
Programmieren in Java
<http://proglang.informatik.uni-freiburg.de/teaching/java/2017/>

shapes*Geometrische Figuren (korrigiert: 2017-06-15)*

Woche 07 Aufgabe 2/2

Herausgabe: 2017-06-13

Abgabe: 2017-06-23

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.Project **shapes**Package **shapes**

Klassen

V2
<pre>public V2(double x, double y) public double getX() public double getY()</pre>

Box
<pre>public Box(V2 upperLeftCorner, V2 dimensions) public V2 getUpperLeftCorner() public V2 getDimensions()</pre>

<i>Shape</i>
<pre>public boolean contains(V2 point) public Shape move(V2 displacement) public Box boundingBox()</pre>

Shapes
<pre>public static Shape makeEllipse(V2 center, V2 radii) public static Shape makeRectangle(V2 upperLeftCorner, V2 dimensions) public static Shape makePicture(List<Shape> shapes)</pre>

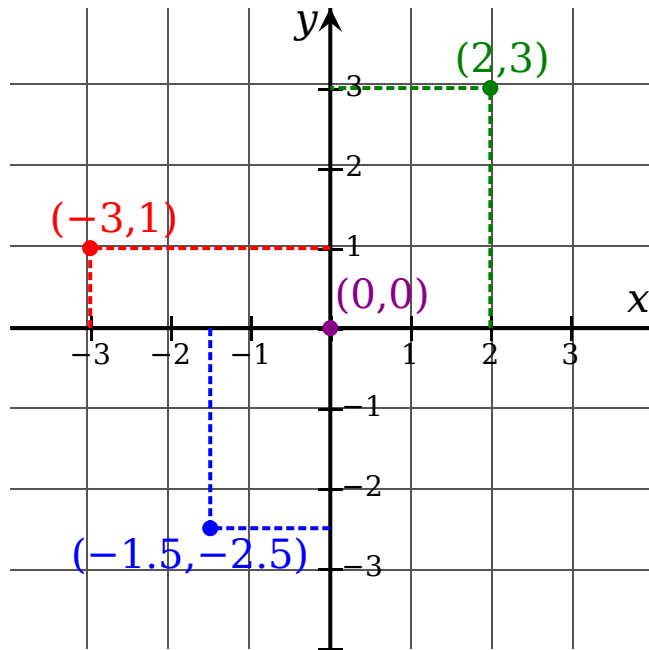
In dieser Aufgabe sollen verschiedene geometrische Figuren in einem Interface *Shape* zusammengefasst werden.

Zunächst werden zwei Hilfsklassen benötigt:

- V2 ist eine Klasse für 2D Vektoren. Sie hat jeweils Getter-Methoden für die x und y Koordinate.
- Box ist eine rechteckige Box. Sie hat einen Getter für den Ortsvektor ihrer linken, oberen Ecke und einen Getter für ihre *Maße*. Die Maße werden als V2-Vektor zurückgegeben, der die Breite in der x -Koordinate und die Höhe in der y -Koordinate angibt. Höhe und Breite sollen positiv, d.h. > 0 , sein.

Die Hilfsklassen sind im Skelett bereits implementiert.

Wir benutzen in dieser Aufgabe kartesische Koordinaten:



Das Interface *Shape* enthält folgende Methoden:

- **contains:** Nimmt den Ortsvektor eines Punktes *point* als *V2*-Objekt und gibt *true* zurück, genau dann wenn der Punkt sich innerhalb der Figur befindet.
- **move:** Verschiebt die Figur um den *V2*-Vektor *displacement*.
- **boundingBox:** Gibt die „Bounding Box“ der Figur als *Box*-Objekt zurück. Die Bounding Box einer Figur ist das kleinste Rechteck, dass alle Punkte der Figur enthält.

Das Interface *Shape* ist auch bereits im Skelett gegeben.

Die *Shape*-Objekte sollen unveränderlich (eng. *immutable*) sein; die Methode *move* soll also ein frisch konstruiertes Objekt zurückgeben.

Diese Aufgabe ist in zwei Teile aufgeteilt:

1. Implementieren Sie zunächst die folgenden zwei Klassen für Rechtecke und Ellipsen:
 - **Ellipse:** wird aus einem Ortsvektor für das Zentrum und einem Vektor, der zwei Radien enthält, konstruiert. Die *x*-Komponente des Vektors enthält den Radius in Richtung der *x*-Achse, die *y*-Komponente den in Richtung der *y*-Achse.
 - **Rectangle:** wird aus dem Ortsvektor für die linke, obere Ecke und einem Vektor für die Maße konstruiert. Die Komponenten des Vektors für die Maße sollen positiv, d.h. > 0 , sein.

¹By K. Bolino - Made by K. Bolino (Kbolino), based upon earlier versions., Public Domain, <https://commons.wikimedia.org/w/index.php?curid=869195>

Implementieren Sie außerdem die Funktionen `makeEllipse` und `makeRectangle` in der Klasse `Shapes`, die neue Objekte der beiden Klassen zurückgeben sollen.

2. Implementieren Sie nun die *Shape*-Klasse `Picture`, die eine Vereinigung von Figuren darstellt: Ein `Picture` wird mit einer Liste der Figuren, aus der es besteht, konstruiert und besteht aus der Vereinigung der Punkte der enthaltenen Figuren. Implementieren Sie auch die Funktion `makePicture` in der Klasse `Shapes`, die ein *Shape*-Object der Klasse `Picture` zurück gibt.

Jenkins wird für jeden Teil separate Tests ausführen. Für beide Teile können Sie jeweils zwei Punkte erlangen. Diese werden nach den üblichen Kriterien vergeben.

Hinweise

- Hier sind zwei Links zur Ellipsendarstellung, die hier verwendet wird:

<http://www.mathopenref.com/coordgeneralellipse.html>

[https://math.stackexchange.com/questions/76457/
check-if-a-point-is-within-an-ellipse](https://math.stackexchange.com/questions/76457/check-if-a-point-is-within-an-ellipse)

- Sie dürfen die Klassen, die im Skelett mitgeliefert werden auch verändern (insbesondere bei `Box` kann das nützlich sein). Achten Sie aber darauf, dass diese immer noch die gleichen Methoden besitzen, die oben auf dem Blatt verlangt sind...sonst wird Jenkins sich beschweren.

Beispieltestfälle

```
package shapes;

import java.util.Arrays;

import static org.junit.Assert.*;

public class ExampleTests {

    @org.junit.Test
    public void testEllipse() throws Exception {
        Shape e = Shapes.makeEllipse(new V2(0, 0), new V2(1, 2));
        V2 p1 = new V2(0.5, 1);
        V2 p2 = new V2(1,1);
        assertTrue(e.contains(p1));
        assertFalse(e.contains(p2));

        assertFalse(e.move(new V2(-2, 0))
            .contains(p1));
    }

    @org.junit.Test
    public void testPicture() throws Exception {
        Shape e = Shapes.makeEllipse(new V2(0, 0), new V2(2, 1));
        Shape r = Shapes.makeRectangle(new V2(0,2), new V2(1, 2));
        V2 p1 = new V2(0.5, 1);
        V2 p2 = new V2(1,1);

        Shape pict = Shapes.makePicture(Arrays.asList(e, r));

        assertTrue(pict.contains(p1));
        assertTrue(pict.contains(p2));
        assertFalse(pict.contains(new V2(2, 0.5)));
    }

}
```