Grundlagen der Informatik für Ingenieure I

6.1 Applets

- Java-Applikationen laufen auf dem Rechner ab, auf dem sie gestartet werden. Die Ladeobjekte liegen im lokalen Netz-(NFS-) Dateisystem. Diesbezüglich unterscheiden sie sich nicht von Programmen, die in C, C++ oder Fortran geschrieben sind.
- Java-Applets laufen “in der Umgebung” eines *“javaenabled” WWW-Browsers* ab. Die Referenz auf ein Applet ist in einer HTML-Seite eingebettet. Daraus ergeben sich u. a. folgende Konsequenzen:
  - Ein Java-Applet kann auf einem beliebigen Web-Server im Internet liegen. Der Webbrowser lädt das Applet vom Webserver herunter und bringt es **lokal** zur Ausführung.
  - Ein Java-Applet kann direkt die grafischen Möglichkeiten des Browsers nutzen.

6.1 Applets

- Sicherheit(Security)
  - Die Tatsache, dass ein Applet beliebig aus dem Internet geladen werden kann, führt zu ernsthaften Sicherheitsproblemen.
    (Viren, Trojanische Pferde, sicherheitstechnologische Schwachstellen).
  - Aus diesen Gründen ist es Applets grundsätzlich nicht erlaubt irgendwelche Aktionen auszuführen, die geeignet sind, Daten auf dem lokalen System in irgendeiner Weise zu verändern.
  - Folgende Restriktionen sind für Applets vorgesehen:
    - Keine E/A-Operationen auf Dateien und
    - keine Kommunikation mit 3. Rechensystemen (also nur mit dem Server und dem lokalen System).
    - Applets sind nicht in der Lage, andere Programmsysteme zu starten; im Unix-Umfeld also kein “fork”!
    - Applets können keine Programme oder Libraries laden.
  - Diese Restriktionspolitik wird im Fachjargon mit *sandboxing* bezeichnet: Dem Applet ist das Spielen nur im Sandkasten - nicht darüber hinaus - erlaubt.

6.2 Applets kreieren

- Kreiert man ein Applet, dann kreiert man eine Subklasse der Klasse *Applet:*

```
public class MeineKlasse extends java.applet.Applet {
....
}
```

  - Die *Applet-Class* ist Teil des *java.applet package* und stellt das nowendige Umfeld bereit, damit ein Browser mit einer *Java-Virtual-Maschine(JVM)* in der Lage ist, ein Java-Applet ablaufen zu lassen.
  - Das grafische Umfeld wird vom *Abstract Windowing Toolkit(AWT)* bereitgestellt. Die *AWT-Classes* sind Teil des *Java.awt package.*
  - Klassenhierachie:

java.lang.Object
↳ java.awt.Component
↳ java.awt.Container
↳ java.awt.Panel
↳ java.applet.Applet

6.2 Applets kreieren

- Im Gegensatz zur Applikation gibt es bei Applets keine ausgezeichnete Signatur *main()*.
  Die Klasse *Applet* stellt eine Anzahl *"activity-methods"* bereit.
  Will bzw. muss man diese Methoden für das eigene Applet nutzen, so muss man sie *"überschreiben" (overriding)*.
- *activity methods:*
  - zur Initialisierung

```
public void init(){
...
}
```

Diese Routine wird ähnlich der Methode *main()* nach einem Ladevorgang (oder *reloading)* aufgerufen. Sie dient der Initialisierung, der Bereitstellung von übergebenen Parametern, etc.

6.2 Applets kreieren

- *activity methods(cont)*:
  - Starten

```
public void start() {
...
{
```

Nach der Initialisierung wird das Applet gestartet. Dies kann während des Lebenszyklus eines Applets, im Gegensatz zur Initialisierung, mehrfach vorkommen, z. B. falls ein "Leser" eine Seite verlassen hat und später wieder auf sie zurückkommt.

  - Stoppen

```
public void stop() {
...
{
```

Das Gegenteil von *start();* z. B. beim Verlassen der Seite.

6.2 Applets kreieren

◆ *activity methods(cont)*:

- Zerstören

```
public void destroy() {
...
{
```

Diese Methode dient der geordneten Terminierung eines Applets und setzt das Applet selbst in die Lage, seine genutzten Ressourcen an das Ressourcemanagement zurückzugeben.

6.2 Applets kreieren

◆ Ein vollständiges Beispiel:

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Event;

public class SimpleClick extends Applet {
    StringBuffer buffer;

    public void init() {
    buffer = new StringBuffer();
        addItem("initializing... ");
    }
    public void start() {
        addItem("starting... ");
    }
    public void stop() {
        addItem("stopping... ");
    }
    public void destroy() {
        addItem("preparing for unloading...");
    }
    void addItem(String newWord) {
        System.out.println(newWord);
        buffer.append(newWord);
        repaint();
    }
```

6.2 Applets kreieren

```
    public void paint(Graphics g) {
      //Draw a Rectangle around the applet's display area.
       g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);

      //Draw the current string inside the rectangle.
        g.drawString(buffer.toString(), 5, 15);
    }

    public boolean mouseDown(Event event, int x, int y) {
      addItem("click!... ");
      return true;
    }
}

-----------------------------------------------------------------
<html>
<head>
<title>SimpleClick</title>
</head>
<body>
<p>Output Simple Click:
<applet code="SimpleClick.class" width=600 height=25>
</applet>
</body>
</html>
```


6.3 *Painting*

- *Painting*

```
public void paint(Graphics g) {
...
{
```

Diese Methode wird immer dann benutzt, wenn irgendetwas auf den Screen auszugeben ist, z. B. wenn das Fenster aus dem Hintergrund wieder in den Vordergrund kommt oder verschoben wird, das Applet selbst Daten ausgeben will, etc. Diese Methode ist in der Klasse *Component* definiert und wird üblicherweise überschrieben.

Als Parameter wird eine Instanz der Klasse *Graphics* übergeben, die bereits instantiiert wurde. Wir müssen allerdings sicherstellen, dass die Definition dieser Klasse, die Teil des *java.awt packages* ist, auch dem Compiler zu Verfügung steht. Dies geschieht mit dem Import-Statement:

```
import java.awt.Graphics;
```

6.3 Painting

◆ Ein Beispiel:

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class HelloAgainApplet extends java.applet.Applet {
  Font f = new Font("TimesRoman",Font.BOLD,36);

  public void paint(Graphics g) {
    g.setFont(f);
    g.setColor(Color.red);
    setBackground(Color.green);
    g.drawString("Hello again!", 5, 40);
  }
}
```


6.3 Painting

◆ Painting

- Dieses Applet gibt einige Buchstaben auf den Screen aus. Es ist also notwendig, eine eigene *paint()-method* zu implementieren, die die *default*-Methode überschreibt. Da sonst nichts passiert, kann auf das Überschreiben der anderen Methoden verzichtet werden.
- Der Zustand des *Graphics-Objects*, das *paint()* als Parameter übergeben wird, beschreibt den Graphikstatus des Applets, wie z. B.
  - *colors oder*
  - *fonds*
- Mit den Anweisungen

```
g.setFont(f);
g.setColor(Color.red);
```

  wird dem *Graphics-Object* ein neuer Fontsatz und ein *Color-Object,* welches die Farbe ‘rot’ repräsentiert, zugewiesen.

6.3 Painting

- Painting
  - Die *drawString()-method* gibt den *String* im 1. Parameter aus, gemäß *“f”* und *“color.red”,* an Position x = 5(points) und y = 40(points), beschrieben durch den 2. bzw. 3. Parameter.

6.4 Applet - HTML - Umgebung

- Die Referenz auf Java-Applets wird, wie wir bereits wissen, in HTML-Seiten eingebettet. Dafür stellt HTML das *Applet-Tag* bereit.
- Seit spätestens HTML 4.0 sollten für *Tags* nur noch Kleinbuchstaben verwendet werden. Möglicherweise ist das noch nicht in allen Ihnen zur Verfügung gestellten Beispielen der Fall. Die Funktionalität wird dadurch z. Zt. noch nicht beeinflusst.

```
<html>
<head>
<title>This page has an applet on it</title>
</head>
<body>
<h2>Hello Again</h2>
<p>My second Java applet says:
<br><applet code="HelloAgainApplet.class" width=200 heigth=50>
</applet><p>
<a href="HelloAgainApplet.java">The Source</a>
</body>
</html>
```

6.4 Applet - HTML - Umgebung

- Dem *Applet-Tag* werden verschiedene Attribute mitgegeben
  - **`code=filename.class`**
    Name der Datei, die den Code des Applets enthält. Im Gegensatz zum JDK muss hier die Extension “class” mit angegeben werden.
    `code` wird verwendet, wenn sich HTML-Datei und Applet-Datei in der gleichen Directory befinden. Ist dies nicht der Fall, gibt man an, wo diese Datei zu finden ist. Hierzu verwendet man
  - **`codebase`** mit
    einer URL oder einem relativen Pfadnamen:

```
<applet code=”filename.class” codebase=”../applets”
width=120 heigth=100></applet>
oder
<applet code=”filename.class”
codebase=”http://www4/informatik.uni-erlangen.de/j-applets”
width=120 heigth=100></applet>
```


6.4 Applet - HTML - Umgebung

- Dem Applet-Tag werden verschiedene Attribute mitgegeben(cont)
  - **`width, height`**
    Gibt die Größe der Applet-Box an; die Maßeinheit ist *“pixel”*.
  - **`align`**
    `left, right, top, texttop, middle, absmiddle, baseline, bottom` oder **`absbottom`**
  - href (*Link*)
    Der Text aus der angegebenen Datei - hier der Sourcecode des Java-Applets - wird beim Anklicken des *Links* dargestellt.

6.4 Applet - HTML - Umgebung

- **`align`**-Beispiel
  (Sie sollten die anderen Möglichkeiten selbst einmal durchspielen!)

```
<html>
<head>
<tiltle>This page has an applet on it, aligned left</title>
</head>
<body>
<h2>Hello Again (Align)</h2>
<p><APPLET CODE="HelloAgainApplet.class"
width=200 height=50 align=left>Hello Again!</applet>
To the left of this paragraph is an applet. It's a
simple, unassuming applet, in which a small string is
printed in red type, set in 36 point Times bold.
<br clear=all>
<p>In the next part of the page, we demonstrate how
under certain conditions, styrofoam peanuts can be
used as a healthy snack.
<p>
<a HREF="HelloAgainApplet.java">The Source</a>
</body>
</html>
```

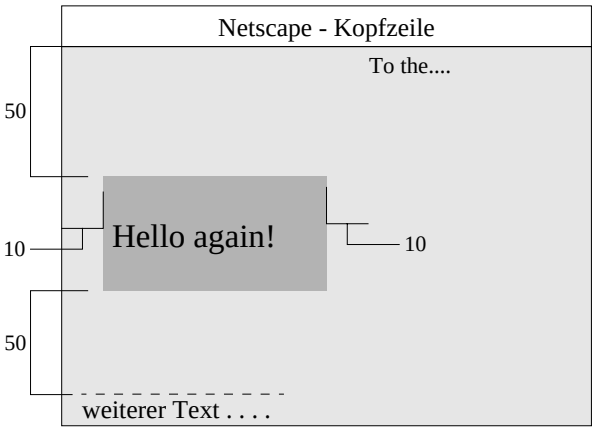

6.4 Applet - HTML - Umgebung

- Dem Applet-Tag werden verschiedene Attribute mitgegeben(cont)
  - **`hspace, vvspace`**
    Abstand in *Pixel* zum “umfließenden Text”; zum Beispiel:

```
<html>
<head>
<title>This page has an applet on it, with space around it</title>
</head>
<body>
<h2>Hello Again (Space)</h2>
<p><applet code="HelloAgainApplet.class" width=200 height=50
align=left vspace=50 hspace=10>Hello Again!</applet>
To the left of this paragraph is an applet. It's a
simple, unassuming applet, in which a small string is
printed in red type, set in 36 point Times bold.
<br clear=all>
<p>In the next part of the page, we demonstrate how
under certain conditions, styrofoam peanuts can be
used as a healthy snack.
<p>
<a href="HelloAgainApplet.java">The Source</a>
</body>
</html>
```

6.4 Applet - HTML - Umgebung

- Dem Applet-Tag werden verschiedene Attribute mitgegeben(cont)
  - **`hspace, vspace`**



6.4 Applet - HTML - Umgebung

- Parameterübergabe an Applets
  - Zur Bereitstellung der Parameter gibt es das *HTML-Tag*
    - **`param`**
      mit den beiden Attributen
      - ***`name`***; Name des Parameters und
      - ***`value`***; Wert des Parameters

```
<apple code="filename.class" codebase="../applets"
                                        WIDTH=120 HEIGHT=100>
<param name="font" value="TimesRoman">
Pparam name="size" value="36">
</applet>
```

  - Zur Übernahme der Parameter verwendet man in der *init()-method* die Methode *getParameter()*, z. B.

```
String theFondName = getParameter("font");
```

Aufgabe: Ändern Sie "HelloWorldApplet" so ab, dass Sie Fonttyp und Fondgröße als Parametern übergeben!