Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «12» октября 2022 г. по «01» ноября 2022 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 19-2к  Берестенников Дмитрий Михайлович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2022

Содержание

Введение…………………………………………………………………………...3

Массивы……………………………………………………………………………3

1.Раздел…………………………………………………………………………….4

1.1 Задача 1………………………………………………………………………...4

1.2 Задача 2………………………………………………………………………...7

1.3 Задача 3……………………………………………………………………….12

1.4 Задача 4……………………………………………………………………….16

1.5 Задача 5……………………………………………………………………….22

1.6 Задача 6……………………………………………………………………….25

2.Раздел

2.1 Задача 1………………………………………………………………….........32

2.2 Диаграмма……………………………………………………………............33

2.3 Тестирование…………………………………………………………............40

Список использованных источников…………………………………………...42

Приложения

Введение

Массивы

Массив в Java представляет собой класс, при этом имя объекта класса массива является объектной ссылкой на динамическую память, в которой хранятся элементы массива. Элементами массива, в свою очередь, могут быть значения базового типа или объекты. Элементы массива проиндексированы, индексирование элементов начинается с нуля. Для объявления ссылки на массив можно записать пустые квадратные скобки после имени типа, например: int a[]. Аналогичный результат получится при записи int []a. Массивы в языке Java являются динамическими. Существует два способа создания массива: с помощью оператора new или с помощью прямой инициализации присваиванием значений элементам массива в фигурных скобках. Значения элементов неинициализированного массива, для которого выделена память, устанавливаются в значения по умолчанию для массива базового типа или null для массива объектных ссылок.

1 Основной раздел

1.1 Задача 1

Условие:

Найти корни квадратного уравнения. Параметры уравнения передавать с командной строкой.

Осуществить проверку корректности ввода чисел.

Блок-схема:

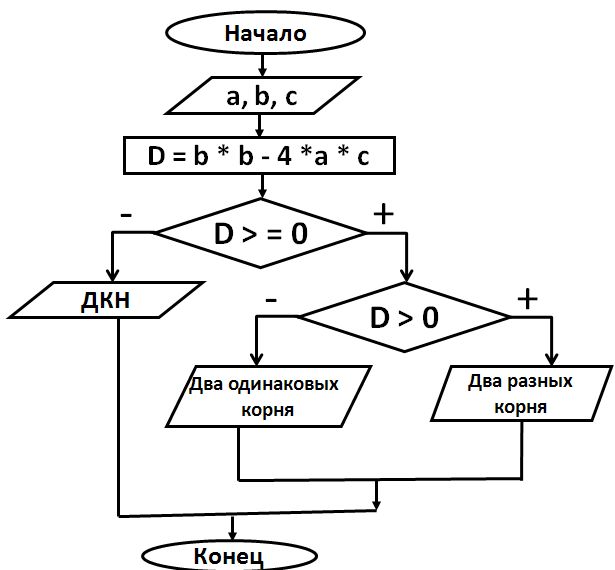


Рисунок 1.1 – Блок схема

Код программы:

package quadraticequation;  
import java.util.Scanner;  
public class Equation {  
  
 public static void main(String[] args) {  
  
 double a, b, c;  
  
 double D;  
  
 System.out.println("Программа решает квадратное уравнение вида:");  
  
 System.out.println("ax^2 + bx + c = 0");  
  
 System.out.println("Введите a, b и c:");  
  
 Scanner in = new Scanner(System.in);  
  
 a = in.nextDouble();  
  
 b = in.nextDouble();  
  
 c = in.nextDouble();  
  
 D = b \* b - 4 \* a \* c;  
  
 if (D > 0) {  
  
 double x1, x2;  
  
 x1 = (-b - Math.sqrt(D)) / (2 \* a);  
  
 x2 = (-b + Math.sqrt(D)) / (2 \* a);  
  
 System.out.println("Корни уравнения: x1 = " + x1 + ", x2 = " + x2);  
  
 }  
  
 else if (D == 0) {  
  
 double x;  
  
 x = -b / (2 \* a);  
  
 System.out.println("Уравнение имеет единственный корень: x = " + x);  
  
 }  
  
 else {  
  
 System.out.println("Уравнение не имеет действительных корней!");  
  
 }  
  
 }  
  
}

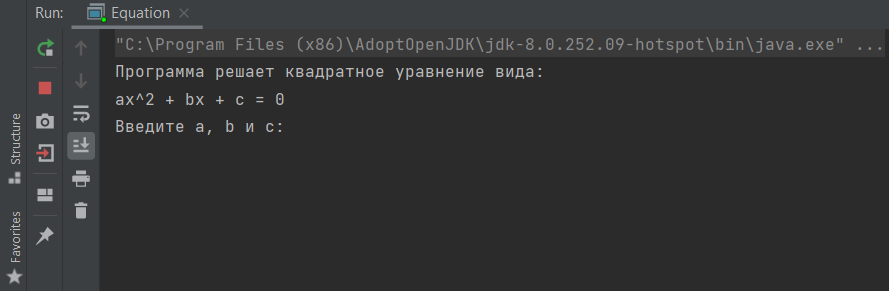


Рисунок 1 - Результат выполнения

1.2 Задача 2

Условие:

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно.

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 которых находится в заданном интервале.

Блок схема:

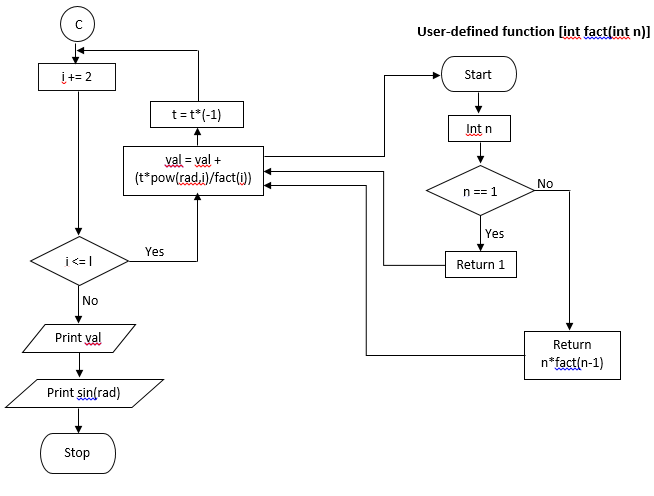


Рисунок 1.2 – Блок схема

Код программы:

package com.company;  
  
public class Main {  
  
 private int id;  
  
 private String name;  
  
 private String surname;  
  
 private String middle\_name;  
  
 private String address;  
  
 private String phone\_number;  
  
 private int med\_card;  
  
 private String diagnosis;  
  
 public Main(){  
 this.id = 0;  
 this.name = "";  
 this.surname = "";  
 this.middle\_name = "";  
 this.address = "";  
 this.med\_card = 0;  
 this.phone\_number = "";  
 this.diagnosis = "";  
 }  
  
 public Main(int id, String name, String middle\_name, String surname, String address, String phone\_number, int med\_card, String diagnosis) {  
 this.id = id;  
 this.name = name;  
 this.surname = surname;  
 this.middle\_name = middle\_name;  
 this.address = address;  
 this.med\_card = med\_card;  
 this.phone\_number = phone\_number;  
 this.diagnosis = diagnosis;  
 }  
  
 void setId(int id){  
 this.id = id;  
 }  
  
 int getId(){  
 return id;  
 }  
  
 void setName(String name){  
 this.name =name;  
 }  
  
 String getName(){  
 return name;  
 }  
  
 void setSurname(String surname){  
 this.surname = surname;  
 }  
  
 String getSurname(){  
 return surname;  
 }  
  
 void setMiddle\_name(String middle\_name){  
 this.middle\_name = middle\_name;  
 }  
  
 String getMiddle\_name(){  
 return middle\_name;  
 }  
  
 void setAddress(String address){  
 this.address =address;  
 }  
  
 String getAddress(){  
 return address;  
 }  
  
 void setPhone\_number(String phone\_number){  
 this.phone\_number =phone\_number;  
 }  
  
 String getPhone\_number(){  
 return phone\_number;  
 }  
  
 void setMed\_card(int med\_card){  
 if(med\_card>0) {  
 this.med\_card = med\_card;  
 } else System.out.println("Номер медицинской карты не может быть отрицательным или нулевым!");  
 }  
  
 int getMed\_card(){  
 return med\_card;  
 }  
  
 void setDiagnosis(String diagnosis){  
 this.diagnosis =diagnosis;  
 }  
  
 String getDiagnosis(){  
 return diagnosis;  
 }  
  
 @Override  
 public String toString() {  
 return "Пациент --- " +  
 "id=" + id +  
 ", Фамилия='" + surname +  
 ", Имя=" + name +  
 ", Отчество=" + middle\_name +  
 ", Адрес='" + address +  
 ", Номер телефона='" + phone\_number +  
 ", Номер медицинской карты=" + med\_card +  
 ", Диагноз='" + diagnosis;  
 }  
}

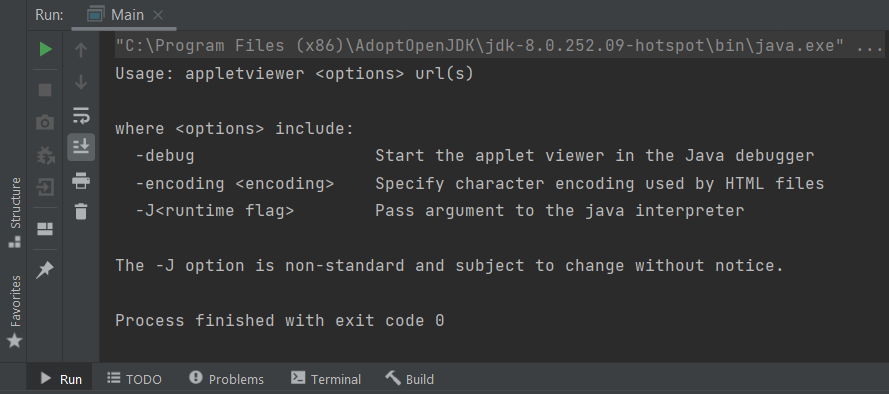


Рисунок 2 - Результат выполнения

1.3 Задача 3

Условие:

Создать объект класса Самолет, используя классы Крыло, Шасси, Двигатель. Методы: летать, задавать маршрут, вывести на консоль маршрут.

Блок схема:

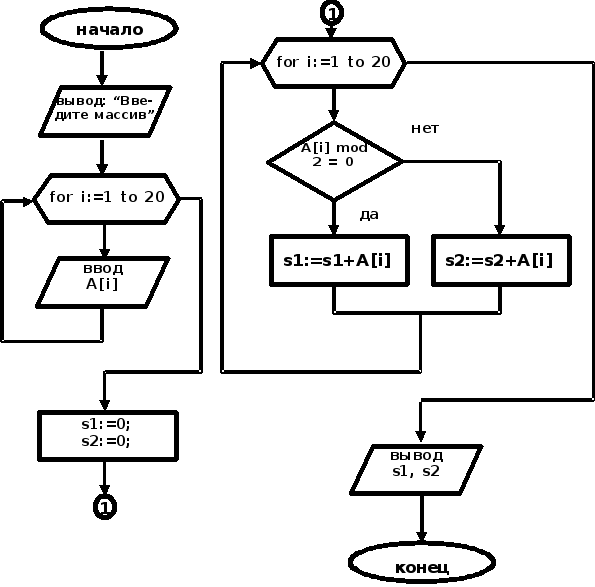


Рисунок 1.3 – Блок схема

Код программы:

Aircraft aircraft = new Aircraft("МС-21");

aircraft.setEngine(new Engine("/nДвигатель: Турбовентиляторный двигатель 870 км/ч"));

aircraft.setWing(new Wing("/nКрыло: из полимерных композитных материалов, размах 36 м"));

aircraft.setChassis(new Chassis("/nШасси: обычные!"));

aircraft.getInfo();

public class Aircraft {

    private Engine engine = null;

    private Chassis chassis = null;

    private Wing wing = null;

    private String name;

    public Aircraft(String name, Wing wing, Chassis chassis, Engine engine) {

        this.wing = wing;

        this.chassis = chassis;

        this.engine = engine;

        this.name = name;

    }

    public void setWing(Wing wing){

        this.wing = wing;

    }

    public void setChassis(Chassis chassis){

        this.chassis = chassis;

    }

    public void setEngine(Engine engine){

        this.engine = engine;

    }

    public String getInfo(){

        return this.getInfo() +

                this.engine.getInfo() +

                this.chassis.getInfo() +

                this.wing.getInfo();

    }

}

public class Engine{

    private String name = "";

    public Engine(String name) {

        this.name = name;

    }

    public String getInfo() {

        return this.name;

    }

}

public class Chassis{

    private String name = "";

    public Chassis(String name) {

        this.name = name;

    }

    public String getInfo() {

        return this.name;

    }

}

public class Wing{

    private String name = "";

    public Wing(String name) {

        this.name = name;

    }

    public String getInfo() {

        return this.name;

    }

}

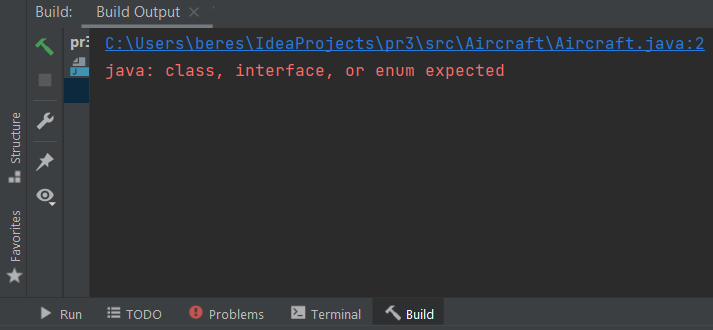


Рисунок 3 - Результат выполнения

1.4 Задача 4

Условие:

Создать класс Account с внутренним классом, с помощью объектов которого можно хранить информацию обо всех операциях со счетом (снятие, платежи, поступления).

Блок схема:

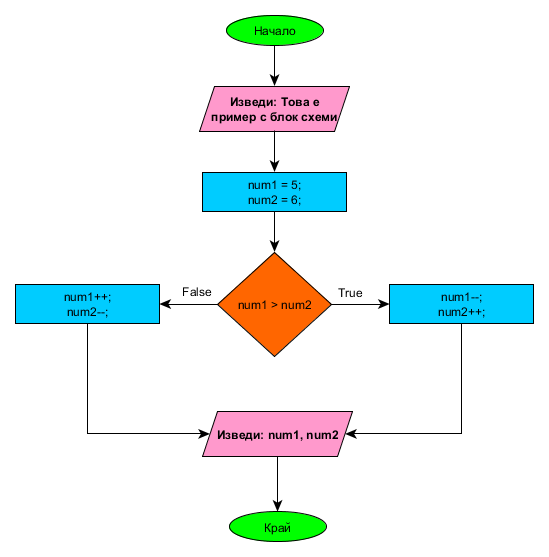


Рисунок 1.4 – Блок схема

Код программы:

package com.company.Main;  
  
 import java.util.ArrayList;  
 import java.util.Date;  
 import java.util.List;  
  
public class Bank {  
  
  
 public static void main(String[] args) {  
  
 Account accaunt = new Account();  
  
 accaunt.newTransaction(Account.typeTransaction.RECEIPT, 2000);  
 accaunt.newTransaction(Account.typeTransaction.PAYMENT, 100);  
 accaunt.newTransaction(Account.typeTransaction.WITHDRAWAL, 1500);  
  
 System.out.println(accaunt.print());  
  
 }  
  
}  
  
  
  
class Account {  
  
 enum typeTransaction {  
 WITHDRAWAL, PAYMENT, RECEIPT  
 }  
  
 int accountBalance = 0;  
 int nextId = 0;  
  
 List<Transaction> history = new ArrayList<>();  
  
 abstract class Transaction {  
 int id = 0;  
 int amount = 0;  
 int balance = 0;  
 Date date = new Date();  
 typeTransaction type;  
  
 public Transaction() {  
 this.id = nextId++;  
 }  
  
 @Override  
 public String toString() {  
 String string;  
 string = '\n' + "Transaction ID: " + this.id + '\n';  
 string += "Type: " + this.type + '\n';  
 string += "Date: " + this.date + '\n';  
 string += "Amount: " + this.amount + '\n';  
 string += "Balance after operation: " + this.balance + '\n';  
 return string;  
 }  
  
 }  
  
 class Withdrawal extends Transaction {  
  
 public Withdrawal(int amount) {  
 this.balance = accountBalance -= amount;  
 this.amount = amount;  
 }  
  
  
 }  
  
 class Payment extends Transaction {  
  
 public Payment(int amount) {  
 this.balance = accountBalance -= amount;  
 this.amount = amount;  
  
 }  
  
  
 }  
  
 class Receipt extends Transaction {  
  
 public Receipt(int amount) {  
 this.balance = accountBalance += amount;  
 this.amount = amount;  
 }  
  
  
 }  
  
 public void newTransaction(typeTransaction type, int amount)  
 throws UnsupportedOperationException {  
  
 Transaction newTransaction;  
  
 switch (type) {  
 case WITHDRAWAL:  
 newTransaction = new Withdrawal(amount);  
 break;  
  
 case PAYMENT:  
 newTransaction = new Payment(amount);  
 break;  
  
 case RECEIPT:  
 newTransaction = new Receipt(amount);  
 break;  
  
 default:  
 throw new UnsupportedOperationException("Not supported yet.");  
  
 }  
  
 newTransaction.type = type;  
 history.add(newTransaction);  
 }  
  
 public String print() {  
 String string = "";  
  
 for( Transaction t : history)  
 {  
 string += t.toString();  
 }  
  
  
 return string;  
 }

}

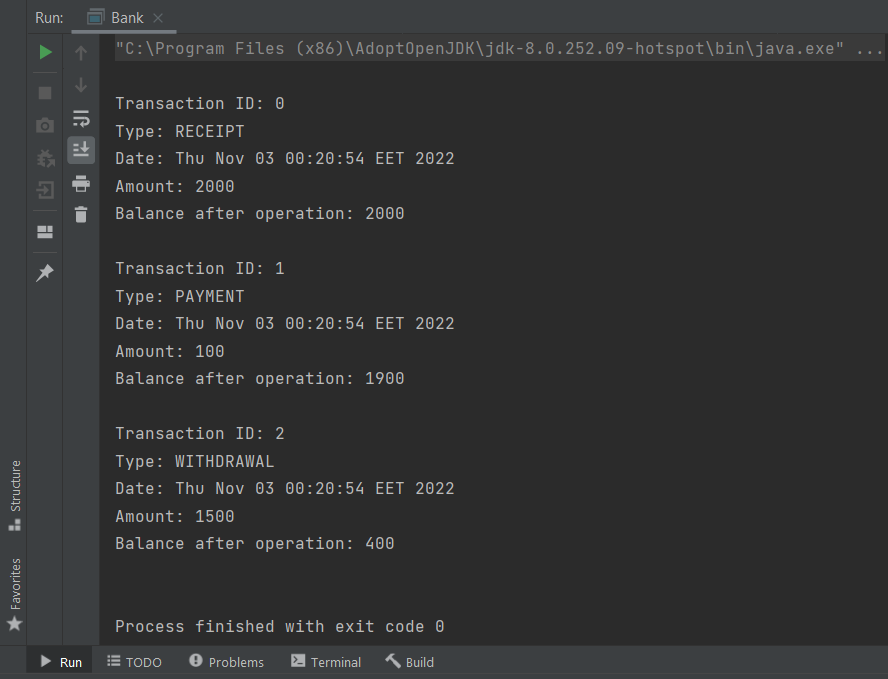


Рисунок 5 – Результат выполнения

1.5 Задача 5

Условие:

В тексте после буквы Р, если она не последняя в слове, ошибочно напечатана буква А вместо О. Внести исправления в текст.

Блок схема:

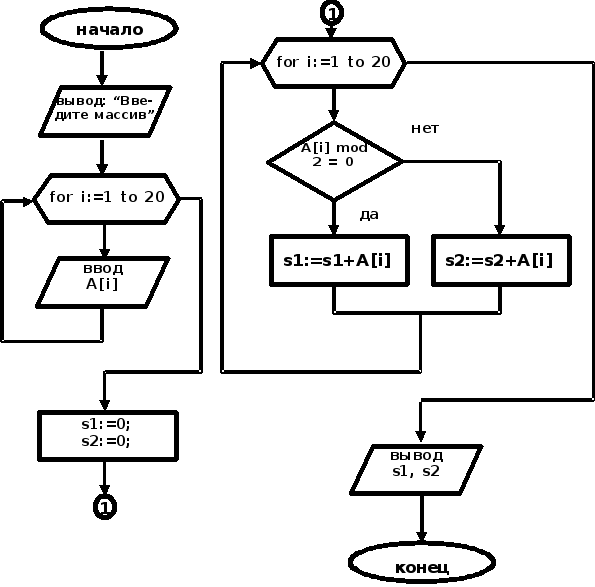


Рисунок 1.5 – Блок схема

Код программы:

package com.company;  
  
public void replace() {  
 for (int i = 0; i < wrongText.length(); i++) {  
 if (wrongText.charAt(i) == 'р') {  
 if (wrongText.lastIndexOf("р") == wrongText.length()-1) {  
 break;  
 } else {  
 rep(i);  
 }  
 } else if (wrongText.charAt(i) == 'Р') {  
 if (wrongText.lastIndexOf("р") == wrongText.length()-1) {  
 break;  
 } else {  
 rep(i);  
 }  
 }  
  
 }  
 System.out.println(wrongText);  
 }  
  
private void rep(int i) {  
 if (wrongText.charAt(i + 1) != ' ' && wrongText.charAt(i + 1) == 'а') {  
 wrongText = replaceCharAt(wrongText, i + 1, 'о');  
 } else if (wrongText.charAt(i + 1) != ' ' && wrongText.charAt(i + 1) == 'А') {  
 wrongText = replaceCharAt(wrongText, i + 1, 'О');  
 }  
 }  
  
  
public String replaceCharAt(String s, int pos, char c) {  
  
 return s.substring(0, pos) + c + s.substring(pos + 1);  
  
 }

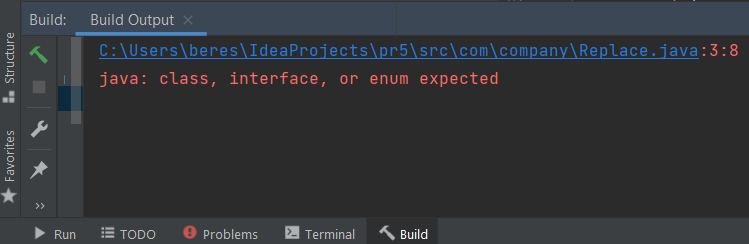


Рисунок 5 – Результат выполнения

1.6 Задача 6

Условие:

Вариант A

Выполнить задания на основе варианта А задачи 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Вариант B

Выполнить задания из варианта В задачи 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Блок схема:

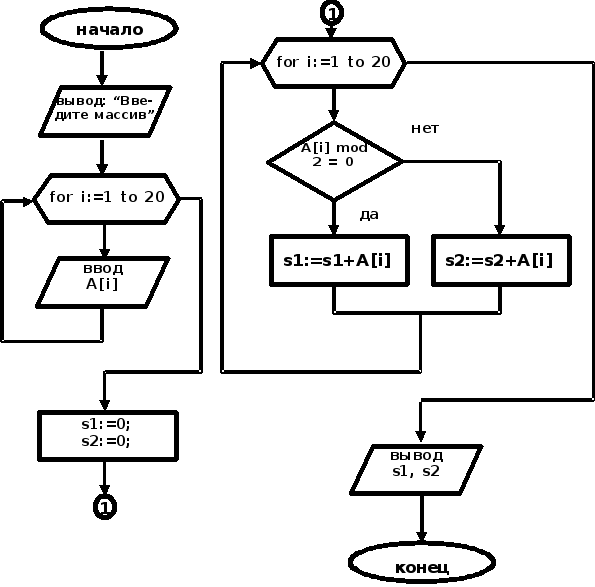


Рисунок 1.6 – Блок схема

Код программы:

package com.company;

import java.io.\*;

import java.util.\*;

public class Matrix {

    private double element[][];//Двумерный массив, чтобы хранить элементы матрицы

    private double q; //Чтобы хранить знак матрицы

    private int degree;//Хранит n(размер)

    public Matrix() {

        q=1;

        degree=1;

        element=new double[degree-1][degree-1];

        element[0][0]=0;

    }

    public Matrix(int n){

        q=1;

        degree=n;

        element=new double[degree][degree];

    }

    public Matrix(String name)throws IOException{

        q=1;

        Scanner in=new Scanner(new FileReader(name));

        degree=in.nextInt();

        element=new double[degree][degree];

        for(int i=0;i<degree;++i){

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

                element[i][j]=in.nextDouble();

            }

        }

        in.close();

    }

    public Matrix(Matrix B){

        q=B.q;

        degree=B.degree;

        element=new double [degree][degree];

        for(int i=0;i<degree;++i){

            System.arraycopy(B.element[i], 0, element[i], 0, degree);

        }

    }

    int getDegree(){

        return degree;

    } //Получить n(размер матрицы)

    public void show(){

        for(int i=0;i<degree;++i){

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

                System.out.format("%6.2f",element[i][j]);//Чтобы вывести на экран

            }

            System.out.println();

        }

    }

    Matrix Transpose(){

        Matrix result=new Matrix(this);

        for(int i=0;i<degree;++i){

            for(int j=i+1;j<degree;++j){

                double buf=result.element[i][j];

                result.element[i][j]=result.element[j][i];

                result.element[j][i]=buf;

            }

        }

        return result;

    }

    double getFirstNorm(){

        double nextS = 0;

        double max = -Double.MIN\_VALUE;

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

            nextS = 0;

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

                nextS += this.element[i][j];

            }

            if (max<nextS) max = nextS;

        }

        return max;

    }

    @Override

    public String toString(){

        Formatter f=new Formatter();

        for(int i=0;i<degree;++i){

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

                f.format("%6.2f", element[i][j]);

            }

            f.format("\n");

        }

        return f.toString();

    }

    public void set(String name)throws IOException{

        q=1;

        degree=0;

        Scanner in=new Scanner(new FileReader(name));

        degree=in.nextInt();

        element=new double[degree][degree];

        for(int i=0;i<degree;++i){

            element[i]=new double [degree];

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

                element[i][j]=in.nextDouble();

            }

        }

        in.close();

    }

    boolean Equals(Matrix B){

        if(degree!=B.degree)

            return false;

        for(int i=0;i<degree;++i){

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

                if(element[i][j]!=B.element[i][j])

                    return false;

            }

        }

        return true;

    }

    Matrix Multiply(Matrix B){          //Умножение

        Matrix result=new Matrix(degree);

        double s=0;

        for(int i=0;i<degree;++i){

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

                s=0;

                for(int k=0;k<degree;++k){

                    s+=element[i][k]\*B.element[k][j];

                }

                result.element[i][j]=s;

            }

        }

        return result;

    }

    Matrix Multiply(double n){       //умножение на число

        Matrix result=new Matrix(degree);

        for(int i=0;i<degree;++i){

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

                result.element[i][j]=element[i][j]\*n;

            }

        }

        return result;

    }

    Matrix Plus(Matrix B){                  //сложение

        Matrix result=new Matrix(degree);

        for(int i=0;i<degree;++i){

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

                result.element[i][j]=element[i][j]+B.element[i][j];

            }

        }

        return result;

    }

    Matrix Minus(Matrix B){                  //вычитание

        Matrix result=new Matrix(degree);

        for(int i=0;i<degree;++i){

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

                result.element[i][j]=element[i][j]-B.element[i][j];

            }

        }

        return result;

    }

}

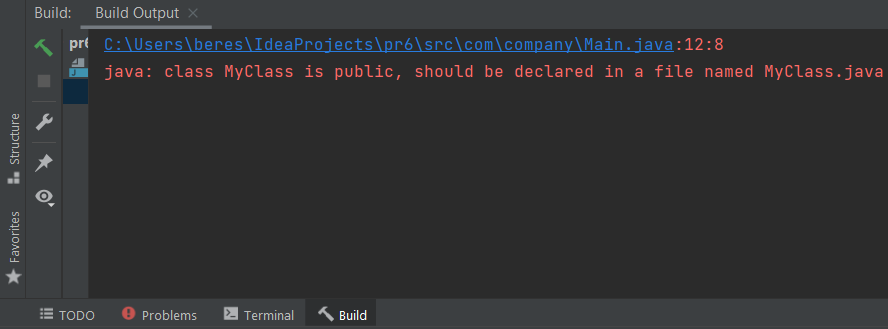


Рисунок 6 – Результат выполнения

2 Основной раздел

2.1 Задача 1

Условие:

В качестве серверной части разработать базу данных в СУБД MySQL в соответствии с вариантом. База данных содержит таблицу, состоящую из не менее чем 5-ти полей и 20-ти записей.

     Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

     Таким образом, структура клиента должна быть такой:

1. Класс, содержащий статический метод main() для демонстрации возможностей программы.
2. Класс, содержащий поля и методы для работы с удалённой базой данных:
3. Обязательные методы (запросы) клиента:

- выборка содержимого таблиц базы данных;

- запись в файл результата выборки содержимого таблиц БД.

- вывод в консоль результатов выполнения каждого метода.

1. Дополнительные методы (запросы) клиента
   1. добавление записи в таблицу.
   2. удаление записи из таблицы.
   3. обновление записи в таблице.
   4. поиск записи по признаку (на ваш выбор).

Схема диаграммы:

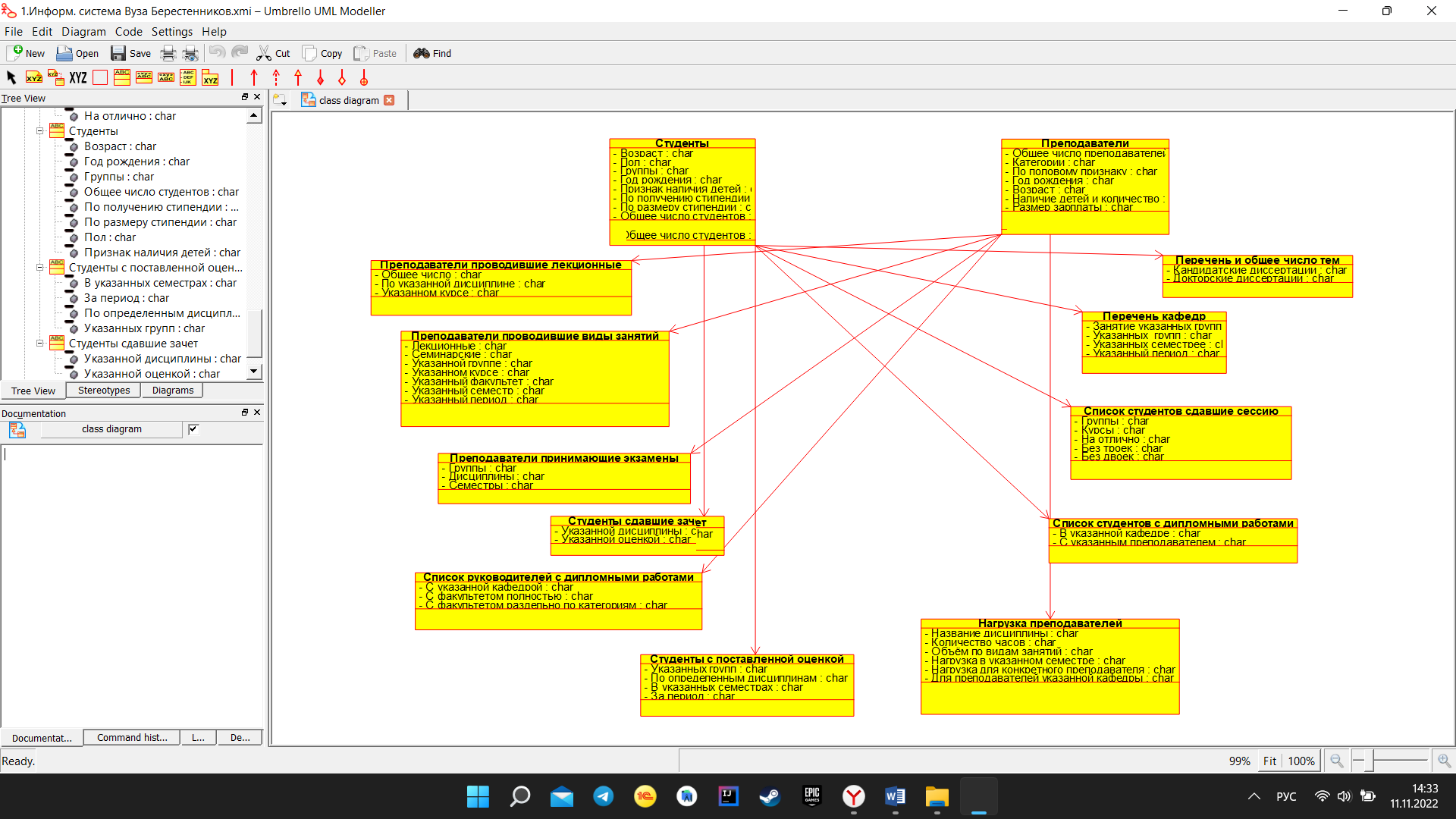


Рисунок 7 – Амбрелло Диаграмма «Студенты»

Код программы:

package com.company;  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.List;  
import com.company.Entity;  
public abstract class AbstractDAO <K, T extends Entity> {  
 public abstract List<T> findAll();  
 public abstract T findEntityById(K id);  
 public abstract boolean delete(K id);  
 public abstract boolean delete(T entity);  
 public abstract boolean create(T entity);  
 public abstract T update(T entity);  
}  
 public void close(Statement st) {  
 try {  
 if (st != null) {  
 st.close();  
 }  
 } catch (SQLException e) {  
// лог о невозможности закрытия Statement  
 }  
 }  
 public void close(Connection connection) {  
 try {  
 if (connection != null) {  
 connection.close();  
 }  
 } catch (SQLException e) {  
// генерация исключения, т.к. нарушается работа пула  
 }  
 }  
 package com.company.simpledao;  
 import java.sql.Connection;  
 import java.sql.PreparedStatement;  
 import java.sql.ResultSet;  
 import java.sql.SQLException;  
 import java.sql.Statement;  
 import java.util.\*;  
 import by.bsac.pool.ConnectionPool;  
 import by.bsac.subject.Abonent;  
public class AbonentDAO extends AbstractDAO <Integer, Abonent> {  
 public static final String SQL\_SELECT\_ALL\_ABONENTS = "SELECT \* FROM  
 phonebook";  
 public static final String SQL\_SELECT\_ABONENT\_BY\_LASTNAME =  
 "SELECT idphonebook,phone FROM phonebook WHERE lastname=?";  
 @Override  
 public List<Abonent> findAll() {  
 List<Abonent> abonents = new ArrayList<>();  
 Connection cn = null;  
 Statement st = null;  
 try {  
 cn = ConnectionPool.getConnection();  
 st = cn.createStatement();  
 ResultSet resultSet =  
 st.executeQuery(SQL\_SELECT\_ALL\_ABONENTS);  
 while (resultSet.next()) {  
 Abonent abonent = new Abonent();  
 abonent.setId(resultSet.getInt("idphonebook"));  
 abonent.setPhone(resultSet.getInt("phone"));  
 abonent.setLastName(resultSet.getString("lastname"));  
 abonents.add(abonent);  
 }  
 } catch (SQLException e) {  
 System.err.println("SQL exception (request or table failed): " + e);  
 } finally {  
 close(st);  
// код возвращения экземпляра Connection в пул  
 }  
 return abonents;  
 }  
 @Override  
 public Abonent findEntityById(Integer id) {  
 throw new UnsupportedOperationException();  
 }  
 @Override  
 public boolean delete(Integer id) {  
 throw new UnsupportedOperationException();  
 }  
 @Override  
 public Abonent create(Abonent entity) {  
 throw new UnsupportedOperationException();  
 }  
 @Override  
 public Abonent update(Abonent entity) {  
 throw new UnsupportedOperationException();  
 }  
 // собственный метод DAO  
 public Abonent findAbonentByLastName(String name) {  
 Abonent abonent = new Abonent();  
 Connection cn = null;  
 PreparedStatement st = null;  
 try {  
 cn = ConnectionPool.getConnection();  
 st =  
 cn.prepareStatement(SQL\_SELECT\_ABONENT\_BY\_LASTNAME);  
 st.setString(1, name);  
 ResultSet resultSet =st.executeQuery();  
 resultSet.next();  
 abonent.setId(resultSet.getInt("idphonebook"));  
 abonent.setPhone(resultSet.getInt("phone"));  
 abonent.setLastName(name);  
 } catch (SQLException e) {  
 System.err.println("SQL exception (request or table failed): " + e);  
 } finally {  
 close(st);  
// код возвращения экземпляра Connection в пул  
 }  
 return abonent;  
 }  
 @Override  
 public boolean delete(Abonent entity) {  
 throw new UnsupportedOperationException();  
 }  
}  
package com.company.pool;  
 import java.sql.Connection;  
 import java.sql.SQLException;  
 import javax.naming.Context;  
 import javax.naming.InitialContext;  
 import javax.naming.NamingException;  
 import javax.sql.DataSource;  
public class ConnectionPool {  
 private static final String DATASOURCE\_NAME = "jdbc/testphones";  
 private static DataSource dataSource;  
 static {  
 try {  
 Context initContext = new InitialContext();  
 Context envContext = (Context) initContext.lookup("java:/comp/env");  
 dataSource = (DataSource) envContext.lookup(DATASOURCE\_NAME);  
 } catch (NamingException e) {  
 e.printStackTrace();  
 }  
 }  
 private ConnectionPool() { }  
 public static Connection getConnection() throws SQLException {  
 Connection connection = dataSource.getConnection();  
 return connection;  
 }  
// метод возвращения Connection в пул  
}  
  
public class Student {  
 private String name;  
 private Integer personalNumber;  
 private String groupNumber;  
 private Integer course;  
 // класс с информацией о студенте  
 public Student(String name, Integer personalNumber, String groupNumber, int course) {  
 this.name = name;  
 this.personalNumber = personalNumber;  
 this.groupNumber = groupNumber;  
 this.course = course;  
 System.out.println("Created student " + getName() + " with personalId " +  
 getPersonalNumber() + ", course "  
 + getCourse() + ", group " + getGroupNumber());  
 }  
 public Student() {  
 System.out.println("New student has been created");  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getGroupNumber() {  
 return groupNumber;  
 }  
 public void setGroupNumber(String groupNumber) {  
 this.groupNumber = groupNumber;  
 }  
 public Integer getCourse() {  
 return course;  
 }  
 public void setCourse(Integer course) {  
 this.course = course;  
 }  
 public void setPersonalNumber(Integer personalNumber) {  
 this.personalNumber = personalNumber;  
 }  
 public int getPersonalNumber() {  
 return personalNumber;  
 }  
  
 @Override  
 public String toString() {  
 return "Student " + getName() + ", personalId = " + getPersonalNumber() + ", course = " +  
 getCourse()  
 + ", group = " + getGroupNumber();  
 }  
}

Тестирование:

#class TestDAO  
package com.company;  
 import com.company.dao.DAOFactory;  
 import com.company.dao.StudentDAO;  
 import com.company.model.Student;  
public class TestDAO {  
 public static void main(String[] args) {  
 DAOFactory mysqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);  
 StudentDAO studentDAO = mysqlFactory.getStudentDAO();  
 // CREATE  
 Student student = new Student();  
 student.setPersonalNumber(1);  
 student.setName("Arsenij Alferov");  
 student.setCourse(1);  
 student.setGroupNumber("SP441");  
 int systemId = studentDAO.create(student);  
 System.out.println("Student's system id = " + systemId);  
 // READ  
 Student studentFromDB1 = studentDAO.readBySystemlId(systemId);  
 System.out.println("after create: " + studentFromDB1);  
 // UPDATE  
 studentFromDB1.setGroupNumber("SP442");  
 studentDAO.update(studentFromDB1);  
 // READ  
 Student studentFromDB2 = studentDAO.readBySystemlId(systemId);  
 System.out.println("after update: " + studentFromDB2);  
 // DELETE  
 studentDAO.delete(studentFromDB2.getPersonalNumber());  
 }  
}

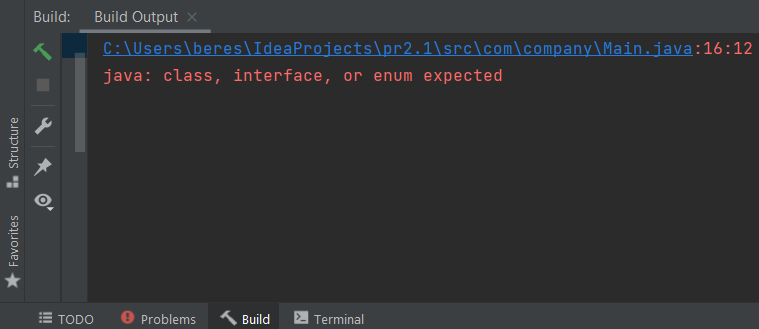


Рисунок 8 - Результат выполнения

Ссылка на Github https://github.com/dmitryberest/Practice

Список использованных источников

1.Java Массивы (Java Array)

<https://javarush.ru/groups/posts/massivy-java>

2. Учебник Java Методы Программирования

3. Учебник Семакин И.Г. Основы алгоритмизации и программирования