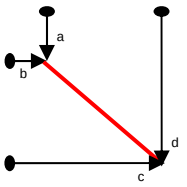
Prozeduale Programmierung Zusammenfassung

Java.awt (Zeichnen)

Grundbefehele

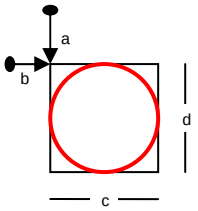
g.drawLine(a,b,c,d);
(Linie)

a) Horizontaler Wert des Linienanfangs
b) Vertikaler Wert des Linienanfangs
c) Horizontaler Wert des Linienendes
d) Vertikaler Wert des Linienendes



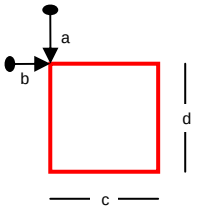
g.drawRect(a,b,c,d);
(Rechteck)

a) Horizontaler Wert der oberen rechten Ecke
b) Vertikaler Wert der oberen rechten Ecke
c) Breite des Rechtecks
d) Höhe des Rechtecks



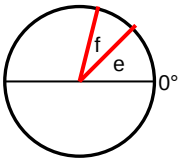
g.drawOval(a,b,c,d);
(Ellipsen / Kreise) (werden immer in (unsichtbaren) Rechtecken gezeichnet)

a) Horizontaler Wert der oberen linken Ecke des Rechtecks
b) Vertikaler Wert der oberen linken Ecke des Rechtecks
c) Breite des Rechtecks
d) Höhe des Rechtecks



g.drawArc(a,b,c,d,e,f);
Ein Kreissegment zeichnen.

a) Horizontaler Wert der oberen linken Ecke des Rechtecks
b) Vertikaler Wert der oberen linken Ecke des Rechtecks
c) Breite des Rechtecks
d) Höhe des Rechtecks
e) Anfangswinkel
f) Winkel des Bogens



Farben setzen:

g.setBackground(Color.lightGray);
g.setColor(Color.red);
Verfügbare Farben:
black, gray, orange, yellow, blue,green, pink, cyan, lightGray, red, darkGray, magenta, white

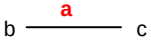
Formen ausfüllen

fillRect
fillArc
fillOval

Zeichenketten

g.drawString(“a“,b,c)

a) Text in Anführungszeichen
b) Horizontale Position des Anfangs der Zeichenkette
c) Vertikale Position des Endes der Zeichenkette



Nice 2 Know

Kommentar	\\
Verknüpfunge	+
Einschliessen von Parametern	( )
Kennzeichnen von Ketten	„…“

Variablen

Deklaration von Variablen

- Variablen müssen mit einem Buchstaben beginnen.
- Können danach aus einer beliebigen Anzahl Ziffern und Buchstaben bestehen.
- Können _ und $ enthalten
- Keine Leerzeichen. Keine Umlaute (funktioniert zwar, doch gibt häufiger Fehler)
- Zur besseren Lesbarkeit, jedes neue Wort mit einem Grossbuchstaben beginnen. hoeheDerBox z.B.
- Keine reservierten Namen verwenden wie void, extends, int

Deklaration
int x;

Deklaration mit Wertzuweisung
Iit x = 100;

Wertzuweisung (Variable muss irgendwo früher Deklariert worden sein)
Wertzuweisung erfolgt von rechts nach links.

x = 100;

a = b; a hat nun den Wert b
a = c: vor der neuen Zuweisung hat a noch den Wert b, dieser wird nun mit Wert c überschrieben.

Variablentypen

boolean	true / false	1 Bit	
char	Text Zeichen	16 Bit	
byte		-128 bis + 127	
short		-32’768 bis + 32’676	
integer	Ganzzahl von ca. -2 Mio bis +2 Mio.	-2'147'483'648	int
	Nimmt immer Ganzzahl. Beim rechnen aufpassen!	bis	
		+2'147'483'647	5
long		64 bit	
float	Gleitkommazahl	-3.4 x10^{38}	float
		bis	
		+ 3.4 x10^{38}	5.5f
double	Gleitkommazahl ohne genauere Definition		double
	Double wird angenommen wenn kein f hinter der Zahl steht.		5.5

Konstante

Werden immer in Grossbuchstaben geschrieben. (damit sie klar erkennbar sind)
Können nicht mehr geändert werden im Programm

```
final Typ NAME = Wert;

final double GRAVITATION = 9.81;
```

final

Operatoren

Punkt vor Strich Regel gilt auch in Java. Zur Sicherheit besser Klammern setzen.

Der Ausführungsreihenfolge nach:

Multiplikation	Etwas multiplizieren	a * b	*
Division	Etwas teilen	a / b	/
Modulo	Den Rest einer Division ausgeben	a % b	%
		5 % 2 = 0.5	
Addition	Etwas addieren	a + b	+
Subtraktion	Etwas subtrahieren	a – b	-

Spezielle Operatioren

Addition +1	1 dazuzählen	a + 1	++
Subtraktion – 1	1 wegzählen	a - 1	--

Variable anzeigen

g.drawString(“Ergebnis ist “ + 1 + 2, 100, 100);	Ergebnis ist 12
g.drawString(“Ergebnis ist “ + (1 + 2), 100, 100);	Ergebnis ist 3

Vergleichsoperatoren		
grösser als	a > b	>
kleiner als	a < b	<
gleich	a == b	==
ungleich	a != b	!=
kleiner gleich	a <= b	<=
grösser gleich	a >= b	>=
Logische Operatoren		
AND	alter > 6 && < 16 -> Kindertarif	&&
OR		||
NOT		!

Variablentyp umwandeln (casten)

- int kann immer in float umgewandelt werden. (Es können keine Informationen verloren gehen)
- float kann in int konvertiert werde4n. Doch es geht Information verloren (die Kommastellen von float)

```
int x = 5;
x = (float) x/2;
// Ergebnis wird 2.5 sein. Da x in einen floattyp umgewandelt wurde.
```

„cast“

Lokale Variablen

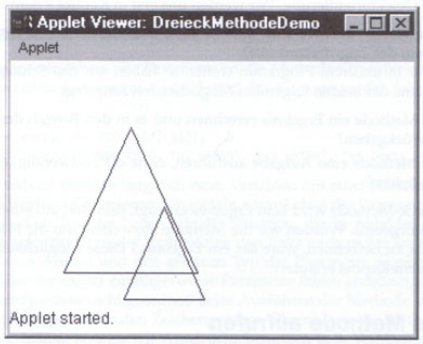
- Werden in die aktuelle Methode reinkopiert..
- Es wird nur innerhalb der Methode damit gearbeitet.
- Wird beim beenden der Methode zerstört.

Methode

Aufbau einer Methode

Abbildung 5.1 Programm mit Methode zeichneDreieck, die zweimal aufgerufen wird

```
import java.awt.*;
import java.applet.Applet;
public class DreieckMethodeDemo extends Applet {
    public void paint (Graphics g) {
3       zeichneDreieck(g, 40, 160, 100, 110);
        zeichneDreieck(g, 85, 180, 60, 70);
    }                          // paint
1   private void zeichneDreieck (Graphics g, int untenX,
                                 int untenY, int basis,
                                 int hoehe) {
2       g.drawLine(untenX, untenY, untenX + basis, untenY);
        g.drawLine(untenX + basis, untenY, untenX + basis/2, untenY – hoehe);
        g.drawLine(untenX + basis/2, untenY – hoehe, untenX, untenY);
    } // zeichneDreieck
} // class DreieckMethodeDemo
```



1 **Methodenkopf**
- Gibt der Methode den Namen
- Zeigt die angegebenen Parameter an

`Private void beliebigerName (parameter liste)`

2 **Methodenkörper**
- Steht immer zwischen den geschweiften Klammern { }

3 **Methodenaufruf**
- Methode wird über ihren Namen aufgerufen

Aufruf der Methode: `nameMeinerMethode( );`

Allgemeine Form der Methode

```
Private void beliebigerName (parameter liste)
    {
        Methodenkörper
    }
```

Optionen

```
private void zeichneDreieck (Graphics g, int untenX,
```

public und private
Access control modifiers – Modifikation der Zugriffskontrolle

public
- Kann von überall her aufgerufen werden

private
- Kann nur aus dem Aktuellen Applet/Programm aufgerufen werden.

Void
Gibt keinen „globalen“ Rückgabewert. Der Rückgabewert ist nur innerhalb der Methode gültig.

Parameter

```
zeichneDreieck(  g, 40, 160, 100, 100  );
```

- Zuweisung der Parameter findet von links nach rechts statt.
- Anzahl und Typ der Parameter muss genau stimmen
- Parameter werden der Reihe nach zugewiesen. (Sie werden nicht geordnet) Werden durch Kommas voneinander getrennt.

Formale Parameter
- Die Liste der Namen und Typen, über die derjenige entscheidet, der die Methode schreibt.
- Wenn ähnliche Namen in anderen Methoden benutzt werden, treten keine Probleme auf, da jede Methode ihre eigene Kopie an Parametern hat.

Bezeichner, der in einer Methode verwendet wird, um einen Wert aufzunehmen, der an die Methode vom Aufrufer übergeben wird.
Zum Beispiel ist betrag ein formaler Parameter von verarbeiteEinzahlung(int betrag)

Aktuelle Parameter
- Parameter die beim Aufruf zur Verfügung stehen müssen.

der tatsächliche Wert, der an die Methode durch den Aufrufer übergeben wird.
Zum Beispiel werden die 200, die beim Aufruf an die Methode verarbeite

Return und Ergebnisse

```
import java.awt.*;
import java.applet.Applet;
public class ReturnDemo extends Applet {
    public void paint(Graphics g) {
        int antwort = flaecheRechteck(30, 40);
        g.drawString("Fläche des Rechtecks ist " + antwort, 100, 100);
    }
    private int flaecheRechteck(int seite1, int seite2) {
        int flaeche = seite1 * seite2;
        return flaeche;
    }
}
```

1
2

1 Anstelle von void wurde int verwendet, d.h. dass die Methode einen Integer Rückgabewert erwartet
2 Um den Wert zurück zu erhalten.
`Return Ausdruck;`

Return zusammen mit void verwenden

```
Private void demo()
{
   //mach etwas
   return;
   //mach etwas anderes
}
```

Hier wurde `return;` verwendet um `//mach etwas anderes` nicht aufzurufen, nicht auszuführen. Um einen vorzeitigen Abbruch des Programms zu erzwingen.

Ereignis Kapitel noch nicht komplett. Weiter im Buch S. 91 bis 105

- Benutzer drückt auf eine Taste oder klickt mit der Maus
- Benutzer klickt eine Schaltfläche, eine Laufleiste, eine Menüoption usw. an…

Entscheidungen

if

```
If (Bedingung)
{
   auszuführende Anweisung(en), wenn Bedingung wahr ist;
}
```

```
if ( )
  then;
```

ACHTUNG! Kein Semikolon am Ende der If Zeile!

if else

```
If (Bedingung)
{
   auszuführende Anweisung(en), wenn Bedingung wahr ist;
}
else
{
   auszuführende Anweisung(en), wenn Bedingung falsch ist;
}
```

```
if ( )
{
  then;
}
  else;
  { }
```

ACHTUNG! Kein Semikolon am Ende der if und else Zeilen!

Wiederholung

while

```
Variable  zuweisen
while (Bedingung wahr)
{
   auszuführende Anweisung(en);
Variable++;
}
```

```
while ()
{ }
```

ACHTUNG! Kein Semikolon am Ende der while Zeile!

Beispiel

```
int zaehler, x = 10
zaehler = 0
while (zaehler < 8)
{
   g.drawString("*", x, 20);
   x = x+10;
   zaehler++;
}
```

Ausgabe:
******

1. Stern wird gezeichnet,
Zähler wird um 1 erhöht.
Schlaufe wird so lange wiederholt bis die Bedingung nicht mehr stimmt

for

```
for (Anfangsanweiseung; Testbedingung; Anweisung)
{
   auszuführende Anweisung(en);
}
```

```
for ( ; ; s)
{ }
```

ACHTUNG! Kein Semikolon am Ende der for Zeile!

Anfangsanweisung	Was ist zu machen, bevor die Schleife startet
Testbedingung	Wird vor dem Ausführen der Schleife geprüft Wenn wahr, wird die Schleife ausgeführt, wenn falsch, wird abgebrochen.
Anweisung	Wird direkt vor dem Ende jeder Wiederholung ausgeführt

do …while

die Schleife wird mindestens einmal ausgeführt, da die Überprüfung erst am Ende ist

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

```
Do
{ }
while ( );
```

switch

```
switch
{
  case Bedingung1:
    Anweisung
    break;
  case Bedingung2:
    Anweisung
    break;
  case Bedingungx:
    Anweisung
    break;
  default
    Anweisung
}
```

```
switch( )
{
case 1:
break;
case x:
break;
default;
}
```

Bedingung 1 bis x	Nach dem was ich filtern will.
default	Alles was zu keiner Bedingung passt, wird mit den default Angaben behandelt

Beispiele

```
Switch (obst)
{
      case „birne“:
            g.drawString( “Es ist eine Birne! “, 100, 100);
            break;
      case „apfel“:
            g.drawString( “Es ist ein Apfel! “, 100, 100);
            break;
      default:
            g.drawString( “es ist irgendetwas! “, 100, 100);
}
```

```
Switch (geradeZahlen)
{
      case 2
            g.drawString( “Es ist eine 2! “, 100, 100);
            break;
      case 4:
            g.drawString( “Es ist eine 4! “, 100, 100);
            break;
      default:
            g.drawString( “es ist irgendetwas! “, 100, 100);
}
```