# Einstieg in die Informatik mit Java

## Swing

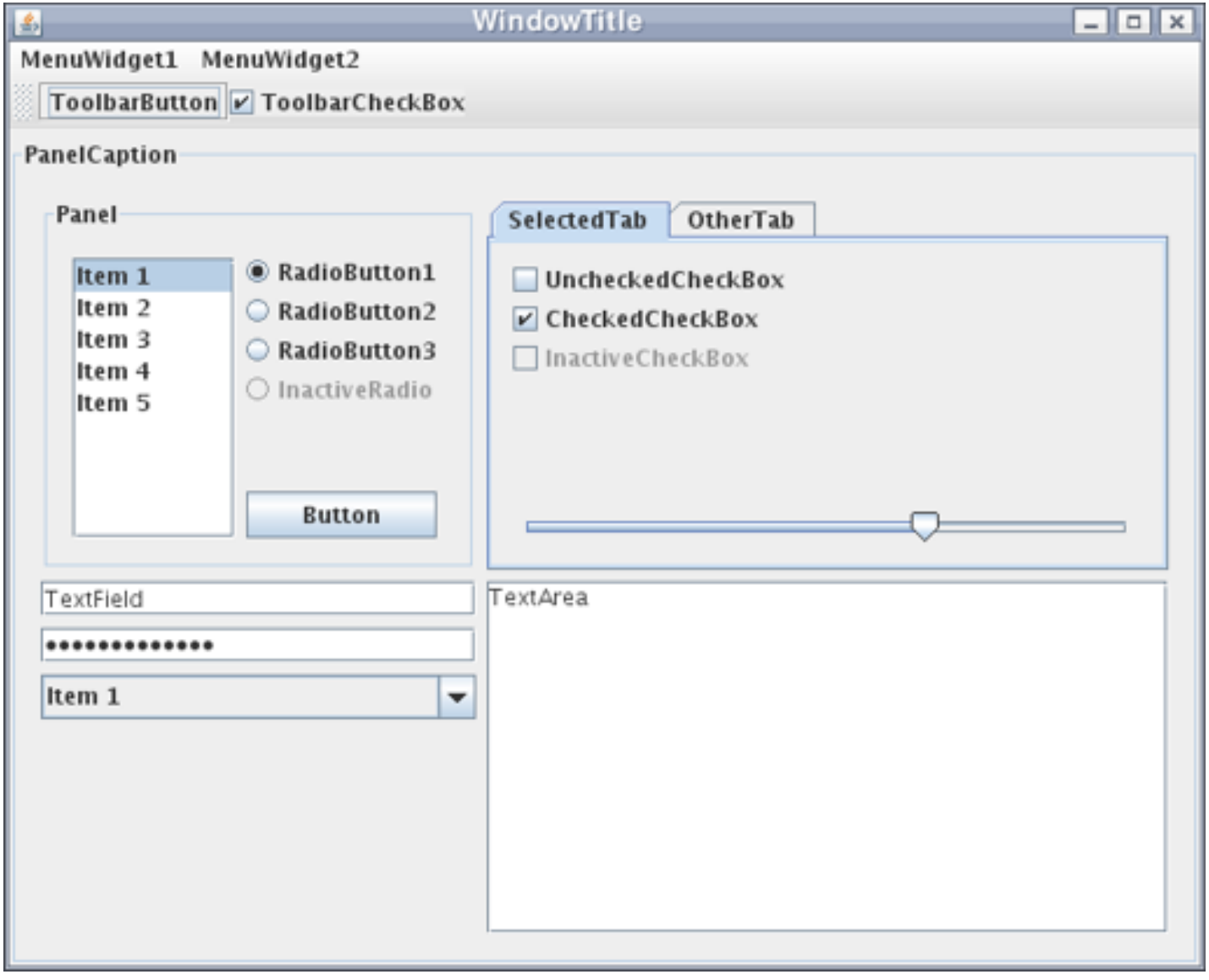
Gerd Bohlender

Institut für Angewandte und Numerische Mathematik

# Gliederung

# Gliederung

# Einführendes Beispiel




Quelle: `http://upload.wikimedia.org/wikipedia/commons/c/cc/Gui-widgets.png`

# Gliederung

# Eigenschaften von Swing

- leistungsfähige Grafikbibliothek
- baut auf AWT auf
- ist in Java seit Version 1.2 enthalten
- Applets werden abgeleitet von javax.Swing.JApplet, entsprechend JFrame
- Swing-Komponenten (JButton, JLabel, ...) sind in Java geschrieben („leichtgewichtige“ Komponenten)
- im Gegensatz dazu werden Komponenten bei AWT vom Betriebssystem erzeugt („schwergewichtige“ Komponenten)

# Vor- und Nachteile von Swing

- Vorteil „leichtgewichtiger“ Komponenten: kein separater Betriebssystemaufruf für das Erzeugen der Komponente
- Vorteil: flexibler, es sind auch Komponenten möglich, die das Betriebssystem nicht vorsieht
- bei AWT sind im Gegensatz dazu nur Komponenten möglich, die auf „allen“ Betriebssystemen verfügbar sind
- Vorteil/Nachteil: gleiches Aussehen auf allen Betriebssystemen
- Vorteil: wählbares Look-and-Feel (wie Windows, wie Motif/Linux, eigene Java-Version, ...)
- weitere Vorteile wie Tooltips, Drag-and-Drop, Steuerung von Komponenten durch Tastenkombinationen, Multiple Document Interface, Doppelpufferung, ...
- Nachteil: auf älteren Rechnern ist Swing träge aufgrund seiner Komplexität

# Ebenen einer Swing-Oberfläche

Ein JApplet oder JFrame besteht aus mehreren Ebenen (Container vom Typ Pane):

- RootPane enthält LayeredPane und GlassPane
- LayeredPane enthält ContentPane und MenuBar
- ContentPane enthält die Komponenten
- MenuBar enthält eine Menüleiste
- GlassPane ist ein durchsichtiger Container, der über allen anderen Containern liegt

Am wichtigsten für Anwendungen ist die ContentPane, sie kann beschafft werden mit

```
Container content = getContentPane();
```

Dieser werden Komponenten hinzugefügt mit

```
content.add ...
```

# Gliederung

# Typisches Swing-Applet

```
import java.awt.*;
import javax.swing.*;
public class SwingApplet extends JApplet {
      JLabel jl;
      public void init() {
            // zuerst Content Pane beschaffen
            Container content = getContentPane();
            // alle Komponenten etc. auf Content Pane
            content.setBackground(Color.white);
            jl = new JLabel("Ich bin ein Swing-JLabel");
            // Default-Layout bei Swing ist BorderLayout
            content.add(jl, BorderLayout.SOUTH);
      }
}
```

# Gliederung

# Swing und Threads

Swing ist nicht thread-safe:

- bei „paralleler“ Ausführung mehrerer Threads kann es zu unkontrollierten Effekten kommen
- Swing-Komponenten sollten stets in einem sogenannten Event Dispatch Thread von AWT ausgeführt werden
- dieser führt die gesamte Ereignisverarbeitung grafisch-interaktiver Java-Anwendungen aus
- die Hilfsklasse SwingUtilities stellt zwei Methoden invokeLater und invokeAndWait bereit
- diese werden aus anderen Threads heraus aufgerufen und besitzen ein ausführbares Objekt vom Typ Runnable als Parameter

# Swing und Threads

Sichere Behandlung von Threads unter Swing:

- der gesamte Aufbau der Oberfläche wird in eine Methode verpackt
- es wird ein Objekt erzeugt, das das Interface Runnable implementiert; dieses ruft in seiner Methode run die obige Methode auf
- invokeLater reiht das ausführbare Objekt in die Ereigniswarteschlange von AWT ein und kehrt noch vor dessen Ausführung in den aufrufenden Code zurück. Der aufrufende Thread wird also nicht blockiert
- invokeAndWait reiht das ausführbare Objekt in die Ereigniswarteschlange von AWT ein und wartet, bis es abgearbeitet wurde. Der aufrufende Thread wird also blockiert

# Typische Thread-Behandlung (Beispiel JFrame)

```
import javax.swing.*;

public class HelloWorldSwing {
    private static void createAndShowGUI() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```