Aufgaben Tag 2

Aufgabe 1: Polymorphie 1

In dieser Aufgabe sollen Sie Java-Klassen zur Verwaltung einer Fußballmannschaft entwerfen. Folgende Objekte und Eigenschaften sollen verwaltet werden:

- Mannschaft: hat einen Namen, einen Trainer und genau 11 Spieler
- Spieler(in): hat einen Namen, ein Alter und eine Anzahl geschossener Tore
- Trainer(in): hat einen Namen, ein Alter und Erfahrung (Anzahl Jahre)

a) Für manche Objekte in der obigen Liste bietet es sich an eine Oberklasse zu verwenden. Definieren Sie zuerst diese Oberklasse und danach jeweils eine Java-Klasse für alle Objekte aus der obigen Liste.
Hinweis: Alle Instanzvariablen sollen außerhalb der Klasse nicht zugreifbar sein, auch nicht über Getter-und-Setter-Methoden!

b) Implementieren Sie für jede Klasse aus a) einen Konstruktor.

c) Schreiben Sie eine kurze Test-Klasse mit einer `main`-Methode in welcher eine Instanz vom Typ Mannschaft erzeugt und alle zugehörigen Variablen initialisiert werden.
Hinweis: Für die Spieler reichen einfache Testdaten!

Aufgabe 2: Polymorphie 2

Gegeben seien nachstehende Java-Klassen. Wichtig sind deren Vererbungsbeziehungen, sowie die Methoden `aufhalten`.

```
public class Lebewesen { }

public class Mensch extends Lebewesen { }

public class Tier extends Lebewesen { }

public class Vogel extends Tier  {}

public class Platz {
   public void aufhalten(Lebewesen l1, Lebewesen l2) {
      System.out.println("Lebewesen 1 und Lebewesen 2 an einem unbekannten Platz.");
   }
}

public class Haus extends Platz {
   public void aufhalten(Mensch m, Tier t) {
      System.out.println("Mensch und Tier im Haus.");
   }
}

public class Wald extends Platz {
   public void aufhalten(Lebewesen l, Vogel v) {
      System.out.println("Lebewesen und Vogel im Wald.");
   }

   public void aufhalten(Mensch m, Tier t) {
      System.out.println("Mensch und Tier im Wald.");
   }
}
```

Geben Sie für die nachstehenden Befehlssequenzen an was jeweils auf der Konsole ausgegeben wird, falls die Befehlssequenz fehlerfrei ist. Falls ein Fehler auftritt, beschreiben Sie diesen kurz.
Achtung: Die obigen Klassen beinhalten keine Fehler!

a) Befehlssequenz 1:

```
Lebewesen lebewesen  = new Tier();
Wald      wald       = new Wald();
wald.aufhalten(lebewesen, new Vogel());
```

b) Befehlssequenz 2:

```
Wald wald = new Wald();
wald.aufhalten(new Tier(), new Wald());
```

c) Befehlssequenz 3:

```
Haus  haus  = new Haus();
Platz platz = haus;
platz.aufhalten(new Mensch(), new Tier());
```

d) Befehlssequenz 4:

```
wald = new Wald();
wald.aufhalten(new Tier(), new Tier());
```

Aufgabe 3: Referenzen

a) Was gibt das folgende Programm aus?

```
1  public class References1 {
2      private static void change(int integer) {
3          integer = 42;
4      }
5
6      private static void change2(int[] integers) {
7          for(int i : integers) {
8              i += 10;
9          }
10     }
11
12     public static void main(String[] args) {
13         int a = 1;
14         int b = 2;
15
16         int[] numbers = new int[3];
17         change2(numbers);
18
19         numbers[1] = b++;
20         numbers[0] = ++a;
21
22         change(a);
23
24         numbers[2] = a;
25
26         System.out.println(a);
27         System.out.println(b);
28         System.out.println(numbers[0]);
29         System.out.println(numbers[1]);
30         System.out.println(numbers[2]);
31     }
32 }
```

b) Was gibt das folgende Programm aus?

```java
class Book {
    public String title;

    public Book(String title) {
        this.title = title;
    }
}

public class References2 {
    private static void change(Book book) {
        book.title = "Dancing Jax";
    }

    public static void main(String[] args) {
        Book book = new Book("Gone");
        Book[] books = new Book[2];
        books[0] = new Book("Justifiers");
        books[1] = new Book("Erebos");

        books[1] = book;
        book = books[0];
        books[0].title = "Ender's Game";
        change(book);
        books[0] = book;

        Book newBook = new Book("Dancing Jax");

        System.out.println(book.title);
        System.out.println(books[0].title);
        System.out.println(books[1].title);

        System.out.println(books[0] == newBook);
    }
}
```

Aufgabe 4: Vererbung und statische Methoden

Gegeben seien folgende Klassen:

```java
public class Drive {
    public static String connector = "unknown";
    public int capacity = 0;
    public String name = "unknown";
    public Drive(String name, int capacity) {
        this.name = name;
        this.capacity = capacity;
    }
    public static String getConnector() {
        return connector;
    }
    public int getCapacity() {
        return capacity;
    }
    public String getName() {
        return name;
    }
    public String doSomething(USBDrive drive) {
        return "done";
    }
}

public class USBDrive extends Drive {
    public static String connector = "USB";
    public USBDrive(String name, int capacity) {
        super(name, capacity);
    }
    public static String getConnector() {
        return connector;
    }
    public String getName() {
        return name;
    }
    public String doSomething(Drive drive) {
        return "USB done";
    }
}
```

```
39  public class StaticStuff {
40      public static void main(String args[]) {
41          USBDrive usbDrive1 = new USBDrive("Erstistick", 8000);
42          Drive usbDrive2 = new USBDrive("Lubuntu Live-Stick", 2000);
43          Drive drive = new Drive("Floppy Drive", 1);
44
45          System.out.println(usbDrive1.name);
46          System.out.println(usbDrive1.getCapacity());
47          System.out.println(usbDrive1.connector);
48          System.out.println(usbDrive1.doSomething(usbDrive1));
49          System.out.println(usbDrive1.doSomething(usbDrive2));
50          System.out.println(usbDrive1.doSomething(drive));
51          System.out.println();
52          System.out.println(usbDrive2.getName());
53          System.out.println(usbDrive2.capacity);
54          System.out.println(usbDrive2.getConnector());
55          System.out.println(usbDrive2.doSomething(usbDrive1));
56          System.out.println();
57          System.out.println(drive.name);
58          System.out.println(drive.getCapacity());
59          System.out.println(drive.connector);
60          System.out.println(usbDrive2.doSomething(usbDrive1));
61          System.out.println();
62          System.out.println(Drive.getConnector());
63          System.out.println(Drive.connector);
64          System.out.println(USBDrive.getConnector());
65          System.out.println(USBDrive.connector);
66          System.out.println();
67          usbDrive1.connector = "USB 2.0";
68          USBDrive usbDrive3 = new USBDrive("Erstistick 2017", 16000);
69          System.out.println(USBDrive.connector);
70          System.out.println(usbDrive1.getConnector());
71          System.out.println(usbDrive3.getConnector());
72
73      }
74  }
```

a) Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antworten.

1. In der Klasse `USBDrive` wird die Methode `getConnector()` von `Drive` überschrieben.
2. In der Klasse `USBDrive` wird die Methode `getName()` von `Drive` überschrieben.
3. In der Klasse `USBDrive` wird die Methode `doSomething()` von `Drive` überschrieben.
4. Um die Methode `getConnector()` der Klasse `USBDrive` aufzurufen, braucht man ein Objekt vom Typ `USBDrive`.
5. Auf die Variable `connector` in der Klasse `Drive` kann auch ohne ein Objekt vom Typ `Drive` zugegriffen werden.

b) Was wird ausgegeben, wenn das Programm ausgeführt wird?

c) Begründen Sie Ihre Antworten.

1. Kann `System.out.println(usbDrive2.doSomething(usbDrive2));` in der `main`-Methode eingefügt werden?
2. Kann in der `main`-Methode der Befehl `System.out.println(usbDrive2.doSomething(drive));` eingefügt werden?
3. Ändert sich die Ausgabe von `System.out.println(usbDrive2.getConnector());`, wenn `getConnector()` in `Drive` und `USBDrive` nicht statisch wären?