|  |  |
| --- | --- |
| 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 №** | 3 |

**Название:** Классы, наследование, полиморфизм.

**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

Определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

# Текст программы

class Quadratic {

private double a;

private double b;

private double c;

public Quadratic(double a, double b, double c) {

//checkQuadratic(a);

this.a = a;

this.b = b;

this.c = c;

}

public double getA() {

return a;

}

public void setA(double a) {

this.a = a;

}

public double getB() {

return b;

}

public void setB(double b) {

this.b = b;

}

public double getC() {

return c;

}

public void setC(double c) {

this.c = c;

}

public double getDiscriminant() {

return b \* b - 4 \* a \* c;

}

public double[] getX() {

double d = getDiscriminant();

double[] x;

if (d > 0) {

x = new double[2];

x[0] = (-b + Math.sqrt(d)) / (2 \* a);

x[1] = (-b - Math.sqrt(d)) / (2 \* a);

return x;

} else if (d == 0) {

x = new double[1];

x[0] = -b / (2 \* a);

return x;

} else {

x = new double[0];

}

return x;

}

public double getExtremum() {

return -b / (2 \* a);

}

public double[] getDecreasingInterval() {

double[] interval = new double[2];

if (a > 0) {

interval[0] = Double.NEGATIVE\_INFINITY;

interval[1] = getExtremum();

} else {

interval[0] = getExtremum();

interval[1] = Double.POSITIVE\_INFINITY;

}

return interval;

}

public double[] getIncreasingInterval() {

double[] interval = new double[2];

if (a > 0) {

interval[0] = getExtremum();

interval[1] = Double.POSITIVE\_INFINITY;

} else {

interval[0] = Double.NEGATIVE\_INFINITY;

interval[1] = getExtremum();

}

return interval;

}

public double getMaxRoot(double[] roots) throws Exception {

if (roots.length == 2) {

double maxroot = roots[0];

if (roots[1] > maxroot)

maxroot = roots[1];

return maxroot;

}

else if (roots.length == 1)

return roots[0];

else

throw new Exception("Корней нет");

}

public double getMinRoot(double[] roots) throws Exception {

if (roots.length == 2) {

double minroot = roots[0];

if (roots[1] < minroot)

minroot = roots[1];

return minroot;

}

else if (roots.length == 1)

return roots[0];

else

throw new Exception("Корней нет");

}

}

public class Main {

public static void main(String[] args) {

List<Quadratic> quadratics = new ArrayList<>();

List<Double> maxroots = new ArrayList<Double>();

List<Double> minroots = new ArrayList<Double>();

//double maxx = first.getMaxRoot(first.getX());

quadratics.add(new Quadratic(2., 1., 5.));

quadratics.add(new Quadratic(1., 10.5, 3.));

quadratics.add(new Quadratic(2.5, 20., 3.));

quadratics.add(new Quadratic(1., 0, 0));

quadratics.add(new Quadratic(1., 0, -5.));

//double maxx = quadratics.get(0).getMaxRoot(quadratics.get(0).getX);

int narr = 0;

for (Quadratic quadratic : quadratics) {

System.out.println("y = " + quadratic.getA() + "x^2 + (" + quadratic.getB() + ")x + (" + quadratic.getC() + ")");

System.out.print("Discriminant = ");

System.out.println(quadratic.getDiscriminant());

double[] roots = new double[2];

roots = quadratic.getX();

System.out.print("Root(s) = ");

System.out.println(Arrays.toString(roots));

try {

double max = quadratic.getMaxRoot(roots);

double min = quadratic.getMinRoot(roots);

narr += 1;

maxroots.add(max);

minroots.add(min);

}

catch (Exception e) {

System.out.println(e.getMessage());

}

//double max = quadratic.getMaxRoot(roots);

//System.out.println(max);

System.out.print("Decreasing interval = ");

System.out.println(Arrays.toString(quadratic.getDecreasingInterval()));

System.out.print("Increasing interval = ");

System.out.println(Arrays.toString(quadratic.getIncreasingInterval()));

System.out.println();

}

System.out.println(narr);

System.out.println(maxroots);

System.out.println(minroots);

if (maxroots.isEmpty()) {

System.out.println("Max root = None");

}

else {

double mx = maxroots.get(0);

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

if (maxroots.get(i) > mx)

mx = maxroots.get(i);

}

System.out.println("Max root = " + mx);

}

if (minroots.isEmpty()) {

System.out.println("Min root = None");

}

else {

double mn = minroots.get(0);

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

if (minroots.get(i) < mn)

mn = minroots.get(i);

}

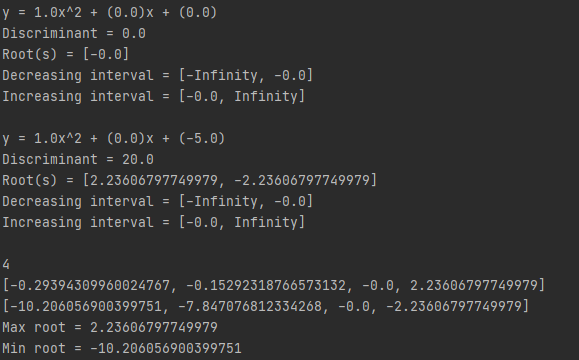
System.out.println("Min root = " + mn);

}

}

}

# Результат



**Вариант 1**

**Задача 2**

# Задание

Определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности n из комплексных координат. Передать их в метод, который выполнит их сложение.

# Текст программы

class Complex {

double re;

double im;

public Complex(double re, double im) {

this.re = re;

this.im = im;

}

public Complex() {

re = 0;

im = 0;

}

public double getRe() {

return re;

}

public void setRe(double re) {

this.re = re;

}

public double getIm() {

return im;

}

public void setIm(double im) {

this.im = im;

}

@Override

public String toString() {

final StringBuilder sb = new StringBuilder("{");

sb.append("re=").append(re);

sb.append(", im=").append(im);

sb.append('}');

return sb.toString();

}

public Complex add(Complex z1) {

Complex z = new Complex();

z.re = re + z1.re;

z.im = im + z1.im;

return z;

}

public Complex mult(Complex z1) {

Complex z = new Complex();

z.re = re\*z1.re - im\*z1.im;

z.im = re\*z1.im + z1.re\*im;

return z;

}

}

class ComplexList {

List<Complex> complexList;

public ComplexList() {

complexList = new ArrayList<>();

}

public boolean remove(int k) {

boolean f;

if (f = isPossibleToRemove(k))

complexList.remove(k);

return f;

}

private boolean isPossibleToRemove(int k) {

return (k>=0 && k<complexList.size());

}

public void addNumber (Complex z) {

complexList.add(z);

}

public void addRandomNComplex(int n, double max) {

Random rand = new Random();

double re;

double im;

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

re = (rand.nextDouble()-0.5)\*2\*max;

im = (rand.nextDouble()-0.5)\*2\*max;

complexList.add(new Complex(re, im));

}

}

public void output() {

int i=0;

for (Complex z: complexList) {

System.out.println("Число " + ++i + ": " + z.toString());

}

}

public Complex summ() {

Complex res = new Complex();

for (Complex z: complexList) {

res = res.add(z);

}

return res;

}

public Complex mult() {

Complex res = new Complex();

int i = 0;

for (Complex z: complexList) {

if (i==0) res = z;

else res = res.mult(z);

i++;

}

return res;

}

}

public class Main {

public static void main(String[] args) {

Complex z1 = new Complex(2, -2);

Complex z2 = new Complex(1, 2);

Complex z3 = new Complex(3, -4);

ComplexList numbers = new ComplexList();

numbers.addNumber(z1);

numbers.addNumber(z2);

numbers.addNumber(z3);

numbers.addNumber(z2.add(z3));

numbers.addNumber(z2.mult(z3));

//Вывод комплексных чисел

numbers.output();

//Сумма вектора комплексных чисел

System.out.println(numbers.summ());

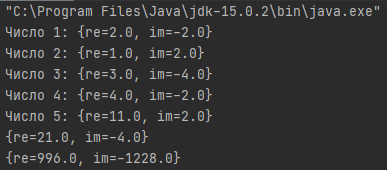
//Умножение вектора комплексных чисел

System.out.println(numbers.mult());

}

}

# Результат



**Вариант 2**

**Задача 1**

# Задание

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: a) список покупателей в алфавитном порядке; b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

# Текст программы

class Customer {

private String name;

private String name2;

private String name3;

private String address;

private int creditCardId;

private int bankNumberCard;

Customer(String name, String name2, String name3, String address, int creditCardId, int bankNumberCard) {

this.name = name;

this.name2 = name2;

this.name3 = name3;

this.address = address;

this.creditCardId = creditCardId;

this.bankNumberCard = bankNumberCard;

}

public String getName() {

return name;

}

public String getName2() {

return name2;

}

public String getName3() {

return name3;

}

public String getAddress() {

return address;

}

public int getCreditCardId() {

return creditCardId;

}

public int getBankNumberCard() {

return bankNumberCard;

}

public String toString() {

return String.format("ФИО: %s %s %s\t Страна: %s Номер карты: %d Номер банка: %d",

name2, name, name3, address, creditCardId, bankNumberCard);

}

}

class Shop {

private String shopName;

private ArrayList<Customer> customersList = new ArrayList<>();

Shop(String shopName) {

this.shopName = shopName;

}

void addCustomerToShopList(Customer customer) {

customersList.add(customer);

}

List<Customer> getListByName() {

List<Customer> list = new ArrayList<>(customersList);

Collections.sort(list, new Comparator<Customer>() {

@Override

public int compare(Customer o1, Customer o2) {

return o1.getName2().compareToIgnoreCase(o2.getName2());

}

});

return list;

}

List<Customer> getListByDiapasonCreaditCard(int diapazonStart, int diapazonEnd) {

List<Customer> list = new ArrayList<>();

for (Customer c : customersList) {

if (c.getCreditCardId() >= diapazonStart && c.getCreditCardId() <=diapazonEnd) {

list.add(c);

}

}

return list;

}

}

public class Main {

public static void main(String[] args) {

Shop shop1 = new Shop("Shop1");

shop1.addCustomerToShopList(new Customer("Емельян", "Парамедов", "Михайлович", "Украина", 20202020, 12345));

shop1.addCustomerToShopList(new Customer("Алексей", "Дмитриев", "Иванович", "Россия", 30303030, 23456));

shop1.addCustomerToShopList(new Customer("Станислав", "Даудов", "Иванович", "Италия", 40404040, 34567));

shop1.addCustomerToShopList(new Customer("Виктор", "Викторов", "Иванович", "Украина", 41414141, 45678));

shop1.addCustomerToShopList(new Customer("Ян", "Милов", "Иванович", "Латвия", 77777777, 56789));

shop1.addCustomerToShopList(new Customer("Диана", "Ылева", "Ивановна", "Россия", 99909090, 67890));

shop1.addCustomerToShopList(new Customer("Маргарита", "Ретузова", "Ивановна", "Украина", 10101010, 78901));

shop1.addCustomerToShopList(new Customer("Степан", "Яблоко", "Иванович", "Россия", 87879787, 89012));

shop1.addCustomerToShopList(new Customer("Стапан", "Даудов", "Иванович", "США", 61686868, 90123));

shop1.addCustomerToShopList(new Customer("Антон", "Антон", "Иванович", "Украина", 12121212, 10234));

System.out.println("\nСортировка по ФИО:");

List<Customer> listSortByName = shop1.getListByName();

for (Customer c : listSortByName) {

System.out.println(c);

}

System.out.println("\nКарты в диапозoне '20000000 - 70000000':");

List<Customer> listSortByDiapazonCreditCard = shop1.getListByDiapasonCreaditCard(20000000, 70000000);

for (Customer c : listSortByDiapazonCreditCard) {

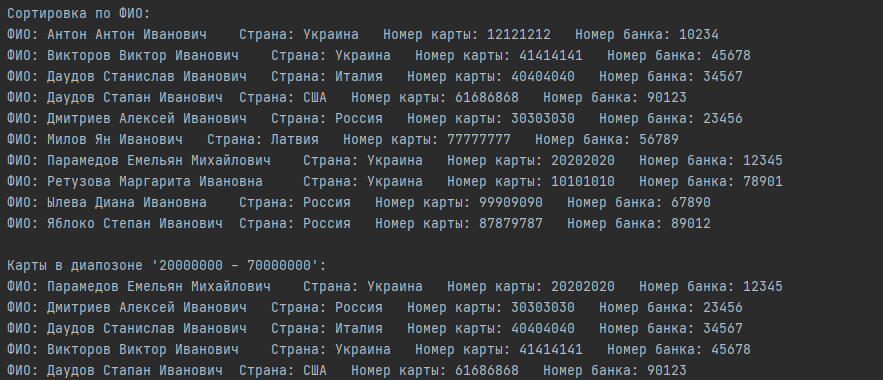
System.out.println(c);

}

}

}

# Результат



**Вариант 2**

**Задача 2**

# Задание

Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты у которых находится в заданном интервале..

# Текст программы

class Patient {

private String name;

private String name2;

private String name3;

private String address;

private int patientCardId;

private String diagnoz;

Patient(String name, String name2, String name3, String address, int patientCardId, String diagnoz) {

this.name = name;

this.name2 = name2;

this.name3 = name3;

this.address = address;

this.patientCardId = patientCardId;

this.diagnoz = diagnoz;

}

public String getName() {

return name;

}

public String getName2() {

return name2;

}

public String getName3() {

return name3;

}

public String getAddress() {

return address;

}

public int getCreditCardId() {

return patientCardId;

}

public String getDiagnoz() {

return diagnoz;

}

public String toString() {

return String.format("ФИО: %s %s %s\nАдрес: %s \nНомер карты: %d\nДиагноз: %s\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_",

name2, name, name3, address, patientCardId, diagnoz);

}

}

class Hospital {

private String hospitalName;

private ArrayList<Patient> patientsList = new ArrayList<>();

Hospital(String hospitalName) {

this.hospitalName = hospitalName;

}

void addPatientToHospitalList(Patient patient) {

patientsList.add(patient);

}

List<Patient> getListByDiag(String diag) {

List<Patient> list = new ArrayList<>();

for (Patient c : patientsList) {

if (c.getDiagnoz() == diag) {

list.add(c);

}

}

return list;

}

List<Patient> getListByDiapasonCreaditCard(int diapazonStart, int diapazonEnd) {

List<Patient> list = new ArrayList<>();

for (Patient c : patientsList) {

if (c.getCreditCardId() >= diapazonStart && c.getCreditCardId() <=diapazonEnd) {

list.add(c);

}

}

return list;

}

}

public class Main {

public static void main(String[] args) {

Hospital hospital1 = new Hospital("hospital1");

hospital1.addPatientToHospitalList(new Patient("Емельян", "Парамедов", "Михайлович", "Украина", 20202020, "Нервоз"));

hospital1.addPatientToHospitalList(new Patient("Алексей", "Дмитриев", "Иванович", "Россия", 30303030, "Грыжа"));

hospital1.addPatientToHospitalList(new Patient("Станислав", "Даудов", "Иванович", "Италия", 40404040, "Коронавирус"));

hospital1.addPatientToHospitalList(new Patient("Виктор", "Викторов", "Иванович", "Украина", 41414141, "Гепатит С"));

hospital1.addPatientToHospitalList(new Patient("Ян", "Милов", "Иванович", "Латвия", 77777777, "Перелом ребер"));

hospital1.addPatientToHospitalList(new Patient("Диана", "Ылева", "Ивановна", "Россия", 99909090, "Коронавирус"));

hospital1.addPatientToHospitalList(new Patient("Маргарита", "Ретузова", "Ивановна", "Украина", 10101010, "Бессонница"));

hospital1.addPatientToHospitalList(new Patient("Степан", "Яблоко", "Иванович", "Россия", 87879787, "ОРВИ"));

hospital1.addPatientToHospitalList(new Patient("Стапан", "Даудов", "Иванович", "США", 61686868, "Коронавирус"));

hospital1.addPatientToHospitalList(new Patient("Антон", "Антон", "Иванович", "Украина", 12121212, "ОРВИ"));

System.out.println("\nПациенты с коронавирусом:");

List<Patient> listSortByDiag = hospital1.getListByDiag("Коронавирус");

for (Patient c : listSortByDiag) {

System.out.println(c);

}

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

System.out.println("\nКарты в диапозoне '20000000 - 70000000':");

List<Patient> listSortByDiapazonCreditCard = hospital1.getListByDiapasonCreaditCard(20000000, 70000000);

for (Patient c : listSortByDiapazonCreditCard) {

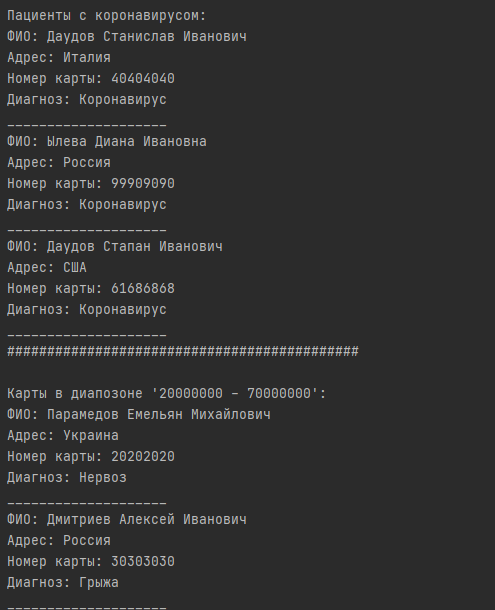
System.out.println(c);

}

}

}

# Результат



**Вариант 3**

**Задача 1**

# Задание

Создать объект класса Сутки, используя классы Час, Минута. Методы: вывести на консоль текущее время, рассчитать время суток (утро, день, вечер, ночь).

# Текст программы

class Minute {

private int intMin;

private String strMin;

//Empty constructor

public Minute () {

this.intMin = 0;

this.strMin = "0";

}

//Creates Word from given String and extends charLength of word on number of chars in String

public Minute (int intMin) {

this.intMin = intMin;

this.strMin = String.valueOf(intMin);

}

//Getter for body of Word

public int getBody() {

return intMin;

}

public String getBodyStr() {

return strMin;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

if (!super.equals(obj)) return false;

Minute min = (Minute) obj;

if (intMin == min.intMin%60) return true;

else return false;

}

@Override

public int hashCode() {

int result = intMin;

return result;

}

@Override

public String toString() {

if (intMin < 60 && intMin >= 0) return strMin + " мин";

else if (intMin > 0) return String.valueOf(intMin%60) + " мин";

else return "Отрицательное число мин";

}

}

class Hour {

private int intHour;

//private Minute intMin;

private String strHour;

//Empty constructor

public Hour () {

this.intHour = 0;

//this.intMin = new Minute(0);

this.strHour = "0";

}

//Creates Word from given String and extends charLength of word on number of chars in String

public Hour (int intHour) {

this.intHour = intHour;

//this.intMin = intMin;

this.strHour = String.valueOf(intHour);

}

//Getter for body of Word

public int getBody() {

return intHour;

}

public String getBodyStr() {

return strHour;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

if (!super.equals(obj)) return false;

Hour hour = (Hour) obj;

if (intHour == hour.intHour%24) return true;

else return false;

}

@Override

public int hashCode() {

int result = intHour\*60;

return result;

}

@Override

public String toString() {

if (intHour < 24 && intHour >= 0) return strHour + " час";

else if (intHour > 0) return String.valueOf(intHour%24) + " час";

else return "Отрицательное число час";

}

}

class Day {

private Minute m;

private Hour h;

private int rh;

//Empty constructor

public Day () {

this.h = new Hour(0);

this.m = new Minute(0);

this.rh = 0;

}

public Day (Hour h, Minute m) {

this.h = h;

this.m = m;

this.rh = (h.getBody() + m.getBody()/60)%24;

}

//Getters

public Hour getHour() {

return new Hour(rh);

}

public Minute getMinute() {

return m;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

if (!super.equals(obj)) return false;

Day day = (Day) obj;

if (h.equals(day.h) && m.equals(day.m) && h.getBody() == day.h.getBody() && m.getBody() == day.m.getBody()) return true;

else return false;

}

@Override

public int hashCode() {

int result = h.getBody()\*60 + m.getBody();

return result;

}

@Override

public String toString() {

int rm = m.getBody()%60;

String mins = String.valueOf(rm);

if (rm / 10 == 0 || rm == 0) mins = "0" + mins;

String hours = String.valueOf(rh);

if (m.getBody() < 0 || h.getBody() < 0) return "Введены Отрицательные Значенияя!!!!1";

else return "Время: " + hours + " : " + mins;

}

public String getTime() {

return this.toString();

}

public String dayTime() {

if (rh >= 23 || rh < 5) return "Ночь";

else if (rh >= 5 && rh < 11) return "Утро";

else if (rh >= 11 && rh < 17) return "День";

else return "Вечер";

}

}

public class Main {

public static void main(String[] args) {

Minute min = new Minute(62);

Hour hour = new Hour(18);

Day day = new Day(hour, min);

System.out.println(day.getTime());

System.out.println(day.dayTime());

}

}

}

# Результат



**Вариант 3**

**Задача 2**

# Задание

Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление..

# Текст программы

class Numb {

private int numBody;

private String strBody;

//Empty constructor

public Numb () {

this.numBody = 0;

this.strBody = "";

}

//Creates Word from given String and extends charLength of word on number of chars in String

public Numb (int numBody) {

this.numBody = numBody;

this.strBody = String.valueOf(numBody);

}

//Getter for body of Word

public int getBody() {

return numBody;

}

public String getBodyStr() {

return strBody;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

if (!super.equals(obj)) return false;

Numb number = (Numb) obj;

if (numBody == number.numBody) return true;

else return false;

}

@Override

public int hashCode() {

int result = numBody;

return result;

}

@Override

public String toString() {

return strBody;

}

}

class Fraction {

private Numb chisl;

private Numb znam;

//Empty constructor

public Fraction () {

this.chisl = new Numb(0);

this.znam = new Numb(1);

}

//Creates Word from given String and extends charLength of word on number of chars in String

public Fraction (Numb chisl, Numb znam) {

this.chisl = chisl;

this.znam = znam;

}

public Numb getChisl() {

return chisl;

}

public Numb getZnam() {

return znam;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

if (!super.equals(obj)) return false;

Fraction fraction = (Fraction) obj;

if (chisl.equals(fraction.chisl) && znam.equals(fraction.znam)) return true;

else return false;

}

@Override

public int hashCode() {

int result = chisl.hashCode() \* znam.hashCode();

return result;

}

//поиск НОД для упрощения дроби

public int gcd() {

int a = Math.abs(chisl.getBody());

int b = Math.abs(znam.getBody());

if (a == 0)

return b;

while (b != 0) {

if (a > b)

a = a - b;

else

b = b - a;

}

return a;

}

//поиск НОК для умножения дробей

public int lcm(int a, int b) {

return a / gcd() \* b;

}

@Override

public String toString() {

if (znam.getBody() == 0) {

if (chisl.getBody() > 0) return "[+ inf]";

else if (chisl.getBody() < 0) return "[- inf]";

else return "[0]";

}

if (chisl.getBody() == 0) return "[0]";

else

return "[" + chisl.getBody()/gcd() + " / " + znam.getBody()/gcd() + "]";

}

//Сложение

public Fraction summ(Fraction fraction) {

int a = fraction.getChisl().getBody();

int b = fraction.getZnam().getBody();

Numb zn = new Numb(lcm(znam.getBody(), b));

Numb ch = new Numb(chisl.getBody()\*b + a\*znam.getBody());

Fraction result = new Fraction(ch, zn);

return result;

}

//Вычитание

public Fraction dec(Fraction fraction) {

int a = fraction.getChisl().getBody();

int b = fraction.getZnam().getBody();

Numb zn = new Numb(lcm(znam.getBody(), b));

Numb ch = new Numb(chisl.getBody()\*b - a\*znam.getBody());

Fraction result = new Fraction(ch, zn);

return result;

}

//Умножение

public Fraction mult(Fraction fraction) {

int a = fraction.getChisl().getBody();

int b = fraction.getZnam().getBody();

Numb zn = new Numb(znam.getBody()\* b);

Numb ch = new Numb(chisl.getBody()\* a);

Fraction result = new Fraction(ch, zn);

return result;

}

//Деление

public Fraction del(Fraction fraction) {

int a = fraction.getChisl().getBody();

int b = fraction.getZnam().getBody();

Numb zn = new Numb(znam.getBody()\* a);

Numb ch = new Numb(chisl.getBody()\* b);

Fraction result = new Fraction(ch, zn);

return result;

}

}

public class Main {

public static void main(String[] args) {

Numb number11 = new Numb(-3);

Numb number12 = new Numb(5);

Fraction fraction1 = new Fraction(number11, number12);

System.out.println("Дробь 1: " + fraction1.toString());

Numb number21 = new Numb(4);

Numb number22 = new Numb(12);

Fraction fraction2 = new Fraction(number21, number22);

System.out.println("Дробь 2: " + fraction2.toString());

//Сложение дробей

Fraction summa = fraction1.summ(fraction2);

System.out.println("Сумма дробей: " + summa.toString());

//Вычитание дробей

Fraction decr = fraction1.dec(fraction2);

System.out.println("Разность дробей: " + decr.toString());

//Умножение дробей

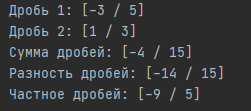
Fraction delen = fraction1.del(fraction2);

System.out.println("Частное дробей: " + delen.toString());

}

}}

# Результат



**Вариант 4**

**Задача 1**

# Задание

Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.

# Текст программы

//класс Бригадир, рабочий

class Brigada {

private double price; //Стоимость сотрудника

private int n; //Число бригадиров

public Brigada(double price, int n) {

this.price = price;

this.n = n;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getN() {

return n;

}

public void setN(double price) {

this.n = n;

}

public double getCost() {

return n\*price;

}

}

//класс Дом = ТЗ на дом

class House {

int etaz; //этажи

int meters; //кв. м. этажа

public House(int etaz, int meters) {

this.etaz = etaz;

this.meters = meters;

}

public int getEtaz() {

return etaz;

}

public void setEtaz(int etaz) {

this.etaz = etaz;

}

public int getMeter() {

return meters;

}

public void setMeter(int meters) {

this.meters = meters;

}

//расчет метража

public int getHouseMetrage() {

return etaz\*meters;

}

}

//класс Заказчик

class Client {

double money; //цена за кв. м.

House house; //ТЗ

public Client(double money, House house) {

this.money = money;

this.house = house;

}

public House getHouse() {

return house;

}

public void setHouse(House house) {

this.house = house;

}

public double getMoney() {

return money;

}

public void setMoney(double money) {

this.money = money;

}

}

//класс Конструктор

class Consructor {

Client client;

Brigada brig;

double coef = 0.5; // коэффициент оплаты рабочих

public Consructor(Client client) {

this.client = client;

this.brig = new Brigada(client.money \* coef, 5);

}

public Client getClient() {

return client;

}

public Brigada getBrigada() {

return brig;

}

public void setClient(Client client) {

this.client = client;

}

public double countCost() {

return client.house.getHouseMetrage()\*client.getMoney() + brig.getCost();

}

public String countCostString() {

double price1 = client.house.getHouseMetrage()\*client.getMoney();

double price2 = brig.getCost();

return "Цена дома: " + price1 + "$. Цена работ: " + price2 + "$";

}

}

public class Main {

public static void main(String[] args) {

House house = new House(1, 100); //Формирование ТЗ

Client client = new Client(3000, house); //Утверждение суммы выплаты за кв.м.

Consructor constr = new Consructor(client); //Расчёт стоимости работ

System.out.println(constr.countCostString()); //Отчет

}

}

}

# Результат



**Вариант 4**

**Задача 2**

# Задание

Система Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

# Текст программы

class Exam {

private int min; //Проходной балл

private int max; //Максимальный балл

private String name;

public Exam(String name, int min, int max) {

this.name = name;

this.min = min;

this.max = max;

}

public String getName() {

return name;

}

public int getMin() {

return min;

}

public int getMax() {

return max;

}

}

class Exams {

private Exam ex1 = new Exam("Математика", 75, 100);

private Exam ex2 = new Exam("Русскмй язык", 60, 100);

private Exam ex3 = new Exam("Обществознание", 50, 100);

private Exam ex4 = new Exam("Математика", 60, 100);

// Проходной балл за все экзамены

private int pBall = ex1.getMin() + ex2.getMin() + ex3.getMin() + ex4.getMin() + 50;

public Exam[] getExams() {

return new Exam[] {ex1, ex2, ex3, ex4};

}

public int getPBall() {

return pBall;

}

}

class Abiturient {

private int ex1ball;

private int ex2ball;

private int ex3ball;

private int ex4ball;

private String name;

public Abiturient(int ex1ball, int ex2ball, int ex3ball, int ex4ball, String name) {

this.ex1ball = ex1ball;

this.ex2ball = ex2ball;

this.ex3ball = ex3ball;

this.ex4ball = ex4ball;

this.name = name;

}

public int getEx1Ball() {

return ex1ball;

}

public int getEx2Ball() {

return ex2ball;

}

public int getEx3Ball() {

return ex3ball;

}

public int getEx4Ball() {

return ex4ball;

}

public String getName() {

return name;

}

public double getBall() {

return (ex1ball + ex2ball + ex3ball + ex4ball)/4;

}

public String toString() {

return name + ": " + ex1ball + ", " + ex2ball + ", " + ex3ball + ", " + ex4ball;

}

}

class Prepod {

String abiName; //абитуриент

int[] marks = new int[4]; //оценка

public Prepod(String abiName) {

this.abiName = abiName;

}

public Abiturient getMarks() {

for (int i = 0; i < 4; i++) {

int min = 33;

int max = 100;

int diff = max - min;

Random random = new Random();

marks[i] = random.nextInt(diff + 1) + min;

}

return new Abiturient(marks[0], marks[1], marks[2], marks[3], abiName);

}

}

public class Main {

public static void main(String[] args) {

Exams exs = new Exams();

Prepod pr = new Prepod("Иванов А.А.");

Abiturient ab = pr.getMarks();

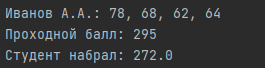
System.out.println(ab.toString());

System.out.println("Проходной балл: " + exs.getPBall() + "\nСтудент набрал: " + ab.getBall()\*4);

}

}

# Результат



**Ссылка на репозиторий**

<https://github.com/bmstu-iu6-21m/pract3-alpolyakov>

# Вывод

Был получен опыт работы с классами, получены навыки применения наследования, полиморфизма и переопределения методов при написании программ на языке Java.