|  |
| --- |
|  |
| 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 №1**

Выполнил студент группы ИНБО-02-20 Абдусалямова С. К.

Принял Степанов П.В.

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

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

Отметка о выполнении

Москва 2021г.

## **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.

## **Ход Работы**

На основе поставленной задачи был написан код программы:

Shape.java – абстрактный родительский суперкласс

package com.company;  
  
public abstract class Shape {  
 protected String color;  
 protected boolean filled;  
  
 public Shape() {  
 this.filled = false;  
 this.color = "blue";  
 }  
  
 public Shape(String color, boolean filled) {  
 this.color = color;  
 this.filled = filled;  
 }  
  
 public String getColor() {  
 return color;  
 }  
  
 public boolean isFilled() {  
 return filled;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 public void setFilled(boolean filled) {  
 this.filled = filled;  
 }  
  
 abstract double getArea();  
 abstract double getPerimeter();  
 @Override  
 abstract public String toString();

Rectangle.java

package com.company;  
  
public class Rectangle extends Shape {  
 protected double width;  
 protected double length;  
  
 public Rectangle() {  
 this.length = 1;  
 this.width = 1;  
 }  
  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 }  
  
 public Rectangle(double width, double length, String color, boolean filled) {  
 super(color, filled); //экземляр родительского класса  
 this.width = width;  
 this.length = length;  
 }  
  
 public double getWidth() {  
 return width;  
 }  
  
 public double getLength() {  
 return length;  
 }  
  
 public void setWidth(double width) {  
 this.width = width;  
 }  
  
 public void setLength(double length) {  
 this.length = length;  
 }  
  
 @Override  
 double getArea() {  
 return this.length \* this.width;  
 }  
  
 @Override  
 double getPerimeter() {  
 return 2 \* (width + length);  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: rectangle, width: " + this.width  
 + ", length:" + this.length + ", color: " + this.color;  
 }  
}

Circle.java

package com.company;  
  
public class Circle extends Shape{  
 protected double radius;  
  
 public Circle(double radius) {  
 this.radius = radius;  
 }  
  
 public Circle(double radius, String color, boolean filled) {  
 super(color, filled);   
 this.radius = radius; //экземпляр текущего класса  
 }  
  
 public Circle() {  
 radius = 1;  
 }  
  
 public double getRadius() {  
 return radius;  
 }  
  
 public void setRadius(double radius) {  
 this.radius = radius;  
 }  
  
 @Override  
 double getArea() {  
 return Math.*PI*\*radius\*radius;  
 }  
  
 @Override  
 double getPerimeter() {  
  
 return 2\*Math.*PI*\*radius;  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: circle, radius: "+this.radius+", color: "+this.color;  
 }  
}

Square.java

package com.company;  
  
public class Square extends Rectangle {  
 private double side;  
  
 public Square() {  
 this.side = 1;  
 }  
  
 public Square(double side) {  
 this.side = side;  
 }  
  
 public Square(String color, boolean filled, double side) {  
 super(side, side, color, filled);  
 this.side = side;  
 }  
  
 @Override  
 public void setWidth(double side) {  
 super.setWidth(width);  
 }  
  
 public double getSide() {  
 return side;  
 }  
  
 @Override  
 public void setLength(double side) {  
 super.setLength(length);  
 }  
  
 @Override  
 public String toString() {  
 return "Shape: square, side: " + this.side + ", color: " + this.color;  
 }  
}

Movable.java – интерфейс для реализации движения форм.

package com.company;  
  
public interface Movable {  
  
 public void moveUp();  
 public void moveDown();  
 public void moveLeft();  
 public void moveRight();  
}

MovableCircle.java

package com.company;  
  
public class MovableCircle implements Movable{  
  
 private int radius;  
 private MovablePoint center;  
  
 public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {  
 center = new MovablePoint(x, y, xSpeed, ySpeed);  
 this.radius = radius;  
 }  
  
 @Override  
 public String toString() {  
 return "Center coords: (" + center.x + ", " + center.y +")";  
 }  
  
 @Override  
 public void moveUp() {  
 center.moveUp();  
 }  
  
 @Override  
 public void moveDown() {  
 center.moveDown();  
 }  
  
 @Override  
 public void moveLeft() {  
 center.moveLeft();  
 }  
  
 @Override  
 public void moveRight() {  
 center.moveRight();  
 }  
}

MovablePoint.java – движущийся прямоугольник, реализующий интерфейс Movable.

package com.company;  
  
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 "Point ccords: (" + this.x+ ", "+ this.y+")";  
 }  
  
 @Override  
 public void moveUp() {  
 this.y = this.y + this.ySpeed;  
 }  
  
 @Override  
 public void moveDown() {  
 this.y = this.y - this.ySpeed;  
 }  
  
 @Override  
 public void moveLeft() {  
 this.x = this.x - this.xSpeed;  
 }  
  
 @Override  
 public void moveRight() {  
 this.x = this.x + this.xSpeed;  
 }  
}

MovablePoint.java

package com.company;  
  
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 "Point ccords: (" + this.x+ ", "+ this.y+")";  
 }  
  
 @Override  
 public void moveUp() {  
 this.y = this.y + this.ySpeed;  
 }  
  
 @Override  
 public void moveDown() {  
 this.y = this.y - this.ySpeed;  
 }  
  
 @Override  
 public void moveLeft() {  
 this.x = this.x - this.xSpeed;  
 }  
  
 @Override  
 public void moveRight() {  
 this.x = this.x + this.xSpeed;  
 }  
}

## **ВЫВОД**

В ходе данной работы я ознакомилась с понятием абстрактный класс, интерфейсов, а также закрепила навыки их использования.