

Übung 09: Vererbung und Dynamische Bindung

Abgabetermin: TT.MM.JJJJ

Name: _____

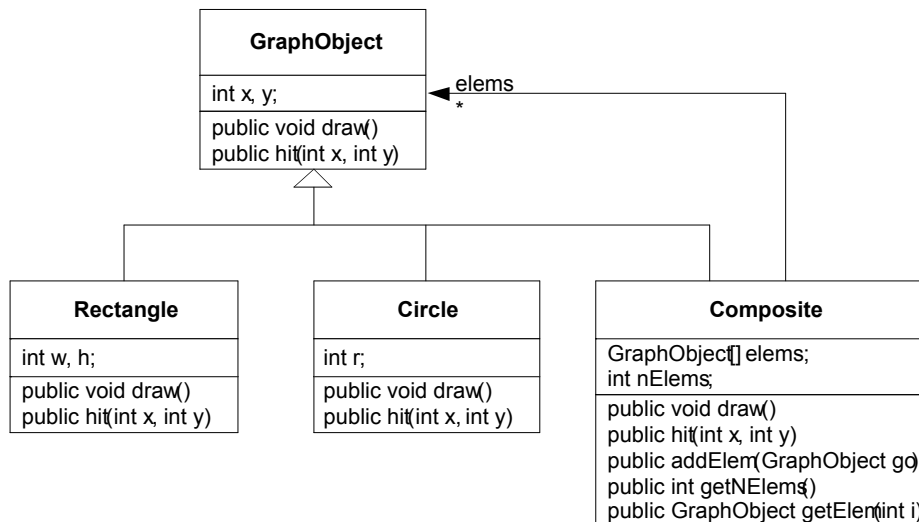
Matrikelnummer: _____

Gruppe: G1 (Prähofer) G2 (Prähofer) G3 (Wolfinger) G4 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Pkte
Aufgabe 09.X	12	<input type="checkbox"/>	Prosabeschreibung Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	

Aufgabe 09.X: Graphische Objekte

Realisieren Sie ein Klassensystem für graphische Objekte wie in der folgenden Abbildung dargestellt:



Schreiben Sie eine Basisklasse `GraphObject` und Unterklassen `Rectangle` und `Circle` für graphische Objekte. `GraphObjects` haben eine Position. `Rectangles` haben zusätzlich eine Breite und Höhe, `Circles` einen Radius. `Composites` bestehen aus einer Reihe von Unterobjekten (`elems`), die in einem Array vom Typ `GraphObject[]` gespeichert werden. Methoden `addElem`, `getNElem` und `getElement` dienen zur Definition und zum Zugriff auf die Unterelemente.

Neben Konstruktoren und Zugriffsmethoden sollen die Klassen folgende Operation implementieren:

- `void draw()`: zur Ausgabe mit Klasse `Window`
- `boolean hit(int x, int y)`: stellt fest, ob die Position `x / y` innerhalb des Objektes liegt.

Gehen Sie bei der Lösung folgend vor:

1. Skizzieren Sie Klassen und die Methoden in Prosa.
2. Implementieren Sie die Klassen und die Methoden in Java
3. Testen Sie das Programm interaktiv

Prosabeschreibung:

Klasse Point

- Int-Variablen x und y
- Konstruktor mit x und y als Parameter

Klasse GraphObject

Basisklasse für alle graphischen Objekte, abstract

- Feld name von Typ String
- Feld pos für Position vom Typ Point
- Konstruktor mit name als Parameter – pos ist 0/0
- Konstruktor mit name und x, y für Position
- Abstrakte Methode hit für Feststellen, ob gegebene Position x/y ein Treffer
- Abstrakte Methode draw für die Ausgabe auf Window
- Konkrete Methode move für das Verschieben um eine Distanz dx / dy

Klasse Rectangle

Klasse für Rechtecke, abgeleitet von GraphObject

- Felder w und h für Breite und Höhe
- Konstruktor mit name als Parameter – pos ist 0/0, w und h sind 0
- Konstruktor mit name und x, y für Position und w und h für Breite und Höhe
- Konkrete Methode hit die feststellt, ob innerhalb Rechteck
- Konkrete Methode draw für die Ausgabe auf Window

Klasse Circle

Klasse für Kreise, abgeleitet von GraphObject

- Feld r für Radius
- Konstruktor mit name als Parameter – pos ist 0/0, r ist 0
- Konstruktor mit name und x, y für Position und r für Radius
- Konkrete Methode hit die feststellt, ob innerhalb Kreis
- Konkrete Methode draw für die Ausgabe auf Window

Klasse Composite

Klasse für zusammengesetzte Objekte, abgeleitet von GraphicObject

- Array mit GraphObject-Elementen
- Int-Variable für Menge der Elemente
- Konstruktor mit name als Parameter – pos ist 0/0
- Konstruktor mit name und x, y für Position
- Methode addElem, welches ein GraphObject anfügt; beim Anfügen wird die Position des Elementes relativ zu diesem zusammengesetzten Element betrachtet und die Position angepasst.
- Methode getElem(int i) für den Zugriff auf das i-te Element
- Methode getNElems() für den Zugriff auf die Anzahl der Elemente
- Konkrete Methode hit die feststellt, ob innerhalb der Figur, geht sequentiell die Unterelemente durch
- Konkrete Methode draw für die Ausgabe auf Window, ruft draw für alle Unterelemente auf
- Konkrete Methode move für das Verschieben um eine Distanz dx / dy, wobei rekursiv move aller Elemente aufgerufen wird

Klasse GraphicalObjects

Testklasse mit main-Methode; beinhaltet die Schleife mit Einlesen des Schusses und Ausgabe der Treffer

Methode main:

- Aufbau einer Figur snowMan als Composite
- In einer Schleife
 - Einlesen der Position des Schusses
 - Ausgabe, ob Treffer

Java-Programm:

```
// Übungsprogramm graphische Objekte aufbauen und Enthaltensein prüfen

import java.awt.Color;

class Point {
    int x;
    int y;
}

abstract class GraphObject {

    String name;
    Point pos = new Point();

    public GraphObject(String name) {
        this(name, 0, 0);
    }

    public GraphObject(String name, int x, int y) {
        this.name = name;
        pos.x = x;
        pos.y = y;
    }

    abstract public boolean hit(int px, int py);

    abstract public void draw();

    public void move(int dx, int dy) {
        pos.x = pos.x + dx;
        pos.y = pos.y + dy;
    }
}

class Rectangle extends GraphObject {
    int w;
    int h;

    public Rectangle(String name) {
        this(name, 0, 0, 1, 1);
    }

    public Rectangle(String name, int x, int y, int w, int h) {
        super(name, x, y);
        this.w = w;
        this.h = h;
    }

    public boolean hit(int px, int py) {
        return px > pos.x && py > pos.y && px < pos.x + w && py < pos.y + h;
    }

    public void draw() {
        Window.fillRectangle(pos.x, pos.y, w, h, Color.LIGHT_GRAY);
        Window.drawRectangle(pos.x, pos.y, w, h);
    }
}

class Circle extends GraphObject {

    int r;

    public Circle(String name) {
        this(name, 0, 0, 1);
    }

    public Circle(String name, int x, int y, int r) {
        super(name, x, y);
        this.r = r;
    }

    public boolean hit(int px, int py) {
        double dpx = Math.abs(px - pos.x);
        double dpy = Math.abs(py - pos.y);
        double dist = Math.sqrt(dist = dpx * dpx + dpy * dpy);

        return dist < r;
    }
}
```

```

    public String toString() {
        return ("Circle" + "(" + pos.x + "," + pos.y + "," + r + "): " + name );
    }

    public void draw() {
        Window.fillCircle(pos.x, pos.y, r, Color.LIGHT_GRAY);
        Window.drawCircle(pos.x, pos.y, r);
    }
}

class Composite extends GraphObject {
    final static int CAPACITY = 100;
    GraphObject[] elems = new GraphObject[CAPACITY];
    int nElems = 0;

    public Composite(String name) {
        this(name, 0, 0);
    }

    public Composite(String name, int x, int y) {
        super(name, x, y);
    }

    public void addElem(GraphObject elem) {
        elem.move(pos.x, pos.y);
        elems[nElems] = elem;
        nElems++;
    }

    public GraphObject getElem(int i) {
        if (i >= 0 && i < nElems) {
            return elems[i];
        } else {
            return null;
        }
    }

    public int getNElems() {
        return nElems;
    }

    @Override
    public void draw() {
        for (int i = 0; i < nElems; i++) {
            elems[i].draw();
        }
    }

    @Override
    public boolean hit(int px, int py) {
        for (int i = 0; i < nElems; i++) {
            if (elems[i].hit(px, py)) return true;
        }
        return false;
    }

    @Override
    public void move(int dx, int dy) {
        for (int i = 0; i < nElems; i++) {
            elems[i].move(dx, dy);
        }
        super.move(dx, dy);
    }
}

public class GraphicalObjects {

    public static void main(String args[]) {

        Composite snowMan = new Composite("Snowman");
        snowMan.add(new Rectangle("BeinL", 100, 300, 45, 100));
        snowMan.add(new Rectangle("BeinR", 155, 300, 45, 100));
        snowMan.add(new Circle("Body", 150, 250, 60));
        snowMan.add(new Circle("Kopf", 150, 160, 30));
        Composite hut = new Composite("Hut", 130, 90);
        hut.add(new Rectangle("Hut1", 0, 30, 40, 10));
        hut.add(new Rectangle("Hut2", 10, 0, 20, 30));
        snowMan.add(hut);
        snowMan.add(new Rectangle("ArmL", 60, 210, 40, 30));
        snowMan.add(new Rectangle("ArmR", 200, 210, 40, 30));

        snowMan.move(10, 10);
    }
}

```

```
Window.open();

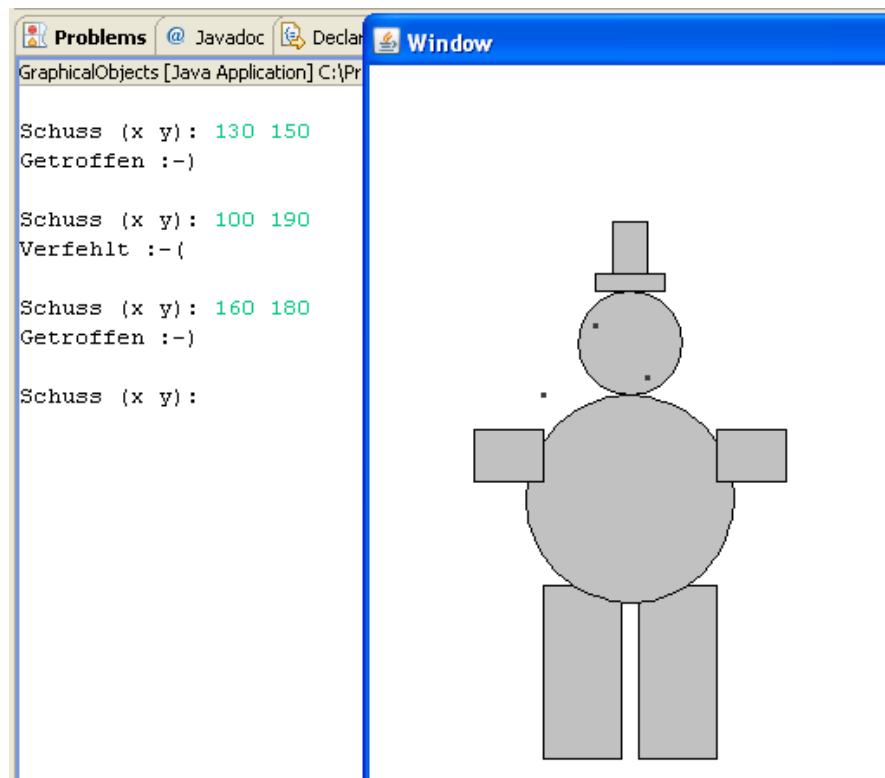
snowMan.draw();

int nrHits = 0;
Point shot;
shot = readShot();
while (shot != null) {
    Window.drawPoint(shot.x, shot.y, Color.darkGray);

    if (snowMan.hit(shot.x, shot.y)) {
        Out.println("Getroffen :-) ");
        nrHits++;
    } else {
        Out.println("Verfehlt :-(");
    }
    shot = readShot();
}
Out.println("Gratuliere: " + nrHits + " Treffer insgesamt!");
}

static Point readShot() {
    Out.println();
    Out.print("Schuss (x y): ");
    Point p = new Point();
    p.x = In.readInt();
    p.y = In.readInt();
    if (p.x >= 0 && p.y >= 0) return p; else return null;
}

static GraphObject findHit(GraphObject[] gos, Point shot) {
    for (int i = 0; i < gos.length; i++) {
        if (gos[i].hit(shot.x, shot.y)) {
            return gos[i];
        }
    }
    return null;
}
}
```

Test:

The screenshot shows a Java IDE window titled "GraphicalObjects [Java Application] C:\Pr". The console window on the left displays the following log output:

```
Schuss (x y): 130 150  
Getroffen :-)  
  
Schuss (x y): 100 190  
Verfehlt :-(  
  
Schuss (x y): 160 180  
Getroffen :-)  
  
Schuss (x y):
```

The graphical window on the right displays a simple stick figure character. The character is composed of several gray shapes: a small rectangle for the head, a larger circle for the torso, two small rectangles for arms, and two vertical rectangles for legs. The character is centered in the window.