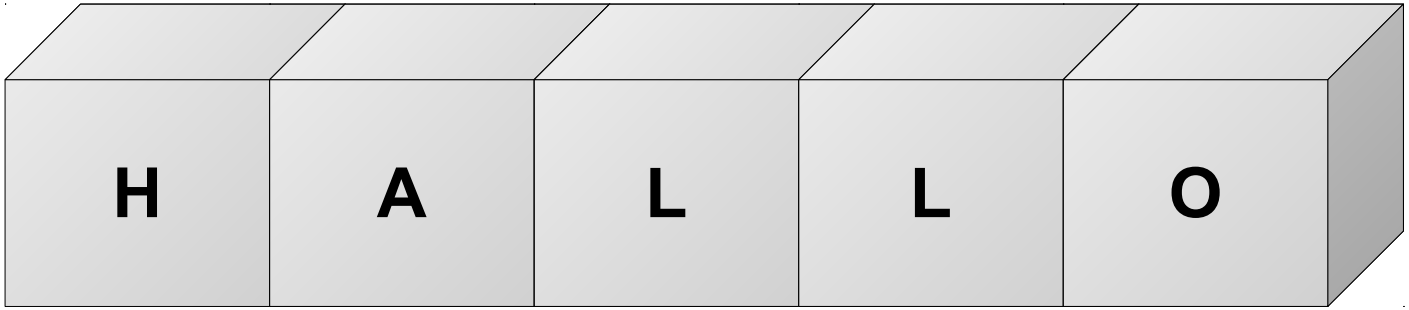
Java - Character und Strings



Character (ein Zeichen)

```
char buchstabe = 'A';
char ziffer = 65;
```

- Ein Zeichen wird als primitiver Datentyp `char` gespeichert.
- Literale als Character beginnen und enden mit einem Apostroph.
- Das Zeichen kann als Ganzzahl dargestellt werden. Es können Zeichen von 0 bis 65535 gespeichert werden.

ASCII-Zeichensatz

```
char buchstabe = 'A';
char ziffer = 65;
```

- ASCII (American Standard Code for Information Interchange). Definition von 128 Zeichen.
- ANSI oder Latin-1-Zeichensatz. Darstellung von 256 Zeichen. Die ersten 128 Zeichen entsprechen den ASCII-Zeichensatz.
- Siehe http://cs.stanford.edu/people/miles/iso8859.html.
- In dem obigen Beispiel wird der Buchstabe A als Literal und mit Hilfe der Ganzzahl 65 der Variablen zugewiesen. Die Ganzzahl 65 symbolisiert den Buchstaben A im ASCII-Zeichensatz.

Unicode-Zeichen

```
char buchstabe = 'A';
char ziffer = 65;
char unicode = '\u0041';
```

- \u maskiert die nachfolgende Zeichenfolge als Kodierung in einem Unicode-Zeichensatz.
- Das Unicode-Zeichen wird mit Hilfe von Hexadezimalzahlen kodiert.

Hinweise

- Das erste Zeichen im Unicode-Zeichensatz wird mit Hilfe von `'\u0000'` angegeben.
- Die ersten 127 Zeichen des UTF-8-Zeichensatzes sind mit den Zeichen im ASCII-Zeichensatz identisch.
- Die ersten 256 Zeichen des UTF-8-Zeichensatzes entsprechen den Zeichen in dem ISO 8859-1 (Latin 1)-Zeichensatz.
- Siehe https://unicode-table.com/de/.

Hexadezimalzahlen

- Zahlensystem zur Basis 16.
- Die Ziffern 0 bis 9 für die Zahlen 0 bis 9 aus dem Dezimalsystem. Die Buchstaben A bis F für die Zahlen 10 bis 15 aus dem Dezimalsystem.

„Maskierung“ von Zeichen

```
char unicode = '\u0041';
final char newLine = '\n';
```

- Mit Hilfe des Schrägstrichs wird ein Zeichen maskiert.
- Zeichen, die mit einem Schrägstrich beginnen, haben eine besondere Bedeutung für den Compiler.
- Die Zeichen, die dem `\u` folgen, werden als Unicode-Zeichen interpretiert. Das Zeichen `\n` fügt eine neue Zeile ein.

Escape-Sequenzen

```
final char newLine = '\n';
final char apostroph = '\'';
```

- Steuerzeichen für den Drucker etc.
- Nicht druckbare Zeichen eines Zeichensatzes.
- Maskierung von Zeichen, die in Java in einer besonderen Funktion genutzt werden.

Möglichkeiten

Escape-Sequenz	Beschreibung
\b	Rückschritt (Backspace)
\t	Tabulator
\n	Zeilenumbruch (Newline)
\f	Seitenvorschub (Form Feed)
\r	Wagenrücklauf. (Carriage Return)
\uxxxx	Zum Beispiel \u0041 für „A”
\"	Anführungszeichen
\'	Apostroph
\\	Backslash

Zeichencodes in NetBeans

- Rechtsklick auf das Projekt.
- Auswahl von *Properties* im Kontextmenü.
- Die Kategorie *Sources* ist aktiv.
- Das Kombinationsfeld *Encoding* auf der Seite rechts, unten gibt Auskunft über den genutzten Zeichensatz.

... und Microsoft Eingabeaufforderung

- Die Microsoft Eingabeaufforderung unterstützt standardmäßig nur ASCII-Zeichencode (siehe http://www.ascii-code.com/).
- Der Befehl *chcp* zeigt die aktuelle genutzte Codepage an.
- Der Befehl *chcp 1252* stellt die aktuelle Codepage auf „West European Latin“ um.

Strings

- Zeichenketten, zusammengesetzt aus Buchstaben, Zahlen, Satzzeichen, Sonderzeichen und mathematischen Symbolen
- Folge von Zeichen, die mit Hilfe des Unicode-Zeichensatzes kodiert sind.

Implementierung in Java

- Array vom Datentyp `char[]`.
- Die vordefinierte Klasse „String“ stellt Attribute und Methoden für Zeichenketten bearbeitet. Ein direkter Zugriff auf einen String ist in Java nicht möglich.

Implementierung als Array von char

```
char[] wort;
wort = new char[]{'H', 'a', 'l', 'l', 'o'};
```

- Jedes Zeichen eines Strings wird in einem Array abgelegt.
- Die Anzahl der Elemente spiegelt die Anzahl der Zeichen in der Zeichenkette wieder.
- Mit Hilfe des Index kann auf ein Zeichen zugegriffen werden.

Implementierung als Instanz der Klasse „String“

```
String strWort;
strWort = "Hallo";
```

- Der Typ `String` ist von der Klasse `java.lang.String` definiert.
- Die Klasse `java.lang.String` ist standardmäßig eingebunden und kann direkt genutzt werden.

Klasse

- Abstraktion von Dingen aus der realen Welt.
- Allgemeine Beschreibung von einem Objekt. Welche Eigenschaften hat ein Objekt? Wie verhält sich das Objekt?
- Bauplan für ein bestimmtes Ding zum Beispiel für Strings, dynamische Arrays etc.
- Formale Beschreibung einer bestimmten Objektgruppe.
- Vorlage für die Erzeugung eines Objektes.

Deklaration von Objekt-Variablen

`String`	`strWort`	`;`
`Klasse`	`variablenname`	`;`

- Statt eines Datentyps wird ein Klassenname als Typ angegeben. Der Klassenname muss exakt wie im Kopf (`public class klasseName`) der Klasse beschrieben, geschrieben werden.
- Der Variablenname ist frei wählbar.

Objekte

- Ein Ding (Exemplar, Instanz) aus der realen Welt.
- Objekte wie Strings etc.
- Eine abgeschlossen Einheit.
- Alle Elemente einer Kategorie von Dingen haben die gleichen Attribute, aber in unterschiedlichen Ausprägungen. Sie nutzen die gleichen Methoden zum Ändern ihrer Attribut-Werte.

Objekt-Variable

`String`	`strWort`	`;`
`Klasse`	`variablenname`	`;`

- Entsprechend der Klasse kann die Variable einen Verweis auf ein Objekt speichern.
- Instanzen von einer Klasse.
- Platzhalter für eine Referenz auf ein konkretes Objekt.

Initialisierung von Objekt-Variablen

`strWord`	=	`"Hallo"`
`variablenname`	=	`referenz`

- Mit Hilfe des Gleichheitszeichens wird ein Objekt von der Klasse erzeugt.
- In diesem Beispiel wird eine Referenz auf das Objekt `"Hallo"` erzeugt.
- Strings beginnen und enden immer mit den Anführungszeichen.

Strings deklarieren und initialisieren

```
String strSatz = "Java – Einführ. in die Programm.";
```

- Strings können gleichzeitig deklariert und initialisiert werden.

Nutzung eines Apostrophs

```
char chrApostroph = '\'';
String strApostroph = "'";
```

- Ein Apostroph begrenzt ein Wert vom Typ `char`. Das Apostroph als Zeichen muss in diesem Typ maskiert werden.
- In einem String wird das Apostroph als Zeichen behandelt.

Nutzung eines Anführungszeichen

```
char chrAnfuehrungszeichen = '"';
String strAnfuehrungszeichen = "\"";
```

- Ein Apostroph begrenzt ein Wert vom Typ `char`. Das Anführungszeichen wird als ASCII-Zeichen behandelt.
- Ein Anführungszeichen begrenzt ein String. Das Anführungszeichen als Zeichen muss in einem String maskiert werden.

Zahlen und Strings

```
String strZahl;
int intWert = 3;
double dblWert = 0.5;

strZahl = intWert;
strZahl = dblWert;
```

- Fehler: `incompatible types`.

Lösung

```
String strZahl;
int intWert = 3;
double dblWert = 0.5;

strZahl = String.valueOf(intWert);
strZahl = String.valueOf(dblWert);
```

- Mit Hilfe der Methode `.valueOf()` der Klasse `String` wird der Zahlenwert in den entsprechenden Typ konvertiert.

Methoden

- Was kann ein Objekt machen?
- Beschreibung der Funktionalität eines Objekts.
- Veränderungen von Objekten von außen her.
- Prozeduren, die in Klassen gekapselt sind.
- Alle Objekte einer Klasse haben die gleichen Methoden.

Nutzung des Punktoperators

- Methoden und Objekt-Variablen werden mit Hilfe des Punktoperators verbunden. Die Methode verändert das angegebene Objekt. Die aufzurufende Methode muss in der Klasse, von dem das Objekt ist, definiert sein.
- Klassenmethoden und Klassen werden mit Hilfe des Punktoperators verbunden. Klassenmethoden können ohne Erzeugung einer Instanz aufgerufen werden. Die aufzurufende Methode ist in der Klasse, links vom Punkt, definiert.

Verknüpfung von Strings

```
String strSatz;

strSatz = "GATCAAGGGC" + "TTTTATATAC";
strSatz = 16 + " Teilnehmer";
strSatz = "Teilnehmer " + 'A';
```

- Das Pluszeichen verknüpft Strings.
- Strings können mit Zahlen und Werten von Typ `char` verknüpft werden.
- Der Ausdruck wird von links nach rechts ausgewertet.

Operator „Pluszeichen“

```
String strSatz;

strSatz = 'a' + 'b' + 'c';
strSatz = 1 + 2 + 3;
```

- Fehler: incompatible types.
- In den oben geschilderten Beispielen werden die Integer-Werte addiert. Die Zuweisung des Ergebnisses erzeugt den Fehler.

Lösung

```
strSatz = "" + 'a' + 'b' + 'c';

strSatz = String.valueOf(1 + 2 + 3);
strSatz = String.valueOf(1) +  String.valueOf(2)
         + String.valueOf(3);
```

- In dem ersten Beispiel werden die Werte vom Datentyp `char` mit einem leeren String `""` verknüpft.
- In den letzten Beispielen werden die Zahlen mit Hilfe der Methode `.valueOf()` konvertiert.

Lösung

```
strSatz = "" + 'a' + 'b' + 'c';

strSatz = String.valueOf(1 + 2 + 3);
strSatz = String.valueOf(1) +  String.valueOf(2)
         + String.valueOf(3);
```

- In dem ersten Beispiel werden die Werte vom Datentyp `char` mit einem leeren String `""` verknüpft.
- In den letzten Beispielen werden die Zahlen mit Hilfe der Methode `.valueOf()` konvertiert.

Verknüpfungsmethode

```
String strSatz;

strSatz = "Teilnehmer " + 'A';
strSatz.concat(" und B");
```

- Mit Hilfe der Methode .concat() wird der String links vom Punktoperator mit dem String in den runden Klammern verbunden.
- Der verknüpfte String kann als Variable gespeichert werden.

Methoden der Klasse String

- Alle Instanzen der Klasse String haben die gleichen Methoden.
- Mit Hilfe von Methoden kann von außen auf Strings zugegriffen werden.
- Die Methode bezieht sich immer auf den String rechts vom Punktoperator.
- Methoden der Klasse String: https://docs.oracle.com/javase/tutorial/java/data/strings.html

Beispiele

```
String strSatz;
String strWort;
int anzahlZeichen;

strSatz = "GATCAAGGGCTTTTATATAC";
anzahlZeichen = strSatz.length();

strWort = strSatz.substring(0, anzahlZeichen / 2);
```

Aufruf von Methoden

`strSatz`	`.`	`length`	`(`				`)`
`strSatz`	`.`	`substring`	`(`	`0`	`,`	`10`	`)`
`Objekt`	`.`	`methode`	`(`	`parameterliste`			`)`

- Die Methode wird immer mit ihrem Namen aufgerufen. Der Name ist ein Platzhalter für einen bestimmten Abschnitt von Code in der Klasse.
- Mit Hilfe des Punktoperators wird die Methode mit einem Objekt verbunden. Die Methode wird auf das Objekt links vom Punktoperator angewendet.

Parameterliste einer Methode

`strSatz`	`.`	`length`	`(`				`)`
`strSatz`	`.`	`substring`	`(`	`0`	`,`	`10`	`)`
`Objekt`	`.`	`methode`	`(`	`parameterliste`			`)`

- Dem Methoden-Namen folgt eine Parameterliste.
- Die Parameterliste beginnt und endet mit den runden Klammern.
- Die Parameterliste kann leer sein.
- Die Parameterliste kann x Parameter in Abhängigkeit der Deklaration enthalten.

Parameter in der Liste

strSatz	.	substring	(	p01	,	p02	)
Objekt	.	methode	(	0	,	10	)

- Die Parameter werden als Startwerte für Attribute, Variablen etc. in einer Methode benötigt.
- Die Parameterliste kann beliebig viele Parameter in Abhängigkeit der Definition in der Klasse enthalten.
- Die Parameter werden durch Kommata getrennt.
- Der Typ eines Parameters wird bei der Definition der Methode festgelegt.

Rückgabewert der Methode

`strSatz`	`.`	`length`	`(`				`)`
`strSatz`	`.`	`substring`	`(`	`0`	`,`	`10`	`)`
`Objekt`	`.`	`methode`	`(`	`parameterliste`			`)`

- Die Methode kann einen Wert an den Aufrufer zurückgeben. Die Methode `.length()` gibt eine Ganzzahl zurück. Die Methode `.substring()` gibt einen String zurück.
- Die Art des Rückgabewertes ist abhängig von der Definition der Methode.
- Der Rückgabewert kann in einer Variablen gespeichert werden.

Anzahl der Zeichen

```
anzahlZeichen = strSatz.length();
```

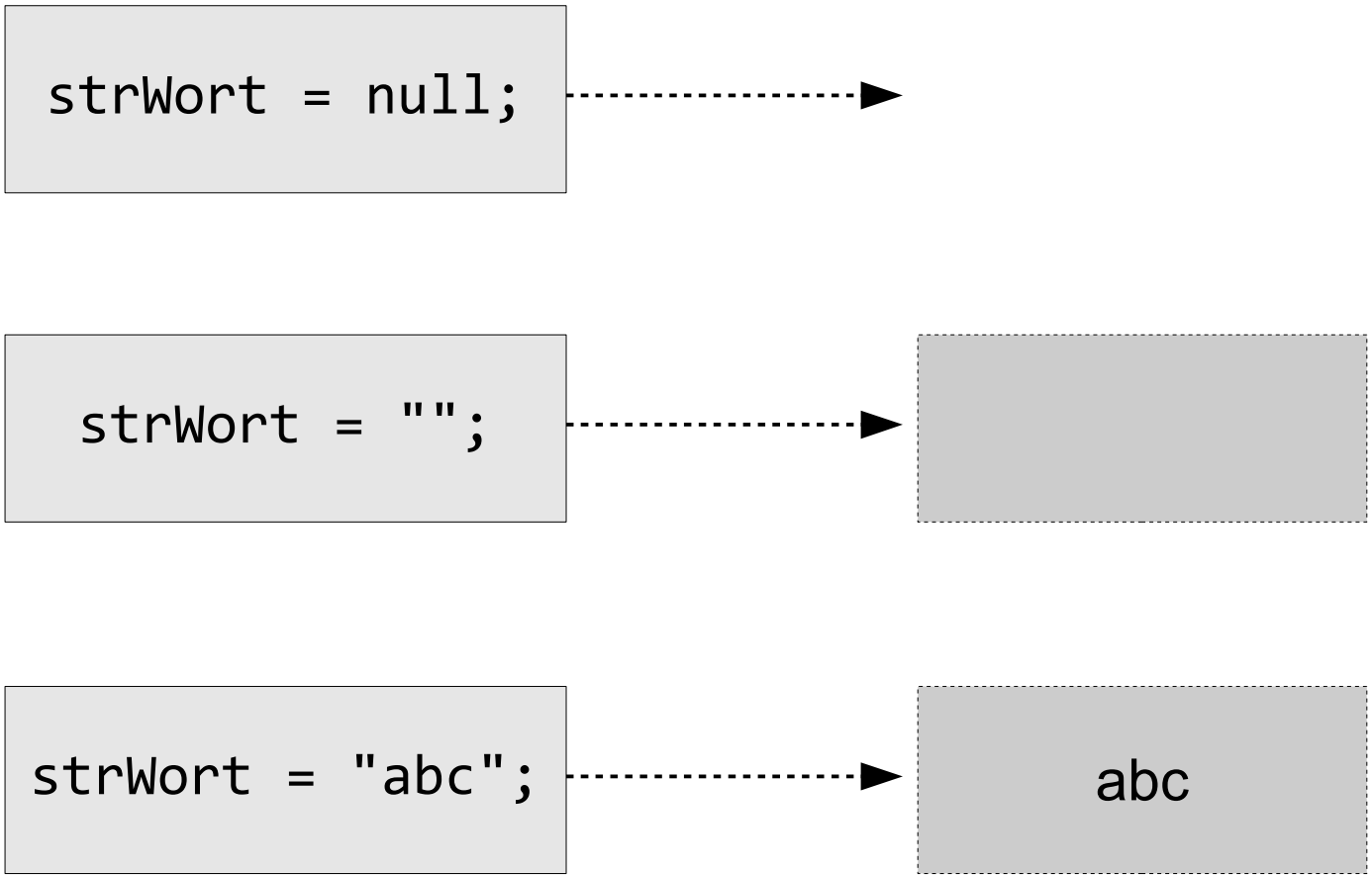
- Länge eines Strings.
- Wie viele Zeichen hat der String links vom Punktoperator?
- Der Methode werden keine Parameter übergeben.

Teilstring aus einem String

```
strSatz.substring(0, anzahlZeichen / 2);
```

- Die Methode `.substring()` liefert einen String aus dem String links vom Punktoperator zurück.
- Als erster Parameter wird der Methode der Beginn des Teilstrings übergeben. Das erste Zeichen hat den Index 0 und so weiter.
- Der zweite Parameter gibt die Länge des Teilstrings an.
- Falls der Index die Grenzen des Strings über- oder unterschreitet oder eine zu große Länge gewählt wurde, wird eine Fehler zurückgegeben.

Beispiele für Strings



Leerer String

```
String strWortLeer;
boolean resulat;

strWortLeer = "";
resulat = strWortLeer.isEmpty();
```

- Der String hat einen definierten Inhalt.
- Der String ist leer. Der String enthält keine Zeichen.
- Mit Hilfe der Methode `.isEmpty()` wird abgefragt, ob ein String leer ist.

Undefinierter String

```
String strWortUndefined;
boolean resulat;

strWortUndefined = null;
resulat = strWortUndefined.isEmpty();
```

- Der String hat einen undefinierten Inhalt.
- Der String ist „nichts“ (`null`).
- Die Objekt-Variable verweist auf kein Objekt.

Ist der String undefiniert?

```
strWortUndefined = null;

resulat = strWortUndefined.isEmpty();
resulat = strWortUndefined == null;
resulat = (strWortUndefined instanceof String)
```

- Bei einen undefinierten String liefert die Methode `.isEmpty()` einen Fehler.
- Mit Hilfe von Vergleichsoperatoren kann ein abgefragt werden, ob ein String undefiniert ist.
- Der Operator `instanceof` fragt ab, ob das gespeicherte Objekt von der Klasse String ist.

Vergleich von Strings

```
boolean resulat;

resulat = ("Abc" == "abc");

resulat = ("Abc".equals("abc"));

resulat = ("Abc".equalsIgnoreCase("abc"));
```

… mit Hilfe der Vergleichsoperatoren

```
resulat = ("Abc" == "abc");
```

- Ist die Referenz der beiden Objekte gleich?
- Verweisen die beiden Objekt-Variablen auf den gleichen Speicherplatz.

… mit Hilfe der Methoden

```
resulat = ("Abc".equals("abc"));

resulat = ("Abc".equalsIgnoreCase("abc"));
```

- Die Methode `.equals()` beachtet die Groß- und Kleinschreibung. Das Zeichen „A“ hat eine andere Codierung als das Zeichen „a“.
- Die Methode `.equalsIgnoreCase()` wird die Groß- und Kleinschreibung bei Buchstaben nicht beachtet.

Ist … im String vorhanden

```
String strSatz;
boolean resulat;

strSatz = "GATCAAGGGCTTTTATATAC";

resulat = strSatz.contains("A");
resulat = strSatz.startsWith("A");
resulat = strSatz.endsWith("A");
```

Erläuterung

- Die Methode `.contains()` trifft nur eine Aussage darüber, ob die Zeichenkette in dem String vorhanden ist.
- Die Methode `.startsWith()` überprüft, ob der String mit dem Parameter beginnt.
- Die Methode `.endsWith()` überprüft, ob der String mit dem Parameter endet.

Ist … im String an der Position vorhanden

```
String strSatz;
int resulat;

strSatz = "GATCAAGGGCTTTTATATAC";

for(resulat = strSatz.indexOf('A'); resulat > 0;)
{
  System.out.println(resulat);
  resulat = strSatz.indexOf("A", resulat +1);
}
```

Erläuterung

- Der Methode `.indexof()` wird der zu suchende String übergeben.
- In der Schleife wird der Methode als zweiten Parameter ein Index übergeben. Ab diesem Index wird der erste Parameter gesucht.
- Die Methode gibt den Fundort des zu suchenden String in dem String links vom Punktoperator zurück. Falls der zu suchende String nicht in dem String enthalten ist, wird -1 zurückgegeben.

Ausgabe eines einzelnen Zeichens

```
String strSatz;
char zeichen;

strSatz = "GATCAAGGGCTTTTATATAC";

for(int index = 0; index < strSatz.length();
      index++)
{
    zeichen = strSatz.charAt(index);
     System.out.println(zeichen);
 }
```

Erläuterung

- Der Methode `.charAt(index)` gibt ein einzelnes Zeichen vom Typ `char` zurück..
- Das erste Zeichen in einem String hat den Index 0. Das letzte Zeichen hat den Index `strSatz.length() - 1`.

Konvertierung von Zahlen nach Strings

```
int intZahl;
Double dblZahl;
String ausgabe;

intZahl = 5;
ausgabe = Integer.toString(intZahl);

dblZahl = 3.2;
ausgabe = dblZahl.toString();
```

Konvertierung in primitive Datentypen

- Eine Konvertierung von Strings in primitive Datentypen ist nur über die Wrapper-Klassen zu dem Datentyp möglich.
- Mit Hilfe der Methode `.charAt()` kann ein einzelnes Zeichen als char gespeichert werden.

Wrapper-Klassen

- Schnittstelle zwischen den primitiven Datentypen und Objekttypen in Java.
- Kapselung von primitiven Datentypen in einem einfachen Objekt.

Hinweise im Internet

- https://docs.oracle.com/javase/tutorial/java/data/numberclasses.html
- http://www.java-blog-buch.de/0606-wrapper-klassen/

Leibniz Universität Hannover

Wrapper-Klassen

Primitiver Datentyp	Wrapper-Klasse
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Deklaration der Objekt-Variablen

`Double`	`dblClass`	`;`
`Klasse`	`variablenname`	`;`

- Statt eines Datentyps wird der Name der gewünschten Wrapper-Klasse angegeben.
- Der Variablenname ist frei wählbar.

Initialisierung von Objekt-Variablen

`dblClass`	`=`	`new`	`Double`	`(`	`temp`	`)`
`objektvariable`	`=`	`new`	`Klasse`	`(`	`variable`	`)`

- Das Schlüsselwort `new` erzeugt ein neues Objekt von einer Klasse.
- In den runden Klammern werden der Klasse Werte für die Konstruktion des Objekts übergeben.
- Wrapper-Klassen wird die umzuwandelnde Variable übergeben. Falls der Wert nicht konvertiert werden kann, wird der Fehler `no suitable constructor` ausgegeben.

String → Wrapper-Klasse

```
dblClass = Double.valueOf(temperatur)
```

- Die Methode `.valueOf()` konvertiert einen String in die entsprechende Klasse um.
- Wenn der String undefiniert ist, wird eine Fehlermeldung ausgegeben.

Wrapper-Klasse → String

```
dblClass.toString()
```

- Die Methode `.toString()` konvertiert ein Objekt in ein String.

Wrapper-Klasse → double

```
dblWert = dblClass.doubleValue();
```

- Die Methode `.doubleValue()` konvertiert das Objekt in den entsprechenden primitiven Datentyp.

String → double

```
dblWert = Double.parseDouble(temperatur);
```

- Die Methode `.parseDouble(String)` konvertiert ein String über die Wrapper-Klasse in den entsprechenden primitiven Datentyp.
- Hinweis: Die Konvertierung von "3 Äpfel" oder "3,456" erzeugt den Fehler `java.lang.NumberFormatException`.

Streams (Datenströme)

- Transport von Daten.
- Input-Streams (Eingabe) lesen Daten von einer Quelle. In diesem Kurs werden die Daten von der Konsole gelesen. Die Daten werden mit Hilfe der Tastatur in die Konsole eingegeben.
- Output-Streams schreiben Daten in eine Senke. In diesem Kurs werden die Daten auf die Konsole geschrieben und damit am Bildschirm ausgegeben.

Konsolen

- Kommandozeile. Terminal.
- Die Microsoft Eingabeaufforderung wird als Konsole unter dem Betriebssystem Windows genutzt.
- Die IDE „NetBeans“ nutzt das Fenster output als „Konsole“.

Standard-Streams in Java

- `System.out`. Ausgabe auf die Konsole.
- `System.in`. Lesen von der Konsole. Mit Hilfe der Tastatur werden Zeichen auf der Konsole eingegeben.
- `System.err`. Ausgabe von Fehlermeldungen auf die Konsole. NetBeans kennzeichnet diese mit einer roten Schriftfarbe.

Zeichenorientierte Streams

- Verarbeitung von Textdaten.
- Ein- und Ausgabe von Daten, die in einem Texteditor angesehen und bearbeitet werden können.
- Arbeiten mit Text in einer bestimmten Zeichencodierung. In Java werden Daten im Unicode-Format verarbeitet.

… in Java

- Klassen zum Schreiben und Lesen von Datenströmen sind im Paket `java.io` definiert.
- `java.io.Reader` und `java.io.Writer`.
- `java.io.FileReader` und `java.io.FileWriter`.

Byteorientierte Streams

- Verarbeitung von Binärdaten.
- Die Art der Anwendung legt den Inhalt der Bytes fest.

… in Java

- Klassen zum Schreiben und Lesen von Datenströmen sind im Paket `java.io` definiert.
- `java.io.InputStream` und `java.io.OutputStream`.
- `java.io.FileInputStream` und `java.io.FileOutputStream`.

Ausgabe in die Konsole

```
String satz01 = "Mr Mousebender: Tell me, do you have
                 any cheese at all?";
String satz02 = "Henry Wenslydale: Yes.";

System.out.println(satz01);
System.out.println(satz02);

System.out.print(satz01 + '\n');
System.out.print(satz02 + '\n');
```

Erläuterung

- Beiden Methoden wird als Parameter der auszugebende Wert übergeben.
- Die Methode `.println()` beendet jede Ausgabe mit einem Zeilenumbruch.
- Die Methode `.print()` gibt den Wert so aus wie dieser übergeben wurde.

Formatierung der Ausgabe

```
System.out.printf("Temperatur im %-20s %5.2f %n",
                  "Januar", 7.6);
```

- Die Methode `.printf()` formatiert die übergebenen Variablen entsprechend der Angaben.
- Als erster Parameter wird der Methode ein Formatstring übergeben. Dieser String besteht aus nicht veränderbaren Zeichen, Platzhalter für die dynamischen Bestandteile und Formatierungszeichen zu den Platzhaltern.
- Die anderen Parameter enthalten die Werte für die dynamischen Bestandteile des Formatierungssstrings.

Formatierungszeichen

```
System.out.printf("Temperatur im %-20s %5.2f %n",
                  "Januar", 7.6);
```

- Beginn mit dem Prozentzeichen.
- Der Buchstabe kennzeichnet den Datentyp. `%s` ist ein Platzhalter für ein String. `%f` stellt eine Gleitkommazahl dar.
- In Abhängigkeit des Typs stehen weitere Formatierungsmöglichkeiten zur Verfügung.

Beispiele für Platzhalter

Platzhalter	Datentyp
%s	String
%d	Ganzzahl
%f	Gleitkommazahl
%n	New Line

- Siehe https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

Formatierungszeichen

%	schalter	breite	.	genauigkeit	typ

- Schalter. Wird der Wert rechts- oder linksbündig ausgerichtet?
- Breite. Mindestanzahl der auszugebenden Zeichen.
- Genauigkeit. Bei Gleitkommazahlen wird zum Beispiel die Anzahl der Nachkommastellen angegeben.
- Typ. Ein Buchstabe, der eine bestimmte Gruppe von Werten symbolisiert.

Beispiele für Strings

Das	Erläuterung
%s	Unformatierter String
%20s	Rechtsbündige Ausgabe. Die Angabe 20 legt die Mindestanzahl der auszugebenden Zeichen fest. Falls der String kleiner ist, wird mit Leerzeichen aufgefüllt.
%-20s	Das Minuszeichen symbolisiert eine linksbündige Ausgabe.
%%	Prozentzeichen
%n	Neue Zeile

Beispiele für Ganzzahlen

	Erläuterung
%b %B	Boolean
%d	Ganzzahl.
%4d	Mindestens 4 Stellen werden genutzt. Falls die Zahl weniger Stellen hat, wird diese mit Leerzeichen aufgefüllt.
%04d	Mindestens 4 Stellen werden genutzt. Falls die Zahl weniger Stellen hat, wird diese mit Nullen aufgefüllt.

Beispiele für Gleitkommazahlen

	Erläuterung
%f	Gleitkommazahl
%.2f	Die Gleitkommazahl wird in diesem Beispiel auf 2 Stellen nach dem Komma gerundet.
%5.2f	Rechts vom Punkt wird die Gesamtzahl der auszugebenden Zeichen angegeben. Links vom Punkt wird die Anzahl der Nachkommastellen angegeben. In diesem Beispiel würde eine Zahl mit zwei Stellen vor und nach dem Komma angezeigt. Falls die Zahl mehr Stellen benötigt, werden diese angezeigt.

Input von der Konsole

- `java.util.Scanner`. Einführung mit der JDK 5.0.
- Einlesen von Zeichen von der Tastatur, aus Dateien und Strings.
- Nutzung bei einem zeichenorientierten Stream.

Import der Klasse

```
import java.util.Scanner;
```

- Mit Hilfe des Schlüsselwortes `import` wird eine definierte Klasse in einem anderen Java-Programm eingebunden.
- Die eingebundene Klasse ist in dem Programm sichtbar und kann genutzt werden.
- Die Klasse `Scanner` ist in dem Paket `java.util` definiert. Der Punktoperator verbindet die Ordner (Pakete) mit den darin enthaltenen Dateien (Klassen). Ordner können wiederum Unterordner enthalten.

Erzeugung einer Instanz der Klasse Scanner

```
Scanner input = new Scanner(System.in);
```

- Das Schlüsselwort `new` erzeugt ein neues Objekt von der Klasse `Scanner`.
- Die Instanz wird mit Hilfe des, in den runden Klammern, übergebenen Parameter konstruiert.
- In diesem Beispiel wird als Quelle die Standard-Eingabe genutzt.

Schließen des Datenstroms

```
Scanner input = new Scanner(System.in);
input.close();
```

- Die Methode `.close()` schließt einen Datenstrom (in diesem Fall `System.in`).
- Das Objekt links vom Punktoperator ist mit einem Input-Stream verbunden. Der, mit dem Objekt verbundene Datenstrom wird geschlossen.
- Falls kein Datenstrom vorhanden ist, wird der Fehler `NoSuchElementException` ausgegeben.

Einlesen von Strings

```
strTemperatur[countMonat] = input.nextLine();
strTemperatur[countMonat] = input.next();
```

- Die Methode `.next()` liest eine Zeichenkette ein. Die Zeichenkette endet mit einem Leerzeichen.
- Die Methode `.nextLine()` liest einen String bis zum Zeile-Endezeichen ein. Das Zeilen-Endezeichen wird nicht gelesen.

Einlesen von Zahlen

```
dblTemperatur[countMonat] = input.nextDouble();
intTemperatur[countMonat] = input.nextInt();
```

- Die Methode `.nextDouble()` liest eine Gleitkommazahl vom Typ `double` ein.
- Die Methode `.nextInt()` liest eine Ganzzahl vom Typ `int` ein.

Hinweise

- Für jeden primitiven Datentyp gibt es eine entsprechende Methode.
- Die Methoden lesen entsprechend der Länderkennung ein. Eine Länderkennung „De“ liest Gleitkommazahlen ein, die als Dezimaltrennzeichen ein Komma haben. Falls die Gleitkommazahl einen Punkt als Dezimaltrennzeichen besitzt, wird ein Fehler geworfen.

Einzulesende Werte überprüfen

```
import java.util.Scanner;

public class Anweisung_Referenztyp {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        if (input.hasNextInt() == true)
        {
            strWert = input.next();
            zahl = Integer.parseInt(strWert);
            System.out.println("Ganzzahl: " + zahl);
        }
```

… überprüfen

```
if (input.hasNextInt() == true)
```

- Kann der nächste einzulesende String als Ganzzahl vom Datentyp `int` interpretiert werden?
- Wenn ja, führe die Anweisungen aus.

… in Abhängigkeit des Wertes einlesen

```
if (input.hasNextDouble() == true)
{
   dezimal = input.nextDouble();
   System.out.println("Dezimalzahl: " + dezimal);
}
```

- Kann der nächste String als Wert vom Typ `double` interpretiert werden?
- Wenn ja, lese den nächsten String entsprechend des Datentyps ein.

Falsche Eingabe

```
while(countMonat < MAXMONAT )
{
  System.out.printf("Temperaturmittelwerte %s:%n",
                    monat[countMonat]);

            while (input.hasNextDouble() == false)
            {
                System.out.println("Falsche Eingabe! ");
                System.out.printf("Temperaturmittelwerte %s:%n",
                                  monat[countMonat]);
                input.next();
            }

            dblTemperatur[countMonat] = input.nextDouble();

            countMonat++;
}
```

Einlesen von Umlauten

```
String strUmlaute = "";
Scanner input = new Scanner(System.in, "iso-8859-1");

System.out.println("Umlaute: ");
strUmlaute = input.nextLine();

input.close();
```

- In diesem Kurs wird die Konsole *Output* der IDE NetBeans genutzt.
- In diesem Fenster werden von der Konsole eingelesene Umlaute wie zum Beispiel ä, ü, ö und so weiter als Rechtecke ausgegeben.

Festlegung des Zeichensatzes

```
Scanner input = new Scanner(System.in, "iso-8859-1");
```

- Mit Hilfe von `new` wird ein neues Objekt vom Typ `Scanner` erzeugt.
- Für die Konstruktion eines neuen Scanners werden die angegebenen Werte in der Parameterliste benötigt. Die Parameterliste wird durch die runden Klammern begrenzt.
- Die einzelnen Parameter in der Liste werden durch ein Komma getrennt.

Parameter bei der Konstruktion

```
Scanner input = new Scanner(System.in, "iso-8859-1");
```

- Als erster Parameter wird eine Vorlage für die Konstruktion an den Konstrukteur übergeben. In diesem Beispiel wird die Standardeingabe (`System.in`) als Vorlage übergeben.
- Als zweiter Parameter wird der Zeichensatz angegeben, der von der Konsole oder Datei genutzt wird. In diesem Beispiel wird für das Schreiben in die Konsole der Zeichensatz ISO-8859-1 genutzt.