Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
Dr. Markus Brenneis

08. Februar 2022

Hauptklausur

Programmierung
WS 21/22

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

Hinweise:

- Diese Klausur enthält 20 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 11. 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	9	Σ
Punktzahl	7	5	2	13	13	8	9	6	27	90
Erreicht										

Aufgabe 1

_____ / 7 Punkte

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

```
/mnt/projects/blatt02 % ls
Hello.java
/mnt/projects/blatt02 %
```

Sie wollen die Klasse `Hello` kompilieren. Welchen Befehl müssen Sie dazu eingeben?

(b) [4 Punkte] Gegeben sei die folgende Klasse:

Person.java Java

```java
public class Person {

    private String name;
    private String mail;

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

    @Override
    public String toString() {
        return name + " " mail;
    }

}
```

Beim Compilieren der Klasse gibt es folgende Fehlermeldungen:

```
Person.java:6: error: <identifier> expected
    public Person(String name, mail) {
                               ^
Person.java:13: error: ';' expected
        return name + " " mail;
                        ^
Person.java:13: error: not a statement
        return name + " " mail;
                         ^
3 errors
```

Geben Sie an, in welchen 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.

Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

Zeilennummer: _____

Fehlerbeschreibung: _____

Korrektur: _____

(c) [2 Punkte] Sie haben eine Reihe von ganzzahligen Messwerten zeilenweise in einer Textdatei `/projekt/werte.txt` gespeichert; ungültige Messwerte sind als -1 gespeichert. Außerdem haben Sie ein Java-Programm `Filter`, das zeilenweise Integer einliest und alle Integer, die nicht -1 sind, wieder zeilenweise ausgibt.

```
/projekt % ls
Filter.class Filter.java werte.txt
/projekt %
```

Geben Sie einen Befehl an, mit dem Sie alle **gültigen** Messwerte aus `werte.txt` in einer Datei `bereinigt.txt` speichern können:

_____

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-Schleife in eine for-each-Schleife um:

Vorgabe — Java

```java
String[] names = {"Alice", "Bob", "Charlie"};
for(int index = 0; index < names.length; index++) {
    System.out.print(names[index]);
}
```

Ihre Lösung — Java

```java
String[] names = {"Alice", "Bob", "Charlie"};
```

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

Vorgabe — Java

```java
int choice = 2;
switch(choice) {
    case 1:
    case 2:
        System.out.print("Feuer");
        break;
    case 3:
        System.out.print("Luft");
        break;
    default:
        System.out.print("ungültig");
}
```

Ihre Lösung — Java

```java
int choice = 2;
```

Aufgabe 3

_____ / 2 Punkte

(a) [1 Punkt] Gegeben sei der reguläre Ausdruck `[a-z0-9]+[a-z]`. Kreuzen Sie alle Strings an, die vollständig von diesem Ausdruck gematcht werden:

- ☐ `123a`
- ☐ `123A`
- ☐ `a123b`
- ☐ `1ab`
- ☐ `1a`
- ☐ `a`

(b) [1 Punkt] Der oben angegebene Ausdruck ist gleichbedeutend mit:

- ☐ `[A-Z0-9]+[A-Z]`
- ☐ `[a-z0-9]*`
- ☐ `[a-z0-9]*[a-z]`
- ☐ `[a-z0-9][a-z0-9]*[a-z]`

Aufgabe 4

_____ / 13 Punkte

Für eine Finanzanwendung soll ein Programm für Zinseszinsrechnung erstellt werden. Vervollständigen Sie dafür die Klasse `Zinsen`:

(a) [3 Punkte] Schreiben Sie eine **private, statische** Methode `endkapital(double k0, double p, int n)`, die berechnet, auf welches Endkapital K_n das Startkapital K_0 nach n Jahren bei einer jährlichen Verzinsung von p gewachsen ist, und K_n zurückgibt. Die Berechnungsvorschrift lautet:

$$K_n = K_0 \cdot (1 + p)^n$$

Zur Erinnerung: `Math.pow(a, b)` berechnet a^b.

(b) [10 Punkte] Legen Sie eine `main`-Methode an, die das Startkapital (Kommazahl größer 0) und den Zinssatz (Kommazahl größer 0) in dieser Reihenfolge als Konsolenargumente entgegennimmt. Es soll dann jeweils das Endkapital nach 0, 1, 2 usw. Jahren ausgegeben werden, bis das Endkapital **größer** als das doppelte Startkapital ist.

Wenn **etwas anderes** als Zahlen, **zu kleine** Zahlen oder **zu wenige** Zahlen angegeben werden, soll die Fehlermeldung „ungültige Eingabe“ ausgegeben werden. *Zur Erinnerung: Die Methoden zum Parsen werfen im Fehlerfall eine* `NumberFormatException`.

Beispiele:

```
% java Zinsen 100 0.05
100.0
105.0
115.7625
134.0095640625
162.88946267774418
207.89281794113683
% java Zinsen 100
ungültige Eingabe
% java Zinsen 100 0
ungültige Eingabe
% java Zinsen 100 zwei
ungültige Eingabe
```

Zinsen.java Java

```
public class Zinsen {
```

Zinsen.java (Fortsetzung) Java

```
}
```

Aufgabe 5

_____ / 13 Punkte

Gehen Sie in dieser Aufgabe davon aus, dass $n \in \mathbb{N}$ und $n \geq 1$ gilt.

Eine natürliche Zahl n ist eine perfekte Zahl, wenn die Summe ihrer natürlichen Teiler (n ausgeschlossen) **gleich** n ist. Die kleinste perfekte Zahl ist 6 (Teilersumme $1 + 2 + 3 = 6$).

Unten ist ein Programm `PerfekteZahlen` mit einer Beispielausgabe vorgegeben. Erweitern Sie dieses Programm um folgende **private**, **statische** Methoden, damit das Programm wie im Beispiel funktioniert:

(a) [4 Punkte] Schreiben Sie eine Methode `int teilersumme(int n)`, welche die Summe aller natürlichen Teiler von `n` berechnet, wobei n selbst **ausgeschlossen** ist.

(b) [3 Punkte] Schreiben Sie eine Methode `boolean istPerfekt(int n)`, die genau dann `true` zurückgibt, wenn `n` eine perfekte Zahl ist.

(c) [6 Punkte] Schreiben Sie eine Methode `int[] perfekteZahlen(int anzahl)`, die ein Array zurückgibt, das genau die ersten `anzahl` perfekten Zahlen enthält. Falls `anzahl < 0` ist, soll eine `IllegalArgumentException` geworfen werden.

Beispiel:

```
% java PerfekteZahlen
6
true
false
6 28
```

PerfekteZahlen.java — Java

```
public class PerfekteZahlen {

    public static void main(String[] args) {
        System.out.println(teilersumme(6));
        System.out.println(istPerfekt(6));
        System.out.println(istPerfekt(12));

        for(int zahl: perfekteZahlen(2)) {
            System.out.print(zahl + " ");
        }
    }

    // Ergänzen Sie hier die Methoden teilersumme, istPerfekt und perfekteZahlen.
```

PerfekteZahlen.java (Fortsetzung) Java

```
}
```

Aufgabe 6

_____ / 8 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 = 0; 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:

Circle.java — Java

```
public class Circle {
    private final double radius;

    public Circle(double r) {
        radius = r;
    }

    // gibt den Flächeninhalt des Kreises zurück
    public double area() {
        return Math.PI * radius * radius;
    }
```

(a) [6 Punkte] Ergänzen Sie die Klasse um eine **öffentliche, statische** Methode `sort`, die ein Array von `Circle`-Objekten übergeben bekommt und **absteigend** nach ihrem Flächeninhalt sortiert; das übergebene Array soll dabei von der Methode verändert werden (die Methode hat also keinen Rückgabewert). Gehen Sie davon aus, dass kein Wert im übergebenen Array `null` ist. Falls der Methode `null` übergeben wird, soll es keine Exception geben (die Methode macht also nichts).

Circle.java (Fortsetzung) — Java

Circle.java (Fortsetzung) Java

```
}
```

(b) [2 Punkte] Angenommen, Sie müssten häufiger 1 Millionen Kreise nach Ihrer Größe sortieren. Würden Sie dafür eine Sortiermethode benutzen, die Insertion Sort verwendet? Begründen Sie Ihre Antwort in 1–2 ausformulieren Sätzen und schlagen Sie ggf. eine Alternative vor.

Aufgabe 7

_____ / 9 Punkte

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

List.java — Java

```java
public class List {
    private class Node {
        private int data;
        private Node next;

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

    private Node head;

    // fügt value vorne in die Liste ein
    public void prepend(int value) {
        head = new Node(value, head);
    }
```

(a) [6 Punkte] Vervollständigen Sie die Implementierung der Methode `int removeNegative()`, die alle Zahlen aus der Liste entfernt, die negativ sind. Der Rückgabewert soll angeben, wie viele Zahlen entfernt worden sind.

List.java (Fortsetzung) — Java

```java
  public int removeNegative() {

    int anzahlLoeschungen = _______________;

    while(head != null && _____________________________) {

      ___________________________________________;

      ___________________________________________;

    }

    ___________________________________________;

    while(_____________________________________) {

      while(current.next != null

            && _____________________________________________) {

        current.next = _____________________________________;

        anzahlLoeschungen++;

      }

      current = ___________________________________________;

    }

    ___________________________________________;

  }
}
```

(b) [3 Punkte] Alice möchte in einem größeren Programmierprojekt verkettete Listen benutzen. Sie schlägt vor, eine generische Klasse `List<T>` selbst zu implementieren. Bob sagt, dass sie lieber die fertige Klasse `java.util.LinkedList<T>` aus dem JDK verwenden sollte.

i) Würden Sie eher den Vorschlag von Alice oder den von Bob umsetzen? Geben Sie einen nachvollziehbaren Grund in 1–3 ausformulierten Sätzen an.

Alice erwidert, dass sie ihre eigene Klasse bevorzugt, weil sie nicht jedes Mal, wenn sie ein LinkedList erstellt, das längliche `new java.util.LinkedList<>()` benutzen möchte.

ii) Halten Sie dieses Argument von Alice für ein nachvollziehbares/starkes Argument? Begründen Sie Ihre Antwort in 1–2 ausformulierten Sätzen.

Aufgabe 8

_____ / 6 Punkte

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

SearchTree.java — Java

```
public class SearchTree {

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

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

    private BinaryNode root;

    // fügt newNumber in den Baum ein
    public void insert(double newNumber) {
        // ... (Implementierung nicht abgedruckt)

    }
```

Vervollständigen Sie die Methode `double sumGreaterThan(double t)`, die die **Summe** aller Einträge im Baum zurückgibt, die **größer** als `t` sind; bei einem leeren Suchbaum ist diese Summe 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.

SearchTree.java (Fortsetzung) — Java

```
public double sumGreaterThan(double t) {
```

SearchTree.java (Fortsetzung) Java

```
    }
}
```

Aufgabe 9

_____ / 27 Punkte

In dieser Aufgabe sollen Sie Klassen und Methoden für ein Computerspiel implementieren.

Hinweise:

- Sie dürfen alle Variablennamen frei wählen.
- Wählen Sie sinnvolle Datentypen für Ihre Variablen.
- Alle Instanzvariablen müssen **privat** sein.
- Das genaue Format der 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 im selben Package liegen.

(a) [8 Punkte]

Schreiben Sie eine **abstrakte, öffentliche** Klasse `Task`, in welcher es eine Objektvariable gibt, die angibt, ob der Task beendet ist oder nicht; standardmäßig sind neu erstellte Tasks nicht beendet. Über eine **öffentliche** Methode `void finish()` soll ein Task auf beendet gesetzt werden können. Mit einer **öffentlichen** Methode `isFinished()` soll abgefragt werden können, ob der Task beendet ist.

Außerdem soll es einen **öffentlichen Konstruktor** geben, dem der Name des Tasks übergeben werden kann, und eine **abstrakte, öffentliche** Methode `getDuration()`, die die Länge des Tasks (in Sekunden) zurückgibt.

Die `toString`-Methode soll überschrieben werden, sodass **Name** und **Länge** des Tasks und ob dieser **beendet** ist oder nicht zurückgegeben werden.

`Task.java` `Java`

(b) [6 Punkte]

Implementieren Sie zwei **nicht abstrakte, öffentliche** Klassen `LongTask` und `ShortTask`, die sinnvoll von `Task` erben. Jeder LongTask dauert 10 Sekunden, jeder ShortTask 5 Sekunden.

Den **öffentlichen** Konstruktoren beider Klassen soll übergeben werden, wie die Tasks heißen; diese Information soll auf geeignete Weise mithilfe der Oberklasse gespeichert werden.

`LongTask.java` `Java`

`ShortTask.java` `Java`

(c) [8 Punkte]

Erstellen Sie eine nicht abstrakte, **öffentliche** Klasse `Player`. Jedes Player-Objekt speichert einen Namen und ein Array von beliebig vielen Tasks, die beide dem **öffentlichen** Konstruktor als Parameter übergeben werden.

Schreiben Sie eine **öffentliche** Objektmethode `finishTask(int number)`, die den Task an der Stelle `number` im Array auf beendet setzt und keinen Rückgabewert hat.

Außerdem soll die `toString`-Methode überschrieben werden, sodass zuerst der Spielername, alle **nicht beendeten** Tasks und **danach** alle **beendeten** Tasks als ein String zurückgegeben werden.

Player.java Java

```
public class Player {









}
```

(d) [5 Punkte]

Ergänzen Sie die `main`-Methode der Klasse `Game`, sodass folgendes passiert:

1. Ein ShortTask mit Namen „A“ und ein LongTask mit Namen „B“ werden erstellt.
2. Ein Player mit Namen „S“ und den Tasks A und B wird erstellt.
3. S beendet genau einen der beiden Tasks (egal welchen).
4. Die String-Repräsentation des Player-Objekts wird ausgegeben.

Game.java Java

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









    }
}
```