|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования«МИРЭА - Российский технологический университет»РТУ МИРЭА |

Институт технологий управления (ИТУ)

Кафедра информационных технологий в государственном управлении (ИТГУ)

Объектно-ориентированный анализ и программирование

Практическая работа №3  
Гордиенко Анна Александровна   
УИБО-02-21

// Abstract superclass Shape

abstract class Shape {

protected String color;

protected boolean filled;

public Shape(String color, boolean filled) {

this.color = color;

this.filled = filled;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public boolean isFilled() {

return filled;

}

public void setFilled(boolean filled) {

this.filled = filled;

}

public abstract double getArea();

public abstract double getPerimeter();

@Override

public String toString() {

return "Shape[color=" + color + ", filled=" + filled + "]";

}

}

// Circle subclass

class Circle extends Shape {

private double radius;

public Circle(String color, boolean filled, double radius) {

super(color, filled);

this.radius = radius;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

@Override

public double getArea() {

return Math.PI \* radius \* radius;

}

@Override

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

@Override

public String toString() {

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

}

}

// Rectangle subclass

class Rectangle extends Shape {

private double width;

private double length;

public Rectangle(String color, boolean filled, double width, double length) {

super(color, filled);

this.width = width;

this.length = length;

}

public double getWidth() {

return width;

}

public void setWidth(double width) {

this.width = width;

}

public double getLength() {

return length;

}

public void setLength(double length) {

this.length = length;

}

@Override

public double getArea() {

return width \* length;

}

@Override

public double getPerimeter() {

return 2 \* (width + length);

}

@Override

public String toString() {

return "Rectangle[" + super.toString() + ", width=" + width + ", length=" + length + "]";

}

}

// Square subclass

class Square extends Rectangle {

public Square(String color, boolean filled, double side) {

super(color, filled, side, side);

}

public double getSide() {

return getWidth(); // or getLength()

}

public void setSide(double side) {

setWidth(side);

setLength(side);

}

@Override

public String toString() {

return "Square[" + super.toString() + "]";

}

}

// Movable interface

interface Movable {

void moveUp();

void moveDown();

void moveLeft();

void moveRight();

}

// MovableRectangle class

class MovableRectangle implements Movable {

private MovablePoint topLeft;

private MovablePoint bottomRight;

public MovableRectangle(int x1, int y1, int x2, int y2, int xSpeed, int ySpeed) {

topLeft = new MovablePoint(x1, y1, xSpeed, ySpeed);

bottomRight = new MovablePoint(x2, y2, xSpeed, ySpeed);

}

public boolean sameSpeed() {

return topLeft.xSpeed == bottomRight.xSpeed && topLeft.ySpeed == bottomRight.ySpeed;

}

@Override

public void moveUp() {

topLeft.moveUp();

bottomRight.moveUp();

}

@Override

public void moveDown() {

topLeft.moveDown();

bottomRight.moveDown();

}

@Override

public void moveLeft() {

topLeft.moveLeft();

bottomRight.moveLeft();

}

@Override

public void moveRight() {

topLeft.moveRight();

bottomRight.moveRight();

}

}

// MovablePoint class

class MovablePoint implements Movable {

private int x;

private int y;

private int xSpeed;

private int ySpeed;

public MovablePoint(int x, int y, int xSpeed, int ySpeed) {

this.x = x;

this.y = y;

this.xSpeed = xSpeed;

this.ySpeed = ySpeed;

}

@Override

public void moveUp() {

y -= ySpeed;

}

@Override

public void moveDown() {

y += ySpeed;

}

@Override

public void moveLeft() {

x -= xSpeed;

}

@Override

public void moveRight() {

x += xSpeed;

}

}

Ссылка на GitHub: https://github.com/AnnaGoko/java\_oop\_mirea