Programmiertechnik 1 – Klausur WS2020/2021
22.2.2021, 14:00–16:00 Uhr

Die Klausur besteht aus 8 Aufgaben, die zusammen 100 Punkte ergeben.
Als einziges Hilfsmittel ist ein Spickzettel im Format DIN-A4 zugelassen. Viel Erfolg!

Aufgabe 1: Zahlensysteme und Zeichencodes *(10 Punkte)*

a) Geben Sie die Dezimalzahl **`88`** in folgenden Schreibweisen an:
- als ganzzahliges binäres Java-Literal
- als ganzzahliges oktales Java-Literal
- als ganzzahliges hexadezimales Java-Literal
- als Java-Gleitkommaliteral

Ihre Lösung muss den vollständigen Rechenweg enthalten. *(4 Punkte)*

b) Geben Sie die negative Dezimalzahl **`-88`** als Binärzahl in 16-stelliger Zweierkomplementdarstellung an. Ihre Lösung muss den vollständigen Rechenweg enthalten. *(2 Punkte)*

c) Geben Sie die Binärzahl mit Festkomma **`101,101`** als Dezimalzahl mit Festkomma an. Ihre Lösung muss den vollständigen Rechenweg enthalten. *(2 Punkte)*

d) Die Ziffer **`1`** hat laut ASCII-Tabelle den oktalen Zeichencode **`061`**. Java verwendet intern UTF-16. Geben Sie den UTF-16-Zeichencode der Ziffer **`1`** in Binärdarstellung an. *(2 Punkte)*

Hinweis: ein UTF-16-Code lässt sich leicht aus einem ASCII-Code herleiten

Aufgabe 2: Datentypen *(13 Punkte)*

a) Betrachten Sie die folgenden Variablenzuweisungen:

```
a = 12 <= 34
b = 56 / 78
c = 90 * 0.1
d = "23" + 45
e = new double[67]
```

Geben Sie zu jeder Variablen an:
- den Typ, den sie haben muss, damit ihr Typ genau dem Typ des zugewiesenen Werts entspricht
- den Wert, den sie nach der Zuweisung hat

(5 Punkte)

b) Nennen Sie alle Werttypen (= primitive Datentypen) sowie alle Arten von Referenztypen in Java. *(4 Punkte)*

c) Die Zuweisung **`=`** und der Vergleich **`==`** haben bei Werttypen (= primitive Datentypen) einerseits und Referenztypen andererseits eine unterschiedliche Bedeutung. Erklären Sie den Unterschied. *(2 Punkte)*

d) Nennen Sie den allgemeinsten Referenztyp von Java und erklären Sie, welche Rolle dieser Typ in der folgenden Ausgabeanweisung spielt: **`System.out.println(new Beispiel());`** *(2 Punkte)*

Aufgabe 3: Ausdrücke *(8 Punkte)*

a) Gegeben ist der folgende Java-Ausdruck:

```
c == b || c == (char)(b >>> 8 | b << 8)
```

Zeichnen Sie den Auswertungsbaum des Ausdrucks und nummerieren Sie darin die Knoten entsprechend der Ausführungsreihenfolge (Nummer eins für den als Erster auszuführenden Operator). *(7 Punkte)*

Hinweis: So wie die Operanden b, c und 8 mehrfach im Ausdruck vorkommen, kommen sie natürlich auch mehrfach im Auswertungsbaum vor.

b) Wenn **b** den Typ **char** und den Wert **'\ufeff'** hat, welchen Wert hat dann der folgende Ausdruck?

```
(char)(b >>> 8 | b << 8)
```

(1 Punkt)

Aufgabe 4: Anweisungen *(12 Punkte)*

Betrachten Sie die folgende Java-Anwendung:

```
 1  public final class Aufgabe4 {
 2      private Aufgabe4() { }
 3      public static void main(final String[] args) {
 4          String s = "";
 5          for (String vorname: args) {
 6              System.out.print(s + vorname);
 7              char c = vorname.charAt(vorname.length() - 1);
 8              switch (c) {
 9              case 's':
10              case 'x':
11              case 'z':
12                  System.out.print('\'');
13                  break;
14              default:
15                  System.out.print('s');
16              }
17              s = " und ";
18          }
19          System.out.println(" erstes Java-Programm funktioniert!");
20      }
21  }
```

a) Was gibt die obige Anwendung auf dem Bildschirm aus, wenn sie wie folgt aufgerufen wird: *(5 Punkte)*

```
java Aufgabe4 Hans Heinz Herta
```

b) Formulieren Sie die obige **for-each**-Schleife über alle Arrayelement als **for**-Schleife mit Index-Laufvariabler. Die umformulierte **for**-Schleife soll dasselbe tun wie die ursprüngliche Schleife. *(4 Punkte)*

Hinweis: Sie brauchen Anweisungen im Schleifenrumpf, die unverändert bleiben, nicht abzuschreiben. Geben Sie stattdessen nur die Zeilennummern an

c) Ersetzen Sie die obige **switch-case-default**-Anweisung durch **if-else**-Anweisungen. Die umformulierte Anweisungsfolge soll dasselbe tun wie die ursprüngliche Anweisung. *(3 Punkte)*

Aufgabe 5: Zeichenketten *(8 Punkte)*

a) In Java lässt sich alles in ein String-Objekt umwandeln. Welche mehrfach überladene Klassenmethode der Klasse **`String`** kann als Fabrikmethode für die Umwandlung verwendet werden? Geben Sie ein Beispiel für den Methodenaufruf an. *(2 Punkte)*

b) Wie sieht die von der Instanzmethode **`toString()`** gelieferte Stringdarstellung eines Objekts aus, wenn in der Klasse des Objekts keine besonderen Vorkehrungen zur Stringdarstellung getroffen wurden? *(1 Punkt)*

c) Wie testet man Strings auf Wertgleichheit? Geben Sie ein Beispiel an. *(2 Punkte)*

d) Warum gibt es in der Java-Bibliothek zusätzlich zur Klasse **`String`** noch eine Klasse **`StringBuilder`**? Wofür kann die Klasse **`StringBuilder`** zum Beispiel verwendet werden? *(3 Punkte)*

Aufgabe 6: Variablen *(15 Punkte)*

a) Definieren Sie in Java eine öffentliche instanziierbare Klasse **`A`** mit folgenden Bestandteilen:
- einer öffentlichen Klassenvariablen **`kv`** mit ganzzahligem Typ und Anfangswert **`11`**
- einer öffentlichen Instanzvariablen **`iv`** mit ganzzahligem Typ und Anfangswert **`22`**
- einer öffentlichen Instanzmethode **`set`** ohne Rückgabewert mit einem ganzzahligen Parameter **`p`**, die der Instanzvariablen **`iv`** den Wert von **`p`** zuweist

(5 Punkte)

b) Definieren Sie in Java eine Main-Klasse **`B`**. Deren **`main`**-Methode soll eine lokale Variable **`v`** vom Typ **`A`** enthalten, die ein Objekt der Klasse **`A`** aus a) referenziert, und soll die Werte der Variablen **`kv`** und **`iv`** aus a) mit **`println`** auf die Konsole schreiben. *(6 Punkte)*

c) Instanziierbare Klassen sollten im Sinne der Kapselung Konsistenzbedingungen für ihre Instanzvariablen definieren. Mit welchen vier Maßnahmen stellt man in Java sicher, dass Instanzvariablen nur gültige Werte gemäß den Konsistenzbedingungen ihrer Klasse annehmen können? *(4 Punkte)*

Aufgabe 7: Methoden *(15 Punkte)*

a) Definieren Sie in Java eine öffentliche abstrakte Klasse **`C`** mit folgenden Bestandteilen:
- einer privaten Instanzvariablen **`pv`** vom Typ **`int`**
- einem öffentlichen Konstruktor mit einem Parameter **`p`**, der zum Initialisieren der privaten Instanzvariablen **`pv`** verwendete wird
- einer öffentlichen, nicht überschreibbaren Instanzmethode **`get`** ohne Parameter zum Abfragen des Werts der Instanzvariablen **`pv`**

(6 Punkte)

b) Definieren Sie in Java eine Klasse **`D`** als Unterklasse von **`C`** mit einem öffentlichen Standardkonstruktor, der die private Instanzvariable **`pv`** der Oberklasse mit **`7`** initialisiert. *(4 Punkte)*

c) Definieren Sie in Java eine Main-Klasse **`E`**, deren **`main`**-Methode alle Konstruktoren und Instanzmethoden von **`C`** und **`D`** direkt oder indirekt aufruft. *(4 Punkte)*

d) Wie ist der heimliche Parameter **`this`** im Konstruktor und der Instanzmethode von **`C`** definiert? *(1 Punkt)*

Aufgabe 8: Objektorientierung *(19 Punkte)*

Für die Angabe von Geschwindigkeiten sind die Einheiten Kilometer pro Stunde, Meilen pro Stunde und Knoten in Gebrauch. Diese Einheiten lassen sich in einem Java Aufzählungstyp zusammenfassen:

```
public enum Einheit { KMH, MPH, KN }
```

Die folgende beispielhafte Anweisungsfolge nutzt den Aufzählungstyp bei einer Klasse **`Geschwindigkeit`**:, um eine Geschwindigkeit von 8,8 km/h umgerechnet in Knoten auf die Standardausgabe zu schreiben:

```
Geschwindigkeit v = Geschwindigkeit.valueOf(8.8, Einheit.KMH);
System.out.printf("%f%s%n", v.getValue(Einheit.KN), Einheit.KN);
```

a) Schreiben Sie die in der obigen Anweisungsfolge verwendete Java-Klasse **`Geschwindigkeit`** mit folgenden Eigenschaften: *(14 Punkte)*

Halten Sie die gewohnten Programmierkonventionen ein, aber verzichten Sie auf Javadoc-Kommentare

- Die Klasse soll nicht als Oberklasse verwendbar sein.
- Jede Instanz der Klasse soll eine Geschwindigkeit in Meter pro Sekunde mit Typ **`double`** kapseln. Die Geschwindigkeit soll nach der Initialisierung nicht mehr änderbar sein.
- Die Klasse hat einen privaten Konstruktor mit einem Parameter vom Typ **`double`** zum Initialisieren der Geschwindigkeit. Der Konstruktor überwacht die folgenden Konsistenzbedingungen:
  - die Geschwindigkeit darf nicht negativ sein
  - die Geschwindigkeit darf nicht größer als die Lichtgeschwindigkeit 299792458 m/s sein

  Hinweis: Verwenden Sie zum Prüfen der Konsistenzbedingungen die Klassenmethode
  `Double.compare(double, double)`
  Die Methode liefert eine ganze Zahl kleiner 0, wenn das erste Argument kleiner als das zweite ist, größer 0, wenn das erste Argument größer ist, sonst 0.
- Die Klasse hat eine öffentliche Klassenmethode **`valueOf`** mit zwei Parametern, die als Rückgabewert eine Referenz auf ein neues Objekt der Klasse **`Geschwindigkeit`** liefert. Der erste Parameter ist der Betrag der Geschwindigkeit als Gleitkommazahl, der zweite die Einheit als Element der Aufzählung **`Einheit`** (siehe den Beispielaufruf in der obigen Anweisungsfolge).

  Hinweis: Der Konstruktor erwartet die Geschwindigkeit in Meter pro Sekunde.
  Es gelten folgende Umrechnungsfaktoren:
  1 km/h = 0,27778 m/s
  1 kn = 0,51444 m/s
  1 mph = 0,44704 m/s
- Die Klasse hat eine öffentliche Instanzmethode **`getValue`** zum Abfragen der gespeicherten Geschwindigkeit. Ein Parameter legt fest, in welcher der drei Einheiten der Betrag geliefert wird (siehe den Beispielaufruf in der obigen Anweisungsfolge)

  Hinweis: Die Objekte speichern die Geschwindigkeit in Meter pro Sekunde.
  Beachten Sie die bei ***`valueOf`*** *genannten Umrechnungsfaktoren.*

b) Die Klasse **`Geschwindigkeit`** soll ein Bauplan für Wertobjekte *(value objects)* sein. Solche Klassen müssen eigentlich bestimmte Methoden der Oberklasse **`Object`** überschreiben. Geben Sie die Signaturen und Rückgabetypen dieser Methoden an. *(3 Punkte)*

c) Was muss bei der Klasse **`Geschwindigkeit`** ergänzt werden, damit die Bibliotheksmethode **`java.util.Arrays.sort(Object[])`** ein Array von **`Geschwindigkeit`**-Objekten sortieren kann? *(2 Punkte)*

Hinweis: Es ist nur das Prinzip gefragt, keine vollständige Implementierung.

<u>Nach Ende der Bearbeitungszeit</u>:

1. Laden Sie Fotos bzw. Scans Ihrer handschriftlichen Lösungen in Moodle hoch.
   Sie haben dafür maximal 15 Minuten Zeit.

   Die hochgeladenen Fotos bzw. Scans dienen als Nachweis,
   dass Sie die Bearbeitungszeit eingehalten haben.

2. Stecken Sie Ihre handschriftlichen Lösungen in den vorbereiteten Briefumschlag.
   Unterschreiben Sie die eidesstattliche Erklärung und die Datenschutzhinweise und stecken Sie das Blatt mit in den Briefumschlag oder laden Sie ein Foto bzw. einen Scan davon in Moodle hoch.

3. Werfen Sie den zugeklebten Briefumschlag so schnell wie möglich an der Hochschule oder korrekt frankiert in einen Briefkasten ein.

   Prof. Dr. H. Drachenfels
   HTWG Konstanz
   Alfred-Wachtel-Straße 8
   78462 Konstanz
   (Frankieren Sie bei Postversand mit 1,55 Euro)

4. Teilen Sie mir anschließend per E-Mail mit, wann Sie den Brief an der Hochschule eingeworfen bzw. per Post abgeschickt haben.