Java-Merkblatt

AvHG, Inf, My

Grundlagen

Datentypen

`boolean`	Wahrheitswerte
`char`	Buchstaben
`int`	ganze Zahlen
`double`	Fließkommazahlen
`String`	Zeichenketten

Operatoren

`==`	gleich
`!=`	ungleich
`<=`	kleiner oder gleich
`>=`	größer oder gleich
`&&`	und
`||`	oder
`!`	nicht
`=`	Wert-Zuweisung
`%`	Modulo (Rest bei ganzzahliger Division)

Bedingungen

```
if (Bedingung) {
    Anweisungen;
} else {
    Anweisungen;
}
```

```
switch (Ausdruck) {
case Wert1:
    Anweisungen1;
    break;
case Wert2:
    Anweisungen2;
    break;
...
default:
    Anweisungen3;
}
```

Schleifen

```
while (Bedingung) {
    Anweisungen;
}
```

```
do {
    Anweisungen;
}while (Bedingung);
```

```
for (Vorab; Bedingung; Wert-Verändern) {
    Anweisungen;
}
```

Methoden

```
Sichtbarkeit Rückgabewert Name(Typ1 Parameter1, Typ2 Parameter2, ... ) {
    Anweisungen;
    return Wert;  // Rückgabe eines berechneten Wertes. Kann entfallen.
}
```

Kommentare

```
// Kommentar bis zum Zeilenende
/* Kommentar über
   mehrere Zeilen
*/
```

Methoden der Klasse `Graphics` (Auswahl)

`void` **`setFont`**`(Font font)`	Setzen der Schriftart
`void` **`setColor`**`(Color c)`	Setzen der Vordergrundfarbe
`void` **`drawString`** `(String str, int x, int y)`	schreibt im aktuellen Font einen Text (x,y = linke untere Ecke)
`void` **`drawLine`**`(int x1, int y1, int x2, int y2)`	malt eine Linie von (x1, y1) nach (x2, y2)
`void` **`drawRect`**`(int x, int y, int width, int height)` `void` **`fillRect`**`(int x, int y, int width, int height)`	malt ein Rechteck (x,y = linke obere Ecke) `fillRect()`: ausgefüllt, `drawRect()`: hohl
`void` **`drawOval`**`(int x, int y, int width, int height)` `void` **`fillOval`**`(int x, int y, int width, int height)`	malt im Bereich des angegebenen Rechtecks einen Kreis oder eine Ellipse (x,y = linke obere Ecke) `fillOval()`: ausgefüllt, `drawOval()`: hohl

Grundgerüst einer Java-Klasse

```
import java.awt.*;

public class Beispiel {
    private static int klassenVariable;
    protected int objektVariable;

    public Beispiel(int objektVariable) {
        this.objektVariable = objektVariable;
    }

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

Vererbung

```
public class SpezialBeispiel extends Beispiel {
    public SpezialBeispiel(int objektVariable) {
        super(objektVariable);
        ...
    }

    @Override
    public void methode() {
        super.methode();
        ...
    }
}
```

Hilfsbibliothek

```
import hilfe.*;
```

Klasse `HZeichnen`

Methoden:

```
static void drawDreieck (Graphics g, int x1, int y1, int x2, int y2, int x3, int y3)
static void fillDreieck (Graphics g, int x1, int y1, int x2, int y2, int x3, int y3)
```

Animationen erzeugen

```
import javax.swing.Timer;
```

Klasse `Timer`

Konstruktor: `public Timer(int millisec, ActionListener frame)`

Methoden: `void start()`
`void stop()`

Standard-Dialoge

```
import javax.swing.*;
```

Die **Klasse `JOptionPane`** besitzt die Methoden:

```
public static void showMessageDialog (Component parentComponent, Object message)
```

```
public static String showInputDialog (Object message)
```

Umwandlung von Strings in Zahlen

```
int zahl = Integer.parseInt(text);          //text: Variable vom Typ String
double zahl2 = Double.parseDouble(text);
```

Farben Mischen

```
Color hellrot = new Color(rot,gruen,blau);
```
// rot, gruen und blau sind jeweils Farbwerte zwischen 0 und 255

Bitmaps

Image-Objekt erzeugen

```
Image bild1, bild2;
bild1 = getToolkit().getImage(getClass().getResource("name1.gif"));
bild2 = getToolkit().getImage(getClass().getResource("name2.gif"));

MediaTracker mt = new MediaTracker(frame);  // frame ist das Anwendungsfenster
mt.addImage(bild1, 1);
mt.addImage(bild2, 1);
try {
    mt.waitForAll();
} catch (Exception e) {
    e.printStackTrace();
}
```

Malen eines Image-Objekts

Die Klasse **`Graphics`** besitzt die Methode:

```
public boolean drawImage (Image img, int x, int y, ImageObserver observer)
```

Als letzten Parameter übergibt man ein Objekt des Anwendungsfensters (`frame`).

Zufallszahlen

```
import java.util.*;
```

Man muss ein Objekt der Klasse **`Random`** erzeugen (der Konstruktor besitzt keine Parameter). Anschließend kann man mit folgender Methode der Klasse eine Zufallszahl erzeugen:

```
public int nextInt(int n)
```
// gibt eine Zahl zwischen 0 und (n-1) zurück

Arrays

Für einfache Datentypen (`int, char, boolean, double`):

```
int[] zahl;                                 // Variable anlegen
zahl = new int[200];                        // Feld erzeugen
for (int i=0; i < zahl.length(); i++) {     // Werte in das Feld schreiben
    zahl[i] = i+1;
}
```

Für Klassen-Typen:

```
Auto[] a;                                   // Variable anlegen
a = new Auto[4];                            // Feld erzeugen
for (int x = 0; x < a.length(); x++) {      // Objekte des Feldes erzeugen
    a[x] = new Auto(30+x*100, 20);
}
```

Tastatureingaben abfangen

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

public class Beispiel implements KeyListener {
    JFrame frame;

    public Beispiel(JFrame frame) {
        this.frame = frame;
        frame.addKeyListener (this);
    }

    public void keyPressed(KeyEvent e) {
    }

    public void keyReleased(KeyEvent e) {
    }

    public void keyTyped(KeyEvent e) {
        char c = e.getKeyChar();
        ...
        frame.repaint();     // Anwendungsfenster neu zeichnen
    }
}
```

Mauseingaben abfangen

Normale Mausereignisse

Man muss das Interface **`MouseListener`** implementieren:

Methoden des Interfaces `MouseListener`	**Wird aufgerufen, wenn...**
`public void` **`mouseClicked`** `(MouseEvent e)`	... eine Maustaste gedrückt und wieder losgelassen wurde.
`public void` **`mousePressed`** `(MouseEvent e)`	... eine Maustaste gedrückt wurde.
`public void` **`mouseReleased`** `(MouseEvent e)`	... eine Maustaste losgelassen wurde.
`public void` **`mouseEntered`** `(MouseEvent e)`	… der Mauszeiger das Fenster “betreten” hat.
`public void` **`mouseExited`** `(MouseEvent e)`	… der Mauszeiger das Fenster verlassen hat.

Eine Klasse, die das Interface `MouseListener` implementiert, muss beim Frame durch Aufruf der folgenden Methode registriert werden:

`public void` **`addMouseListener`**`(MouseListener l)`

Mausbewegungen abfangen

Für Mausbewegungen gibt es ein eigenes Interface **`MouseMotionListener`**:

Methoden des Interfaces `MouseMotionListener`	**Wird aufgerufen, wenn...**
`public void` **`mouseDragged`** `(MouseEvent e)`	… die Maus bei gedrückter Taste bewegt wurde.
`public void` **`mouseMoved`** `(MouseEvent e)`	… die Maus ohne gedrückte Taste bewegt wurde.

Eine Klasse, die das Interface `MouseMotionListener` implementiert, muss beim Frame durch Aufruf der folgenden Methode registriert werden:

`public void` **`addMouseMotionListener`**`(MouseMotionListener l)`

Details über das Ereignis erfahren

Bei allen Methoden erhält man ein Objekt der Klasse **`MouseEvent`**, in dem Details über das Mausereignis gespeichert sind. Die Klasse `MouseEvent` besitzt folgende Methoden:

Methoden der Klasse `MouseEvent` (Auswahl)	**Beschreibung**
`public int` **`getButton`**`()`	Gibt an, welcher Knopf gedrückt wurde: `MouseEvent.BUTTON1`, `MouseEvent.BUTTON2` oder `MouseEvent.BUTTON3`.
`public int` **`getClickCount`**`()`	Anzahl der Mausklicks
`public int` **`getX`**`()`	X-Position der Maus
`public int` **`getY`**`()`	Y-Position der Maus