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 И СИСТЕМЫ УПРАВЛЕНИЯ**

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

НАПРАВЛЕНИЕ ПОДГОТОВКИ 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: Языка программирования для работы с большими данными

Студент ИУ6-23М _____ Д.Д. Капитонов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

Вариант 1 и 2, номера 9,10.

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

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

Код программы:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

9. interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.
10. interface Фильм <- class Отечественный Фильм <- class Комедия.

Код программы:

```
package daniil;

import java.util.ArrayList;

public class lab {
   public static class Park{
      String name;
      public ArrayList<Attraction> list_attr;

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

      public Park(String name) {
         this.name = name;
         list_attr = new ArrayList<>();
      }

      public void add_attr(String name, int drive_time, int open_time, int close_time, int cost){
         list_attr.add(new Attraction(name, drive_time, open_time, close_time, cost));
      }


      public static class Attraction{
//          хранить информацию об аттракционах, времени их работы и стоимости.
         String name;
         int drive_time;
         int open_time;
         int close_time;
         int cost;

         public Attraction(String name, int drive_time, int open_time, int close_time, int cost) {
            this.name = name;
            this.drive_time = drive_time;
```

```
        this.open_time = open_time;
        this.close_time = close_time;
        this.cost = cost;
    }

    @Override
    public String toString() {
        return "Attraction{" +
                "name='" + name + '\'' +
                ", drive_time=" + drive_time +
                ", open_time=" + open_time +
                ", close_time=" + close_time +
                ", cost=" + cost +
                '}';
    }
}

public static class Cinema{
    String name;
    ArrayList<Addreses> cin_list;

    public Cinema() {
        cin_list = new ArrayList<>();
    }

    public Cinema(String name) {
        this.name = name;
        cin_list = new ArrayList<>();
    }
    public void add_new_address(String addres, int time_open, int time_close){
        cin_list.add(new Addreses(addres, time_open, time_close));
    }

    @Override
    public String toString() {
        return "Cinema{" +
                "name='" + name + '\'' +
                ", cin_list=" + cin_list +
                '}';
    }

    public static class Addreses{
        String addres;
        int time_open;
        int time_close;
        ArrayList<String> film_list;
        ArrayList<String> time_spend;
        ArrayList<Integer> cost;

        public Addreses() {
            film_list = new ArrayList<>();
            time_spend = new ArrayList<>();
```

```
            cost = new ArrayList<>();
        }

        public Addreses(String addres, int time_open, int time_close) {
            this.addres = addres;
            this.time_open = time_open;
            this.time_close = time_close;
            film_list = new ArrayList<>();
            time_spend = new ArrayList<>();
            cost = new ArrayList<>();
        }
        public void add_new_film(String film, String time, Integer prise){
            film_list.add(new String(film));
            time_spend.add(new String(time));
            cost.add(new Integer(prise));
        }

        @Override
        public String toString() {
            return "Addreses{" +
                    "addres='" + addres + '\'' +
                    ", time_open=" + time_open +
                    ", time_close=" + time_close +
                    ", film_list=" + film_list +
                    ", time_spend=" + time_spend +
                    ", cost=" + cost +
                    '}';
        }
    }
  }
}

interface Furniture{
    public String getname();
}

public static abstract class Wardrobe implements Furniture{
    String name;
    int cost;
    public String getname(){
        return this.name;
    }

    public Wardrobe() {
    }

    public Wardrobe(String name, int cost) {
        this.name = name;
        this.cost = cost;
    }
    public void test(){
        System.out.println("Test");
```

```
    }
  }
  public static class Book_war extends Wardrobe{
    @Override
    public void test(){
      System.out.println("Book");
    }

  }


  interface Film{
    String getname();
  }
  public static class Rus_film implements Film{
    String name;
    int duration;

    public Rus_film() {
    }

    public Rus_film(String name, int duration) {
      this.name = name;
      this.duration = duration;
    }

    public String getname(){
      return this.name;
    }

  }
  public static class Comedy extends Rus_film{
    int time_release;

    public Comedy(int time_release) {
      this.time_release = time_release;
    }

    public Comedy() {
    }
    public String getname(){
      return Integer.toString(this.time_release);
    }
  }
  public static void main(String[] args) {


  }
}
```

Вывод: научились работать с внутренними классами и интерфейсами.