# Übungsaufgaben Präsenzübung

## 1 Java-Basics[1]

### 1.1 Operatoren

Schreibe die Ausgabe jeder Zeile als Kommentar hinter den Code.

```
1   public static void main(String[] args) {
2      boolean a = true;
3      boolean b = false;
4      boolean c = true;
5
6      System.out.println(!a && b); //
7      System.out.println(!a || !c); //
8      c = !a | !c;
9      System.out.println(!c ^ b); //
10     System.out.println(!(a | b) == (!a) & (!b)); //
11     b = c & true;
12     System.out.println(b ? a : !a); //
13  }
```

### 1.2 Operatoren, Datentypen, Typumwandlung

Schreibe die Ausgabe jeder Zeile als Kommentar hinter den Code.

```
1   public static void main(String[] args) {
2      int i = 5;
3      int n = i++%5;
4      System.out.println(i%5); //
5      System.out.println(n); //
6      System.out.println(8^4); //
7      System.out.println((1<<5) + 10); //
8      i = Integer.MIN_VALUE; //-2147483648;
9      i-= 1;
10     System.out.println(i); //
11  }
```

[1] Evtl. übersteigt die Schwierigkeit dieser Aufgabe die Anforderungen der Präsenzübung

## 1.3 Operatoren, Datentypen, Typumwandlung

Korrigiere folgenden Code, sodass er fehlerfrei kompiliert werden kann.

```
1  public static void main(String[] args) {
2     float a = 1.034;
3     float b = 1.52;
4     float c = a+b;
5
6     System.out.println(c);
7     System.out.println("A"+125 == 190);
8
9     byte d = 30;
10    byte e = 12;
11    byte f = d + e;
12 }
```

## 1.4 Klassen, Methoden, Sichtbarkeit, Überschattung

Schreibe die Ausgabe jeder Zeile als Kommentar hinter den Code.

```
1  public class ClassA {
2     private int x = 5;
3     public static int y = 6;
4
5     public static void main(String[] args) {
6        int x = 1;
7        int y = 7;
8        ClassA inst = new ClassA();
9        System.out.println(x);
10       System.out.println(y);
11
12       inst.print(x);
13
14       System.out.println(x);
15       System.out.println(ClassA.y+y);
16
17       inst.print(y);
18
19       System.out.println(x);
20       System.out.println(ClassA.y);
21 }
22
23    public void print(int x) {
24       System.out.println(++x);
25       x = 3;
26       System.out.println(x);
27       int y = 4;
28       System.out.println(y);
29       System.out.println(this.x);
30    }
31 }
```

### 1.5 Klassen, Methoden, Sichtbarkeit, Überschattung

Ergänze die folgende Klasse so, dass gilt: `MAX_ID` ist eine öffentliche, statische Konstante, welche mit den Wert 100 initialisiert wird, `characters` ist ein privates Attribut, das ein Feld (Array) von Zeichen enthält und mit den Werten `'A'`, `'B'`, `'C'` initialisiert wird.

```
1  public class Main {
2        MAX_ID =
3        characters =
4  }
```

## 2 Kontrollfluss

### 2.1 Schleifen

Schreibe die Methode `print()` erneut auf, verwende jetzt aber eine `for`-Schleife. Die Ausgabe der Methoden soll identisch sein.

```
1  public static void print(int count){
2     do {
3        System.out.println("Count is: " + count);
4        count++;
5     } while (count < 10);
6  }
7  public static void print(){
8
9
10
11
12 }
```

### 2.2 Schleifen, Arrays

Ergänze die folgenden Methode, sodass alle Werte des Felds `values` ausgegeben werden, die sich restlos durch 3 teilen lassen. Verwende zur Ausgabe die Methode `System.out.println`.

```
1  public static void printModThree(int[][] values) {
2
3
4
5
6
7
8
9  }
```

### 2.3 switch

Ergänze die folgenden Methode, sodass sie für einen Monat ausgibt, wie viele Tages dieser hat. Wird ein ungültiger Wert, also ein Wert der nicht zwischen 1 und 12 liegt übergeben, soll eine Exception geworfen werden.

```
1   private int printDaysOfMonth(int month, boolean isLeap){
2
3
4      switch (     ) {
5         case 1: case 3: case 5:
6         case 7: case 8: case 10:
7         case 12:
8
9
10        case 4: case 6:
11        case 9: case 11:
12
13
14        case 2:
15
16
17
18        throw new IllegalArgumentException("Error, month has to be <= 12 und
              >= 1");
19     }
20
21  }
```

## 3 Objekt-Orientierung

### 3.1 Datenkapselung, Vererbung

Sorge dafür, dass die Attribute `numberOfPlayers` und `name` vor dem Zugriff aus anderen Klassen geschützt sind. Ergänze dann die Klasse SportsTeam um getter- und setter-Methoden, falls diese sinnvoll sind. Ergänze die `main`-Methode so, dass ein neues Basketballteam erzeugt wird. Ergänze dann die Klasse `BasketballTeam`, sodass die `main`-Methode fehlerfrei ausgeführt werden kann.

```
1   public class SportsTeam {
2      final int numberOfPlayers;
3      String name;
4
5      public SportsTeam(int numberOfPlayers, String name) {
6         this.numberOfPlayers = numberOfPlayers;
7         this.name = name;
8      }
9
10
11
12 }
```

```java
public class BasketballTeam extends SportsTeam {




   @Override
   public void score(int score) {
      System.out.println("The basketball score is "+ score);
   }

   public static void main(String[] args) {
      SportsTeam team = new BasketballTeam   ;
      team.score(9);
   }
}
```

## 3.2 Modellierung

Ergänzen Sie die Klasse `Animal`, sodass sie die Interfaces `Comparable<T>` und `InterfaceB` implementiert. Zwei Tiere sind gleich, wenn sie gleich alt sind. Ist ein Tier älter als ein anderes, ist es von höherer Ordnung.

```java
   public class Animal {
      private int age;
      private boolean isAlive;

      @Override









      @Override




   }
```

```java
public interface InterfaceB {
   boolean isAlive();
}
```