Java Intro Workshop

4 Aufgaben zu Klassen

Aufgabe 1: SuperHero

In dieser Aufgabe soll eine Klasse `SuperHero` implementiert werden, die Superhelden mit ihren Namen, ihren Superkräften und ihrer Stärke modelliert. Zudem soll eine Methode `boolean isStrongerThan(SuperHero)` prüfen, ob ein Superheld stärker als ein anderer ist. Schließlich ist eine allgemeingültige Prüfung zu realisieren, die den stärksten Superhelden aus einer Menge zurückgibt. Dazu soll eine Methode `SuperHero strongestOf(SuperHero...)` mithilfe von Var Args, einer variablen Parameterliste, erstellt werden. Folgendes Hauptprogramm zeigt die Verwendung und mögliche Aufrufe – variieren Sie gern die Eigenschaften.

```
public static void main(String[] args)
{
    SuperHero superMan =
              new SuperHero("Superman", "Kryptonite-Power", 1_000);
    SuperHero batMan = new SuperHero("Batman", "Techno-Power", 100);
    SuperHero ironMan = new SuperHero("Ironman", "Techno-Power", 500);
    System.out.println("Superman stronger than Batman? " +
                       superMan.isStrongerThan(batMan));
    System.out.println(SuperHero.strongestOf(superMan, batMan, ironMan));
}
```

Aufgabe 2: Counter

Nachdem die Grundbegriffe beim objektorientierten Entwurf mit Java bekannt sind, soll nun ein Zähler als Klasse entworfen werden, der folgende Anforderungen erfüllt:

1. Er lässt sich auf den Wert 0 zurücksetzen.
2. Er lässt sich um eins erhöhen.
3. Der aktuelle Wert lässt sich abfragen.

Dabei existiert bereits das folgende, rudimentäre Grundgerüst:

```
public class Counter
{
    public int count = 0;

    public Counter()
    {
    }
```

```
    public void setCounter(int count)
    {
        count = count;
    }
}
```

Das Attribut `count` speichert den Zähler und kann von überall abgefragt und verändert werden. Das Rücksetzen erfolgt durch Übergabe von 0. Die obige Implementierung sollen Sie nun korrigieren, erweitern und verbessern, indem Sie zu den Anforderungen passende Methoden implementieren und als Kür das Attribut „verstecken“.

Aufgabe 3: Vererbung

Erstellen Sie zwei Basisklassen `ExcludeFilter` und `IncludeFilter` sowie basierend darauf einen SPAM-Filter und einen Namensfilter, die jeweils eine `filter()`-Methode anbieten und wie folgt aufgerufen werden:

```
public static void main(String[] args)
{
    IFilter spamfilter = new SPAMFilter();
    System.out.println(spamfilter.filter(List.of("SPAM",
"DAS", "IST", "SPAM", "SPAM", "DETECTED")));

    IFilter namefilter = new NameFilter();
    System.out.println(namefilter.filter(List.of("DAS",
"SIND", "MICHAEL", "UND", "SOPHIE")));
}
```

Das gewünschte Ergebnis ist wie folgt:

```
[DAS, IST, DETECTED]
[MICHAEL, SOPHIE]
```

BONUS: Führen Sie nun noch eine abstrakte Basisklasse `AbstractFilter` oder ein Interface `IFilter` ein, die lediglich die Signatur der Methode vorgibt. Modifizieren Sie die Konstruktion und Aufrufe passend.