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

09. August 2021

Klausur

Professionelle Softwareentwicklung
SS 2021

Hinweise:

- Diese Klausur enthält 9 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	Σ
Punktzahl	6	3	3	5	7	24
Erreicht						

Aufgabe 1 [6 Punkte]

(a) [5 Punkte] Eine Prumzahl (benannt nach Jakobus Prum) ist eine Zahl x, die durch 3, 5 oder durch 7 teilbar ist. Die ersten Prumzahlen lauten 3, 5, 6, 7, 9, 10, 12, 14, ... In der Klausur Programmierung haben Sie vielleicht die Methode `boolean isPrum(int n)` geschrieben, die `true` zurückgibt, genau dann wenn `n` eine Prumzahl ist. Sie können die Methode als gegeben ansehen.

Schreiben Sie eine Methode `List<Integer> prum(int n)`, die eine Liste mit den ersten `n` Prumzahlen zurückgibt. Sie müssen eine reine Stream-Lösung schreiben und Sie müssen eine Methodenreferenz auf die `isPrum` Methode verwenden.

```
public class Prumzahlen {

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









}
```

(b) [1 Punkt] Wir wollen die Elemente in einem Stream duplizieren. Aus dem Stream mit den Elementen 1,2,3 soll der Stream mit den Elementen 1,1,2,2,3,3 erzeugt werden. Dazu haben wir folgenden Codeschnipsel gegeben. Was muss in die Lücke eingesetzt werden, damit der Code funktioniert?

```
public static <T> Stream<T> dupe(Stream<T> input) {

  return input._____________________(e -> Stream.of(e,e));

}
```

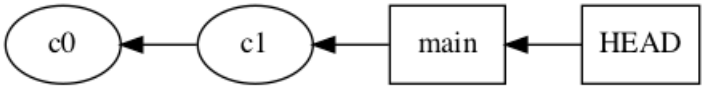
Aufgabe 2 [3 Punkte]

(a) [1 Punkt] Worum handelt es sich bei dem Code Smell „Divergent Change“ (Divergierende Änderungen)? Beschreiben Sie in 1-3 Sätzen um was es sich bei dem Smell handelt.

(b) [2 Punkte] Welches der SOLID Prinzipien wird verletzt, wenn der Code Smell „Divergent Change“ (Divergierende Änderungen) vorliegt? Beschreiben Sie warum Code, welcher den Smell hat, das Prinzip verletzen könnte.

Aufgabe 3 [3 Punkte]

Wir haben ein git Repository, in dem zwei Commits gemacht sind:



Zeichnen Sie den resultierenden Graphen, der entsteht, wenn wir die folgenden git Kommandos in dem Repository ausführen:

```
git commit
git checkout c0
git branch fix
git checkout fix
git commit
git checkout main
git merge fix
```

Zeichnen Sie alle Commits und die Position aller Branches und des HEAD Pointers ein. Sie können davon ausgehen, dass vor jedem Commit eine Änderung gestaged wurde. Wie bei dem Simulator, dessen Level Sie auf Blatt 14 lösen sollten, verwenden wir `c0`, `c1`, `c2`, etc. als Ersatz für die SHA-Codes.

Aufgabe 4 [5 Punkte]

Angenommen, wir wollen das Wörterbuch aus der praktischen Übung 4 in Spring umsetzen. Wir haben eine Spring Bean `Bloomfilter`, die wir in die Klasse `Woerterbuch` injecten. Die Wörterbuchklasse sieht folgendermaßen aus:

```
public class Woerterbuch {

  private final Bloomfilter store;

  public Woerterbuch(Bloomfilter store) {
    this.store = store;
  }

  public void addWord(String word) {
    String cleaned = prepareWord(word);
    store.add(cleaned);
  }

  public boolean contains(String word) {
    String cleaned = prepareWord(word);
    return store.contains(cleaned);
  }

  private String prepareWord(String word) {
    // Satzzeichen und Leerzeichen entfernen, umwandeln in lowercase
  }

}
```

(a) [1 Punkt] Was müssen wir ändern, um aus der Klasse `Woerterbuch` eine Spring Bean zu machen?

(b) [4 Punkte] Wir wollen die Klasse `Woerterbuch` testen. Schreiben Sie einen solitary Unit-Test, der prüft, dass, wenn wir `"Klausur!!! "` als Eingabe haben, das Wort `"klausur"` im Bloomfilter gespeichert wird. Sie können davon ausgehen, dass die üblichen Dependencies (JUnit, Mockito, JAssert) vorhanden sind. Es reicht, wenn Sie die Testmethode schreiben. Markieren Sie, welche Zeilen zu den einzelnen Teilen im AAA-Muster gehören.

Aufgabe 5 [7 Punkte]

Wir wollen eine Datenstruktur schreiben, in der wir Integer Schlüssel-Wert-Paare speichern können. Eine normale `HashMap` können wir leider nicht verwenden, weil wir sehr schnell nachprüfen können müssen, ob ein bestimmter Wert vorhanden ist. Bei der normalen `HashMap` müssten dazu alle Werte durchsucht werden.

Schreiben Sie eine Klasse `IntegerLookupMap` mit den Methoden `put`, `get` und `contains`. Die Methode `get` verhält sich wie bei einer `HashMap<Integer,Integer>`. Die Methode `put` verhält sich ähnlich zu der `put` Methode aus der Map, ignoriert aber alle Aufrufe, bei denen ein schon vorhandener Wert gespeichert werden soll. Die Methode `contains` soll `true` zurückgeben, genau dann, wenn der Wert unter einem beliebigen Schlüssel gespeichert ist. Alle Methoden müssen performant sein, es dürfen nicht schlimmstenfalls alle Elemente durchlaufen werden. Wir dürfen zugunsten der Laufzeitperformance aber Speichereffizienz opfern. Sie dürfen vorhandene Datenstrukturen aus Java zur Implementierung verwenden.