http://schuelerlabor.informatik.rwth-aachen.de

Leitprogramm Informatik:

Einführung in die Java-Programmierung

```java
import java.util.Random;

public class Zufall{

  public static void main(String[] args){
    /*Random rando = new Random(); //erstelle Objekt der importierten klasse Random

    System.out.println(rando.nextInt()); //R⍰ckgabewert test*/
  }

  /*
  Die Funktion getInt erzeugt eine zuf⍰llige ganze Zahl, zwischen -99 und 99
  und gibt diese zur⍰ck.
  */
  public static int getInt(){
    int rueck; //Initialisierung des R⍰ckgabewertes
    Random Zahl = new Random(); //Generierung eines Zuf⍰lligen Objektes
    Random vor = new Random(); //Generierung eines weiteren zuf⍰lligen Objektes
    rueck = Zahl.nextInt(99); //cast Zuf⍰lliges Objekt auf ganze Zahl zwischen 0 und 9

    //Vorzeichen Bestimmung und R⍰ckgabe
    if(vor.nextInt(2)==1){return rueck;}
    else{rueck = rueck * (-1); return rueck;}
  }
}
```

Verfasser: C. Roeckerath, J. Nowak

Kurz-Info:
Informatischer Inhalt: Einführung in die Programmierung in Java
Jahrgangsstufe: 8 bis EF
Vorwissen: Von Neumann-Rechner, Prozessor, Maschinensprache

CC BY SA

KURZINFORMATION FÜR DIE LEHRKRAFT

Titel: Einführung in die Programmierung in Java

Schulstufe: gymnasiale Mittel- und Oberstufe (ab 8. Klasse)

optimale Jahrgangsstufe: EF

Themenbereich: Einstieg in die Java-Programmierung; Struktur, Schleifen

EINORDNUNG IN GESETZLICHE RAHMENBEDINGUNGEN

Lehrplan NRW:

Kompetenzbereiche: *Modellieren*: Modifizierung, Erweiterung und Konstruktion informatischer Modelle zu kontextbezogenen Problemstellungen; *Implementieren*: Implementierung, Modifizierung, Test und Korrektur von Computerprogrammen im *JavaEditor*

Inhaltsfelder: *Daten und ihre Strukturierung*: Objekte und Klassen: Ermittlung von Objekten, ihrer Eigenschaften, ihrer Operationen und Beziehungen bei der Analyse einfacher Problemstellungen; *Algorithmen*: Analyse, Entwurf und Implementierung einfacher Algorithmen: Analyse, Modifizierung, Entwurf und Implementierung einfacher Algorithmen und ihre Darstellung in Umgangssprache; *Formale Sprachen und Algorithmen*: Implementierung einfacher Algorithmen im *JavaEditor* unter Berücksichtigung der Syntax und Semantik der Programmiersprache Java, Interpretation von Fehlermeldungen und Korrektur des Quellcodes; *Informatiksysteme*: Internet: Nutzung des Internets zu Recherchezwecken; *Informatik, Mensch und Gesellschaft*: Einsatz von Informatiksystemen: selbstständige, sichere und zielführende Nutzung von Informatiksystemen

Einbindung in den Unterricht: z. B. zu Beginn einer Unterrichtsreihe, die sich mit höheren Programmiersprachen oder auch ganz konkret Java befasst.

Vorkenntnisse: Von Neumann-Rechner, Prozessor, Maschinensprache

Inhaltsbeschreibung: In diesem Leitprogramm können Schülerinnen und Schüler den Umgang mit der Programmiersprache Java erlernen. Es wird auf die Grammatik und Semantik der Sprache Java eingegangen, und außerdem werden erste Richtlinien für sauberen Code angesprochen. Da es sich um ein Selbstlernangebot handelt, erarbeiten die Lernenden das neue (Programmier-)Wissen weitestgehend selbstständig. Lediglich im Rahmen der Kapitelabschlusstests ist es ratsam, dass die Schülerinnen und Schüler sich mit einem Betreuenden oder einer Lehrkraft austauschen. Disziplinierte Schülerinnen und Schüler können ihre Lösungen jedoch auch selbstständig mit Hilfe von Musterlösungen überprüfen, sodass theoretisch auch eine vollkommen selbstständige Bearbeitung möglich ist. Die Kapiteltests dienen der Überprüfung, ob die Inhalte des jeweiligen Kapitels, die die Schülerinnen und Schüler immer anhand von mehreren Aufgaben erarbeiten, auch wirklich verstanden wurden. Ist dies nicht der Fall, können die Jugendlichen einzelne Kapitelinhalte noch einmal gezielt wiederholen.

EINFÜHRUNG

In den nächsten Wochen, in denen du dieses Leitprogramm durcharbeitest, verfolgst du das Ziel, die ersten Schritte des Programmierens (in Java) zu erlernen. Programmieren bedeutet, dass man einen Text in einer speziellen Sprache verfasst, mit deren Hilfe man dem Computer Anweisungen erteilen kann, die er ausführen muss. Diese spezielle Sprache nennt man Programmiersprache.

Java ist eine moderne Programmiersprache, die in der Arbeitswelt, im Internet und an Universitäten viel genutzt wird. Mit Java kann man nicht nur kleine Programme schreiben, sondern auch komplexe Software konstruieren. Nutzt du den Computer regelmäßig? Wenn das der Fall ist, dann hast du sicherlich auch schon oft, ohne es zu merken, Software genutzt, die in Java geschrieben wurde.

Dieses Leitprogramm wird dir nicht nur zu den ersten Schritten des Programmierens in Java verhelfen. Nach dem Durcharbeiten wirst du auch dazu in der Lage sein, eine Vielzahl an Programmiersprachen zu verstehen und mit ein bisschen Übung auch selbst zu nutzen.

Arbeite dieses Leitprogramm sorgfältig durch! Für das Programmieren ist es beispielsweise wichtig, dass du immer erst weiterarbeitest, wenn dein Programm auch wirklich funktioniert. Es ist ganz normal, dass nicht jedes Programm auf Anhieb funktioniert. Wenn dies der Fall ist, musst du dich unter Umständen längere Zeit auf Fehlersuche begeben. Auch das gehört zu den Aufgaben eines Informatikers oder einer Informatikerin. Lass dich also nicht unterkriegen, wenn nicht alles von Beginn an so funktioniert, wie du es dir vorgestellt hast. Bleib dran! Es lohnt sich. Du schaffst das.

Viel Erfolg!

INHALT

Kurzinformation für die Lehrkraft 2
Einführung 3
Arbeitsanleitung 5
Kapitel 1: Erste Schritte 6
Kapitel 2: Programmaufbau 13
Kapitel 3: Datentypen und Operationen 23
Kapitel 4: Variablen 35
Kapitel 5: Verzweigungen 48
Kapitel 6: Schleifen 61
Kapitel 7: Arrays 77
Anhang A: Kapiteltests 105
Anhang B: Mediothek 135
Anhang C: Literaturangaben 135
AnHang D: Abbildungsverzeichnis 135

ARBEITSANLEITUNG

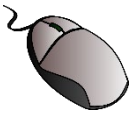
Das vorliegende Leitprogramm ist für das selbständige Durcharbeiten gedacht. Die Kapitel bestehen aus unterschiedlichen Teilen, die du an den folgenden Symbolen erkennen kannst:

Lernziel

Was wirst du nach dem Bearbeiten des Kapitels (mehr) können?

Theorie

Lies dir diesen Teil sorgfältig durch. Hier lernst du alles, was du für die praktischen Aufgaben und für die Tests brauchst.

Übungsaufgaben am Computer 

Führe diese Aufgaben am Computer aus. Arbeite erst weiter, wenn dein Programm läuft. Du darfst diese Aufgaben alleine oder mit einem Partner bearbeiten.

Übungsaufgaben im Heft

Bearbeite die Übungsaufgaben, und vergleiche sie anschließend mit den Musterlösungen. Soweit nicht anders angegeben, bearbeitest du diese Aufgaben alleine.

Sicherungsphase

Hier soll das Gelernte gesichert werden. Dieser Teil hilft dir beim Einprägen der gelernten Themen!

Lernkontrolle

Jetzt kannst du prüfen, ob du die Inhalte des letzten Kapitels verstanden hast. Entscheide selbst, wann du dich fit fühlst für diesen Test. Nur wenn du diesen Test erfolgreich meisterst, darfst du das nächste Kapitel oder das Additum in Angriff nehmen.

Additum

Diesen Teil bearbeiten nur die schnellen Schüler und Schülerinnen. Nach der Lernkontrolle für das Kapitel erfährst du von der Lehrkraft, ob du dazugehörst. Diese Aufgaben sind extra anspruchsvoller. Sie können auch unbekannten Stoff behandeln.

KAPITEL 1: ERSTE SCHRITTE

ÜBERSICHT

In diesem Kapitel wird erklärt, wie ein Programm entsteht. Du wirst lernen, wie man aus einem Text ein Programm machen kann.

Was du an dieser Stelle noch nicht lernen wirst, das ist, wie man ein Programm selbst schreibt. Zunächst geht es nur darum, aus einem vorgegebenen Text ein Programm in Maschinensprache zu machen, das man auf dem Computer laufen lassen kann.

LERNZIEL

Nachdem du dieses Kapitel durchgearbeitet hast, kannst du einen vorgegebenen (Programm-)Text in ein Maschinenprogramm umwandeln und dieses Programm am Computer starten.

AUFGABEN

Aufgabe1

Führe die nächsten Arbeitsaufträge Schritt für Schritt aus! Wenn du unsicher bist, was mit einigen der Anweisungen gemeint sein könnte, dann helfen dir die Abbildung 1: Compiler Zeile und Abbildung 2: Der Editor vielleicht weiter.

a) Suche dir einen freien Rechner!
b) Logge dich mit deinem Benutzernamen und Passwort ein!
c) Starte das Programm JavaEditor! Jetzt siehst du das in Abbildung 1 dargestellte Fenster.
d) Öffne ein neues Dokument! (Datei → Neu)
e) Gib folgenden Javaprogrammcode ein! Achte darauf, dass du kein Zeichen vergisst!

```
a. public class Mein_erstes_Programm {


      i.  public static void main (String [] arguments){
             1. System.out.print("Hallo Welt");
     ii.  }


    iii.  }
```

f) Speichere das Programm unter diesem Namen: **Mein_erstes_Programm.java**
g) Klicke auf das Symbol ! Jetzt hast du das Programm vom Computer kompilieren lassen. Was das bedeutet, wird dir später im Theorieteil erklärt.

Wenn du bis hierhin alles richtig gemacht hast, dann steht dies im unteren Feld:

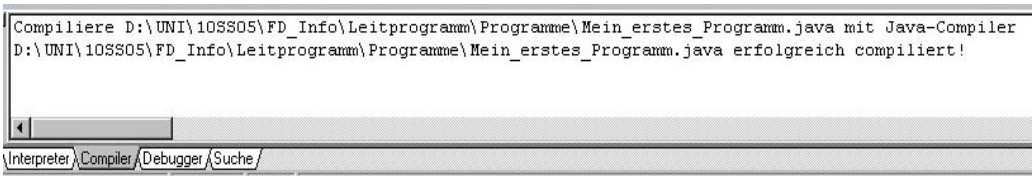


Abbildung 1: Compiler-Zeile

Wenn dies nicht der Fall ist, dann taucht in dem Feld der Begriff „error“ (also Fehler) auf. Das bedeutet, dass du irgendeinen Teil des Textes nicht richtig übertragen hast. Das kann schnell passieren. Versuche, den Fehler zu finden, und kompiliere dann noch einmal! Wenn du den Fehler gar nicht finden kannst, dann bitte einen Mitschüler oder eine Mitschülerin um Hilfe.

Klicke auf das grüne Dreieck ! Dann öffnet sich ein schwarzes Fenster, in dem unter anderem Folgendes steht:

Hallo Welt!

Du hast das Programm ausgeführt. Was das bedeutet, erfährst du ebenfalls später im Theorieteil.

Hier noch einmal eine kurze Übersicht des Editors:

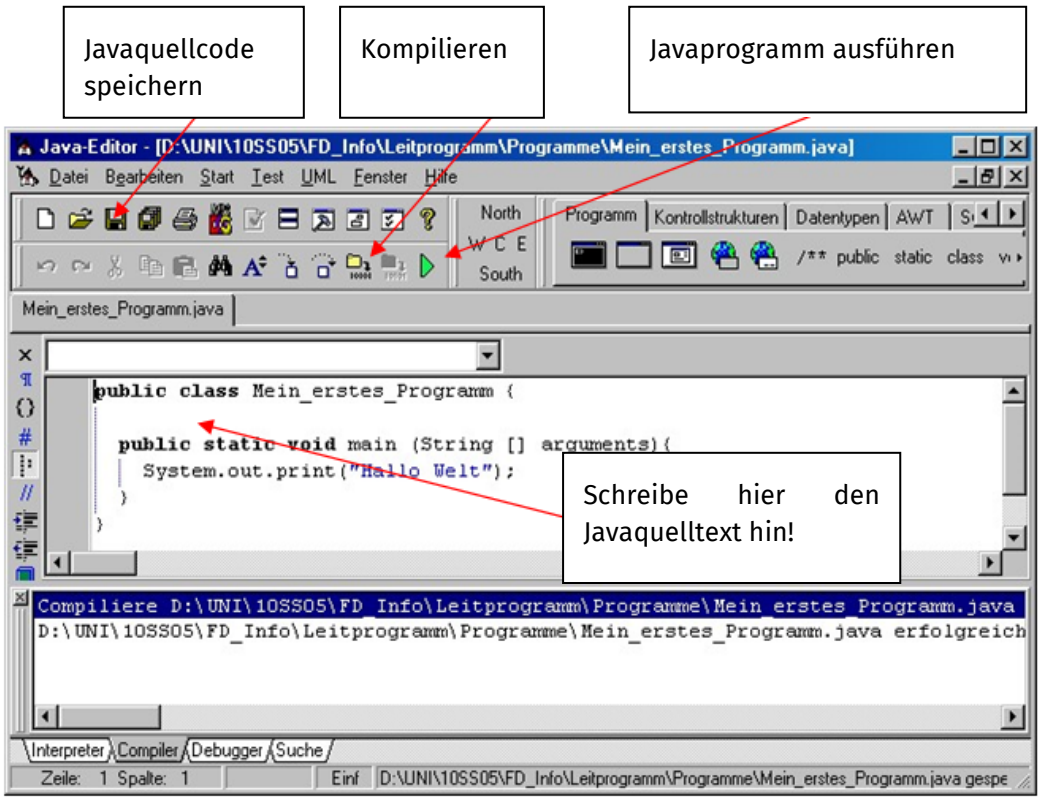


Abbildung 2: Der Editor

THEORIE

Was hast du in der Aufgabe oben gemacht?

Du hast, vielleicht das erste Mal, ein Programm geschrieben. Dazu hast du einen vorgegebenen Text, den **Quelltext** deines Programms, mit Hilfe des *JavaEditors* geschrieben und gespeichert. Diesen Teil des Programmierens nennt man **editieren**. Deshalb nennt man Programme, mit denen man Quellcodes schreiben und speichern kann auch **Editoren**.

Im nächsten Schritt hast du das Programm kompiliert. Weißt du, was das bedeutet? Ein Computer bzw. sein Prozessor versteht zunächst einmal nur die Anweisungen des Befehlssatzes. Dabei handelt es sich um sehr wenige und sehr einfache Befehle. Wenn man mit diesen wenigen Anweisungen ein größeres Programm schreiben müsste, bräuchte man sehr lange und der Quelltext würde enorm unübersichtlich, zumal dieser binär codiert wäre. Aus diesem Grund hat man **höhere Programmiersprachen** entwickelt, die viel mehr Anweisungen bieten. Der Quelltext, den du geschrieben hast, ist in der höheren Programmiersprache Java geschrieben. Mit höheren Programmiersprachen ist das Programmieren leichter und weniger aufwändig. Java ist die Programmiersprache, die du in den nächsten Wochen lernen wirst. Wie erfährt der Computer aber nun, welche Maschinenbefehle ausgeführt werden müssen, wenn er einen Befehl in einer höheren Programmiersprache erhält? Für diese Aufgabe gibt es **Compiler**. Sie übersetzen einen Quelltext, der in einer höheren Programmiersprache geschrieben ist, in Maschinensprache, also so, dass der Prozessor ihn versteht. Nach dem Kompilieren weiß der Computer also, was er tun soll. Das Javaprogramm wurde dann nämlich in ein Programm in Maschinensprache übersetzt.

Dein nächster Schritt war in der obigen Aufgabe das **Ausführen des Programms**. Es öffnete sich ein schwarzes Fenster in dem „Hallo Welt“ stand. Dieses schwarze Fenster nennt man **Eingabeaufforderung**. Ein „Programm ausführen“ bedeutet also, dass der Computer die Anweisungen befolgt, die im Quelltext des Programms formuliert sind. Du kannst dein Programm jetzt beliebig oft vom Computer ausführen lassen.

Eine Software, mit der man Quellcodes nicht nur editieren und speichern, sondern auch kompilieren und ausführen kann, heißt **Entwicklungsumgebung**. Der *JavaEditor* ist die Entwicklungsumgebung, mit der du im Rahmen dieses Leitprogramms arbeiten wirst.

SICHERUNGSPHASE

Aufgabe 2

Finde einen Mitschüler und/ oder eine Mitschülerin, der/die auch gerade bei der Aufgabe 2 angekommen ist!

a) Diskutiert die Bedeutung der fettgedruckten Begriffe im Theorieteil!
b) Fragt euch gegenseitig die Begriffe ab! Dabei darf der/die Befragte natürlich nicht im Theorieteil nachsehen. Der/die Abfragende sollte dagegen genau nachsehen, ob der/die Befragte die richtigen Antworten gibt.

Aufgabe 3

Siehe dir die Punkte an, die in Abbildung 3 durch die Pfeile markiert sind. Beantworte folgende Fragen zu den einzelnen Punkten.

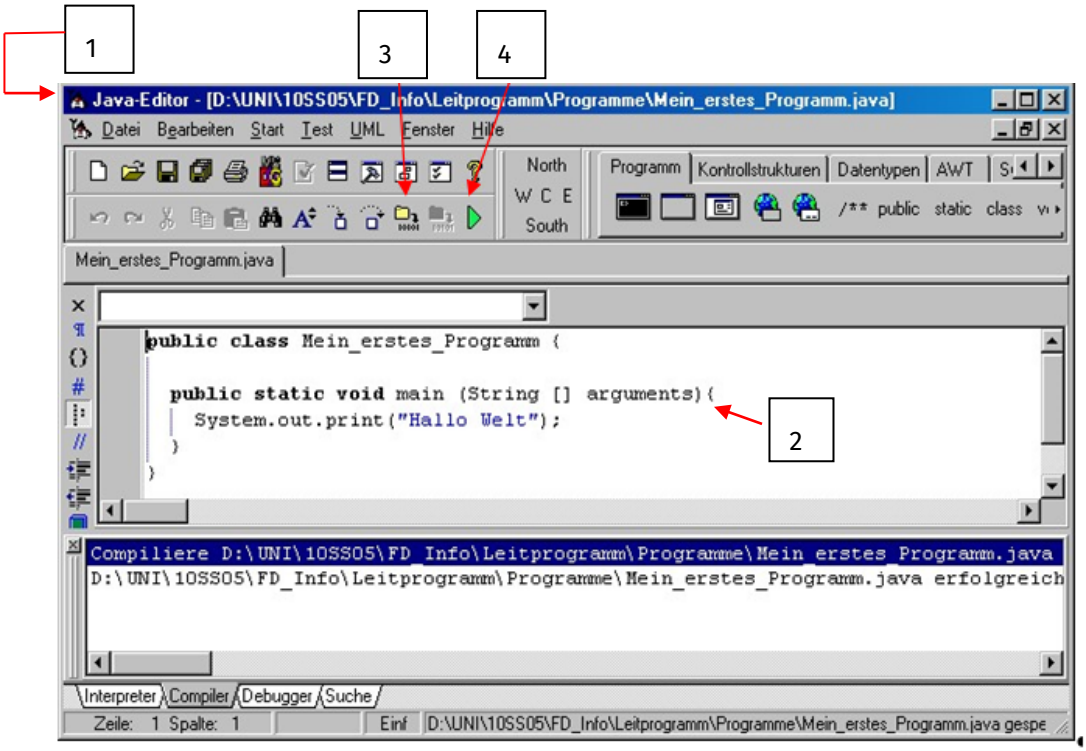


Abbildung 3: Aufgabe zum Editor

Zu 1

a) Zu welchem Programm gehört dieses Fenster?

_____

b) Wie nennt man solche Programme?

_____

c) Was kann man mit ihnen machen?

_____

Zu 2

a) Was schreibt man hier hinein?

_____

b) Wie nennt man das Schreiben davon?

_____

Zu 3

a) Wie nennt man das, was der Computer macht, wenn du diesen Knopf drückst?

_____

b) Was bedeutet das?

_____

c) Warum schreibt man ein Quellprogramm nicht so, dass der Computer es direkt verstehen kann?

_____

Zu 4

a) Wie nennt man das, was der Computer macht, wenn du diesen Knopf drückst?

b) Was bedeutet das?

Vergleiche deine Lösungen mit den Musterlösungen am Ende des Kapitels!

THEORIE

Bis hierhin hast du erstmal nur gelernt, wie man ein vorgegebenes Programm editiert, kompiliert und ausführt. Aber was macht das Programm "Mein_erstes_Programm.java"?

Nun wirst du dich mit der Bedeutung, der sogenannten Semantik, des Quelltextes befassen:

```
public class Mein_erstes_Programm {


  public static void main (String [] arguments){
    System.out.print("Hallo Welt");
  }


}
```

Die Ausführung dieses Programms bewirkt, dass die Textzeile **Hallo Welt** auf dem Bildschirm angezeigt wird. Man nennt das **Ausgabe** oder **Output**.

SICHERUNGSPHASE

Aufgabe 4

Betrachte noch einmal den Quelltext. Welche Zeile des Programms könnte die Ausgabe bewirkt haben?

Versuche das Programm so umzuschreiben, dass es einen anderen Text als *Hallo Welt* ausgibt!

LERNKONTROLLE

Fühlst du dich sicher im Stoff? Dann kannst du bei deiner Lehrkraft den Kapiteltest machen. Wenn du ihn bestehst, darfst du das nächste Kapitel oder das Additum bearbeiten.

ADDITUM

Für diese Aufgabe darfst du dir gerne einen Partner oder eine Partnerin suchen. Diese/r sollte an derselben Stelle angelangt sein wie du.

In diesem Kapitel hast du gelernt, dass ein Computer zunächst einmal nur sehr einfache und sehr wenige Anweisungen versteht. Man hat höhere Programmiersprachen eingeführt, weil man, wenn man mit diesen wenigen Anweisungen ein größeres Programm schreiben würde, sehr lange bräuchte und der Quelltext sehr unübersichtlich würde.

Recherchiere im Internet, und beantworte die folgenden Fragen!

a) Wie nennt man die Sprache, die der Computer direkt versteht?

b) Welcher Teil des Computers führt die Befehle in dieser Sprache wirklich aus?

c) In welcher Form liegen die Befehle in dieser Sprache vor?

d) Nenne ein paar Beispiele dafür, was die einzelnen Befehle bewirken!

e) Ist es möglich, ein Programm direkt in dieser Sprache zu schreiben?

f) Wie nennt man die Menge aller Befehle eines Prozessors?

g) Bei welchen Prozessoren hat man eher viele und bei welchen eher wenige Befehle?

Aufgabe 3

1.
   a) JavaEditor
   b) Entwicklungsumgebung
   c) Editieren, speichern, kompilieren, ausführen
2.
   a) Quellcode des Programms
   b) Editieren
3.
   a) Kompilieren
   b) Übersetzten eines Quelltextes in einer höheren Programmiersprache in Anweisungen, die der Computer versteht
   c) Ein Computer versteht nur sehr einfache und sehr wenige Anweisungen – die Befehle des Befehlssatzes. Wenn man mit diesen wenigen Anweisungen ein größeres Programm schreiben müsste, bräuchte man sehr lange und der Quelltext würde sehr unübersichtlich.
4.
   a) Ausführen
   b) Der Computer tut das, was ihm durch den Quelltext aufgetragen wird.

Aufgabe 4

Die Ausgabe am Bildschirm wird durch die folgende Programmzeile bewirkt:

```
System.out.print("Hallo Welt");
```

Ersetzt man `"Hallo Welt"` durch einen anderen Text, z. B. `"Dieses ist mein erstes Programm"`, dann wird dieser Text beim Ausführen des Programms am Bildschirm ausgegeben.

a) Maschinensprache, Befehlssatz, Maschinenbefehle

b) Der Prozessor

c) Die Befehle liegen in Binärcode vor, d. h. es sind Folgen von 0en und 1en.

d) Arithmetische Operationen: führen Berechnungen durch

   Speicheroperationen: übertragen Daten zwischen Prozessorregistern und Speicher

   Vergleichsoperationen: Vergleich von Werten

   Steueroperationen: Verzweigungen, die den Ablauf des Programms beeinflussen

e) Das ist sehr schwer. Es handelt sich um eine für den Menschen kaum lesbare Sprache, die allenfalls von Experten mit einem so genannten Maschinensprachemonitor bearbeitet werden kann.

f) Will man die Menge aller Befehle eines Prozessors beschreiben, so wird der Begriff Befehlssatz bevorzugt.

g) Relativ große Befehlssätze findet man in CISC-Prozessoren, möglichst kleine Befehlssätze werden in RISC-Prozessoren angestrebt.

KAPITEL 2: PROGRAMMAUFBAU

ÜBERSICHT

Das Programm *Mein_erstes_Programm.java* aus Kapitel 1 hat einen sehr einfachen Quelltext. Bevor man kompliziertere Java-Quelltexte verstehen bzw. selber schreiben kann, muss man wissen, wie sie grundsätzlich aufgebaut sind. Sie bestehen aus einer Vielzahl von Komponenten, wovon einige immer vorkommen müssen und einige nicht. Diese einzelnen Komponenten werdet ihr euch im Rahmen dieses Kapitels ansehen und ihre Bedeutung kennenlernen.

LERNZIEL

In diesem Kapitel wirst du lernen, wie Javaquelltexte aufgebaut sind. Du lernst also, welche Bestandteile zu einem Javaprogramm gehören und welche Funktion sie haben.

THEORIE

Anhand des Programms *Kreisumfang* soll der grundsätzliche Aufbau von Javaprogrammen veranschaulicht werden. Es ist nicht nötig, dass du dieses Programm verstehst.

```
public class Kreisumfang {
  //Ausgabe der Überschrift
static void ueberschrift(){
  System.out.println("\n KREISUMFANG \n");
}
//Berechnung des Umfangs
static double umfang (double r){
  return r* 2.0 * 3.14159;
}
public static void main (String[] arg) {
  double r;
  double u;
  ueberschrift();
  System.out.println("  r  | Kreisumfang ");
  for (r=1; r<=9; r++) {
    u = umfang(r);
    System.out.print(" ");
    System.out.print(r);
    System.out.print(" | ");
    System.out.println(u);
}}}
```

Methoden

Deklarationen

Hauptprogramm

```
public class Kreisumfang {…}
```

Der Name der Klasse muss gleich dem Namen der Datei sein, in der der Quelltext gespeichert ist. Deshalb ist dieses Programm in der Datei *Kreisumfang.java* gespeichert. In den geschweiften Klammern steht das gesamte Programm.

```
public static void main (String[] arg) {...}
```

Ein Programm hat ein **Hauptprogramm**. Es heißt immer *main*. Bei der Ausführung des Programms beginnt der Rechner beim Hauptprogramm. Der Quellcode des Hauptprogramms wird in die geschweiften Klammern geschrieben.

```
double r;
double u;
```

Die zu verarbeitenden Daten eines Programms werden in dafür vorgesehenen Behältern, sogenannten **Variablen**, gespeichert. Hier haben wir die Variablen r und u, die beide Dezimalzahlen (double) speichern können. Wird im Quelltext eine Variable benutzt, so steht sie für den in ihr gespeicherten Wert. Durch die **Deklaration** gibt man dem Computer an, welche Variablen genutzt werden sollen und welche Art von Werten in den Variablen gespeichert werden soll.

Nach den Deklarationen folgen die **Anweisungen**. Diese sagen dem Computer, was er tun soll. Jede Deklaration und jede Anweisung wird durch ein Semikolon (;) abgeschlossen.

```
//Ausgabe der Ueberschrift
```

Kommentare geben Erläuterungen zum Programm. Sie beginnen mit // und gehen bis zum Zeilenende. Bei der Kompilierung und Ausführung werden sie vom Computer ignoriert. Sie dienen der besseren Lesbarkeit des Programms für den Programmierer oder andere Personen, die sich den Quelltext ansehen.

Sie haben allerdings keinerlei Einfluss auf das Programm.

```
static void ueberschrift(){
System.out.println("\n KREISUMFANG \n");
```

Um den Quelltext besser strukturieren zu können, können Teile von ihm in sogenannten **Methoden** ausgelagert werden. Mit Hilfe von Methoden kann häufig genutzter Code mehrfach verwendet werden.

```
ueberschrift();
```

Das Ausführen des in einer Methode ausgelagerten Quellcodes wird durch einen Methodenaufruf angestoßen.

AUFGABEN

Aufgabe 1 – Programmaufbau

```
public class Rechnung {

  //Ausgabe der Ueberschrift
  static void ueberschrift(){
    System.out.println("\n Irgendeine Rechnung \n");
  }
  // Dieses Programm gibt das Ergebnis von x+y aus.
  public static void main (String[] arg) {
    int x;
    int y;

    ueberschrift();

    x = 100 + 4 * 3 / 4;
    y = 12345;

    System.out.println(x + y);
  }

}
```

Wenn du den Theorieteil gut verstanden hast, kannst du versuchen, die folgende Aufgabe zu lösen. Versuche dabei, so wenig wie möglich im Theorieteil nachzusehen.

a) Beschrifte das Programm wie oben im Theorieteil mit den folgenden Begriffen:

Kommentare, Hauptprogramm, Anweisungen, Methode, Methodenaufruf, Deklarationen

Und beantworte die Fragen:

b) Was sind hier die Variablen?

c) Unter welchem Namen muss das Programm gespeichert werden?

d) Wie heißt das Hauptprogramm?

e) Wo steht der Quellcode des Hauptprogramms?

f) Wird die Variable y im Hauptprogramm oder in der Methode deklariert?

g) Was macht der Computer beim Kompilieren und Ausführen mit den Kommentaren?

h) Was darf bei Anweisungen und Deklarationen nie fehlen?

i) Womit beginnen Kommentare?

j) Wozu dienen Methoden?

k) Wie kann man den Quelltext einer Methode ausführen?

l) Wozu braucht man Variablen?

m) Wozu braucht man Deklarationen?

n) Wozu braucht man Anweisungen?

LERNKONTROLLE

Bevor du die Lernkontrolle machst, solltest du die fettgedruckten Begriffe aus dem Theorieteil kennen und verstanden haben. Du solltest auch den Aufbau eines Java-Programms kennen. Wenn das der Fall ist, dann kannst du bei deiner Lehrkraft den Kapiteltest machen. Wenn du ihn bestehst, darfst du das nächste Kapitel oder das Additum bearbeiten.

Du kannst diese Tests durchaus auch alleine machen. Sei bei der Kontrolle dir selbst gegenüber aber ehrlich, oder lasse die Tests durch deine Eltern oder Geschwister kontrollieren.

ADDITUM

Diese Aufgabe kannst du mit einem Partner bearbeiten.

In diesem Kapitel kamen zwei Programme vor: *Rechnung.java* und *Kreisumfang.java*

Du hast dich ausschließlich mit dem Aufbau und den Komponenten dieser Programme befasst. Hast du auch eine Vorstellung davon, was die Programme machen?

TIPP: Die Anweisungen eines Programms werden immer der Reihe nach durchgegangen.

Betrachte den Quelltext von *Rechnung.java* !

```
public class Rechnung {

  //Ausgabe der Ueberschrift
  static void ueberschrift(){
    System.out.println("\n Irgendeine Rechnung \n");
  }
  // Dieses Programm gibt das Ergebnis von x+y aus
  public static void main (String[] arg) {
    int x;
    int y;

    ueberschrift();

    x = 100 + 4 * 3 / 4;
    y = 12345;

    System.out.println(x + y);
  }

}
```

Wenn du vor diesem Leitprogramm noch nie programmiert hast, dann ist es schwer, die folgenden Aufgaben zu beantworten. Es könnte dir helfen, das Programm auf dem Computer auszuführen! Wie das geht, hast du in Kapitel 1 gelernt.

a) Was bedeuten die einzelnen Deklarationen und Anweisungen?
b) Was macht das Programm?
c) Schreibe das Programm so um, dass es eine andere Rechnung durchführt!

Betrachte den Quelltext von Kreisumfang.java !

```
public class Kreisumfang {
  //Ausgabe der Ueberschrift
  static void ueberschrift(){
    System.out.println("\n KREISUMFANG \n");
  }

  //Berechnung des Umfangs
  static double umfang(double r){
    return r * 2.0 * 3.14159;
  }

  public static void main (String[] arg) {
    double r;
    double u;
     ueberschrift();
    System.out.println("  r  | Kreisumfang ");
    for (r=1; r<=9; r++){

      u = umfang(r);

      System.out.print(" ");
      System.out.print(r);
      System.out.print(" | ");
      System.out.println(u);
    }
  }

}
```

a) Was macht das Programm?
b) Schreibe das Programm so um, dass es die Kreisfläche zu einem Radius bestimmt. Wenn du die Formel dazu nicht mehr auswendig kannst, dann besorg sie dir.

MUSTERLÖSUNGEN ZU DEN AUFGABEN AUS KAPITEL 2

Aufgabe 1

a)

```
public class Rechnung {

  //Ausgabe der Ueberschrift
  static void ueberschrift(){
    System.out.println("\n Irgendeine Rechnung \n");
  }

  // Dieses Programm gibt das Ergebnis von x+y aus
  public static void main (String[] arg) {
    int x;
    int y;

    ueberschrift();

    x = 100 + 4 * 3 / 4;
    y = 12345;

    System.out.println(x + y);
  }

}
```

Methode

Deklarationen

Methodenaufruf

Haupt-programm

Anweisungen

b) x, y
c) Rechnung.java
d) main
e) Das Hauptprogramm steht bei

```
public static void main (String[] arg) {…}
```

zwischen den geschweiften Klammern.

f) Die Variable y wird im Hauptprogramm deklariert.
g) Beim Kompilieren und Ausführen werden die Kommentare vom Rechner ignoriert.
h) Nach Anweisungen und Deklarationen steht immer ein Semikolon (;).
i) Ein Kommentar beginnt mit //.
j) Methoden dienen der Strukturierung des Quelltextes. Häufig verwendeter Quellcode muss nur einmal geschrieben werden.
k) Der Quelltext einer Methode kann durch einen Methodenaufruf ausgeführt werden.

l) Variablen braucht man, um Werte bzw. Daten zu speichern. Durch den Namen der Variable erhält man den gespeicherten Wert.
m) Durch Deklarationen gibt man dem Computer an, welche Variablen genutzt werden sollen und welche Art von Werten in den Variablen gespeichert werden soll.
n) Die Anweisungen sagen dem Computer, was er tun soll.

Additum

1.+2.

```
int x;
int y;
```

Deklaration von den Variablen x und y, zum Speichern von ganzzahligen Werten. Die gespeicherten Werte können mit x und y angesprochen werden.

```
ueberschrift();
```

Die Methode *ueberschrift* wird aufgerufen und ihr Quelltext dadurch ausgeführt. Das bewirkt, dass die Zeile „Irgendeine Rechnung“ am Bildschirm ausgegeben wird.

```
x = 100 + 4 * 3 / 4;
y = 12345;
```

Die Ergebnisse der Rechnungen werden in x und y gespeichert.

```
System.out.println(x + y);
```

Das Ergebniss von x+y wird am Bildschirm ausgeben.

4.

```
double r;
double u;
```

Deklaration von Variablen für Dezimalzahlen.

```
ueberschrift();
```

Aufruf der Methode ueberschrift(), die eine Überschrift am Bildschirm ausgibt

```
for (r=1; r<=9; r++){...}
```

Die Anweisungen in den geschweiften Klammern werden solange ausgeführt, bis r < oder = 9 ist. r ist beim ersten Durchgang = r und wird dann pro Durchgang um eins erhöht.

```
u = umfang(r);
```

Die Methode *umfang()* wird aufgerufen und das, was die Methode berechnet, wird in u gespeichert.

```
static double umfang(double r){
        return r * 2.0 * 3.14159;
 }
```

Das Ergebnis von r*2.0*3.14159 wird berechnet und zurückgegeben, so dass es in u gespeichert werden kann.

```
System.out.print(" ");
System.out.print(r);
```

```
System.out.print(" | ");
System.out.println(u);
```

Ausgabe von r und u und einigen Zeichen

Das Programm berechnet also den Kreisumfang u für Radien r von 1 bis 9 und gibt ihn jeweils am Bildschirm aus.

KAPITEL 3: DATENTYPEN UND OPERATIONEN

ÜBERSICHT

Du hast schon gelernt, dass der Rechner Daten unterschiedlicher Typen verarbeiten kann. Wie funktioniert nun diese Verarbeitung? Was kann man also mit diesen Daten machen? Man kann mit ihnen operieren. Beispiele dafür sind die Grundrechenarten. Wenn man nun eine Operation auf Daten bestimmter Datentypen ausführt, von welchem Datentyp ist das Ergebnis dann? Das musst du beim Programmieren wissen, damit du weißt, mit welchem Datentyp die Variable, in die das Ergebnis gespeichert werden soll, deklariert sein muss.

LERNZIEL

In diesem Kapitel wirst du lernen, welche einfachen Datentypen es gibt, welche Operationen man auf ihnen ausführen kann und von welchem Datentyp die Ergebnisse sind.

THEORIE

Ein Computer muss viele Daten verarbeiten können. Du weißt bereits, dass es verschiedene Arten von Daten, sogenannte **Datentypen** oder einfach nur **Typen**, gibt. Ein Computer kennt z. B. unter anderem Zahlen, Zeichen oder auch Texte.

In den letzten beiden Kapiteln bist du schon auf Daten und Datentypen gestoßen. Im Programm *Mein_erstes_Programm.java* hattest du mit Texten zu tun.

Die Ausgabe mit

```
System.out.print( )
```

ist unter anderem für Daten vom Datentyp Text (bei Java String) vorgesehen. Um dem Computer zu signalisieren, dass es sich bei einem Datum um einen Text handelt, muss er vom Programmierer als Text gekennzeichnet werden. Dazu setzt man Texte in Anführungszeichen (oben).

“Dies ist ein Text!“

Die Anführungszeichen signalisieren dem Rechner also, dass es sich um einen Text handelt. In der Fachsprache bezeichnet man einen so gekennzeichneten Text als **String**.

Im letzten Kapitel wurde im Programm *Kreisumfang.java* mit Dezimalzahlen gerechnet, um zu einem Radius den Kreisumfang zu bestimmen. In Java sind das Werte vom Datentyp **double**.

Welche Datentypen gibt es sonst noch bei Java?

Zunächst betrachtest du nur die sogenannten „einfachen Datentypen“:

Einfache Datentypen

Datentyp	**in Java**	**Beispiele**
ganze Zahlen (Integer)	int	1 -2 0 23 42
Gleitkommazahlen, Dezimalzahlen	double	3.14159 -2.71828
Wahrheitswerte	boolean	true false
Zeichen (Character)	char	’a’ ’3’ ’+’ ’$’ ’.’
Zeichenkette, Text	String	“Hallo Welt“ “3 ist eine ganze Zahl“

Ganze Zahlen und Dezimalzahlen sind dir aus der Mathematik bekannt. Auch bei Java können sie negativ oder positiv sein. Es gibt nur zwei Wahrheitswerte: wahr (true) oder falsch (false). Zeichen werden in Apostrophe (’ ’) und Texte in Anführungszeichen (“ “) gesetzt.

Bei Zeichen und Texten muss man beachten, dass zwischen Groß- und Kleinbuchstaben unterschieden wird. Für einen Rechner sind ein Kleinbuchstabe und sein zugehöriger Großbuchstabe völlig verschiedene Zeichen.

Mit Hilfe von Java kann man Daten verarbeiten. Z. B. kann man eine einfache Addition zweier ganzer Zahlen durchführen. Die Berechnung wird durch einen sogenannten **Operator** bewirkt. In diesem Fall handelt es sich bei diesem Operator um ein +.

Diese Kombination von Daten eines bestimmten Datentyps mit Hilfe von Operatoren nennt man **Ausdrücke**. Die Daten bezeichnet man dabei als **Operanden**. Bei der Auswertung eines Ausdrucks ergibt sich ein Datum, das wiederum einen Datentyp hat. Dieser ist der **Ergebnisdatentyp**.

Addiert man beispielsweise zwei Werte vom Datentyp int, so erhält man wieder einen Wert vom Datentyp int.

Beispiel:

3 und 2 sind Daten vom Datentyp int.

+ ist ein Operator für Daten vom Datentyp int.

3 + 2 ist ein Ausdruck mit den Operanden 3 und 2 und dem Operator +.

Der Ergebnisdatentyp des Ausdrucks 3 + 2 ist vom Datentyp int.

SICHERUNGSPHASE

Aufgabe 1

Fülle die Lücken [ ] aus und beantworte die Fragen:

1. 'a' ist vom Datentyp [ ].

2. – ist ein [ ].

3. 2 <= 3 oder 3 + 4 bezeichnet man als [ ].

4. 2 und 3 sind die [ ] in dem Ausdruck 2 + 3.

5. 2 < 3 wird zu [ ] ausgewertet.

6. Sind bei dem Ausdruck 3 == 4 (== bedeutet „ist gleich“) die Datentypen der Operanden und der Ergebnistyp gleich?

7. Sind bei dem Ausdruck 3 + 4 die Datentypen der Operanden und der Ergebnistyp gleich?

8. + ist [ ] im Ausdruck 3+2.

9. “a“ ist vom Datentyp [ ].

10. Haben Dezimalzahlen den Datentyp int?

11. 1/2 ist vom Datentyp [ ].

THEORIE

Was gibt es nun für Operationen? Die meisten Operationen sind dir bekannt.

Grundrechenarten

Das sind zunächst einmal die **Grundrechenarten** (+, - , * , /). Diese kann man sowohl auf int- als auch auf double-Werte anwenden.

3 + 2

3.2 + 2.1

Die Bedeutung dieser Operatoren ist bis auf die Division bei int, die sogenannte ganzzahlige Division, klar.

Bei der **ganzzahligen Division** werden einfach die Stellen hinter dem Komma vernachlässigt.

Beispiel: 3/2 wird ausgewertet zu 1

4/7 wird ausgewertet zu 0

2/2 wird ausgewertet zu 1

Die ganzen Zahlen sind eine Teilmenge der natürlichen Zahlen. Deshalb kann man auch Operationen durchführen, bei denen einer der beiden Operanden vom Typ int und der andere vom Typ double ist. Das Ergebnis ist dann natürlich vom Typ double.

1.2 + 2 wird ausgewertet zu 3.2

Intern formt der Computer den Integerwert zu einem Wert vom Typ double um und führt dann die Operation durch. Das Umformen zum passenden Typ nennt man **Typkonversion**. Sie kommt auch bei anderen Datentypen vor.

Verkettung

Beim Datentyp String hat der Operator + eine ganz andere Bedeutung. Er verkettet zwei Strings.

“Hal“ +“lo“ wird ausgewertet zu “Hallo“.

Vergleichsoperationen

Um zwei Werte miteinander zu vergleichen, gibt es Vergleichsoperationen.

3 < 2

3 != 4

Das sind die folgenden Operationen:

<	kleiner
<=	kleiner oder gleich (kleinergleich)
>	größer
>=	größer oder gleich (größergleich)
==	gleich
!=	ungleich

Diese Operationen können natürlich auf Zahlen angewendet werden. Zum Teil lassen sie sich aber auch auf die anderen Datentypen anwenden. Z. B. können Texte oder Buchstaben auf Gleichheit überprüft werden. Der Ergebnistyp einer Vergleichsoperation ist immer boolean.

SICHERUNGSPHASE

Aufgabe 2

a) Wie werden die folgenden Ausdrücke ausgewertet?
b) Von welchen Datentypen sind die Ergebnisse?

Klammern haben dieselbe Bedeutung wie in der Mathematik.

Beispiel: 23.4 + 7 Ergebnis: 30.4

Ergebnistyp: double

- 30 - 5
- (10 / 3) + 0.5
- ‘a’ == ‘b’
- “text” == “Text”
- “Pro” + “gramm”
- "ab“ != “cd“
- 6.6 / 3.3
- (10 / 4) == 2
- (1/ 3) * 1234567891234
- ‘Q’ == ‘q’
- (“Progr” + “amm”) == “Program”
- 11 <= (22/2)
- 11 < (22/2)
- + 2) == 3.0

THEORIE

Boolsche Operationen

Boolsche Operationen werden auf Wahrheitswerte angewandt. Es gibt das logische Und (&&), das logische Oder (||) und das logische Nicht (!).

A && B ist nur dann erfüllt, wenn sowohl A als auch B erfüllt ist.

A || B ist dann erfüllt, wenn A oder B oder beides gilt.

! A ist erfüllt, wenn A nicht gilt.

Wenn du dir noch unsicher bist, dann schau in der Tabelle nach:

	Ausdruck	**Auswertung**
logisches Und	true && true false && true true && false false && false	true false false false
logisches Oder	true || true false || true true || false false || false	true true true false
logisches Nicht	! true ! false	false true

Jetzt hast du alle wichtigen Operationen kennengelernt. Die Tabelle auf der folgenden Seite stellt eine Übersicht davon dar:

Datentyp Operanden	**Operator**	**Bedeutung**	**Ergebnis-datentyp**	**Beispiele**
int	+	Addition	int	3 + 2
	*	Multiplikation	int	3 * 2
	-	Subtraktion	int	3 - 2
	-	Negation	int	-3
	/	**ganzzahlige** Division	int	3 / 2
	==	gleich	boolean	3 ==2
	!=	ungleich	boolean	3 != 2
	<	kleiner	boolean	3 < 2
	>	größer	boolean	3 > 2
	<=	kleinergleich	boolean	3 <= 2
	>=	größergleich	boolean	3 >= 2
double	+	Addition	double	3.2 + 2.0
	*	Multiplikation	double	3.2 * 2.0
	-	Subtraktion	double	3.2 – 2.0
	-	Negation	double	-3.0
	/	Division	double	3.2 / 2.0
	==	gleich	boolean	3.2 == 2.0
	!=	ungleich	boolean	3.2 != 2.0
	<	kleiner	boolean	3.2 < 2.0
	>	größer	boolean	3.2 > 2.0
	<=	kleinergleich	boolean	3.2 <= 2.0
	>=	größergleich	boolean	3.2 >= 2.0
boolean	||	logisches Oder	boolean	true || false
	&&	logisches Und	boolean	true && false
	!	logisches Nicht	boolean	! true
	==	gleich	boolean	true == false
	!=	ungleich	boolean	true != false
char	==	gleich	boolean	'a' == 'b'
	!=	ungleich	boolean	'a' != 'b'
String	+	Verkettung	String	"Te" + "xt"
	==	gleich	boolean	"Te" == "xt"

Du brauchst die Tabelle jetzt nicht auswendig zu lernen. Nutze sie einfach zum Nachschauen.

Mit den unterschiedlichen Operatoren können auch komplexere Ausdrücke erstellt werden:

(2 > 3) && (3 < 9)

Wie in der Mathematik werden die Ausdrücke in den Klammern zuerst ausgewertet. Daraus ergibt sich für den obigen Ausdruck

false && true

und das wird zu false ausgewertet.

SICHERUNGSPHASE

Aufgabe 3

Welche Ergebnisse haben die Ausdrücke? Von welchen Datentypen sind die Ergebnisse?

- !((23 + 17) == 40)
- ((23.0 + 17) != 40.0) && true
- (10 / 3) + 3.1
- true || (2 > 3)
- (!('a' == 'b')) II (!(!(2==2)))
- ('a' == 'a') && (2 < 3)
- (true && ('x' == 'x')) || false
- ("ab"+"cd") == "ab cd"
- (6.6 / 3.3) == (2 + 0.2)
- (10 / 4 == 2) || ('a' != 'b')
- (10 / 3 – 3) * 1234567891234
- 'Q' == 'q'
- !("Hallo" != "Hallo")
- (!(,A' == ,a')) == true

LERNKONTROLLE

Hast du verstanden, was man unter Datentypen, Operanden, Operatoren und Ausdrücke versteht? Kennst du die Bedeutung der einzelnen Operatoren? Wenn ja, dann melde dich zum Kapiteltest. Um den Kapiteltest zu bestehen, musst du in der Lage sein, eine ähnliche Aufgabe wie Aufgabe 3 zu lösen.

ADDITUM

Betrachte den folgenden Ausdruck: 'A' + 0

a) Auf welche Datentypen wird hier die Operation + angewandt?

b) Schreibe ein Javaprogramm, das dir den Wert des Ausdrucks ausgibt! Was gibt das Programm aus?

c) Versuche herauszubekommen, warum genau dieser Wert ausgegeben wird!
TIPP: ASCII

d) Was muss demnach bei 'B' + 0 ausgegeben werden?

e) Welchen Wert hat der Ausdruck 'A' + 'B'?

f) Welchen Wert hat der Ausdruck '1' + '2'?

MUSTERLÖSUNGEN ZU DEN AUFGABEN AUS KAPITEL 3

Aufgabe 1

a) char
b) Operator
c) Ausdrücke
d) Operanden
e) true
f) Nein, die Operanden 3 und 4 sind vom Datentyp int und der Ergebnistyp ist boolean.
g) Ja, Operanden und Ergebnis sind vom Typ int.
h) der Operator
i) “a“ ist vom Datentyp String
j) Nein, Dezimalzahlen haben den Datentyp double.
k) double (,da 0.5 ist)

Aufgabe 2

	Ergebnis	**Ergebnisdatentyp**
23.4 + 7	30.4	double
30 – 5	25	int
(10 / 3) + 0.5	3.5	double
‘a’ == ‘b’	false	boolean
“text” == “Text”	false	boolean
“Pro” + “gramm”	“Programm“	String
"ab“ != “cd“	true	boolean
6.6 / 3.3	2.0	double
10 / 4 == 2	true	boolean
1/ 3 * 1234567891234	0	int
‘Q’ == ‘q’	false	boolean
(“Progr” + “amm”) == “Program”	false	boolean
11 <= (22/2)	true	boolean
11 < (22/2)	false	boolean
(1.0 + 2) == 3.0	true	boolean

Aufgabe 3

	Ergebnis	**Ergebnisdatentyp**
!((23 + 17) == 40)	false	boolean
((23.0 + 17) != 40.0) && true	false	boolean
10 / 3 + 3.1	6.1	double
true || (2 > 3)	true	boolean
(!('a' == 'b')) II (!(!(2==2)))	true	boolean
('a' == 'a') && (2 < 3)	true	boolean
(true && ('x' == 'x')) || false	true	boolean
("ab"+"cd") == "ab cd"	false	boolean
(6.6 / 3.3) == (2 + 0.2)	true	boolean
(10 / 4 == 2) || ('a' != 'b')	true	boolean
((10 / 3) – 3) * 1234567891234	0	int
'Q' == 'q'	false	boolean
!("Hallo" != "Hallo")	true	boolean
(!(,A' == ,a')) == true	true	boolean
!((23 + 17) == 40)	false	boolean

Aufgaben zum Additum

Betrachtung des folgenden Ausdrucks: 'A' + 0

a) Auf welche Datentypen wird hier die Operation + angewandt?

char und int

b) Schreibe ein Javaprogramm, das dir den Wert des Ausdrucks ausgibt.

```
public class Test {

  public static void main (String [] arguments){
    System.out.print( 'A' + 0 );
  }
}
```

Was gibt das Programm aus? 65

c) Versuche herauszubekommen, warum genau dieser Wert ausgegeben wird!

0 ist vom Typ Integer. Durch die Addition, die für Zahlen vorgesehen ist, wird das Zeichen 'A' in eine Zahl umgewandelt, und dann werden die beiden Werte addiert. Jedes Zeichen ist einer Integer-Zahl zugeordnet. Diese Zuordnung kann man in einer ASCII-Tabelle nachsehen.

d) Was muss demnach bei 'b' + 0 ausgegeben werden? 66

e) Welchen Wert hat der Ausdruck 'A' + 'B'? 131

f) Welchen Wert hat der Ausdruck '1' + '2'? 99

KAPITEL 4: VARIABLEN

ÜBERSICHT

Du weißt aus den vorherigen Kapiteln schon ungefähr, was Variablen sind. Jetzt wirst du dir Variablen noch einmal genauer ansehen. Um die Eigenschaften von Variablen besser zu veranschaulichen, wirst du ihren Inhalt am Bildschirm ausgeben lassen. Dafür musst du dich noch einmal mit der Ausgabe am Bildschirm befassen.

In diesem Kapitel wirst du interaktive Programme schreiben. Das bedeutet, dass die kompilierten Programme während der Ausführung über den Bildschirm und die Tastatur mit dem Benutzer kommunizieren. Um ein interaktives Programm schreiben zu können, brauchst du also noch die Möglichkeit, Eingaben über die Tastatur einzulesen und zu verarbeiten.

LERNZIEL

Du lernst, was Variablen sind, wie man sie deklariert und verwendet. Obwohl das Kapitel den Titel Variablen trägt, wirst du zudem lernen, welche Möglichkeiten man hat, Ausgaben am Bildschirm zu machen und Eingaben vom Benutzer über die Tastatur im Programm zu verwenden.

Aufgabe 1 – Vorbereitung

Bevor du nun mit den Variablen anfangen kannst, musst du ein paar Vorbereitungen treffen.

Ausgabe am Bildschirm

Wie du weißt, sollst du interaktive Programme schreiben. Wie Texte während der Ausführung des Programms ausgeben werden, weißt du.

Welche Anweisung verwendet man dafür?

Außerdem kann man mit

```
System.out.println("irgendein String")
```

eine ganze Zeile ausgeben. Das bedeutet, dass der in den Klammern angegebene String ausgegeben wird und dann in die nächste Zeile gesprungen wird.

Beispiel:

Programmtext	Ausgabe am Bildschirm
`System.out.println("**Einkaufsliste**");`	****Einkaufsliste****
`System.out.println("");`	
`System.out.println("3 Liter Milch" );`	**3 Liter Milch**
`System.out.println("500gr Weintrauben" );`	**500gr Weintrauben**
`System.out.println("5 x Joghurt");`	**5 x Joghurt**
`System.out.println("2 x Brot" );`	**2 x Brot**

Schreibt man in einen String `\n`, dann bewirkt das bei der Ausgabe des Strings einen Zeilenumbruch an der Stelle, wo \n steht.

Ein `\t` in einem String bewirkt einen Zeilenvorschub. Das sieht dann so aus, wie wenn man in einem Textverarbeitungsprogramm die Tabulatortaste verwendet.

Programmtext	Ausgabe
`System.out.print("1 \n 2 \n 3 \n 4");`	**1** **2** **3** **4**

Oft lassen sich Zeichen wie ö, ü, ä und ß nicht mit System.out.print... ausgeben. Deshalb schreibt man stattdessen in der Regel oe, ue, ae und ss.

Eingabe über die Tastatur

Wie bekommst du ein Programm nun dazu, während der Ausführung Eingaben über die Tastatur entgegenzunehmen, um sie dann weiterzuverarbeiten? Das ist nicht so einfach. Woher soll ein Programm wissen, ob eine Zahl oder ein Zeichen gemeint ist, wenn man die 3 auf der Tastatur drückt? Bis jetzt hast du noch nicht das Wissen, um ein Programm zu schreiben, das dies kann. Aber du kannst dir Hilfe holen:

1. **Besorge Dir die Datei Kon.java!**
2. **Lege sie in den Ordner, in dem du deine Java-Quelltexte liegen hast.**
3. **Kompiliere sie!**

Wenn du jetzt ein Programm schreibst und es im **gleichen Ordner** ablegst, hast du die Möglichkeit, Eingaben von der Tastatur zu lesen.

Mit den folgenden Methodenaufrufen kannst du einen Wert vom jeweiligen Datentyp über die Tastatur einlesen lassen:

`Kon.readInt( );`	Einlesen von ganzen Zahlen (int)
`Kon.readDouble( );`	Einlesen von Dezimalzahlen (double)
`Kon.readChar( );`	Einlesen von Zeichen (char)
`Kon.readString( );`	Einlesen von Texten (String)

Probiere es doch einmal aus:

1. **Schreibe ein Programm, das nach dem Namen des Benutzers fragt!**

   TIPP: Wenn du nicht genau weißt, wie du das hinbekommen sollst, dann orientiere dich an dem Programm *Mein_erstes_Programm.java*.

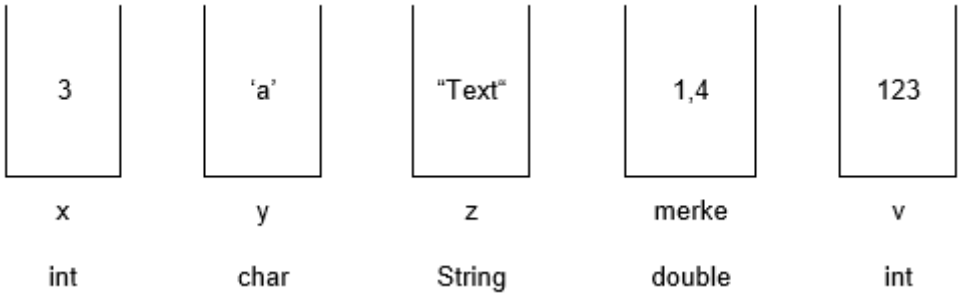
2. **Wenn das geklappt hat, dann erweitere dein Programm so, dass bei der Ausführung ein Wert über die Tastatur eingelesen wird!**

Arbeite erst weiter, wenn dein Programm funktioniert.

Dein Programm liest jetzt bei der Ausführung einen String über die Tastatur ein. Allerdings wird dieser String noch nicht verarbeitet. Das Programm ist in dieser Form also relativ sinnlos. Der String muss weiterverarbeitet werden. Dazu muss er zunächst einmal gespeichert werden. Du brauchst also Variablen.

THEORIE

Nun wirst du dir Variablen etwas genauer ansehen. Variablen sind benannte Behälter für Daten. Sie können ihren Wert ändern und haben einen Datentyp. Das bedeutet, dass man in einer Variablen nur Werte speichern kann, die den gleichen Datentyp wie die Variable haben. Man sagt: Wert und Variable müssen zuweisungskompatibel sein.



In der Darstellung gibt es die Variablen x, y, z und merke. x ist vom Datentyp int und enthält den Integerwert 3. y ist vom Datentyp char und enthält den Wert 'a' usw..

Jede Variable muss vor ihrer Verwendung **deklariert** werden. Das hast du im Laufe der letzten Kapitel sicherlich mitbekommen. Es bedeutet, dass Namen und Datentyp der Variablen bekannt gemacht werden. Der Compiler reserviert Speicherplatz für den Wert, der in der Variable abgelegt werden soll.

Die Deklaration für die obigen Variablen sieht so aus:

```
int x;
char y;
String z;
double merke;
int v;
```

Übrigens: Wenn in einer Deklaration mehrere Variablen von demselben Datentyp angelegt werden sollen, dann kann man diese auch alle durch Komma getrennt hinter die Typangabe schreiben.

Die Deklaration für die obigen Variablen kann also auch so aussehen:

```
int x, v;
char y;
String z;
double merke;
```

Mit dem =-Zeichen bewirkt man eine **Wertzuweisung**. Damit kann man einen Wert in einer Variablen speichern.

Die Wertzuweisungen für die obigen Variablen sehen wie folgt aus:

```
x = 3;
y = 'a';
z = "Text";
merke = 1,4;
```

```
v = 123;
```

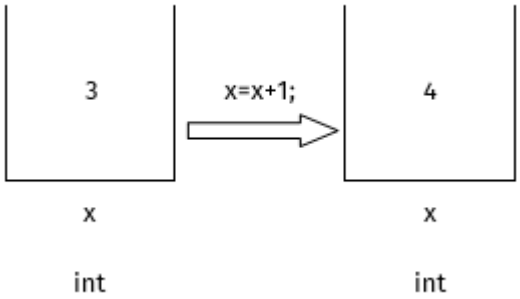
Da Variable und Wert zuweisungskompatibel sein müssen, müssen die linke und rechte Seite vom =-Zeichen in der Regel denselben Typ haben. Z. B. kann man in einer Variablen, die für int-Werte vorgesehen ist, keine Strings speichern.

```
x = "INDA - Gymnasium";
```
geht nicht

Nach der Wertzuweisung kann man die Variable anstelle des Werts benutzen. Man kann z. B. folgendes machen:

```
x = x+1;
```

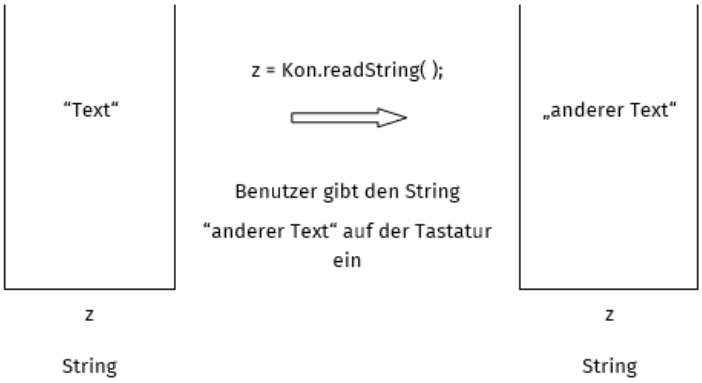
Zunächst wird der Ausdruck *x + 1* ausgewertet. Da 3 in *x* gespeichert ist, wird *x+1* zu 4 ausgewertet. Dann wird der ermittelte Wert der Variablen x zugewiesen. Also enthält x jetzt den Wert 4.



Man kann den Wert der Variablen am Bildschirm ausgeben lassen.

```
System.out.print(x);
```
4

Außerdem kann man mit `Kon.readInt( )`, `Kon.readDouble( )`, `Kon.readChar( )`, und `Kon.readString( )` Werte, die über die Tastatur eingegeben werden, in einer Variablen speichern.

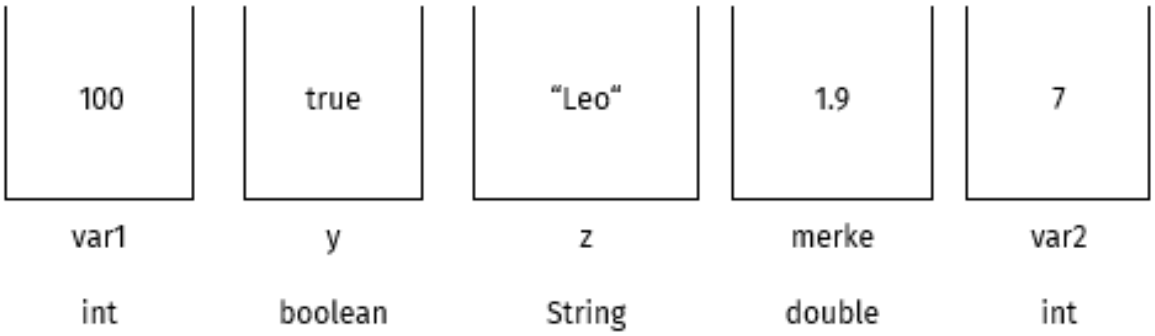


SICHERUNGSPHASE

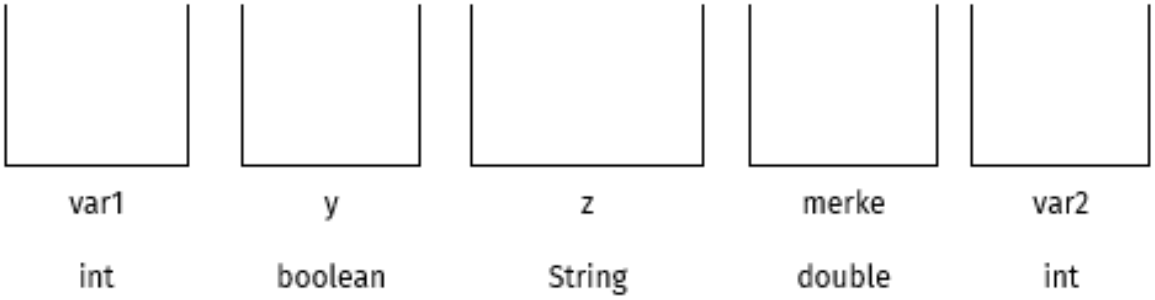
Aufgabe 2 – Umgang mit Variablen

Zeichne die durch die Anweisungen bewirkten Veränderungen in den Variablen ein.

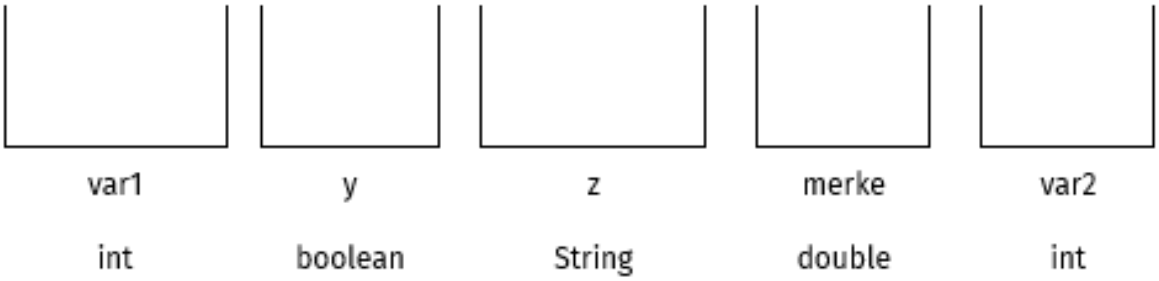
Empfehlung: Ziehe in regelmäßigen Abständen die Musterlösung heran, um sie mit deinem Programm zu vergleichen, da sich ein früh gemachter Fehler auf die ganze Lösung auswirken kann.



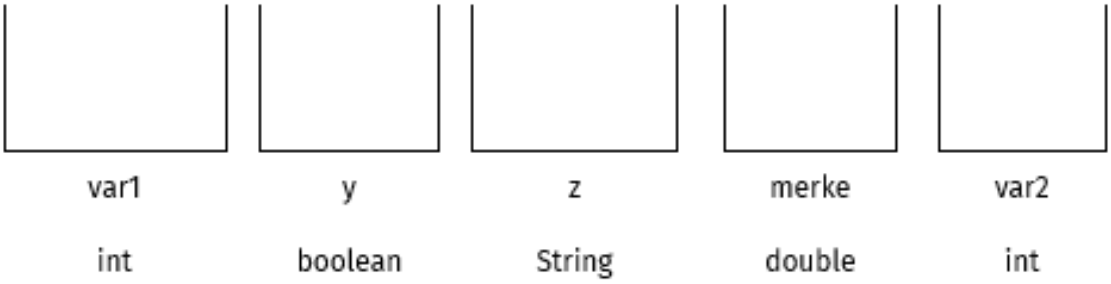

```
var1 = 4;
y = (3 == 3);
z = "Lina";
merke = 7.2;
```



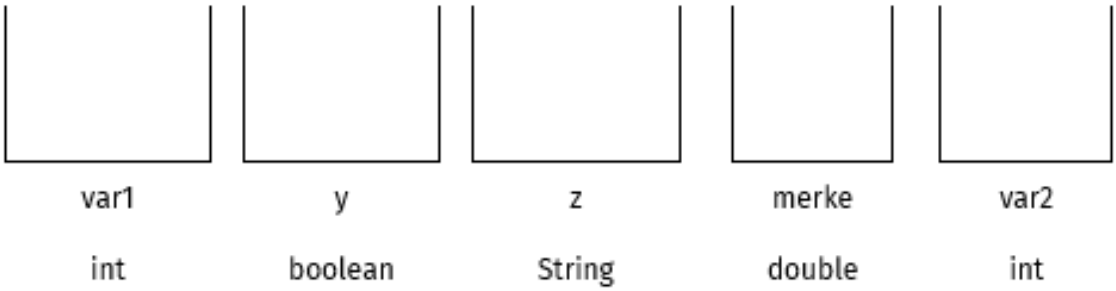

```
var1 = var2;
y = y && true;
z = "Max & " + z;
var2 = var2 + 11;
```



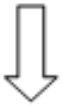
```
merke = merke + var1 + var2;
var2 = var2 * (var1-5);
y = (!y) && (var1 != var2)
```

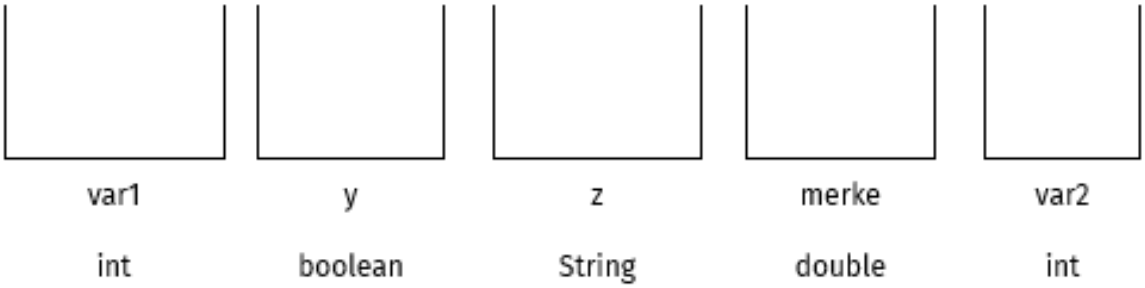



```
var1 = 4;
y = !(var2 == (6*6));
z = Kon.readString();    Eingabe: Lone
merke = merke - 0.2;
```




```
var1=Kon.readInt();3;Eingabe: 40
merke=Kon.readDouble(); Eingabe: 40
```





Aufgabe 3 – Name einlesen und ausgeben

Erweitere dein Programm aus Aufgabe 1, so dass der eingegebene Name in einer Variablen gespeichert wird und dann am Bildschirm wieder ausgegeben wird.

Aufgabe 4 – Alter ausrechnen

Schreibe ein Programm, das den Namen, das aktuelle Jahr und das Geburtsjahr des Benutzers einliest. Dann soll das Alter des Benutzers ausgegeben werden. Dazu muss noch in Erfahrung gebracht werden, ob der Benutzer in diesem Jahr schon Geburtstag hatte.

Der Dialog, der bei der Ausführung des Programms entsteht, könnte zum Beispiel so aussehen:

Hallo, wie heißt du?	Mia
Welches Jahr haben wir, Mia?	Jahr: 2005
In welchem Jahr bist du geboren?	Jahr: 2001
Hattest du dieses Jahr schon Geburtstag?	Fuer ja gebe 0, fuer nein 1 ein: 0

Dann bist du 4 Jahre alt.

Aufgabe 5 – Spritkostenberechnung

Schreibe ein Programm zur Spritkostenberechnung!

Das Programm liest

- den aktuellen Spritpreis,
- den durchschnittlichen Spritverbrauch des Autos pro 100 km
- und die Anzahl der zu fahrenden Kilometer ein.

Dann gibt das Programm aus, wie teuer die Fahrt wird.

Was würde es kosten, mit dem Auto deiner Eltern oder eines anderen Bekannten von Aachen nach

- Madrid
- Istanbul
- Peking
- Köln

zu fahren?

Finde dazu im Internet (oder anders)

- die aktuellen Spritpreise,

  _____

- wie viel Liter Benzin das Auto deiner Eltern bzw. des Bekannten pro 100 km verbraucht

  _____

  und
- die entsprechenden Entfernungen heraus (Luftlinie ist nicht realistisch).

  _____

LERNKONTROLLE

Du solltest für die Lernkontrolle mit Variablen umgehen können. Das heißt, dass du sie anlegen und benutzen können solltest. Außerdem musst du in der Lage sein, ein interaktives Programm zu schreiben. Das äußere Gerüst des Programms wird dir allerdings vorgegeben.

Aufgabe 1 – Vorbereitung

```
System.out.print();


public class Name {
  public static void main (String [] arguments){
    String name;
    System.out.print("Wie heisst du? ");
    name = Kon.readString();
  }
}
```

Aufgabe 2 – Umgang mit Variablen

100	true	"Leo"	1.9	7
var1	y	z	merke	var2
int	boolean	String	double	Int

```
var1 = 4;
y = (3 == 3);
z = "Lina";
merke = 7.2;
```

4	true	"Lina"	7.2	7
Var1	y	z	merke	var2
Int	boolean	String	double	Int

```
var1 = var2;
y = y && true;
z = "Max & " + z;
var2 = var2 + 11;
```

7	true	"Max & Lina"	7.2	18
var1	y	z	merke	var2
Int	boolean	String	double	Int

```
merke = merke + var1 + var2;
var2 = var2 * (var1-5);
y = (!y) && (var1 != var2)
```

var1	y	z	merke	var2
7	false	"Max & Lina"	32.2	36
Int	boolean	String	double	Int

```
var1 = 4;
y = !(var2 == (6*6));
z = Kon.readString();    Eingabe: Lone
merke = merke - 0.2;
```

Var1	y	z	merke	var2
4	false	"Lone"	32.0	36
Int	boolean	String	double	Int

```
var1=Kon.readInt()+3;Eingabe: 40
merke = Kon.readDouble(); Eingabe: 40
```

var1	y	z	merke	var2
43	false	"Lone"	40.0	36
Int	boolean	String	double	Int

Aufgabe 3 – Name einlesen und ausgeben

```
public class Name {
  public static void main (String [] arguments){
    String name;

    System.out.print("Wie heisst du? ");
    name = Kon.readString();
    System.out.print("Hallo ");
    System.out.print(name);
    System.out.print("!");

  }
}
```

Aufgabe 4 – Alter ausrechnen

```
public class Dialog {
  public static void main (String [] arguments){
    String name;
    int jahr;
    int gebjahr;
    int abzug, alter;
    System.out.print("Hallo, wie heisst du? \n");
    name = Kon.readString();
    System.out.print("Welches Jahr haben wir, ");
    System.out.print(name);
    System.out.print("? \n");
    jahr = Kon.readInt();
    System.out.print("In welchem Jahr bist du geboren, ");
    System.out.print(name);
    System.out.print("? \n");
    System.out.print("Geburtsjahr: ");
    gebjahr = Kon.readInt();
    System.out.println("Dieses Jahr schon Geburtstag gehabt?“);
    System.out.print("Fuer ja tippe 0, fuer nein 1 ein: ");
    abzug = Kon.readInt();;
    alter = jahr - gebjahr - abzug;
    System.out.print("Dann bist du ");
    System.out.print(alter);
    System.out.print(" Jahre alt.");
  }
}
```

Aufgabe 5 – Spritkostenberechnung

Beispiel: VW Golf FSI mit 102 PS

Aktueller Spritpreis: 1,44 Super

Verbrauch des Autos: ca. 7l (Herstellerangaben Mittel außer- und innerorts)

Kosten Aachen (nicht Luftlinie)

- Madrid: 198,6 €
- Istanbul: 286,50 €
- Peking: 980,46 €
- Köln: 8,06 €

Programmvorschlag:

```
public class Spritverbrauch {
  public static void main (String [] arguments){
    double preis, strecke, kosten, verbrauch ;
    System.out.println("*****Spritkostenrechner*******");
    System.out.print("Spritpreis pro Liter in Cent: ");
    preis = Kon.readDouble();
    preis = preis/100;
    System.out.print("Durchschn. Verbrauch in L pro 100 km %: ");
    verbrauch = Kon.readDouble();
    System.out.print("Strecke in km: ");
    strecke = Kon.readDouble();
    kosten = preis * verbrauch/100 * strecke;
    System.out.print("Kosten: ");
    System.out.print(kosten);
    System.out.print(" Euro ");
  }
}
```

KAPITEL 5: VERZWEIGUNGEN

ÜBERSICHT

Eine Verzweigung ist eine Kontrollstruktur. Was ist eine Kontrollstruktur? Du hast gelernt, dass ein Programm Zeile für Zeile abgearbeitet wird. Es gibt allerdings Anweisungen, die einen Sprung zu einem anderen Teil des Programms oder die Wiederholung eines Programmteils bewirken. Diese Anweisungen nennt man Kontrollstrukturen. In diesem Kapitel wirst du dich mit Verzweigungen befassen. Verzweigung bedeutet, dass aus einer Auswahl an Möglichkeiten im Programm fortzufahren eine ausgewählt wird. Verzeigungen kann man mit if- und switch-Anweisungen bewirken.

LERNZIEL

In diesem Kapitel lernst du, mit Verzweigungen umzugehen. Du wirst den Umgang mit der If-Anweisung und der Switch-Anweisung erlernen.

Außerdem wirst du lernen, deine Programme durch Kommentare und Einrückungen übersichtlich zu gestalten.

AUFGABEN

Aufgabe 1

```
if (zahl < 0) {
  System.out.print("Zahl ist negativ");
} else {
  System.out.print("Zahl ist nicht negativ");
}
```

Was passiert in diesem Programmausschnitt? Übersetze in Umgangssprache:

Schreibe ein Programm, das für eine Zahl, die über die Tastatur eingelesen wird, ausgibt, ob die Zahl negativ oder nicht negativ ist!

THEORIE

In Aufgabe 1 hast du die Bedeutung der if-Anweisung herausgefunden.

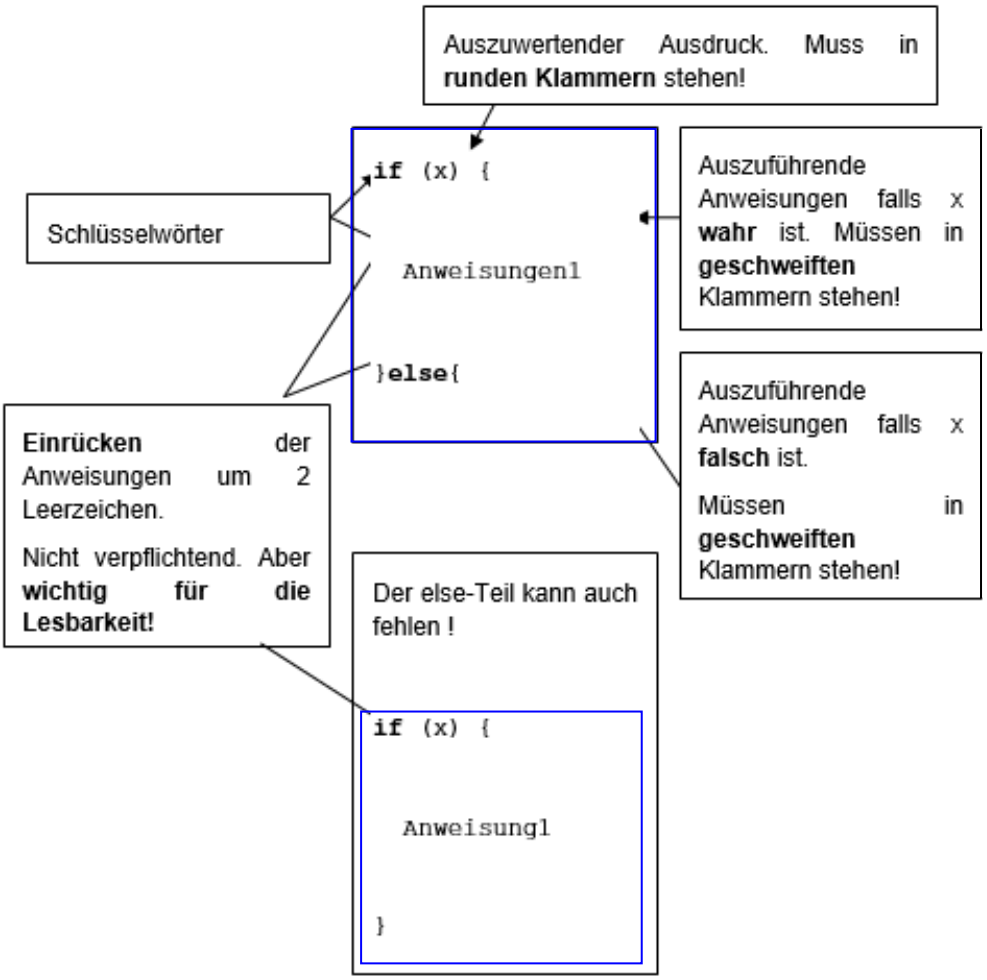
Hier noch einmal eine zusammenfassende Übersicht:

Programmcode	**Bedeutung**

if (x) {
  Anweisung1
} else {
  Anweisung2
}
	Falls der Ausdruck x wahr ist,
	führe Anweisung1 aus
	sonst
	Anweisung2.

Die Regeln, wie man Anweisungen schreiben muss, damit der Computer sie verstehen kann, nennt man **Syntaxregeln**. Die richtige Schreibweise bezeichnet man deshalb auch als die richtige **Syntax**.

Bei der **Syntax** der if-Anweisung (blau umrahmt) muss auf einiges genau geachtet werden:



Aufgabe 2

Schreibe ein Programm für einen Bankautomaten! Der Kunde, der sich gerade am Bankautomaten mit seiner PIN legitimiert hat, hat ein Guthaben von 136.34 €. Das Konto kann nicht überzogen werden. Schreibe ein Programm, das einliest, wie viel Geld dieser Kunde abheben möchte, und nur dann Geld ausgibt, wenn dieser Betrag das Guthaben nicht übersteigt.

THEORIE

Da eine If-Anweisung natürlich auch eine Anweisung ist, kannst du If–Anweisungen auch ineinander schachteln. Du kannst also eine If-Anweisung auch als Anweisung in eine andere If-Anweisung schreiben.

Das kann zum Beispiel so aussehen:

```
public class Betrag{
  public static void main (String [] arguments){
    // Bildet den Bertag einer Zahl
    double x, betrag;
    x = Kon.readDouble();
    if (x < 0) {      //Wenn x < 0 ist, wird -x in betrag
                      //gespeichert
      betrag=-x;
    }else{            //Sonst..
      if (x > 0) {    //..ist entweder x > 0,
        betrag=x;     //dann wird x in betrag gespeichert
      }else{          //..oder x = 0,
        betrag=0;     //dann wird 0 in betrag gespeichert
      }
    }
    System.out.print(betrag);
  }
}
```

Dir ist sicherlich schon aufgefallen, dass die Quelltexte in diesem Leitprogramm auf bestimmte Art und Weise editiert sind. Einrückungen und Kommentare werden eingesetzt, um die Quelltexte besser lesbar zu machen. Besonders bei Schachtelungen sollte man darauf achten, Einrückungen zu machen. Eine Einrückung besteht immer aus zwei Leerzeichen. Nutze nicht die Tabulatortaste!

In Zukunft werden Schachtelungen immer häufiger vorkommen. Das obige Beispiel ist trotz Einrückungen und Kommentaren schon etwas unübersichtlich. Aber ohne diese sähe es so aus:

```
public class Betrag{
public static void main (String [] arguments){double x, betrag; x = Kon.readDouble(); if (x < 0) {betrag=-x;}else{if (x > 0) {betrag=x;}else{betrag=0;}} System.out.print(betrag); }}
```

So ist das Programm kaum noch nachzuvollziehen, oder?

Versuche also in Zukunft, deine Quellcodes durch Einrückungen und Kommentare gut lesbar zu gestalten! Der *JavaEditor* gibt dabei eine automatische Hilfestellung. Orientiere dich außerdem an den Quellcodes im Leitprogramm.

Aufgabe 3

Erweitere dein Programm aus Aufgabe 1 so, dass für die eingegebene Zahl entschieden wird, ob sie

- negativ,
- gleich Null oder
- positiv ist.

Tipp: Orientiere dich an dem Programm *Betrag.java* aus dem Theorieteil.

Sorge durch Einrückungen und Kommentare für eine gute Lesbarkeit deines Programms!

Aufgabe 4

Schreibe ein Programm, das anhand des aktuellen Datums und des Geburtsdatums des Benutzers das Alter des Benutzers ausgibt! Wenn der Benutzer am aktuellen Datum Geburtstag hat, wird zusätzlich eine Gratulation ausgegeben. Das aktuelle und das Geburtsdatum werden über die Tastatur während der Ausführung eingelesen.

Sorge durch Einrückungen und Kommentare für eine gute Lesbarkeit deines Programms!

Beispiel für Programmablauf:

*****Geburtstagsrechner*****

aktuelles Datum

Tag: 13

Monat: 9

Jahr: 2020

Geburtsdatum

Tag: 21

Monat: 9

Jahr: 1996

Du bist 23 Jahre alt.

THEORIE

Nun wirst du noch eine weitere Kontrollstruktur kennenlernen: die Switch-Anweisung. Sie ist der if-Anweisung ähnlich.

Programmcode	**Bedeutung**

`switch (x){`
`  case Wert1:{`
`  Anweisung1`
`  break;`
`  }`
`  case Wert2: {`
`  Anweisung2`
`  break;`
`  }`
`  ...`
`  Default : {`
`  letzteAnweisung`
`  }`
`}`
	Falls x
	gleich Wert1 ist,
	dann führe Anweisung1 aus.
	
	Falls x gleich Wert2 ist,
	dann führe Anweisung2 aus.
	usw.
	
	In allen anderen Fällen führe letzteAnweisung aus.

Die Werte, die jeweils nach **case** stehen, dürfen nur vom Datentyp **int** oder **char** sein. Außerdem müssen dort direkt Werte eingesetzt werden. Variablen sind nicht zulässig.

Genau wie bei der Verwendung der if-Anweisungen muss hier ganz genau auf die **Syntax** und zur Übersichtlichkeit auf Einrückungen geachtet werden.

Beispiel zur Verwendung von Switch:

```
// Je nachdem, welche Zahl zwischen 1 und 3 in zahl
// gespeichert ist, wird eine Übersetzung auf Deutsch,
// Englisch und Italienisch ausgegeben.


    switch (zahl) {
      case 1 :{
        System.out.println("1 : Eins, one, uno"); break;
      }
      case 2 :{
        System.out.println("2 : Zwei, two, due"); break;
      }
      case 3 :{
```

```
        System.out.println("3 : Drei, three, tre"); break;
      }
      default:{
        System.out.println("Nicht in der Datenbank.");
      }
    }
```

SICHERUNGSPHASE

Aufgabe 5

Schreibe einen „Taschenrechner", der die Funktionen +, -, *, / hat.

Das Programm liest

- zuerst die erste Zahl, dann
- den Operator und dann
- die zweite Zahl ein.

Dann wird das Ergebnis ausgegeben.

Sorge durch Einrückungen und Kommentaren für eine gute Lesbarkeit deines Programms!

Diese Aufforderung gilt ab jetzt für alle Programmieraufgaben.

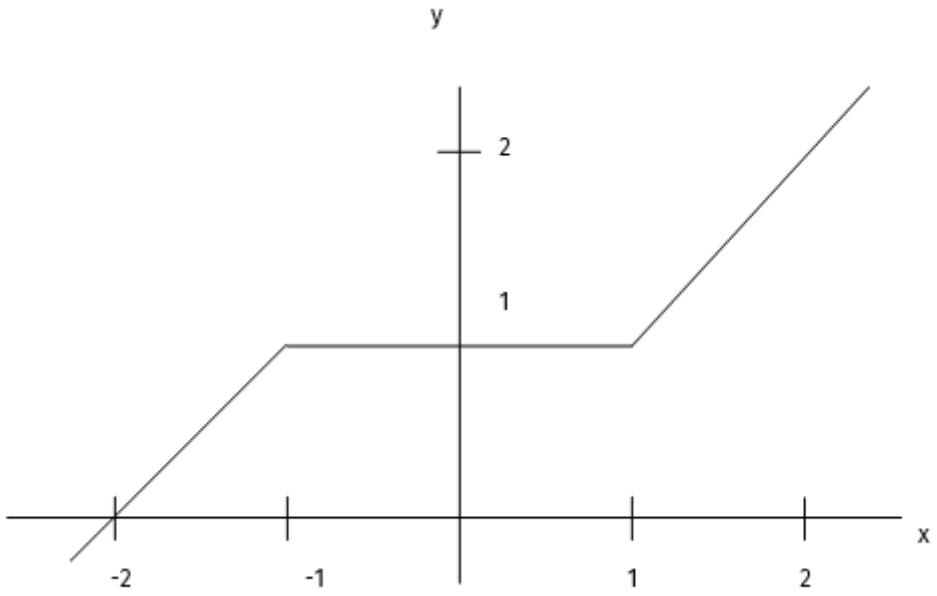
Deshalb wird sie ab jetzt nicht mehr wiederholt.

LERNKONTROLLE

Fühlst du dich sicher im Stoff? Dann kannst du bei deiner Lehrkraft den Kapiteltest machen. Wenn du ihn bestehst, darfst du das nächste Kapitel oder das Additum bearbeiten.

ADDITUM

Schreibe ein Programm für die Funktion, die in der Abbildung beschrieben ist.



MUSTERLÖSUNGEN ZU DEN AUFGABEN AUS KAPITEL 5

Aufgabe 1

```
if (zahl < 0) {
  System.out.print("Diese Zahl ist negativ.");
} else {
  System.out.print("Diese Zahl ist nicht negativ.");
}
```

Was passiert in diesem Programmausschnitt? Übersetze in Umgangssprache:

Wenn zahl kleiner ist als 0, dann gebe "Die Zahl ist negativ." und sonst "Die Zahl ist nicht negativ." am Bildschirm aus.

Aufgabe 2

Schreibe ein Programm für einen Bankautomaten! Der Kunde, der sich gerade am Bankautomaten mit seiner PIN legitimiert hat, hat ein Guthaben von 136.34 €. Das Konto kann nicht überzogen werden. Schreibe ein Programm, das einliest, wie viel Geld dieser Kunde abheben möchte, und nur dann Geld ausgibt, wenn dieser Betrag das Guthaben nicht übersteigt.

```
public class Bankautomat {
  public static void main (String [] arguments){
    double guthaben, abbuchung;
    guthaben = 136.34;
    System.out.print("Wie viel Geld wollen Sie abheben? ");
    abbuchung = Kon.readDouble(); //einlesen der gewün. Abbuch.
    if (guthaben - abbuchung < 0) { //abbuchung zu hoch
      System.out.print("Keine Auszahlung.")
      System.out.print("Betrag übersteigt Guthaben. ");
    } else { //genug Guthaben für Abbuchung
      System.out.print("Es wird ");
      System.out.print( abbuchung );
      System.out.print(" Euro ausgezahlt. ");
    }
  }
}
```

Aufgabe 3

```
public class Zahlentest2 {
  public static void main (String [] arguments){
    double zahl;
    System.out.print("Geben Sie eine ganze Zahl ein? ");
    zahl = Kon.readDouble(); //Einlesen einer Zahl
    if (zahl < 0) {
      System.out.print("Diese Zahl ist negativ.");
    } else {    //zahl ist nicht negativ
      if (zahl==0){
        System.out.print("Diese Zahl ist gleich 0.");
      } else {
        System.out.print("Diese Zahl ist positiv.");
      }
    }
  }
}
```

Aufgabe 4

Schreibe ein Programm, das anhand des aktuellen Datums und des Geburtsdatums des Benutzers das Alter des Benutzers ausgibt! Wenn der Benutzer am aktuellen Datum Geburtstag hat, wird zusätzlich eine Gratulation ausgegeben. Das aktuelle und das Geburtsdatum werden über die Tastatur während der Ausführung eingelesen.

```
public class Alter {
  public static void main (String [] arguments){
    String name;
    int datJahr, datMonat, datTag;
    int gebJahr, gebMonat, gebTag;
    int alter;
    //Einlesen des aktuellen Datums
    System.out.println("Welches Datum haben wir?");
    System.out.print("Tag : ");
    datTag = Kon.readInt();
    System.out.print("Monat : ");
    datMonat = Kon.readInt();
    System.out.print("Jahr : ");
    datJahr = Kon.readInt();
    //Einlesen des Geburtsdatums
    System.out.println("Wann bist du geboren?");
    System.out.print("Tag : ");
    gebTag = Kon.readInt();
    System.out.print("Monat : ");
    gebMonat = Kon.readInt();
    System.out.print("Jahr : ");
    gebJahr = Kon.readInt();
    // Um das Alter zu bestimmen, muss man das Geburtsjahr
    // vom aktuellen Jahr
    // abziehen und dann...
    alter = datJahr - gebJahr;
    // ..überprüfen, ob der Benutzer in diesem Jahr
    //schon Geburtstag hatte.
    if (gebMonat < datMonat){              //Geburtstag war schon
      System.out.print("Du bist ");
```

```
        System.out.print(alter);
        System.out.print(" Jahre alt.");
      } else {
        if (gebMonat == datMonat){   //Geburtstag in diesem Monat
          if (gebTag < datTag){   //Geburtstag war schon in diesem Monat
            System.out.print("Du bist ");
            System.out.print(alter);
            System.out.print(" Jahre alt.");
          } else {                //Geburtstag kommt noch in diesem Monat
            if (gebTag == datTag){       //Geburtstag heute
              System.out.println("HAPPY BIRTHDAY!");
              System.out.print("Heute bist du ");
              System.out.print(alter);
              System.out.print(" Jahre alt geworden.");
            } else {            //Geb. später (nicht heute) in diesem Monat
              System.out.print("Du bist ");
              System.out.print(alter-1);
              System.out.print(" Jahre alt.");
            }
          }
        } else {  //Geb. kommt noch in diesem Jahr (nicht in diesem Monat)
          System.out.print("Du bist ");
          System.out.print(alter-1);
          System.out.print(" Jahre alt.");
        }
      }
    }
  }
```

Aufgabe 5

Schreibe einen „Taschenrechner“, der die Funktionen +, -, *, / hat.

```
public class Taschenrechner {
  public static void main (String [] arguments){
    double zahl1, zahl2;
    int operator;
    System.out.print("****Taschenrechner****\n");
    zahl1 = Kon.readDouble();  //Der 1. Operand wird eingelesen.
    operator = Kon.readChar(); //Die gew. Operation wird eingelesen.
    switch (operator) {
      case '+' :{  //Operand ist +
        zahl2 = Kon.readDouble();  //Der 2. Operand wird eingelesen.
        System.out.println("=");
        System.out.print(zahl1 + zahl2);
        break;
      }
      case '-' :{  //Operand ist -
        zahl2 = Kon.readDouble();  //Der 2. Operand wird eingelesen.
        System.out.println("=");
        System.out.print(zahl1 - zahl2);
        break;
      }
      case '*' :{ //Operand ist *
        zahl2 = Kon.readDouble(); //Der 2. Operand wird eingelesen.
        System.out.println("=");
        System.out.print(zahl1 * zahl2);
        break;
      }
      case '/' :{ //Operand ist /
        zahl2 = Kon.readDouble(); //Der 2. Operand wird eingelesen.
        System.out.println("=");
        System.out.print(zahl1 / zahl2);
        break;
      }
```

```
      default: {
        System.out.println("FEHLER");
      }
    }
  }
}
```

Additum

Es handelt sich um folgende Funktion:

$$f(x) = \begin{cases} x + 2 & \text{falls } x < -1 \\ 1 & \text{falls } -1 <= x <= 1 \\ x & \text{falls } 1 < x \end{cases}$$

```
public class Funktion {
  public static void main (String [] arguments){
    int x,y;
    System.out.print("x = ");
    x=Kon.readInt();
    if (x < -1) {
      y=x+2;
    } else {
      if (x <= 1) {
        y=1;
      } else {
        y=x;
      }
    }
    System.out.print("f(x) = ");
    System.out.print(y);
  }
}
```

KAPITEL 6: SCHLEIFEN

ÜBERSICHT

Schleifen sind – genau wie Verzweigungen – Kontrollstrukturen. Sie bewirken eine Abweichung von der üblichen Abarbeitung der Anweisungen. Solange keine Kontrollstrukturen vorkommen, werden die Anweisungen in einem Programmquelltext vom Compiler der Reihe nach von oben nach unten durchgearbeitet. Du weißt inzwischen, dass die Kontrollstrukturen if und switch bewirken, dass an einer bestimmten Stelle im Quelltext eine Auswahl zwischen zwei oder mehreren abzuarbeitenden Quelltextteilen getroffen wird. Schleifen bewirken dagegen, dass ein bestimmter Quelltextteil **gar nicht, einmal oder mehrmals wiederholt** wird. In diesem Kapitel wirst du dich mit **for**- und **while**-Schleifen befassen. Bei for-Schleifen handelt es sich um eine fest vorgegebene Anzahl von Wiederholungen. Bei while-Schleifen dagegen wird die Anzahl der Wiederholungen von einer Bedingung abhängig gemacht. Der Quelltext wird solange wiederholt, bis diese Bedingung verletzt, also false, ist.

LERNZIEL

In diesem Kapitel wirst du lernen, wie du die Wiederholung von Programmteilen mit for- und while-Schleifen programmieren und sinnvoll in deine Programme einbauen kannst.

AUFGABEN

Aufgabe 1 – Wie funktioniert eine for-Schleife?

Die folgende Aufgabe ist nicht einfach, wenn du wenige oder keine Vorkenntnisse im Programmieren hast. Du sollst hier herausfinden, wie for-Schleifen funktionieren. Eine Hilfestellung könnte sein, dass du das Programm am Computer austestest. Betrachte das folgende Programm:

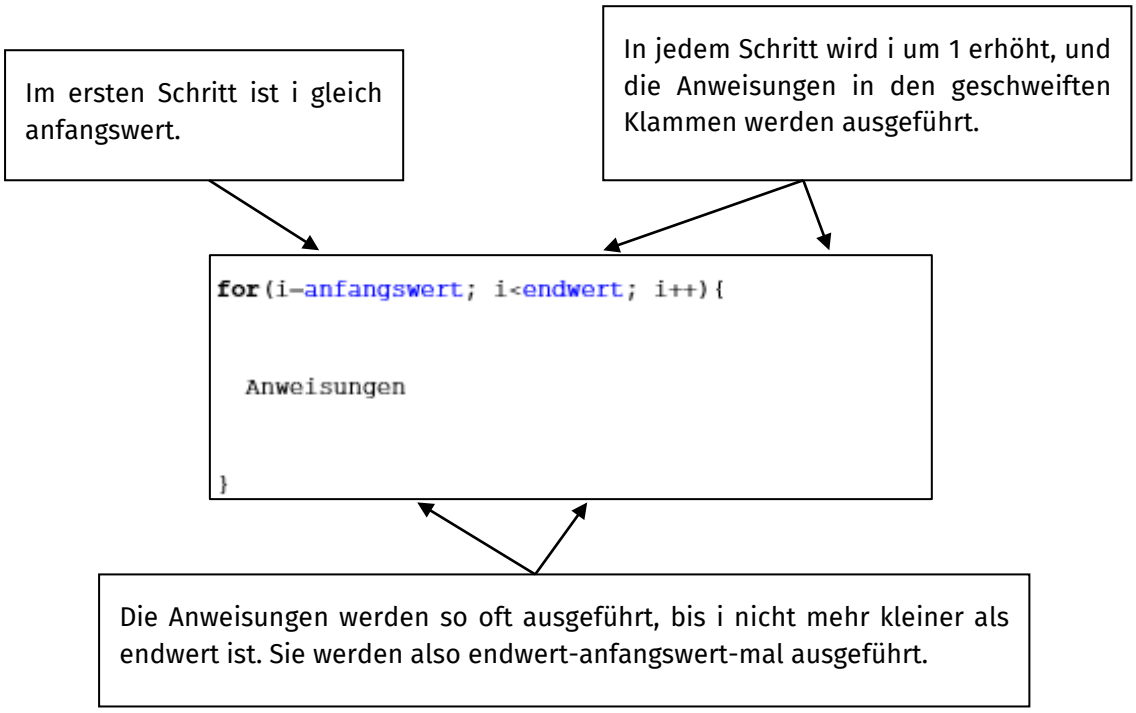
```
public class BedeutungForSchleife {
  public static void main (String [] arguments){
    int i;
    for(i=0; i<10; i++){
      System.out.println(i);
    }
  }
}
```

Übersetze den Quelltext in Umgangssprache!

Tipp: x++ hat dieselbe Bedeutung wie x = x+1.

THEORIE

Wenn du mit Aufgabe 1 Schwierigkeiten hattest, dann hilft dir der folgende Theorieteil. Wie funktioniert eine for-Schleife? Bei der for-Schleife ist fest vorgegeben, wie oft die Anweisungen in den geschweiften Klammern ausgeführt werden. Es gibt eine Zählvariable, die am Anfang auf einen Anfangswert gesetzt wird. Dann wird sie solange hochgezählt, bis sie nicht mehr kleiner als ein vorgegebener Endwert ist. Jedes Mal, wenn die Zählvariable um eins hochgezählt wird, werden auch die Anweisungen in den geschweiften Klammern ausgeführt.



Beispiel: **Das kleine 1 x 3**

```
public class Einmaldrei {
  public static void main (String [] arguments){
    int i,x;
    x=0;
    for(i=0; i<11; i++){
      System.out.print(i);
      System.out.print(" * 3 = ");
      System.out.println(x);
      x = 3 + x;
    }
  }
}
```

Für i=0 bis i=10 wird i je um eins erhöht und das Ergebnis von i*3 ausgegeben. i*3 wird allerdings nicht durch Multiplikation berechnet. Das Ergebnis von i*3 steht in der Variablen x. Das passiert dadurch, dass in jedem Schritt zu der Variablen x die Zahl 3 addiert wird und dieser Wert wiederum in x gespeichert wird. Am Anfang ist x durch 0 initialisiert.

SICHERUNGSPHASE

Aufgabe 2 – for-Schleife benutzen

Schreibe ein Programm, das alle Zahlen von 1 bis 100 aufaddiert. Es soll das Ergebnis jeder Addition ausgegeben werden.

Aufgabe 3 – for-Schleife und if-Anweisung

Schreibe ein Programm, das für alle Zahlen zwischen 1 und 100 testet, ob sie

1. durch 3 teilbar sind.
2. nicht durch 5 aber durch 4 teilbar sind.
3. Primzahlen sind.

Lasse jeweils die Zahlen, auf die die Bedingungen zutreffen, ausgeben.

Du brauchst:

Mit dem Operator % erhält man den Rest der ganzzahligen Division.

5 % 3 wird also ausgewertet zu 2.

Tipp zu 3.:

Schachtelung von zwei for-Schleifen: Eine Zahl i kann solange als Primzahl angesehen werden, bis eine Zahl zwischen 2 und i-1 gefunden wird, die i teilt.

boolean-Variable, die false wird, wenn die aktuell untersuchte Zahl keine Primzahl ist.

Aufgabe 4 – Wie funktioniert eine while-Schleife?

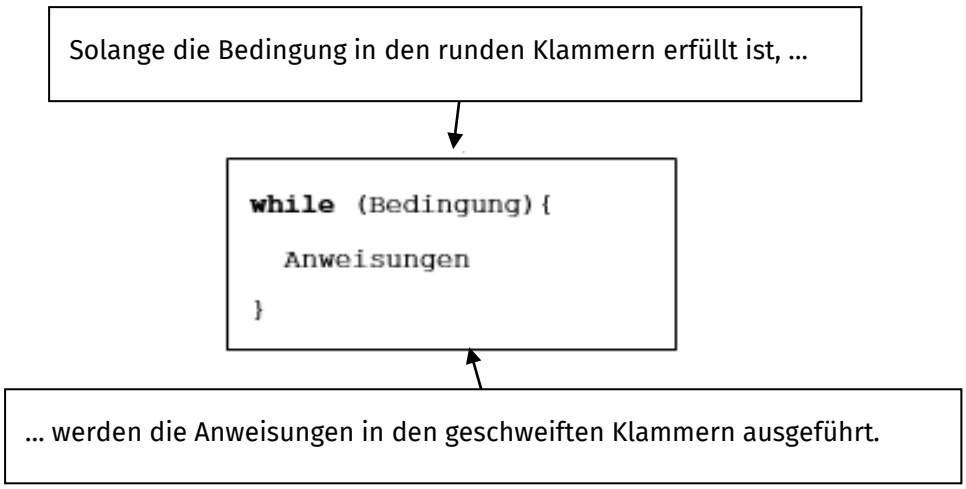
Was passiert in dem folgenden Programm? Bei der Beantwortung der Frage könnte es helfen, das Programm am Computer auszutesten. Betrachte das folgende Programm:

```
public class WhileSchleife{
  public static void main (String [] arguments){
    int x;
    x = 10;
    while (x > 0){
       System.out.println(x);
       x = x-3;
    }
  }
}
```

Übersetze den Quelltext in Umgangssprache!

THEORIE

Es folgt die Theorie zu Aufgabe 4, um zu klären, wie eine while-Schleife funktioniert.



Wie oft die Anweisungen wiederholt werden, hängt also davon ab, ob die Bedingung noch erfüllt ist. Damit die Wiederholungen abbrechen, muss die Bedingung verletzt werden. Folglich muss die Bedingung von den Anweisungen abhängen.

Beispiel: **Das kleine 1 x 4**

```
public class Einmalvier{
  public static void main (String [] arguments){
    int x;
    x=0;
    while (x <= 40){
      System.out.println(x);
      x = x+4;
    }
  }
}
```

Zuerst wird x auf 0 gesetzt. Solange x kleiner gleich 40 ist, wird erst x ausgegeben und dann 4 + x in x gespeichert. Also werden die Zahlen 0, 4, 8,...,40 ausgegeben.

SICHERUNGSPHASE

Aufgabe 5 – while-Schleife benutzen

Schreibe die Programme aus Aufgabe 2 und 3 um! Ersetze die for-Schleifen durch while-Schleifen.

Hinweis: Benenne die Dateien, in denen die Programmtexte aus Aufgabe 2 und 3 stehen, um, bevor du sie änderst, so dass ihre Quelltexte erhalten bleiben.

Aufgabe 6 – Kopfrechentrainer

Schreibe einen Kopfrechentrainer für die Addition von ganzen Zahlen zwischen -99 und 99.

Das Programm soll folgendermaßen ablaufen:

- Der Trainer stellt eine Additionsaufgabe.
- Der Benutzer gibt seine Lösung ein.
- Wenn die Aufgabe richtig gelöst wurde, stellt der Trainer eine neue Aufgabe.
- Wenn die Aufgabe falsch gelöst wurde, gibt der Trainer eine Fehlermeldung, das richtige Ergebnis und die Anzahl der richtig gelösten Aufgaben aus und beendet sich.

Du brauchst:

Um zufällige Aufgaben generieren zu können, brauchst du Zufallszahlen.

Die Datei (Klasse) Zufall.java stellt die Methode

```
Zufall.getInt();
```

zur Verfügung. Damit wird eine ganze Zufallszahl zwischen -99 und 99 erzeugt. Stelle also sicher, dass die Klasse **Zufall.java** im gleichen Verzeichnis wie dein Programm liegt und kompiliert ist.

LERNKONTROLLE

Für den Kapiteltest solltest du sicher mit for- und while-Schleifen umgehen können. Auch die if-Anweisung musst du beherrschen. Das bedeutet, dass du wissen musst, wie die Syntax dieser Kontrollstrukturen ist und wie man sie anwendet. Wenn du den Kapiteltest bestehst, darfst du das nächste Kapitel bearbeiten.

ADDITUM

Erweitere den Kopfrechentrainer aus Aufgabe 6 um die anderen drei Grundrechenarten. Der Benutzer kann über ein Menü auswählen, welche Rechenart er üben will.

MUSTERLÖSUNGEN ZU DEN AUFGABEN AUS KAPITEL 6

Aufgabe 1

```
for(i=0; i<10; i++){
      System.out.println(i);
}
```

Übersetze den Quelltext in Umgangssprache!

Es wird 10 mal (zwischen i=0 und i=9) der Wert der Variable i in einer Schleife ausgegeben. Bei jedem Schleifendurchlauf wird die Variable i um 1 erhöht.

Aufgabe 2

Schreibe ein Programm, das alle Zahlen von 1 bis 100 aufaddiert. Es soll das Ergebnis jeder Addition ausgegeben werden.

```
public class Aufaddieren {
  public static void main (String [] arguments){
    int i,x;
    x=0;
    for(i=0; i<101; i++){
      x = x + i;
      System.out.println(x);
    }
  }
}
```

Aufgabe 3

Schreibe ein Programm, das für alle Zahlen zwischen 1 und 100 testet,

- ob sie durch 3 teilbar sind.
- nicht durch 5 aber durch 4 teilbar sind.
- Primzahlen sind.

Lasse jeweils die Zahlen, auf die die Bedingungen zutreffen, ausgeben.

```
public class Zahlen{
  public static void main (String [] arguments){
    int i,j;
    boolean primzahl;
```

```
//Alle Zahlen zwischen 1 und 100, die durch 3 teilbar sind
System.out.println("\nAlle Zahlen zwischen 1 und 100,..");
System.out.println("\n\n.. die durch 3 teilbar sind:");
for (i=1; i<101; i++){
  if (i % 3 == 0){
    System.out.print(i);
    System.out.print(" ");
  }
}

//Alle Zahlen zwischen 1 und 100, die nicht durch 5 aber
//durch 4 teilbar sind.
System.out.println("..die nicht durch 5 aber durch 4 teilbar:");
for (i=1; i<101; i++){
  if ((i % 5 != 0)&& (i%4==0)){
    System.out.print(i);
    System.out.print(" ");
  }
}

//Alle Zahlen Primzahlen zwischen 1 und 100
System.out.println("\n\n.., die Primzahlen sind:");

for (i=2; i<101; i++){
  primzahl = true; // i wird als Primzahl angesehen,..
  for (j=2; j<i; j++){
    //..solange keine Zahl zwischen 2 und i-1
    // gefunden wurde, die i teilt.
    if (i % j == 0){
      primzahl = false;
    }
  }
  //Wenn i eine Primzahl ist, dann wird i ausgegeben.
  if (primzahl == true){
    System.out.print(i);
```

```
        System.out.print(" ");
      }
    }

  }
}
```

Aufgabe 4

Was passiert in dem folgenden Programm?

```
x = 10;
  while (x > 0){
     System.out.println(x);
     x = x-3;
  }
```

Übersetze in Umgangssprache!

Solange x größer als 0 ist, wird erst x am Bildschirm ausgegeben und das Ergebnis von x-3 in x gespeichert.

Ausgabe:

```
10
7
4
1
```

Aufgabe 5

Schreibe die Programme aus Aufgabe 2 und 3 um! Ersetze die for-Schleifen durch while-Schleifen.

```
public class AufaddierenWhile{
  public static void main (String [] arguments){
    int i,x;
    i=1;
    x=0;
    //i wird von 1 bis 100 hochgezählt.
    while (i<=100){
      x = x + i; // aufaddieren der Zahlen von 1 bis 100
```

```
      i = i+1;
      System.out.println(x);
    }

  }
}

public class ZahlenWhile{
  public static void main (String [] arguments){
    int i,j;
    boolean primzahl;

    //Alle Zahlen zwischen 1 und 100, die durch 3 teilbar sind.
    System.out.println("\nAlle Zahlen zwischen 1 und 100,..");
    System.out.println("\n\n.. die durch 3 teilbar sind:");
    i = 1;
    while (i<=100){
      if (i % 3 == 0){
        System.out.print(i);
        System.out.print(" ");
      }
      i=i+1;
    }

    //Alle Zahlen zwischen 1 und 100, die nicht durch 5 aber
    //durch 4 teilbar sind.
    System.out.println("..die nicht durch 5 aber durch 4 teilbar:");
    i = 1;
    while (i<=100){
      if ((i % 5 != 0)&& (i%4==0)){
        System.out.print(i);
        System.out.print(" ");
      }
      i=i+1;
    }
```

```
    //Alle Zahlen Primzahlen zwischen 1 und 100
    System.out.println("\n\n.., die Primzahlen sind:");
    i=2;
    while (i<=100){
      j=2;
      primzahl = true; // i wir als Primzahl angesehen,..
      while (j<=i-1){
        //..solange keine Zahl zwischen 2 und i-1 gefunden wurde,
        // die i teilt.
        if (i % j == 0){
          primzahl = false;
        }
        j=j+1;
      }
      //Wenn i eine Primzahl ist, dann wird i ausgegeben.
      if (primzahl == true){
        System.out.print(i);
        System.out.print(" ");
      }
      i=i+1;
    }
  }
}
```

Aufgabe 6

Schreibe einen Kopfrechentrainer für die Addition von ganzen Zahlen zwischen -99 und 99. Das Programm soll folgendermaßen ablaufen:

- Der Trainer stellt eine Additionsaufgabe.
- Der Benutzer gibt seine Lösung ein.
- Wenn die Aufgabe richtig gelöst wurde, stellt der Trainer eine neue Aufgabe.
- Wenn die Aufgabe falsch gelöst ist, gibt der Trainer eine Fehlermeldung, das richtige Ergebnis und die Anzahl der richtig gelösten Aufgaben aus und beendet sich.

```
public class Kopfrechnen{
  public static void main (String [] arguments){
    boolean fehler;
    int zahl1, zahl2, ergebnis, eingabe, anzahl;
```

```
    fehler = false;
    anzahl = 0;

    System.out.println("Kopfrechentrainer : ADDITION\n");
    //Solange der Benutzer keinen Fehler gemacht hat,
    //wird das Programm ausgeführt.
    while (fehler == false){
      //Speichern von zwei Zufallszahlen
      zahl1 = Zufall.getInt();
      zahl2 = Zufall.getInt();
      //Aufgabe stellen
      System.out.print("(");
      System.out.print(zahl1);
      System.out.print(") + (");
      System.out.print(zahl2);
      System.out.print(") = ");
      //Lösung des Benutzers einlesen lassen
      eingabe = Kon.readInt();

      //richtige Lösung errechnen
      ergebnis = zahl1 + zahl2;
      //Wenn die Lösung des Benutzers richtig ist, dann
      if (eingabe != ergebnis){
        //Fehlerausgabe und Anzahl der richtigen Lösungen ausgeben
        System.out.print("FALSCH\n Du hast ");
        System.out.print(anzahl);
        System.out.print(" richtige Loesungen erzielt!");
        // In Variable fehler speichern, dass Fehler aufgetreten ist.
        fehler = true;
      }
      anzahl = anzahl + 1;
    }

  }
}
```

Additum

Erweitere den Kopfrechentrainer aus Aufgabe 6 um die anderen drei Grundrechenarten. Der Benutzer kann über ein wiederkehrendes Menü auswählen, welche Rechenart er üben will.

```
public class Kopfrechnenplus{
  public static void main (String [] arguments){
    boolean fehler;
    int zahl1, zahl2, ergebnis, eingabe, ergebnisrest,
    int eingaberest, anzahl;
    char menuepunkt;

    fehler = false;
    anzahl = 0;
    menuepunkt = ' ';
    while (menuepunkt != 'E'){
      System.out.println("\n\n*** Kopfrechentrainer plus ***");
      System.out.println("(A)ddition     (S)ubtraktion     (M)ultiplikation
(D)ivision  (E)nde");
      menuepunkt = Kon.readChar();
      //Solange der Benutzer keinen Fehler gemacht hat,
      //wird das Programm ausgeführt.
      switch (menuepunkt){

        //Addition
        case 'A':{
           while (fehler == false){
             //Speichern von zwei Zufallszahlen
             zahl1 = Zufall.getInt();
             zahl2 = Zufall.getInt();
             //Aufgabe stellen
             System.out.print("(");
             System.out.print(zahl1);
             System.out.print(") + (");
             System.out.print(zahl2);
             System.out.print(") = ");
```

```
      //Lösung des Benutzers einlesen lassen
      eingabe = Kon.readInt();
      //richtige Lösung errechnen
      ergebnis = zahl1 + zahl2;
      //Wenn die Lösung des Benutzers richtig ist, dann
      if (eingabe != ergebnis){
        //Fehlerausgabe und Anzahl der richtigen Lösungen
        // ausgeben.
        System.out.print("FALSCH\n Du hast ");
        System.out.print(anzahl);
        System.out.println(" richtige Loesungen erzielt!");
        // In Variable fehler speichern, dass Fehler
        // aufgetreten ist.
        fehler = true;
      }
      anzahl = anzahl + 1;
    }
    fehler = false;
    anzahl = 0;
    break;
  }



  //Subtraktion
  case 'S':{
    ( .. analog zur Addition)
  }
  //Multiplikation
  case 'M':{
    ( .. analog zur Addition)
  }
  //Division
  case 'D':{
    while (fehler == false){
      //Speichern von zwei Zufallszahlen
```

```
      zahl1 = Zufall.getInt();
      zahl2 = Zufall.getInt();

      //Aufgabe stellen
      System.out.print("(");
      System.out.print(zahl1);
      System.out.print(") / (");
      System.out.print(zahl2);
      System.out.print(") = ");

      //Lösung des Benutzers einlesen lassen
      eingabe = Kon.readInt();
      System.out.print("Rest : ");
      eingaberest = Kon.readInt();

      //richtige Lösung errechnen
      ergebnis = zahl1 / zahl2;
      ergebnisrest = zahl1 % zahl2;

      //Wenn die Lösung des Benutzers richtig ist, dann
    if ((eingabe != ergebnis) || (eingaberest != ergebnisrest)){
        //Fehlerausgabe und Anzahl der richtigen
        //Lösungen ausgeben
        System.out.print("FALSCH\n Du hast ");
        System.out.print(anzahl);
        System.out.print(" richtige Loesungen erzielt!");
        // In Variable fehler speichern, dass
        // Fehler aufgetreten ist.
       fehler = true;
      }
      anzahl = anzahl + 1;
    }
    fehler = false;
    anzahl = 0;
    break;
  }
```

```
            }
        }
    }
}
```

KAPITEL 7: ARRAYS

ÜBERSICHT

Ein **Array** ist eine **Datenstruktur**. In einer Datenstruktur werden auf eine bestimmte Art und Weise Daten gespeichert. Die Variablen, die du bisher kennengelernt hast, sind also auch Datenstrukturen. In einer Variablen wurde ein Wert von einem bestimmten Datentyp gespeichert. In einem Array dagegen werden **mehrere Daten vom gleichen Datentyp** gespeichert.

Man kann sich ein Array wie eine Liste vorstellen. In einem Array steht also eine Information nach der anderen. Deshalb nennt man Arrays auch Reihungen. Es gibt auch komplexere Arrays, die sogenannten zweidimensionalen Arrays. Diese kann man sich am ehesten als Tabellen vorstellen.

LERNZIEL

In diesem Kapitel lernst du den Umgang mit ein- und zweidimensionalen Arrays. Dazu benutzt du die Kontrollstrukturen und Anweisungen, die du in den vorigen Kapiteln kennengelernt hast.

THEORIE

Du weißt inzwischen schon, dass man in einem Array mehrere Daten vom gleichen Typ speichern kann. Man kann sich ein Array wie im folgenden **Beispiel** vorstellen:

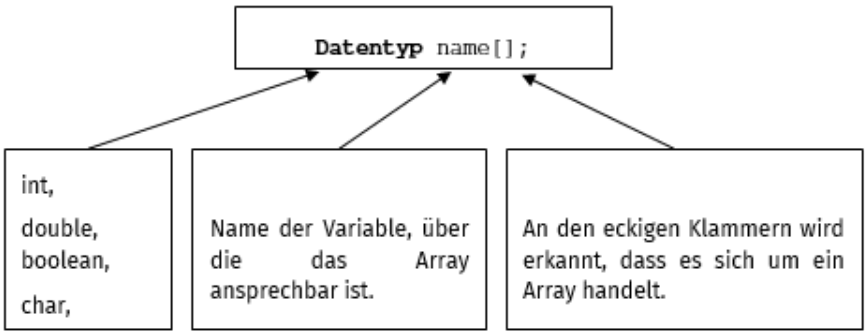
> x ist ein Array für Integerwerte und hat 3 Speicherzellen.
>
> In der Speicherzelle 0 befindet sich eine 12, in der Speicherzelle 1 die Zahl -23 und in der Speicherzelle 2 eine 3.

	x
0	12
1	-23
2	3

Wie realisiert man nun Arrays in einem Programm?

Deklaration eines Arrays

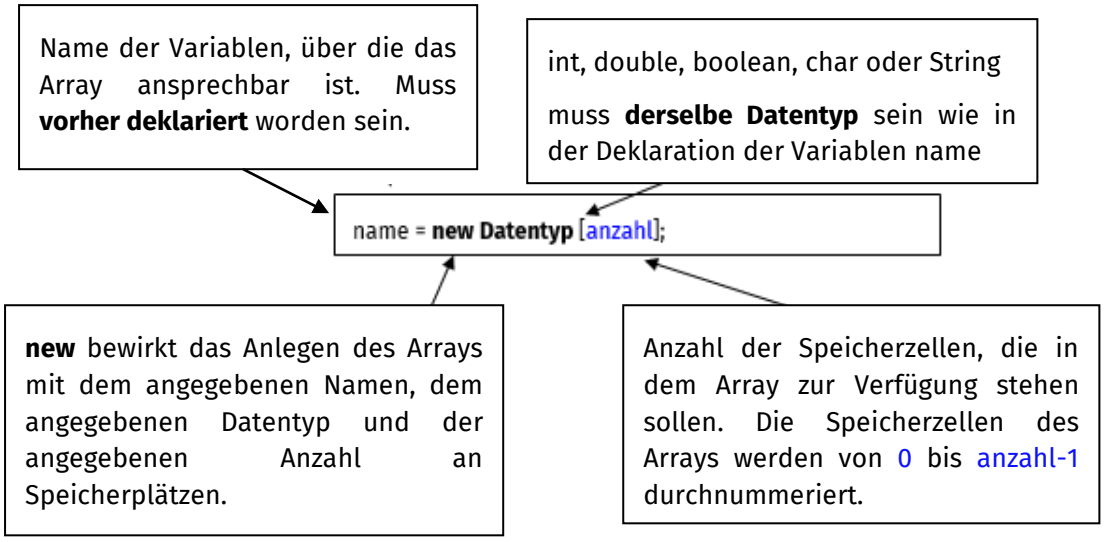
Zunächst muss das Array deklariert werden. In der Deklaration von Arrays wird mitgeteilt, welcher Variablen das Array zugewiesen werden soll und von welchem Datentyp die Werte des Arrays sind.



Anlegen eines Arrays

Nach der Deklaration muss das Array angelegt werden. Das bedeutet, dass Speicherplatz für das Array reserviert werden muss. Damit das passieren kann, muss bekannt sein, wie viele Daten in

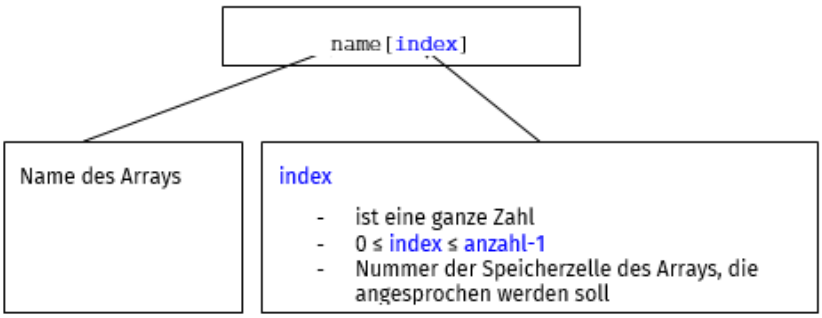
dem Array abgelegt werden sollen. Für jeden Wert, der gespeichert werden soll, wird eine Speicherzelle angelegt. Die Speicherzellen des Arrays werden von 0 an durchnummeriert.



Nachdem das passiert ist, existiert das Array und ist über den Namen, den man ihm zugewiesen hat, ansprechbar. Jede Speicherzelle des Arrays ist bereits mit einem Standardwert belegt worden. Das nennt man **Initialisierung**. Ein Array mit dem Datentyp Integer wird mit dem Wert 0 initialisiert.

Speichern und Auslesen bei Arrays

Jede einzelne Speicherzelle kann mit dem Namen des Arrays und der Nummer der Speicherzelle (hier index) angesprochen werden. Die Werte für index müssen zwischen 0 und anzahl-1 liegen. anzahl ist die Zahl, die beim Anlegen des Arrays für die Anzahl der Speicherzellen angegeben wurde.



name[index] kann jetzt wie eine Variable verwendet werden:

Wert wird im Array name an die Speicherzelle mit der nummer index gespeichert.	`name[index] = Wert;`
Der Wert, der im Array name an der Speicherzelle mit der Nummer index gespeichert ist, wird mit 2 addiert und in der Variablen x gespeichert.	`x = name[index]+2;`

Beispielspielprogramm zu dem Array:

	x
0	12
1	-23
2	3

Deklaration von **x** als Array mit integer-Werten

Anlegen des Arrays **x** als Array mit 3 integer-Werten.

Programm Reihung

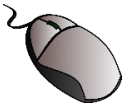
```
public class Reihung {
  public static void main (String [] arguments){

    int x[];

    x = new int [3];

    x[0] = 12;
    x[1] = -23;
    x[2] = 3;
  }
}
```

Belegung des Arrays mit integer-Werten. Die einzelnen Speicherzellen des Arrays können über die Indizes 0, 1 und 2 angesprochen werden.

SICHERUNGSPHASE

Aufgabe 1 – Arrays deklarieren, anlegen und beschreiben

Programmiere die folgenden Arrays!



	x
0	12
1	11
2	-1

	y
0	1.3
1	1.4
2	-12.3
3	2.23

	z
0	'a'
1	'#'

	v
0	"Ar"
1	"ra"
2	"y"

Lasse in deinem Programm folgende Werte am Bildschirm ausgeben:

- Inhalt der Speicherzelle 0 von Array x
- Inhalt der Speicherzelle 3 von Array y
- Inhalt der Speicherzelle 1 von Array z
- Inhalt der Speicherzelle 0 von Array v

Aufgabe 2 – Arrays deklarieren, anlegen und beschreiben

Schreibe ein Programm, in dem du je ein Array für zwei unterschiedliche Datentypen deklarierst, anlegst und mit Werten füllst.

Die beiden Arrays sollen eine unterschiedliche Anzahl an Speicherzellen haben.

Der Inhalt jedes Arrays soll am Bildschirm ausgegeben werden. Formatiere deine Ausgabe wie in dem folgenden **Beispiel**:

doubleArray ist ein array, das 3 double-Werte enthält.

```
  | doubleArray
-------------------
0 | 3.1
1 | 12.0
2 | -3.4
```

Aufgabe 3 – Schleifen und Arrays

Schreibe ein Programm, in dem alle Zahlen von 1 bis 100 in ein geeignetes Array geschrieben werden. Verwende eine Schleife.

Aufgabe 4 – Notenverwaltung

Schreibe ein Programm zur Verwaltung der Noten einer Schulklasse.

Das Programm hat vier Funktionen.

1. Noten der Klausur einlesen lassen, wobei die Klassengröße interaktiv in Erfahrung gebracht wird
2. Notenliste der Klausur ausgeben lassen
3. Durchschnittsnote der Klasse ermitteln lassen
4. Programm beenden

Die Funktionen werden über ein immer wiederkehrendes Menü angesteuert (Tipp: switch und while).

Aufgabe 5 – Standardbelegung der Arrays

Im Theorieteil hast du gelernt, dass die Arrays nach dem Anlegen mit

```
name = new Datentyp [anzahl];
```

schon mit einem Standardwert initialisiert sind. Beim Datentyp int ist das die 0.

Finde heraus, welcher Wert je nach Datentyp für die Standardbelegung gewählt wird! Trage deine Ergebnisse hier ein:

int: 0

double

---

char

String

THEORIE

Die Arrays, die du kennengelernt hast, sind **eindimensionale Arrays**. Jetzt wirst du lernen, **zweidimensionale Arrays** einzusetzen. Zweidimensionale Arrays sind **Tabellen** zum Speichern von Werten des gleichen Datentyps. Schaue dir zunächst das Beispiel an:

- y ist ein zweidimensionales Array für Integerwerte.
- y hat 3 Zeilen und 4 Spalten.
- Die Speicherzelle in der Zeile 1 und der Spalte 2 enthält eine 12.

y

	0	1	2	3
0	1	-9	34	-8
1	6	-6	12	0
2	9	89	0	0

Wie realisiert man zweidimensionale Arrays in einem Programm?

Deklaration eines zweidimensionalen Arrays

```
Datentyp name [] [];
```

An den **zwei** eckigen Klammern wird erkannt, dass es sich um ein zweidimensionales Array handelt.

Anlegen eines zweidimensionalen Arrays

```
name = new Datentyp [zeilenanzahl] [spaltenanzahl];
```

Anzahl der Zeilen, die in dem Array zur Verfügung stehen sollen. Die Zeilen des Arrays werden von 0 bis zeilenanzahl-1 durchnummeriert.

Anzahl der Spalten, die in dem Array zur Verfügung stehen sollen. Die Spalten des Arrays werden von 0 bis spaltenanzahl-1 durchnummeriert.

Speichern und Auslesen bei zweidimensionalen Arrays

Jede einzelne Speicherzelle kann mit dem Namen des Arrays, der Nummer seiner Spalte und der Nummer seiner Zeile angesprochen werden.

```
name[zeile][spalte]
```

zeile
- ist eine ganze Zahl
- 0 ≤ zeile ≤ spaltenanzahl-1
- Nummer der Zeile des Arrays, die angesprochen werden soll.

spalte
- ist eine ganze Zahl
- 0 ≤ spalte ≤ spaltenanzahl-1
- Nummer der Spalte des Arrays, die angesprochen werden soll.

Beispiel zu dem oben angegebenen Array

y

	0	1	2	3
0	1	-9	34	-8
1	6	-6	12	0
2	9	89	0	0

Programm ZweidimensionalesArray

```
public class ZweidimensionalesArray{
  public static void main (String [] arguments){

    //Deklaration von y als zweidimensionales
    //int-Array:
    int y[][];

    //Anlegen von y mit 3 Zeilen und 4 Spalten
    y = new int [3][4];

    // Speichern der Werte in Zeile 0
    y[0][0] = 1;
    y[0][1] = -9;
    y[0][2] = 34;
    y[0][3] = -8;

    // Speichern der Werte in Zeile 1
    y[1][0] = 6;
    y[1][1] = -6;
    y[1][2] = 12;
    y[1][3] = 0;

    // Speichern der Werte in Zeile 2
    y[2][0] = 9;
    y[2][1] = 89;
    y[2][2] = 0;
    y[2][3] = 0;
  }
}
```

SICHERUNGSPHASE

Aufgabe 6 – Stundenplanmanager

Schreibe einen Stundenplanmanager! Bearbeite die folgenden Schritte alle im gleichen Programmquellcode der Reihe nach. Gehe immer erst zum nächsten Schritt über, wenn der aktuelle gemeistert ist.

1. Lege ein zweidimensionales Array an, das sich dazu eignet, einen Stundenplan zu speichern (z. B. 5 Spalten, 6 Zeilen).
2. Programmiere eine entsprechende Ausgabe für den Stundenplan (also für das in 1 angelegte Array).

   TIPP: \t innerhalb eines Strings bewirkt bei der Ausgabe eine Einrückung wie bei der Nutzung der Tabulatortaste.

   Beispiel:

```
    Aktueller Stundenplan

Mo      Di      Mi      Do      Fr

-----------------------------

M       If      M       E       Ph

M       Sp      M       E       Ph

E       D       Sp      M       E

E       D       Sp      F       E

Ph      F       K       If      K

        F               If      K
```

3. Erweitere dein Programm um die Möglichkeit Eintragungen in den Stundenplan vornehmen zu lassen.

   Beispiel:

```
 ** Stunden eintragen **

Tage 1=Mo   2=Di   3=Mi   4=Do oder 5=Fr

Stunden 1 - 6

Welche Stunde soll eingetragen werden?

Tag: 2
```

4. Erweitere dein Programm so, dass man die Möglichkeit hat, in einem Menü aus den folgenden drei Punkten auszuwählen:
   a. Stundenplan ausgeben
   b. Stunde eintragen
   c. Programm beenden

   Nach Ausführung des entsprechenden Menüpunktes wird zum Menü zurückgekehrt.

   TIPP: switch und while!

Beispiel:

```
****** Stundenplanmanager ******
(1) Stundenplan ausgeben
(2) Stunde eintragen
(3) Programm beenden
Bitte waehle einen Menuepunkt: 2


** Stunden eintragen **
Tage 1=Mo   2=Di   3=Mi   4=Do oder 5=Fr
Stunden 1 - 6
Welche Stunde soll eingetragen werden?
Tag: 2
Stunde: 1
Fach: If
```

LERNKONTROLLE

Um den Kapiteltest zu bestehen, musst du den Umgang mit ein- und zweidimensionalen Arrays beherrschen.

ADDITUM – NOTENVERWALTUNG PLUS

Erweitere dein Programm aus Aufgabe 4 so, dass die Noten mehrerer Klausuren eingelesen werden können.

Die Anzahl der Notenlisten, die eingegeben werden sollen, wird vorher interaktiv erfragt. Es steht die Möglichkeit zur Verfügung, für jeden Schüler eine Durchschnittsnote seiner bisher erzielten Noten zu ermitteln.

ADDITUM – MATRIZEN

Man kann ein zweidimensionales Array auch als Matrix auffassen. Wenn du nicht weißt, was eine Matrix ist, dann schaue im Internet nach.

1. Finde im Internet heraus, wie zwei Matrizen multipliziert werden.

2. Multipliziere!

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 0 \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

3. Schreibe ein Programm, das
- zwei Matrizen als zweidimensionale Arrays speichert,
- multipliziert und
- das Ergebnis in einem weiteren zweidimensionalen Array speichert.

4. Überprüfe deine Lösung aus 2. mit dem Programm.

Aufgabe 1

Programmiere die folgenden Arrays!

	x		y		z		v
0	12	0	1.3	0	'a'	0	"Ar"
1	11	1	1.4	1	'#'	1	"ra"
2	-1	2	-12.3			2	"y"
		3	2.23				

Lasse in deinem Programm folgende Werte am Bildschirm ausgeben:

- Inhalt der Speicherzelle 0 von Array x
- Inhalt der Speicherzelle 3 von Array y
- Inhalt der Speicherzelle 1 von Array z
- Inhalt der Speicherzelle 0 von Array v

```
public class mehrereArrays {
  public static void main (String [] arguments){
    // Deklaration der Arrays
    int x[];
    double y[];
    char z[];
    String v[];

    // Anlegen der Arrays
    x = new int [3];
    y = new double [4];
    z = new char [2];
    v = new String [3];

    // Schreiben von Werten in das Array x
    x[0] = 12;
    x[1] = 11;
    x[2] = -1;

    // Schreiben von Werten in das Array y
    y[0] = 1.3;
```

```
    y[1] = 1.4;
    y[2] = -12.3;
    y[3] = 2.23;

    // Schreiben von Werten in das Array z
    z[0] = 'a';
    z[1] = '#';

    // Schreiben von Werten in das Array v
    v[0] = "Ar";
    v[1] = "ra";
    v[2] = "y";

    // Ausgabe der verlangten Werte
    System.out.print("x[0] = "); System.out.println(x[0]);
    System.out.print("y[3] = "); System.out.println(y[2]);
    System.out.print("z[1] = "); System.out.println(z[1]);
    System.out.print("v[0] = "); System.out.println(v[0]);

  }
}
```

Aufgabe 2

Schreibe ein Programm, in dem du je ein Array für zwei unterschiedliche Datentypen deklarierst, anlegst und beschreibst.

Die beiden Arrays sollen eine unterschiedliche Anzahl an Speicherzellen haben.

Der Inhalt jedes Arrays soll am Bildschirm ausgegeben werden.

```
public class Arrays {
  public static void main (String [] arguments){
    // Deklaration der Arrays
    double doubleArray[];
    String stringArray[];

    // Anlegen der Arrays
    doubleArray = new double [3];
```

```
    stringArray = new String [4];

    // Schreiben von Werten in das Array doubleArray
    doubleArray[0] = 12.2;
    doubleArray[1] = -12.2;
    doubleArray[2] = 120.0;

    // Schreiben von Werten in das Array StringArray
    stringArray[0] = "Das";
    stringArray[1] = "sind";
    stringArray[2] = "vier";
    stringArray[3] = "Woerter!";

    // Ausgabe der Werte des Arrays stringArray in der
    // verlangten Formatierung
    System.out.println("  |  stringArray");
    System.out.println("-------------------------");
    System.out.print("0 | "); System.out.println(stringArray[0]);
    System.out.print("1 | "); System.out.println(stringArray[1]);
    System.out.print("2 | "); System.out.println(stringArray[2]);
    System.out.print("3 | "); System.out.println(stringArray[3]);

    // Ausgabe der Werte des Arrays doubleArray in der
    // verlangten Formatierung
    System.out.println("");
    System.out.println("  |  doubleArray");
    System.out.println("-------------------------");
    System.out.print("0 | "); System.out.println(doubleArray[0]);
    System.out.print("1 | "); System.out.println(doubleArray[1]);
    System.out.print("2 | "); System.out.println(doubleArray[2]);

  }
}
```

Aufgabe 3

Schreibe ein Programm, in dem alle Zahlen von 1 bis 100 in ein geeignetes Array geschrieben werden. Verwende eine Schleife.

```
public class Array {
  public static void main (String [] arguments){
    int i;
    int x[];
    x = new int [101];

    for(i=1; i<101; i++){
      x[i] = i;
    }
  }
}
```

oder

```
public class Array {
  public static void main (String [] arguments){
    int i;
    int x[];
    x = new int [101];
    i=1;
    while (i<=100) {
      x[i] = i;
      i = i+1;
    }
  }
}
```

Aufgabe 4

Schreibe ein Programm zur Verwaltung der Noten einer Schulklasse.

Das Programm hat 4 Funktionen.

1. Noten der Klausur einlesen lassen, wobei die Klassengröße interaktiv in Erfahrung gebracht wird
2. Notenliste der Klausur ausgeben lassen
3. Durchschnittsnote der Klasse ermitteln lassen
4. Programm beenden.

Die Funktionen werden über ein immer wiederkehrendes Menü angesteuert.

```
public class Noten{
  public static void main (String [] arguments){
    // Notenliste (Array)
    double noten[];
    double erg;
    int anzahl, menuepunkt, i;

    //Variable zur Steuerung der Menueauswahl
    menuepunkt = 0;

    //Anzahl der Schüler. Wird während der Ausführung eingelesen.
    anzahl = 0;

    //Anfangs ist unbekannt, wie viele Schüler mitgeschrieben haben.
    noten = new double [anzahl];

    //Solange nicht "Programm beenden" gewählt, wird Menü ausgeben.
    while (menuepunkt!= 4) {
      System.out.println("\nNOTENVERWALTUNG GK 11 If ");
      System.out.println("(1) Noten eintagen");
      System.out.println("(2) Notenliste ausgeben");
      System.out.println("(3) Durchschnittsnote Kurs ermitteln");
      System.out.println("(4) Programm beenden");
      System.out.print("Bitte waehle einen Menuepunkt: ");

      //Einlesen der Menuepunktauswahl des Benutzers
      menuepunkt = Kon.readInt();
```

```
//Je nach Menüpunktauswahl werden versch. Aktionen ausgeführt.
switch (menuepunkt){

  // Noten einlesen lassen
  case 1:{
    System.out.println("\nNOTEN EINLESEN");
    // Anzahl der Schüler einlesen lassen
    System.out.print("Anzahl Schueler: ");
    anzahl = Kon.readInt();
    // Notenliste für anzahl Schüler anlegen
    noten = new double [anzahl];
    System.out.println("Schueler / Note ");
    System.out.println("----------------");
    for(i=0; i<anzahl; i++){
      System.out.print(" ");
      System.out.print(i);
      System.out.print(" / ");
      noten[i] = Kon.readInt();
    }
    break;
  }

  // Notenliste ausgeben lassen
  case 2:{
    System.out.println("\nNOTENLISTE");
    System.out.println("Schueler\t|    Note ");
    System.out.println("-------------------------------------");
    for(i=0; i<anzahl; i++){
      System.out.print("     ");
      System.out.print(i);
      System.out.print("  \t|    ");
      System.out.println(noten[i]);
    }
    break;
```

```
        }

        //Durchschnittsnote bestimmen
        case 3:{
          erg = 0;
          for(i=0; i<anzahl; i++){
            erg = erg + noten[i];
          }
          erg = erg/anzahl;
          System.out.print("Durchschnittsnote : ");
          System.out.println(erg);
          break;
        }

        //Programm beenden
        case 4:{
          System.out.println("Programm wird beendet.");
          break;
        }

        //Fehlerhafte Eingabe
        default:{
          System.out.println("Fehlerhafte Eingabe!");
        }
      }

    }

  }
}
```

Aufgabe 5

Finde heraus, welcher Wert je nach Datentyp für die Standardbelegung gewählt wird!

int: 0

double: 0.0

char: Leerzeichen

string: null

Anhand dieses Programms lassen sich die Werte, mit denen beim Anlegen der Arrays initialisiert wird, herausfinden.

```
public class Standardbelegung {
  public static void main (String [] arguments){
    // Deklaration der Arrays
    int intArray[];
    char charArray[];
    double doubleArray[];
    String stringArray[];

    // Anlegen der Arrays
    intArray = new int [2];
    doubleArray = new double [2];
    charArray = new char [2];
    stringArray = new String [2];

    // Ausgeben der Werte, die standardmäßig beim Anlegen
    // in die Arrays geschrieben werden
    System.out.print("Initialisierung fuer int-Arrays : |");
    System.out.print(intArray[0]);
    System.out.println("|");
    System.out.print("Initialisierung fuer double-Arrays : |");
    System.out.print(doubleArray[0]);
    System.out.println("|");
    System.out.print("Initialisierung fuer char-Arrays : |");
    System.out.print(charArray[0]);
    System.out.println("|");
    System.out.print("Initialisierung fuer string-Arrays : |");
    System.out.print(stringArray[0]);
    System.out.println("|");
  }
}
```

Aufgabe 6

Schreibe einen Stundenplanmanager! Bearbeite die folgenden Schritte alle im gleichen Programmquellcode der Reihe nach. Gehe immer erst zum nächsten Schritt über, wenn der aktuelle gemeistert ist.

1. Lege ein zweidimensionales Array an, das sich dazu eignet, einen Stundenplan zu speichern (z. B. 5 Spalten, 6 Zeilen).

```
 public class Stundenplan{
  public static void main (String [] arguments){
    String plan[][];

    //Stundenplan wird angelegt, Standardwert für
    //jede Stunde ist null
    plan = new String [7][6];
  }
}
```

2. Programmiere eine entsprechende Ausgabe für den Stundenplan (also für das in 1. angelegte Array).

```
public class Stundenplan{
  public static void main (String [] arguments){
    String plan[][];
    int menuepunkt, tag, stunde, i, j;

    //Stundenplan wird angelegt, Standardwert für jede Stunde ist null
    plan = new String [7][6];

    // Ausgeben des Stundenplans
    System.out.println("\n      Aktueller Stundenplan ");
    System.out.println("Mo\tDi\tMi\tDo\tFr");
    System.out.println("-------------------------------------");
    for(i=1; i<=6; i++){                    //i = Stunde
      for(j=1; j<=5; j++){                  //j = Tag
        System.out.print(plan[i][j]);
        System.out.print("\t");
      }
```

```
      System.out.println("");
    }
  }
}
```

3. Erweitere dein Programm um die Möglichkeit Eintragungen in den Stundenplan vornehmen zu lassen.

```
public class Stundenplan{
  public static void main (String [] arguments){
    String plan[][];
    int menuepunkt, tag, stunde, i, j;

    //Stundenplan wird angelegt, Standardwert für jede Stunde ist null
    plan = new String [7][6];

    // Ausgeben des Stundenplans
    System.out.println("\n      Aktueller Stundenplan ");
    System.out.println("Mo\tDi\tMi\tDo\tFr");
    System.out.println("-------------------------------------");
    for(i=1; i<=6; i++){                    //i = Stunde
      for(j=1; j<=5; j++){                  //j = Tag
        System.out.print(plan[i][j]);
        System.out.print("\t");
      }
      System.out.println("");
    }

     //Eingeben einer Stunde in den Stundenplan
     System.out.println("\n** Stunde eintragen **");
     System.out.println("Tage 1=Mo 2=Di 3=Mi  4=Do oder 5=Fr");
     System.out.println("Stunden 1 - 6");
     System.out.println("Welche Stunde soll eingetragen werden?");
     System.out.print("Tag: ");
     tag = Kon.readInt();
     System.out.print("Stunde: ");
     stunde = Kon.readInt();
```

```
      System.out.println("Fach: ");
      plan[stunde][tag]=Kon.readString();

    }
  }
```

4. Erweitere dein Programm so, dass man die Möglichkeit hat in einem Menü aus den folgenden drei Punkten auszuwählen:
   a. Stundenplan ausgeben
   b. Stunde eintragen
   c. Programm beenden

   Nach Ausführung des entsprechenden Menüpunktes wird zum Menü zurückgekehrt.

```
public class Stundenplan{
  public static void main (String [] arguments){
    String plan[][];
    int menuepunkt, tag, stunde, i, j;

    //Stundenplan wird angelegt, Standardwert für jede Stunde ist null
    plan = new String [7][6];

    //Variable zur Steuerung der Menueauswahl
    menuepunkt = 0;

    //Solange nicht "Programm beenden" gewählt wird Menü ausgeben
    while (menuepunkt!= 3) {
      System.out.println("\n******   Stundenplanmanager  *********");
      System.out.println("(1) Stundenplan ansehen");
      System.out.println("(2) Stunde eintragen");
      System.out.println("(3) Programm beenden");
      System.out.print("Bitte waehle einen Menuepunkt: ");

      //Einlesen der Menuepunktauswahl des Benutzers
      menuepunkt = Kon.readInt();

      //Je nach Menuepunktauswahl werden versch. Aktionen ausgeführt.
      switch (menuepunkt){
```

```
// Ausgeben des Stundenplans
case 1:{
  System.out.println("\n      Aktueller Stundenplan ");
  System.out.println("Mo\tDi\tMi\tDo\tFr");
  System.out.println("-------------------------------------");
  for(i=1; i<=6; i++){                    //i = Stunde
    for(j=1; j<=5; j++){                  //j = Tag
      System.out.print(plan[i][j]);
      System.out.print("\t");
    }
    System.out.println("");
  }
  break;
}

//Eingeben einer Stunde in den Stundenplan
case 2:{
  System.out.println("\n** Stunde eintragen **");
  System.out.println("Tage 1=Mo 2=Di 3=Mi  4=Do oder 5=Fr");
  System.out.println("Stunden 1 - 6");
  System.out.println("Welche Stunde eintragen?");
  System.out.print("Tag: ");
  tag = Kon.readInt();
  System.out.print("Stunde: ");
  stunde = Kon.readInt();
  System.out.println("Fach: ");
  plan[stunde][tag]=Kon.readString();
  break;
}

//Programm beenden
case 3:{
  System.out.println("Programm wird beendet.");
  break;
}
```

```
        //Fehlerhafte Eingabe
        default:{
          System.out.println("Fehlerhafte Eingabe!");
        }
      }
    }

  }
}
```

Additum – Notenverwaltung plus

Erweitere dein Programm aus Aufgabe 4 so, dass die Noten mehrerer Klausuren eingelesen werden können.

Die Anzahl der Notenlisten, die eingegeben werden sollen, wird vorher interaktiv erfragt. Es steht die Möglichkeit zur Verfügung, für jeden Schüler eine Durchschnittsnote seiner bisher erzielten Noten zu ermitteln.

```
public class Notenplus{
  public static void main (String [] arguments){
    // Notenliste (Array)
    double noten[][];
    double erg;
    int schueleranzahl, klausurenanzahl, klausur, schueler,
    int menuepunkt, i;

    //Variable zur Steuerung der Menüauswahl
    menuepunkt = 0;
    System.out.println("\nNOTENVERWALTUNG GK 11 If ");
    System.out.println("Wie viele Schueler umfasst der Kurs? ");
    schueleranzahl = Kon.readInt();
    System.out.println("Wie viele Klausuren werden geschrieben? ");
    klausurenanzahl = Kon.readInt();

    //Anlegen des Arrays pro Klausur eine Zeile, pro Schüler eine
    //Zeile
```

```
noten = new double [klausurenanzahl][schueleranzahl];

//Solange nicht "Programm beenden" gewählt, wird Menü ausgeben
while (menuepunkt!= 5){
  System.out.println("\n(1) Noten eintagen");
  System.out.println("(2) Notenliste ausgeben");
  System.out.println("(3) Durchschnittsnote Kurs ermitteln");
  System.out.println("(4) Durchschnittsnote Schueler ermitteln");
  System.out.println("(5) Programm beenden");
  System.out.print("Bitte waehle einen Menuepunkt: ");

  //Einlesen der Menüpunktauswahl des Benutzers
  menuepunkt = Kon.readInt();

  //Je nach Menuepunktauswahl werden versch. Aktionen ausgeführt
  switch (menuepunkt){

    // Noten einlesen lassen für eine bestimmte Klausur
    case 1:{
      System.out.println("\nNOTEN EINLESEN");
      System.out.print("Klausuren : 1 - ");
      System.out.println(klausurenanzahl);
      System.out.print("Schueler : 0 - ");
      System.out.println(schueleranzahl);
      // Anzahl der Schüler einlesen lassen
      System.out.print("Klausurnr. : ");
      klausur = Kon.readInt()-1;
      System.out.println("Schueler / Note ");
      System.out.println("----------------");
      for(i=0; i<schueleranzahl; i++){
        System.out.print(" ");
        System.out.print(i);
        System.out.print(" / ");
        noten[klausur][i] = Kon.readInt();
      }
```

```
  break;
}

// Notenliste ausgeben lassen für eine bestimmte Klausur
case 2:{
  System.out.println("\nNOTENLISTE");
  System.out.print("Klausuren : 1 - ");
  System.out.println(klausurenanzahl);
  System.out.print("Schueler : 0 - ");
  System.out.println(schueleranzahl);
  System.out.print("Klausurnr. : ");
  klausur = Kon.readInt()-1;
  System.out.println("Schueler\t|    Note ");
  System.out.println("-------------------------------------");
  for(i=0; i<schueleranzahl; i++){
    System.out.print("     ");
    System.out.print(i);
    System.out.print("  \t|    ");
    System.out.println(noten[klausur][i]);
  }
  break;
}

//Durchschnittsnote Kurs bestimmen für eine bestimmte Klausur
case 3:{
  System.out.print("Klausuren : 1 - ");
  System.out.println(klausurenanzahl);
  System.out.print("Schueler : 0 - ");
  System.out.println(schueleranzahl);
  System.out.print("Klausurnr. : ");
  klausur = Kon.readInt()-1;
  erg = 0;
  for(i=0; i<schueleranzahl; i++){
    erg = erg + noten[klausur][i];
  }
```

```
  erg = erg/schueleranzahl;
  System.out.print("Durchschnittsnote Kurs: ");
  System.out.println(erg);
  break;
}

//Durchschnittsnote aller geschrieben Klausuren eines
//Schülers bestimmen
case 4:{
  System.out.print("Klausuren : 1 - ");
  System.out.println(klausurenanzahl);
  System.out.print("Schueler : 0 - ");
  System.out.println(schueleranzahl);
  //Einlesen des gewünschten Schuelers
  System.out.print("Durchschittsnote fuer Schüler: ");
  schueler = Kon.readInt();
  erg = 0;
  //Berechnen der Durchschnittsnote für den Schüler
  for(i=0; i<klausurenanzahl; i++){
    erg = erg + noten[i][schueler];
  }
  erg = erg/klausurenanzahl;
  System.out.print("Durchschnittsnote: ");
  System.out.println(erg);
  break;
}

//Programm beenden
case 5:{
  System.out.println("Programm wird beendet.");
  break;
}

//Fehlerhafte Eingabe
default:{
```

```
            System.out.println("Fehlerhafte Eingabe!");
          }
        }

      }

    }
}
```

Additum – Matrizen

Man kann ein zweidimensionales Array auch als Matrix auffassen. Wenn du nicht weißt, was eine Matrix ist, dann schaue im Internet nach.

1. Finde im Internet heraus, wie zwei Matrizen multipliziert werden.
2. Multipliziere!

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 1 & 2 \\ 3 & 3 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 4 \\ 2 & 1 & 2 \\ 3 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 17 & 16 & 8 \\ 10 & 11 & 10 \\ 9 & 9 & 18 \end{pmatrix}$$

3. Schreibe ein Programm, das
- zwei Matrizen als zweidimensionale Arrays speichert,
- multipliziert und
- das Ergebnis in einem weiteren zweidimensionalen Array speichert.

```
public class Matrixmultiplikation {
  public static void main (String [] arguments){
    // Deklaration einer 3 x 3 Matrix
    int i,j,k;
    int x[][];
    int ergebnis[][];
    // Anlegen der Arrays
    x = new int [3][3];
    ergebnis = new int [3][3];

    // Schreiben von Werten in die Matrix x
    x[0][0] = 1;
    x[0][1] = 2;
```

```
    x[0][2] = 4;
    x[1][0] = 2;
    x[1][1] = 1;
    x[1][2] = 2;
    x[2][0] = 3;
    x[2][1] = 3;
    x[2][2] = 0;

    //Ergebnis der Multiplikation bestimmen
    for (i=0; i<3; i++){
      for (j=0; j<3; j++){
        for (k=0; k<3; k++){
          ergebnis[i][j] = ergebnis[i][j] + x[i][k]*x[k][j];
        }
      }
    }
    //Ergebnis ausgeben
    for (i=0; i<3; i++){
      for (j=0; j<3; j++){
        System.out.print(ergebnis[i][j]);
        System.out.print("\t");
      }
      System.out.print("\n");
    }
  }
}
```

ANHANG A: KAPITELTESTS

A.1 LERNKONTROLLE FÜR DAS ERSTE KAPITEL

Bearbeite die folgenden Aufgaben sorgfältig und ohne Hilfsmittel (z. B. Theorieteil des Leitprogramms)! Lasse sie von deiner Lehrkraft korrigieren. Wenn du nicht mehr als zwei Fehler machst, dann darfst du mit dem nächsten Kapitel beginnen.

1. Wie läuft das Erstellen eines ausführbaren Programms ab? Bringe die einzelnen Schritte in die richtige Reihenfolge, indem du sie richtig nummerierst!

   _____Ausführen des Programms

   _____Kompilieren des Quellcodes

   _____Starten der Entwicklungsumgebung

   _____Speichern des Quellcodes

   _____Editieren des Quellcodes

2. Bilde sinnvolle Paare!

1	Sinn höherer Programmiersprachen	A	Editieren, speichern, kompilieren, ausführen
2	Übersetzen eines Quelltextes in eine höhere Programmiersprache in Anweisungen, die der Computer versteht	B	Ein Computer versteht nur sehr einfache und sehr wenige Anweisungen. Wenn man mit diesen wenigen Anweisungen ein größeres Programm schreiben müsste, bräuchte man sehr lange und der Quelltext würde sehr unübersichtlich.
3	Quellcode	C	Java
4	Ausführen eines Programms	D	Text in einer höheren Programmiersprache verfassen
5	Eine Entwicklungsumgebung	E	Kompilieren eines Quelltextes
6	Was man mit einer Entwicklungsumgebung machen kann	F	Der Computer tut das, was ihm durch den Quelltext des Programms aufgetragen wird.
7	Eine höhere Programmiersprache	G	JavaEditor

Paare:

Zahl	Buchstabe

MUSTERLÖSUNG FÜR DEN TEST ZUM ERSTEN KAPITEL

1. Wie läuft das Erstellen eines ausführbaren Programms ab? Bringe die einzelnen Schritte in die richtige Reihenfolge, indem du sie richtig nummerierst!

   5 Ausführen des Programms

   4 Kompilieren des Quellcodes

   1 Starten der Entwicklungsumgebung

   3 Speichern des Quellcodes

   2 Editieren des Quellcodes

2. Bilde sinnvolle Paare!
   (1,B) (2,E) (3,D) (4,F) (5,G) (6,A) (7,C)

A2 LERNKONTROLLE FÜR DAS ZWEITE KAPITEL

Bearbeite die folgenden Aufgaben sorgfältig und ohne Hilfsmittel (z. B. Theorieteil des Leitprogramms)! Lasse sie von deiner Lehrkraft korrigieren. Wenn du nicht mehr als zwei Fehler machst, dann darfst du mit dem nächsten Kapitel beginnen.

1. Worum handelt es sich bei den angegebenen Quelltextausschnitten?

```
int x;
int y;
double z ;
```

a)

```
static double umfang(double r){
    return r * 2.0 * 3.14159;
  }
```

b)

```
x = 100 + 4 * 3 / 4;
y = 12345;
```

c)

```
umfang(r);
```

d)

//Berechnung des Umfangs

e)

```
public static void main (String[] arg) {…}
```

f)

2. Was ist gemeint?

   Dadurch wird dem Computer angegeben, welche Variablen genutzt werden sollen und welche Art von Werten in den Variablen gespeichert werden sollen.

   Sie sind Behälter für Daten bzw. Werte.

   Damit kann häufig genutzter Code mehrfach verwendet werden.

   Das Ausführen des in einer Methode ausgelagerten Quellcodes wird dadurch angestoßen.

   Dort beginnt der Computer beim Ausführen des Programms.

   Das muss immer nach Anweisungen und Deklarationen stehen.

   Sie dienen der besseren Lesbarkeit des Programms

MUSTERLÖSUNG FÜR DEN TEST ZUM ZWEITEN KAPITEL

1. Worum handelt es sich bei den angegebenen Quelltextausschnitten?

```
int x;
int y;
double z ;
```

a) Deklaration

```
static double umfang(double r){
    return r * 2.0 * 3.14159;
}
```

b) Methode

```
x = 100 + 4 * 3 / 4;
y = 12345;
```

c) Anweisungen

```
umfang(r);
```

d) Methodenaufruf

```
//Berechnung des Umfangs
```

e) Kommentar

```
public static void main (String[] arg) {…}
```

f) Hauptprogramm. Quelltext zwischen den geschweiften Klammern.

2. Was ist gemeint?

Dadurch wird dem Computer angegeben, welche Variablen genutzt werden sollen und welche Art von Werten in den Variablen gespeichert werden sollen.

Deklaration

Sie sind Behälter für Daten bzw. Werte.

Variablen

Damit kann häufig genutzter Code mehrfach verwendet werden.

Methoden

Das Ausführen des in einer Methode ausgelagerten Quellcodes wird dadurch angestoßen.

Methodenaufruf

Dort beginnt der Computer beim Ausführen des Programms.

Hauptprogramm

Das muss immer nach Anweisungen und Deklarationen stehen.

Semikolon

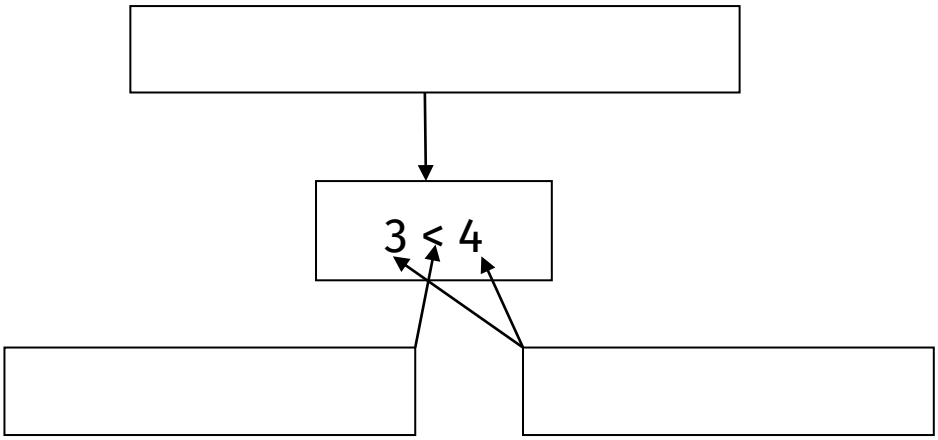
Sie dienen der besseren Lesbarkeit des Programms.

Kommentare

LERNKONTROLLE FÜR DAS DRITTE KAPITEL

Bearbeite die folgenden Aufgaben sorgfältig und ohne Hilfsmittel (z. B. Theorieteil des Leitprogramms)! Lasse sie von deiner Lehrkraft korrigieren. Wenn du nicht mehr als zwei Fehler machst, dann darfst du mit dem nächsten Kapitel beginnen.

1. Wie nennt man das, worauf die Pfeile zeigen im Allgemeinen? Trage die korrekten Bezeichnungen ein!



2. Welche Datentypen sind hier im Spiel?

_____

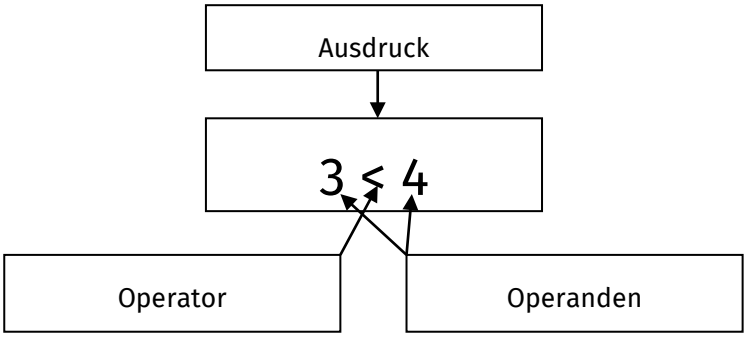
3. Fülle die Tabelle aus!

	Ergebnis	**Ergebnisdatentyp**
!((20 + 17) == 40)		
((23.0 + 17) != 4.0) && true		
10 / 3 + 2.1		
true && (2 > 3)		

(!(‘a’ == ‘b’)) && (!(!(2==2)))		
(‘a’ == ‘a’) && (2 > 3)		
(true && (‘x’ == ’x’)) || false		
(“ab“+“cd“) == ”abcd“		
(6.6 / 3.3) != (2 + 0.2)		
(10 / 4 == 1) || (‘a’ == ‘b’)		
(13 / 3 – 3) * 1234567891234		
‘Q’ != ‘q’		
!(“Hallo” == “Hallo”)		
(!(‘a’ == ‘a’)) == true		

MUSTERLÖSUNG FÜR DEN TEST ZUM DRITTEN KAPITEL

1. Wie nennt man das, worauf die Pfeile zeigen im Allgemeinen? Trage die korrekten Bezeichnungen ein



2. Welche Datentypen sind hier im Spiel?

   int und boolean

3. Fülle die Tabelle aus!

	Ergebnis	**Ergebnisdatentyp**
!((20 + 17) == 40)	true	boolean
((23.0 + 17) != 4.0) && true	true	boolean
10 / 3 + 2.1	5.1	double
true && (2 > 3)	false	boolean
(!('a' == 'b')) && (!(!(2==2)))	true	boolean
('a' == 'a') && (2 > 3)	false	boolean
(true && ('x' == 'x')) || false	true	boolean
("ab"+"cd") == "abcd"	true	boolean
(6.6 / 3.3) != (2 + 0.2)	true	boolean
(10 / 4 == 1) || ('a' == 'b')	false	boolean
(13 / 3 – 3) * 1234567891234	1234567891234	int
'Q' != 'q'	true	boolean
!("Hallo" == "Hallo")	false	boolean

(!(‘a’ == ‘a’)) == true	false	boolean

LERNKONTROLLE FÜR DAS VIERTE KAPITEL

Du hast bestanden, wenn du nicht mehr als zwei Fehler machst.

1. **Fülle die Kästen sinnvoll!**

```
public class Zinsen {
  public static void main (String [] arguments){
   String name;
   [        ] anlage, zins, dauer, gewinn ;

    System.out.println("*****ZINSRECHNER*******");

    System.out.print("Anlagebetrag in Euro: ");

    anlage = [                          ]

    System.out.print("Zinssatz in %: ");

    zins = [                          ]

    System.out.print("Anlagedauer in Jahren : ");

    dauer = [                          ]

    gewinn = [                          ]

    System.out.print("Gewinn: ");

    System.out.print([                ]);

    System.out.print(" Euro ");
  }
}
```

2. **Was macht das Programm?**

3. **Womit bewirkst du einen Zeilenumbruch innerhalb eines Strings?**

4. **Schreibe die Deklaration so kurz wie möglich um!**

```
int zahl1;
```

```
char y;
```

```
double zahl2 ;
int x;
int ergebnis;
double z ;
String Text2 ;
int zahl3;
String string;
double grossezahl;
```

---

5. Zeichne die durch die Anweisungen bewirkten Veränderungen ein!

x	y	z	merke	v
1	'a'	" tra "	3.1	5
int	char	String	double	int

```
x = 3 + v;
y = 'b';
z = "tri " + z;
merke = v + 1.4;
```

x	y	z	merke	v
int	char	String	double	int

```
x = v;
v = x+1;
z = z + "trullala"
```

x	y	z	merke	v
int	char	String	double	int

MUSTERLÖSUNG FÜR DEN TEST ZUM VIERTEN KAPITEL

1. **Fülle die Kästen sinnvoll!**

```
public class Zinsen {
  public static void main (String [] arguments){
    String name;
    double anlage, zins, dauer, gewinn ;

    System.out.println("*****ZINSRECHNER*******");
    System.out.print("Anlagebetrag in Euro: ");
    anlage = Kon.readDouble();
    System.out.print("Zinssatz in %: ");
    zins = Kon.readDouble();
    System.out.print("Anlagedauer in Jahren : ");
    dauer = Kon.readDouble();
    gewinn = anlage * zins/100 * dauer;
    System.out.print("Gewinn: ");
    System.out.print(gewinn);
    System.out.print(" Euro ");
  }
}
```

2. Was macht das Programm?

   Berechnung der Zinsen nach Einlesen von Anlagebetrag, Zinssatz und Laufzeit

3. Womit bewirkst du einen Zeilenumbruch innerhalb eines Strings?

   `\n`

4. Schreibe die Deklaration so kurz wie möglich um!

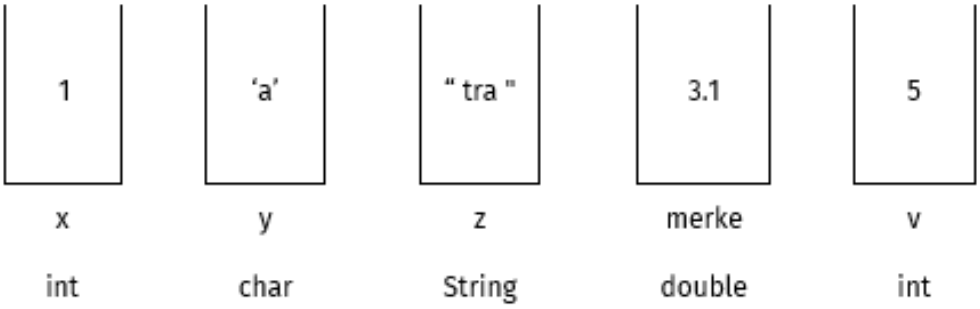
   int zahl1, x, ergebnis, zahl3;

   char y;

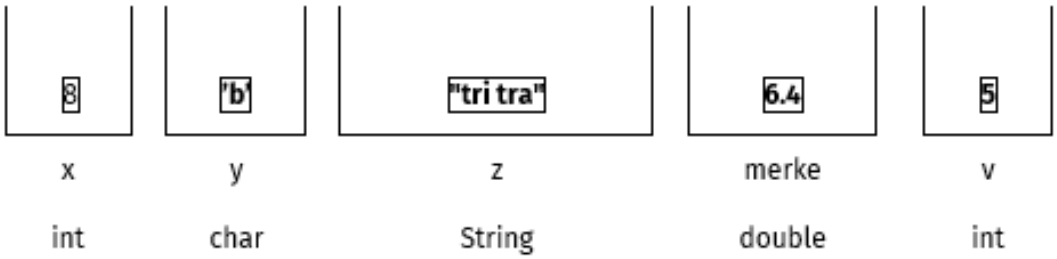
   double zahl2, z grossezahl;

   String Text2, string;

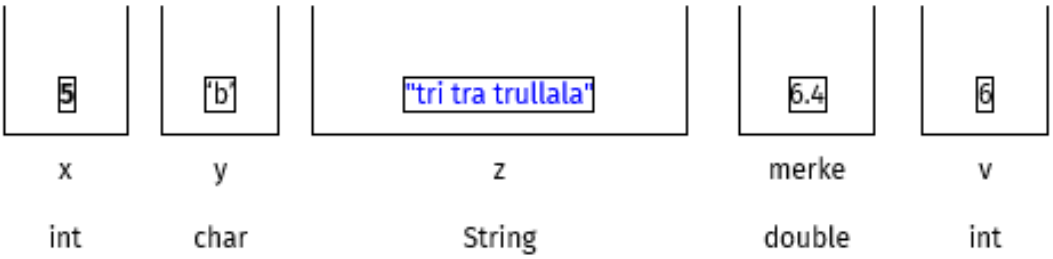
5. Zeichne die durch die Anweisungen bewirkten Veränderungen ein!




```
x = 3 + v;
y = 'b';
z = "tri " + z;
merke = v + 1.4;
```




```
x = v;
v = x+1;
z = z + "trullala"
```



A.5 LERNKONTROLLE FÜR DAS FÜNFTE KAPITEL

Du hast bestanden, wenn du nicht mehr als zwei Fehler machst.

Programm *Termine*:

```
public class Termine {
  public static void main (String [] arguments){
    char wochentag;
    System.out.println("******Deine festen Termine*****\n");
    System.out.println("(M)ontag,(D)ienstag,(M)ittwoch,(D)onnerstag,
                                                          usw.?");
    wochentag = Kon.readChar();
    switch (wochentag) {

      case 'M' :{
        System.out.print("M(o)ntag oder M(i)ittwoch?");
        wochentag = Kon.readChar();
        switch (wochentag) {
          case 'o':{
             System.out.print("15:00  Nachhilfe"); break;
          }
          case 'i':{
            System.out.print("16:30 Schwimmen"); break;
          }
          default:{
             System.out.print("FEHLER 1!");
          }
        }
        break;
      }
...Teile ausgeschnitten...
      case 'F' :{
        System.out.print("17:00 Fussball"); break;
      }

      case 'S' :{
        System.out.print("S(a)mstag oder S(o)nntag?");
```

```
        wochentag = Kon.readChar();
        switch (wochentag) {
          case 'a':{
             System.out.print("22:00 PARTY"); break;
          }
          case 'o':{
            System.out.print("bis 15:00 ausschlafen :-)"); break;
          }
          default:{
             System.out.print("FEHLER");
          }
        }
        break;
      }

      default:{
        System.out.println("FEHLER 2!");
      }
    }
  }
}
```

Beantworte die folgenden Fragen zu dem Programm *Termine*.

1. Was macht der Besitzer des Programms freitags?

   _____

2. Was macht er sonntags?

   _____

3. Was passiert, wenn man beim Ablauf des Programms erst ein M und dann ein x eingibt?

   _____

4. Was passiert, wenn man beim Ablauf des Programms als erstes ein X eingibt?

5. Wofür ist der Teil des Quellcodes, der fehlt, zuständig?

6. Hätte man dieses Programm auch mit if-Anweisungen schreiben können?

Weiter geht's auf der nächsten Seite ...

7. **Schreibe ein Programm für die Außentemperaturanzeige im Auto mit Hilfe der If-Anweisung.** Wenn die Temperatur unter 0 Grad fällt, dann wird zusätzlich zur Temperaturangabe eine Warnung vor Glätte ausgegeben. Nimm dazu an, dass in der Variable temperatur bereits die aktuelle Temperatur gespeichert ist.

```
public class Temperaturanzeige{
  public static void main (String [] arguments){
    // Temperaturanzeige mit Glättewarnung
    double temperatur; //aktuelle Temperatur
```

```
  }
}
```

1. Was macht der Besitzer des Programms freitags?
   17:00: Fußball spielen

2. Was macht er sonntags?

   Bis 15:00 Uhr ausschlafen ☺

3. Was passiert, wenn man beim Ablauf des Programms erst ein M und dann ein x eingibt?

   Es wird FEHLER 1 ausgegeben.

4. Was passiert, wenn man beim Ablauf des Programms als erstes ein X eingibt?
   Es wird FEHLER 2 ausgegeben.

5. Wofür ist der Teil des Quellcodes, der fehlt, zuständig?

   Er gibt die Termine für Donnerstag und Freitag aus.

6. Hätte man dieses Programm auch mit if-Anweisungen schreiben können?
   Ja.

7. Schreibe ein Programm für die Außentemperaturanzeige im Auto mit Hilfe der If-Anweisung. Wenn die Temperatur unter 0 Grad fällt, dann wird zusätzlich zur Temperaturangabe eine Warnung vor Glätte ausgegeben. Nehme dazu an, dass in der Variable temperatur bereits die aktuelle Temperatur gespeichert ist.

```
public class Temperaturanzeige{
  public static void main (String [] arguments){
    // Temperaturanzeige mit Glättewarnung
    double temperatur; //aktuelle Temperatur
    temperatur = -5;
    if (temperatur < 0){
      System.out.print(temperatur);
      System.out.print("Grad Celsiua: Glaettegefahr!");
    }else{
      System.out.print(temperatur);
      System.out.print("° c");
    }
  }
}
```

A.6 LERNKONTROLLE FÜR DAS SECHSTE KAPITEL

Du hast bestanden, wenn du nicht mehr als zwei Fehler machst.

1. Schreibe zwei Programme zum Potenzieren von ganzen Zahlen.

   Die Programme berechnen

   $$a^b$$

   und speichern das Ergebnis in

   `ergebnis`.

a) Verwende eine for-Schleife!

```
public class HochFor {
  public static void main (String [] arguments){
     int a, b, i, ergebnis;
     ergebnis = 1;
     a = Kon.readInt();
     System.out.println("hoch");
     b = Kon.readInt();
     System.out.println("=");









     System.out.println(ergebnis);
  }
}
```

b) Verwende eine while-Schleife!

```
public class HochWhile {
  public static void main (String [] arguments){
     int a, b, i, ergebnis;
     ergebnis = 1;
     a = Kon.readInt();
     System.out.println("hoch");
     b = Kon.readInt();
     System.out.println("=");
     System.out.println(ergebnis);
  }
}
```

2. Was geben die folgenden Programmteile aus?

```
i = 1;
while (i<=20){
  if (i % 4 == 0){
    System.out.print(i);
    System.out.print(" ");
  }
  i=i+1;
}
```

Ausgabe:

```
for (i=20; i<50; i++){
  if ((i % 9 == 0)||(i>45)){
    System.out.print(i);
    System.out.print(" ");
  }
}
```

Ausgabe:

```
for (i=1; i<=5; i++){
    for (j=1; j<=i; j++){
      System.out.print(j);
      System.out.print(" ");
    }
    System.out.println("");
  }
```

Ausgabe:

1. Schreibe zwei Programme zum Potenzieren von ganzen Zahlen.

   Die Programme berechnen

   $$a^b$$

   und speichern das Ergebnis in

   `ergebnis`.

a) Verwende eine for-Schleife!

```
public class HochFor {
  public static void main (String [] arguments){
     int a, b, i, ergebnis;
     ergebnis = 1;
     a = Kon.readInt();
     System.out.println("hoch");
     b = Kon.readInt();
     System.out.println("=");

     for (i=1; i<=b; i++){
       ergebnis = ergebnis * a;
     }

     System.out.println(ergebnis);
  }
}
```

2. Verwende eine while-Schleife!

```
public class HochWhile {
  public static void main (String [] arguments){
     int a, b, i, ergebnis;
     ergebnis = 1;
     a = Kon.readInt();
     System.out.println("hoch");
     b = Kon.readInt();
     System.out.println("=");
```

```
      i=1;
      while (i<=b){
        ergebnis = ergebnis * a;
        i = i+1;
      }

      System.out.println(ergebnis);
    }
  }
```

3. Was geben die folgenden Programmteile aus?

```
i = 1;
while (i<=20){
  if (i % 4 == 0){
    System.out.print(i);
    System.out.print(" ");
  }
  i=i+1;
}
```

Ausgabe: 48 12 16 20

```
for (i=20; i<50; i++){
  if ((i % 9 == 0)||(i>45)){
    System.out.print(i);
    System.out.print(" ");
  }
}
```

Ausgabe: 27 36 45 46 47 48 49

```
for (i=1; i<=5; i++){
  for (j=1; j<=i; j++){
    System.out.print(j);
    System.out.print(" ");
  }
  System.out.println("");
}
```

Ausgabe:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

A.7 LERNKONTROLLE FÜR DAS SIEBTE KAPITEL

Du hast bestanden, wenn du nicht mehr als zwei Fehler machst.

1. Erzeuge diese Arrays, und ergänze die Kästen im Quellcode:

			y	
	x		0	1
0	3.14	0	'*'	'-'
1	-1.0	1	'-'	'*'
2	2.7	2	'+'	'-'

```
public class mehrereArrays {
  public static void main (String [] arguments){
    int i,j;
    // Deklaration der Arrays
    double x[];
    char y[][];

    // Anlegen der Arrays
```

```
```

```
    // Schreiben von Werten in das Array x
```

```
```

MUSTERLÖSUNG

1. Erzeuge diese Arrays:

x
3.14
-1.0
2.7

	0	1
0	'*'	'-'
1	'-'	'*'
2	'+'	'-'

```
public class mehrereArrays {
  public static void main (String [] arguments){

    int i,j;
    // Deklaration der Arrays
    double x[];
    char y[][];
    // Anlegen der Arrays
    x = new double [3];
    y = new char [3][2];
    // Schreiben von Werten in das Array x
    x[0] = 3.14;
    x[1] = -1.0;
    x[2] = 2.7;
    // Schreiben von Werten in das Array y
    y[0][0] = '*';
    y[0][1] = '-';
    y[1][0] = '-';
    y[1][1] = '*';
    y[2][0] = '+';
    y[2][1] = '-';

    //Ausgabe des Arrays x
    System.out.println("x");
    for (i=0; i<3; i++){
```

ANHANG B: MEDIOTHEK

Für das Erstellen und Testen der Programme und die Arbeit mit dem Internet stehen Arbeitsplätze mit Rechnern zur Verfügung. Je nach Angebot kann alleine oder zu zweit gearbeitet werden.

- Verwendete Programmiersprache: Java
- Entwicklungsumgebung: JavaEditor (Freeware)
- Browser: Firefox

Bei den hier dargestellten Anwendungen handelt es sich ausschließlich um freiverfügbare Software. Sie lassen sich auf den folgenden Internetseiten herunterladen:

- Java 2 Platform, Standard Edition, v 1.4.2 (J2SE):

  http://java.sun.com/j2se/1.4.2/download.html

- JavaEditor:

  http://www.bildung.hessen.de/abereich/inform/skii/material/java/installation.htm

- Firefox for Windows:

  http://www.mozilla.org/

ANHANG C: LITERATURANGABEN

Mössenböck H.: Sprechen Sie Java? Eine Einführung in das systematische Programmieren. 3. Auflage, dpunkt.verlag, 2005.

Bishop J.: Java lernen. 2. Auflage, Addison-Wesley, 2001.

Barnes D., Kölling M.: Objektorientierte Programmierung mit Java, Pearson Studium, 2003.

Mössenböck H.: Grundlagen der Programmierung WS 2003/2004 an der Johannes-Kepler-Universität Linz.

Lichter H.: Vorlesung Programmierung WS 2000/2001 an der RWTH-Aachen.

Giesel J.: Vorlesung Programmierung WS 2001/2002 an der RWTH-Aachen.

ANHANG D: ABBILDUNGSVERZEICHNIS

- **Titelbild** und **Modullogo** – *Quelle: Screenshot des JavaEditors*, http://www.bildung.hessen.de/abereich/inform/skii/material/java/installation.htm
- **Arbeitssymbole** , , , , , , , – *Quelle: InfoSphere*
- Alle weiteren Abbildungen und Tabellen wurden durch die Autoren dieses Leitprogramms erstellt.