

## Aufgabe 1

Datentypen

## Aufgabe 2

Casting von Datentypen

## Aufgabe 3

- a) Quellcode lesen

```
Public static int f(int[] myArray, int parameter){
```

```
    Int value=0;
```

```
    Switch(parameter)
```

```
        Case 1
```

```
            For(int x:myArray){
```

```
                Value+=x*x;
```

```
}
```

```
        Case 0
```

```
            For(int x:myArray){
```

```
                Value+=x;
```

```
}
```

```
        Return value;
```

```
}
```

- b) Code aus 3 ohne switch und for Schleife analog implementieren

## Aufgabe 4

Arrays

Schreibe static boolean g(int[] myArray), so dass true herausgegeben wird, wenn mind. ein Eintrag von myArray genau 2 mal vorkommt, sonst false.

## Aufgabe 5

Rekursion mit String

Schreibe Methode g(String s) so, dass

- $3 \cdot g(t)$ , wenn  $s = "a" + t$
- $5 \cdot g(t)$ , wenn  $s = "b" + t + "c"$
- $s.length()$ , sonst [s könnte auch „ sein]

## Aufgabe 6

Finde heraus ob ein String s eine Aneinanderkettung zweier identischer Strings t ist ( $s.equals(t+t)$ ). Falls ja, gib t an, ansonsten gib die Nullreferenz zurück.

### **Aufgabe 7**

Collections, Klassen. Klassen Student und Professor. Professor hat int officeHours, String domain, String namen. Student hat String domain, namen und int time. Es geht darum, dass eine Sprechstunde beim Prof abgebildet werden soll. Dazu werden der Reihe nach in der Main Studenten erzeugt, die eine gewisse Zeit time für ein Gespräch mit dem Prof brauchen. Dies wird aber nur gemacht, wenn die Domains von Prof und Student übereinstimmen, falls nicht, wird ein Text der Art: „Professor „+name+“ ist nicht zuständig für“ student.name.

Ist die Domain identisch, dann reduziert sich die verfügbare Zeit officeHours des Professors um die Zeit, die der entsprechende Student benötigt. Solange der Professor noch Zeit hat, wird der nächste Student zur Sprechstunde zugelassen und es wird ausgegeben „Professor „+name+“ hilft gerade “ +student.name.

Hat der Prof keine Zeit mehr Ausgabe „Leider keine Zeit mehr“

### **Aufgabe 8**

Exceptions, Klassen

Int isbn, String title, String abstract in der Form „123;asdas;opio“ übergeben. Exceptions, wenn Part1 kein Int ist, oder einer der durch „;“ getrennten tokens leer, oder null übergeben wird,...

### **Aufgabe 9**

Vererbung Kamera

Abstract Klasse Kamera, Vererbung auf Klassen Smartphone, und Spiegelreflex

### **Aufgabe 10**

Klasse List analog Vorlesung.

# Aufgaben Programmierung I

## Wintersemester 2007/2008

Schreiben Sie eine Java-Applikation, die die folgende Problemstellung löst und auf dem Bildschirm ausgibt. Es soll eine Klassenstruktur für eine Mediathek entwickelt werden, die es erlaubt Medien wie DVDs, CDs etc. zu erfassen. **Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten beginnen.**

Aufgabe 41: Medium

Schreiben Sie eine abstrakte Klasse `Medium`, die für jedes Objekt eine eindeutige Identifikationsnummer und einen Titel speichert. Die Identifikationsnummern sollen dabei iterativ generiert werden (d.h. das erste erzeugte Objekt soll die Nummer 1 bekommen, das zweite die 2, usw.). Darüber hinaus sollen Methoden `boolean istVerliehen()` und `setVerliehen(boolean)` bereitgestellt werden.

Aufgabe 42: CD/DVD

Leiten Sie von der Klasse `Medium` eine Klasse `DVD` ab, die zusätzlich den Regisseur und die Spielzeit der DVD speichert. Schreiben Sie die entsprechenden `get`-Methoden. Das Setzen der Klassenvariablen sollte hingegen nur im Konstruktor möglich sein. Analog dazu leiten Sie ebenfalls von `Medium` eine Klasse `CD` ab, die den Künstler und die Spielzeit speichert.

Aufgabe 43: Mediathek I

Eine Mediathek repräsentiert in unserem Falle eine Menge beliebiger Größe von verschiedenen Medien. Dazu werden die Medien in einer doppelt verketteten Liste (unsortiert) verwaltet.

- Schreiben Sie deshalb eine Klasse `MedListElem`, deren Objekte ein Listenelement mit einem Wert vom Typ `Medium` darstellen.
- Schreiben Sie eine Klasse `Mediathek`, die eine Methode zum einfachen Einfügen am Ende `void add(Medium m)`, eine zum Löschen eines Mediums aus der Liste `void delete(Medium m)` und eine zum Auffinden des entsprechenden Listenelements `MedListElem lookUp(Medium m)` bereitstellt. Die `lookUp`-Methode soll dabei außerhalb der Klasse nicht sichtbar sein. Des Weiteren sollten Sie eine Methode `String toString()` in der Klasse `Medienliste` schreiben, die die Ausgabe der Liste als String erlaubt. Die Verwendung einer Implementierung vom Typ `Collection`, d.h. von `ArrayList` u.ä., ist **nicht** zulässig.

#### Aufgabe 44:

#### Mediathek II

Erweitern sie ihre Klasse **Mediathek**, so dass der folgende Sachverhalt modelliert wird:

- Eine Mediathek hat einen Namen, der über den Konstruktor festgelegt wird. Das Auslesen soll mittels `getName()` erfolgen.
- Erweitern Sie die Methode `toString()`, so dass zusätzlich der Name der Mediathek ausgegeben wird.
- Schreiben Sie eine Methode `Mediathek find(String str)` zum Suchen nach Medien anhand des Titels. Der Methode soll ein `String str` übergeben werden und in einer Liste alle Medien zurückliefern, die diesen String `str` im Titel enthalten. Der Rückgabewert soll den Namen der Mediathek in der gesucht wird zusammen mit der gesuchten Zeichenkette `str` tragen.

#### Aufgabe 45:

#### MediaTest

Schreiben Sie eine Klasse **MediaTest** zum Testen der bereits entwickelten Klassen **Medium**, **DVD**, **CD**, **Mediathek**. Legen Sie dazu eine Mediathek an und ordnen Sie dieser eine beliebige Anzahl an Medien zu, sowohl CDs als auch DVDs. Geben Sie die eingegebenen Objekte der Mediathek aus. Durchsuchen Sie die Mediathek nach einem von Ihnen festgelegten Suchterminus und geben die erhaltenen Medien iterativ aus.

#### Aufgabe 46:

#### Mediathek III

Schreiben Sie in der Klasse **Mediathek** eine Methode `boolean ausleihen(Medium m)`, die eine Ausnahme `NoSuchElementException` wirft, falls das Medium `m` nicht in der Mediathek enthalten ist. Die Methode liefert `true` zurück, falls das Medium erfolgreich ausgeliehen werden konnte. Analog dazu soll eine Methode `boolean zurueckgeben(Medium m)` bereitgestellt werden, die ebenfalls eine Ausnahme `NoSuchElementException` wirft.

# Grundlagen der Programmierung

Norbert Müller

Universität Trier – Fachbereich IV – Informatik

20. Februar 2024

Bearbeitungszeit: **120 Minuten**

Name	_____
Matr.Nr.	_____
Rechner	_____

- Hiermit bestätige ich, dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur zur Vorlesung Grundlagen der Programmierung (Programmierung I) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.
- Ich fühle mich gesundheitlich in der Lage, die Klausur anzutreten.

Ort, Datum: \_\_\_\_\_

Unterschrift: \_\_\_\_\_

## Beachten Sie folgende Regelungen:

- Als Hilfsmittel sind nur die zur Verfügung gestellten Editoren (inkl. Java-Dokumentation) sowie die elektronisch zur Verfügung gestellten Unterlagen (Kurs zur Klausur auf der moodle-Plattform) erlaubt. Andere Hilfsmittel wie z.B. das Einloggen in andere Moodle-Kurse oder die Verwendung von Smartphones sind nicht erlaubt.
- In der Klausur sind **112 Punkte** erreichbar. Ziel ist, dass Sie davon etwa **100 Punkte** bearbeiten. Die Zahl der Punkte pro Aufgabe entspricht also in etwa der Bearbeitungszeit (in Minuten).
- Alle Lösungen sind über das Moodle-System abzugeben, d.h. dort abzuspeichern.
- Programme können i.d.R. unter Moodle auch getestet und vor-evaluierter werden. Die Testfälle werden bei der Korrektur allerdings durch neue Fälle ersetzt werden! Abgaben, die nicht compilierbar sind, führen zu deutlichen Punktabzügen!

Aufgabe	1	2	3	4	5	6	7	8	9	10	Gesamt
Möglich	7	6	14	8	8	12	15	15	12	15	112
Erreicht											

Note: \_\_\_\_\_

## 1. Aufgabe: Primitive Datentypen und Termauswertung (7 P)

Bei folgenden Deklarationen von Variablen

```
float eins = 1.0f; int zwei = 2; long drei = 3L; double vier = 4.0d;
```

soll der folgende Ausdruck ausgewertet werden:

```
drei / zwei - eins / zwei * drei + vier  
(a)   (b)   (c)   (d)   (e)
```

Geben Sie zunächst an, wie dieser Ausdruck (entsprechend der Auswertungsregeln in Java) ausgewertet wird, indem Sie im Ausdruck 5 öffnende und 5 schließende Klammern hinzufügen.

Beispiel: `eins + zwei + drei` würde ausgewertet als `((eins + zwei) + drei)`

Geben Sie zudem an, welchen Typ und welchen Wert die Zwischenresultate der Operationen (a) bis (e) **bei der von Ihnen gewählten Klammerung** an den jeweiligen Positionen haben. Verwenden Sie für die Angabe des Wertes immer Literale des jeweiligen Typs.

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

## 2. Aufgabe: Deklarationen und Casts (6 P)

Betrachten Sie die im folgenden angegebenen kurzen Abschnitte aus (evtl. nicht kompilierbaren) Java-Quelltexten. Geben Sie bei den syntaktisch korrekten Textabschnitten den jeweils zugewiesenen Wert an. Bei syntaktisch nicht korrekten Texten geben Sie bitte an, warum sie nicht korrekt sind.

1. boolean bit = (12 != 34);
2. int value = 1.234;
3. double val = (int) 123.4;
4. int byte = 123;
5. string text = "1234";
6. int array[] = {12,34};

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

### 3. Aufgabe: Quelltext verstehen und umformulieren

Betrachten Sie die folgende Methode:

```
public static int f( int[] data, int value ) {

    if ( data == null ) throw new IllegalArgumentException();

    int i = data.length - 1;

    while ( i != 0 ) {
        i--;
        if ( data[i] == value ) return 0;
    }
    return 1;
}
```

- (a) Erklären Sie in Worten, bei welchen Kombinationen der Parameter `data` und `value` diese Methode den Wert 1 zurückgibt, wann den Wert 0 und welche anderen Programmverläufe es gibt. Drei oder vier ordentlich formulierte Sätze sollten dazu reichen. Achten Sie auf Sonderfälle! (8 P)

Beschreiben Sie dabei *nicht* den Ablauf des Algorithmus, also nicht die einzelnen Zeilen des Quelltextes! Sie sollen stattdessen nur das beim Aufruf sichtbare Verhalten beschreiben.

- (b) Implementieren Sie den Algorithmus aus `f` analog neu in einer Methode `g`: (6 P)
- Die neue Methode `g` muss für alle(!) möglichen Parameter die gleichen Resultate wie `f` liefern.
  - Dabei darf `g` jedoch *keine* `while`-Anweisung enthalten.

Den Quelltext gibt es in Moodle sowohl unter Aufgabe 03a als auch unter Aufgabe 03b. Ihre Lösung zu Aufgabenteil (a) soll als Freitext in Aufgabe 03a eingetragen werden, die Lösung zu Teil (b) soll unter A3b.java bei Aufgabe 03b in Moodle gespeichert werden; eine zusätzliche Klasse Evaluation.java zum Test einfacher Beispiele (aber nicht aller interessanten Fälle!) ist dort vorgesehen.

### 4. Aufgabe: Strings

(8 P)

Schreiben Sie eine Funktion `stringTest`, die einen formalen Parameter `String s` hat und testet, ob `s` die Verkettung `t + t` eines anderen (zunächst unbekannten und von Ihnen zu bestimmenden) Strings `t` ist. Wenn dies der Fall ist, soll `t` zurückgegeben werden, ansonsten die `null`-Referenz.

**Beispiele:**

```
stringTest("xyzxyz") ~ "xyz"
stringTest("1111111111") ~ "11111"
stringTest("ababab") ~ null
```

Achten Sie darauf, dass der Parameter `s` hier sogar `null` sein kann!

(Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 04, dort ist auch die Lösung in der Datei A4.java abzuspeichern. Eine passende `main`-Methode mit der Möglichkeit zur testweisen Evaluierung ist in einer Klasse Evaluation bereits vorgegeben.)

## 5. Aufgabe: Rekursion

(8 P)

Implementieren Sie die folgende Funktion  $g$  als rekursive Funktion mit dem Rückgabetyp **String** und einem Parameter **s** vom Typ **String**:

$$g(s) = \begin{cases} s, & \text{falls die Länge von } s \text{ eine gerade Zahl ist,} \\ g(s1) + "#" + g(s2), & \text{in allen anderen Fällen, wobei } s1 \text{ und } s2 \text{ sich ergeben} \\ & \text{aus } s1.length() = s2.length() = (s.length() - 1)/2 \\ & \text{und } s = s1+c+s2 \text{ mit einem einzelnen(!) Character } c. \\ & c \text{ ist also dabei der Character genau in der Mitte von } s, \text{ der durch } "#" \text{ ersetzt wird; der} \\ & \text{vordere und hintere Teil von } s \text{ werden rekursiv analog behandelt.} \end{cases}$$

**Beispiele:**

$$\begin{aligned} g("ab") &\rightsquigarrow "ab", & g("abc") &\rightsquigarrow "###", \\ g("abcde") &\rightsquigarrow "ab#de", & g("abcdefghijklm") &\rightsquigarrow "ab#de#gh#jk" \end{aligned}$$

Ihre Lösung darf dabei außer den Parametern und evtl. lokalen Variablen keine weiteren Variablen verwenden, auch Schleifen sind nicht erlaubt. (Ansonsten wird die Abgabe mit 0 Punkten bewertet, selbst wenn sie die richtigen Resultate liefert. Dies gilt analog für Lösungen, die nur die obigen Beispielresultate reproduzieren.)

Sie dürfen dabei davon ausgehen, dass **s** nicht den Wert **null** hat. Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 05** in der Datei **A5.java**, mit der Möglichkeit zur testweisen Evaluierung durch **Evaluation.java**. In der dortigen Datei **A5.java** ist die Lösung abzuspeichern.

## 6. Aufgabe: Arrays

(12 P)

Implementieren Sie eine Methode `int[] mirrorArray(int[] data)`, die ein (neues!) Array zurückgibt, in dem die Werte aus `data` in umgekehrter Reihenfolge enthalten sind.

Die Inhalte des Feldes `data` müssen dabei erhalten bleiben. Es ist möglich, dass `data` beim Aufruf die `null`-Referenz ist; in diesem Sonderfall soll auch die Null-Referenz zurückgegeben werden.

**Beispiel:**

Parameterfeld `data`: { 1, 2, 3, 4, 5 }  
`mirrorArray(data)`: { 5, 4, 3, 2, 1 }

Parameterfeld `data`: { 12345, 54321 }  
`mirrorArray(data)`: { 54321, 12345 }

Parameterfeld `data`: `null`  
`mirrorArray(data)`: `null`

(Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 06**, inklusive einer passenden `main`-Methode und der Möglichkeit zur testweisen Evaluierung. Dort ist auch die Lösung abzuspeichern.)

## 7. Aufgabe: Exception und lokale Klassen

(15 P)

Vorgegeben ist eine Klasse `Evaluation` mit zwei Methoden: Eine kurze `main`-Methode und eine Methode `void exceptionTrigger()`.

- `main` ruft eine Methode `exceptionCheck` auf, die Sie noch schreiben sollen.
- `void exceptionTrigger()` liest eine `int`-Zahl `n` von der Konsole ein. Sollte die eingegebene Zahl nicht zwischen 1 und 5 liegen, läuft `exceptionTrigger` ohne Probleme durch. Liegt `n` aber in diesem Bereich, wird in Abhängigkeit von `n` eine von fünf unterschiedlichen Exceptions entstehen.

Schreiben Sie nun in der Klasse `Checker` die Methode `int exceptionCheck()` und zudem eine lokale Exception-Klasse `NoProblemException` mit folgenden Eigenschaften:

- `exceptionCheck` soll `exceptionTrigger` aufrufen und durch geeignetes Fangen der jeweils geworfenen Exception die eingelesene Zahl bestimmen und an `main` zurückgeben.
- Welche Zahl bei `exceptionTrigger` zu welcher Exception führt, können Sie im Quelltext der Klasse `Evaluation` und der `Evaluation` unter Moodle leicht erkennen.
- Wenn `exceptionTrigger` also zum Beispiel eine 1 einliest, wird eine `ArithmeticException` geworfen. Dann muss `exceptionCheck` auch eine 1 zurückgeben.
- Ist die von `exceptionTrigger` gelesene Eingabe nicht zwischen 1 und 5, so läuft die Methode *ohne* Exception ab. In solchen Fällen soll nun von `exceptionCheck` selbst eine `NoProblemException` (als Subklasse von `RuntimeException`) geworfen werden.

An der Klasse `Evaluation` dürfen Sie keine Änderungen vornehmen; Ihre Lösung ist komplett in der Datei `Checker.java` abzuspeichern.

## 8. Aufgabe: Vererbung

(15 P)

Vorgegeben ist eine Klasse `Song`. Sie speichert über ein Musikstück Informationen wie den Titel (`String title`) und die Musikgruppe (`String band`). Folgende Methoden werden dabei bereitgestellt:

- Ein Konstruktor der Form

```
Song(String title, String band)
```

- Die getter-Methoden für einen `Song` (also `getTitle()` und `getBand()`)

Implementieren Sie passend dazu ein Interface `Playlist`. Der Zweck dieses Interfaces ist die Speicherung einer Sammlung von Musikstücken (`Song`). Daher soll das Interface die folgenden drei Methoden vorsehen:

- `void addSong(Song mySong)`
- `Song play(String title)`
- `int getSize()`

Leiten Sie schließlich die Klasse `SimplePlaylist` vom Interface ab. Die Klasse `SimplePlaylist` speichert Musikstücke in einem Array (`Song[] songs`). Dabei wird die Größe des Arrays im Konstruktor angegeben. Zudem speichert sie die Anzahl der über `addSong` hinzugefügten Musikstücke in einer geeigneten Variablen `size`.

Implementieren Sie `SimplePlaylist` wie folgt:

- Implementieren Sie einen Konstruktor `public SimplePlaylist(int maxsize)`, der als Parameter die Größe des Arrays hat.
- Implementieren Sie in `SimplePlaylist` die Methode `addSong`, die ihren Parameter-Song zur Playlist-Instanz hinzufügt. Sie können dabei eine beliebige Reihenfolge der Speicherung im Array `songs` wählen, ein Anfügen bei der ersten "freien" Stelle ist sicherlich am einfachsten. Wenn kein freier Platz mehr im Array existiert, geben Sie den folgenden String auf der Konsole aus und ignorieren ansonsten den Aufruf:

```
Playlist full
```

- `getSize()` kann als getter-Methode implementiert werden.
- Zum Schluss sollen Sie in `SimplePlaylist` die Methode `play` des Interfaces implementieren. Diese Methode soll im Array `songs` nach einem Stück mit diesem Titel suchen und dieses, wenn gefunden, zurückgeben. Wenn der gesuchte Titel nicht in `songs` enthalten ist, so geben Sie die `null`-Referenz zurück.

Eine rudimentäre Testklasse `Evaluation` ist vorgegeben, um Ihre Implementierung zu testen. Diese Klasse enthält einige Tests zur Überprüfung, ob die genannten Methoden und Funktionalitäten implementiert wurden.

**9. Aufgabe: Objekte**

(12 P)

Betrachten Sie die Klassen `Client` und `Parser`. Die Klasse `Client` speichert für einen Kunden eine ID, den Namen und eine Geldsumme, die vom Kunden für Waren ausgegeben wurde. Außerdem sind die Getter-Funktionen und eine `toString()`-Funktion bereits vorgegeben.

Erweitern Sie die Klasse `Parser` um eine Methode `createClient(String string)`, die in der Lage ist, Strings der folgenden Form zu verarbeiten und als Objekte zu speichern:

```
{cnr:1234;name:Nyso;value:200}  
{cnr:1234;value:200;name:Nyso}  
{cnr:1234      ;value      :200      ;name      : Nyso}
```

Berücksichtigen Sie, dass die Reihenfolge der Elemente des Triples unterschiedlich sein kann und ebenfalls unnötige Leerzeichen in den Strings vorkommen können.

**Weitere Fehler müssen Sie nicht betrachten, da nur die erwähnten Fehler vorkommen!**

## 10. Aufgabe: Stacks

(15 P)

Betrachten Sie die gegebenen Klassen `Stack` und `Elem`, die den in der Vorlesung vorgestellten Varianten sehr ähnlich sind. Sie enthalten einen Teil einer Stack-Implementierung aus verketteten Listenelementen, wobei jedoch der Typ `Integer` für die Datenkomponente fest voreingestellt ist. Eine `push`-Methode ist bereits vorgegeben, ebenso wie eine Methode `toString()`.

Erweitern Sie die Klasse `Stack` um die folgenden drei Methoden `pop()`, `invert()` und `removeZeroes()`:

1. Die Methode `pop()` ergänzt das Verhalten der Klasse um die (noch fehlende) Pop-Operation, so dass tatsächlich eine lauffähige Stack-Implementierung entsteht.

Bei einem leeren Stack soll `pop()` der Einfachheit halber den Zahlwert `-123456` zurückgeben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 1 (dh. Eingabe von 1) grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 15 10 5 ]
Resultat: 15 10 5 -123456
```

2. Die Methode `invert()` soll das Vorzeichen aller im Stack enthaltenen Einträge umdrehen, d.h. aus positiven Zahlen werden negative Zahlen und umgekehrt. Die Reihenfolge der Werte soll ansonsten unverändert bleiben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 2 grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ -4 -3 0 1 2 0 ]
Resultat: [ 4 3 0 -1 -2 0 ]
```

3. Die Methode `removeZeroes()` soll im Stack alle Einträge entfernen, die den Wert 0 haben. Alle anderen Werte sollen in der Original-Reihenfolge erhalten bleiben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 3 grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 0 0 1 0 2 0 ]
Resultat: [ 1 2 ]
```

Beim Testfall 4 werden wichtige Sonderfälle von `removeZeroes()` getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 0 0 0 ]
Resultat: [ ]
```

Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 10**, dort ist auch die Lösung abzuspeichern. Zu dieser Aufgabe finden Sie neben `Stack` und `Elem` auch eine Testklasse `Evaluation` mit einer `main`-Methode in Moodle. Die Klassen `Elem` und `Evaluation` dürfen nicht verändert werden.

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013

Bearbeitungszeit: 120 Minuten

Name	<hr/>
Matr.Nr	<hr/>
Punkte Gesamt	<hr/> von 40
Punkte Note	<hr/>

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) 4. Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) 5. Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

# Klausur Programmierung I

## Norbert Müller

### Wintersemester 2020/2021

#### Aufgabe 1:

man sollte beurteilen, ob die Syntax stimmt, wenn ja sollte man den Wert angeben der dabei rauskam, wenn nein, sollte man erklären, wo der Fehler liegt

- 1) boolean b = !(4<2);
- 2) int Int = 42;
- 3) Integer int = 42;
- 4) double d = (int) 42;

die letzten beiden weiß ich leider nicht mehr

#### Aufgabe 2:

Auswertungen von Funktionen

```
long eins = 1; int zwei = 2; float drei = 3.0; double vier = 4.0;  
eins / zwei + eins / vier * drei
```

man sollte Klammern setzen, um zu verdeutlichen, in welcher Reihenfolge das ausgewertet wird und auch angeben welchen Typ das Ergebnis hat wenn man sich die 4 Operatoren unabhängig betrachtet (also welchen Typ hat das Ergebnis, wenn man eins / zwei rechnet usw.) und man sollte den Wert angeben der dabei rauskommt

#### Aufgabe 3:

- das Verhalten einer vorgegebenen Methode beschreiben
- diese Methode umschreiben, sodass keine while-Schleife mehr vorhanden ist

#### Aufgabe 4:

Strings

man sollte eine Methode schreiben, die zwei Strings als Parameter hat und überprüft, ob der erste zweimal den zweiten String enthält, wenn ja soll das was dahinter steht zurückgegeben werden, wenn nein soll „null“ zurückgegeben werden

z.B. s1 = „xyzxyzABC“ s2 = „xyz“ dann sollte „ABC“ zurückgegeben werden  
s1 = „abab“ s2 = „ab“ dann sollte „“ zurückgegeben werden

#### Aufgabe 5:

Rekursion

es war eine Funktion gegeben

g(x) = „leer“, wenn der übergebene String leer ist  
= s (der String selbst), wenn die Länge des Strings ungerade ist  
= g(s1) + „#“ + g(s2), in allen anderen Fällen

s1 ist dabei die erste Hälfte von dem String und s2 die zweite Hälfte

z.B. s = „ab“ → „a#b“  
s = „abcd“ → „a#b#c#d“  
s = „abcdef“ → „abc#def“

#### Aufgabe 6:

Arrays

man sollte eine Methode schreiben, die zwei Arrays als Parameter hat und überprüft, ob mindestens zwei Zahlen aus dem ersten Array auch in dem zweiten Array vorkommen

wenn ja sollte „true“ zurückgegeben werden, wenn nein „false“

z.B. A = {1,2,3}; B = {4,5,6,7} → false  
A = {1,2,3}; B = {1,2,6} → true

$A = \{\}; B = \{1,2,3,4,5,6\} \rightarrow \text{false}$

### Aufgabe 7:

#### Collections

es war eine Klasse Meeting vorgegeben, date und visitor als Variablen enthält und man sollte eine Klasse „PersonalCalender“ schreiben, die alle Meetings speichert

### Aufgabe 8:

#### Vererbung

man hatte eine Klasse „Song“ vorgegeben und sollte ein Interface „Playlist“ schreiben und dann eine Klasse „SimplePlaylist“, die das Interface implementiert  
in „Playlist“ waren die Methoden: void addSong(Song song), String play(String titel) und noch eine Methode, an die ich mich leider nicht mehr erinner  
bei addSong sollte ein neuer Song in ein Array Song[] songs aufgenommen werden  
bei play sollte der Song mit dem passenden Titel gesucht werden und zurückgegeben werden, falls der Song nicht enthalten ist, sollte „null“ zurückgegeben werden

### Aufgabe 9:

#### Stack

man musste zwei Methoden schreiben, die erste sollte prüfen, ob ein Element in dem Stack enthalten ist  
die zweite Methode sollte ein Element daraus löschen (pop wurde nicht vorgegeben; man musste mit Referenzen arbeiten)

### Aufgabe 10:

#### StringTokenizer

man hatte eine Klasse A(mir fällt der genaue Name nicht mehr ein), wo der Konstruktor 7 Parameter hat  
dann hatte man eine andere Klasse, bei der im Konstruktor ein String übergeben wurde der alle 7 Parameter-Inforationen enthält, die immer durch „:“ getrennt wurden  
man sollte die trennen und dann in den Variablen speichern und damit ein neues Objekt von A erstellen  
falls der String falsch formatiert ist, sollte „falsches Format“ zurückgegeben werden

# Grundlagen der Programmierung

Norbert Müller

Universität Trier – Fachbereich IV – Informatik

17. Februar 2022

Bearbeitungszeit: **120 Minuten**

Name	_____
Matr.Nr.	_____
Rechner	_____

- Hiermit bestätige ich, dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur zur Vorlesung Grundlagen der Programmierung (Programmierung I) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.
- Ich fühle mich gesundheitlich in der Lage, die Klausur anzutreten.

Ort, Datum: \_\_\_\_\_

Unterschrift: \_\_\_\_\_

Beachten Sie folgende Regelungen:

- Als Hilfsmittel sind nur die zur Verfügung gestellten Editoren (inkl. Java-Dokumentation) sowie die elektronisch zur Verfügung gestellten Unterlagen (Kurs zur Klausur auf der moodle-Plattform) erlaubt. Andere Hilfsmittel wie z.B. das Einloggen in andere Moodle-Kurse oder die Verwendung von Smartphones sind nicht erlaubt.
- In der Klausur sind **112 Punkte** erreichbar. Ziel ist, dass Sie davon etwa **100 Punkte** bearbeiten. Die Zahl der Punkte pro Aufgabe entspricht also in etwa der Bearbeitungszeit (in Minuten).
- Alle Lösungen sind über das Moodle-System abzugeben, d.h. dort abzuspeichern.
- Programme können i.d.R. unter Moodle auch getestet und vor-evaluierter werden. Die Testfälle werden bei der Korrektur allerdings durch neue Fälle ersetzt werden! Abgaben, die nicht compilierbar sind, führen zu deutlichen Punktabzügen!

Aufgabe	1	2	3	4	5	6	7	8	9	10	Gesamt
Möglich	7	6	14	8	8	12	15	15	15	12	112
Erreicht											

Note: \_\_\_\_\_

**1. Aufgabe: Primitive Datentypen und Termauswertung**

(7 P)

Bei folgenden Deklarationen von Variablen

```
double eins = 1.0d; double zwei = 2.0d; int drei = 3; int vier = 4;
```

soll der folgende Ausdruck ausgewertet werden:

```
eins / zwei + drei / vier * zwei  
// (1)   (2)   (3)   (4)
```

Geben Sie zunächst an, wie dieser Ausdruck ausgewertet wird, indem Sie im Ausdruck Klammern (entsprechend den Auswertungsregeln in Java) hinzufügen.

Geben Sie zudem an, welchen Typ und welchen Wert die Zwischenresultate der Operationen (1) bis (4) (entsprechend den Auswertungsregeln in Java) an den jeweiligen Positionen haben. Verwenden Sie für die Angabe des Wertes immer Literale des jeweiligen Typs.

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

**2. Aufgabe: Deklarationen und Casts**

(6 P)

Betrachten Sie die im folgenden angegebenen kurzen Abschnitte aus (evtl. nicht kompilierbaren) Java-Quelltexten. Geben Sie bei den syntaktisch korrekten Textabschnitten den jeweils zugewiesenen Wert an. Bei syntaktisch nicht korrekten Texten geben Sie bitte an, warum sie nicht korrekt sind.

1. double real = 0,101;
2. boolean test = !(0 == 1);
3. int value = 010-1;
4. string text = "0101";
5. int data = 0x10-1;
6. int veryLarge = 1234567890;

Tragen Sie Ihre Antworten bei der entsprechenden Aufgabe in Moodle ein.

### 3. Aufgabe: Quelltext verstehen und umformulieren

Betrachten Sie die folgende Methode:

```
public static boolean f( int[] data ) {

    if ( data == null ) return false;

    int i = 0;
    int value = data[i];

    while ( i < data.length ) {
        if ( data[i] != value ) return false;
        i++;
    }
    return true;
}
```

- (a) Erklären Sie in Worten das Verhalten dieser Methode (d.h. bei welchen Parametern sie welchen Rückgabewert hat): Wann liefert sie Rückgabewert `true`, wann `false`? Beschreiben Sie dabei NICHT die Arbeit einzelner Zeilen des Quelltextes; zwei oder drei ordentlich formulerte Sätze sollten reichen! Achten Sie auf Sonderfälle! (8 P)
- (b) Implementieren Sie den Algorithmus aus `f` analog neu in einer Methode `g`: (6 P)
- Die neue Methode `g` muss für alle(!) möglichen Parameter die gleichen Resultate wie `f` liefern.
  - Dabei darf `g` jedoch *keine* `while`-Anweisung enthalten.

Den Quelltext gibt es in Moodle sowohl unter Aufgabe 03a als auch unter Aufgabe 03b. Ihre Lösung zu Aufgabenteil (a) soll als Freitext in Aufgabe 03a eingetragen werden, die Lösung zu Teil (b) soll unter A3b.java bei Aufgabe 03b in Moodle gespeichert werden; eine zusätzliche Klasse Evaluation.java zum Test einfacher Beispiele (aber nicht aller interessanten Fälle!) ist dort vorgesehen.

### 4. Aufgabe: Strings

(8 P)

Schreiben Sie eine Funktion `stringTest`, die zwei formale Parameter `String s1` und `String s2` hat und testet, ob `s1` die Verkettung `t + s2 + t` des Strings `s2` mit einem zweifach zu verwendenden anderen (von Ihnen zu findenden und zunächst unbekannten) String `t` ist. Wenn dies der Fall ist, soll `t` zurückgegeben werden, ansonsten die `null`-Referenz.

Beispiele:

```
stringTest("xyzABCxyz", "ABC") ~> "xyz"
stringTest("11222211", "2222") ~> "11"
stringTest("ABC", "ABC") ~> ""
stringTest("XaBab", "B") ~> null
```

Sie können davon ausgehen, dass weder `s1` noch `s2` den Wert `null` hat.

Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 04 in der Datei A4.java, mit der Möglichkeit zur testweisen Evaluierung durch Evaluation.java. In der dortigen Datei A4.java ist die Lösung abzuspeichern.

## 5. Aufgabe: Rekursion

Implementieren Sie die folgende Funktion  $g$  als rekursive Funktion mit dem Rückgabetyp `String`, einem Parameter `s` vom Typ `String` und zwei Parametern `m, n` vom Typ `int`:

$$g(s, m, n) = \begin{cases} \text{null}, & \text{falls } m \geq n \text{ oder } m < 0 \text{ ist oder } n > s.length() . \\ s.substring[m, n], & \text{falls ansonsten } m+1 = n \text{ ist,} \\ g(s, (m+n)/2, n) + g(s, m, (m+n)/2), & \text{in allen anderen Fällen} \end{cases}$$

Beispiele:  $g("abcde", 0, 5) = "edcba"$ ,  $g("abcde", 0, 3) = "cba"$ ,  
 $g("abcde", 1, 4) = "dcb"$ ,  $g("abcde", 2, 3) = "c"$

(Semantik von  $g$ : Es wird aus `s` der Teilstring von Position `m` bis Position `n-1` bestimmt, aber in umgekehrter Zeichenreihenfolge. Dies ist aber für die Lösung nicht relevant.)

Ihre Lösung darf dabei außer den Parametern und evtl. lokalen Variablen keine weiteren Variablen verwenden, insbesondere sind auch Schleifen nicht erlaubt. (Ansonsten wird die Abgabe mit 0 Punkten bewertet, selbst wenn sie die richtigen Resultate liefert. Dies gilt analog für Lösungen, die nur die obigen Beispielresultate reproduzieren.)

Sie dürfen dabei davon ausgehen, dass `s` nicht die `null`-Referenz ist. Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 05 in der Datei `A5.java`, mit der Möglichkeit zur testweisen Evaluierung durch `Evaluation.java`. In der dortigen Datei `A5.java` ist die Lösung abzuspeichern.

(12 P)

## 6. Aufgabe: Arrays

Implementieren Sie eine Methode `int[] mirrorArray(int[] data)`, die ein (neues!) Array zurückgibt, in dem die Werte aus `data` in umgekehrter Reihenfolge enthalten sind. Die Inhalte des Feldes `data` müssen dabei erhalten bleiben. Es ist möglich, dass `data` beim Aufruf die `null`-Referenz ist; in diesem Sonderfall soll auch die Null-Referenz zurückgegeben werden.

Beispiel:

Parameterfeld `data`: { 1, 2, 3, 4, 5 }  
`mirrorArray(data)`: { 5, 4, 3, 2, 1 }

Parameterfeld `data`: { 12345, 54321 }  
`mirrorArray(data)`: { 54321, 12345 }

Parameterfeld `data`: `null`  
`mirrorArray(data)`: `null`

(Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 06, inklusive einer passenden `main`-Methode und der Möglichkeit zur testweisen Evaluierung. Dort ist auch die Lösung abzuspeichern.)

## 7. Aufgabe: Collections

(15 P)

Vorgegeben ist eine Klasse `ExamResult`, bei der ein Objekt den Namen einer Person (`String name`) und die Note (`int mark`) speichern kann, die die Person bei einer Prüfung erreicht hat.

Ein passender Konstruktor und alle benötigten Getter-Methoden sind hier vorgegeben, so dass diese Klasse ebenso wie die testende Klasse `Evaluation` unverändert bleiben können (und müssen).

Sie sollen nun als Ergänzung dazu eine Klasse `ClassResults` implementieren. Realisieren Sie dafür die folgenden Teilaufgaben:

1. Jedes Objekt der Klasse `ClassResults` soll folgende Informationen speichern: Den Namen einer Prüfung (`String exam`) und zudem eine Liste an Einzelergebnissen (`ArrayList<ExamResult> results`), die die bei dieser Prüfung erreichten Resultate enthält.
2. Implementieren Sie den von `Evaluation` benutzten Konstruktor und ebenfalls die Getter-Methoden `getExam()` und `getResults()`.
3. Eine Methode `void registerMark(ExamResult result)` wird benötigt, um neue Prüfungsergebnisse zur Ergebnisliste eines `ClassResults`-Objektes hinzuzufügen.
4. Außerdem soll jede `ClassResults`-Instanz dazu in der Lage sein, zu einer gegebenen Note alle Personen zu finden, die diese Note erlangt haben. Sie sollen eine entsprechende Methode

```
ArrayList<String> findExaminees(int mark)
```

implementieren, die eine Liste mit den entsprechenden Personen zurückgibt.

5. Zugriff auf die Instanzvariablen der Klasse soll NUR mit dem Konstruktor und den obigen Methoden möglich sein. Diese Methoden und der Konstruktor sollen hingegen frei zugänglich sein.

Zum Testen Ihrer Implementierung ist die Klasse `Evaluation` gegeben. Diese überprüft aber bewußt nur rudimentär, ob die einzelnen Methoden vorhanden sind und bei den wenigen Testfällen die richtigen Rückgabewerte geliefert werden.

(15)

## 8. Aufgabe: Vererbung

Schreiben Sie eine konkrete Klasse `Cover` und ein Interface `Stream` wie im folgenden spezifiziert:

Vorgegeben ist eine abstrakte Klasse `Song`. Sie liefert über vier abstrakte Methoden Daten über ein Musikstück: den Titel (`String title()`), die Künstler (`(String artist())`) und die Spieldauer (`double length()`) des Stücks. Außerdem gibt es Zusatzinformationen (`String additionalInfo()`) zum Stück. Diese Klasse und die Testklasse `Evaluation` dürfen nicht verändert werden.

- Implementieren Sie nun eine konkrete Subklasse `Cover` von `Song`. Es geht hierbei um Musikstücke, die bereits früher einmal veröffentlicht wurden und jetzt neu gecovert worden sind.
- Zusätzlich zu den Daten über Titel, aktuelle Künstler und Spieldauer, die über `Song` bereits abfragbar sind, soll `Cover` daher folgende Informationen bereitstellen:
  - welche Künstler das Original des Songs eingespielt haben  
(zu speichern in `String originalArtist`) und
  - in welchem Jahr das geschehen ist  
(zu speichern in `int originalPublication`).
- Folgende Methoden müssen Sie dazu implementieren:
  - Ein Konstruktor der Form

```
Cover(String title, String artist, double length,  
      String originalArtist, int originalPublication)
```

- Zwei getter-Methoden für die zusätzlichen Objektvariablen  
(also `getOriginalArtist()` und `getOriginalPublication()`)
- Die vier konkretisierten Methoden für `Song`. Bei `additionalInfo()` soll die Form "originally by ... from year ..." mit den zusätzlichen Daten aus `originalArtist` und `originalPublication` gewählt werden; ansonsten sollen die entsprechenden Komponenten aus dem Konstruktor unverändert zurückgegeben werden.

Vergeben Sie dabei den Zugriffsmodifikator `private` und das Attribut `@Override` so oft wie möglich.

- Für eine Austrahlung im Radio wird als Kerninformation nur die Spieldauer benötigt. Daher implementiert `Song` ein (von Ihnen zu schreibendes) Interface `Stream`, das aber nur `length()` bereitstellen soll.

Eine rudimentäre Testklasse `Evaluation` ist vorgegeben, um Ihre Implementierung zu testen. Diese Klasse enthält einige Tests zur Überprüfung, ob die genannten Methoden und Funktionalitäten implementiert wurden.

## 9. Aufgabe: Exceptions und Strings

(15 P)

Implementieren Sie eine Methode

```
public static String checkComparison(String calculation)
```

die überprüft, ob der Parameter `calculation` eine korrekte (und korrekt formatierte) Ungleichung entsprechend folgender Form darstellt:

```
"1 < 2"      "54321 > 12345"      "-1111 < +1111"
```

Im Parameter dürfen dabei nur folgende Teile vorkommen:

- ganze Zahlen in der üblichen Dezimalform, evtl. mit Vorzeichen, so dass `int` als Zahltyp reicht und `Integer.parseInt()` verwendet werden kann (auch zum Test auf Korrektheit der Syntax),
- eines der Zeichen '`>`' und '`<`' (als Vergleichsoperatoren)

Diese drei Teile sind durch ein oder mehrere Leerzeichen voneinander getrennt, so dass problemlos ein `StringTokenizer` zur Zerlegung benutzt werden kann.

Ihre Lösung soll nun folgendes leisten:

- Sollte der String korrekt formatiert sein und zudem die Ungleichung stimmen, so geben Sie folgenden String zurück:

```
valid
```

- Sollte der String zwar korrektes Format haben, aber die Werte nicht übereinstimmen, geben Sie einen String im folgenden Format zurück, wobei `left` und `right` die gefundenen Werte sein sollen:

```
left is not smaller than right
```

bzw.

```
left is not greater than right
```

- Wenn der String falsch formatiert ist, sollen Sie eine `RuntimeException` werfen, die folgende Nachricht enthhält:

```
syntax error
```

Zur Vereinfachung: Auf Zahlierläufe brauchen Sie nicht zu achten. Der Parameter ist nicht `null`. Eventuelle intern auftretende Exceptions, z.B. aus `Integer.parseInt()`, müssen Sie aber geeignet abfangen.

Einige Beispiele für falsche Gleichungen sind:

```
"-1 > +1"      "54321 < 12345"      "1111 < 1111"
```

Einige Beispiele für falsch formatierte Strings sind:

```
"5<10"          "10 = 5"           "5 < 1 0"  
"+ 10 > 9"       "10.0 < 11"        "-5 > - 4"
```

## 10. Aufgabe: Listen

Betrachten Sie die gegebenen Klassen `Liste` und `Elem`, die den in der Vorlesung vorgestellten Varianten sehr ähnlich sind. Sie implementieren eine Minimalfassung einer verketteten Liste von `String`-Werten. Die bereits vorgegebene Methode `addFront(String s)` fügt den `String s` am Anfang der Liste ein.

Erweitern Sie die Klasse `Liste` um die folgenden zwei Methoden:

- Die Methode `ausgeben()` soll einen String mit den Werten sämtlicher Elemente in der Reihenfolge ausgeben, in welcher sie ausgehend von `start` in der Liste enthalten sind. Sie sollen dabei durch einzelne '+'-Zeichen getrennt werden. Insbesondere soll nach dem letzten Element kein '+' stehen.

Der erste Aufruf von `ausgeben()` in der `Main.main()`-Methode soll also die folgende Ausgabe produzieren::

was+wie+wer+das+die+der

- Die Methode `int ersetze(String s, String t)` soll in der Liste den String `s` überall durch den String `t` ersetzen und zudem als Rückgabewert angeben, wie oft `s` ersetzt wurde.

Die weiteren Aufrufe von `ausgeben()` in der `Main.main()`-Methode sollen daher die folgende Ausgabe produzieren:

was+wie+wer+das+die+wer  
was+wie+wie+das+die+wie  
was+wie+wie+das+die+wie

Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 10, dort ist auch die Lösung abzuspeichern. Zu dieser Aufgabe finden Sie neben `Liste` und `Elem` auch eine Testklasse `Main` mit einer `main`-Methode in Moodle. Die Klassen `Elem` und `Main` dürfen nicht verändert werden. Sie dürfen zur Vereinfachung davon ausgehen, dass keine `null`-Strings zu verarbeiten sind.

## Gedankenprotokoll Nachklausur Programmierung 1 am 10.04.2018

### Aufgabe 1 + 2

Datentypen (Ergebnis, Datentyp d. Ergebnis, Erklärung)

1.  $1 / 2.0 * 3$
  2. "12"+10
  3. int 1.999
  4. double x = (int) 1.999
  5. double int = 2.0
- ...

### Aufgabe 3

a) Code schriftlich erklären.

```
static int funktion (int [] a) {  
    int summe = 0;  
    for (int x : a) {  
        summe = summe + x;  
    }  
    return summe;  
}
```

b) gleichen Code nochmal implementieren, aber nicht in einer for-Schleife und ohne auf den Originalcode zuzugreifen

### Aufgabe 4

Methode schreiben, um zu zählen wie oft String t ein Teilstring von String s ist  
z.B. String s = „aabaaba“ und String t = „ab“ dann soll der Wert 2 zurückgegeben werden

### Aufgabe 5

Methode schreiben: deleteMinMax

In einem Feld sollen die Minima und Maxima gefunden werden und anschließend durch Nullen ersetzt werden, z.B. Feld der Größe 5: 1 2 4 1 3 sieht danach so aus: 0 2 0 0 3

### Aufgabe 6

Rekursion mit Strings

$$g(s) = \begin{cases} 0, & \text{falls String } s \text{ leer ist} \\ 2*g(t), & \text{falls } s \text{ aus 'a' + t besteht} \\ 3+g(t), & \text{falls } s \text{ auf 'b' + t + 'c' besteht} \\ s.length(), & \text{in jedem anderen Fall} \end{cases}$$

'a' , 'b' , 'c' sollen ganz normale Buchstaben darstellen

### Aufgabe 7

?

### *Aufgabe 8*

Abstrakte Klasse Prüfung, davon abgeleitete Klassen Klausur und Mündlich schreiben

Prüfung besitzt fach (String) und eine id (int), wobei die id ab 1 hochzählen soll

Klausur besitzt geschrieben (boolean)

Mündlich besitzt zudem ein Datum

### *Aufgabe 9*

Listen: vorgegebenes Programm wie folgt ergänzen

Methode 1 ergänze (String s, String t): String t in der Liste hinter String s angehangen werden

Methode 2 ... ?

anschließende Ausgabe der richtigen Reihenfolge der Liste

### *Aufgabe 10*

File IO mit Exception: 3 gegebene txt.-Dateien mit jeweils normalen Zahlen je Spalte (int) und

Kommentare (gekennzeichnet durch #)

Es soll berechnet werden die Summe der Zahlen in den jeweiligen Dateien. Eine Exception soll geworfen werden, wenn Kommentare in der Datei vorhanden sind (mit Lokalisation)

### *Aufgabe 11*

Collections: Dominospiel realisieren

# Programmierung I Klausur WiSe 2018/2019

## **Aufgabe 1 (schriftlich):**

Man bekommt Rechnungen mit unterschiedlichen Zahltypen gegeben und muss das Ergebnis, den Datentyp des Ergebnisses und eine kurze Begründung dafür abgeben, in dieser Aufgabe mit eher untypischen Typen, z.B. Hexadezimal, Long, Float etc.

## **Aufgabe 2 (schriftlich):**

Man bekommt wie in Aufgabe 1 eine Rechnung/Zuweisung, bestehend aus 2-4 Zahlen verschiedener Typen und muss das Ergebnis angeben. Falls die Rechnung nicht funktionieren würde (also zu einem Error führen würde), muss man eine kurze Begründung angeben, warum das Programm hier abstürzt. Ein paar Beispiele für die Rechnungen:

- 1) int a = 1.999;
- 2) int double = 3.0;
- 3) "12"+10
- 4) int b = 1 / 3.0 \* 2;

## **Aufgabe 3:**

Teil a) Schriftlich, Code auf Papier gegeben, man muss kurz erklären, was der Code macht. Der Code in der Klausur war eine Methode mit einem int[] **feld** und einem int **parameter**. Die Methode hatte eine int **summe** gegeben. Mit dem **parameter** wurde ein switch aufgerufen, falls **parameter == 0**, wurde mit einer for-each-Schleife der jeweilige Feldeintrag zur **summe** addiert, falls **parameter == 1**, dann wurde mit einer for-each-Schleife zuerst der Feldeintrag quadriert und dann zur **summe** addiert. Am Ende der Methode wurde **summe** zurückgegeben. Code:

```
public int methode(int[] feld, int parameter) {  
    int summe = 0;  
    switch (parameter) {  
        case 0: {  
            for(int i : feld) {  
                summe += i;  
            }  
        }  
        case 1: {  
            for(int i : feld) {  
                summe += i*i;  
            }  
        }  
    }  
    return summe;  
}
```

Teil b) Den gleichen Code implementieren, aber ohne for-Schleifen

## **Aufgabe 4:**

Man soll eine Methode schreiben, die einen String **str** einliest, falls dieser String zweimal genau aus dem gleichen String **t** besteht, also **str = t + t**, dann soll String **t** zurückgegeben werden, ansonsten **null**. (Beispiel: "TestTest" soll "Test" zurückliefern, "Auto" soll **null** zurückliefern) Lösung:

```
public static boolean stringTest(String str) {  
    int haelfte = str.length() / 2;  
    String teil1 = str.substring(0, haelfte);  
    String teil2 = str.substring(haelfte, str.length());  
    return teil1.equals(teil2);  
}
```

## **Aufgabe 5:**

Feld mit Integers wird in eine Methode übergeben, falls ein Feldeintrag **genau** zweimal im Feld vorkommt, soll *TRUE* zurückgegeben werden, sonst *FALSE*. Lösung:

```
public static boolean feldTest(int[] feld) {  
    for(int i=0; i < feld.length; i++) {  
        int eintrag = feld[i];  
        int counter = 0;  
        for(int j=0; j < feld.length; j++) {  
            if(feld[j] == feld[i]) counter++;  
        }  
        if(counter == 2) return true;  
    }  
    return false;  
}
```

## **Aufgabe 6:**

Man muss zwei Klassen Professor und Student schreiben. Professor hat einen Namen, Fachbereich und eine int-Variable **zeit** (diese werden im Konstruktor gesetzt, wird von der Evaluation automatisch getan). Jeder Student bekommt im Konstruktor einen Namen, Fachbereich und einen Professor **prof** übergeben. Falls **prof** den gleichen Fachbereich besitzt, wie der gerade konstruierte Student, dann wird dieser Student in eine ArrayList des Typs Student in der Professor-Klasse eingefügt. Man soll dann eine Methode **beratung** schreiben, die die ArrayList durchläuft und einen String der Form „Student xyz wird beraten“ ausgibt, falls der Professor noch **zeit** hat, ansonsten wird der String „Professor xy hat keine Beratungszeit mehr übrig“ ausgegeben.

## **Aufgabe 7:**

Man bekommt Strings der Form “1234;Harry Potter; This is a book“ übergeben, die erste Zahl soll die ISBN-Nummer darstellen, der Mittelteil ist der Name des Buchs und der letzte Teilstring ist ein Kommentar. Für jede Eingabe sollte die Ausgabe wie folgt aussehen (am Beispiel):

isbn:	1234
Name:	Harry Potter
Abstrct:	This is a book

Falls ein Teilstring leer ist oder zu wenige Teilstrings vorhanden sind, soll eine Exception geworfen und gecatched werden, die dann “Invalid input“ ausgibt.

## **Aufgabe 8:**

Man bekommt eine abstrakte Klasse Kamera gegeben, davon abgeleitet sind die Klassen Smartphone und Spiegelreflexkamera. Jede der Klassen hat eine int **bilder**, die von 1 hochzählen soll wenn ein Bild gemacht wird (dafür gibt es Methode **takePicture()**). Zudem hat jede neue Instanz (von Smartphone oder Spiegelreflexkamera) eine individuelle ID.

## **Aufgabe 9:**

Man bekommt eine verkettete Liste **L**, ein Element **x** und eine int **n** gegeben und soll eine Methode schreiben, die **x** an der **n**-ten Stelle der Liste einfügt (darauf achten, dass **n** nicht größer ist als die Liste)

Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage nach Ergebnis, Datentyp,  
Erklärung in eine Tabelle auf Klausurblatt schreiben

Bsp: int m = 1; double n = (int) 1.0;

Aufgabe 2 (6P)

6 Aufgaben: Frage zu Programmschnipseln nach Ergebnis oder wenn es nicht läuft den Grund

**Aufgabe 3**

- a) Programm auf Blatt Papier schriftlich erklären – (6P)
- b) Programm ohne for und ohne switch programmieren (8P)

```
Public static int xx (long feld[], int/long? parameter){
```

```
    Int value = 0;
```

```
    Switch(parameter)
```

```
        Case 0: for (long k : feld) value += v; return value;
```

```
        Case 1: for(long k : feld) value += v*v; return value;
```

```
}
```

**Aufgabe 4 (8P)**

Methode schreiben die einen String übergeben bekommt;  
dieser String s soll untersucht werden, ob er aus zwei gleichen Teilstrings t besteht;  
wenn ja soll der Teilstring t zurückgegeben werden – sonst null;

Bspw: String s = abab hat t = ab;

**Aufgabe 5 (8P)**

Rekursion: mit if und else if und else die Anweisungen auf dem Blatt entsprechend implementieren

## Aufgabe 6 (12P) - Arrays

Es wurde ein Array einer Methode übergeben;

Man sollte schauen, ob irgendeine Zahl im Array genau 2x vorkommt,  
wenn ja diese ausgeben,  
wenn nicht false;

## Aufgabe 7 (15P) – Collections

Es gab: eine Klasse Professoren und eine Klasse Studenten und eine main-Klasse

Klasse Professor:

- String domain (Fachbereich)
  - int officeHours
  - String name
- LinkedList<studenten> list;

Klasse studenten:

- String domain (Fachbereich)
- int time (benötigte Zeit für Beratung)
- String name

Man sollte nun zwei Methoden schreiben:

1. Methode bekommt ein student übergeben  
ist dieser vom selben Fachbereich (domain) wie der Prof;  
so soll er in die Liste vom Prof eingetragen werden  
sonst sollte eine Ausgabe auf Konsole erfolgen (Prof x ist nicht zuständig für Student y)
2. Methode soll die Liste vom Prof abarbeiten  
es sollen die studentnamen ausgegeben werden  
jeder student benötigt seine „time“  
diese wird von „officeHours „ bei jeder Ausgabe vom studenten um studenten.getTime()  
verringert  
Ist officeHours auf 0 so sollen alle restlichen Studenten in der Liste ausgegeben werden;  
mit den Zusatz (in etwa) „ Prof x konnte Student y nicht bearbeiten“  
(alle Ausgaben war nicht auf deutsch sondern auf englisch)

## Aufgabe 8 (15P) Exceptiones

Es wurde ein String an eine Methode übergeben, die geschrieben werden sollte.

Dieser String beinhaltet das Schema „ISBN; titelvonBuch; abstract“

bspw: „123;rotkappe; abstract“

Es konnte passieren, dass der String null ist, nicht alle Angaben enthalten sind;

Oder „;“ 1x fehlt

Der String sollte auf drei String aufgeteilt werden;

Die ISBN sollte in einen int Variable gespeichert werden;

Falls isbn zu lang war oder keine Zahl war sollte eine Exception fliegen und den Wert auf -1 von isbn setzen;

Ansonsten sollte eine Ausgabe wie folgt erfolgen; wichtig war hier die Bündigkeit(tabellenartig)

ISBN: 125

Titel: TitelBuch

Abstract: Nicht abstract!

## Aufgabe 9 (15P) Vererbung

- Abstracte class camera schreiben:
  - o String model
  - o Int id (soll bei 0 starten ; inklusive )
  - o Int pictures( Anzahl der getätigten Bilder, soll bei 0 starten, inklusive)
  - o Boolean hasObjektiv (ob man Objektiv abschrauben kann)
  - o Abstracte methode void takePicture();
  - o toString methode
- von camera abgeleitet
  - o class Smartphone
    - sollte im Konstruktor name übergeben bekommen; und im Konstruktor (super(name, false) wegen kein Objektiv abnehmbar)
  - o class SRL-Kamera
    - sollte im Konstruktor (model, String Objektivname) erhalten;
    - im Konstruktor true wegen Objektiv
  - o Methode takePicture in Smartphone und SRL-Kamera implementieren; wenn aufgerufen soll sich die Zahl picture erhöhen und ausgeben werden plus mit welcher Kamera ( id und model) und wenn vorhanden Objektiv

### Aufgabe 10 (15P)

Abgewandelte Version von Queue mit Elem; 2 Methoden schreiben;  
es gab keine getter/setter-Methoden bei Elem; es gab nur String value und Elem next;  
methode enqueue, toString(?) waren vorgegeben;

1. Methode: Insert ( int index, String word) – soll an stelle index word einfügen
2. Methode: bekommt einen String übergeben; der aus mehreren Teilen besteht;  
diese Teile sind mittels Leerzeichen voneinander getrennt;  
diese Teile sollen nun in die bestehe Queue hinten angehängt werden;  
Jedes Teil ein neues Elem;

# Programmierung 1 Wintersemester 2017/2018

Bei: Prof. Dr. Norbert Müller

Insgesamt gab es 110 Punkte, wovon 100 erreicht werden mussten.

**Aufgabe 1) (7 Punkte)**

Grundrechenoperationen von verschiedenen Dateitypen. Ergebnis, Dateityp vom Ergebnis, Erklärung.

**Aufgabe 2) (7 Punkte)**

Zuweisungen. Zum Beispiel: int a = 1,2? Was passiert dann? Double d = (int) 1.2;

**Aufgabe 3) (6 + 8 Punkte)**

- a) Algorithmus erklären (auf Papier)
- b) Den selben Algorithmus programmieren aber ohne Switch / For

**Aufgabe 4) (8 Punkte)**

Teilstring testen (Aufgabe von SS17):

„Man bekommt einen String, der aus 3 Teilstrings besteht und so aufgebaut ist:  
String1String2String1. Die Strings sind alle gleich lang. Das Programm soll prüfen ob der erste  
Teilstring gleich dem dritten Teilstring ist, und wenn das der Fall ist, den mittleren  
zurückgeben. Als Beispiel: Der String lautet "AutoBuchAuto". Es soll "Buch" zurück gegeben  
werden.“

**Aufgabe 5) (8 Punkte)**

Integer Array mit 5 Zahlen. Alle positiven Zahlen sollten die Position tauschen.

Also: [1, -1, 2, 4, -3, 5] => [5, -1, 4, 2, 1]

**Aufgabe 6) () Rekursion**

Es gab eine Methode G(int k) die ein Integer zurückgibt, je nach Wert von k sollte ein  
bestimmter Wert zurückgegeben werden, der teilweise noch einmal G aufruft.

**Aufgabe 7) ()**

**Aufgabe 8) (12 Punkte)**

Abstrakte Klassen und Vererbung

**Aufgabe 9) (12 Punkte)**

Verkettete Listen mit Elem (siehe VL).

**Aufgabe 10) (12 Punkte)**

File I/O

# Programmierung I – 1. Klausur WS 21/22 – bei Herr Prof. Dr. Müller

100 von 112 Punkte für 1.0 erreichen

## Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage was System.out.println(...) ausgeben wird

Bsp.: System.out.println("10" + 20) Ausgabe: 1020

## Aufgabe 2 (6P)

6 Aufgaben: Initialisierung von Datentypen, Frage nach korrekter Syntax, wenn ja, Wert angeben, wenn nein, den Grund angeben

Bsp.: string test = "abc"; (Fehler, String muss großgeschrieben werden)  
Double d = 6,5; (Fehler, double Zahlen werden mit einem Punkt initialisiert)

## Aufgabe 3 (14P)

- Programm auf Blatt Papier schriftlich erklären – (6P)
- Programm ohne while-Schleife schreiben, z.B. mit einer for-Schleife (8P)

## Aufgabe 4 (8P) – Stringverarbeitung

Eine Methode bekommt String s1 und String s2 übergeben. Zu überprüfen war, ob der String s2 in s1 enthalten ist und dann der Teilstring t vor s2 und hinter s2 wieder stand, also quasi die Form hat:

s1 = (t + s2 + t) - Bsp.: s1 = abcxyzabc , s2 = xyz

Wenn das gestimmt hat, sollte t übergeben werden, ansonsten null und wenn t = "" war, dann sollte "" zurückgegeben werden

## Aufgabe 5 (8P) – Rekursion

Rekursion: einfach nur die verschiedenen Fälle abschreiben und wieder rekursiv aufrufen lassen

## Aufgabe 6 (12P) – Arrays

Es wurde ein Array einer Methode übergeben und man sollte wieder ein Array zurückgeben, allerdings sollten dann alle Werte umgedreht sein.

Falls das Array eine Null-Referenz ist, sollte auch null zurückgegeben werden

## Aufgabe 7 (15P) – Collections

Man hatte eine ArrayList <ExamResults> (ExamResults waren wiederum Objekte, bestehend aus Person, Note und Fach) und sollte dafür 2 Methoden schreiben:

1. Eine Methode die neue ExamResults einfügt
2. Eine Methode, welche eine „**int mark**“ übergegeben bekommt, die alle Objekte der ArrayList durchläuft (Iterator bietet sich da an) und dann alle Personen in eine neue ArrayList einträgt, welche die Note „**mark**“ haben.  
Anschließend sollte die ArrayList<String> mit den Personennamen zurückgegeben werden

## Aufgabe 8 (15P) – Exceptions

Man bekam einen String der Form : „Zahl“ gefolgt von „> oder <“ und dann wieder eine „Zahl“. Getrennt waren die Strings nur mit Leerzeichen, sodass sich ein StringTokenizer anbietet, in Kombination mit der trim() Funktion.

Man sollte überprüfen, ob der String diesem Format entspricht, wenn nein, sollte eine RuntimeException geworfen werden, wenn ja, dann sollte überprüft werden, ob die Werte der beiden Zahlen auch mit dem „> oder <“ Zeichen übereinstimmen. Wenn das der Fall war, dann sollte „valid“ zurückgegeben werden, ansonsten zwei Texte, wie „links ist nicht größer als rechts“ oder analog „links ist nicht kleiner als rechts“.

## Aufgabe 9 (15P) – Vererbung

Man hatte 4 Klassen und musste ein Interface schreiben, wovon eine abstrakte Klasse „Song“ geerbt hat und dann sollte man eine Klasse „Cover“ schreiben, welche von Song erbt.

Es wurde ein Konstruktor verlangt, sowie Initialisierung von den abstrakten Methoden, sowie ein @Override einer Stringmethode.

## Aufgabe 10 (12P) – Liste

Abgespeckte Version von der Vorlesung, man hatte nur die Klasse Elem und List (mit String Objekten) und musste folgende Methoden schreiben:

1. Eine Methode, die einmal durch die Liste läuft und alle Stringwerte zu einem String zusammenfasst, sodass man jeden Stringwert durch ein „+“ getrennt hat. Ein Beispiel für die Form ist: was+wie+wer+wo+wann  
Verlang war nur, dass hinter dem letzten String kein Plus mehr gesetzt wird
2. Eine Methode size(), welche beim Durchlaufen einmal zählen sollte, wie viele Objekte in der Liste drinnen sind

**Exam was in total 115 ; 15 is bonus.**

**(7 pts) Q1** It was about Data types.

<u>Example</u>	<u>Type</u>	<u>Explanation</u>
31+42		
42.0/10+ 5/4		
3.2d+4.3f		
!(5>6)		
True		

**(6 pts) Q2** It was about again data types and assignment of these data types and if the compiler works or not

e.g.

<u>Example</u>	<u>If the compiler works or not, if not the reason</u>
double r=(int)4;	

**Q3)**

a) Read the given code and explain what it is doing

```
public static boolean f(int[] data){  
    if(data==null) return false;  
    int i=0;  
    while(i<data.length){  
        if(data[i-1]>data[i]) return false;  
        i++;  
    }  
    return true;
```

b) Implement the above code without using while or do while.

**Q4)** You must write a method which return "abc" if "abcabcXY", "XY" (it is an example) like  
string t + t + s return t

```
public static String method(String first, String s){  
}
```

If String first in the form t+t+s and the methods should return t else null

e.g

"xyzxyzABC" must return "xyz"

"xyxyABC" must return "xy"

“11111111” must return “111”

“xyzxyyABC” must return null;

**Q5)** Writing method which return true if a char repeated three or more in an array ( ex return true for array={1 1 1 2 3} )

```
public boolean charRepeated(String[] data){  
}
```

Examples were like

{“1”, “2”, “3”, “4”, “5”, “5” } must return false

{“1”, “1”, “1”, “2”, “3”, “5” } must return true

**Q6)** Arrays

**Q7)** String and StringTokenizer

You are given Strings in the form of (.....;.....;.....;.....)

(String ;String ; String ;String)  
↓

This part must be converted from String to int  
In the question it says that use Integer.parseInt()

Eg. (1234;Marcus Halle; Bla Bla Publishing; Something More)

You need to printout;

ISBN : ..... (integer)

Author: ..... (String)

Publisher: .....(String)

Some other thing: ..... (String)

**(15 pts)Q8**) Interface

**(15 pts)Q9**) Collections

**(15 pts)Q10**) Queue

You have implement a method into Queue.java that add new element to the desired position

```
Public void addElem(Elem new, int pos){  
}
```

# **Programmierung I – 1. Klausur – Gedankenprotokoll**

## **– WS 19/20 – 18.02.2020 –**

Klausur bei Herrn Prof. Dr. Müller und Herrn Jun.-Prof. Dr. Weyers für Prog I und Prog Ia & Ib.  
**Insgesamt gab es 110 Punkte. 100 Punkte für 1,0.**

Bearbeitung der Aufgaben mit Eclipse oder IntelliJ ist sehr empfehlenswert.  
**110 Minuten Zeit.**

### **Aufgabe 1 (auf dem Blatt zu beantworten):**

Rechenoperationen mit verschiedenen Zahlen und Typen in tabellarischer Form.

Gefragt waren das **Ergebnis**, der **Typ** und eine **Begründung** des Zustandekommens des Ergebnisses – unter anderem mit „exotischen“ Typen wie Long, Float, Hexadezimal...

**Beispiele:** 42L – 0xa; 42.f + 13.d; „42“+32; 42./10+18/10; ...

### **Aufgabe 2 (auf dem Blatt zu beantworten):**

Ausdrücke mit verschiedenen Datentypen in tabellarischer Form.

Gefragt war, ob der dargestellte **Ausdruck** sich **kompilieren** lässt oder nicht.

Wenn ja, dann war das **Ergebnis** des Ausdrucks gefragt,  
wenn nicht, dann der **Grund**, warum es nicht kompilieren sollte.

**Beispiele:** boolean b = (4 < 2); Int int = 42; int Int = 42.0;  
int [] a = {43,22} → int [2]; char c = „d“

### **Aufgabe 3:**

Quelltextverständnis.

#### **a) (auf dem Blatt zu beantworten):**

Es war ein Quellcode gegeben mit der Methode „f“, die einen boolean-Wert **true** liefert, wenn das übergebene Array absteigend sortiert ist, **false** liefert, wenn es nicht absteigend sortiert ist, und wenn die Null-Referenz übergeben wird, wurde dies per **if-Anweisung** geprüft und eine **Exception** geworfen. Der Durchlauf des Arrays geschah in einer **while-Schleife**.

#### **b) (unter moodle abzugeben)**

Programmierung des Algorithmus als Methode „g“ ohne **while** oder **do-while-Schleife**.  
Somit war hier eine **for-Schleife** gefragt.

### **Aufgabe 4 (unter moodle abzugeben):**

Strings.

Es sollte eine Methode geschrieben werden, die überprüft,  
ob der String **s1** aus dem **Schema t + s2 + t** besteht und anschließend soll **t** zurückgegeben  
werden.

#### **Beispiele:**

**s1** = „AutoBuchAuto“, **s2** = „Buch“ → Rückgabe von **t** = „Auto“  
oder **s1** = „xxxyxxxxBeispielxxxxxxx“, **s2** = „Beispiel“  
→ Rückgabe von **t** = „xxxyxxxx“  
oder **s1** = „aaWsApfelwAss“, **s2** = „Apfel“ → Rückgabe einer **Null-Referenz**.

# Programmierung I – 1. Klausur – Gedankenprotokoll

## **– WS 19/20 – 18.02.2020 –**

Somit wurden in diese Methode die Strings **s1** und **s2** übergeben. Man musste zuerst herausfinden, was dieser **String t** ist und wenn **s1** aus **diesem Schema (also: t + s2 + t)** besteht, sollte **t** zurückgegeben werden. Andernfalls eine **Null-Referenz**.

### **Aufgabe 5 (unter moodle abzugeben):**

Rekursion.

Die zu programmierende Methode **g(String s, int n)** war mit den genauen Anweisungen auf dem Blatt angegeben.

Hier als vereinfachte Darstellung der zu programmierenden Abfragen der Rekursion:

$$g(\text{String } s, \text{int } n) = \begin{cases} 0, & \text{falls } n < 0 \\ g(s, s.\text{length}() - 1), & \text{falls } n < s.\text{length}() \\ g(s, s.\text{length}()), & \text{falls } s.\text{charAt}(n) == 'X' \\ g(s, s.\text{length}() + 1), & \text{falls } s.\text{charAt}(n) == 'Y' \\ 0, & \text{sonst} \end{cases}$$

### **Aufgabe 6 (unter moodle abzugeben):**

Arrays.

Es sollte eine Methode geschrieben werden, die **true** zurückgibt, falls das Array aus **mind. 3 gleichen Einträgen** besteht, andernfalls **false**.

Hier bietet sich eine 2-fach verschachtelte **for-Schleife** an, die mit einem Counter die Häufigkeit eines Eintrages überprüft und bei einem Counter von 3 **true** returnt.

Es bietet sich auch eine 3-fach verschachtelte **for-Schleife** ohne Counter an.

### **Aufgabe 7 (unter moodle abzugeben):**

Collections.

Es sollte eine **Klasse „Webshop“** geschrieben werden, die die Eigenschaften **String name** und **List <Produkte> products = new ArrayList<Produkte>();** enthält.

Die **Klasse „Produkte“** war schon vorgegeben mit **Namen, Preis** und den jeweiligen Gettern und Settern war vorgegeben.

Somit sollte ein **Konstruktor** `public Webshop (String name) {....}` geschrieben werden, der den übergebenen Namen in die Instanz hinzufügt. Also: `name = this.name;`  
Zudem waren folgende 2 Methoden zu schreiben:

**Product getMostExpensiveProduct (String s) {...}**

**Product getCheapestProduct (String s) {...}**

Hierbei sollte überprüft werden, ob der übergebene **Artikelname s** in der List **products** vorhanden ist, und anschließend sollte das **teuerste Produkt** und das **günstigste Produkt** unter dem **Artikelnamen s** zurückgegeben werden.

# Programmierung I – 1. Klausur – Gedankenprotokoll

**– WS 19/20 – 18.02.2020 –**

## **Aufgabe 8 (unter moodle abzugeben):**

Vererbung.

Es sollte zuerst ein **Interface „Publication“** geschrieben wurde, die die **abstrakten** Methoden `getDescription` und `getVenueDescription` enthält.

Als Nächstes sollte eine **Klasse „Article“** geschrieben werden, die das **Interface “Publication”** implementiert, der die Variablen `String title, String author, String venue, int year` und eine **eindeutig** und einmalig bezeichnete `int ID` (also static) zugeschrieben wird und letztlich ein passender **Konstruktor** geschrieben werden sollte.

Es sollten danach die jeweiligen **Getter** und **Setter**, die vorgegeben in der Main-Methode implementiert waren, als Methoden angelegt werden.

Als zusätzliche **Methoden** sollten `getDescription` und `getVenueDescription` die jeweiligen Inhalte des Artikels als **String** returnen.

Danach sollte eine **Klasse „Journal“** geschrieben werden, die die **Klasse „Article“** extendet, der die Variablen `int numberOfArticles` zugeschrieben werden und im **Konstruktor** zusätzlich zu den o. g. Eigenschaften auch `numberOfArticles` übergeben werden und letztlich per `super(...)` von „**Article**“ die gleichen Eigenschaften gesetzt werden sollten.

Zum Schluss sollte ein das folgende Array angelegt werden:

```
Article [] articles = new Article [numberOfArticles];
```

und die Methode `getDescription` sollte per `@Override` überschrieben werden und sollte zusätzlich zu dem o. g. Text auch Folgendes ausgeben:

```
return (...) + " containing " + numberOfArticles + " articles";
```

## **Aufgabe 9 (unter moodle abzugeben):**

Listen.

Es war vereinfacht eine **Klasse „Elem“** nur mit `next` und `value` - ohne `.getNext()` oder `.setNext()` - gegeben und eine **Klasse „Queue“** gegeben, in welcher folgende beiden Methoden implementiert werden sollten:

**1. Methode:** `public boolean isElem (String s) {...}`

**2. Methode:** `public void swap (String s1, String s2) {...}`

Die **1. Methode** sollte überprüfen, ob der **String s** in der Queue vorhanden ist, wenn ja sollte **true** zurückgegeben werden, wenn nein, sollte **false** zurückgegeben werden.

Die **2. Methode** sollte den 2 Referenzen vertauschen:

**Beispiel: „This for is fun lol“**

sollte mit `swap(“for”, “is”)` zu **“This is for fun lol”** werden.

# **Programmierung I – 1. Klausur – Gedankenprotokoll**

**– WS 19/20 – 18.02.2020 –**

## **Aufgabe 10 (unter moodle abzugeben):**

Objekte.

Es sollte eine **Klasse „Parser“** erstellt werden, die in einer Methode eine Instanz der **Klasse „Baby“**, die vorgegeben war, mit den Variablen **String name, String birthdate und int weight** erstellt.

Hierbei sollte der „**Parser**“ in dieser Methode einen String, wie z.B.

{name:Clara; weight: 1023; birthdate: 02-18-2020 },  
die Informationen „Clara“, 2023 und „02-18-2020“  
durch einen StringTokenizer token = new StringTokenizer („;:{ }“) oder  
der .split(“[ { }; : ]“)-Methode über den Tag „name“, „weight“ und „birthdate“  
filtern.

Wichtig war hier die Benutzung der Methode .trim(), die die Leerzeichen entfernt.

Anschließend sollte ein **Baby** konstruiert werden und **returnt** werden mit den gefilterten Variablen. Also: Baby baby = new Baby (name, weight, birthdate);

1. Schreiben Sie eine Java-Applikation, welche die folgende Problemstellungen löst und auf dem Bildschirm ausgibt. Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten Teilaufgabe fortfahren.

1. Lesen Sie vom Benutzer zwei beliebige Strings  $S_1$  und  $S_2$  ein.
  2. Bestimmen Sie die Länge  $n$  von  $S_1$  und  $m$  von  $S_2$ .
  3. Legen Sie eine Matrix *Needleman* mit  $(n + 1)$  Zeilen und  $(m + 1)$  Spalten an.
  4. Schreiben Sie eine Methode  $\text{int cost(char } a, \text{char } b\text{)}$ , die  $-1$  zurückgibt, wenn  $a \neq b$  ist und  $1$  zurückgibt wenn  $a = b$  ist.
  5. Schreiben Sie eine Methode  $\text{int max(int } a, \text{int } b, \text{int } c\text{)}$ , die das Maximum dreier Zahlen bestimmt.
  6. Schreiben Sie eine Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$ , die eine Matrix auf dem Bildschirm ausgibt.
  7. Initialisieren Sie
    - (a) die Matrixposition  $\text{Needleman}(0, 0)$  mit  $0$ .
    - (b)  $\text{Needleman}(i, 0)$  mit  $-i$  für  $i = 1 \dots n$  mit  $i$ .
    - (c)  $\text{Needleman}(0, j)$  mit  $-j$  für  $j = 1 \dots m$  mit  $j$ .
  8. Berechnen Sie alle anderen Matrixpositionen  $\text{Needleman}(i, j)$  wie folgt
- $$\text{Needleman}(i, j) = \max \begin{cases} \text{Needleman}(i - 1, j) - 1 \\ \text{Needleman}(i, j - 1) - 1 \\ \text{Needleman}(i - 1, j - 1) + \text{cost}(S_1[i - 1], S_2[j - 1]) \end{cases}$$
9. Geben Sie den Wert  $\text{Needleman}(n, m)$  aus.
  10. Geben Sie die Matrix *Needleman* aus.  
(für eine 4,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  11. Erweitern Sie die Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$  derart, dass nun auch das  $i$ -te Zeichen von  $S_1$  vor der  $i$ -ten Zeile und das  $j$ -te Zeichen von  $S_2$  vor der  $j$ -ten Spalte steht.  
(für eine 3,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  12. Bestimmen Sie einen möglichen maximalen Pfad von  $\text{Needleman}(0, 0)$  nach  $\text{Needleman}(n, m)$ . Der Pfad soll dabei jeweils der Maximumauswahl des 8. Schrittes folgen. Orientieren Sie sich dabei an folgenden Tipps.
    - (a) Fangen Sie am Ende der Matrix an und bestimmen Sie das Maximum der davon links, darüber und diagonal links darüber liegenden Zelle.
    - (b) Das Maximum liegt auf dem Pfad und kann nun als neuer Ausgangspunkt verwendet werden.
    - (c) Suchen Sie solange neue Maxima bis Sie  $\text{Needleman}(0, 0)$  als Maximum identifiziert haben.
    - (d) Speichern Sie den Pfad in einer Matrix der gleichen Größe wie *Needleman*. Verwenden Sie String als Datentyp der Matrix. Alle Positionen sind von Anfang an auf ein Leerzeichen initialisiert. Die jeweils gefunden Maxima werden in dieser Matrix an der entsprechenden Position gespeichert.

(für eine 1,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)

0	-1	-2	-3	-4
-1	1	0	-1	-2
-2	0	0	-1	0
-3	-1	1	0	-1
-4	-2	0	2	1
-5	-3	-1	1	3

Ausgabe 10.)

	A	G	T	C
0	-1	-2	-3	-4
A	-1	1	0	-1
C	-2	0	0	-1
G	-3	-1	1	0
T	-4	-2	0	2
C	-5	-3	-1	1

Ausgabe 11.)

	A	G	T	C
0				
A				1
C				0
G				1
T				2
C				3

Ausgabe 12.)

**Aufgabe 1:** Primitive Datentypen und Termauswertung  
Siehe Anhang

**Aufgabe 2:** Deklarationen und Casts  
Siehe Anhang

**Aufgabe 3a:** Quelltext verstehen  
Siehe Anhang

**Aufgabe 3b:** Den Quelltext aus Aufgabe 3a so umformulieren, das er genau die gleichen Ergebnisse liefert, das Programm jedoch keine while-Schleife enthält.

**Aufgabe 4:** Strings

man sollte eine Methode schreiben, welche 2 Strings als Input hat.

Beispiel: s1= abcDEFabc, s2 = DEF. Man sollte bei dieser Eingabe abc zurückliefern, heißt man muss überprüfen ob der vordere String und der hintere String gleich sind. Ist dies der Fall, so sollte man diesen zurückgeben. s1 = abab, s2 = „ → Ausgabe: ab,

weiteres Beispiel: s1: abaaba, s2 = aa → Ausgabe null.

**Aufgabe 5:** ein Array wird eingelesen, die Aufgabe ist es eine KOPIE dieses Arrays und dessen Elemente in der Reihenfolge vertauscht. Beispiel: [1,2,3,4,5] → [5,4,3,2,1]. Dabei darf das eingelesene Array nach dem Aufruf der Methode sich nicht verändert haben.

**Aufgabe 6:** Rekursion

die Aufgabe fällt mir nicht mehr ein, da die Beschriebene Rekursion sehr kompliziert war (ich weiß nichtmals was diese Rekursion gemacht hat). Die Aufgabe ließ sich jedoch innerhalb einer Minute bewältigen, dazu ist es unwichtig was die Rekursion macht, man muss diese nur stur implementieren!

**Aufgabe 7:** Collectitons, **Aufgabe 8:** Vererbung

→ Diese Aufgaben waren komplex und Materialbezogen, diese kann ich hier leider nicht mehr sinnvoll wiedergeben.

**Aufgabe 9:** Exceptions

Seltsamerweise wurde diese Aufgabe nach 2 Tagen auf Moodle wieder freigeschaltet (die anderen leider nicht). Daher habe ich die exakten Dateien, welche auch in der Klausur gegeben waren (siehe Anhang)

Aufgabe war es: Evaluation.java darf nicht geändert werden

Man muss eine Methode String checkComparison(String calculation){} schreiben. Diese Methode soll einen String auswerten. Richtige Kombinationen für einen String sind:

„0 < 1“  
„1 > 0“  
„-1 < 0“  
„+1 > 11111“  
„234 < 435“  
„-233 < -345“

falsche Kombinationen sind:

„0< 1“  
„+0 < 1“ //2 Leerzeichen

„0 = 0“

Die Aufgabe ist nun:

-bei einer Eingabe mit falscher Kombination soll eine RuntimeException geworfen werden, welche „syntax error“ ausgibt.

-ist die Eingabe eine richtige Kombination, jedoch macht die Eingabe logischerweise keine Sinn (also „+1 > 1111“), so ist die Rückgabe entweder

x is not greater than y

oder

x is not smaller than y

-ist die Eingabe eine richtige Kombination und macht auch logischerweise Sinn (also „+1 < 0“), so soll die Rückgabe „valid“ sein

Im Ordner „Lösungen“ befindet sich meine Lösung zu dieser Aufgabe, die Lösung hat volle Punktzahl erreicht.

### Aufgabe 10: Listen

Auch diese Aufgabe ist zu Komplex, als das ich sie hier verständlich ausdrücken könnte.

#### Letzte Worte meinerseits:

-lese dir die Aufgaben zuerst komplett durch, bei manchen Aufgaben ist dies für das Verständnis notwendig!

-2h Bearbeitungszeit, 112 zu erreichende Punkte

-mache dich vertraut damit wie du neue Projekte auf Eclipse/ IntelliJ anlegst, öffnest du IntelliJ auf dem Uni Rechner hat dieser kein Projekt und du musst selber eins anlegen (ich denke nicht, dass das ein Problem sein sollte, aber lieber zu viel gesagt als zu wenig)

Miniklausur 1 zur Vorlesung

# Grundlagen der Programmierung

## Aufgabenblatt

- Sie dürfen NICHT in Gruppen arbeiten. Mit der Teilnahme versichern Sie, dass Sie die Aufgaben alleine bearbeiten. Bearbeitungszeit ist jeweils von **09:00** bis **10:00**. Sie benötigen am Semesterende 50% der Gesamtpunktzahl aller 6 Miniklausuren.
- Die Angaben zur Evaluation bedeuten:
  - Bei “*Evaluation: Zahlen*” wird nur auf Übereinstimmung der Zahlen mit der Vorgabe getestet, sonstige Formatierungen und ausgegebener Text werden ignoriert. Sie brauchen also nicht auf Zeilenenden o.Ä. zu achten.
  - Bei “*Evaluation: Zahlen/Text*” müssen alle ausgegebenen Zeichen mit der Vorgabe übereinstimmen. Die exakte Formatierung (Zeilenumbrüche, etc.) ist jedoch nicht relevant.
- Es wird bei allen Aufgaben verlangt, Eingaben über den Scanner vorzunehmen. Wenn Sie das Einlesen anders implementieren, werden die Evaluierungen NICHT funktionieren und Sie erhalten KEINE Punkte, selbst wenn Ihre Lösung ansonsten in Ordnung sein sollte.

### Aufgabe 1 (Evaluation: Zahlen) (10 Punkte)

Die Aufgabe besteht darin, über den Scanner 4 Zahlen  $a$ ,  $b$ ,  $c$  und  $d$  mit Hilfe von `sc.nextInt()` einzulesen und dann den Wert  $(a - b) * c + d$  auszugeben, etwa in der Form

Eingabe: 2 3 4 5

Ausgabe: 1

### Aufgabe 2 (Evaluation: Zahlen) (10 Punkte)

Lesen Sie zwei Zahlen  $a$ ,  $b$  mit Hilfe von `sc.nextInt()` über den Scanner ein und bestimmen Sie die Summe  $a + (a+1) + (a+2) + \dots + b$  aller Zahlen, die zwischen diesen beiden Zahlen liegen (jeweils inklusive).

Sie dürfen davon ausgehen, dass  $a \leq b$  gilt. (Der Fall  $a = b$  ist also erlaubt!)

Bei der Eingabe 4 6 müssen Sie also die Summe  $4 + 5 + 6 = 15$  bestimmen, analog bei Eingabe 1 10 die Summe  $1 + 2 + 3 + \dots + 10 = 55$ .

Eingabe: 4 6

Ausgabe: 15

### Aufgabe 3 (Evaluation: Zahlen)

(10 Punkte)

Lesen Sie zwei Zahlen  $a, b$  mit Hilfe von `sc.nextInt()` über den Scanner ein und geben Sie dann alle Zahlen aus, die zwischen diesen beiden Zahlen liegen (jeweils inklusive). Bei der Ausgabe müssen Sie jedoch die Reihenfolge umdrehen, d.h. Sie müssen die Zahlenfolge  $b \ b-1 \dots a+1 \ a$  als Ausgabe erzeugen.

Sie dürfen davon ausgehen, dass  $a \leq b$  gilt.

Bei der Eingabe 4 7 müssen Sie also die Ausgabe 7 6 5 4 erzeugen.

Eingabe: 4 7

Ausgabe: 7 6 5 4

### Aufgabe 4 (Evaluation: Zahlen)

(10 Punkte)

Im Beispiel K3B01\_GCD der Vorlesung werden zwei Zahlen über popup-Fenster eingelesen und dann deren größter gemeinsamer Teiler in einem weiteren Fenster ausgegeben. Sie finden das Programm bei Moodle im Abschnitt **Folien/Skript/Programmcodes** in der zip-Datei **Progammbeispiele zur Vorlesung**.

Kopieren Sie den Kern dieses Programms in das zur Aufgabe vorgegebene Lösungsprogramm und modifizieren Sie es dort so, dass keine popup-Fenster benutzt werden, sondern die zwei Zahlen mit Hilfe von `sc.nextInt()` über den Scanner eingelesen werden und das Resultat auf der Konsole ausgegeben wird, z.B. in der Form

Eingabe: 15 9

Ausgabe: 3

### Aufgabe 5 (Evaluation: Zahlen/Text)

(10 Punkte)

Geben Sie eine Eingabeaufforderung der Form “Bitte zwei Zahlen eingeben.” aus und lesen Sie dann zwei Zahlen  $a, b$  mit Hilfe von `sc.nextInt()` über den Scanner ein.

Dabei gibt es zwei Fälle:  $a \leq b$  oder  $a > b$ . Erzeugen Sie entsprechend einen der folgenden zwei Texte als Ausgabe:

“Die erste Zahl ist kleiner oder gleich der zweiten Zahl.”

bzw.

“Die erste Zahl ist grösser als die zweite Zahl.”

Ein Beispiel für den Programmablauf inkl.(!) Eingaben wäre also:

Bitte zwei Zahlen eingeben.

3

8

Die erste Zahl ist kleiner oder gleich der zweiten Zahl.

Achtung: Bei dieser Aufgabe wird auch der **komplette Ausgabetext** evaluiert.

## Grundlagen der Programmierung

### Aufgabenblatt

- Sie dürfen NICHT in Gruppen arbeiten. Mit der Teilnahme versichern Sie, dass Sie die Aufgaben alleine bearbeiten. Bearbeitungszeit ist jeweils von **09:00** bis **10:00**. Sie benötigen am Semesterende 50% der Gesamtpunktzahl aller 6 Miniklausuren.
- Die Angaben zur Evaluation bedeuten:
  - Bei “*Evaluation: Zahlen*” wird nur auf Übereinstimmung der Zahlen mit der Vorgabe getestet, sonstige Formatierungen und ausgegebener Text werden ignoriert. Sie brauchen also nicht auf Zeilenenden o.Ä. zu achten.
  - Bei “*Evaluation: Zahlen/Text*” müssen alle ausgegebenen Zeichen mit der Vorgabe übereinstimmen. Die exakte Formatierung (Zeilenumbrüche, etc.) ist jedoch nicht relevant.
  - Die von der Evaluierungsroutine ausgewerteten Teile des Ein/Ausgabeprotokolls werden in **blau** dargestellt, die Eingaben in **rot**.
- Es wird bei allen Aufgaben verlangt, Eingaben über den Scanner vorzunehmen. Wenn Sie das Einlesen anders implementieren, werden die Evaluierungen NICHT funktionieren und Sie erhalten KEINE Punkte, selbst wenn Ihre Lösung ansonsten in Ordnung sein sollte.

#### Aufgabe 1 (Evaluation: Zahlen/Text)

(10 Punkte)

Schreiben Sie ein Programm, das zwei `int`-Zahlen  $n$  und  $m$  einliest.

- Fall  $n < m$ : Das Programm soll die Summe der Zahlen zwischen  $n$  und  $m$  (**exklusive  $n$  und inklusive  $m$** ) mit Hilfe einer **for-Schleife** bestimmen. Bei  $n = -2$  und  $m = 2$  müssen Sie also die Summe  $(-1) + 0 + 1 + 2 = 2$  berechnen und ausgeben.

**Eingabe:** **-2 2**

**Ausgabe:** **2**

- Für den Fall  $n \geq m$  soll keine Berechnung vorgenommen werden, sondern nur eine Fehlermeldung wie folgt ausgegeben werden (d.h. Ihr Ausgabetext wird entsprechend evaluiert):

**Eingabe:** **100 10**

**falsche Eingabe**

Hinweis: Nutzen Sie eine zusätzliche Variable für die Summe, die mit 0 initialisiert wird.

#### Aufgabe 2 (Evaluation: Zahlen)

(10 Punkte)

Im vorgegebenen Anfangsteil des Programmes wird zunächst eine ganze Zahl `anzahl` (mit Wert  $> 0$ ) eingelesen und dann ein Array `int[] data` der Größe `anzahl` erstellt. Dieses Array wird dann mit zufällig bestimmten Zahlen gefüllt, wozu ein Zufallszahlengenerator mit einem zweiten eingelesenen Wert initialisiert wird. Die Zahlen im Array können dabei sowohl positiv als auch negativ sein. Bis hierhin ist das Programm vorgegeben und darf nicht verändert werden.

Sie sollen nun herausfinden, (a) wie viele der Zahlen im Array  $< 0$  sind und (b) wie viele  $> 0$  sind. Als Beispiel ergeben sich bei Eingabe von **3 3** als Array-Inhalt die drei Zahlen **4 -1 6** und damit

Anzahl: 3  
Zufallsstart: 3  
(a) Anzahl negativer Werte: 1  
(b) Anzahl positiver Werte: 2

### Aufgabe 3 (ohne Evaluationsmöglichkeit)

(10 Punkte)

Schreiben Sie ein Programm, mit dem Sie (z.B. durch Suche mit einer `while`-Schleifen) die folgenden zwei Zahlen bestimmen können:

- die kleinste positive int-Zahl  $n (> 0)$ , für die in Java der boolesche Ausdruck  $(n*n*n)/(n*n) == n$  den Wert `false` ergibt, sowie
- die kleinste positive long-Zahl  $m (> 0)$ , für die in Java der Ausdruck  $(m*m*m*m)/m == m*m*m$  den Wert `false` ergibt.

Eine Eingabe wird hier nicht benötigt, Sie brauchen also z.B. keinen Scanner. Die Ausgabe soll im Wesentlichen die folgende Form haben:

int: 1234  
long: 12345

Achtung: Die hier angegebenen Werte **1234** und **123456** sind nicht die korrekten Resultate, geben aber grob deren Größenordnung wieder. *Während der Miniklausur* ist auch kein korrekter Wert für die Evaluation vorgegeben, d.h. Sie können Ihr Programm nicht durch die Evaluation auf Korrektheit testen. Die korrekten Werte werden erst für die Korrektur in Moodle eingestellt.

### Aufgabe 4 (Evaluation: Zahlen)

(10 Punkte)

Programmieren Sie den folgenden Algorithmus:

Deklarieren Sie zwei Variablen: a vom Typ int und x vom Type double.		
Lesen Sie in a eine int-Zahl vom Scanner ein. (Sie können annehmen, dass a positiv ist.)		
Setzen Sie x auf den Wert 2.		
Führen Sie genau zehnmal den folgenden Schleifenrumpf aus: <table border="1"><tr><td>Geben Sie x aus.</td></tr><tr><td>Bestimmen Sie einen neuen Wert für x über die Formel <math>x = (x + a/x) / 2</math></td></tr></table>	Geben Sie x aus.	Bestimmen Sie einen neuen Wert für x über die Formel $x = (x + a/x) / 2$
Geben Sie x aus.		
Bestimmen Sie einen neuen Wert für x über die Formel $x = (x + a/x) / 2$		

Es werden also genau 10 double-Zahlen ausgegeben. Beispiel:

Eingabe: 16  
Ausgabe: 2.0 5.0 4.1 4.001219512195122 4.0000001858445895  
4.00000000000004 4.0 4.0 4.0 4.0

Anmerkung: Dies ist das sogenannte “Heron”-Verfahren zur Berechnung der Quadratwurzel.

**Aufgabe 5 (Evaluation: Zahlen/Text)**

(10 Punkte)

Schreiben Sie ein Programm, das drei `int`-Zahlen  $k$ ,  $m$  und  $n$  einliest (in dieser Reihenfolge) und eine Liste der Zahlen zwischen  $m$  und  $n$  (jeweils inklusive) ausgibt. Allerdings soll dabei jede Zahl, die ohne Rest durch  $k$  teilbar ist, durch den Text `beep` ersetzt werden.

Für den Fall  $m \leq n$  soll die Liste aufsteigend sortiert sein, wie im folgenden Beispiel:

Eingabe: 3 4 10

Ausgabe: 4 5 beep 7 8 beep 10

Für den Fall  $m > n$  soll die Liste absteigend sortiert sein, wie im folgenden Beispiel:

Eingabe: 2 11 3

Ausgabe: 11 beep 9 beep 7 beep 5 beep 3

Tipp: Setzen Sie dazu den %-Operator geeignet ein. Sie dürfen davon ausgehen, dass der Wert  $k$  positiv ist. Trennen Sie die einzelnen Ausgaben zum Beispiel durch ein oder mehrere Leerzeichen oder Zeilenenden voneinander.

Miniklausur 3 zur Vorlesung

Grundlagen der Programmierung

Aufgabenblatt

- Sie dürfen NICHT in Gruppen arbeiten. Mit der Teilnahme versichern Sie, dass Sie die Aufgaben alleine bearbeiten. Bearbeitungszeit ist jeweils von **09:00** bis **10:00**. Sie benötigen am Semesterende 50% der Gesamtpunktzahl aller 6 Miniklausuren.
- Die Angaben zur Evaluation bedeuten:
  - Bei “*Evaluation: Exakter Text*” müssen alle ausgegebenen Zeichen inklusive ihrer Formatisierungen mit der Vorgabe übereinstimmen.
  - Bei “*Evaluation: vorgegebene main*” gibt es eine vorgegebene `main`-Methode, die die von Ihnen zu schreibenden Methoden aufruft. Diese Methode darf NICHT verändert werden.
  - Die von der Evaluierungsroutine ausgewerteten Teile des Ein/Ausgabeprotokolls werden in **blau** dargestellt, die Eingaben in **rot**.
- Es wird bei allen Aufgaben verlangt, Eingaben über den Scanner vorzunehmen. Wenn Sie das Einlesen anders implementieren, werden die Evaluierungen NICHT funktionieren und Sie erhalten KEINE Punkte, selbst wenn Ihre Lösung ansonsten in Ordnung sein sollte.

**Aufgabe 1 (Evaluation: Exakter Text)**

(10 Punkte)

Schreiben Sie ein Java-Programm, das als Eingabe eine `int`-Zahl  $n$  einliest und als Ausgabe (ähnlich dem Beispiel K3B14\_Flagge und der entsprechenden Übungsaufgabe) Muster (aus  $n \times n$  Zeichen) folgender Art generiert:

<b>Eingabe:</b> 6	<b>Eingabe:</b> 7
XXXXXX	XXXXXX
XXXXXo	XXXXXo
XXXXoo	XXXXoo
XXXooo	XXXooo
XXoooo	XXoooo
Xooooo	Xooooo

Achten Sie auch darauf, dass keine Zeilenenden fehlen und dass Sie keine unnötigen Leerzeilen erzeugen, ansonsten funktioniert die Evaluation nicht. Die im Muster verwendeten Buchstaben sind das große X und das kleine o.

**Aufgabe 2 (Evaluation: vorgegebene main)**

(10 Punkte)

Schreiben Sie eine statische Java-Methode `meineFunktion` mit vier Parametern, die die folgende Formel berechnet:

$$\text{meineFunktion}(a, b, c, d) = \begin{cases} a + b, & \text{falls } c < d \\ a - b, & \text{falls } c \geq d \end{cases}$$

<b>Eingabe:</b> 1 2 3 4	<b>Eingabe:</b> 4 3 2 1
<b>Ausgabe:</b> 3	<b>Ausgabe:</b> 1

Eine passende `main`-Methode ist vorgegeben und darf nicht verändert werden. Welche Signatur `meineFunktion` haben sollte (passend zur `main`-Methode), müssen Sie selbst herausfinden.

### Aufgabe 3 (Evaluation: vorgegebene main)

(10 Punkte)

Mit der vorgegebenen `main`-Methode wird ein globales(!) Array `data` initialisiert. Schreiben Sie dazu eine statische Methode `meinMaximum(a,b)`, die den größten Wert von `data` für Indizes im Bereich von `a` bis `b` (jeweils inklusive) bestimmt und zurückgibt. Sie können dabei davon ausgehen, dass  $a \leq b$  gilt und diese beiden Werte auch im für Indizes erlaubten Bereich liegen.

Die vorgegebene `main`-Methode enthält einige Aufrufe der Methode `meinMaximum(a,b)`.

Anzahl: 5 Werte: 1 5 2 4 3 Tests: 0 1 5 2 3 4 4 4 3 0 4 5	Anzahl: 6 Werte: 5 4 3 2 1 0 Tests: 1 2 4 2 3 3 3 5 2 5 5 0
---	---

### Aufgabe 4 (Evaluation: vorgegebene main)

(10 Punkte)

Schreiben Sie eine statische Java-Methode `meineAusgabe` des Typs `void`, die ihren `int`-Parameter in Dezimalschreibweise ausgibt (mit `System.out.print(...)`), wobei aber die **Reihenfolge der Stellen umgedreht** ist (erst Einer, dann Zehner, dann Hunderter etc.). Sie können davon ausgehen, dass der Parameter  $> 0$  ist. Eine `main`-Methode ist wieder vorgegeben:

Eingabe: 13579 Ausgabe: 97531	Eingabe: 1000 Ausgabe: 0001
----------------------------------	--------------------------------

Tipp: Sie können z.B. die Operatoren "%" und "/" in einer Schleife geeignet einsetzen, aber es gibt auch andere Wege, um Zahlen in Strings umzuwandeln.

### Aufgabe 5 (Evaluation: vorgegebene main)

(10 Punkte)

Das vorgegebene Programm liest wie üblich an Anfang Daten in ein Array ein und gibt sie am Ende schließlich wieder aus. Vertauschen Sie nun die Werte im Array an der gekennzeichneten Stelle in der Mitte des Programmes so, dass die Reihenfolge der Werte bei der abschließenden Ausgabe umgedreht ist. Das vorgegebene Programm darf ansonsten nicht verändert werden. Beispiel:

Anzahl: 5 Werte: 1 2 3 4 5 umgedrehte Reihenfolge: 5 4 3 2 1
--

Achtung: Ziel ist nicht, dass Sie selbst die Werte in umgekehrter Reihenfolge ausgeben, sondern dass Sie die Werte im Array entsprechend vertauschen! Die Ausgabe geschieht bereits über das vorgegebene Programm!

## Aufgabe 1

Datentypen

## Aufgabe 2

Casting von Datentypen

## Aufgabe 3

- a) Quellcode lesen

```
Public static int f(int[] myArray, int parameter){
```

```
    Int value=0;
```

```
    Switch(parameter)
```

```
        Case 1
```

```
            For(int x:myArray){
```

```
                Value+=x*x;
```

```
}
```

```
        Case 0
```

```
            For(int x:myArray){
```

```
                Value+=x;
```

```
}
```

```
        Return value;
```

```
}
```

- b) Code aus 3 ohne switch und for Schleife analog implementieren

## Aufgabe 4

Arrays

Schreibe static boolean g(int[] myArray), so dass true herausgegeben wird, wenn mind. ein Eintrag von myArray genau 2 mal vorkommt, sonst false.

## Aufgabe 5

Rekursion mit String

Schreibe Methode g(String s) so, dass

- $3 \cdot g(t)$ , wenn  $s = "a" + t$
- $5 \cdot g(t)$ , wenn  $s = "b" + t + "c"$
- $s.length()$ , sonst [s könnte auch „ sein]

## Aufgabe 6

Finde heraus ob ein String s eine Aneinanderkettung zweier identischer Strings t ist ( $s.equals(t+t)$ ). Falls ja, gib t an, ansonsten gib die Nullreferenz zurück.

### **Aufgabe 7**

Collections, Klassen. Klassen Student und Professor. Professor hat int officeHours, String domain, String namen. Student hat String domain, namen und int time. Es geht darum, dass eine Sprechstunde beim Prof abgebildet werden soll. Dazu werden der Reihe nach in der Main Studenten erzeugt, die eine gewisse Zeit time für ein Gespräch mit dem Prof brauchen. Dies wird aber nur gemacht, wenn die Domains von Prof und Student übereinstimmen, falls nicht, wird ein Text der Art: „Professor „+name+“ ist nicht zuständig für“ student.name.

Ist die Domain identisch, dann reduziert sich die verfügbare Zeit officeHours des Professors um die Zeit, die der entsprechende Student benötigt. Solange der Professor noch Zeit hat, wird der nächste Student zur Sprechstunde zugelassen und es wird ausgegeben „Professor „+name+“ hilft gerade “ +student.name.

Hat der Prof keine Zeit mehr Ausgabe „Leider keine Zeit mehr“

### **Aufgabe 8**

Exceptions, Klassen

Int isbn, String title, String abstract in der Form „123;asdas;opio“ übergeben. Exceptions, wenn Part1 kein Int ist, oder einer der durch „;“ getrennten tokens leer, oder null übergeben wird,...

### **Aufgabe 9**

Vererbung Kamera

Abstract Klasse Kamera, Vererbung auf Klassen Smartphone, und Spiegelreflex

### **Aufgabe 10**

Klasse List analog Vorlesung.

# Aufgaben Programmierung I

## Wintersemester 2007/2008

Schreiben Sie eine Java-Applikation, die die folgende Problemstellung löst und auf dem Bildschirm ausgibt. Es soll eine Klassenstruktur für eine Mediathek entwickelt werden, die es erlaubt Medien wie DVDs, CDs etc. zu erfassen. **Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten beginnen.**

Aufgabe 41: Medium

Schreiben Sie eine abstrakte Klasse `Medium`, die für jedes Objekt eine eindeutige Identifikationsnummer und einen Titel speichert. Die Identifikationsnummern sollen dabei iterativ generiert werden (d.h. das erste erzeugte Objekt soll die Nummer 1 bekommen, das zweite die 2, usw.). Darüber hinaus sollen Methoden `boolean istVerliehen()` und `setVerliehen(boolean)` bereitgestellt werden.

Aufgabe 42: CD/DVD

Leiten Sie von der Klasse `Medium` eine Klasse `DVD` ab, die zusätzlich den Regisseur und die Spielzeit der DVD speichert. Schreiben Sie die entsprechenden `get`-Methoden. Das Setzen der Klassenvariablen sollte hingegen nur im Konstruktor möglich sein. Analog dazu leiten Sie ebenfalls von `Medium` eine Klasse `CD` ab, die den Künstler und die Spielzeit speichert.

Aufgabe 43: Mediathek I

Eine Mediathek repräsentiert in unserem Falle eine Menge beliebiger Größe von verschiedenen Medien. Dazu werden die Medien in einer doppelt verketteten Liste (unsortiert) verwaltet.

- Schreiben Sie deshalb eine Klasse `MedListElem`, deren Objekte ein Listenelement mit einem Wert vom Typ `Medium` darstellen.
- Schreiben Sie eine Klasse `Mediathek`, die eine Methode zum einfachen Einfügen am Ende `void add(Medium m)`, eine zum Löschen eines Mediums aus der Liste `void delete(Medium m)` und eine zum Auffinden des entsprechenden Listenelements `MedListElem lookUp(Medium m)` bereitstellt. Die `lookUp`-Methode soll dabei außerhalb der Klasse nicht sichtbar sein. Des Weiteren sollten Sie eine Methode `String toString()` in der Klasse `Medienliste` schreiben, die die Ausgabe der Liste als String erlaubt. Die Verwendung einer Implementierung vom Typ `Collection`, d.h. von `ArrayList` u.ä., ist **nicht** zulässig.

#### Aufgabe 44:

#### Mediathek II

Erweitern sie ihre Klasse **Mediathek**, so dass der folgende Sachverhalt modelliert wird:

- Eine Mediathek hat einen Namen, der über den Konstruktor festgelegt wird. Das Auslesen soll mittels `getName()` erfolgen.
- Erweitern Sie die Methode `toString()`, so dass zusätzlich der Name der Mediathek ausgegeben wird.
- Schreiben Sie eine Methode `Mediathek find(String str)` zum Suchen nach Medien anhand des Titels. Der Methode soll ein `String str` übergeben werden und in einer Liste alle Medien zurückliefern, die diesen String `str` im Titel enthalten. Der Rückgabewert soll den Namen der Mediathek in der gesucht wird zusammen mit der gesuchten Zeichenkette `str` tragen.

#### Aufgabe 45:

#### MediaTest

Schreiben Sie eine Klasse **MediaTest** zum Testen der bereits entwickelten Klassen **Medium**, **DVD**, **CD**, **Mediathek**. Legen Sie dazu eine Mediathek an und ordnen Sie dieser eine beliebige Anzahl an Medien zu, sowohl CDs als auch DVDs. Geben Sie die eingegebenen Objekte der Mediathek aus. Durchsuchen Sie die Mediathek nach einem von Ihnen festgelegten Suchterminus und geben die erhaltenen Medien iterativ aus.

#### Aufgabe 46:

#### Mediathek III

Schreiben Sie in der Klasse **Mediathek** eine Methode `boolean ausleihen(Medium m)`, die eine Ausnahme `NoSuchElementException` wirft, falls das Medium `m` nicht in der Mediathek enthalten ist. Die Methode liefert `true` zurück, falls das Medium erfolgreich ausgeliehen werden konnte. Analog dazu soll eine Methode `boolean zurueckgeben(Medium m)` bereitgestellt werden, die ebenfalls eine Ausnahme `NoSuchElementException` wirft.

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013  
Bearbeitungszeit: 120 Minuten

Name	_____
Matr.Nr	_____
Punkte Gesamt	_____ von 40
Punkte Note	_____

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) **4.** Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) **5.** Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

# Klausur Programmierung I

## Norbert Müller

### Wintersemester 2020/2021

#### Aufgabe 1:

man sollte beurteilen, ob die Syntax stimmt, wenn ja sollte man den Wert angeben der dabei rauskam, wenn nein, sollte man erklären, wo der Fehler liegt

- 1) boolean b = !(4<2);
- 2) int Int = 42;
- 3) Integer int = 42;
- 4) double d = (int) 42;

die letzten beiden weiß ich leider nicht mehr

#### Aufgabe 2:

Auswertungen von Funktionen

```
long eins = 1; int zwei = 2; float drei = 3.0; double vier = 4.0;  
eins / zwei + eins / vier * drei
```

man sollte Klammern setzen, um zu verdeutlichen, in welcher Reihenfolge das ausgewertet wird und auch angeben welchen Typ das Ergebnis hat wenn man sich die 4 Operatoren unabhängig betrachtet (also welchen Typ hat das Ergebnis, wenn man eins / zwei rechnet usw.) und man sollte den Wert angeben der dabei rauskommt

#### Aufgabe 3:

- das Verhalten einer vorgegebenen Methode beschreiben
- diese Methode umschreiben, sodass keine while-Schleife mehr vorhanden ist

#### Aufgabe 4:

Strings

man sollte eine Methode schreiben, die zwei Strings als Parameter hat und überprüft, ob der erste zweimal den zweiten String enthält, wenn ja soll das was dahinter steht zurückgegeben werden, wenn nein soll „null“ zurückgegeben werden

z.B. s1 = „xyzxyzABC“ s2 = „xyz“ dann sollte „ABC“ zurückgegeben werden  
s1 = „abab“ s2 = „ab“ dann sollte „“ zurückgegeben werden

#### Aufgabe 5:

Rekursion

es war eine Funktion gegeben

g(x) = „leer“, wenn der übergebene String leer ist  
= s (der String selbst), wenn die Länge des Strings ungerade ist  
= g(s1) + „#“ + g(s2), in allen anderen Fällen

s1 ist dabei die erste Hälfte von dem String und s2 die zweite Hälfte

z.B. s = „ab“ → „a#b“  
s = „abcd“ → „a#b#c#d“  
s = „abcdef“ → „abc#def“

#### Aufgabe 6:

Arrays

man sollte eine Methode schreiben, die zwei Arrays als Parameter hat und überprüft, ob mindestens zwei Zahlen aus dem ersten Array auch in dem zweiten Array vorkommen

wenn ja sollte „true“ zurückgegeben werden, wenn nein „false“

z.B. A = {1,2,3}; B = {4,5,6,7} → false  
A = {1,2,3}; B = {1,2,6} → true

$A = \{\}; B = \{1,2,3,4,5,6\} \rightarrow \text{false}$

### Aufgabe 7:

#### Collections

es war eine Klasse Meeting vorgegeben, date und visitor als Variablen enthält und man sollte eine Klasse „PersonalCalender“ schreiben, die alle Meetings speichert

### Aufgabe 8:

#### Vererbung

man hatte eine Klasse „Song“ vorgegeben und sollte ein Interface „Playlist“ schreiben und dann eine Klasse „SimplePlaylist“, die das Interface implementiert  
in „Playlist“ waren die Methoden: void addSong(Song song), String play(String titel) und noch eine Methode, an die ich mich leider nicht mehr erinner  
bei addSong sollte ein neuer Song in ein Array Song[] songs aufgenommen werden  
bei play sollte der Song mit dem passenden Titel gesucht werden und zurückgegeben werden, falls der Song nicht enthalten ist, sollte „null“ zurückgegeben werden

### Aufgabe 9:

#### Stack

man musste zwei Methoden schreiben, die erste sollte prüfen, ob ein Element in dem Stack enthalten ist  
die zweite Methode sollte ein Element daraus löschen (pop wurde nicht vorgegeben; man musste mit Referenzen arbeiten)

### Aufgabe 10:

#### StringTokenizer

man hatte eine Klasse A(mir fällt der genaue Name nicht mehr ein), wo der Konstruktor 7 Parameter hat  
dann hatte man eine andere Klasse, bei der im Konstruktor ein String übergeben wurde der alle 7 Parameter-Inforationen enthält, die immer durch „:“ getrennt wurden  
man sollte die trennen und dann in den Variablen speichern und damit ein neues Objekt von A erstellen  
falls der String falsch formatiert ist, sollte „falsches Format“ zurückgegeben werden

# Grundlagen der Programmierung

Norbert Müller

Universität Trier – Fachbereich IV – Informatik

17. Februar 2022

Bearbeitungszeit: **120 Minuten**

Name	_____
Matr.Nr.	_____
Rechner	_____

- Hiermit bestätige ich, dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur zur Vorlesung Grundlagen der Programmierung (Programmierung I) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.
- Ich fühle mich gesundheitlich in der Lage, die Klausur anzutreten.

Ort, Datum: \_\_\_\_\_

Unterschrift: \_\_\_\_\_

Beachten Sie folgende Regelungen:

- Als Hilfsmittel sind nur die zur Verfügung gestellten Editoren (inkl. Java-Dokumentation) sowie die elektronisch zur Verfügung gestellten Unterlagen (Kurs zur Klausur auf der moodle-Plattform) erlaubt. Andere Hilfsmittel wie z.B. das Einloggen in andere Moodle-Kurse oder die Verwendung von Smartphones sind nicht erlaubt.
- In der Klausur sind **112 Punkte** erreichbar. Ziel ist, dass Sie davon etwa **100 Punkte** bearbeiten. Die Zahl der Punkte pro Aufgabe entspricht also in etwa der Bearbeitungszeit (in Minuten).
- Alle Lösungen sind über das Moodle-System abzugeben, d.h. dort abzuspeichern.
- Programme können i.d.R. unter Moodle auch getestet und vor-evaluierter werden. Die Testfälle werden bei der Korrektur allerdings durch neue Fälle ersetzt werden! Abgaben, die nicht compilierbar sind, führen zu deutlichen Punktabzügen!

Aufgabe	1	2	3	4	5	6	7	8	9	10	Gesamt
Möglich	7	6	14	8	8	12	15	15	15	12	112
Erreicht											

Note: \_\_\_\_\_

**1. Aufgabe: Primitive Datentypen und Termauswertung**

(7 P)

Bei folgenden Deklarationen von Variablen

```
double eins = 1.0d; double zwei = 2.0d; int drei = 3; int vier = 4;
```

soll der folgende Ausdruck ausgewertet werden:

```
eins / zwei + drei / vier * zwei  
// (1)   (2)   (3)   (4)
```

Geben Sie zunächst an, wie dieser Ausdruck ausgewertet wird, indem Sie im Ausdruck Klammern (entsprechend den Auswertungsregeln in Java) hinzufügen.

Geben Sie zudem an, welchen Typ und welchen Wert die Zwischenresultate der Operationen (1) bis (4) (entsprechend den Auswertungsregeln in Java) an den jeweiligen Positionen haben. Verwenden Sie für die Angabe des Wertes immer Literale des jeweiligen Typs.

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

**2. Aufgabe: Deklarationen und Casts**

(6 P)

Betrachten Sie die im folgenden angegebenen kurzen Abschnitte aus (evtl. nicht kompilierbaren) Java-Quelltexten. Geben Sie bei den syntaktisch korrekten Textabschnitten den jeweils zugewiesenen Wert an. Bei syntaktisch nicht korrekten Texten geben Sie bitte an, warum sie nicht korrekt sind.

1. double real = 0,101;
2. boolean test = !(0 == 1);
3. int value = 010-1;
4. string text = "0101";
5. int data = 0x10-1;
6. int veryLarge = 1234567890;

Tragen Sie Ihre Antworten bei der entsprechenden Aufgabe in Moodle ein.

### 3. Aufgabe: Quelltext verstehen und umformulieren

Betrachten Sie die folgende Methode:

```
public static boolean f( int[] data ) {

    if ( data == null ) return false;

    int i = 0;
    int value = data[i];

    while ( i < data.length ) {
        if ( data[i] != value ) return false;
        i++;
    }
    return true;
}
```

- (a) Erklären Sie in Worten das Verhalten dieser Methode (d.h. bei welchen Parametern sie welchen Rückgabewert hat): Wann liefert sie Rückgabewert `true`, wann `false`? Beschreiben Sie dabei NICHT die Arbeit einzelner Zeilen des Quelltextes; zwei oder drei ordentlich formulierte Sätze sollten reichen! Achten Sie auf Sonderfälle! (8 P)
- (b) Implementieren Sie den Algorithmus aus `f` analog neu in einer Methode `g`: (6 P)
- Die neue Methode `g` muss für alle(!) möglichen Parameter die gleichen Resultate wie `f` liefern.
  - Dabei darf `g` jedoch *keine* `while`-Anweisung enthalten.

Den Quelltext gibt es in Moodle sowohl unter Aufgabe 03a als auch unter Aufgabe 03b. Ihre Lösung zu Aufgabenteil (a) soll als Freitext in Aufgabe 03a eingetragen werden, die Lösung zu Teil (b) soll unter A3b.java bei Aufgabe 03b in Moodle gespeichert werden; eine zusätzliche Klasse Evaluation.java zum Test einfacher Beispiele (aber nicht aller interessanten Fälle!) ist dort vorgesehen.

### 4. Aufgabe: Strings

(8 P)

Schreiben Sie eine Funktion `stringTest`, die zwei formale Parameter `String s1` und `String s2` hat und testet, ob `s1` die Verkettung `t + s2 + t` des Strings `s2` mit einem zweifach zu verwendenden anderen (von Ihnen zu findenden und zunächst unbekannten) String `t` ist. Wenn dies der Fall ist, soll `t` zurückgegeben werden, ansonsten die `null`-Referenz.

Beispiele:

```
stringTest("xyzABCxyz", "ABC") ~> "xyz"
stringTest("11222211", "2222") ~> "11"
stringTest("ABC", "ABC") ~> ""
stringTest("XaBab", "B") ~> null
```

Sie können davon ausgehen, dass weder `s1` noch `s2` den Wert `null` hat.

Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 04 in der Datei A4.java, mit der Möglichkeit zur testweisen Evaluierung durch Evaluation.java. In der dortigen Datei A4.java ist die Lösung abzuspeichern.

## 5. Aufgabe: Rekursion

Implementieren Sie die folgende Funktion  $g$  als rekursive Funktion mit dem Rückgabetyp `String`, einem Parameter `s` vom Typ `String` und zwei Parametern `m, n` vom Typ `int`:

$$g(s, m, n) = \begin{cases} \text{null}, & \text{falls } m \geq n \text{ oder } m < 0 \text{ ist oder } n > s.length() . \\ s.substring[m, n], & \text{falls ansonsten } m+1 = n \text{ ist,} \\ g(s, (m+n)/2, n) + g(s, m, (m+n)/2), & \text{in allen anderen Fällen} \end{cases}$$

Beispiele:  $g("abcde", 0, 5) = "edcba"$ ,  $g("abcde", 0, 3) = "cba"$ ,  
 $g("abcde", 1, 4) = "dcb"$ ,  $g("abcde", 2, 3) = "c"$

(Semantik von  $g$ : Es wird aus `s` der Teilstring von Position `m` bis Position `n-1` bestimmt, aber in umgekehrter Zeichenreihenfolge. Dies ist aber für die Lösung nicht relevant.)

Ihre Lösung darf dabei außer den Parametern und evtl. lokalen Variablen keine weiteren Variablen verwenden, insbesondere sind auch Schleifen nicht erlaubt. (Ansonsten wird die Abgabe mit 0 Punkten bewertet, selbst wenn sie die richtigen Resultate liefert. Dies gilt analog für Lösungen, die nur die obigen Beispielresultate reproduzieren.)

Sie dürfen dabei davon ausgehen, dass `s` nicht die `null`-Referenz ist. Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 05 in der Datei `A5.java`, mit der Möglichkeit zur testweisen Evaluierung durch `Evaluation.java`. In der dortigen Datei `A5.java` ist die Lösung abzuspeichern.

(12 P)

## 6. Aufgabe: Arrays

Implementieren Sie eine Methode `int[] mirrorArray(int[] data)`, die ein (neues!) Array zurückgibt, in dem die Werte aus `data` in umgekehrter Reihenfolge enthalten sind. Die Inhalte des Feldes `data` müssen dabei erhalten bleiben. Es ist möglich, dass `data` beim Aufruf die `null`-Referenz ist; in diesem Sonderfall soll auch die Null-Referenz zurückgegeben werden.

Beispiel:

Parameterfeld `data`: { 1, 2, 3, 4, 5 }  
`mirrorArray(data)`: { 5, 4, 3, 2, 1 }

Parameterfeld `data`: { 12345, 54321 }  
`mirrorArray(data)`: { 54321, 12345 }

Parameterfeld `data`: `null`  
`mirrorArray(data)`: `null`

(Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 06, inklusive einer passenden `main`-Methode und der Möglichkeit zur testweisen Evaluierung. Dort ist auch die Lösung abzuspeichern.)

## 7. Aufgabe: Collections

(15 P)

Vorgegeben ist eine Klasse `ExamResult`, bei der ein Objekt den Namen einer Person (`String name`) und die Note (`int mark`) speichern kann, die die Person bei einer Prüfung erreicht hat.

Ein passender Konstruktor und alle benötigten Getter-Methoden sind hier vorgegeben, so dass diese Klasse ebenso wie die testende Klasse `Evaluation` unverändert bleiben können (und müssen).

Sie sollen nun als Ergänzung dazu eine Klasse `ClassResults` implementieren. Realisieren Sie dafür die folgenden Teilaufgaben:

1. Jedes Objekt der Klasse `ClassResults` soll folgende Informationen speichern: Den Namen einer Prüfung (`String exam`) und zudem eine Liste an Einzelergebnissen (`ArrayList<ExamResult> results`), die die bei dieser Prüfung erreichten Resultate enthält.
2. Implementieren Sie den von `Evaluation` benutzten Konstruktor und ebenfalls die Getter-Methoden `getExam()` und `getResults()`.
3. Eine Methode `void registerMark(ExamResult result)` wird benötigt, um neue Prüfungsergebnisse zur Ergebnisliste eines `ClassResults`-Objektes hinzuzufügen.
4. Außerdem soll jede `ClassResults`-Instanz dazu in der Lage sein, zu einer gegebenen Note alle Personen zu finden, die diese Note erlangt haben. Sie sollen eine entsprechende Methode

```
ArrayList<String> findExaminees(int mark)
```

implementieren, die eine Liste mit den entsprechenden Personen zurückgibt.

5. Zugriff auf die Instanzvariablen der Klasse soll NUR mit dem Konstruktor und den obigen Methoden möglich sein. Diese Methoden und der Konstruktor sollen hingegen frei zugänglich sein.

Zum Testen Ihrer Implementierung ist die Klasse `Evaluation` gegeben. Diese überprüft aber bewußt nur rudimentär, ob die einzelnen Methoden vorhanden sind und bei den wenigen Testfällen die richtigen Rückgabewerte geliefert werden.

(15)

## 8. Aufgabe: Vererbung

Schreiben Sie eine konkrete Klasse `Cover` und ein Interface `Stream` wie im folgenden spezifiziert:

Vorgegeben ist eine abstrakte Klasse `Song`. Sie liefert über vier abstrakte Methoden Daten über ein Musikstück: den Titel (`String title()`), die Künstler (`(String artist())`) und die Spieldauer (`double length()`) des Stücks. Außerdem gibt es Zusatzinformationen (`String additionalInfo()`) zum Stück. Diese Klasse und die Testklasse `Evaluation` dürfen nicht verändert werden.

- Implementieren Sie nun eine konkrete Subklasse `Cover` von `Song`. Es geht hierbei um Musikstücke, die bereits früher einmal veröffentlicht wurden und jetzt neu gecovert worden sind.
- Zusätzlich zu den Daten über Titel, aktuelle Künstler und Spieldauer, die über `Song` bereits abfragbar sind, soll `Cover` daher folgende Informationen bereitstellen:
  - welche Künstler das Original des Songs eingespielt haben  
(zu speichern in `String originalArtist`) und
  - in welchem Jahr das geschehen ist  
(zu speichern in `int originalPublication`).
- Folgende Methoden müssen Sie dazu implementieren:
  - Ein Konstruktor der Form

```
Cover(String title, String artist, double length,  
      String originalArtist, int originalPublication)
```

- Zwei getter-Methoden für die zusätzlichen Objektvariablen  
(also `getOriginalArtist()` und `getOriginalPublication()`)
- Die vier konkretisierten Methoden für `Song`. Bei `additionalInfo()` soll die Form "originally by ... from year ..." mit den zusätzlichen Daten aus `originalArtist` und `originalPublication` gewählt werden; ansonsten sollen die entsprechenden Komponenten aus dem Konstruktor unverändert zurückgegeben werden.

Vergeben Sie dabei den Zugriffsmodifikator `private` und das Attribut `@Override` so oft wie möglich.

- Für eine Austrahlung im Radio wird als Kerninformation nur die Spieldauer benötigt. Daher implementiert `Song` ein (von Ihnen zu schreibendes) Interface `Stream`, das aber nur `length()` bereitstellen soll.

Eine rudimentäre Testklasse `Evaluation` ist vorgegeben, um Ihre Implementierung zu testen. Diese Klasse enthält einige Tests zur Überprüfung, ob die genannten Methoden und Funktionalitäten implementiert wurden.

## 9. Aufgabe: Exceptions und Strings

(15 P)

Implementieren Sie eine Methode

```
public static String checkComparison(String calculation)
```

die überprüft, ob der Parameter `calculation` eine korrekte (und korrekt formatierte) Ungleichung entsprechend folgender Form darstellt:

```
"1 < 2"      "54321 > 12345"      "-1111 < +1111"
```

Im Parameter dürfen dabei nur folgende Teile vorkommen:

- ganze Zahlen in der üblichen Dezimalform, evtl. mit Vorzeichen, so dass `int` als Zahltyp reicht und `Integer.parseInt()` verwendet werden kann (auch zum Test auf Korrektheit der Syntax),
- eines der Zeichen '`>`' und '`<`' (als Vergleichsoperatoren)

Diese drei Teile sind durch ein oder mehrere Leerzeichen voneinander getrennt, so dass problemlos ein `StringTokenizer` zur Zerlegung benutzt werden kann.

Ihre Lösung soll nun folgendes leisten:

- Sollte der String korrekt formatiert sein und zudem die Ungleichung stimmen, so geben Sie folgenden String zurück:

```
valid
```

- Sollte der String zwar korrektes Format haben, aber die Werte nicht übereinstimmen, geben Sie einen String im folgenden Format zurück, wobei `left` und `right` die gefundenen Werte sein sollen:

```
left is not smaller than right
```

bzw.

```
left is not greater than right
```

- Wenn der String falsch formatiert ist, sollen Sie eine `RuntimeException` werfen, die folgende Nachricht enthhält:

```
syntax error
```

Zur Vereinfachung: Auf Zahlierläufe brauchen Sie nicht zu achten. Der Parameter ist nicht `null`. Eventuelle intern auftretende Exceptions, z.B. aus `Integer.parseInt()`, müssen Sie aber geeignet abfangen.

Einige Beispiele für falsche Gleichungen sind:

```
"-1 > +1"      "54321 < 12345"      "1111 < 1111"
```

Einige Beispiele für falsch formatierte Strings sind:

```
"5<10"          "10 = 5"           "5 < 1 0"  
"+ 10 > 9"       "10.0 < 11"        "-5 > - 4"
```

## 10. Aufgabe: Listen

Betrachten Sie die gegebenen Klassen `Liste` und `Elem`, die den in der Vorlesung vorgestellten Varianten sehr ähnlich sind. Sie implementieren eine Minimalfassung einer verketteten Liste von `String`-Werten. Die bereits vorgegebene Methode `addFront(String s)` fügt den `String s` am Anfang der Liste ein.

Erweitern Sie die Klasse `Liste` um die folgenden zwei Methoden:

- Die Methode `ausgeben()` soll einen String mit den Werten sämtlicher Elemente in der Reihenfolge ausgeben, in welcher sie ausgehend von `start` in der Liste enthalten sind. Sie sollen dabei durch einzelne '+'-Zeichen getrennt werden. Insbesondere soll nach dem letzten Element kein '+' stehen.

Der erste Aufruf von `ausgeben()` in der `Main.main()`-Methode soll also die folgende Ausgabe produzieren::

was+wie+wer+das+die+der

- Die Methode `int ersetze(String s, String t)` soll in der Liste den String `s` überall durch den String `t` ersetzen und zudem als Rückgabewert angeben, wie oft `s` ersetzt wurde.

Die weiteren Aufrufe von `ausgeben()` in der `Main.main()`-Methode sollen daher die folgende Ausgabe produzieren:

was+wie+wer+das+die+wer  
was+wie+wie+das+die+wie  
was+wie+wie+das+die+wie

Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 10, dort ist auch die Lösung abzuspeichern. Zu dieser Aufgabe finden Sie neben `Liste` und `Elem` auch eine Testklasse `Main` mit einer `main`-Methode in Moodle. Die Klassen `Elem` und `Main` dürfen nicht verändert werden. Sie dürfen zur Vereinfachung davon ausgehen, dass keine `null`-Strings zu verarbeiten sind.

## Gedankenprotokoll Nachklausur Programmierung 1 am 10.04.2018

### Aufgabe 1 + 2

Datentypen (Ergebnis, Datentyp d. Ergebnis, Erklärung)

1.  $1 / 2.0 * 3$
  2. "12"+10
  3. int 1.999
  4. double x = (int) 1.999
  5. double int = 2.0
- ...

### Aufgabe 3

a) Code schriftlich erklären.

```
static int funktion (int [] a) {  
    int summe = 0;  
    for (int x : a) {  
        summe = summe + x;  
    }  
    return summe;  
}
```

b) gleichen Code nochmal implementieren, aber nicht in einer for-Schleife und ohne auf den Originalcode zuzugreifen

### Aufgabe 4

Methode schreiben, um zu zählen wie oft String t ein Teilstring von String s ist  
z.B. String s = „aabaaba“ und String t = „ab“ dann soll der Wert 2 zurückgegeben werden

### Aufgabe 5

Methode schreiben: deleteMinMax

In einem Feld sollen die Minima und Maxima gefunden werden und anschließend durch Nullen ersetzt werden, z.B. Feld der Größe 5: 1 2 4 1 3 sieht danach so aus: 0 2 0 0 3

### Aufgabe 6

Rekursion mit Strings

$$g(s) = \begin{cases} 0, & \text{falls String } s \text{ leer ist} \\ 2*g(t), & \text{falls } s \text{ aus 'a' + t besteht} \\ 3+g(t), & \text{falls } s \text{ auf 'b' + t + 'c' besteht} \\ s.length(), & \text{in jedem anderen Fall} \end{cases}$$

'a' , 'b' , 'c' sollen ganz normale Buchstaben darstellen

### Aufgabe 7

?

### *Aufgabe 8*

Abstrakte Klasse Prüfung, davon abgeleitete Klassen Klausur und Mündlich schreiben

Prüfung besitzt fach (String) und eine id (int), wobei die id ab 1 hochzählen soll

Klausur besitzt geschrieben (boolean)

Mündlich besitzt zudem ein Datum

### *Aufgabe 9*

Listen: vorgegebenes Programm wie folgt ergänzen

Methode 1 ergänze (String s, String t): String t in der Liste hinter String s angehangen werden

Methode 2 ... ?

anschließende Ausgabe der richtigen Reihenfolge der Liste

### *Aufgabe 10*

File IO mit Exception: 3 gegebene txt.-Dateien mit jeweils normalen Zahlen je Spalte (int) und

Kommentare (gekennzeichnet durch #)

Es soll berechnet werden die Summe der Zahlen in den jeweiligen Dateien. Eine Exception soll geworfen werden, wenn Kommentare in der Datei vorhanden sind (mit Lokalisation)

### *Aufgabe 11*

Collections: Dominospiel realisieren

# Programmierung I Klausur WiSe 2018/2019

## **Aufgabe 1 (schriftlich):**

Man bekommt Rechnungen mit unterschiedlichen Zahltypen gegeben und muss das Ergebnis, den Datentyp des Ergebnisses und eine kurze Begründung dafür abgeben, in dieser Aufgabe mit eher untypischen Typen, z.B. Hexadezimal, Long, Float etc.

## **Aufgabe 2 (schriftlich):**

Man bekommt wie in Aufgabe 1 eine Rechnung/Zuweisung, bestehend aus 2-4 Zahlen verschiedener Typen und muss das Ergebnis angeben. Falls die Rechnung nicht funktionieren würde (also zu einem Error führen würde), muss man eine kurze Begründung angeben, warum das Programm hier abstürzt. Ein paar Beispiele für die Rechnungen:

- 1) int a = 1.999;
- 2) int double = 3.0;
- 3) "12"+10
- 4) int b = 1 / 3.0 \* 2;

## **Aufgabe 3:**

Teil a) Schriftlich, Code auf Papier gegeben, man muss kurz erklären, was der Code macht. Der Code in der Klausur war eine Methode mit einem int[] **feld** und einem int **parameter**. Die Methode hatte eine int **summe** gegeben. Mit dem **parameter** wurde ein switch aufgerufen, falls **parameter == 0**, wurde mit einer for-each-Schleife der jeweilige Feldeintrag zur **summe** addiert, falls **parameter == 1**, dann wurde mit einer for-each-Schleife zuerst der Feldeintrag quadriert und dann zur **summe** addiert. Am Ende der Methode wurde **summe** zurückgegeben. Code:

```
public int methode(int[] feld, int parameter) {  
    int summe = 0;  
    switch (parameter) {  
        case 0: {  
            for(int i : feld) {  
                summe += i;  
            }  
        }  
        case 1: {  
            for(int i : feld) {  
                summe += i*i;  
            }  
        }  
    }  
    return summe;  
}
```

Teil b) Den gleichen Code implementieren, aber ohne for-Schleifen

## **Aufgabe 4:**

Man soll eine Methode schreiben, die einen String **str** einliest, falls dieser String zweimal genau aus dem gleichen String **t** besteht, also **str = t + t**, dann soll String **t** zurückgegeben werden, ansonsten **null**. (Beispiel: "TestTest" soll "Test" zurückliefern, "Auto" soll **null** zurückliefern) Lösung:

```
public static boolean stringTest(String str) {  
    int haelfte = str.length() / 2;  
    String teil1 = str.substring(0, haelfte);  
    String teil2 = str.substring(haelfte, str.length());  
    return teil1.equals(teil2);  
}
```

## **Aufgabe 5:**

Feld mit Integers wird in eine Methode übergeben, falls ein Feldeintrag **genau** zweimal im Feld vorkommt, soll *TRUE* zurückgegeben werden, sonst *FALSE*. Lösung:

```
public static boolean feldTest(int[] feld) {  
    for(int i=0; i < feld.length; i++) {  
        int eintrag = feld[i];  
        int counter = 0;  
        for(int j=0; j < feld.length; j++) {  
            if(feld[j] == feld[i]) counter++;  
        }  
        if(counter == 2) return true;  
    }  
    return false;  
}
```

## **Aufgabe 6:**

Man muss zwei Klassen Professor und Student schreiben. Professor hat einen Namen, Fachbereich und eine int-Variable **zeit** (diese werden im Konstruktor gesetzt, wird von der Evaluation automatisch getan). Jeder Student bekommt im Konstruktor einen Namen, Fachbereich und einen Professor **prof** übergeben. Falls **prof** den gleichen Fachbereich besitzt, wie der gerade konstruierte Student, dann wird dieser Student in eine ArrayList des Typs Student in der Professor-Klasse eingefügt. Man soll dann eine Methode **beratung** schreiben, die die ArrayList durchläuft und einen String der Form „Student xyz wird beraten“ ausgibt, falls der Professor noch **zeit** hat, ansonsten wird der String „Professor xy hat keine Beratungszeit mehr übrig“ ausgegeben.

## **Aufgabe 7:**

Man bekommt Strings der Form „1234;Harry Potter; This is a book“ übergeben, die erste Zahl soll die ISBN-Nummer darstellen, der Mittelteil ist der Name des Buchs und der letzte Teilstring ist ein Kommentar. Für jede Eingabe sollte die Ausgabe wie folgt aussehen (am Beispiel):

isbn:	1234
Name:	Harry Potter
Abstrct:	This is a book

Falls ein Teilstring leer ist oder zu wenige Teilstrings vorhanden sind, soll eine Exception geworfen und gecatched werden, die dann „Invalid input“ ausgibt.

## **Aufgabe 8:**

Man bekommt eine abstrakte Klasse Kamera gegeben, davon abgeleitet sind die Klassen Smartphone und Spiegelreflexkamera. Jede der Klassen hat eine int **bilder**, die von 1 hochzählen soll wenn ein Bild gemacht wird (dafür gibt es Methode **takePicture()**). Zudem hat jede neue Instanz (von Smartphone oder Spiegelreflexkamera) eine individuelle ID.

## **Aufgabe 9:**

Man bekommt eine verkettete Liste **L**, ein Element **x** und eine int **n** gegeben und soll eine Methode schreiben, die **x** an der **n**-ten Stelle der Liste einfügt (darauf achten, dass **n** nicht größer ist als die Liste)

Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage nach Ergebnis, Datentyp,  
Erklärung in eine Tabelle auf Klausurblatt schreiben

Bsp: int m = 1; double n = (int) 1.0;

Aufgabe 2 (6P)

6 Aufgaben: Frage zu Programmschnipseln nach Ergebnis oder wenn es nicht läuft den Grund

**Aufgabe 3**

- a) Programm auf Blatt Papier schriftlich erklären – (6P)
- b) Programm ohne for und ohne switch programmieren (8P)

```
Public static int xx (long feld[], int/long? parameter){
```

```
    Int value = 0;
```

```
    Switch(parameter)
```

```
        Case 0: for (long k : feld) value += v; return value;
```

```
        Case 1: for(long k : feld) value += v*v; return value;
```

```
}
```

**Aufgabe 4 (8P)**

Methode schreiben die einen String übergeben bekommt;  
dieser String s soll untersucht werden, ob er aus zwei gleichen Teilstrings t besteht;  
wenn ja soll der Teilstring t zurückgegeben werden – sonst null;

Bspw: String s = abab hat t = ab;

**Aufgabe 5 (8P)**

Rekursion: mit if und else if und else die Anweisungen auf dem Blatt entsprechend implementieren

## Aufgabe 6 (12P) - Arrays

Es wurde ein Array einer Methode übergeben;

Man sollte schauen, ob irgendeine Zahl im Array genau 2x vorkommt,  
wenn ja diese ausgeben,  
wenn nicht false;

## Aufgabe 7 (15P) – Collections

Es gab: eine Klasse Professoren und eine Klasse Studenten und eine main-Klasse

Klasse Professor:

- String domain (Fachbereich)
  - int officeHours
  - String name
- LinkedList<studenten> list;

Klasse studenten:

- String domain (Fachbereich)
- int time (benötigte Zeit für Beratung)
- String name

Man sollte nun zwei Methoden schreiben:

1. Methode bekommt ein student übergeben  
ist dieser vom selben Fachbereich (domain) wie der Prof;  
so soll er in die Liste vom Prof eingetragen werden  
sonst sollte eine Ausgabe auf Konsole erfolgen (Prof x ist nicht zuständig für Student y)
2. Methode soll die Liste vom Prof abarbeiten  
es sollen die studentnamen ausgegeben werden  
jeder student benötigt seine „time“  
diese wird von „officeHours „ bei jeder Ausgabe vom studenten um studenten.getTime()  
verringert  
Ist officeHours auf 0 so sollen alle restlichen Studenten in der Liste ausgegeben werden;  
mit den Zusatz (in etwa) „ Prof x konnte Student y nicht bearbeiten“  
(alle Ausgaben war nicht auf deutsch sondern auf englisch)

## Aufgabe 8 (15P) Exceptiones

Es wurde ein String an eine Methode übergeben, die geschrieben werden sollte.

Dieser String beinhaltet das Schema „ISBN; titelvonBuch; abstract“

bspw: „123;rotkappe; abstract“

Es konnte passieren, dass der String null ist, nicht alle Angaben enthalten sind;

Oder „;“ 1x fehlt

Der String sollte auf drei String aufgeteilt werden;

Die ISBN sollte in einen int Variable gespeichert werden;

Falls isbn zu lang war oder keine Zahl war sollte eine Exception fliegen und den Wert auf -1 von isbn setzen;

Ansonsten sollte eine Ausgabe wie folgt erfolgen; wichtig war hier die Bündigkeit(tabellenartig)

ISBN: 125

Titel: TitelBuch

Abstract: Nicht abstract!

## Aufgabe 9 (15P) Vererbung

- Abstracte class camera schreiben:
  - o String model
  - o Int id (soll bei 0 starten ; inklusive )
  - o Int pictures( Anzahl der getätigten Bilder, soll bei 0 starten, inklusive)
  - o Boolean hasObjektiv (ob man Objektiv abschrauben kann)
  - o Abstracte methode void takePicture();
  - o toString methode
- von camera abgeleitet
  - o class Smartphone
    - sollte im Konstruktor name übergeben bekommen; und im Konstruktor (super(name, false) wegen kein Objektiv abnehmbar)
  - o class SRL-Kamera
    - sollte im Konstruktor (model, String Objektivname) erhalten;
    - im Konstruktor true wegen Objektiv
  - o Methode takePicture in Smartphone und SRL-Kamera implementieren; wenn aufgerufen soll sich die Zahl picture erhöhen und ausgeben werden plus mit welcher Kamera ( id und model) und wenn vorhanden Objektiv

### Aufgabe 10 (15P)

Abgewandelte Version von Queue mit Elem; 2 Methoden schreiben;  
es gab keine getter/setter-Methoden bei Elem; es gab nur String value und Elem next;  
methode enqueue, toString(?) waren vorgegeben;

1. Methode: Insert ( int index, String word) – soll an stelle index word einfügen
2. Methode: bekommt einen String übergeben; der aus mehreren Teilen besteht;  
diese Teile sind mittels Leerzeichen voneinander getrennt;  
diese Teile sollen nun in die bestehe Queue hinten angehängt werden;  
Jedes Teil ein neues Elem;

# Programmierung 1 Wintersemester 2017/2018

Bei: Prof. Dr. Norbert Müller

Insgesamt gab es 110 Punkte, wovon 100 erreicht werden mussten.

**Aufgabe 1) (7 Punkte)**

Grundrechenoperationen von verschiedenen Dateitypen. Ergebnis, Dateityp vom Ergebnis, Erklärung.

**Aufgabe 2) (7 Punkte)**

Zuweisungen. Zum Beispiel: int a = 1,2? Was passiert dann? Double d = (int) 1.2;

**Aufgabe 3) (6 + 8 Punkte)**

- a) Algorithmus erklären (auf Papier)
- b) Den selben Algorithmus programmieren aber ohne Switch / For

**Aufgabe 4) (8 Punkte)**

Teilstring testen (Aufgabe von SS17):

„Man bekommt einen String, der aus 3 Teilstrings besteht und so aufgebaut ist:  
String1String2String1. Die Strings sind alle gleich lang. Das Programm soll prüfen ob der erste  
Teilstring gleich dem dritten Teilstring ist, und wenn das der Fall ist, den mittleren  
zurückgeben. Als Beispiel: Der String lautet "AutoBuchAuto". Es soll "Buch" zurück gegeben  
werden.“

**Aufgabe 5) (8 Punkte)**

Integer Array mit 5 Zahlen. Alle positiven Zahlen sollten die Position tauschen.

Also: [1, -1, 2, 4, -3, 5] => [5, -1, 4, 2, 1]

**Aufgabe 6) () Rekursion**

Es gab eine Methode G(int k) die ein Integer zurückgibt, je nach Wert von k sollte ein  
bestimmter Wert zurückgegeben werden, der teilweise noch einmal G aufruft.

**Aufgabe 7) ()**

**Aufgabe 8) (12 Punkte)**

Abstrakte Klassen und Vererbung

**Aufgabe 9) (12 Punkte)**

Verkettete Listen mit Elem (siehe VL).

**Aufgabe 10) (12 Punkte)**

File I/O

# Programmierung I – 1. Klausur WS 21/22 – bei Herr Prof. Dr. Müller

100 von 112 Punkte für 1.0 erreichen

## Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage was System.out.println(...) ausgeben wird

Bsp.: System.out.println("10" + 20) Ausgabe: 1020

## Aufgabe 2 (6P)

6 Aufgaben: Initialisierung von Datentypen, Frage nach korrekter Syntax, wenn ja, Wert angeben, wenn nein, den Grund angeben

Bsp.: string test = "abc"; (Fehler, String muss großgeschrieben werden)  
Double d = 6,5; (Fehler, double Zahlen werden mit einem Punkt initialisiert)

## Aufgabe 3 (14P)

- Programm auf Blatt Papier schriftlich erklären – (6P)
- Programm ohne while-Schleife schreiben, z.B. mit einer for-Schleife (8P)

## Aufgabe 4 (8P) – Stringverarbeitung

Eine Methode bekommt String s1 und String s2 übergeben. Zu überprüfen war, ob der String s2 in s1 enthalten ist und dann der Teilstring t vor s2 und hinter s2 wieder stand, also quasi die Form hat:

s1 = (t + s2 + t) - Bsp.: s1 = abcxyzabc , s2 = xyz

Wenn das gestimmt hat, sollte t übergeben werden, ansonsten null und wenn t = "" war, dann sollte "" zurückgegeben werden

## Aufgabe 5 (8P) – Rekursion

Rekursion: einfach nur die verschiedenen Fälle abschreiben und wieder rekursiv aufrufen lassen

## Aufgabe 6 (12P) – Arrays

Es wurde ein Array einer Methode übergeben und man sollte wieder ein Array zurückgeben, allerdings sollten dann alle Werte umgedreht sein.

Falls das Array eine Null-Referenz ist, sollte auch null zurückgegeben werden

## Aufgabe 7 (15P) – Collections

Man hatte eine ArrayList <ExamResults> (ExamResults waren wiederum Objekte, bestehend aus Person, Note und Fach) und sollte dafür 2 Methoden schreiben:

1. Eine Methode die neue ExamResults einfügt
2. Eine Methode, welche eine „**int mark**“ übergegeben bekommt, die alle Objekte der ArrayList durchläuft (Iterator bietet sich da an) und dann alle Personen in eine neue ArrayList einträgt, welche die Note „**mark**“ haben.  
Anschließend sollte die ArrayList<String> mit den Personennamen zurückgegeben werden

## Aufgabe 8 (15P) – Exceptions

Man bekam einen String der Form : „Zahl“ gefolgt von „> oder <“ und dann wieder eine „Zahl“. Getrennt waren die Strings nur mit Leerzeichen, sodass sich ein StringTokenizer anbietet, in Kombination mit der trim() Funktion.

Man sollte überprüfen, ob der String diesem Format entspricht, wenn nein, sollte eine RuntimeException geworfen werden, wenn ja, dann sollte überprüft werden, ob die Werte der beiden Zahlen auch mit dem „> oder <“ Zeichen übereinstimmen. Wenn das der Fall war, dann sollte „valid“ zurückgegeben werden, ansonsten zwei Texte, wie „links ist nicht größer als rechts“ oder analog „links ist nicht kleiner als rechts“.

## Aufgabe 9 (15P) – Vererbung

Man hatte 4 Klassen und musste ein Interface schreiben, wovon eine abstrakte Klasse „Song“ geerbt hat und dann sollte man eine Klasse „Cover“ schreiben, welche von Song erbt.

Es wurde ein Konstruktor verlangt, sowie Initialisierung von den abstrakten Methoden, sowie ein @Override einer Stringmethode.

## Aufgabe 10 (12P) – Liste

Abgespeckte Version von der Vorlesung, man hatte nur die Klasse Elem und List (mit String Objekten) und musste folgende Methoden schreiben:

1. Eine Methode, die einmal durch die Liste läuft und alle Stringwerte zu einem String zusammenfasst, sodass man jeden Stringwert durch ein „+“ getrennt hat. Ein Beispiel für die Form ist: was+wie+wer+wo+wann  
Verlang war nur, dass hinter dem letzten String kein Plus mehr gesetzt wird
2. Eine Methode size(), welche beim Durchlaufen einmal zählen sollte, wie viele Objekte in der Liste drinnen sind

**Exam was in total 115 ; 15 is bonus.**

**(7 pts) Q1** It was about Data types.

<u>Example</u>	<u>Type</u>	<u>Explanation</u>
31+42		
42.0/10+ 5/4		
3.2d+4.3f		
!(5>6)		
True		

**(6 pts) Q2** It was about again data types and assignment of these data types and if the compiler works or not

e.g.

<u>Example</u>	<u>If the compiler works or not, if not the reason</u>
double r=(int)4;	

**Q3)**

a) Read the given code and explain what it is doing

```
public static boolean f(int[] data){  
    if(data==null) return false;  
    int i=0;  
    while(i<data.length){  
        if(data[i-1]>data[i]) return false;  
        i++;  
    }  
    return true;
```

b) Implement the above code without using while or do while.

**Q4)** You must write a method which return "abc" if "abcabcXY", "XY" (it is an example) like  
string t + t + s return t

```
public static String method(String first, String s){  
}
```

If String first in the form t+t+s and the methods should return t else null

e.g

"xyzxyzABC" must return "xyz"

"xyxyABC" must return "xy"

“11111111” must return “111”

“xyzxyyABC” must return null;

**Q5)** Writing method which return true if a char repeated three or more in an array ( ex return true for array={1 1 1 2 3} )

```
public boolean charRepeated(String[] data){  
}
```

Examples were like

{“1”, “2”, “3”, “4”, “5”, “5” } must return false

{“1”, “1”, “1”, “2”, “3”, “5” } must return true

**Q6)** Arrays

**Q7)** String and StringTokenizer

You are given Strings in the form of (.....;.....;.....;.....)

(String ;String ; String ;String)  
↓

This part must be converted from String to int  
In the question it says that use Integer.parseInt()

Eg. (1234;Marcus Halle; Bla Bla Publishing; Something More)

You need to printout;

ISBN : ..... (integer)

Author: ..... (String)

Publisher: .....(String)

Some other thing: ..... (String)

**(15 pts)Q8**) Interface

**(15 pts)Q9**) Collections

**(15 pts)Q10**) Queue

You have implement a method into Queue.java that add new element to the desired position

```
Public void addElem(Elem new, int pos){  
}
```

# **Programmierung I – 1. Klausur – Gedankenprotokoll**

## **– WS 19/20 – 18.02.2020 –**

Klausur bei Herrn Prof. Dr. Müller und Herrn Jun.-Prof. Dr. Weyers für Prog I und Prog Ia & Ib.  
**Insgesamt gab es 110 Punkte. 100 Punkte für 1,0.**

Bearbeitung der Aufgaben mit Eclipse oder IntelliJ ist sehr empfehlenswert.  
**110 Minuten Zeit.**

### **Aufgabe 1 (auf dem Blatt zu beantworten):**

Rechenoperationen mit verschiedenen Zahlen und Typen in tabellarischer Form.

Gefragt waren das **Ergebnis**, der **Typ** und eine **Begründung** des Zustandekommens des Ergebnisses – unter anderem mit „exotischen“ Typen wie Long, Float, Hexadezimal...

**Beispiele:** 42L – 0xa; 42.f + 13.d; „42“+32; 42./10+18/10; ...

### **Aufgabe 2 (auf dem Blatt zu beantworten):**

Ausdrücke mit verschiedenen Datentypen in tabellarischer Form.

Gefragt war, ob der dargestellte **Ausdruck** sich **kompilieren** lässt oder nicht.

Wenn ja, dann war das **Ergebnis** des Ausdrucks gefragt,  
wenn nicht, dann der **Grund**, warum es nicht kompilieren sollte.

**Beispiele:** boolean b = (4 < 2); Int int = 42; int Int = 42.0;  
int [] a = {43,22} → int [2]; char c = „d“

### **Aufgabe 3:**

Quelltextverständnis.

#### **a) (auf dem Blatt zu beantworten):**

Es war ein Quellcode gegeben mit der Methode „f“, die einen boolean-Wert **true** liefert, wenn das übergebene Array absteigend sortiert ist, **false** liefert, wenn es nicht absteigend sortiert ist, und wenn die Null-Referenz übergeben wird, wurde dies per **if-Anweisung** geprüft und eine **Exception** geworfen. Der Durchlauf des Arrays geschah in einer **while-Schleife**.

#### **b) (unter moodle abzugeben)**

Programmierung des Algorithmus als Methode „g“ ohne **while** oder **do-while-Schleife**.  
Somit war hier eine **for-Schleife** gefragt.

### **Aufgabe 4 (unter moodle abzugeben):**

Strings.

Es sollte eine Methode geschrieben werden, die überprüft,  
ob der String **s1** aus dem **Schema t + s2 + t** besteht und anschließend soll **t** zurückgegeben  
werden.

#### **Beispiele:**

**s1** = „AutoBuchAuto“, **s2** = „Buch“ → Rückgabe von **t** = „Auto“  
oder **s1** = „xxxyxxxxBeispielxxxxxxx“, **s2** = „Beispiel“  
→ Rückgabe von **t** = „xxxyxxxx“  
oder **s1** = „aaWsApfelwAss“, **s2** = „Apfel“ → Rückgabe einer **Null-Referenz**.

# Programmierung I – 1. Klausur – Gedankenprotokoll

## **– WS 19/20 – 18.02.2020 –**

Somit wurden in diese Methode die Strings **s1** und **s2** übergeben. Man musste zuerst herausfinden, was dieser **String t** ist und wenn **s1** aus **diesem Schema (also: t + s2 + t)** besteht, sollte **t** zurückgegeben werden. Andernfalls eine **Null-Referenz**.

### **Aufgabe 5 (unter moodle abzugeben):**

Rekursion.

Die zu programmierende Methode **g(String s, int n)** war mit den genauen Anweisungen auf dem Blatt angegeben.

Hier als vereinfachte Darstellung der zu programmierenden Abfragen der Rekursion:

$$g(\text{String } s, \text{int } n) = \begin{cases} 0, & \text{falls } n < 0 \\ g(s, s.\text{length}() - 1), & \text{falls } n < s.\text{length}() \\ g(s, s.\text{length}()), & \text{falls } s.\text{charAt}(n) == 'X' \\ g(s, s.\text{length}() + 1), & \text{falls } s.\text{charAt}(n) == 'Y' \\ 0, & \text{sonst} \end{cases}$$

### **Aufgabe 6 (unter moodle abzugeben):**

Arrays.

Es sollte eine Methode geschrieben werden, die **true** zurückgibt, falls das Array aus **mind. 3 gleichen Einträgen** besteht, andernfalls **false**.

Hier bietet sich eine 2-fach verschachtelte **for-Schleife** an, die mit einem Counter die Häufigkeit eines Eintrages überprüft und bei einem Counter von 3 **true** returnt.

Es bietet sich auch eine 3-fach verschachtelte **for-Schleife** ohne Counter an.

### **Aufgabe 7 (unter moodle abzugeben):**

Collections.

Es sollte eine **Klasse „Webshop“** geschrieben werden, die die Eigenschaften **String name** und **List <Produkte> products = new ArrayList<Produkte>();** enthält.

Die **Klasse „Produkte“** war schon vorgegeben mit **Namen, Preis** und den jeweiligen Gettern und Settern war vorgegeben.

Somit sollte ein **Konstruktor** `public Webshop (String name) {....}` geschrieben werden, der den übergebenen Namen in die Instanz hinzufügt. Also: `name = this.name;`  
Zudem waren folgende 2 Methoden zu schreiben:

**Product getMostExpensiveProduct (String s) {...}**

**Product getCheapestProduct (String s) {...}**

Hierbei sollte überprüft werden, ob der übergebene **Artikelname s** in der List **products** vorhanden ist, und anschließend sollte das **teuerste Produkt** und das **günstigste Produkt** unter dem **Artikelnamen s** zurückgegeben werden.

# Programmierung I – 1. Klausur – Gedankenprotokoll

**– WS 19/20 – 18.02.2020 –**

## **Aufgabe 8 (unter moodle abzugeben):**

Vererbung.

Es sollte zuerst ein **Interface „Publication“** geschrieben wurde, die die **abstrakten** Methoden `getDescription` und `getVenueDescription` enthält.

Als Nächstes sollte eine **Klasse „Article“** geschrieben werden, die das **Interface “Publication”** implementiert, der die Variablen `String title, String author, String venue, int year` und eine **eindeutig** und einmalig bezeichnete `int ID` (also static) zugeschrieben wird und letztlich ein passender **Konstruktor** geschrieben werden sollte.

Es sollten danach die jeweiligen **Getter** und **Setter**, die vorgegeben in der Main-Methode implementiert waren, als Methoden angelegt werden.

Als zusätzliche **Methoden** sollten `getDescription` und `getVenueDescription` die jeweiligen Inhalte des Artikels als **String** returnen.

Danach sollte eine **Klasse „Journal“** geschrieben werden, die die **Klasse „Article“** extendet, der die Variablen `int numberOfArticles` zugeschrieben werden und im **Konstruktor** zusätzlich zu den o. g. Eigenschaften auch `numberOfArticles` übergeben werden und letztlich per `super(...)` von „**Article**“ die gleichen Eigenschaften gesetzt werden sollten.

Zum Schluss sollte ein das folgende Array angelegt werden:

```
Article [] articles = new Article [numberOfArticles];
```

und die Methode `getDescription` sollte per `@Override` überschrieben werden und sollte zusätzlich zu dem o. g. Text auch Folgendes ausgeben:

```
return (...) + " containing " + numberOfArticles + " articles";
```

## **Aufgabe 9 (unter moodle abzugeben):**

Listen.

Es war vereinfacht eine **Klasse „Elem“** nur mit `next` und `value` - ohne `.getNext()` oder `.setNext()` - gegeben und eine **Klasse „Queue“** gegeben, in welcher folgende beiden Methoden implementiert werden sollten:

**1. Methode:** `public boolean isElem (String s) {...}`

**2. Methode:** `public void swap (String s1, String s2) {...}`

Die **1. Methode** sollte überprüfen, ob der **String s** in der Queue vorhanden ist, wenn ja sollte **true** zurückgegeben werden, wenn nein, sollte **false** zurückgegeben werden.

Die **2. Methode** sollte den 2 Referenzen vertauschen:

**Beispiel: „This for is fun lol“**

sollte mit `swap(“for”, “is”)` zu **“This is for fun lol”** werden.

# **Programmierung I – 1. Klausur – Gedankenprotokoll**

**– WS 19/20 – 18.02.2020 –**

## **Aufgabe 10 (unter moodle abzugeben):**

Objekte.

Es sollte eine **Klasse „Parser“** erstellt werden, die in einer Methode eine Instanz der **Klasse „Baby“**, die vorgegeben war, mit den Variablen **String name, String birthdate und int weight** erstellt.

Hierbei sollte der „**Parser**“ in dieser Methode einen String, wie z.B.

{name:Clara; weight: 1023; birthdate: 02-18-2020 },  
die Informationen „Clara“, 2023 und „02-18-2020“  
durch einen StringTokenizer token = new StringTokenizer („;:{ }“) oder  
der .split(“[ { }; : ]“)-Methode über den Tag „**name**“, „**weight**“ und „**birthdate**“  
filtern.

Wichtig war hier die Benutzung der Methode **.trim()**, die die Leerzeichen entfernt.

Anschließend sollte ein **Baby** konstruiert werden und **returnt** werden mit den gefilterten Variablen. Also: Baby baby = new Baby (name, weight, birthdate);

1. Schreiben Sie eine Java-Applikation, welche die folgende Problemstellungen löst und auf dem Bildschirm ausgibt. Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten Teilaufgabe fortfahren.

1. Lesen Sie vom Benutzer zwei beliebige Strings  $S_1$  und  $S_2$  ein.
  2. Bestimmen Sie die Länge  $n$  von  $S_1$  und  $m$  von  $S_2$ .
  3. Legen Sie eine Matrix *Needleman* mit  $(n + 1)$  Zeilen und  $(m + 1)$  Spalten an.
  4. Schreiben Sie eine Methode  $\text{int cost(char } a, \text{char } b\text{)}$ , die  $-1$  zurückgibt, wenn  $a \neq b$  ist und  $1$  zurückgibt wenn  $a = b$  ist.
  5. Schreiben Sie eine Methode  $\text{int max(int } a, \text{int } b, \text{int } c\text{)}$ , die das Maximum dreier Zahlen bestimmt.
  6. Schreiben Sie eine Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$ , die eine Matrix auf dem Bildschirm ausgibt.
  7. Initialisieren Sie
    - (a) die Matrixposition  $\text{Needleman}(0, 0)$  mit  $0$ .
    - (b)  $\text{Needleman}(i, 0)$  mit  $-i$  für  $i = 1 \dots n$  mit  $i$ .
    - (c)  $\text{Needleman}(0, j)$  mit  $-j$  für  $j = 1 \dots m$  mit  $j$ .
  8. Berechnen Sie alle anderen Matrixpositionen  $\text{Needleman}(i, j)$  wie folgt
- $$\text{Needleman}(i, j) = \max \begin{cases} \text{Needleman}(i - 1, j) - 1 \\ \text{Needleman}(i, j - 1) - 1 \\ \text{Needleman}(i - 1, j - 1) + \text{cost}(S_1[i - 1], S_2[j - 1]) \end{cases}$$
9. Geben Sie den Wert  $\text{Needleman}(n, m)$  aus.
  10. Geben Sie die Matrix *Needleman* aus.  
(für eine 4,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  11. Erweitern Sie die Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$  derart, dass nun auch das  $i$ -te Zeichen von  $S_1$  vor der  $i$ -ten Zeile und das  $j$ -te Zeichen von  $S_2$  vor der  $j$ -ten Spalte steht.  
(für eine 3,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  12. Bestimmen Sie einen möglichen maximalen Pfad von  $\text{Needleman}(0, 0)$  nach  $\text{Needleman}(n, m)$ . Der Pfad soll dabei jeweils der Maximumauswahl des 8. Schrittes folgen. Orientieren Sie sich dabei an folgenden Tipps.
    - (a) Fangen Sie am Ende der Matrix an und bestimmen Sie das Maximum der davon links, darüber und diagonal links darüber liegenden Zelle.
    - (b) Das Maximum liegt auf dem Pfad und kann nun als neuer Ausgangspunkt verwendet werden.
    - (c) Suchen Sie solange neue Maxima bis Sie  $\text{Needleman}(0, 0)$  als Maximum identifiziert haben.
    - (d) Speichern Sie den Pfad in einer Matrix der gleichen Größe wie *Needleman*. Verwenden Sie String als Datentyp der Matrix. Alle Positionen sind von Anfang an auf ein Leerzeichen initialisiert. Die jeweils gefunden Maxima werden in dieser Matrix an der entsprechenden Position gespeichert.

(für eine 1,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)

0	-1	-2	-3	-4
-1	1	0	-1	-2
-2	0	0	-1	0
-3	-1	1	0	-1
-4	-2	0	2	1
-5	-3	-1	1	3

Ausgabe 10.)

	A	G	T	C
0	-1	-2	-3	-4
A	-1	1	0	-1
C	-2	0	0	-1
G	-3	-1	1	0
T	-4	-2	0	2
C	-5	-3	-1	1

Ausgabe 11.)

	A	G	T	C
0				
A				1
C				0
G				1
T				2
C				3

Ausgabe 12.)

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013

Bearbeitungszeit: 120 Minuten

Name	<hr/>
Matr.Nr	<hr/>
Punkte Gesamt	<hr/> von 40
Punkte Note	<hr/>

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) **4.** Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) **5.** Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

## Aufgabe 1

Datentypen

## Aufgabe 2

Casting von Datentypen

## Aufgabe 3

- a) Quellcode lesen

```
Public static int f(int[] myArray, int parameter){
```

```
    Int value=0;
```

```
    Switch(parameter)
```

```
        Case 1
```

```
            For(int x:myArray){
```

```
                Value+=x*x;
```

```
}
```

```
        Case 0
```

```
            For(int x:myArray){
```

```
                Value+=x;
```

```
}
```

```
        Return value;
```

```
}
```

- b) Code aus 3 ohne switch und for Schleife analog implementieren

## Aufgabe 4

Arrays

Schreibe static boolean g(int[] myArray), so dass true herausgegeben wird, wenn mind. ein Eintrag von myArray genau 2 mal vorkommt, sonst false.

## Aufgabe 5

Rekursion mit String

Schreibe Methode g(String s) so, dass

- $3 \cdot g(t)$ , wenn  $s = "a" + t$
- $5 \cdot g(t)$ , wenn  $s = "b" + t + "c"$
- $s.length()$ , sonst [s könnte auch „ sein]

## Aufgabe 6

Finde heraus ob ein String s eine Aneinanderkettung zweier identischer Strings t ist ( $s.equals(t+t)$ ). Falls ja, gib t an, ansonsten gib die Nullreferenz zurück.

### **Aufgabe 7**

Collections, Klassen. Klassen Student und Professor. Professor hat int officeHours, String domain, String namen. Student hat String domain, namen und int time. Es geht darum, dass eine Sprechstunde beim Prof abgebildet werden soll. Dazu werden der Reihe nach in der Main Studenten erzeugt, die eine gewisse Zeit time für ein Gespräch mit dem Prof brauchen. Dies wird aber nur gemacht, wenn die Domains von Prof und Student übereinstimmen, falls nicht, wird ein Text der Art: „Professor „+name+“ ist nicht zuständig für“ student.name.

Ist die Domain identisch, dann reduziert sich die verfügbare Zeit officeHours des Professors um die Zeit, die der entsprechende Student benötigt. Solange der Professor noch Zeit hat, wird der nächste Student zur Sprechstunde zugelassen und es wird ausgegeben „Professor „+name+“ hilft gerade “ +student.name.

Hat der Prof keine Zeit mehr Ausgabe „Leider keine Zeit mehr“

### **Aufgabe 8**

Exceptions, Klassen

Int isbn, String title, String abstract in der Form „123;asdas;opio“ übergeben. Exceptions, wenn Part1 kein Int ist, oder einer der durch „;“ getrennten tokens leer, oder null übergeben wird,...

### **Aufgabe 9**

Vererbung Kamera

Abstract Klasse Kamera, Vererbung auf Klassen Smartphone, und Spiegelreflex

### **Aufgabe 10**

Klasse List analog Vorlesung.

# Aufgaben Programmierung I

## Wintersemester 2007/2008

Schreiben Sie eine Java-Applikation, die die folgende Problemstellung löst und auf dem Bildschirm ausgibt. Es soll eine Klassenstruktur für eine Mediathek entwickelt werden, die es erlaubt Medien wie DVDs, CDs etc. zu erfassen. **Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten beginnen.**

Aufgabe 41: Medium

Schreiben Sie eine abstrakte Klasse `Medium`, die für jedes Objekt eine eindeutige Identifikationsnummer und einen Titel speichert. Die Identifikationsnummern sollen dabei iterativ generiert werden (d.h. das erste erzeugte Objekt soll die Nummer 1 bekommen, das zweite die 2, usw.). Darüber hinaus sollen Methoden `boolean istVerliehen()` und `setVerliehen(boolean)` bereitgestellt werden.

Aufgabe 42: CD/DVD

Leiten Sie von der Klasse `Medium` eine Klasse `DVD` ab, die zusätzlich den Regisseur und die Spielzeit der DVD speichert. Schreiben Sie die entsprechenden `get`-Methoden. Das Setzen der Klassenvariablen sollte hingegen nur im Konstruktor möglich sein. Analog dazu leiten Sie ebenfalls von `Medium` eine Klasse `CD` ab, die den Künstler und die Spielzeit speichert.

Aufgabe 43: Mediathek I

Eine Mediathek repräsentiert in unserem Falle eine Menge beliebiger Größe von verschiedenen Medien. Dazu werden die Medien in einer doppelt verketteten Liste (unsortiert) verwaltet.

- Schreiben Sie deshalb eine Klasse `MedListElem`, deren Objekte ein Listenelement mit einem Wert vom Typ `Medium` darstellen.
- Schreiben Sie eine Klasse `Mediathek`, die eine Methode zum einfachen Einfügen am Ende `void add(Medium m)`, eine zum Löschen eines Mediums aus der Liste `void delete(Medium m)` und eine zum Auffinden des entsprechenden Listenelements `MedListElem lookUp(Medium m)` bereitstellt. Die `lookUp`-Methode soll dabei außerhalb der Klasse nicht sichtbar sein. Des Weiteren sollten Sie eine Methode `String toString()` in der Klasse `Medienliste` schreiben, die die Ausgabe der Liste als String erlaubt. Die Verwendung einer Implementierung vom Typ `Collection`, d.h. von `ArrayList` u.ä., ist **nicht** zulässig.

#### Aufgabe 44:

#### Mediathek II

Erweitern sie ihre Klasse **Mediathek**, so dass der folgende Sachverhalt modelliert wird:

- Eine Mediathek hat einen Namen, der über den Konstruktor festgelegt wird. Das Auslesen soll mittels `getName()` erfolgen.
- Erweitern Sie die Methode `toString()`, so dass zusätzlich der Name der Mediathek ausgegeben wird.
- Schreiben Sie eine Methode `Mediathek find(String str)` zum Suchen nach Medien anhand des Titels. Der Methode soll ein `String str` übergeben werden und in einer Liste alle Medien zurückliefern, die diesen String `str` im Titel enthalten. Der Rückgabewert soll den Namen der Mediathek in der gesucht wird zusammen mit der gesuchten Zeichenkette `str` tragen.

#### Aufgabe 45:

#### MediaTest

Schreiben Sie eine Klasse **MediaTest** zum Testen der bereits entwickelten Klassen **Medium**, **DVD**, **CD**, **Mediathek**. Legen Sie dazu eine Mediathek an und ordnen Sie dieser eine beliebige Anzahl an Medien zu, sowohl CDs als auch DVDs. Geben Sie die eingegebenen Objekte der Mediathek aus. Durchsuchen Sie die Mediathek nach einem von Ihnen festgelegten Suchterminus und geben die erhaltenen Medien iterativ aus.

#### Aufgabe 46:

#### Mediathek III

Schreiben Sie in der Klasse **Mediathek** eine Methode `boolean ausleihen(Medium m)`, die eine Ausnahme `NoSuchElementException` wirft, falls das Medium `m` nicht in der Mediathek enthalten ist. Die Methode liefert `true` zurück, falls das Medium erfolgreich ausgeliehen werden konnte. Analog dazu soll eine Methode `boolean zurueckgeben(Medium m)` bereitgestellt werden, die ebenfalls eine Ausnahme `NoSuchElementException` wirft.

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013

Bearbeitungszeit: 120 Minuten

Name	<hr/>
Matr.Nr	<hr/>
Punkte Gesamt	<hr/> von 40
Punkte Note	<hr/>

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) **4.** Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) **5.** Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

# Praktische Informatik (15.07.2011)

## Info 1:

1.

Ableiten einer vergebenen Klasse

```
class Document{  
    int id;  
    public Document(int id){  
        this.id=id;  
    }  
}
```

Einmal ein Buch welches zusätzlich Seitenzahlen hat und einmal eine Klasse Zeitschrift mit zusätzlich einer Nummer. Dabei soll man im Sinne der Datenkapslung arbeiten.

Außerdem soll ein interface Bibliothek implementiert werden, welches eine Liste von Dokumenten speichern und in Eingabe Reihenfolge ausgeben kann.

2.

Erstellen einer Methode double kreiszahl(double delta) zum berechnen der Kreiszahl Pi. Dabei war die Funktion angeben. Man sollte solange iterieren bis die Werte sich nur noch um delta verändern.

3.

Zwei Klassen: abstract A, mit einem static int counter und B extends A, welche einen counter bzw. verändern setzen. In einem kleinen Aufruf der Klassen sollte man bestimmen welchen Wert die Ausgabe hat.

Dann noch ein kleiner 4 Zeilen Beispiel-Code bei dem man Fehler suchen, weswegen diese nicht kompiliert werden.

## Info 2:

Eigentlich wie immer:

- Quicksort mit Laufzeit im schlechtesten und besten Fall
- Heapsort mit Laufzeit (ohne sink)
- Einen O( $n + k$ ) Sortieralgorithmus
- Topsort mit Laufzeit
- DFS mit compnum und dfsnum
- löschen eines Knoten im Knoten-Suchbaum
- aus einem sortierten Feld eine Blatt-Orientierten Suchbaum erstellen
- Topsort mit Laufzeit

## Gedankenprotokoll Nachklausur Programmierung 1 am 10.04.2018

### Aufgabe 1 + 2

Datentypen (Ergebnis, Datentyp d. Ergebnis, Erklärung)

1.  $1 / 2.0 * 3$
  2. "12"+10
  3. int 1.999
  4. double x = (int) 1.999
  5. double int = 2.0
- ...

### Aufgabe 3

a) Code schriftlich erklären.

```
static int funktion (int [] a) {  
    int summe = 0;  
    for (int x : a) {  
        summe = summe + x;  
    }  
    return summe;  
}
```

b) gleichen Code nochmal implementieren, aber nicht in einer for-Schleife und ohne auf den Originalcode zuzugreifen

### Aufgabe 4

Methode schreiben, um zu zählen wie oft String t ein Teilstring von String s ist  
z.B. String s = „aabaaba“ und String t = „ab“ dann soll der Wert 2 zurückgegeben werden

### Aufgabe 5

Methode schreiben: deleteMinMax

In einem Feld sollen die Minima und Maxima gefunden werden und anschließend durch Nullen ersetzt werden, z.B. Feld der Größe 5: 1 2 4 1 3 sieht danach so aus: 0 2 0 0 3

### Aufgabe 6

Rekursion mit Strings

$$g(s) = \begin{cases} 0, & \text{falls String } s \text{ leer ist} \\ 2*g(t), & \text{falls } s \text{ aus 'a' + t besteht} \\ 3+g(t), & \text{falls } s \text{ auf 'b' + t + 'c' besteht} \\ s.length(), & \text{in jedem anderen Fall} \end{cases}$$

'a' , 'b' , 'c' sollen ganz normale Buchstaben darstellen

### Aufgabe 7

?

### *Aufgabe 8*

Abstrakte Klasse Prüfung, davon abgeleitete Klassen Klausur und Mündlich schreiben

Prüfung besitzt fach (String) und eine id (int), wobei die id ab 1 hochzählen soll

Klausur besitzt geschrieben (boolean)

Mündlich besitzt zudem ein Datum

### *Aufgabe 9*

Listen: vorgegebenes Programm wie folgt ergänzen

Methode 1 ergänze (String s, String t): String t in der Liste hinter String s angehangen werden

Methode 2 ... ?

anschließende Ausgabe der richtigen Reihenfolge der Liste

### *Aufgabe 10*

File IO mit Exception: 3 gegebene txt.-Dateien mit jeweils normalen Zahlen je Spalte (int) und

Kommentare (gekennzeichnet durch #)

Es soll berechnet werden die Summe der Zahlen in den jeweiligen Dateien. Eine Exception soll geworfen werden, wenn Kommentare in der Datei vorhanden sind (mit Lokalisation)

### *Aufgabe 11*

Collections: Dominospiel realisieren

# Programmierung I Klausur WiSe 2018/2019

## **Aufgabe 1 (schriftlich):**

Man bekommt Rechnungen mit unterschiedlichen Zahltypen gegeben und muss das Ergebnis, den Datentyp des Ergebnisses und eine kurze Begründung dafür abgeben, in dieser Aufgabe mit eher untypischen Typen, z.B. Hexadezimal, Long, Float etc.

## **Aufgabe 2 (schriftlich):**

Man bekommt wie in Aufgabe 1 eine Rechnung/Zuweisung, bestehend aus 2-4 Zahlen verschiedener Typen und muss das Ergebnis angeben. Falls die Rechnung nicht funktionieren würde (also zu einem Error führen würde), muss man eine kurze Begründung angeben, warum das Programm hier abstürzt. Ein paar Beispiele für die Rechnungen:

- 1) int a = 1.999;
- 2) int double = 3.0;
- 3) "12"+10
- 4) int b = 1 / 3.0 \* 2;

## **Aufgabe 3:**

Teil a) Schriftlich, Code auf Papier gegeben, man muss kurz erklären, was der Code macht. Der Code in der Klausur war eine Methode mit einem int[] **feld** und einem int **parameter**. Die Methode hatte eine int **summe** gegeben. Mit dem **parameter** wurde ein switch aufgerufen, falls **parameter == 0**, wurde mit einer for-each-Schleife der jeweilige Feldeintrag zur **summe** addiert, falls **parameter == 1**, dann wurde mit einer for-each-Schleife zuerst der Feldeintrag quadriert und dann zur **summe** addiert. Am Ende der Methode wurde **summe** zurückgegeben. Code:

```
public int methode(int[] feld, int parameter) {  
    int summe = 0;  
    switch (parameter) {  
        case 0: {  
            for(int i : feld) {  
                summe += i;  
            }  
        }  
        case 1: {  
            for(int i : feld) {  
                summe += i*i;  
            }  
        }  
    }  
    return summe;  
}
```

Teil b) Den gleichen Code implementieren, aber ohne for-Schleifen

## **Aufgabe 4:**

Man soll eine Methode schreiben, die einen String **str** einliest, falls dieser String zweimal genau aus dem gleichen String **t** besteht, also **str = t + t**, dann soll String **t** zurückgegeben werden, ansonsten **null**. (Beispiel: "TestTest" soll "Test" zurückliefern, "Auto" soll **null** zurückliefern) Lösung:

```
public static boolean stringTest(String str) {  
    int haelfte = str.length() / 2;  
    String teil1 = str.substring(0, haelfte);  
    String teil2 = str.substring(haelfte, str.length());  
    return teil1.equals(teil2);  
}
```

## **Aufgabe 5:**

Feld mit Integers wird in eine Methode übergeben, falls ein Feldeintrag **genau** zweimal im Feld vorkommt, soll *TRUE* zurückgegeben werden, sonst *FALSE*. Lösung:

```
public static boolean feldTest(int[] feld) {  
    for(int i=0; i < feld.length; i++) {  
        int eintrag = feld[i];  
        int counter = 0;  
        for(int j=0; j < feld.length; j++) {  
            if(feld[j] == feld[i]) counter++;  
        }  
        if(counter == 2) return true;  
    }  
    return false;  
}
```

## **Aufgabe 6:**

Man muss zwei Klassen Professor und Student schreiben. Professor hat einen Namen, Fachbereich und eine int-Variable **zeit** (diese werden im Konstruktor gesetzt, wird von der Evaluation automatisch getan). Jeder Student bekommt im Konstruktor einen Namen, Fachbereich und einen Professor **prof** übergeben. Falls **prof** den gleichen Fachbereich besitzt, wie der gerade konstruierte Student, dann wird dieser Student in eine ArrayList des Typs Student in der Professor-Klasse eingefügt. Man soll dann eine Methode **beratung** schreiben, die die ArrayList durchläuft und einen String der Form „Student xyz wird beraten“ ausgibt, falls der Professor noch **zeit** hat, ansonsten wird der String „Professor xy hat keine Beratungszeit mehr übrig“ ausgegeben.

## **Aufgabe 7:**

Man bekommt Strings der Form “1234;Harry Potter; This is a book“ übergeben, die erste Zahl soll die ISBN-Nummer darstellen, der Mittelteil ist der Name des Buchs und der letzte Teilstring ist ein Kommentar. Für jede Eingabe sollte die Ausgabe wie folgt aussehen (am Beispiel):

isbn:	1234
Name:	Harry Potter
Abstrct:	This is a book

Falls ein Teilstring leer ist oder zu wenige Teilstrings vorhanden sind, soll eine Exception geworfen und gecatched werden, die dann “Invalid input“ ausgibt.

## **Aufgabe 8:**

Man bekommt eine abstrakte Klasse Kamera gegeben, davon abgeleitet sind die Klassen Smartphone und Spiegelreflexkamera. Jede der Klassen hat eine int **bilder**, die von 1 hochzählen soll wenn ein Bild gemacht wird (dafür gibt es Methode **takePicture()**). Zudem hat jede neue Instanz (von Smartphone oder Spiegelreflexkamera) eine individuelle ID.

## **Aufgabe 9:**

Man bekommt eine verkettete Liste **L**, ein Element **x** und eine int **n** gegeben und soll eine Methode schreiben, die **x** an der **n**-ten Stelle der Liste einfügt (darauf achten, dass **n** nicht größer ist als die Liste)

Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage nach Ergebnis, Datentyp,  
Erklärung in eine Tabelle auf Klausurblatt schreiben

Bsp: int m = 1; double n = (int) 1.0;

Aufgabe 2 (6P)

6 Aufgaben: Frage zu Programmschnipseln nach Ergebnis oder wenn es nicht läuft den Grund

**Aufgabe 3**

- a) Programm auf Blatt Papier schriftlich erklären – (6P)
- b) Programm ohne for und ohne switch programmieren (8P)

```
Public static int xx (long feld[], int/long? parameter){
```

```
    Int value = 0;
```

```
    Switch(parameter)
```

```
        Case 0: for (long k : feld) value += v; return value;
```

```
        Case 1: for(long k : feld) value += v*v; return value;
```

```
}
```

**Aufgabe 4 (8P)**

Methode schreiben die einen String übergeben bekommt;  
dieser String s soll untersucht werden, ob er aus zwei gleichen Teilstrings t besteht;  
wenn ja soll der Teilstring t zurückgegeben werden – sonst null;

Bspw: String s = abab hat t = ab;

**Aufgabe 5 (8P)**

Rekursion: mit if und else if und else die Anweisungen auf dem Blatt entsprechend implementieren

## Aufgabe 6 (12P) - Arrays

Es wurde ein Array einer Methode übergeben;

Man sollte schauen, ob irgendeine Zahl im Array genau 2x vorkommt,  
wenn ja diese ausgeben,  
wenn nicht false;

## Aufgabe 7 (15P) – Collections

Es gab: eine Klasse Professoren und eine Klasse Studenten und eine main-Klasse

Klasse Professor:

- String domain (Fachbereich)
  - int officeHours
  - String name
- LinkedList<studenten> list;

Klasse studenten:

- String domain (Fachbereich)
- int time (benötigte Zeit für Beratung)
- String name

Man sollte nun zwei Methoden schreiben:

1. Methode bekommt ein student übergeben  
ist dieser vom selben Fachbereich (domain) wie der Prof;  
so soll er in die Liste vom Prof eingetragen werden  
sonst sollte eine Ausgabe auf Konsole erfolgen (Prof x ist nicht zuständig für Student y)
2. Methode soll die Liste vom Prof abarbeiten  
es sollen die studentnamen ausgegeben werden  
jeder student benötigt seine „time“  
diese wird von „officeHours „ bei jeder Ausgabe vom studenten um studenten.getTime()  
verringert  
Ist officeHours auf 0 so sollen alle restlichen Studenten in der Liste ausgegeben werden;  
mit den Zusatz (in etwa) „ Prof x konnte Student y nicht bearbeiten“  
(alle Ausgaben war nicht auf deutsch sondern auf englisch)

## Aufgabe 8 (15P) Exceptiones

Es wurde ein String an eine Methode übergeben, die geschrieben werden sollte.

Dieser String beinhaltet das Schema „ISBN; titelvonBuch; abstract“

bspw: „123;rotkappe; abstract“

Es konnte passieren, dass der String null ist, nicht alle Angaben enthalten sind;

Oder „;“ 1x fehlt

Der String sollte auf drei String aufgeteilt werden;

Die ISBN sollte in einen int Variable gespeichert werden;

Falls isbn zu lang war oder keine Zahl war sollte eine Exception fliegen und den Wert auf -1 von isbn setzen;

Ansonsten sollte eine Ausgabe wie folgt erfolgen; wichtig war hier die

Bündigkeit(tabellenartig)

ISBN: 125

Titel: TitelBuch

Abstract: Nicht abstract!

## Aufgabe 9 (15P) Vererbung

- Abstracte class camera schreiben:
  - o String model
  - o Int id (soll bei 0 starten ; inklusive )
  - o Int pictures( Anzahl der getätigten Bilder, soll bei 0 starten, inklusive)
  - o Boolean hasObjektiv (ob man Objektiv abschrauben kann)
  - o Abstracte methode void takePicture();
  - o toString methode
- von camera abgeleitet
  - o class Smartphone
    - sollte im Konstruktor name übergeben bekommen; und im Konstruktor (super(name, false) wegen kein Objektiv abnehmbar)
  - o class SRL-Kamera
    - sollte im Konstruktor (model, String Objektivname) erhalten;
    - im Konstruktor true wegen Objektiv
  - o Methode takePicture in Smartphone und SRL-Kamera implementieren; wenn aufgerufen soll sich die Zahl picture erhöhen und ausgeben werden plus mit welcher Kamera ( id und model) und wenn vorhanden Objektiv

### Aufgabe 10 (15P)

Abgewandelte Version von Queue mit Elem; 2 Methoden schreiben;  
es gab keine getter/setter-Methoden bei Elem; es gab nur String value und Elem next;  
methode enqueue, toString(?) waren vorgegeben;

1. Methode: Insert ( int index, String word) – soll an stelle index word einfügen
2. Methode: bekommt einen String übergeben; der aus mehreren Teilen besteht;  
diese Teile sind mittels Leerzeichen voneinander getrennt;  
diese Teile sollen nun in die bestehe Queue hinten angehängt werden;  
Jedes Teil ein neues Elem;

# Programmierung 1 Wintersemester 2017/2018

Bei: Prof. Dr. Norbert Müller

Insgesamt gab es 110 Punkte, wovon 100 erreicht werden mussten.

**Aufgabe 1) (7 Punkte)**

Grundrechenoperationen von verschiedenen Dateitypen. Ergebnis, Dateityp vom Ergebnis, Erklärung.

**Aufgabe 2) (7 Punkte)**

Zuweisungen. Zum Beispiel: int a = 1,2? Was passiert dann? Double d = (int) 1.2;

**Aufgabe 3) (6 + 8 Punkte)**

- a) Algorithmus erklären (auf Papier)
- b) Den selben Algorithmus programmieren aber ohne Switch / For

**Aufgabe 4) (8 Punkte)**

Teilstring testen (Aufgabe von SS17):

„Man bekommt einen String, der aus 3 Teilstrings besteht und so aufgebaut ist:  
String1String2String1. Die Strings sind alle gleich lang. Das Programm soll prüfen ob der erste  
Teilstring gleich dem dritten Teilstring ist, und wenn das der Fall ist, den mittleren  
zurückgeben. Als Beispiel: Der String lautet "AutoBuchAuto". Es soll "Buch" zurück gegeben  
werden.“

**Aufgabe 5) (8 Punkte)**

Integer Array mit 5 Zahlen. Alle positiven Zahlen sollten die Position tauschen.

Also: [1, -1, 2, 4, -3, 5] => [5, -1, 4, 2, 1]

**Aufgabe 6) () Rekursion**

Es gab eine Methode G(int k) die ein Integer zurückgibt, je nach Wert von k sollte ein  
bestimmter Wert zurückgegeben werden, der teilweise noch einmal G aufruft.

**Aufgabe 7) ()**

**Aufgabe 8) (12 Punkte)**

Abstrakte Klassen und Vererbung

**Aufgabe 9) (12 Punkte)**

Verkettete Listen mit Elem (siehe VL).

**Aufgabe 10) (12 Punkte)**

File I/O

1. Schreiben Sie eine Java-Applikation, welche die folgende Problemstellungen löst und auf dem Bildschirm ausgibt. Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten Teilaufgabe fortfahren.

1. Lesen Sie vom Benutzer zwei beliebige Strings  $S_1$  und  $S_2$  ein.
  2. Bestimmen Sie die Länge  $n$  von  $S_1$  und  $m$  von  $S_2$ .
  3. Legen Sie eine Matrix *Needleman* mit  $(n + 1)$  Zeilen und  $(m + 1)$  Spalten an.
  4. Schreiben Sie eine Methode  $\text{int cost(char } a, \text{char } b\text{)}$ , die  $-1$  zurückgibt, wenn  $a \neq b$  ist und  $1$  zurückgibt wenn  $a = b$  ist.
  5. Schreiben Sie eine Methode  $\text{int max(int } a, \text{int } b, \text{int } c\text{)}$ , die das Maximum dreier Zahlen bestimmt.
  6. Schreiben Sie eine Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$ , die eine Matrix auf dem Bildschirm ausgibt.
  7. Initialisieren Sie
    - (a) die Matrixposition  $\text{Needleman}(0, 0)$  mit  $0$ .
    - (b)  $\text{Needleman}(i, 0)$  mit  $-i$  für  $i = 1 \dots n$  mit  $i$ .
    - (c)  $\text{Needleman}(0, j)$  mit  $-j$  für  $j = 1 \dots m$  mit  $j$ .
  8. Berechnen Sie alle anderen Matrixpositionen  $\text{Needleman}(i, j)$  wie folgt
- $$\text{Needleman}(i, j) = \max \begin{cases} \text{Needleman}(i - 1, j) - 1 \\ \text{Needleman}(i, j - 1) - 1 \\ \text{Needleman}(i - 1, j - 1) + \text{cost}(S_1[i - 1], S_2[j - 1]) \end{cases}$$
9. Geben Sie den Wert  $\text{Needleman}(n, m)$  aus.
  10. Geben Sie die Matrix *Needleman* aus.  
(für eine 4,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  11. Erweitern Sie die Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$  derart, dass nun auch das  $i$ -te Zeichen von  $S_1$  vor der  $i$ -ten Zeile und das  $j$ -te Zeichen von  $S_2$  vor der  $j$ -ten Spalte steht.  
(für eine 3,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  12. Bestimmen Sie einen möglichen maximalen Pfad von  $\text{Needleman}(0, 0)$  nach  $\text{Needleman}(n, m)$ . Der Pfad soll dabei jeweils der Maximumauswahl des 8. Schrittes folgen. Orientieren Sie sich dabei an folgenden Tipps.
    - (a) Fangen Sie am Ende der Matrix an und bestimmen Sie das Maximum der davon links, darüber und diagonal links darüber liegenden Zelle.
    - (b) Das Maximum liegt auf dem Pfad und kann nun als neuer Ausgangspunkt verwendet werden.
    - (c) Suchen Sie solange neue Maxima bis Sie  $\text{Needleman}(0, 0)$  als Maximum identifiziert haben.
    - (d) Speichern Sie den Pfad in einer Matrix der gleichen Größe wie *Needleman*. Verwenden Sie String als Datentyp der Matrix. Alle Positionen sind von Anfang an auf ein Leerzeichen initialisiert. Die jeweils gefunden Maxima werden in dieser Matrix an der entsprechenden Position gespeichert.

(für eine 1,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)

0	-1	-2	-3	-4
-1	1	0	-1	-2
-2	0	0	-1	0
-3	-1	1	0	-1
-4	-2	0	2	1
-5	-3	-1	1	3

Ausgabe 10.)

	A	G	T	C
0	-1	-2	-3	-4
A	-1	1	0	-1
C	-2	0	0	-1
G	-3	-1	1	0
T	-4	-2	0	2
C	-5	-3	-1	1

Ausgabe 11.)

	A	G	T	C
0				
A				1
C				0
G				1
T				2
C				3

Ausgabe 12.)

# Aufgaben Programmierung I

## Wintersemester 2007/2008

Schreiben Sie eine Java-Applikation, die die folgende Problemstellung löst und auf dem Bildschirm ausgibt. Es soll eine Klassenstruktur für eine Mediathek entwickelt werden, die es erlaubt Medien wie DVDs, CDs etc. zu erfassen. **Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten beginnen.**

Aufgabe 41: Medium

Schreiben Sie eine abstrakte Klasse `Medium`, die für jedes Objekt eine eindeutige Identifikationsnummer und einen Titel speichert. Die Identifikationsnummern sollen dabei iterativ generiert werden (d.h. das erste erzeugte Objekt soll die Nummer 1 bekommen, das zweite die 2, usw.). Darüber hinaus sollen Methoden `boolean istVerliehen()` und `setVerliehen(boolean)` bereitgestellt werden.

Aufgabe 42: CD/DVD

Leiten Sie von der Klasse `Medium` eine Klasse `DVD` ab, die zusätzlich den Regisseur und die Spielzeit der DVD speichert. Schreiben Sie die entsprechenden `get`-Methoden. Das Setzen der Klassenvariablen sollte hingegen nur im Konstruktor möglich sein. Analog dazu leiten Sie ebenfalls von `Medium` eine Klasse `CD` ab, die den Künstler und die Spielzeit speichert.

Aufgabe 43: Mediathek I

Eine Mediathek repräsentiert in unserem Falle eine Menge beliebiger Größe von verschiedenen Medien. Dazu werden die Medien in einer doppelt verketteten Liste (unsortiert) verwaltet.

- Schreiben Sie deshalb eine Klasse `MedListElem`, deren Objekte ein Listenelement mit einem Wert vom Typ `Medium` darstellen.
- Schreiben Sie eine Klasse `Mediathek`, die eine Methode zum einfachen Einfügen am Ende `void add(Medium m)`, eine zum Löschen eines Mediums aus der Liste `void delete(Medium m)` und eine zum Auffinden des entsprechenden Listenelements `MedListElem lookUp(Medium m)` bereitstellt. Die `lookUp`-Methode soll dabei außerhalb der Klasse nicht sichtbar sein. Des Weiteren sollten Sie eine Methode `String toString()` in der Klasse `Medienliste` schreiben, die die Ausgabe der Liste als String erlaubt. Die Verwendung einer Implementierung vom Typ `Collection`, d.h. von `ArrayList` u.ä., ist **nicht** zulässig.

#### Aufgabe 44:

#### Mediathek II

Erweitern sie ihre Klasse **Mediathek**, so dass der folgende Sachverhalt modelliert wird:

- Eine Mediathek hat einen Namen, der über den Konstruktor festgelegt wird. Das Auslesen soll mittels `getName()` erfolgen.
- Erweitern Sie die Methode `toString()`, so dass zusätzlich der Name der Mediathek ausgegeben wird.
- Schreiben Sie eine Methode `Mediathek find(String str)` zum Suchen nach Medien anhand des Titels. Der Methode soll ein `String str` übergeben werden und in einer Liste alle Medien zurückliefern, die diesen String `str` im Titel enthalten. Der Rückgabewert soll den Namen der Mediathek in der gesucht wird zusammen mit der gesuchten Zeichenkette `str` tragen.

#### Aufgabe 45:

#### MediaTest

Schreiben Sie eine Klasse **MediaTest** zum Testen der bereits entwickelten Klassen **Medium**, **DVD**, **CD**, **Mediathek**. Legen Sie dazu eine Mediathek an und ordnen Sie dieser eine beliebige Anzahl an Medien zu, sowohl CDs als auch DVDs. Geben Sie die eingegebenen Objekte der Mediathek aus. Durchsuchen Sie die Mediathek nach einem von Ihnen festgelegten Suchterminus und geben die erhaltenen Medien iterativ aus.

#### Aufgabe 46:

#### Mediathek III

Schreiben Sie in der Klasse **Mediathek** eine Methode `boolean ausleihen(Medium m)`, die eine Ausnahme `NoSuchElementException` wirft, falls das Medium `m` nicht in der Mediathek enthalten ist. Die Methode liefert `true` zurück, falls das Medium erfolgreich ausgeliehen werden konnte. Analog dazu soll eine Methode `boolean zurueckgeben(Medium m)` bereitgestellt werden, die ebenfalls eine Ausnahme `NoSuchElementException` wirft.

1. Schreiben Sie eine Java-Applikation, welche die folgende Problemstellungen löst und auf dem Bildschirm ausgibt. Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten Teilaufgabe fortfahren.

1. Lesen Sie vom Benutzer zwei beliebige Strings  $S_1$  und  $S_2$  ein.
  2. Bestimmen Sie die Länge  $n$  von  $S_1$  und  $m$  von  $S_2$ .
  3. Legen Sie eine Matrix *Needleman* mit  $(n + 1)$  Zeilen und  $(m + 1)$  Spalten an.
  4. Schreiben Sie eine Methode  $\text{int cost(char } a, \text{char } b\text{)}$ , die  $-1$  zurückgibt, wenn  $a \neq b$  ist und  $1$  zurückgibt wenn  $a = b$  ist.
  5. Schreiben Sie eine Methode  $\text{int max(int } a, \text{int } b, \text{int } c\text{)}$ , die das Maximum dreier Zahlen bestimmt.
  6. Schreiben Sie eine Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$ , die eine Matrix auf dem Bildschirm ausgibt.
  7. Initialisieren Sie
    - (a) die Matrixposition  $\text{Needleman}(0, 0)$  mit  $0$ .
    - (b)  $\text{Needleman}(i, 0)$  mit  $-i$  für  $i = 1 \dots n$  mit  $i$ .
    - (c)  $\text{Needleman}(0, j)$  mit  $-j$  für  $j = 1 \dots m$  mit  $j$ .
  8. Berechnen Sie alle anderen Matrixpositionen  $\text{Needleman}(i, j)$  wie folgt
- $$\text{Needleman}(i, j) = \max \begin{cases} \text{Needleman}(i - 1, j) - 1 \\ \text{Needleman}(i, j - 1) - 1 \\ \text{Needleman}(i - 1, j - 1) + \text{cost}(S_1[i - 1], S_2[j - 1]) \end{cases}$$
9. Geben Sie den Wert  $\text{Needleman}(n, m)$  aus.
  10. Geben Sie die Matrix *Needleman* aus.  
(für eine 4,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  11. Erweitern Sie die Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$  derart, dass nun auch das  $i$ -te Zeichen von  $S_1$  vor der  $i$ -ten Zeile und das  $j$ -te Zeichen von  $S_2$  vor der  $j$ -ten Spalte steht.  
(für eine 3,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  12. Bestimmen Sie einen möglichen maximalen Pfad von  $\text{Needleman}(0, 0)$  nach  $\text{Needleman}(n, m)$ . Der Pfad soll dabei jeweils der Maximumauswahl des 8. Schrittes folgen. Orientieren Sie sich dabei an folgenden Tipps.
    - (a) Fangen Sie am Ende der Matrix an und bestimmen Sie das Maximum der davon links, darüber und diagonal links darüber liegenden Zelle.
    - (b) Das Maximum liegt auf dem Pfad und kann nun als neuer Ausgangspunkt verwendet werden.
    - (c) Suchen Sie solange neue Maxima bis Sie  $\text{Needleman}(0, 0)$  als Maximum identifiziert haben.
    - (d) Speichern Sie den Pfad in einer Matrix der gleichen Größe wie *Needleman*. Verwenden Sie String als Datentyp der Matrix. Alle Positionen sind von Anfang an auf ein Leerzeichen initialisiert. Die jeweils gefunden Maxima werden in dieser Matrix an der entsprechenden Position gespeichert.

(für eine 1,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)

0	-1	-2	-3	-4
-1	1	0	-1	-2
-2	0	0	-1	0
-3	-1	1	0	-1
-4	-2	0	2	1
-5	-3	-1	1	3

Ausgabe 10.)

	A	G	T	C
0	-1	-2	-3	-4
A	-1	1	0	-1
C	-2	0	0	-1
G	-3	-1	1	0
T	-4	-2	0	2
C	-5	-3	-1	1

Ausgabe 11.)

	A	G	T	C
0				
A				1
C				0
G				1
T				2
C				3

Ausgabe 12.)

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013

Bearbeitungszeit: 120 Minuten

Name	<hr/>
Matr.Nr	<hr/>
Punkte Gesamt	<hr/> von 40
Punkte Note	<hr/>

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) 4. Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) 5. Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

## Aufgabe 1

Datentypen

## Aufgabe 2

Casting von Datentypen

## Aufgabe 3

- a) Quellcode lesen

```
Public static int f(int[] myArray, int parameter){
```

```
    Int value=0;
```

```
    Switch(parameter)
```

```
        Case 1
```

```
            For(int x:myArray){
```

```
                Value+=x*x;
```

```
}
```

```
        Case 0
```

```
            For(int x:myArray){
```

```
                Value+=x;
```

```
}
```

```
        Return value;
```

```
}
```

- b) Code aus 3 ohne switch und for Schleife analog implementieren

## Aufgabe 4

Arrays

Schreibe static boolean g(int[] myArray), so dass true herausgegeben wird, wenn mind. ein Eintrag von myArray genau 2 mal vorkommt, sonst false.

## Aufgabe 5

Rekursion mit String

Schreibe Methode g(String s) so, dass

- $3 \cdot g(t)$ , wenn  $s = "a" + t$
- $5 \cdot g(t)$ , wenn  $s = "b" + t + "c"$
- $s.length()$ , sonst [s könnte auch „ sein]

## Aufgabe 6

Finde heraus ob ein String s eine Aneinanderkettung zweier identischer Strings t ist ( $s.equals(t+t)$ ). Falls ja, gib t an, ansonsten gib die Nullreferenz zurück.

### **Aufgabe 7**

Collections, Klassen. Klassen Student und Professor. Professor hat int officeHours, String domain, String namen. Student hat String domain, namen und int time. Es geht darum, dass eine Sprechstunde beim Prof abgebildet werden soll. Dazu werden der Reihe nach in der Main Studenten erzeugt, die eine gewisse Zeit time für ein Gespräch mit dem Prof brauchen. Dies wird aber nur gemacht, wenn die Domains von Prof und Student übereinstimmen, falls nicht, wird ein Text der Art: „Professor „+name+“ ist nicht zuständig für“ student.name.

Ist die Domain identisch, dann reduziert sich die verfügbare Zeit officeHours des Professors um die Zeit, die der entsprechende Student benötigt. Solange der Professor noch Zeit hat, wird der nächste Student zur Sprechstunde zugelassen und es wird ausgegeben „Professor „+name+“ hilft gerade “ +student.name.

Hat der Prof keine Zeit mehr Ausgabe „Leider keine Zeit mehr“

### **Aufgabe 8**

Exceptions, Klassen

Int isbn, String title, String abstract in der Form „123;asdas;opio“ übergeben. Exceptions, wenn Part1 kein Int ist, oder einer der durch „;“ getrennten tokens leer, oder null übergeben wird,...

### **Aufgabe 9**

Vererbung Kamera

Abstract Klasse Kamera, Vererbung auf Klassen Smartphone, und Spiegelreflex

### **Aufgabe 10**

Klasse List analog Vorlesung.

# Aufgaben Programmierung I

## Wintersemester 2007/2008

Schreiben Sie eine Java-Applikation, die die folgende Problemstellung löst und auf dem Bildschirm ausgibt. Es soll eine Klassenstruktur für eine Mediathek entwickelt werden, die es erlaubt Medien wie DVDs, CDs etc. zu erfassen. **Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten beginnen.**

Aufgabe 41: Medium

Schreiben Sie eine abstrakte Klasse `Medium`, die für jedes Objekt eine eindeutige Identifikationsnummer und einen Titel speichert. Die Identifikationsnummern sollen dabei iterativ generiert werden (d.h. das erste erzeugte Objekt soll die Nummer 1 bekommen, das zweite die 2, usw.). Darüber hinaus sollen Methoden `boolean istVerliehen()` und `setVerliehen(boolean)` bereitgestellt werden.

Aufgabe 42: CD/DVD

Leiten Sie von der Klasse `Medium` eine Klasse `DVD` ab, die zusätzlich den Regisseur und die Spielzeit der DVD speichert. Schreiben Sie die entsprechenden `get`-Methoden. Das Setzen der Klassenvariablen sollte hingegen nur im Konstruktor möglich sein. Analog dazu leiten Sie ebenfalls von `Medium` eine Klasse `CD` ab, die den Künstler und die Spielzeit speichert.

Aufgabe 43: Mediathek I

Eine Mediathek repräsentiert in unserem Falle eine Menge beliebiger Größe von verschiedenen Medien. Dazu werden die Medien in einer doppelt verketteten Liste (unsortiert) verwaltet.

- Schreiben Sie deshalb eine Klasse `MedListElem`, deren Objekte ein Listenelement mit einem Wert vom Typ `Medium` darstellen.
- Schreiben Sie eine Klasse `Mediathek`, die eine Methode zum einfachen Einfügen am Ende `void add(Medium m)`, eine zum Löschen eines Mediums aus der Liste `void delete(Medium m)` und eine zum Auffinden des entsprechenden Listenelements `MedListElem lookUp(Medium m)` bereitstellt. Die `lookUp`-Methode soll dabei außerhalb der Klasse nicht sichtbar sein. Des Weiteren sollten Sie eine Methode `String toString()` in der Klasse `Medienliste` schreiben, die die Ausgabe der Liste als String erlaubt. Die Verwendung einer Implementierung vom Typ `Collection`, d.h. von `ArrayList` u.ä., ist **nicht** zulässig.

#### Aufgabe 44:

#### Mediathek II

Erweitern sie ihre Klasse **Mediathek**, so dass der folgende Sachverhalt modelliert wird:

- Eine Mediathek hat einen Namen, der über den Konstruktor festgelegt wird. Das Auslesen soll mittels `getName()` erfolgen.
- Erweitern Sie die Methode `toString()`, so dass zusätzlich der Name der Mediathek ausgegeben wird.
- Schreiben Sie eine Methode `Mediathek find(String str)` zum Suchen nach Medien anhand des Titels. Der Methode soll ein `String str` übergeben werden und in einer Liste alle Medien zurückliefern, die diesen String `str` im Titel enthalten. Der Rückgabewert soll den Namen der Mediathek in der gesucht wird zusammen mit der gesuchten Zeichenkette `str` tragen.

#### Aufgabe 45:

#### MediaTest

Schreiben Sie eine Klasse **MediaTest** zum Testen der bereits entwickelten Klassen **Medium**, **DVD**, **CD**, **Mediathek**. Legen Sie dazu eine Mediathek an und ordnen Sie dieser eine beliebige Anzahl an Medien zu, sowohl CDs als auch DVDs. Geben Sie die eingegebenen Objekte der Mediathek aus. Durchsuchen Sie die Mediathek nach einem von Ihnen festgelegten Suchterminus und geben die erhaltenen Medien iterativ aus.

#### Aufgabe 46:

#### Mediathek III

Schreiben Sie in der Klasse **Mediathek** eine Methode `boolean ausleihen(Medium m)`, die eine Ausnahme `NoSuchElementException` wirft, falls das Medium `m` nicht in der Mediathek enthalten ist. Die Methode liefert `true` zurück, falls das Medium erfolgreich ausgeliehen werden konnte. Analog dazu soll eine Methode `boolean zurueckgeben(Medium m)` bereitgestellt werden, die ebenfalls eine Ausnahme `NoSuchElementException` wirft.

# Klausur Programmierung I

Universität Trier  
Fachbereich IV-Informatik

Wintersemester 2012/2013

**Prof. Dr. Bernd Walter und Peter Birke**

14.02.2013

Bearbeitungszeit: 120 Minuten

Name	_____
Matr.Nr	_____
Punkte Gesamt	_____ von 40
Punkte Note	_____

**Beachten Sie die Rückseite**

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den \_\_\_\_\_

Unterschrift: \_\_\_\_\_

**Auszufüllen von Studierenden mit Studienziel Bachelor oder Master**

Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja:

Nein:

**Regeln**

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

**Programmierung I (40 Punkte sind zu erreichen)**

- (4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen  $x$  und  $y$  ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

- (6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
    private int id;
    private double balance;
    private String customer;

    public Account(int id, double balance, String customer) {
        this.id = id;
        this.balance = balance;
        this.customer = customer;
    }

    public final void deposit(double d) {
        if (d > 0) this.balance += d;
    }

    public void withdraw(double d) {
        if (d > 0) this.balance -= d;
    }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id`, `balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen. Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

- (7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.
- (a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.
- (b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

- (12 Pkt.) 4. Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

- (a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.
- (b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

- (11 Pkt.) 5. Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
    private int[] internalArray;
    private int size;

    public UArray(int initCapacity) {
        internalArray = new int[initCapacity];
        size = 0;
    }

    void ensureCapacity() {
        // TODO Implementieren
    }

    public void add(int arg) {
        ensureCapacity();
        internalArray[size++] = arg;
    }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

- (a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.
- (b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während
- (c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.
- (d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.
- (e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode `-1` zurückgeben.

# Klausur Programmierung I

## Norbert Müller

### Wintersemester 2020/2021

#### Aufgabe 1:

man sollte beurteilen, ob die Syntax stimmt, wenn ja sollte man den Wert angeben der dabei rauskam, wenn nein, sollte man erklären, wo der Fehler liegt

- 1) boolean b = !(4<2);
- 2) int Int = 42;
- 3) Integer int = 42;
- 4) double d = (int) 42;

die letzten beiden weiß ich leider nicht mehr

#### Aufgabe 2:

Auswertungen von Funktionen

```
long eins = 1; int zwei = 2; float drei = 3.0; double vier = 4.0;  
eins / zwei + eins / vier * drei
```

man sollte Klammern setzen, um zu verdeutlichen, in welcher Reihenfolge das ausgewertet wird und auch angeben welchen Typ das Ergebnis hat wenn man sich die 4 Operatoren unabhängig betrachtet (also welchen Typ hat das Ergebnis, wenn man eins / zwei rechnet usw.) und man sollte den Wert angeben der dabei rauskommt

#### Aufgabe 3:

- das Verhalten einer vorgegebenen Methode beschreiben
- diese Methode umschreiben, sodass keine while-Schleife mehr vorhanden ist

#### Aufgabe 4:

Strings

man sollte eine Methode schreiben, die zwei Strings als Parameter hat und überprüft, ob der erste zweimal den zweiten String enthält, wenn ja soll das was dahinter steht zurückgegeben werden, wenn nein soll „null“ zurückgegeben werden

z.B. s1 = „xyzxyzABC“ s2 = „xyz“ dann sollte „ABC“ zurückgegeben werden  
s1 = „abab“ s2 = „ab“ dann sollte „“ zurückgegeben werden

#### Aufgabe 5:

Rekursion

es war eine Funktion gegeben

g(x) = „leer“, wenn der übergebene String leer ist  
= s (der String selbst), wenn die Länge des Strings ungerade ist  
= g(s1) + „#“ + g(s2), in allen anderen Fällen

s1 ist dabei die erste Hälfte von dem String und s2 die zweite Hälfte

z.B. s = „ab“ → „a#b“  
s = „abcd“ → „a#b#c#d“  
s = „abcdef“ → „abc#def“

#### Aufgabe 6:

Arrays

man sollte eine Methode schreiben, die zwei Arrays als Parameter hat und überprüft, ob mindestens zwei Zahlen aus dem ersten Array auch in dem zweiten Array vorkommen

wenn ja sollte „true“ zurückgegeben werden, wenn nein „false“

z.B. A = {1,2,3}; B = {4,5,6,7} → false  
A = {1,2,3}; B = {1,2,6} → true

$A = \{\}; B = \{1,2,3,4,5,6\} \rightarrow \text{false}$

### Aufgabe 7:

#### Collections

es war eine Klasse Meeting vorgegeben, date und visitor als Variablen enthält und man sollte eine Klasse „PersonalCalender“ schreiben, die alle Meetings speichert

### Aufgabe 8:

#### Vererbung

man hatte eine Klasse „Song“ vorgegeben und sollte ein Interface „Playlist“ schreiben und dann eine Klasse „SimplePlaylist“, die das Interface implementiert  
in „Playlist“ waren die Methoden: void addSong(Song song), String play(String titel) und noch eine Methode, an die ich mich leider nicht mehr erinner  
bei addSong sollte ein neuer Song in ein Array Song[] songs aufgenommen werden  
bei play sollte der Song mit dem passenden Titel gesucht werden und zurückgegeben werden, falls der Song nicht enthalten ist, sollte „null“ zurückgegeben werden

### Aufgabe 9:

#### Stack

man musste zwei Methoden schreiben, die erste sollte prüfen, ob ein Element in dem Stack enthalten ist  
die zweite Methode sollte ein Element daraus löschen (pop wurde nicht vorgegeben; man musste mit Referenzen arbeiten)

### Aufgabe 10:

#### StringTokenizer

man hatte eine Klasse A(mir fällt der genaue Name nicht mehr ein), wo der Konstruktor 7 Parameter hat  
dann hatte man eine andere Klasse, bei der im Konstruktor ein String übergeben wurde der alle 7 Parameter-Inforationen enthält, die immer durch „:“ getrennt wurden  
man sollte die trennen und dann in den Variablen speichern und damit ein neues Objekt von A erstellen  
falls der String falsch formatiert ist, sollte „falsches Format“ zurückgegeben werden

# Praktische Informatik (15.07.2011)

## Info 1:

1.

Ableiten einer vergebenen Klasse

```
class Document{  
    int id;  
    public Document(int id){  
        this.id=id;  
    }  
}
```

Einmal ein Buch welches zusätzlich Seitenzahlen hat und einmal eine Klasse Zeitschrift mit zusätzlich einer Nummer. Dabei soll man im Sinne der Datenkapslung arbeiten.

Außerdem soll ein interface Bibliothek implementiert werden, welches eine Liste von Dokumenten speichern und in Eingabe Reihenfolge ausgeben kann.

2.

Erstellen einer Methode double kreiszahl(double delta) zum berechnen der Kreiszahl Pi. Dabei war die Funktion angeben. Man sollte solange iterieren bis die Werte sich nur noch um delta verändern.

3.

Zwei Klassen: abstract A, mit einem static int counter und B extends A, welche einen counter bzw. verändern setzen. In einem kleinen Aufruf der Klassen sollte man bestimmen welchen Wert die Ausgabe hat.

Dann noch ein kleiner 4 Zeilen Beispiel-Code bei dem man Fehler suchen, weswegen diese nicht kompiliert werden.

## Info 2:

Eigentlich wie immer:

- Quicksort mit Laufzeit im schlechtesten und besten Fall
- Heapsort mit Laufzeit (ohne sink)
- Einen O( $n + k$ ) Sortieralgorithmus
- Topsort mit Laufzeit
- DFS mit compnum und dfsnum
- löschen eines Knoten im Knoten-Suchbaum
- aus einem sortierten Feld eine Blatt-Orientierten Suchbaum erstellen
- Topsort mit Laufzeit

## Gedankenprotokoll Nachklausur Programmierung 1 am 10.04.2018

### Aufgabe 1 + 2

Datentypen (Ergebnis, Datentyp d. Ergebnis, Erklärung)

1.  $1 / 2.0 * 3$
  2. "12"+10
  3. int 1.999
  4. double x = (int) 1.999
  5. double int = 2.0
- ...

### Aufgabe 3

a) Code schriftlich erklären.

```
static int funktion (int [] a) {  
    int summe = 0;  
    for (int x : a) {  
        summe = summe + x;  
    }  
    return summe;  
}
```

b) gleichen Code nochmal implementieren, aber nicht in einer for-Schleife und ohne auf den Originalcode zuzugreifen

### Aufgabe 4

Methode schreiben, um zu zählen wie oft String t ein Teilstring von String s ist  
z.B. String s = „aabaaba“ und String t = „ab“ dann soll der Wert 2 zurückgegeben werden

### Aufgabe 5

Methode schreiben: deleteMinMax

In einem Feld sollen die Minima und Maxima gefunden werden und anschließend durch Nullen ersetzt werden, z.B. Feld der Größe 5: 1 2 4 1 3 sieht danach so aus: 0 2 0 0 3

### Aufgabe 6

Rekursion mit Strings

$$g(s) = \begin{cases} 0, & \text{falls String } s \text{ leer ist} \\ 2*g(t), & \text{falls } s \text{ aus 'a' + t besteht} \\ 3+g(t), & \text{falls } s \text{ auf 'b' + t + 'c' besteht} \\ s.length(), & \text{in jedem anderen Fall} \end{cases}$$

'a' , 'b' , 'c' sollen ganz normale Buchstaben darstellen

### Aufgabe 7

?

### *Aufgabe 8*

Abstrakte Klasse Prüfung, davon abgeleitete Klassen Klausur und Mündlich schreiben

Prüfung besitzt fach (String) und eine id (int), wobei die id ab 1 hochzählen soll

Klausur besitzt geschrieben (boolean)

Mündlich besitzt zudem ein Datum

### *Aufgabe 9*

Listen: vorgegebenes Programm wie folgt ergänzen

Methode 1 ergänze (String s, String t): String t in der Liste hinter String s angehangen werden

Methode 2 ... ?

anschließende Ausgabe der richtigen Reihenfolge der Liste

### *Aufgabe 10*

File IO mit Exception: 3 gegebene txt.-Dateien mit jeweils normalen Zahlen je Spalte (int) und

Kommentare (gekennzeichnet durch #)

Es soll berechnet werden die Summe der Zahlen in den jeweiligen Dateien. Eine Exception soll geworfen werden, wenn Kommentare in der Datei vorhanden sind (mit Lokalisation)

### *Aufgabe 11*

Collections: Dominospiel realisieren

# Programmierung I Klausur WiSe 2018/2019

## **Aufgabe 1 (schriftlich):**

Man bekommt Rechnungen mit unterschiedlichen Zahltypen gegeben und muss das Ergebnis, den Datentyp des Ergebnisses und eine kurze Begründung dafür abgeben, in dieser Aufgabe mit eher untypischen Typen, z.B. Hexadezimal, Long, Float etc.

## **Aufgabe 2 (schriftlich):**

Man bekommt wie in Aufgabe 1 eine Rechnung/Zuweisung, bestehend aus 2-4 Zahlen verschiedener Typen und muss das Ergebnis angeben. Falls die Rechnung nicht funktionieren würde (also zu einem Error führen würde), muss man eine kurze Begründung angeben, warum das Programm hier abstürzt. Ein paar Beispiele für die Rechnungen:

- 1) int a = 1.999;
- 2) int double = 3.0;
- 3) "12"+10
- 4) int b = 1 / 3.0 \* 2;

## **Aufgabe 3:**

Teil a) Schriftlich, Code auf Papier gegeben, man muss kurz erklären, was der Code macht. Der Code in der Klausur war eine Methode mit einem int[] **feld** und einem int **parameter**. Die Methode hatte eine int **summe** gegeben. Mit dem **parameter** wurde ein switch aufgerufen, falls **parameter == 0**, wurde mit einer for-each-Schleife der jeweilige Feldeintrag zur **summe** addiert, falls **parameter == 1**, dann wurde mit einer for-each-Schleife zuerst der Feldeintrag quadriert und dann zur **summe** addiert. Am Ende der Methode wurde **summe** zurückgegeben. Code:

```
public int methode(int[] feld, int parameter) {  
    int summe = 0;  
    switch (parameter) {  
        case 0: {  
            for(int i : feld) {  
                summe += i;  
            }  
        }  
        case 1: {  
            for(int i : feld) {  
                summe += i*i;  
            }  
        }  
    }  
    return summe;  
}
```

Teil b) Den gleichen Code implementieren, aber ohne for-Schleifen

## **Aufgabe 4:**

Man soll eine Methode schreiben, die einen String **str** einliest, falls dieser String zweimal genau aus dem gleichen String **t** besteht, also **str = t + t**, dann soll String **t** zurückgegeben werden, ansonsten **null**. (Beispiel: "TestTest" soll "Test" zurückliefern, "Auto" soll **null** zurückliefern) Lösung:

```
public static boolean stringTest(String str) {  
    int haelfte = str.length() / 2;  
    String teil1 = str.substring(0, haelfte);  
    String teil2 = str.substring(haelfte, str.length());  
    return teil1.equals(teil2);  
}
```

## **Aufgabe 5:**

Feld mit Integers wird in eine Methode übergeben, falls ein Feldeintrag **genau** zweimal im Feld vorkommt, soll *TRUE* zurückgegeben werden, sonst *FALSE*. Lösung:

```
public static boolean feldTest(int[] feld) {
    for(int i=0; i < feld.length; i++) {
        int eintrag = feld[i];
        int counter = 0;
        for(int j=0; j < feld.length; j++) {
            if(feld[j] == feld[i]) counter++;
        }
        if(counter == 2) return true;
    }
    return false;
}
```

## **Aufgabe 6:**

Man muss zwei Klassen Professor und Student schreiben. Professor hat einen Namen, Fachbereich und eine int-Variable **zeit** (diese werden im Konstruktor gesetzt, wird von der Evaluation automatisch getan). Jeder Student bekommt im Konstruktor einen Namen, Fachbereich und einen Professor **prof** übergeben. Falls **prof** den gleichen Fachbereich besitzt, wie der gerade konstruierte Student, dann wird dieser Student in eine ArrayList des Typs Student in der Professor-Klasse eingefügt. Man soll dann eine Methode **beratung** schreiben, die die ArrayList durchläuft und einen String der Form „Student xyz wird beraten“ ausgibt, falls der Professor noch **zeit** hat, ansonsten wird der String „Professor xy hat keine Beratungszeit mehr übrig“ ausgegeben.

## **Aufgabe 7:**

Man bekommt Strings der Form „1234;Harry Potter; This is a book“ übergeben, die erste Zahl soll die ISBN-Nummer darstellen, der Mittelteil ist der Name des Buchs und der letzte Teilstring ist ein Kommentar. Für jede Eingabe sollte die Ausgabe wie folgt aussehen (am Beispiel):

ISBN:	1234
Name:	Harry Potter
Abstrct:	This is a book

Falls ein Teilstring leer ist oder zu wenige Teilstrings vorhanden sind, soll eine Exception geworfen und gecatched werden, die dann „Invalid input“ ausgibt.

## **Aufgabe 8:**

Man bekommt eine abstrakte Klasse Kamera gegeben, davon abgeleitet sind die Klassen Smartphone und Spiegelreflexkamera. Jede der Klassen hat eine int **bilder**, die von 1 hochzählen soll wenn ein Bild gemacht wird (dafür gibt es Methode **takePicture()**). Zudem hat jede neue Instanz (von Smartphone oder Spiegelreflexkamera) eine individuelle ID.

## **Aufgabe 9:**

Man bekommt eine verkettete Liste **L**, ein Element **x** und eine int **n** gegeben und soll eine Methode schreiben, die **x** an der **n**-ten Stelle der Liste einfügt (darauf achten, dass **n** nicht größer ist als die Liste)

Aufgabe 1 (7P)

7 Aufgaben: Programmschnipsel: Frage nach Ergebnis, Datentyp,  
Erklärung in eine Tabelle auf Klausurblatt schreiben

Bsp: int m = 1; double n = (int) 1.0;

Aufgabe 2 (6P)

6 Aufgaben: Frage zu Programmschnipseln nach Ergebnis oder wenn es nicht läuft den Grund

**Aufgabe 3**

- a) Programm auf Blatt Papier schriftlich erklären – (6P)
- b) Programm ohne for und ohne switch programmieren (8P)

```
Public static int xx (long feld[], int/long? parameter){
```

```
    Int value = 0;
```

```
    Switch(parameter)
```

```
        Case 0: for (long k : feld) value += v; return value;
```

```
        Case 1: for(long k : feld) value += v*v; return value;
```

```
}
```

**Aufgabe 4 (8P)**

Methode schreiben die einen String übergeben bekommt;  
dieser String s soll untersucht werden, ob er aus zwei gleichen Teilstrings t besteht;  
wenn ja soll der Teilstring t zurückgegeben werden – sonst null;

Bspw: String s = abab hat t = ab;

**Aufgabe 5 (8P)**

Rekursion: mit if und else if und else die Anweisungen auf dem Blatt entsprechend implementieren

## Aufgabe 6 (12P) - Arrays

Es wurde ein Array einer Methode übergeben;

Man sollte schauen, ob irgendeine Zahl im Array genau 2x vorkommt,  
wenn ja diese ausgeben,  
wenn nicht false;

## Aufgabe 7 (15P) – Collections

Es gab: eine Klasse Professoren und eine Klasse Studenten und eine main-Klasse

Klasse Professor:

- String domain (Fachbereich)
  - int officeHours
  - String name
- LinkedList<studenten> list;

Klasse studenten:

- String domain (Fachbereich)
- int time (benötigte Zeit für Beratung)
- String name

Man sollte nun zwei Methoden schreiben:

1. Methode bekommt ein student übergeben  
ist dieser vom selben Fachbereich (domain) wie der Prof;  
so soll er in die Liste vom Prof eingetragen werden  
sonst sollte eine Ausgabe auf Konsole erfolgen (Prof x ist nicht zuständig für Student y)
2. Methode soll die Liste vom Prof abarbeiten  
es sollen die studentnamen ausgegeben werden  
jeder student benötigt seine „time“  
diese wird von „officeHours „ bei jeder Ausgabe vom studenten um studenten.getTime()  
verringert  
Ist officeHours auf 0 so sollen alle restlichen Studenten in der Liste ausgegeben werden;  
mit den Zusatz (in etwa) „ Prof x konnte Student y nicht bearbeiten“  
(alle Ausgaben war nicht auf deutsch sondern auf englisch)

## Aufgabe 8 (15P) Exceptiones

Es wurde ein String an eine Methode übergeben, die geschrieben werden sollte.

Dieser String beinhaltet das Schema „ISBN; titelvonBuch; abstract“

bspw: „123;rotkappe; abstract“

Es konnte passieren, dass der String null ist, nicht alle Angaben enthalten sind;

Oder „;“ 1x fehlt

Der String sollte auf drei String aufgeteilt werden;

Die ISBN sollte in einen int Variable gespeichert werden;

Falls isbn zu lang war oder keine Zahl war sollte eine Exception fliegen und den Wert auf -1 von isbn setzen;

Ansonsten sollte eine Ausgabe wie folgt erfolgen; wichtig war hier die

Bündigkeit(tabellenartig)

ISBN: 125

Titel: TitelBuch

Abstract: Nicht abstract!

## Aufgabe 9 (15P) Vererbung

- Abstracte class camera schreiben:
  - o String model
  - o Int id (soll bei 0 starten ; inklusive )
  - o Int pictures( Anzahl der getätigten Bilder, soll bei 0 starten, inklusive)
  - o Boolean hasObjektiv (ob man Objektiv abschrauben kann)
  - o Abstracte methode void takePicture();
  - o toString methode
- von camera abgeleitet
  - o class Smartphone
    - sollte im Konstruktor name übergeben bekommen; und im Konstruktor (super(name, false) wegen kein Objektiv abnehmbar)
  - o class SRL-Kamera
    - sollte im Konstruktor (model, String Objektivname) erhalten;
    - im Konstruktor true wegen Objektiv
  - o Methode takePicture in Smartphone und SRL-Kamera implementieren; wenn aufgerufen soll sich die Zahl picture erhöhen und ausgeben werden plus mit welcher Kamera ( id und model) und wenn vorhanden Objektiv

### Aufgabe 10 (15P)

Abgewandelte Version von Queue mit Elem; 2 Methoden schreiben;  
es gab keine getter/setter-Methoden bei Elem; es gab nur String value und Elem next;  
methode enqueue, toString(?) waren vorgegeben;

1. Methode: Insert ( int index, String word) – soll an stelle index word einfügen
2. Methode: bekommt einen String übergeben; der aus mehreren Teilen besteht;  
diese Teile sind mittels Leerzeichen voneinander getrennt;  
diese Teile sollen nun in die bestehe Queue hinten angehängt werden;  
Jedes Teil ein neues Elem;

# Programmierung 1 Wintersemester 2017/2018

Bei: Prof. Dr. Norbert Müller

Insgesamt gab es 110 Punkte, wovon 100 erreicht werden mussten.

**Aufgabe 1) (7 Punkte)**

Grundrechenoperationen von verschiedenen Dateitypen. Ergebnis, Dateityp vom Ergebnis, Erklärung.

**Aufgabe 2) (7 Punkte)**

Zuweisungen. Zum Beispiel: int a = 1,2? Was passiert dann? Double d = (int) 1.2;

**Aufgabe 3) (6 + 8 Punkte)**

- a) Algorithmus erklären (auf Papier)
- b) Den selben Algorithmus programmieren aber ohne Switch / For

**Aufgabe 4) (8 Punkte)**

Teilstring testen (Aufgabe von SS17):

„Man bekommt einen String, der aus 3 Teilstrings besteht und so aufgebaut ist:  
String1String2String1. Die Strings sind alle gleich lang. Das Programm soll prüfen ob der erste  
Teilstring gleich dem dritten Teilstring ist, und wenn das der Fall ist, den mittleren  
zurückgeben. Als Beispiel: Der String lautet "AutoBuchAuto". Es soll "Buch" zurück gegeben  
werden.“

**Aufgabe 5) (8 Punkte)**

Integer Array mit 5 Zahlen. Alle positiven Zahlen sollten die Position tauschen.

Also: [1, -1, 2, 4, -3, 5] => [5, -1, 4, 2, 1]

**Aufgabe 6) () Rekursion**

Es gab eine Methode G(int k) die ein Integer zurückgibt, je nach Wert von k sollte ein  
bestimmter Wert zurückgegeben werden, der teilweise noch einmal G aufruft.

**Aufgabe 7) ()**

**Aufgabe 8) (12 Punkte)**

Abstrakte Klassen und Vererbung

**Aufgabe 9) (12 Punkte)**

Verkettete Listen mit Elem (siehe VL).

**Aufgabe 10) (12 Punkte)**

File I/O

**Exam was in total 115 ; 15 is bonus.**

**(7 pts) Q1** It was about Data types.

<u>Example</u>	<u>Type</u>	<u>Explanation</u>
31+42		
42.0/10+ 5/4		
3.2d+4.3f		
!(5>6)		
True		

**(6 pts) Q2** It was about again data types and assignment of these data types and if the compiler works or not

e.g.

<u>Example</u>	<u>If the compiler works or not, if not the reason</u>
double r=(int)4;	

**Q3)**

a) Read the given code and explain what it is doing

```
public static boolean f(int[] data){  
    if(data==null) return false;  
    int i=0;  
    while(i<data.length){  
        if(data[i-1]>data[i]) return false;  
        i++;  
    }  
    return true;
```

b) Implement the above code without using while or do while.

**Q4)** You must write a method which return "abc" if "abcabcXY", "XY" (it is an example) like  
string t + t + s return t

```
public static String method(String first, String s){  
}
```

If String first in the form t+t+s and the methods should return t else null

e.g

"xyzxyzABC" must return "xyz"

"xyxyABC" must return "xy"

“11111111” must return “111”

“xyzxyyABC” must return null;

**Q5)** Writing method which return true if a char repeated three or more in an array ( ex return true for array={1 1 1 2 3} )

```
public boolean charRepeated(String[] data){  
}
```

Examples were like

{“1”, “2”, “3”, “4”, “5”, “5” } must return false

{“1”, “1”, “1”, “2”, “3”, “5” } must return true

**Q6)** Arrays

**Q7)** String and StringTokenizer

You are given Strings in the form of (.....;.....;.....;.....)

(String ;String ; String ;String)  
↓

This part must be converted from String to int  
In the question it says that use Integer.parseInt()

Eg. (1234;Marcus Halle; Bla Bla Publishing; Something More)

You need to printout;

ISBN : ..... (integer)

Author: ..... (String)

Publisher: .....(String)

Some other thing: ..... (String)

**(15 pts)Q8**) Interface

**(15 pts)Q9**) Collections

**(15 pts)Q10**) Queue

You have implement a method into Queue.java that add new element to the desired position

```
Public void addElem(Elem new, int pos){  
}
```

1. Schreiben Sie eine Java-Applikation, welche die folgende Problemstellungen löst und auf dem Bildschirm ausgibt. Überprüfen Sie zu Ihrer eigenen Sicherheit, ob eine Teilaufgabe richtig gelöst ist, bevor Sie mit der nächsten Teilaufgabe fortfahren.

1. Lesen Sie vom Benutzer zwei beliebige Strings  $S_1$  und  $S_2$  ein.
  2. Bestimmen Sie die Länge  $n$  von  $S_1$  und  $m$  von  $S_2$ .
  3. Legen Sie eine Matrix *Needleman* mit  $(n + 1)$  Zeilen und  $(m + 1)$  Spalten an.
  4. Schreiben Sie eine Methode  $\text{int cost(char } a, \text{char } b\text{)}$ , die  $-1$  zurückgibt, wenn  $a \neq b$  ist und  $1$  zurückgibt wenn  $a = b$  ist.
  5. Schreiben Sie eine Methode  $\text{int max(int } a, \text{int } b, \text{int } c\text{)}$ , die das Maximum dreier Zahlen bestimmt.
  6. Schreiben Sie eine Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$ , die eine Matrix auf dem Bildschirm ausgibt.
  7. Initialisieren Sie
    - (a) die Matrixposition  $\text{Needleman}(0, 0)$  mit  $0$ .
    - (b)  $\text{Needleman}(i, 0)$  mit  $-i$  für  $i = 1 \dots n$  mit  $i$ .
    - (c)  $\text{Needleman}(0, j)$  mit  $-j$  für  $j = 1 \dots m$  mit  $j$ .
  8. Berechnen Sie alle anderen Matrixpositionen  $\text{Needleman}(i, j)$  wie folgt
- $$\text{Needleman}(i, j) = \max \begin{cases} \text{Needleman}(i - 1, j) - 1 \\ \text{Needleman}(i, j - 1) - 1 \\ \text{Needleman}(i - 1, j - 1) + \text{cost}(S_1[i - 1], S_2[j - 1]) \end{cases}$$
9. Geben Sie den Wert  $\text{Needleman}(n, m)$  aus.
  10. Geben Sie die Matrix *Needleman* aus.  
(für eine 4,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  11. Erweitern Sie die Methode  $\text{String printMatrix(int } [][] \text{ matrix)}$  derart, dass nun auch das  $i$ -te Zeichen von  $S_1$  vor der  $i$ -ten Zeile und das  $j$ -te Zeichen von  $S_2$  vor der  $j$ -ten Spalte steht.  
(für eine 3,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)
  12. Bestimmen Sie einen möglichen maximalen Pfad von  $\text{Needleman}(0, 0)$  nach  $\text{Needleman}(n, m)$ . Der Pfad soll dabei jeweils der Maximumauswahl des 8. Schrittes folgen. Orientieren Sie sich dabei an folgenden Tipps.
    - (a) Fangen Sie am Ende der Matrix an und bestimmen Sie das Maximum der davon links, darüber und diagonal links darüber liegenden Zelle.
    - (b) Das Maximum liegt auf dem Pfad und kann nun als neuer Ausgangspunkt verwendet werden.
    - (c) Suchen Sie solange neue Maxima bis Sie  $\text{Needleman}(0, 0)$  als Maximum identifiziert haben.
    - (d) Speichern Sie den Pfad in einer Matrix der gleichen Größe wie *Needleman*. Verwenden Sie String als Datentyp der Matrix. Alle Positionen sind von Anfang an auf ein Leerzeichen initialisiert. Die jeweils gefunden Maxima werden in dieser Matrix an der entsprechenden Position gespeichert.

(für eine 1,0 muss diese Aufgabe und alle vorherigen Teilaufgaben gelöst werden.)

0	-1	-2	-3	-4
-1	1	0	-1	-2
-2	0	0	-1	0
-3	-1	1	0	-1
-4	-2	0	2	1
-5	-3	-1	1	3

Ausgabe 10.)

	A	G	T	C
0	-1	-2	-3	-4
A	-1	1	0	-1
C	-2	0	0	-1
G	-3	-1	1	0
T	-4	-2	0	2
C	-5	-3	-1	1

Ausgabe 11.)

	A	G	T	C
0				
A				1
C				0
G				1
T				2
C				3

Ausgabe 12.)

# Grundlagen der Programmierung

Norbert Müller

Universität Trier – Fachbereich IV – Informatik

20. Februar 2024

Bearbeitungszeit: **120 Minuten**

Name	_____
Matr.Nr.	_____
Rechner	_____

- Hiermit bestätige ich, dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur zur Vorlesung Grundlagen der Programmierung (Programmierung I) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.
- Ich fühle mich gesundheitlich in der Lage, die Klausur anzutreten.

Ort, Datum: \_\_\_\_\_

Unterschrift: \_\_\_\_\_

## Beachten Sie folgende Regelungen:

- Als Hilfsmittel sind nur die zur Verfügung gestellten Editoren (inkl. Java-Dokumentation) sowie die elektronisch zur Verfügung gestellten Unterlagen (Kurs zur Klausur auf der moodle-Plattform) erlaubt. Andere Hilfsmittel wie z.B. das Einloggen in andere Moodle-Kurse oder die Verwendung von Smartphones sind nicht erlaubt.
- In der Klausur sind **112 Punkte** erreichbar. Ziel ist, dass Sie davon etwa **100 Punkte** bearbeiten. Die Zahl der Punkte pro Aufgabe entspricht also in etwa der Bearbeitungszeit (in Minuten).
- Alle Lösungen sind über das Moodle-System abzugeben, d.h. dort abzuspeichern.
- Programme können i.d.R. unter Moodle auch getestet und vor-evaluierter werden. Die Testfälle werden bei der Korrektur allerdings durch neue Fälle ersetzt werden! Abgaben, die nicht compilierbar sind, führen zu deutlichen Punktabzügen!

Aufgabe	1	2	3	4	5	6	7	8	9	10	Gesamt
Möglich	7	6	14	8	8	12	15	15	12	15	112
Erreicht											

Note: \_\_\_\_\_

## 1. Aufgabe: Primitive Datentypen und Termauswertung (7 P)

Bei folgenden Deklarationen von Variablen

```
float eins = 1.0f; int zwei = 2; long drei = 3L; double vier = 4.0d;
```

soll der folgende Ausdruck ausgewertet werden:

```
drei / zwei - eins / zwei * drei + vier  
(a)   (b)   (c)   (d)   (e)
```

Geben Sie zunächst an, wie dieser Ausdruck (entsprechend der Auswertungsregeln in Java) ausgewertet wird, indem Sie im Ausdruck 5 öffnende und 5 schließende Klammern hinzufügen.

Beispiel: `eins + zwei + drei` würde ausgewertet als `((eins + zwei) + drei)`

Geben Sie zudem an, welchen Typ und welchen Wert die Zwischenresultate der Operationen (a) bis (e) **bei der von Ihnen gewählten Klammerung** an den jeweiligen Positionen haben. Verwenden Sie für die Angabe des Wertes immer Literale des jeweiligen Typs.

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

## 2. Aufgabe: Deklarationen und Casts (6 P)

Betrachten Sie die im folgenden angegebenen kurzen Abschnitte aus (evtl. nicht kompilierbaren) Java-Quelltexten. Geben Sie bei den syntaktisch korrekten Textabschnitten den jeweils zugewiesenen Wert an. Bei syntaktisch nicht korrekten Texten geben Sie bitte an, warum sie nicht korrekt sind.

1. boolean bit = (12 != 34);
2. int value = 1.234;
3. double val = (int) 123.4;
4. int byte = 123;
5. string text = "1234";
6. int array[] = {12,34};

Die Abgabe der Antwort erfolgt bei der entsprechenden Aufgabe in Moodle.

### 3. Aufgabe: Quelltext verstehen und umformulieren

Betrachten Sie die folgende Methode:

```
public static int f( int[] data, int value ) {

    if ( data == null ) throw new IllegalArgumentException();

    int i = data.length - 1;

    while ( i != 0 ) {
        i--;
        if ( data[i] == value ) return 0;
    }
    return 1;
}
```

- (a) Erklären Sie in Worten, bei welchen Kombinationen der Parameter `data` und `value` diese Methode den Wert 1 zurückgibt, wann den Wert 0 und welche anderen Programmverläufe es gibt. Drei oder vier ordentlich formulierte Sätze sollten dazu reichen. Achten Sie auf Sonderfälle! (8 P)

Beschreiben Sie dabei *nicht* den Ablauf des Algorithmus, also nicht die einzelnen Zeilen des Quelltextes! Sie sollen stattdessen nur das beim Aufruf sichtbare Verhalten beschreiben.

- (b) Implementieren Sie den Algorithmus aus `f` analog neu in einer Methode `g`: (6 P)
- Die neue Methode `g` muss für alle(!) möglichen Parameter die gleichen Resultate wie `f` liefern.
  - Dabei darf `g` jedoch *keine* `while`-Anweisung enthalten.

Den Quelltext gibt es in Moodle sowohl unter Aufgabe 03a als auch unter Aufgabe 03b. Ihre Lösung zu Aufgabenteil (a) soll als Freitext in Aufgabe 03a eingetragen werden, die Lösung zu Teil (b) soll unter A3b.java bei Aufgabe 03b in Moodle gespeichert werden; eine zusätzliche Klasse Evaluation.java zum Test einfacher Beispiele (aber nicht aller interessanten Fälle!) ist dort vorgesehen.

### 4. Aufgabe: Strings

(8 P)

Schreiben Sie eine Funktion `stringTest`, die einen formalen Parameter `String s` hat und testet, ob `s` die Verkettung `t + t` eines anderen (zunächst unbekannten und von Ihnen zu bestimmenden) Strings `t` ist. Wenn dies der Fall ist, soll `t` zurückgegeben werden, ansonsten die `null`-Referenz.

**Beispiele:**

```
stringTest("xyzxyz") ~ "xyz"
stringTest("1111111111") ~ "11111"
stringTest("ababab") ~ null
```

Achten Sie darauf, dass der Parameter `s` hier sogar `null` sein kann!

(Einen Rahmen zur Lösung finden Sie in Moodle unter Aufgabe 04, dort ist auch die Lösung in der Datei A4.java abzuspeichern. Eine passende `main`-Methode mit der Möglichkeit zur testweisen Evaluierung ist in einer Klasse Evaluation bereits vorgegeben.)

## 5. Aufgabe: Rekursion

(8 P)

Implementieren Sie die folgende Funktion  $g$  als rekursive Funktion mit dem Rückgabetyp **String** und einem Parameter **s** vom Typ **String**:

$$g(s) = \begin{cases} s, & \text{falls die Länge von } s \text{ eine gerade Zahl ist,} \\ g(s1) + "#" + g(s2), & \text{in allen anderen Fällen, wobei } s1 \text{ und } s2 \text{ sich ergeben} \\ & \text{aus } s1.length() = s2.length() = (s.length() - 1)/2 \\ & \text{und } s = s1+c+s2 \text{ mit einem einzelnen(!) Character } c. \\ & c \text{ ist also dabei der Character genau in der Mitte von } s, \text{ der durch } "#" \text{ ersetzt wird; der} \\ & \text{vordere und hintere Teil von } s \text{ werden rekursiv analog behandelt.} \end{cases}$$

**Beispiele:**

$$\begin{aligned} g("ab") &\rightsquigarrow "ab", & g("abc") &\rightsquigarrow "###", \\ g("abcde") &\rightsquigarrow "ab#de", & g("abcdefghijklm") &\rightsquigarrow "ab#de#gh#jk" \end{aligned}$$

Ihre Lösung darf dabei außer den Parametern und evtl. lokalen Variablen keine weiteren Variablen verwenden, auch Schleifen sind nicht erlaubt. (Ansonsten wird die Abgabe mit 0 Punkten bewertet, selbst wenn sie die richtigen Resultate liefert. Dies gilt analog für Lösungen, die nur die obigen Beispielresultate reproduzieren.)

Sie dürfen dabei davon ausgehen, dass **s** nicht den Wert **null** hat. Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 05** in der Datei **A5.java**, mit der Möglichkeit zur testweisen Evaluierung durch **Evaluation.java**. In der dortigen Datei **A5.java** ist die Lösung abzuspeichern.

## 6. Aufgabe: Arrays

(12 P)

Implementieren Sie eine Methode `int[] mirrorArray(int[] data)`, die ein (neues!) Array zurückgibt, in dem die Werte aus `data` in umgekehrter Reihenfolge enthalten sind.

Die Inhalte des Feldes `data` müssen dabei erhalten bleiben. Es ist möglich, dass `data` beim Aufruf die `null`-Referenz ist; in diesem Sonderfall soll auch die Null-Referenz zurückgegeben werden.

**Beispiel:**

Parameterfeld `data`: { 1, 2, 3, 4, 5 }  
`mirrorArray(data)`: { 5, 4, 3, 2, 1 }

Parameterfeld `data`: { 12345, 54321 }  
`mirrorArray(data)`: { 54321, 12345 }

Parameterfeld `data`: `null`  
`mirrorArray(data)`: `null`

(Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 06**, inklusive einer passenden `main`-Methode und der Möglichkeit zur testweisen Evaluierung. Dort ist auch die Lösung abzuspeichern.)

## 7. Aufgabe: Exception und lokale Klassen

(15 P)

Vorgegeben ist eine Klasse `Evaluation` mit zwei Methoden: Eine kurze `main`-Methode und eine Methode `void exceptionTrigger()`.

- `main` ruft eine Methode `exceptionCheck` auf, die Sie noch schreiben sollen.
- `void exceptionTrigger()` liest eine `int`-Zahl `n` von der Konsole ein. Sollte die eingegebene Zahl nicht zwischen 1 und 5 liegen, läuft `exceptionTrigger` ohne Probleme durch. Liegt `n` aber in diesem Bereich, wird in Abhängigkeit von `n` eine von fünf unterschiedlichen Exceptions entstehen.

Schreiben Sie nun in der Klasse `Checker` die Methode `int exceptionCheck()` und zudem eine lokale Exception-Klasse `NoProblemException` mit folgenden Eigenschaften:

- `exceptionCheck` soll `exceptionTrigger` aufrufen und durch geeignetes Fangen der jeweils geworfenen Exception die eingelesene Zahl bestimmen und an `main` zurückgeben.
- Welche Zahl bei `exceptionTrigger` zu welcher Exception führt, können Sie im Quelltext der Klasse `Evaluation` und der `Evaluation` unter Moodle leicht erkennen.
- Wenn `exceptionTrigger` also zum Beispiel eine 1 einliest, wird eine `ArithmeticException` geworfen. Dann muss `exceptionCheck` auch eine 1 zurückgeben.
- Ist die von `exceptionTrigger` gelesene Eingabe nicht zwischen 1 und 5, so läuft die Methode *ohne* Exception ab. In solchen Fällen soll nun von `exceptionCheck` selbst eine `NoProblemException` (als Subklasse von `RuntimeException`) geworfen werden.

An der Klasse `Evaluation` dürfen Sie keine Änderungen vornehmen; Ihre Lösung ist komplett in der Datei `Checker.java` abzuspeichern.

## 8. Aufgabe: Vererbung

(15 P)

Vorgegeben ist eine Klasse `Song`. Sie speichert über ein Musikstück Informationen wie den Titel (`String title`) und die Musikgruppe (`String band`). Folgende Methoden werden dabei bereitgestellt:

- Ein Konstruktor der Form

```
Song(String title, String band)
```

- Die getter-Methoden für einen `Song` (also `getTitle()` und `getBand()`)

Implementieren Sie passend dazu ein Interface `Playlist`. Der Zweck dieses Interfaces ist die Speicherung einer Sammlung von Musikstücken (`Song`). Daher soll das Interface die folgenden drei Methoden vorsehen:

- `void addSong(Song mySong)`
- `Song play(String title)`
- `int getSize()`

Leiten Sie schließlich die Klasse `SimplePlaylist` vom Interface ab. Die Klasse `SimplePlaylist` speichert Musikstücke in einem Array (`Song[] songs`). Dabei wird die Größe des Arrays im Konstruktor angegeben. Zudem speichert sie die Anzahl der über `addSong` hinzugefügten Musikstücke in einer geeigneten Variablen `size`.

Implementieren Sie `SimplePlaylist` wie folgt:

- Implementieren Sie einen Konstruktor `public SimplePlaylist(int maxsize)`, der als Parameter die Größe des Arrays hat.
- Implementieren Sie in `SimplePlaylist` die Methode `addSong`, die ihren Parameter-Song zur Playlist-Instanz hinzufügt. Sie können dabei eine beliebige Reihenfolge der Speicherung im Array `songs` wählen, ein Anfügen bei der ersten "freien" Stelle ist sicherlich am einfachsten. Wenn kein freier Platz mehr im Array existiert, geben Sie den folgenden String auf der Konsole aus und ignorieren ansonsten den Aufruf:

```
Playlist full
```

- `getSize()` kann als getter-Methode implementiert werden.
- Zum Schluss sollen Sie in `SimplePlaylist` die Methode `play` des Interfaces implementieren. Diese Methode soll im Array `songs` nach einem Stück mit diesem Titel suchen und dieses, wenn gefunden, zurückgeben. Wenn der gesuchte Titel nicht in `songs` enthalten ist, so geben Sie die `null`-Referenz zurück.

Eine rudimentäre Testklasse `Evaluation` ist vorgegeben, um Ihre Implementierung zu testen. Diese Klasse enthält einige Tests zur Überprüfung, ob die genannten Methoden und Funktionalitäten implementiert wurden.

**9. Aufgabe: Objekte**

(12 P)

Betrachten Sie die Klassen `Client` und `Parser`. Die Klasse `Client` speichert für einen Kunden eine ID, den Namen und eine Geldsumme, die vom Kunden für Waren ausgegeben wurde. Außerdem sind die Getter-Funktionen und eine `toString()`-Funktion bereits vorgegeben.

Erweitern Sie die Klasse `Parser` um eine Methode `createClient(String string)`, die in der Lage ist, Strings der folgenden Form zu verarbeiten und als Objekte zu speichern:

```
{cnr:1234;name:Nyso;value:200}  
{cnr:1234;value:200;name:Nyso}  
{cnr:1234      ;value      :200      ;name      : Nyso}
```

Berücksichtigen Sie, dass die Reihenfolge der Elemente des Triples unterschiedlich sein kann und ebenfalls unnötige Leerzeichen in den Strings vorkommen können.

**Weitere Fehler müssen Sie nicht betrachten, da nur die erwähnten Fehler vorkommen!**

## 10. Aufgabe: Stacks

(15 P)

Betrachten Sie die gegebenen Klassen `Stack` und `Elem`, die den in der Vorlesung vorgestellten Varianten sehr ähnlich sind. Sie enthalten einen Teil einer Stack-Implementierung aus verketteten Listenelementen, wobei jedoch der Typ `Integer` für die Datenkomponente fest voreingestellt ist. Eine `push`-Methode ist bereits vorgegeben, ebenso wie eine Methode `toString()`.

Erweitern Sie die Klasse `Stack` um die folgenden drei Methoden `pop()`, `invert()` und `removeZeroes()`:

1. Die Methode `pop()` ergänzt das Verhalten der Klasse um die (noch fehlende) Pop-Operation, so dass tatsächlich eine lauffähige Stack-Implementierung entsteht.

Bei einem leeren Stack soll `pop()` der Einfachheit halber den Zahlwert `-123456` zurückgeben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 1 (dh. Eingabe von 1) grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 15 10 5 ]
Resultat: 15 10 5 -123456
```

2. Die Methode `invert()` soll das Vorzeichen aller im Stack enthaltenen Einträge umdrehen, d.h. aus positiven Zahlen werden negative Zahlen und umgekehrt. Die Reihenfolge der Werte soll ansonsten unverändert bleiben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 2 grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ -4 -3 0 1 2 0 ]
Resultat: [ 4 3 0 -1 -2 0 ]
```

3. Die Methode `removeZeroes()` soll im Stack alle Einträge entfernen, die den Wert 0 haben. Alle anderen Werte sollen in der Original-Reihenfolge erhalten bleiben.

In der `Evaluation.main()`-Methode wird dies beim Testfall 3 grob getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 0 0 1 0 2 0 ]
Resultat: [ 1 2 ]
```

Beim Testfall 4 werden wichtige Sonderfälle von `removeZeroes()` getestet. Es sollte dabei die folgende Ausgabe produziert werden:

```
Startwerte: [ 0 0 0 ]
Resultat: [ ]
```

Einen Rahmen zur Lösung finden Sie in Moodle unter **Aufgabe 10**, dort ist auch die Lösung abzuspeichern. Zu dieser Aufgabe finden Sie neben `Stack` und `Elem` auch eine Testklasse `Evaluation` mit einer `main`-Methode in Moodle. Die Klassen `Elem` und `Evaluation` dürfen nicht verändert werden.