Aufgabenzettel 7

Dynamic Binding und Collections

Dieser Aufgabenzettel muss spätestens in Ihrer Praktikumsgruppe in **KW 24** vorgestellt werden
Ausnahme: Gruppen 1, 2 und 4 können bis zum 23. Juni abgeben, da der 9. Juni ein Feiertag ist und der 16. Juni in der Projektwoche liegt.

Alle für diesen Aufgabenzettel entwickelten Programme sollen sich in einem Paket **de.hsruhrwest.oop.ss2025.assignment7** oder entsprechenden **Unterpaketen** befinden.

Aufgabe 1

Betrachten Sie die folgende Klassenhierarchie:

```
public class Top {
    private int left;
    private int right;

    public Top(int left, int right) {
        this.left = left;
        this.right = right;
    }
    public int getLeft()      { return left; }
    public int getRight()     { return right; }

    public int getResult()    { return getLeft() + getRight(); }
}


public class Middle extends Top {
    public Middle(int left, int right) {
        super(left, right);
    }

    @Override
    public int getResult() {
        return getLeft() - getRight();
    }
}


public class Bottom extends Middle {
    public Bottom(int left, int right) {
        super(left, right);
    }

    @Override
    public int getResult() {
        return super.getLeft() * -1;
    }
}
```

Welche Ausgaben produzieren die folgenden Codeausschnitte? Begründen Sie Ihre Antwort.

a)
```
Top object = new Middle(5, 6);
System.out.println( object.getResult() );
```

b)
```
Top object = new Bottom(5, 6);
System.out.println( object.getResult() );
```

c)
```
Top object = new Top(5, 6);
System.out.println( object.getResult() );
```

Aufgabe 2

Implementieren Sie eine einfache Datenbank für Computerspieler.

a) Implementieren Sie eine Klasse **Player**, die einen Computerspieler repräsentiert. Ein Spieler besitzt einen Namen, eine Punktzahl und eine Menge (Set) von Lieblingsspielen. Ein Lieblingsspiel soll hier als einfacher String modelliert werden. Implementieren Sie einen Konstruktor, Getter, Setter, toString, sowie die equals- und hashCode-Methode für die Klasse. equals- und hashCode dürfen generiert werden.

b) Die Klasse **Player** soll nun die Comparable-Schnittstelle implementiert und Spieler natürlicherweise nach Punktzahl sortieren.

c) Implementieren Sie zusätzlich einen Comparator, der Spieler nach der **Anzahl** der Lieblingsspiele sortiert.

d) Implementieren Sie eine Spielerdatenbank **PlayerDatabase**. Sie speichert intern eine Liste (List) von Spielern und bietet folgende Funktionen:

- Das **Hinzufügen** eines Spielers.
- Die **Rückgabe** aller Spieler.
- Das **Löschen** von Spielern anhand eines Suchstrings. Die Methode bekommt einen String als Parameter übergeben und löscht alle Spieler, die diesen String im Namen enthalten. Verwenden Sie für die Aufzählen und Löschen der Spieler einen Iterator.
- Eine Methode, die als Parameter ein Computerspiel (String) erhält. Sie gibt alle Spieler als Menge (Set) zurück, die das Spiel in ihrer Liste der Lieblingsspiele enthalten.

- Zwei Methoden, die die Liste der Spieler **sortieren**, einmal nach ihrer natürlichen Ordnung (siehe b) und einmal nach der Anzahl der Lieblingsspiele (siehe c).
- Eine Methode, die ein Spiel-Histogramm zurückgibt. Das Histogramm ist eine Abbildung (Map), die jedem Computerspiel die Häufigkeit zuordnet, mit der es von Spielern als Lieblingsspiel gewählt wurde. Sie bildet also Namen von Computerspielen (String) auf Anzahlen (Long) ab.

e) Testen Sie die gesamte Funktionalität in einem Hauptprogramm.

Aufgabe 3

Betrachten Sie die folgende Klasse:

```
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }
}
```

In diesem Hauptprogramm werden zwei Personen einer Menge hinzugefügt:

```
public static void main(String[] args) {
     var set = new HashSet<Person>();
     set.add(new Person("Bob"));
     set.add(new Person("Bob"));

     System.out.println("Size of set: " + set.size());
 }
```

Obwohl zwei Personen mit gleichen Attributen („Bob“) hinzugefügt werden, enthält die Menge zwei Einträge statt einem. Worin besteht der Fehler?