Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 8

з дисципліни “Інженерія ПЗ”

“Шаблони, що породжують. Шаблони Prototype, Singleton та Factory Method.”

Виконала: студентка 2 курсу групи ІО-32

Руденко Т.А.

Перевірив: ст.в. Антонюк А.І..

Київ – 2014

**Варіант завдання:**

**3224** MOD 11 = **1**

Визначити специфікації класів та реалізацію методів для механізму клонування графічних елементів у редакторі векторної графіки. Забезпечити можливість як глибокого так і поверхневого клонування.

**Лістинг:**

/\*\*

\* Cloner interface

\*/

**package** com.lab111.labwork8;

/\*\*

\* **@author** TRudenko

\* **@version** 8.1

\*/

**public** **interface** PrototypeClone {

**public** PrototypeClone surfCloner();

**public** PrototypeClone deepCloner();

}

/\*\*

\* Class of a point, represents a ConcretePrototype

\*/

package com.lab111.labwork8;

/\*\*

\* @author TRudenko

\* @version 8.1

\*/

public class Point implements PrototypeClone {

private int x;

private int y;

/\*\*

\* Constructor

\* @param x coordinate of a point on the x axis

\* @param y coordinate of a point on the x axis

\*/

public Point(int x, int y){

this.x = x;

this.y = y;

}

/\*\*

\* @return the x

\*/

public int getX() {

return x;

}

/\*\*

\* @param x the x to set

\*/

public void setX(int x) {

this.x = x;

}

/\*\*

\* @return the y

\*/

public int getY() {

return y;

}

/\*\*

\* @param y the y to set

\*/

public void setY(int y) {

this.y = y;

}

/\*\*

\* Returns a string with the properties of the object

\*/

@Override

public String toString() {

return "Point [x=" + x + ", y=" + y + "]";

}

/\*\*

\* Returns a surface clone of the object

\*/

@Override

public PrototypeClone surfCloner() {

return new Point(this.x, this.y);

}

/\*\*

\* Returns a deep clone of the object

\*/

@Override

public PrototypeClone deepCloner() {

return this.surfCloner();

}

}

/\*\*

\* Class of a circle, represents a ConcretePrototype

\*/

package com.lab111.labwork8;

/\*\*

\* @author TRudenko

\* @version

\*/

public class Circle implements PrototypeClone {

private Point center;

private int radius;

/\*\*

\* Constructor

\* @param center coordinates of the center

\* @param radius the radius of a circle

\*/

public Circle(Point center, int radius){

this.center = center;

this.radius = radius;

}

/\*\*

\* Returns a surface clone of the object

\*/

@Override

public PrototypeClone surfCloner() {

return new Circle(this.center, this.radius);

}

/\*\*

\* Returns a deep clone of the object

\*/

@Override

public PrototypeClone deepCloner() {

return new Circle((Point)center.deepCloner(), this.radius);

}

/\*\*

\* @return the center

\*/

public Point getCenter() {

return center;

}

/\*\*

\* @param center the center to set

\*/

public void setCenter(Point center) {

this.center = center;

}

/\*\*

\* @return the radius

\*/

public int getRadius() {

return radius;

}

/\*\*

\* @param radius the radius to set

\*/

public void setRadius(int radius) {

this.radius = radius;

}

/\*\*

\* Returns a string with the properties of the object

\*/

@Override

public String toString() {

return "Circle [center=" + center.toString() + ", radius=" + radius + "]";

}

}

/\*\*

\* Class of a rectangle, represents a ConcretePrototype

\*/

package com.lab111.labwork8;

/\*\*

\* @author TRudenko

\* @version 8.1

\*/

public class Rectangle implements PrototypeClone {

private Point c1,c2;

/\*\*

\* Constructor

\* @param c1 coordinates of the top left corner

\* @param c2 coordinates of the bottom right corner

\*/

public Rectangle(Point c1, Point c2){

this.c1 = c1;

this.c2 = c2;

}

/\*\*

\* Returns a surface clone of the object

\*/

@Override

public PrototypeClone surfCloner() {

return new Rectangle(this.c1, this.c2);

}

/\*\*

\* Returns a deep clone of the object

\*/

@Override

public PrototypeClone deepCloner() {

return new Rectangle((Point)c1.deepCloner(), (Point)c2.deepCloner());

}

/\*\*

\* @return the c1

\*/

public Point getC1() {

return c1;

}

/\*\*

\* @param c1 the c1 to set

\*/

public void setC1(Point c1) {

this.c1 = c1;

}

/\*\*

\* @return the c2

\*/

public Point getC2() {

return c2;

}

/\*\*

\* @param c2 the c2 to set

\*/

public void setC2(Point c2) {

this.c2 = c2;

}

/\*\*

\* Returns a string with the properties of the object

\*/

@Override

public String toString() {

return "Rectangle [corner1=" + c1.toString() + ", corner2=" + c2.toString() + "]";

}

}

/\*\*

\* Class of an icon, represents a ConcretePrototype

\*/

package com.lab111.labwork8;

/\*\*

\* @author TRudenko

\* @version 8.1

\*/

public class Icon implements PrototypeClone {

private Rectangle r;

private Circle c;

/\*\*

\* Constructor

\* @param r a rectangle

\* @param c a circle

\*/

public Icon(Rectangle r, Circle c){

this.r = r;

this.c = c;

}

/\*\*

\* Returns a surface clone of the object

\*/

@Override

public PrototypeClone surfCloner() {

return new Icon(this.r, this.c);

}

/\*\*

\* Returns a deep clone of the object

\*/

@Override

public PrototypeClone deepCloner() {

return new Icon((Rectangle)r.deepCloner(), (Circle)c.deepCloner());

}

/\*\*

\* @return the r

\*/

public Rectangle getR() {

return r;

}

/\*\*

\* @param r the r to set

\*/

public void setR(Rectangle r) {

this.r = r;

}

/\*\*

\* @return the c

\*/

public Circle getC() {

return c;

}

/\*\*

\* @param c the c to set

\*/

public void setC(Circle c) {

this.c = c;

}

/\*\*

\* Returns a string with the properties of the object

\*/

@Override

public String toString() {

return "Icon [rectangle=" + r.toString() + ", circle=" + c.toString() + "]";

}

}

/\*\*

\* Main class that represents the work of Prototype design pattern

\*/

package com.lab111.labwork8;

/\*\*

\* @author TRudenko

\* @version 8.1

\*/

public class Lab8 {

public static void main(String[] args) {

//Creating a new circle and making a deep copy of it

Circle c = new Circle(new Point(2,4), 8);

Circle copyC = (Circle) c.deepCloner();

System.out.println(c.toString());

System.out.println(copyC.toString());

//Creating a new rectangle and making a surface copy of it

Rectangle r = new Rectangle(new Point(8,7), new Point(2,1));

Rectangle copyR = (Rectangle) r.surfCloner();

System.out.println(r.toString());

System.out.println(copyR.toString());

//Creating a new icon and making a deep copy of it

Icon ico = new Icon(r,c);

Icon copyIco = (Icon) ico.deepCloner();

System.out.println(ico.toString());

System.out.println(copyIco.toString());

}

}

Результат виконання:

Circle [center=Point [x=2, y=4], radius=8]

Rectangle [corner1=Point [x=8, y=7], corner2=Point [x=2, y=1]]

Rectangle [corner1=Point [x=8, y=7], corner2=Point [x=2, y=1]]

Icon [rectangle=Rectangle [corner1=Point [x=8, y=7], corner2=Point [x=2, y=1]], circle=Circle [center=Point [x=2, y=4], radius=8]]

Icon [rectangle=Rectangle [corner1=Point [x=8, y=7], corner2=Point [x=2, y=1]], circle=Circle [center=Point [x=2, y=4], radius=8]]

