DRAFT

Technische Universität Braunschweig

IAS | INSTITUTE FOR APPLICATION SECURITY

Klausur zur Vorlesung

Programmieren 1

12. September 2022

Zugelassene Hilfsmittel: *Es sind **keine** Taschenrechner, Mobiltelefone, Headsets, Smartwatches, Smartglasses oder sonstige (kommunikationsfähige) Hilfsmittel zugelassen! In der Hand, auf dem Tisch oder anderweitig in Reichweite befindliche Hilfsmittel werden unmittelbar als Täuschungsversuch gewertet!*

Bearbeitungszeit: 120 Minuten

Hinweise: Bitte ausschließlich dokumentenechte Stifte in einer anderen Farbe als Rot benutzen. **Antworten bzw. Zeichnungen mit Bleistift oder rotem Stift gelten als nicht geschrieben.**

Die Antworten sind in deutscher oder englischer Sprache zu verfassen. Zusätzlich verwendete Blätter sind mit dem Klausur-Code zu versehen. Diese Klausur besteht aus **24 nummerierten Seiten** inkl. Arbeitsblättern. Bitte prüfen Sie Ihre Klausur auf Vollständigkeit!

Name:

Vorname:

Matrikelnummer:

Fachrichtung: ☐ Informatik ☐ WInfo ☐ IST ☐ MeWi ☐ Wirtschaftsingenieurwesen (☐ Bauing ☐ E-Technik ☐ Maschbau) ☐ Maschinenbau ☐ Elektrotechnik ☐ Sonstiges:

Studiengang: ☐ Bachelor ☐ Master ☐ Sonstiges:

Code: DRAFT

DRAFT

Diese Seite nicht beschreiben!

DRAFT

Technische Universität Braunschweig

IAS | INSTITUTE FOR APPLICATION SECURITY

Klausur zur Vorlesung

Programmieren 1

12. September 2022

Code: **DRAFT**

Bewertung

Aufgabe 1	max.	5	Punkte	Punkte	
Aufgabe 2	max.	10	Punkte	Punkte	
Aufgabe 3	max.	15	Punkte	Punkte	
Aufgabe 4	max.	10	Punkte	Punkte	
Aufgabe 5	max.	13	Punkte	Punkte	
Aufgabe 6	max.	6	Punkte	Punkte	
Aufgabe 7	max.	31	Punkte	Punkte	
Summe	max.	90	Punkte	Punkte	

Note:

DRAFT

Diese Seite nicht beschreiben!

Aufgabe 1

```
1  abstract class Fabelwesen {
2      String ort;
3  }
4
5  class Feuerfabelwesen extends Fabelwesen {
6  }
7
8  class Erdfabelwesen extends Fabelwesen {
9  }
10
11 class Drache extends Feuerfabelwesen {
12     private void feuerSpeien() { /* mehr Code */ }
13 }
14
15 class Phoenix extends Feuerfabelwesen {
16     void ausDerAscheAufsteigen() { /* mehr Code */ }
17 }
18
19 class Main {
20     void aufgabe() {
21         Drache smaug = new Drache();
22         smaug.ort = "Höhle";
23         smaug.feuerSpeien();
24         Fabelwesen fw1 = new Phoenix();
25         Fabelwesen fw2 = new Erdfabelwesen();
26         Feuerfabelwesen ffw = (Feuerfabelwesen) fw2;
27         Object o = new Phoenix();
28         Feuerfabelwesen phoenix = o;
29         Feuerfabelwesen glumanda = new Feuerfabelwesen();
30         glumanda.ort = "Alabastia";
31         glumanda.ausDerAscheAufsteigen();
32         Drache erdwurm = new Erdfabelwesen();
33         fw1.ausDerAscheAufsteigen();
34     }
35 }
```

Dieser Java-Code definiert eine Vererbungshierarchie und nutzt diese in der Methode `aufgabe()`. In dieser Methode sind mindestens 6 Fehler enthalten. Für die volle Punktzahl müssen Sie 5 Zeilennummern mit Fehlern nennen und diesen in einem Satz beschreiben.

(5 Punkte)

(Gesamt 5 Punkte)

Aufgabe 2

Jedem der folgenden Ausdrücke geht die Deklaration `int x = 42;` voraus.
Geben Sie für jeden der Ausdrücke den Java-Datentyp und den Wert des Ausdrucks an.

1. `x++ % 2`

   (2 Punkte)

   Typ: _____ Wert: _____

2. `2 * x * 0.5`

   (2 Punkte)

   Typ: _____ Wert: _____

3. `"A" + x + "B" + 3.14`

   (2 Punkte)

   Typ: _____ Wert: _____

4. `5 / 2`

   (2 Punkte)

   Typ: _____ Wert: _____

5. `++x - 20`

   (2 Punkte)

   Typ: _____ Wert: _____

(Gesamt 10 Punkte)

Aufgabe 3

Beantworten Sie die folgenden Fragen im Bezug auf die gegebene rekursive Funktion `r`.

1. In welcher Reihenfolge und mit welchen Parametern wird *r* bei `r(9, "")` aufgerufen?

   Geben Sie eine Liste der Aufrufe an.

   (11 Punkte)

2. Welchen Wert liefert der Aufruf `r(9, "")`?

   (2 Punkte)

3. Wie groß ist die maximale Rekursionstiefe, das heißt die maximale Anzahl gleichzeitig aktiver Aufrufe, für `r(9, "")`?

   (2 Punkte)

```
1  static String[] buffer = {
2    "p", "f", "u", "r", "n", "f", "o", "u", "n", "g"
3  };
4
5  static String r(int i, String s) {
6      if (i < 0) {
7          return s;
8      }
9      if (i % 3 != 0) {
10         return r(i - 1, s);
11     }
12     return r(i - 1, buffer[i] + s);
13 }
```

(Gesamt 15 Punkte)

Aufgabe 4

Ihnen liegt die Dokumentation einer Klasse und ihrer Methoden vor. Sie sollen die nachfolgenden Aufgaben ausschließlich unter Verwendung der gegebenen Methoden lösen! Hierbei dürfen Sie die gegebene Klasse `MyList` nicht verändern, keine eigenen Methoden hinzufügen oder andere syntaktische Elemente von Java verwenden.

Gefordert ist lediglich jener Quellcode, der zum Lösen der Aufgabenstellung erforderlich ist. Eine main-Methode oder eine Klasse sind nicht notwendig. Lediglich das Deklarieren von Variablen, mathematischen Operatoren, casting auf primitive Datentypen, `System.out.println()`, die Instantiierung der gegebenen Klasse `MyList` sowie das Aufrufen ihrer Methoden ist erlaubt!

Ihnen sind in allen 4 Teilaufgaben neue Instanzen a und b der Klasse `MyList` gegeben und Sie sollen mit diesen weiter arbeiten. Sie können davon ausgehen, dass jede Instanz mindestens ein Element enthält. Die Teilaufgaben sind unabhängig von einander zu bearbeiten.

1. Geben Sie von jeder Instanz der Liste das jeweils größte Element aus.

   (2 Punkte)

2. Erstellen Sie eine Liste, die aus den Zahlen 1, 2 und 3 besteht.

   (2 Punkte)

3. Geben Sie den Durchschnittswert von Instanz `a` als Fließkommazahl aus.

   (3 Punkte)

4. Geben Sie die Summe der Werte in der Schnittmenge von `a` und `b` aus.

   (3 Punkte)

```java
/**
 * Die Implementierung einer Liste.
 *
 * Die Reihenfolge der Elemente in der Liste entspricht
 * der Reihenfolge in der sie hinzugefügt wurden,
 * mit dem Element aus dem Konstruktor an erster Stelle.
 */
class MyList {

      /**
       * Konstruktor, der eine Instanz mit nur einem Element erstellt.
       */
      public MyList(int arg);

      /**
       * Fügt ein neues Element an das Ende der Liste hinzu.
       *
       * @param arg Das neue Element.
       * @return Die aktuelle Liste mit dem zusätzlichen Element.
       */
      public MyList add(int arg);

      /**
       * Summiert alle Elemente.
       * @return Die Summe aller Elemente der Liste.
       */
      public int sum();

      /**
       * Sortiert die Liste aufsteigend.
       * @return Die aktuelle Liste mit aufsteigend sortierten
          Elementen.
       */
       public MyList sort();

       /**
        * Gibt das letzte Element der Liste zurück.
        * @return Das Element, das aktuell am Ende der Liste steht.
        */
       public int last();
```

```
40          /**
41           * Gibt die Anzahl der Elemente in der Liste zurück.
42           * @return Die Anzahl der Elemente.
43           */
44          public int length();
45
46          /**
47          * Erstellt eine neue Instanz, bestehend aus der Schnittmenge
                der beiden Listen.
48           * Weder die aktuelle noch die übergebene Instanz werden dabei
                 verändert.
49           *
50           * @param other Die übergebene Instanz.
51           * @return Eine neue Instanz bestehend aus der Schnittmenge
                der aktuellen und der übergebenen Instanz.
52           */
53          public MyList intersection(MyList other);
54  }
```

(Gesamt 10 Punkte)

Aufgabe 5

Füllen Sie die Lücken im folgenden Java-Programmtext so aus, dass das Programm die nachfolgende Ausgabe erzeugt:

> Bojack H., spoken by the famous Will Arnett, has Todd C. living with him/her
> Todd C., spoken by the beloved Aaron Paul
> Diane N., spoken by the beloved Alison B.

Sie dürfen dabei den vorhandenen Code auf keine Art und Weise erweitern oder ändern. Nicht jede Lücke muss notwendigerweise auch gefüllt werden.

(13 Punkte)

```
1  _____ _____ {
2    _____ print();
3  }
4  _____ Kind { _____ , Recurring }
5
6  _____ class Person implements Printable {
7      private String name;
8
9      public Person(String name) {
10         this.name = name;
11     }
12
13     public String getName() {
14         return name;
15     }
16 }
17
18 class VoiceActor _____ _____ {
19     private Kind kind;
20
21     public VoiceActor(String name, Kind kind) {
22       _____ (name);
23         this.kind = kind;
24     }
25
26     _____ String print() {
27         String s = _____ ;
28         switch (this.kind) {
29             case Star:
30                 return "the famous " + s;
31             default:
32                 return "the beloved " + s;
33         }
34     }
35 }
```

```
36 class Character _____  Person  {
37     private VoiceActor actor;
38     private Character guest;
39
40     public Character(String name, VoiceActor actor) {
41         super(name);
42         this.actor = actor;
43     }
44
45     public Character(String name, VoiceActor actor, _____
           guest) {
46         super(name);
47         this.actor = actor;
48         this.guest = guest;
49     }
50
51     public String print() {
52       return this._____
             + ", spoken by " + this.actor._____
             + (this.guest != null ? ", has " + this.guest._____
             + " living with him/her" : "");
53     }
54 }
55
56 public class HorsinAround {
57     public static void main(String[] args) {
58         VoiceActor _____  = new VoiceActor(_____
              , _____ );
59         Character todd = new Character("Todd C.", aaron);
60         VoiceActor _____  = new VoiceActor(_____
              , _____ );
61         Character diane = new Character("Diane N.", new VoiceActor("
              Alison B.", Kind.Recurring));
62         Character bojack = new Character("Bojack H.", will,
              _____
              );
63
64         Person[] persons = {
65           _____ , todd, _____
66         };
67         for(Printable p : persons) {
68           System.out.println(p._____ );
69         }
70     }
71 }
```

Aufgabe 6

Schreiben Sie ein Java-Programm, welches angefangen mit der 1 in 3-er Schritten jede Ganzzahl bis einschließlich der 10000 ausgibt, wobei bei Zahlen, die ein Vielfaches von 5, oder 13, oder beidem sind, **stattdessen** `"TUBS"` ausgegeben werden soll.

Beispielausgabe:

```
1
4
7
TUBS
TUBS
16
19
22
TUBS
28
```

Sie dürfen davon ausgehen, dass Ihr Code in der `main`-Methode einer Klasse steht. Die Deklaration einer Klasse und einer `main`-Methode ist nicht notwendig. Erstellen Sie keine weiteren Klassen oder Methoden.

Der Import und die Verwendung von Klassen und Methoden aus der Java-Standardbibliothek, wie den `Scanner`, ist nicht gestattet. Ausnahmen sind die Klasse `String`, `System.out.print` und `System.out.println`.

Bitte beachten Sie, dass Sie nur dann Punkte erhalten, wenn Sie ein Java-Programm schreiben, das erkennbar geeignet um ist die Problemstellung zu lösen.

(6 Punkte)

(Gesamt 6 Punkte)

Aufgabe 7

Schreiben Sie ein Programm, welches bestimmt, welche*r Spieler*in nach einer Partie Tic-Tac-Toe gewonnen hat.

Dafür bekommt Ihr Programm als Argument auf der Kommandozeile das Spielbrett. Das Spielbrett wird hierfür von links nach rechts und von oben nach unten mit den Zeichen `"x"`, `"o"` und `"-"` definiert. `"x"` und `"o"` sind hier die Spielersymbole und `"-"` das Symbol für ein leeres Feld. Die einzelnen Zeichen sind hierbei separate Argumente.

Ihr Programm soll nun bestimmen welche*r Spieler*in gewonnen hat. Dazu soll entweder `"x"` oder `"o"` ausgegeben werden, wenn die jeweilige Person gewonnen hat oder `"-"` falls niemand gewonnen hat.

Sie können davon ausgehen, dass in dem ausgefüllten Spielbrett höchstens ein Gewinner zu erkennen ist. D.h. entweder hat nur eine Person drei Zeichen in einer Reihe oder niemand.

Implementieren Sie eine Funktion `einlesen`, die das Spielbrett einliest und eine `IllegalArgumentException` wirft, wenn zu wenige oder falsche Zeichen im Input vorhanden sind. Implementieren Sie außerdem eine Funktion `bestimmen`, die bestimmt wer gewonnen hat.

Implementieren Sie eine main-Funktion, die die beiden Hilfsfunktionen nutzt, um die Anforderungen zu erfüllen. Alle Exceptions die von ihrem Code geworfen werden, müssen in der main-Funktion auch behandelt und sinnvolle Fehlermeldungen ausgegeben werden.

Bitte beachten Sie, dass Sie nur dann Punkte erhalten, wenn Sie ein Java-Programm schreiben, das erkennbar geeignet ist um die Problemstellung zu lösen.

Verwenden Sie aussagekräftige Variablen- und Funktionsnamen und kommentieren Sie Ihren Code!

Hinweise

- Der Import und die Verwendung von Klassen und Methoden aus der Java-Standardbibliothek, wie den `Scanner`, ist nicht gestattet.
- Lösungen, die dagegen verstoßen, werden **nicht gewertet**. Ausnahmen sind die Klasse `String`, `System.out.print` und `System.out.println`.
- Sie sollen das Spiel Tic-Tac-Toe an sich **nicht** implementieren! Einzig die Prüfung, wer gewonnen hat, soll von Ihnen implementiert werden.

Auf der nächsten Seite finden Sie die Spielregeln und ein Beispiel.

Spielregeln

- Das Spielbrett ist 3x3 Felder groß.
- Es gibt zwei Spieler*innen.
- Ein*e Spieler*in hat das `x` als Symbol, die andere Person das `o`.
- Die Spieler*innen schreiben abwechselnd ihr Symbol in ein freies Feld.
- Das Spiel ist zu Ende, wenn das Spielbrett voll ist oder ein*e Spieler*in gewonnen hat.
- Ein*e Spieler*in gewinnt, wenn drei ihrer Zeichen direkt nebeneinander liegen. Das ist in horizontaler, vertikaler und diagonaler Richtung möglich.

Beispiel

`java IhrProgramm x x x - x o - o o` ist der Aufruf ihres Programmes mit den Argumenten, die folgendes Spielbrett beschreiben. Beachten Sie, dass die Argumente mit Leerzeichen getrennt sind.

x	**x**	**x**
	x	o
	o	o

Die erwartete Ausgabe ist `"x"`, da die Person mit dem Symbol `x` durch die oberste Reihe gewonnen hat.

(31 Punkte)

(Gesamt 31 Punkte)