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

HEINRICH HEINE
UNIVERSITÄT
DÜSSELDORF

28. September 2019

Klausur

Professionelle Softwareentwicklung
Sommersemester 2019

Nachname: _____ Vorname: _____

Matrikelnummer: _____

Unterschrift: _____

☐ Ich möchte im WS 19/20 **nicht** an der Veranstaltung *Softwareentwicklung im Team* teilnehmen.

Zugelassene Hilfsmittel: Eine handschriftlich beschriebene A4 Seite. Matrikelnummer und Name muss auf der Seite stehen

Sollten Sie zusätzliche Blätter benötigen, bekommen Sie diese von uns. Sie dürfen keine eigenen Blätter verwenden!

Diese Klausur enthält 9 nummerierte Seiten. Prüfen Sie bitte zuerst, ob alle Seiten vorhanden sind.

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	7	8	9	Σ
Punktzahl	4	8	6	10	10	8	8	16	10	80
Erreicht										

Aufgabe 1 [4 Punkte]

Es gibt zwei grundsätzlichen Sorten von Stream Operationen. Erklären Sie den Unterschied zwischen den beiden Sorten und geben Sie jeweils eine Beispieloperation an.

Aufgabe 2 [8 Punkte]

Schreiben Sie eine statische Higher Order Funktion `applytwice`, die eine generische Funktion `f` als Parameter übergeben bekommt und eine Funktion zurückgibt, die `f` zweifach anwendet. Java stellt das Interface `UnaryOperator` zur Verfügung, dass Sie verwenden können. Folgendermaßen soll `applytwice` benutzt werden:

```
UnaryOperator<Integer> pow4 = applytwice(x -> x * x);
System.out.println(pow4.apply(2)); // Ausgabe ist 16
UnaryOperator<String> concatTwice = applytwice(x -> x + x);
System.out.println(concatTwice.apply("*")); // Ausgabe ist ****
```

Das Interface `UnaryOperator` ist etwa folgendermaßen definiert:

```
@FunctionalInterface
public interface UnaryOperator<T> {
  T apply(T t);
}
```

Aufgabe 3

[6 Punkte]

Da Software immateriell ist und daher nicht verschleißen kann, ist es auf den ersten Blick nicht intuitiv von Software-Alterung zu sprechen. Trotzdem existiert dieses Phänomen. Erklären Sie, was damit gemeint ist und geben Sie ein Beispiel an, wo Software-Alterung auftreten könnte.

Aufgabe 4

[10 Punkte]

Angenommen wir haben eine Liste von Nutzern, die wir über die URL `/shop/users` ansprechen können. Dieser Ressource wollen wir einen neuen Benutzer hinzufügen. Das wäre sowohl mit einem `PUT` als auch einem `POST` umsetzbar. Vergleichen Sie die beiden Umsetzungen und erklären Sie, welches HTTP Verb aus welchem Grund zu bevorzugen ist.

Aufgabe 5

[10 Punkte]

In dem Buch *Einführung in die Programmierung mit Java* von Sedgewick und Wayne, das Sie aus der Veranstaltung *Programmierung* kennen, wird im Abschnitt *Unit-Tests* vorgeschlagen, die Funktionen der Klasse StdRandom folgendermaßen zu testen:

```
public static void main(String[] args) {
  for (int i = 0; i < 5000; i++) {
    StdOut.printf("%2d", StdRandom.uniform(100));
    // teste weitere Funktionen von StdRandom
    StdOut.println();
  }
}
```

Die Autoren schreiben dazu „Wenn wir diesen Code [...] aufrufen, dürfte die Ausgabe niemanden überraschen: Die Zahlen [...] dürften mit gleicher Wahrscheinlichkeit Werte zwischen 0 und 99 sein [...]“

Was halten Sie von dieser Art, die `uniform`-Methode zu testen? Vergleichen Sie das mit den Prinzipien, die Sie über Tests gelernt haben.

Aufgabe 6 [8 Punkte]

Wir wollen eine Kontrollsoftware für einen Mars Rover (ohne Beachtung der Object Calisthenics) schreiben. Die Software soll das Information Hiding Prinzip nach David Parnas beachten. Geben Sie an, welche Klassen existieren sollten. Ihre Antwort soll für jede Klasse die folgenden Informationen enthalten:

- Name der Klasse
- Welche Aufgabe hat die Klasse? (ca. 1-2 Sätze)
- Welche Entscheidung wird gekapselt? (ca. 1-2 Sätze)
- Liste der öffentlichen Methoden (Die Methodennamen müssen selbsterklärend sein)

Aufgabe 7 [8 Punkte]

Für unseren Papageien-Sprachnachrichtendienst haben wir die Entscheidung getroffen, Polymorphismus einzusetzen und sind in etwa bei folgendem Design angekommen:

```
public interface Parrot {
  double getSpeed();
}

class EuropeanParrot implements Parrot {
  public double getSpeed() {
    return 12;
  }
}

class AfricanParrot implements Parrot {
  private int coconuts;
  public AfricanParrot(int coconuts) {
    this.coconuts = coconuts;
  }
  public double getSpeed() {
    return Math.max(0, 12 - 0.9 * coconuts);
  }
}
```

(a) Der Entscheidung Polymorphismus zu verwenden, liegt eine Vermutung über die Art und Weise zugrunde, wie sich das System in Zukunft entwickeln wird. Welche Vermutung ist das?

(b) Welche Art der Änderung wird von dem Design nicht unterstützt und welche alternative Vermutung könnte man über die zukünftige Entwicklung aufstellen, die in einem unterschiedlichen Design enden würde?

Aufgabe 8 [16 Punkte]

Wir wollen eine Webanwendung schreiben, die eine Liste von Kunden anzeigen kann. Die Klassen für die Daten sind folgendermaßen definiert

```
@Data  // Lombok
class Kunde {
  private String vorname;
  private String nachname;
  private Adresse adresse;
}

@Data  // Lombok
class Adresse {
  private String strasse;
  private String plz;
}
```

Gespeichert sind die Kunden in einer Datenbank und können über die Instanzmethode `List<Kunde> all()` der Spring-Bean `KundenDB` abgefragt werden. Es gibt außerdem einen Service `Postleitzahlen`, über dessen Instanzmethode `String getStadt(String plz)` der Name der Stadt für eine gegebene Postleitzahl geholt werden kann.

(a) Schreiben Sie die Controllerklasse inklusive der notwendigen Annotationen, die die Kundenliste unter der URL `/kunden/alle` zur Verfügung stellt. Der Controller soll die Datenbank-Klasse und den Postleitzahlenservice per Dependency Injection übergeben bekommen. Imports können Sie weglassen.

(b) An den Stellen A bis G in folgendem, aus der Designabteilung gelieferten, Template fehlen die Thymeleaf-Attribute, damit die tatsächlichen Kundendaten dargestellt werden. Ergänzen Sie die Attribute.

```
<html>
   <head><title>Kunden</title></head>
   <body>
    <table A>
      <tr B>
        <td C>Jens</td>
        <td D>Bendisposto</td>
        <td E>Universitätsstr. 1</td>
        <td F>40225</td>
        <td G>Düsseldorf</td>
      </tr>
      <tr>
        <td>Christian</td>
        <td>Meter</td>
        <td>Universitätsstr. 1</td>
        <td>40225</td>
        <td>Düsseldorf</td>
      </tr>
    </table>
   </body>
</html>
```

A	
B	
C	
D	
E	
F	
G	

Aufgabe 9 [10 Punkte]

Zu dieser Aufgabe gehört das Programm auf der folgenden Seite der Klausur.

Das Programm liest eine Datei ein und gibt die Zeilen und deren Länge aus. Zeilen, die länger als 80 Zeichen sind, werden verkürzt dargestellt.

Das Programm ist nicht einfach zu testen. Schreiben Sie das Programm so um, dass es besser testbar ist. Gehen Sie davon aus, dass die Methoden `output` und `readFromFile` korrekt funktionieren. Sie dürfen auch neue Klassen und Interfaces einführen. Wenn Sie Code aus der Implementierung wiederverwerten, müssen Sie diesen nicht erneut aufschreiben. Referenzieren Sie stattdessen die Zeilennnummern.

Anhang: Quelltext für Aufgabe 9

Sie können diese Seite (vorsichtig) aus der Klausur herausreißen. Die Seite muss nicht mit abgegeben werden.

```
01 public class ShortFilePrinter {
02
03   public void summarize(String file) {
04     List<String> lines = readFromFile(file);
05     for (String line : lines) {
06       int length = line.length();
07       if (length > 80) {
08         line = line.substring(0, 77) + "...";
09       }
10       if (length > 0)
11         output(line, length);
12     }
13   }
14
15   public void output(String line, int length) {
16     System.out.printf("%3d: %s%n", length, line);
17   }
18
19   public List<String> readFromFile(String file) {
20     Path p = Paths.get(file);
21     try {
22       return Files.readAllLines(p);
23     } catch (IOException e) {
24       return Collections.emptyList();
25     }
26   }
27
28   public static void main(String[] args) {
29     new ShortFilePrinter().summarize("/Users/bendisposto/klausur.tex");
30   }
31 }
```