WS2016/17 Klausur Programmierung 1 Donnerstag, 02.02.2017

Es sind keine Hilfsmittel außer dem Schreibgerät zugelassen.

Es sind keine Unterlagen zugelassen.

Es ist ausschliesslich das vom Dozenten ausgeteilte Papier zugelassen.

Bearbeitungszeit: 80 min

Bitte Vollständig ausfüllen:

Vorname:
Nachname:
Matrikelnummer:
Letzter Prüfungsversuch:

Leistungsbewertung (integrierte Lehrveranstaltung):

Maximal sind 200 Punkte erreichbar.

Ab 100 Punkten (gesamt) ist die Klausur bestanden.

Aufgabe	1	2	3	4	5	6	7	Klausur	Übung	Gesamt	LV-Note
Punkte max:	8	11	16	14	14	22	15	100	100	200	XXXXXX
Punkte	8	11	13	10	12	15	15	84	97	181	1,0

Weitere Hinweise:

1) Kommunikation jeder Art ist nur mit dem Dozenten gestattet.
2) Lassen Sie auf allen Blätter links, rechts, oben und unten mindestens je 2 cm Rand
3) Richtige Groß-/ Kleinschreibung ist zu beachten
4) Die Lösungen sind programmiertechnisch sinnvoll, verständlich und den jeweiligen Konventionen entsprechend auszuführen. Unnötig komplizierte oder unsinnige Programmkonstrukte führen zu Punktabzug.
5) Es sind die in der Lehrveranstaltung eingeführten Fachbegriffe zu verwenden.
6) Ausschweifende oder unnötig komplizierte Erklärungen führen zu Punktabzug.
7) Bei Aufgaben mit Ankreuzen einer von zwei Antwortmöglichkeiten führt jede falsche Anworten zur Nicht-Wertung einer richtigen Antwort. Bei Unsicherheit sollten Sie einzelne Fragen ggf. nicht beantworten.
8) Falls Sie Teile Ihrer Lösung auf Extrablätter schreiben:
   * Verwenden Sie für jede Aufgabe ein eigenes Extrablatt.
   * Auf allen Extrablätter tragen Sie oben Ihren Namen und Ihre Matrikelnummer ein.

18 **Aufgabe1 (8 Punkte)** Vervollständigen Sie die Methoden:

a) [4 Punkte]

4
```
// Berechnet den Kreisumfang nach der Formel 2 π r
public static double getKreisumfang(double radius){

        return   2 * Math.PI * radius;

}
```

b) [4 Punkte]

4
```
// Berechnet die Fläche eines Rechtecks aus den Seitenlängen a und b
public static double getArea( double a, double b          ){

        return   a * b;

}
```

11 **Aufgabe 2 (11 Punkte)** Vervollständigen Sie die Methoden:

a) [4 Punkte]

4
```
public static File[] toFileArray(String[] filenames) {
        Files[] files = new Files [filenames.length];

        for(int i=0; i< filenames.length; i++) {
                files[i] = new File (filenames[i]);

        }
        return files;
}
```

b) [7 Punkte]

7
```
// Die Methode ermittelt das Minimum und das Maximum der Werte im übergebenen Array.
// Das Minimum und das Maximum werden dann (in dieser Reihenfolge) als Array
// zurückgegeben.
public static double[] getMinmumAndMaximum(double[] a){
        double [redacted] min=a[0];
        double [redacted] max=a[0];
        double[] minMax = new double [2];

        for(int i=[redacted]; i< a.length; i++) {
                min = Math.min (min, a[i]);
                max = Math.max (max, a[i]);
        }
        minMax[0] = min;
        maxMax[1] = max;
        return minMax;
}
```

Aufgabe 3 (16 Punkte):

3.a) [7 Punkte] Geben Sie eine Implementierung an für:
*/** Gibt nur dann true zurück, falls text das Zeichen in theChar mindestens so oft enthält wie in nMin angegeben. */*

```
public static boolean containsCharacterMultiple
                                    (String text, char theChar, int nMin) {
    int counter = 0;
    for (int i = 0; i < text.length(); i++) {
        if( [redacted] text.charAt(i) == theChar) {
            counter++;
        }
    }
    if ( counter >= nMin) {
        return true;
    } else {
        return false;
    }
}
```

3.b) [9 Punkte] Gegeben Sei:

```
Set<Person> rentner   = createSetFrom("./statistik/brd/rentner.txt") ;
Set<Person> maenner   = createSetFrom("./statistik/brd/maenner.txt");
Set<Person> berliner  = createSetFrom("./statistik/brd/berliner.txt");
```

Geben Sie jeweils durch einen kurzen und präzisen Javacode an, wie die folgenden Mengen ohne Veränderung der gegebenen Set<Person> -Objekte gebildet werden können. Deklarieren Sie jeweils eine aussagekräftig benannte Variable, in der Sie das Ergebnis speichern.

a) die Vereinigungsmenge aller Renter und Berliner

```
Set<Person> vereinRuB = rentner;
vereinRuB.addAll(berliner);
```

b) die Menge aller Rentner, die nicht Männer sind

```
Set<Person> rentnerOM = rentner;
rentnerOM.removeAll(maenner);
```

c) die Menge aller Personen, die sowohl Männer als auch Berliner sind

```
Set<Person> mUb = maenner;
[redacted]
mUb.retainAll(berliner);
```

10/

Aufgabe 4 (14 Punkte):

a) [9 Punkte] Gegeben sei eine Klasse Point, mit der ein Point-Objekt initialisiert und gezeichnet werden kann. Die Klasse hat keine weiteren Methoden oder Konstruktoren:

```
public class Point implements Drawable {
      ...
      public Point(int x, int y)      ...
      public void draw(Graphics g)    ...
}
```

5 Schreiben Sie eine vollständige Klasse ColoredPoint im Package shapes, die von Point erbt, sich von Point aber dadurch unterscheidet, dass mit einer Farbe gezeichnet wird, die durch einen zusätzlichen Konstruktorparameter festlegt wird. Beachten Sie DRY!

```
package shapes;

public class ColoredPoint extends Point {
      private [redacted] Color c;

      public ColoredPoint (int x, int y, Color c){
            super (x,y);
            this.c = c;
      }

      N draw (-4)

}
```

b) [5 Punkte]

JComponent
↑
JLabel
↑
CellRenderer

5 Es sei JLabel die direkte Superklasse von CellRenderer und es sei JComponent die direkte Superklasse von JLabel. Ausserdem haben alle Klassen einen Standardkonstruktor.

Welche der folgenden aufeinanderfolgenden Codezeilen ist korrekt?
*Hinweis zur Bewertung: Punktzahl = (richtige Antworten – falsche Anworten) * 1 P*

	korrekt	fehlerhaft
CellRenderer a = new CellRenderer();	⊗	O
JLabel e = new CellRenderer();	⊗	O
CellRenderer b = new JLabel();	O	⊗
JComponent c = a;	⊗	O
a = new JComponent();	O	⊗

Aufgabe 5 (14 Punkte): Geben Sie für die im folgenden beschriebenen Methoden jeweils eine Implementierung an:

a) [5 Punkte] Von der Klasse `NamedPoint` sei bekannt, dass sie als einzigen Konstruktor `public NamedPoint(String name, double x, double y)` hat.

*/** Liest aus der Eingabe nacheinander die Zeichenkette name und die zwei Gleitkommazahlen x und y und erzeugt daraus ein NamedPoint-Objekt, das zurückgegeben wird. Sie können ohne Prüfung davon ausgehen, dass der Inhalt der Eingabe korrekt ist und der in der Eingabe enthaltene Name keinen Leerraum enthält. */*

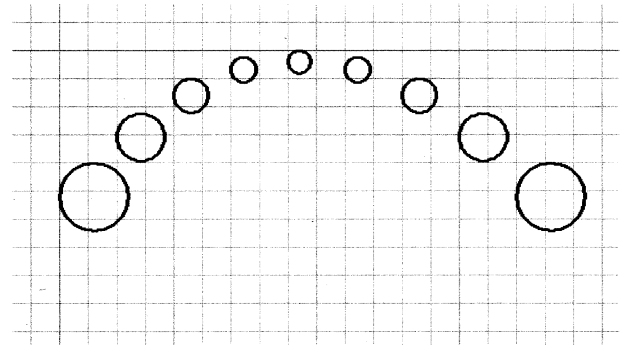
```
public static NamedPoint createNamedPoint(Scanner in) {
    return new NamedPoint(in.next(), in.nextDouble(), in.nextDouble);
}
```

b) [9 Punkte] Die Methode drawCircles(Graphics g) soll mithilfe des Graphics-Objekts mehrere **blaue** Kreise zeichnen. Ein von drawCircles(..) gezeichneter Bildschirmbereich sieht mit zusätzlichem 25-Pixel-Hilfsraster und Koordinatenachsen so aus:

Für die Koordinaten (ex, ey) der linken oberen Ecken der umfassenden Rechtecke der Kreise gilt:

ex hat für die aufeinanderfolgenden Kreise die Werte 0, 50, 100, 150, etc. im Bereich 0 <= ex <= 400

ey = (200 – ex)² / 400 .



Die Radius r eines Kreises hängt von seiner Position ex nach der Formel

r(ex) = ½ * (20 + 0,001 * (ex -200)²) ab.

```
public void drawCircles(Graphics g) {
    g.setColor(Color.BLUE);
    for(int i=0; i<=400; i+=50) {
        int ex = i;
        int ey = (int)((200-ex)(200-ex)/400);
        int d = (int)(20 + 0.001 * (ex-200)(ex-200));
        g.drawOval(ex, ey, d, d);
    }
}
```

→ doppelte Schleifenzähler

15/

Aufgabe 6 (22 Punkte) Gegeben sei:

```
public class Joghurt {
	protected String marke;
	protected int gewicht;

	public Joghurt(String marke, int gewicht) {
		this.marke   = marke;
		this.gewicht = gewicht;
	}

	public String getProductCategorie(){
		return "Joghurt";
	}

	public String toString() {          // vgl. Aufgabe a)

	   return marke + "␣" + getProductCategorie() + ",␣" + gewicht + "g";


	}
}

// Hinweis: Frucht ist eine Immutable-Klasse
public final class Frucht {
	public final String sorte;
	public Frucht(String sorte) {
		this.sorte = sorte;
	}
}
```

a) [4 Punkte] Ergänzen Sie die Methode toString() von Joghurt so, dass die Ausgabe eines `Joghurt`-Objekts (*im Bsp.*: Marke *Jalmj*) in der Art `System.out.println(joghurt);` eine Ausgabe von Marke, Produktkategorie und Gewicht nach dem Muster **`Jalmj Joghurt, 480g`** erzeugt.

4

b) [18 Punkte] Geben Sie die Deklaration einer Klasse `Fruchtjoghurt` an, die von `Joghurt` erbt. Die Klasse `Fruchtjoghurt` besitzt eine Instanzvariable vom Typ `Frucht` und eine Instanzvariable vom Typ `double`, in der der `fruchtanteil` (angegeben in %) gespeichert wird.

11

Die Klasse `Fruchtjoghurt` soll folgende Konstruktoren haben:

(b.k.1) einen vollständigen Konstruktor,

(b.k.2) einen Konstruktor, mit drei Parametern der Typen Joghurt, String (zur Angabe der fruchtsorte) und double.

(b.k.3) einen Standardkonstruktor, der als Standard die Marke Fampf, Fruchtsorte Kirsch, 128g mit 14% Fruchtanteil verwendet.

Die Ausgabe eines `Fruchtjoghurt`-Objekts (*im Bsp.*: Marke *Fampf* und Produkt-Kategorie *Kirsch-Joghurt*) in der Art `System.out.println(fruchtjoghurt);` soll eine Ausgabe von Marke, Produktkategorie, Gewicht und Fruchtanteil nach dem Muster

`Fampf Kirsch-Joghurt, 128g (14,00% Frucht)` erzeugen.

<u>Vermeiden Sie unnötige Codevervielfachungen, beachten Sie DRY!</u>

```
class Fruchtjoghurt extends Joghurt {

    // Attribute
    private Frucht f;
    private double fAnteil;
```

und was machen die Subklassen?

```
    // (b.k.1) vollständiger Konstruktor
    public Fruchtjoghurt (String marke, int gewicht, Frucht f, double fAnt
        super (marke, gewicht);
        this.f = f;
        this.fAnteil = fAnteil;
    }
    // (b.k.2) Konstruktor ... (vgl. Aufgabe)
    public Fruchtjoghurt (Joghurt j, String fSorte, double fAnteil) {
        this (j.marke, j.gewicht, new Frucht (fSorte), fAnteil);
    }

    // (b.k.3) Standardkonstruktor
    public Fruchtjoghurt () {
        this ("Fampf", 128, new Frucht ("Kirsch"), 14);
    }

    // .toString Methode
    public String toString () {
        return String.format ("%s %s - %s, %d g (%3.2f %% Frucht)",
            marke, f.sorte, getProductCategorie(), gewicht, fAnteil);
    }
```

Ah, stimmt! ☺

super

überschreiben

getP... C... ()

(-4)

```
}
```

Aufgabe 7 (15 Punkte):
*Hinweis zur Bewertung: Punktzahl = (richtige Antworten – falsche Anworten) * 1,5 P*

Geben Sie ohne Begründung an, welche der folgenden Aussagen zutreffen und welche nicht zutreffen:

	trifft zu	trifft nicht zu
Eine abstrakte Klasse kann Konstruktoren haben	⊗	O
Jede Bauplanklasse hat mindestens eine statische Methode	O	⊗
Eine abstrakte Klasse muss abstrakte Methoden enthalten	O	⊗
super() ist ein Aufruf der Methode super	O	⊗
Interfaces sind primitive Typen	O	⊗
Stringvergleiche mit == können ein falsches Ergebnis liefern	⊗	O
Jede Klasse muss toString() überschreiben	O	⊗
Immutables des Typs J können als Elemente in Listen des Typs ArrayList<J> gespeichert werden	⊗	O
Collection ist Supertyp von Set	⊗	O
ArrayList ist Subtyp von Collection	⊗	O

ohne Antwort / richtig / falsch / bewertet:

Punkte: 15