Aufgabe 1:
a)
Binär: 1101011
Dezimal: 107
Oktal: 0153 (0=Java Literal für Oktal)
Hex: 0x6B (0x=Java Literal für Hex)
Gleitkomma: 107. ; 0x6Bp0 (Gibt mehrer Lösungen, da nicht klar ist welches Zahlensystem verlangt wird)

b)
```
     000000000|1101011
1er: 111111111|0010100
2er: 111111111|0010101
```

c)
1/3 * 2 = 2/3 Rest 0
2/3 * 2 = 1/3 Rest 1 (Ergebnis zeigt auf den Anfang; periodisch)

2/3 = 0,01 (01 mit Überstrich für die Periodizität)

Aufgabe 2:
a)
1/2 = 0 (int)
3-4.5 = -1.5 (double)
6 == 7 = false (boolean)
"89" + '0' = "890" (String)

b)
int var1 = 2;
String var2 = "Hallo Welt!"; (String ist ein Referenztyp; Siehe 2-18)

c)
x = new Beispiel(); (Java führt einen Upcast aus und wandelt es vom Datentyp Beispiel in Object um)
x = 1; (Zuerst wird ein Autoboxing int=>Integer ausgeführt und dann wieder ein Upcast)

d)
```
Integer.toString(1234);
new Integer(1234).toString();
```

Aufgabe 3:
a)
```
a = (++b + c--) >>> 1
```

1. ++b
2. c--
3. b + c
4. >>> 1
5. a = ...

b) a=25; b=21; c=29;

Aufgabe 4:
a)
```
public static Integer max(final Integer n, final Integer m){
    if (n.intValue() > m.intValue()) {
        return n;
    } else {
        return m;
    }
}
```

b)
```
if (zahlen.length = 0) {
    m = null;
} elseif (zahlen.length = 1) {
    m = zahlen[0];
} else {
    m = zahlen[0];
    for (int i = 1; i < zahlen.length; ++i) {
        n = max(m, zahlen[i]);
    }
}
```

c)
```
int i = 1;
while (i < zahlen.length) {
    m = max(m, n);
    ++i;
}
```

d)
max() => null (2. Max-Funktion)
max(1) => 1 (2. Max-Funktion)
max(2,3) => 3 (1. Max-Funktion)
max(4,5,6) => 6 (2. Max-Funktion)

e)
new Integer[] = {4,5,6} (Alle Übergaben werden in ein neu erstelltes Int-Array übergeben)

Aufgabe 5:
a) aufgabe5.Klasse.java

b)
```
public final class Klasse{
    private Klasse() {}
    public static int methode() {
        return 1;
    }
    public statuc int methode(final int i) {
        return i;
    }
```

```
    public final static int KONSTANTE = 2;
    public static int variable;
}
```

c)
Überladen bedeutet, das eine Funktion mehrfach mit dem selben Namen definiert werden kann,
jedoch muss sie entweder eine andere Anzahl von Übergabeparameter oder der Übergabeparameter muss einen anderen Datentyp haben.

d)

```
import aufgabe5.Klasse;
import static aufgabe5.Klasse.KONSTANTE;
```

Aufgabe 6:
a)

```
public final class Aufgabe {
    private String t;
    private int[] p;

    public Aufgabe(final String t, final int[] p){

        this.validate(p);
        this.t = t;
        this.p = p;
    }

    private void validate(final int[] p) {

        if (p == null || p.length <= 0) {
            throw new IllegalArgumentException();
        }

        for (int i : p) {

            if (i <= 0) {
                throw new IllegalArgumentException();
            }
        }
    }

    public int getAnzahlTeile() {
        return this.p.length;
    }

    public String getTitel() {
        return this.t;
    }

    public String getGesamtPunkte() {
        int gesamt = 0;
        for (int i : p) {
```

```
            gesamt+= i;
        }
        return gesamt;
    }

}
```

b)
"this" verweist auf das aktuelle initialisierte Objekt indem sich die Methode befindern in der this aufgerufen wird.
"this" kann nur in Instanzmethoden benutzt werden, nicht in statischen Methoden.
"this" ist immer der erste Übergabeparameter bei einer Instanzmethode (unsichtbar).

c) `new StringBuilder().append("Aufgabe ").append((i + 1)).toString();`

Aufgabe 7:
Kapselung: private/public int variable;
Vererbung: public final class Unterklasse extends Oberklasse() {...}
Polymorphie: Object x = new Integer(2);
dynamische Bindung: new Integer(1234).intValue(); (Methode intValue ist auch in der Oberklasse Number definiert und wird erst zur Laufzeit erkannt, welches intValue aufgerufen wird)

Aufgabe 8:

```
import java.util.Comparator<String>;

public final class GermanComparator implements Comparator<String>{
    public GermanComparator() {}

    private String normalize(final String s) {
        StringBuilder b = new StringBuilder();

        for(int i = 0; i < s.length(); ++i) {
            char c = s.charAt(i);
            c = Character.toUpperCase(c);

            switch (c) {
                case 'Ä':
                    b.append('A');
                    break;
                case 'Ö':
                    b.append('O');
                    break;
                case 'Ü':
                    b.append('Ü');
                    break;
                case 'ß':
                    b.append("SS");
                    break;
                default:
                    b.append(c);
```

```
            }
        }

        return b.toString();
    }

    @Override
    public int compare(final String s1, final String s2) {
        String n1 = this.normalize(s1);
        String n2 = this.normalize(s2);

        return n1.compareTo(n2);
    }
}
```