Softwarepraktikum: Enigma

Martin Steffen

Sommersemester 2003

Abschnitt I

Applets

Inhalt: Es geht um die Programmierung von Applets/ Parameterübergabe.

Literatur: Die Rohinformation zu diesem Abschnitt bieten die Klassen aus dem Paket `java.applet` bzw. die Klasse `javax.swing.JApplet`. Dazu Kapitel 7 und 8 aus [Fla99a] und Kapitel 4 aus [Fla97]. Die Beschreibung des `jar`-Tools findet sich im Kapitel [Fla99b, Kapitel 16] und den Abschnitt über Jar in [CW96].

Allgemeines

- "Mini"-Applikationen, oft zum download vom Netz
- Applets: Unterklassen von `java.applet.Applet`, bzw. `javax.JApplet`
- Applets stehen unter der Kontrolle des *Appletviewers/Browsers*

  $\Rightarrow$ keine `main()`-Methode notwendig
  $\Rightarrow$ keine Command-line-Parameterübergabe, dafür `<PARAM>`-tags
  $\Rightarrow$ eine Reihe von Standardmethoden, die der Appletviewer aufrufen kann, wie z.B. `paint` und die überschrieben werden
  $\Rightarrow$ Applet muß auf diese Methodenaufrufe prompt antworten, d.h., für *Animationen* benötigt es Threads.

- Sonstiges:
  - Sicherheitseinschränkungen für Applets
  - digitale *Signaturen*

Erstes Applet

```
import java.applet.Applet;                    // Importieren
import java.awt.*;

public class FirstApplet extends Applet {   // JApplet Analog
  public void paint (Graphics page) {
    System.out.println("paint");             // wird aufgerufen wenn
                                              // applet sichtbar wird, neu startet etc
    page.drawString ("Hello World", 50, 50);
  }                                           // Methode paint
}                                             // Klasse FirstApplet
```

- jedes Applet muss auf jeden Fall eine der Methoden

  `init()`, `start()`, oder `paint()`

  überschreiben. `main()` nicht notwendig

Einfaches Applet

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

public class MessageApplet extends Applet { //geht analog mit JApplet
  protected String message;
  protected Font font;

  public void init(){  //beim Laden in den Viewer
    message = this.getParameter("message");
    font = new Font("Helvetica", Font.BOLD, 48);
  }

  public void paint(Graphics g) {
    // could be done nicer with java.awt.Graphics2D
    g.setColor(Color.red);
    g.drawOval(10,10,330,100); g.drawOval(9,9,332,102);
    g.setColor(Color.black);
    g.setFont(font);
    g.drawString(message,40,75);
  }
}
```

java.applet-**Paket,** javax.JApplet

- relativ kleines Paket, Hauptklasse `Applet`
- Hierarchie:

```
Component => Container => Panel => Applet => swingx.JApplet
```

  viele (auch) für Applets wichtige Methoden finden sich in vor allem `Component`

- Vorgehen: appletviewer "bedient" das Applet ⇒ Unterklassenbildung von `(J)Applet` + überschreiben der "Steuer"methoden des Applets
- Swing-Komponente `javax.JApplet`,
  - Swing-fähige Variante von applets
  - als Toplevel-Komponente (wie `JFrame`, `JWindow` etc.):

- Kind `JRootPane`, man bekommt es mittels `getContentpane`[1]
- implementiert `RootPaneContainer`

[1] Vergleiche auch das `Frame`-Beispiel aus der vorangegangenen Woche

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



public class AppTester extends JApplet {
  public void init () {
    Container c = getContentPane();    // get the  RootPaneContainer
                                       // to add the other components
    JButton jb = new JButton ("Default");
    c.add (jb, BorderLayout.WEST);
    jb = new JButton ("LayoutManager");
    c.add (jb, BorderLayout.CENTER);
    jb = new JButton ("is");
    c.add (jb, BorderLayout.EAST);
    jb = new JButton ("BorderLayout: " +
                      (c.getLayout() instanceof BorderLayout)); // boolean test
    c.add (jb, BorderLayout.SOUTH);
  }
                                  }
```

Wichtige Methoden bei Applets

Name	Ausführung bei	typische Verwendung
`init()`	Laden des Applets	Initialisierung, Parameterübergabe. Funktion ähnlich einem Konstruktor
`start()`	Sichtbar-Werden	Malen, Starten der Animation
`destroy()`	Unload	Recourcenfreigabe
`stop()`	Unsichtbar-Werden	temp. Unterbrechen (z.B. der der Animation)
`getAppletInfo()`		Darstellbar in Dialogboxen
`getParameterInfo()`		Parameter des Applets
Aus Oberklassen (`Object => Component => Container => Panel => Applet`)		
`repaint(_)`		(`Component`), ruft `update` auf, welches den Bildschirm löscht und `paint` aufruft
`paint(Grapics)`		Malen (`Container`)
`print(Graphics)`		Drucken

Tabelle 1: Wichtige Methoden “an” Applets

Applet-Lebenszyklus

```
import java.awt.Graphics;                    // abstract class, for general graphics

public class Lebenszyklus extends java.applet.Applet {
  StringBuffer buffer = new StringBuffer();  // from java.lang

  public void init () {resize(500, 20);addItem("Initializing ...");};
  public void start (){addItem("Starting ...");};
  public void stop(){addItem("Stop ...");};
  public void destroy(){addItem("Preparing for unloading ...");};

  public void addItem(String meldung) { // print some message
    System.out.println(meldung);        // on ``stdout'' and
    buffer.append(meldung);             // onto the graphics
    repaint();                          // orig. from abstr. class Component
  };                                    // calls paint

  public void paint(Graphics g) {       // overriding paint from Container
    g.drawRect(0,0,getSize().width -1, getSize().height -1);
    // size deprecated in 1.2
    g.drawString(buffer.toString(), 5, 15);
  };
};
```

Häufig in Applets verwendete Methoden

Die Klasse *Applet* stellt auch Methoden zur Verfügung, die man häufig in Applets brauchen kann.

`Image`	`getImage(URL)`	Laden von Bildern
`String`	`getParameter(String)`	für Parameterübergabe aus HTML-Seite
`URL`	`getDocumentBase()`	Url der HTML-Seite
`URL`	`getCodeBase()`	Url der Klasse
	...	

Tabelle 2: In Applet definierte Methoden (Auswahl)

Parameter von Applets

- Parameterübergabe aus HTML-Seiten
- Lesen der Parameter in der `init()`-Methode, mittels `getParameter(<param_string>)`
- HTML-Fragment:

```
<applet code=myclass.class width=200 height=200>
<param  name="bild"  value="meinbild.gif">
....
</applet>
```

Applets in HTML-Seiten

- mittels des <APPLET>-Tags

```
<APPLET
    CODE = applet-filename
    WIDTH = pixel-width
    HEIGHT = pixel-height
    [OBJECT = serialized-applet-filename]
    [ARCHIVE = jar-file-list]
    [CODEBASE = applet-url]
    [ALT = alternate-text]
    [NAME = applet-name]
    [ALIGN = alignment]
    [VSPACE = vertical-pixel-space]
    [HSPACE = horizontal-pixel-space]
>

[<PARAM NAME = parameter VALUE = <value>]
[<PARAM NAME = parameter VALUE = <value>]
...
[alternat-text]
</APPLET>
```

Applet-Tag

- Notwendige Angaben
  - Code = Name der bytecode-Datei[2], Höhe, Breite[3]
- optional: `ARCHIVE`: Liste von JAR-files, mit Komma getrennt
- vor dem Applet geladen (z.B. weitere Klassendateien, Bilder etc.)
- optional: `CODEBASE` in welchem Verzeichnis liegt der Code
- optional zum Ausrichten: vertikaler und horizontaler Abstand, Ausrichtung

[2] ohne `.class`.

[3] Alternativ kann man auch ein *serialisiertes Applet-Objekt* laden, dort wird die `init()`-Methode nicht aufgerufen, nur `start()`

Applets & Jar-Dateien

- JAR = *Java Archive*
- `ARCHIVE`-HTML-Attribut
- Tool: `jar`[4]
- Aufruf zum Erstellen:[5]

```
jar cvf myapplet.jar *.class <paket>/*.class *.gif *.au
jar cvfm myapplet.jar META-INF/MANIFEST.MF *.class
```

- `Manifest`
  - "Meta"-Informationen über das Archiv

[4] auf die 1.4-Version achten

[5] `c` für create, `f` für file, `m` für Manifest

  - liegt standardmäßig in `./META-INF/MANIFEST.MF`
  - kann verschiedenes Info's enthalten, Autor, Signieren etc. Wichtig für Standalone-Versionen (keine Applets): Angabe der Hauptdatei

    `Main-Class: <paket>.<KLASSE>`

- Einbetten:

```
<APPLET ARCHIVE = "myapplet.jar"
CODE="myapplet.class"
WIDTH=100 HEIGHT=100>
</APPLET>
```

Applets als Stand-alone Programme

- Applets müssen immer irgendwo eingebettet sein
- dies wird jetzt nicht vom *Appletviewer/Browser gemacht:* $\Rightarrow$ `Frame` selber erzeugen (auch Größe setzen)
- Darstelle des `Frames: show`
- Aufrufen der Applet-`init`-Methode durch `main`

folgendes Beispiel erweitert (und vereinfacht) aus [Fla97]

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

public class StandaloneScribble extends Applet { // standalone + applet!
  public static void main (String[] args) {   //  Application need main method!
    Frame f = new Frame();                    // surrounding heavyweight compontent
    Applet a = new StandaloneScribble();      // ``self''-call!
    f.add(a, "Center");                       // Put the applet to main window (frame)
    a.init();
    f.setSize(400, 400);
    f.show();                                 // show l"a"st einen Frame erscheinen
    f.setBackground(bgcolor);
    f.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
          System.exit(0);};
    });
  };                                          // end of main


  public void init() {          // auch als applet verwendbar
    // add the Listeners to the applet
    this.addMouseListener(new MouseAdapter() {
        public void mousePressed(MouseEvent e) {
          lastx = e.getX();                 //
          lasty = e.getY();
```

```
        };
      });

    this.addMouseMotionListener(new MouseMotionAdapter() {  // anon. class
        public void mouseDragged(MouseEvent e) {            // overwrite
          Graphics g = getGraphics ();
          int x = e.getX();
          int y = e.getY();
          g.setColor(StandaloneScribble.this.drawcolor);
          // wiederum: this alleine geht nicht!
          g.drawLine(lastx, lasty, x, y);
          lastx = x; lasty = y;
        };
      });


    Button b = new Button("Blau");
    b.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          drawcolor = Color.blue;
        };
      });

    this.add(b);
    // Left out the other buttons

  };                                  // end of init
  protected int lastx, lasty;
```

```
  protected static Color bgcolor   = Color.white;
  protected Color drawcolor = bgcolor;
};
```

Literatur

[CW96] Mary Campione and Kathy Walrath. *The Java Tutorial*. The Java series. Addison-Wesley, 1996.

[Fla97] David Flanagan. *Java Examples in a Nutshell*. O'Reilly, 1 edition, September 1997.

[Fla99a] David Flanagan. *Java Foundation Classes in a Nutshell*. O'Reilly, 1 edition, September 1999.

[Fla99b] David Flanagan. *Java in a Nutshell*. O'Reilly, 3 edition, November 1999.