Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Университет ИТМО»

**ОТЧЕТ**

по лабораторной работе

# «Абстрактные классы и интерфейсы»

Выполнил:

Студент гр.K33204

Мороз Илья Олегович

Проверил:

Тьютор

Дубаков Анатолий Алексеевич

Санкт-Петербург

2020 г.

**Abstract суперкласс Shape и его подклассы**

Создадим суперкласс Shape и его подклассы Circle, Rectangle и Square.

**Shape.java**

public abstract class Shape {  
 protected String color;  
   
 protected boolean filled;  
  
 Shape() {  
 color = "";  
 filled = false;  
 }  
  
 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;  
 }  
  
 public abstract double getArea();  
  
 public abstract double getPerimeter();  
  
 @Override  
 public String toString() {  
 return "Shape with color = " + color;  
 };  
}

**Circle.java**

public class Circle extends Shape {  
 protected double radius;  
  
 public Circle() {  
 radius = 5;  
 }  
  
 public Circle(double radius) {  
 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;  
 }  
  
 @Override  
 public double getArea() {  
 return Math.*PI* \* Math.*pow*(radius, 2);  
 }  
  
 @Override  
 public double getPerimeter() {  
 return 2 \* Math.*PI* \* radius;  
 }  
  
 @Override  
 public String toString() {  
 return "Circle with radius = " + radius + ", which is a subclass of " + super.toString();  
 }  
}

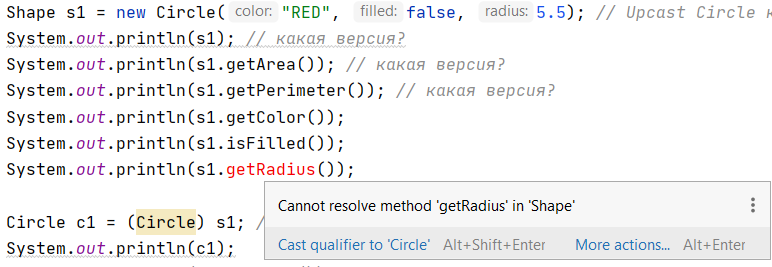
**Rectangle.java**

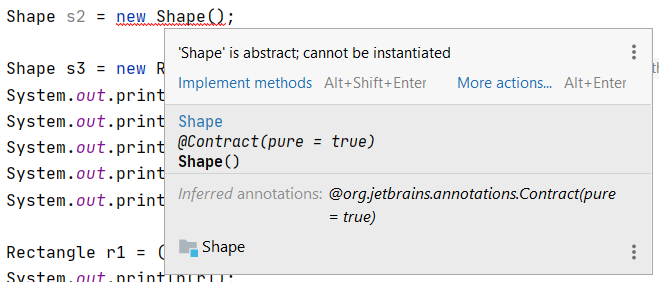
public class Rectangle extends Shape {  
 protected double width;  
  
 protected double length;  
  
 public Rectangle() {  
 width = 1;  
 length = 2;  
 }  
  
 public Rectangle(double width, double length) {  
 this.width = width;  
 this.length = length;  
 }  
  
 public Rectangle(String color, 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;  
 }  
  
 @Override  
 public double getArea() {  
 return width \* length;  
 }  
  
 @Override  
 public double getPerimeter() {  
 return 2 \* (width + length);  
 }  
  
 @Override  
 public String toString() {  
 return "Rectangle with width = " + width + " and length = " + length + ", which is a subclass of " + super.toString();  
 }  
}

**Square.java**

public class Square extends Rectangle{  
 public Square() {  
 super(5, 5);  
 }  
  
 public Square(double side) {  
 super(side, side);  
 }  
  
 public Square(String color, boolean filled, double side) {  
 super(color, filled, side, side);  
 }  
  
 public double getSide() {  
 return getWidth();  
 }  
  
 public void setSide(double side) {  
 super.setLength(side);  
 super.setWidth(side);  
 }  
  
 @Override  
 public void setWidth(double side) {  
 super.setWidth(side);  
 super.setLength(side);  
 }  
  
 @Override  
 public void setLength(double side) {  
 super.setLength(side);  
 super.setWidth(side);  
 }  
  
 @Override  
 public String toString() {  
 return "Square with side = " + getSide() + ", which is a subclass of " + super.toString();  
 }  
}

Теперь протестируем созданные классы с помощью метода main, который предлагается в практике. В коде содержатся ошибки, которые не позволяют скомпилировать код. Например, создание экземпляра абстрактного класса и вызов метода, которые не определён в классе Shape:



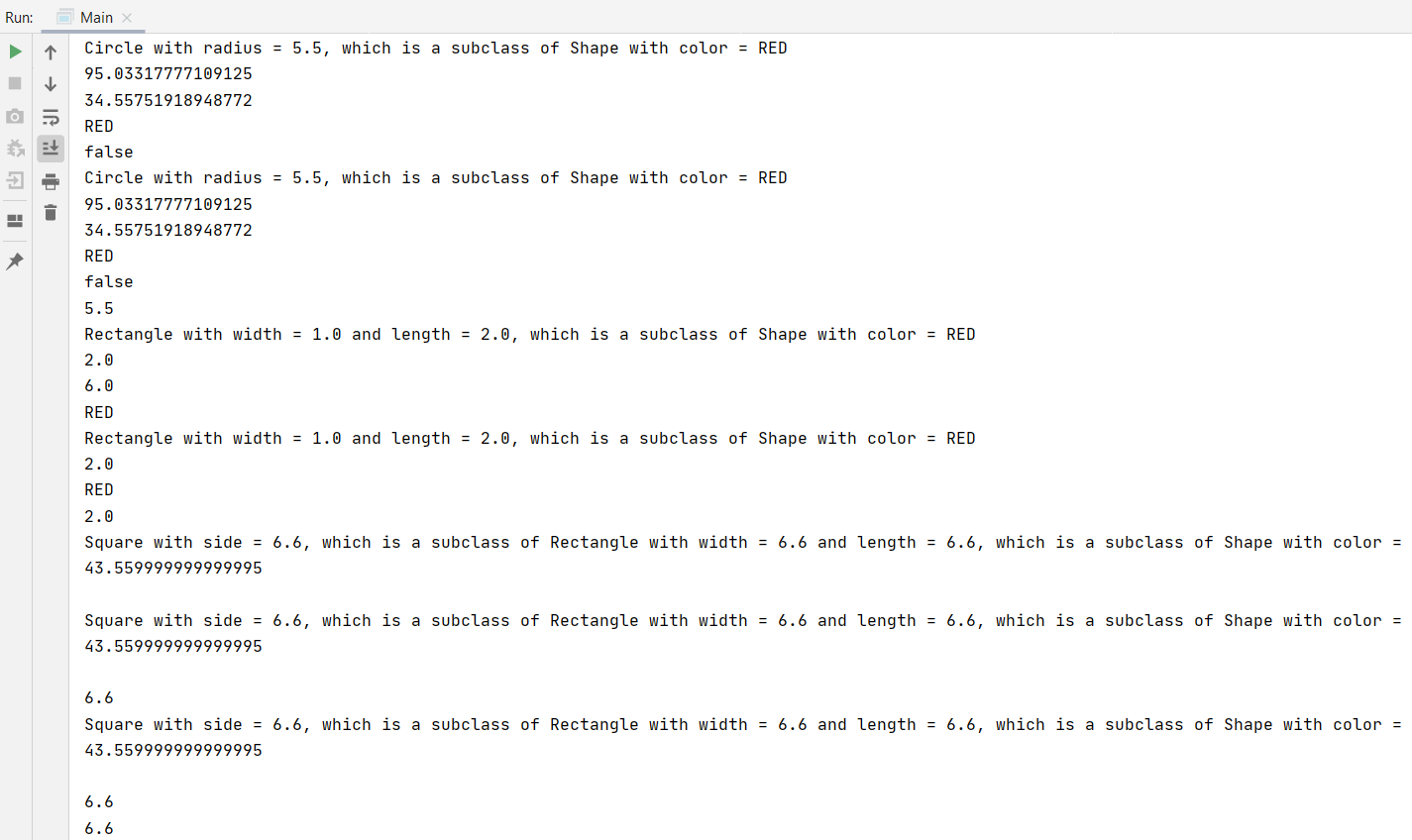


Уберём неработающий код. Получаем следующий файл для тестирования:

**Main.java**

public class Main {  
 public static void main(String[] args) {  
 Shape s1 = new Circle("RED", false, 5.5); *// Upcast Circle к Shape* System.*out*.println(s1); *// какая версия?* System.*out*.println(s1.getArea()); *// какая версия?* System.*out*.println(s1.getPerimeter()); *// какая версия?* System.*out*.println(s1.getColor());  
 System.*out*.println(s1.isFilled());  
  
 Circle c1 = (Circle) s1; *// Downcast обратно к Circle* 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 s3 = new Rectangle("RED", false, 1.0, 2.0); *// Upcast* System.*out*.println(s3);  
 System.*out*.println(s3.getArea());  
 System.*out*.println(s3.getPerimeter());  
 System.*out*.println(s3.getColor());  
  
 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());  
  
 *// Обратите внимание, что выполняется downcast Shape s4 к Rectangle,  
 // который является суперклассом Square, а не объектом Square* Rectangle r2 = (Rectangle) s4;  
 System.*out*.println(r2);  
 System.*out*.println(r2.getArea());  
 System.*out*.println(r2.getColor());  
 System.*out*.println(r2.getLength());  
  
 *// Downcast Rectangle r2 к Square* 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());  
 }  
}

**Console**



**Интерфейсы**

Рассмотрим работу с интерфейсами на примере интерфейса IntSequence.

**IntSequence.java**

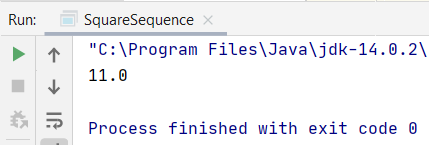
public interface IntSequence {  
 boolean hasNext();  
  
 int next();  
}

Протестируем этот интерфейс.

**SquareSequence.java**

public class SquareSequence implements IntSequence {  
 private int i;  
  
 public boolean hasNext() {  
 return true;  
 }  
  
 public int next() {  
 i++;  
 return i \* i;  
 }  
  
 public static double average(IntSequence seq, int n) {  
 int count = 0;  
 double sum = 0;  
 while (seq.hasNext() && count < n) {  
 count++;  
 sum += seq.next();  
 }  
 return count == 0 ? 0 : sum / count;  
 }  
  
 public static void main(String[] args) {  
 SquareSequence squares = new SquareSequence();  
 double avg = *average*(squares, 5);  
 System.*out*.println(avg);  
 }  
}

**Console**

****

Рассмотрим ещё один пример работы.

**DigitSequence.java**

public class DigitSequence implements IntSequence {  
 private int number;  
  
 public DigitSequence(int n) {  
 number = n;  
 }  
  
 public boolean hasNext() {  
 return number != 0;  
 }  
  
 public int next() {  
 int result = number % 10;  
 number /= 10;  
 return result;  
 }  
  
 public static double average(IntSequence seq, int n) {  
 int count = 0;  
 double sum = 0;  
 while (seq.hasNext() && count < n) {  
 count++;  
 sum += seq.next();  
 }  
 return count == 0 ? 0 : sum / count;  
 }  
  
 public static void main(String[] args) {  
 DigitSequence digits = new DigitSequence(17292344);  
 double avg = *average*(digits, 6);  
 System.*out*.println(avg);  
 }  
}

**Console**

