**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**
**Ордена Трудового Красного Знамени**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

# Отчет по лабораторной работе №2

по дисциплине «Информационные технологии и программирование»

Выполнил: студент группы БПИ2401

Юлдашев Всеволод

Проверил: Харрасов Камиль Раисович

Москва

2025

1. **Цель работы:** Целью лабораторной работы является закрепление знаний и навыков работы с основными принципами объектно-ориентированного программирования (ООП) в языке Java.

   В процессе выполнения необходимо:

- разработать иерархию классов в соответствии с заданным вариантом;
- продемонстрировать использование принципов наследования, инкапсуляции, абстракции и полиморфизма;
- реализовать абстрактный базовый класс и дочерние классы двух уровней наследования;
- использовать конструкторы (в том числе конструктор по умолчанию);
- применить геттеры и сеттеры для работы с полями класса;
- реализовать статическую переменную для подсчёта количества созданных объектов;
- обеспечить ввод и вывод информации об объектах программы.

2. **Ход работы:**

```
abstract class Establishment {
    protected String name;
    protected String address;
    protected double area;


    public Establishment() {}


    public Establishment(String name, String address, double area) {
        this.name = name;
        this.address = address;
        this.area = area;
    }
```

```
    public abstract void open();

    public void close() {
        System.out.println(name + " закрывается.");
    }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }

    public double getArea() { return area; }
    public void setArea(double area) { this.area = area; }

    public String toString() {
        return name + " (" + address + "), площадь: " + area + " м²";
    }
}

class Cafe extends Establishment {
    private int tables;
```

```
    private static int count = 0;


    public Cafe(String name, String address, double area, int tables) {
        super(name, address, area);
        this.tables = tables;
        count++;
    }


    public Cafe() { count++; }


    public void open() {
        System.out.println(name + " открыто. Добро пожаловать в кафе!");
    }


    public void serve() {
        System.out.println(name + " обслуживает посетителей.");
    }


    public static int getCount() { return count; }


    public String toString() {
        return super.toString() + ", столов: " + tables;
```

```
    }
}


class Store extends Establishment {
    private String goodsType;

    public Store(String name, String address, double area, String goodsType) {
        super(name, address, area);
        this.goodsType = goodsType;
    }

    public Store() {}

    public void open() {
        System.out.println(name + " открыт. Добро
пожаловать в магазин!");
    }

    public void restock() {
        System.out.println(name + " пополняет запасы.");
    }

    public String toString() {
        return super.toString() + ", тип товара: " + goodsType;
```

```
    }
}


class Library extends Establishment {
    private int books;

    public Library(String name, String address, double area, int books) {
        super(name, address, area);
        this.books = books;
    }

    public Library() {}

    public void open() {
        System.out.println(name + " открыта. Приглашаем
читать!");
    }

    public void lendBook() {
        System.out.println(name + " выдает книгу читателю.");
    }

    public String toString() {
        return super.toString() + ", книг: " + books;
```

```
    }
}


public class Main {
    public static void main(String[] args) {
        Cafe cafe = new Cafe("Кафе "ВОСТОЧНОЕ"", "ул. Авиамоторная, 8А", 100.0, 10);
        Store store = new Store("Магазин "МТУСИСТ"", "пр. Мира, 7", 80.0, "продовольственные");
        Library library = new Library("Библиотека МТУСИ", "ул. Народного Ополчения, 32", 120.0, 5000);


        System.out.println("Информация о заведениях:");
        System.out.println(cafe);
        System.out.println(store);
        System.out.println(library);


        System.out.println("\nРабота заведений:");
        cafe.open();
        store.open();
        library.open();


        cafe.serve();
        store.restock();
        library.lendBook();
```

```
        System.out.println("\nКоличество созданных кафе: "
+ Cafe.getCount());
    }
}
```

```
> java Main.java
Информация о заведениях:
Кафе 'ВОСТОЧНОЕ' (ул. Авиамоторная, 8А), площадь: 100.0 м², столов: 10
Магазин 'МТУСИСТ' (пр. Мира, 7), площадь: 80.0 м², тип товара: продовольственные
Библиотека МТУСИ (ул. Народного Ополчения, 32), площадь: 120.0 м², книг: 5000

Работа заведений:
Кафе 'ВОСТОЧНОЕ' открыто. Добро пожаловать в кафе!
Магазин 'МТУСИСТ' открыт. Добро пожаловать в магазин!
Библиотека МТУСИ открыта. Приглашаем читать!
Кафе 'ВОСТОЧНОЕ' обслуживает посетителей.
Магазин 'МТУСИСТ' пополняет запасы.
Библиотека МТУСИ выдает книгу читателю.

Количество созданных кафе: 1
```

**Контрольные вопросы:**

1. Абстракция — выделение главных характеристик объекта без деталей. В Java реализуется через абстрактные классы (abstract) и интерфейсы (interface).

2. Инкапсуляция — сокрытие данных и ограничение прямого доступа к ним. Реализуется с помощью модификатора private и геттеров/сеттеров.

3. Наследование — создание нового класса на основе существующего. В Java используется ключевое слово extends.

4. Полиморфизм — возможность вызывать одинаковые методы с разным поведением. Реализуется через переопределение (@Override) и наследование.

5. Множественное наследование — наследование от нескольких классов. В Java нет множественного наследования классов (чтобы избежать конфликтов), но есть множественное наследование интерфейсов.

6. Ключевое слово final — запрещает изменение: класса (нельзя наследовать), метода (нельзя переопределить), переменной (нельзя изменить значение).

7. Модификаторы доступа: public — доступен везде; protected — доступен в пакете и наследниках; (без модификатора) — только в пакете; private — только внутри класса.

8. Конструктор — метод, вызываемый при создании объекта. Типы: по умолчанию (без параметров), параметризированный (с аргументами), копирующий (редко используется вручную).

9. this — ссылка на текущий объект класса. Используется, чтобы отличать поля от аргументов (this.name = name;).

10. super — обращение к родительскому классу. Используется для вызова его конструктора или методов.

11. Геттеры и сеттеры — методы для безопасного чтения и изменения приватных полей. Обеспечивают контроль и защиту данных.

12. Переопределение (override) — изменение поведения унаследованного метода в потомке. Используется аннотация @Override.

13. Перегрузка (overload) — создание нескольких методов с одинаковым именем, но разными параметрами (типами или количеством аргументов).

**Вывод:** В ходе лабораторной работы была реализована иерархия классов на языке Java с применением основных принципов ООП: наследования, инкапсуляции, абстракции и полиморфизма. Создан абстрактный базовый класс и дочерние классы с конструкторами, геттерами, сеттерами и статическим счётчиком объектов. Цель работы достигнута.

https://github.com/zevy3/java-labs/tree/main/java2