Strings

Ein String ist in Java eigentlich eine Klasse, wir können ihn aber zunächst als Datentyp betrachten, der zur Speicherung von Zeichenketten dient.

Beispiel:

```
String zeile   = "Cusanus-Gymnasium Wittlich";
String code    = "234XF56ΓΨ";
String vorname = "Nikolaus";
String zuname = "Cusanus";
```

Stringsvariable können mittels Konkatenation zu einem einzigen String zusammengefasst werden.

```
String name = vorname + " " + zuname;
```

In eine Konkatenation können auch Variable mit anderen Datentypen einbezogen werden. Diese werden intern zunächst in Strings umgewandelt.

Im folgenden Beispiel werden zwei Stringvariable, eine Integervariable und ein Textstring in einer neuen String-Variable zusammengefasst:

```
 int  geburtsjahr = 1401;
String: ausgabe;

ausgabe = vorname + "   " +zuname+"  ist  "+geburtsjahr+
                " in Kues an der Mosel geboren.";

System.out.print( ausgabe);
```

ergibt: Nikolaus Cusanus ist 1401 in Kues an der Mosel geboren.

Ein- und Ausgabe von Variablen

**In Javt existiert eine Klasse IO (für Input Output)
In der die Methoden (bzw. Funktionen) für die Ein- und Ausgabe enthalten sind.**

Diese Klasse muss zuerst einmal mit der Anweisung Import java.io.* ; importiert werden. Der Stern ist eine sogenannte Wildcard (Joker) und bewirkt, dass die Klasse io mit……

Für die Ausgabe verwenden wir die Anweisung
System.out.print(……);

**Die Eingabe ist allerdings etwas aufwändiger:
Zuerst definieren wir eine neues Objekt eingabe als Instanz der Klasse BufferedReader. Diese Klasse enthält u.a. eine Methode (Funktion), die es gestattet, eine Zeichenfolge von der Tastatur einzulesen.**

```
String einstr;
BufferedReader eingabe = new BufferedReader(new
                                InputStreamReader(System.in));
einstr=eingabe.readln();
```

Danach steht der über die Tastatur eingegebene (Abschluss mit der Enter-Taste) Eingabestring in der Variablen einstr zur Verfügung.

Vorsicht: Zahlen werden hierbei zuerst als Zeichenfolgen eingelesen und müssen danach in echte Zahlen des ent-Sprechenden Typs umgewandelt werden.

Umwandlung von Strings in Zahlen

int i = Integer.parseInt("-2314") ;

belegt die Integer-Variable i mit demjenigen Integerwert, der dem Zahlenwert des Strings "-2314" entspricht. Entsprechend weisen

byte b = Byte.parseByte("107");

float f = Float.parseFloat("- 45.3678");

double d = Double.parseDouble("2.346 E6") ;

den Variablen die entsrechende Zahl zu.

Wenn die Strings für die Darstellung der jeweiligen Zahl ungeeignete Zeichen enthalten (z.B. Buchstaben oder im Fall von Integern auch einen Dezimalpunkt), so wird die Ausführung mit einer Fehlermeldung unterbrochen.

Bei Gebrauch dieser Typumwandlungen muss die Methode main() folgendermaßen erweitert werden:

Public static void main(String[ ] args)throws IOException

Umwandlung von Zahlen in Strings

Beispielprogamm Kreis1 gibt nach der Eingabe des Radius den Umfang und den Flächeninhalt des zugehörigen Kreises aus:

```
import java.io.*;

public class Kreis1
{
  public static void main(String argv[])throws IOException
  {
     double pi=3.1415;
     String radiusstr;
    double  radius,umfang,flaeche;

     BufferedReader eingabe = new BufferedReader(new
                                InputStreamReader(System.in));
    // Eine neue Instanz bzw. ein neues Objekt der Klasse BufferedReader
     System.out.println("Brechnung von Kreisumfang und
                                                    Kreisfläche ");
     System.out.print("Radius r=  ");
     radiusstr=eingabe.readLine();
    // Tastatur-Eingabe als Zeichenfolge in die Variable str
     radius=Float.parseFloat(radiusstr);
    // Umwandlung des Strings in eine Kommazahl   z.B. "2.56"→ 2.56

     umfang=2*pi*radius;
     flaeche=pi*radius*radius;

     System.out.println("Kreisumfang U= " +umfang);
     System.out.println("Kreisfläche   A= " +flaeche);

    System.out.println("Programmende");
   }                                    // Ende von main
}                                       // Ende von classs
```

args

Programmaufruf mit Argumenten:
Um Programm HalloWelt die ausgegebene Nachricht zu ändern, muss es neu editiert und kompiliert werden.

Das Programm soll nun so geändert werden, dass es durch Eingabe von *Argumenten* von außen beeinflusst werden kann.
Der Aufruf von HalloWelt2 Cusanus soll die Ausgabe "Hallo Welt hier ist Cusanus !" erzeugen.

Beim Aufruf des Programms mit mehreren Argumenten, zum Beispiel HalloWelt3 Cusanus Kopernikus werden die Argumente fortlaufend in die Parameterliste args der Methode main() gelesen. args ist ein Array (eine Liste) von Stringvariablen, die in der Klammer von main() deklariert ist.

In der Methode main() können diese Strings wie Variablen als args[0], args[1], args[2],.. angesprochen und verwendet werden.
Beachte dass die Zählung mit dem *Index* 0 für das erste Argument beginnt.

```
public class HalloWelt2
{
 public static void main(String[] args)
 {
  System.out.println("Hallo Welt hier ist "+args[0]+" !");
 }
}
```

```
public class HalloWelt3
{
 public static void main(String[] args)
 {
  System.out.println("Hallo Welt hier sind "+args[0]+
                     "  "+args[1]+ " !");
 }
}
```

Aufgabe 1.10:
Schreiben Sie das Programm Kreis1 so um, dass der dort verwendete Radius als Argument an das Programm Kreis2 übergeben wird. Teste das Programm durch Aufruf mit verschiedenen Parameterwerten.

Aufgabe 1.20
Schreiben Sie ein Programm Rechnen1, das mit zwei ganzen Zahlen als Argumenten aufzurufen ist. Das Programm soll die Summe, die Differenz, das Produkt, den Quotienten und den ganzzahligen Rest bei der Division dieser beiden Zahlen ausgeben.

Der Aufruf Rechnen1 97 7 soll folgende Ausgabe erzeugen:
97+7= 104
97-7 = 90
97*7= 679
97 DIV 7= 13
97 MOD 7 = 6
Beispiel:

```
import java.io.*;

public class Subtraktion2 {

  public static void main(String args[ ])throws IOException {
    int Minuend, Subtrahend, Differenz ;
    String str;
    BufferedReader eingabe = new BufferedReader(new
                                                InputStreamReader(System.in));
    System.out.println("Subtraktion zweier Zahlen");
    System.out.print("Minuend: ");
    str=eingabe.readLine();                  // Tastatureingabe in String str
    Minuend=Integer.parseInt(str);           // Eingabestring in INTEGER

    System.out.print("Subtrahend: ");
    str=eingabe.readLine();
    Subtrahend=Integer.parseInt(str);

    Differenz=Minuend-Subtrahend;
    System.out.println(Minuend+" - "+Subtrahend+" = "+Differenz);
    System.out.println("-----------------------------------------------------------------");
   }   //  end of main
}
```

Typumwandlung

Manchmal ist es nötig, den Typ primitiver Datentypen umzuwandeln.
(String ist kein primitiver Datentyp !)

Implizite Typumwandlung

geschieht automatisch, wenn es sich um eine *weitende Typumwandlung* handelt, d.h. der Zieldatentyp einen größeren Bereich umfasst als der Quelldatentyp.
Die Sequenz

```
double d = 3.0;
int i = 2;
d = i;
```

wird von Java klaglos akzeptiert. Der Inhalt von i wird vor der Zuweisung an d in eine Fließkommazahl umgewandelt. d enthält am Ende den Wert 2.0.

Die Reihenfolge der Datentypen von "groß" nach "klein" lautet:

double float long int short byte

Explizite Typumwandlung (Cast)

ist bei einengenden Typumwandlungen nötig.
So liefert in obigem Beispiel `i = d;` eine Fehlermeldung, da beim Umwandeln einer Fließkommazahl in eine ganze Zahl Information verloren gehen kann.
Der Speicherplatz der Variablen i ist i.a. nicht groß genug, um den Inhalt der Variablen d aufzunehmen.
Solche einengenden Typumwandlungen erfordern deshalb den Typumwandlungsoperator (*cast-operator*).

In obigem Beispiel: i = (int) d;

Analog dazu wandelt (float) d die Variable d in ein Fließkommazahl vom Typ float um.

Vorsicht: Beim „Casten“ kann Information verlorengehen!

Konstante

Konstante **sind Werte, die bei der Erstellung eines Programmes festgelegt werden und während des Programmablaufs nicht geändert werden können.**

So sind alle direkt in den Programmtext geschriebenen Werte, wie Zahlen (22.1, 1.96..) oder Texte ("Hallo", "Name: ",...) Konstanten. Diese Konstanten, die keinen Namen besitzen, werden als *Literale* bezeichnet.

Konstante können aber auch als unveränderliche Variablen definiert werden. Bei der Deklaration wird dann das Attribut final eingefügt:

```
final double EUROKURS = 1.95583;
final int  MW_STEUER_SATZ = 16;
```

Konstante müssen bei ihrer Deklaration immer initialisiert werden.

Schreiben Sie die Namen von Konstanten in Gross-buchstaben, um sie von den Variablennamen abzuheben.