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

8. Oktober 2020

Klausur

Professionelle Softwareentwicklung
Sommersemester 2020

Nachname: _____ Vorname: _____

Matrikelnummer: _____

Unterschrift: _____

☐ Ich möchte **nicht** für die Veranstaltung *Softwareentwicklung im Team* im WS 20/21 automatisch angemeldet werden, falls ich bestanden habe.

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

Der Klausur hängen Seiten für weitere Notizen an. Sie dürfen keine eigenen Blätter verwenden!

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

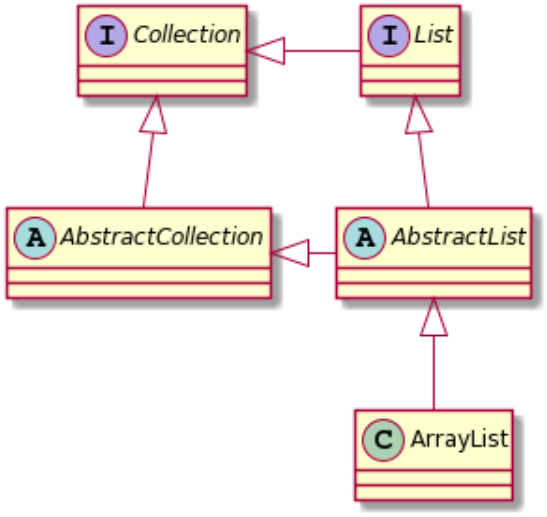
Aufgrund der Coronaschutzverordnung sind Sie verpflichtet persönliche Informationen zur Rückverfolgbarkeit auf einem separaten Zettel mit anzugeben. Bitte füllen Sie diesen Zettel zunächst aus.

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	Σ
Punktzahl	10	12	13	5	10	50
Erreicht						

Aufgabe 1 [10 Punkte]

Folgender Ausschnitt zeigt die Vererbungsbeziehungen zwischen verschiedenen Sammlungen in Java:



Die Buchstaben `I`, `A` und `C` stehen für `Interface`, `Abstract Class` und `Class`.

Betrachten Sie die folgende Codeschnipsel und beurteilen Sie, ob die Schnipsel ...

- nicht compilieren, weil ein Fehler vorliegt;
- zwar compilieren, aber zur Laufzeit immer eine `ClassCastException` werfen;
- compilieren und keine `ClassCastException` zur Laufzeit werfen.

Nehmen Sie an, der Code ist passend in einem Programm eingebettet, es sind also z.B., alle notwendigen Imports vorhanden.

Schreiben Sie pro Schnipsel 1–2 Sätze mit einer Begründung warum der Code einen Fehler hat oder warum er funktioniert. Diese Aufgabe erstreckt sich über zwei Seiten.

(a) [2 Punkte]

```
List x = new List();
```

(b) [2 Punkte]

```
AbstractList x = new ArrayList();
```

Fortsetzung von Aufgabe 1

(c) [2 Punkte]

```
List x = new ArrayList();
ArrayList y = x;
```

(d) [2 Punkte]

```
Collection x = new ArrayList();
AbstractList y = (ArrayList) x;
```

(e) [2 Punkte]

```
ArrayList<String> x = new ArrayList<>();
List<String> y = x;
```

Aufgabe 2 [12 Punkte]

Es soll eine Klasse geschrieben werden, die eine vereinfachte Version eines Repositories mithilfe einer HashMap implementiert. Das Repository soll Objekte vom Typ `Person` speichern, die jeweils eine UUID als Schlüssel haben. Die `Person` Klasse hat eine Methode `getID()`, um an den Schlüssel des Objekts zu kommen.

```
public class PersonRepository {

}
```

(a) [4 Punkte] Ergänzen Sie die obenstehende Klasse `PersonRepository` um die notwendigen Attribute sowie ggf. notwendigen Initialisierungscode.

(b) [2 Punkte] Ergänzen Sie die obenstehende Klasse um eine Methode `save`, die eine Instanz von `Person` bekommt und zu der HashMap hinzufügt oder eine bereits unter dem Schlüssel vorhandene Person überschreibt.

(c) [4 Punkte] Ergänzen Sie die obenstehende Klasse um eine Methode `findByID`, die zu einer übergebenen UUID das passende Objekt heraussucht. Es soll ein `Optional` zurückgegeben werden.

(d) [2 Punkte] Erklären Sie in 1–2 Sätzen, welchen Sinn es hat, ein `Optional` als Rückgabetyp von `findByID` zu verwenden.

Aufgabe 3 [13 Punkte]

Eine Middleware ist ein Wrapper um eine Funktion, die die Ein- und Ausgaben der Funktion modifizieren kann. Die Middleware kann exakt so verwendet werden, wie die gewrappte Funktion. Die Middleware Klasse bekommt im Konstruktor drei Funktionen übergeben: Die Funktion `f` vom Typ `Function<T,S>`, die gewrappt werden soll, sowie eine Funktion `pre`, welche die Eingabe transformiert, und eine Funktion `post`, welche die Ausgabe transformiert.

Ein Verwendungsbeispiel für Middleware sieht so aus:

```
Function<Integer, String> middleware =
    new Middleware<Integer, String> (
        i -> i * i,                    // pre
        i -> String.valueOf(i),        // f
        i -> "Das Ergebnis ist " + i   // post
    );
    System.out.println(middleware.apply(3)); // Ausgabe: Das Ergebnis ist 9
}
```

Beachten Sie, dass es sich nur um ein Beispiel handelt. **Die Middleware-Klasse muss auch mit anderen Funktionen und anderen Typen funktionieren.**

(a) [1 Punkt] Welchen Typ hat die Funktion `pre`?

(b) [1 Punkt] Welchen Typ hat die Funktion `post`?

(c) [2 Punkte] Mit welcher Lombok Annotation können Sie einen passenden Konstruktor erzeugen?

(d) [2 Punkte] Welches Interface implementiert bzw. welche abstrakte Klasse erweitert die Klasse `Middleware<T,S>`? Geben Sie auch ggf. notwendige Typparameter an.

(e) [3 Punkte] Welche Attribute hat die Klasse `Middleware<T,S>`?

(f) [4 Punkte] Implementieren Sie die `apply` Methode der Middleware Klasse.

Aufgabe 4 [5 Punkte]

Cousin Vincenzo und Jens sollen für Onkel Giacomo eine Erweiterung der Marketinganwendung schreiben. Die Kunden sollen mithilfe eines Machine Learning Ansatzes in Gruppen aufgeteilt werden, um noch gezielter werben zu können. Da Jens von Machine Learning keine Ahnung hat, übernimmt Vincenzo die Aufgabe, die Aufteilungskomponente zu implementieren, während Jens die Integration in den bestehenden Anwendungscode übernimmt. Um unabhängig zu arbeiten, einigen die beiden sich darauf konsequent in unterschiedlichen Packages zu arbeiten und Dependency Inversion und Injection zu verwenden, also ein Interface zu definieren, das Jens benutzt und Vincenzo implementiert. Wer von den Beiden übernimmt die Aufgabe, das Interface festzulegen und warum? Es gibt nur für die Begründung Punkte, nicht für die 50:50 Entscheidung.

Aufgabe 5 [10 Punkte]

Für einen Kontoauszugsdrucker wurde nach klassischer Chicagoer Schule des TDD eine Software entwickelt. Dabei sind die drei Testklassen auf der nächsten Seite entstanden. Die Software ist ein so großer Erfolg, dass sie auch für den amerikanischen Markt adaptiert werden soll. Dabei muss natürlich das Datumsformat auf das amerikanische Format umgestellt werden.

(a) [2 Punkte] Welche Nachteile ergeben sich für diese konkrete Änderung aus der Verwendung der klassischen Chicagoer Schule? Begründen Sie Ihre Antwort kurz.

(b) [8 Punkte] Schreiben Sie die Testmethode `test_format_auszugzeile` der Klasse `AuszugZeileTest` so um, dass sie die Nachteile der klassischen Schule für die oben beschriebene Änderung nicht mehr hat und dem modernen Chicagoer Stil entspricht. Gehen Sie davon aus, dass notwendige Bibliotheken vorhanden und importiert sind.

```
@DisplayName("Der DatumFormatter")
public class DatumFormatterTest {
  @Test
  @DisplayName("soll ein Datum im deutschen Format erzeugen")
  void test_formatter(){
    DatumFormatter formatter = new DatumFormatter();
    LocalDate datum = LocalDate.of(2001, 1, 2);
    assertThat(formatter.format(datum)).isEqualTo("02.01.2001");
  }
}
```

```
@DisplayName("Ein Kontoauszug")
public class KontoAuszugTest {
  @Test
  @DisplayName("soll einen Ausdruck mit zwei Buchungen ausgeben können")
  void test_zwei_buchungen_Auszug(){
    Konto konto = new Konto();
    konto.haben(LocalDate.of(2018,4,18), 100);
    konto.haben(LocalDate.of(2018,4,19), 320);
    KontoAuszug auszug = new KontoAuszug(konto, new DatumFormatter());
    String result = auszug.format();
    assertThat(result).isEqualTo(
       "Datum\tBetrag\tSaldo\n"+
          "18.04.2018\t100\t100\n" +
          "19.04.2018\t320\t420");
  }
}
```

```
@DisplayName("Die format Methode einer AuszugZeile")
public class AuszugZeileTest {
  @Test
  @DisplayName("soll einen korrekt formatierten String zurückgeben")
  void test_format_auszugzeile(){
    Transaktion transaktion = new Transaktion(LocalDate.of(2004,4,19), 102);
    AuszugZeile zeile = new AuszugZeile(transaktion, 4201, new DatumFormatter());
    assertThat(zeile.format()).isEqualTo("19.04.2004\t102\t4201");
  }
}
```

Raum für Notizen. Sollten Sie hier Lösungen formulieren, dann machen Sie das für uns kenntlich und markieren Sie für welche Aufgabe sich hier die Lösung befindet.

Raum für Notizen. Sollten Sie hier Lösungen formulieren, dann machen Sie das für uns kenntlich und markieren Sie für welche Aufgabe sich hier die Lösung befindet.