

# JavaC - CheatSheet

## Allgemein

### Primitive Datentypen

Typ	Beschreibung	Wertebereich	Wrapper-Klasse
<b>boolean</b>	Boolescher Wert	true, false	Boolean
<b>char</b>	Textzeichen (UTF16)	'a', 'B', '0', 'é' etc.	Character
<b>byte</b>	Ganzzahl (8 Bit)	-128 bis 127	Byte
<b>short</b>	Ganzzahl (16 Bit)	-32'768 bis 32'767	Short
<b>int</b>	Ganzzahl (32 Bit)	-2 <sup>31</sup> bis 2 <sup>31</sup> -1	Integer
<b>long</b>	Ganzzahl (64 Bit)	-2 <sup>63</sup> bis 2 <sup>63</sup> -1, 1L (L Suffix)	Long
<b>float</b>	Gleitkommazahl(32 Bit)	0.1f, 2e4f (2*10 <sup>4</sup> )	Float
<b>double</b>	Gleitkommazahl(64 Bit)	0.1, 2e4	Double

**Überlauf bzw. Unterlauf** ist in Java definiert. Bei einem Überlauf wird ganz unten weitergezählt, bei einem Unterlauf wird von ganz oben fortgesetzt. Bei Gleitkommazahlen wird führt ein Über-/Unterlauf zu POSITIVE\_INFINITY bzw. NEGATIVE\_INFINITY.

### Explizite Typkonversion

Nur C-Style Cast: (int)3.5; → 3

**TO DO: evtl. noch Arrays und Mehrdimensionale Arrays**

**TO DO: evtl. "var" beschreiben (folie 7, Woche 4)**

## Collections

Collection sind Datenstrukturen für Gruppen von Elementen und fordern einen Import aus dem Paket java.util

### List

Eine Liste ist eine Folge von Elementen und kann wie folgt definiert werden:

```
ArrayList<Obj> name = new ArrayList<Obj>();
ArrayList<Obj> name = new ArrayList<>();
var name = new ArrayList<Obj>();
List<Obj> name = new ArrayList<Obj>();
List<Obj> name = new ArrayList<>();
```

Einige nützliche Operationen mit Listen:

```
var stringList = new ArrayList<String>();

stringList.add("one");           // add at the end
stringList.add(0, "two");        // insert at pos 0
// add -> umkopieren in doppelt so grosses Array (gibt leere plaetze)
String x = stringList.get(1);    // get at pos 1
stringList.set(0, "three");      // replace at pos 0
stringList.remove("two");        // removes the FIRST "two" in List
stringList.remove(1);            // remove at pos 1
// remove -> alles dahinter wird nach vorne geschoben
stringList.contains("three");    // true -> "three" is in List, else -> false
long size = stringList.size();   // get size (number of Elements)
```

### Iteration mit Enhanced for

Besucht jedes Element in einer Collection:

```
for (String s: stringList){
    System.out.println(s);
}
```

### Set

Ein Set ist eine Menge von Elementen, in welchem jedes Element genau einmal vorkommt und wird wie folgt verwendet:

```
Set<String> carModels = new HashSet<>();
carModels.add("Ferrari");
carModels.add("Maserati");
carModels.add("Lamborghini");
carModels.add("Ferrari"); // already present (no effect)
if(carModels.contains("Volkswagen")){...}
```

## Map

Abbildung Schlüssel → Werte

```
Map<Integer, Student> map = new HashMap<>();

Student st1 = new Student("Andrea", "Meier");
Student st2 = new Student("Bertha", "Mueller");
Student st3 = new Student("Clara", "Schneider");

map.put(20000, st1);    // Bei gleichem Schluessel
map.put(70000, st2);    // wird der Wert ueberschrieben
map.put(13000, st3);    // -> nur ein Key pro Map

Student st = map.get(12345);    // Finden nach Schluessel (sehr effizient)

for(Student s : map.values()){  // .values() gibt Collection aller Werte
    System.out.println(s);
}
```

## Wrapper-Klassen

Collections (bzw. alle Generics) nehmen nur Referenzen, welche primitive Datentypen nicht bringen. Um dennoch int's oder double's in Listen zu speichern, gibt es sogenannte Wrapper-Klassen.

```
// Implizites Boxing (Integer.valueOf(123))
Integer wrapper = 123;

// Implizites Unboxing (wrapper.intValue())
int value = wrapper;
```

