Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
Dr. Jens Bendisposto

20. September 2021

Klausur

Professionelle Softwareentwicklung
SS 2021

Hinweise:

- Diese Klausur enthält 10 nummerierte Klausurseiten. Prüfen Sie bitte zuerst, ob die Klausur alle Seiten enthält.
- Täuschungsversuche führen zum sofortigen Ausschluss von der Klausur. Die Klausur wird dann als nicht bestanden gewertet.
- Schreiben Sie **nicht** mit radierbaren Stiften und auch **nicht** mit rot!

Nachname: _____ Vorname: _____

Matrikelnummer: _____ _____

Hörsaal: _____ Sitzplatznummer: _____

Unterschrift: _____

Viel Erfolg!

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	Σ
Punktzahl	4	3	5	3	2	3	20
Erreicht							

Aufgabe 1 [4 Punkte]

Jakobus Prum hat neben den Prumzahlen (Zahlen, die durch 3,5 oder 7 teilbar sind) auch die Prumvektoren definiert. Ein Prumvektor ist ein Vektor, dessen Länge eine Prumzahl ist. Wir haben bereits eine Klasse `Vektor`, die beliebige Vektoren speichern kann. In der Klasse gibt es eine Methode `int getLaenge()`, die die Länge des Vektors berechnet.

(a) [3 Punkte] Schreiben Sie eine Methode `findPrumVector(Stream<Vector> input)`, die einen Stream von Vektoren als Eingabe bekommt, und in dem Stream einen Prumvektor sucht und diesen zurückgibt. Es gibt einige Bedingungen an die Methode:

- Die Methode muss mit jedem beliebigen, nicht verbrauchten, Stream klarkommen. Insbesondere darf die Methode keine Exception werfen.
- Die Methode darf niemals den Wert `null` oder einen nicht im Stream vorhandenen Vektor zurückgeben.
- Wenn es einen Prumvektor in dem Stream gibt, dann muss die Methode terminieren.
- Die Methode soll außer dem Verbrauchen des Streams keine Seiteneffekte haben.

Sie müssen einen geeigneten Rückgabetyp der Methode selber wählten. Aufrufer der Methode sollen über den zurückgegebenen Wert Zugriff auf den Prumvektor bekommen.

```
public class PrumvektorUtil {

  public static boolean isPrum(int n) {
    return n % 3 == 0 || n % 5 == 0 || n % 7 == 0;
  }









}
```

(b) [1 Punkt] Warum können wir keine Methodenreferenz auf die Methode `isPrum` verwenden?

Aufgabe 2 [3 Punkte]

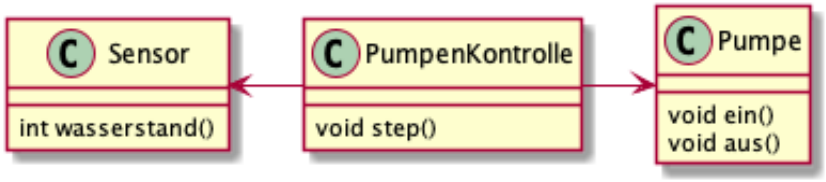
Unser Kollege Peter hat eine FIFO (First In, First Out) Queue implementiert. Die Queue soll bei einer Microblogging Platform benutzt werden und kann daher zwischenzeitlich sehr groß werden, zum Teil werden dort einige Millionen Objekte zwischengespeichert. Peters Implementierung sieht wie folgt aus:

```
public class Fifo<E> {

  private final List<E> queue = new ArrayList<>();

  public void push(E e) {
    queue.add(e);
  }

  public E pop() {
    return queue.remove(0);
  }

  public boolean isEmpty() {
    return queue.isEmpty();
  }

}
```

Die Lösung funktioniert korrekt. Aber ist das auch eine adäquate Lösung für unser Problem? Begründen Sie Ihre Antwort. Beschreiben Sie, welche Probleme unter welchen Umständen auftreten könnten und machen Sie gegebenenfalls Vorschläge zur Verbesserung.

Aufgabe 3 [5 Punkte]

Wir wollen folgendes automatische Pumpsystem testen:



Wir haben auch schon ein Grundgerüst für einen Test entwickelt, bei dem wir prufen wollen, dass die Pumpe eingeschltet wrd, wenn der Wasserstand den Wert 10 übersteigt.

```
@Test
void testOverflow() {
// <1>
  Sensor sensor = mock(Sensor.class);
// <2>
  Pumpe pumpe = mock(Pumpe.class);
// <3>
  PumpenKontrolle controller = new PumpenKontrolle(pumpe);
// <4>
  assertThat(pumpe.aktiv()).isFalse();
// <5>
  controller.setSensor(sensor);
// <6>
  controller.step(); // Schaltet die Pumpe ein, wenn der Wasserstand höher ist als 10
// <7>
}
```

(a) [2 Punkte] An einer oder mehreren der mit Zahlen markierten Stellen benötigen wir Code, damit der Test korrekt funktioniert. Schreiben Sie auf welcher zusätzliche Code an welchen Stellen benötigt wird. Wenn der Code an mehr als einer Stelle stehen könnte dürfen Sie frei wählen

Beispiel:`<1> System.out.println("Beispiel");`

(b) [1 Punkt] Welche Zeile oder welche Zeilen machen den Act Schritt des Tests aus?

(c) [2 Punkte] Es werden die beiden Test-Doubles sensor und pumpe verwendet. Welche davon sind Stubs bzw. Mocks. Begründen Sie die Antwort.

Aufgabe 4 [3 Punkte]

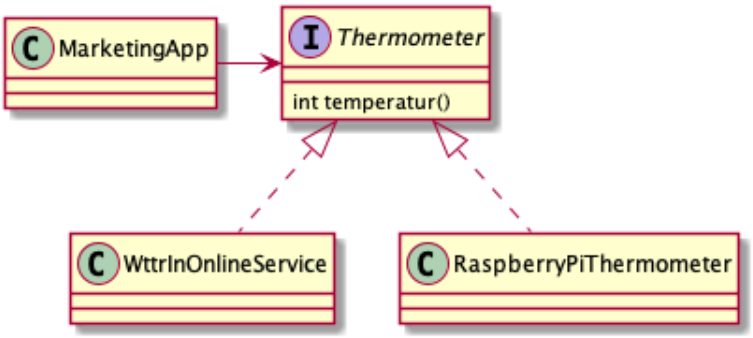
Als wir über Datenabstraktion gesprochen haben, haben wir ein Beispiel kennengelernt, bei dem Daten für Klausurzulassungen in zwei Arrays gespeichert wurden.

```
public class Zulassungen {
  private String[] logins;
  private int[] punkte;

  public void printZulassung() {
    for (int i=0; i < punkte.length; i++) {
      if (punkte[i] > 300) System.out.println(logins[i]);
    }
  }

  // Konstruktor und weitere Methoden
}
```

Wir haben damals, die Klasse `Student` eingeführt, in der wir jeweils einen Login und eine Punktzahl gespeichert haben. Ohne es damals zu wissen, haben wir einen Code Smell beseitigt. Um welchen Code Smell handelt es sich dabei? Geben Sie den Namen des Smalls an und beschreiben Sie kurz, was den Code Smell ausmacht und warum er hier Ihrer Ansicht nach vorliegt.

Aufgabe 5 [2 Punkte]

Wir wollen unsere Eisdielen Marketinganwendung mit Spring Boot umsetzen. Zunächst wollen wir uns auf die Temperaturbestimmung einschränken und wir haben daher folgende Spring Beans schon implementiert:



(a) [1 Punkt] Wir haben die Klassen mit `@Component` annotiert und injizieren die Klasse `MarketingApp` in eine Instanz von `CommandLineRunner`. Welcher Fehler taucht dabei auf und was ist die Ursache?

(b) [1 Punkt] Wie können wir das Problem ausschließlich mithilfe von Spring Boot Annotationen zu beheben.

Aufgabe 6 [3 Punkte]

Wir haben die Klasse `Versand` in einem Webshop-Systen. In der Klasse gibt es die Methode `darfVerschicktWerden`, die prüft, ob ein Produkt legal an einem Kunden verschickt werden darf. Die Methode funktioniert, verletzt aber das Gesetz von Demeter. Erklären Sie, an welcher Stelle das Gesetz verletzt wird und wie der Code geändert werden müsste, damit das Gesetz von Demeter eingehalten wird.

```
public class Versand {
  public boolean darfVerschicktWerden(Kunde k, Produkt p) {
    if (p.isAltersbegrenzt()) {
      Stammdaten sd = k.getStammdaten();
      if (sd.getAlter() < 18) return false;
    }
    return true;
  }
}
```