|  |  |
| --- | --- |
| 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 **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 4**

**Вариант № 13**

**Название:** внутренние классы и интерфейсы

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | С.П.Пантелеев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**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 класса Main:

package Zadanie\_1;

public class Main {

public static void main(String[] args) {

Mobile mobile1 = new Mobile("samsung");

Mobile mobile2 = new Mobile("sony");

mobile1.add\_model("2","12", "A212", "snap\_dragon1", "3");

mobile1.add\_model("6","13", "A242", "snap\_dragon2", "32");

mobile2.add\_model("3","15", "A252", "snap\_dragon3", "23");

mobile2.add\_model("2","16", "A272", "snap\_dragon4", "35");

}

}

Код класса Mobile:

package Zadanie\_1;

import java.util.ArrayList;

import java.util.List;

public class Mobile {

private String brend\_name;

private List<Model> model;

public void add\_model(String kol\_ader, String kol\_px, String number\_model, String processor, String op){

model.add(new Model(kol\_ader, kol\_px, number\_model, processor, op));

}

public Mobile(String brend\_name) {

this.brend\_name = brend\_name;

model = new ArrayList<>();

}

public String getBrend\_name() {

return brend\_name;

}

public void setBrend\_name(String brend\_name) {

this.brend\_name = brend\_name;

}

public List<Model> getModel() {

return model;

}

public void setModel(List<Model> model) {

this.model = model;

}

private class Model{

private String kol\_ader;

private String kol\_px;

private String number\_model;

private String processor;

private String op;

public String getKol\_ader() {

return kol\_ader;

}

public void setKol\_ader(String kol\_ader) {

this.kol\_ader = kol\_ader;

}

public String getKol\_px() {

return kol\_px;

}

public void setKol\_px(String kol\_px) {

this.kol\_px = kol\_px;

}

public String getNumber\_model() {

return number\_model;

}

public void setNumber\_model(String number\_model) {

this.number\_model = number\_model;

}

public String getProcessor() {

return processor;

}

public void setProcessor(String processor) {

this.processor = processor;

}

public String getOp() {

return op;

}

public void setOp(String op) {

this.op = op;

}

public Model(String kol\_ader, String kol\_px, String number\_model, String processor, String op) {

this.kol\_ader = kol\_ader;

this.kol\_px = kol\_px;

this.number\_model = number\_model;

this.processor = processor;

this.op = op;

}

}

}

Работа программы показана на рисунке 1.

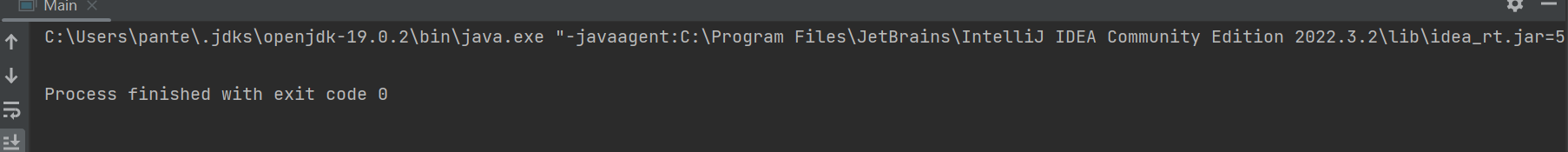


Рисунок 1 – Работа программы

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

Код класса Main:

package Zadanie\_2;

public class Main {

public static void main(String[] args) {

Vistovka vistovka1 = new Vistovka("pank", "9:00", "19:00", "20.01.2023");

Vistovka vistovka2 = new Vistovka("classic", "9:00", "19:00", "21.01.2023");

vistovka1.add\_picture("1", "roman");

vistovka1.add\_picture("2", "ivan");

vistovka2.add\_picture("3", "Sasha");

vistovka2.add\_picture("4", "Dasha");

}

}

Код класса Vistovka:

package Zadanie\_2;

import java.util.ArrayList;

import java.util.List;

public class Vistovka {

private String name\_vistovka;

private String time\_begin;

private String time\_end;

private String date;

private List<Picture> pictures;

public void add\_picture(String name\_picture, String name\_avtor){

pictures.add(new Picture(name\_picture, name\_avtor));

}

public Vistovka(String name\_vistovka, String time\_begin, String time\_end, String date) {

this.name\_vistovka = name\_vistovka;

this.time\_begin = time\_begin;

this.time\_end = time\_end;

this.date = date;

pictures = new ArrayList<>();

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getName\_vistovka() {

return name\_vistovka;

}

public void setName\_vistovka(String name\_vistovka) {

this.name\_vistovka = name\_vistovka;

}

public String getTime\_begin() {

return time\_begin;

}

public void setTime\_begin(String time\_begin) {

this.time\_begin = time\_begin;

}

public String getTime\_end() {

return time\_end;

}

public void setTime\_end(String time\_end) {

this.time\_end = time\_end;

}

public List<Picture> getPictures() {

return pictures;

}

public void setPictures(List<Picture> pictures) {

this.pictures = pictures;

}

private class Picture{

private String name\_of\_picture;

private String name\_of\_avtor;

public Picture(String name\_of\_picture, String name\_of\_avtor) {

this.name\_of\_picture = name\_of\_picture;

this.name\_of\_avtor = name\_of\_avtor;

}

public String getName\_of\_picture() {

return name\_of\_picture;

}

public void setName\_of\_picture(String name\_of\_picture) {

this.name\_of\_picture = name\_of\_picture;

}

public String getName\_of\_avtor() {

return name\_of\_avtor;

}

public void setName\_of\_avtor(String name\_of\_avtor) {

this.name\_of\_avtor = name\_of\_avtor;

}

}

}

Работа программы показана на рисунке 2.

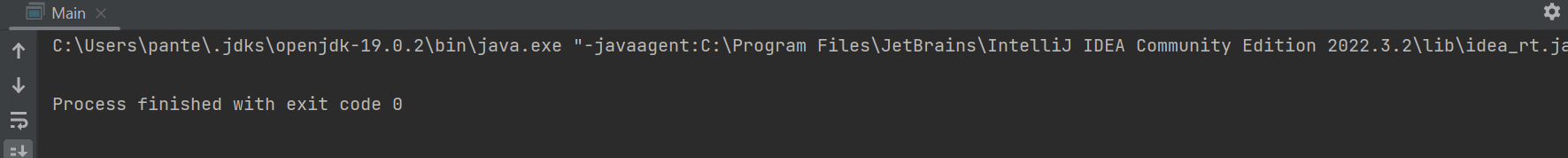


Рисунок 2 – Работа программы

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

Код класса Main:

package Zadanie\_3;

public class Main {

public static void main(String[] args) {

}

}

interface Worker{

void work();

}

class Inginer implements Worker{

private String name;

@Override

public void work() {

System.out.println("инжинер работает!!!!!");

}

public Inginer(String name) {

this.name = name;

}

}

class Rukovoditel extends Inginer{

private String name;

@Override

public void work() {

System.out.println("Руководитель работает!!!!!");

}

public Rukovoditel(String name, String name1) {

super(name);

this.name = name1;

}

}

Работа программы показана на рисунке 3.

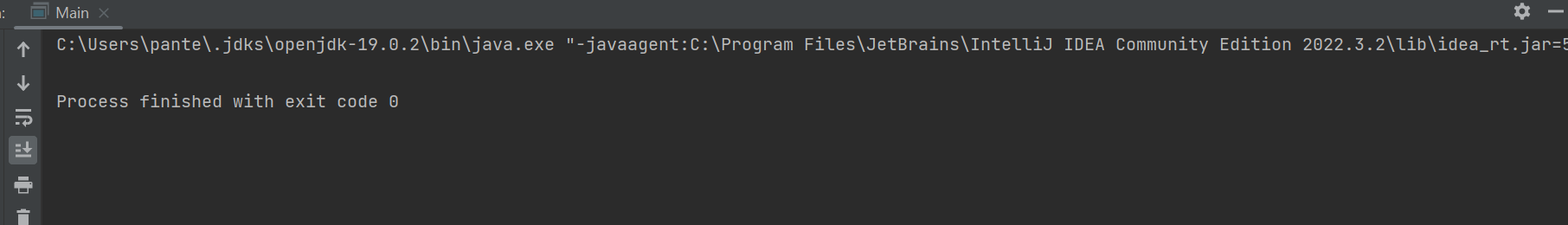


Рисунок 3 – Работа программы

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

Код класса Main:

package Zadanie\_4;

public class Main {

public static void main(String[] args) {

}

interface Zdanie{

void build();

}

abstract class Soshial\_zdanie implements Zdanie{

public String name;

public Soshial\_zdanie(String name) {

this.name = name;

}

@Override

public void build() {

System.out.println("Здание стоится");

}

abstract void close();

}

class Theater extends Soshial\_zdanie{

@Override

void close() {

System.out.println("Здание закрылось");

}

public Theater(String name) {

super(name);

}

}

}

Работа программы показана на рисунке 4.

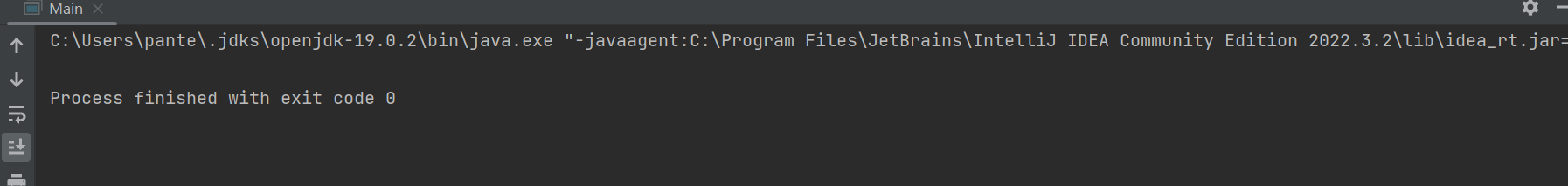


Рисунок 4 – Работа программы

**Вывод:** была изучена работа внутренних классов и интерфейсов в java.