Probeklausur
Hochschule Zittau/Görlitz,
Prüfer: Prof. Dr. Georg Ringwelski

Einführung in die Programmierung

(K-)II/Wb17

Name:

Matrikelnummer:

Punkte:

1	2	3	4	5	6	Gesamt
/21	/19	/20	/20	/20	/20	/120

Spielregeln:

- Die Bearbeitungszeit beträgt **120 Minuten** ab Ausgabe der Klausur
- Die maximale Punktzahl ist 120, für jede Minute einen. Achten Sie bei der Bearbeitung darauf die **Zeit pro Aufgabe** an den zu erreichenden Punkten auszurichten.
- Abzugeben ist **ausschließlich das ausgegebene Papier**. Lösungen werden direkt unter die Fragen und ggf. auf Rückseiten der Blätter geschrieben. Lösungen auf eigenem Papier werden nicht bewertet.
- Zur Bearbeitung sind **dokumentenechte Stifte** zu verwenden (kein Bleistift, Füller o.ä.)
- Wer die **Prüfung verlässt** um zur Toilette zu gehen, gibt währenddessen seine Klausur ab. Es kann immer nur eine Person gehen.
- Jeder **Betrugsversuch** führt zum sofortigen Ende der Klausur für alle Beteiligten und deren Beurteilung mit der Note 5.
- Die Klausur besteht aus x **Seiten, prüfen Sie den vollständigen Erhalt**. Alle Seiten sind abzugeben und Name und Matrikelnummer sind auf jedem separaten Blatt einzutragen.
- Es sind **keine elektrischen Hilfsmittel** zugelassen! (Hilfsmittel aus Papier (Skripte etc) sind erlaubt, fremde Handschrift bitte vorher kopieren)

Viel Erfolg !

Name: MatNr:

1. Vermischtes

Für jede Teilaufgabe gibt es bis zu 3 Punkte. Es sind beliebig viele Antworten richtig.

a) In welchen der folgenden Zeilen finden zulässige Typumwandlungen statt, so dass sie sich im Kontext eines Java-Programms übersetzen lassen.

- ☐ `int x = 3; long y = x;`
- ☐ `double x = 3; float y = x;`
- ☐ `String x = "X"; char y = x;`
- ☐ `char x = 5; int y = x;`
- ☐ `long[] x = new long[3]; int[] y = x;`

b) Was ist die Ausgabe folgenden Programms?

```
class Test{
     public static void main(String[] a){
          int x = 5; int y = 6;
          System.out.println("Ergebnis= "+ ++x+2+y++);
     }
}
```

- ☐ Kompiliert nicht
- ☐ Laufzeitfehler
- ☐ `Ergebnis=` 527
- ☐ `Ergebnis=` 626
- ☐ `Ergebnis=` 14
- ☐ `Ergebnis=` 15
- ☐ `Ergebnis=` 627
- ☐ `Ergebnis=` 87

c) Welche Ausgabe erzeugt folgender Code?

```
class X{
   public static void main(String[] a){
        int[] x;
        x[0] = 42;
   }
   System.out.println("x="+x);
}
```

(Eine Antwort ist richtig)

- ☐ Das Programm kann man nicht übersetzen
- ☐ Das Programm kann übersetzt werden, erzeugt aber einen Laufzeitfehler
- ☐ Das Programm läuft, erzeugt aber keine Ausgabe
- ☐ x=42
- ☐ x=0
- ☐ x=[I@15db9742

Name: MatNr:

d) Was ist die Ausgabe folgenden Programms?

```
class Test{
     public static void main(String[] args){
          int[] a = {1,2,3};
          for(int i=0; i< a.length; i++){
               for(int  j=0; j < a.length;j++){
                    if(a[i] <a[j])
                         System.out.print(a[i]);
               }
          }
     }
}
```

- ☐ Kompiliert nicht
- ☐ Laufzeitfehler
- ☐ Terminiert nicht
- ☐ Das Programm macht keine Ausgaben
- ☐ 111223
- ☐ 123
- ☐ 12
- ☐ 112

e) Welche Aussagen über den Standardkonstruktor der Klasse K sind zutreffend?

- ☐ Der Standardkonstruktor ist eine Methode von K.
- ☐ Der Rückgabetyp des Konstruktors ist K.
- ☐ Der Standardkonstruktor kann einen beliebigen Namen haben
- ☐ Der Standardkonstruktor hat keine Parameter
- ☐ Vor der Definition des Standardkonstruktors muss das Wort static stehen

f) Welche Aussagen über statische Methoden und Attribute sind zutreffend?

- ☐ In statischen Methoden dürfen Objekte erzeugt werden
- ☐ Statische Methoden dürfen nicht-statische Methoden ohne ein Objekt aufrufen
- ☐ Nicht-statische Methoden dürfen statische Methoden ohne ein Objekt aufrufen
- ☐ Auch bei statischen Attributen einer Klasse kann jedes Objekt dieser Klasse einen unterschiedlichen Wert haben
- ☐ Statische Attribute werden nicht vererbt

Name: MatNr:

g) Welche Aussagen sind zutreffend?

- ☐ Ein Programm terminiert, genau dann wenn es bei den gleichen Eingabewerten immer die selbe Ausgabe erzeugt
- ☐ Ein Programm heißt deterministisch genau dann wenn die Reihenfolge seiner Befehle bei jedem Ablauf gleich ist.
- ☐ Ein deterministischer Algorithmus ist immer auch determiniert
- ☐ Die Komplexität eines Algorithmus wird durch die Anzahl der in ihm verwendeten Steueranweisungen (for, while, if) bestimmt
- ☐ Der Zustand eines Objekts in der OOP ist bestimmt durch die aktuellen Werte seiner Attribute und lokalen Variablen

2. Imperative Programmierung

a) (10 Punkte) Schreiben Sie eine Java Prozedur **`char[] reverse(char[] a)`**, die Arrays von Buchstaben „umdreht“. Z.B liefert **`reverse([a,b,c])`** das Array **`[c,b,a]`**.

b) (9 Punkte) Schreiben Sie unter Verwendung dieser Funktion eine Methode **`boolean isPalindrome(char[] a)`**, die überprüft, ob die Buchstabenliste im Parameter ein Palindrom ist.

(Palindrome sind solche Wörter, die vorwärts und rückwärts gelesen gleich sind, wie z.B.“otto“, „rentner“ oder „reliefpfeiler“.)

3. Imperative Programmierung

a) (13 Punkte) Schreiben Sie eine Java Prozedur **`int teiler(int x`**), die die Anzahl der ganzzahligen Teiler von **`x`** berechnet.

Eine Zahl y ist ein ganzzahliger Teiler von x, wenn $1 \leq y \leq x$ und $x \bmod y = 0$ gilt.

Hinweis 1: Um die Teiler y zu finden kann man für alle Werte zwischen 1 und x ausprobieren, ob sie bei der ganzzahligen Division den Rest 0 ergeben.

b) (7 Punkte) Schreiben Sie eine Testprogramm für **`teiler`**

Name: MatNr:

4. Testen

(20 Punkte) Implementieren Sie ein Testprogramm, das die u.g. Klasse `StrUtil` zur Arbeit mit Zeichenketten testet. Realisieren Sie je Methode zwei qualitativ unterschiedliche Testfälle.

Sie brauchen nicht die Klasse `StrUtil` zu implementieren!

```
class StrUtil{
  // umwandeln aller Zeichen der Zeichenkette in Großbuchstaben
  static String toUpperCase(String s){…}

  // liefert die Anazhal der Zeichen
  static int length(String s){…}

  // prüft, ob s in t vorkommt, wie „all“ in „Hallo“ vorkommt
  static boolean contains(String t, String s){…}
}
```

5. OOP Syntax

(20 Punkte) Implementieren Sie die Java-Klassen, die notwendig sind, damit sich folgender Code kompilieren lässt und bei Ausführung in allen Fällen “test ok” ausgibt.

```
class Test{
  public static void main(String[] a){
    erwarteGleich(42, C.methode());

    C x = new C(10);
    erwarteGleich(10, x.foo());

    C y = new C();
    erwarteGleich(43, y.foo())
    erwarteGleich(10, x.foo());

    if(x.equals(y))
       erwarteGleich(1,1);
    else
       erwarteGleich(-1,1);
  }

  static void erwarteGleich(int soll, int ist){
    if(soll == ist) System.out.println(“test ok“);
    else System.out.println(“fehler“);
  }
}
```

Name: MatNr:

6. OOP

(20 Punkte)

a) Geben Sie die Vererbungshierarchie der Klassen folgender Verkehrsmittel an: Auto, Fahrrad, Bahn, Flugzeug, Verkehrsmittel, Offentliche Verkehrsmittel, Individualverkehrsmittel.

b) Implementieren Sie die entsprechenden Klassen, so dass

i. … man für Bahn und Flugzeug Verbindungen abfragen kann, indem man zwei Ortsnamen als Zeichenketten eingibt. Zur Berechnung von Verbindungen sei die Klasse `Verbindung` gegeben, die einen Konstruktor `Verbindung(String start, String ziel, Verkehrsmittel v)` zur Verfügung stellt

ii. … bei jedem Individualverkehrsmittel ein Kaufpreis mit einem getter abgefragt werden kann.

iii. … jedem Verkehrsmittel im Konstruktor der Name seines Besitzers als Zeichenkette zugeordnet werden muss. Standardkonstruktoren gibt es nicht.