|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №2**

Выполнил студент группы ИНБО-02-20 Маркарянц Р.Д.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

Упражнение 1. Создайте абстрактный родительский суперкласс Shape и его дочерние классы (подклассы).

Упражнение 2. Перепишите суперкласс Shape и его подклассы, так как это представлено на диаграмме Circle, Rectangle and Square.

Упражнение 3. Вам нужно написать тестовый класс, чтобы самостоятельно это проверить, необходимо объяснить полученные результаты и связать их с понятием ООП - полиморфизм. Некоторые объявления могут вызвать ошибки компиляции. Объясните полученные ошибки, если таковые имеются.

Упражнение 4. Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable.

Упражнение 5. Напишите новый класс MovableRectangle (движущийся прямоугольник). Его можно представить как две движущиеся точки MovablePoints (представляющих верхняя левая и нижняя правая точки) и реализующие интерфейс Movable.

## **Ход Работы**

В ходе выполнения работы были получены следующие исходные коды:

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public abstract class Shape {  
 protected String color;  
 protected boolean filled;  
  
 public Shape() {  
 this.color = "blue";  
 this.filled = false;  
 }  
 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;  
 }  
  
 abstract double getArea();  
 abstract double getPerimeter();  
  
 @Override  
 public String toString() {  
 return "Shape{" +  
 "color='" + color + '\'' +  
 ", filled=" + filled +  
 '}';  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class Circle extends Shape{  
 protected double radius;  
  
 public Circle(){  
 this.filled = false;  
 this.color = "blue";  
 this.radius = 1;  
 }  
 public Circle(double radius){  
 this.filled = false;  
 this.color = "blue";  
 this.radius = radius;  
 }  
 public Circle(double radius, String color, boolean filled){  
 this.radius = radius;  
 this.color = color;  
 this.filled = filled;  
 }  
 public double getRadius() {  
 return radius;  
 }  
 public void setRadius(double radius) {  
 this.radius = radius;  
 }  
 public double getArea() {  
 return Math.*PI*\*radius\*radius;  
 }  
 public double getPerimeter() {  
 return 2\*Math.*PI*\*radius;  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: circle, radius: "+this.radius+", color: "+this.color;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class Rectangle extends Shape{  
 protected double width;  
 protected double length;  
  
 public Rectangle() {  
 this.width = 10;  
 this.length = 20;  
 this.filled = false;  
 this.color = "blue";  
 }  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 this.filled = false;  
 this.color = "blue";  
 }  
 public Rectangle(String color, boolean filled, double width, double length) {  
 this.width = width;  
 this.length = length;  
 this.color = color;  
 this.filled = filled;  
 }  
 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;  
 }  
 public double getArea() {  
 return length\*width;  
 }  
 public double getPerimeter() {  
 return 2\*(length+width);  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: rectangle, width: "+this.width+", length: "+this.length;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class Square extends Rectangle{  
 protected double side;  
  
 public Square() {  
 this.side = 10;  
 }  
 public Square(double side) {  
 this.side = side;  
 this.filled = false;  
 this.color = "blue";  
 }  
 public Square(String color, boolean filled, double side) {  
 this.side = side;  
 this.color = color;  
 this.filled = filled;  
 }  
 public double getSide() {  
 return side;  
 }  
 public void setSide(double side) {  
 this.side = side;  
 }  
 public void setWidth(double side) {  
 this.side = side;  
 }  
 public void setLength(double side) {  
 this.side = side;  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: square, side: "+this.side;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
  
public class Test {  
 public static void main(String[] args) {  
 Shape s1 = new Circle(5.5, "RED", false);  
 System.out.println(s1);  
 System.out.println(s1.getArea());  
 System.out.println(s1.getPerimeter());  
 System.out.println(s1.getColor());  
 System.out.println(s1.isFilled());  
 System.out.println(s1.getRadius()); // У объекта shape нет радиуса  
  
 Circle c1 = (Circle)s1;  
 System.out.println(c1);  
 System.out.println(c1.getArea());  
 System.out.println(c1.getPerimeter());  
 System.out.println(c1.getColor());  
 System.out.println(c1.isFilled());  
 System.out.println(c1.getRadius());  
  
 Shape s2 = new Shape(); // Нельзя создать экземпляр абстрактного класса  
  
 Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Заданы неверные параметры  
 System.out.println(s3);  
 System.out.println(s3.getArea());  
 System.out.println(s3.getPerimeter());  
 System.out.println(s3.getColor());  
 System.out.println(s3.getLength());  
  
 Rectangle r1 = (Rectangle)s3;  
 System.out.println(r1);  
 System.out.println(r1.getArea());  
 System.out.println(r1.getColor());  
 System.out.println(r1.getLength());  
  
 Shape s4 = new Square(6.6);  
 System.out.println(s4);  
 System.out.println(s4.getArea());  
 System.out.println(s4.getColor());  
 System.out.println(s4.getSide()); // Обращене к несуществующему методу  
 Rectangle r2 = (Rectangle)s4;  
 System.out.println(r2);  
 System.out.println(r2.getArea());  
 System.out.println(r2.getColor());  
 System.out.println(r2.getSide()); // Обращене к несуществующему методу  
 System.out.println(r2.getLength());  
  
 Square sq1 = (Square)r2;  
 System.out.println(sq1);  
 System.out.println(sq1.getArea());  
 System.out.println(sq1.getColor());  
 System.out.println(sq1.getSide());  
 System.out.println(sq1.getLength());  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
  
public interface Movable {  
 public void moveUp();  
 public void moveDown();  
 public void moveLeft();  
 public void moveRight();  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class MovablePoint implements Movable{  
 protected int x;  
 protected int y;  
 protected int xSpeed;  
 protected 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 String toString() {  
 return "MovablePoint{" +  
 "x=" + x +  
 ", y=" + y +  
 ", xSpeed=" + xSpeed +  
 ", ySpeed=" + ySpeed +  
 '}';  
 }  
 public void moveUp() {  
 this.y+=1;  
 }  
 public void moveDown() {  
 this.y-=1;  
 }  
 public void moveLeft() {  
 this.x-=1;  
 }  
 public void moveRight() {  
 this.x+=1;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class MovableCircle extends MovablePoint{  
 private int radius;  
 private MovablePoint center;  
  
 public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {  
 super(x, y, xSpeed, ySpeed);  
 this.radius = radius;  
 }  
 public void moveUp() {  
 this.center.y+=1;  
 }  
 public void moveDown() {  
 this.center.y-=1;  
 }  
 public void moveLeft() {  
 this.center.x-=1;  
 }  
 public void moveRight() {  
 this.center.x+=1;  
 }  
  
 @Override  
 public String toString() {  
 return "MovableCircle{" +  
 "center=" + center +  
 ", x=" + x +  
 ", y=" + y +  
 ", xSpeed=" + xSpeed +  
 ", ySpeed=" + ySpeed +  
 '}';  
 }  
}

package ru.mirea.inbo0220.markaryants.pr3;  
import java.lang.\*;  
  
public class MovableRectangle extends MovablePoint{  
 private MovablePoint topLeft;  
 private MovablePoint bottomRight;  
  
 public MovableRectangle(int x, int y, int xSpeed, int ySpeed) {  
 super(x, y, xSpeed, ySpeed);  
 }  
 public void moveUp() {  
 this.topLeft.y+=1;  
 this.bottomRight.y+=1;  
 }  
 public void moveDown() {  
 this.topLeft.y-=1;  
 this.bottomRight.y-=1;  
 }  
 public void moveLeft() {  
 this.topLeft.x-=1;  
 this.bottomRight.x-=1;  
 }  
 public void moveRight() {  
 this.topLeft.x+=1;  
 this.bottomRight.x+=1;  
 }  
}

## **Вывод**

Выполнив практическую работу, я получил навыки работы с абстрактными классами, интерфейсом и наследованием.