# Java - AWT

***Basispraktikum SystemArchitektur WS 2008/2209***
***09.12.2008***

***Atanas Dimitrov***

**Was besprechen wir heute:**

- AWT – Klassen
- Die Klasse Graphics
- AWT – Methoden
- Die Klasse ToolKit
- Ausgabe von Bilder
- Die Klasse MediaTracker
- Fluende Animation mittels Double Buffering

**Was besprechen wir heute nicht:**

- Menus
- Dialogfenster und Dialogelemente
- Event Handling
- Swing
- Java 3D

**Java - Abstract Windowing Toolkit (AWT)**

AWT ist eine von Java angebotene Bibliothek zur Entwicklung von grafischer Oberflächen.

**Fähigkeiten**:

- Grafische Primitivoperationen zum Zeichnen von Linien oder Füllen von Flächen und zur Ausgabe von Text
- Methoden zur Steuerung des Programmablaufs
- Dialogelemente zur Kommunikation mit dem Anwender
- Fortgeschrittenere Grafikfunktionen zur Darstellung und Manipulation von Bildern und zur Ausgabe von Sound

**Java - Abstract Windowing Toolkit (AWT)**

**Was brauchen wir:**

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

**Einige Klassen**

**Canvas** – ist die Basis für selbstgestaltete Interaktionsbausteine. Man kann und muss hier selbst Ausgabe und Eingabe programmieren, es gibt keine Standardreaktion auf Events. Canvas allein bietet keine Funktionalität.

**Window** – ist ein selbstständiges Fenster ohne Rahmen mit den typischen Unterklassen Frame (Fenster mit Rahmen) und Dialog (DialogBox). Innerhalb eines Window können Komponenten angeordnet sein.

**Frame** – ist ein Anwendungsfenster einer selbstständigen Java – Anwendung.

Konstruktor: `public Frame(String windowName);`

**Andere Klassen**: `Applet, Panel, Dialog, FileDialog, Button, CheckBox, ...`

**Java - Abstract Windowing Toolkit (AWT)**

## Die Klasse Graphics

`Graphics`: Stellt die Abstrakton eines universellen Ausgabegeräts für Grafik und Schrift dar.

Die Klasse `Graphics` bietet Methoden zur:

- Ausgabe von Primitiven (Linien, Rechtecke, Ovale, ...)

  z.B: `drawRect(), drawLine()`

- Färben von Flächen

  z.B: `fillRect(), fillOval()`

- Ausgabe von Text (Strings)

  z.B: `drawString()`

- Ausgabe von Bildern (*jpg*, *gif* und *png* werden ohne weitere Bibliotheken unterstützt)

  z.B: `drawImage()`

Andere Methoden: `setColor(Color c), setFont(Font f), copyArea(), ...`

**Java - Abstract Windowing Toolkit (AWT)**

**Wichtige Methoden:**

- **`void paint( Graphics g )`:** Enthält alles, was auf dem Bildschirm gezeichnet werden muss.
- **`void update( Graphics g )`:** Löscht den Inhalt des Fensters mit der Backgroundfarbe. Ruft paint auf. update ist schon vorimplementiert, kann und muss (in manche Fälle) aber überschrieben werden.
- **`void repaint()`:** Aktualisiert den Bildschirm. Ruft update( Graphics g ) auf. Schon vorimplementiert.

**Java - Abstract Windowing Toolkit (AWT)**

**Die Klasse ToolKit**

Die Klasse `ToolKit` diehnt als Hilfsklasse bei der Abbildung der portablen AWT-Eigenschaften.

Wird der Anwendung von der Klassenmethode `getDefaultToolKit()` zur Verfügung gestellt.

Viele der Methoden von ToolKit sind wichtig nur für Implementatoren von AWT-Portierungen.

```
Toolkit tk = Toolkit.getDefaultToolkit();
```

**Java - Abstract Windowing Toolkit (AWT)**

**Ausgabe von Bildern**

Die Klasse `Graphics` unterstützt Ausgabe von Bildern:

```
drawImage(Image i, int x, int y, ImageObserver observer);
```

`ImageObserver` überwacht Ladezustand des Bildes.
Der `this` - Pointer kann hier übergeben werden.
`this` – Referenz auf das Fensterobjekt.

**Wie erstellt man aber ein Image Objekt?**

Die Hilfsklasse `ToolKit` bietet hier eine Möglichkeit:

```
Toolkit tk = Toolkit.getDefaultToolkit();
Image im = tk.getImage("DukeTux.gif");
```

**Java - Abstract Windowing Toolkit (AWT)**

**Die Klasse `MediaTracker` - Laden, dann Ausgeben**

```
Image im = tk.getImage("DukeTux.gif");
```

`getImage()` ladet die Datei nicht. Das Bild wird erst dann geladen, wenn es eigentlich benötigt wird.

Resultat ist das mögliche Fehlen eines oder mehrerer Bilder am Anfang.

Die Klasse `MediaTracker` bietet hier eine Lösung.

```
MediaTracker mt = new MediaTracker(this);
mt.addImage(Image m, int index);
mt.waitForAll();
```

Pointer zum Fenster.

Der Thread wartet bei `waitForAll()` bis alle Bilder vollständig geladen sind.

**Java - Abstract Windowing Toolkit (AWT)**

**Double Buffering – Fluende Animation**

Wenn man die Methoden von `Graphics` benutzt, dann wird jede Linie, Rechteck usw. allein an die Grafikkarte geschickt und danach ausgegeben.

Als Resultat kann es passieren, dass man den Background sieht noch bevor etwas darauf gezeichnet wird.

Double Buffering ist eine Möglichkeit dieses Problem zu beheben.

**Was brauchen wir:**

- Eine Instanz von `Graphics` (offscreenGraphics), mit deren Hilfe wir im Hauptspeicher zeichnen, aber auf dem Bildschirm nicht ausgeben.
- Eine Instanz von `Image` (offscreenImage), die wir mit offscreenGraphics zusammensetzen.

**Vorgehensweise:**

- Erstelle die beiden Objekte
- Setze das offscreenImage zusammen.
- Gebe Am Ende das eine zusammengesetzte Bild mittels der normalen `Graphics` Instanz aus.

**Java - Abstract Windowing Toolkit (AWT)**

**Double Buffering – Fluende Animation**

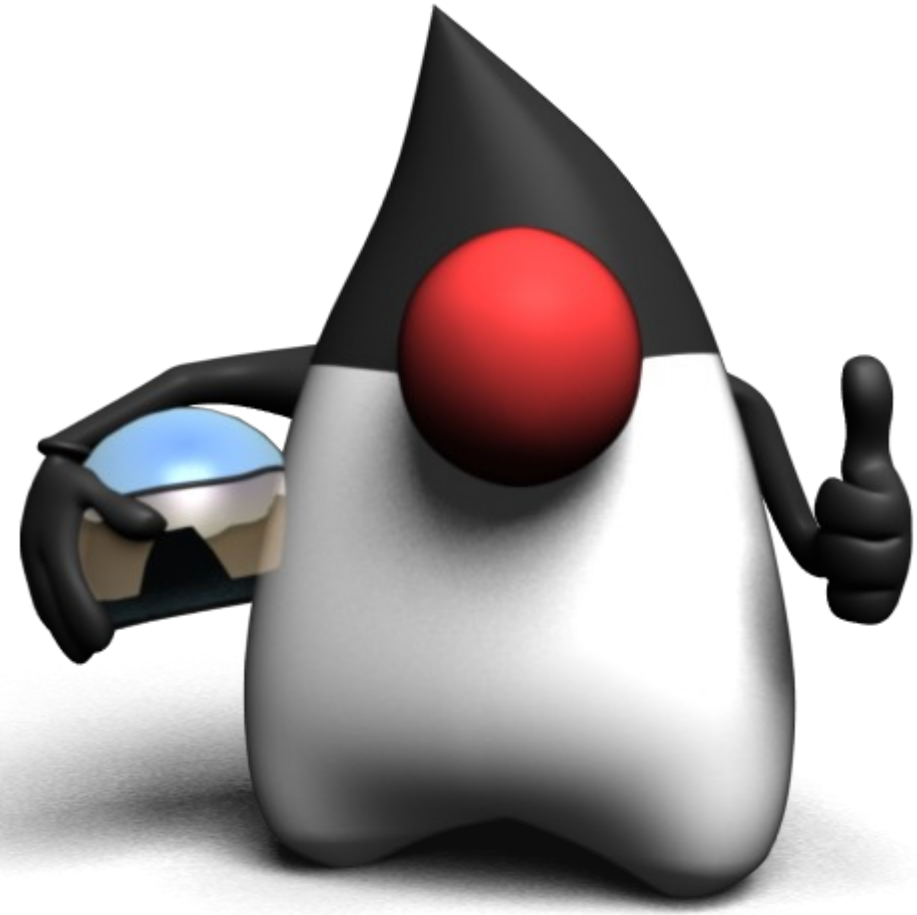
```
private Image offscreenImage;
private Graphics offscreenGraph;

...

offscreenImage = createImage(H_SIZE, V_SIZE);
offscreenGraph = offscreenImage.getGraphics();

...

offscreenGraph.drawString(" Hello World !", 50, 300);
offscreenGraph.drawImage(im, 300, 200, this);

...

g.drawImage(offscreenImage, 0, 0, this);
```

Erstellen der beiden Instanzen.

Erstellt ein `Graphics`-Objekt mit dessen Hilfe man auf das Bild zeichnen kann. Nur für off-screen Bilder..

Zeichnen im Hauptspeicher.

Ausgabe auf dem Bildschirm



# Noch Fragen ???

# *Danke sehr für die Aufmerksamkeit!*