|  |  |
| --- | --- |
| 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 № 3**

**Название:** Внутренние классы и интерфейсы

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

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

Москва, 2023

**Цель:** изучить и освоить 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 Mobile:

public class Mobile {  
 private Model model;  
  
 public Mobile(String name, long ram, String cpu, long memory) {  
 this.model = new Model(name, ram, cpu, memory);  
 }  
  
 public Model getModel() {  
 return model;  
 }  
  
 public void setModel(Model model) {  
 this.model = model;  
 }  
  
 public class Model {  
 private String name;  
 private long ram;  
 private String cpu;  
 private long memory;  
  
 public Model(String name, long ram, String cpu, long memory) {  
 this.name = name;  
 this.ram = ram;  
 this.cpu = cpu;  
 this.memory = memory;  
 }  
  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public long getRam() {  
 return ram;  
 }  
  
 public void setRam(long ram) {  
 this.ram = ram;  
 }  
  
 public String getCpu() {  
 return cpu;  
 }  
  
 public void setCpu(String cpu) {  
 this.cpu = cpu;  
 }  
  
 public long getMemory() {  
 return memory;  
 }  
  
 public void setMemory(long memory) {  
 this.memory = memory;  
 }  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 Mobile mobile = new Mobile("Redmi 9", 8, "i5", 32);  
 Mobile.Model model = mobile.getModel();  
 System.out.println(model.getName());  
 }  
}

Результат работы программы представлен на рисунке 1.

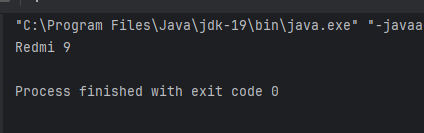


Рисунок 1 – Результат работы программы

**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 Gallery:

import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
public class Gallery {  
 List<Picture> pictures;  
  
 public Gallery() {  
 this.pictures = new ArrayList<>();  
 }  
  
 public void addPicture(String name, String author, Date date) {  
 Picture picture = new Picture(name, author,date);  
 this.pictures.add(picture);  
 }  
  
 public void removePicture(int i) {  
 this.pictures.remove(i);  
 }  
  
 public List<Picture> getPictures() {  
 return this.pictures;  
 }  
  
 public class Picture {  
 private String name;  
 private String author;  
 private Date date;  
  
 public Picture(String name, String author, Date date) {  
 this.name = name;  
 this.author = author;  
 this.date = date;  
 }  
  
 public void printInfo() {  
 System.out.printf("%s, %s, дата выставки: %s", name,author, date.toString());  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public Date getDate() {  
 return date;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
 }  
}

Код класса Main:

import java.util.Date;  
  
public class Main {  
 public static void main(String[] args) {  
 Gallery gallery = new Gallery();  
 gallery.addPicture("Чёрный квадрат", "Малевич", new Date());  
 gallery.getPictures().get(0).printInfo();  
 }  
}

Результат работы программы показан на рисунке 2.

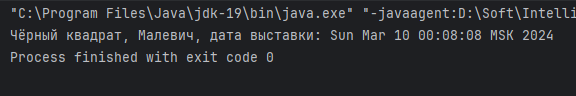


Рисунок 2 – Результат работы программы

**Задание 3:** реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Сотрудник <- class Инженер <- class Руководитель.

Код интерфейса Employer:

public interface Employer {  
 public void work();  
}

Код класса Engineer:

public class Engineer implements Employer {  
 protected String name;  
  
 public Engineer(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public void work() {  
 System.out.printf("Инженер %s работает за на предприятии\n", name);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

Код класса Manager:

import java.util.Date;  
  
public class Manager extends Engineer {  
 private Date date;  
 public Manager(String name, Date date) {  
 super(name);  
 this.date = date;  
 }  
  
 @Override  
 public void work() {  
 super.work();  
 System.out.printf("%s руководит начиная с %s\n", name, date.toString());  
 }  
  
 public Date getDate() {  
 return date;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
}

Код класса Main:

import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
 List<Employer> employers = new ArrayList<>();  
 employers.add(new Engineer("Вася"));  
 employers.add(new Manager("Петя", new Date()));  
  
 employers.forEach(Employer::work);  
  
 }  
}

Результат работы программы представлен на рисунке 3.

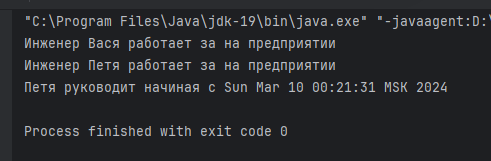
****

Рисунок 3 – Результат работы 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 интерфейса Building:

public interface Building {  
 public void build();  
}

Код класса SocialBuilding:

public abstract class SocialBuilding implements Building {  
 protected String name;  
 protected String address;  
  
 public SocialBuilding(String name, String address) {  
 this.name = name;  
 this.address = address;  
 }  
  
 @Override  
 public void build() {  
 System.out.printf("Было построено здание %s на %s\n", name, address);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
}

Код класса Theater:

import java.util.Date;  
  
public class Theater extends SocialBuilding {  
 private Date date;  
  
 public Theater(String name, String address, Date date) {  
 super(name, address);  
 this.date = date;  
 }  
  
 @Override  
 public void build() {  
 super.build();  
 System.out.printf("Выступление в %s будет %s\n", name, date.toString());  
 }  
  
 public Date getDate() {  
 return date;  
 }  
  
 public void setDate(Date date) {  
 this.date = date;  
 }  
}

Код класса Main:

import java.util.Date;  
  
public class Main {  
 public static void main(String[] args) {  
 Building building = new Theater("Дюсалей", "ул. Бауманская", new Date());  
 building.build();  
 }  
}

Результат работы программы показан на рисунке 4.

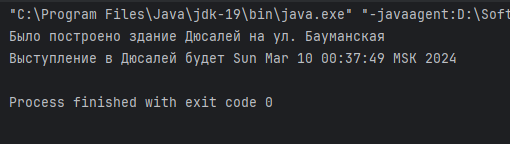


Рисунок 4 – Результат работы программы

**Вывод:** была освоена работа внутренних классов и интерфейсов в Java.