Java Grundlagen Sheet

(Stand WS 16/17)

Datentypen:

- Ganze Zahlen
  - `byte` — `byte b = 1;` — 08 Bit — -2^7 bis 2^7-1
  - `short` — `short s = 5;` — 16 Bit — -2^{15} bis $2^{15}-1$
  - `int` — `int i = 10;` — 32 Bit — -2^{31} bis $2^{31}-1$
  - `long` — `long l = 20L;` — 64 Bit — -2^{63} bis $2^{63}-1$
- Gebrochene Zahlen
  - `float` — `float f = 1.3F;` — 32 Bit — Einfache Genauigkeit
  - `double` — `double d = 1.3;` — 64 Bit — Doppelte Genauigkeit
- Wahrheitswerte
  - `boolean` — `boolean b = true;` — 01 Bit — Werte: `true` oder `false`
- Zeichen
  - `char` — `char a = 'a';` — 16 Bit — ASCII Code/UNICODE
- Text
  - `String` — `String text = "Hallo";`

Operatoren:

- Sonstige
  - Zuweisungsoperator — `=` — `int b = 1;` — `// 1`
  - Zeilenendoperator — `;` — `;`
- Arithmetische
  - Addition — `+` — `int i = 10 + 5;` — `// 15`
    `String text = "Hall" + "o";` — `// Hallo`
  - Subtraktion — `-` — `int i = 10 - 5;` — `// 5`
  - Multiplikation — `*` — `int i = 10 * 5;` — `// 50`
  - Division — `/` — `int i = 8 / 3;` — `// 2`
    `double d = 8.0 / 3.0;` — `// 2.666666666`
  - Modulo — `%` — `int i = 5 % 2;` — `// 1`
- Vergleiche
  - Ist-Gleich (Äquivalent) — `==` — `boolean b = 2 == 2;` — `// true`
  - Ungleich — `!=` — `boolean b = 2 ! = 2;` — `// false`
  - Echt-Kleiner — `<` — `boolean b = 1 < 2;` — `// true`
  - Kleiner-Gleich — `<=` — `boolean b = 2 <= 2;` — `// true`
  - Echt-Größer — `>` — `boolean b = 1 > 2;` — `// false`
  - Größer-Gleich — `>=` — `boolean b = 2 >= 2;` — `// true`
- Boolsche
  - Negation(Umkehrung) — `!` — `boolean b = ! false;` — `// true`
  - UND — `&&` — `boolean b = true && true;` — `// true`
  - ODER — `||` — `boolean b = false || false;` — `// false`
  - ENTWEDER-ODER(XOR) — `^` — `boolean b = true ^ false;` — `// true`

Verzweigungen:

Einfache if-Anweisung:

```
if (Bedingung) {
	// Wenn Bedingung true ist
}
```

if-else-Anweisung:

```
if (Bedingung) {
	// Wenn Bedingung true ist
}
else {
	// Wenn Bedingung false ist
}
```

if-else if-else-Anweisung:

```
if (Bedingung) {
	// Wenn Bedingung true ist
}
else {
	if (AndereBedingung) {
	   // Wenn Bedingung false ist und AndereBedingung true ist
	} else {
	   // Wenn Bedingung false ist und AndereBedingung false ist
	}
}
```

n-seitige Verzweigung: switch-case

```
switch (variable) {
	case 1: System.out.println(„variable ist “ + 1);
		break;
	case 2: System.out.println(„variable ist “ + 2);
		break;
	default: System.out.println(„variable ist etwas anderes“);
		break;
}
```

Wiederholungen / Schleifen:

Kopfgesteuert: while-do

```
	int intVariable = 0;
	while(intVariable < 10) {
	    System.out.println(intVariable);
	    intVariable++;
	}
// gibt die Zahlen von 0 bis 9 aus
```

Fußgesteuert: do-while

```
	int intVariable = 0;
	do {
	    System.out.println(intVariable);
	    intVariable++;
	} while(intVariable < 10);
// gibt die Zahlen von 0 bis 9 aus
```

Zählschleife: for-Schleife

```
for(Initialisierung der Zählvariable ; Bedingung ; Verändern der Zählvariable){
	//code
}

	for (int intVariable = 0; intVariable < 10; intVariable ++) {
		System.out.println(intVariable);
	}
// gibt die Zahlen von 0 bis 9 aus
```

Methoden:

Prinzipieller Aufbau:

```
Sichtbarkeit [static] Rückgabetyp Name(Parameter) {
    //weiterer Quellcode
    return Rückgabe;
}
```

- Die **Sichtbarkeit** kann entweder **private**, **public**, **protected** oder weggelassen werden.
- **static**: Kann gesetzt sein oder nicht. Innerhalb von **static** Methoden können direkt nur **static** Methoden aufgerufen werden.
- Der **Rückgabetyp** kann ein normaler **Datentyp** (wie int), eine **Klasse** (wie String), ein **Array**, oder **void** sein.
    - **void** bedeutet keine Rückgabe. Also auch kein **return**.
- Der **Name** der Methode wird klein geschrieben und sollte beschreiben was die Methode macht.
- **Parameter**: Eine Methode kann **0** bis **n** Parameter haben. Die einzelnen Parameter werden durch Komma abgetrennt aufgelistet. Die Parameter bestehen aus **Datentyp** und einen **Namen**.

Beispiel:

```
public static int summe(int summand1, int summand2){
    return summand1 + summand2;
}

//Verwendung der Methode:
public static void main(String[] args) {
    int zahl1 = 3;
    int zahl2 = 4;
    int sum = summe(zahl1,zahl2);   // sum hat den Wert 7
}
```

Casting:

- Wandelt **Basisdatentypen** sinnvoll ineinander um.
- Dabei kann Genauigkeit verloren gehen.

```
(neuer Datentyp) wertZumUmWandeln;
```

Beispiel double zu int casten:

```
double doubleWert = 3.84;
int integerWert = (int) doubleWert; // integerWert = 3
```

Beispiel char zu int casten:

```
char buchstabe='a';
System.out.println((int)buchstabe); // 97 siehe ASCII Tabelle
```

Arrays (Felder):

- Speichert mehrere Variablen **gleichen** Datentyps.
- Arrays haben eine **feste** Länge, die beim Erstellen des Arrays festgelegt werden muss.
- Die Nummerierung von Arrays beginnt immer bei 0.

Erstellen eines Arrays:

```
int[] myArray = new int[5];          // erstellt ein Array der Länge 5
```

oder

```
int[] myArray2 = {1,4,9};  //erstellt ein Array der Länge 3 mit den Werten 1,4,9
```

Zugriff auf das Array:

```
int wert = myArray2[1];              // wert ist 4
```

Länge eines Array:

```
int length= myArray2.length;         // 3
```

Exceptions:

Prinzipieller Aufbau:

```
try {
        // Kritische Anweisungen, die schief gehen könnten
}
catch (fehler) { //Fehler der überwacht wird
        /* Anweisungen, die nur ausgeführt werden,
           wenn dieser Fehler auftritt */
}
```

Beispiel: (Abfangen falscher Nutzereingaben)

```
Scanner scn = new Scanner(System.in);
System.out.println("Gib eine ganze Zahl ein");
try {
        int zahl = scn.nextInt();
} catch (Exception e) {
        System.out.println("Das war keine Zahl");
}
```