

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ  
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Ордена трудового Красного Знамени федеральное государственное  
бюджетное**  
**образовательное учреждение высшего образования**  
**«Московский технический университет связи и информатики»**

Кафедра Математическая кибернетика и информационные технологии

**Лабораторная работа №2**  
**Объектно-ориентированное программирование**

Выполнил: студент группы БВТ2402

Голиков Михаил Вячеславович

Руководитель: Мосева Марина Сергеевна

Москва, 2077

## Цель работы

Цель данной лабораторной работы — освоить основные концепции объектно-ориентированного программирования на языке Java: абстракция, инкапсуляция, наследование и полиморфизм. В рамках работы реализуются классы, демонстрирующие данные принципы.

## Индивидуальное задание

Реализовать абстрактный класс Figure с полями цвета и координат, а также абстрактным методом volume(). Создать три класса-наследника: Sphere, Parallelepiped и Cylinder. В классах реализовать методы доступа (геттеры и сеттеры), а также переопределить метод getInfo() для вывода информации об объектах.

## Основная часть

### Задание 1. Вариант 6

В ходе работы были реализованы три класса:

1. Sphere — класс сферы, переопределяющий метод volume().
2. Parallelepiped — класс параллелепипеда с параметрами длины, ширины и высоты.
3. Cylinder — класс цилиндра с параметрами радиуса и высоты.

```
package Lab_2;

import java.lang.Math;

// абстракция
abstract class Figure{

    private String color;
    private double corX;
    private double corY;
    abstract double volume();

    private static int countObjectsCreated = 0;

    //-----
    // конструктор
    public Figure(String color, double corX, double corY){
        this.color=color;
        this.corX=corX;
```

```

        this.corY=corY;
        countObjectsCreated++;
    }

    // статический полиморфизм (перегрузка)
    public Figure() {
        this("None", 0.0, 0.0);
    }
    //-----

    //-----
    // инкапсуляция
    public String getColor(){
        return this.color;
    }
    public void setColor(String color){
        if (color.equals("Black")){
            System.out.println("No black figures allowed! Do not use color -
Black");
        }
        else {
            this.color = color;
        }
    }

    public double getCorX(){
        return this.corX;
    }
    public void setCorX(double corX){
        this.corX=corX;
    }

    public double getCorY(){
        return this.corY;
    }
    public void setCorY(double corY){
        this.corY=corY;
    }
    //-----

    public void getInfo(){
        System.out.println("color: " + this.color);
        System.out.println("corX: " + this.corX);
        System.out.println("corY: " + this.corY);
    }

    public void printCountObjectsCreated(){
        System.out.println("countObjectsCreated: " + countObjectsCreated);
    }
}

// наследование
class Sphere extends Figure{
    private double radius;

    public Sphere(String color, double corX, double corY, double radius){

```

```

        super(color, corX, corY);
        this.radius = radius;
    }
    public Sphere(){
        super();
        this.radius = 0.0;
    }

    public double getRadius(){
        return this.radius;
    }
    public void setRadius(double radius){
        if (radius < 0){
            this.radius= -radius;
        }
        else {
            this.radius = radius;
        }
    }

    // реализация абстрактного метода
    @Override
    public double volume(){
        return (double)(4/3) * Math.PI * this.radius * this.radius;
    }

    // динамический полиморфизм (переопределение метода)
    @Override
    public void getInfo() {
        super.getInfo();
        System.out.println("radius: " + this.radius);
    }
}

class Parallelepiped extends Figure{
    private double length;
    private double width;
    private double height;

    public Parallelepiped(String color, double corX, double corY, double
length, double width, double height){
        super(color, corX, corY);
        this.length = length;
        this.width = width;
        this.height = height;
    }
    public Parallelepiped(){
        super();
        this.length = 0.0;
        this.width = 0.0;
        this.height = 0.0;
    }

    public double getLength(){
        return this.length;
    }
    public void setLength(double length){
        // вспомнил про тернарный оператор

```

```

        this.length = length < 0 ? -length : length;
    }

    public double getWidth(){
        return this.width;
    }
    public void setWidth(double width){
        this.width = width < 0 ? -width : width;
    }

    public double getHeight(){
        return this.height;
    }
    public void setHeight(double height){
        this.height = height < 0 ? -height : height;
    }

    @Override
    public double volume(){
        return this.length * this.width * this.height;
    }

    @Override
    public void getInfo() {
        super.getInfo();
        System.out.println("length: " + this.length);
        System.out.println("width: " + this.width);
        System.out.println("height: " + this.height);
    }
}

class Cylinder extends Figure{
    private double radius;
    private double height;

    public Cylinder(String color, double corX, double corY, double radius,
double height){
        super(color, corX, corY);
        this.radius = radius;
        this.height = height;
    }
    public Cylinder(){
        super();
        this.radius = 0.0;
        this.height = 0.0;
    }

    public double getRadius(){
        return this.radius;
    }
    public void setRadius(double radius){
        this.radius = radius < 0 ? -radius : radius;
    }

    public double getHeight(){
        return this.height;
    }
    public void setHeight(double height){
        this.height = height < 0 ? -height : height;
    }

    @Override

```

```

    public double volume(){
        return Math.PI * this.radius * this.radius * this.height;
    }

    @Override
    public void getInfo() {
        super.getInfo();
        System.out.println("radius: " + this.radius);
        System.out.println("height: " + this.height);
    }
}

public class Var6 {
    public static void main(String[] args){
        Sphere sphere = new Sphere();
        System.out.println("Before changes -----");
        sphere.getInfo();

        sphere.setRadius(-15.0);
        sphere.setColor("Black");
        sphere.setCorX(10.0);
        sphere.setCorY(20.0);

        String sphereColor = sphere.getColor();
        if (sphereColor.equals("None")){
            sphere.setColor("Pink");
        }

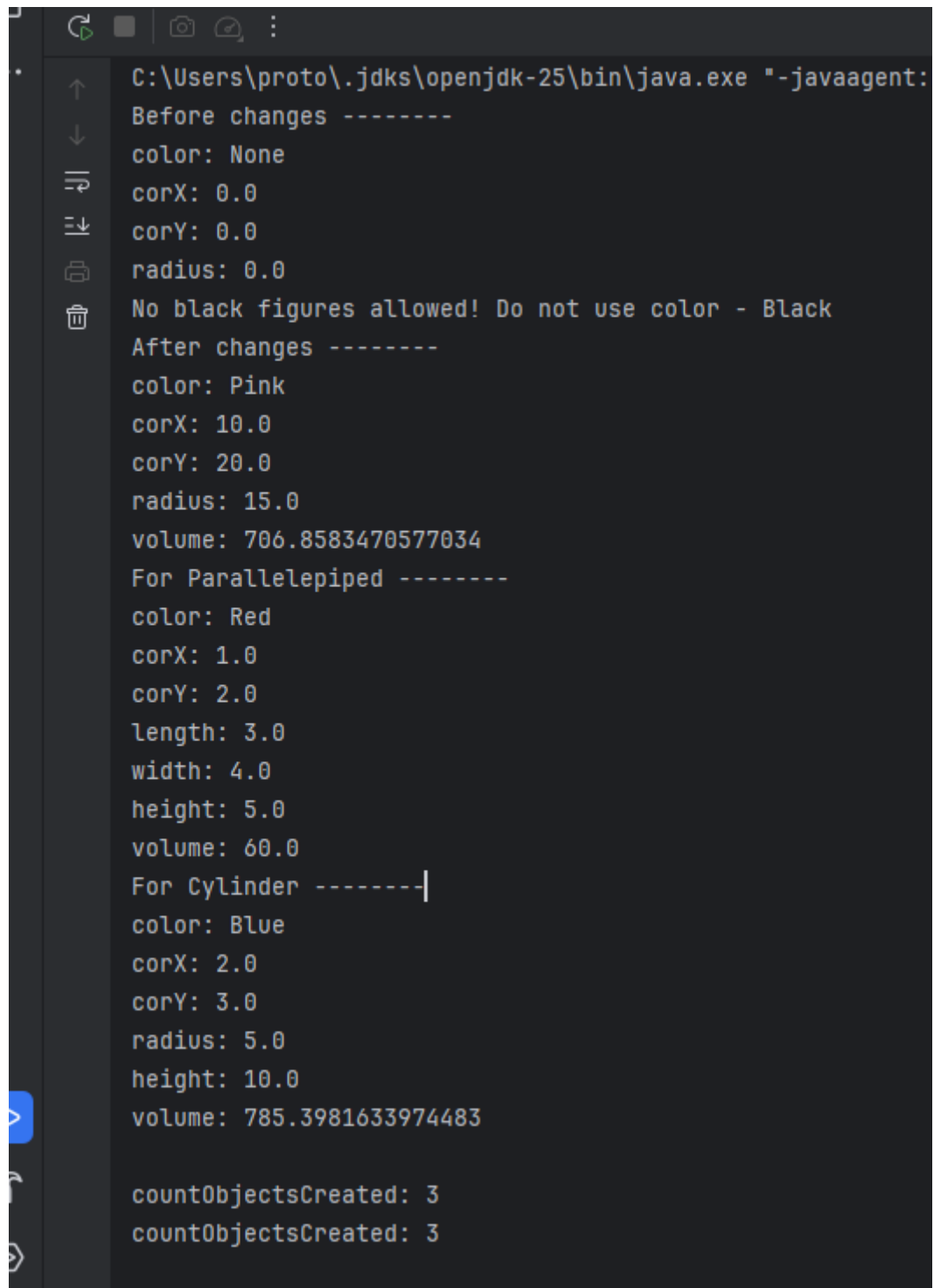
        System.out.println("After changes -----");
        sphere.getInfo();
        System.out.println("volume: " + sphere.volume());

        Parallelepiped box = new Parallelepiped("Red", 1.0, 2.0, 3.0, 4.0,
5.0);
        System.out.println("For Parallelepiped -----");
        box.getInfo();
        System.out.println("volume: " + box.volume());

        Cylinder cyl = new Cylinder("Blue", 2.0, 3.0, 5.0, 10.0);
        System.out.println("For Cylinder -----");
        cyl.getInfo();
        System.out.println("volume: " + cyl.volume() + '\n');

        sphere.printCountObjectsCreated();
        box.printCountObjectsCreated();
    }
}

```



```
C:\Users\proto\.jdk\openjdk-25\bin\java.exe "-javaagent:
Before changes -----
color: None
corX: 0.0
corY: 0.0
radius: 0.0
No black figures allowed! Do not use color - Black
After changes -----
color: Pink
corX: 10.0
corY: 20.0
radius: 15.0
volume: 706.8583470577034
For Parallelepiped -----
color: Red
corX: 1.0
corY: 2.0
length: 3.0
width: 4.0
height: 5.0
volume: 60.0
For Cylinder -----|
color: Blue
corX: 2.0
corY: 3.0
radius: 5.0
height: 10.0
volume: 785.3981633974483

countObjectsCreated: 3
countObjectsCreated: 3
```

Элемент 2 — Пример вывода программы

## Заключение

В ходе выполнения лабораторной работы были реализованы классы для трёх фигур с использованием основных принципов ООП. Были изучены перегрузка и переопределение методов, работа с абстрактными классами и

инкапсуляцией. Получен практический опыт разработки объектно-ориентированных программ на Java.

**Github:** <https://github.com/Prototype721/Java>