|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |
| Институт инновационных технологий и государственного управления (ИНТЕГУ) | |
| Кафедра информационных технологий в государственном управлении (ИТГУ) | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3** | |
| **по дисциплине** | |
| **«Объективно-ориентированное программирование»** | |
|  | |
| Выполнил студент группы ГИБО-04-18 | *Алиев В.Г.* |
| Принял  *Старший преподаватель* | *Зорина Н.В.*  *.* |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_\_ г. |  |

Москва 2020

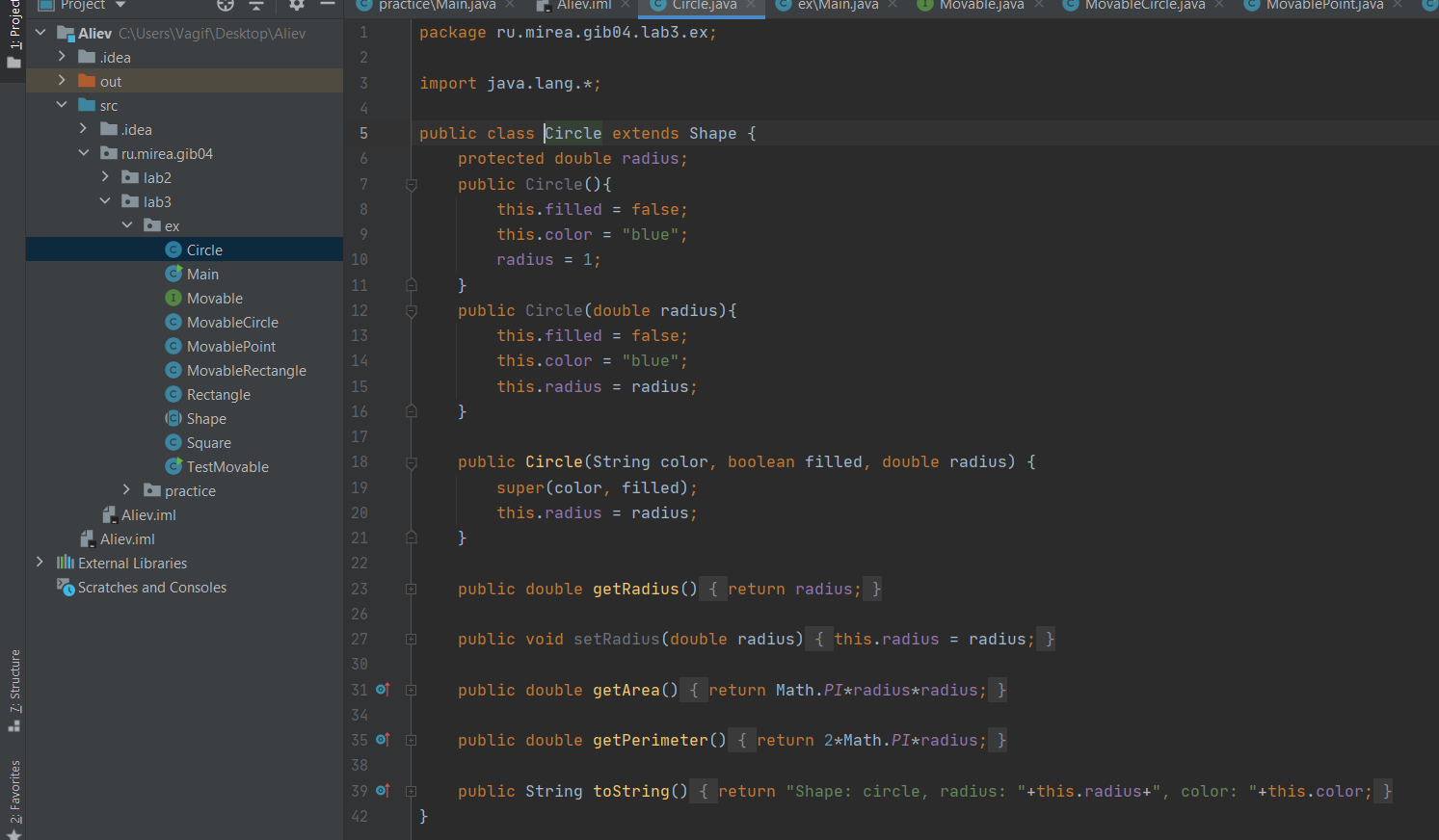
# Цель работы

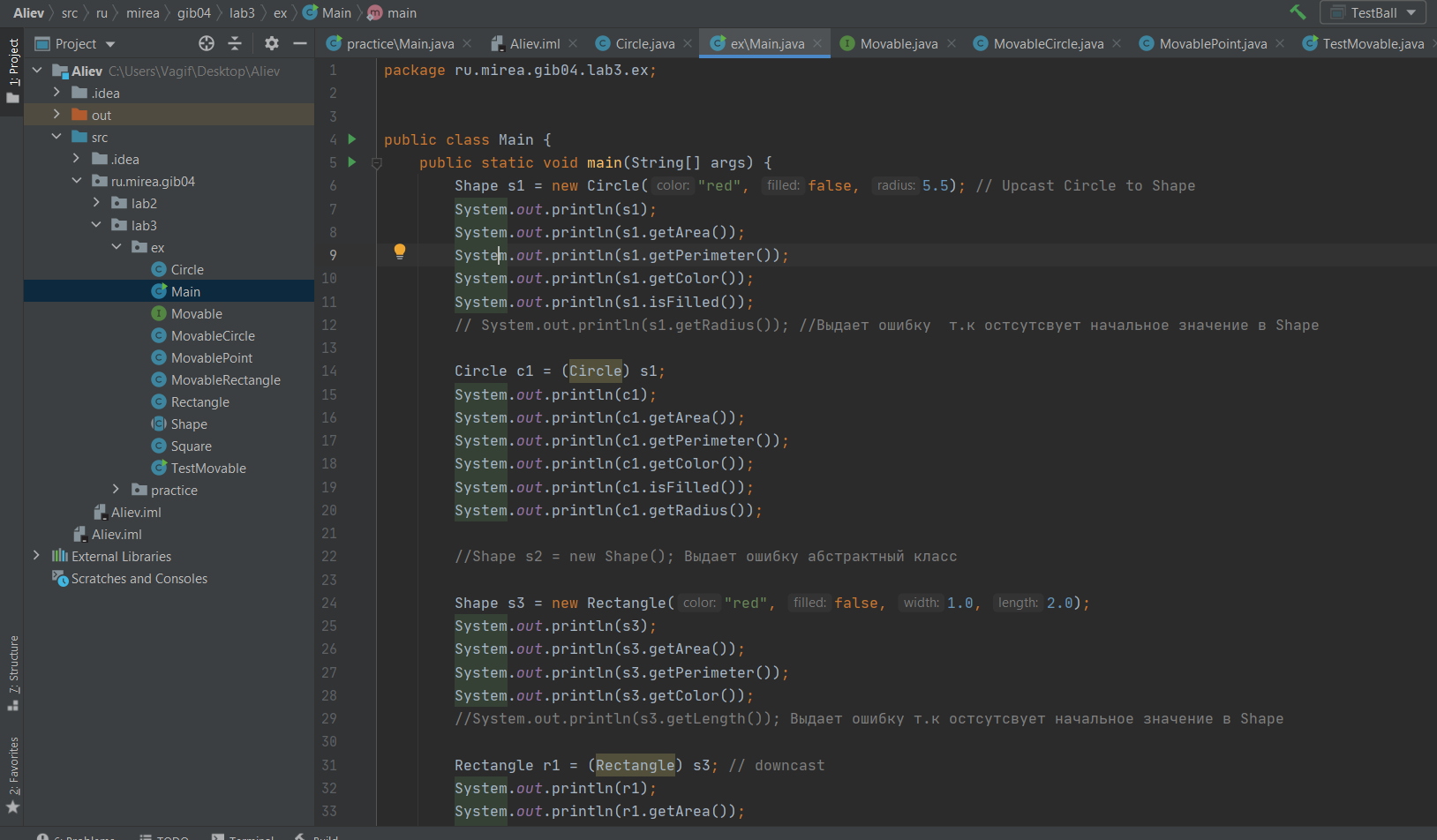
Цель данной практической работы – освоить на практике работу с абстрактными классами и наследованием на Java

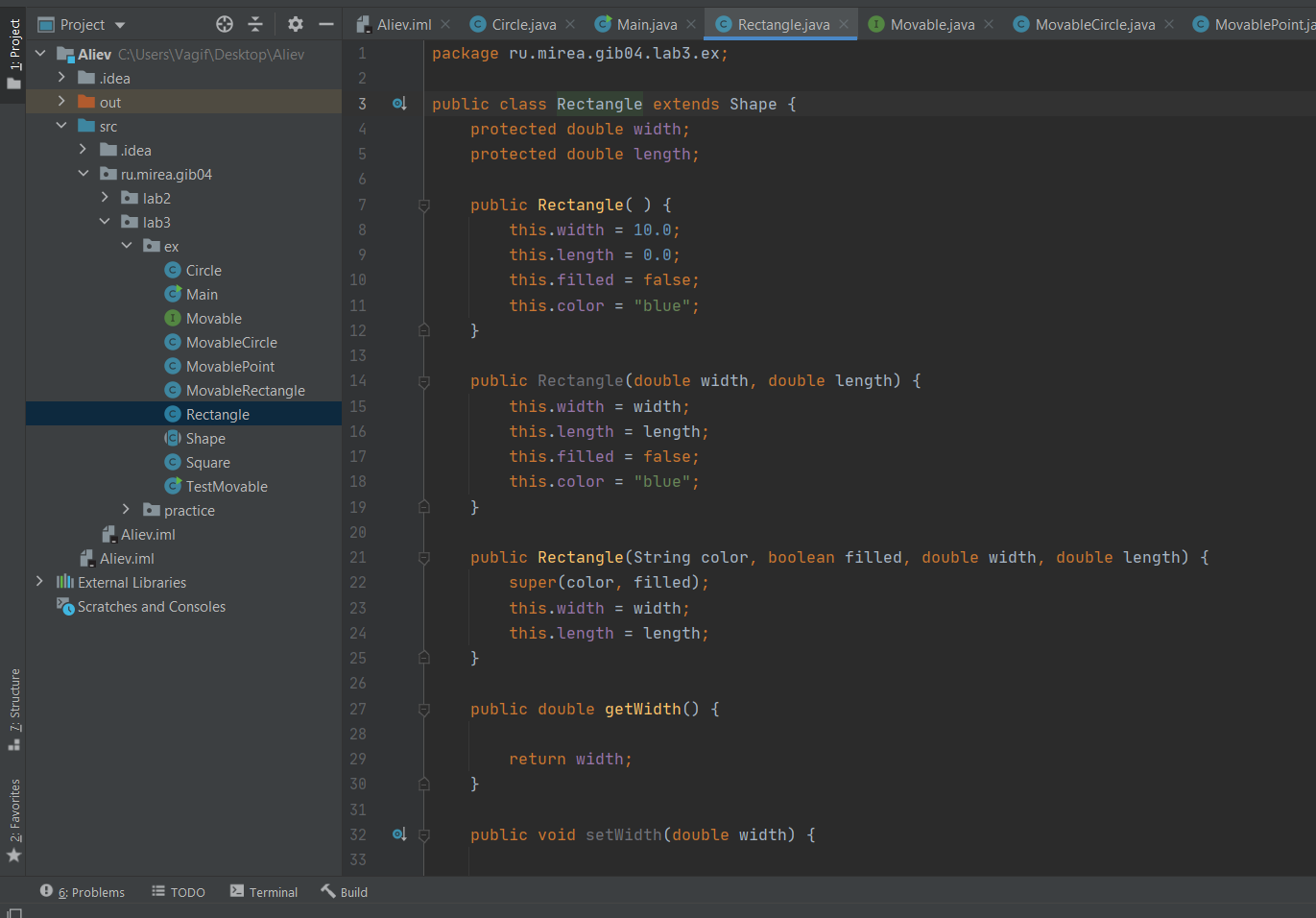
# Код и результат

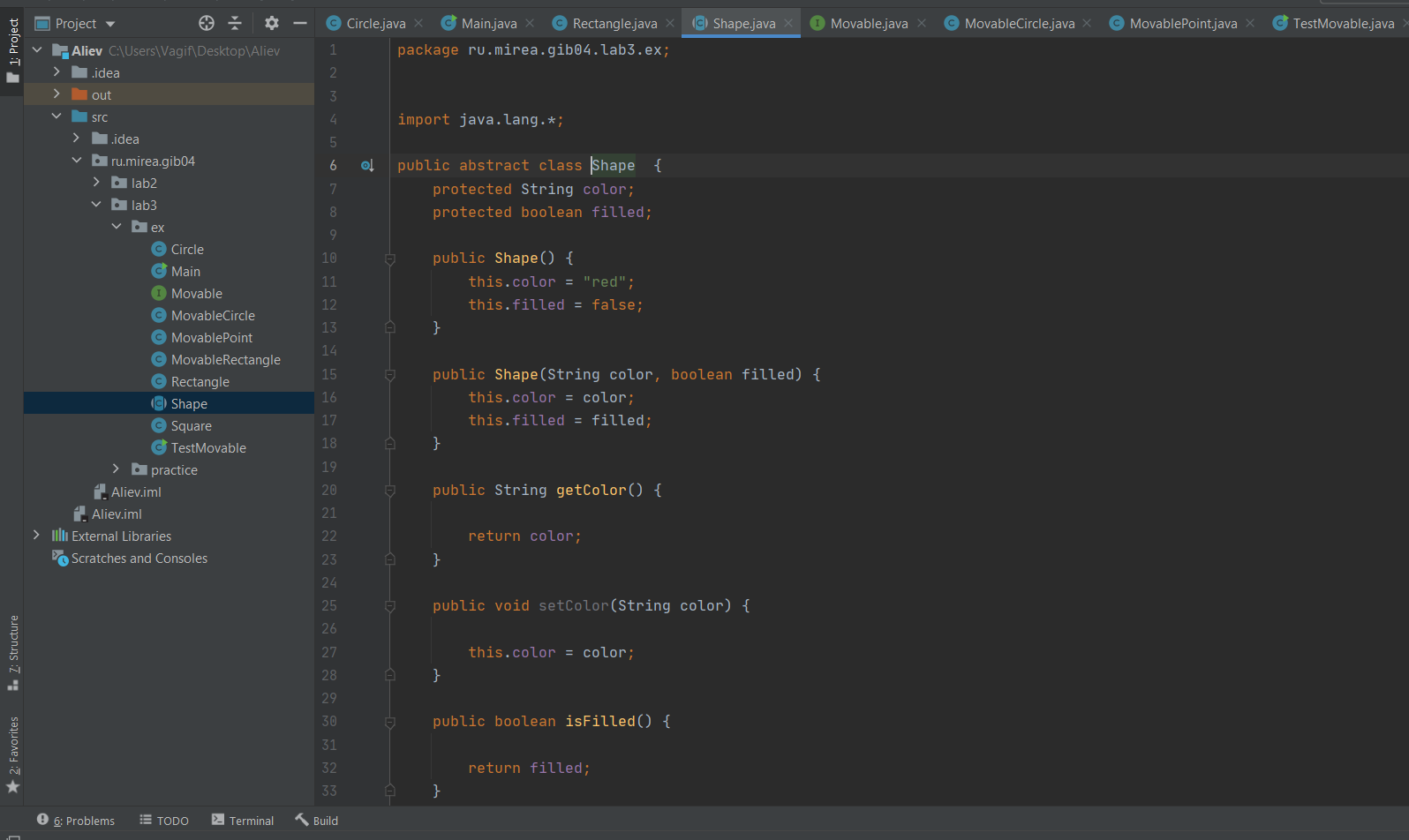
**Скриншоты:**

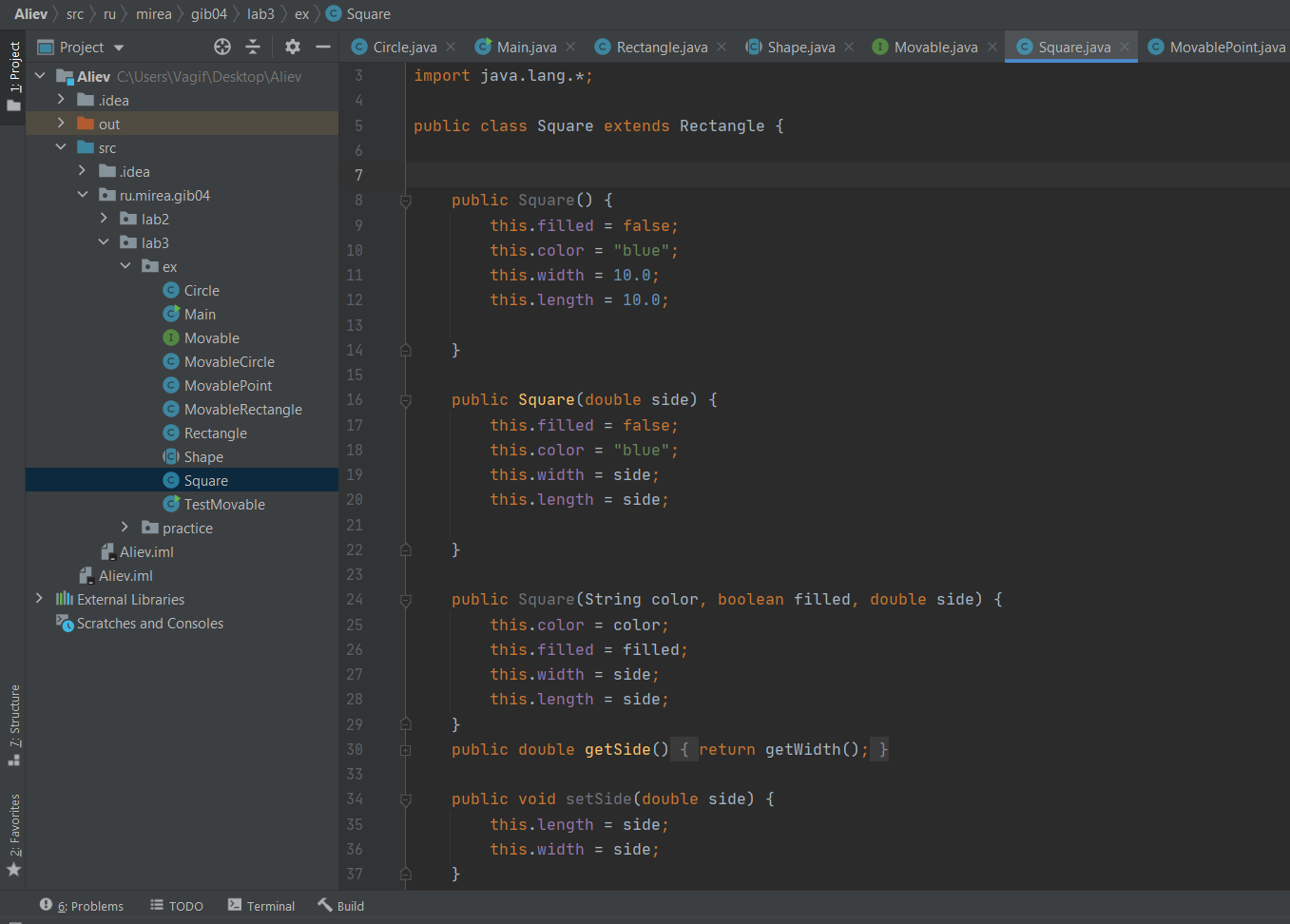
**Задание 1**

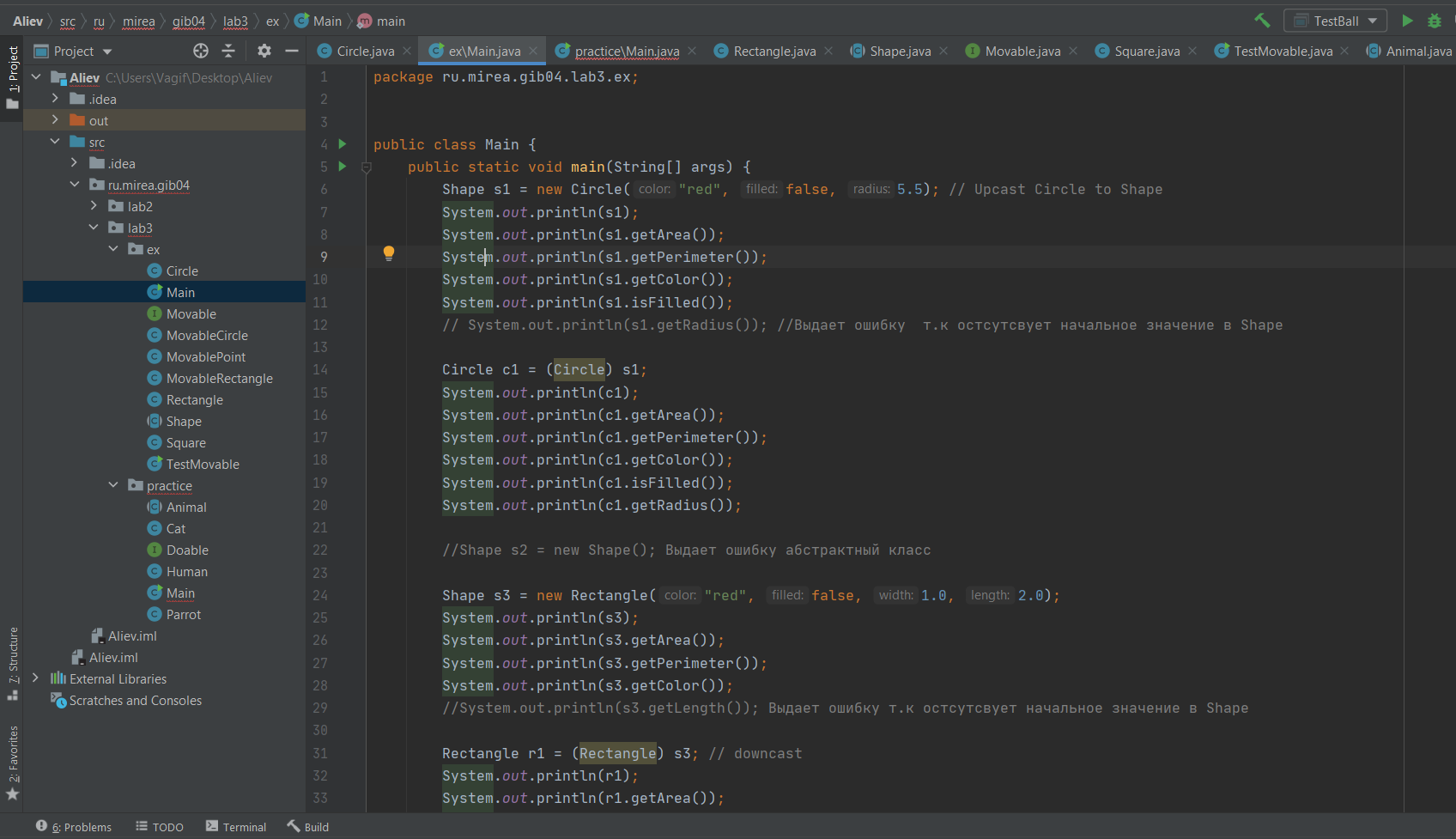


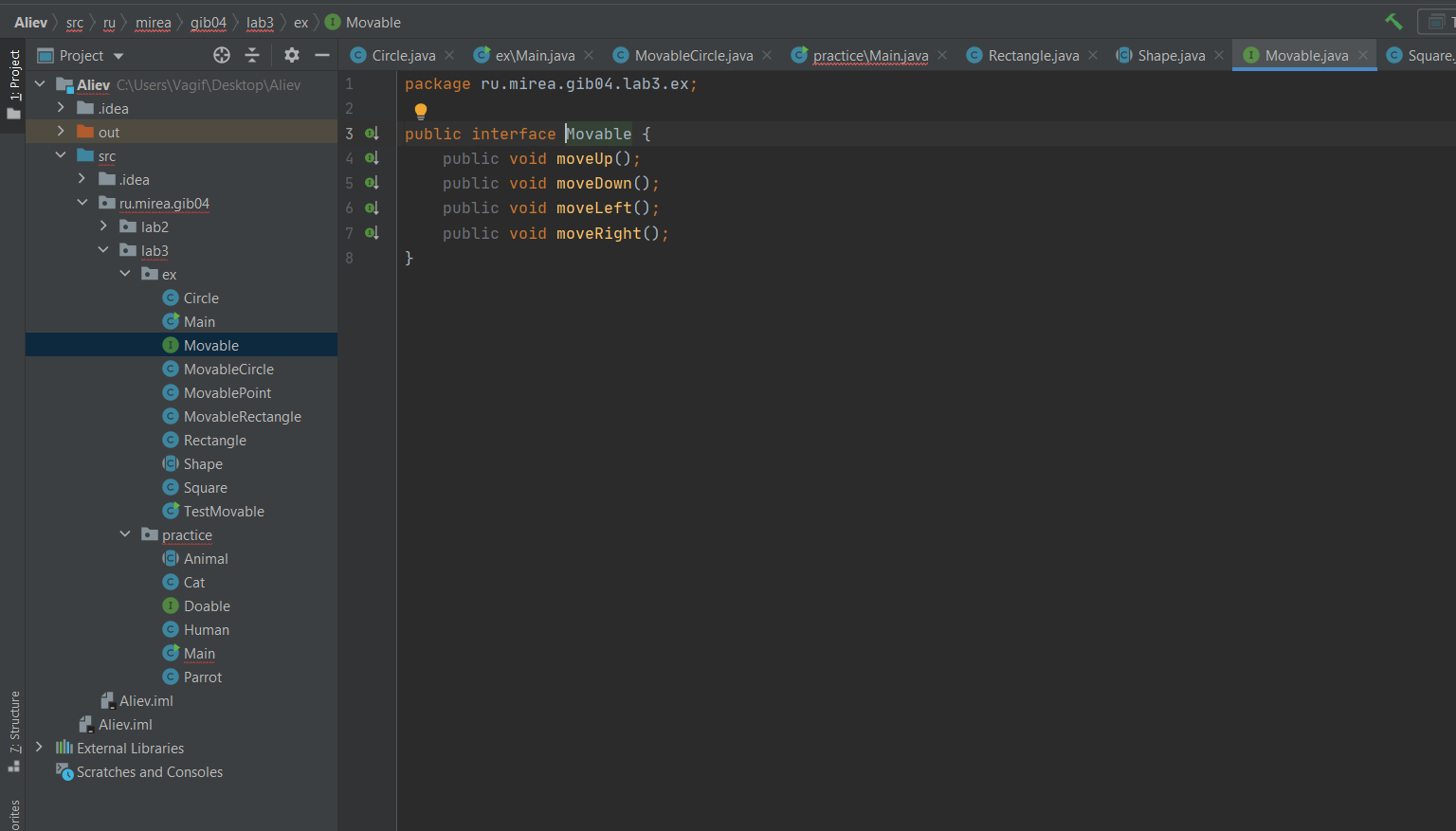


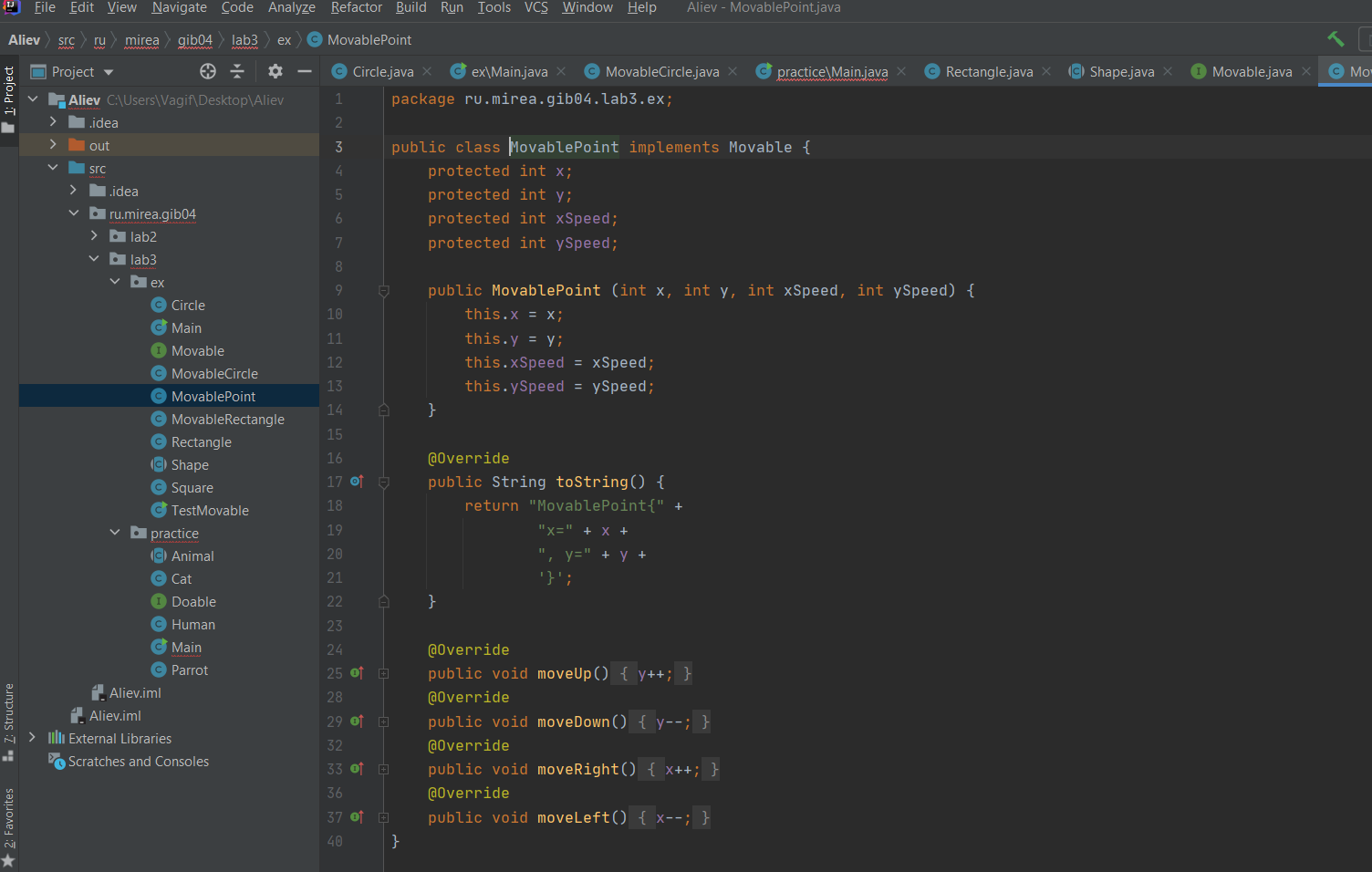


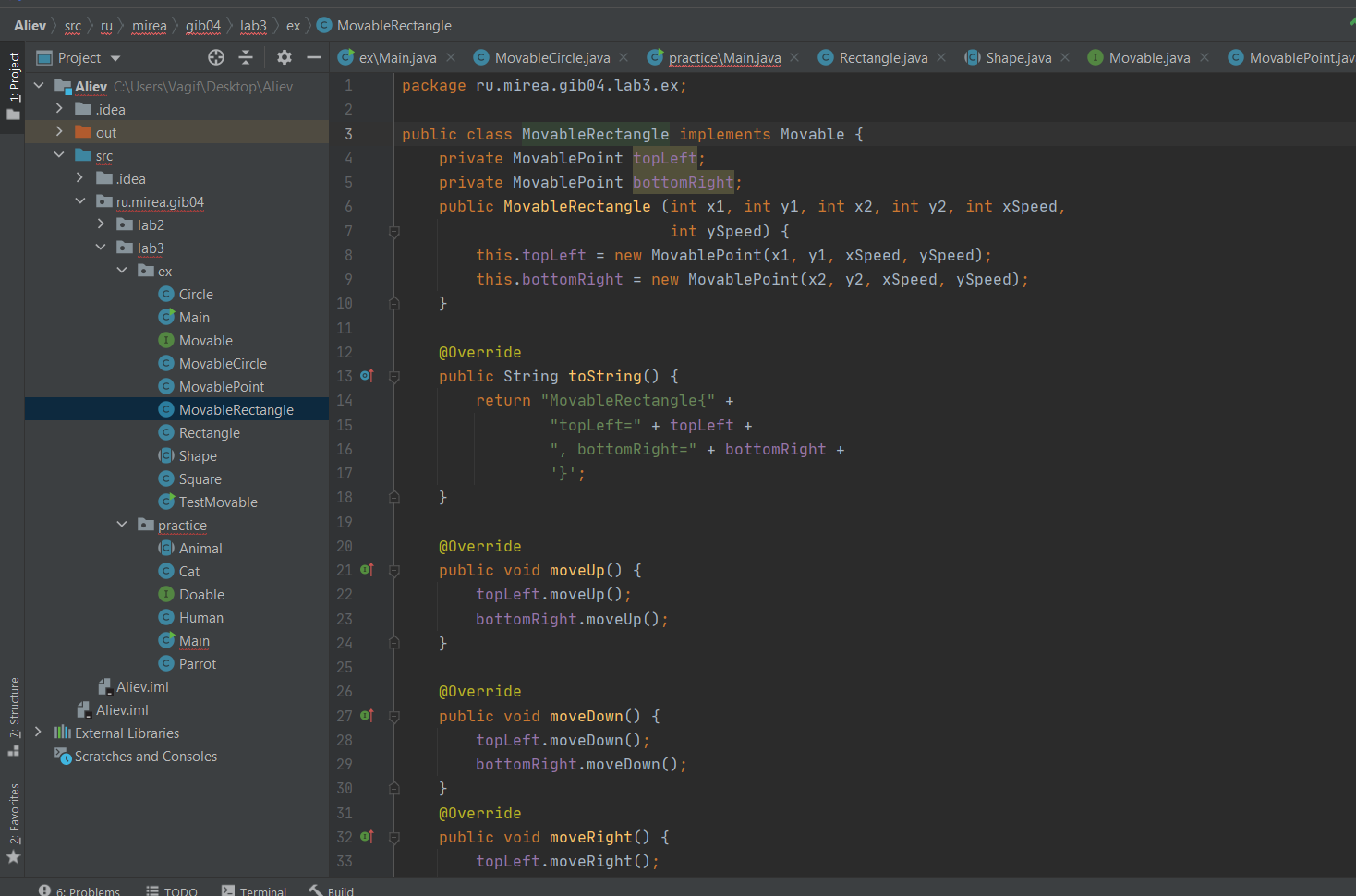




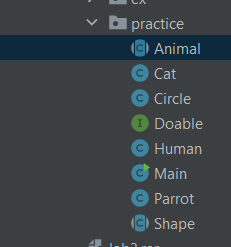


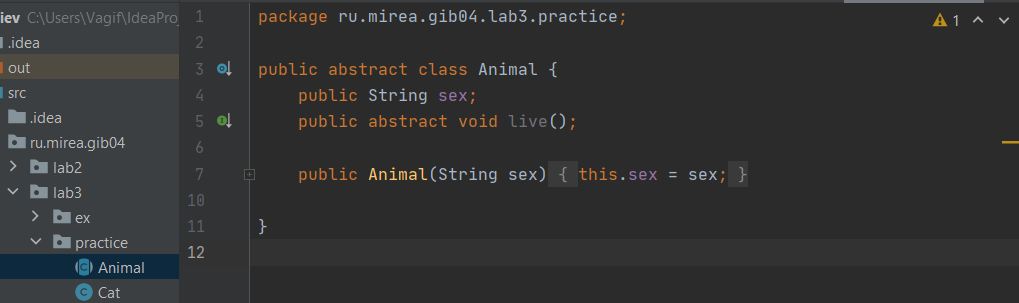


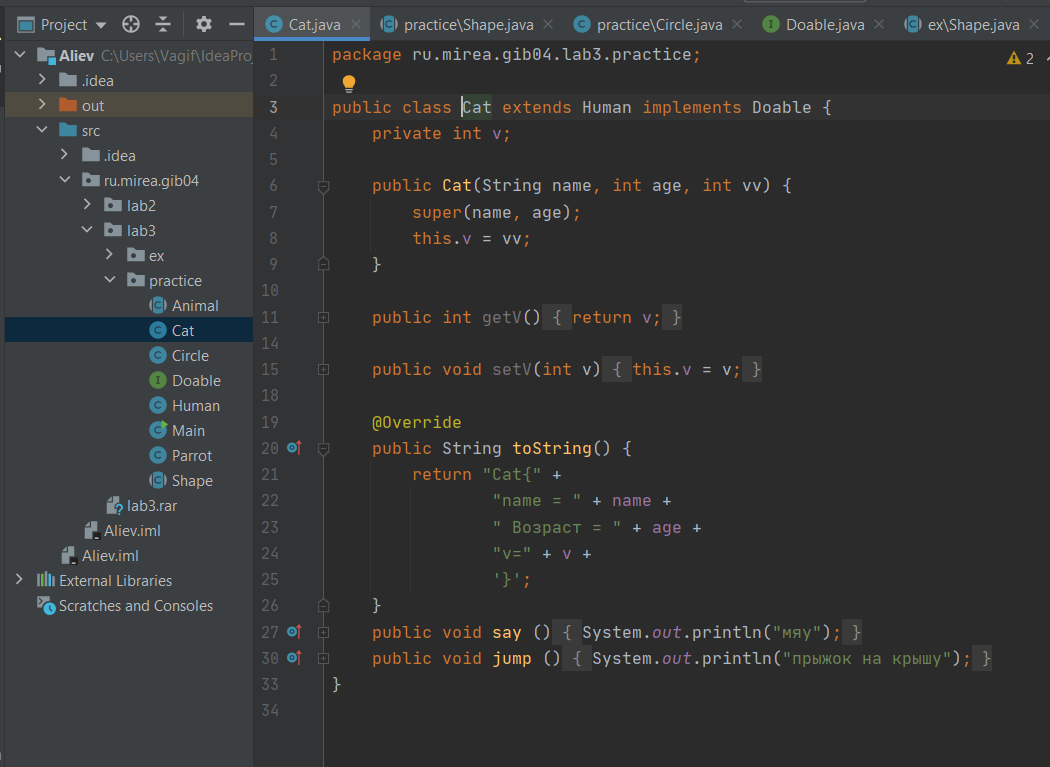


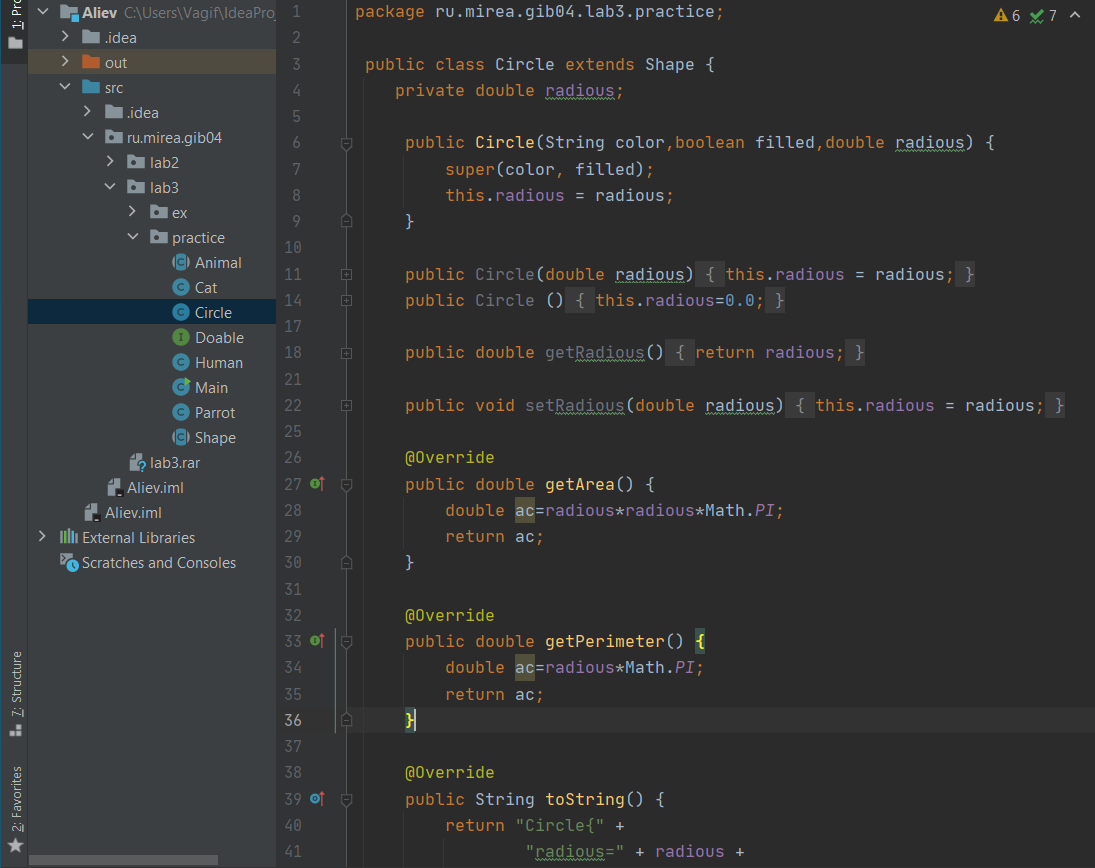


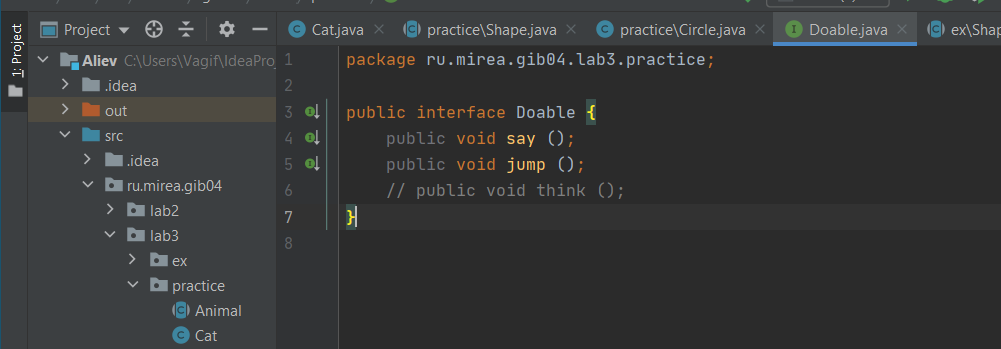
**ПРАКТИКА:**

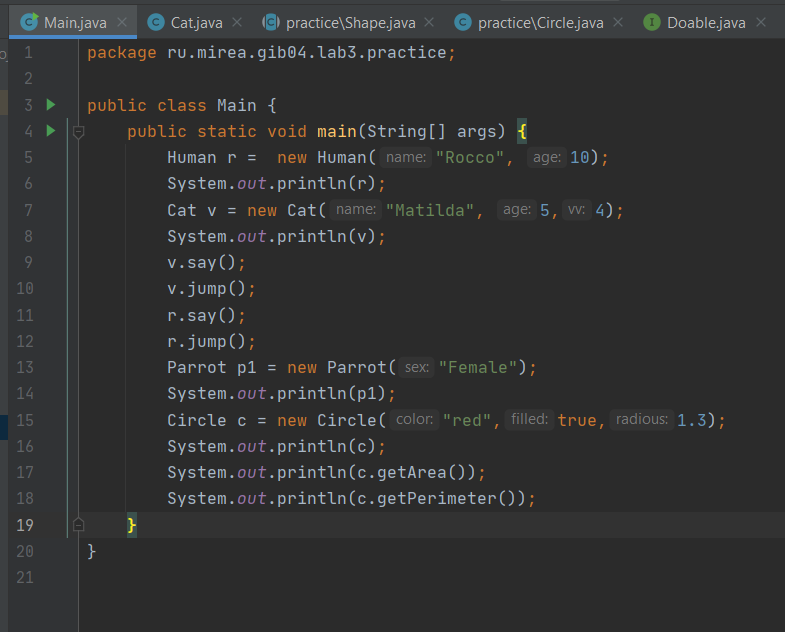
****

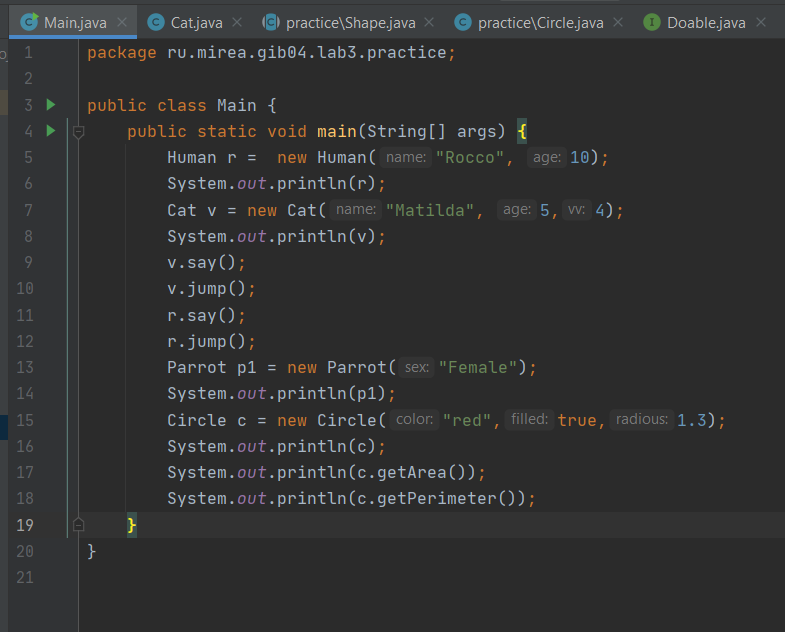
****

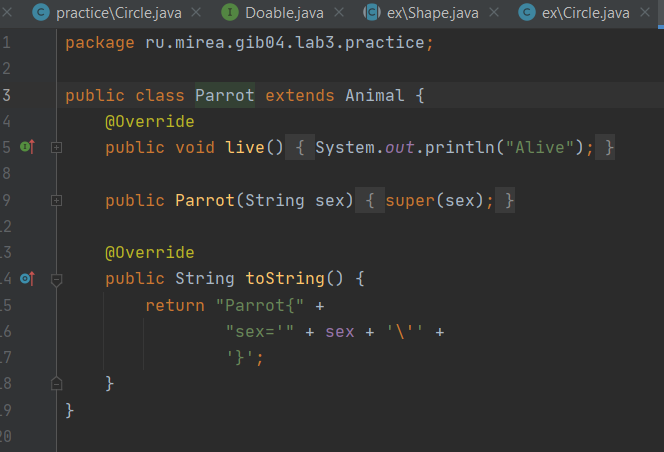
****

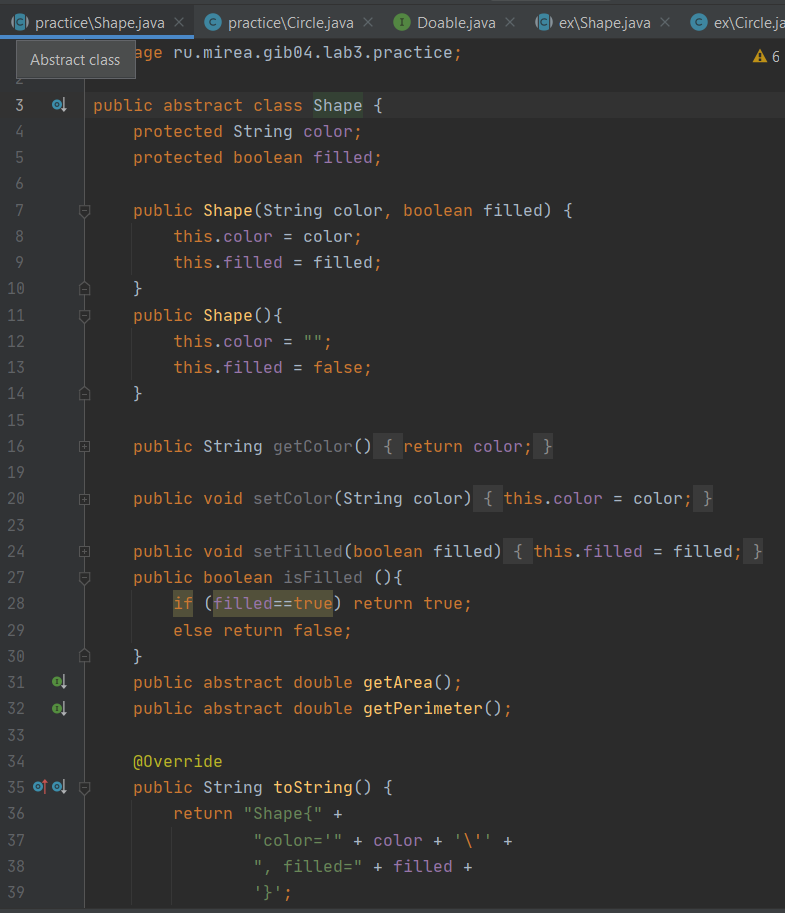
****

****

****

****

****

****

**Код программы**

**Задание 1**

* **Класс Сircle**
* package ru.mirea.gib04.lab3.ex;  
    
  import java.lang.\*;  
    
  public class Circle extends Shape {  
   protected double radius;  
   public Circle(){  
   this.filled = false;  
   this.color = "blue";  
   radius = 1;  
   }  
   public Circle(double radius){  
   this.filled = false;  
   this.color = "blue";  
   this.radius = radius;  
   }  
    
   public Circle(String color, boolean filled, double radius) {  
   super(color, filled);  
   this.radius = radius;  
   }  
    
   public double getRadius() {  
   return radius;  
   }  
    
   public void setRadius(double radius) {  
   this.radius = radius;  
   }  
    
   public double getArea() {  
   return Math.PI\*radius\*radius;  
   }  
    
   public double getPerimeter() {  
   return 2\*Math.PI\*radius;  
   }  
    
   public String toString() {  
   return "Shape: circle, radius: "+this.radius+", color: "+this.color;  
   }  
  }
* **Класс Main**

package ru.mirea.gib04.lab3.ex;  
  
  
public class Main {  
 public static void main(String[] args) {  
 Shape s1 = new Circle("red", false, 5.5); // Upcast Circle to Shape  
 System.out.println(s1);  
 System.out.println(s1.getArea());  
 System.out.println(s1.getPerimeter());  
 System.out.println(s1.getColor());  
 System.out.println(s1.isFilled());  
 // System.out.println(s1.getRadius()); //Выдает ошибку т.к остсутсвует начальное значение в Shape  
  
 Circle c1 = (Circle) 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());  
  
 //Shape s2 = new Shape(); Выдает ошибку абстрактный класс  
  
 Shape s3 = new Rectangle("red", false, 1.0, 2.0);  
 System.out.println(s3);  
 System.out.println(s3.getArea());  
 System.out.println(s3.getPerimeter());  
 System.out.println(s3.getColor());  
 //System.out.println(s3.getLength()); Выдает ошибку т.к остсутсвует начальное значение в Shape  
  
 Rectangle r1 = (Rectangle) s3; // downcast  
 System.out.println(r1);  
 System.out.println(r1.getArea());  
 System.out.println(r1.getColor());  
 System.out.println(r1.getLength());  
  
 Shape s4 = new Square(6.6); // Upcast  
 System.out.println(s4);  
 System.out.println(s4.getArea());  
 System.out.println(s4.getColor());  
 //System.out.println(s4.getSide()); Тоже самое  
  
  
 Rectangle r2 = (Rectangle) s4;  
 System.out.println(r2);  
 System.out.println(r2.getArea());  
 System.out.println(r2.getColor());  
 //System.out.println(r2.getSide()); Выдает ошибку т.к остсутсвует начальное значение в Shape и Rectangle  
 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());  
  
 }  
}

* **Класс Rectangle**

package ru.mirea.gib04.lab3.ex;  
  
public class Rectangle extends Shape {  
 protected double width;  
 protected double length;  
  
 public Rectangle( ) {  
 this.width = 10.0;  
 this.length = 0.0;  
 this.filled = false;  
 this.color = "blue";  
 }  
  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 this.filled = false;  
 this.color = "blue";  
 }  
  
 public Rectangle(String color, boolean filled, double width, double length) {  
 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);  
 }  
  
 @Override  
 public String toString() {  
 return "Rectangle{" +  
 "width=" + width +  
 ", length=" + length +  
 '}';  
 }  
}

* **Абстрактный класс Shape**

package ru.mirea.gib04.lab3. zadanie1;

public abstract class Shape {

protected String color;

protected boolean filled;

public Shape (String color, boolean filled) {

setColor(color);

setFilled(filled);

}

public Shape () {this("red", true);}

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;}

abstract public double getArea();

abstract public double getPerimeter();

abstract public String toString();

}

* **Класс Square**

package ru.mirea.gib04.lab3.ex;  
  
import java.lang.\*;  
  
public class Square extends Rectangle {  
  
  
 public Square() {  
 this.filled = false;  
 this.color = "blue";  
 this.width = 10.0;  
 this.length = 10.0;  
  
 }  
  
 public Square(double side) {  
 this.filled = false;  
 this.color = "blue";  
 this.width = side;  
 this.length = side;  
  
 }  
  
 public Square(String color, boolean filled, double side) {  
 this.color = color;  
 this.filled = filled;  
 this.width = side;  
 this.length = side;  
 }  
 public double getSide(){  
 return getWidth();  
 }  
  
 public void setSide(double side) {  
 this.length = side;  
 this.width = side;  
 }  
  
 public void setWidth(double side) {  
 this.width = side;  
 }  
 public void setLength(double side) {  
 this.length = side;  
 }  
  
 @Override  
 public String toString() {  
 return "Square{" +  
 "width=" + width +  
 ", length=" + length +  
 ", color='" + color + '\'' +  
 ", filled=" + filled +  
 '}';  
 }  
}

* **Интерфейс Movable**

package ru.mirea.gib04.lab3.ex;  
  
public interface Movable {  
 public void moveUp();  
 public void moveDown();  
 public void moveLeft();  
 public void moveRight();  
}

* **Класс MovablePoint**

package ru.mirea.gib04.lab3.ex;  
  
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 "MovablePoint{" +  
 "x=" + x +  
 ", y=" + y +  
 '}';  
 }  
  
 @Override  
 public void moveUp() {  
 y++;  
 }  
 @Override  
 public void moveDown() {  
 y--;  
 }  
 @Override  
 public void moveRight() {  
 x++;  
 }  
 @Override  
 public void moveLeft() {  
 x--;  
 }  
}

* **Класс MovableRectangle**

package ru.mirea.gib04.lab3.ex;  
  
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);  
 }  
  
 @Override  
 public String toString() {  
 return "MovableRectangle{" +  
 "topLeft=" + topLeft +  
 ", bottomRight=" + bottomRight +  
 '}';  
 }  
  
 @Override  
 public void moveUp() {  
 topLeft.moveUp();  
 bottomRight.moveUp();  
 }  
  
 @Override  
 public void moveDown() {  
 topLeft.moveDown();  
 bottomRight.moveDown();  
 }  
 @Override  
 public void moveRight() {  
 topLeft.moveRight();  
 bottomRight.moveRight();  
 }  
 @Override  
 public void moveLeft() {  
 topLeft.moveLeft();  
 bottomRight.moveLeft();  
 }  
}