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

**Отчет по лабораторной работе № 3**

***по дисциплине «Языки программирования для работы с большими данными»***

Студент ИУ6-21М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Щербакова**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П. В. Степанов**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**Задание 1:**

**Вариант 4:** Определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы. Определить, какая из матриц имеет наименьшую первую и вторую нормы.

**Код программы:**

**Файл Matrix.java**

package Matrix;

public class Matrix {

public int n = 0;

public int[][] head = null;

public Matrix() {}

public Matrix(int[][] matrix, int n) {

this.n = n;

this.head = new int[n][n];

this.fill(matrix);

}

public Matrix(int n) {

this.n = n;

this.head = new int[n][n];

}

public void fillRandom() {

for (int i = 0; i < this.n; i++) {

for (int j = 0; j < this.n; j++) {

this.head[i][j] = (int) (Math.*random*() \* (n \* 2 + 1) - n);

}

}

}

public void fill(int[][] matrix) {

for (int i = 0; i < this.n; i++) {

for (int j = 0; j < this.n; j++) {

this.head[i][j] = matrix[i][j];

}

}

}

public void print() {

for (int[] row: this.head) {

for (int item: row) {

System.***out***.printf("%d ", item);

}

System.***out***.println();

}

System.***out***.println();

}

public Matrix doAddition(Matrix matrix) {

for (int i = 0; i < this.n; i++) {

for (int j = 0; j < this.n; j++) {

this.head[i][j] += matrix.head[i][j];

}

}

return this;

}

public Matrix doSubtraction(Matrix matrix) {

for (int i = 0; i < this.n; i++) {

for (int j = 0; j < this.n; j++) {

this.head[i][j] -= matrix.head[i][j];

}

}

return this;

}

public Matrix doMultiplication(Matrix matrix) {

if (this.n != matrix.n) {

return null;

}

Matrix result = new Matrix(this.n);

for (int i = 0; i < this.n; i++) {

for (int j = 0; j < this.n; j++) {

int sum = 0;

for (int k = 0; k < this.n; k++) {

sum += this.head[i][k] \* matrix.head[k][j];

}

result.head[i][j] = sum;

}

}

this.fill(result.head);

return result;

}

public int norm(int range) {

int max = Integer.***MIN\_VALUE***;

for (int i = 0; i < this.n; i++) {

int sum = 0;

for (int j = 0; j < this.n; j++) {

switch (range) {

case 1:

sum += this.head[i][j];

break;

case 2:

sum += this.head[j][i];

}

}

if (max < sum) {

max = sum;

}

}

return max;

}

}

**Файл Main.java**

package Matrix;

import Matrix.Matrix;

public class Main {

public static Matrix findMatrixWithMaxNorm(int range, Matrix[] array) {

int max = Integer.***MIN\_VALUE***;

int maxI = -1;

for (int i = 0; i < array.length; i++) {

int norm = array[i].norm(range);

if (max < norm) {

max = norm;

maxI = i;

}

}

return array[maxI];

}

public static void main(String []args) {

int n = 5;

Matrix matrix = new Matrix(n);

Matrix matrix2 = new Matrix(n);

matrix.fillRandom();

matrix2.fillRandom();

matrix.print();

matrix2.print();

matrix.doAddition(matrix2);

matrix.print();

matrix.doSubtraction(matrix2);

matrix.print();

matrix.doMultiplication(matrix2);

matrix.print();

System.***out***.println(matrix.norm(1));

System.***out***.println(matrix.norm(2));

Matrix[] array = new Matrix[n];

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

array[i] = new Matrix(n);

array[i].fillRandom();

}

Main.*findMatrixWithMaxNorm*(1, array).print();

Main.*findMatrixWithMaxNorm*(2, array).print();

}

}

**Вывод программы**

3 -3 -4 -4 3

5 3 -1 1 -4

3 0 1 2 5

4 -3 3 3 -4

1 4 0 0 -3

1 0 -4 -1 2

1 3 4 -4 -2

-4 3 -2 -4 0

-3 3 2 5 4

5 -2 -3 -3 1

4 -3 -8 -5 5

6 6 3 -3 -6

-1 3 -1 -2 5

1 0 5 8 0

6 2 -3 -3 -2

3 -3 -4 -4 3

5 3 -1 1 -4

3 0 1 2 5

4 -3 3 3 -4

1 4 0 0 -3

43 -39 -33 -4 -1

-11 17 8 4 4

18 -1 -25 -12 19

-40 17 -16 23 22

-10 18 21 -8 -9

22

35

1 -2 5 4 0

2 5 -2 4 2

5 -3 -2 4 3

3 1 -3 -1 -2

5 0 5 2 1

1 -2 5 4 0

2 5 -2 4 2

5 -3 -2 4 3

3 1 -3 -1 -2

5 0 5 2 1

**Задание 2**

**Вариант 6:** House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.

**Код программы:**

**Файл Main.java**

package House;

import House.House;

public class Main {

public static void main(String[] args) {

int n = 10;

House[] houses = House.*createFactory*(n);

System.***out***.println("Список квартир");

for (House house: houses) {

System.***out***.println(house.toString());

}

int roomCount = 5;

System.***out***.println("Список квартир , имеющих заданное число комнат");

for (House house: houses) {

boolean isValid = house.filterByRoomCount(roomCount);

if (isValid) {

System.***out***.println(house.toString());

}

}

int low = 2;

int high = 15;

System.***out***.println("Список квартир , имеющих заданное число комнат и этаж в диапазоне");

for (House house: houses) {

boolean isValid = house.filterByRoomCount(roomCount) && house.filterByFloor(low, high);

if (isValid) {

System.***out***.println(house.toString());

}

}

int square = 100;

System.***out***.println("Список квартир , имеющих площадь, превосходящую заданную");

for (House house: houses) {

boolean isValid = house.filterBySquare(square);

if (isValid) {

System.***out***.println(house.toString());

}

}

}

}

**Файл House.java**package House;

import java.util.Date;

public class House {

public static enum *BuildingTypes* {

***Living***,

***Country***,

***Industrial***,

};

private int id = 0;

private int number = 0;

private int square = 0;

private int floor = 0;

private int roomCount = 0;

private String street = "";

private House.*BuildingTypes* buildingTtype;

private Date lifetime;

public House() {

this.setId((int) Math.*random*());

this.setNumber((int) (Math.*random*() \* 1000 + 1));

this.setSquare((int) (Math.*random*() \* 200 + 1));

this.setFloor((int) (Math.*random*() \* 20 + 1));

this.setRoomCount((int) (Math.*random*() \* 5 + 1));

this.setStreet("Street");

this.setBuildingTtype(House.*BuildingTypes*.***Living***);

this.setLifetime(new Date());

}

public House(House house) {

this.setId((int) Math.*random*());

this.setNumber(house.number);

this.setSquare(house.square);

this.setFloor(house.floor);

this.setRoomCount(house.roomCount);

this.setStreet(house.street);

this.setBuildingTtype(house.buildingTtype);

this.setLifetime(house.lifetime);

}

public House(

int number,

int square,

int floor,

int roomCount,

String street,

House.*BuildingTypes* buildingType,

Date lifetime

) {

this.setId((int) Math.*random*());

this.setNumber(number);

this.setSquare(square);

this.setFloor(floor);

this.setRoomCount(roomCount);

this.setStreet(street);

this.setBuildingTtype(buildingTtype);

this.setLifetime(lifetime);

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getNumber() {

return number;

}

public void setNumber(int number) {

this.number = number;

}

public int getSquare() {

return square;

}

public void setSquare(int square) {

this.square = square;

}

public int getFloor() {

return floor;

}

public void setFloor(int floor) {

this.floor = floor;

}

public int getRoomCount() {

return roomCount;

}

public void setRoomCount(int roomCount) {

this.roomCount = roomCount;

}

public String getStreet() {

return street;

}

public void setStreet(String street) {

this.street = street;

}

public House.*BuildingTypes* getBuildingTtype() {

return buildingTtype;

}

public void setBuildingTtype(House.*BuildingTypes* buildingTtype) {

this.buildingTtype = buildingTtype;

}

public Date getLifetime() {

return lifetime;

}

public void setLifetime(Date lifetime) {

this.lifetime = lifetime;

}

public String toString() {

return "number: " + Integer.*toString*(this.getNumber()) +

"\nsquare: " + Integer.*toString*(this.getSquare()) +

"\nfloor: " + Integer.*toString*(this.getFloor()) +

"\nrooms count: " + Integer.*toString*(this.getRoomCount()) +

"\nstreet: " + this.getStreet() +

"\nbuilding type: " + this.getBuildingTtype() +

"\nlifetime: " + this.getLifetime() + "\n";

}

public static House[] createFactory(int count) {

House[] houses = new House[count];

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

houses[i] = new House();

}

return houses;

}

public boolean filterByRoomCount(int roomCount) {

return this.getRoomCount() == roomCount;

}

public boolean filterByFloor(int low, int high) {

return this.getFloor() >= low && this.getFloor() <= high;

}

public boolean filterBySquare(int square) {

return this.getSquare() > square;

}

}

}

**Вывод программы**

Список квартир

number: 861

square: 62

floor: 20

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 1000

square: 133

floor: 10

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 563

square: 96

floor: 19

rooms count: 1

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 505

square: 73

floor: 4

rooms count: 1

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 697

square: 190

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 270

square: 194

floor: 12

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 151

square: 39

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 941

square: 175

floor: 13

rooms count: 2

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 876

square: 134

floor: 13

rooms count: 1

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 805

square: 136

floor: 15

rooms count: 2

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

Список квартир , имеющих заданное число комнат

number: 861

square: 62

floor: 20

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 1000

square: 133

floor: 10

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 697

square: 190

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 270

square: 194

floor: 12

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 151

square: 39

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

Список квартир , имеющих заданное число комнат и этаж в диапазоне

number: 1000

square: 133

floor: 10

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 697

square: 190

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 270

square: 194

floor: 12

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 151

square: 39

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

Список квартир , имеющих площадь, превосходящую заданную

number: 1000

square: 133

floor: 10

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 697

square: 190

floor: 2

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 270

square: 194

floor: 12

rooms count: 5

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 941

square: 175

floor: 13

rooms count: 2

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 876

square: 134

floor: 13

rooms count: 1

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

number: 805

square: 136

floor: 15

rooms count: 2

street: Street

building type: Living

lifetime: Sun May 31 13:33:15 MSK 2020

**Задание 3:**

**Вариант 6:** Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.

**Код программы:**

**Файл Petal.java**

package Rose;

public class Petal {

public static enum *Colors* {

***Red***,

***White***,

}

public *Colors* color = Petal.*Colors*.***Red***;

public boolean alive = false;

public Petal() {}

public Petal(Petal.*Colors* color) {

this.color = color;

}

public Petal(Petal.*Colors* color, boolean alive) {

this.color = color;

this.alive = alive;

}

public void bloom() {

this.alive = true;

}

public void wilt() {

this.alive = false;

}

*@Override*

public String toString() {

return "color: " + this.color +

"\nalive: " + this.alive + "\n";

}

*@Override*

public boolean equals(Object petal) {

return this.color == ((Petal) petal).color && this.alive == ((Petal) petal).alive;

}

*@Override*

public int hashCode() {

return this.color.hashCode() + (this.alive ? 1 : 0);

}

}

**Файл Bud.java**

package Rose;

public class Bud {

public Petal[] petals;

public Bud(int petalsCount) {

this.petals = new Petal[petalsCount];

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

this.petals[i] = new Petal();

}

}

public Bud(int petalsCount, Petal.*Colors* color) {

this.petals = new Petal[petalsCount];

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

this.petals[i] = new Petal(color);

}

}

public void bloom() {

for (int i = 0; i < this.petals.length; i++) {

this.petals[i].bloom();

}

}

public void wilt() {

for (int i = 0; i < this.petals.length; i++) {

this.petals[i].wilt();

}

}

*@Override*

public String toString() {

String result = "";

for (int i = 0; i < this.petals.length; i++) {

result += this.petals[i].toString();

}

return result;

}

*@Override*

public boolean equals(Object o) {

Bud bud = (Bud) o;

boolean isEqualCount = this.petals.length == bud.petals.length;

if (!isEqualCount) {

return false;

}

boolean isEqual = true;

for (int i = 0; (i < this.petals.length) && isEqual; i++) {

isEqual = this.petals[i].equals(bud.petals[i]);

}

return isEqual;

}

*@Override*

public int hashCode() {

int result = 0;

for (int i = 0; i < this.petals.length; i++) {

result += this.petals[i].hashCode();

}

return result;

}

}

**Файл Rose.java:**

package Rose;

public class Rose extends Bud {

public Rose(int petalsCount) {

super(petalsCount);

}

public Rose(int petalsCount, Petal.*Colors* color) {

super(petalsCount, color);

}

}

**Файл Main.java:**

package Rose;

import Rose.Rose;

public class Main {

public static void main(String[] args) {

int roseCount = 5;

Rose rose = new Rose(roseCount);

System.***out***.println(rose.toString());

rose.bloom();

System.***out***.println(rose.toString());

rose.wilt();

System.***out***.println(rose.toString());

Rose redRose = new Rose(roseCount, Petal.*Colors*.***Red***);

System.***out***.println(rose.equals(redRose));

System.***out***.println(redRose.hashCode());

}

}

**Задание 4:**

**Вариант 1:** Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

**Код программы:**

Файл Person.java:

package Optional;

public class Person {

public String name;

public String surname;

public Person(String name, String surname) {

this.name = name;

this.surname = surname;

}

}

Файл Student.java:

package Optional;

public class Student extends Person {

public Student(String name, String surname) {

super(name, surname);

}

public void goToCourse(Course course) {

course.addStudent(this);

}

}

Файл Teacher.java:

package Optional;

public class Teacher extends Person {

public Teacher(String name, String surname) {

super(name, surname);

}

public void startCourse(Course course) {

course.teacher = this;

}

public Mark createMark(int mark, Student student, Course course) {

return new Mark(mark, student, course);

}

}

Файл Course.java:

package Optional;

import java.util.ArrayList;

public class Course {

public String name;

public ArrayList<Student> students = new ArrayList<Student>();

public Teacher teacher;

public Course(String name) {

this.name = name;

}

public void addStudent(Student student) {

this.students.add(student);

}

}

Файл Mark.java:

package Optional;

public class Mark {

public int mark;

public Student student;

public Course course;

public Mark(int mark, Student student, Course course) {

this.mark = mark;

this.student = student;

this.course = course;

}

}

Файл Optional.java:

package Optional;

import java.util.ArrayList;

public class Optional {

public ArrayList<Course> courses = new ArrayList<Course>();

public ArrayList<Student> students = new ArrayList<Student>();

public ArrayList<Teacher> teachers = new ArrayList<Teacher>();

public Archive archive = new Archive();

public void addCourse(Course course) {

this.courses.add(course);

}

public void addTeacher(Teacher teacher) {

this.teachers.add(teacher);

}

public void addStudent(Student student) {

this.students.add(student);

}

public void startCourse(Teacher teacher, Course course) {

teacher.startCourse(course);

}

public void goToCourse(Student student, Course course) {

student.goToCourse(course);

}

public void putMark(Teacher teacher, Student student, Course course, int mark) {

Mark newMark = teacher.createMark(mark, student, course);

this.archive.addMark(newMark);

}

}

Файл Archive.java:

package Optional;

import java.util.ArrayList;

public class Archive {

public ArrayList<Mark> marks = new ArrayList<Mark>();

public void addMark(Mark mark) {

this.marks.add(mark);

}

}

Файл Main.java:

package Optional;

public class Main {

public static void main(String[] args) {

Optional optional = new Optional();

optional.addTeacher(new Teacher("Иван", "Иванов"));

optional.addStudent(new Student("Петр", "Петров"));

optional.addCourse(new Course("Физика"));

optional.startCourse(optional.teachers.get(0), optional.courses.get(0));

optional.goToCourse(optional.students.get(0), optional.courses.get(0));

optional.putMark(optional.teachers.get(0), optional.students.get(0), optional.courses.get(0), 5);

}

}

**Вывод:** Программы работают корректно.