Übersicht

Inhalte

✓ Einfache Beispielprogramme

Lexikalische Konventionen

Aufgabe 02.01

- Hello, world! in Eclipse
- Legen Sie ein neues Projekt in Eclipse namens OOP2 (für Kapitel 2) an
- Legen Sie die Klasse ***HelloWorld*** an und implementieren Sie diese wie in der Vorlage

```
public class HelloWorld{
    public static void main(String args[]){
        System.out.println("Hello, world!");
    }
}
```

Lokale Variable, Ausdrücke und Schleifen I

- Symbolische Konstanten
- Vergleichsoperator vs. Zuweisungsoperator
- Schleife

```
1  // Datei: Fahrenheit.java -->  Klasse zur Wandlung von Temperaturen von Fahrenheit nach Celsius
2  public class Fahrenheit
3  {
4     public static void main (String[] args)
5     {  // Konstanten
6        final int UPPER = 300;    // obere Grenze, UPPER ist eine symbol. Konstante,
7                                  //300 ist eine literale Konstante
8        final int LOWER =   0;    // untere Grenze
9        final int STEP =   20;    // Schrittweite
10
11       // Variablen
12       int fahr;                 // Definition der lokalen Variablen, fahr für die Temperatur in  Fahrenheit
13       int celsius;              // Definition der lokalen Variablen, celsius für die Temperatur in Celsius
14
15      // Anweisungen
16       fahr = LOWER;             // als Anfangswert wird fahr der Wert 0 zugewiesen
17
18       while (fahr <= UPPER)
19       {
20          celsius = 5 * (fahr - 32) / 9;     // nach dieser Formel berechnet sich der Celsius-Wert aus einem Fahrenheit-Wert
21
22
23          System.out.print (fahr);       // der Wert von fahr wird auf
24                                         // den Bildschirm ausgegeben
25          System.out.print ("     ");    // Leerzeichen in derselben Zeile
26
27          System.out.println (celsius);
28                                 // Der Wert von Celsius wird in
29                                 // derselben Zeile ausgegeben.
30                                 // Anschließend springt der Cursor
31                                 // zum Anfang der naechsten Zeile.
32
33          fahr = fahr + STEP;    // Der naechste Wert von fahr  wird berechnet
34       }
35    }
36 }
```

Lokale Variable, Ausdrücke und Schleifen II

- Nur mit int (ganzzahligen Ausdrücken)
- Schleife
- Hinweis zur letzten Zeile: Statt einer Variablen kann auch ein komplizierter Ausdruck von diesem Typ stehen

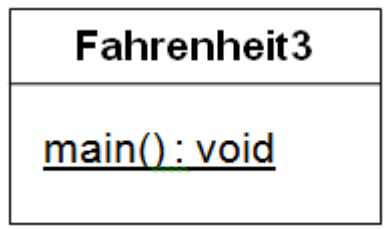
```
1   // Datei: Fahrenheit2.java
2
3   public class Fahrenheit2
4   {
5      public static void main (String[] args)
6      {
7         int fahr;
8
9         for (fahr = 0; fahr <= 300; fahr = fahr + 20)
10        {
11           System.out.print (fahr);
12           System.out.print ("     ");
13           System.out.println (5 * (fahr - 32) / 9);
14        }
15     }
16  }
```

Lokale Variable, Ausdrücke und Schleifen III

- Implizite Konvertierung bei 5.0/9,
- dann explizite Typkonvertierung von double nach float mit Hilfe des Cast-Operators

```
1  // Datei: Fahrenheit3.java
2
3  public class Fahrenheit3
4  {
5     public static void main (String[] args)
6     {
7        // Konstanten
8        final int UPPER = 300;    // obere Grenze
9        final int LOWER =   0;    // untere Grenze
10       final int STEP =   20;    // Schrittweite
11
12       int fahr;
13       float celsius;
14
15       fahr = LOWER;
16
17       while (fahr <= UPPER)
18       {
19          celsius = (float) (5.0 / 9) * (fahr - 32);
20          System.out.println (fahr + "      " +  celsius);
21          fahr = fahr + STEP;
22       }
23    }
24 }
```

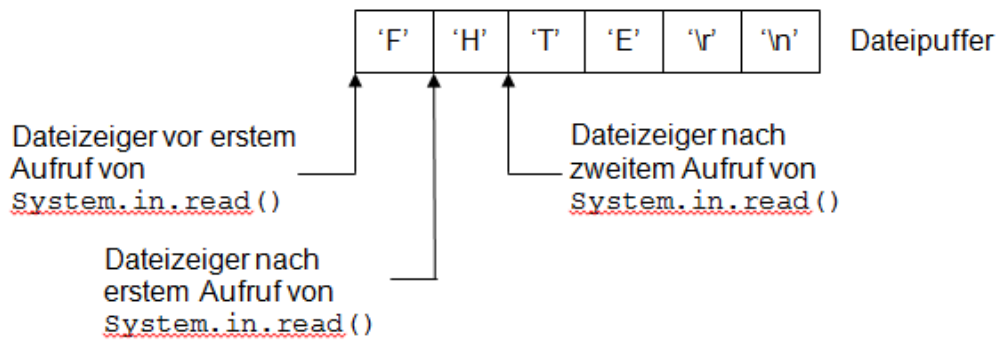
Visualisierung



Zeichen von der Tastatur einlesen

Einlesen von Zeichen

```
1  // Datei: Zeichen.java
2
3  public class Zeichen
4  {
5     // beachten Sie die Deklaration der Methode main() nicht
6     public static void main (String[] args) throws Exception
7     {
8        int c = 0;
9        int anzahl = 0;
10
11       System.out.print ("Bitte eine Folge von Zeichen eingeben ");
12       System.out.println ("und mit <RETURN> abschliessen");
13
14       do
15       {
16          // System.in.read() gibt einen int-Wert im Bereich 0 bis
17          // 255 zurueck. -1 wird zurueckgegeben, wenn kein Zeichen
18          // mehr im Dateipuffer steht.
19          c = System.in.read();
20
21          // Mit (char) c wird die int-Variable c
22          // in ein Zeichen gewandelt.
23          System.out.println (
24             "ASCII-Code: " + c + " Zeichen: " + (char) c);
25          anzahl = anzahl + 1;
26
27       } while (c != -1);
28       System.out.println ("Anzahl: " + anzahl);
29    }
30 }
```



Erzeugen von Objekten

```
Punkt p = new Punkt();
// oder...
Punkt p;
p = new Punkt();
```

- Typ-Deklaration der Variablen p und Instanziieren eines Objektes vom Typ p.
- Wenn nur innerhalb eines Methodenrumpfes sichtbar, spricht man von einer lokalen Variablen.
- Aufruf des Konstruktors, da dieser den gleichen Namen wie die Klasse trägt.
- Referenzvariable p zeigt auf ihr Objekt im Speicher.

Initialisierung von Objekten mit Konstruktoren

- Nicht jede Klasse muss eine ***main***-Methode haben
- In der Regel wird jede Klasse in einer eigenen Datei gespeichert
- Wenn mehrere Klassen in einer Datei sind, dann kann nur eine ***public*** sein!

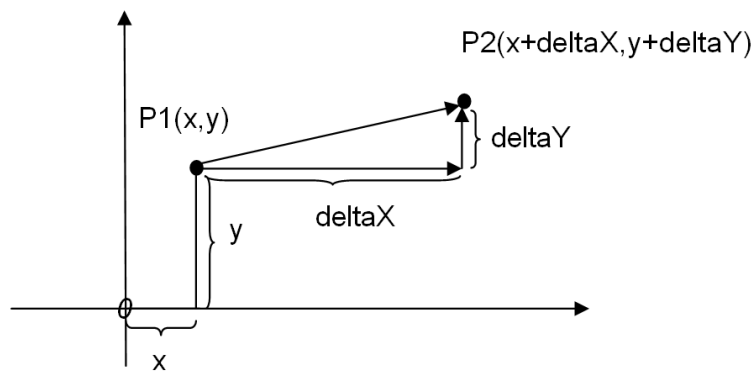
```java
// Datei: Punkt2.java
public class Punkt2
{
   private int x;
   public Punkt2()              // Dieser Konstruktor wird
   {                            // noch erklaert
      System.out.println ("Default-Konstruktor");
      x = 1;
   }
   public Punkt2 (int u)        // Dieser Konstruktor wird noch
   {                            // erklaert
      System.out.print ("Konstruktor mit einem Parameter:");
      System.out.println (" x = " + u);
      x = u;
   }
   public void print()
   {
      System.out.println ("x = " + x);
   }
}
```

```java
// Datei: TestPunkt2.java
public class TestPunkt2
{
   public static void main (String[] args)
   {
      Punkt2 p1 = new Punkt2();    // Erzeugen eines Punktes.
                                   // x wird durch Default-
                                   // konstruktor auf 1 gesetzt
      Punkt2 p2 = new Punkt2 (3); // Erzeugen eines Punktes.
                                   // x wird auf 3 gesetzt
      Punkt2 p3 = new Punkt2 (10);// Erzeugen eines Punktes.
                                   // x wird auf 10 gesetzt
      System.out.println ("Koordinate des Punktes p1:");
      p1.print();
      System.out.println ("Koordinate des Punktes p2:");
      p2.print();
      System.out.println ("Koordinate des Punktes p3:");
      p3.print();
   }
}
```

Schreiben von Instanzmethoden I

- Verschieben eines Punktes (Klasse Punkt3 erweitert um die Methode *verschiebe*)

```
1  // Datei: Punkt4.java
2
3  public class Punkt4   // Deklaration der Klasse Punkt4
4  {
5     private int x;     // Datenfeld für die x-Koordinate vom Typ int
6     private int y;     // Datenfeld für die y-Koordinate vom Typ int
7     public int getX()                // eine Methode, um den x-Wert
8     {                                // abzuholen
9        return x;
10    }
11
12    public int getY()                // eine Methode, um den y-Wert
13    {                                // abzuholen
14       return y;
15    }
16
17    public void setX (int i)         // eine Methode, um den x-Wert
18    {                                // zu setzen
19       x = i;
20    }
21
22    public void setY (int i)         // eine Methode, um den y-Wert
23    {                                // zu setzen
24       y = i;
25    }
26
27    public void verschiebe (int deltaX, int deltaY)
28    {
29       x = x + deltaX;
30       y = y + deltaY;
31    }
32 }
```



Schreiben von Instanzmethoden II

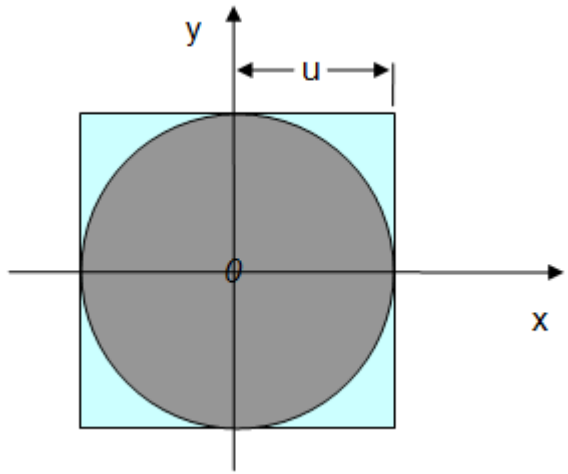
- Formalparameter mit aktuellem Parameter initialisieren

```
1  // Datei: TestPunkt4.java
2
3  public class TestPunkt4
4  {
5     public static void main (String[] args)
6     {
7        Punkt4 p = new Punkt4();    // hiermit wird ein Punkt erzeugt
8        p.setX (1);                 // Aufruf der Methode setX()
9        p.setY (2);                 // Aufruf der Methode setY()
10
11       System.out.println ("Die Koordinaten des Punktes p sind: ");
12       System.out.println (p.getX());
13       System.out.println (p.getY());
14
15       p.verschiebe (4, 1);
16       System.out.println ("Die Koordinaten des Punktes p sind: ");
17       System.out.println (p.getX());
18       System.out.println (p.getY());
19    }
20 }
```

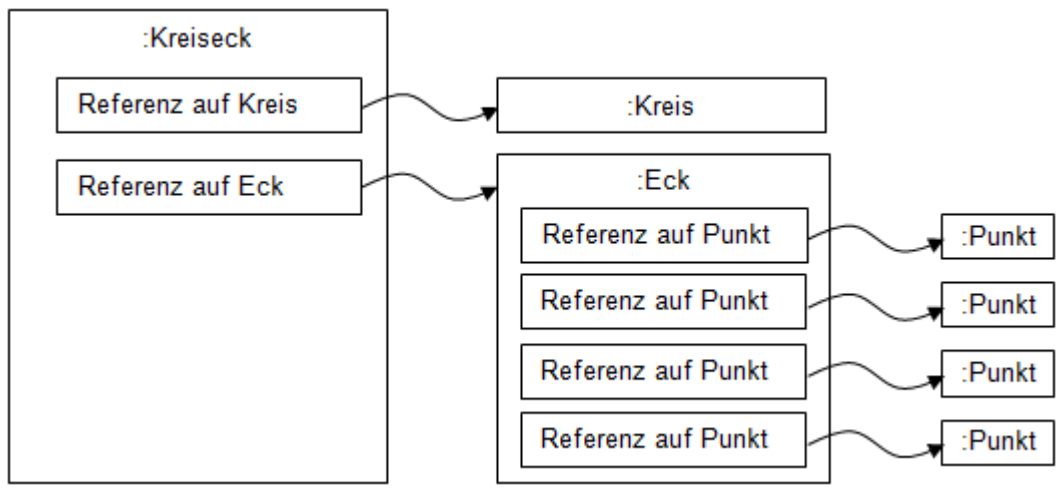
Zusammengesetzte Objekte I

Beispiel KreisEck

```
1  // Datei: Eck.java
2  public class Eck
3  {
4     private Punkt5 p1;
5     private Punkt5 p2;
6     private Punkt5 p3;
7     private Punkt5 p4;
8
9     public Eck (double u)    // u soll eine halbe Seitenlaenge
10    {                        // des Quadrats darstellen
11       System.out.println ("Viereck wird erzeugt aus 4 Eckpunkten");
12       p1 = new Punkt5 (u, u);
13       p2 = new Punkt5 (-u, u);
14       p3 = new Punkt5 (u, -u);
15       p4 = new Punkt5 (-u, -u);
16    }
17    public void skaliere (double u)
18    {
19       p1.setX (p1.getX() * u);
20       p1.setY (p1.getY() * u);
21       p2.setX (p2.getX() * u);
22       p2.setY (p2.getY() * u);
23       p3.setX (p3.getX() * u);
24       p3.setY (p3.getY() * u);
25       p4.setX (p4.getX() * u);
26       p4.setY (p4.getY() * u);
27    }
28    public double berechneFlaeche()
29    {
30       return (2 * p1.getX()) * (2 * p1.getY());
31    }
32 }
```



Visualisierung



Zusammengesetzte Objekte II

- Beispiel KreisEck

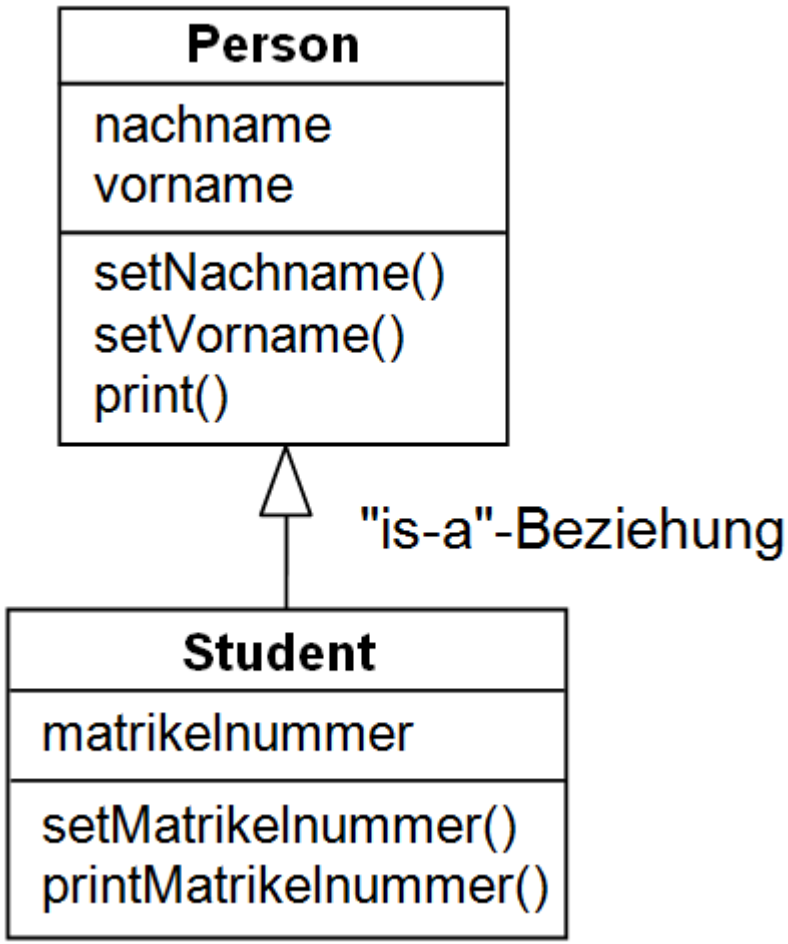
```java
// Datei: Kreis.java
public class Kreis
{
   private double radius;
   static final double PI = 3.1415;
   // PI ist eine konstante Klassenvar
   public Kreis (double u)
   {
      radius = u;
   }
   public void skaliere (double u)
   {
      radius = radius * u;
   }
   public double berechneFlaeche()
   {
      return (PI * radius * radius);
   }
   public double getRadius()
   {
      return radius;
   }
}
```

```java
// Datei: Kreiseck.java
public class Kreiseck
{
   private Kreis kreisref;
   private Eck eckref;
   public Kreiseck (double alpha) // alpha ist der Radius des
                                  // Inkreises
   {
      kreisref = new Kreis (alpha);
      eckref = new Eck (alpha);
   }
   public void skaliere (double u)
   {
      kreisref.skaliere (u); // Delegationsprinzip. Der Aufruf
      eckref.skaliere (u);   // skaliere (u) wird an die Komponenten
   }                         // weitergeleitet.
   public double flaechendifferenz()
   {
      return (eckref.berechneFlaeche() -
              kreisref.berechneFlaeche());
   }
   public Kreis getKreis()
   {
      return kreisref;
   }
}
```

- Testen mit der Klasse KreisEckTest

Selbst definierte Untertypen durch Vererbung I

- Abgeleitet werden Klassen. Objekte können nicht abgeleitet werden.
- Durch Ableitung wird ein Untertyp geschaffen, dies wird auch als Subtyping bezeichnet



Selbst definierte Untertypen durch Vererbung II

Klasse Person

```
1  // Datei: Person.java
2  import java.util.Scanner;
3  public class Person
4  {
5     private String name;
6     private String vorname;
7     // und können in einer Variable vom
8     // Namen sind konstante Zeichenketten
9     // Typ String gespeichert werden. Die
10    // Klasse String ist eine Bibliotheksklasse.
11    public Person()
12    {
13       System.out.print ("\nNachnamen eingeben ");
14       System.out.print ("(Ende mit <CR>): ");
15       name = input();
16
17       System.out.print ("Vornamen eingeben ");
18       System.out.print ("(Ende mit <CR>): ");
19       vorname = input();
20    }
21    public String input() // bitte überlesen Sie die
22    {
23       Scanner eingabe = new Scanner (System.in);
24       return eingabe.next();
25    }
26    public void print()
27    {
28       System.out.print ("\nNachname: " + name);
29       System.out.print ("\nVorname: " + vorname);
30    }
31 }
```

Klasse Student

```
1  // Datei: Student.java
2  public class Student extends Person
3  {
4     private String matrikelnummer;
5     public Student()
6     {
7        super();  // Aufruf des Konstruktors der Vaterklasse
8
9        System.out.print ("Matrikelnummer eingeben ");
10       System.out.print ("(Ende mit <CR>): ");
11       matrikelnummer = input();
12    }
13    public void printMatrikelnummer()
14    {
15       System.out.print ("\nMatrikelnummer : " + matrikelnummer);
16    }
17    public static void main (String[] args)
18    {
19       System.out.print ("\nErfasse Person");
20       Person pers1 = new Person();
21
22       System.out.print ("\nErfasse Student");
23       Student stud1 = new Student();
24
25       System.out.print ("\nAusgabe Person");
26       pers1.print();
27
28       System.out.print ("\n\nAusgabe Student");
29       stud1.print();
30       stud1.printMatrikelnummer();
31    }
32 }
```

Methode printf()

- Erwartet als ersten Parameter einen Formatierungsstring

Sonderzeichen	Bedeutung
%d	Integer (32 Bit, mit Vorzeichen)
%u	Integer (32 Bit, positiv, ohne Vorzeichen)
%hu	liest weiterhin ein int, konvertiert es jedoch zu unsigned short int
%x, %X	integer in hexadezimaler Schreibweise (hex, kein Standard!)
%o	Integer in oktaler Schreibweise
%f	Fließkommazahl
%e, %E	Fließkommazahl in Exponentialdarstellung
%c	Wert als ASCII-Zeichen (character)
%s	Adresse als Zeichenkette (String)

- Nach dem Formatstring kommen als weitere Parameter die auszugebenden Variablen bzw. Ausdrücke

printf-Befehl	Ausgabe
`printf("Insg. %d Euro %d Cent", 15, 20);`	`Insg. 15 Euro 20 Cent`
`printf("%5d Euro", 123);`	`...123 Euro`
`printf("%8d Dollar", 123);`	`.....123 Dollar`
`printf("%-8d Cent", 123);`	`123..... Cent`
`printf("%3d $", 12345);`	`12345 $`
`printf("Nr. %d", 255);`	`Nr. 255`
`printf("Nr. %o", 255);`	`Nr. 377`
`printf("Nr. %h", 255);`	`Nr. ff`
`printf("Euro %.2f", 34.565);`	`Euro 34.57`
`printf("Euro %7.2f", 34.564);`	`Euro ..34.56`
`printf("%5tT", new Calendar());`	`19:47:19`
`printf("%5tF", new Calendar());`	`2004-10-09`

Klasse Scanner

- Einfache, mächtige Eingabemöglichkeit mit integrierter Typdefinition

```
1  // Datei: EingabeTest.java
2  import java.util.Scanner;
3  public class EingabeTest
4  {
5     public static void main (String[] args)
6     {
7        // Erzeugen eines Objektes der Klasse Scanner, um von
8        // der Standard-Eingabe (Tastatur) einzulesen.
9        Scanner eingabe = new Scanner (System.in);
10
11       System.out.print ("Geben Sie Ihren Namen ein: ");
12       String name = eingabe.next();
13       System.out.println ("Hallo " + name +
14                   "! Heute wollen wir zwei Zahlen addieren.");
15
16       System.out.print (name + ", gib die erste Zahl ein: ");
17       int zahl1 = eingabe.nextInt();
18       System.out.print ("OK. Und nun die zweite Zahl: ");
19       int zahl2 = eingabe.nextInt();
20       System.out.println ("Das Ergebnis ist: " + zahl1 +
21                     " + " + zahl2 + " = " + (zahl1 + zahl2));
22    }
23 }
```

Übersicht

Inhalte

- ✓ Einfache Beispielprogramme
- ✓ Lexikalische Konventionen

Zeichenvorrat

- Java benutzt den Unicode-Zeichensatz
- Die Wörter oder Zeichengruppen, aus denen Programmtext aufgebaut ist, werden als **lexikalische** Einheiten bzw. als **Token** bezeichnet , dieses sind:
  - ✓ Namen
  - ✓ Schlüsselwörter
  - ✓ Konstanten
  - ✓ Satzzeichen (Interpunktionszeichen, engl. separators)
  - ✓ Operatoren
- Werden durch Trenner begrenzt
  - ✓ Zwischenraum (Whitespacezeichen, Leerzeichen…)
  - ✓ Kommentare (Block, Zeile, Dokumentation)
  - ✓ Satzzeichen
  - ✓ Operatoren
- Java ist case sensitive

Style-Guide für die Programmierung

- Gestaltungsrichtlinie für das Erstellen von Programmen, um die Lesbarkeit zu erhöhen
- Dazu gehören auch Konventionen über das Einrücken in Blöcken etc. Für Namen hat sich in Java der folgende Programmierstil durchgesetzt:

Name	Konvention	Beispiel
Variablennamen	Kleinbuchstaben	`variable`
Datenfeldnamen	Kleinbuchstaben	`vorname`
Methodennamen	Kleinbuchstaben	`methode()`
Klassennamen	1. Buchstaben groß, Rest klein	`Person`
symbolische Konstanten	alle Buchstaben groß	`MAXIMUM`

- Aus mehreren Wörtern zusammengesetzte Namen werden ohne Unterstrich, das zweite Worte groß geschrieben. Z.B. *verschiebePunkt()*
- Keine Sonderzeichen wie z.B. % & §, aber $ ist erlaubt
- Umlaute sind erlaubt, sollten aber besser nicht genutzt werden
- Keine Zahl am Anfang
- Keine Schlüsselwörter

Style-Guide für die Programmierung

- Schlüsselwörter Auszug

boolean	double	int	return
break	else	long	static
case	extends	new	super
catch	final	null	switch
char	float	package	this
class	for	private	try
default	if	protected	void
do	import	public	while

Aufgabe 02.02

- Welche Bezeichner können in Java verwendet werden?

  - anfang&endVariable
  - porsche 911
  - erster_Eintrag_nach_Beendigung_von_2_Schleifen
  - new
  - 1objekt
  - objekt_extends_class
  - else123

Code Formatierung

- Einrückung, wenn neuer Block beginnt (Tabulator oder 4 Leerzeichen)
- Öffnende geschweifte Klammer immer als letztes Zeichen am Ende einer Zeile (K&R-Stil)

```
public class HelloWorld{
    public static …
}
```

- oder als einziges Zeichen in der darauf folgenden Zeile in Höhe des ersten Zeichens (Allman-Stil)

```
public class HelloWorld
{
    public static …
}
```

- Die schließende geschweifte Klammer steht einzeln in Zeile auf Höhe des ersten Zeichen

Kommentare

- Kommentarblock `/* dies ist ein Blockkommentar */`
- Zeilenkommentar `// Zeilenkommentar`
- Dokumentationsdokumentar

```
1  // Datei: DocuTest2.java
2
3  /** Ich bin ein Kommentar und erlaeutere die Klasse DocuTest2
4    * @version 1.0
5    * @author  Rainer Brang
6    */
7  public class DocuTest2
8  {
9     /** Ich bin ein Kommentar und erlaeutere das Datenfeld x */
10    public int x;
11
12    /** Erlaeuterung der Methode meth()
13      * @param para Hier die Beschreibung des Parameters
14      * @return  Kein Rueckgabewert
15      */
16    public void meth (int para)
17    {
18       // Anweisungen
19    }
20 }
```

Kommentare

- Dokumentationskommentar
- Zusätzliche Kommentierung von z.B. Übergabeparameter, Rückgabewerte etc. über sogenannte Tags

Tag	Bedeutung	Kommentartyp
@see	erstellt einen Link zu anderen Klassen	Klasse, Datenfeld, Methode
@version	gibt die Version an	Klasse
@author	gibt den Autor an	Klasse
@param	beschreibt einen Parameter näher	Methode
@return	beschreibt den Rückgabewert	Methode
@exception	beschreibt die Exception näher	Methode
@deprecated	markiert ein Element als deprecated	Klasse, Datenfeld, Methode

Aufgabe 02.03

- Hello, world! in Eclipse

```
public class HelloWorld{
    public static void main(String args[])
    {
        System.out.println("Hello, world!");
    }
}
```

- Fügen Sie Kommentare für Autor, Version, Klasse und main-Methode mit Parameter in den Quelltext ein, und erzeugen Sie das zugehörige javadoc.

Aufgabe 02.04

- Fehlerbehandlung
- Finden Sie den Fehler im Programm! Was könnten Sie bei dieser einfachen Klasse besser machen?

```
public class HelloWorld {
    public static void main (String[] args)
    {
        int zahl = 7;
        System.out.println("Bildschirmtext");
        System.out.println(zahl)
        System.out.println("Toll, oder?");
    }
}
```