Fakultät IV - Elektrotechnik und Informatik
Algorithmen und Datenstrukturen SoSe 2017
Brock / Mabrouk / Sieverling / Stahl

Aufgabenblatt 1

letzte Aktualisierung: 28. April, 17:04 Uhr

Ausgabe: 28.04.2017 19:59
Abgabe: 10.05.2016 19:59

Thema: *Klassen, Objekte, Methoden, Iteratoren, Listen, Unit testing*

Wichtige Ankündigungen

- Bestimmte Unit Tests in dieser Aufgabe sind Teil der Abgabe. Diese dürfen nicht geteilt oder weitergegeben werden!! Unit Tests die nicht Teil der Abgabedateien sind (SphereTest.java) dürfen immer geteilt werden.
- Aufgrund des Feiertages fallen die Montags-Tutorien aus. Wir werden ein Tutorium aufnehmen und auf ISIS hochladen.
- Das Abgabe-Repository steht unter
  `https://teaching.inet.tu-berlin.de/services/svn/algodat-sose2017/`
  mittels eurem tubit-account zur Verfügung,
- Erinnerung: Registriere dich (für die meisten: in QISPOS) falls du das noch nicht gemacht hast!
- All Übungen sind in Einzelarbeit zu erledigen. Kopiert niemals Code und gebt auch keine Kopien heraus. Finden wir ein Plagiat führt das zum nichtbestehen des Kurses für beide StudentInnen.

Abgabe

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

Geforderte Dateien:	
`Blatt01/src/Sphere.java`	Aufgabe 2.6
`Blatt01/src/ArrayCheck.java`	Aufgabe 3.3
`Blatt01/tests/ArrayCheckTest.java`	Aufgabe 3.4

Als Abgabe wird nur die neueste Version im svn gewertet.

1. Aufgabe: Einführung

1.1. (Tut) Arbeiten mit Eclipse Importiert das Hausaufgaben-Projekt in Eclipse. Geht dazu wie folgt vor:

1. Entpackt das Vorgabenarchiv und öffnet Eclipse.
2. Klickt mit der rechten Maustaste in den Package Explorer und klickt auf Import...
3. Wählt unter General Existing Projects into Workspace.
4. Wählt unter Select root directory den Ordner aus, in welchen ihr das Vorgabenarchiv entpackt habt.
5. Klickt auf Finish.

In dem Eclipse-Projekt findet ihr nun ein Programmskelett als Vorgabe, in welche ihr eure Lösung der Hausaufgabe schreibt. Seht euch die Dateien im Ordner `src/` an. Öffnet die Datei `Point.java` für die nächste Teilaufgabe

1.2. (Tut) Java erklären Was bedeuten die folgenden Begriffe oder Anweisungen?

1. `.java Datei vs. .class Datei`
2. `Klasse vs. Objekt/Instanz`
3. `public static void main(String[] arguments)`
4. `Primitive Datentypen vs. Referenzen`
5. `Point p = new Point(1, 2, 3);`
6. `p.y = p.y + 1; p.y + 1; p.getY()`
7. `p.toString();`

2. Aufgabe: Objektorientierung

2.1. (Tut) Klasse Quadrat programmieren Implementiert eine Klasse, welche Quadrate repräsentiert. Ein Quadrat ist durch seine Seitenlänge eindeutigt bestimmt.

Hinweis: Beachtet die Java-Namenskonventionen!

- Klassennamen sind Substantive und beginnen mit Großbuchstaben (z.B. `Student`).
- Variablennamen sind kurze Substantive und beginnen mit Kleinbuchstaben, jedes weitere Wort beginnt mit einem Großbuchstaben (z.B. `dateOfBirth`, `camelCase`).
- Konstanten werden komplett mit Großbuchstaben geschrieben (z.B. `RED`).
- Methodennamen sind Verben und beginnen mit Kleinbuchstaben, jedes weitere Wort beginnt mit einem Großbuchstaben (z.B. `countElements`).
- Programme sind mit Javadoc-Kommentaren zu versehen.
- Inhalte eines Blocks werden mit vier Leerzeichen oder einem Tabulator eingerückt.

Mehr über die *Java Code Conventions* findet ihr unter
`http://www.oracle.com/technetwork/java/codeconvtoc-136057.html`.

2.2. (Tut) Eigenschaften und Konstruktor Schreibt einen Konstruktor für die Quadrat-Klasse. Dem Konstruktor soll als Argument die Seitenlänge des Quadrats übergeben werden.

2.3. (Tut) Methoden Schreibt eine Methode `scaleBy`, mit welcher man das Quadrat vergrößern oder verkleinern kann. Dazu wird der gesuchten Methode ein Faktor s als Parameter übergeben, welcher mit der Seitenlänge multipliziert wird.

2.4. (Tut) Objekte der Quadrat-Klasse instanziieren Erstellt eine Klasse `SquareObjects`; in der ihr zwei Objekte der Quadrat-Klasse instanziiert und eines der Quadrate um den Faktor 2 vergrößert.

2.5. Klasse `Sphere` implementieren (40 Punkte) Im Eclipse-Projekt zur Hausaufgabe findet ihr die Klasse `Sphere`, welche verschiedene leere Methoden enthält. Eure Aufgabe ist es, die folgenden Methoden zu implementieren:

1. `Sphere(int x, int y, int z, double r)` ist ein Konstruktor, dem x-, y-, z-Koordinate des Mittelpunkts sowie der Radius r der Kugel übergeben werden.
2. `Sphere(Point c, double r)` ist ein zweiter Konstruktor, dem der Mittelpunkt als Point-Objekt übergeben wird, sowie der Radius r.
3. `int getX()` gibt die x-Koordinate der Kugel zurück.
4. `int getY()` gibt die y-Koordinate der Kugel zurück.
5. `int getZ()` gibt die z-Koordinate der Kugel zurück.
6. `double getRadius()` gibt den Radius der Kugel zurück.
7. `double calculateDiameter()` berechnet und gibt den Durchmesser der Kugel zurück.
8. `double calculateSurfaceArea()` berechnet und gibt die Oberfläche der Kugel zurück.
9. `double calculateVolume()` berechnet und gibt das Volumen der Kugel zurück.

Hinweis:
- Die Methode `Math.pow(double a, double b)` (Exponent) und die Konstante `Math.PI` (die Zahl π) sind eure Freunde.
- Java unterscheidet zwischen Integer und Fließkomma-Division (Überprüft beispielsweise das Ergebnis von $1/2$ und $1.0/2.0$.). Überzeugt Euch vom Unterschied und verwendet den richtigen Datentyp.
- Verwendet für die Bearbeitung dieser Aufgabe die Klasse `Point.java` aus den Vorgaben, verändert sie aber nicht.
- Die Implementierung muss Null Pointer in Methodenparametern nicht extra behandeln.

2.6. (Tut) Testen mit jUnit Öffnet die Datei `tests/SphereTest.java`. Die Datei enthält einfache Tests, die euer Programm auf jeden Fall bestehen muss. Wir werden mindestens einmal täglich eure letzte Version im svn mit unseren eigenen vollständigen Tests überprüfen. Ihr habt die Aufgabe bestanden, wenn ihr bei den Tests volle Punkte bekommt und euren Code ausreichend kommentiert.

Führt die Tests mit Eclipse aus und betrachtet die Anzeige.

Hinweis: Ihr könnt auch selbst weitere Tests in der Klasse `SphereTest.java` hinzufügen, um Euch die Entwicklung der Sphere-Klasse zu erleichtern.
Es gibt verschiedene `assert`-Anweisungen in jUnit, wovon die einfachste `assertTrue` ist, es gibt z.B. noch `assertFalse`, `assertEquals`, `assertNull` u.v.m. Ihr findet weitere Informationen zu jUnit unter http://junit.org/.

3. Aufgabe: Arrays

Als Arrays (dt. „Felder“), werden zusammenhängende Speicherbereiche gleichen Typs bezeichnet, auf einzelne Elemente kann direkt über Indices zugegriffen werden.

3.1. (Tut) Einführung Initialisiert ein ArrayList von Zahlen und weist ihm Werte zu. Besprecht den Unterschied der folgenden zwei Lösungen.

```
1 int[] a = new int[3]; //Legt array der Groesse 3 an
2 a[0] = 10;
3 a[1] = 20;
4 a[2] = 30;
5 System.out.println(a[2]); //gibt 30 zurueck
```



```
1 ArrayList<Integer> a = new ArrayList<Integer>(); //legt dynamisches Array an
2 a.add(10);
3 a.add(20);
4 a.add(30);
5 System.out.println(a.get(2)); //gibt 30 zurueck
```

3.2. (Tut) Ablaufen eines Arrays Besprecht die Methode `calcExamAverage`, die ein Array von Klausurnoten bekommt, die Durchschnittsnote errechnet und diese zurück gibt.

3.3. Reihen als Arrays (30 Punkte) In dieser Aufgabe sollt ihr die Klasse `ArrayCheck` implementieren, welche Methoden enthält, die mit Arrays arbeiten.

Genauer soll `ArrayCheck` dabei folgende 2 Methoden enthalten (nutzt hier die Vorgabe von `ArrayCheck`):

1. `boolean allDivisibleBy(ArrayList<Integer> arr, int divisor)`, die überprüft, ob alle Zahlen im übergebenen Array `arr` ganzzahlig durch die übergebene Zahl `divisor` teilbar sind. Hinweise:
   - Ist das übergebene Array nicht existent (d.h. die Referenz hat den Wert `null`) oder leer, soll die Methode `false` zurückgeben.
   - Keine Zahl ist durch 0 teilbar.
   - 0 ist durch alle Zahlen teilbar (ausser durch 0).
2. `boolean isAnagram(ArrayList<Char> arr1, ArrayList<Char> arr2)`, die überprüft, ob die beiden übergebenen Character-Arrays ein Anagramm sind.
   Hinweis:
   - Zwei Zeichenketten sind ein Anagram, wenn die eine Zeichenkette nur durch Umstellen der Buchstaben der anderen Zeichenkette generiert werden kann.
   - Anagramme ignorieren Gross- und Kleinschreibung sowie Leerzeichen und Sonderzeichen.
   - Für leere oder nicht-existente Arrays soll die Methode `false` zurückgeben.
   - mit dem Befehl `Collections.sort(arr)` kann ein Array sortiert weden
   - Nützlich sind auch `Character.toLowerCase(x)` und `Character.isLetter(x)`

   Beispiele:
   die Methode gibt für folgende Paare `true` zurück:
   `([abc],[cba])` `([AlgoDat],[Gala Tod!])` `([Eclipse],[PC, Leise])`
   die Methode gibt für folgende Paare `false` zurück:
   `([aaa],[aa])` `([aab],[ba])`

3.4. JUnit-Test für ArrayCheck (30 Punkte) Erweitert die JUnit-Testklasse `ArrayCheckTest`, die Unit-tests für die Klasse `ArrayCheck` enthält. Schreibt für jede Methode von `ArrayCheck` genau einen Unit-test. Achtet darauf, dass eure Tests weder fehlerhafte Implementierungen von `ArrayCheck` akzeptieren, noch korrekte Implementierungen verwerfen.

Haltet euch bei der Implementierung der Unit-tests an die zu testendende Verhaltensbeschreibung in der Vorgabe! Die Vorgabe findet ihr in `Aufgaben/Blatt01/Vorgaben/tests/`.