

Oft will man ein Statement nur dann ausführen, wenn eine bestimmte Bedingung gilt. Das geht mit dem if Statement:

```
if (zinsSatz > 100.0) {  
    System.out.println("Fehler.");  
} else  
    rate = restschuld * zinsSatz/100.0 / 12.0 + tilgung;
```

Bemerkung: Ausgaben an System.out sind dilettantisch.

Bedingungen

...stehen immer in Klammern und sind von der Form $x_1 \text{ op } x_2$ wobei

- op ist <, >, <=, >=, ==
- Die Operanden sind Zahlen des gleichen Typs, evtl. wird implizite Typkonversion vorgenommen.
- Später lernen wir noch andere Bedingungen kennen.

Vorsicht beim Testen von Doubles auf Gleichheit. Besser

```
double final EPSILON = 1E-10; // zum Beispiel  
if ( Math.abs(x - y) <= EPSILON )
```

```
if (zinsSatz > 100.0) {  
    Fehlerbearbeitung.fehler(falscherZinsSatz);  
} else  
    rate = restschuld * zinsSatz/100.0 / 12.0 + tilgung;
```

- Fehlerbearbeitung.fehler ist eine **benutzerdefinierte** Methode, die **Fehlerobjekte** anzeigt und entsprechend verfährt.
- Ein (benutzerdefiniertes) **Fehlerobjekt** (hier falscherZinsSatz) beinhaltet den Anzeigetext (üblicherweise in verschiedenen Sprachen) und Instruktionen wie bei seinem Auftreten zu verfahren ist.

Die Klammern

Formal steht nach der Bedingung **ein** Statement.

Braucht man mehrere, so fasst man sie mit geschweiften Klammern ({}) zu einem **Block** zusammen.

Das ganze if - else Konstrukt wird als ein einziges Statement aufgefasst, ein **zusammengesetztes Statement**.

Man kann den else Teil auch weglassen.

Was ist hier falsch?

```
if (betrag <= kontostand)
    double neuerKontostand = kontostand - betrag;
    kontostand = neuerKontostand;
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 66

Meherere if-Statements

Natürlich können in den Zweigen eines if-Statements wiederum solche stehen:

```
if ( richter >= 8.0 )
    s = "Grosse Verwuestung"
else if ( richter >= 7.0 )
    s = "Viele Gebaeude zerstoert"
else if ( richter >= 6.0 )
    s = "Viele Gebaeude beschaedigt"
else if ( richter >= 4.5 )
    ...
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 68

Antwort

Die geschweiften Klammern fehlen.

Einrückungen dienen der Lesbarkeit werden aber vom Compiler ignoriert.

Richtig:

```
if (betrag <= kontostand) {
    double neuerKontostand = kontostand - betrag;
    kontostand = neuerKontostand;
}
```

Oder so:

```
if (betrag <= kontostand)
{
    double neuerKontostand = kontostand - betrag;
    kontostand = neuerKontostand;
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 67

Dangling else

Ein **else** bezieht sich immer auf das nächstgelegene **if**.

Ungeachtet der Einrückung.

Wir wollen aus **int a, b, c** as größte und das kleinste berechnen.

Einführung in die Informatik: Programmierung und Softwareentwicklung 69

Maximum und Minimum

```
if (a >= b)
    if (a >= c) {
        max = a;
        if (c >= b)
            min = b;
        else
            min = c;
    } else {
        max = c; min = b;
    }
else
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 70

Einführung in Klassen und Objekte

Wir wollen Bankkontos verwalten.

Ein Bankkonto enthält einen Kontostand.

(außerdem noch Name, Überziehungslimit, etc. pp.)

Der Kontostand kann

- abgerufen werden
- erhöht werden durch Einzahlung
- erniedrigt werden durch Abhebung

Einführung in die Informatik: Programmierung und Softwareentwicklung 72

Maximum und Minimum

```
if (b >= c) {
    max = b;
    if (c >= a)
        min = b;
    else
        min = c;
} else {
    max = c; min = a;
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 71

Verwendung von Bankkonten

Wir möchten eine Klasse von Bankkonten, die man etwa so verwenden kann:

```
public class BankkontoTest {
    public static void main(String[] args) {
        Bankkonto matthiasGiro = new Bankkonto();
        Bankkonto johannasSpar = new Bankkonto();
        double zinsSatz = 1.25;

        johannasSpar.einzahlen(2000.00);
        matthiasGiro.einzahlen(30000.00);
        matthiasGiro.abheben(10000.00);
        johannasSpar.einzahlen(
            johannasSpar.getKontostand() * zinsSatz / 100.0);
    }
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 73

```
System.out.println("Johannas Sparkonto: " +
                    johannasSpar.getKontostand());
System.out.println("Matthias Girokonto: " +
                    matthiasGiro.getKontostand());
    }
}
```

Ausgabe wäre hier:

```
Johannas Sparkonto: 2025.0
Matthias Girokonto: 20000.0
```

Wie kann man so eine Klasse schreiben?

Deklaration der Daten

Die Daten eines Objekts bestehen aus Variablen, den **Instanzvariablen**, die in der Klasse deklariert werden.

Beim Bankkonto brauchen wir nur eine Instanzvariable vom Typ `double`.

Wir schreiben also unter *Deklaration der Daten*:

```
private double kontostand;
```

Damit wird gesagt, dass jedes Objekt der Klasse **Bankkonto** sich einen Double-Wert “merken” kann.

Die Qualifikation **private** besagt, dass dieser Wert von außen nicht direkt einsehbar ist. Nur auf dem Umweg über Methodenaufrufe.

Kommt in eine Datei `Bankkonto.java`.

```
public class Bankkonto {
    Deklaration der Daten           Die Einträge
    Definition der Methoden
    Definition des Konstruktors }
können in beliebiger Reihenfolge stehen.
```

Instanzvariablen als public

Man kann Instanzvariablen auch **public** deklarieren. Dann kann man ihren Wert von außen abrufen. Etwa

```
Rectangle r = Rectangle(10,10,30,50);
r.height;
```

Es ist im allgemeinen schlecht, Instanzvariablen **public** zu deklarieren.

Grund: interne Repräsentation der Daten kann dann nicht mehr verändert werden.

Beispiel: Kontostand als Integer (in Cents).

Ein `Rectangle` besteht aus den Koordinaten der linken oberen Ecke, sowie Breite und Höhe, jeweils als `int`.

Wie sehen die Instanzvariablen der Klasse `Rectangle` aus?

Methodendefinitionen

Die Methode `einzahlen` definieren wir durch

```
public void einzahlen(double betrag) {  
    kontostand = kontostand + betrag;  
}
```

Das bedeutet im einzelnen:

- Die Methode kann von außen aufgerufen werden (`public`)
- Der Aufruf erfolgt mit einem Parameter vom Typ `double`
- Der Aufruf liefert keinen Wert zurück (`void`)
- Beim Aufruf werden die Statements zwischen den `{ }` ausgeführt. Hier

```
    kontostand = kontostand + betrag;
```
- Dabei kann sowohl auf die Instanzvariablen, wie auch auf die Parameter zugegriffen werden.

```
public int x;  
public int y;  
public int width;  
public int height;
```

Wie gesagt, `public` sollte hier vermieden werden.

Methodendefinitionen

Die Methode `abheben` definiert man durch

```
public void abheben(double betrag) {  
    kontostand = kontostand - betrag;  
}
```

Rectangle

Ein Objekt der Klasse `Rectangle` kann man so verschieben:

```
r.translate(34,12)
```

Wie sieht die **Deklaration** der Methode `translate` aus?

Einführung in die Informatik: Programmierung und Softwareentwicklung 82

Antwort

```
public void translate(int dx, int dy) {  
    x = x + dx;  
    y = y + dy;  
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 84

Antwort

```
public void translate(int dx, int dy) {  
    ...  
}
```

Und wie sieht der **Methodenrumpf** (method body) aus?

Einführung in die Informatik: Programmierung und Softwareentwicklung 83

Methodendefinitionen

Die Methode `getKontostand` liefert einen Double-Wert zurück.

```
public double getKontostand() {  
    return kontostand;  
}
```

Die Deklaration `public double getKontostand()` besagt, dass die Methode keine Parameter erwartet, aber einen `double` zurückliefert.

Das `return` Statement legt den zurückgegebenen Wert fest.

Wie würden wir die Methode `union` der Klasse `Rectangle` deklarieren, die das kleinste achsenparallele Rechteck ausgibt, welches das gegebene Rechteck und ein weiteres umfasst.

Einführung in die Informatik: Programmierung und Softwareentwicklung 85

Antwort

```
public Rectangle union(Rectangle r){  
    ...  
}
```

und wie würden wir sie implementieren?

Einführung in die Informatik: Programmierung und Softwareentwicklung 86

Antwort

```
    if (y + height >= r.y + r.height)  
        resheight = y + height - resy;  
    else  
        resheight = r.y + r.height - resy;  
    return new Rectangle(resx, resy, reswidth, resheight);  
}
```

Beachte hier:

- Im Methodenrumpf kann man jederzeit neue Variablen deklarieren.
- Bei Objekterzeugung das `new` nicht vergessen.

Einführung in die Informatik: Programmierung und Softwareentwicklung 88

Antwort

```
Rectangle union(Rectangle r) {  
    int resx, resy, reswidth, resheight;  
  
    if (x <= r.x)  
        resx = x;  
    else  
        resx = r.x;  
    if (y <= r.y)  
        resy = y;  
    else  
        resy = r.y;  
    if (x + width >= r.x + r.width)  
        reswidth = x + width - resx;  
    else  
        reswidth = r.x + r.width - resx;  
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 87

Konstruktoren

Ein neues Bankkonto erzeugt man mit

```
new Bankkonto
```

Die Instanzvariable `kontostand` ist dann **automatisch** mit Null vorbesetzt.

Besser ist es, das Verhalten von “new Bankkonto” selber zu definieren.

Dazu schreibt man in die Klasse ein oder mehrere

Konstruktordefinitionen.

Zum Beispiel:

```
Bankkonto() {  
    kontostand = 0.0;  
}
```

Einführung in die Informatik: Programmierung und Softwareentwicklung 89

Konstruktoren

Man kann auch noch zusätzlich schreiben:

```
Bankkonto(double betrag) {  
    kontostand = betrag;  
}
```

Schreibt man nun `b = new Bankkonto()` so wird eins mit Kontostand 0 erzeugt.

Schreibt man dagegen `b = new Bankkonto(134.0)` so wird eins mit Kontostand 134.0 erzeugt.

Welcher Konstruktor zur Ausführung kommt, richtet sich nach Anzahl und Typen der Parameter.

Antwort

```
Rectangle(Point p) {  
    x = p.x;  
    y = p.y;  
    width = 0;  
    height = 0;  
}
```

Rectangles

Ein `Rectangle` kann man auch so erzeugen:

```
Point p;  
Rectangle b = new Rectangle(p);
```

gemeint ist das Rechteck mit linker oberer Ecke `p` und Breite, Höhe beide Null.

Wie ist das in der Klasse `Rectangle` definiert?

Grafikfenster

Für das Practical stellen wir Ihnen ein Grafikfenster zur Verfügung.

Es versteht die folgenden Methoden:

```
public void drawLine(Point x, Point y)  
public void switchToBackgroundColour()  
public void switchToForegroundColour()  
public Point mouseClicked()  
public void setText(String text)  
public void getText(String text)
```

Die Methode `mouseClick` wartet auf einen Mausklick und gibt dann dessen Position als `Point` zurück.

`switchToBackgroundColour()` schaltet den "Zeichenstift" auf die Hintergrundfarbe um. Dient zum Löschen.

`setText` schreibt Text in das Grafikfenster.

`Points` haben integer `x` und `y` Koordinaten als `public`

Instanzvariablen. Näheres siehe Java-Referenz.

Wir wollen eine Klasse **Dreieck** implementieren, die durch ihre Eckpunkte (in integer) repräsentierte Dreiecke enthält.

Ein Dreieck soll folgendes können:

- Seine Eckpunkte liefern (durch Methodenaufruf)
- Sich verschieben lassen
- Sich zeichnen lassen (in einem gegebenen Grafikfenster)
- Sich löschen lassen (in einem gegebenen Grafikfenster)