

# Java Kurzreferenz – Für Fortgeschrittene

## 1. Initialisierung von Applets: Die Methode init

Wenn ein Applet gestartet wird, werden immer zwei Methoden aufgerufen. Diese heissen `init` und `start`.

Um gewisse Berechnungen auszuführen, bevor das Applet gestartet wird, bietet es sich an, die Methode `init` zu verwenden. Dazu schreibt man eine eigene Methode `init`.

Die Methode `init` erhält keinen Parameter und gibt keinen Wert zurück. Sie sieht folgendermassen aus:

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

Die Methode darf nicht umbenannt werden.

## 2. Die Datentypen Graphics und Image

`Graphics` ist ein spezieller Datentyp, welcher einen Bildschirmausschnitt beschreibt und alle notwendigen Beschreibungen und Informationen zusammenfasst (Farben, Fonts, Linien, etc.).

Um eine Variable zu erhalten, die einen Bildschirmausschnitt zwischenspeichern kann, kann man folgendermassen vorgehen:

1. Erzeuge ein Bild als Ausgabe:

Um ein Bild zu erzeugen, kann die Methode: **`createImage`** verwendet werden. Sie erhält die Breite und die Höhe des zu erzeugenden Bildes als Parameter.

Beispiel:

```
Image meinBild;  
meinBild = createImage (fensterBreite, fensterHoeh);
```

2. Definiere einen Bildschirmausschnitt, der das Bild beschreibt:

Jedes Objekt der Klasse `Image` hat die Methode **`getGraphics`**, mit der man einen dazugehörigen Bildschirmausschnitt definieren kann.

Beispiel:

```
Graphics meinBildschirmAusschnitt;  
meinBildschirmAusschnitt = meinBild.getGraphics();
```

### 3. Verwende den Bildschirmausschnitt

Bei der Verwendung eines nach 1. und 2. erzeugten Bildschirmausschnittes können dieselben Methode verwendet werden wie für den Bildschirmausschnitt screen des Applets.

Um ein Bild (also ein Objekt vom Typ Image) auf dem Bildschirm darzustellen, gibt es die Methode **drawImage**. Sie erhält vier Parameter:

- das Bild, das dargestellt werden soll
- die x-Koordinate des Punktes, an den die linke obere Ecke des Bildes platziert werden soll
- die y-Koordinate des Punktes, an den die linke obere Ecke des Bildes platziert werden soll
- mit dem vierten Parameter kann man ein Objekt übergeben, das es erlaubt den Fortschritt der Darstellung zu überwachen. Wenn man dieses nicht übergeben möchte, so kann man auch **this** übergeben. Mit this verweist man auf das Applet, in welchem die Methode aufgerufen wird.

Um meinBild von der linken oberen Ecke ausgehend auf dem Bildschirm darzustellen, kann also folgender Methodenaufruf verwendet werden:

```
screen.drawImage (meinBild, 0, 0, this);
```

### 3. Farben

Es gibt unterschiedliche Modelle, um Farben zu beschreiben. In Java wird das RGB Farbenmodell verwendet. RGB steht für red, green, blue – rot, grün, blau.

Wie Ihr sicherlich wisst, können alle Farben aus den drei Grundfarben rot, blau und gelb hergestellt werden. Dabei tauchen die einzelnen Grundfarben in unterschiedlichen Anteilen auf. Da grün aus blau und gelb entsteht, kann man auf Basis von blau, grün und rot ebenfalls alle Farben darstellen. Im RGB Farbenmodell werden Farben beschrieben, indem man den Anteil von rot, blau und grün angibt, den man braucht, um diese Farbe zusammenzumischen..

Die Angaben erfolgen über ganze Zahlen. Es werden Werte zwischen 0 und 255 angegeben. 255 bedeutet, dass man von der Farbe so viel nimmt, wie möglich. 0 bedeutet, dass die Farbe nicht verwendet wird.

Die Werte für einige mögliche Farben werden in der folgenden Tabelle angegeben:

Farbe	rot	grün	blau
weiss	255	255	255
schwarz	0	0	0
rot	255	0	0
blau	0	0	255
gelb	255	255	0
grün	0	255	0
grau	127	127	127

Um eine Farbe zu verwenden, wird eine Methode aufgerufen, die eine Farbe erzeugt. Ihr werden die RGB Werte mitgegeben.

```
Color col; /* dies erzeugt eine Variable des Typs Color */
```

```
col = new Color (255, 0, 0); /* hier wird eine neue Farbe erzeugt (rot) und diese Farbe wird der Variablen Color zugewiesen */
```

Das Wörtchen **new** in Java wird immer dann verwendet, wenn man ein neues Objekt erzeugen möchte. Bei den Datentypen int, float und char braucht man diesen Befehl nicht, da es sich um einfache Datentypen handelt. Wenn sie erstellt werden, muss Java nichts tun, ausser den Speicherplatz zu reservieren.

Bei komplexeren Objekten benötigt man jedoch das new. In unserem Fall erlaubt es uns, dass nicht nur Speicherplatz für die Farbe bereitgestellt wird, sondern wirklich eine Farbe mit den angegebenen RGB Werten kreiert wird.

Um eine Farbe für die Ausgabe zu setzen, muss die Methode **setColor** aufgerufen werden.

Beispiel:

```
screen.setColor (col); /* setze die aktuelle Farbe auf rot */  
screen.drawLine (0,0,100,100); /* die hier gezeichnete Linie wird rot */
```

**Beachte:** Java bietet auch die Möglichkeit eine vordefinierte Farbe auszuwählen. In diesem Falle könnte man auch **screen.setColor(Color.red)** verwenden.

## 4. Ereignisse, Buttons und Textfelder

Ein User Interface, eine Benutzerschnittstelle, erlaubt es den BenutzerInnen, den Programm-durchlauf zu steuern. Dies erfolgt häufig mit Hilfe von Menüs, Buttons, Textfeldern, usw. Ein besonderes Merkmal an einem User Interface besteht darin, dass die Reihenfolge von Aktionen nicht vorhersehbar ist. Aus diesem Grund braucht man einen bestimmten „Mechanismus“, um ein User Interface realisieren zu können.

Dieser Mechanismus besteht in einem Ereignismodell. Eine Aktion, welche vom Benutzer/von der Benutzerin gewünscht wird, löst ein sogenanntes Ereignis aus. Dieses Ereignis wird von dem Programm abgefangen und führt dazu, dass sofort eine Methode aufgeführt wird, die zu diesem Ereignis gehört.

Auch in Java gibt es ein solches Ereignismodell, um Benutzerschnittstellen zu implementieren. In diesem Schnupperstudium werden wir die veraltete Version des Ereignismodells kennenlernen, da diese etwas einfacher realisiert werden kann. Wir benötigen für diese Version weniger Befehle und müssen uns das Ereignismodell nicht im Detail ansehen.

Zum besseren Verständnis kann folgende Vorstellung hilfreich sein: Parallel zur Ausführung des Apfelmännchen-Programmes wird ein zweites Programm (Ereignis-Programm) ausgeführt, welches nichts weiter tut, als auf Ereignisse zu warten. Wenn dann ein Ereignis eintritt, verweist das Ereignis-Programm auf eine bestimmte Stelle im Apfelmännchen-Programm, welche darauf vorbereitet ist, das Ereignis zu verarbeiten.

Ein Ereignis kann z.B. dann auftreten, wenn der Benutzer/die Benutzerin mit der Maus auf einen Button (ein Kontrollfeld) geht und die Maus betätigt.

Ein **Button** entspricht einem Java Objekt des Typs Button. Um einen Button zu kreieren, muss die Methode Button aufgerufen werden. Davor muss wieder das Wörtchen new stehen, das anzeigt, dass hier ein neues Objekt kreiert werden soll. Der Methode Button wird der Text mitgegeben, welcher auf dem Button erscheinen soll.

Beispiel:

```
Button meinButton = new Button (“Druecke mich”);
```

Ein Button ermöglicht es, eine Aktion auszulösen. Manchmal möchte der Benutzer/die Benutzerin dem Programm einen oder mehrere Werte übermitteln. Eine Möglichkeit, um dies zu tun, besteht in der Verwendung von Textfeldern. Ein Textfeld erlaubt die Eingabe von Zeichen durch BenutzerInnen.

Ein Textfeld ist vom Datentyp TextField. Die Methode, welche es kreiert, erhält zwei Parameter: den Wert und die Länge des Feldes.

Beispiel:

```
TextField meinTextFeld = new TextField (“5“, 2);
```

```
int wert = 5;
```

```
TextField meinTextFeld2 = new TextField (““ + wert,5);
```

Im ersten Beispiel wird ein Textfeld der Länge 2 kreiert, welches als initialen Wert den Wert 5 erhält. Auch das zweite Textfeld hat als initialen Wert den Wert 5. Es ist 5 Zeichen lang.

Mit Hilfe obiger Anweisungen kann man Buttons und Textfelder definieren. Diese müssen dann aber noch zum Applet hinzugefügt werden. Hierzu dient die Methode **add**. Sie erhält den Variablennamen des Objektes, welches dem Applet hinzugefügt werden soll.

Beispiel:

```
add (meinButton);  
add (meinTextFeld);
```

Diese Befehle werden am Besten in die Methode `init()` eingeführt. So werden die Buttons und Textfelder zu Beginn dem Applet hinzugefügt.

In Java kann man als ProgrammiererIn bestimmen, wo und wie Elemente des User Interfaces auf dem Bildschirm dargestellt werden. Dies wird dann allerdings ein wenig komplizierter, daher verzichten wir im Schnupperstudium darauf.

Wie bereits erwähnt, wird ein Ereignis ausgelöst, wenn der Benutzer/die Benutzerin mit der Maus auf einen Button klickt. Es gibt eine Methode, die von einem solchen Ereignis aufgerufen wird. In dieser Methode können wir den Programmcode einfügen, der beim Auftreten des Ereignisses ausgeführt werden soll. Die Methode heisst **action**.

Im folgenden wird der Rahmen der Methode erklärt. Du musst Deinen Programmcode nur an der bezeichneten Stelle einfügen.

```
public boolean action (Event evt, Object arg) {  
    if (evt.target == meinButton) { /* meinButton durch den Namen Deines Buttons  
        ersetzen */  
        /* hier kommt Dein Programmcode hinzu */  
        return true;  
    } else {  
        return false;  
    }  
}
```

### **Erklärung der Methode:**

Die Methode heisst `action` und gibt einen Wert vom Typ `boolean` zurück. Ein `boolean` kann die Werte `true` und `false` annehmen. In obigem Methodenrahmen wird ein `true` zurückgegeben, wenn es für das eingetretene Ereignis eine `if` Anweisung gab.

Die Methode `action` erhält zwei Parameter: ein `Event` und ein Objekt. Das Objekt wird unserer Methode zwar übergeben, wir benötigen es jedoch nicht. Ein **Event**, Ereignis, enthält verschiedene Informationen. Eine dieser Informationen beschreibt das Element des User Interfaces, auf das sich das Ereignis bezieht. Auf diese Information kann man über ein **.target** nach dem Variablennamen zugreifen.

Mit `(evt.target == meinButton)` kann man also überprüfen, ob sich das Ereignis auf das Element namens `meinButton` bezogen hat.

Wie wir oben gesehen haben, dient ein Textfeld dazu, dass BenutzerInnen Informationen an das Programm weitergeben können. Diese Information muss dann vom Applet gelesen und

verarbeitet werden. Eine Möglichkeit ist, die Information zu lesen, nachdem der Benutzer/die Benutzerin einen Button betätigt hat.

Jedes Textfeld hat eine Methode namens **getText()**. Diese ermöglicht das Lesen des in dem Textfeld stehenden Textes. Die Methode wird für das Textfeld aufgerufen und erhält keine Parameter.

Beispiel:

```
meinTextfeld.getText();
```

Diese Methode gibt den Wert des Textfeldes als Zeichenkette zurück. Wenn wir diese Zeichenkette als ganze Zahl lesen möchten, so können wir eine Methode des Datentyps Integer verwenden: **parseInt(...)**, welche eine Zeichenkette in eine ganze Zahl umwandelt.

Um also den Text in einem Textfeld einzulesen und in eine Integer Variable abzuspeichern, kann man folgende Anweisung verwenden:

```
int zahl = Integer.parseInt (meinTextfeld.getText());
```

Im Verlauf eines Programmablaufes kann es notwendig sein, ein Applet neu zu zeichnen. Dies erfolgt immer dann automatisch, wenn sich an der Fensteroberfläche etwas verändert hat, das das Applet betrifft. Man kann das erneute Zeichnen des Applets aber auch erzwingen. Dazu dient die Methode **repaint ()**.

Wird diese Methode aufgerufen, so wird die Methode paint des Applets ausgeführt.