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

Выполнил студент группы ИНБО-02-20 Лукьяненко Д.В.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.
Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Цель данной практической работы – освоить на практике работу с абстрактными классами и наследованием на Java.

Задание

Задание для первого пункта

Абстрактный суперкласс Shape и его подклассы

Задание перепишите суперкласс Shape и его подклассы так как это представлено на диаграмме Circle, Rectangle and Square.

В этом задании, класс Shape определяется как абстрактный класс , который содержит:

Два *protected* (защищенных) переменных color(String) и filled(boolean). Защищенные переменные могут быть доступны в подклассах и классах в одном пакете . Они обозначаются со знаком '#' в диаграмме классов в нотации языка UML.

Методы геттеры и сеттеры для всех переменных экземпляра класса, и метод toString().

Два абстрактных метода getArea() и getPerimeter() выделены курсивом в диаграмме класса.

В подклассах Circle(круг) и Rectangle(прямоугольник) должны переопределяться абстрактные методы getArea() и getPerimeter(), чтобы обеспечить их надлежащее выполнение для конкретных экземпляров типа подкласс. Также необходимо для каждого подкласса переопределить toString().

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.

Задание для второго пункта

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. Убедитесь, что две точки имеет одну и ту же скорость (нужен метод это проверяющий).

Репозиторий

Ссылка: https://github.com/neluckoff/mirea-java-lessons/tree/master/src/ru/luckoff/mirea/practice_3

Выполнение работы

Выполнение первого пункта

Первым делом мною был создан абстрактный суперкласс Shape и добавлены protected переменные color и filled и абстрактные методы getArea() и getPerimeter(). Полностью пользуясь представленной в задании UML Диаграммой я добавил конструтор, геттеры и сеттеры, которые были там представлены.

Далее я создал классы Circle, Rectangle и Square, соблюдая все наследования, указанные в UML Диаграммах. Также мною были заполнены все абстрактные методы в этих классах.

После того, как программа была готова я перешел к ее тестам.

```
Shape s1 = new Circle( color: "RED",   filled: false,  radius: 5.5); // Upcast Circle to Shape
System.out.println(s1); // which version?
System.out.println(s1.getArea()); // which version?
System.out.println(s1.getPerimeter()); // which version?
System.out.println(s1.getColor());
System.out.println(s1.isFilled());
System.out.println(s1.getRadius());
```

Рисунок 1 – Первый тест

На рисунке 1 представлен первый тест программы. Ошибка возникла из-за того, что в классе Shape отсутствует метод getRadius(). Чтобы починить программу нужно привести тип ((Circle)s1).

Все ошибки в тестах были одинаковыми, с одной и той же проблемой, поэтому я решил не уделять этому описанию много времени.

Ниже мною будет представлена UML Диаграмма, которая была создана после написании программы.

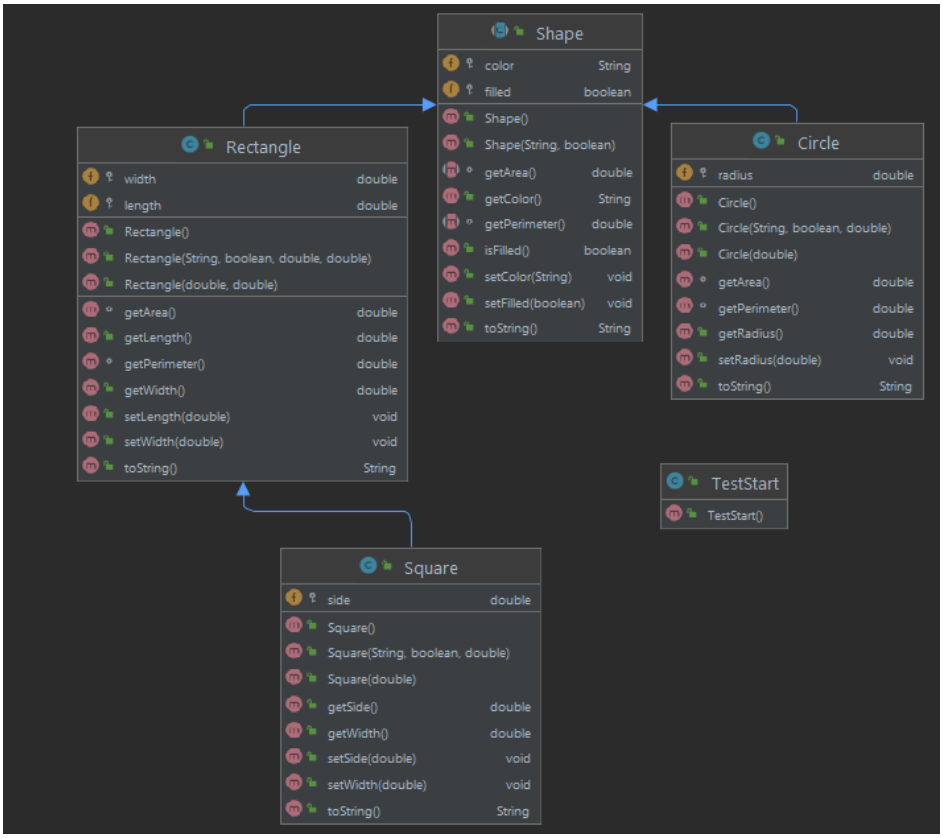


Рисунок 2 – Первая полученная UML Диаграмма

Выполнение второго пункта

Второе задание было выполнено аналогично с первым, за исключением того, что теперь классы были связаны между собой при помощи имплементации и агрегации.

Ниже мною будет представлена UML Диаграмма, которая была создана после написании программы.

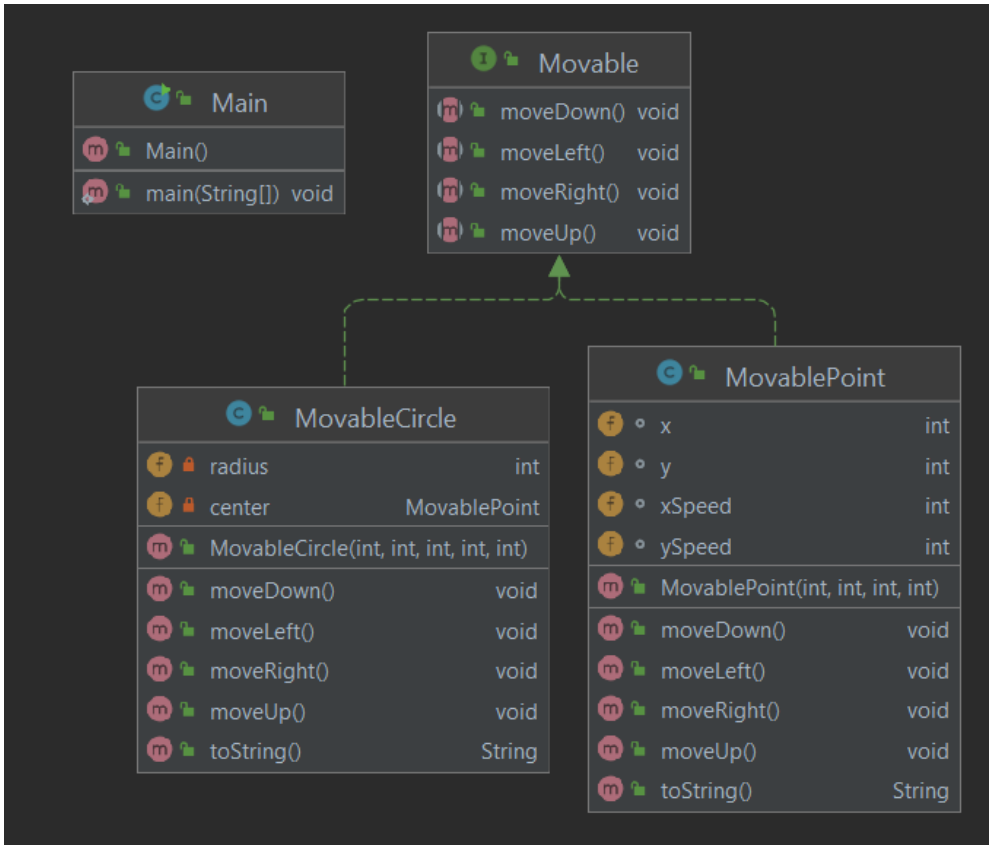


Рисунок 3 – Вторая полученная UML Диаграмма

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть как выглядит код выполненного задания.

Код задачи первого пункта

```java
package ru.luckoff.mirea.lesson_3.taskOne;

public abstract class Shape {
    protected String color;
    protected boolean filled;
    abstract double getArea();
    abstract double getPerimeter();

    public Shape() {
        this.color = "Red";
        this.filled = true;
    }
    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; }

    @Override
    public String toString() {
        return "Shape{" +
                "color='" + color + '\'' +
                ", filled=" + filled +
                '}';
    }
}
```

Рисунок 4 – Абстрактный класс Shape

```java
package ru.luckoff.mirea.lesson_3.taskOne;

public class Circle extends Shape {
    protected double radius;

    public Circle() {
        this.color = "Red";
        this.filled = true;
        this.radius = 0;
    }
    public Circle(double radius) {
        this.color = "Red";
        this.filled = true;
        this.radius = radius;
    }
    public Circle(String color, boolean filled, double radius) {
        this.color = color;
        this.filled = filled;
        this.radius = radius;
    }

    public double getRadius() { return radius; }
    public void setRadius(double radius) { this.radius = radius; }
    @Override
    double getArea() { return 3.14*radius*radius; }
    @Override
    double getPerimeter() { return 2*3.14*radius; }
    @Override
    public String toString() {
        return "Circle{"+"radius="+radius+", color='"+color+'\''+", filled="+filled+'}';
    }
}
```

Рисунок 5 – Класс Circle

```java
package ru.luckoff.mirea.lesson_3.taskOne;

public class Rectangle extends Shape {
    protected double width;
    protected double length;

    public Rectangle() {
        this.color = "Red";
        this.filled = true;
        this.width = 0;
        this.length = 0;
    }
    public Rectangle(double width, double length) {
        this.color = "Red";
        this.filled = true;
        this.width = width;
        this.length = length;
    }
    public Rectangle(String color, boolean filled, double width, double length) {
        this.color = color;
        this.filled = filled;
        this.width = width;
        this.length = length;
    }
```

Рисунок 6 – Класс Rectangle (1)

```java
    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
    double getArea() { return width*length; }
    @Override
    double getPerimeter() { return 2*width+2*length; }
    @Override
    public String toString() {
        return "Rectangle{" +
                "width=" + width +
                ", length=" + length +
                ", color='" + color + '\'' +
                ", filled=" + filled +
                '}';
    }
}
```

Рисунок 7 – Класс Rectangle (2)

```java
package ru.luckoff.mirea.lesson_3.taskOne;

import ru.luckoff.mirea.lesson_3.taskOne.Rectangle;

public class Square extends Rectangle {
    protected double side;

    public Square() {
        this.color = "Red";
        this.filled = true;
        this.side = 0;
    }
    public Square(double side) {
        this.color = "Red";
        this.filled = true;
        this.side = side;
    }
    public Square(String color, boolean filled, double side) {
        this.color = color;
        this.filled = filled;
        this.side = side;
    }
    public double getSide() { return side; }
    public void setSide(double side) { this.side = side; }
    public double getWidth() { return width; }
    public void setWidth(double width) { this.width = width; }
    @Override
    public String toString() {
        return "Square{"+"width="+width+", length="+ length
                +", color='"+color+'\''+", filled="+filled+", side="+side+'}';
    }
}
```

Рисунок 8 – Класс Square

Код задачи второго пункта

```java
package ru.luckoff.mirea.lesson_3.taskTwo;

public interface Movable {
    public void moveUp();
    public void moveDown();
    public void moveLeft();
    public void moveRight();
}
```

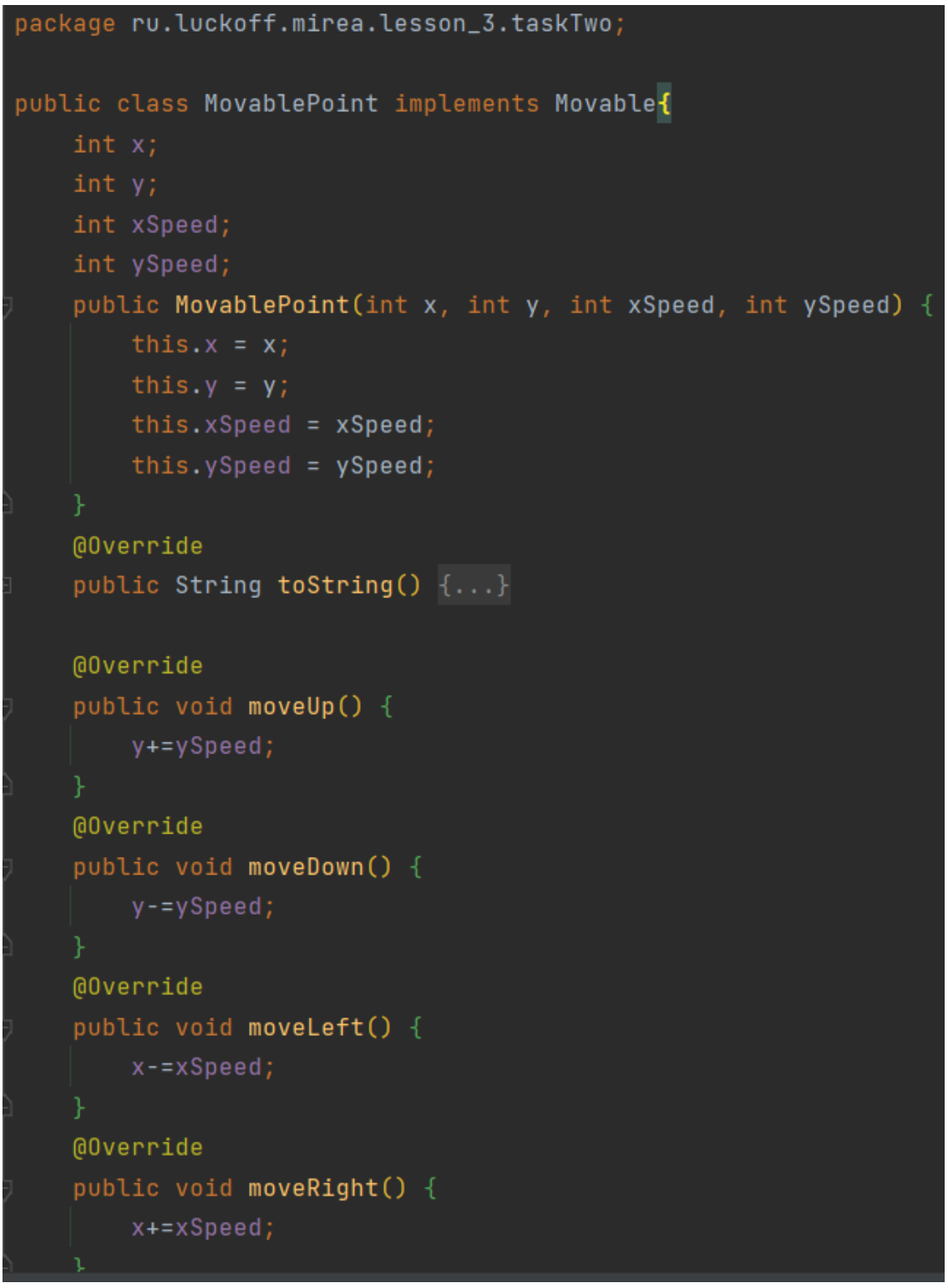
Рисунок 9 – Интерфейс Movable

```java
package ru.luckoff.mirea.lesson_3.taskTwo;

public class MovableCircle implements Movable {
    private int radius;
    private MovablePoint center;

    public MovableCircle(int radius, int x, int y, int xSpeed, int ySpeed) {
        this.radius = radius;
        this.center = new MovablePoint(x, y, xSpeed, ySpeed);
    }
    @Override
    public String toString() {
        return "MovableCircle{" +", radius=" + radius + ", center=" + center +
    }
    @Override
    public void moveUp() {
        center.moveUp();
    }
    @Override
    public void moveDown() {
        center.moveDown();
    }
    @Override
    public void moveLeft() {
        center.moveLeft();
    }
    @Override
    public void moveRight() {
        center.moveRight();
    }
}
```

Рисунок 10 – Класс MovableCircle



```java
package ru.luckoff.mirea.lesson_3.taskTwo;

public class MovablePoint implements Movable{
    int x;
    int y;
    int xSpeed;
    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() {...}

    @Override
    public void moveUp() {
        y+=ySpeed;
    }
    @Override
    public void moveDown() {
        y-=ySpeed;
    }
    @Override
    public void moveLeft() {
        x-=xSpeed;
    }
    @Override
    public void moveRight() {
        x+=xSpeed;
    }
```

Рисунок 11 – Класс MovablePoint.

Вывод

В результате выполнения данной практической работы я освоил на практике работу с абстрактными классами и наследованием на Java.