Fakultät IV - Elektrotechnik und Informatik
Algorithmen und Datenstrukturen SoSe 2016
Brock / Deimel / Foucard / Mabrouk / Sieverling

Aufgabenblatt 2

letzte Aktualisierung: 10. May, 15:20 Uhr

Ausgabe: 08.05.2016
Abgabe: 18.05.2016 19:59

Thema: *Klassen, Objekte*

Wichtige Ankündigungen

- Unit tests dürfen geteilt werden. Lösungen dürfen auf keinen Fall geteilt werden! Wir testen auf Plagiate.
- Das Abgabe-Repository steht unter `https://teaching.inet.tu-berlin.de/tubitauth/svn/algodat-sose2016` mittels eurem tubit-account zur Verfügung. Im Ordner /Material/Tutorien/tXX werden die Tutoren das Material aus den Tutorien hochladen.
- Erinnerung: Registriere dich (für die meisten: in QISPOS) falls du das noch nicht gemacht hast! Deadline ist am 15.5.2016, es gibt keine spätere Möglichkeit zum registrieren!
- Es gibt ein neues Tutorium am Montag von 10 bis 12 Uhr. Kommt dorthin, falls ihr wechseln wollt.

Abgabe

Die folgenden Dateien müssen für eine erfolgreiche Abgabe im svn Ordner `Tutorien/txx/Studierende/deinname@TU-BERLIN.DE/Abgaben/` eingecheckt sein:

Geforderte Dateien:	
`Blatt02/src/Cuboid.java`	Aufgabe 1.7
`Blatt02/src/Cube.java`	Aufgabe 1.8
`Blatt02/src/Picture.java`	Aufgabe 2

Als Abgabe wird die nur neueste Version im svn gewertet.

1. Aufgabe: Vererbung

Vererbung ist ein grundlegendes Konzept der Objektorientierung und hat deshalb große Bedeutung für die Softwareentwicklung, Die folgenden Aufgaben sollen euch die Begriffe Basisklasse, Ableitung und abstrakte Klasse näher bringen.

1.1. (Tut) Modifikatoren Was bedeuten die Modifikatoren `public`, `private`, `protected`, `static`?

1.2. (Tut) Annotationen Mit Annotationen können Metadaten in den Quelltext eingebunden werden. Was bedeutet `@Deprecated` und `@Override`?

1.3. (Tut) Hierarchie Erstellt eine Hierarchie von 3D-Körpern - Würfel, Tetraeder, Quader, Körper und erklärt dabei Generalisierung und Spezialisierung.

1.4. (Tut) Die Basisklasse Body Erstellt neue Klassen `Body` und `Tetrahedron`. Versucht nun folgenden Java-Code auszuführen:

```
1 Body b = new Tetrahedron();
```

Wieso ist das momentan noch nicht möglich?
Lasst jetzt `Tetrahedron` von `Body` erben und führt den Code noch einmal aus.

1.5. (Tut) Body als abstrakte Klasse Wandelt `Body` in eine abstrakte Klasse um gebt dieser Klasse die zwei abstrakten Methoden `double calculateVolume()`, `double calculateSurface()`.

1.6. (Tut) Tetraeder Lasst `Tetrahedron` von `Body` erben. Kompiliert anschließend `Tetrahedron` und beseitigt alle Fehlermeldungen.

1.7. Quader (20 Punkte) Erstellt eine Klasse `Cuboid`, die ein Quader repräsentiert. Beachtet dabei die Vererbungshierarchie, die in Aufgabe 1.3 entwickelt wurde. Die Javadoc-Kommentare der Klasse `Cuboid` sollen die Darstellung des Quaders möglichst genau spezifizieren. Neben der Implementierung der abstrakten Methoden soll `Cuboid` mindestens einen leeren Konstruktor und einen parametrisierten Konstruktor besitzen.

1.8. Würfel (20 Punkte) Erstellt eine Klasse `Cube`. Diese soll einen Würfel darstellen. Beachtet auch hier die Vererbungshierarchie, die in Aufgabe 1.3 entwickelt wurde. Erstellt auch für `Cube` sowohl einen leeren als auch einen parametrisierten Konstruktor.

2. Aufgabe: Bildreparatur

Zur Speicherung von Bildern können Matrizen sehr gut genutzt werden. Ein RGB Bild besteht aus width×height Pixeln, welche jeweils drei Farbwerte enthalten.
Wir betrachten die folgende Darstellung:

```
1  public class RGBColor {
2
3          private int red;
4          private int green;
5          private int blue;
6
7          public RGBColor(int r, int g, int b){
8                  this.red=r;
9                  this.green=g;
10                 this.blue=b;
11         }
12 ...
13
14
15 public class Picture {
16
```

```
17          private RGBColor imageMatrix[][];
18          private int width;
19          private int height;
20
21          /**
22           * initialize a picture by opening given path
23           * @param path path of *.bmp picture
24           */
25          public Picture(String path){
26                  openAndSetPicture(path);
27          }
28  ...
```

2.1. (Tut) Arrays Was sind die Unterschiede zwichen einem Array und einer ArrayList?

2.2. (Tut) Bilderzeugung Erweitert die Klasse Picture um die Methode `public void createWhitePicture()`, die ein weißes Bild generiert. Weiße Pixel werden in RGB als (255,255,255) representiert.

```
1       public Picture(int height, int width){
2         this.height = height;
3         this.width = width;
4         createWhitePicture();
5       }
6       public void createWhitePicture(){
7           this.imageMatrix = new RGBColor[this.width][this.height];
8           for (int w=0; w< this.width; w++){
9               for(int h=0; h< this.height; h++){
10                  //set this colors in picture
11                  this.imageMatrix[w][h] = new RGBColor(255, 255, 255);
12              }
13          }
14      }
```

2.3. Bildreparatur (30 Punkte) Bei dem Kopiervorgang unserer Bilder sind Fehler passiert und einige Pixel sind verloren gegangen. Die verlorenen Pixel werden nun als weiß(255,255,255) repräsentiert. Erweitert die Klasse Picture um die Methode `public void repairPicture()`, die ein Bild repariert, indem sie die weißen Pixel anhand der Mittelung der Nachbarschaftsfarben rekonstruiert.

Hinweis: Als Nachbarn verstehen wir alle 8 direkt angrenzenden Pixel. Nachbarn die weiß sind, oder nicht existieren sollen ignoriert werden.

2.4. Bilddrehung 90 Grad rechts (20 Punkte) Erweitert die Klasse Picture um die Methode `public void rot90DegRight()`, die ein Bild um 90 Grad im Uhrzeigersinn dreht.

Hinweis: Die Methode soll auch für nicht-symmetrische Bilder (width $\neq$ height) funktionieren. In solchen Fällen, wird ein Bild mit vertauschten Dimensionen (height $\times$ width statt width $\times$ height) erzeugt.

2.5. Bilddrehung 180 Grad (10 Punkte) Erweitert die Klasse Picture um die Methode `public void rot180Deg()`, die ein Bild um 180 Grad dreht.