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 verschieden 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…
Bespiele: `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`** = „xxyxxxBeispielxxyxxxx", **`s2`** = „Beispiel"
→ Rückgabe von **`t`** = „xxyxxxx"
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:

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

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** zurückgeben werde.

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 Articles [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);`