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

Отчет по выполнению практической работы 3

**Тема. НАСЛЕДОВАНИЕ. АБСТРАКТНЫЕ СУПЕРКЛАССЫ И ИХ ПОДКЛАССЫ В JAVA.**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Задание 2](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc35859_811265792)

[​ Circle.java 3](#__RefHeading___Toc42463_811265792)

[​ Square.java 5](#__RefHeading___Toc42465_811265792)

[​ Shape.java 6](#__RefHeading___Toc42467_811265792)

[​ Rectangle.java 7](#__RefHeading___Toc42469_811265792)

[​ Main.java 8](#__RefHeading___Toc42471_811265792)

[​ Скриншот 9](#__RefHeading___Toc3748_811265792)

[​ Заключение 10](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 10](#__RefHeading___Toc3752_811265792)

# Теория

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

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип. 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.

# Задание

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.

Shape s1 = new Circle(5.5, "RED", false); // 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());

Circle c1 = (Circle)s1; // Downcast back to 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 s2 = new

Shape();

Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Upcast

System.out.println(s3);

System.out.println(s3.getArea());

System.out.println(s3.getPerimeter());

System.out.println(s3.getColor());

System.out.println(s3.getLength());

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());

// Take note that we downcast Shape s4 to Rectangle,

// which is a superclass of Square, instead of Square

Rectangle r2 = (Rectangle)s4;

System.out.println(r2);

System.out.println(r2.getArea());

System.out.println(r2.getColor());

System.out.println(r2.getSide());

System.out.println(r2.getLength()); // Downcast Rectangle r2 to 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());

# Код

### Circle.java

package dev.ky3he4ik.pr.pr3;  
  
public class Circle extends Shape {  
 protected double radius;  
  
 public Circle() {  
 super();  
 radius = 0;  
 }  
  
 public Circle(double radius) {  
 super();  
 this.radius = radius;  
 }  
  
 public Circle(double radius, String color, boolean filled) {  
 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 \* radius \* radius;  
 }  
  
 @Override  
 public double getPerimeter() {  
 return Math.PI \* radius \* 2;  
 }  
  
 @Override  
 public String toString() {  
 return "Circle with r = " + radius + ' ' + super.toString();  
 }  
}

### Square.java

package dev.ky3he4ik.pr.pr3;  
  
public class Square extends Rectangle {  
 public Square() {  
 super();  
 }  
  
 public Square(double side) {  
 super(side, side);  
 }  
  
 public Square(double side, String color, boolean filled) {  
 super(side, side, color, filled);  
 }  
  
 public double getSide() {  
 return length;  
 }  
  
 public void setSide(double side) {  
 this.length = side;  
 this.width = side;  
 }  
  
 @Override  
 public void setWidth(double side) {  
 this.length = side;  
 this.width = side;  
 }  
  
 @Override  
 public void setLength(double side) {  
 this.length = side;  
 this.width = side;  
 }  
  
 @Override  
 public String toString() {  
 return "Square " + length + 'x' + length + "#" + color + (filled ? " not" : " ") + " filled";  
 }  
}

### Shape.java

package dev.ky3he4ik.pr.pr3;  
  
// 3.2  
public abstract class Shape {  
 protected String color;  
 protected boolean filled;  
  
 public Shape() {  
 color = "000000";  
 filled = false;  
 }  
  
 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;  
 }  
  
 public abstract double getArea();  
  
 public abstract double getPerimeter();  
  
 @Override  
 public String toString() {  
 return "#" + color + (filled ? " not" : " ") + " filled";  
 }  
}

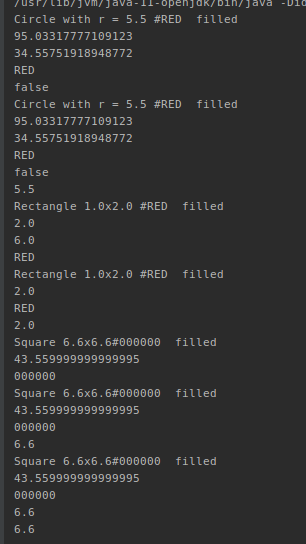
### Rectangle.java

package dev.ky3he4ik.pr.pr3;  
  
public class Rectangle extends Shape {  
 protected double width;  
 protected double length;  
  
 public Rectangle() {  
 super();  
 length = 0;  
 width = 0;  
 }  
  
 public Rectangle(double width, double length) {  
 super();  
 this.width = width;  
 this.length = length;  
 }  
  
 public Rectangle(double width, double length, String color, boolean filled) {  
 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 (width + length) \* 2;  
 }  
  
 @Override  
 public String toString() {  
 return "Rectangle " + width + 'x' + length + ' ' + super.toString();  
 }  
}

### Main.java

package dev.ky3he4ik.pr.pr3;  
  
public class Main {  
 public static void main(String[] args) {  
 Shape s1 = new Circle(5.5, "RED", false); // 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());  
 Circle c1 = (Circle) s1; // Downcast back to 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 s2 = new Shape();  
 Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Upcast  
 System.out.println(s3);  
 System.out.println(s3.getArea());  
 System.out.println(s3.getPerimeter());  
 System.out.println(s3.getColor());  
// System.out.println(s3.getLength());  
 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());  
 // Take note that we downcast Shape s4 to Rectangle,  
 // which is a superclass of Square, instead of Square  
 Rectangle r2 = (Rectangle) s4;  
 System.out.println(r2);  
 System.out.println(r2.getArea());  
 System.out.println(r2.getColor());  
// System.out.println(r2.getSide());  
 System.out.println(r2.getLength()); // Downcast Rectangle r2 to 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());  
 }  
}

# Скриншот



# Заключение

В данной практической работе я освоил на практике работу с абстрактными классами и наследованием на Java.

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.