|  |  |
| --- | --- |
| 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**

**Название:** Внутренние классы, интерфейсы

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

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

Москва, 2023

**Задание 1:** создать класс Календарь с внутренним классом, с помощью объектов которого можно хранить информацию о выходных и праздничных днях.

Решение задания представлено в листингах 1-2.

Листинг 1 – Программный код модуля Main.java для задания 1

public class Main {

public static void main(String[] args) {

Calendar calendar = new Calendar();

calendar.addHoliday("2023-01-01", "New Year's Day");

calendar.addHoliday("2023-03-08", "International Women's Day");

calendar.printHolidays();

}

}Листинг 2 – Программный код модуля Matrix.java для задания 1

import java.util.concurrent.ThreadLocalRandom;

public class Matrix{

int colNum;

int rowNum;

int[][] matrix;

public void output(){

for (int i = 0; i < rowNum; i++){

for (int j = 0; j < colNum; j++){

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

}

private void init() {

for (int i = 0; i < rowNum; i++) {

for (int j = 0; j < colNum; j++) {

matrix[i][j] = ThreadLocalRandom.current().nextInt(50 \* (-1), 50 + 1);

}

}

}

public Matrix(int col, int row){

colNum = col;

rowNum = row;

matrix = new int[rowNum][colNum];

init();

}

public Matrix(int colRow) {

colNum = colRow;

rowNum = colRow;

init();

}

}

Листинг 2 – Программный код модуля Calendar.java для задания 1

import java.util.ArrayList;

import java.util.List;

public class Calendar {

private List<Holiday> holidays;

public Calendar() {

holidays = new ArrayList<>();

}

public void addHoliday(String date, String name) {

Holiday holiday = new Holiday(date, name);

holidays.add(holiday);

}

public void printHolidays() {

for (Holiday holiday : holidays) {

System.out.println(holiday.getDate() + " - " + holiday.getName());

}

}

public class Holiday {

private String date;

private String name;

public Holiday(String date, String name) {

this.date = date;

this.name = name;

}

public String getDate() {

return date;

}

public String getName() {

return name;

}

}

}

**Задание 2:** создать класс Shop (магазин) с внутренним классом, с помощью объектов которого можно хранить информацию об отделах, товарах и услуг.

Решение задания представлено в листингах 3-4.

Листинг 3 – Программный код модуля Main.java для задания 2

public class Main {

public static void main(String[] args) {

Shop shop = new Shop("MyShop", 3);

shop.addDepartment(0, "Electronics", 100);

shop.addDepartment(1, "Clothing", 200);

shop.addDepartment(2, "Home Appliances", 150);

shop.displayDepartments();

}

}

Листинг 4 – Программный код модуля Shop.java для задания 2

public class Shop {

private String name;

private Department[] departments;

public Shop(String name, int numDepartments) {

this.name = name;

this.departments = new Department[numDepartments];

}

public void addDepartment(int index, String departmentName, int numProducts) {

departments[index] = new Department(departmentName, numProducts);

}

public void displayDepartments() {

for (Department department : departments) {

System.out.println("Department: " + department.getName());

System.out.println("Products: " + department.getNumProducts());

System.out.println("-------------------------");

}

}

public class Department {

private String name;

private int numProducts;

public Department(String name, int numProducts) {

this.name = name;

this.numProducts = numProducts;

}

public String getName() {

return name;

}

public int getNumProducts() {

return numProducts;

}

}

}

**Задание 3:** реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующего класса.

interface Mobile <- abstract class Siemens Mobile <- class Model.

Решение задания представлено в листингах 5-8.

Листинг 5 – Программный код модуля Main.java для задания 3

public class Main {

public static void main(String[] args) {

SiemensMobile mobile = new Model("Siemens A100");

mobile.call("123456789");

mobile.sendMessage("987654321", "Hello!");

mobile.customizeSettings();

}

}

Листинг 6 – Программный код модуля Mobile.java для задания 3

public interface Mobile {

void call(String number);

void sendMessage(String number, String message);

}Листинг 7 – Программный код модуля BookArray.java для задания 3

import java.util.ArrayList;

public class BookArray {

private Book[] books;

public BookArray(Book[] books) {

this.books = books;

}

public Book[] getBooks() {

return books;

}

public Book[] getByAuthor(String author) {

ArrayList<Book> result = new ArrayList<Book>();

for (Book book : books) {

for (String bookAuthor : book.getAuthors()) {

if (bookAuthor.equals(author)) {

result.add(book);

break;

}

}

}

return result.toArray(new Book[result.size()]);

}

public Book[] getByPublisher(String publisher) {

ArrayList<Book> result = new ArrayList<Book>();

for (Book book : books) {

if (book.getPublisher().equals(publisher)) {

result.add(book);

}

}

return result.toArray(new Book[result.size()]);

}

public Book[] getAfterYear(int year) {

ArrayList<Book> result = new ArrayList<Book>();

for (Book book : books) {

if (book.getYear() > year) {

result.add(book);

}

}

return result.toArray(new Book[result.size()]);

}

}

Листинг 7 – Программный код модуля Mobile.java для задания 3

public class Model extends SiemensMobile {

public Model(String model) {

super(model);

}

// Implement the customizeSettings method specific to Model

@Override

public void customizeSettings() {

System.out.println("Customizing settings for " + getModel());

}

}

Листинг 8 – Программный код модуля SiemenseMobile.java для задания 3

public abstract class SiemensMobile implements Mobile {

private String model;

public SiemensMobile(String model) {

this.model = model;

}

public String getModel() {

return model;

}

@Override

public void call(String number) {

System.out.println("Calling " + number + " from " + model);

}

// Implement the sendMessage method from the Mobile interface

@Override

public void sendMessage(String number, String message) {

System.out.println("Sending message to " + number + " from " + model + ": " + message);

}

public abstract void customizeSettings();

}

**Задание 4:** реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующего класса.

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

Решение задания представлено в листингах 9-12.

Листинг 9 – Программный код модуля Main.java для задания 4

public class Main {

public static void main(String[] args) {

Warship ship = new AircraftCarrier("USS Enterprise", 50);

ship.sail();

ship.dock();

ship.fire();

if (ship instanceof AircraftCarrier) {

AircraftCarrier aircraftCarrier = (AircraftCarrier) ship;

aircraftCarrier.launchAircraft();

}

}

}

Листинг 10 – Программный код модуля AircraftCarrier.java для задания 4

public class AircraftCarrier extends Warship {

private int aircraftCount;

public AircraftCarrier(String name, int aircraftCount) {

super(name);

this.aircraftCount = aircraftCount;

}

// Реализация метода fire(), специфичного для Авианосца

@Override

public void fire() {

System.out.println(name + " is launching aircraft for an airstrike.");

}

// Дополнительный метод специфичный для Авианосца

public void launchAircraft() {

System.out.println(name + " is launching aircraft.");

}

}

Листинг 11 – Программный код модуля Ship.java для задания 4

public interface Ship {

void sail();

void dock();

}

Листинг 12 – Программный код модуля Warship.java для задания 4

public abstract class Warship implements Ship {

protected String name;

public Warship(String name) {

this.name = name;

}

public String getName() {

return name;

}

// Реализация метода sail() из интерфейса Корабль

@Override

public void sail() {

System.out.println(name + " is sailing.");

}

// Реализация метода dock() из интерфейса Корабль

@Override

public void dock() {

System.out.println(name + " is docking.");

}

// Абстрактный метод, специфичный для Военного Корабля

public abstract void fire();

}

**Вывод:** в ходе выполнения данной лабораторной работы была освоена работа с внутренними классами и изучены приемы работы с интерфейсами на языке Java.