HHU Düsseldorf
Institut für Informatik
Dr. Markus Brenneis

*Daten des Prüflings*

Matrikelnummer: _____

Nachname: _____

Vorname: _____

# 2. Klausur

**Programmierung WS 23/24**
**22. März 2024**

- 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!

☐ Falls ich diese Klausur bestehe, möchte ich **nicht** automatisch zum Programmierpraktikum 1 im kommenden Sommersemester angemeldet werden.

Zusätzliche Blätter: _____ **Wir wünschen Ihnen viel Erfolg!**

---

Diesen Teil bitte nicht ausfüllen:

| Aufgabe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Σ |
|---|---|---|---|---|---|---|---|---|---|
| Punkte | 8 | 5 | 2 | 12 | 14 | 13 | 8 | 28 | 90 |
| Erreicht | | | | | | | | | |

## Aufgabe 1

_____ / 8 Punkte

___/5 (a) Gegeben seien die folgenden Klassen:

Resistor.java

```java
public class Resistor {
    private final r;

    public Resistor(double r) {
        this.r = r;
    }

    public String toString() {
        return "Resistor " + r + " Ohm";
    }
}
```

Main.java

```java
public class Main {
    public static void main(String[] args) {
        Resistor resistor = new Resistor(4.5);
        // sollte "Resistor 4.5 Ohm" ausgeben:
        System.out.print(Resistor.toString());
    }
}
```

Beim Compilieren der Klassen gibt es folgende Fehlermeldungen:

```
% javac Main.java
./Resistor.java:2: error: <identifier> expected
    private final r;
                  ^
./Resistor.java:2: error: cannot find symbol
    private final r;
                  ^
  symbol:   class r
  location: class Resistor
Main.java:5: error: non-static method toString() cannot be referenced from a static
   context
        System.out.println(Resistor.toString());
                                   ^
./Resistor.java:5: error: cannot find symbol
        this.r = r;
            ^
  symbol: variable r
./Resistor.java:9: error: cannot find symbol
        return "Resistor " + r + " Ohm";
                             ^
  symbol:   variable r
  location: class Resistor
5 errors
```

Geben Sie auf der nächsten Seite 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 ein Java-Programm `Quadrat` geschrieben, das beliebig viele ganzen Zahlen quadrieren kann.

```
% ls
Quadrat.java zahlen
% cat zahlen
8 -5 4
```

Quadrat.java Java

```java
public class Quadrat {
    public static void main(String[] args) {
        for(String arg: args) {
            int zahl = Integer.parseInt(arg);
            System.out.println(zahl * zahl);
        }
    }
}
```

Geben Sie Konsolen-Befehle an, sodass mithilfe des Programms `Quadrat` die Quadrate der Zahlen 8, −5, 4 berechnet und in der Datei `ergebnisse.txt` gespeichert werden. Denken Sie daran, dass das Programm vorher noch compiliert werden muss.

_____

_____

_____

_____

## 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 for-Schleife in eine for-each-Schleife um.

Vorgabe Java

```
int[] digits = {7, 1, 8, 2, 8, 1, 8};
for(int i = 0; i < digits.length; i++) {
    System.out.print(digits[i]);
}
```

Ihre Lösung Java

```
int[] digits = {7, 1, 8, 2, 8, 1, 8};
```

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

Vorgabe Java

```
int zahl = 1;
String wort;
if(zahl == 0) {
    wort = "kein";
} else if(zahl == 1) {
    wort = "ein";
} else {
    wort = "viele";
}
System.out.println(wort);
```

Ihre Lösung Java

```
int zahl = 1;
String wort;



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

## Aufgabe 3

_____ / 2 Punkte

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

- ☐ `abcdef`
- ☐ `abc123def`
- ☐ `a1`
- ☐ `a1a`
- ☐ `123`

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

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

## Aufgabe 4

_____ / 12 Punkte

Die letzte Ziffer der Matrikelnummern ist eine Prüfziffer, die sich aus den ersten sechs Ziffern berechnen lässt. Für eine Matrikelnummer $a_6a_5a_4a_3a_2a_1a_0$ gilt:

$$a_0 = 9a_6 + 7a_5 + 3a_4 + 9a_3 + 7a_2 + 3a_1 \text{ modulo } 10$$

Gilt diese Beziehung nicht, ist die Matrikelnummer ungültig.

Beispiel: Die Matrikelnummer $\underline{2185237}$ ist gültig, denn

$$\begin{aligned}\underline{7} &= (9 \cdot \underline{2} + 7 \cdot \underline{1} + 3 \cdot \underline{8} + 9 \cdot \underline{5} + 7 \cdot \underline{2} + 3 \cdot \underline{3}) \text{ modulo } 10 \\ &= 117 \text{ modulo } 10\end{aligned}$$

___/4 (a) Schreiben Sie eine **private, statische** Methode `xteZiffer(int zahl, int x)`, die die `x`-te Ziffer **von hinten** der Zahl `zahl` zurückgibt; die Zählung beginnt mit $x = 0$. Sie dürfen davon ausgehen, dass die Zahl positiv ist und mindestens $x$ Ziffern hat.
Beispiel: `xteZiffer(6789, 2) == 7`

*Tipps:*

- `Math.pow(10, 2) == 100.0`
- `12365 / 100 == 123`
- `123 % 10 == 3`

___/4 (b) Schreiben Sie eine **private, statische** Methode `pruefziffer(int nr)`, die für eine gegebene (eventuell ungültige) Matrikelnummer die Prüfziffer gemäß der Formel oben berechnet und zurückgibt. Sie dürfen davon ausgehen, dass die übergebene Zahl positiv ist und genau sieben Ziffern hat.
Beispiele: `pruefziffer(2185237) == 7`, `pruefziffer(2185230) == 7`

___/4 (c) Schreiben Sie eine **öffentliche, statische** Methode `istMatrikelnummer(int nr)`, die genau dann `true` zurückgibt, wenn

- die übergebene Zahl positiv ist,
- genau sieben Ziffern hat und
- die Prüfziffer korrekt ist.

Andernfalls wird `false` zurückgegeben.

Matrikelnummer.java Java

```
public class Matrikelnummer {
```

Matrikelnummer.java (Fortsetzung) Java

```
}
```

## Aufgabe 5

_____ / 14 Punkte

Vervollständigen Sie die Klasse `Exp`.

*Hinweise:*

- `Math.pow(double x, double y)` berechnet $x^y$ und gibt einen `double` zurück.
- Die Parse-Methoden werfen im Fehlerfall eine `NumberFormatException`.
- Für diese Aufgabe dürfen Sie ignorieren, dass `int`-Variablen keine beliebig großen Zahlen speichern können. Die Rückgabetypen können Sie (sinnvoll) frei wählen.

___/$3\frac{1}{2}$ (a) Implementieren Sie eine **private, 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 $0! = 1$. Gehen Sie davon aus, dass das übergebene $n \geq 0$ ist; werfen Sie keine Exceptions.

___/4 (b) Implementieren Sie eine **öffentliche, statische** Methode `exp`, die eine reelle Zahl `x` übergeben bekommt und das Ergebnis der folgenden Formel zurückgibt:

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^9}{9!} = \sum_{n=0}^{9} \frac{x^n}{n!}$$

*(Ignorieren Sie den letzten Teil mit dem $\Sigma$, wenn Sie das Zeichen nicht kennen.)*

___/$6\frac{1}{2}$ (c) Implementieren Sie die `main`-Methode, sodass folgendes passiert:

- Wird nicht **genau ein** Argument übergeben, beendet sich das Programm mit der Ausgabe `err`.
- Ist das übergebene Argument **keine reelle Zahl** oder ist **kleiner** als $-2$ oder **größer** als 4, beendet sich das Programm mit der Ausgabe `err`.
- In allen anderen Fällen wird der Wert von `exp(x)` auf der Standardausgabe ausgegeben.

Exp.Java Java

```
public class Exp {
```

Exp.Java (Fortsetzung) Java

```
}
```

## Aufgabe 6

_____ / 13 Punkte

___/2 (a) Angenommen, Sie speichern 1 Mio. Zahlen in einer Datenstruktur. Die Zahl an Position 100.000 soll um 1 erhöht werden. Ist diese Operation in einer einfach verketteten Liste oder einem Array wesentlich langsamer? Begründen Sie Ihre Antwort mit einem Argument in 1–3 ausformulierten Sätzen.

Eine ArrayList ist eine Liste, die bestimmte Vorteile (und Nachteile) von Arrays und Listen kombiniert.

In unserer Klasse `ArrayList` können `double`-Werte gespeichert werden. Die Werte der Liste werden im Array `data` gespeichert: Das erste Element der Liste ist an Index 0, das zweite an Index 1 usw. Das Array `data` ist ggf. länger als die Liste Elemente hat. Wie viele Elemente gerade in der Liste sind, ist in `size` gespeichert.

ArrayList.java — Java

```java
public class ArrayList {
    private double[] data;
    private int size = 0;

    public ArrayList() {
        this.data = new double[2]; // ein frisch erstelltes Array enthält 0en
    }
```

Wenn eine Zahl zur `ArrayList` hinzugefügt wird, wird sie hinten in `data` an der Position `size` gespeichert und `size` um 1 erhöht. Hat das Array `data` keinen Platz mehr, werden vorm Einfügen alle gespeicherten Zahlen in ein neues, doppelt so großes Array kopiert und `data` durch dieses neue Array ersetzt.

Das folgende Beispiel zeigt, wie die `ArrayList` funktionieren soll und welche Daten in dem Objekt gespeichert sind:

Beispiel — Java

```java
ArrayList zahlen = new ArrayList(); // size: 0, data: {0, 0}
zahlen.add(1.5);                    // size: 1, data: {1.5, 0}
zahlen.add(5.25);                   // size: 2, data: {1.5, 5.25}
zahlen.add(-2.5);                   // size: 3, data: {1.5, 5.25, -2.5, 0}

System.out.println(zahlen.get(0)); // 1.5
System.out.println(zahlen.get(2)); // -2.5
System.out.println(zahlen.get(3)); // wirft IndexOutOfBoundsException
```

___/4 (b) Implementieren Sie die **öffentliche Objektmethode** `double get(int index)`, die die Zahl am gegebenen Index der Liste zurückgibt. Falls der übergebene Index negativ ist oder größer oder gleich der Listenlänge ist, soll von Ihrem Code eine `IndexOutOfBoundsException` geworfen werden.

___/7 (c) Implementieren Sie die **öffentliche Objektmethode** `void add(double element)`, die die übergebene Zahl hinten in die Liste einfügt, `size` aktualisiert, und dabei wie oben beschrieben ggf. das `data`-Array vergrößert.

ArrayList.java (Fortsetzung) Java

```
}
```

## Aufgabe 7

_____ / 8 Punkte

Gegeben sei die folgende Klasse `Tree`, die einen binären Suchbaum implementiert, in dem Integer gespeichert werden können, und die folgende fertige Methoden hat:

Tree.java

```java
public class Tree {
    private class BinaryNode {
        private int element;
        private BinaryNode left, right;

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

    private BinaryNode root;

    public void insert(int newNumber) {
        // ... (Code nicht abgedruckt)

    }

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

    }
```

Wie aus der Vorlesung bekannt befindet sich ganz links im Baum die kleinste gespeicherte Zahl.

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 `Tree` soll eine Instanzmethode `void foreach(Printer p)` implementiert werden, die für jede Zahl `x` im Baum einmal `p.print(x)` ausführt; die Elemente werden dabei in aufsteigender Reihenfolge betrachtet.

Folgendes Beispiel zeigt, wie `foreach` funktionieren soll:

Beispiel — Java

```
Tree tree = new Tree();
tree.insert(1);
tree.insert(-2);
tree.insert(3);
tree.insert(-3);
System.out.println(tree); // -3,-2,1,3
tree.foreach(new StarPrinter()); // *-3**-2**1**3*
tree.foreach(new SquarePrinter()); // 9419
// für Aufgabenteil b) - gibt die Absolutbeträge der Zahlen aus
tree.foreach(new AbsPrinter()); // 3213
```

___/5 (a) Implementieren Sie `foreach`; Sie müssen dabei eine private, rekursive Hilfsmethode benutzen:

Tree.java (Fortsetzung) — Java

```
public void foreach(Printer p) {




}
```

___/3 (b) Schreiben Sie die Klasse `AbsPrinter`, sodass sie wie im Beispiel oben verwendet werden kann.

AbsPrinter.java — Java

## Aufgabe 8

_____ / 28 Punkte

In dieser Aufgabe schreiben wir Klassen für einen kleinen Vokabeltrainer.

**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 in dieser Aufgabe geschriebenen Klassen im selben Package liegen.

__/10 (a) Vervollständigen Sie die Klasse `Vokabel`, die eine Vokabel bestehend aus Frage, Antwort und der Informationen, ob sie beim letzten Abfragen richtig beantwortet wurde, besteht.

- Schreiben Sie einen **öffentlichen Konstruktor**, der Frage und Antwort entgegennimmt und abspeichert. Außerdem wird gespeichert, dass die Vokabel beim letzten Abfragen nicht richtig beantwortet wurde.
- Schreiben Sie zwei **öffentliche Methoden** `getFrage()` und `getAntwort()`, die die Frage bzw. Antwort zurückgeben.
- Überschrieben Sie **toString**, sodass alle im `Vokabel`-Objekt gespeicherten Informationen zurückgegeben werden.
- Schreiben Sie eine **öffentliche Objektmethode** `void antworten(String antwort)`, die prüft, ob die übergebene Antwort mit der im Objekt gespeicherten Antwort übereinstimmt. Stimmen die Antworten überein, soll `Richtig!` auf der Standardausgabe ausgegeben werden, ansonsten `Falsch`. In beiden Fällen soll die Informationen, ob die Vokabel zuletzt richtig beantwortet wurde, entsprechend aktualisiert werden.

Vokabel.java Java

```
public class Vokabel {









}
```

___/6 (b) Unser Vokabeltrainer wird zwei verschiedene Abfrage-Modi haben. Vervollständigen Sie die abstrakte Klasse `Abfrage`:

- Die vorgegebene Methode `antwortErfragen` benutzt die Klasse `java.util.Scanner`. Fügen Sie an der richtigen Stelle das Statement ein, das erforderlich ist, um `Scanner` wie im vorgegebenen Code angegeben verwenden zu können.
- Die Klasse speichert ein Array von `Vokabel`-Objekten, die dem **öffentlichen Konstruktor** übergeben werden.
- Schreiben Sie eine **öffentliche, abstrakte** `void`-Methode `abfrage`, die genau ein `Vokabel`-Objekt als Parameter nimmt.
- Schreiben Sie eine **öffentliche Objektmethode** `abfragen`, die für alle gespeicherten `Vokabel`-Objekte zuerst die `abfrage`-Methode aufruft, dann `antwortErfragen()` aufruft und dann auf dem jeweiligen `Vokabel`-Objekt die Methode `antworten` aufruft, wobei der Parameter die Rückgabe von `antwortErfragen()` ist.

Abfrage.java Java

```
public abstract class Abfrage {

    private String antwortErfragen() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextLine();
    }
}
```

___/8 (c) Implementieren Sie zwei nicht abstrakte, **öffentliche Klassen** `SalatAbfrage` und (auf der nächsten Seite) `NormaleAbfrage`, die sinnvoll von `Abfrage` erben. Die Konstruktoren nehmen jeweils ein Array von `Vokabel`-Objekten entgegnen und speichern es mithilfe der Oberklasse. Beide Klassen überschreiben die Methode `abfrage(Vokabel)` wie folgt:

- `SalatAbfrage`: gibt die in dem übergebenen `Vokabel`-Objekt gespeicherte **Antwort** aus, wobei die Buchstaben alphabetisch sortiert sind.
- `NormaleAbfrage`: gibt die in dem übergebenen `Vokabel`-Objekt gespeicherte **Frage** auf der Standardausgabe aus.

*Hinweise:*

- `"foo".toCharArray()` gibt ein Array mit den Elementen `'f'`, `'o'` und `'o'` zurück.
- `void java.util.Arrays.sort(char[])` sortiert das übergebene Array alphabetisch, indem das übergebene Array verändert wird.

`SalatAbfrage.java` `Java`

NormaleAbfrage.java Java

```
```

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

- eine Instanz von `SalatAbfrage` wird erstellt; dabei werden die drei bereits angelegten `Vokabel`-Objekte auf geeignete Weise übergeben
- die Instanzmehode `abfragen` wird aufgerufen
- die String-Repräsentation von `v1` wird auf der Standardausgabe ausgegeben

Demo.java Java

```
public class Demo {
    public static void main(String[] args) {
        Vokabel v1 = new Vokabel("Urlaub", "semester");
        Vokabel v2 = new Vokabel("Stunde", "timme");
        Vokabel v3 = new Vokabel("Krankenpfleger", "sjuksköterska");









    }
}
```