|  |
| --- |
|  |
| 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 №5**

Выполнил студент группы ИВБО-07-19 Гудаев И.И.

Принял Степанин П.С.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2020г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2020г.

Отметка о выполнении

**Москва – 2020 г.**

## **Задание**

Вам нужно написать два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable на основе классов, разработанных в практической работе № 4. Изучите UML диаграмму и представьте реализацию класса

## **Ход Работы**

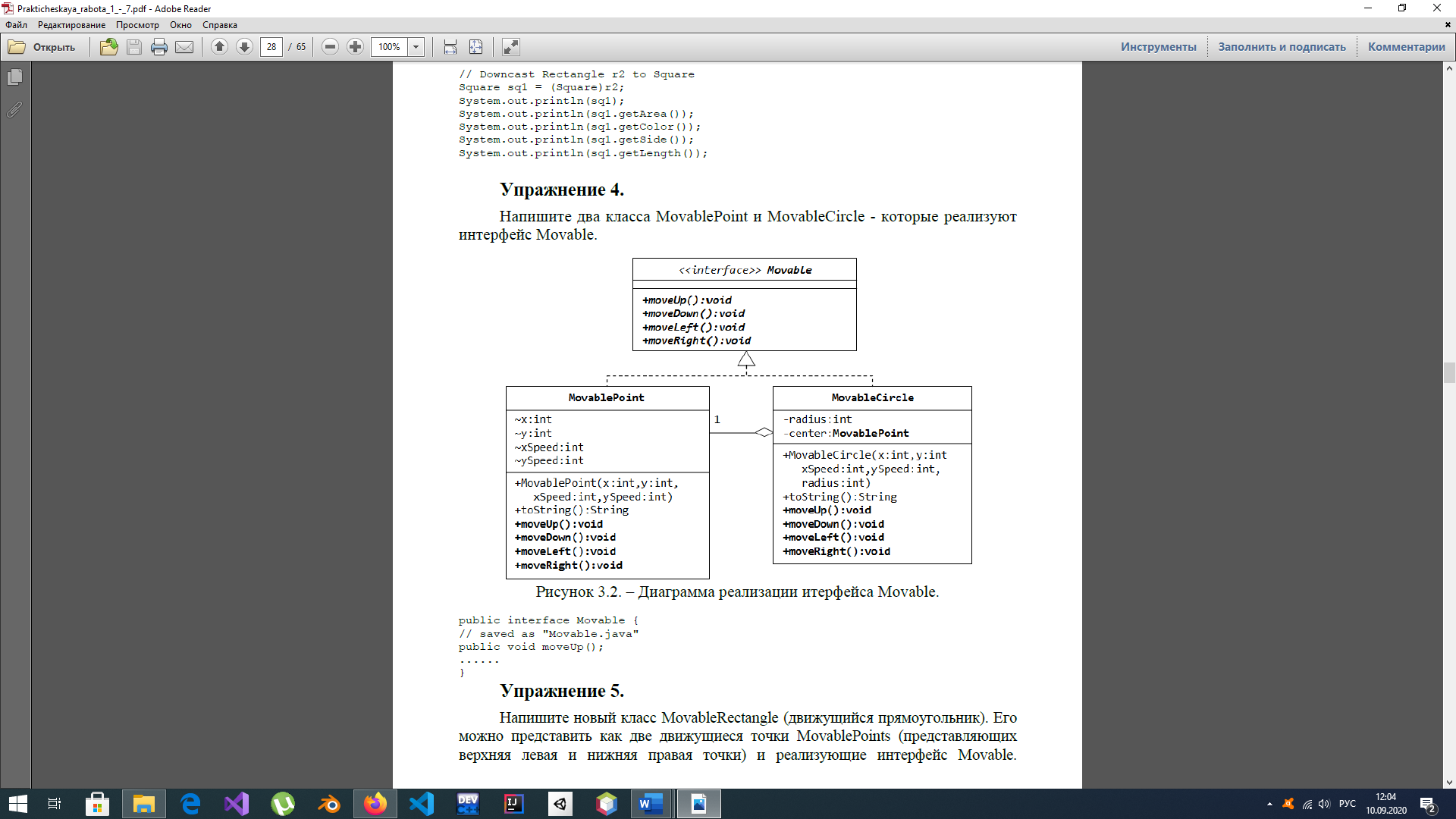


Рисунок 1. UML диаграмма

На UML диаграмме представлен интерфейс Movable с методами движения по плоскости. Классы MovablePoint и MovableCircle имплементируются от этого интерфейса, причем класс MovablePoint связан агрегацией с классом MovableCircle. Реализация данной диаграммы представлена на листингах 1-4.

В ходе выполнения работы были получены следующие исходные коды:

**package** com.company;

**public** **interface** Movable {

void moveUp();

void moveDown();

void moveLeft();

void moveRight();

}

Листинг 1. Задание 1 (интерфейс Movable)

**package** com.company;

**import** java.time.Clock;

**public** **class** MovablePoint **implements** Movable{

int x, y, xSpeed, 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 moveDown() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

x += 0;

y += ySpeed;

}

}

**@Override**

**public** void moveLeft() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

x -= xSpeed;

y += 0;

}

}

**@Override**

**public** void moveRight() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

x += xSpeed;

y += 0;

}

}

**@Override**

**public** void moveUp() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

x += 0;

y -= ySpeed;

}

}

}

Листинг 2. Задание 1 (класс MovablePoint)

**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) {

**this**.radius = radius;

center.x = x;

center.y = y;

center.ySpeed = ySpeed;

center.xSpeed = xSpeed;

}

**@Override**

**public** void moveDown() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

center.x += 0;

Листинг 3.1. Задание 1 (класс MovableCircle)

center.y += center.ySpeed;

}

}

**@Override**

**public** void moveLeft() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

center.x -= center.xSpeed;

center.y += 0;

}

}

**@Override**

**public** void moveRight() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

center.x += center.xSpeed;

center.y += 0;

}

}

**@Override**

**public** void moveUp() {

**for**(float i = 0; i < 10f; i+= 0.01f)

{

center.x += 0;

center.y -= center.ySpeed;

}

}

}

Листинг 3.2. Задание 1 (класс MovableCircle)

**package** com.company;

**public** **class** Main {

**public** **static** void main(**String**[] args) {

// write your code here

MovableCircle cr = **new** MovableCircle(0, 0, 1, 1, 2);

cr.moveDown();

}

}

Листинг 4. Задание 1 (класс Main)

## **Вывод**

Мы разобрались с понятием интерфейса, научились его создавать и реализовывать с помощью имплементированных классов, написав соответствующую программу.