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 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №4

Название: Внутренние классы, интерфейсы

Дисциплина: Языки программирования для работы с большими данными

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

Москва, 2023 г.

Цель: получить опыт работы с внутренними классами и интерфейсами.

Вариант 1 – задание 9

Условие: создать класс Park (парк) с внутренним классом, с помощью объектов которого можно хранить информацию об аттракционах, времени их работы и стоимости.

Код:

```
import java.util.ArrayList;
import java.util.List;

class Park {
    private final List<Attraction> attractions;

    public Park() {
        attractions = new ArrayList<>();
    }

    public void addAttraction(String name, int price, String
startTime, String endTime) {
        attractions.add(new Attraction(name, price, startTime,
endTime));
    }

    public void removeAttraction(String name) {
        for (int i = 0; i < attractions.size(); i++) {
            if (attractions.get(i).getName().equals(name)) {
                attractions.remove(i);
                return;
            }
        }
    }

    @Override
    public String toString() {
        StringBuilder res = new StringBuilder();
        res.append("Аттракционы парка:\n");
        for (Park.Attraction attraction : this.attractions)
            res.append(attraction.getName()).append(" -
").append(attraction.getPrice())
                    .append(" рублей\nРаботает с
").append(attraction.getStartTime())
                    .append(" до
").append(attraction.getEndTime()).append("\n");
        return res.toString();
    }

    public static class Attraction {
        private final String name;
        private final int price;
        private final String startTime;
        private final String endTime;
```

```
        public Attraction(String name, int price, String
startTime, String endTime) {
            this.name = name;
            this.price = price;
            this.startTime = startTime;
            this.endTime = endTime;
        }

        public String getName() {
            return name;
        }

        public int getPrice() {
            return price;
        }

        public String getStartTime() {
            return startTime;
        }

        public String getEndTime() {
            return endTime;
        }
    }
}

public class var1_ex9 {
    public static void main(String[] args) {
        Park park = new Park();

        park.addAttraction("Карусель", 100, "10:00", "18:00");
        park.addAttraction("Горка", 150, "11:00", "19:00");
        park.addAttraction("Качели", 50, "12:00", "20:00");

        System.out.println(park);

        park.removeAttraction("Качели");

        System.out.println(park);
    }
}
```

Результат выполнения:

```
Аттракционы парка:
Карусель - 100 рублей
Работает с 10:00 до 18:00
Горка - 150 рублей
Работает с 11:00 до 19:00
Качели - 50 рублей
Работает с 12:00 до 20:00

Аттракционы парка:
Карусель - 100 рублей
Работает с 10:00 до 18:00
Горка - 150 рублей
Работает с 11:00 до 19:00
```

Рисунок 1 – Результат выполнения 1.9

Вариант 1 – задание 10

Условие: создать класс Cinema (кино) с внутренним классом, с помощью объектов которого можно хранить информацию об адресах кинотеатров, фильмах и времени сеансов.

Код:

```
import java.util.ArrayList;

class Cinema {
    private final ArrayList<String> addresses = new
ArrayList<>();
    private final ArrayList<Movie> movies = new ArrayList<>();

    public void addAddress(String address) {
        addresses.add(address);
    }

    public void removeAddress(String address) {
        addresses.remove(address);
    }

    public void addMovie(String title, String director, int
duration) {
        movies.add(new Movie(title, director, duration));
    }

    public void removeMovie(String title) {
        for (Movie movie : movies) {
            if (movie.getTitle().equals(title)) {
                movies.remove(movie);
                break;
            }
        }
    }
```

```
    @Override
    public String toString() {
        StringBuilder res = new StringBuilder();
        res.append("Адреса кинотеатров:\n");
        for (String address : this.addresses)
            res.append(address).append("\n");
        res.append("Фильмы:\n");
        for (Cinema.Movie movie : this.movies)
            res.append(movie.getTitle()).append(" (реж.
").append(movie.getDirector())
                    .append(") -
").append(movie.getDuration()).append("\n");
        return res.toString();
    }

    public static class Movie {
        private final String title;
        private final String director;
        private final int duration;

        public Movie(String title, String director, int
duration) {
            this.title = title;
            this.director = director;
            this.duration = duration;
        }

        public String getTitle() {
            return title;
        }

        public String getDirector() {
            return director;
        }

        public int getDuration() {
            return duration;
        }
    }
}

public class var1_ex10 {
    public static void main(String[] args) {
        Cinema cinema = new Cinema();

        cinema.addAddress("ул. Ленина, 5");
        cinema.addAddress("ул. Пушкина, 10");

        cinema.addMovie("Звездные войны", "Джордж Лукас", 120);
        cinema.addMovie("Терминатор", "Джеймс Кэмерон", 108);

        System.out.println(cinema);
```

```
    }
}
```

Результат выполнения:

```
Адреса кинотеатров:
ул. Ленина, 5
ул. Пушкина, 10
Фильмы:
Звездные войны (реж. Джордж Лукас) - 120
Терминатор (реж. Джеймс Кэмерон) - 108
```

Рисунок 2 – Результат выполнения 1.10

Вариант 2 – задание 9

Условие: Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.

Код:

```
interface Furniture {
    void move();
    void disassemble();
}

// Абстрактный класс Closet, который реализует интерфейс
Furniture
abstract class Closet implements Furniture {
    protected int height;
    protected int width;
    protected int depth;
    protected String material;

    public Closet(int height, int width, int depth, String
material) {
        this.height = height;
        this.width = width;
        this.depth = depth;
        this.material = material;
    }

    public void move() {
        System.out.println("Перемещаем шкаф");
    }

    public abstract void assemble();

    public void disassemble() {
        System.out.println("Разбираем шкаф");
    }
}
```

```
// Класс BookCloset, который наследует абстрактный класс Closet
class BookCloset extends Closet {
    private final int shelvesCount;

    public BookCloset(int height, int width, int depth, String
material, int shelvesCount) {
        super(height, width, depth, material);
        this.shelvesCount = shelvesCount;
    }

    public void assemble() {
        System.out.println("Собираем книжный шкаф");
    }

    public void storeBooks() {
        System.out.println("Книги хранятся на " + shelvesCount +
" полках");
    }
}

public class var2_ex9 {
    public static void main(String[] args) {
        BookCloset bookCloset = new BookCloset(200, 80, 40,
"дерево", 4);
        bookCloset.move();
        bookCloset.assemble();
        bookCloset.storeBooks();
        bookCloset.disassemble();
    }
}
```

Результат выполнения:

```
Перемещаем шкаф
Собираем книжный шкаф
Книги хранятся на 4 полках
Разбираем шкаф
```

Рисунок 3 – Результат выполнения 2.9

Вариант 2 – задание 10

Условие: Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Фильм <- class Отечественный Фильм <- class Комедия.

Код:

```
interface Film {
    void play(); // метод для проигрывания фильма
    void stop(); // метод для остановки проигрывания фильма
}

// Абстрактный класс DomesticFilm
```

```
abstract class DomesticFilm implements Film {
    protected String title; // название фильма
    protected int duration; // продолжительность фильма в
минутах

    public DomesticFilm(String title, int duration) {
        this.title = title;
        this.duration = duration;
    }

    public String getTitle() {
        return title;
    }

    public int getDuration() {
        return duration;
    }
}

// Класс Comedy
class Comedy extends DomesticFilm {
    private final int year;

    public Comedy(String title, int duration, int year) {
        super(title, duration);
        this.year = year;
    }

    public int getYear() {
        return year;
    }

    @Override
    public void play() {
        // код для проигрывания комедии
        System.out.println("Проигрывается комедия \"" + title +
"\" " + year +
                " года продолжительностью " + duration + "
минут");
    }

    @Override
    public void stop() {
        // код для остановки проигрывания комедии
        System.out.println("Проигрывание комедии \"" + title +
"\" было остановлено");
    }
}

public class var2_ex10 {
    public static void main(String[] args) {
        Comedy comedy = new Comedy("Брильянтовая рука", 98,
1968);
```

```
        comedy.play();

        comedy.stop();
    }
}
```

Результат выполнения:

```
Проигрывается комедия "Брильянтовая рука" 1968 года продолжительностью 98 минут
Проигрывание комедии "Брильянтовая рука" было остановлено
```

Рисунок 4 – Результат выполнения 2.10

Вывод: в ходе выполнения лабораторной работы был получен опыт работы с внутренними классами и интерфейсами.