|  |
| --- |
|  |
| 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 №3-4**

Выполнил студент группы ИНБО-02-20 Судариков Д.А.

Принял Степанов П.В.

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

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

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

Написать два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable .

Создайте JFrame приложение у которого есть компоненты GUI

## **Ход Работы**

В ходе выполнения работы были получены следующие исходные коды:

package ru.mirea.lab.lab3.ex1;  
  
public class Circle extends Shape {  
 @Override  
 public void typeShape(){  
 System.*out*.println("Это круг");  
 }  
}

package ru.mirea.lab.lab3.ex1;  
  
public class RECTANGLE extends Shape {  
 @Override  
 public void typeShape(){  
 System.*out*.println("Это прямоугольник");  
 }  
}

package ru.mirea.lab.lab3.ex1;  
  
public abstract class Shape {  
 public void square(){  
 System.*out*.println("Любая фигура имеет площадь");  
 }  
 public abstract void typeShape();  
}

package ru.mirea.lab.lab3.ex1;  
  
import ru.mirea.lab.lab3.ex1.Circle;  
import ru.mirea.lab.lab3.ex1.RECTANGLE;  
  
public class TestShape {  
 public static void main(String[] args) {  
 RECTANGLE a = new RECTANGLE();  
 Circle b = new Circle();  
 a.square();  
 a.typeShape();  
 b.typeShape();  
 }  
}

package ru.mirea.lab.lab3.ex2ex3;  
  
public class Circle1 extends Shape1 {  
 protected double radius;  
 public Circle1(){}  
 public Circle1 (double radius){  
 this.radius = radius;  
 }  
 public Circle1(double radius,String color,boolean filled){  
 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* \* Math.*pow*(radius, 2);  
 }  
  
 @Override  
 public double getPerimeter() {  
 return 2 \* Math.*PI* \* radius;  
 }  
  
 @Override  
 public String toString() {  
 return "Круг имеет радиус равный: "+ radius + ", площадь равную: "+ getArea() + ", и периметр равный: "+ getPerimeter();  
 }  
}

package ru.mirea.lab.lab3.ex2ex3;  
  
public class Rectangle1 extends Shape1 {  
 protected double width;  
 protected double length;  
 public Rectangle1() {}  
 public Rectangle1(double width,double length){  
 this.width=width;  
 this.length=length;  
 }  
 public Rectangle1(double width,double length,String color,boolean filled){  
 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;  
 }  
 public double getArea() {  
 return width \* length;  
 }  
 public double getPerimeter() {  
 return 2\* (width + length);  
 }  
 public String toString() {  
 return "Прямоуголник с шириной=" + width + ", длиной=" + length + ", площадь которого равна=" + getArea() + ", периметр=" + getPerimeter();  
 }  
}

package ru.mirea.lab.lab3.ex2ex3;  
  
 public abstract class Shape1 {  
 protected String color;  
 protected boolean filled;  
 public Shape1(String color,boolean filled){  
 this.color=color;  
 this.filled=filled;  
 }  
  
 public String getColor() {  
 return color;  
 }  
 public Shape1 () {this("RED", true);}  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
 public boolean isFilled(){  
 return filled;  
 }  
 public void setFilled (boolean filled) {  
 this.filled = filled;  
 }  
 abstract public double getArea();  
 abstract public double getPerimeter();  
 abstract public String toString();  
 }

package ru.mirea.lab.lab3.ex2ex3;  
  
public class Square extends Rectangle1{  
 public Square (){};  
 public Square (double side){  
 super(side,side);  
 }  
 public Square(double side,String color,boolean filled) {  
 super(side,side,color,filled);  
 }  
 public double getSide() {  
 return width;  
 }  
 public void setSide(double side){  
 width = side;  
 length = side;  
 }  
 public void getWidth(double side){  
 setSide(side);  
 setSide(side);  
 }  
 public String toString(){  
 return "Квадрат с стороной=" + getSide() + ", площадью=" + getArea() + ", периметром=" + getPerimeter();  
 }  
}

package ru.mirea.lab.lab3.ex2ex3;  
  
public class TestShape1 {  
 public static void main(String[] args) {  
 Shape1 s1 = new Circle1(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());  
 // У класса Shape нет метода getRadius()  
 //System.out.println(s1.getRadius());  
 Circle1 c1 = (Circle1)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());  
 // Нельзя создать экземпляр абстрактного класса  
 //Shape s2 = new Shape();  
 Shape1 s3 = new Rectangle1(1.0, 2.0, "RED", false);  
 System.*out*.println(s3);  
 System.*out*.println(s3.getArea());  
 System.*out*.println(s3.getPerimeter());  
 System.*out*.println(s3.getColor());  
 // У класса Shape не метода getLength()  
 //System.out.println(s3.getLength());  
 Rectangle1 r1 = (Rectangle1)s3;  
 System.*out*.println(r1);  
 System.*out*.println(r1.getArea());  
 System.*out*.println(r1.getColor());  
 System.*out*.println(r1.getLength());  
 Shape1 s4 = new Square(6.6);  
 System.*out*.println(s4);  
 System.*out*.println(s4.getArea());  
 System.*out*.println(s4.getColor());  
 // У класса Shape нет метода getSide()  
 //System.out.println(s4.getSide());  
 Rectangle1 r2 = (Rectangle1)s4;  
 System.*out*.println(r2);  
 System.*out*.println(r2.getArea());  
 System.*out*.println(r2.getColor());  
 // У класса Rectangle тоже нет метода getSide()  
 //System.out.println(r2.getSide());  
 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.lab.lab3.ex4ex5;  
  
public interface Movable {  
 public void moveUp();  
 public void moveDown();  
 public void moveLeft();  
 public void moveRight();  
}

package ru.mirea.lab.lab3.ex4ex5;  
  
public class MovableCircle implements Movable{  
 protected int radius;  
 protected MovablePoint center;  
 public MovableCircle (int x, int y, int xSpeed, int ySpeed,int radius){  
 this.radius = radius;  
 this.center = new MovablePoint(x,y,xSpeed,ySpeed);  
 }  
 public String toString(){  
 return "Круг ("+ center.x + ", " + center.y + ") с радиусом равным: " + radius;  
 }  
  
 @Override  
 public void moveUp() {  
 center.moveUp();  
 }  
  
 @Override  
 public void moveDown() {  
 center.moveDown();  
 }  
  
 @Override  
 public void moveLeft() {  
 center.moveLeft();  
 }  
  
 @Override  
 public void moveRight() {  
 center.moveRight();  
 }  
}

package ru.mirea.lab.lab3.ex4ex5;  
  
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;  
 }  
 public String toString(){  
 return "Точка (" + x + ", " + y +")";  
 }  
 public void moveUp(){y++;}  
 public void moveDown(){y--;}  
 public void moveLeft(){x++;}  
 public void moveRight(){x--;}  
 }

package ru.mirea.lab.lab3.ex4ex5;  
  
public class MovableRectangle implements Movable {  
 private MovablePoint topLeft;  
 private MovablePoint bottomRight;  
 public MovableRectangle(int x1, int y1, int x2, int y2, int xSpeed,int ySpeed){  
 this.topLeft = new MovablePoint(x1,y1,xSpeed,ySpeed);  
 this.bottomRight = new MovablePoint(x2,y2,xSpeed,ySpeed);  
 }  
 public String toString(){  
 return "Прямоугольник (" + topLeft.x + " , " + topLeft.y + " , " + bottomRight.x + " , " + bottomRight.y + ")";  
 }  
  
 @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();  
 }  
}

package ru.mirea.lab.lab3.ex4ex5;  
  
public class TestMovable {  
 public static void main(String[] args) {  
 MovablePoint point = new MovablePoint(0,0,1,1);  
 System.*out*.println(point);  
 point.moveUp();  
 point.moveLeft();  
 System.*out*.println(point);  
 MovableCircle circle = new MovableCircle(0,0,1,1,2);  
 System.*out*.println(circle);  
 circle.moveDown();  
 circle.moveRight();  
 System.*out*.println(circle);  
 MovableRectangle rectangle = new MovableRectangle(0, 0, 1,1,1,1);  
 System.*out*.println(rectangle);  
 rectangle.moveUp();  
 rectangle.moveLeft();  
 System.*out*.println(rectangle);  
 }  
}

**4 практика**. Я решил данную задачу с помощью JavaFX и программы Scene Builder.

В ходе выполнения работы были получены следующие исходные коды:

package ru.mirea.lab.lab4;  
  
import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
  
public class GUI extends JFrame{  
 private JFrame frame = new JFrame("Match!");  
 private JButton button1 = new JButton("AC Millan");  
 private JButton button2 = new JButton("Real Madrid");  
 private JButton button3 = new JButton("End Match");  
 private JButton button4 = new JButton("New Match");  
 private JLabel label1 = new JLabel("Result: 0X0");  
 private JLabel label2 = new JLabel("Last scorer: N/A");  
 private JLabel label3 = new JLabel("Winner: none");  
 public int click\_but1 = 0;  
 public int click\_but2 = 0;  
  
  
 public GUI() {  
 super("Example");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(null);  
 setSize(555, 455);  
 button1.setBounds(25, 15, 200, 50);  
 button1.setActionCommand("Millan");  
 button2.setBounds(250, 15, 200, 50);  
 button2.setActionCommand("Madrid");  
 button3.setBounds(25, 120, 200, 50);  
 button3.setActionCommand("End");  
 button4.setBounds(250, 120, 200, 50);  
 button4.setActionCommand("Reset");  
 label1.setBounds(25, 280, 150, 20);  
 label2.setBounds(25, 300, 150, 20);  
 label3.setBounds(25, 320, 150, 20);  
 add(button1);  
 add(button2);  
 add(button3);  
 add(button4);  
 add(label1);  
 add(label2);  
 add(label3);  
 ActionListener actionListener = new TestActionListener();  
  
 button1.addActionListener(actionListener);  
 button2.addActionListener(actionListener);  
 button3.addActionListener(actionListener);  
 button4.addActionListener(actionListener);  
 }  
  
 public class TestActionListener implements ActionListener{  
 public void actionPerformed(ActionEvent e) {  
 String butText = e.getActionCommand();  
 switch(butText){  
 case "Millan":  
 click\_but1++;  
 label1.setText("Result: " + click\_but1 + "X" + click\_but2);  
 label2.setText("Last scorer: " + butText);  
 if (click\_but1 > click\_but2){  
 label3.setText("Winner: " + butText);  
 } else if (click\_but2 == click\_but1){  
 label3.setText("Winner: none");  
 }  
 break;  
 case "Madrid":  
 click\_but2++;  
 label1.setText("Result: " + click\_but1 + "X" + click\_but2);  
 label2.setText("Last scorer: " + butText);  
 if (click\_but2 > click\_but1){  
 label3.setText("Winner: " + butText);  
 } else if (click\_but2 == click\_but1){  
 label3.setText("Winner: none");  
 }  
 break;  
 case "Reset":  
 click\_but1 = 0; click\_but2 = 0;  
 label1.setText("Result: " + click\_but1 + "X" + click\_but2);  
 label2.setText("Last scorer: N/A");  
 label3.setText("Winner: none");  
 break;  
 case "End":  
 if (click\_but1 > click\_but2){  
 JOptionPane.*showMessageDialog*(null, "Winner in this match is AC Millan!!!", "Result", JOptionPane.*INFORMATION\_MESSAGE*);  
 } else if (click\_but2 > click\_but1){  
 JOptionPane.*showMessageDialog*(null, "Winner in this match is Real Madrid!!!", "Result", JOptionPane.*INFORMATION\_MESSAGE*);  
 } else {  
 JOptionPane.*showMessageDialog*(null, "The match ended in a draw", "Result", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
 break;  
 }  
 }  
 }  
 public static void main(String[] args){  
 new GUI().setVisible(true);  
 }  
}

## **Вывод**

Получены знания по работе с интерфейсом, а также я продолжил изучать работу с классами. Также мной были получены знания по работе с GUI, JavaFX и Scene Builder.