Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
Prof. Dr. Michael Schöttner

7. Fabruar 2023

Hauptklausur

Programmierung
WS 22/23

Nachname:		Vorname:	
Matrikelnummer:		Sitzplatznummer:	
Zusätzliche Blätter:		Unterschrift:	

Hinweise:

- Diese Klausur enthält 22 nummerierte Klausurseiten. Prüfen Sie bitte zuerst, ob die Klausur alle Seiten enthält. Sie dürfen die Heftung der Klausur nicht auftrennen.
- Sie erhalten außerdem von uns leere Blätter. Sollten Sie auf diesen Blättern für die Korrektur relevante Teile Ihrer Lösung notieren, markieren Sie dies deutlich an der entsprechenden Aufgabe und auf dem zusätzlichen Blatt. Schreiben Sie auf alle zusätzlichen Blätter Ihren Namen. Geben Sie außerdem oben auf dem Deckblatt an, wie viele zusätzliche Blätter Sie zur Korrektur abgeben. Wenn Sie weiteres Papier benötigen, melden Sie sich bitte.
- Alle Fachbegriffe in dieser Klausur werden wie in der Vorlesung definiert verwendet. Alle Fragen beziehen sich auf die in der Vorlesung vorgestellte Java-Version 17. Programmcode muss in Java geschrieben werden.
- Antworten dürfen auf Deutsch oder Englisch gegeben werden. Sofern keine ausformulierten Sätze verlangt sind, reichen nachvollziehbare Stichworte als Antwort.
- Zugelassene Hilfsmittel: eine beidseitig beschriebene oder bedruckte DIN-A4-Seite, Wörterbuch (Wörterbücher müssen vor Beginn der Klausur den Aufsichtspersonen zur Kontrolle vorgelegt werden.)
- Schalten Sie Ihr Mobiltelefon aus. 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!

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	7	8	Σ
Punktzahl	9	5	8	24	6	6	4	28	90
Erreicht									

Aufgabe 1

_____ / 9 Punkte

(a) [6 Punkte] Gegeben seien die folgenden Klassen:

Person.java

```java
public class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public toString() {
        return name;
    }
}}
```

Employee.java

```java
public class Employee extends Person {
    private integer salary;

    public Employee(String name, int salary) {
        super(name);
        this.salary = salary;
    }

    @Override
    public String toString() {
        return super.toString() + ", " + salary;
    }
}
```

Beim Compilieren der Klassen gibt es folgende Fehlermeldungen:

```
./Person.java:9: error: invalid method declaration; return type required
    public toString() {
           ^
./Person.java:12: error: class, interface, or enum expected
}}
 ^
Employee.java:2: error: cannot find symbol
    private integer salary;
            ^
  symbol:   class integer
  location: class Employee
./Person.java:8: error: annotation type not applicable to this kind of declaration
    @Override
    ^
./Person.java:10: error: incompatible types: unexpected return value
        return name;
               ^
5 errors
```

Geben Sie an, in welchen Klassen und Zeilen die **Ursachen** für die Fehler sind, beschreiben Sie die Fehler jeweils kurz (max. 1 Satz) und geben Sie die korrigierte Codezeile **vollständig** an, sodass der Code das tut, was bei der Programmierung wahrscheinlich vorgesehen war. Geben Sie keine Folgefehler an, die durch Korrektur eines vorherigen Fehlers behoben werden.

Klasse, Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Klasse, Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Klasse, Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Klasse, Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Klasse, Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

(b) [1 Punkt] Ihr Terminal sieht gerade wie folgt aus:

```
/home/dennis/project % ls
Hello.class Hello.java
```

Die Klasse `Hello` ist bereits kompiliert und enthält eine `main`-Methode. Mit welchem Befehl können Sie diese `main`-Methode mit dem Argument `Info` ausführen? *Geben Sie nur den für das Ausführen notwendigen Befehl an.*

(c) [2 Punkte] Wir haben die Punkte einer Klausur in der Textdatei `punkte` gespeichert. Außerdem haben wir ein bereits kompiliertes Java-Programm `Minimum`, das das Minimum von ganzen Zahlen berechnen und ausgeben kann.

```
% ls
Minimum.class Minimum.java punkte
% cat punkte
80 82 80 70 69 87 47 58 29 50 69
```

Minimum.java Java

```java
import java.util.Scanner;
import java.util.LinkedList;
import java.util.List;

public class Minimum {
    public static void main(String[] args) {
        Scanner eingabe = new Scanner(System.in);
        List<Integer> zahlen = new LinkedList<>();

        // liest Zahlen ein
        while(eingabe.hasNext()) {
            int zahl = eingabe.nextInt();
            zahlen.add(zahl);
        }

        // berechnet das Minimum (den Code müssen Sie nicht nochvollziehen)
        int minimum = zahlen.stream()
            .mapToInt(Integer::intValue)
            .min()
            .orElse(0);

        // gibt das Minimum aus
        System.out.println(minimum);
    }
}
```

Geben Sie einen Aufruf des Programms `Minimum` an, sodass das Minimum der Zahlen in der Datei `punkte` berechnet und auf der Standardausgabe ausgegeben wird.

Aufgabe 2

_____ / 5 Punkte

Formen Sie die Kontrollstrukturen in den folgenden Codeausschnitten um, sodass sich die Semantik des Codes nicht ändert. Benutzen Sie in Ihrem Code jeweils nur die von der Aufgabe vorgegebene Art von Kontrollstruktur.

(a) [2 Punkte] Formen Sie die folgende for-each-Schleife in eine for-Schleife um:

Vorgabe Java

```java
String[] names = {"Kuruk", "Kyoshi", "Roku"};
int index = 0;
for(String name: names) {
    System.out.print(index + ": " + name);
    index++;
}
```

Ihre Lösung Java

```java
String[] names = {"Kuruk", "Kyoshi", "Roku"};
```

(b) [3 Punkte] Formen Sie die folgende switch-Verzweigung in eine if-Verzweigung um:

Vorgabe Java

```java
int ticket = 1;
switch(ticket) {
    case 1:
    case 3:
        System.out.print("2 €");
        break;
    case 2:
        System.out.print("3 €");
        break;
    default:
        System.out.print("err");
}
System.out.print(" bitte.");
```

Ihre Lösung Java

```java
int ticket = 1;









System.out.print(" bitte.");
```

Aufgabe 3

_____ / 8 Punkte

Marlin möchte zum Üben für die RA-Klausur Zweierpotenzen lernen, also die Folge der Zahlen $2^0, 2^1, 2^2, 2^3, \ldots$

Schreiben Sie für Marlin ein Java-Programm `Binaer`, das eine ganze Zahl x als Konsolenargument entgegennimmt und die Zweierpotenzen von 2^0 bis 2^x als **ganze Zahlen** auf der Standardausgabe ausgibt. Die Ausgabe soll die Form `2^5 = 32` haben (siehe Beispiel). Falls **kein Argument** oder **keine ganze Zahl** übergeben wird, soll sich das Programm mit der Ausgabe `err` beenden.

Zur Erinnerung: Die Parse-Methoden werfen im Fehlerfall eine `NumberFormatException`. `Math.pow(a,b)` *berechnet* a^b *und gibt einen* ***Double*** *zurück.*

Beispiel-Aufrufe:

```
% java Binaer 4
2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
% java Binaer -3
% java Binaer
err
% java Binaer zwei
err
```

Binaer.java Java

Binaer.java (Fortsetzung) Java

Aufgabe 4

_____ / 24 Punkte

Aus der Vorlesung kennen Sie folgende Implementierung von Insertion Sort, die ein Array von Integern aufsteigend sortiert:

```java
public static void sort(int[] numbers) {
    for(int currentIndex = 1; currentIndex < numbers.length; currentIndex++) {
        int currentNumber = numbers[currentIndex];
        int insertionPosition = currentIndex;
        while(insertionPosition > 0 && numbers[insertionPosition - 1] > currentNumber) {
            numbers[insertionPosition] = numbers[insertionPosition - 1];
            insertionPosition--;
        }
        numbers[insertionPosition] = currentNumber;
    }
}
```

Gegeben sei die folgende Klasse `Studi`, die Studis mit Namen und Durchschnittsnote repräsentiert.

Studi.java

```java
public class Studi {

    private final String name;
    private final double schnitt;

    public Studi(String name, double notendurchschnitt) {
        this.name = name;
        this.schnitt = notendurchschnitt;
    }

    public double getSchnitt() {
        return schnitt;
    }

    public String getName() {
        return name;
    }
```

(a) [6 Punkte] Vervollständigen Sie die Klassenmethode `nachNote`, die ein Array von `Studi`-Objekten übergeben bekommt und ein neues Array zurückgeben soll, in dem dieselben Objekte **aufsteigend** nach ihrer Durchschnittsnote sortiert sind (beste (also kleinste) Note am Anfang); die Reihenfolge der Objekte im übergebenen Array soll dabei von der Methode **nicht** verändert werden; das Original-Array und das sortierte Array sollen dieselben Objekte im Heap referenzieren.

Falls der Methode `null` übergeben wird, soll eine `IllegalArgumentException` geworfen werden. Sie dürfen davon ausgehen, dass kein Wert im übergebenen Array `null` ist.

Studi.java (Fortsetzung) Java

```
  public static _____________ nachNote(Studi[] studisOrig) {
    if(studisOrig == null) {
      ____________________________________________________________;
    }
    _____________ studis = new Studi[_____________________________];
    for(int i = 0; i < studis.length; i++) {
      ____________________________________________________________;
    }
    for(int i = _____________; i < studis.length; i++) {
      ____________________________________________________________;
      int einfPos = i;
      while(einfPos > 0
       && ____________________________________________________) {
        ____________________________________________________________;
        einfPos--;
      }
      ____________________________________________________________;
    }
    ____________________________________________________;
  }
}
```

Ergänzen Sie die Klasse `Auflistung` um folgende **private, statische** Methoden. Sie dürfen immer davon ausgehen, dass kein Array und kein Array-Wert `null` ist.

(b) [7 Punkte] `Studi[] zugelassen(Studi[])`: Gibt ein neues Studi-Array zurück, das nur genau die Studi-Objekte aus dem übergebenen Array enthält, deren Durchschnittsnote kleiner als 3,0 ist

(c) [4 Punkte] `String[] namen(Studi[])`: Gibt die Namen aller Studi-Objekte im übergebenen Array zurück (lässt Reihenfolge unverändert)

(d) [3 Punkte] `void ausgeben(String[])`: Gibt die übergebenen Strings auf der Standardausgabe aus (lässt Reihenfolge unverändert)

(e) [4 Punkte] Vervollständigen Sie die `main`-Methode (nächste Seite), sodass die Namen aller bereits angelegten Studis mit einer Durchschnittsnote kleiner 3,0 ausgegeben werden, wobei die Studis aufsteigend nach Durchschnittsnote sortiert sind. Sie müssen dabei alle in dieser Aufgabe geschriebenen Methoden verwenden; vergessen Sie nicht, dass `nachNote` in einer anderen Klasse steht. Der Code muss auch dann korrekt funktionieren, wenn die Eigenschaften der drei vorgegebenen Objekte anders wären.

Auflistung.java Java

```
public class Auflistung {
```

Auflistung.java (Fortsetzung) Java

```
    public static void main(String[] args) {
        Studi kim = new Studi("Kim", 2.5);
        Studi marlin = new Studi("Marlin", 3.3);
        Studi sascha = new Studi("Sascha", 1.4);

    }
}
```

Aufgabe 5

_____ / 6 Punkte

Gegeben sei die folgende Klasse `List`, die eine einfach verkettete Liste implementiert, in der Doubles gespeichert werden können:

List.java Java

```
public class List {
    private class Node {
        private double data;
        private Node next;

        private Node(double data, Node next) {
            this.data = data;
            this.next = next;
        }
    }

    private Node head;

    public List(double[] initialValues) {
        for(int i = initialValues.length - 1; i >= 0; i--) {
            head = new Node(initialValues[i], head);
        }
    }

    public String toString() {
        // ... (nicht abgedruckt)

    }

    public void removeNegative() {
        Node current = head;
        while(current != null) {
            while(current.next != null && current.next.data < 0) {
                current.next = current.next.next;
            }
            current = current.next;
        }
    }

}
```

Die Methode `void removeNegative()` soll alle Doubles aus der Liste entfernen, die kleiner als 0 sind. Die Methode funktioniert aber nicht richtig; beim folgenden Testaufruf gibt es eine unerwartete Ausgabe:

Test.java Java

```
public class Test {
    public static void main(String[] args) {
        double[] listElements = {-5.3, -2, 5, 3, -3.3, -2.2};
        List list = new List(listElements);
        System.out.println(list); // erwartet: -5.3, -2.0, 5.0, 3.0, -3.3, -2.2,
        list.removeNegative();
        System.out.println(list); // erwartet: 5.0, 3.0,
    }
}
```

```
% java Test
-5.3, -2.0, 5.0, 3.0, -3.3, -2.2,
-5.3, 5.0, 3.0,
```

Geben Sie eine korrigierte Implementierung der Methode an; alternativ können Sie auch **eindeutig** beschreiben, wie die fehlerhafte Implementierung korrigiert werden kann:

Ihre Lösung Java

```
public void removeNegative() {









}
```

Aufgabe 6

_____ / 6 Punkte

Gegeben sei die Klasse `Tree` für einen binären Suchbaum, in dem Doubles gespeichert werden können:

Tree.java — Java

```java
public class Tree {

    private class BinaryNode {
        private double element;
        private BinaryNode left, right;

        private BinaryNode(double element) {
            this.element = element;
        }
    }

    private BinaryNode root;

    public void insert(double newNumber) {
        // ... (Implementierung nicht abgedruckt)

    }
```

Vervollständigen Sie die Methode `int countLarge()`, die die **Anzahl** aller Einträge im Baum zurückgibt, die **größer oder gleich** 100 sind; bei einem leeren Suchbaum ist die Anzahl gleich 0. Nutzen Sie in Ihrem Code die Eigenschaften eines binären Suchbaumes aus, um die Anzahl der betrachteten Knoten minimal zu halten. Sie dürfen zusätzliche Hilfsmethoden mit minimaler Sichtbarkeit schreiben.

Tree.java (Fortsetzung) — Java

```java
public int countLarge() {
```

Tree.java (Fortsetzung) Java

```
    }
}
```

Aufgabe 7

_____ / 4 Punkte

(a) [1 Punkt] Gegeben sei der reguläre Ausdruck `[ABCD][0-9]+`. Kreuzen Sie alle Strings an, die vollständig von diesem Ausdruck gematcht werden:

- ☐ `a0`
- ☐ `A4`
- ☐ `B10`
- ☐ `A-4`
- ☐ `G7`
- ☐ `5G`

(b) [2 Punkte] Wir wollen in einer Datenstruktur speichern, welche Matrikelnummern sich für den Studiengang Informatik angemeldet haben; die Anzahl der Matrikelnummern ist vorab nicht bekannt, die gespeicherte Reihenfolge ist egal. Später wollen wir sehr oft überprüfen, ob eine Matrikelnummer in der Datenstruktur vorhanden ist. Troy schlägt vor, ein Hashset zu benutzen. Gabi meint, es sollte besser ein Array verwendet werden.

Würden Sie eher Troys oder Gabis Vorschlag umsetzen? Geben Sie einen nachvollziehbaren Grund in 1–2 ausformulierten Sätzen an.

(c) [1 Punkt] Angenommen, Sie wollen eine einfach verkettete Liste in einer Java-Anwendung verwenden. Warum ist es sinnvoller, die JDK-Klasse `java.util.LinkedList<E>` zu verwenden, anstatt eine Liste selbst zu implementieren? Geben Sie einen nachvollziehbaren Grund in 1–2 ausformulierten Sätzen an.

Aufgabe 8

_____ / 28 Punkte

In dieser Aufgabe sollen Sie Klassen und Methoden für den Versand von Briefen und Paketen zwischen verschiedenen Standorten eines Unternehmens programmieren.

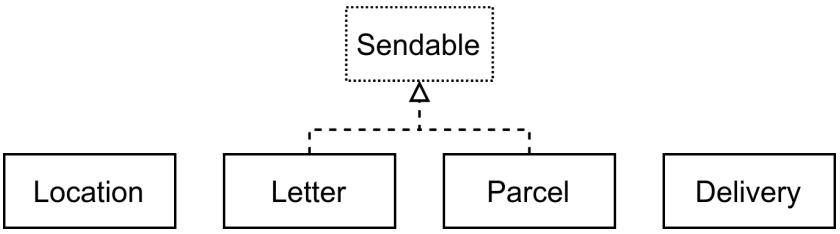
Hinweise:

- Sie dürfen alle Variablennamen frei wählen.
- Wählen Sie sinnvolle Datentypen für Ihre Variablen.
- Alle Instanzvariablen müssen **privat** sein.
- Wenn kein Konstruktorverhalten vorgeschrieben ist, reicht der Default-Konstruktor.
- Das genaue Format von Textausgaben ist Ihnen überlassen.
- Innerhalb dieser Aufgabe müssen Sie keine Exceptions abfangen. Sie müssen keine Parameter validieren.
- Lesen Sie sich die Aufgabenstellung vor Beginn der Implementierung **vollständig** durch, um einen besseren Überblick über das Gesamtbild zu erhalten.
- Gehen Sie davon aus, dass alle in dieser Aufgabe genannten Klassen und Interfaces im selben Package liegen.

Gegeben ist das Interface `Sendable`, das eine einzelne Sendung beschreibt:

Sendable.java

```java
public interface Sendable {
    // gibt den Absende-Standort zurück
    Location senderLocation();
    // gibt den Empfangs-Standort zurück
    Location receiverLocation();
    // gibt das Gewicht einer Sendung in Gramm (g) zurück
    double weight();
}
```



(a) [4 Punkte]

Vervollständigen Sie die Klasse `Location`, die einen Standort mit Postleitzahl (ganze Zahl) und Land (Text) repräsentiert. Der **öffentliche** Konstruktor `Location(int, String)` bekommt diese beiden Informationen übergeben und speichert sie in Instanzvariablen ab. Schreiben Sie **öffentliche** Methoden `getCountry()` und `getPostCode()`, die das Land bzw. die Postleitzahl zurückgeben.

Location.java Java

```
public class Location {









}
```

(b) [6 Punkte]
Schreiben Sie eine nicht abstrakte, **öffentliche** Klasse `Letter`, die das `Sendable`-Interface sinnvoll implementiert. Der öffentliche Konstruktor nimmt Absender- und Empfänger-Standort in Form von `Location`-Objekten entgegen. Jeder Brief wiegt 80 g.

`Letter.java` `Java`

(c) [6 Punkte]
Schreiben Sie eine nicht abstrakte, **öffentliche** Klasse `Parcel`, die das `Sendable`-Interface sinnvoll implementiert. Der öffentliche Konstruktor nimmt das Gewicht, den Absender- und den Empfänger-Standort (`Location`s) entgegen.

Parcel.java Java

(d) [10 Punkte]

Vervollständigen Sie die Klasse `Delivery`, die eine Lieferung bestehend aus einer oder mehreren Sendungen repräsentiert. Ein `Delivery`-Objekt speichert dazu ein Array von `Sendable`-Instanzen. Es gibt zwei **öffentliche** Konstruktoren, über die die Sendungen der Lieferung gesetzt werden:

- `Delivery(Sendable[])` nimmt ein Array von Sendungen entgegen.
- `Delivery(Sendable)` nimmt eine einzelne Sendung entgegen.

Schreiben Sie eine **private, statische** Methode `int postage(Sendable)`, die die Portokosten einer einzelnen Sendung in Euro berechnet:

- Sendungen unter 100 g: 1 €
- Sendungen $\geq$ 100 g: 2 €
- Sendungen, bei denen Empfangs- und Absende-**Land unterschiedlich** sind: zusätzlich 1 €

Schreiben Sie eine **öffentliche** Instanzmethode `int postage()`, die das Gesamtporto für die Sendungen der Lieferung zurückgibt. **(Rückseite beachten)**

Delivery.java Java

```
public class Delivery {
```

Delivery.java (Fortsetzung) Java

```
}
```

(e) [2 Punkte]
Ergänzen Sie die `main`-Methode der Klasse `DispatchOffice`, sodass eine Lieferung bestehend aus dem angelegten `Letter`-Objekt erstellt wird, und das Gesamtporto mithilfe der `Delivery`-Instanz berechnet und ausgegeben wird.

DispatchOffice.java Java

```
public class DispatchOffice {
  public static void main(String[] args) {

    Letter letter = new Letter(new Location(40225, "Germany"),
                               new Location(40530, "Sweden"));

    Delivery delivery = ____________________________________;

    System.out.println(____________________________________);

  }
}
```