E5 Java-Applets

Die Applet-Welt

Applets sind in Java geschriebene "kleine" Applikationen, die *in eine HTML-Seite eingebettet* sind. Mit dieser werden sie von einem lokalen Rechner oder aus dem Inter- / Intranet von einem Webserver geladen und im *Browser* des Webclients ausgeführt und dargestellt. Dazu verfügen alle gängigen Browser über eine eingebaute JVM. Innerhalb des HTML-Codes kann genau spezifiziert werden, von welcher Quelle das Applet zu laden ist und welcher (rechteckige) Bereich im Browserfenster ihm zur Ausführung und Darstellung zur Verfügung steht.

Applets sind ein Mittel, um Webseiten mit dynamischen interaktiven Programminhalten zu versehen. Innerhalb des Appletscodes steht der gesamte J2SE-Sprachumfang zur Verfügung – insbesondere auch alle Möglichkeiten der Grafikprogrammierung. Da wir im Rahmen dieses Buches auf Grafik nicht eingehen, beschränken wir uns auf Applet-spezifischen Programmstrukturen und Abläufe. Zur Vertiefung: [POM05].

Jede aus dem Internet geladene HTML-Seite (eventuell unbekannten Ursprungs) kann Applets enthalten. Applets unterliegen im Vergleich zu lokalen Java-Applikationen *besonderen Sicherheitsbeschränkungen* beim Zugriff auf Rechnerressourcen, um Daten und Funktionsfähigkeit des ausführenden Systems zu schützen.

Programmtechnisch ist ein Applet eine Java-Klasse, die von der J2SE-Klasse `java.applet.Applet` erbt und spezifische Methoden implementiert, die in definierter Reihenfolge bei der Ausführung des Applets aufgerufen werden.

Applets sind nicht für sich (standalone) lauffähig (sie besitzen keine `main()`-Methode), sondern werden im Browser geladen und ausgeführt, wenn die umgebene HTML-Seite prozessiert wird und der Browser auf ein `<applet>` Tag stösst. Auch außerhalb des Browsers können Applets mit dem JDK-Tool **appletviewer** (aus *bin*-Verzeichnis) getestet werden – allerdings wird dabei nur das reine Appletcoding ausgeführt, nicht jedoch das umgebende HTML dargestellt.

Aufrufsyntax: `appletviewer [options] urls`

Als `urls` lassen sich ein oder mehrer HTML-Dokumente angeben, z.B. `appletviewer Bank.html`

Kurzeinführung HTML

- **HTML** = Hyper Text Markup Language - eine *Textauszeichnungssprache*, d.h. HTML zeichnet Inhalte von Dokumenten aus wie Überschriften, Textabsätze, Tabellen etc.
- HTML wird für Darstellung von Web-Seiten verwendet. HTML ist *nicht* case sensitive
- Eine HTML-Seite gliedert sich grundsätzlich in zwei Teile: **head** und **body**
  - Im **head-Teil** finden sich Informationen zur HTML-Seite (META-Daten), sowie der Titel der Seite.
  - Im **body-Teil** steht der eigentliche Inhalt der Seite. Nur dieser Teil wird vom Browser optisch dargestellt.
- **HTML-Tags:** Der Inhalt von HTML-Dateien steht in **HTML-Elementen**. HTML-Elemente werden durch so genannte **Tags** (Markierungen) markiert. *Beispiele:*
  - **<h1>** Das ist eine Überschrift 1.Ordnung **</h1>**
  - **<b>** Dieser Text ist fett **<b>**
  - Hiernach wird die Zeile umgebrochen: **
** Weiter in der nächsten Zeile!
  - Tags bestehen meist aus einem ***Anfangs*-Tag** und einem ***End*-Tag**. Es gibt es aber auch „Standalone“-Tags, z.B. **
** für den Zeilenumbruch.

Einbetten von Java-Applets in HTML 3.2 : `applet` Tag

- Applets werden über Tag **`<applet>`** im `body`-Teil einer HTML-Seite eingebunden. Ausführung geschieht stets durch die Browser-JVM. Alle notwendigen Applet-Angaben werden in diesem Tag vermerkt. Dient als Schnittstelle zwischen HTML-Seite und Applet.
  - Zwischen Anfangs- und End-Tag kann Text stehen, der angezeigt wird, falls Ausführung des Java Applets fehlschlägt. Die im Attribut `code` genannte Klasse enthält Bytecode des Applets. Speicherort kann im Attribut `codebase` angegeben werden. Ein rechteckiger Bereich in HTML-Seite ist zur Darstellung vorgesehen, dessen Abmessungen mit Attributen `width` und `height` definiert sind. Die Browser-JVM instantiiert das Applet und ruft dessen `init()` -Methode auf.

```
<applet code = "Klassenname"    codebase = "URL"
        archive = "JAR-File"     name = "..."          align = "..."
        width = ...    height = ...    vspace = ...     hspace = ...
>
<param  name = "Parametername"  value = "Parameterwert" >
    Ersatztext
</applet>
```

- Notwendige Attribute :
  - **`code`** Dateiname des Applets-Bytecodes inclusive Extension `.class`
  - **`width`** Breite des Applets in Pixeln **`height`** Höhe des Applets in Pixeln
  - Bsp:

```
< applet code = "HelloWorld.class"   width = 300  height = 100  >
    Ausführung fehlgeschlagen!
</applet>
```

Einbetten von Java-Applets in HTML 3.2 : `applet` Tag

- Optionale Attribute :
  - **`codebase`** Speicherort des Applet-Bytecodes im Filesystem: Klassenpfad (URL) für das Applet und den von ihm benutzten Dateien. Angabe relativ zum Verzeichnis der HTML-Seite oder absoluter Pfad. Kann weggelassen werden, wenn sich das Applet im selben Verzeichnis wie die HTML-Seite befindet.
    - Attribut ermöglicht, ein Applet aus irgendeinem Ort im Internet innerhalb einer HTML-Seite zu referenzieren
    - Wird `codebase` angegeben, dann ist der unter `code` angegebene Dateiname relativ zum `codebase`-Pfad
  - **`archive`** Name des JAR-Files, das den Bytecode enthält. Alle vom Applet benötigten Klassen können in ein `.jar`-File gepackt und somit in einem *einzigen* HTTP-Zugriff vom Server geladen werden, was die Ladezeit des Applets deutlich reduziert.
  - **`name`** Name des Applets unter dem dieses z.B. durch andere Applets auf HTML-Seite oder durch JavaScript referenziert werden kann
  - **`align`** Ausrichtung des Applets innerhalb HTML-Seite. Mögliche Werte: `left, right, top, middle, texttop, absmiddle, baseline, bottom, absbottom.`
  - **`alt`** Text, den der Browser als Alternative ausgibt, der `applet` Tag versteht, jedoch Applet nicht ausführen kann
  - **`vspace`** Vertikaler Freiraum um Applet (in Pixeln). Nur gültig, wenn `align = left` oder `right`
  - **`hspace`** Horizontaler Freiraum (in Pixeln). Nur gültig, wenn `align = left` oder `right`

Parameterübergabe an Applets: <param> Tag

- Im <applet> Tag können **Parameter** an Applet übergeben werden. **<param>** Tag definiert Übergabewert, durch den HTML-Seite einen Parameterwert an Applet-Code übergibt. Beliebig viele <param> Tags können auftreten.
- Parameter werden im Appletcode abgerufen mittels Methode:
  - **String** Applet.**getParameter(String name)**
  - Parameter werden als String übergeben und von als String zurückgegeben
- Angaben: **<param** name="[name]" value="[wert]" >
  - **name** Parametername, muss mit Parameternamen im Appletcode übereinstimmen
  - **value** Wert der an Applet übergeben werden soll.

```
<applet code = "MyApplet.class"   codebase ="../applets/classes"
        archive = "Demo.jar"
        name = "MyApplet"         align = top
        width = 200               height = 100
        vspace = 10               hspace = 20        >
<param name = "Gruss"   value = "Hallo miteinander">
   Das Applet sollte nur eine kurze Grussformel ausgeben
</applet>
```

Ab HTML 4.0: Einbetten von Applets alternativ mittels `object` Tag

- Applets und multimediale Objekte können über Tag **`<object>`** im `body` einer HTML-Seite eingebunden werden. Mit diesem Tag kann anstelle der Browser-JVM eine andere JVM als Plug-In zum Einsatz gebracht werden.
- Notwendige Attribute :
  - **`width`** Breite des Applets in Pixeln **`height`** Höhe des Applets in Pixeln
  - **`classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"`**
    ClassID des Java-JVM-PlugIns, sollte immer diesen Wert tragen
- Optionale Attribute des `object` Tags:
  - **`codebase`** URL der auf (als Plug-In zu ladende) JVM verweist
  - **`align`** Ausrichtung des Applets innerhalb HTML-Seite. Mögliche Werte: left, right, top, middle, texttop,absmiddle, baseline, bottom, absbottom.
  - **`vspace`** Freiraum um Applet (in Pixeln) in vertikaler Richtung
  - **`hspace`** Freiraum um Applet (in Pixeln) in horizontaler Richtung
  - **`name`** Name des Applets unter dem dieses z.B. durch JavaScript referenziert werden kann
- Weitere Applet-Informationen werden als *Parameter* angegeben :
  - **`param name="codebase" value="..."`**
  - **`param name="code" value="..."`**
  - **`param name="archive"  value="..."`**

Einbetten von Java-Applets in HTML 4.0 : `object` Tag


```
<object classid = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        width = 300    height = 100
>
<param name = "code"      value = "MyApplet.class" >
<param name = "codebase"  value = "../applets/classes">
<param name = "archive"   value = "Demo.jar">
<param name = "Gruss"     value = "Hallo miteinander">
   Das Applet sollte nur eine kurze Grussformal ausgeben
</object>
```

Die Klasse Applet

Jede implementierte Applet-Klasse ist Unterklasse der J2SE-Klasse `java.applet.Applet`.

Jedes Applet enthält somit den Code:

```
import java.awt.Applet;
public class MeinApplet extends Applet { /* ... */ }
```

Die Klasse **`java.applet.Applet`** ist direkt von **`java.awt.Panel`** abgeleitet und verfügt über die Merthoden und Attribute der folgenden **Vererbungshierachie:**

```
java.lang.Object
   |
   +--java.awt.Component
      |
      +--java.awt.Container
            |
            +--java.awt.Panel
                  |
                  +--java.applet.Applet
```

Abstract Window Toolkit (**AWT**) gehört zu J2SE-Paket **`java.awt`**

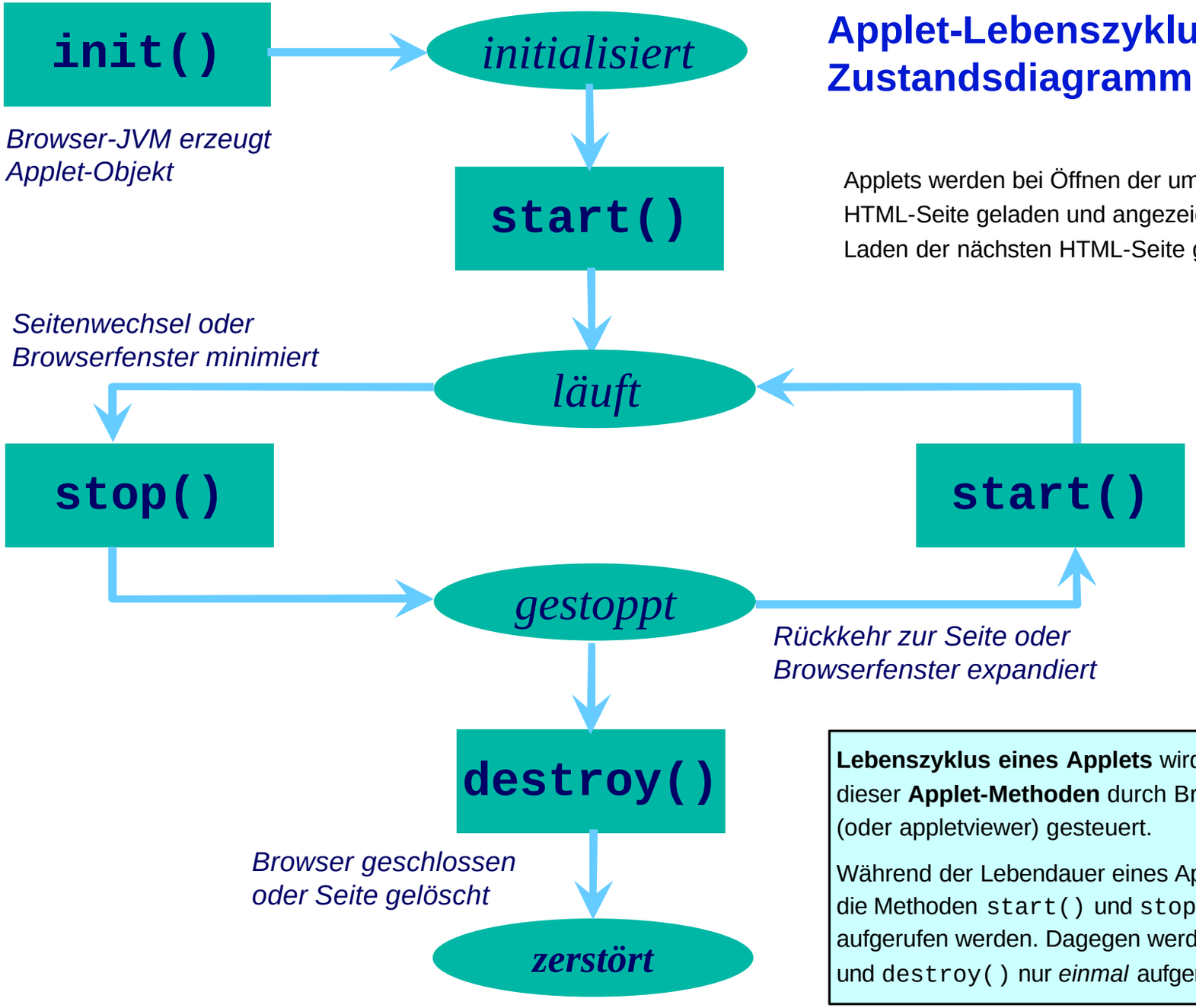
Enthält Grafik-Klassen und Methoden zur Layout-Gestaltung und Verwendung interaktiver Oberflächenelementen (wie Buttons, Textfelder, Scrollbars etc) sowie Fenstern und Dialogen.

Wird teilweise auch innerhalb Applet-Programmierung verwendet.

Klasse `java.applet.Applet` stellt einen Container dar ⇒ Im Zeichenbereich des Applets können wie in einem Fenster andere Komponenten dargestellt werden

Viele Grafikfunktionen stehen zur Vfg; können durch Objekt (Grafik-Kontext) der Klasse `java.awt.Graphics` angesprochen werden

Applet-Lebenszyklus: Zustandsdiagramm



Applets werden bei Öffnen der umgebenden HTML-Seite geladen und angezeigt und beim Laden der nächsten HTML-Seite gestoppt.

Lebenszyklus eines Applets wird durch Aufruf dieser **Applet-Methoden** durch Browser-JVM (oder appletviewer) gesteuert.

Während der Lebendauer eines Applets können die Methoden `start()` und `stop()` *mehrfach* aufgerufen werden. Dagegen werden `init()` und `destroy()` nur *einmal* aufgerufen.

Methoden eines Applets

Applet benötigt keine `main()`-Methode. Falls doch vorhanden wird sie bei Ausführung innerhalb Applet-Lebenszyklus nicht berücksichtigt.

- **`public void init()`**
  - Wird nur **einmal** während der Lebenszeit des Applets beim Laden des Applets aufgerufen. Dient z.B. zum Anlegen und Initialisieren von verwendeten Objekten und Daten, Abfragen und Auswerten von Parametern , Laden von Bildern oder anderen Medien, Aufbau und Initialisieren der Benutzeroberfläche. (Quasi ein Konstruktorersatz)
- **`public void start()`**
  - Aufruf nach `init()` und jedesmal, wenn HTML-Seite und eingebettetes Applet erneut angezeigt wird. Nach (Neu-)Start des Applets und wenn Neuzeichnen des Applets (nach Verdecken) erforderlich ist. Es wird Methode `java.awt.Component.paint()` aufgerufen und von Browser-JVM mit Referenz auf Graphics-Objekt versorgt.
- **`public void stop()`**
  - Hält Applet-Ausführung nur an, dieses bleibt jedoch geladen. Aufruf z.B. wenn Browser minimiert, Applet aus sichtbarem Bereich gescrollt oder verdeckt oder Web-Seite verlassen wird. Sollte z.B. laufende Threads anhalten und Ressourcen freigeben, die nur das laufende Applet benötigt. Bei erneutem Aufruf der Applet-Seite wird Applet durch erneuten Aufruf von `start()` wieder gestartet.
- **`public void destroy()`**
  - Aufruf erfolgt nach `stop()`, z.B. wenn Browser geschlossen und HTML-Seite und Applet endgültig aus dem Speicher entfernt wird. Wenn Applet noch aktiv ist wird zuvor `stop()` aufgerufen. Sollte Ressourcen freigeben und alle Threads beenden. (Aufgaben werden jedoch größtenteils schon von Java-Garbage Collection übernommen.)

Jede Applet-Klasse ***erbt*** diese Methoden, muß sie jedoch durch ***Überschreiben*** mit konkretem Inhalt füllen. Bei Überschreiben bleibt Aufgabe der Methode und deren relative Aufrufreihenfolge erhalten. Es wird jedoch festgelegt, was ***konkret*** ausgeführt wird, sobald die Methode von Browser-JVM gestartet wird.

Ausgaben innerhalb Applet-Bereich: `paint()`

Ausgabe soll *nicht* auf Standardausgabe gehen, sondern im ***Zeichenbereich*** des Applets innerhalb HTML-Seite $\Rightarrow$ *nicht* durch `System.out.println()` erreichtbar.

Geerbte Methode `java.awt.Component.`**`paint()`** kann verwendet werden. Durch Überschreiben von **`paint()`** kann in Applet-Oberfläche gezeichnet werden

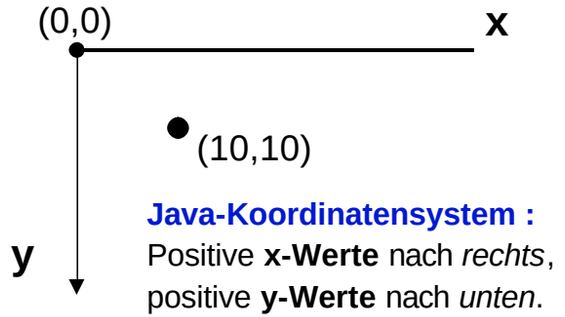
Methode `paint()` wird durch JVM jedesmal aufgerufen, wenn Komponente neugezeichnet wird (zB nach Verdecken des Fensters). Coding von `paint()` übernimmt Neuzeichnen.

Methode `paint()` erhält durch JVM **`java.awt.Graphics`**-Referenz **`g`** auf Applet-Zeichenbereich innerhalb HTML-Seite.

Auf `Graphic`-Referenz `g` stehen Zeichen- und Ausgabemethoden zur Vfg, zB Methode **`drawString()`** zur Textausgabe.

Parameter: 1. Auszugebender Text, 2. x-Koordinate, 3. y-Koordinate des Textes innerhalb Applet-Zeichenbereich.

```
import java.applet.Applet;
import java.awt.Graphics;
public class FirstTry extends Applet {
   public void init(){
   }
   public void paint( Graphics g ) {
        // Referenz auf Zeichenbereich verwendet
        g.drawString( "Hallo miteinander", 20, 20 );
   }
}
```



Java-Koordinatensystem :
Positive **x-Werte** nach *rechts*, positive **y-Werte** nach *unten*. Pixel-Angabe in ganzen Zahlen.

Implementieren eines minimalen Applets

1. Datei **`FirstTry.java`** anlegen mit Code :

```
import java.applet.Applet;
import java.awt.Graphics;
public class FirstTry extends Applet {
     public void init(){ /* ... */ }
     public void paint( Graphics g ) {
         g.drawString( "Applet läuft! ", 10, 10 );
     }
}
```

2. Datei kompilieren - Bytecode-Datei **`FirstTry.class`** wird erstellt

3. Eine HTML-Seite **`First.html`** erstellen und im *selben* Verzeichnis ablegen :

```
<html>
<body>
  Ab hier wird das Applet eingefügt   <br>
  <applet code="FirstTry.class"
          width=200    height=100
  >
     Problem bei Darstellung des Applets
  </applet>
</body>
</html>
```

4. Datei **`First.html`** im Browser öffnen, Java-Applet wird ausführt.

Demonstration des Applet-Lebenszyklus

Quelle: Goll, Weiß, Müller, "Java als erste Programmiersprache", Teubner, 2001

Werden verschiedene HTML-Seiten durch Browser aufgerufen und immer wieder zur Applet-HTML-Seite zurückgekehrt, kann man Veränderung der Zähler und somit Aufrufreihenfolge der Aufrufe der Applet-Methoden beobachten.

// Applet-Code `Lebenszyklus.java`

```
import java.applet.*;
import java.awt.*;
public class Lebenszyklus extends Applet {
   private int initZaehler = 0;
   private int startZaehler = 0;
   private int stopZaehler = 0;
   private int destroyZaehler = 0;

   public void init()    {  initZaehler++;  }
   public void start()   {  startZaehler++; }
   public void stop()    {  stopZaehler++;  }
   public void destroy() {  destroyZaehler++;  }
   public void paint( Graphics g ) {
        g.drawString( "Lebenszyklus:", 0, 10 );
        g.drawString( "init-Zaehler:  " + initZaehler, 0, 30 );
        g.drawString( "start-Zaehler:  " + startZaehler, 0, 50 );
        g.drawString( "stop-Zaehler:  " + stopZaehler, 0, 70 );
        g.drawString( "destroy-Zaehler:  " + destroyZaehler, 0, 90 );
   }
}
```

Übergabe von Parametern

1. Die HTML-Datei `Parameters.html`

```
<html>
<body>
 Applet wird eingefügt    <br>
 <applet code = "NextTry.class"
         width = 200   height = 200 >
 <param name= "info"   value = "Hallo allerseits!">
 </applet>
</body>
</html>
```

2. Der Applet-Code `NextTry.java`

```
import java.applet.Applet;
import java.awt.*;
public class NextTry extends Applet {
   private String text = "noch initial";
   public void init() {
        String temp = getParameter( "info" );
        if( temp != null ) {  text = temp;  }
   }
   public void paint( Graphics g ) {
        g.drawString( "Parameter: " + text, 30, 30 );
   }
}
```

Einbetten von Bildern

1. Die HTML-Datei `Picture.html`

```
<html>
<body>  Applet wird eingefügt   <br>
 <applet code="Bild.class"   codebase="."
         width=400   height=300    >
 </applet>
</body>
</html>
```

Bilder werden mit Methode **`Applet.getImage()`** ins Applet geladen. Mögliche Formate sind `.jpg` oder `.gif`

1. Parameter: Verzeichnis (URL-Adresse) von dem Bild geladen wird. Durch Methode **`Applet.getcodeBase()`** erhältlich, falls Bild und Applet im gleichen Verzeichnis liegen.

2. Parameter: Dateiname des Bildes

Sicherheitsaspekt: Applet kann nur Verbindung zu Server öffnen, von dem es geladen wurde. Deshalb ist zum Öffnen im Browser **Server-URL** anzugeben. Andernfalls wird Sicherheitsverletzung ausgelöst.

2. Der Applet-Code `Bild.java`

```
import java.awt.Image;
import java.awt.Graphics;
import java.applet.Applet;
public class Bild extends Applet {
 private Image abbildung;
 public void init() {
  abbildung = getImage( getCodeBase(), "foto.jpg" );
 }
 public void paint( Graphics g ) {
    g.drawImage( abbildung, 0, 0, getWidth(), getHeight(), this );
 }
}
```

Zeichnen des Bildes mittels Mehode **`Graphics.drawImage()`**

1. Parameter: Das zu zeichnende Image-Objekt

2. + 3. Parameter: x- und y-Koordinate der oberen linken Bild-Ecke relativ zu Applet-Zeichenbereich

4. + 5. Parameter: Breite und Höhe des Bildes. Mit Methoden `Applet.getWidth()` und `Applet.getHeight()`kann gesamte Breite und Höhe des Applet-Bereichs übergeben werden.

6. Parameter: Referenz auf Objekt einer Klasse, die Interface `ImageObserver` implementiert. Passiert in Klasse `Component`, von der `Applet` erbt. Somit kann mit `this` einfach Referenz auf `Applet`-Objekt selbst übergeben werden

Einbetten von Bildern in Applets

1. Methoden zum Importieren und Zeichen von Bildern :

   `Image Applet.`**`getImage( URL url )`** lädt Bild-Dateien

   `boolean Graphics.`**`drawImage()`** zeigt Bild-Dateien an

   // von `drawImage()` existieren mehrere überladene Versionen

2. Methoden zur Ermittlung der Server-URL (Adresse):

   (Rückgabewert `URL` ist Objekt der Klasse `java.net.URL`)

```
// Liefert URL des Applet-Verzeichnisses:
public URL getCode Base()
// Liefert URL des Verzeichnisses der
// zugehörigen HTML-Seite:
public URL getDocumentBase()
```

Bestimmung von Server-URL zur Laufzeit :

Applets dürfen *nur* Verbindungen zu Rechner aufbauen, von dem sie geladen wurden. Nur von diesem dürfen weitere Dateien + Ressourcen (z.B. Bilder) vom Applet nachgeladen werden. Deshalb kann auf Bilddateien *nicht direkt* zugegriffen werden, sondern nur über URL, die auf **Server** verweist, von dem HTML-Seite und Applet geladen wurde. Dazu dienen entsprechende Methoden der Klasse `Applet` :

Sicherheitsaspekte (Sandbox-Prinzip)

Unsignierte Applets unterliegen besonderen Sicherheitsrestriktionen. Auf lokalem Rechner installierter Java-Code wird als sicher betrachtet - jeglicher Java-Code von "außen" als unsicher und potentiell gefährlich. Vor diesem müssen alle lokalen Daten und Ressourcen geschützt werden.

- Zuständig für Sicherheit ist die "Sandbox" = Sicherheitskonzept für Java
- Sandbox = Klassenlader (ClassLoader) + Sicherheitsmanager (SecurityManager)
- Normalen Applikationen werden nur durch Klassenlader überwacht. Aus dem Internet geladener Programmcode bringt zusätzlich SecurityManger ins Spiel:
  - Security Manager kontrolliert, dass ein Applet keine Dateien lesen oder löschen und keine Dateien oder Verzeichnisse erstellen darf
  - ClassLoader kontrolliert, dass Applet nur Klassen vom eigenen Server lädt.
- Daraus ergeben sich folgende **Restriktionen** (für *unsignierte* Applets *ohne* digitale Signatur)
  - Im Dateisystem des lokalen Rechners darf weder gelesen noch geschrieben werden; temporäre Dateien dürfen nicht angelegt werden; Dateiinformationen dürfen nicht abgefragt werden.
  - Es dürfen keine neuen Verzeichnisse angelegt oder bestehene Verzeichnisse sowie deren Attribute verändert oder auch nur ermittelt werden.
  - Weiergabe von Informationen über das lokale System und dessen Benuter sind verboten.
  - Alle Operationen müssen im Speicher oder auf dem Server ausgeführt werden.
  - Applets dürfen keine Netzverbindung zu Fremdrechnern aufbauen, sie dürfen nur mit dem Internet-Server kommunizieren, von dem die Applet-Seite stammt.
  - Applets dürfen keine Programme starten oder Bibliotheken (DLLs) laden. JVM darf nicht beendet werden.
  - Druckerausgaben sind verboten.
  - Klassen der Java-Standardbibliothek `java.*` dürfen nicht überschrieben werden.

Übungen

1. Schreiben Sie eine HTML-Datei, die einen Text im Browser ausgibt. Öffnen Sie das HTML-File im Browser.

2. Schreiben Sie ein Applet das einen Text innerhalb des Appletbereichs der Breite 300 Pixel und Höhe 100 Pixel ausgibt.

Betten Sie in die HTML-Datei mittels `<applet>`-Tag das Applet ein und öffnen Sie die Seite im Browser. Das Applet sollte ausgeführt werden.

3. Führen Sie das Applet mit dem JDK-Tool `appletviewer` aus.

4. Erweitern Sie das Applet, so dass es mit einer Hintergrundfarbe dargestellt wird und führen Sie es im Browser aus.

Hinweis zu Übung 4: Nutzen Sie folgendes Codefragment :

```
import java.awt.Color;
import java.awt.Graphics;
// ..........................
public void init() {
   setBackground( Color.green );
}
// ........................
```

Beispiele zur Applet-Programmierung:

JDK-Verzeichnis: `\demos\applets`

`http://java.sun.com/applets/`

`http://www.jars.com`

Weiterführende Literatur: [POM05]