Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt
Fakultät Informatik und Wirtschaftsinformatik
Prof. Heinzl SS 2020

Prüfung im Fach
Programmieren I (Java)

Termin: 22.07.2020
Zeit: 13.30 – 15.00 Uhr
Hilfsmittel: - keine -

Hinweise:

1. Beginnen Sie eine neue Aufgabe stets auf einer neuen Seite.
2. Falls bei einer Aufgabe nicht ausdrücklich erlaubt, dürfen Methoden oder Attribute von vordefinierten Java-Klassen nicht verwendet werden! Unter diese Regelung fallen **nicht**
   - `length` im Zusammenhang mit Feldern/Arrays
   - die Ausgabe-Methoden, wie `print` oder `println`.
   - Methoden der Klasse `Scanner`
   - die Methoden `length()`, `charAt()`, `toCharArray()` der Klasse String
   - die Methode `Math.random()`
3. Mit **45 Punkten** haben Sie die Prüfung in jedem Fall bestanden.

Nachname: _____
(in Druckbuchstaben)

Vorname: _____
(in Druckbuchstaben)

Matrikelnummer: _____

Aufgabe	**1**	**2**	**3**	**4**	**∑**
Maximale Punktzahl	20	18	19	33	90
Erreichte Punktzahl					

Aufgabe 1 (20 Punkte):

Schreiben Sie eine Methode, die den natürlichen Logarithmus von x im Bereich von 0 bis 2 nach folgender Definition berechnet.

$$\ln(x) \coloneqq \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \cdot (x-1)^n \text{, für } 0 < x < 2$$

Wenn der x Wert die Bedingung nicht erfüllt, werfen Sie mit *throw new RuntimeException()* eine Fehlermeldung!
Die Berechnung soll solange laufen, wie der betragsmäßige Wertzuwachs größer als $|10^{-16}|$ ist.

Zur Erinnerung: Sie dürfen **keine** Methoden aus der Klasse `Math` verwenden!

Aufgabe 2 (18 Punkte):

a) Schreiben Sie eine Methode *flatten*, die ein beliebiges zweidimensionales int-Array in ein eindimensionales int-Array umwandelt. Dabei sollen alle Werte des zweidimensionalen Arrays im eindimensionalen Array wiederauftauchen. Die Reihenfolge soll - wie in folgendem Beispiel ersichtlich - umgesetzt werden:

Beispiel:

1	2	3	4	5
6				
7	8			
9	10	11	12	

⇨

1	2	3	4	5	6	7	8	9	10	11	12

Aufgabe 3 (9 + 10 = 19 Punkte):

Implementieren Sie eine verkettete Liste. Gegeben ist eine Knotenklasse, die über ein Namensattribut vom Typ String sowie über eine Referenz auf einen Knoten verfügt, die den *Vorgänger* speichern kann. Ihre Liste soll anstelle einer Referenz auf das *erste* Element, eine Referenz auf das *letzte* Element speichern!

a) Implementieren Sie eine *einfuegen*-Methode, die ein neues Element an das Ende der Liste einfügt.
b) Überschreiben Sie die *toString*-Methode, so dass sie einen String zurückgibt, in der alle Namen von vorne nach hinten mit Leerzeichen getrennt auftauchen. Der String darf am Anfang und/oder am Ende Leerzeichen enthalten.

Beispiel:
In die Liste werden drei Knoten in folgender Reihenfolge eingefügt: Cool, Zylla, Aaronson
Die *ende*-Referenz verweist auf Aaronson.
Die *toString*-Methode gibt den String "Cool Zylla Aaronson" zurück.

Aufgabe 4 (9 + 10 + 14 = 33 Punkte):

a) Im Internet findet ein Austausch mit Webseiten hauptsächlich über die HTTP-Methoden GET und POST statt. Ein Browser benutzt die HTTP-*Methode* GET um bspw. eine Webseite abzurufen. Formulare (bspw. ein Registrierungsformular) nutzen meist die HTTP-*Methode* POST, wenn Sie auf dem Server einen Datensatz anlegen.

Schreiben Sie eine Klasse, die eine HTTP-Anfrage abbilden kann. Dabei soll die HTTP-Anfrage über die HTTP-*Methode* für den Abruf verfügen sowie über die *URL*, von der der Abruf erfolgen soll. Schreiben Sie einen dazu passenden Konstruktor! Achten Sie darauf, dass nur die Werte GET oder POST als HTTP-Methode akzeptiert werden. Bei der Übergabe von anderen Werten soll die Methode auf GET gesetzt werden.

b) Ein HTTP-Header besteht aus einem Name-Wert-Paar und dient der Angabe von weiteren Informationen für den Abruf. Hier kann beispielsweise angegeben werden, welcher Content-Type (z. B. HTML, XML, JSON) der Aufrufer unterstützt.

Beispiel für einen Header:
```
Content-Type: text/html
```

Content-Type ist der Name.
text/html ist der Wert.
Doppelpunkt + Leerzeichen sind die Trennzeichen zwischen Name und Wert.

Erweitern Sie die Klasse aus Teilaufgabe a) so, dass die HTTP-Anfrage über maximal 20 Header verfügen kann. Ergänzen Sie die Klasse um eine Methode *addHeader*, die einen Header übergeben bekommt und als einen der 20 möglichen Header ergänzt.

Hinweis: Sie brauchen sich nicht um das Entfernen von Headern aus der HTTP-Anfrage zu kümmern!

c) Wenn Sie eine Abfrage an einen Webserver schicken, muss diese je nach Protokoll einem bestimmten Format entsprechen. Das HTTP 1.0 Protokoll schreibt folgendes **Format** vor.

```
<http-method> <url> HTTP/1.0CRLF
<header1>CRLF
<header2>CRLF
…
<headerN>CRLF
CRLF
```

CRLF steht dabei für einen Zeilenumbruch (CRLF = Carriage Return Line Feed)

Beispiel: Ein GET Request auf die URL http://www.google.de/index.html mit einem Content-Type Header sieht wie folgt aus:

```
GET http://www.google.de/index.html HTTP/1.0
Content-Type: text/html
```

Überschreiben Sie die *toString*-Methode, so dass Sie die HTTP-Anfrage in oben beschriebenem **Format** zurückgibt.