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

**Институт информационных технологий**

КАФЕДРА ИНСТРУМЕТНАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИППО)

ПРАКТИЧЕСКАЯ РАБОТА № 3

по дисциплине «Объектно-ориентированное программирование»

«Абстрактные суперклассы и его подклассы в Java»

ВАРИАНТ 2

Выполнил студент группы ИНБО-01-18 *Трофимов С. М.*

Принял Батанов А.О.

Практическая работа выполнена «\_\_\_»\_\_\_\_\_\_\_2019г.

«Зачтено» «\_\_\_»\_\_\_\_\_\_\_2019г.

Москва 2019

**Цель работы**

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

**Теоретическое введение**

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы помечаются ключевым словом abstract.

Абстрактный метод не завершён. Он состоит только из объявления и не имеет тела:

abstract void yourMethod();

По сути, мы создаём шаблон метода. Например, можно создать абстрактный метод для вычисления площади фигуры в абстрактном классе Фигура. А все другие производные классы от главного класса могут уже реализовать свой код для готового метода. Ведь площадь у прямоугольника и треугольника вычисляется по разным алгоритмам и универсального метода не существует.

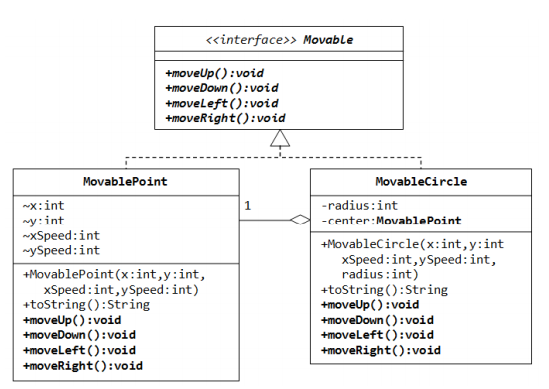
Если вы объявляете класс, производный от абстрактного класса, но хотите иметь возможность создания объектов нового типа, вам придётся предоставить определения для всех абстрактных методов базового класса. Если этого не сделать, производный класс тоже останется абстрактным, и компилятор заставит пометить новый класс ключевым словом abstract.

Абстрактный класс не может содержать какие-либо объекты, а также абстрактные конструкторы и абстрактные статические методы. Любой подкласс абстрактного класса должен либо реализовать все абстрактные методы суперкласса, либо сам быть объявлен абстрактным.

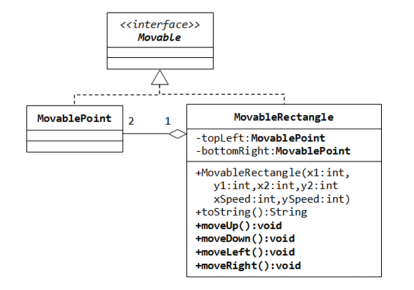
**Выполнение практической работы**

*Задание:*

Вам нужно написать два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable.



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).



*Решение:*

Файл Movable.java

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

Файл MovablePoint.java

**package** com.gradess.company;  
  
**public class** MovablePoint **implements** Movable {  
  
 **private int x**;  
 **private int y**;  
 **private int xSpeed**;  
 **private 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 void** moveUp() {  
 **y** += **ySpeed**;  
 }  
  
 @Override  
 **public void** moveDown() {  
 **y** -= **ySpeed**;  
 }  
  
 @Override  
 **public void** moveLeft() {  
 **x** += **xSpeed**;  
 }  
  
 @Override  
 **public void** moveRight() {  
 **x** -= **ySpeed**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "MovablePoint{"** +  
 **"x="** + **x** +  
 **", y="** + **y** +  
 **", xSpeed="** + **xSpeed** +  
 **", ySpeed="** + **ySpeed** +  
 **'}'**;  
 }  
}

Файл MovableCircle.java

**package** com.gradess.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**.**center** = **new** MovablePoint(x, y, xSpeed, ySpeed);  
 **this**.**radius** = 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();  
 }  
  
 @Override  
 **public** String toString() {  
 **return "MovableCircle{"** +  
 **"radius="** + **radius** +  
 **", center="** + **center** +  
 **'}'**;  
 }  
}

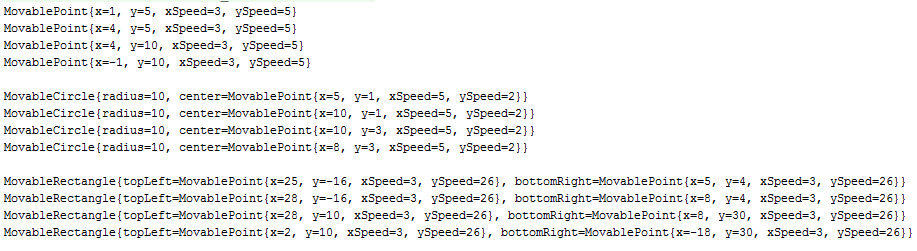
Файл MovableRectangle.java

**package** com.gradess.company;  
  
**public class** MovableRectangle **implements** Movable {  
  
 **private** MovablePoint **topLeft**;  
 **private** MovablePoint **bottomRight**;  
  
 **public** MovableRectangle(**int** x1, **int** x2, **int** y1, **int** y2, **int** xSpeed, **int** ySpeed) {  
 **this**.**topLeft** = **new** MovablePoint(x1, y1, xSpeed, ySpeed);  
 **this**.**bottomRight** = **new** MovablePoint(x2, y2, xSpeed, ySpeed);  
 }  
  
 @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();  
 }  
  
 @Override  
 **public** String toString() {  
 **return "MovableRectangle{"** +  
 **"topLeft="** + **topLeft** +  
 **", bottomRight="** + **bottomRight** +  
 **'}'**;  
 }  
}

Файл Main.java

**package** com.gradess.company;  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 MovablePoint point = **new** MovablePoint(1,10,3,5);  
 *moveWithLog*(point);  
 System.***out***.println();  
  
 MovableCircle circle = **new** MovableCircle(5, 3, 5, 2, 10);  
 *moveWithLog*(circle);  
 System.***out***.println();  
  
  
 MovableRectangle rectangle = **new** MovableRectangle(25,5, 10, 30, 3, 26);  
 *moveWithLog*(rectangle);  
 System.***out***.println();  
  
  
 }  
  
 **private static void** moveWithLog(Movable movableObject) {  
 movableObject.moveDown();  
 *print*(movableObject);  
  
 movableObject.moveLeft();  
 *print*(movableObject);  
  
 movableObject.moveUp();  
 *print*(movableObject);  
  
 movableObject.moveRight();  
 *print*(movableObject);  
  
 }  
  
 **private static void** print(Object circle) {  
 System.***out***.println(circle);  
 }  
}

**Результат выполнения программы**



**Выводы по работе**

Я освоил на практике работу с абстрактными классами и наследованием на Java.

**Используемая литература:**

1. К. Сьерра, Б. Бэйтс. Изучаем Java, 2-е издание.
2. С. Макконнелл. Совершенный код, 3-е издание.