ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
СЕВЕРО-КАВКАЗСКИЙ ФИЛИАЛ
ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО
БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО
ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
СВЯЗИ И ИНФОРМАТИКИ»

КАФЕДРА ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

МУ к ПЗ 1 и 2

по дисциплине

ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

Ростов-на-Дону

2019 г.

МУ к ПЗ 1 и 2

по дисциплине

ПРОЕКТИРОВАНИЕ КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

Для студентов очной и заочной форм обучения

Направление подготовки - **09.03.01 «**Информатика и вычислительная техника»

Составитель: П.В. Лобзенко, доцент кафедры ИВТ

Рассмотрено и одобрено
на заседании кафедры ИВТ Протокол от «26» августа 2019 г. № 1

МУ к ПЗ-1

Простые двухуровневые клиент-серверные приложения в Java

2.1.1 Цели занятия

Выработать умения и навыки по составлению программ простых одноуровневых клиент-серверных приложений на основе технологии сокетов.

2.1.2 Теоретические основы и пример выполнения

2.1.2.1 Создание сетевых приложений с использованием TCP

Java поддерживает классы и методы, которые позволяют устанавливать соединение с удаленным компьютером, используя протокол TCP. В отличие от UDP, TCP является протоколом, ориентированным на установление соединения, которое гарантирует надежную связь между приложениями клиента и сервера. Взаимодействие с использованием протокола TCP, начинается после установления соединения между сокетами клиента и сервера. Сокет сервера "слушает" запросы на установление соединения, отправленные сокетами клиентов, и устанавливает соединение. После установления соединения между приложениями клиента и сервера, они могут взаимодействовать друг с другом.

Java упрощает сетевое программирование, путем инкапсуляции функциональности соединения сокета TCP в классы сокета, в которых класс Socket предназначен для создания сокета клиента, а класс ServerSocket для создания сокета сервера.

2.1.2.2 Идентификация методов классов Socket и ServerSocket

Socket является базовым классом, поддерживающим протокол TCP. Класс Socket обеспечивает методы для потокового ввода/вывода, облегчает выполнение операций чтения и записи в сокет и является обязательным для программ, выполняющих сетевое взаимодействие.

Для создания объектов класса Socket используются следующие конструкторы, определенные в классе Socket:

- public Socket (InetAddress IP_address, int port) - создает объект Socket, который соединяется хостом, заданным в параметрами IP_address и port.
- public Socket (String hostname, int port) – создает объект Socket, который соединяется с хостом, заданным параметрами имя хоста или IP адрес и port, который сервер "слушает".

Некоторые полезные методы класса Socket представлены в таблице 2.1.

Таблица 2.1- Основные методы класса Socket

Метод	Описание
public InetAddress() public getInetAddress()	Возвращает объект InetAddress, который содержит IP-адрес, с которым соединяется объект Socket
public InputStream() public getInputStream()	Возвращает входной поток для объекта Socket
public InetAddress() public getLocalAddress()	Возвращает объект InetAddress, содержащий локальный адрес, с которым соединяется объект Socket
public int getPort()	Возвращает удаленный порт, с которым соединяется объект Socket
public int getLocalPort()	Возвращает локальный порт, с которым соединяется объект Socket
public OutputStream() public getOutputStream()	Возвращает выходной поток объекта Socke
void close()	Закрывает объект Socket
public string toString()	Возвращает IP-адрес и номер порта сокета клиента как String

ServerSocket представляет собой класс, используемый программами сервера для прослушивания запросов клиентов. ServerSocket реально не выполняет сервис, но создает объект Socket от имени клиента, через который выполняется взаимодействие с сокетом клиента.

Для создания и инициализации объектов ServerSocket используются следующие конструкторы, определенные в классе ServerSocket:

- public ServerSocket(int port_number)- создает сокет сервера на заданный порт на локальной машине. Клиентам следуетиспользовать этот порт, чтобы общаться с сервером. Если номер порта 0, то сокет сервера создается на любой свободный порт локальной машины;
- public ServerSocket(int port, int backlog) - создает сокет сервера на заданный порт на локальной машине. Второй параметр задает максимальное количество соединений клиентов, которые сокет сервера поддерживает на заданном порту;
- public ServerSocket(int port, int backlog, InetAddress bindAddr) - создает сокет сервера на заданный порт. Третий параметр используется для создания сокета сервера хоста, подключенного к нескольким физическим линиям (multi-homed host). Сокет сервера принимает запросы клиента только с заданных IP адресов.

Некоторые полезные методы класса ServerSocket представлены в таблице 2.2.

Таблица 2.2- Основные методы класса ServerSocket

Метод	Описание
public InetAddress() public getInetAddress()	getInetAddress() Возвращает объект InetAddress, который содержит адрес объекта ServerSocket
public int getLocalPort()	Возвращает номер порта, с которого объект ServerSocket слушает запросы клиента
public Socket accept() throws IOException	Заставляет сокет сервера слушать соединение клиента и принимать его. После установления соединения клиента с сервером метод возвращает сокет клиента
public void bind(SocketAddress address) throws IOException	Связывает объект ServerSocket с заданным адресом (IP адрес и порт). Этот метод вы-

	зывает исключительную ситуацию IOException, когда происходит ошибка
public void close() throws IOException	Закрывает объект ServerSocket. Этот метод вызывает исключительную ситуацию IO-Exception, когда происходит ошибка
public String toString()	Возвращает IP адрес и номер порта сокета сервера как String

2.1.2.3 Создание северной части по протоколу TCP/IP в Java

Для создания серверного приложения сокета TCP, необходимо выполнить следующие шаги:

- создать объект сокета сервера ServerSocket (класс Server);
- прослушивать запросы клиента на соединение;
- запустить сервер;
- создать поток соединения для запросов клиентов.

2.1.2.4 Разработка класса Server

Класс Server наследует класс Thread и, таким образом, поддерживает многопоточное выполнение. Объект ServerSocket "слушает" запросы клиентов и конструктор класса Server создает объект ServerSocket. Сообщение об ошибке отображается, если возникает исключительная ситуация при запуске сервера.

Фрагмент кода для конструктора сервера показан в листинге 2.1

Листинг 2.1- Фрагмент конструктора класса Server

```
public Server()
{
 try
 {
 serverSocket = new ServerSocket(1001);
 }
 catch(IOException e)
 {
```

```
 fail(e, "Невозможно запустить сервер.");
 }
 System.out.println("Сервер запущен. . .");
 this.start(); // Запускается поток
}
```

В приведенном фрагменте кода используется общий метод обработки ошибок fail(), который обеспечивает обработку всех исключительных ситуаций. Метод принимает два аргумента (объект Exception и объект String) и выводит сообщение об ошибке.

Фрагмент кода для метода fail() выглядит следующим образом (листинг 2.2):

Листинг 2.2- Формирование сообщение об ошибке

```
public static void fail(Exception e, String str)
{
 System.out.println(str + "." + e);
}
```

2.1.2.5 «Прослушивание» запросов клиентов

Метод run() класса Server, как любой поток, который реализует интерфейс Runnable, содержит инструкции для потока. В этом случае сервер переходит в бесконечный цикл и прослушивает запросы клиентов. Когда сервер обнаруживает подключение клиента, метод accept() класса ServerSocket выполняет соединение. При этом сервер создает объект класса Connection для клиента. Объект класса Socket передается конструктору класса Connection и взаимодействие между клиентом и сервером выполняется через этот сокет.

Фрагмент кода для метода run() выглядит показан на листинге 2.3.

Листинг 2.3- Метод для прослушивания запросов клиентов

```
public void run()
{
 try
 {
```

```
 while(true)
 {
 Socket client = serverSocket.accept();
 Connection con = new Connection(client);
 }
 }
 catch(IOException e)
 {
 fail(e, “Не прослушивается”);
 }
}
```

2.1.2.6 Запуск сервера

Фрагмент кода для метода main() приведен в листинге 2.4.

Листинг 2.4- Запуск сервера

```
public static void main(String args[])
{
 new Server();
}
```

В этом фрагменте кода создается объект класса Server, который запускает поток.

2.1.2.7 Создание потокового соединения

Класс Connection нужен для потокового соединения с клиентом (листинг 2.5).

Листинг 2.5- Класс для создания потокового соединения

```
class Connection extends Thread
{
protected Socket netClient;
 protected BufferedReader fromClient;
 protected PrintStream toClient;
 public Connection(Socket client)
 {
```

```
 netClient = client;
 try
 {
fromClient = new BufferedReader(new
InputStreamReader(netClient.getInputStream()));
toClient = new PrintStream(netClient.getOutputStream());
 }
 catch(IOException e)
 {
 try
 {
 netClient.close();
 }
 catch(IOException e1)
 {
 System.err.println("Unable to set up streams"
+ e1);
 return;
 }
 }
 this.start();
 }
 public void run()
 {
 String clientMessage;
 try
 {
 for(;;)
 {
 clientMessage = fromClient.readLine();
 if(clientMessage == null)
 break;
 // Посылает подтверждение клиенту
 toClient.println("Received");
 }
```

```
}
catch(IOException e)
{}
finally
{
try
{
netClient.close();
}
catch(IOException e)
{}
}
}
}
```

В представленном фрагменте кода класс Connection создает объект fromClient класса BufferedReader, который получает ввод от клиента, используя метод getInputStream(). Объект класса PrintStream (toClient) дает возможность серверу отправлять клиенту данные, используя метод getOutputStream(). Таким образом, возникают необходимые (прием и передача) возможности взаимодействия.

Когда клиент соединяется с сервером, сервер использует метод readLine() объекта fromClient, чтобы запомнить сообщение, посланное клиентом в переменной clientMessage типа String. Метод println() используется для вывода сообщения “Received ” сокету.

Для выхода из системы сервер завершает цикл. При этом выполняется блок finally для закрытия сокета клиента. Закрытие сокета является важным действием, поскольку сохранение активного соединения неизбежно приводит к потери памяти сервера. Блок finally обеспечивает закрытие ранее установленного соединения. Следует отметить, что сервер является многопоточным, и каждый клиент получает свой собственный поток на сервере.

2.1.2.8 Пример создания класса Server

В листинге 2.6 показан создания класса Server, который принимает запросы соединения клиента и посылает строку Login, как ответное сообщение клиенту.

Листинг 2.6- Пример кода класса Server

```
import java.io.*;
import java.net.*;
public class Server extends Thread {
ServerSocket serverSocket; // Определяется переменная server-
Socket
public Server() {
try {
/*
* Создание объекта ServerSocket, который принимает запросы
* соединения от клиентов через порт 1001
*/
serverSocket = new ServerSocket(1001);
System.out.println(serverSocket.toString());
} catch (IOException e) {
fail(e, "Could not start server.");
}
System.out.println("Сервер запущен . . .");
/* Стартует поток */
this.start();
}
public static void fail(Exception e, String str) {
System.out.println(str + "." + e);
}
public void run() {
try {
while (true) {
/* Принимаются запросы от клиентов */
```

```
   Socket client = serverSocket.accept();
   /*
* Создается объект соединение, чтобы управлять
   взаимодействием
   * кдиента с сервером
   */
   Connection con = new Connection(client);
   }
   } catch (IOException e) {
   fail(e, "Not listening");
   }
   }
   public static void main(String args[]) {
   /* Запускается сервер */
   new Server();
   }
   }
   class Connection extends Thread {
   /* Declare the variables */
   protected Socket netClient;
   protected BufferedReader fromClient;
   protected PrintStream toClient;
   public Connection(Socket client) {
   netClient = client;
   try {
   /* Создается входной поток, чтобы принимать данные от
   клиента */
   fromClient = new BufferedReader(new
   InputStreamReader(
   netClient.getInputStream()));
   /* Создается выходной поток, чтобы посылать данные
   клиенту */
   toClient = new
   PrintStream(netClient.getOutputStream());
   } catch (IOException e) {
```

```
try {
/* Закрывается сокет клиента */
netClient.close();
} catch (IOException e1) {
System.err.println("Unable to set up streams" +
e1);
return;
}
}
/* Start the thread */
this.start();
}
public void run() {
String login, password;
try {
while (true) {
toClient.println("Login: ");
/* Принимается login как ввод от клиента */
login = fromClient.readLine();
/* Завершить соединение, когда Bye вводится как login */
if (login == null || login.equals("Bye")) {
System.out.println("Exit");
return;
} else
System.out.println(login + " logged
in");
// Посылается подтверждение клиенту
toClient.println("Password: ");
/* Принимается пароль то клиента */
password = fromClient.readLine();
}
} catch (IOException e) {
} finally {
try {
netClient.close();
```

```
} catch (IOException e) {
}
}
}
}
```

2.1.2.9 Создание клиента TCP/IP

Первый шаг выполнения клиента состоит в установлении соединения между клиентом и сервером. Для установления соединения между клиентом и сервером необходимо создать объект Socket. Для создания приложения клиента сокета TCP необходимо выполнить следующие задачи:

- создать сокет клиента, используя объект Socket;
- читать и писать в сокет;
- закрыть соединение.

2.1.2.10 Создание сокета клиента

Объект сокета клиента создается с помощью конструктора класса Socket, принимающего два параметра, IP адрес и номер порта, которые прослушиваются сервером. Следующий фрагмент кода используется для создания сокета клиента (листинг 2.7).

Листинг 2.7- Создание сокета клиента

```
try
{
 Socket clientSocket = new Socket("127.0.0.1", 1001);
}
catch(UnknownHostException e)
{
System.err.println("Неопределенное имя хоста ");
System.exit(1);
}
```

В предыдущем фрагменте кода IP адрес равный ‘127.0.0.1’ и порт равный ‘1001’ определяют сокет, на котором сервер прослушивает запросы клиента.

2.1.2.11 Чтение и запись в сокет

После установления соединения между клиентом и сервером, клиент посылает запрос на сервер через сокет. Чтение и запись в сокет аналогичны чтению из файла и записи в файл. Чтобы обеспечить возможность клиенту общаться с сервером, необходимо выполнить следующие действия:

- объявляются два объекта по одному для классов PrintStream и BufferedReader. Эти объекты будут использоваться для чтения и записи в сокет socket. PrintStream out = null; // Объект для записи в сокет BufferedReader in = null; // Объект для чтения из сокета;
- объекты PrintStream и BufferedReader связываются с сокетом. out = new PrintStream(clientSocket.getOutputStream()); in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

Методы getInputStream() и getOutputStream() класса Socket позволяют клиенту взаимодействовать с сервером. Метод getInputStream() позволяет объекту BufferedReader читать из сокета, а метод getOutputStream() позволяет объекту PrintStream писать в сокет.

Объявляется еще один объект класса BufferdReader для связи со стандартным входом, чтобы данные, введенные в приложении клиенте, могли передаваться на сервер. Следующий фрагмент кода используется для чтения данных из окна консоли (листинг 2.8).

Листинг 2.8- Чтение данных из окна консоли

```
BufferdReader stdin = new BufferedReader(new
InputStreamReader(System.in));
String str;
while((str = stdin.readLine()).length() != 0)
{
out.println(str);
```

```
}
```

Представленный фрагмент кода позволяет пользователю вводить данные с клавиатуры. Цикл while продолжается до тех пор, пока пользователь не введет символ завершения ввода (Ctrl-Z).

2.1.2.12 Закрытие соединения

Операторы, представленные ниже, закрывают потоки и соединения с сервером: out.close(); in.close(); stdin.close().

Сокет клиента принимает имя пользователя и пароль, и обеспечивает связь. Для завершения соединения, пользователь должен ввести ‘Bye’. Можно использовать следующий код для создания класса Client (листинг 2.9).

Листинг 2.9- Пример создания класса Client

```
import java.net.*;
import java.io.*;
public class Client {
public static void main(String[] args) throws IOException {
Socket clientSocket;
PrintStream out = null;
BufferedReader in = null;
try {
/* Создается объект сокет, чтобы соединиться с сервером
*/
clientSocket = new Socket("127.0.0.1", 1001);
/* Создается выходной поток, чтобы посылать данные на
сервер */
out = new
PrintStream(clientSocket.getOutputStream());
/* Создается входной поток, чтобы принимать данные с
сервера */
in = new BufferedReader(new InputStreamReader(
clientSocket.getInputStream()));
```

```
    } catch (UnknownHostException e) {
    System.err.println("Unidentified hostname ");
    System.exit(1);
    } catch (IOException e) {
    System.err.println("Couldn't get I/O ");
    System.exit(1);
    }
    /* Создается входной поток, чтобы читать данные из окна консо-
ли
    */
    BufferedReader stdin = new BufferedReader(new
    InputStreamReader(
    (System.in)));
    /* Чтение из сокета */
    String login = in.readLine();
    System.out.println(login);
    /* Прием login */
    String logName = stdin.readLine();
    out.println(logName);
    /* Чтение из сокета */
    String password = in.readLine();
    System.out.println(password);
    /* Прием password */
    String pass = stdin.readLine();
    out.println(pass);
    String str = in.readLine();
    System.out.println(str);
    while ((str = stdin.readLine()) != null) {
    out.println(str);
    if (str.equals("Bye"))
    break;
    }
    out.close();
    in.close();
    stdin.close();
```

```
}
}
```

Представленный выше код сохраняется как Client.java. При выполнении класса Client, отображается сообщение для ввода Login. После приема Login на консоли класса Client появляется сообщение для ввода password. Рис. 7 показывает вывод класса Client, который посылает login XXXX и password для входа на Server. Введенный login в окне консоли класса Client передается классу Server. Когда на сервер передается строка «Bye» в качестве login от клиента, соединение между клиентом и сервером прекращается.

2.1.3 Порядок выполнения задач ПЗ

Алгоритм выполнения задач практического занятия следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить классы Swrver и Client и установить соединение между ними;
- составить методы (функции) решения всех задач, поместить их в класс Server. Все исходные данные вводить в консоли клиента, передавать их на сервер, а полученные результаты решений задач выводить на экран в консоли клиента;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.1.4 Варианты заданий:

1. Составить программу для перевода длины в метрах в длину в сантиметрах, определив функцию, выполняющую это преобразование и передав длину в метрах в качестве параметра.

2. Составить программу для нахождения суммы элементов каждого из трех массивов, введенных с клавиатуры, определив функцию, выполняющую это действие, и передавая массивы в качестве параметра.

3. Даны числа S, T. Получить с использованием функции пользователя F(T,-2S,1.17)+F(2.2,T,S-T) где F(A, B, C) = (2A-B-sin(C))/(5+C)

4. Составить программу перевода двоичной записи натурального числа в десятичную, описав соответствующую функцию с параметром. Перевод осуществлять для чисел, вводимых с клавиатуры. Признак конца ввода - число 0.

5. Даны числа S, T. Получить с использованием функции пользователя с параметрами G(1,sin(S))+2G(T*S,24)-G(5,-S), где G(A,B)=(2A+B*B)/(A*B*2+B*5).

6. Составить программу для расчета значений гипотенузы треугольника, определив функцию, выполняющую этот расчет. Катеты передаются в качестве параметров.

7. Найти периметр десятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, которые передаются функции в качестве параметров из основной программы.

8. Найти периметр шестиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами. Координаты передаются функции в качестве параметров из основной программы.

9. Найти площадь пятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, и процедуру вычисления площади треугольника по трем сторонам. Описать функции с соответствующими формальными параметрами.

10. Составить программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве пара-

метра натуральное число и возвращающую TRUE, если оно удовлетворяет указанному условию.

11. Используя подпрограмму - функцию, составить программу для нахождения максимального из трех чисел. Числа передаются функции в качестве параметров.

12. Используя подпрограмму - функцию, составить программу для печати знаков трех чисел, введенных с клавиатуры и передаваемых функции в качестве параметра.

13. Используя подпрограмму - функцию, составить программу для возведения чисел в целую положительную степень. Число передаются функции в качестве параметра из основной программы. Расчет вести для чисел, пока не будет введено число, равное 0.

14. Используя подпрограмму - функцию, составить программу для вычисления функции Z=(X1+Y1)/(X1*Y1), где X1 - первый корень уравнения X^2-4*X-1=0; Y1 - первый корень уравнения $2*Y^2 + A*Y - A^2 = 0$ (A - произвольное).

15. Задав функцию, вывести на печать средние арифметические двух массивов, введенных с клавиатуры. Массив передается функции в качестве параметра.

16. Задав функцию, рассчитать и вывести на печать максимальные значения в трех парах чисел, вводимых с клавиатуры. Пара чисел передается функции в качестве параметра.

17. Найти периметр восьмиугольника, координаты вершин которого заданы. Определить функцию вычисления расстояния между двумя точками, заданными своими координатами. Координаты передать функции в качестве параметров.

18. Даны четыре пары чисел. Получить с использованием функции пользователя наибольший общий делитель для каждой пары.

19. Даны числа A, B, C. Получить с использованием функции пользователя наименьшее значение. Числа передаются функции из основной программы в качестве параметров.

20. Даны числа x = 1,2,...,N. Получить с использованием функции пользователя значения 3*P(X+3)*P(X) для заданных x, где $P(X) = 10*X^3 - 14*X^2 + 12*X - 2$.

21. Составить программу для расчета значений катета треугольника, определив функцию, выполняющую этот расчет. Гипотенуза и второй катет передаются в качестве параметров.

22. Даны целые числа a,b,c,d. Проверить с использованием функции пользователя их четность. Число для проверки передается в функцию в качестве параметра из основной программы.

23. Для каждого из 10 введенных с клавиатуры чисел напечатать сообщение: является ли оно простым или нет, описав функцию логического типа, возвращающую значение “ИСТИНА”, если число, переданное ей в качестве параметра, является простым.

24. Даны числа S, T. Получить с использованием функции пользователя Y(T,S)=G(12,S)+G(T,S)-G(2S-1,S*T), где G(A,B)=(2*A+B*B)/(A*B*2+B*5).

25. Определите функцию, определяющую, какой целой степенью числа 2 является ее аргумент (если число не является степенью двойки - выдать соответствующее сообщение).

26. Определите функцию, подсчитывающую сумму N первых элементов целочисленного массива A. N и массив A передать в качестве параметров.

27. Вычислить количество простых чисел, не превосходящих заданного N. Описать функцию логического типа, возвращающую значение true, если число простое и false в противном случае.

28. Используя подпрограмму - функцию с параметрами, составить программу для вычисления функции $F(X,Y) = (2X^3-4*X^2+X+1)/(9*Y^3+Y+4) + 3*Y^2+5*Y$.

29. Составить программу для перевода веса в граммах в вес в килограммах, определив функцию, выполняющую это преобразование. Вес в граммах передается функции в качестве параметра.

30. Даны числа S, T. Получить с использованием функции пользователя G(12, S)+G(T, S)-G(2S-1, S*T) где G(A, B) = (2*A+B*B)/(A*B*2+B*5).

МУ к ПЗ-2

Простые двухуровневые клиент-серверные приложения в C#

2.2.1 Цели занятия

Выработать умения и навыки составлять типовые программы решения задач на выбранном языке программирования, снабженные элементами графического интерфейса пользователя в виде клиент-серверных приложений.

2.2.2 Теоретические основы и пример выполнения

Для создания клиент-серверного приложения в C#, аналогично предыдущему ПЗ, необходимо создать два класса Server и Client, используя технологию сокетов.

2.2.2.1 Программирование класса Server

Вначале в этом классе создадим серверное соединение и используем для этого систему сокетов.

Сокеты- это концепция сетевого программирования, когда существуют два вида «программных разъемов»- сокетов: клиентские и серверные.

Серверный сокет как розетка, которая «висит на стене» готовая к работе, в ожидании, когда к ней подключат штекер. Точно так же серверный сокет переходит в режим ожидания подключения на определенном адресе и определенном порту.

Клиентский сокет как вилка, которую втыкают в розетку. Как только клиент подключается к серверному сокету, информация начинает передаваться между ними.

Для того чтобы создать виртуальное подключение между клиентом и сервером, надо знать место, где находится нужный нам серверный сокет.

В нашем случае, создается слушатель событий подключений в сети по протоколу TCP IP – TcpListener, в качестве параметра которому передается так называемая точка входа, которую определяет метод IPEndPoint из переданных

ему IP адреса компьютера, на котором будет запущен сервер (пока это вариант localhost, т.е. IP 127.0.0.1) и номер порта Port, на котором создаваемой серверное подключение будет прослушивать подключения клиентов. Т.о., в конструкторе класса Server появится следующий слушатель (листинг 2.10).

Листинг 2.10- Серверный слушатель подключения клиентов

```
TcpListener listner;
        public Program(int Port)
        {
            listner = new TcpListener(new IPEnd-
Point(IPAddress.Parse("127.0.0.1"), Port));
            listner.Start();
            Console.WriteLine("Сервер запущен, ожидает подклю-
чений...");
. . . }
```

Видно, что после запуска прослушивания сетевых подключений выдается сообщение о том, что сервер готов и ожидает подключений.

Далее, нужно в это подключение с помощью метода AcceptTcpClient() и создать новый поток для него с помощью метода Thread из одноименной библиотеки, которую нужно подключить заранее: using System.Threading.

В качестве параметра этому методу передается метод clientThread, в котором для каждого нового клиента открывается TCP соединение в новом потоке (листинг 2.11).

Листинг 2.11- Метод создания нового TCP соединения для нового клиента

```
static void clientThread(Object StateInfo)
        {
           new Client((TcpClient)StateInfo);
        }
```

В итоге получим следующий текст класса Server (листинг 2.12).

Листинг 2.12- Класс Server

```
class Server
    {
       TcpListener listner;
        public Server (int Port)
        {
            listner = new TcpListener(new IPEnd-
Point(IPAddress.Parse("127.0.0.1"), Port));
            listner.Start();
            Console.WriteLine("Сервер запущен, ожидает подклю-
чений...");
            while (true)
            {

                TcpClient client = listner.AcceptTcpClient();
                // Создаем поток


     Thread Thread = new Thread(new ParameterizedThread-
Start(clientThread));
                // И запускаем этот поток, передавая ему при-
нятого клиента
                Thread.Start(client);
            }
        }

        static void clientThread(Object StateInfo)
        {
           new Client((TcpClient)StateInfo);
        }

        static void Main(string[] args)
        {
            new Program(12000);
        }
```

```
}
```

Видно, что сюда же включен и главный метод модуля – метод Main, из которого происходит запуск всего приложения и сервер начинает «прослушивание» на порту №12000, который в качестве параметра передается конструктору класса.

2.2.2.2 Класс Client

Этот класс необходим для считывания запроса от клиентских приложений, поиска нужного контента согласно запросу и для его передачи в сеть клиентам.

Начнем с задания объектов для методов чтения из потока и записи в поток. Для этого создаются новые экземпляры соответствующих классов (листинг 2.13).

Листинг 2.13- Организация потокового чтения и записи

```
StreamReader sr = new StreamReader(client.GetStream()); // чтение из потока клиента
                StreamWriter sw = new StreamWriter(client.GetStream()); // запись в поток клиента
                sw.AutoFlush = true;
```

Этот пример клиент-серверного приложения обеспечивает запрос от клиента на сервер необходимого файла информации, передачу его клиенту, сохранение в темповском файле на клиенте и отображение в клиентской части приложения. Поэтому, определим какой файл необходим: прочитаем из потока в за-просе клиента значение строковой переменной str, конвертируем его в целочисленную переменную kluch, и по ее значению в структуре switch определим необходимый файл (листинг 2.14).

Листинг 2.14- Определение требуемого контента

```
    // Принимаем передачу от клиента
                string cl = sr.ReadLine(); // Отображение типа обучающего модуля
                Console.WriteLine(cl);
                string str = sr.ReadLine();
                int kluch = int.Parse(str); // по kluch определяем нужный файл
                string FilePath = "C:/www/"; // составляем путь к файлу
                switch (kluch)
                {
                    case 1:
                        {
                            FilePath += "L1.htm";
                            Console.WriteLine("Путь к файлу:   " + FilePath);
                        }
                        break;

                    case 2:
                        {
                            FilePath += "L2.htm";
                            Console.WriteLine("Путь к файлу:   " + FilePath);
                        }
```

```
                    break;
                }
```

Видно, что весь контент преобразован в html формат и находится в соответствующих папках на сервере. Аналогично программируется определение оставшихся файлов обучающих материалов.

Далее нужно открыть найденный файл. Сделаем это, проверяя правильность его открытия с помощью блока try/catch и если возникнет ошибка открытия, выдадим соответствующее сообщение (листинг 2.15).

Листинг 2.15- Безопасное открытие файла контента

```
// Открываем файл, страхуясь на случай ошибки
                FileStream FS;
                try
                {
                    FS = new FileStream(FilePath, FileMode.Open, FileAccess.Read, FileShare.Read);
                    sw.WriteLine("Запрашиваемый файл открыт и готов к передаче");
                }
                catch (Exception)
                {
                    // Если случилась ошибка, посылаем клиенту ошибку открытия файла
                    sw.WriteLine("Ошибка открытия файла");
                    return;
                }
```

Учитывая, что данные передаются в сети в формате байт, нужно зарезервировать буфер для чтения материалов из открытого файла. Затем, перенести эти данные в буфер и через него передать их клиенту в ответе на запрос. Весь этот процесс будет выполняться до конца данных в открытом файле (листинг 2.16).

Листинг 2.16- Передача контента в ответном сообщении клиенту

```
// Задаем буфер для размещения данных из выбранного файла
                byte[] Buffer = new byte[1024];
                // Переменная для хранения количества байт, пере-
давaемых клиенту
                int Count;


                // Пока не достигнут конец файла
                while (FS.Position < FS.Length)
                {
                    // Читаем данные из файла и пишем их в буфер
                    Count = FS.Read(Buffer, 0, Buffer.Length);
                    // И передаем их клиенту
                    client.GetStream().Write(Buffer, 0, Count);
                }
                sw.WriteLine("Файл передан");
                // Закроем файл и соединение
                FS.Close();
                sw.WriteLine("Соединение прервано");
                client.Close();
```

Из листинга следует, что по завершению передачи данных клиенту мы закрываем файл и соединение.

2.2.3 Порядок выполнения задач ПЗ

Алгоритм выполнения задач практического занятия следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов здач, то перейти в его начало);
- составить классы Swrver и Client и установить соединение между ними;
- составить методы (функции) решения всех задач, поместить их в класс Server. Все исходные данные вводить в консоли клиента, передавать их на сервер, а полученные результаты решений задач выводить на экран в консоли клиента;
- оформить отчет для всего приложения вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.2.4 Варианты заданий

1. Вывести на печать положительные значения функции y=sin(x)+5cos(x-2) для x изменяющегося на отрезке [-5, 12] с шагом 1,2.

2. Вывести на печать значения функции z=tg(2x)-sin(x) для x изменяющегося на отрезке [-3, 3] с шагом 0,3.

3. Ввести с клавиатуры и напечатать модули N чисел; если введено отрицательное число, ввод и печать прекратить.

4. Вывести на печать значения функции z=ln(x)+tg(2x), большие 1, для x изменяющегося на отрезке [3, 8] с шагом 0,9.

5. Определить, является ли натуральное число N степенью числа 5 или нет.

6. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

7. Напечатать значения функции y=ln(x-1/x),где значения x вводятся с клавиатуры. При вводе числа,не входящего в область определения функции, вычисления прекратить.

8. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

9. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,2; 0,8) для x изменяющегося на отрезке [4,-6] с шагом 0,91.

10. Дано натуральное число N. Получить наименьшее число вида 4^k, большее N.

11. Для x из интервала (2;8) с шагом 0,75 вычислить y=(4x-3x+cos(x))/A, где A вводится с клавиатуры.

12. Найти первый член последовательности ln(9n)/(n*n), меньший 1, для n изменяющегося следующим образом: n=1,2,3... .

13. Определить, является ли натуральное число N степенью числа 3 или нет.

14. Вывести на печать отрицательные значения функции z=cos(x)-5sin(x-2) для x изменяющегося на отрезке [-3, 11] с шагом 0,9.

15. Ввести с клавиатуры и напечатать квадраты N чисел, если введено кратное 3 положительное число, ввод и печать прекратить.

16. Вывести на печать отрицательные значения функции z=tg(x)+5cos(x-2) для x изменяющегося на отрезке [12, 1] с шагом 1,2.

17. Ввести с клавиатуры и напечатать N чисел, если введено равное нулю или кратное 2 число, ввод и печать прекратить.

18. Вывести на печать значения функции z=ln(|x|)+tg(2x), большие 2 для x изменяющегося на отрезке [3, -8] с шагом 0,9.

19. Найти первый отрицательный член последовательности sin(tg(n/2)) для n изменяющегося на следующим образом: n=1,2,3... .

20. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

21. Найти первую цифру в целом положительном числе.

22. Дано натуральное число N. Получить наибольшее число вида 3^k, меньшее N.

23. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,3;0,7) для x изменяющегося на отрезке [-4,6] с шагом 0,91.

24. Дано натуральное число N. Получить наименьшее число вида 5^k, большее N.

25. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

26. Найти первый член последовательности ln(9n/(n*n+1), меньший 0, для n изменяющегося на следующим образом: n=1,2,3... .

27. Определить, является ли натуральное N степенью числа 4 или нет.

28. Вывести на печать положительные значения функции z=sin(x)-5cos(x-2) для x изменяющегося на отрезке [5,-12] с шагом 1,2.

29. Напечатать значения функции $Y = \sqrt{2x^2 - x^3}$ для произвольных *x,* вводимых с клавиатуры. При вводе числа, не входящего в область определения функции, ввод и печать прекратить.

30. Найти первый отрицательный член последовательности cos(ctg(n)) для n изменяющегося на следующим образом: n=1,2,3... .

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дубков – СП: НИУ ИТМО, 2013. – 248 с.

2 Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API. [Электронный ресурс] // [сайт] [2018], URL: https://tproger.ru/articles/web-api/. (дата обращения: 03.05.2018).

3 Технологии разработки клиент-серверных приложений. [Электронный ресурс] // [сайт] [2018], URL: https://bourabai.ru/dbt/client2.htm. (дата обращения: 05.05.2018).

4 Сенина А.А., Тузовский А.Ф Обзор основных современных технологий разработки web-приложений. [Электронный ресурс] // [сайт] [2018], URL: https://core.ac.uk/download/pdf/53095773.pdf. (дата обращения: 03.05.2018).

5 Басс Лен, Клементс Пол, Кацман Рик. Архитектура программного обеспечения на практике. 2-е издание. — СПб.: Питер, 2006. — 575 с.

6 П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано. Android для программистов: создаём приложения. — СПб.: Питер, 2013. — 560 с.

7 Краткое описание Visual Studio [Электронный ресурс] // [сайт] [2018], URL: http://habrahabr.ru. (дата обращения: 10.05.2018).

8 Фленов М. Е. Библия C#. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 560 с.: