Java Programmierung

main Klasse:

```
public static void main(String[] args){
   //Code
}
```

Bildschirmausgabe:

```
System.out.print("Text ohne Zeilenumbruch");
System.out.println("Text mit Zeilenumbruch");
System.out.printf("Text mit anschließendem Zeilenumbruch%n");
```

Primitive Datentypen:

- Haben immer einen Wert

Datentyp	Größe	Beschreibung
byte	1 byte	Speichert ganze Zahlen von -128 bis 127
short	2 bytes	Speichert ganze Zahlen von -32 bis 32
int	4 bytes	Speichert ganze Zahlen von -2 147 483 648 bis 2 147 483 647
long	8 bytes	Speichert ganze Zahlen von -9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807
float	4 bytes	Speichert Dezimalzahlen mit 6 bis 7 Nachkommastellen
double	8 bytes	Speichert Dezimalzahlen mit bis zu 15 Nachkommastellen
boolean	1 bit	Speichert wahrheitswert true / false
char	2 bytes	Speichert einzelne Zeichen der ASCII Tabelle

Komplexe Datentypen:

- Benutzt man für ArrayList

Primitive Datentyp	Komplexer Datentyp
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

String

Anweisung	Beschreibung
Variable1.equals(variable2)	Vergleich von 2 Strings
.length()	Länge des Strings ermitteln
.toUpperCase()	Alles als Großbuchstaben ausgeben
.toLowerCase()	Alles als kleinbuchstaben ausgeben
.indexOf()	Gibt position von dem ersten Zeichen eines gesuchten Textes wieder
.concat()	Um 2 Strings mit leerzeichen zu verbinden
+	Um 2 Strings ohne Leerzeichen zu verbinden
\’	In String Texteingabe benutzen wenn man Hochzeichen schreiben will
\”	In String Texteingabe benutzen wenn man Anführungszeichen schreiben will -> Escaping

\\	In String Texteingabe nutzen wenn man Backslash schreiben möchte
\n	Neue Zeile
\r	Zurück zum Anfang der Zeile gehen/Carriage Return
\t	Tab
\b	Eins zurück gehen /Backspace
\f	Form Feed/ Zeilenvorschub

Operatoren

Arithmetische Operatoren

Operator	Name	Beschreibung
+	Addition	Addiert 2 Werte
-	Subtraktion	Subtrahiert 2 Werte
*	Multiplikation	Multipliziert 2 Werte
/	Division	Dividiert 2 Werte
%	Modulo	Rest einer Division wird ausgegeben
++	Inkrementieren	Variable wird um 1 erhöht x++ : Inkrementiere x und gib alten Wert an Ausdruck zurück ++x : Inkrementiere x und gibt neuen Wert zurück
--	Dekrementieren	Variable wird um 1 verniedrigt
=	Gleich	Weißt Wert einer Variablen zu
+=	Bsp: x += 5	x=x+5

Vergleichsoperatoren

Operator	Name	Beschreibung
==	Ist gleich	Identisch Wert Bei Objekten (Strings): variable1.equals(variable2)
!=	Ungleich	Nicht identischer Wert
>	Größer	Wert ist größer als Vergleichswert
<	Kleiner	Wert ist kleiner als Vergleichswert
>=	Größer gleich	Wert ist größer gleich Vergleichswert
<=	Kleiner gleich	Wert ist Kleiner gleich Vergleichswert

Logische Operatoren

Operator	Name	Beschreibung
&&	Logisches UND	Ist wahr wenn beide Aussagen wahr sind

||	Logisches ODER	Ist wahr wenn eine der beiden Aussagen wahr ist
!	Logisches NICHT	Invertiert das Ergebnis

Priorität von Operatoren

Priorität	Operator
1	( )
2	++, --, Vorzeichenplus, Vorzeichenminus, ~, !, (Datentyp)
3	*, /, %
4	+, -
5	<, <=, >, >=
6	==, !=
7	&
8	^
9	|
10	&&
11	||

Anweisungen

Anweisung	Beschreibung

If (Bedingung){
//Code wenn Bedingung wahr
} else {
//Code wenn Bedingung nicht wahr ist
}
	If Anweisung
	Wenn if Bedingung wahr ist wird die erste Anweisung ausgeführt, ansonsten wird die zweite Anweisung hinter else ausgeführt

If (1.Bedingung) {
//Code wenn Bedingung wahr ist
} else if (2.Bedingung) {
//Code wenn 1. Bedingung falsch ist und 2. Bedingung wahr ist
} else {
//Code wenn 1. und 2. Bedingung nicht wahr ist
}
	Verschachtelte If- else Bedingung

Return	Beendet if - Codeblock
Variable = (Bedingung) ? AnweisungWahr : AnweisungFalsch	Kurze if else Anweisung

Switch (Datenobjekt/ Variable) {
Case x:
//Code;
Break;
Case y:
//Code;
Break;
Case z:
//Code;
Break;
default:
//Code;
Break;
}
	Switch Case
	Wenn man mehrere Ausgangsmöglichkeiten und Verzweigungen hat nutzt man switch case
	Default wenn es kein Case gibt der übereinstimmt
	
	Nur für Datentypen String, Char, Enum, Int
	
	Break beendet das Case – sonst geht es direkt in das nächste Case den Code über

While (Bedingung) {
//Code
}
	While Schleife
	Wird ausgeführt solange Bedingung wahr ist
	Vergleich mit "=="

do {
  //Code
} while(Bedingung)
	Do While Schleife
	Wird auf jeden Fall 1x ausgeführt auch wenn Bedingung nicht wahr ist und wird nach jedem Durchlauf auf Bedingung geprüft

For (Statement1; Statement2; Statement3) {
  //Code
}
	For Schleife
	Statement1: Durchlaufvariable initialisiert und deklariert nur 1x
	Statement2: Bedingung nach jedem Durchlauf
	Statement3: Durchlaufvariable hochgezählt nach jedem Durchlauf

For (Datentyp variablenName : arrayName) {
  //code
}
	For each Schleife
	Durchläuft ein Array und kann Inhalt von Array ausgeben
	Gibt alle Inhalte vom Array wieder

Break;	Beendet komplette Schleife / bricht Schleife ab
Continue;	Überspringt diesen Schleifendurchlauf / bricht den Schleifendurchgang ab

Arrays

- Größer von Anfang an klar

Anweisung	Beschreibung
Datentyp [ ] arrayName	Array deklarieren
Datentyp [ ] arrayName = new Datentyp [längeDesArrays]	Array mit fester Länge erstellen ohne Elemente im Array (Inhalt) Datentyp int – muss man ausschreiben als Integer
Datentyp [ ] arrayName = { Inhalt, …}	Array deklarieren und initialisieren
arrayName [index]	Zugriff auf Wert index des Arrays – index beginnt bei 0
arrayName.length	Länge des Arrays
Datentyp [ ] [ ] arrayName = {{inhalt } {inhalt}}	2 Dimensionales Array deklarieren und initialisieren

ArrayList

- Hat dynamische Größe

Import java.util.ArrayList;	Benötigt import damit ArrayList funktioniert
ArrayList <Datentyp> arrayListName = new ArrayList < > ( );	Datentyp mit komplexen Datentypen
arrayListName.add(index, Inhalt);	Elemente zu Array Liste hinzufügen man kann Index angeben
arrayListName.remove(index, Inhalt);	Kann Element an bestimmter Stelle entfernen
arrayListName.get(index)	Ein Array Element von einem bestimmten Index ausgeben
arrayListName.set(Index, Inhalt)	Element überschreiben mit .set (stelle, Inhalt)
arrayListName.remove(0)	Element des Arrays an bestimmten Stelle entfernen
arrayListName.clear()	Alle Elemente des Arrays entfernen
arrayListName.size()	Herausfinden wie viele Elemente die ArrayList hat
arrayListName.contains(Inhalt)	Herausfinden ob ArrayList bestimmten Wert enthält -> gibt true / false heraus

for (int i = 0; i < arrayListName.size(); i++) {
  System.out.println(arrayListName.get(i));
  }
	For Schleife Array Liste ausgeben

for (String i : arrayListName) {
  System.out.println(i);
  }
	Mit for-each Schleife Array Liste ausgeben

Klassen

- Klassen beginnen mit Großbuchstaben – heißen gleich wie Dateiname
- Beginnen mit Schlüsselwort class
-

Attribute

- Sind Datenobjekte / werden wie Variablen oder Konstanten definiert – Angabe des Zugriffsrechts legt Sichtbarkeit fest

Methoden

- Methoden beginnen mit kleinbuchstaben
- Können einen Rückgabewert oder keinen Rückgabewert besitzen
    - Datentyp des Rückgabewertes muss angegeben sein und Anweisung „return“ besitzen
- Methoden ohne Rückgabewert werden mit „void“ gekennzeichnet

Konstruktor

- Methoden die zur Initialisierung von Objekten verwendet werden
- Heißen wie Klassen und können beliebig viele Parameter haben

Anweisung	Beschreibung
Access Modifiers - Kontrolliert Zugriff Level	
Static	Kann darauf zugegriffen werden, ohne ein Objekt der Klasse zu erstellen
Public	Zugriff auf Klasse ist von jeder anderen Klasse möglich
Default	Zugriff auf Klasse ist nur von Klassen im selbem package möglich
Private	Zugriff auf Code nur in derselben Klasse möglich - Get(mit return) und set notwendig um Methoden zu schreiben
Protected	Zugriff auf Code ist unterklassen möglich (und im selben Package)
Non – Access Modifiers – für Klassen	
Final	Klasse kann nicht vererbt werden von anderen Klassen - Können nicht abgeleitet werden
Abstract	Klasse kann nicht verwendet werden um Objekte zu erstellen - Können nicht instanziert werden
Non – Access Modifiers – für Attribute und Methoden	
Final	Attribute und Methoden können nicht überschrieben werden - Können nicht überschrieben werden
Static	Attribute und Methoden gehören zu der Klasse und nicht zum Objekt - Wert wird 1x zugewiesen und besitzt für alle Objekte der Klasse gleichen Wert
Abstract	Kann nur in abstract Klassen verwendet werden und nur von Methoden - In abstrakten Klassen definiert - Haben keinen Methodenrumpf und müssen von abgeleiteten Klassen der abstrakten Klasse überschrieben werden
Transient	Attribute und Methoden werden übersprungen

Objekte
- Werden mithilfe von „new“ erstellt – danach muss immer Konstruktor der Klasse kommen

Polymorphie
- Methoden die gleich genannt werden und unterschiedliche Ergebnisse liefern können
- Sie können keine unterschiedlichen Rückgabewerte besitzen

Upcast
- Objektreferenz der Unterklasse in Objektreferenz der Oberklasse umgewandelt – „extends“ schlüsselwort

Downcast
- Objektreferenz der Oberklasse wird in Objektreferenz der Unterklasse umgewandelt – Typ muss explizit angegeben werden

Instanceof
- Prüfung ob Objektreferenz zuweisungskompatibel zu Klasse ist – dann wenn die Klasse des referenzierten Objekts in Vererbungsbeziehung zur Klasse steht

Java Enums
- Enum ist eine spezielle Klasse welche Konstanten repräsentiert
- Sind public, static und final
- Benutzt bei Werten die sich nicht verändern

Bibliotheken

Anweisung	Beschreibung
Import package.name.*	Gesamte Bibliothek wird importiert
Import package.name.Class	Nur einzelne Klasse wird importiert

Scanner

Anweisung	Beschreibung
`import java.util.Scanner;`	Importieren der Scanner Klasse
Scanner sc = new Scanner(System.in);	Variable als Scanner deklarieren und initialisieren
sc.next() variablenName.charAt(x);	Ließt Zeichenkette bis Leerzeichen ein ‚sc‘ – name des Scanners .next() ließt Strings ein aber nur Zeichenfolgen und keine einzelnen Zeichen charAt(0) – ließt ersten Wert (char) der von String eingelesenen Zeichenfolge aus
sc.nextDouble()	Liest double Werte ein
sc.nextInt()	Liest Int Werte ein
sc.nextLine()	Liest komplette Zeile ein
sc.nextBoolean()	Liest wharheitswerte ein
sc.nextByte()	Liest nächten byte ein vom nutzer eingegeben
sc.nextFloat()	Liest Float wert ein
sc.nextLong()	Liest long wert ein
sc.nextShort()	Liest short Wert ein
sc.close()	!!Scanner immer schließen

Random

Anweisung	Beschreibung
`import java.util.Random;`	Importieren der Random Klasse
Random randomVariable = new Random( );	Zum initialisieren des Zufallszählers
randomVariable.nextInt(int Grenzwert);	Zufallszahl zwischen 0 und einer Zahl kleiner als Grenzwert ermittelt

Dates

Anweisung	Beschreibung
`import java.time.LocalDate;`	Datum
`import java.time.LocalTime;`	Zeit
`import java.time.LocalDateTime;`	Datum und Zeit

`import java.time.DateTimeFormatter;`
DateTimeFormatter myFormat =
DateTimeFormatter.ofPattern("dd-MM-yyyy
HH:mm:ss");
String formattedDate =
myTimeDate.format(myFormat);
System.out.println(formattedDate);
	Datum und Zeit formatieren

.ofPattern()	Akzeptiert verschiedene Formate yyyy-MM-dd dd/MM/yyyy dd-MMM-yyyy E, MMM dd yyyy

Methoden

Klasse Math

Anweisung	Beschreibung
Math.max(x,y)	Findet den Höchsten Wert
Math.min(x,y)	Findet den niedrigsten Wert
Math.sqrt(x)	Berechnet die Wurzel
Math.abs(x)	Gibt den positive Wert aus - Betrag
Math.pow(x, Hochzahl)	Gibt Potenz von Wert aus
Math.random()	Gibt eine Zufallszahl zwischen 0.0 und 1.0 aus