|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/04 Компьютерный анализ и интерпретация больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Название:** Внутренние классы, интерфейсы.

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-21М |  | А.А. Поляков |
|  | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Преподаватель |  |  | П.В. Степанов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

2021 г.

# Цель лабораторной работы

Освоить и получить навыки работы с языком программирования Java для применения его в работе с большими данными.

**Вариант 1**

**Задача 1**

# Задание

Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

# Текст программы

import java.util.Scanner;

public class Main {

static public void main(String[]args){

City ct = new City();

System.out.println(ct.info.toString());

}

}

class City {

public static Scanner ob = new Scanner(System.in);

public Info info;

private String city;

public City() {

System.out.print("Введите название города: ");

setCity(ob.next());

info = new Info();

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city=city;

}

public class Info {

int pr,st,sq;

public Info() {

System.out.print("Кол-во проспектов в городе:");

setPr(ob.nextInt());

System.out.print("Количество улиц в городе:");

setSt(ob.nextInt());

System.out.print("кол-во площадей в городе:");

setSq(ob.nextInt());

}

public int getPr(){

return pr;

}

public void setPr(int pr){

this.pr=pr;

}

public int getSt(){

return st;

}

public void setSt(int st){

this.st=st;

}

public int getSq(){

return sq;

}

public void setSq(int sq){

this.sq=sq;

}

@Override

public String toString() {

return "в городе "+getCity()+" "+getPr()+" проспектов,"+getSt()+" улицы и "+getSq()+" площадей";

}

public void print(){

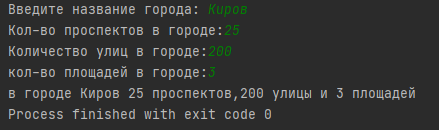
System.out.print("в городе "+getCity()+" "+getPr()+" проспектов,"+getSt()+" улицы и "+getSq()+" площадей");

}

}

}

# Результат



**Вариант 1**

**Задача 2**

# Задание

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

# Текст программы

import java.util.ArrayList;

import java.util.List;

public class Main {

static public void main(String[]args){

Cinema c = new Cinema();

c.printSeanses();

}

}

class Cinema {

List<Info> seanses = new ArrayList<>();

//public Info info;

//public String city;

public Cinema() {

seanses.add(new Info("Октябрь", "Железный человек", "25 янв 19:00"));

seanses.add(new Info("Октябрь", "Джокер", "25 янв 21:30"));

seanses.add(new Info("Октябрь", "1408", "26 янв 00:00"));

}

public void printSeanses() {

for (Info seanse : seanses) {

seanse.print();

}

}

public class Info {

public String k, f, t;

public Info(String k, String f, String t) {

this.k = k;

this.f = f;

this.t = t;

}

public String getK() {

return k;

}

public void setK(String k) {

this.k = k;

}

public String getF() {

return f;

}

public void setF(String f) {

this.f = f;

}

public String getT() {

return t;

}

public void setT(String t) {

this.t = t;

}

public void print() {

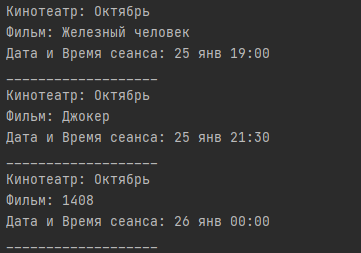
System.out.print("Кинотеатр: " + getK() + "\nФильм: " + getF() + "\nДата и Время сеанса: " + getT() + "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

}

# Результат



**Вариант 2**

**Задача 1**

# Задание

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

interface Корабль <- abstract class Военный Корабль <- class Авианосец.

# Текст программы

interface Ships {

int getDisplacement(); // получить водоизмещение

int getMaxSpeed(); // максимальую скорость

}

abstract class ArmyShip implements Ships {

protected String name; //Название

private int displacement; //водоизмещение

private int maxSpeed; //скорость

private int distanatMax; //максимальная дистанция

private int teamCount; //Число военных в команде

public ArmyShip(String name, int displacement, int maxSpeed, int distanatMax, int teamCount) {

this.name = name;

this.displacement = displacement;

this.maxSpeed = maxSpeed;

this.distanatMax = distanatMax;

this.teamCount = teamCount;

}

public abstract String getName();

@Override

public int getDisplacement() {

return displacement;

}

@Override

public int getMaxSpeed() {

return maxSpeed;

}

public int getDistanatMax() {

return distanatMax;

}

public int getTeamCount() {

return teamCount;

}

}

class AeroCarrier extends ArmyShip {

private String typeShip = "Авианосец";

private int planePlaces; //Вместимость корабля

public AeroCarrier(String name, int displacement, int maxSpeed, int distanatMax, int teamCount, int planePlaces) {

super(name, displacement, maxSpeed, distanatMax, teamCount);

this.planePlaces = planePlaces;

}

@Override

public String getName() {

return name;

}

@Override

public String toString() {

return "~~~~~~'" + name + "\'~~~~~~\n" +

"Тип корабля '" + typeShip + "\';\n" +

"Численность экипажа = " + getTeamCount() + ";\n" +

"Максимальная дальность плаванья = " + getDistanatMax() + " км.\n" +

"Максимальная скорость = " + getMaxSpeed() + " узлов \n" +

"Водоизмещение = " + getDisplacement() + " тысяч тон \n" +

"Количество мест для авиатехники = " + getTeamCount() + " единиц \n" +

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

}

public class Main {

public static void main(String[] args) {

ArmyShip [] ships = new ArmyShip[2];

ships[0] = new AeroCarrier("Иван IV", 20000, 30, 10000, 120, 10);

ships[1] = new AeroCarrier("Иван Третий", 500, 13, 100, 40, 4);

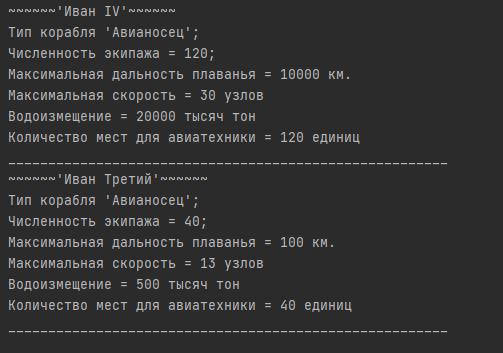
for (ArmyShip tmpSh : ships)

System.out.println(tmpSh);

}

}

# Результат



**Вариант 2**

**Задача 2**

# Задание

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

interface Здание <- abstract class Общественное Здание <- class Театр.

# Текст программы

interface Building {

int getHeight(); // высота

int getSquareMeters(); // площадь

}

abstract class PublicBuilding implements Building {

protected String adress; //Адрес

private int height; //высота

private int squareMeters; //площадь

public PublicBuilding(String adress, int height, int squareMeters) {

this.adress = adress;

this.height = height;

this.squareMeters = squareMeters;

}

public abstract String getAdress();

@Override

public int getHeight() {

return height;

}

@Override

public int getSquareMeters() {

return squareMeters;

}

}

class Theatre extends PublicBuilding {

private String type = "Театр ";

private String name;

private int halls; //Число залов

private int places; //Число мест в залах

public Theatre(String adress, int height, int squareMeters, String name, int halls, int places) {

super(adress, height, squareMeters);

this.name = name;

this.halls = halls;

this.places = places;

}

@Override

public String getAdress() {

return adress;

}

@Override

public String toString() {

return "~~~~~~'" + type + name + "\'~~~~~~\n" +

"Адрес: " + getAdress() + "\n" +

"Число залов: " + halls + "\n" +

"Количество мест: " + places + "\n" +

"Площадь = " + getSquareMeters() + " кв. м.\n" +

"Высота задния = " + getHeight() + " м \n" +

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

}

}

public class Main {

public static void main(String[] args) {

PublicBuilding [] ships = new PublicBuilding[2];

ships[0] = new Theatre("Москва, ул. Кирова, 10", 40, 450, "им. Высоцкого", 1, 3000);

ships[1] = new Theatre("Москва, ул. Смирнова, 34а", 60, 920, "им. Савичева", 3, 10000);

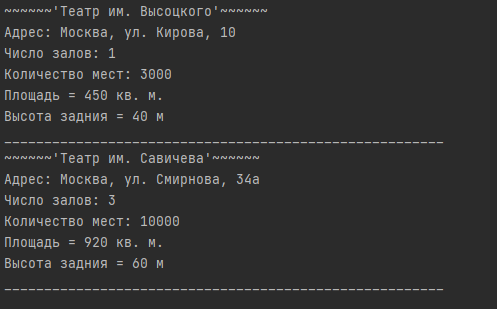
for (PublicBuilding tmpSh : ships)

System.out.println(tmpSh);

}

}

# Результат



**Ссылка на репозиторий**

<https://github.com/bmstu-iu6-21m/pract4-alpolyakov>

# Вывод

Был получен опыт работы с интерфейсами и внутренними классами при написании программ на языке Java.