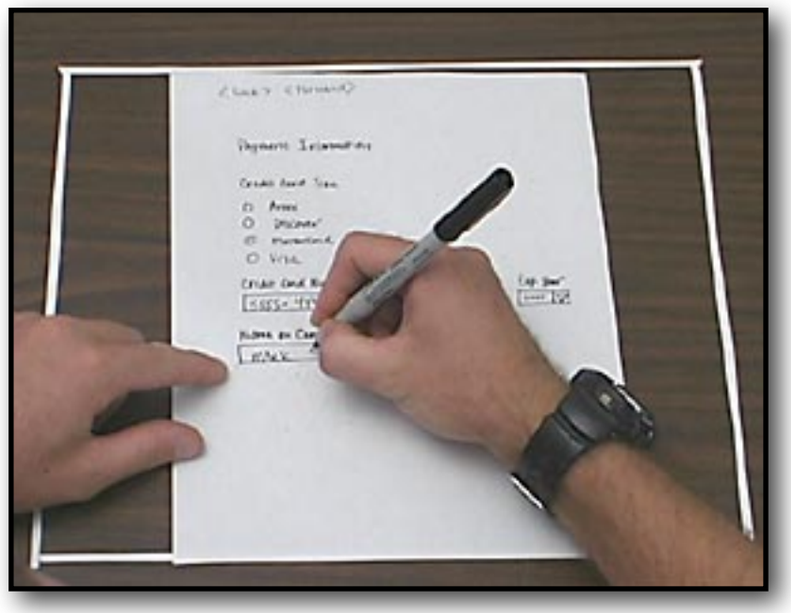


2.



3.

4.

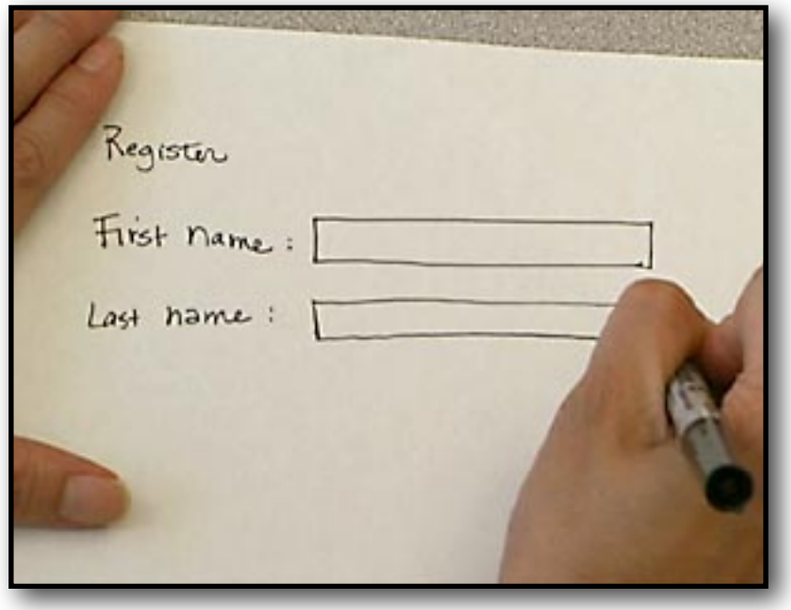


5.

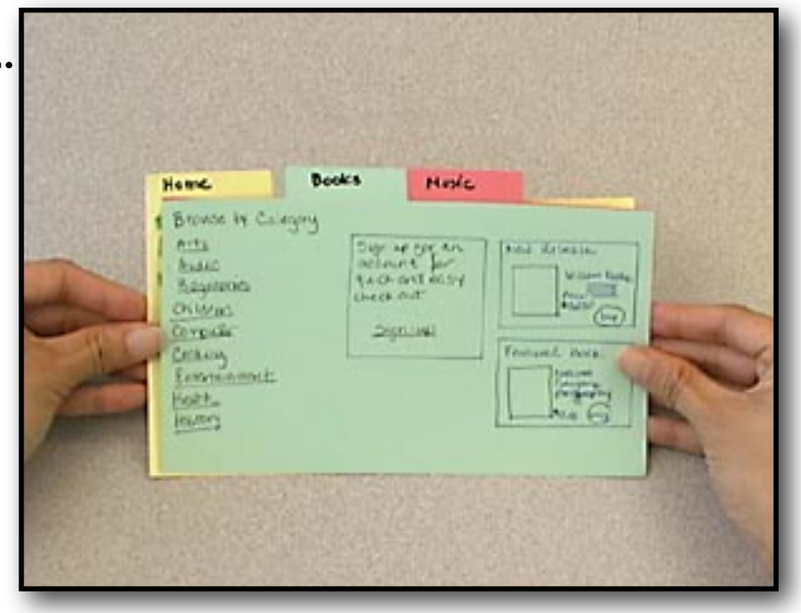
Paper Prototyping

Credits: Nielsen Norman Group

1.

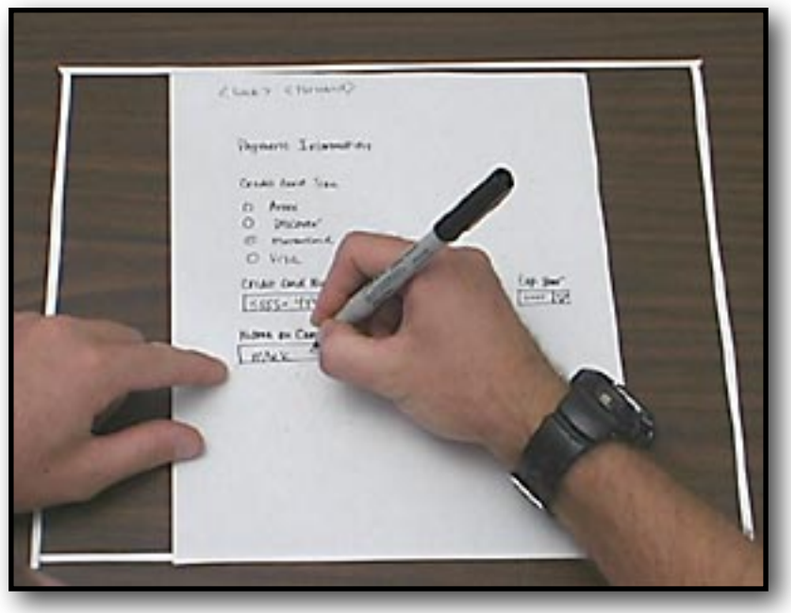


2.



3.

4.



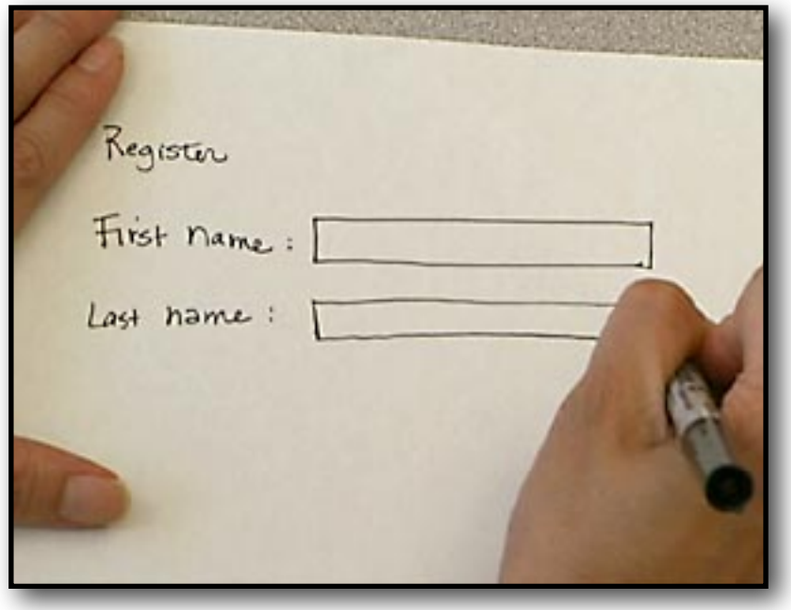
5.

6.

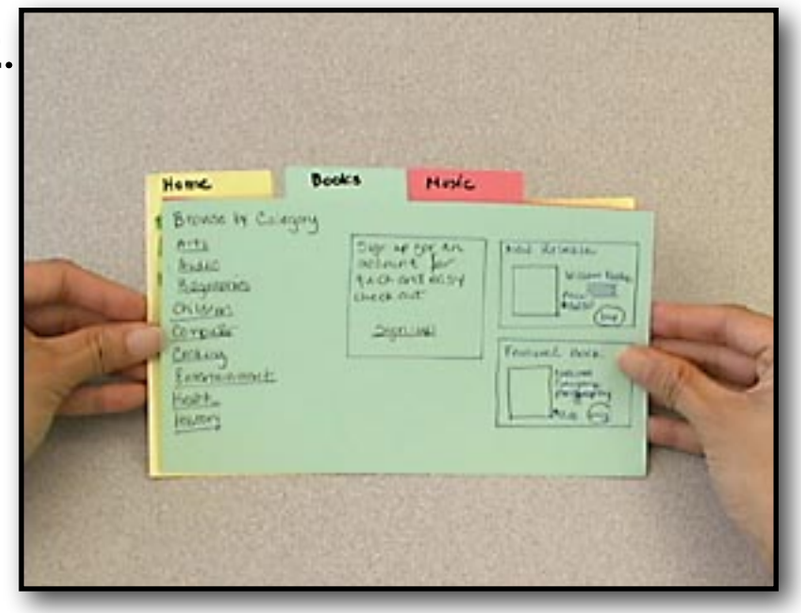
Paper Prototyping

Credits: Nielsen Norman Group

1.

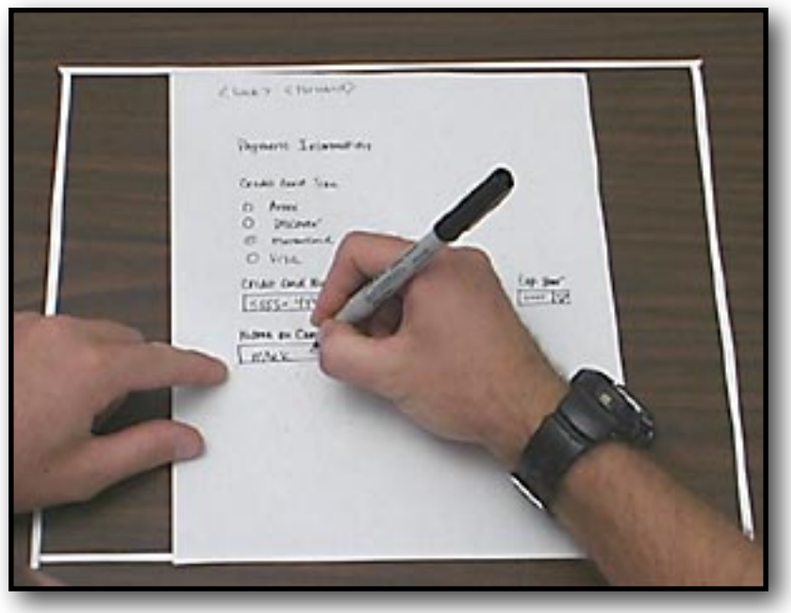


2.



3.

4.

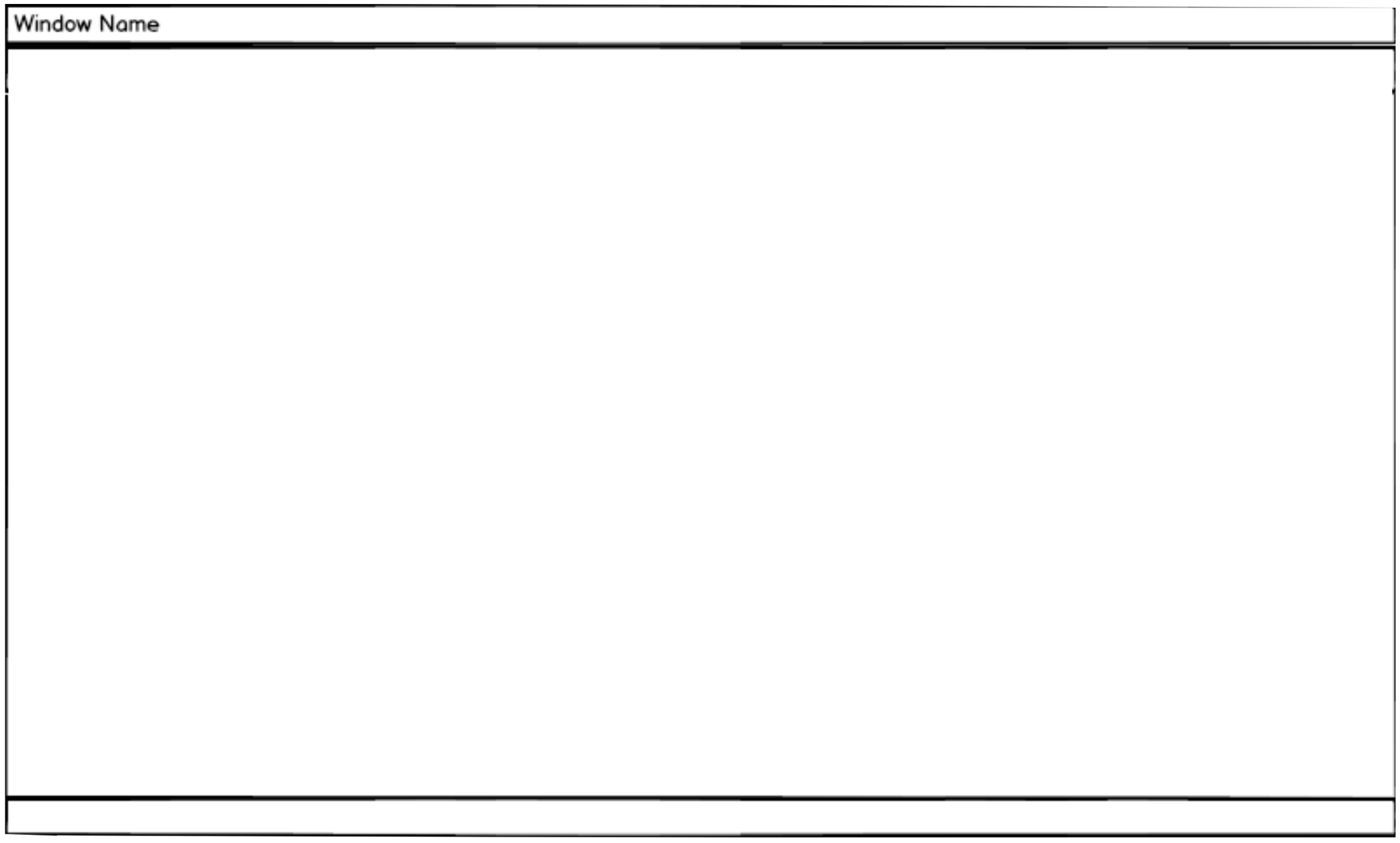


5.

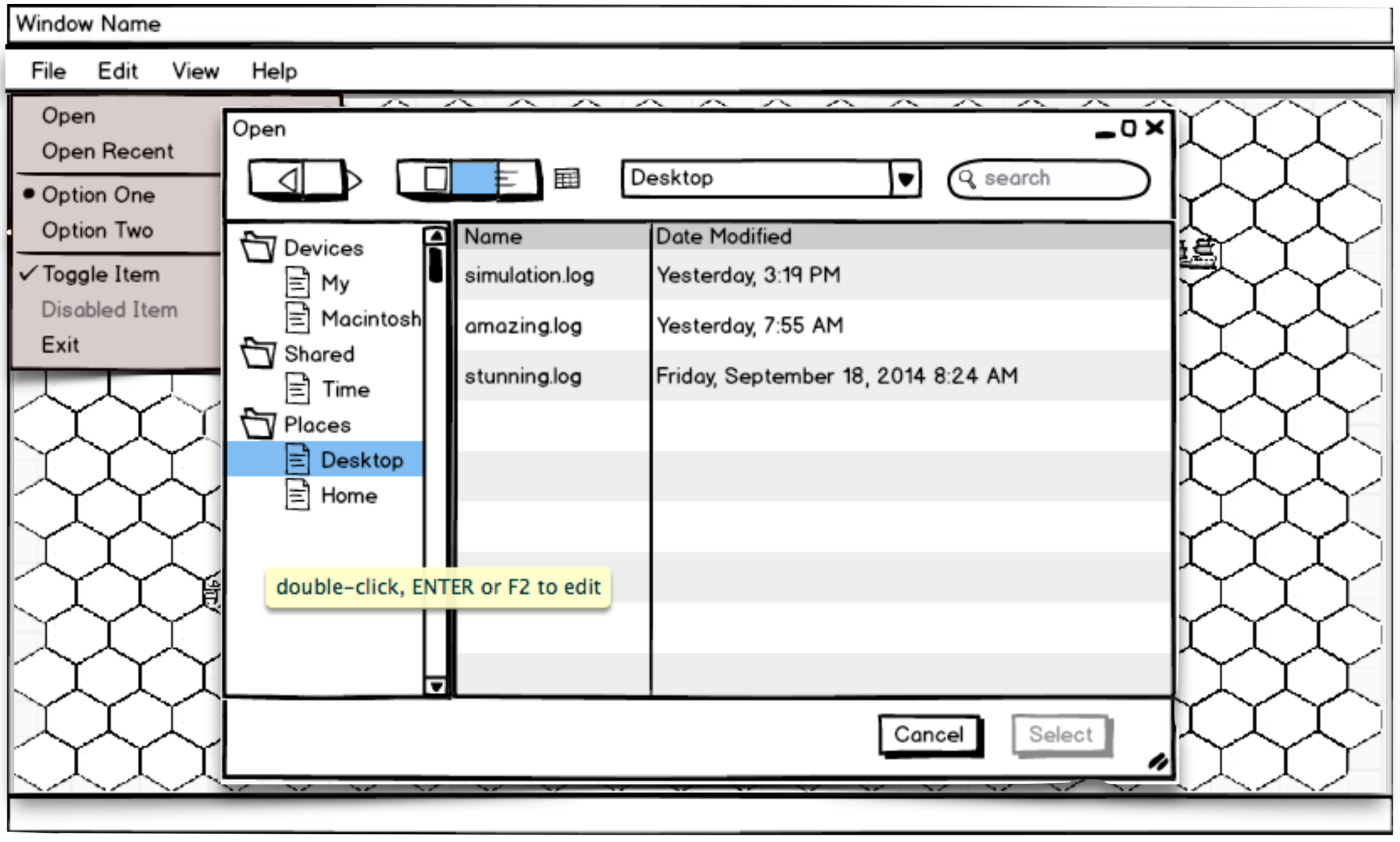
6.

7.

Wireframing



Wireframing



Vorteile

Vorteile

- Schnelles Erstellen eines Mockups ohne dazu programmieren zu müssen.

Vorteile

- Schnelles Erstellen eines Mockups ohne dazu programmieren zu müssen.
- Entdeckt frühzeitig Probleme im Design.

Vorteile

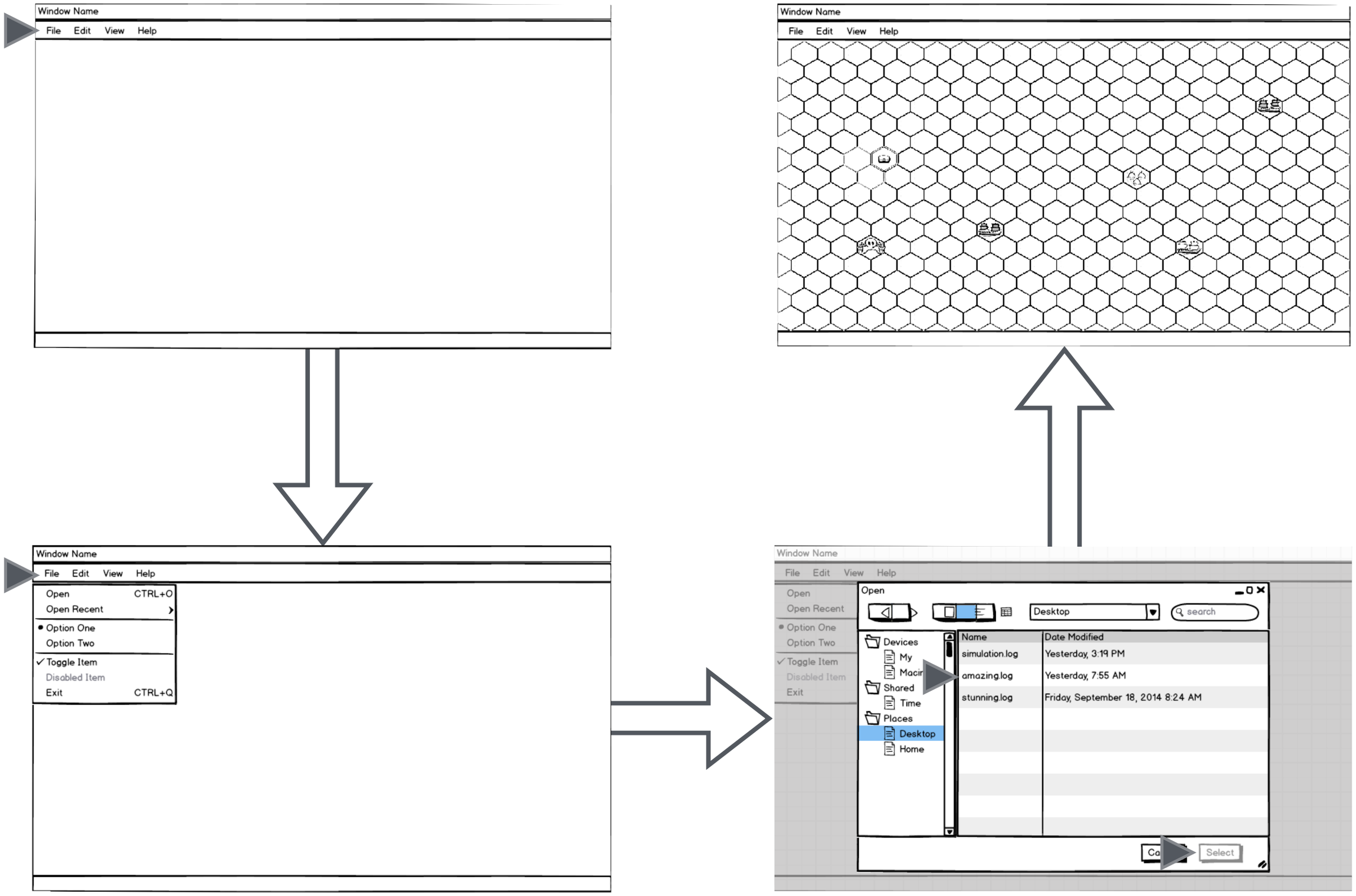
- Schnelles Erstellen eines Mockups ohne dazu programmieren zu müssen.
- Entdeckt frühzeitig Probleme im Design.
- Ermöglicht das Einholen von User-Feedback vor dem Implementieren.

Vorteile

- Schnelles Erstellen eines Mockups ohne dazu programmieren zu müssen.
- Entdeckt frühzeitig Probleme im Design.
- Ermöglicht das Einholen von User-Feedback vor dem Implementieren.
- Ermöglicht Zusammenarbeit von Teams aus mehreren Disziplinen.

Credits: Paper Prototyping

Storyboards



GUI Design Abnahme

Was wir von Ihnen sehen möchten.

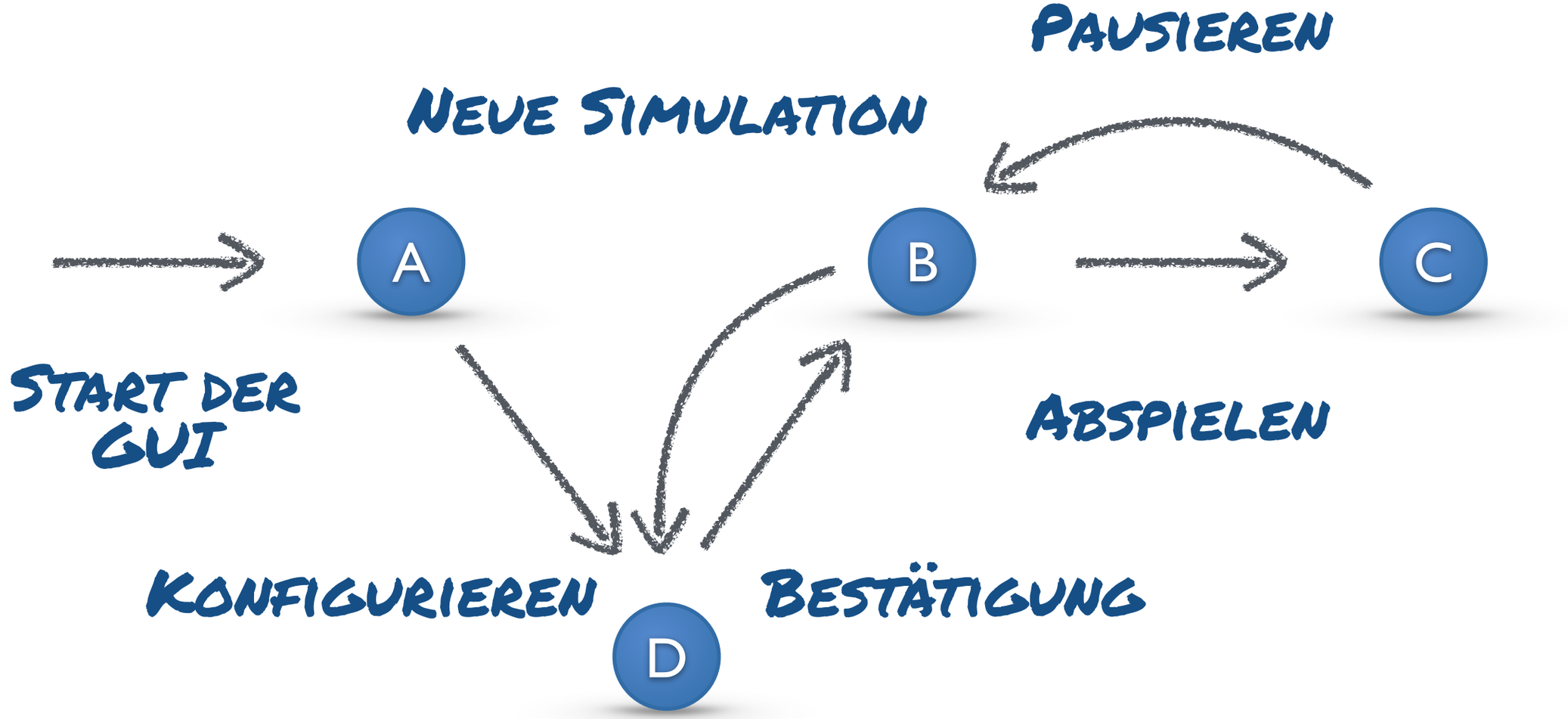
- Paper Prototype/Wireframe (Papier oder Digital) mit klar erkennbaren Elementen.
- Status-Diagramme
- Status-Beschreibungen
- Status-Übergangsbeschreibung
- Interaktionsbeschreibung: GUI zu Datenquelle (Simulator oder Logreader) und umgekehrt.

Status Diagramme

Welchen Status kann Ihre GUI annehmen?

Wie gelangt die GUI in welchen Status?

Status Diagramme



Status Beschreibungen

A. **Uninitialisiert**: Fenster: *Main*. Panels: *Main:Control*. Alle Buttons gesperrt, *Menu>File>(Open, Quit)* verfügbar.
B. **Pausiert**: Play-Button nutzbar, Spielfeld präsent in Panel *Main:Map*. Spielstände in Panel *Main:Info* angezeigt.
C. **Laufende Simulation**: …
D. **Warten auf Konfiguration**: …
E. …

Status Übergänge

Start der GUI: Erstellung des Hauptfensters (*Main*) mit Panel *Main:Control*.

Konfiguration: Konfigurieren-Button öffnet *Dialog:Config*

Bestätigung: *Dialog:Config:OK*-Button + gültige Einstellungen schließt Dialog und erstellen Panel *Main:Main* und *Main:Info*.

Laufende Simulation: Drücken des *Main:Control:Play*-Buttons startet Simulation und setzt Pause-Symbol für *Main:Control:Play*-Button.

…

Interaktion mit Datenquelle

In Ihrem Fall zwei Möglichkeiten:

Simulator

Log Reader

Interaktion mit Simulator

GUI zu Simulator

Main:Control:Play#click	startet neuen Simulator-Thread
Main:Control:Pause#click	signalisiert dem Simulator-Thread zu warten
Main:Control:Turn#change	signalisiert dem Simulator-Thread Daten ab der angegebenen Runde zu senden.

Interaktion mit Simulator

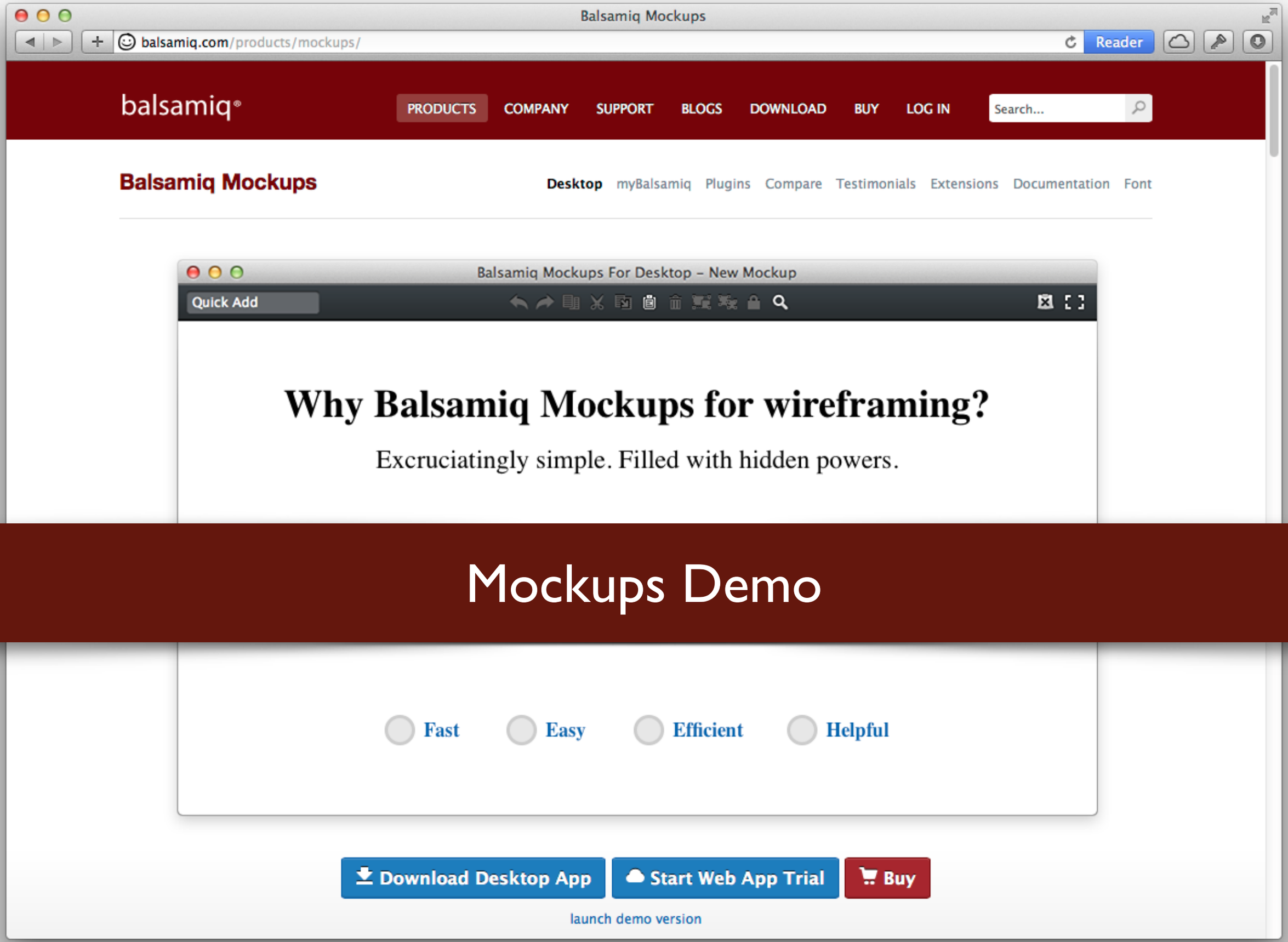
Simulator zu GUI

Simulation bereit

Setzt GUI in Pausiert-Status (freigeben der Elemente: Play, …)

Ende der Simulation

Setzt GUI in End-Status und veranlasst Anzeigen des Punktestands.



Mockups Demo

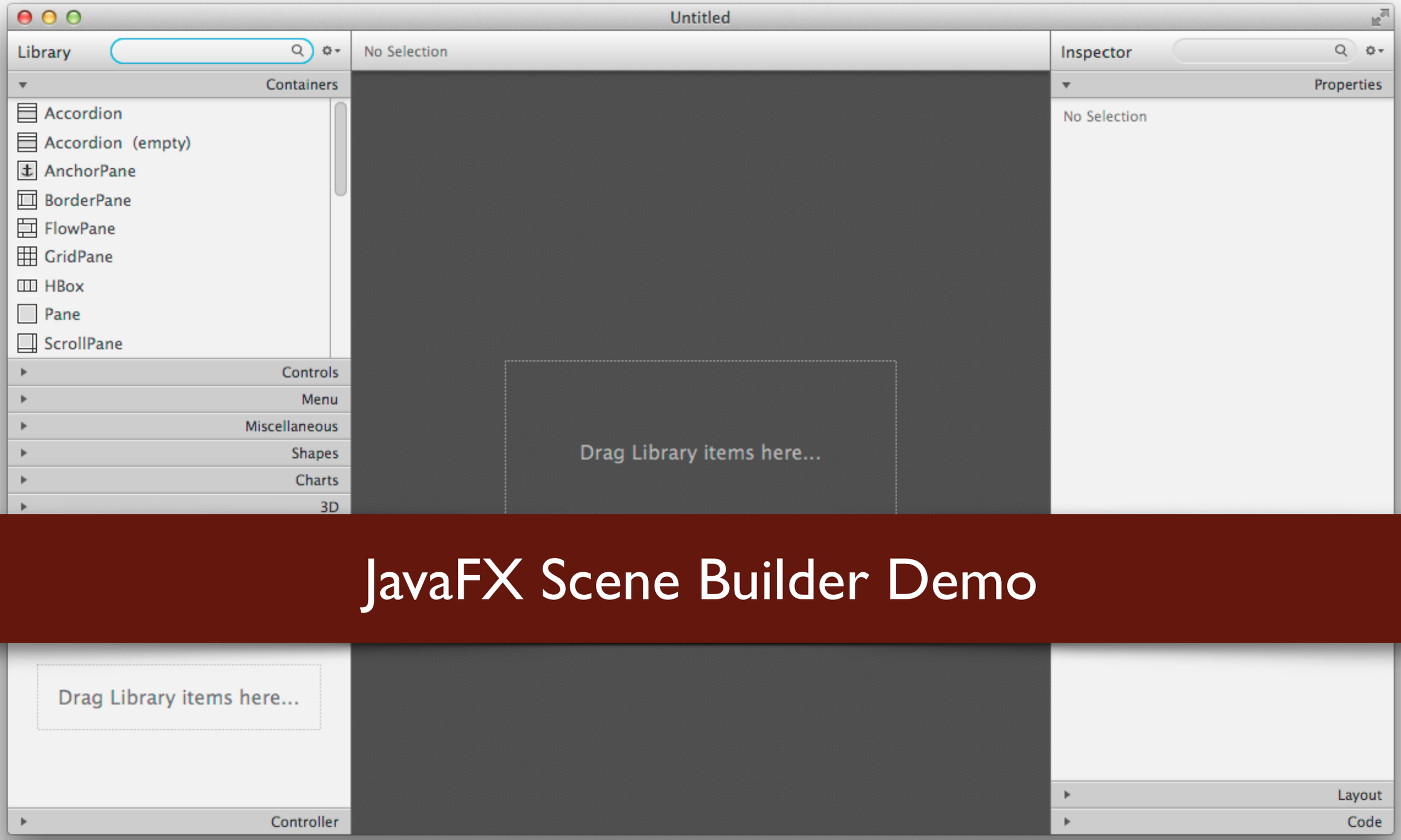
GUI Frameworks

Swing **JavaFX** GWT

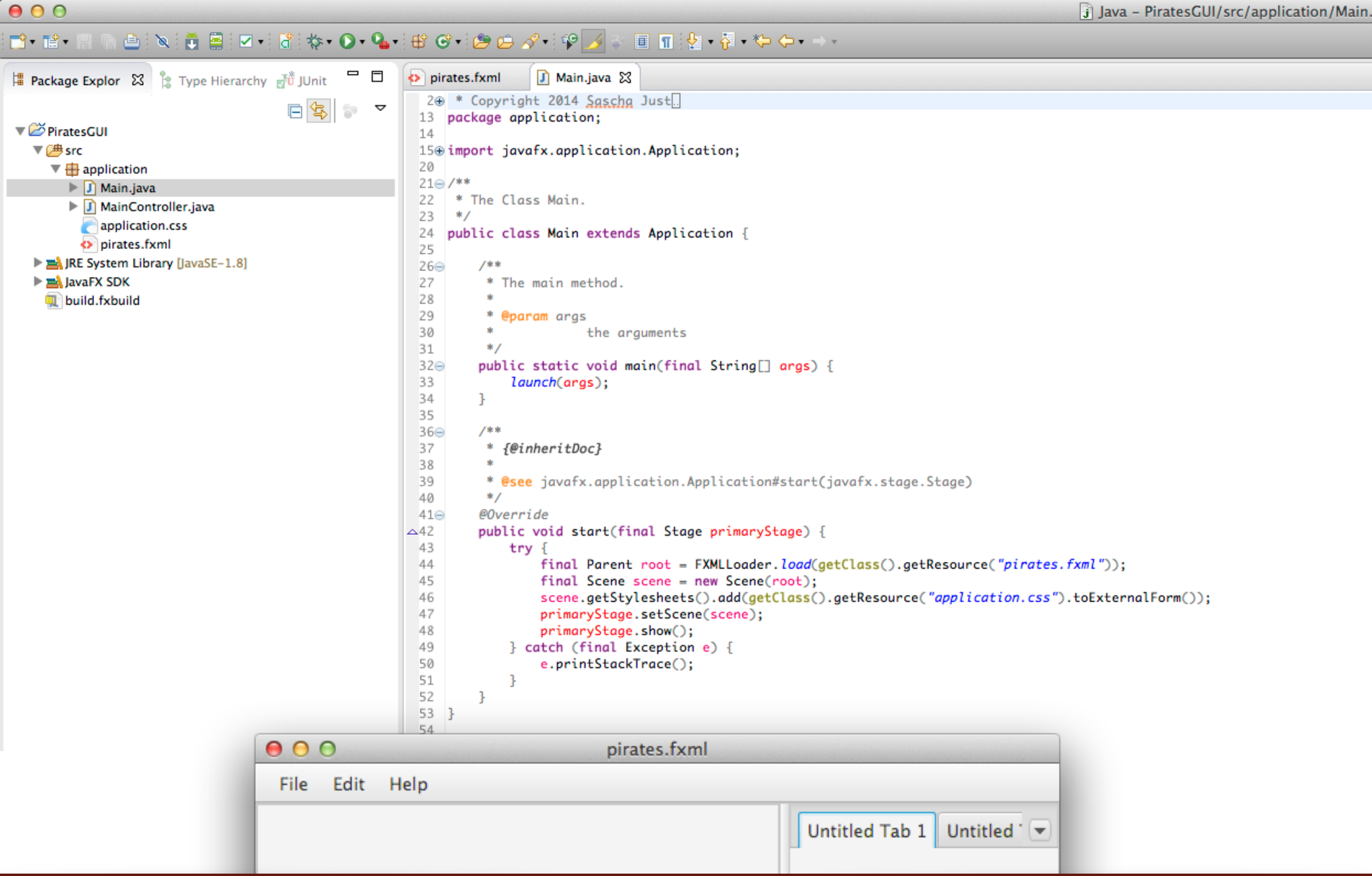
…

http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

Paper Prototype zu GUI.



JavaFX Scene Builder Demo



FXML in Eclipse verknüpfen: Demo

JavaFX Links

http://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm

http://docs.oracle.com/javase/8/javafx/get-started-tutorial/get_start_apps.htm#JFXST804

https://www.youtube.com/watch?v=ij0HwRAlCmo

http://code.makery.ch/java/javafx-8-tutorial-intro/

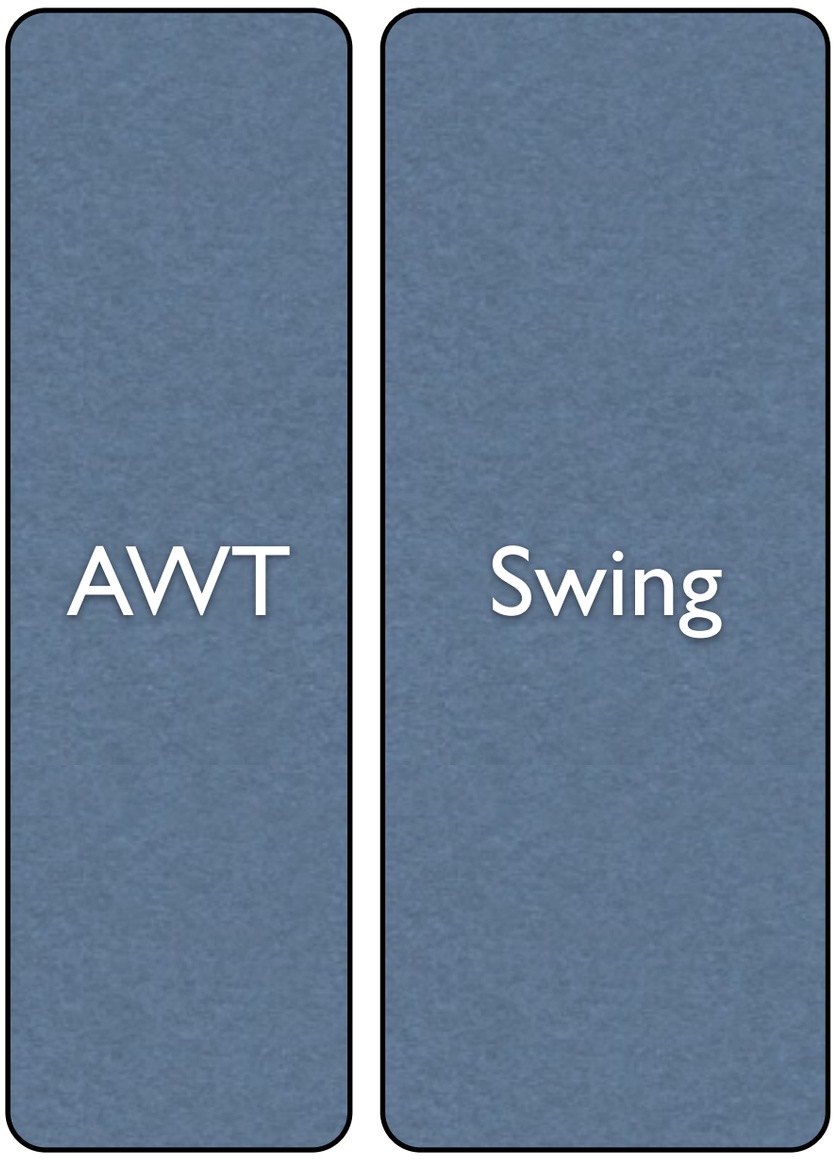
Swing Links

http://docs.oracle.com/javase/tutorial/uiswing/start/index.html

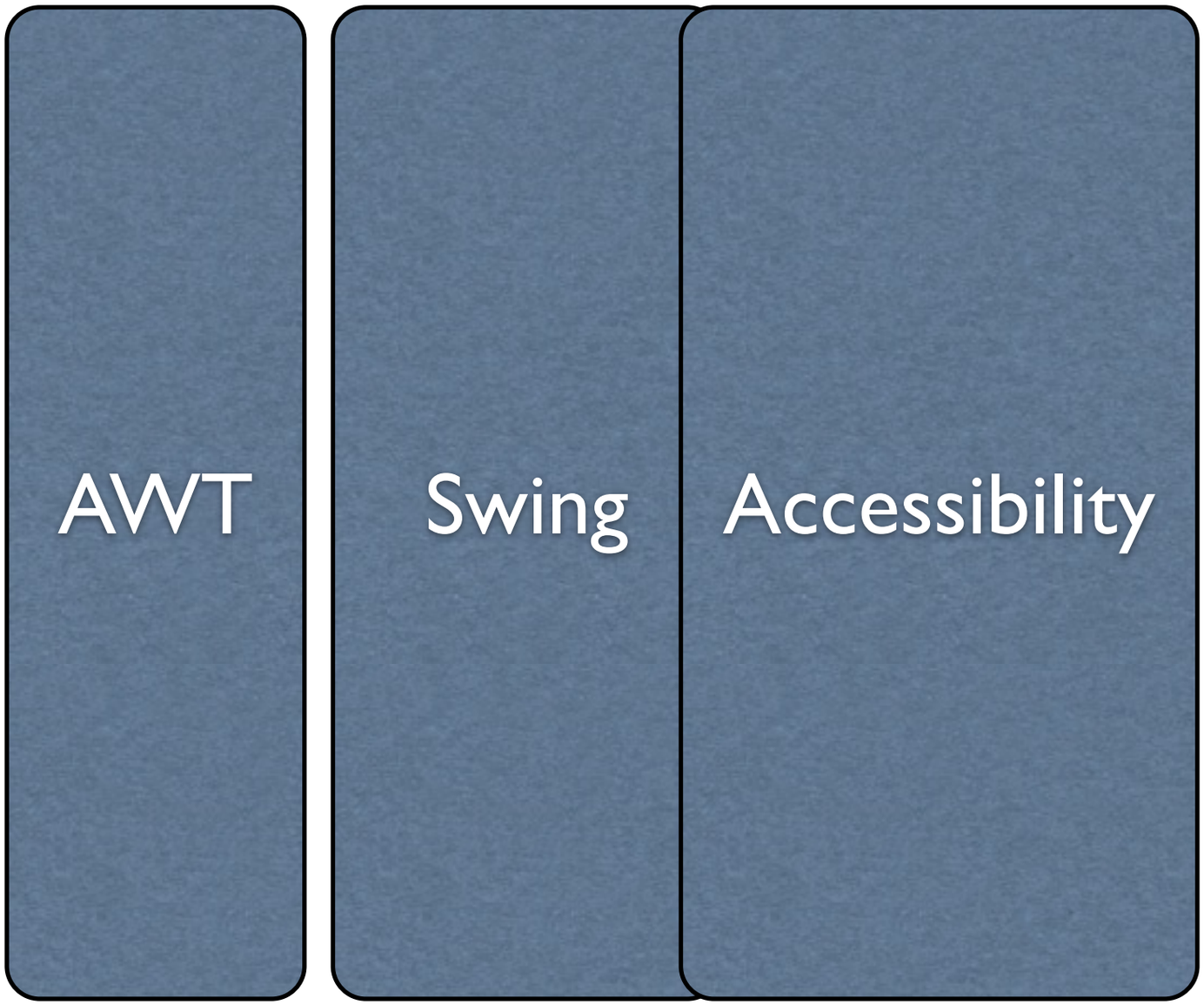
Java GUI Toolkits

Java GUI Toolkits

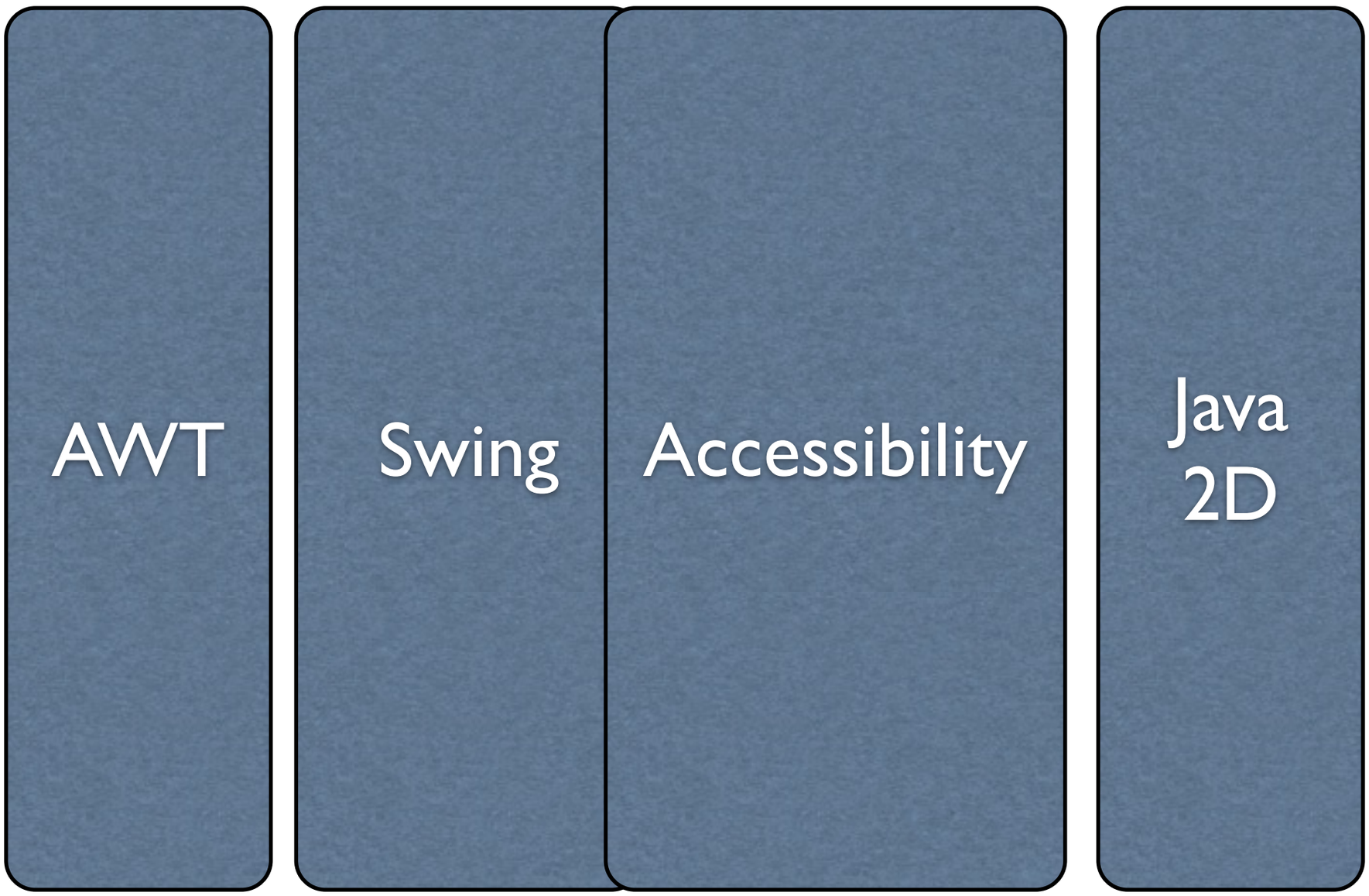
Java GUI Toolkits



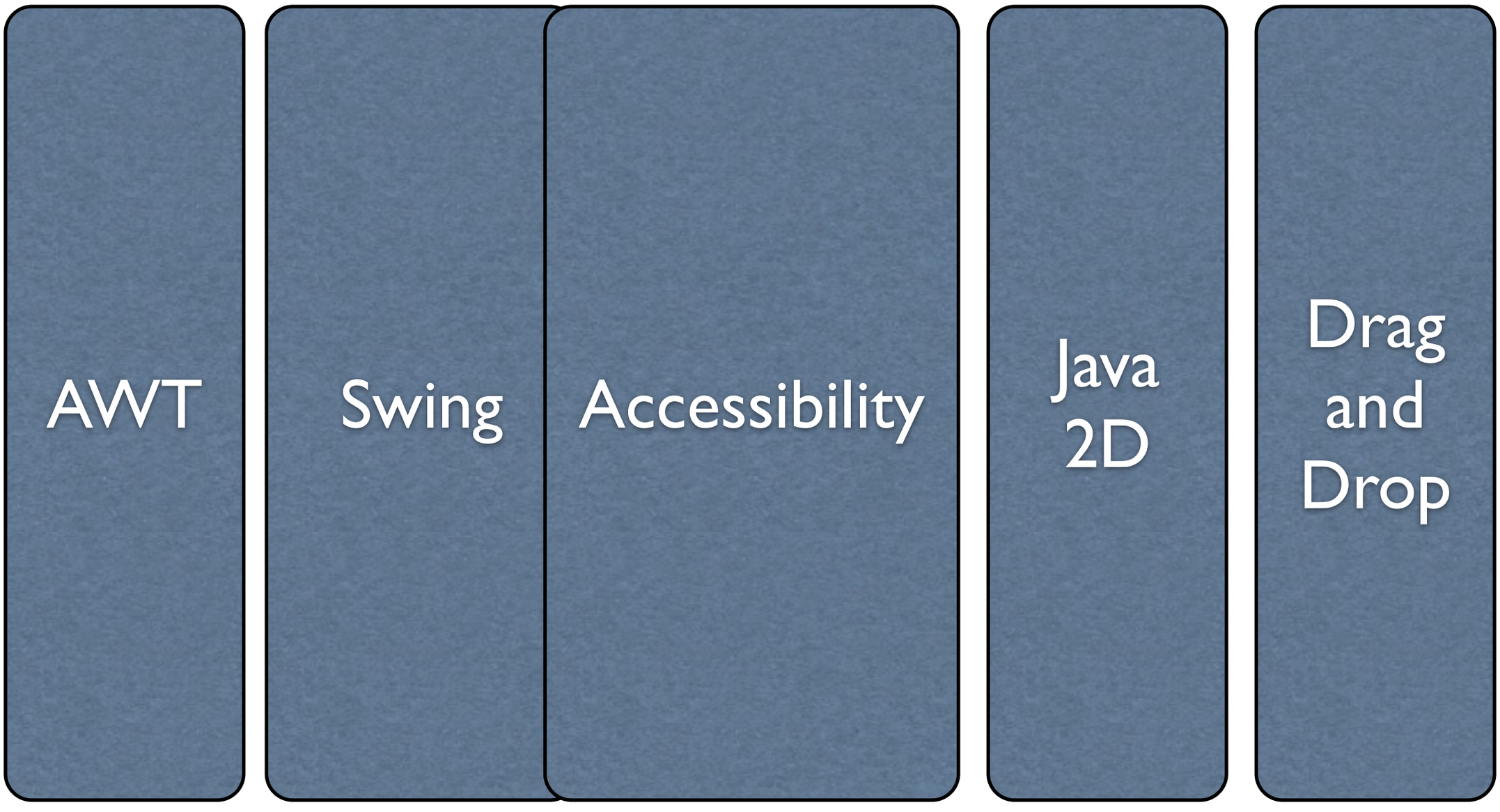
Java GUI Toolkits



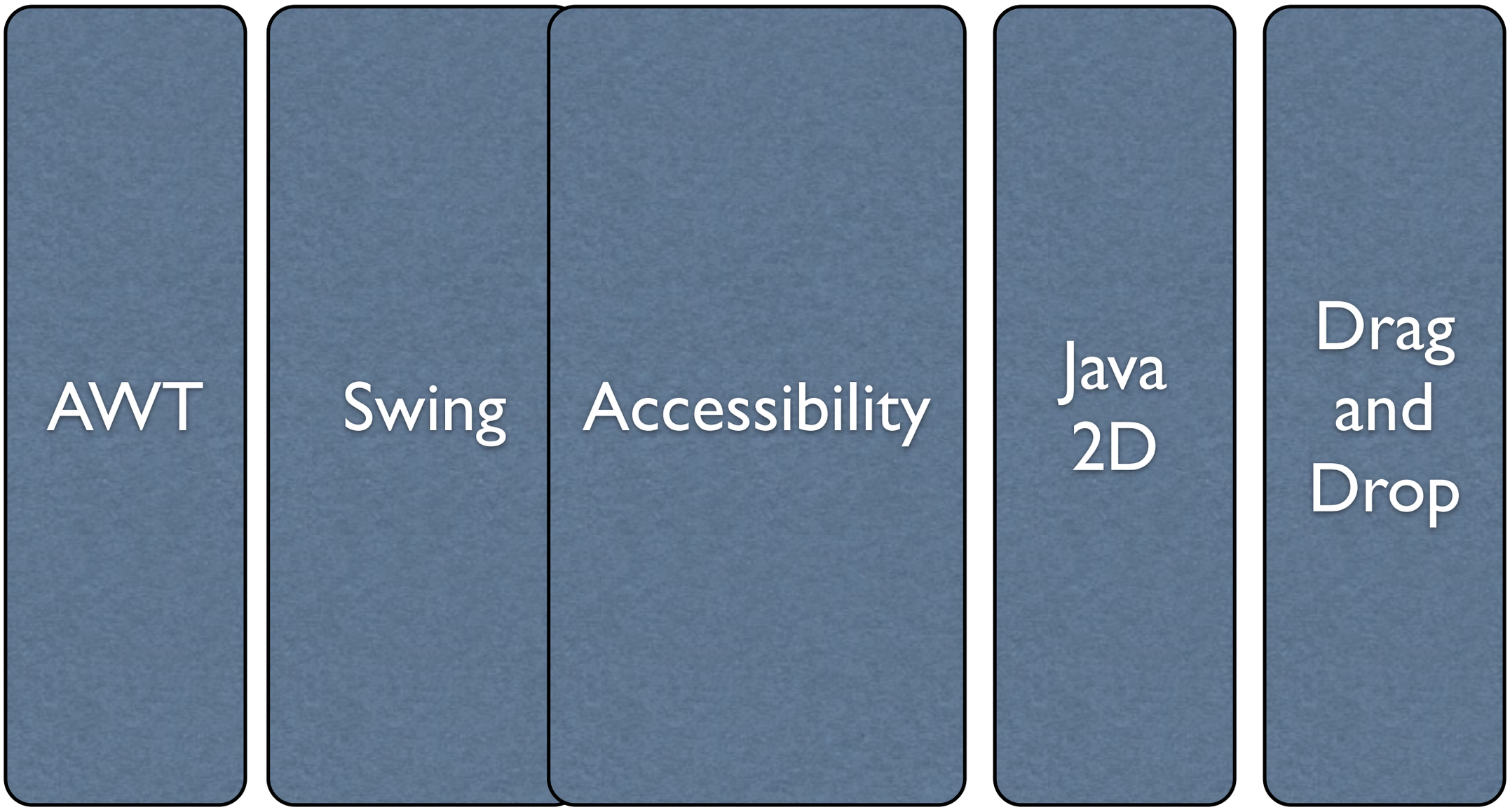
Java GUI Toolkits



Java GUI Toolkits

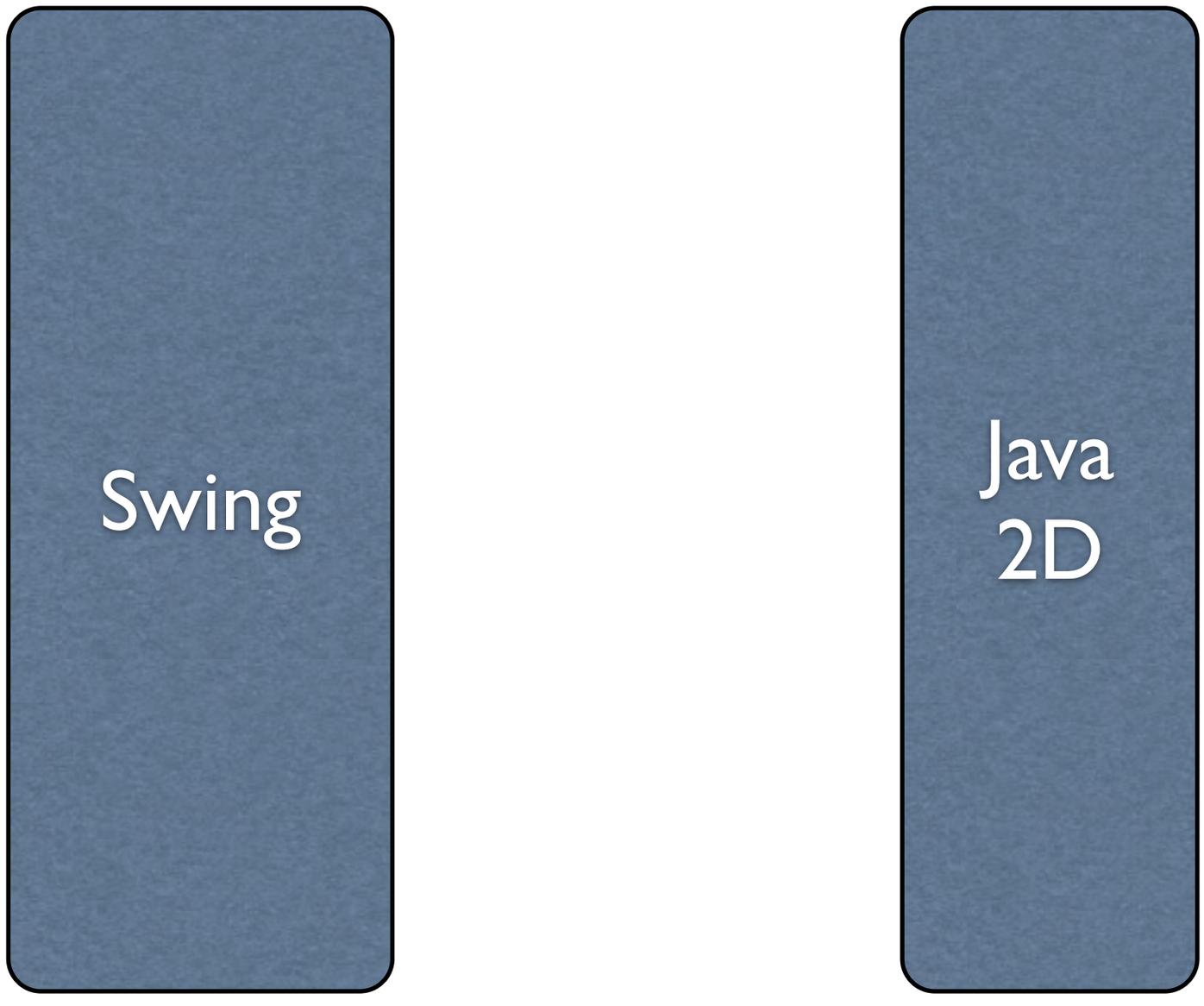


Java GUI Toolkits



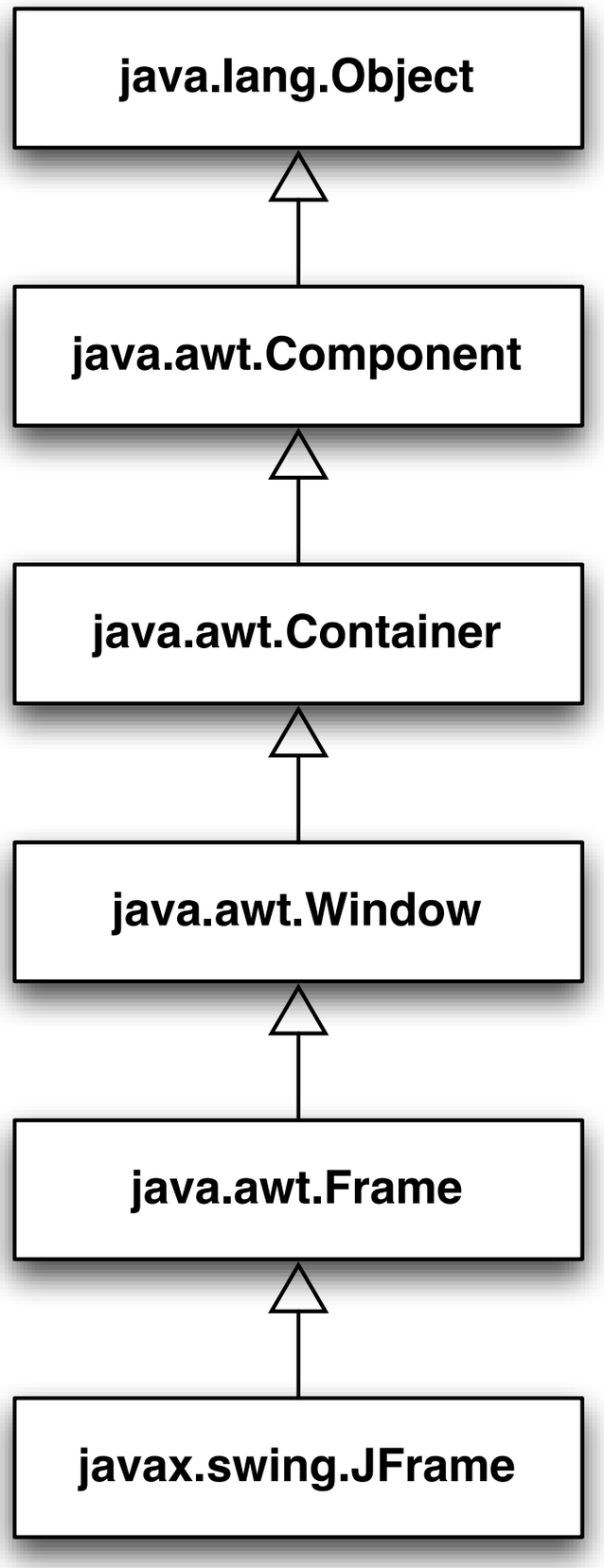
+ *Standard Widget Toolkit (SWT)* • + *Qt Jambi* • …

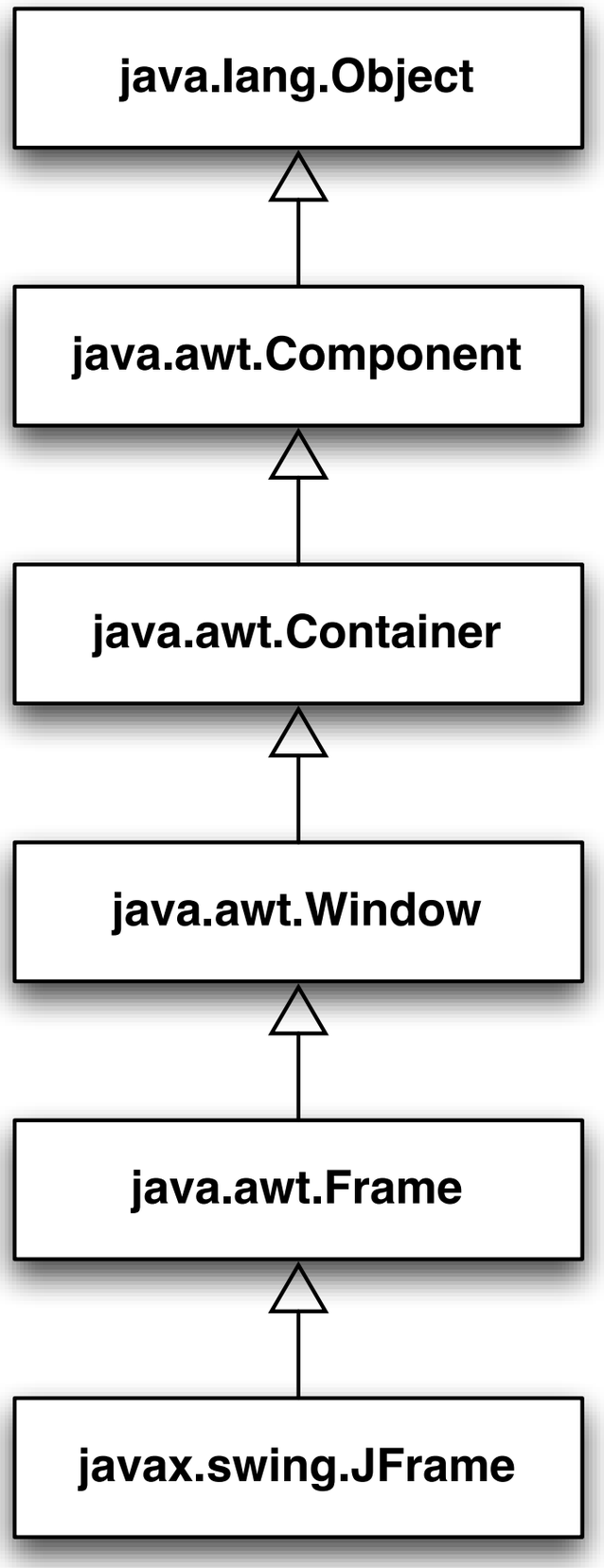
Java GUI Toolkits



Frames

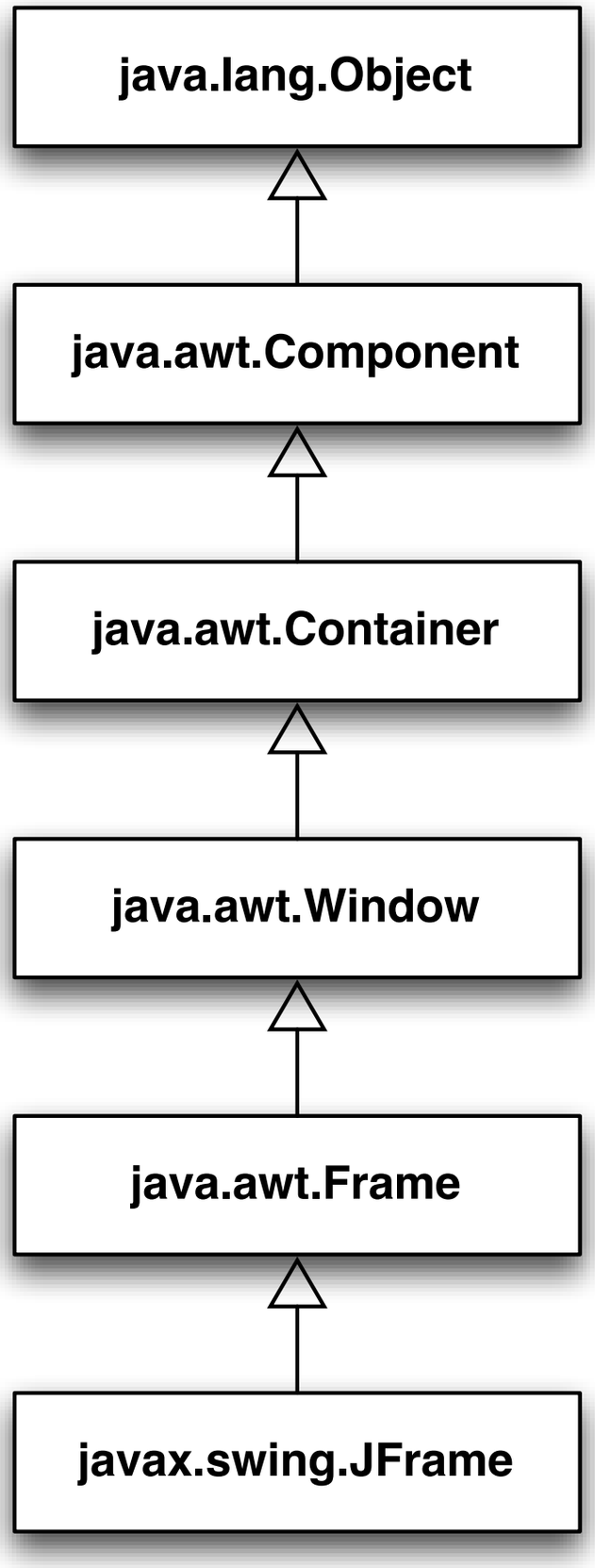
Frames


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

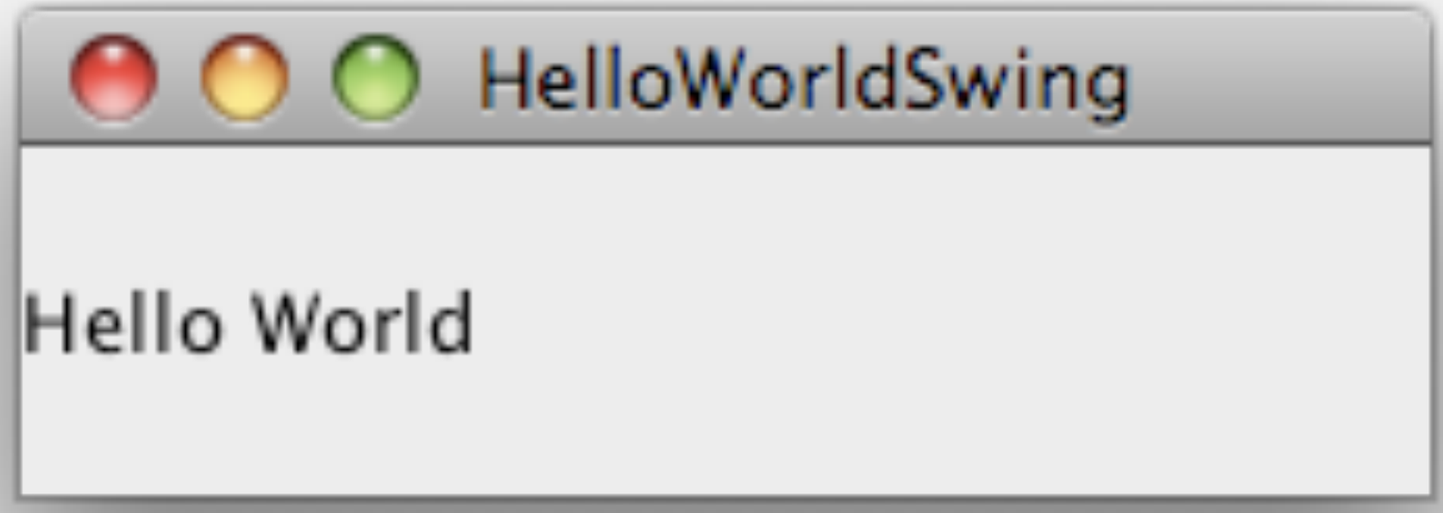


Frames

```
JFrame frame = new JFrame("HelloWorldSwing");
```



Frames




```
JFrame frame = new JFrame("HelloWorldSwing");
```

Hello World

```
private static void createAndShowGUI() {
    //Create and set up the window.
    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    //Add the ubiquitous "Hello World" label.
    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    //Display the window.
    frame.pack();
    frame.setVisible(true);
}
```

Hello World

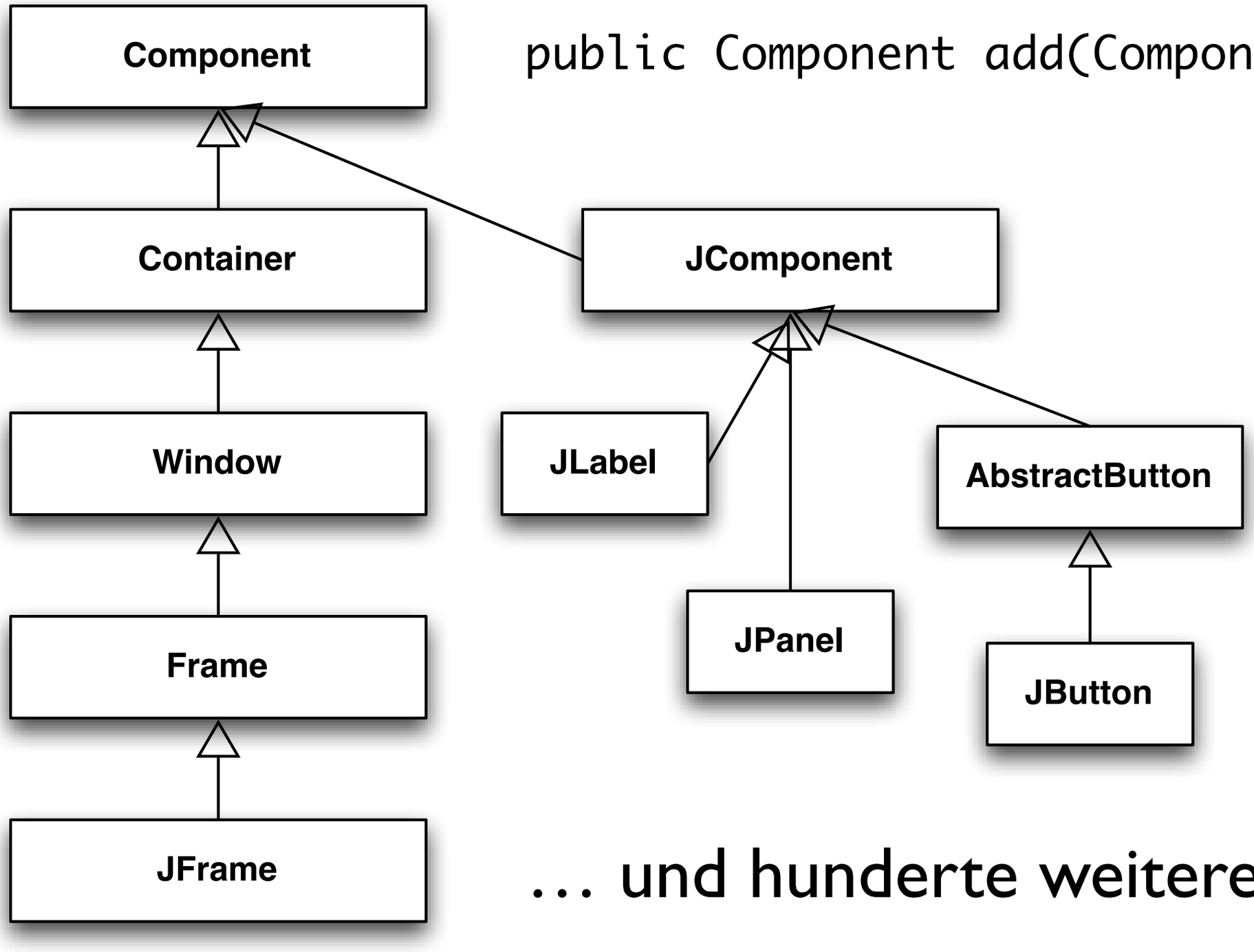
```
import javax.swing.*;

public class HelloWorldSwing {
    private static void createAndShowGUI() { /* see above */ }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Elemente hinzufügen

Elemente hinzufügen

```
public Component add(Component c);
```

Elemente hinzufügen

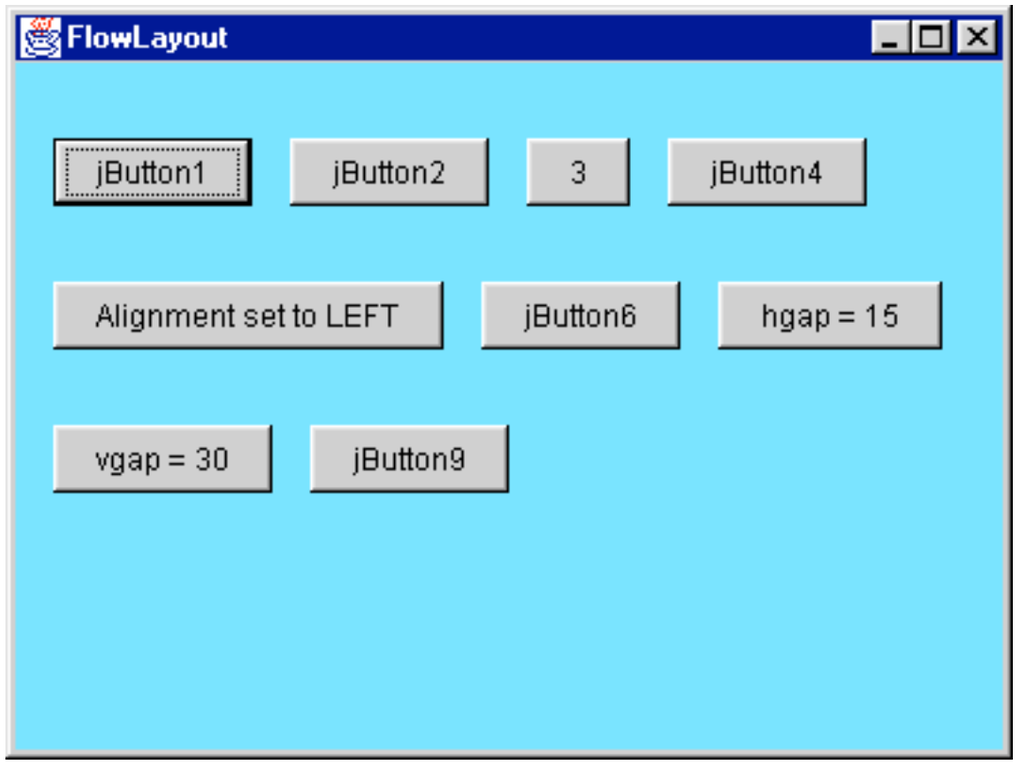



```
public Component add(Component c);
```

… und hunderte weitere Elemente!

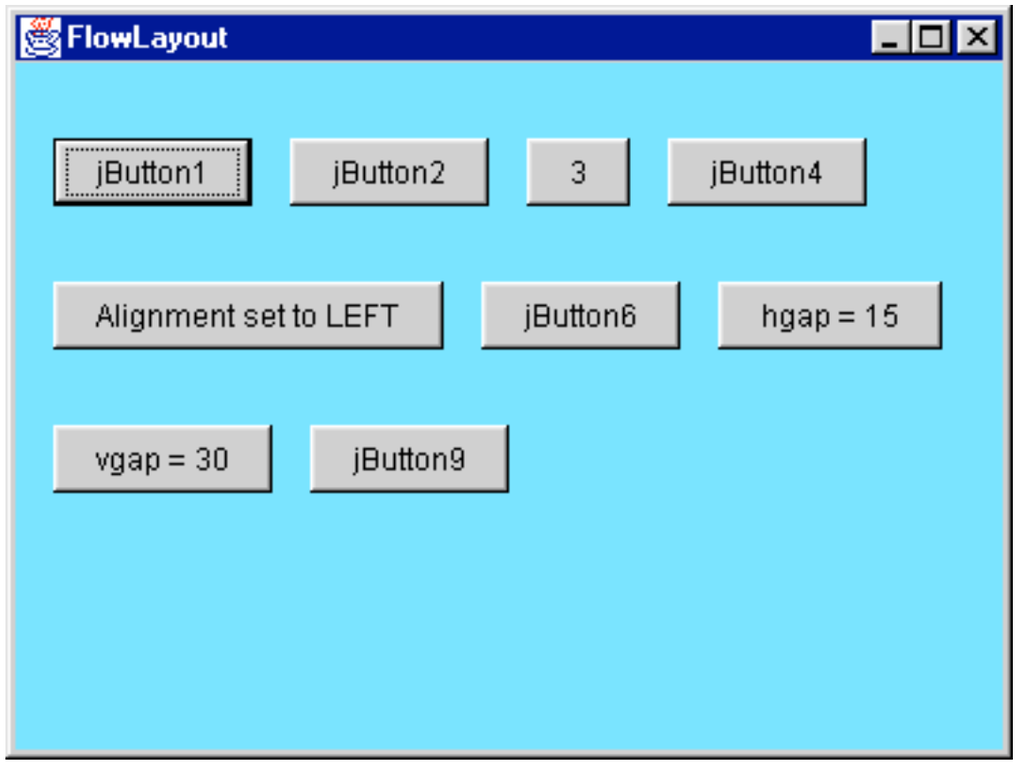
Layout

```
frame.getContentPane().setLayout(new FlowLayout());
```

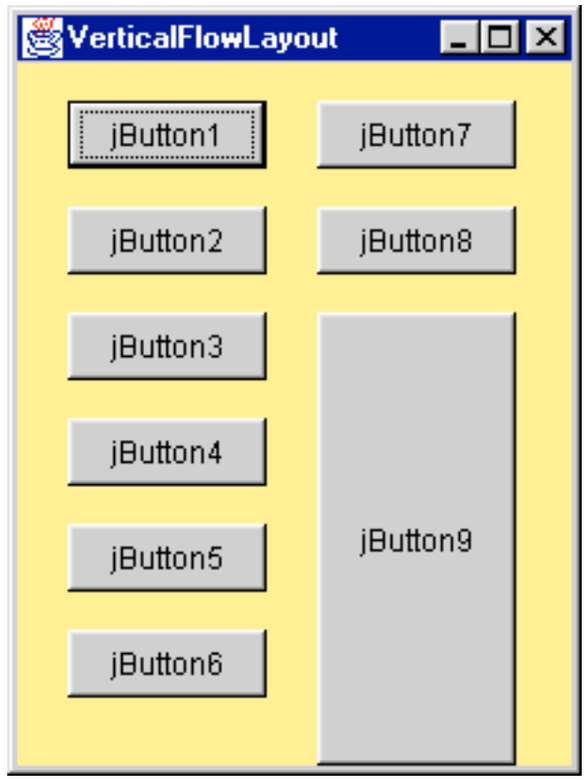


Layout

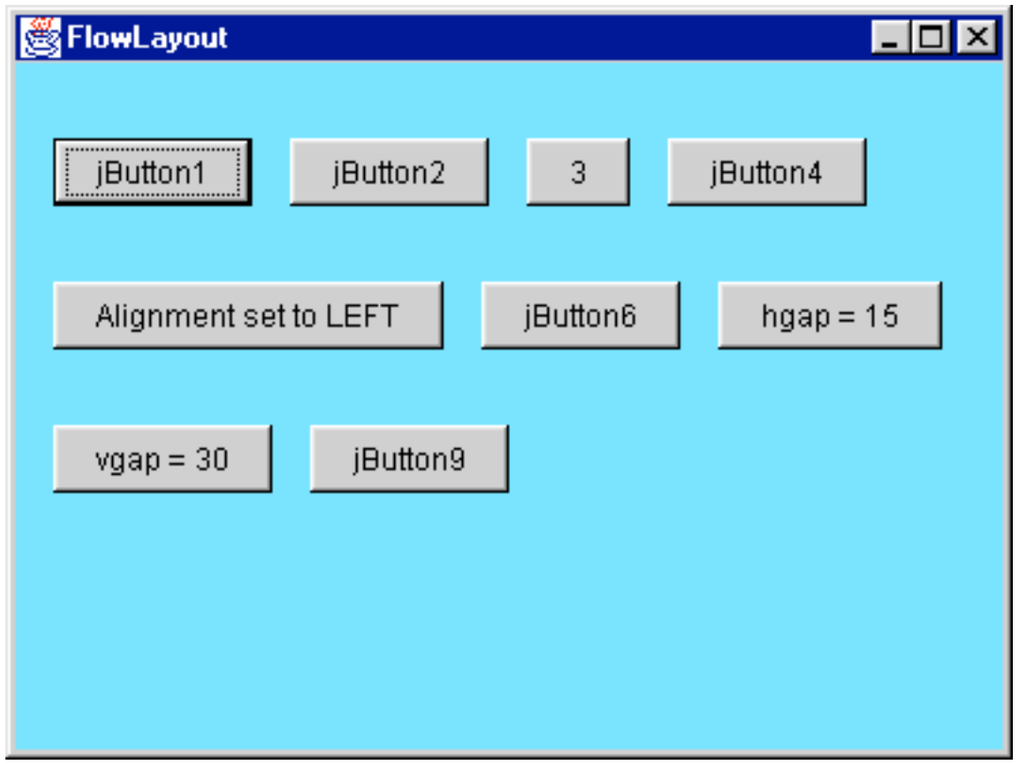
```
frame.getContentPane().setLayout(new FlowLayout());
```



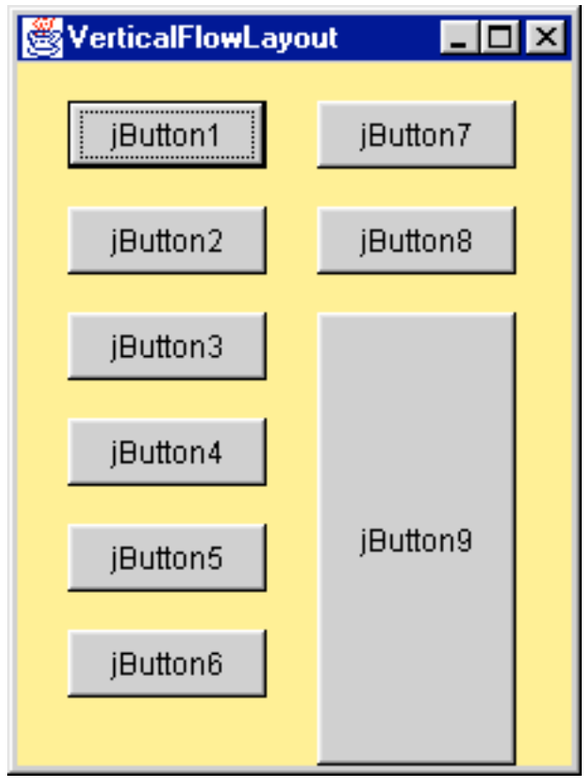
Layout



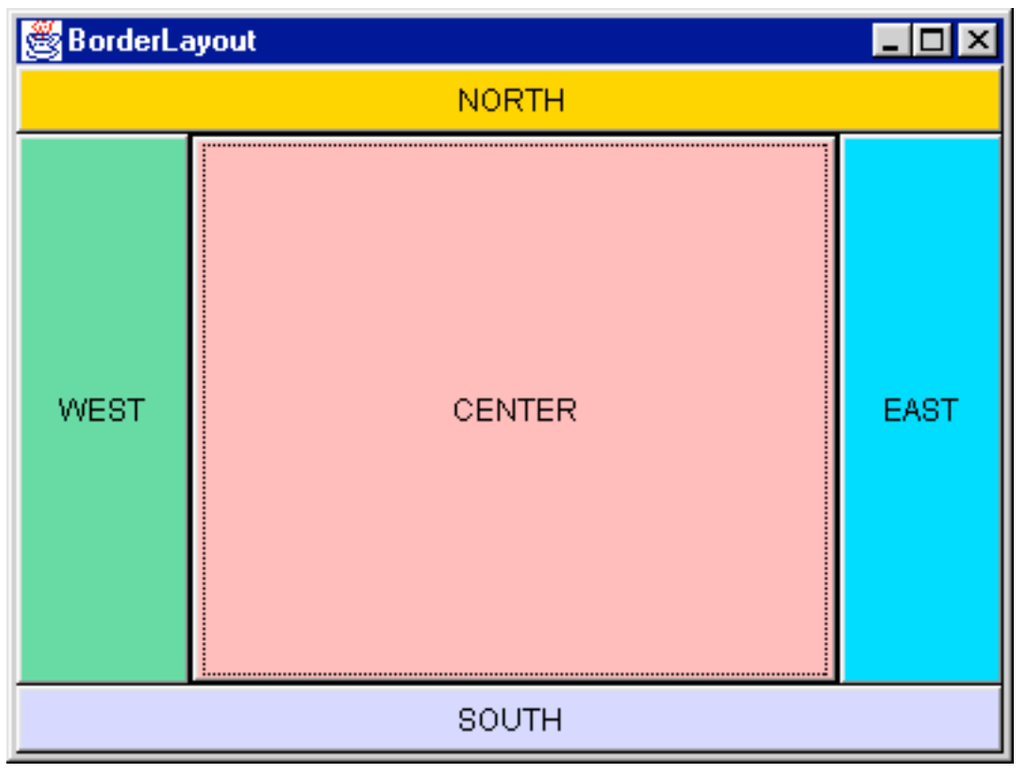

```
frame.getContentPane().setLayout(new FlowLayout());
```

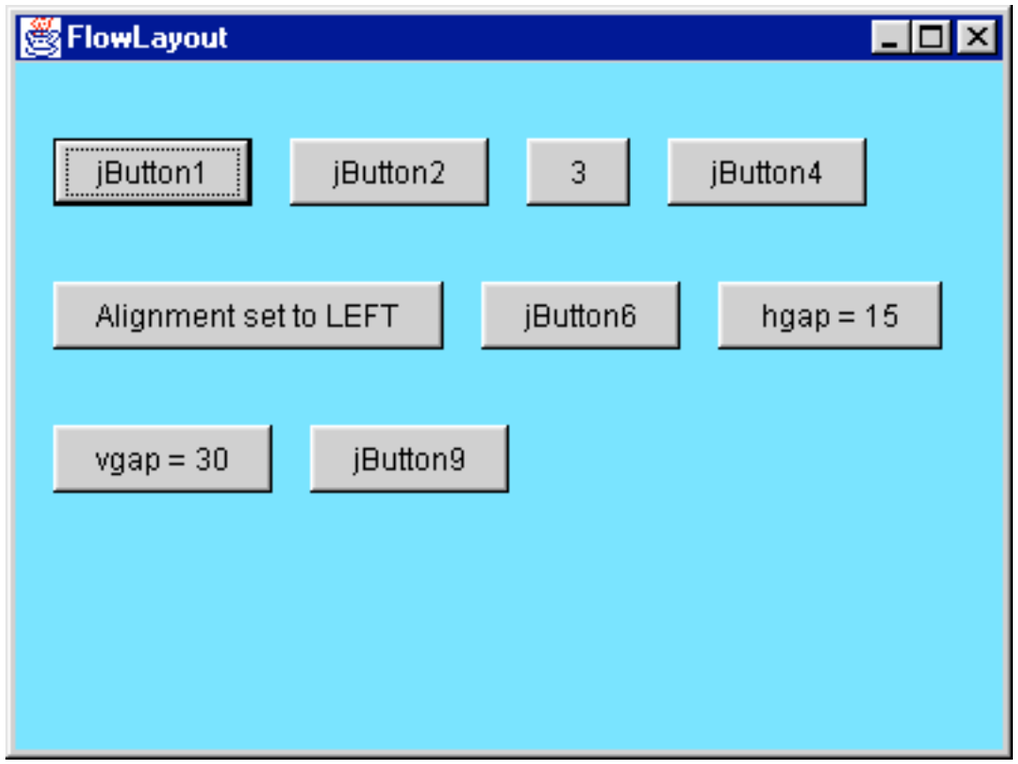


Layout

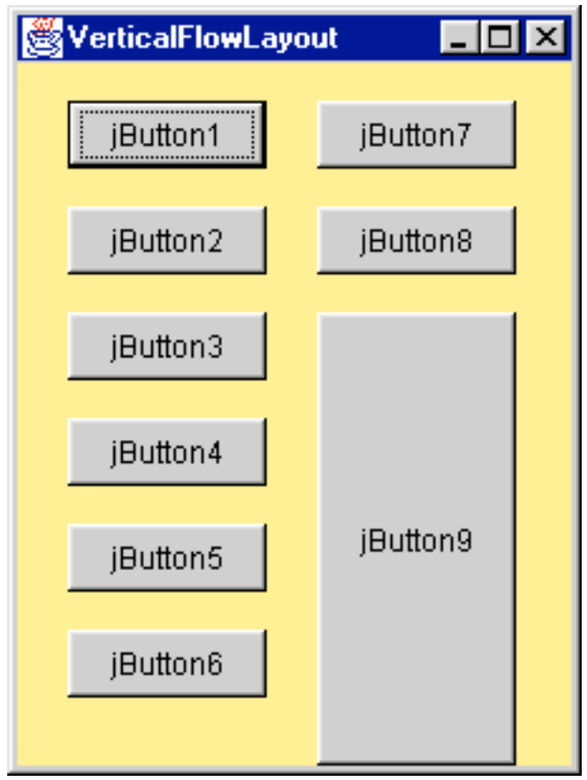



```
frame.getContentPane().setLayout(new FlowLayout());
```

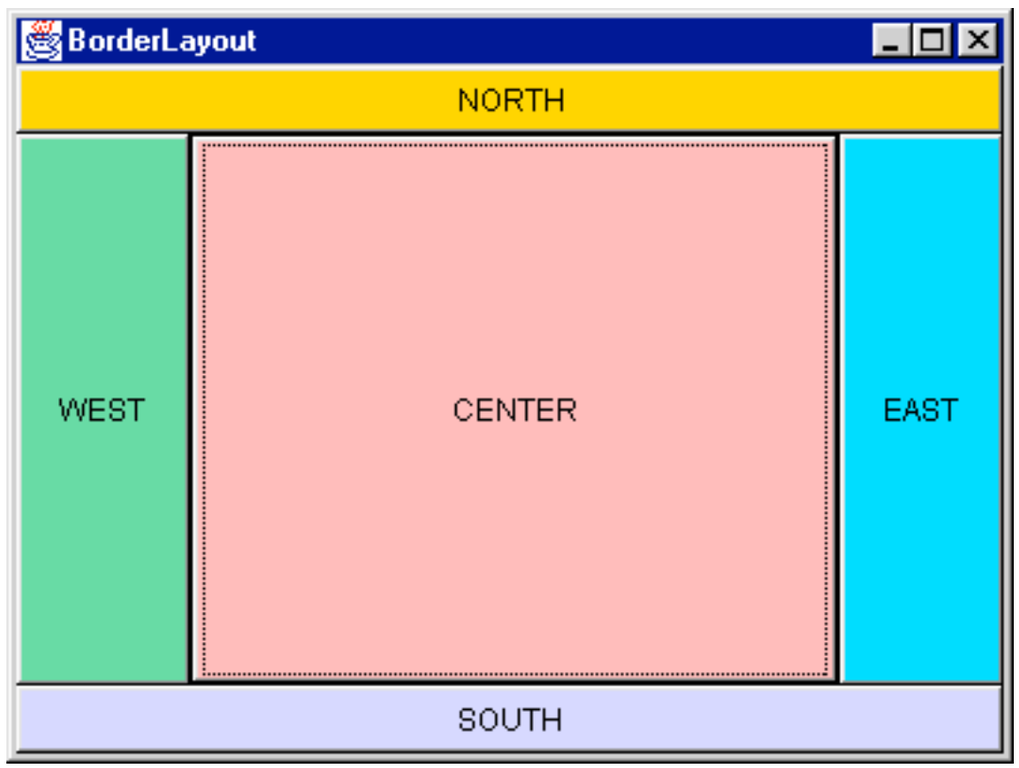


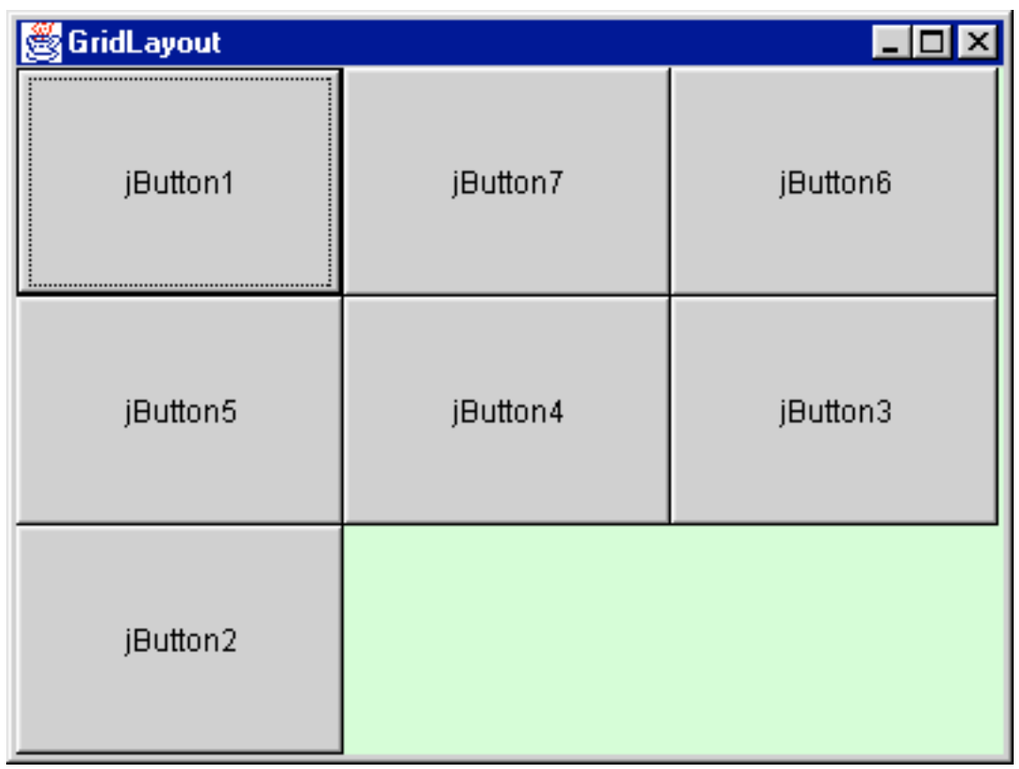


Layout




```
frame.getContentPane().setLayout(new FlowLayout());
```





Aktionen

Aktionen

1. Für jede Aktion eigene Klasse implementieren

Aktionen

1. Für jede Aktion eigene Klasse implementieren

```
public class MyClass implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        ...//code that reacts to the action...
    }
```

Aktionen

1. Für jede Aktion eigene Klasse implementieren

```
public class MyClass implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        ...//code that reacts to the action...
    }
```

2. Objekt der Klasse beim Bedienelement registrieren

Aktionen

1. Für jede Aktion eigene Klasse implementieren

```
public class MyClass implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        ...//code that reacts to the action...
    }
```

2. Objekt der Klasse beim Bedienelement registrieren

```
someComponent.addActionListener(instanceOfMyClass);
```

auch WindowListener, ChangeListener, MouseListener…

Model-View-Controller

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%

Bundestagswahl
22.09.2013

Model-View-Controller

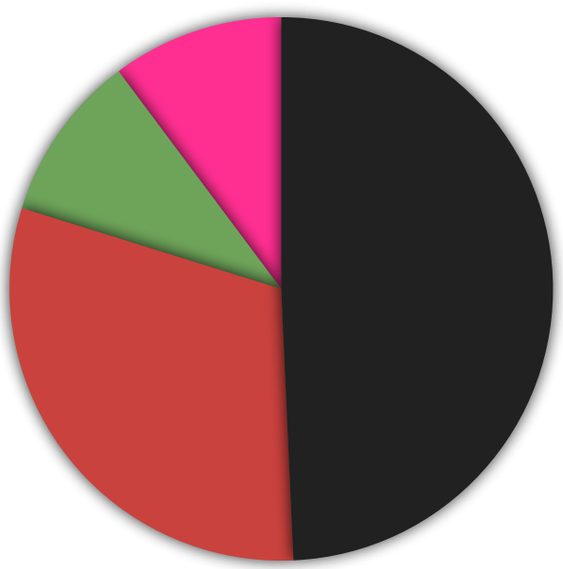
CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%
Sonstige	6,2%

Bundestagswahl
22.09.2013

Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%
Sonstige	6,2%

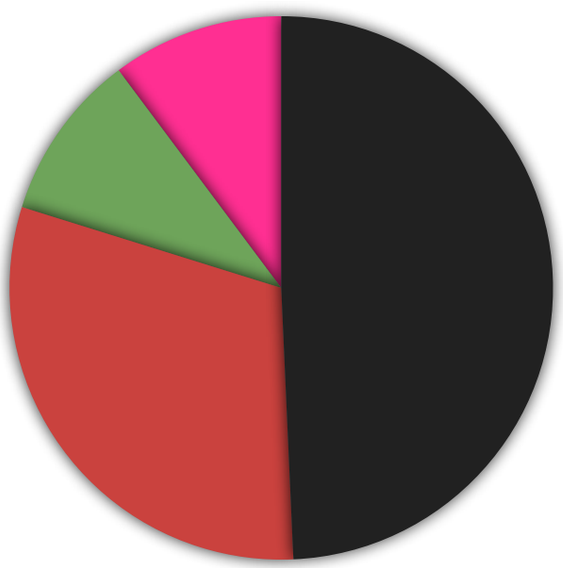
Bundestagswahl
22.09.2013

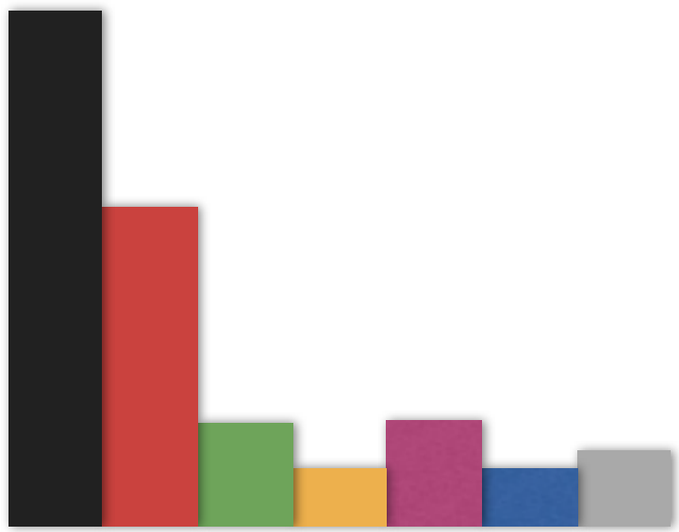


Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%
Sonstige	6,2%

Bundestagswahl
22.09.2013

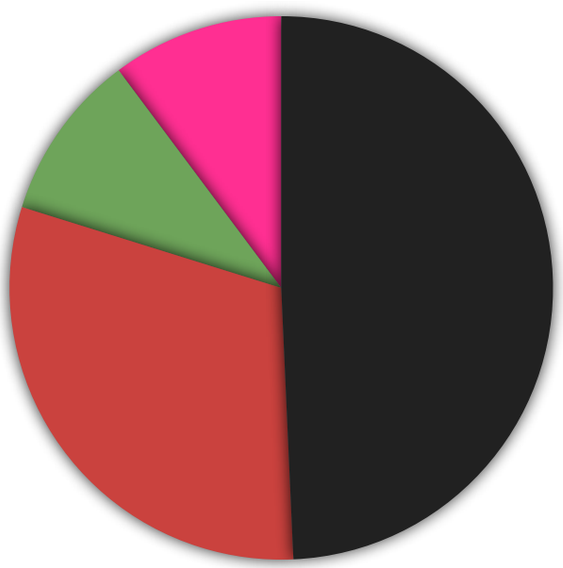


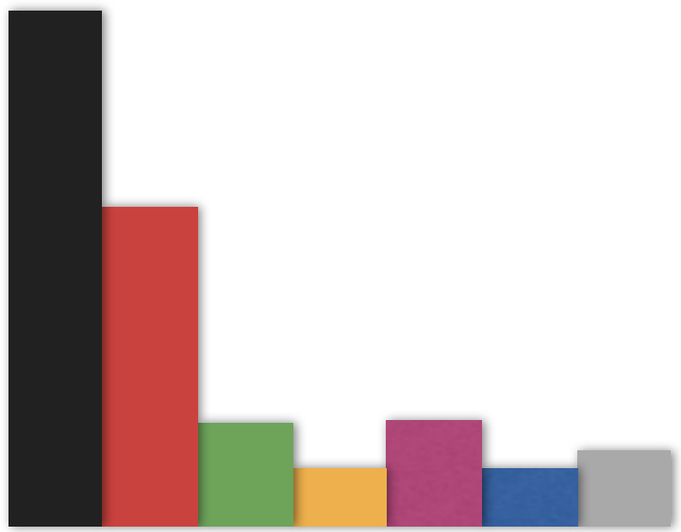


Model-View-Controller

CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%
Sonstige	6,2%

Bundestagswahl
22.09.2013

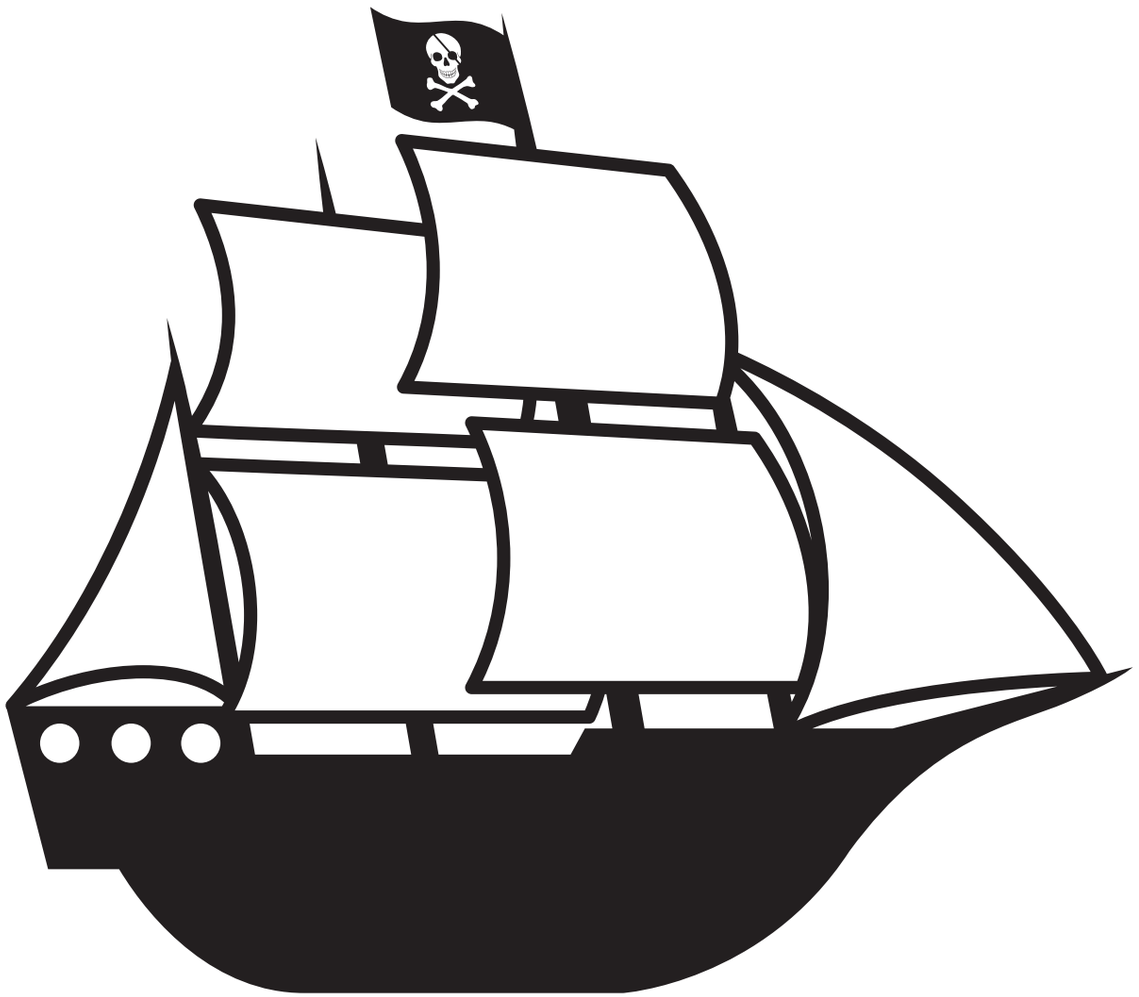




CDU/CSU	41,5%
SPD	25,7%
GRÜNE	8,4%
FDP	4,8%
LINKE	8,6%
AfD	4,7%
Sonstige	6,2%

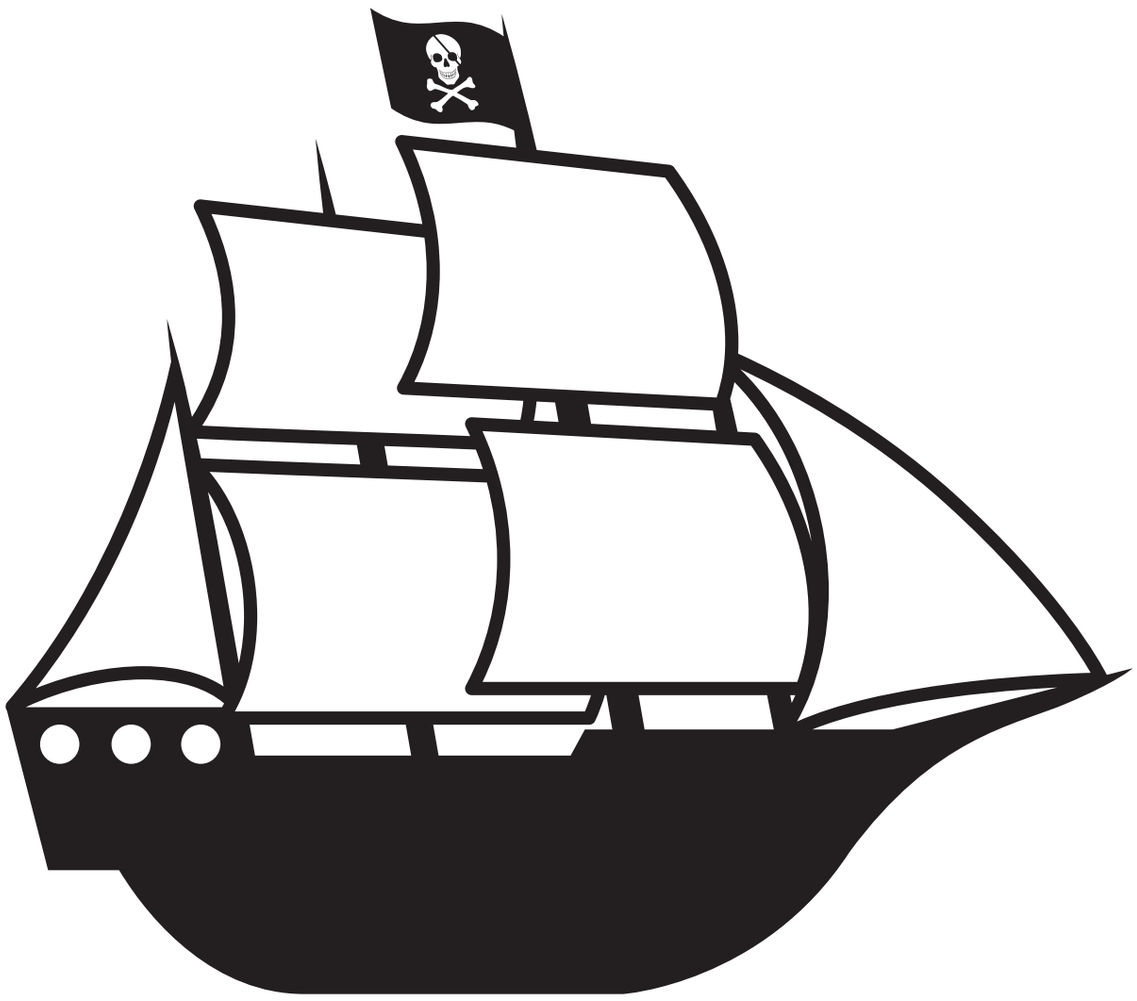
Grafik

Grafik

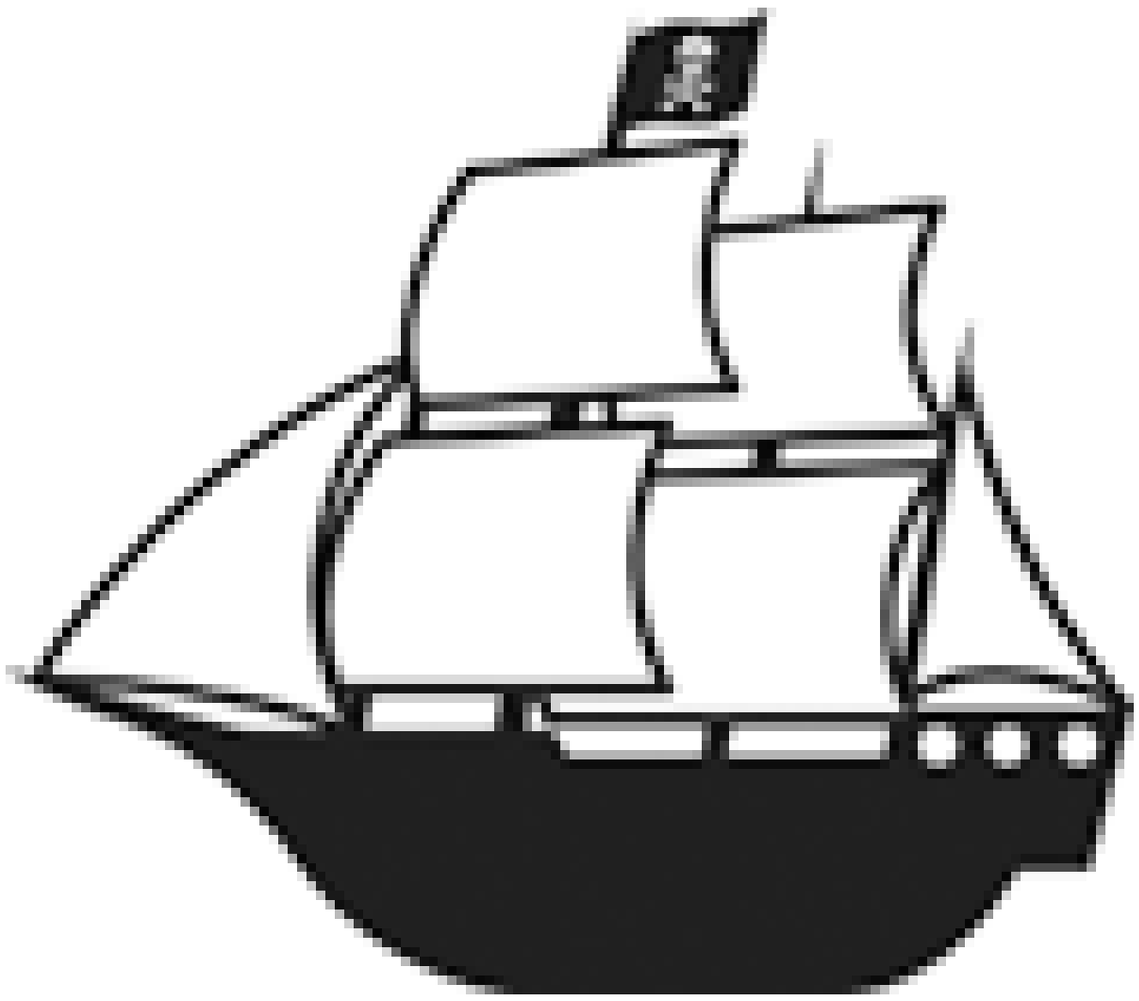


Vektor

Grafik



Vektor



Pixel

Icons

Icons

Icons

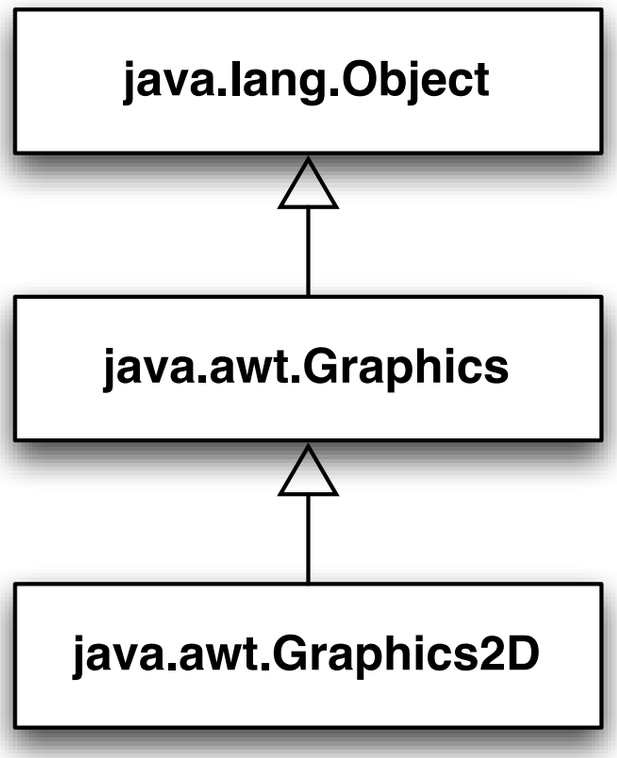

```
mageIcon icon = createImageIcon("images/middle.gif",
                                "pretty but meaningless");
label1 = new JLabel("Image and Text", icon, JLabel.CENTER);
...
label3 = new JLabel(icon);
```

Icons

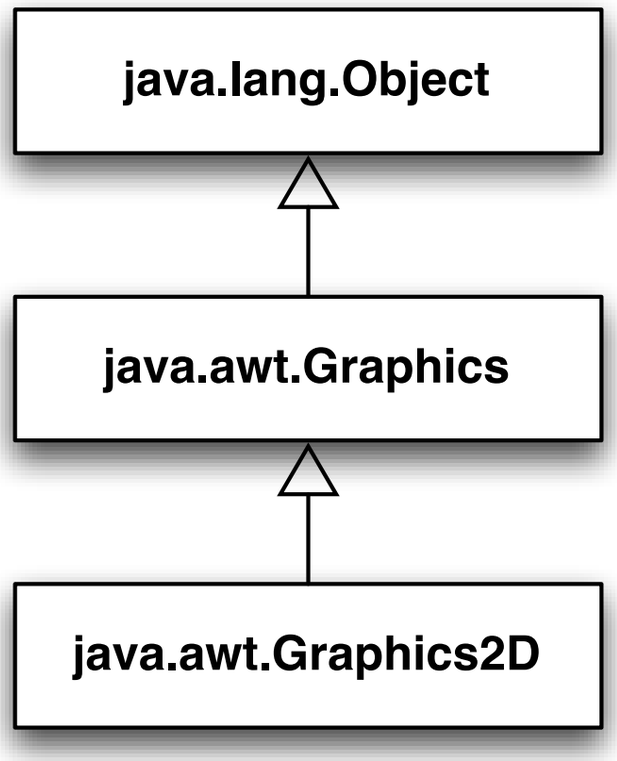

```
mageIcon icon = createImageIcon("images/middle.gif",
                                "pretty but meaningless");
label1 = new JLabel("Image and Text", icon, JLabel.CENTER);
...
label3 = new JLabel(icon);
```

```
/** Returns an ImageIcon, or null if the path was invalid. */
protected ImageIcon createImageIcon(String path, String description) {
    java.net.URL imgURL = getClass().getResource(path);
    if (imgURL != null) {
        return new ImageIcon(imgURL, description);
    } else {
        System.err.println("Couldn't find file: " + path);
        return null;
    }
}
```

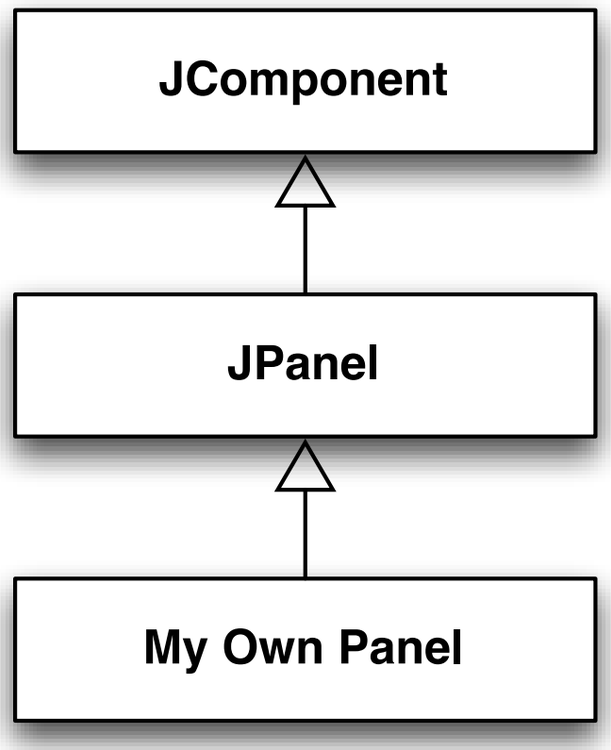
Vektorgrafik

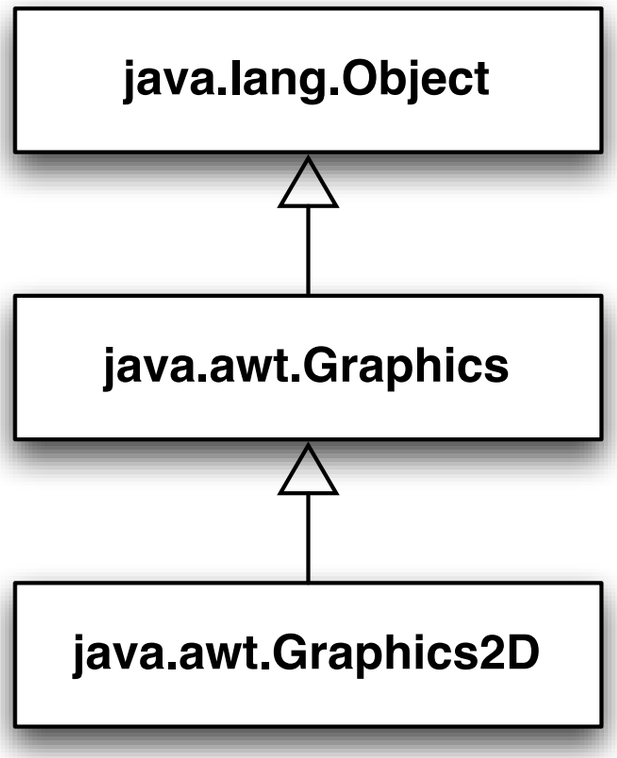


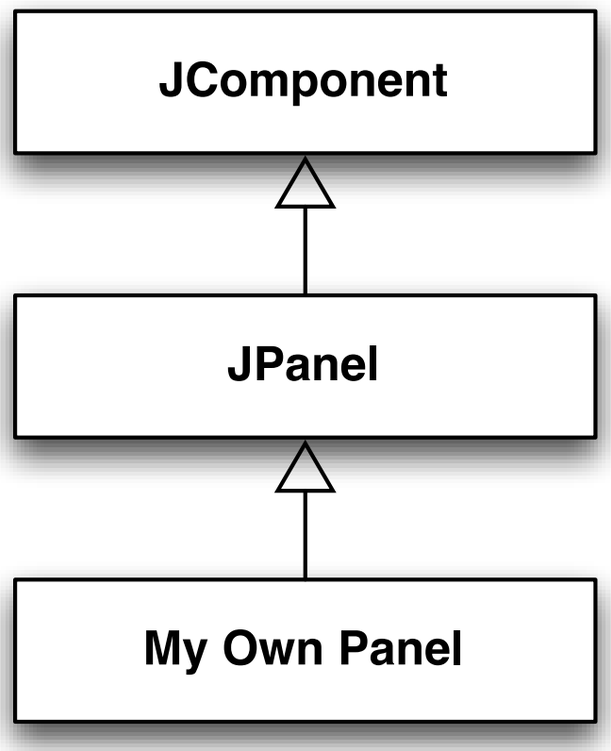
Vektorgrafik



Vektorgrafik







Vektorgrafik

```
import java.awt.Graphics2D;

public class CellPanel extends JPanel {

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        drawItall(g2d);
    }

    private void drawItAll(Graphics2D g2d) {
        g2d.drawImage(…);
        g2d.drawString(…);
        g2d.drawLine(…);
    }
}
```