|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**

по дисциплине «Разработка клиент-серверных приложений»

**Тема лабораторной работы**: «Простое клиент-серверное приложение на сокетах»

|  |  |  |
| --- | --- | --- |
| Студент группыНИКБО-07-18 | Фроленко М.Д | (подпись) |
| Руководитель  лабораторных работ | Скворцова Л.А | (подпись) |
| Отчет представлен к защите | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_201\_г. |  |
| Допущен к защите | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_201\_г. |  |

Москва 2020 г.

**Содержание**

Текстовое описание лабораторной работы3

Заключение4

Список используемых источников5

Приложение6

**Задание практической работы**

Разработать клиент-серверное приложение, выполняющее функции по вариантам. Вариант №6:

Вывести уравнение прямой, проходящей через две точки. На вход поступает 4 числа: х1, у1, х2, у2.. Числа должны поступать в виде строки с некоторым разделителем; в массиве; списком чисел.

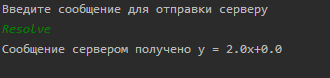
**Результат выполнения работы**

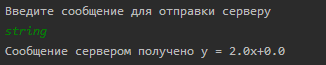
В первую очередь необходимо создать класс Client для работы с клиентской частью приложения – получения сообщения и отправкой его на сервер.

Для удобной работы с сокетами необходимо разработать класс ObjArray, хранящий данные для работы сервера и сообщение.

Далее необходимо создать класс Server для обработки сообщений и отправки результата.

Для подготовки ответа создан интерфейс Result и класс Variant , реализующий его.







**Приложениe**

1. **Server.java**
2. package com.company;  
   import java.io.\*;  
   import java.net.ServerSocket;  
   import java.net.Socket;  
     
   public class Server {  
    public static final int *SERVER\_PORT* = 60001;  
    public static void main (String[] args)  
    { ObjArray Obj;  
    //Прослушивание входящих сообщений на порте 60001  
    try (ServerSocket server = new ServerSocket(*SERVER\_PORT*))  
    {  
    String message =null;  
    while(!"STOP".equals(message)){  
   //принятие входящего клиента  
    System.*out*.println("Ожидание соединения");  
    try{//настройка сокета соединения с клиентом  
    Socket connectionSocket = server.accept();  
   //Настройка считывателя входного потока (от клиента)  
    ObjectInputStream incoming= new  
    ObjectInputStream(connectionSocket.getInputStream());  
   //Настройка записи иходящего потока (к клиенту)  
    ObjectOutputStream outgoing =new  
    ObjectOutputStream(connectionSocket.getOutputStream());  
   //блок try  
   //получение отправленного клиентом сообщения  
    System.*out*.println("Ожидание сообщения клиента");  
    Obj= (ObjArray) incoming.readObject();  
    message = Obj.getM();  
    System.*out*.println("Получено от клиента сообщение "+ message);  
   //Отправить сообщение клиенту  
    String s = new String("STOP");  
    String str=new String("");  
    if (!message.equals(s)) {  
    Variant varray;  
    switch (message){  
    case "array" : varray = new Variant(Obj.getArray());break;  
    case "string" : varray = new Variant(Obj.getReq());break;  
    case "arraylist" : varray = new Variant(Obj.getReq1());break;  
    default:  
    throw new IllegalStateException("Unexpected value: " + message);  
    }  
      
    String max = varray.getResult();  
    str="Сообщение сервером получено " ;  
    //outgoing.writeObject(str);  
    str = str + max;  
    //outgoing.writeUTF(str);  
    } else {  
    str="Соединение с сервером закрыто";  
    // outgoing.writeObject(str);  
    }  
     
    outgoing.writeUTF(str);  
   //Закрытьвыходной поток, так как больше нету данных для  
   // оправки клиенту  
    outgoing.close();  
    }catch (IOException e) {  
    e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
    e.printStackTrace();  
    }  
    }  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }  
   }
3. **Client.java**
4. package com.company;  
   import java.io.\*;  
   import java.net.ServerSocket;  
   import java.net.Socket;  
   import java.util.ArrayList;  
   import java.util.Arrays;  
   import java.util.List;  
   import java.util.Scanner;  
     
   public class Client {  
    public static void main(String[] args) throws UnsupportedEncodingException {  
   //Прослушивание входящих сообщений на порте 50001  
    try (  
   // создание сокета для подключения к серверу  
    Socket clientSocket = new Socket("localhost", 60001);  
   //настройка считывателя входного потока  
   //для ввода с клавиатуры  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*, "UTF-8"));  
   //настройка считывателя входного потока от сервера  
    // ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());  
   //настройка записи исходяего потока потока от сервера  
    ObjectOutputStream out = new ObjectOutputStream(clientSocket.getOutputStream());  
    ) { //ввод сообщения пользователя с клавиатуры  
    System.*out*.println("Введите сообщение для отправки серверу");  
    String message = br.readLine();  
   //отправить сообщение на сервер  
    Double[] req1 = {1.0,2.0,2.0,4.0};  
    String req2 = "1.0,2.0,2.0,4.0";  
    ArrayList<Double> req3 = new ArrayList<Double>(Arrays.*asList*(req1));  
     
    ObjArray Obj = new ObjArray(req3, message);  
    out.writeObject(Obj);  
     
   //считать ответ от сервера  
    try {  
    ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());  
    //BufferedReader br1 = new BufferedReader(new InputStreamReader(in));  
    //String reply = br1.readLine();  
    //System.out.println(reply);  
    String obj = (String)in.readUTF();  
    System.*out*.println(obj);  
     
    } catch (EOFException eof) {  
   //ничего не делать, сервер закрыл соединение  
    }  
   //закрыть соединение  
    clientSocket.close();  
    } catch (IOException e) {  
    e.printStackTrace();  
    }  
    }  
   }
5. **ArrayObj.java**
6. package com.company;  
     
   import java.io.Serializable;  
   import java.util.ArrayList;  
     
   public class ObjArray implements Serializable {  
    private Double[] x;  
    private String req;  
    private ArrayList<Double> req1;  
    private String message;  
    public ObjArray(Double[] a, String m){  
    x=a; message = m;  
    }  
     
    public ObjArray(String req1 , String m){  
    req = req1 ; message = m;  
    }  
     
    public ObjArray(ArrayList<Double> req , String m){  
    req1 = req ; message = m;  
    }  
    public Double[] getArray(){  
    return x;  
    }  
    public String getM(){  
    return message;  
    }  
     
    public String getReq() {  
    return req;  
    }  
     
    public ArrayList<Double> getReq1() {  
    return req1;  
    }  
   }
7. **Result.java**
8. package com.company;  
     
   public interface Result {  
    public String getResult();  
     
   }
9. **Variant.java**
10. package com.company;  
      
      
    import java.util.ArrayList;  
      
    public class Variant implements Result {  
     private Double []x;  
     @Override  
     public String getResult() {  
     String answer;  
     double k,b;  
     k = (x[3] - x[1]) / (x[2] - x[0]);  
     b = x[1] - k \* x[0];  
     answer = "y = " + k + "x+" + b;  
     return answer;  
     }  
     public Variant(String s){  
     String [] s1 = s.split(",");  
     x = new Double[s1.length];  
     for(int i=0;i<s1.length;i++){  
     x[i] =(Double.*parseDouble*(s1[i]));  
     }  
     };  
     public Variant(Double[]a){  
     x=a;  
     }  
     public Variant(ArrayList<Double> L){  
     //L.toArray(x);  
     x = new Double[L.size()];  
     L.toArray(x);  
      
     }  
    }