**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 6**

**Дисциплина:** Технологии кроссплатформенного программирования

**Тема:** Абстрактные классы и интерфейсы

**Выполнила: студентка группы 211-725**

Алюбаева Карина Ислямбековна

**Дата**  28.10.2023

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Москва**

**2023**

Оглавление

[Введение 2](#_Toc149397332)

[Цель работы: 2](#_Toc149397333)

[Задание: 2](#_Toc149397334)

[Ход работы 3](#_Toc149397335)

[Применение абстрактных классов 3](#_Toc149397336)

[Применение интерфейсов 6](#_Toc149397337)

# Введение

## Цель работы:

Научиться применять абстрактные классы и интерфейсы в программе на примере кода с фигурами.

## Задание:

1. Применение абстрактных классов.
   1. Создать не менее трёх классов, которые будут описывать различные геометрические фигуры предварительно согласовав их с преподавателем (например, треугольник, квадрат и круг).
   2. Предусмотреть наличие у фигур таких характеристик, как цвет, линейные размеры, площадь, возможно объём.
   3. В программе должен быть абстрактный класс с названием Figure, на основе которого создаются производные классы фигур.
   4. В главном методе реализовать создание объектов производных классов и продемонстрировать, как получить доступ к объектам через объектные переменные подклассов и через объектную переменную абстрактного суперкласса.
2. Применение интерфейсов.
   1. Для фигур из предыдущего задания создать интерфейсы.
   2. Создать суперкласс, обеспечивающий ввод с клавиатуры вида арифметической операции (например, умножение) и вывод заголовка результата, который будет отображаться на экране.
   3. Создать подкласс, который наследует суперкласс и реализует интерфейсы. В подклассе предусмотреть вычисление результата, получаемого применением вводимой арифметической операции к величинам площадей фигур, и отображение результата на экране после отображения заголовка результата.

# Ход работы

## Применение абстрактных классов

Листинг с программой:

// Абстрактный класс

public abstract class Figure {

    private String color;

    public Figure(String color) {

        this.color = color;

    }

    public abstract double getArea();

    public String getColor() {

        return color;

    }

}

// Класс треугольник

public class Triangle extends Figure {

    private double base;

    private double height;

    public Triangle(String color, double base, double height) {

        super(color);

        this.base = base;

        this.height = height;

    }

    //этот метод переопределяет метод getArea в суперклассе

    @Override

    public double getArea() {

        return 0.5 \* base \* height;

    }

}

// Класс квадрат

// Класс Квадрат

public class Square extends Figure {

    private double side;

    public Square(String color, double side) {

        super(color);

        this.side = side;

    }

    //этот метод переопределяет метод getArea в суперклассе

    @Override

    public double getArea() {

        return side \* side;

    }

}

// Класс круг

// Класс Круг

public class Circle extends Figure {

    private double radius;

    public Circle(String color, double radius) {

        super(color);

        this.radius = radius;

    }

    //этот метод переопределяет метод getArea в суперклассе

    @Override

    public double getArea() {

        return Math.PI \* radius \* radius;

    }

}

// Главный метод

// Главный метод

public class Main {

    public static void main(String[] args) {

        // Создание объектов различных фигур

        Triangle triangle = new Triangle("Красный", 5, 4);

        Square square = new Square("Синий", 3);

        Circle circle = new Circle("Зеленый", 2);

        // Доступ к объектам через объектную переменную суперкласса

        Figure figure1 = triangle;

        Figure figure2 = square;

        Figure figure3 = circle;

        // Вывод информации о треугольнике

        System.out.println("Площадь треугольника: " + triangle.getArea());

        System.out.println("Цвет треугольника: " + figure1.getColor());

        // Вывод информации о квадрате

        System.out.println("Площадь квадрата: " + square.getArea());

        System.out.println("Цвет квадрата: " + figure2.getColor());

        // О круге

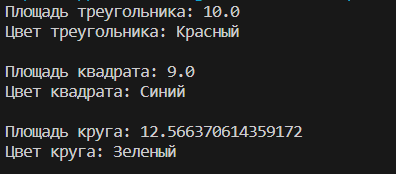
        System.out.println("Площадь круга: " + circle.getArea());

        System.out.println("Цвет круга: " + figure3.getColor());

    }

}

Результат работы программы



## Применение интерфейсов

// Интерфейс для получения цвета

interface Colorful {

    String getColor();

}

// Интерфейс для вычисления площади

interface AreaCalculable {

    double getArea();

}

// Абстрактный класс Figure, который реализует интерфейсы

public abstract class Figure implements Colorful, AreaCalculable {

    private String color;

    public Figure(String color) {

        this.color = color;

    }

    @Override

    public String getColor() {

        return color;

    }

}

// Класс Triangle, который реализует интерфейсы

public class Triangle extends Figure {

    private double base;

    private double height;

    public Triangle(String color, double base, double height) {

        super(color);

        this.base = base;

        this.height = height;

    }

    @Override

    public double getArea() {

        return 0.5 \* base \* height;

    }

}

// Класс Square, который реализует интерфейсы

public class Square extends Figure {

    private double side;

    public Square(String color, double side) {

        super(color);

        this.side = side;

    }

    @Override

    public double getArea() {

        return side \* side;

    }

}

// Класс Circle, который реализует интерфейсы

public class Circle extends Figure {

    private double radius;

    public Circle(String color, double radius) {

        super(color);

        this.radius = radius;

    }

    @Override

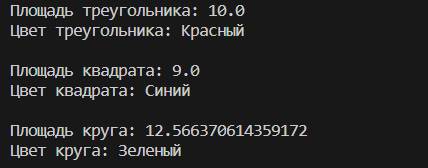
    public double getArea() {

        return Math.PI \* radius \* radius;

    }

}

Результат работы программы:



Суперкласс, обеспечивающий ввод с клавиатуры:

import java.util.Scanner;

// Суперкласс для выполнения арифметических операций

class Calculator {

    private Scanner scanner;

    public Calculator() {

        scanner = new Scanner(System.in);

    }

    public void performCalculation(Figure figure1, Figure figure2, String operation) {

        double result = 0.0;

        switch (operation) {

            case "\*":

                result = figure1.getArea() \* figure2.getArea();

                break;

            case "+":

                result = figure1.getArea() + figure2.getArea();

                break;

            case "-":

                result = figure1.getArea() - figure2.getArea();

                break;

            case "/":

                result = figure1.getArea() / figure2.getArea();

                break;

            default:

                System.out.println("Неизвестная операция");

                return;

        }

        System.out.println("Результат операции: " + result + "\n");

    }

}

// Подкласс, который наследует суперкласс Calculator и реализует интерфейсы

class AdvancedCalculator extends Calculator implements CalculatorResult, DisplayResult {

    public AdvancedCalculator() {

        super(); // Вызываем конструктор суперкласса

    }

    @Override

    public double calculateResult(double area1, double area2) {

        // В данном случае, реализуем сложение площадей фигур

        return area1 + area2;

    }

    @Override

    public void displayResult(double result) {

        System.out.println("Заголовок результата:");

        System.out.println("Результат операции: " + result);

    }

}

// Интерфейс для вычисления результата

interface CalculatorResult {

    double calculateResult(double area1, double area2);

}

// Интерфейс для отображения результата

interface DisplayResult {

    void displayResult(double result);

}