Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

ОТЧЕТ

по лабораторной работе № 4

Вариант 13

Название: Внутренние классы, интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		В.А.Ловцов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель: изучить работу внутренних классов и интерфейсов в java.

Задание 1: создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Код:

```
import java.util.Scanner;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the
gutter.
class Mobile{
    String manufactor;

    public Mobile(String manufactor) {
        this.manufactor = manufactor;
    }


    class Info {
        String model;
        int year;
        int cpu;
        double weigth;
        int memory;
        double display;

        public Info(String model, int year, int cpu, double
weight, int memory, double display) {
            this.model = model;
            this.year = year;
            this.cpu = cpu;
            this.weigth = weight;
            this.memory = memory;
            this.display = display;
        }

        @Override
        public String toString() {
            return manufactor + " " + model +
                    ", year=" + year +
                    ", cpu=" + cpu +
                    ", weight=" + weigth +
                    ", memory=" + memory +
                    ", display=" + display;
        }
    }
```

```
}

public class Third_1 {
    public static void main(String[] args) {
        //TIP Press <shortcut actionId="ShowIntentionActions"/>
with your caret at the highlighted text
        // to see how IntelliJ IDEA suggests fixing it.
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите информацию о телефоне");


        System.out.print("Введите название производителя: ");
        String manufactor = scanner.nextLine();

        Mobile iphone = new Mobile(manufactor);

        System.out.print("Введите название модели: ");
        String model = scanner.nextLine();

        System.out.print("Введите год выпуска: ");
        int year = scanner.nextInt();

        System.out.print("Введите число ядер: ");
        int cpu = scanner.nextInt();

        System.out.print("Введите вес: ");
        double weight = scanner.nextDouble();

        System.out.print("Введите память (ГБ): ");
        int memory = scanner.nextInt();

        System.out.print("Введите диагональ дисплея (дюймы):
");
        double display = scanner.nextDouble();

        Mobile.Info info = iphone.new Info(model, year, cpu,
weight, memory, display);

        System.out.println(info);



    }
}
```

Работа программы показана на рисунке 1.

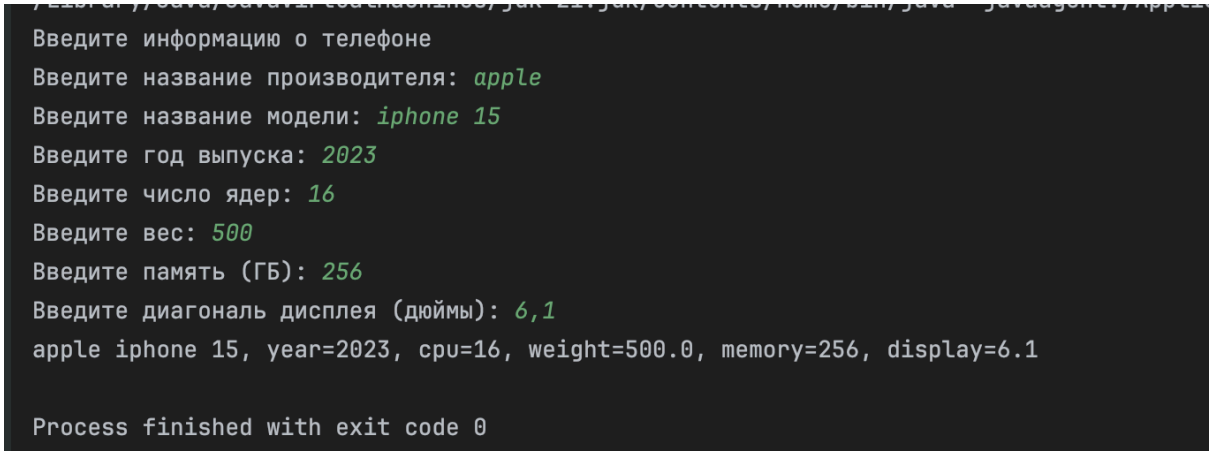

Рисунок 1 – Работа программы

Задание 2: создать класс Художественная Выставка с внутренним классом, с помощью объектов которого можно хранить информацию о картинах, авторах и времени проведения выставок.

Код:

```
import java.util.Scanner;

class Exhibition {
    boolean isOpen = false;

    public void close(){
        this.isOpen = false;
    }

    public void open(){
        this.isOpen = true;
    }

    class Art {
        String name;
        String author;
        int year;

        public Art(String name, String author, int year) {
            this.name = name;
            this.author = author;
            this.year = year;
        }

        @Override
        public String toString() {
```

```
                return name + ", written by " + author +" in "
+ year + ", available = " + isOpen;
            }
        }
    }

    public class Fourth_1 {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);

            Exhibition expo = new Exhibition();

            System.out.println("Введите информацию картине
1:");
            System.out.print("\tname: ");
            String name = scanner.nextLine();
            System.out.print("\tauthor: ");
            String author = scanner.nextLine();
            System.out.print("\tyear: ");
            int year = scanner.nextInt();
            scanner.nextLine();

            Exhibition.Art art1 = expo.new Art(name, author,
year);


            System.out.println("Введите информацию картине
2:");
            System.out.print("\tname: ");
            name = scanner.nextLine();
            System.out.print("\tauthor: ");
            author = scanner.nextLine();
            System.out.print("\tyear: ");
            year = scanner.nextInt();
            scanner.nextLine();

            Exhibition.Art art2 = expo.new Art(name, author,
year);

            System.out.println("До открытия:");
            System.out.println(art1);
            System.out.println(art2);

            expo.open();

            System.out.println("\nОткрытие!");
            System.out.println(art1);
            System.out.println(art2);

        }
    }
```

Работа программы показана на рисунке 2.

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Applic
Введите информацию картине 1:
    name: bears
    author: shishkin
    year: 1700
Введите информацию картине 2:
    name: wave
    author: vasnetsov
    year: 1810
До открытия:
bears, written by shishkin in 1700, available = false
wave, written by vasnetsov in 1810, available = false

Открытие!
bears, written by shishkin in 1700, available = true
wave, written by vasnetsov in 1810, available = true

Process finished with exit code 0
```

Рисунок 2 – Работа программы

Задание 3: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. Interface Сотрудник <- class Инженер <- class Руководитель.

Код:

```
import java.util.Arrays;
import java.util.List;

interface Employer {

    void work();
}

class Engineer implements Employer{

    @Override
    public void work() {
        System.out.println("Engineer is working");
    }

    public void checkScheme(){
```

```
        System.out.println("Engineer is checking scheme");
    }
}

class Manager extends Engineer{
    @Override
    public void work() {
        System.out.println("Manager is working");
    }

    @Override
    public void checkScheme() {
        System.out.println("Manager is checking scheme");
    }

    public void createTask(){
        System.out.println("Manager is creating a task");
    }
}



public class Third_2 {
    public static void main(String[] args) {
        Employer engineer = new Engineer();
        Employer manager = new Manager();

        List<Employer>  employers  =  Arrays.asList(engineer,
manager);
        for (Employer emp : employers) {
            emp.work();
        }


        Engineer engineer_1 = new Engineer();
        Engineer manager_1 = new Manager();

        List<Engineer> employers_1 = Arrays.asList(engineer_1,
manager_1);
        for (Engineer emp : employers_1) {
            emp.checkScheme();
        }

        Manager manager_2 = new Manager();
        manager_2.createTask();

    }
}
```

Работа программы показана на рисунке 3.

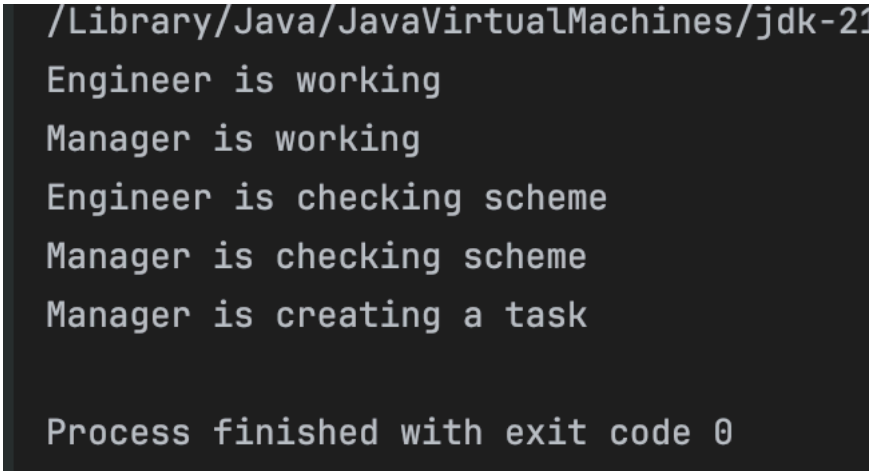

Рисунок 3 – Работа программы

Задание 4: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. Interface Здание <- abstract class Общественное Здание <- class Театр.

Код:

```
interface Building{
    void repair();
}

abstract class PublicBuilding implements Building{
    @Override
    public void repair() {
        System.out.println("Public building is being
repaired");
    }

    public void close(){
        System.out.println("Public building is closed");
    }

    public void open(){
        System.out.println("Public building is open");
    }

    public abstract void activate();
}

class Theatre extends PublicBuilding{

    @Override
```

```
    public void repair() {
        System.out.println("Theatre is being repaired");
    }

    @Override
    public void close(){
        System.out.println("Theatre is closed");
    }
    @Override
    public void open() {
        System.out.println("Theatre is opened");
    }

    @Override
    public void activate() {
        System.out.println("The play is being shown");
    }

    public void antract(){
        System.out.println("The play is on pause");
    }
}


public class Fourth_2 {
    public static void main(String[] args) {
        Building build_1 = new Theatre();
        build_1.repair();

        PublicBuilding build_2 = new Theatre();
        build_2.repair();
        build_2.close();
        build_2.open();
        build_2.activate();

        Theatre build_3 = new Theatre();
        build_3.repair();
        build_3.close();
        build_3.open();
        build_3.activate();
        build_3.antract();

    }
}
```

Работа программы показана на рисунке 4.

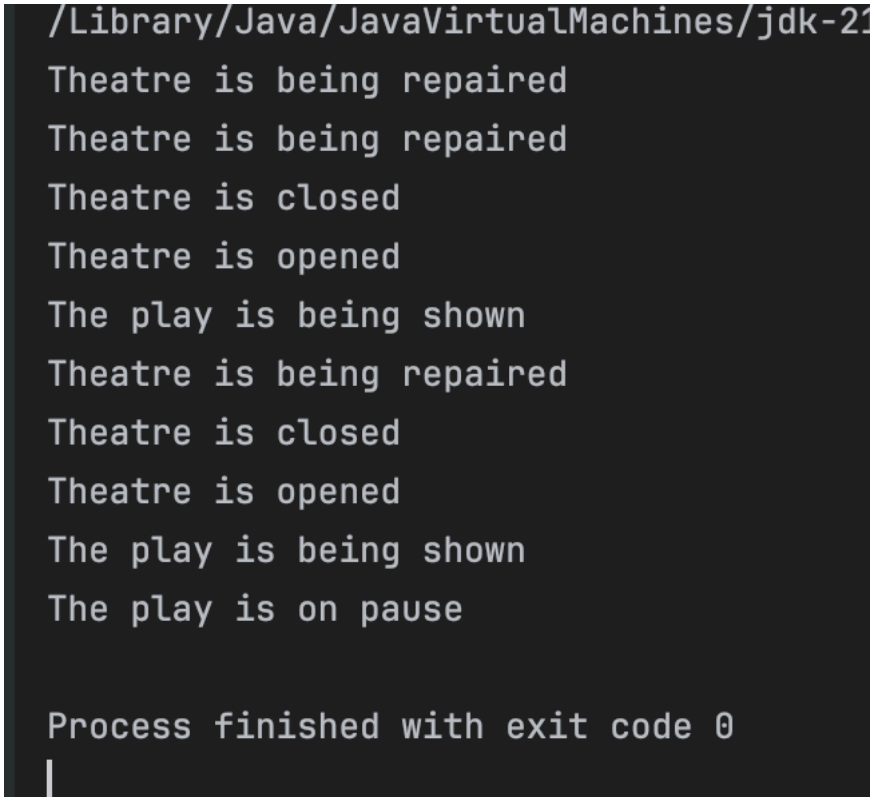

Рисунок 4 – Работа программы

Вывод: была изучена работа внутренних классов и интерфейсов