HHU Düsseldorf
Institut für Informatik
Dr. Markus Brenneis

Daten des Prüflings

Matrikelnummer: _____

Nachname: _____

Vorname: _____

1. Klausur

Programmierung WS 23/24
6. Februar 2024

- Diese Klausur enthält 19 nummerierte Klausurseiten. Prüfen Sie bitte zuerst, ob die Klausur alle Seiten enthält. Sie dürfen die Heftung der Klausur nicht auftrennen.
- Sie erhalten 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 unten auf dem Deckblatt an, wie viele zusätzliche Blätter Sie zur Korrektur abgeben.
- Fachbegriffe werden wie in der Vorlesung definiert verwendet. Die Aufgaben beziehen sich auf die in der Vorlesung vorgestellte Java-Version 21. 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.
- Erlaubte Hilfsmittel: eine beidseitig beschriebene oder bedruckte A4-Seite, Wörterbuch (Wörterbuch muss vor Klausurbeginn der Aufsicht zur Kontrolle vorgelegt werden.)
- Schalten Sie technische Geräte 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!

Zusätzliche Blätter: _____ Unterschrift: _____

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	7	8	Σ
Punkte	11	5	11	16	5	12	3	27	90
Erreicht									

Aufgabe 1

_____ / 11 Punkte

__/6 (a) Gegeben seien die folgenden Klassen:

Statistik.java

```java
public class Statistik {
    private double[] werte;

    public Statistk(double[] w) {
        this.werte = w;
    }

    public double schnitt() {
        double summe = 0;
        for(int wert: werte) {
            summe += wert;
        }
        return summe / werte.length;
    }
}
```

Main.java

```java
public class Main {
    public static void main(String[] args) {
        double[] noten = {2.0, 1.7, 2.3};
        Statistik statistik = new Statistik(noten);
        // Mithilfe des Statistik-Objekts den Schnitt berechnen und ausgeben
        double s = schnitt(statistik);
        System.out.println(s);
    }
}
```

Beim Compilieren der Klassen gibt es folgende Fehlermeldungen:

```
% javac Main.java
./Statistik.java:4: error: invalid method declaration; return type required
    public Statistk(double[] w) {
           ^
Main.java:6: error: cannot find symbol
        double s = schnitt(statistik);
                   ^
  symbol:   method schnitt(Statistik)
  location: class Main
./Statistik.java:10: error: incompatible types: possible lossy conversion from
   double to int
        for(int wert: werte) {
                      ^
3 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: _____

___/3 (b) Wir haben die Punkte einer Klausur in der Textdatei `punkte.txt` gespeichert. Außerdem haben wir ein Java-Programm `Minimum` geschrieben, das das Minimum beliebig vieler ganzer Zahlen berechnen und ausgeben kann.

```
% ls
Minimum.java punkte.txt
% cat punkte.txt
82 80 70
```

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 an, wie im Terminal mithilfe des Programms `Minimum` das Minimum der Zahlen in der Datei `punkte.txt` berechnet und auf der Standardausgabe ausgegeben werden kann. Denken Sie daran, dass das Programm vorher noch compiliert werden muss.

___/2 (c) Marlin sagt, wir hätten im Programm `Minimum` besser ein Array statt einer verketteten Liste zum Speichern der Zahlen verwenden sollen. Stimmen Sie Marlin zu? Begründen Sie Ihre Antwort mit einem Argument in 1–3 ausformulierten Sätzen.

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.

___/2 (a) Formen Sie die folgende while-Schleife in eine for-each-Schleife um.

Vorgabe Java

```java
String[] names = {"Crowley", "Aziraphale", "Agnes"};
int index = 0;
while(index < names.length) {
    System.out.print(names[index]);
    index++;
}
```

Ihre Lösung Java

```java
String[] names = {"Crowley", "Aziraphale", "Agnes"};
```

___/3 (b) Formen Sie die folgende if-Verzweigung in eine switch-Verzweigung um.

Vorgabe Java

```java
char op = 'x';
if(op == 'x' || op == '*') {
    System.out.print("mal");
} else if(op == '+') {
    System.out.print("plus");
} else {
    System.out.print("unknown");
}
System.out.println(".");
```

Ihre Lösung Java

```java
char op = 'x';



System.out.println(".");
```

Aufgabe 3

_____ / 11 Punkte

Ein Palindrom ist ein Wort, das vorwärts und rückwärts gelesen gleich ist.

Schreiben Sie ein Programm `Palindrom`, das genau einen String als Argument erwartet und ausgibt, ob dieser String ein Palindrom ist:

- Ausgabe: `ja`, wenn der String unter Beachtung von Groß- und Kleinschreibung ein Palindrom ist; ein leerer String ist auch ein Palindrom.
- Ausgabe: `jein`, wenn der String ein Palindrom ist, wenn man Groß- und Kleinschreibung ignoriert
- Ausgabe: `nein`, wenn der String kein Palindrom ist
- Ausgabe: `err`, wenn nicht genau ein Argument übergeben wird

Beispielaufrufe:

```
% java Palindrom
err
% java Palindrom Abba
jein
% java Palindrom a
ja
% java Palindrom xcx
ja
% java Palindrom mima
nein
```

Zur Erinnerung: `"foo".toCharArray()` gibt ein `char`-Array mit den drei Elementen `'f'`, `'o'` und `'o'` zurück. `"FooBar".toLowerCase()` ist `"foobar"`.

Palindrom.java Java

Palindrom.java (Fortsetzung) Java

Aufgabe 4

_____ / 16 Punkte

Vervollständigen Sie die Klasse `MyMath`.

Hinweise:

- `Math.pow(double x, double y)` berechnet x^y und gibt einen `double` zurück.
- `Math.PI` hat den Wert π.
- Für diese Aufgabe dürfen Sie ignorieren, dass `int`-Variablen keine beliebig großen Zahlen speichern können. Die Rückgabetypen können Sie (sinnvoll) frei wählen.

___/5 (a) Implementieren Sie eine **öffentliche, statische** Methode `fac`, die eine nicht-negative, ganze Zahl n als Parameter übergeben bekommt, die Fakultät von n berechnet und zurückgibt. Die Fakultät von n ist definiert als $n! = 1 \cdot 2 \cdot \cdots \cdot n = \prod_{i=1}^{n} i$. Es gilt $0! = 1$. Falls der Methode eine negative Zahl übergeben wird, soll eine `IllegalArgumentException` geworfen werden.

___/$5\frac{1}{2}$ (b) Implementieren Sie eine **private, statische** Methode `sinSmall`, die eine reelle Zahl `x` übergeben bekommt und das Ergebnis der folgenden Formel zurückgibt:

$$\text{sinSmall}(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + \frac{x^{13}}{13!} = \sum_{n=0}^{6} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

(Ignorieren Sie den hinteren Teil mit dem Σ, wenn Sie das Zeichen nicht kennen.)

___/$5\frac{1}{2}$ (c) Implementieren Sie eine **öffentliche, statische** Methode `sin`, die eine reelle Zahl `x` übergeben bekommt und folgendes Ergebnis liefert:

- falls $0 \leq x < \pi$: sinSmall(x)
- falls $x > \pi$: sinSmall(x modulo (2π))
- in allen anderen Fällen: $-\sin(|x|)$

Ihre Lösung Java

```
public class MyMath {
```

Ihre Lösung Java

```
}
```

Aufgabe 5

_____ / 5 Punkte

Gegeben sei die folgende Klasse `List`, die eine einfach verkettete Liste implementiert, in der Integer gespeichert werden können und welche die folgenden fertigen Methoden hat:

List.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 = null;

    public void add(int element) {
        // fügt element am Ende der Liste ein
        // ... (Code nicht abgedruckt)

    }

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

    }
```

Außerdem gibt es folgendes Interface mit zwei fertigen Implementierungen:

Printer.java

```java
public interface Printer {
    void print(int x);
}
```

StarPrinter.java

```java
public class StarPrinter implements Printer {
    public void print(int x) {
        System.out.print("*" + x + "*");
    }
}
```

SquarePrinter.java

```java
public class SquarePrinter implements Printer {
    public void print(int x) {
        System.out.print(x * x);
    }
}
```

Für die Klasse `List` soll eine Instanzmethode `void foreach(Printer p)` implementiert werden, die für jede Zahl `x` in der Liste einmal `p.print(x)` ausführt. Folgendes Beispiel zeigt, wie `foreach` funktionieren soll:

Beispiel — Java

```java
List liste = new List();
liste.add(1);
liste.add(3);
liste.add(-2);
System.out.println(liste); // 1,3,-2

liste.foreach(new StarPrinter()); // *1**3**-2*
liste.foreach(new SquarePrinter()); // Aufgabenteil a)
```

___/1 (a) Was gibt `liste.foreach(new SquarePrinter())` in diesem Beispiel aus?

___/4 (b) Implementieren Sie `foreach`:

List.java (Fortsetzung) — Java

```java
public void foreach(Printer p) {



}
```

Aufgabe 6

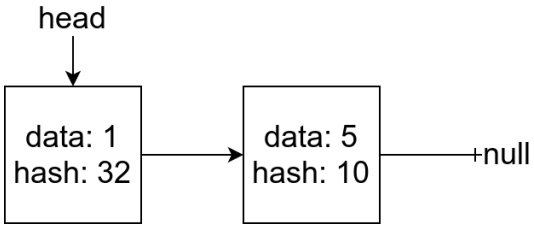
_____ / 12 Punkte

Eine Blockchain ist eine einfach verkettete Liste, bei der nur vorne am Head eingefügt werden kann. Außerdem wird zusätzlich in jedem Knoten (Node) ein sogannter Hashwert gespeichert. Bei uns berechnet sich der Hashwert eines neu eingefügten Knotens wie folgt:

$$(2d + 3h') \text{ modulo } 256$$

Dabei ist d die Zahl, die neu in die Blockchain eingefügt wird, und h' der Hashwert des Knotens, vor dem d eingefügt wird (also der Hashwert des alten Heads). Wird der Hashwert für den ersten Knoten, der zu einer leeren Blockchain hinzugefügt wird, berechnet, setzen wir $h' = 0$.

Beispiel:



Der Hashwert 32 ist das Ergebnis von $(2 \cdot 1 + 3 \cdot 10)$ modulo 256.

___/$5\frac{1}{2}$ (a) Vervollständigen Sie die Objektmethode `void add(int d)`, die vorne einen Knoten mit dem Data-Wert `d` und dem korrekten Hashcode einfügt. Beachten Sie, dass die Blockchain möglicherweise noch leer ist.

___/$6\frac{1}{2}$ (b) Schreiben Sie eine **öffentliche Objektmethode** `boolean verify()`, die für alle Knoten des Blockchain-Objekts prüft, ob der Hashwert korrekt ist. Wenn dies der Fall ist, wird `true` zurückgegeben, ansonsten `false`.

Blockchain.java — Java

```
public class Blockchain {
    private class Node {
        private int data;
        private int hash;
        private Node next;

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

    private Node head = null;

    public void add(int d) {
```

Blockchain.java Java

```
}
```

Aufgabe 7

_____ / 3 Punkte

___/1 (a) Kreuzen Sie alle Strings an, die vollständig vom regulären Ausdruck `[a-z]*[0-9]+[a-z]*` gematcht werden:

- ☐ `abcdef`
- ☐ `abc123def`
- ☐ `a1`
- ☐ `a0a`
- ☐ `123`

___/1 (b) Welche dieser Ausdrücke sind gleichbedeutend mit dem obigen Ausdruck?

- ☐ `[a-z]*[0-9][0-9]*[a-z]*`
- ☐ `[a-z][a-z]+[0-9]+[a-z]*`
- ☐ `[a-z][a-z]+[0-9]+[a-z][a-z]+`

___/1 (c) Der Classpath ist `/mnt`. Wie lautet das erste Statement (≈ die erste Zeile, die kein Kommentar ist) in der Datei `/mnt/de/hhu/progra/Point.java`?

Aufgabe 8

_____ / 27 Punkte

In dieser Aufgabe schreiben wir Klassen für ein kleines Computerspiel.

Hinweise:

- Sie dürfen alle Variablennamen frei wählen.
- Wählen Sie sinnvolle Datentypen für Variablen und Rückgabetypen, wenn es keine genaue Vorgabe gibt.
- 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 Klassen und Interfaces in dieser Aufgabe im selben Package liegen.

Gegeben ist das folgende Interface `Item`:

Item.java — Java

```java
public interface Item {
    int flatBonus();
}
```

___/2 (a) Schreiben Sie eine nicht abstrakte, **öffentliche Klasse** `Sword`, die das Interface `Item` implementiert. Der zurückzugebene Flat-Bonus ist immer 5.

Sword.java — Java

___/$6\frac{1}{2}$ (b) In unserem Spiel gibt es zwei Arten von Entitys: Boss und Player. Jede Entity besitzt Lebenspunkte (Health Points, HP).

Schreiben Sie eine **öffentliche, abstrakte Klasse** `Entity`. Die HP, die eine Entity zu Beginn hat, werden dem **öffentlichen Konstruktor** übergeben und als aktuelle HP gespeichert.

Schreiben Sie außerdem folgende **öffentliche Objektmethoden**:

- `void getDamage(int)` reduziert die HP der Entity um den übergebenen Wert (Schadens-Punkte). Falls die HP nach der Reduktion negativ sind, wird der HP-Wert der Entity auf 0 gesetzt.
- eine **abstrakte** Methode `void attack(Entity other)`, über die einer anderen Entity Schaden zugefügt werden kann; Unterklassen definieren, wie dies genau funktioniert
- überschreiben Sie `toString`, sodass der Rückgabewert die aktuellen HP enthält

Entity.java Java

___/6 (c) Schreiben Sie eine nicht abstrakte, **öffentliche Klasse** `Boss`, die sinnvoll von `Entity` erbt. Der **Konstruktor** nimmt die Start-HP entgegen und speichert sie mithilfe der Oberklasse.

Überschreiben Sie die Methode `attack(Entity other)`, sodass `other` 10 Schadenspunkte zugefügt werden.

Überscheiben Sie `toString`, sodass der Rückgabewert die aktuellen HP und „Boss“ enthält.

Boss.java Java

___/$8\frac{1}{2}$

(d) Schreiben Sie eine nicht abstrakte, **öffentliche Klasse** `Player`, die sinnvoll von `Entity` erbt. Der **Konstruktor** nimmt die Start-HP entgegnen und speichert sie mithilfe der Oberklasse.

Ein Player kann maximal ein Item besitzen. Mit der **öffentlichen** Objektmethode `void setItem(Item)` kann dieses Item gesetzt werden. Ein evtl. bereits vorhandenes Item wird dabei überschrieben.

Überschreiben Sie die Methode `attack(Entity other)`, sodass `other` 3 Schadenspunkte zzgl. dem Flat-Bonus eines evtl. vorhandenen Items zugefügt werden. (Besitzt der Player beispielsweise das `Sword`, werden `other` $3 + 5$ Schadenspunkte hinzugefügt.)

Überscheiben Sie `toString`, sodass der Rückgabewert die aktuellen HP und „Player“ enthält.

`Player.java` `Java`

___/4 (e) Vervollständigen Sie die folgende `main`-Methode, sodass folgendes passiert:

- eine `Player`-Instanz mit 100 HP wird angelegt
- eine `Boss`-Instanz mit 100 HP wird angelegt
- eine `Sword`-Instanz wird angelegt
- die angelegte `Player`-Instanz erhält das `Sword` als Item
- der Player greift den Boss an (die Boss-Instanz verliert also HP)
- die String-Repräsentation des Boss-Objekts wird auf der Standardausgabe ausgegeben

Demo.java Java

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









    }
}
```