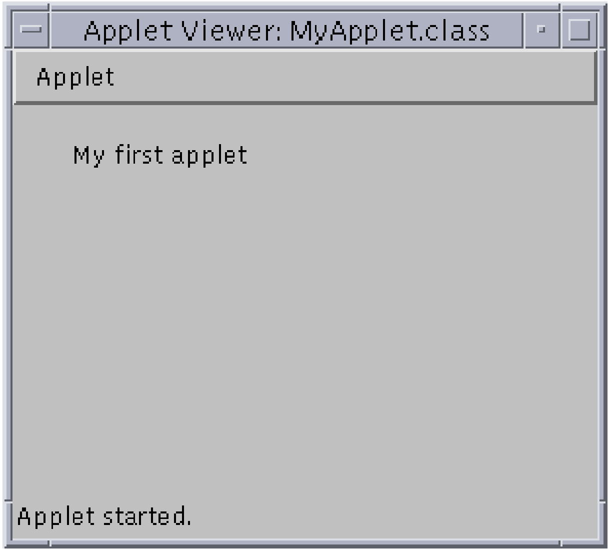
Gliederung:

- Was ist ein Applet?
- Aufbau des Abstract Window Toolkit (AWT)
  - Klassenhierarchie
  - Grundlegende Konzepte
- Eigenschaften von Applets
- Layout
- Event–Handling

Was ist ein Applet?


```
<applet code=MyApplet.class
        width=300
        height=200>
</applet>
```




```
import java.awt.*;
import java.applet.*;

public class MyApplet extends Applet {
  public void paint(Graphics g) {
    g.drawString("My first applet", 30, 30);
  }
}
```

```
appletviewer MyApplet.html
```

Was ist ein Applet?

- „kleines“, einbindbares JAVA–Programm
- Aufpeppen von Web–Seiten vs. sinnvolle Anwendungen
- Inhärent plattform–unabhängig und damit portabel
- Einfache Möglichkeit, GUIs zu entwickeln
- Läuft nur in Verbindung mit Browser bzw. Viewer

Aufbau des AWT

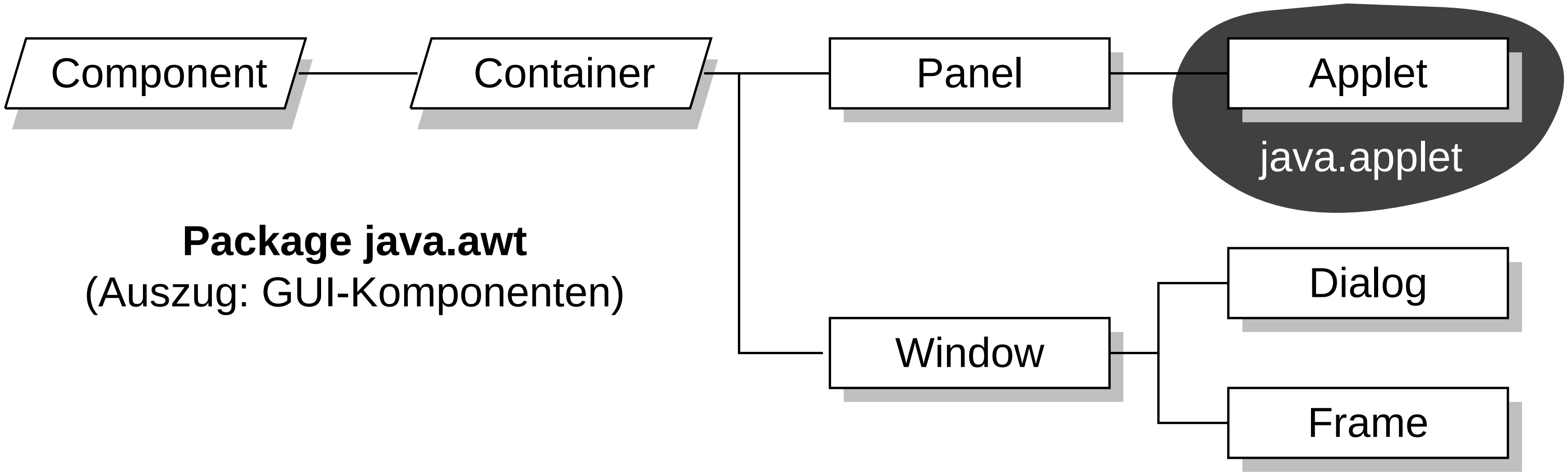
- Klassenkategorien im `java.awt`–Paket:

  Grafik: Klassen für Farben, Zeichensätzen, Bilder, Polygone etc.

  GUI–Komponenten: Buttons, Menüs, Dialogboxen, Container etc.

  Layout–Manager: Erscheinungsbild der Komponenten

- „Pfad“ der Klasse `Applet`:



- Definiert geräteunabhängige Grafikschnittstelle

- Methoden um geometrische Formen zu zeichnen, Bilder darzustellen etc.

```
public abstract class Graphics extends Object {
  // Konstruktor kann nicht direkt aufgerufen werden!
  protected Graphics();

  // Kopiert ein bestehendes Graphics-Objekt
  public abstract Graphics create();

  // Zeichnen eines ausgefüllten Polygons:
  public abstract void fillPolygon(int[] xP, int[] yP, int n);

  // Darstellen einer Zeichenkette
  public abstract void drawString(String s, int x, int y) ;

  ...
}
```

- Spezifische Subklassen von Graphics für verschiedene Plattformen

- Spezifiziert eine Komponente, die andere Komponenten enthält:

```
public abstract class Container extends Component {
  // Kein Konstruktor!

  // Hinzufügen einer Komponente zum Container:
  public Component add(Component comp);

  // Entfernen einer Komponente:
  public synchronized void remove(Component comp);

  // Setzen des Layout-Managers:
  public void setLayout(LayoutManager mgr);

  ...
}
```

- Basisklasse aller grafischen Benutzungsschnittstellen

- Basisklasse aller Ereignisklassen des AWT

- Wichtigste Subklassen:

  ComponentEvent: Beschreibung von „Low-Level“-Ereignissen wie Mouseclick, Fensterscrollen

  ActionEvent,...: „Semantische“ Ereignisse wie das Ausführen eines Kommandos, Zustandsänderung

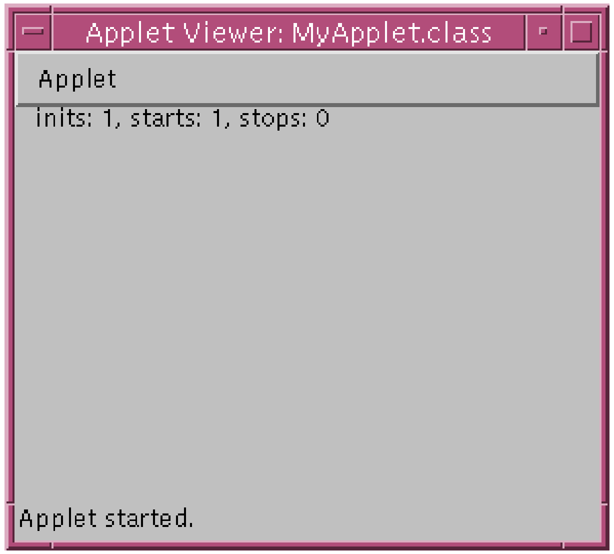
- Teilweise verschiedene Ereignisse in derselben Klasse gekapselt (Beispiel: `MouseEvent`)

- Event-Handling...

Eigenschaften von Applets

- „Hauptprogramm“ abgeleitet von der Klasse `Applet`
- Start: Appletviewer instanziiert Klasse und ruft `init()` und `start()` auf
- Sicherheitsbeschränkungen:
  - Lokale Dateien weder lesen noch schreiben
  - Keine Programme ausführen
- Stets grafik– und ereignisorientiert
- Abspielen von Sounds möglich

- Instanziierung, Initialisierung, Starten und evtl. Stoppen,




```
import java.awt.*;
import java.applet.*;

public class MyApplet extends Applet {
  String s;
  int inits = 0,
      starts = 0,
      stops = 0;
  public void init() {inits++;}
  public void start() {starts++;}
  public void stop() {stops++;}
  public void paint(Graphics g) {
    s = "inits: " + inits +
        ", starts: " + starts +
        ", stops: " + stops;
    g.drawString(s,10,10);
  }
}
```

- Keine Kommandozeile, daher im Applet-Tag codiert:

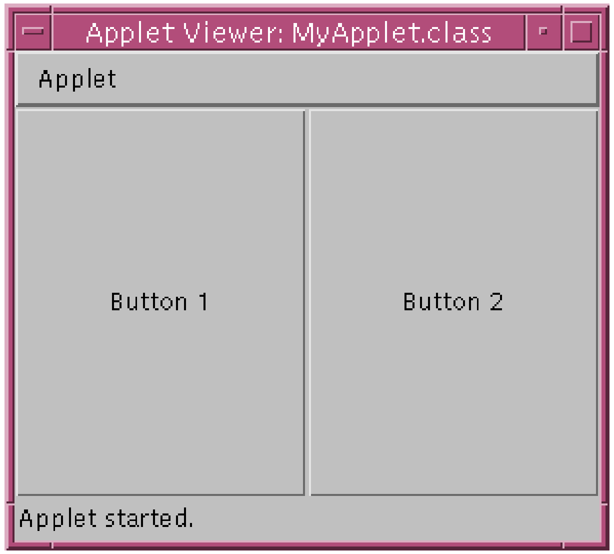
```
<applet code=Example.class width=400 height=200>
<param name="P1" value=42>
<param name="P2" value=77>
</applet>
```

- Einlesen per `getParameter()`-Methode (Achtung: Exception?!)

```
try {
  int p1 = Integer.parseInt(getParameter("P1"));
  int p2 = Integer.parseInt(getParameter("P2"));
} catch (NumberFormatException e) {
}
```

Layout von Applets

- „Layout–Manager“ verwalten Anordnung der Komponenten:




```
...
public class MyApplet extends Applet {
Button b1 = new Button("Button 1"),
       b2 = new Button("Button 2");

public void init() {
  b1.addActionListener(new B1());
  b2.addActionListener(new B2());
  setLayout(new GridLayout(1,2));
  add(b1);
  add(b2);
}
  class B1 implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      getAppletContext().showStatus("Button 1");
    }
  } ...
}
```

- `FlowLayout()` (Default), `BorderLayout()`, `GridLayout(x,y)`

View: Komponente der Anwendung. Realisiert (grafische) Darstellung

Model: Enthält alle Objekte und Methoden mit Bezug zur Anwendung

Controller: Auswertung von Inputs. Veranlaßt View/Model sich zu verändern

- Beispiel: Bestellungs–Anwendung

  View: Eingabemaske mit Textfeldern, Buttons etc.

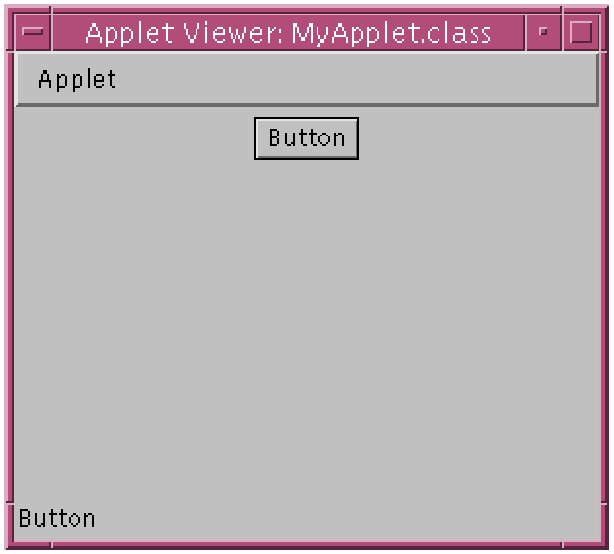
  Model: Klassen `Bestellung` (Gesamtsummenberechnung, Weiterleiten an Datenbank) und `Posten` (Artikelnummer, Preis)

  Controller: Benutzereingaben (per Maus, Tastatur) weiterreichen

Delegation Event Model

- Ereignisquellen lösen sog. *Events* aus
- Ereignisempfänger reagieren auf diese „Nachrichten“
- Weiterleitung der Events nur an registrierte Empfänger
- Reduktion des Nachrichtenaufkommens
- *Vollkommene Trennung von Oberfläche und Anwendung*

- Beispiel:




```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class MyApplet extends Applet {
  Button b = new Button("Button");

  public void init() {
    b.addActionListener(new B());
    add(b);
  }

  class B implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      getAppletContext().showStatus("Button");
    }
  }
}
```

`http://www.javasoft.com/products/jdk/1.1/docs/api/packages.html`
Dokumentation aller Pakete, Klassen und Methoden, u.a. `java.applet` und `java.awt`

`http://www.java.sun.com/docs/books/tutorial/index.html`
Online–Tutorial, u.a. auch mit ausführlicher Darstellung von Applets

`http://www.addison-wesley.de/Service/Krueger/javabuch.htm`
Online–Version eines sehr guten JAVA–Buches, teilweise Quelle der hier zitierten Beispiele.

`http://www.developer.com/directories/pages/dir.java.html/`
Sammlung von Software, Dokumentation etc.