Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
Prof. Dr. Alice McHardy
Robin Küppers
Sebastian Konietzny
Aaron Weimann

WS 2013/2014
10.12.2013
Übungsblatt 7

Einführung in die Informatik I
Übungen zur Vorlesung
Musterlösung

Die Lösungen der zu bearbeitenden Aufgaben müssen spätestens bis **Dienstag**, den **17.11.2013, 11:59 Uhr** über https://ilias.uni-duesseldorf.de abgegeben werden. Bitte achten Sie immer auf einen vollständigen und nachvollziehbaren Lösungsweg, da ansonsten Punkte abgezogen werden können! Vergessen Sie bitte nicht auf Ihrer Abgabe Ihren Namen und Ihre Matrikelnummer zu vermerken (bei Gruppenabgaben müssen sämtliche Mitglieder der Gruppe aufgeführt werden).

7.1 Klassendesign in Java (6 Punkte)

1. Schreiben Sie eine Klasse `Vektor3d`, die einen reellen Vektor (x, y, z) im 3-dimensionalen Raum beschreibt. Es soll möglich sein, Vektoren zu addieren, zu subtrahieren, mit einem reellen Skalar zu multiplizieren und die Länge zu berechnen. (2 Punkte)

2. Schreiben Sie eine Klasse `VektorN`, die einen reellen Vektor im N-dimensionalen Raum beschreibt. Es soll, wie in der ersten Teilaufgabe, möglich sein, Vektoren zu addieren, zu subtrahieren, mit einem reellen Skalar zu multiplizieren und die Länge zu berechnen. Gehen Sie davon aus, dass immer Vektoren mit gleicher Dimensionalität verwendet werden. **Zur Erinnerung:** Die Länge eines Vektors v (der Dimensionalität n) ist definiert durch:

$$|v| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

(4 Punkte)

Musterlösung

1.
```
public class Vektor3d extends VektorN {

    public Vektor3d(double x, double y, double z) {
        super(new double[] { x, y, z });
    }

}
```

```
public class VektorN
{
private  double[] coords;

public double[] getCoords()
{
    return(this.coords);
}

public VektorN(double[] coords )
{
    this.coords = coords;
}

public String toString()
{
    String outString = "";
    for(int i=0;i<this.coords.length; i++)
    {
        outString += this.coords[i] + " ";
    }
    return outString;
}
public VektorN add(VektorN summand)
{
    checkCompatibility(summand);
    for(int i=0;i<summand.getCoords().length; i++)
    {
        this.coords[i] = this.coords[i] + summand.getCoords()[i] ;
    }
    return this;
}
```

```
public VektorN subtract(VektorN subtractant)
{
    checkCompatibility(subtractant);
    for(int i=0;i<subtractant.getCoords().length; i++)
    {
        this.coords[i] = this.coords[i] - subtractant.getCoords()[i] ;
    }
    return this;
}

public double  length()
{
    double sum = 0;
    for(int i=0; i<this.coords.length; i++)
    {
        sum+=Math.pow(this.coords[i], 2);
    }
    return Math.sqrt(sum);
}


public VektorN s_product (double scalar)
{
    for(int i=0; i<this.coords.length; i++)
    {
        this.coords[i] = this.coords[i]*scalar;
    }
    return this;
}
private  void checkCompatibility(VektorN v)
{
    if(!(v.getCoords().length == this.coords.length))
    System.out.println("not compatible");
    System.exit(1);
}
}
```

7.2 Klassen und Instanzen (6 Punkte)

Seien folgende Java-Klassen gegeben:

```
public class Klasse1 {

    private static int x1;
    private int x2;
    public static int x3 = 0;
    public int x4 = 0;

    public void Klasse1() {
        x1 = 0;
        x2 = 0;
    }

    public static void f1() {
        this.x1 = x1 + 1;
    }

    public void f2() {
        this.x2 = x2 + 2;
    }

    private static void f3() {
        x3++;
    }

    private void f4() {
        x4--;
    }

}

public class Klasse2 {

    private Klasse1 klasse1;

    public Klasse2() {
        this.klasse1 = new Klasse1();
    }

}
```

1. Welche der folgenden Behauptungen ist richtig bzw. falsch? Begründen Sie! (5 Punkte)

   (a) `x1` ist eine Klassenvariable von `Klasse1`.

   (b) `x2` ist keine Instanzvariable von `Klasse1`.

   (c) `f1` ist eine Instanzmethode von `Klasse1`.

   (d) `f2` ist keine Instanzmethode von `Klasse1`.

   (e) `x1, x2` werden im parameterlosen Konstruktor von `Klasse1` initialisiert.

   (f) `x3` wird in jeder Instanz von `Klasse1` initialisiert.

   (g) Die Methode `Klasse1` hat eine ungültige Signatur.

   (h) Die Methode `f1` enthält gültigen `Java`-Code.

   (i) `x1` wird für jede Instanz von `Klasse1` initialisiert.

   (j) Die Methode `f2` enthält gültigen `Java`-Code.

2. Auf welche Methoden und Variablen von `Klasse1` kann innerhalb von `Klasse2` zugegriffen werden und auf welche nicht? Wird für den Zugriff eine Instanz von `Klasse1` benötigt? Begründen Sie kurz. (1 Punkt)

Musterlösung

1. (a) richtig

   (b) falsch. `x2` ist keine Instanzvariable, sondern eine statische Variable der Klasse `Klasse1`.

   (c) falsch. `f1` ist eine Klassenmethode von `Klasse1`.

   (d) falsch. `f2` ist eine Instanzmethode von `Klasse1`.

   (e) falsch. `x1,x2` werden in der Methode `Klasse1` initialisiert.

   (f) falsch. `x3` ist eine Klassenvariable und wird nur einmal initialisiert.

   (g) falsch. Die Methodensignatur ist gültig.

   (h) falsch. Ein Zugriff auf `x1` mit dem Schlüsselwort `this` ist ungültig, weil es sich bei `x1` um eine Klassenvariable handelt.

   (i) falsch. Die Klassenvariable `x1` wird nur dann initialisiert, wenn die Methode `Klasse1()` von einer Instanz aufgerufen wird.

   (j) richtig.

2. Die folgenden Variablen und Methoden sind innerhalb von `Klasse2` (nicht) sichtbar:

	sichtbar	nicht sichtbar
Variablen	`x3, x4`	`x1, x2`
Methoden	`f1, f2`	`f3, f4`

   Begründung: Es kann auf Variablen und Methoden zugegriffen werden, die mit dem Schlüsselwort `public` versehen wurden. Alle Variablen und Methoden, die als `private` deklariert wurden, sind von `Klasse2` aus nicht sichtbar. Für Methoden und Variablen, die als `static` deklariert wurden, wird keine Instanz der Klasse `Klasse2` benötigt.

7.3 Objektreferenzen (6 Punkte)

1. Was gibt das folgende Codefragment aus? Erklären Sie, warum die entsprechende Ausgabe erfolgt. (2 Punkte)

```
int[] arr1 = new int[] { 1, 2, 3 };
int[] arr2 = arr1;
arr1 = new int[] { 4, 5 };
System.out.println(arr2.length);
```

2. Was gibt das folgende Codefragment aus? Erklären Sie, warum die entsprechende Ausgabe erfolgt. (2 Punkte)

```
String s1 = new String("Eine Zeichenkette");
String s2 = new String("Eine Zeichenkette");
if (s1 == s2) System.out.println("true");
else System.out.println("false");
```

3. Was gibt das folgende Codefragment aus? Erklären Sie, warum die entsprechende Ausgabe erfolgt. (2 Punkte)

```
int[] a1 = new int[] { 1, 2, 3 };
int[] a2 = new int[] { 4, 5, 6 };
a1 = a2;
a1[1] = 3;
a2[2] = 2;
a2 = a1;
for (int i = 0; i < a2.length; i++) {
    System.out.print(a2[i] + " ");}
```

Musterlösung

1. Ausgabe: 3
   In der ersten Zeile wird `arr1` {1,2,3} zugewiesen. Anschließend wird `arr2` geändert, so dass die Variable auf das gleiche Array wie `arr1` verweist. Bei der Änderung von `arr1` in der dritten Zeile, bleibt die Referenz von `arr2` aber erhalten.

2. Ausgabe: false
   Es handelt sich um Objekte, die unter verschiedenen Adressen im Speicher liegen. Mit dem Gleichheitsoperator (==) werden die Speicheradressen (Referenzen) der String-Objekte verglichen, nicht die Zeichenketten selbst. Um stattdessen die Zeichenketten zu vergleichen, müsste man die Instanzmethode *equals* der Stringobjekte verwenden.

Anmerkung: Wenn man die Variablen `s1, s2` in abkürzender Schreibweise initialisiert (z.B. `s1 = "Eine Zeichenkette"`), dann würde das Code-Fragment `true` zurückliefern, weil Java Zeichenketten im Speicher wiederverwendet (falls möglich), um Resourcen zu schonen. Dieser Mechanismus wird durch die Verwendung von `new` umgangen.

3. Ausgabe: 4 3 2
   Nach der dritten Zeile verweisen `a1` und `a2` auf dasselbe Objekt.

7.4 Grafikprogrammierung mit Java (7 Punkte)

Orientieren Sie sich zur Lösung dieser Aufgabe an `Grayscale.java` aus der Vorlesung.
`Grayscale.java` und alle von `Grayscale.java` referenzierten Klassen finden Sie in Ilias im Vorlesungsmaterial.

Hinweis zur Abgabe: Geben Sie die Lösung als Zip-Archiv mit dem Source-Code der beiden Programme `Invert.java` und `Blend.java`, allen von diesen Programmen benutzten Klassen, sowie den Bildern `orangutan.jpg` bzw. `hund.jpg` und `katze.jpg` ab.

Bevor Sie das Zip-Archiv packen, überprüfen Sie, ob Ihre Programme kompilieren und den gewünschten Output generieren. Fügen Sie keine `.class` Dateien oder Output-Dateien in das Archiv mit ein.

1. Schreiben Sie ein Programm `Invert`, das ein Bild einliest, invertiert und anschließend den Output abspeichert. Implementieren Sie dazu eine Methode `public static Colour invert(Colour col)`, die für eine beliebige Farbe `col` das Inverse zurückliefert. Nutzen Sie dafür die folgende Formel:

   $$x_{invertiert} = 255 - x_{original} \text{ für } x \in \{r, g, b\}$$

   Implementieren Sie eine Main-Methode, in der Sie das Bild `orangutan.jpg` invertieren und unter `orangutan_invert.jpg` abspeichern. (3 Punkte)

2. Schreiben Sie ein Programm `Blender`, das zwei gleichgroße Bilder einliest, miteinander verschmilzt und den Output abspeichert. Schreiben Sie dazu eine Methode `public static Colour blend(Colour c1, Colour c2, double alpha)`. Diese Methode erhält eine Farbe aus Bild 1, eine Farbe aus Bild 2 und einen reellen Parameter $\alpha \in [0, 1]$, der angibt, wie dominant Bild 1 bzw. Bild 2 im Gesamtbild ist. Verwenden Sie die folgende Formel:

   $$x_{gemischt} = \alpha \cdot x_{bild1} + (1 - \alpha) \cdot x_{bild2} \text{ für } x \in \{r, g, b\}$$

   Implementieren Sie eine Main-Methode, in der Sie die Bilder `hund.jpg` und `katze.jpg` einlesen, mit $\alpha = 0.5$ mischen und unter `hund_katze.jpg` wieder abspeichern. (4 Punkte)

Musterlösung

1.
```
public class Invert {
    public static Colour toInverse(Colour c) {
        int r = c.getRed();
        int g = c.getGreen();
        int b = c.getBlue();
        return new Colour(255-r, 255-g, 255-b);
    }

    public static void main(String[] args) {
        Picture pic = new Picture("orangutan.jpg");
        Picture opic = new Picture("orangutan.jpg");
        for (int x = 0; x < pic.width(); x++) {
            for (int y = 0; y < pic.height(); y++) {
             Colour col  =  (Colour) pic.get(x, y);
             Colour inv_col = toInverse(col);
             pic.set(x, y, inv_col);
            }
       }
       //opic.show();
       //pic.show();
       pic.save("orangutan_inv.jpg");
    }
}
```

2.
```
public class Blend {

    public static Colour blend(Colour c1, Colour c2, double a) {
        int r = c1.getRed();
        int g = c1.getGreen();
        int b = c1.getBlue();
        return(new Colour((int) (a*r+(1-a)*c2.getRed()),
        (int) (a*g + (1-a)*c2.getGreen()), (int) (a*b + (1-a)*c2.getBlue())));
    }
```

```
    public static void main(String[] args) {
        Picture pic_1 = new Picture("hund.jpg");
        Picture opic_1 = new Picture("hund.jpg");
        Picture pic_2= new Picture("katze.jpg");
        Picture opic_2 = new Picture("katze.jpg");
        for (int x = 0; x < pic_1.width(); x++) {
            for (int y = 0; y < pic_1.height(); y++) {
             Colour col1  =  (Colour) pic_1.get(x, y);
             Colour col2 =  (Colour) pic_2.get(x, y);
             pic_1.set(x, y, blend(col1,col2,0.5));

            }
       }
       //opic_1.show();
       //pic_2.show();
       //pic_1.show();
       pic_1.save("hund_katze.jpg");


    }
}    }

}
```