МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

ФАКУЛЬТЕТ ТЕЛЕКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

Кафедра Аппаратное и программное обеспечение систем управления в телекоммуникации

О.Н.Джураев, Х.Х.Ахмедова, Ф.К.Тожиева

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по выполнению лабораторных работ по предмету

ОСНОВЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ

ЧАСТЬ 1

Ташкент 2022

Авторы: О.Н.Джураев, Х.Х.Ахмедова, Ф.К.Тожиева

Методическое пособие по выполнению лабораторных работ по предмету "Основы сетевого программирования" часть 1. -Ташкент: ТУИТ. 2022. - 72 стр.

В методическом пособии представлены указания по разработке TCP, UDP и Multicast клиент-серверных сетевых программ; работа с интернет-адресами, формат обмена данными JSON и с гипертекстом; по созданию сетевых приложений по передаче файлов, электронная почта и на основе AJAX. Оно включает в себя название лабораторной работы, цель работы, теоретическую часть, задание, порядок выполнения работы и контрольные вопросы.

Методическое пособие предназначено для бакалавров направления 5350100 – Телекоммуникационные технологии ("Телекоммуникации") для применения в учебном процессе.

Решением научно-методического совета Ташкентского университета информационных технологий имени Мухаммада ал-Хоразмий методическое пособие рекомендуется для публикации. ("___" - протоколом "___" _____ 2022 год).

Ташкентский университет информационных технологий имени Мухаммада Ал-Хоразмий, 2022

ВВЕДЕНИЕ

В республике реализуются комплексные меры по активному развитию цифровой экономики, а также широкому внедрению современных информационно-коммуникационных технологий во все отрасли и сферы, прежде всего, в государственное управление, образование, здравоохранение и сельское хозяйство.

А также, в нашей стране проводится масштабная работа по последовательному внедрению информационно-коммуникационных технологий в государственном управлении, отраслях экономики, социальной сфере и повседневной жизни.

Обеспечение включения Ташкентского университета информационных технологий имени Мухаммада аль-Хорезми в ТОП-1000 самых престижных университетов мира в течение трех лет, укрепление сотрудничества с ведущими университетами и компаниями отрасли, стажировки и обучение преподавателей и студентов за рубежом широко признанный.

Всесторонняя поддержка местных разработчиков программного обеспечения, являющегося важной составляющей информационных и коммуникационных технологий, является постоянным приоритетом нашего государства. В частности, принятые нормативные акты предусматривают комплекс мер по стимулированию развития программного обеспечения.

Первая часть методического пособия по выполнению лабораторных работ по предмету «Основы сетевого программирования» по направлению «5350100-Телекоммуникационные технологии («Телекоммуникации»)» состоит из указания по разработке TCP, UDP и Multicast клиент-серверных сетевых программ; работа с интернет-адресами, формат обмена данными JSON и с гипертекстом; по созданию сетевых приложений по передаче файлов, электронная почта и на основе AJAX.

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Создание сетевого приложения TCP клиент-сервер

Цель работы:

Формирование практических навыков у студентов по созданию сетевого приложения TCP клиент-сервер приложения с использованием классов пакетов java.net.* и java.io.* языка программирования Java.

Теоретическая часть:

Создание TCP-клиента. Первой задачей клиента является связь с сервером. Объект Socket создается для установления соединения между сервером и клиентом. Для создания приложения клиентского сокета TCP выполняются следующие задачи:

1. Создание клиентского сокета с использованием объекта сокета.
2. Запись в сокет и чтение из него.
3. Завершение подключения.

Создание клиентского сокета. Объект клиентского сокета создается с помощью конструктора класса Socket, который получает два параметра: IP-адрес и номер порта, слышимые сервером.

Socket clientSocket = new Socket("127.0.0.1", 1001);

В приведенном выше фрагменте кода IP-адрес, равный 127.0.0.1, и порт 1001 определяют сокет, в котором сервер ожидает клиентских запросов.

Чтение и запись из сокета. Как только соединение между клиентом и сервером установлено, клиент отправляет запрос на сервер через сокет. Чтение и запись из сокета аналогичны чтению и записи из файла. Чтобы клиент мог взаимодействовать с сервером, необходимо сделать следующее:

Два объекта объявлены для классов PrintStream и BufferedReader. Эти объекты используются для чтения и записи из сокета.

```
DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer = 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 DataOutputStream записывать в сокет.

Еще один объект класса BufferdReader объявляется в клиентском приложении для установления соединения со стандартным вводом для передачи введенных данных на сервер. Следующий фрагмент кода используется для чтения данных из окна консоли:

```
BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
```

Этот фрагмент кода позволяет пользователю вводить данные с клавиатуры.

Для завершения подключения используется метод clientSocket.close().

```
package tcp;
import java.io.*;
import java.net.*;
class TCPClient {
  public static void main(String argv[]) throws Exception
  {
    String sentence;
    String modifiedSentence;
     BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
    System.out.println("Kliyent ishga tushdi!!!");
     Socket clientSocket = new Socket("localhost", 6789);
    System.out.println("Kliyent server bilan bog'landi");
    DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
    BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
    System.out.println("Serverga jo'natsh uchun ixtiyoriy matnni
kiriting:");
    sentence = inFromUser.readLine();
    outToServer.writeBytes(sentence + '\n');
    System.out.println("Kiritilgan matn serverga jo'natildi");
     modifiedSentence = inFromServer.readLine();
     System.out.println("Qayta ishlangan matn serverdan keldi: " +
modifiedSentence);
     clientSocket.close();
```

```
        System.out.println("Kliyent soketi yopildi!");
    }
}
```

Приведенный выше код хранится как TCPClient.java.

Создание TCP-сервера. Процесс создания сервера заключается в создании объекта класса ServerSocket, который «слушает» запросы клиентов на подключение от определенного порта. После того как сервер распознает разрешенный запрос, объект ServerSocket принимает объект Socket, созданный клиентом. Соединение между сервером и клиентом осуществляется с помощью этого сокета.

Класс ServerSocket пакета java.net.* используется для создания объекта, позволяющего серверу прослушивать запросы удаленного доступа. Класс BufferedInputStream управляет передачей данных от клиента к серверу, а класс DataOutputStream управляет передачей данных от сервера к клиенту.

Метод accept() ожидает подключения клиента, когда он слышит порт, к которому он подключен. Когда клиент пытается подключиться к серверному сокету, метод принимает соединение и возвращает клиентский сокет, который затем используется клиентом для связи с сервером. Выходной ток этого разъема является входным током для подключенного клиента и наоборот. Статус IOException генерируется при возникновении ошибки во время установки соединения. Java принудительно обрабатывает возникающие исключения.

Чтобы создать серверное приложение для TCP-сервера, необходимо сделать следующее:

- Создать сокет-объект сервера ServerSocket;
- Заслушивание запросов клиентов на подключение;
- Запуск сервера;
- Создание потока подключения для запросов клиентов.

Создать сервер. Объект ServerSocket "слушает" запросы клиентов, а конструктор класса Server создает объект ServerSocket. При запуске сервера

отображается сообщение об ошибке. Фрагмент кода конструктора сервера выглядит так:

```
        serverSocket = new ServerSocket(1001);}
```

```
package tcp;
import java.io.*;
import java.net.*;
class TCPServer
{
  public static void main(String argv[]) throws Exception
  {
    String clientSentence;
    String capitalizedSentence;
    ServerSocket welcomeSocket = new ServerSocket(6789);
    System.out.println("Server ishga tushdi!");
    System.out.println("Kliyentdan so'rovni kutmoqda...");
    while(true)
    {
      Socket connectionSocket = welcomeSocket.accept();
      System.out.println("Kliyent server bilan bog'landi");
      BufferedReader inFromClient = new BufferedReader(new
      InputStreamReader(connectionSocket.getInputStream()));
      DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
      clientSentence = inFromClient.readLine();
      System.out.println("Server kliyentdan so'rovni qabul qildi");
      System.out.println("Kliyentdan qabul qilingan matn: "+clientSentence);
      capitalizedSentence = clientSentence.toUpperCase() + '\n';
      outToClient.writeBytes(capitalizedSentence);
      System.out.println("Server qabul qilingan so'rovni qayta ishlab kliyentga
jo'natdi");
      System.out.println("Qayta ishlangan matn: " +capitalizedSentence);
    }
  }}
```

Приведенный выше код хранится как TCPServer.java.

Задание:

Студенты создают сетевое приложение TCP клиент-сервер, используя варианты, перечисленные в таблице 1.1.

Таблица 1.1

Варианты заданий

№	Задания
1	Найти площадь ромба
2	Рассчитать среднее арифметическое n чисел
3	Найти площадь прямоугольника
4	Рассчитать среднее геометрическое n чисел
5	Найдите площадь треугольника
6	Найти длину окружности произвольного радиуса R
7	Вычислить площадь поверхности шара произвольного радиуса R
8	Найти решение квадратного уравнения с произвольными коэффициентами
9	Сортировать n чисел в порядке возрастания
10	Сортировать n чисел в порядке убывания
11	Найти n степень числа m
12	Вычислить квадратный корень из произвольного числа n
13	Найти наибольшее из n чисел
14	Найти наименьшее из n чисел
15	Найти объем произвольного куба
16	Найти длину биссектрисы треугольника
17	Вычислить сумму любых n чисел
18	Вычислите сумму положительных чисел из любых n чисел
19	Выведите текст в обратном порядке
20	Найти объем произвольного куба
21	Выделяйте нечетные буквы в тексте
22	Вычислить объем шара R радиуса
23	Вычислить объем произвольного цилиндра
24	Вычислить диагональ прямоугольника
25	Найти поверхность шара радиуса R
26	Найти объем произвольного куба
27	Найти четные числа из n чисел
28	Найти среднее арифметическое n чисел
29	Найти среднее геометрическое n чисел
30	Найти n факториал

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки интегрированной среды Eclipse или NetBeans IDE.

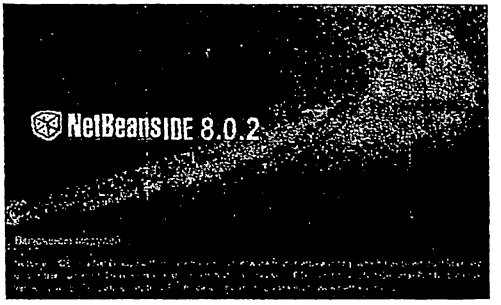


Рисунок 1.1. Процесс запуска NetBeans IDE

Изображение ниже загружает главное окно среды IDE NetBeans.

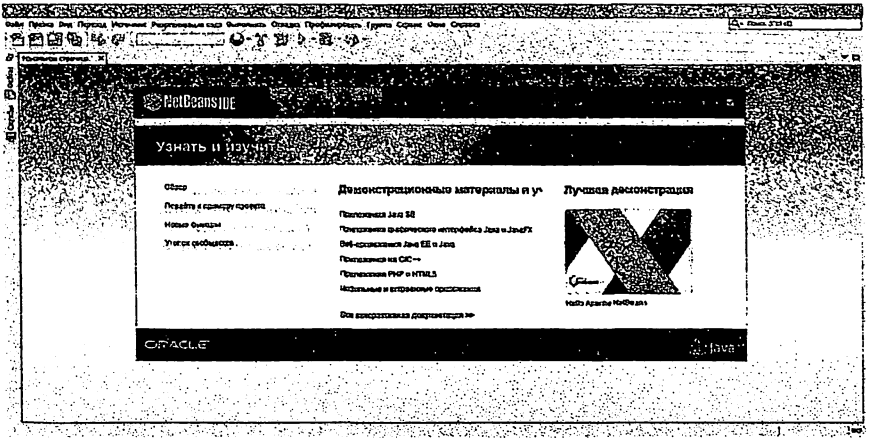


Рисунок 1.2. Главное окно среды NetBeans IDE

Шаг 2. При выборе раздела «Создать проект» в меню «Файл» появится следующее окно с изображением. В этом окне выберите Java в разделе «Категории» и нажмите «Далее».

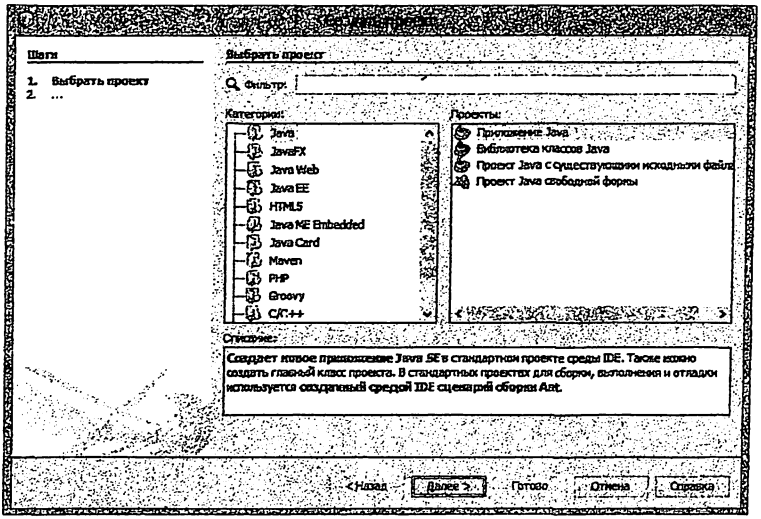


Рисунок 1.3. Окно «Создать проект» среды NetBeans IDE

Шаг 3. В поле «Имя проекта» в окне «Новый Приложение Java» введите «Имя студента» в качестве имени проекта, выберите место, где должен храниться проект, в поле «Расположение проекта», снимите флажок «Создать главный класс» и нажмите «Готово».

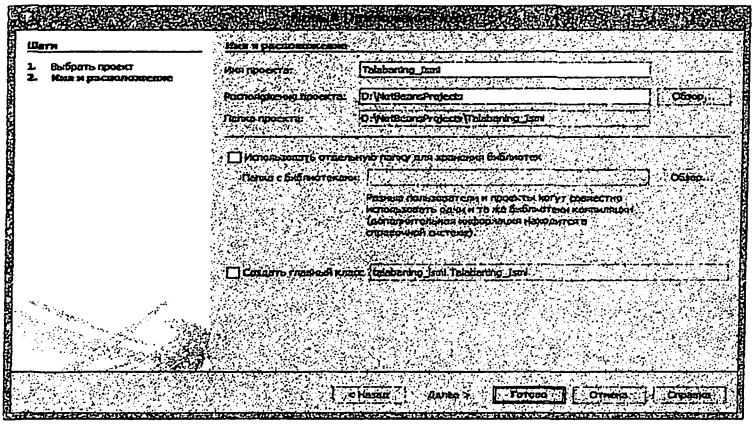


Рисунок 1.4. Окно «Новый Приложения Java» среды NetBeans IDE

Шаг 4. Щелкните правой кнопкой мыши над созданным проектом и выберите «Новый → Класс Java» во конекстном меню.

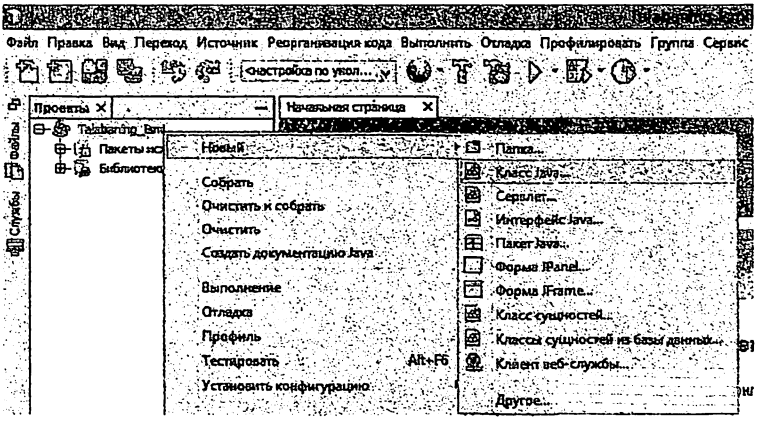


Рисунок 1.5. Создание новый класс в среде IDE NetBeans

Шаг 5. В окне «New Класс Java» введите TCPClient в поле «Имя класса», tcp в поле «Пакет» и нажмите «Готово». Код Java добавляется в созданный файл TCPClient.java. Аналогичный файл TCPServer.java создается и вводится код Java.

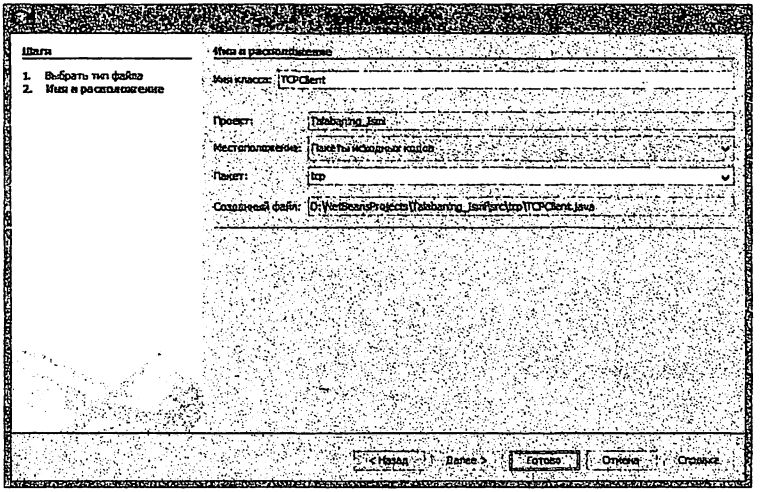


Рисунок 1.6. Окно «New Класс Java» среды NetBeans IDE

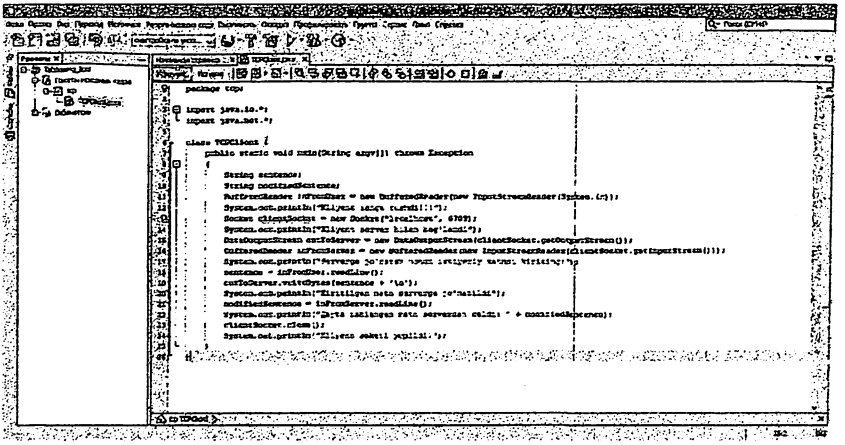


Рисунок 1.7. Файл TCPClient.java среды NetBeans IDE

Рисунок 1.8. Файл TCPServer.java среды NetBeans IDE

Контрольные вопросы

1. Понятие IP.
2. Понятие порта.
3. Понятие сокета.
4. Пакет java.net.*.
5. Пакет java.io.*
6. Класс Socket.
7. Класс InputStream.
8. Класс OutputStream.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Создание сетевого приложения UDP клиент-сервер

Цель работы:

Формирование практических навыков у студентов по созданию сетевого приложения UDP клиент-сервер приложения с использованием классов пакетов java.net.* и java.io.* языка программирования Java.

Теоретическая часть:

Создание UDP-сервера. UDP-сервер — это сетевое приложение, использующее протокол UDP для обслуживания клиентских приложений. Для создания UDP-сервера используется объект DatagramSocket, который принимает объекты DatagramPacket от клиентов. Чтобы создать UDP-сервер, вам необходимо сделать следующее:

- Создать сокет с помощью объекта DatagramSocket;
- Использовать метод receive() для создания объекта класса DatagramPacket и получения клиентских сообщений;
- Использовать метод send() для создания объекта класса DatagramPacket и передачи клиентских сообщений;
- В методе main() вызвать конструктор класса сервера UDP и запустить сервер;
- Объект DatagramPacket, который получает пакет Datagramma имеет буфер для хранения дейтаграмм.

Объект DatagramPacket, отправленный получателю, отличается от полученного объекта данных. Этот объект DatagramPacket содержит IP-адрес хоста, на который был отправлен пакет, и номер порта.

Создается новый объект пакета класса DatagramPacket, который принимает 4 параметра.

- buffer: обеспечивает буфер, содержащий данные;
- length: указывает длину буфера в байтах;
- address: возвращает адрес, на который будет отправлена дейтаграмма;

- port: Предоставляет номер порта, который удаленный компьютер использует для получения дейтаграммы.

Метод send() класса DatagramSocket отправляет объект DatagramPacket по адресу.

Для запуска UDP-сервера вызывается класс-конструктор метода main().

Ниже приведен код Java сервера UDP:

```
package udp;
import java.io.*;
import java.net.*;
class Server
{
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        System.out.println("Server ishga tushdi!");
        while(true)
          {
            DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String( receivePacket.getData());
            System.out.println("Qabul qilindi: " + sentence);
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            String capitalizedSentence = sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket =
            new DatagramPacket(sendData, sendData.length, IPAddress,
port);
            serverSocket.send(sendPacket);
          }        } }
```

Создание UDP-клиента. UDP клиент - это приложение, использующее протокол UDP для отправки запросов на сервер и получения ответов от серверного приложения. В приложении UDP вам нужно создать объект класса DatagramSocket, который получает сообщения от сервера UDP, и вам нужно сделать следующее:

1. Создать пользовательский сокет из объекта класса DatagramSocket

для подключения к серверу.

2. Использовать метод send() для создания объекта класса DatagramPacket и отправки сообщений на сервер.

3. Использовать метод receive() для создания объекта класса DatagramPacket и получения сообщений, отправленных с сервера.

Объект DatagramSocket содержит IP-адрес и номер порта сервера, на который отправляется запрос.

Ниже приведен код Java клиента UDP:

```
package udp;
import java.io.*;
import java.net.*;
class Client
{
  public static void main(String args[]) throws Exception
  {
    System.out.println("Client ishga tushdi!");
    BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
    DatagramSocket clientSocket = new DatagramSocket();
    InetAddress IPAddress = InetAddress.getByName("localhost");
    byte[] sendData = new byte[1024];
    byte[] receiveData = new byte[1024];
    String sentence = inFromUser.readLine();
    sendData = sentence.getBytes();
    DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 9876);
    clientSocket.send(sendPacket);
    DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
    clientSocket.receive(receivePacket);
    String modifiedSentence = new String(receivePacket.getData());
    System.out.println("SERVERDAN:" + modifiedSentence);
    clientSocket.close();
  }
}
```

Задание:

Студенты создают сетевое приложение UDP клиент-сервер, используя варианты, перечисленные в таблице 2.1.

Таблица 2.1

Варианты заданий

№	Задания
1	Вычислить площадь поверхности шара произвольного радиуса R
2	Найти решение квадратного уравнения с произвольными коэффициентами
3	Сортировать n чисел в порядке возрастания
4	Сортировать n чисел в порядке убывания
5	Найти n степень числа m
6	Вычислить квадратный корень из произвольного числа n
7	Найти наибольшее из n чисел
8	Найти наименьшее из n чисел
9	Найти объем произвольного куба
10	Найти длину биссектрисы треугольника
11	Вычислить сумму любых n чисел
12	Вычислите сумму положительных чисел из любых n чисел
13	Выведите текст в обратном порядке
14	Найти объем произвольного куба
15	Выделяйте нечетные буквы в тексте
16	Вычислить объем шара R радиуса
17	Вычислить объем произвольного цилиндра
18	Вычислить диагональ прямоугольника
19	Найти поверхность шара радиуса R
20	Найти объем произвольного куба
21	Найти четные числа из n чисел
22	Найти среднее арифметическое n чисел
23	Найти среднее геометрическое n чисел
24	Найти n факториал
25	Найти площадь ромба
26	Рассчитать среднее арифметическое n чисел
27	Найти площадь прямоугольника
28	Рассчитать среднее геометрическое n чисел
29	Найдите площадь треугольника
30	Найти длину окружности произвольного радиуса R

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE.

Шаг 2. В окне, которое появляется при выборе раздела «Открыть проект» в меню «Файл», выберите проект «Имя_студента» и нажмите кнопку «Открыть проект».

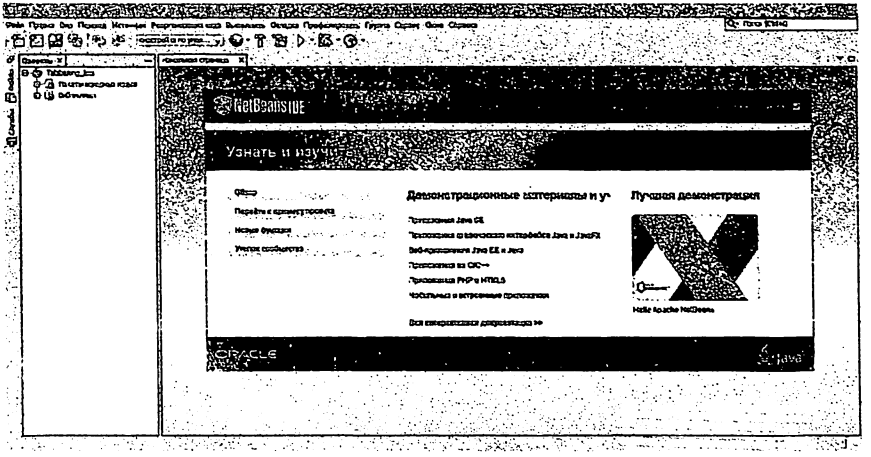


Рисунок 2.1. Главное окно среды NetBeans IDE

Шаг 3. Щелкните правой кнопкой мыши открытый проект «Имя студента», выберите «Создать» → «Класс Java» в контекстном меню.

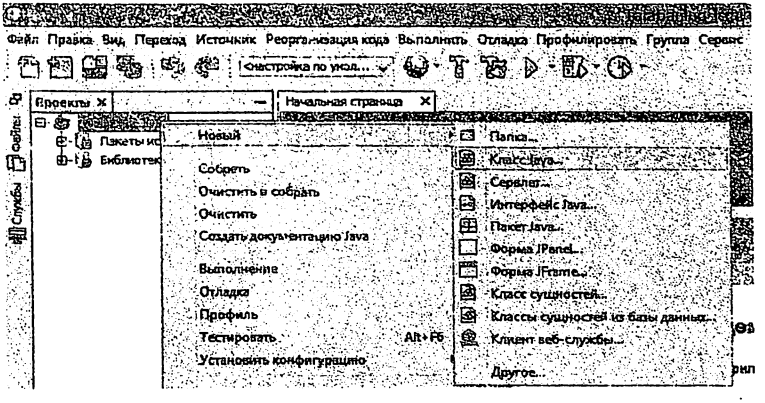


Рисунок 2.2. Создание нового класса в среде NetBeans IDE

Шаг 4. В окне «New Класс Java» введите «UDPClient» в поле «Имя класса», «udp» в поле «Пакет» и нажмите «Готово».

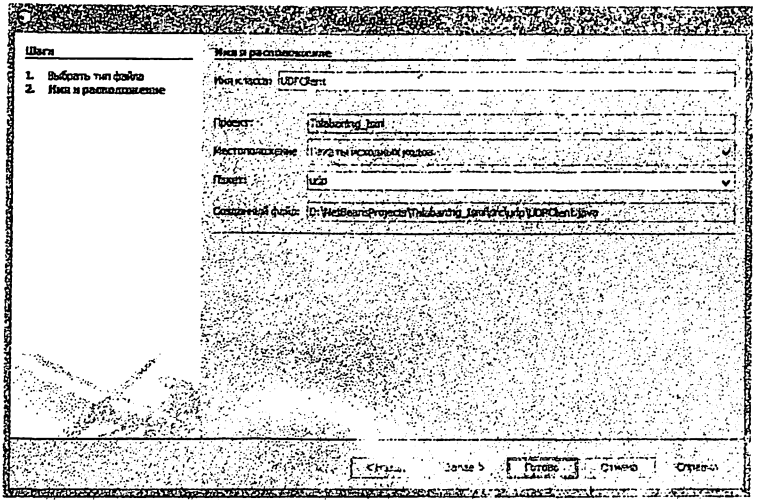


Рисунок 2.3. Окно «New Класс Java» среды NetBeans IDE

Шаг 5. Java-код UDP клиента добавляется в файл UDPClient.java внутри созданного пакета udp.

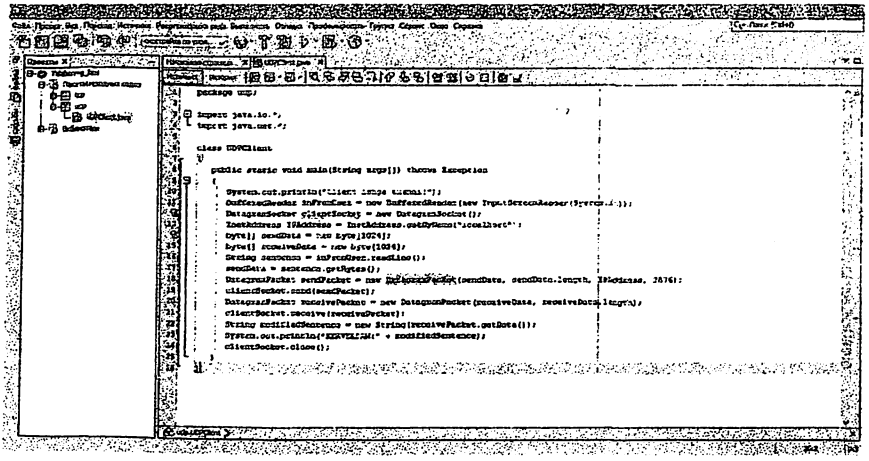


Рисунок 2.4. Файл UDPClient.java среды NetBeans IDE

Шаг 6. Файл UDPServer.java создается так же, как и файл UDPClient.java.

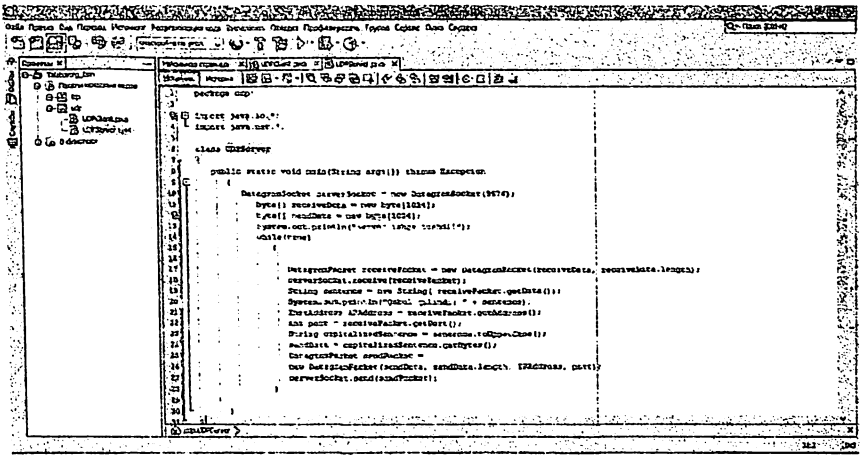


Рисунок 2.5. Файл UDPServer.java среды NetBeans IDE

Контрольные вопросы

1. Протокол UDP
2. Понятие IP.
3. Понятие порта.
4. Понятие сокет.
5. Пакет java.net.*.
6. Пакет java.io.*.
7. Класс DatagramSocket.
8. Класс DatagramPacket.
9. Сокет UDP сервер.
10. Сокет UDP клиент.
11. Методы send() и receive().

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Создание приложения на основе Multicast сокет

Цель работы:

Формирование практических навыков у студентов по созданию приложения на основе Multicast сокет с использованием классов пакетов java.net.* и java.io.* языка программирования Java.

Теоретическая часть:

Создание Multicast программы на основе протокола UDP. Java позволяет разрабатывать сетевые приложения с использованием дейтаграмм UDP и сокетов TCP. Сокеты UDP используют протокол UDP, чтобы приложения могли взаимодействовать по сети. UDP - это быстрый и ненадежный протокол, не требующий соединения. Пакет java.net включает следующие два класса, которые позволяют использовать сокет UDP в приложении Java:

- класс DatagramPacket;
- класс DatagramSocket.

Классы DatagramPacket и DatagramSocket. Объект DatagramPacket - это контейнер данных, содержащий пакеты дейтаграмм, которые передаются или принимаются по сети. Следующие конструкторы используются для инициализации объектов DatagramPacket:

public DatagramPacket (byte[] buffer, int buffer_length): создает объект DatagramPacket, который получает и сохраняет данные в массиве байтов. Длина буфера массива байтов задается вторым параметром buffer_length.

public DatagramPacket (byte[] buffer, int buffer_length, адрес InetAddress, int port): создает объект DatagramPacket, который передает пакеты данных заданной длины. Пакеты данных отправляются на компьютер с IP-адресом и номером порта, указанными в качестве параметра.

Класс DatagramSocket включает функции для управления объектами DatagramPacket. Объекты DatagramPacket передают и получают сохраненные

данные, используя DatagramSocket. Следующие конструкторы используются для инициализации объекта DatagramSocket:

- *public DatagramSocket():* создает объект DatagramSocket и связывает его с разрешенным портом на локальном компьютере.

- *public DatagramSocket (int port):* создает объект и связывает его с портом на локальном хосте, указанном в параметре.

- *public DatagramSocket (порт int, адрес InetAddress):* создает объект и связывает его с заданным портом хоста.

Java код приемника Multicast:

```
package multicast;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
import java.net.UnknownHostException;
public class Receiver {
   final static String INET_ADDR = "224.0.0.3";
   final static int PORT = 8888;
   public static void main(String[] args) throws UnknownHostException {
      InetAddress address = InetAddress.getByName(INET_ADDR);
      byte[] buf = new byte[256];
      try (MulticastSocket clientSocket = new MulticastSocket(PORT)){
         clientSocket.joinGroup(address);
         while (true) {
            DatagramPacket msgPacket = new DatagramPacket(buf, buf.length);
            clientSocket.receive(msgPacket);
            String msg = new String(buf, 0, buf.length);
            System.out.println("Socket 1 received msg: " + msg);
         }
      } catch (IOException ex) {
         ex.printStackTrace();
      }
```

Код Java для Multicast отправителя:

```
package multicast;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class Sender {

  final static String INET_ADDR = "224.0.0.3";
  final static int PORT = 8888;

  public static void main(String[] args) throws UnknownHostException,
InterruptedException {
    InetAddress addr = InetAddress.getByName(INET_ADDR);
    try (DatagramSocket serverSocket = new DatagramSocket()) {
      for (int i = 0; i < 5; i++) {
        String msg = "Sent message no " + i;
        DatagramPacket msgPacket = new
DatagramPacket(msg.getBytes(),msg.getBytes().length, addr, PORT);
        serverSocket.send(msgPacket);
        System.out.println("Server sent packet with msg: " + msg);
        Thread.sleep(500);
      }
    } catch (IOException ex) {
      ex.printStackTrace();
    }
  } }
```

Задание:

Студенты создают приложение на основе Multicast сокет, используя варианты, перечисленные в таблице 3.1.

Таблица 3.1

Варианты заданий

№	Задания
1	Найти n степень числа m
2	Вычислить квадратный корень из произвольного числа n
3	Найти наибольшее из n чисел
4	Найти наименьшее из n чисел
5	Найти объем произвольного куба
6	Найти длину биссектрисы треугольника
7	Вычислить сумму любых n чисел
8	Вычислите сумму положительных чисел из любых n чисел
9	Выведите текст в обратном порядке
10	Найти объем произвольного куба
11	Выделяйте нечетные буквы в тексте
12	Вычислить объем шара R радиуса
13	Вычислить объем произвольного цилиндра
14	Вычислить диагональ прямоугольника
15	Найти поверхность шара радиуса R
16	Найти объем произвольного куба
17	Найти четные числа из n чисел
18	Найти среднее арифметическое n чисел
19	Найти среднее геометрическое n чисел
20	Найти n факториал
21	Найти площадь ромба
22	Рассчитать среднее арифметическое n чисел
23	Найти площадь прямоугольника
24	Рассчитать среднее геометрическое n чисел
25	Найдите площадь треугольника
26	Найти длину окружности произвольного радиуса R
27	Вычислить площадь поверхности шара произвольного радиуса R
28	Найти решение квадратного уравнения с произвольными коэффициентами
29	Сортировать n чисел в порядке возрастания
30	Сортировать n чисел в порядке убывания

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки производственной среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе. Изображение ниже приведено главное окно среды NetBeans IDE.

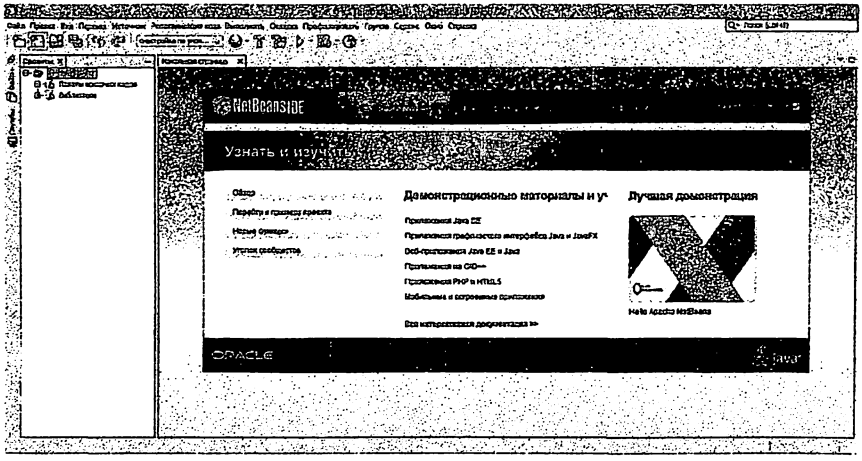


Рисунок 3.1. Главное окно среды NetBeans IDE

Шаг 2. Щелкните проект правой кнопкой мыши, выберите «Создать» → «Класс Java...» в контекстном меню.

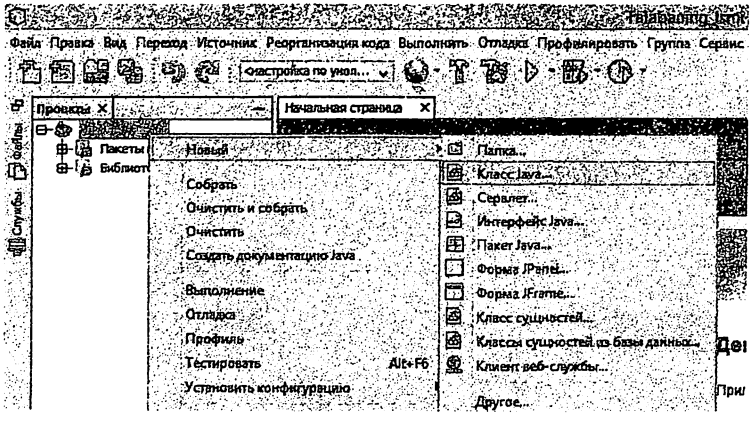


Рисунок 3.2. Создание нового класса в среде NetBeans IDE

Шаг 3. В окне «New Класс Java» введите «Receiver» в поле «Имя класса», «multicast» в поле «Пакет» и нажмите кнопку «Готово».

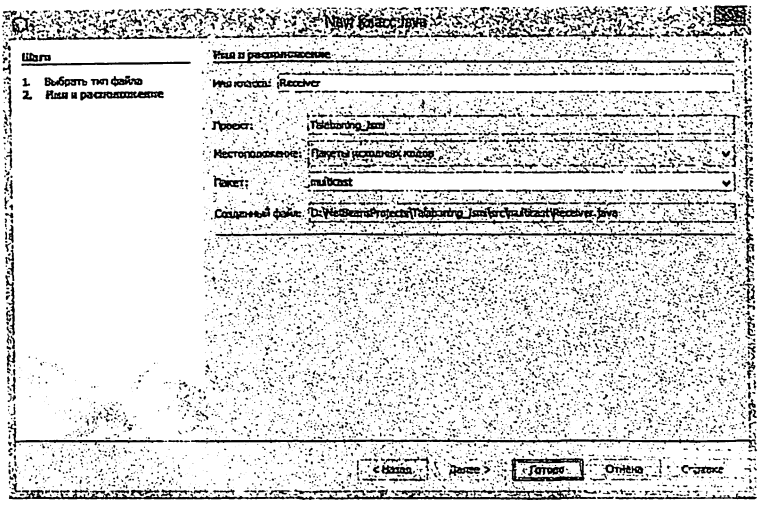


Рисунок 3.3. Окно «New Класс Java» среды NetBeans IDE

Шаг 4. В созданный файл Receiver.java вводится Java код Multicas получателя.

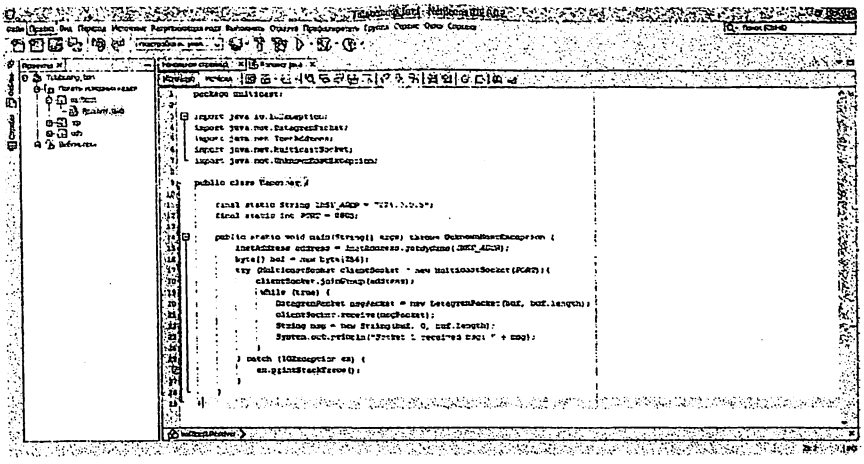


Рисунок 3.4. Файл Receiver.java среды NetBeans IDE

Шаг 5. Аналогичный класс Sender создается и включает Java ко отправителя.

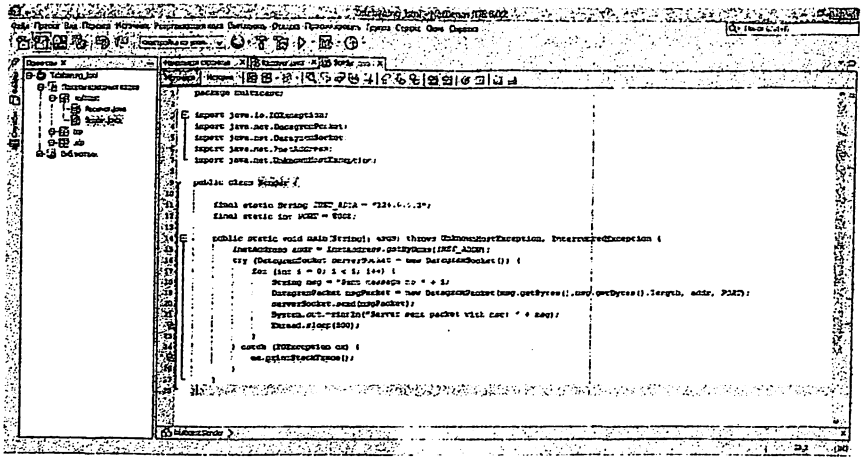


Рисунок 3.5. Файл Sender.java среда NetBeans IDE

Контрольные вопросы

1. Протокол UDP.
2. Multicast IP-адрес.
3. Понятие Multicast сокета.
4. Пакет java.net.*.
5. Пакет java.io.*
6. Класс DatagramSocket.
7. Класс DatagramPacket.

ЛАБОРАТОРНАЯ РАБОТА № 4

Тема: Работа с интернет-адресами и форматом обмена данными JSON

Цель работы:

Формирование практических навыков у студентов по работе с языком программирования Java пакета java.net.* URL-класса и его методы, а также формат обмена данными JSON.

Теоретическая часть:

URI — универсальный идентификатор ресурса.

URL — унифицированный указатель ресурсов.

URN — универсальное имя ресурса.

Анализ. URI: указывает имя и адрес ресурса в сети. Как правило, он делится на URL и URN, поэтому URI состоит из URL и URN.

URL: веб-адрес ресурса. URL-адрес указывает местоположение ресурса и способ доступа к нему.

URN: имя ресурса в Интернете. Смысл URN в том, что он относится только к названию конкретного объекта, который находится во многих конкретных местах.

URI — это абстрактное понятие идентификатора, а URL и URN — это реальные приложения адреса и имени.

Формат запроса/ответа. Форма запроса клиента. Клиент-клиент отправляет запрос на сервер после установления соединения. Этот запрос может принимать две формы: полную и простую. Простой запрос включает в себя метод ввода и запрос ресурсов. Например:

GET http://polyn.net.kiae.su/

Здесь слово GET означает метод доступа, а http://polyn.net.kiae.su/ — запрос ресурса. Клиенты, которые могут поддерживать протокол выше версии 0.9, должны использовать полную форму запроса. При использовании полной формы запрос отображает строку запроса, несколько заголовков (заголовок запроса или общий заголовок) и, возможно, основную часть, представляющую ресурс. Обзор полной съемки Бекус-Наур выглядит следующим образом:

<Полный запрос>:= <строка запроса> (<Общее название>|<Название запроса>|<Название обозначающее ресурс>)<символ новой строки >[<тело ресурса>]

Здесь квадратные скобки обозначают ненужные элементы заголовка. Строка запроса представляет собой запрос ресурсов. Отличие в том, что в

строке запроса могут быть указаны разные способы входа в систему, а версия протокола должна быть указана после запроса ресурса. Например, вы можете использовать следующую подсказку для вызова внешнего приложения:

POST http://polyn.net.kiae.su/cgi-bin/test HTTP/1.0

В этом случае используется метод POST и протокол версии 1.0.

Ответ сервера. Ответ сервера может быть простым или полным, как запрос. В упрощенном ответе сервер возвращает только тело сервера (например, текст HTML-документа). В полном ответе клиенту возвращается строка состояния, общий заголовок, заголовок ответа, заголовок ресурса и тело ресурса. Полный ответ Бекуса-Наура выглядит следующим образом:

<полный ответ> := <статус строки> (<общее название>|<ответ названия>|<название ресурса>) <символ новой строки>[<тело ресурса>]

Строка состояния содержит версию протокола, код возврата и краткое описание кода. Например, это может выглядеть так:

"HTTP/1.0 200 Success"

Заголовок ответа сервера может содержать URI-адрес запрашиваемого ресурса и/или имя серверного приложения, и/или идентификационный код кода безопасности. Содержимое строк заголовка ресурса является общим для запроса клиента и ответа сервера, доступа к методу доступа, типа кодирования тела ресурса (содержимое ресурса), содержимого тела ресурса, типа ресурса, даты истечения срока действия данной копии ресурса, последнего времени использования ресурса. состоит из времени на изменение и расширение заголовка.

Класс URL находится в пакете java.net.*.

Методы класса URL: getFile(), getHost(), getPort(), getDefaultPort(), getProtocol(), getRef(), getQuery(), getPath(), getUserInfo(), getAuthority().

JSON или JavaScript Object Notation — это облегченный стандарт с открытым исходным кодом для обмена удобочитаемой информацией.

JavaScript Object Notation — это открытый формат файла и формат обмена данными, в котором используется удобочитаемый текст для хранения

и передачи объектов данных, состоящих из пар атрибут-значение и типов данных массива.

Возможности JSON используют программисты C, C++, Java, Python, Perl и других языков:

– Формат создан Дугласом Крокфордом;

– Он предназначен для обмена удобочитаемой информацией;

– Расширенный язык сценариев JavaScript;

– Расширение имени файла .json.

– Использование формата обмена данными JSON:

– Используется для написания программ на основе JavaScript, включая расширения браузера и веб-сайты;

– Использование формата обмена данными JSON:

– Используется для написания программ на основе JavaScript, включая расширения браузера и веб-сайты;

– формат JSON используется для сериализации и передачи структурированных данных по сети;

– В основном используется для передачи данных между сервером и веб-приложениями;

– веб-службы и API используют формат JSON для предоставления общей информации;

– Может использоваться в современных языках программирования.

Описание формата обмена данными JSON:

– JSON легко читать и писать;

– JSON — легкий текстовый формат обмена;

– JSON не зависит от языка программирования (независимый).

Синтаксис JSON в основном является частью синтаксиса JavaScript и включает в себя:

– Данные задаются в парах имя/значение;

– Фигурные скобки содержат объекты, а пары ':' имя/значение,

разделенные двоеточием, разделяются запятыми;

– Квадратные скобки содержат массивы, а значения разделяются запятыми.

Типы данных в формате обмена данными JSON:

- Число (Number)

Integer: 1-9 цифры, 0 и положительный или отрицательный

Fraction: .3, -.9 дроби

Exponent: e, e+, e-, E, E+, E-

синтаксиси:

var json-object-name = { string : number_value, ...}

Пример: var obj = {marks: 97}

- Строка (String)

var obj = {name: 'Amit'}

- Логический (Boolean) - true или false

var obj = {distinction: true}

- Массив (Array) – порядковая строка значений

```
{  "books": [
    { "language":"Java" , "edition":"second" },
    { "language":"C++" , "lastName":"fifth" },
    { "language":"C" , "lastName":"third" }
  ] }
```

- Объект - ключ: нерегулярный набор парных значений

- Null – пуст

Задание:

Студентам будут даны практические задания по интернет-адресам и форматам обмена данными JSON.

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе. Изображение ниже представляет главное окно среды NetBeans IDE.

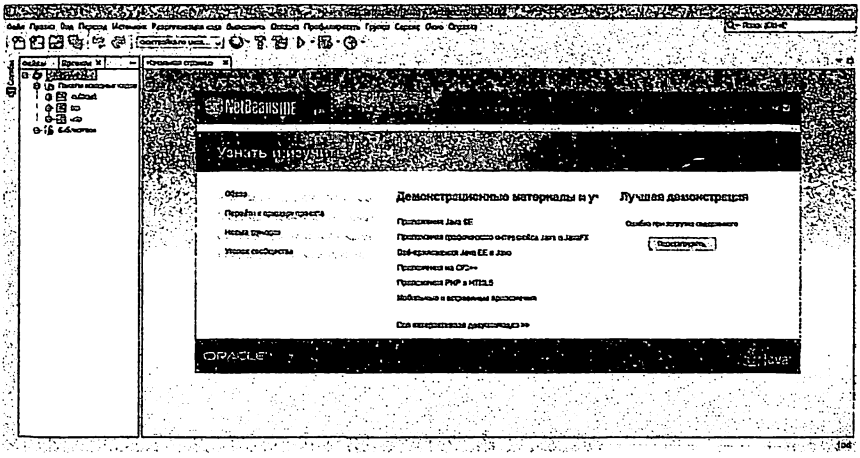


Рисунок 4.1. Главное окно среды NetBeans IDE

Шаг 2. Щелкните проект правой кнопкой мыши, выберите «Создать» → «Класс Java» в контекстном меню.

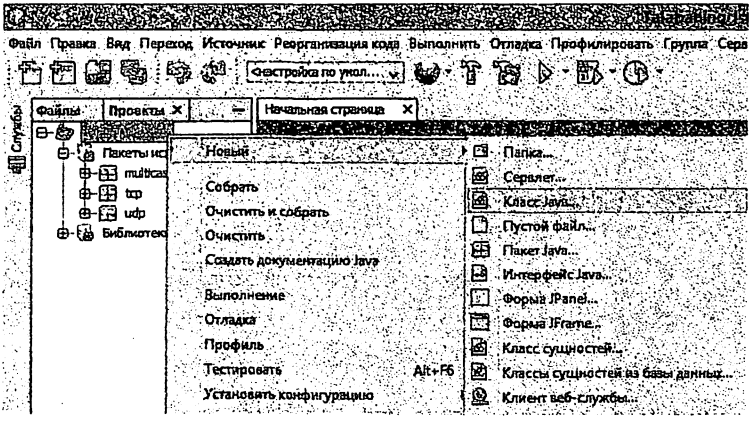


Рисунок 4.2. Создание нового класса в среде NetBeans IDE

Шаг 3. В окне "New Класс Java" введите «URLDemo» в поле «Имя класса», «url» в поле «Пакет» и нажмите «Готово».

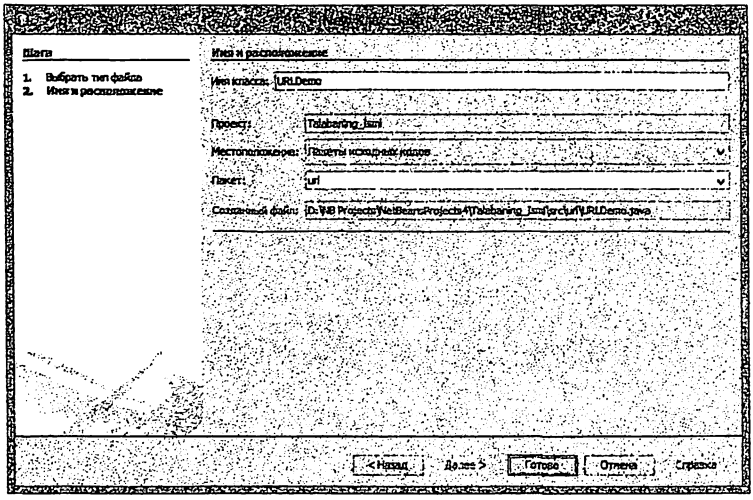


Рисунок 4.3. Окно «New Класс Java» среды NetBeans IDE

Шаг 4. Java код вводится в созданный файл URLDemo.java.

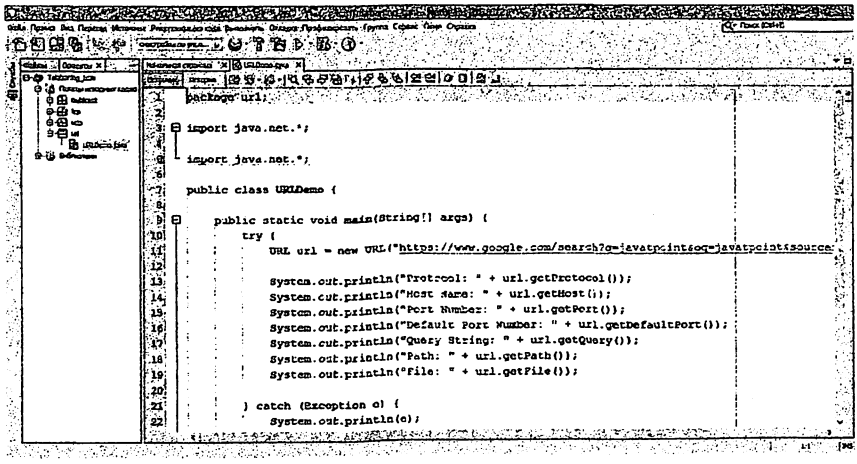


Рисунок 4.4. Файл URLDemo.java среда NetBeans IDE

Контрольные вопросы

1. URI – Uniform Resource Identifiers.
2. URN – Uniform Resource Names.
3. URL – Uniform Resource Locators.
4. Класс URL и его методы.
5. Доменные имена в Интернете.
6. Общая информация JSON
7. Типы данных JSON
8. Описание JSON.

ЛАБОРАТОРНАЯ РАБОТА №5

Тема: Работа с гипертекстами в сети

Цель работы:

Формирование практических навыков у студентов по работе с гипертекстами в сети.

Теоретическая часть:

HTML-формы создаются пользователями для ввода данных и последующего использования введенных данных. Формы могут состоять из управляемых элементов, таких как текстовые окна, кнопки, флажки и меню. Формы помещаются между тегами <form> и </form>.

В некоторых случаях происходит активная коммуникация между пользователем и браузером, то есть ввод и редактирование данных. В этом случае в HTML-документе используется раздел Формы. Пользователь заполняет вышеуказанные формы информацией и отправляет ее на сервер для обработки. Сервер может быть веб-сервером или почтовым сервером. Итак, основной частью любой формы являются ее управляющие элементы. Каждый из этих элементов имеет свое имя, и, кроме того, эти элементы

имеют свое начальное значение и текущее значение. Начальное значение элемента отображается на экране при его первом создании или переустановке (инициализации). Затем пользователь может изменить эти значения. Значения заполненной пользователем формы отправляются на обработку в виде пары «имя = значение». Формы создаются с помощью элемента form. Обзор:

Синтаксис: <FORM>…</FORM> (Блочный элемент)

Аттрибуты: id, class, style, title, lang, dir

action = URL (URL, указанный для обработки форм)

method = GET | POST (Метод HTTP для отправки данных в формах)

enctype = тип файла (для ссылки на форму)

accept = типы файлов (возможные типы файлов)

accept-charset = кодировки (возможные методы кодировки символов)

name = CDATA (имя формы для скрипта)

target = фрейм (фрейм для генерации результатов)

onsubmit = сценарий (отправка формы)

onreset = сценарий (обнуление формы)

Синтаксический вид формы. Все формы начинаются с тега <FORM> и заканчиваются тегом </FORM>. <FORM METHOD="get|post" ACTION="URL">

Элементы формы и другие элементы HTML </FORM>

Атрибут METHOD. Как отправить данные формы. Существует два способа отправки информации:

– GET: информация о форме добавляется в конец URL-адреса, указанного в заголовке формы.

– POST: сразу отправляет всю информацию на URL, указанный в заголовке формы (невидимый).

Атрибут ACTION . Форма ACTION указывает, на какой URL отправлять данные.

Теги формы. TEXTAREA. Тег <TEXTAREA> (обычный текст)

используется для ввода пользователем многострочной информации.

Пример использования тега <TEXTAREA>:

<TEXTAREA NAME="address" ROWS=10 COLS=50>

Текст1

Текст2

</TEXTAREA>

Атрибуты, используемые в <TEXTAREA>. Даны внешний вид и имя.

- NAME - имя объекта ввода данных.

- ROWS - высота входного объекта.

- COLS - ширина входного объекта.

Тег <INPUT> используется для ввода одной строки информации. Атрибуты тега:

Элементы формы и их использование.

Флаги (checkboxes). Флаг — это элемент, который имеет два состояния (выбрано или не выбрано). Флаги создаются с помощью элемента ввода.

Коннекторы (radiobuttons). Коннекторы похожи на флаги, но с той лишь разницей, что все элементы имеют одинаковое имя, и выбран один из них. Соединители также организованы с использованием ввода.

Меню (menus). Меню позволяет пользователю выбрать один или несколько из нескольких вариантов. Меню организовано с использованием элемента select вместе с опциями.

Ввод текста (text input). Есть два элемента, которые заставляют текст вводить пользователь. Элемент input обеспечивает однострочный ввод. textarea — используется для выполнения многострочного ввода.

Селектор файлов (file select). Этот пункт позволяет выбрать файл для отправки на сервер. Селектор файлов также реализован с использованием элемента ввода.

Элемент ввода данных INPUT. Обзор:

Синтаксис: <INPUT> (без закрывающего тега)

Атрибуты: id, class, style, title, lang, dir, event.

type = TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT | RESET | FILE | HIDDEN | IMAGE | BUTTON (Тип ввода)

name = имя элемента

value = начальное значение проверенного

checked = CHECKED (Статус выбора флагов и коннекторов)

readonly = READONLY (Флаги и ярлыки, выбранные для текста и пароля, доступны только для чтения)

disabled = DISABLED (Элемент запрещён)

size = начальная ширина элемента

maxlength = число (максимальное количество символов в тексте)

onfocus = Происходит, когда элемент находится в фокусе

onblur = Происходит, когда элемент теряет фокус

onselect = Когда текст элемента выделен

onchange = При изменении стоимости предмета

password – поле ввода пароля, которое отличается от поля ввода текста, заменяется введенными здесь символами *, а введенные символы не отображаются на экране. Однако введенный текст отправляется на сервер в виде обычного текста без какой-либо кодировки, поэтому с помощью этого элемента не рекомендуется отправлять конфиденциальную информацию (номера кредитных карт).

reset – кнопка отмены формы. Нажатие этой кнопки возвращает все значения в форме к значению по умолчанию.

submit – кнопка отправки формы. При нажатии этой кнопки значения в форме отправляются на сервер как «value = значение». Если в форме есть несколько кнопок отправки, они должны быть названы с использованием атрибута name. Это связано с тем, что программа обработки данных должна определить, какая кнопка была нажата.

image – графическая кнопка для отправки формы. Здесь атрибут src используется для указания адреса графического изображения, а клавиша alt используется для установки альтернативного текста кнопки. Координаты

места нажатия кнопки мыши отправляются на сервер с помощью переменных name.x и name.y, где name — это имя кнопки.

button – кнопка обычного типа. Атрибут value присваивается текст кнопки. Атрибут onclick дается скрипт, который должен срабатывать при нажатии кнопки.

Задание:

Студентам будут даны практические задания по работе с гипертекстами в сети.

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе. Изображение ниже представляет главное окно среды NetBean IDE.

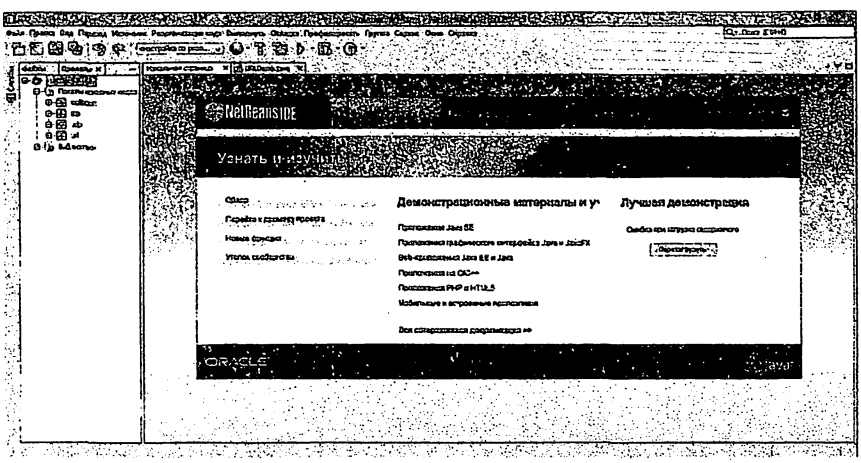


Рисунок 5.1. Главное окно среды NetBeans IDE

Шаг 2. Щелкните проект правой кнопкой мыши, выберите «Создать» → «Пустой файл» в контекстном меню.

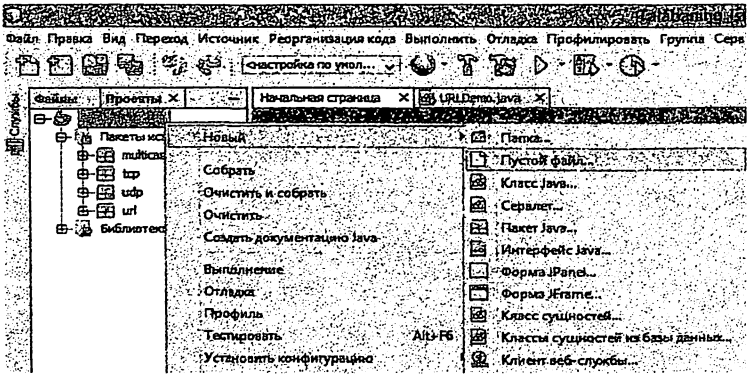


Рисунок 5.2. Создание нового класса в среде NetBeans IDE

Шаг 3. В окне «New Пустой файл» введите index.html в поле «Имя файла» и нажмите «Готово».

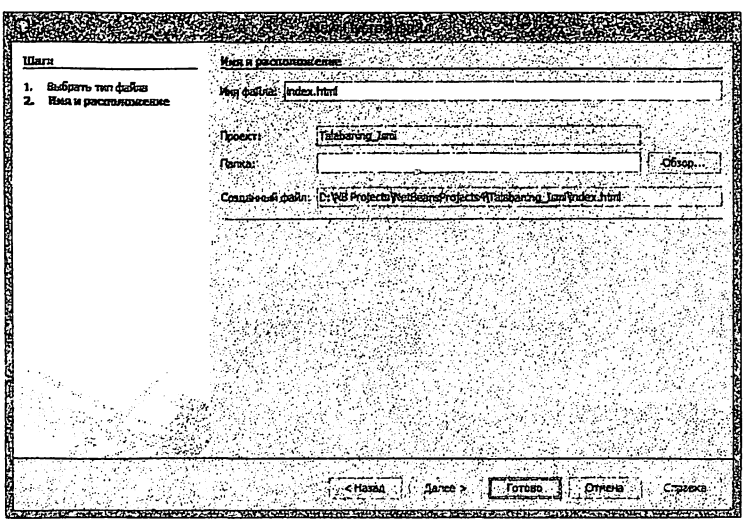
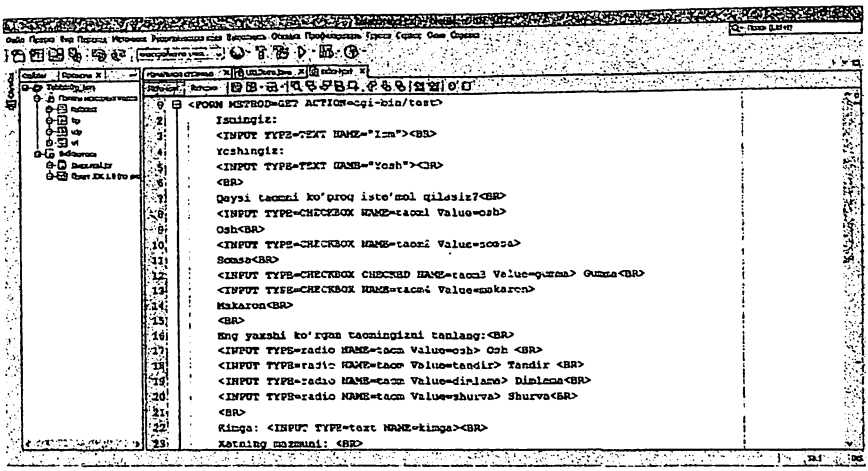


Рисунок 5.3. Окно «New Пустой файл» среды NetBeans IDE

Шаг 4. Код HTML вставляется в созданный файл index.html.



```
<FORM METHOD=GET ACTION=cgi-bin/test>
    Ismingiz:
    <INPUT TYPE=TEXT NAME="Ism"><BR>
    Yoshingiz:
    <INPUT TYPE=TEXT NAME="Yosh"><BR>
    <BR>
    Qaysi taomni ko'proq iste'mol qilasiz?<BR>
    <INPUT TYPE=CHECKBOX NAME=taom1 Value=osh>
    Osh<BR>
    <INPUT TYPE=CHECKBOX NAME=taom2 Value=somsa>
    Somsa<BR>
    <INPUT TYPE=CHECKBOX CHECKED NAME=taom3 Value=gumma> Gumma<BR>
    <INPUT TYPE=CHECKBOX NAME=taom4 Value=makaron>
    Makaron<BR>
    <BR>
    Eng yaxshi ko'rgan taomingizni tanlang:<BR>
    <INPUT TYPE=radio NAME=taom Value=osh> Osh <BR>
    <INPUT TYPE=radio NAME=taom Value=tandir> Tandir <BR>
    <INPUT TYPE=radio NAME=taom Value=dimlama> Dimlama<BR>
    <INPUT TYPE=radio NAME=taom Value=shurva> Shurva<BR>
    <BR>
    Kimga: <INPUT TYPE=text NAME=kimga><BR>
    Xatning mazmuni: <BR>
```

Рисунок 5.4. Файл index.html среде NetBeans IDE

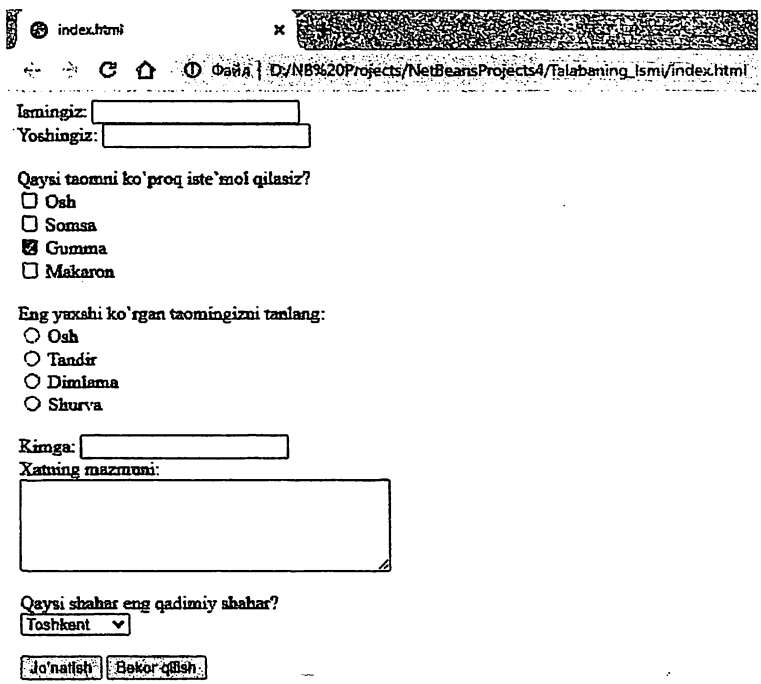


Рисунок 5.5. Результат

Контрольные вопросы

1. Hypertext Transfer Protocol.
2. Понятие гипертекста.
3. Формы HTML.
4. Поля ввода.
5. Кнопки Submit и Reset.
6. Кнопки Checkbox.
7. Кнопки Radio Button.
8. TEXTAREA example.
9. Тег SELECT.

ЛАБОРАТОРНАЯ РАБОТА №6

Тема: Создание сетевого приложения для передачи файлов

Цель работы:

Формирование практических навыков у студентов по созданию сетевого приложения для передачи файлов с использованием классов пакетов java.net.* и java.io.* языка программирования Java.

Теоретическая часть:

FTP — это стандартный механизм, предоставляемый TCP/IP для копирования файлов с одного хоста на другой. Этот протокол устанавливает два типа связи между хостами: команды (порт 21) и передача данных (порт 20).

FTP-клиент в основном состоит из 3 компонентов:

- внешний интерфейс;
- процесс управления;
- процесс передачи данных.

FTP-сервер состоит из 2 компонентов:

- процесс управления;

- процесс передачи данных.

Существует два шага для создания управляющего соединения:

1. Порт 21 сервера всегда пассивно открыт, ожидая клиентского запроса.

2. Клиент активирует 21 порт на сервере, используя временный порт.

Создание канала передачи данных отличается от создания канала управления тремя этапами:

1. Клиент находится в пассивном состоянии, используя временный порт.

2. Номер порта отправляется на сервер по каналу управления.

3. Сервер получает номер порта и переводит 20 номеров портов в активное состояние.

Команды FTP-клиента записываются заглавными буквами в формате ASCII.

Команды можно разделить на 6 групп:

- команды обращения;
- команды управления файлами;
- команды форматирования данных;
- команды обнаружения портов;
- команды для передачи файла;
- различные команды.

Каждая команда FTP генерирует как минимум один ответ, который генерирует трехзначное число xyz, которое, в свою очередь, состоит из двух частей: числовой части (код) и текстовой части (дополнительный комментарий).

Первое число - это одна из зависимостей 1-5, определяющая статус команды:

- 1yz - активность запущена. Отправляет отдельный ответ перед принятием каждого отдельного заказа;

- 2yz - действие завершено. Сервер получает другую команду;

- 3yz - команда принята, но требуется дополнительная информация;

- 4yz - никаких действий, а временная ошибка;

- 5yz - заказ не принят и повторяться не должен.

Java код клиентской части FTP:

```
package ftpclient;
import java.net.*;
import java.io.*;
class FTPClient
{
  public static void main(String args[]) throws Exception
  {
    Socket soc=new Socket("127.0.0.1",8888);
    transferfileClient t=new transferfileClient(soc);
    t.displayMenu();

  } }
class transferfileClient
{
  Socket ClientSoc;
  DataInputStream din;
  DataOutputStream dout;
  BufferedReader br;
  transferfileClient(Socket soc)
  {
    try
    {
      ClientSoc=soc;
      din=new DataInputStream(ClientSoc.getInputStream());
      dout=new DataOutputStream(ClientSoc.getOutputStream());
      br=new BufferedReader(new InputStreamReader(System.in));
    }
    catch(Exception ex)
    { } }
  void SendFile() throws Exception
  {
    String filename;
    System.out.print("Введите название файла :");
    filename=br.readLine();

    File f=new File(filename);
    if(!f.exists())
    {
```

```
        System.out.println("Файл не существует...");
        dout.writeUTF("Fayl topilmadi");
        return;
      }
      dout.writeUTF(filename);
      String msgFromServer=din.readUTF();
      if(msgFromServer.compareTo("Файл заранее существует")==0)
      {
        String Option;
    System.out.println("Файл уже существует. Вы хотите переписать
(Y/N) ?");
        Option=br.readLine();
        if(Option=="Y")
        { dout.writeUTF("Y"); }
        else
        { dout.writeUTF("N");
          return; } }
      System.out.println("Отправка файла ...");
      FileInputStream fin=new FileInputStream(f);
      int ch;
      do
      { ch=fin.read();
        dout.writeUTF(String.valueOf(ch));  }
      while(ch!=-1);
      fin.close();
      System.out.println(din.readUTF());
    }
    void ReceiveFile() throws Exception
    {
      String fileName;
      System.out.print("Введите название файла :");
      fileName=br.readLine();
      dout.writeUTF(fileName);
      String msgFromServer=din.readUTF();

      if(msgFromServer.compareTo("Файл не найден")==0)
      {
        System.out.println("Файл не найден на сервере...");
        return;      }
      else if(msgFromServer.compareTo("ГОТОВО")==0)
      {
        System.out.println("Принятие файла ...");
        File f=new File(fileName);
        if(f.exists())
        {           String Option;
```

```
System.out.println("Файл уже существует. Вы хотите переписать (Y/N)
?");
            Option=br.readLine();
            if(Option=="N")
            {  dout.flush();
               return;    }  }
          FileOutputStream fout=new FileOutputStream(f);
          int ch;
          String temp;
          do
          {
             temp=din.readUTF();
             ch=Integer.parseInt(temp);
             if(ch!=-1)
             { fout.write(ch);                    }
          }while(ch!=-1);
          fout.close();
          System.out.println(din.readUTF());     }     }
    public void displayMenu() throws Exception
    {
      while(true)
      {
        System.out.println("[ МЕНЮ ]");
        System.out.println("1. Отправка файла");
        System.out.println("2. Пирнятие файла");
        System.out.println("3. Выйти");
        System.out.print("\nВведите свой выбор:");
        int choice;
        choice=Integer.parseInt(br.readLine());
        if(choice==1)
        {
          dout.writeUTF("SEND");
          SendFile();     }
        else if(choice==2)
        {  dout.writeUTF("GET");
          ReceiveFile();     }
        else
        {  dout.writeUTF("DISCONNECT");
          System.exit(1); }
      }  }  }
```

Задание:

Студент получает индивидуальное задание на лабораторную работу. Он создает сетевое приложение для передачи файлов.

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Изображение ниже представляет главное окно среды NetBeans IDE.

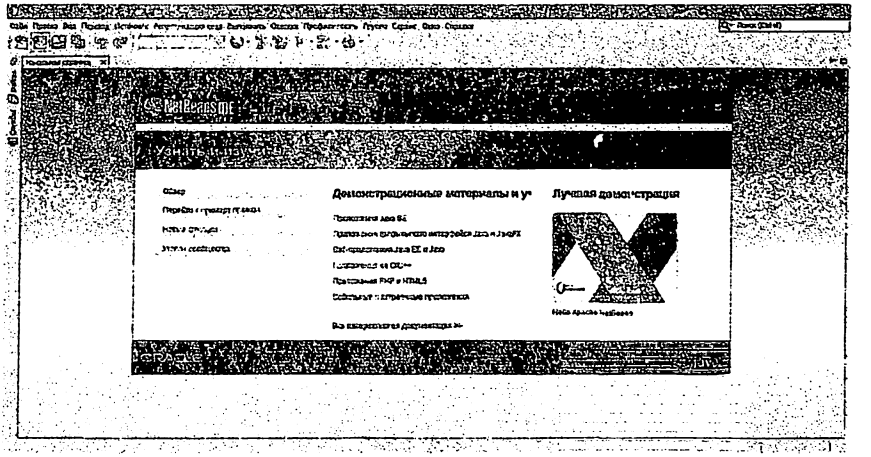


Рисунок 6.1. Главное окно среды NetBeans IDE

Шаг 2. При выборе раздела «Создать проект» в меню «Файл» появится следующее окно. В этом окне выберите Java в разделе «Категории» и нажмите «Далее».

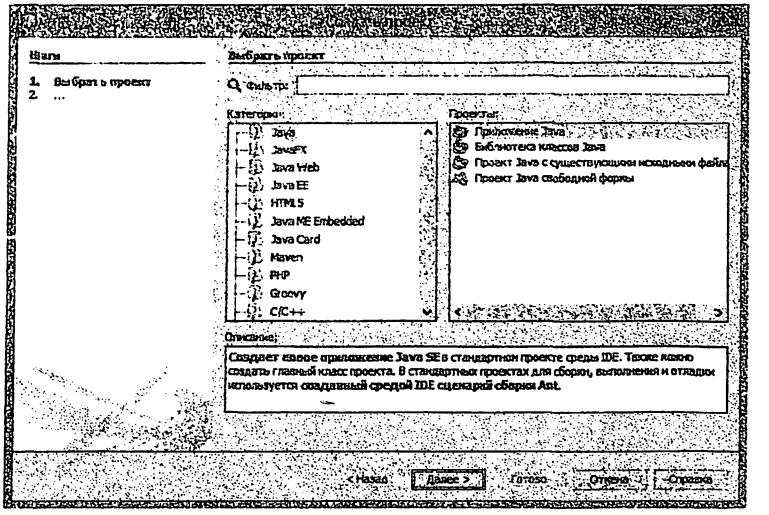


Рисунок 6.2. Окно «Создать проект» среды NetBeans IDE

Шаг 3. Перейдите в окно «Новое приложение Java». В поле «Имя проекта» этого окна введите имя проекта «FTPClient», выберите место, где должен храниться проект, в поле «Расположение проекта» и нажмите «Готово».

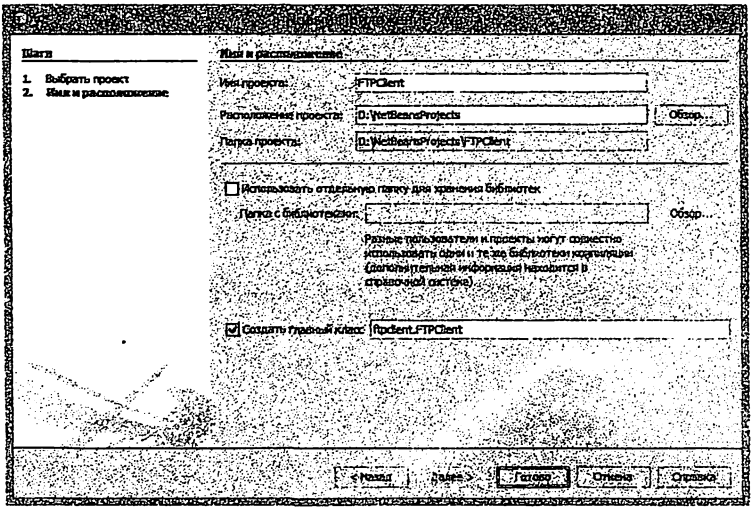


Рисунок 6.3. Среда IDE NetBeans Новое окно приложения Java

Шаг 4. Java код добавляется в созданный файл FTPClient.java.

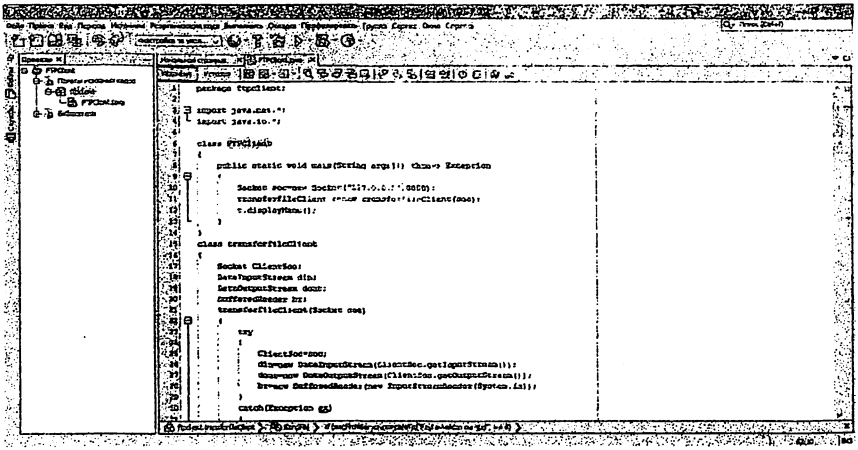


Рисунок 6.4. Файл FTPClient.java среда NetBeans IDE

Шаг 5. Как и в предыдущих шагах, создается проект FTPServer, и код Java добавляется в файл FTPServer.java.

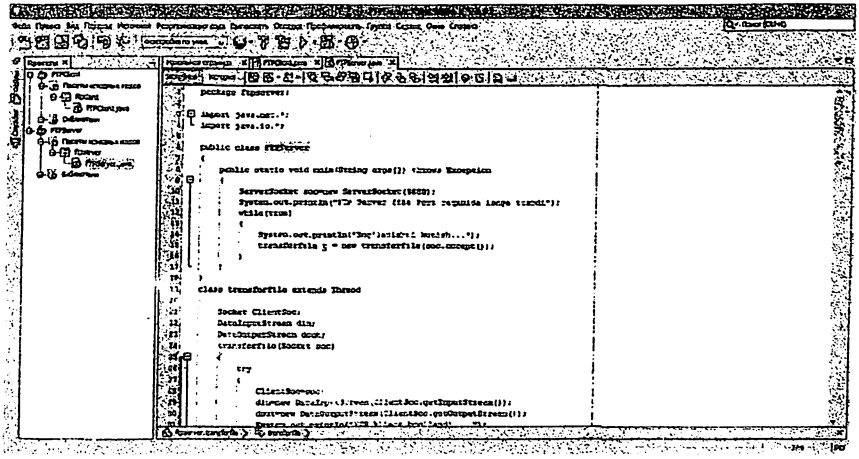


Рисунок 6.5. Файл FTPServer.java среда NetBeans IDE

Контрольные вопросы

1. FTP-протокол.
2. Типы команд в FTP.
3. Ответы в FTP.
4. Пакет java.net.*.
5. Пакет java.io.*
6. Класс сокета.
7. Класс InputStream.
8. Класс OutputStream.

ЛАБОРАТОРНАЯ РАБОТА №7

Тема: Создание сетевого приложения электронной почты

Цель работы:

Формирование практических навыков у студентов по созданию приложений электронной почты с использованием классов пакетов javax.mail.* и java.util.* языка программирования Java.

Теоретическая часть:

JavaMail API поддерживает фреймворки независимые от платформы и протоколы платформы для создания приложений электронной почты. JavaMail API предоставляет набор абстрактных классов, определяющих объекты, включая систему электронной почты. Это пакет для чтения, написания и отправки электронной почты.

Вот некоторые из протоколов, поддерживаемых JavaMail API:

SMTP: простой протокол передачи почты. Предоставляет механизм для отправки электронной почты.

POP3: Post Office Protocol. Поддерживает механизм получения электронной почты. Задает поддержку одного почтового ящика для каждого пользователя. RFC 1939 определяет этот протокол.

IMAP4: Internet Message Access Protocol. Отличный протокол, поддерживающий электронную почту. Предоставляет несколько почтовых ящиков каждому пользователю. Кроме того, почтовый ящик может использоваться несколькими пользователями. RFC определен в 2060 году.

MIME: Multipurpose Internet Mail Extensions. Это не протокол передачи электронной почты. Он определяет, что копируется, т.е. содержание (формат, приложения) письма. API JavaMail используется для написания, получения и отправки электронных писем.

Классы JavaMail API.

JavaMail API включает в себя несколько классов, которые позволяют отправлять, читать и удалять электронные письма. JavaMail API содержит два часто используемых автономных пакета: javax.mail и javax.mail.internet. Эти пакеты включают основные классы JavaMail.

Таблица 7.1

Классы JavaMail API

Классы	Описание
javax.mail.Session	Является ключевым классом API. Многогранный объект, представляющий соединение.
javax.mail.Message	Обеспечивает реализацию электронной почты.
javax.mail.Address	Обеспечивает работу с адресами в сообщении.
javax.mail.Authenticator	Используется для защиты ресурсов на почтовом сервере
javax.mail.Transport	Предоставляет механизм отправки электронной почты.
javax.mail.Store	Обеспечивает хранение и чтение сообщений. Магазин разделен на папки.
javax.mail.Folder	Обеспечивает хранение почтовых сообщений в папках.
javax.mail.internet. MimeMessage	Позволяет работать с электронными письмами в различных форматах.
javax.mail.internet. InternetAddress	Позволяет работать с адресами электронной почты с использованием синтаксиса RFC822.

Класс Session.

Класс сеанса является базовым классом API JavaMail. Объект сеанса создает соединение для JavaMail API для настройки и проверки подлинности.

Объект Session может быть создан следующими способами:

- Путем поиска контролируемого объекта из сервиса JNDI

  InitialContext ctx = new InitialContext ();

  Session session = (Session) ctx.lookup("usersMailSession");
- Второй метод заключается в создании объекта Session с использованием java.util.Properties на основе программирования. Конструктор класса сеанса является закрытым. Класс сеанса предоставляет два метода:
- getDefaultInstance(): объект Session вызывается по умолчанию с помощью метода getDefaultInstance().

  public static Session getDefaultInstance(Properties props)

  public static Session getDefaultInstance(Properties props,Authenticator auth)

- getInstance(): вызывается новый объект Session с помощью метода getInstance().

public static Session getInstance(Properties props)

public static Session getInstance(Properties props,Authenticator auth)

Класс Message

Объект сеанса создан, теперь приступим к созданию сообщения для отправки.

Тип сообщения

- Message является абстрактным классом, поэтому его класс javax.mail.internet.MimeMessage широко используется.
- Конструктор MimeMessage используется для создания сообщения.

Например:

MimeMessage message=new MimeMessage(session);

- Сообщение создано. Теперь нам нужно сохранить сообщение. Вы можете использовать такие методы, как message.setContent() или mimeMessage.setText().
- Широко используемые методы класса MimeMessage

Таблица 7.2

Методы класса MimeMessage

Метод	Описание
public void setFrom(Address address)	Используется для установки адреса отправителя
public void addRecipients (Message.RecipientType type, String addresses)	Определяет тип и адрес получателей.
public void setSubject(String subject)	Используется для определения темы.
public void setText(String textmessage	Используется для обозначения текста сообщения.

Класс Address.

У нас есть объекты Session и Message, и нам нужно добавить адреса в сообщение, используя объект Address.

- Adress — это абстрактный класс. Поэтому широко используется класс javax.mail.internet.
- InternetAddress. Введите свой адрес электронной почты, чтобы создать адрес:*Address address = new InternetAddress("talaba@gmail.com");*
- Еще один способ создания адресного объекта:

Address address = new InternetAddress("talaba@gmail.com", asror);

Класс Authenticator

Класс Authenticator представляет объект проверки подлинности для сетевого подключения.

Обычно он запрашивает у пользователя информацию.

- Authenticator— это абстрактный класс. Вы можете использовать конструктор класса PasswordAuthentication, введя свои Логин и Пароль. Аутентификатор должен быть зарегистрирован при создании объекта сеанса.

Ниже приведен пример аутентификатора:

Properties props = new Properties();

PasswordAuthentication auth = new PasswordAuthentication("talaba", "parol")
Session session = Session.getDefaultInstance(props, auth);

Класс Transport.

Класс Transport используется в механизме передачи сообщений. Обычно этот класс использует протокол SMTP для отправки сообщения.

Вы можете отправить сообщение с помощью static send(): *Transport.send(message);*

Второй способ - открыть сессию, введя логин и пароль, отправить сообщение и закрыть соединение:

message.saveChanges(); // используется вместо send()

//выбор транспорта для session

Transport transport = session.getTransport("smtp");

//связаться

transport.connect(host, username, password);

// повторить при необходимости

transport.sendMessage(message, message.getAllRecipients());

// сделано, закрыть соединение

transport.close();

Класс Store.

Класс, используемый для хранения и чтения сообщений. Класс Store расширяется с помощью класса Service.

Store store = session.getStore("pop3");

store.connect(host, username, password);

Класс Folder.

Класс Folder отображает папки для сообщений электронной почты. Папки могут содержать вложенные папки и сообщения вместо них. После подключения к Store вы можете связаться с Folder.

Folder folder = store.getFolder("INBOX");

folder.open(Folder.READ_ONLY);

Message message[] = folder.getMessages();

Метод getFolder (String name) объекта Folder возвращает имя папки. На картинке ниже вы можете увидеть связь между магазином и папкой.

Задание:

Студент получает индивидуальное задание на лабораторную работу. Для этого задания учащийся создаст программу электронной почты.

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе. Изображение ниже представляет главное окно среды NetBeans IDE.

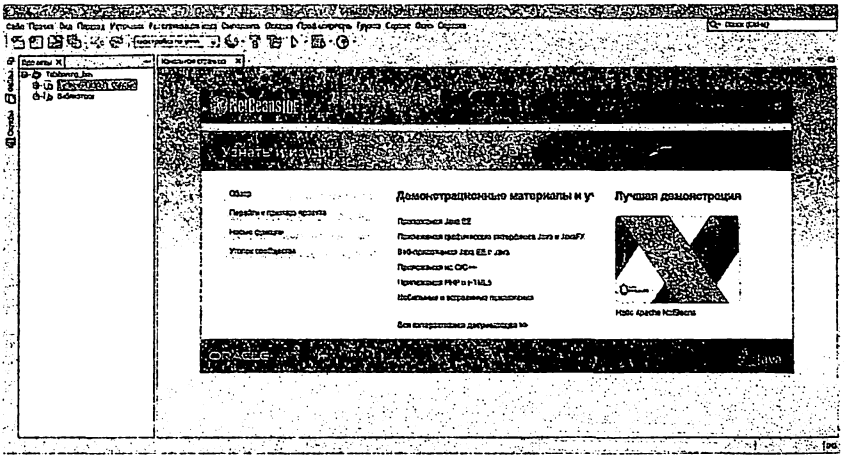


Рисунок 7.1. Главное окно среды NetBeans IDE

Шаг 2. Щелкните проект правой кнопкой мыши и выберите «Новый → Класс Java» в контекстном меню.

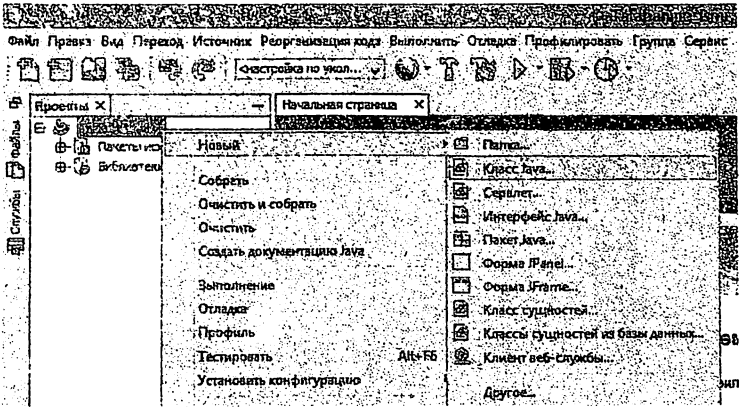


Рисунок 7.2. Создание нового класса в среде NetBeans IDE

Шаг 3. В поле «Имя класса» окна "New Класс Java" введите «SendEmailUsingGMailSMTP», в поле «Пакет» введите «email» и нажмите «Готово».

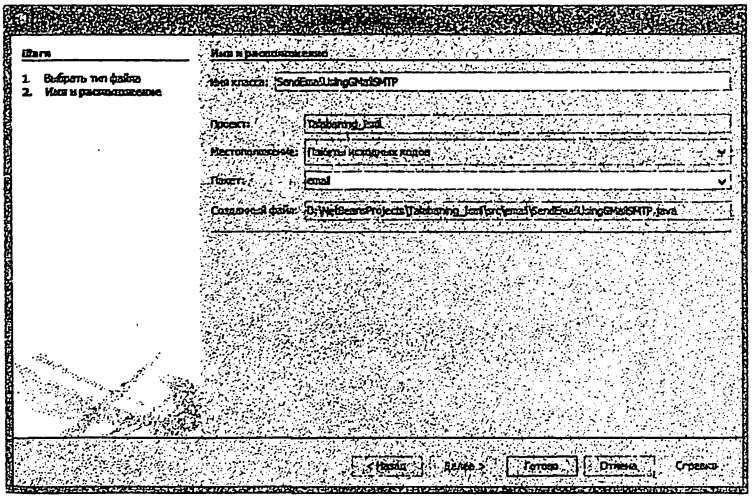


Рисунок 7.3. Окно «New Класс Java» среды NetBeans IDE

Шаг 4. Java код вводится в созданный файл SendEmailUsingGMailSMTP.java.

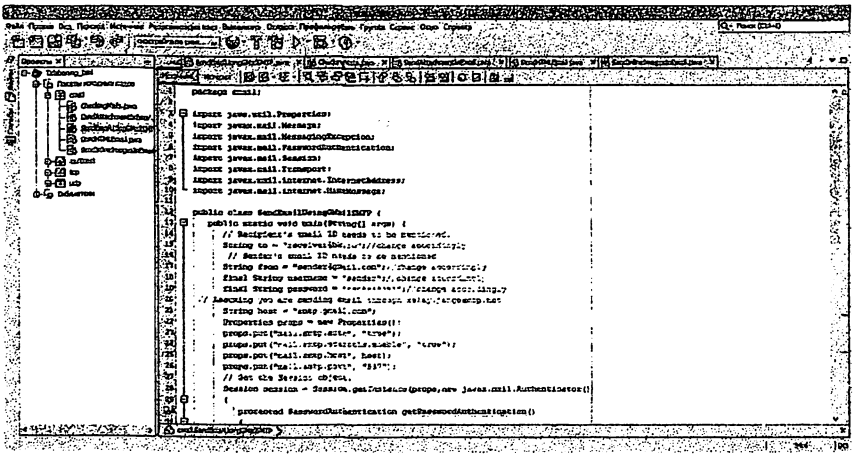


Рисунок 7.4. Файл SendEmailUsingGMailSMTP.java среда NetBeans IDE

Контрольные вопросы

1. Электронная почта
2. Протокол SMTP
3. Протокол POP
4. Протокол IMAP

5. Протокол MIME
6. Заголовок письма
7. Почтовый клиент Mail Transport Agent
8. Mail Delivery Agent
9. Mail User Agent.

ЛАБОРАТОРНАЯ РАБОТА № 8

Тема: Создание сетевой программы на основе AJAX

Цель работы:

Формирование практических навыков у студентов по созданию сетевого приложения на основе AJAX с использованием объекта XMLHttpRequest.

Теоретическая часть:

AJAX (Асинхронный JavaScript и XML) - это набор инструментов для создания асинхронных веб-приложений на стороне клиента с использованием многих веб-технологий. Ajax может отправлять и получать информацию с сервера (в фоновом режиме) асинхронно, не мешая внешнему виду и работе страницы веб-приложения. Ajax позволяет динамически изменять содержимое веб-страниц без перезагрузки всей страницы. Возможности AJAX: - Обновление веб-страницы без перезагрузки страницы - Запрос информации с сервера после загрузки страницы - Получение данных с сервера после загрузки страницы - Отправка данных на сервер в фоновом режиме.

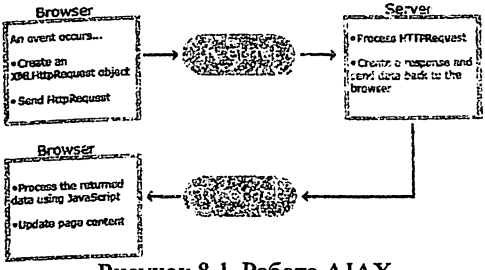


Рисунок 8.1. Работа AJAX

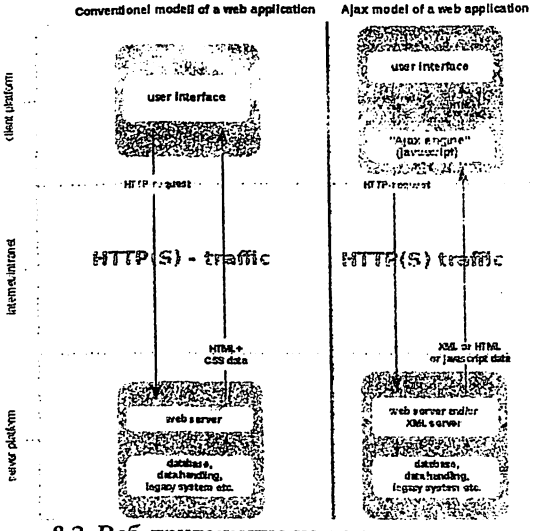


Рисунок 8.2. Веб-приложение на основе технологии Ajax

Объект XMLHttpRequest.

Все современные браузеры поддерживают объект XMLHttpReques. В оновом режиме вы можете использовать объект XMLHttpRequest для бмена информацией с веб-сервером. Это означает, что части веб-страницы ожно обновлять без перезагрузки всей страницы.

var xhttp = new XMLHttpRequest ();

Таблица 8.1

Методы объекта XML HttpRequest

Метод	*Описание*

new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Отправка запроса на сервер:

- xhttp.open("GET", "ajax_info.txt", true); xhttp.send();
- xhttp.open("POST", "demo_post.asp", true); xhttp.send();
- xhttp.open("POST", "ajax_test.asp", true); xhttp.setRequestHeader("Content-type", "application/x-www-form- urlencoded"); xhttp.send("fname=Henry&lname=Ford");
- xhttp.open("GET", "ajax_info.txt", false); xhttp.send();

Таблица 8.2

Ответ сервера

Property	*Description*
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" .....
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Ответ сервера:

```
function loadDoc() {
  var xhttp = new
  XMLHttpRequest();
  xhttp.onreadystatechange =
  function() {
    if (this.readyState == 4 && this.status ==
      200) {
      document.getElementById("demo").innerHT
      ML = this.responseText;
    } };
  xhttp.open("GET", "ajax_info.txt",
  true); xhttp.send();
}
```

Задание:

Студент получает индивидуальное задание на выполнение лабораторных работ. Для этой задачи студент создает сетевую программу на основе AJAX.

Порядок выполнение работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе. Изображение ниже представляет главное окно среды NetBeans IDE.

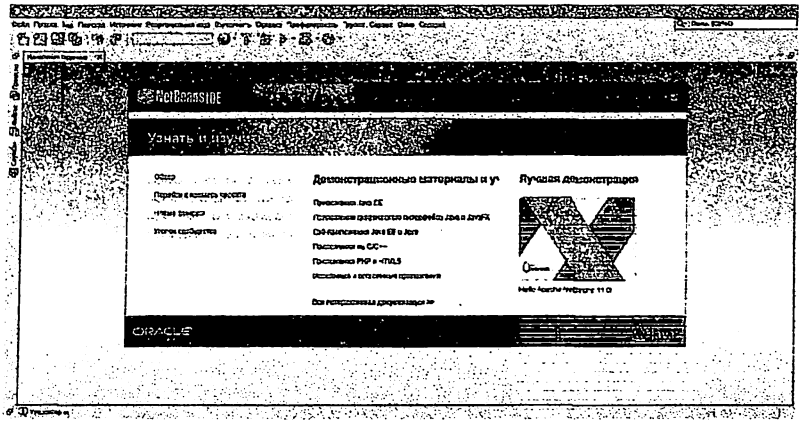


Рисунок 8.3. Главное окно среды NetBeans IDE

Шаг 2. Когда вы выберете раздел «Создать проект» в меню «Файл», появится окно со следующим изображением. В этом окне выберите JavaWeb в разделе «Категории» и нажмите «Далее».

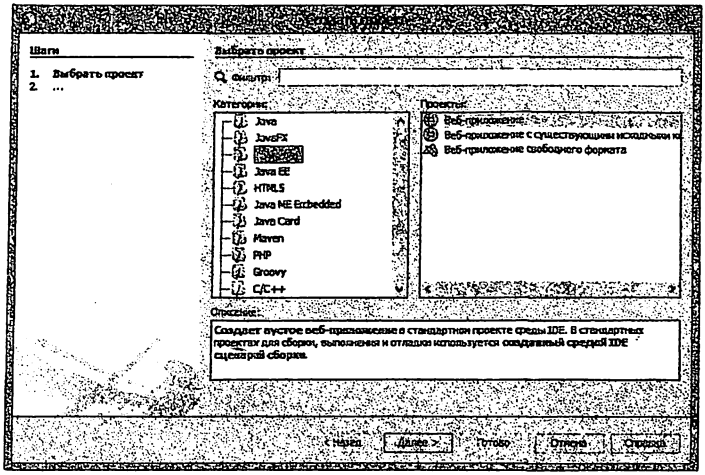


Рисунок 8.4. Окно «Создать проект» среды NetBeans IDE

Шаг 3. Переход к окну «Новый Веб-приложение». В поле «Имя проекта» этого окна введите «Имя_студента_Web» в качестве имени проекта и нажмите «Далее».

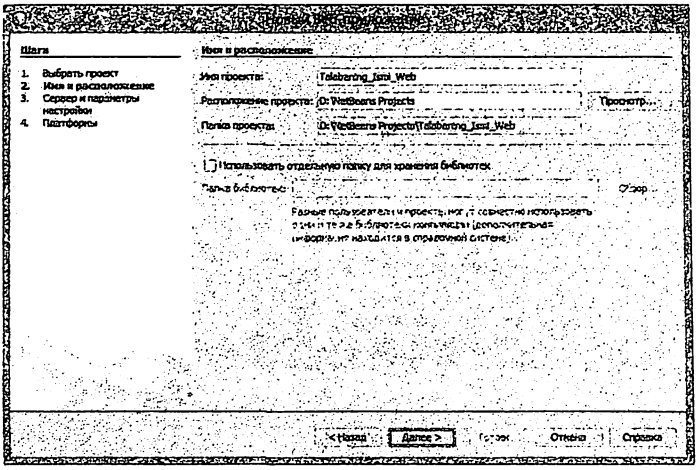


Рисунок 8.5. Окно «Новый Веб-приложение» среды NetBeans IDE

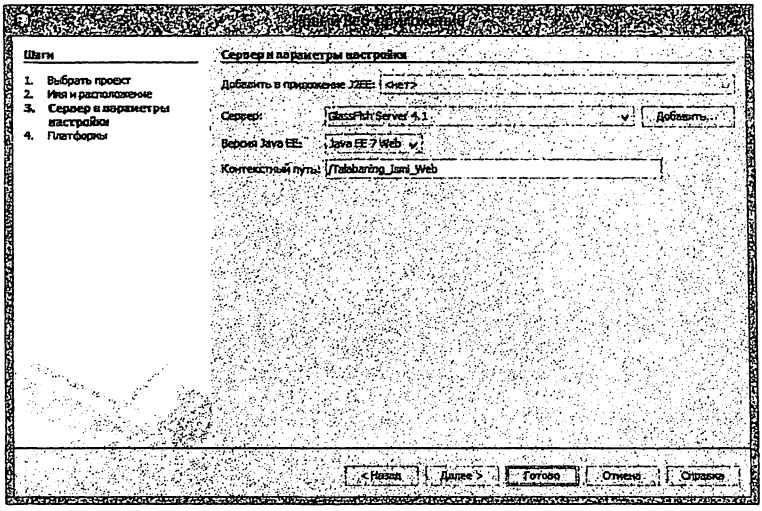


Рисунок 8.6. Окно «Новый Веб-приложение» среды NetBeans IDE

Затем нажмите «Готово» в окне на Рисунке 8.6.

Шаг 4. Щелкните над созданным проектом правой кнопкой мыши и выберите в контекстном меню «Новый → HTML…».

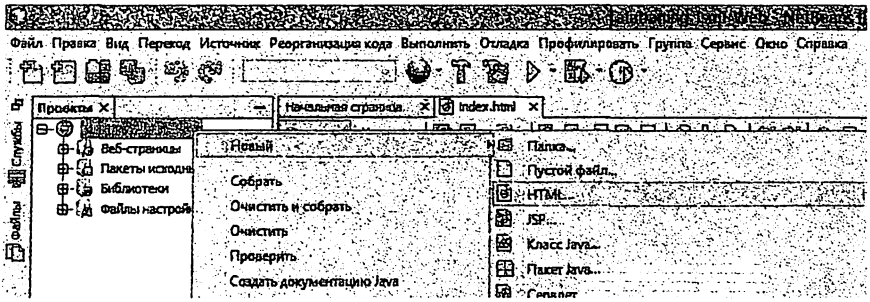


Рисунок 8.7. Среда NetBeans IDE для создания файла HTML

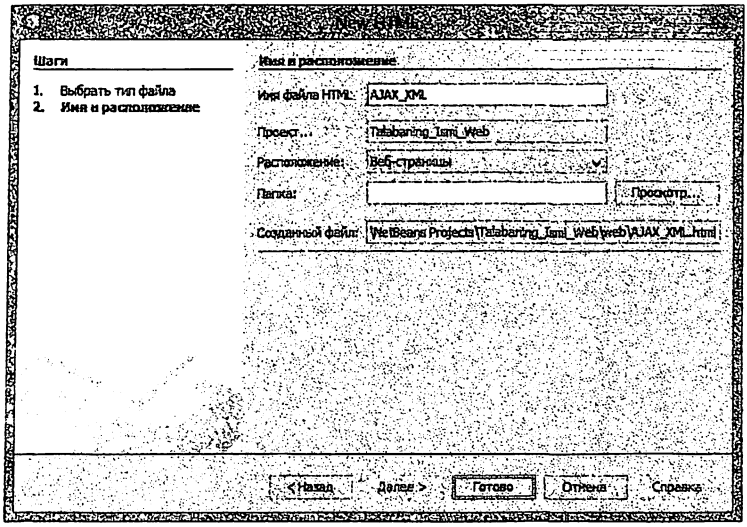


Рисунок 8.8. Окно создания HTML-файла среды NetBeans IDE

Шаг 5. В окне «New HTML» введите AJAX_XML в поле «Имя файла HTML» и нажмите «Готово». Созданный файл AJAX_XML.html содержит следующий код HTML+CSS+JavaScript.

```
<!DOCTYPE html>
<html>
  <style>
    table,th,td {
      border : 1px solid
      black; border-
      collapse: collapse;
    }
    th,td {
      padding: 5px;
    }
  </style>
  <body>
    <h2>The XMLHttpRequest Object</h2>
    <button type="button" onclick="loadDoc()">Get
my CD collection</button>
    <br><br>
    <table id="demo"></table>
    <script>
      function loadDoc() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
          if (this.readyState == 4 && this.status ==
            200) { myFunction(this);
          }
        };
        xhttp.open("GET", "cd_catalog.xml",
        true); xhttp.send();
      }
      function
        myFunction(xml) {
        var i;
        var xmlDoc = xml.responseXML;
        var table =
        "<tr><th>Artist</th><th>Title</th></tr>"; var x
        = xmlDoc.getElementsByTagName("CD");
        for (i = 0; i < x.length;
          i++) { table +=
          "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeVa
              lue + "</td><td>" +
```

```
x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeVa
                    lue + "</td></tr>";
            }
            document.getElementById("demo").innerHTML = table;
        }
    </script>
  </body>
</html>
```

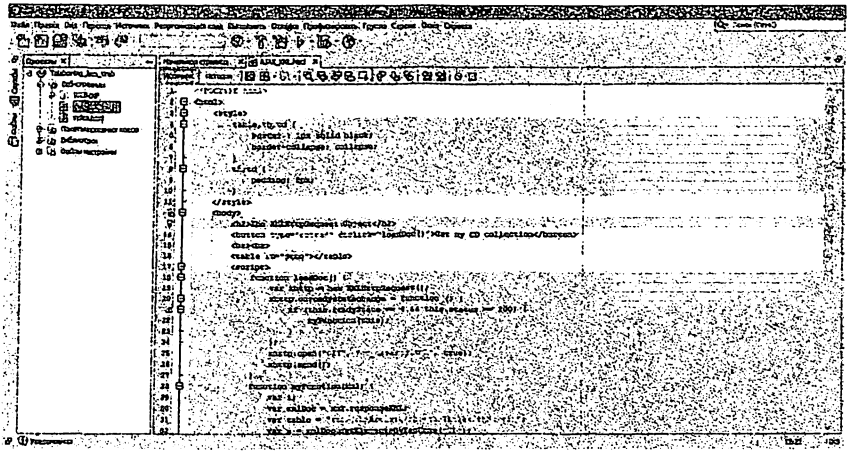


Рисунок 8.9. Содержимое файла AJAX_XML.html среды NetBeans IDE

Шаг 6. Теперь создадим файл cd_catalog.xml. Для этого щелкните правой кнопкой мыши «Веб-страницы» и выберите «Новый → Пустой файл...» в появившемся контекстном меню.

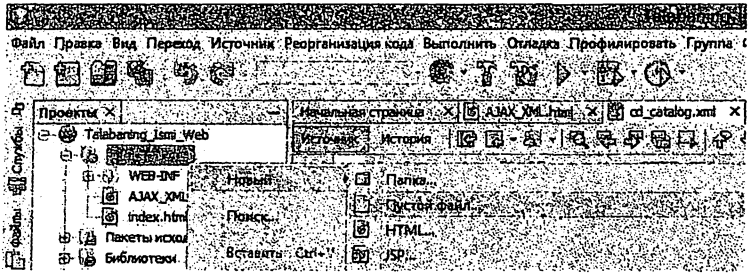


Рисунок 8.10. Создайте файл среды NetBeans IDE

Шаг 7. В окне «New Пустой файл» введите *cd_catalog.xml* в поле «Имя файла» и нажмите «Готово».

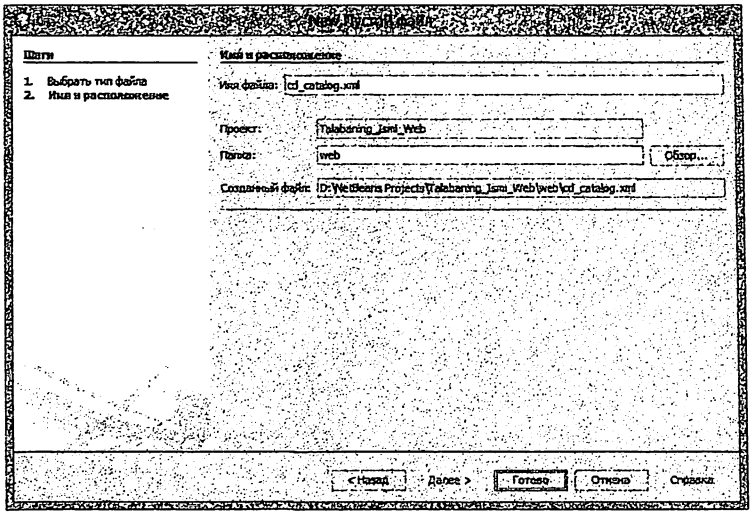


Рисунок 8.11. Создание файл cd_catalog.xml в среде NetBeans IDE

Следующий XML-код вводится в созданный файл “cd_catalog.xml”.

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
 <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
 </CD>
 <CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS Records</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
 </CD>
 <CD>
  <TITLE>Greatest Hits</TITLE>
  <ARTIST>Dolly Parton</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>RCA</COMPANY>
  <PRICE>9.90</PRICE>
```

```
  <YEAR>1982</YEAR>
 </CD>
 <CD>
  <TITLE>Still got the blues</TITLE>
  <ARTIST>Gary Moore</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Virgin records</COMPANY>
  <PRICE>10.20</PRICE>
  <YEAR>1990</YEAR>
 </CD>
 <CD>
  <TITLE>Eros</TITLE>
  <ARTIST>Eros Ramazzotti</ARTIST>
  <COUNTRY>EU</COUNTRY>
  <COMPANY>BMG</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1997</YEAR>
 </CD>
 <CD>
  <TITLE>One night only</TITLE>
  <ARTIST>Bee Gees</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>Polydor</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1998</YEAR>
 </CD>
 <CD>
  <TITLE>Sylvias Mother</TITLE>
  <ARTIST>Dr.Hook</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <COMPANY>CBS</COMPANY>
  <PRICE>8.10</PRICE>
  <YEAR>1973</YEAR>
 </CD>
```

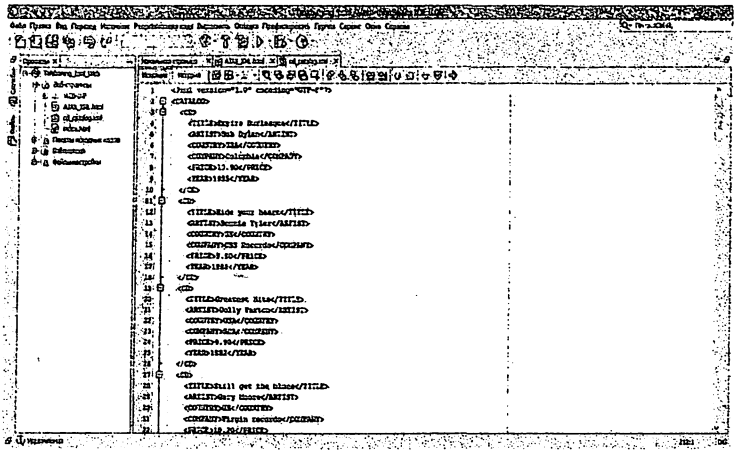


Рисунок 8.12. Содержимое файла cd_catalog.xml среды NetBeans IDE

Шаг 8. Чтобы запустить сетевое приложение, щелкните правой кнопкой мыши над файлом AJAX_XML.html и выберите "Выполнить файл" в всплывающем меню.

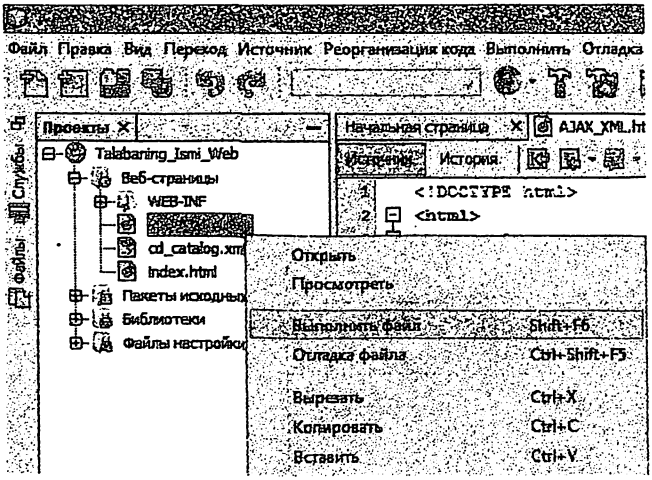


Рисунок 8.13. Запустите среду NetBeans IDE AJAX_XML.html

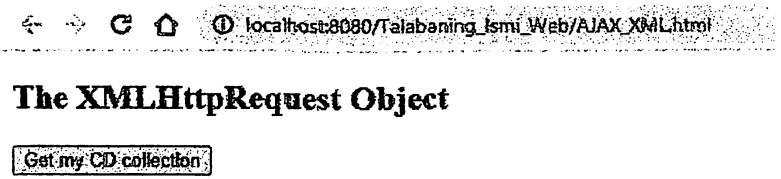


Рисунок 8.14. Результат в браузере

The XMLHttpRequest Object

Get my CD collection

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother
Rod Stewart	Maggie May
Andrea Bocelli	Romanza
Percy Sledge	When a man loves a woman
Savage Rose	Black angel

Рисунок 8.15. Результат в браузере

Контрольные вопросы

1. Возможности AJAX
2. Работа AJAX
3. Объект XMLHttpRequest
4. Запрос и ответ сервера
5. AJAX XML
6. AJAX Database
7. AJAX PHP
8. AJAX ASP

СОДЕРЖАНИЕ

СПИСОК ЛИТЕРАТУРЫ

1. Постановление Президента Республики Узбекистан «О мерах по дальнейшему развитию высшего образования» от 20.04.2017 г. № ПП-2909.
2. Постановление Президента Республики Узбекистан «О мерах по дальнейшему расширению участия отраслей и сфер экономики в повышении качества подготовки специалистов с высшим образованием» от 27.07.2017 г. № ПП-3151.
3. Указ Президента Республики Узбекистан «Об утверждении Стратегии «Цифровой Узбекистан-2030» и мерах по ее эффективной реализации» от 05.10.2020 г. № УП-6079.
4. Постановление Президента Республики Узбекистан «О мерах по дальнейшему расширению доступа к образованию в высших образовательных организациях» от 30.07.2021 г. № ПП-5203.
5. James F. Kurose, Keith W. Ross. Computer networking: a top-down approach.—6th ed. 2013. by Pearson Education, Inc., publishing as Addison-Wesley.
6. Behrouz A. Forouzan. TCP/IP protocol suite.—4th ed. Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2010
7. Elliotte Rusty Harold. Java Network Programming, Fourth Edition. 2014. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol.
8. Jan Graba. An Introduction to Network Programming with Java. Java 7 Compatible. Third Edition. Springer London Heidelberg New York Dordrech. 2013
9. David Reilly and Michael Reilly, Java Network Programming and Distributed Computing, Addison-Wesley (ISBN: 0-201-71037-4).
10. Агзамов Ф.С., Эргашев А.Қ., Туляганов А.А., Гультураев Н.Х. Ўқув адабиётларни ишлаб чиқиш ва нашр этишга тайёрлаш бўйича услубий кўрсатмалар. - Тошкент: Муҳаммад ал-Хоразмий номидаги ТАТУ. 2018. - 52 б.
11. https://www.javatpoint.com
12. https://www.tutorialspoint.com
13. https://www.w3schools.com

Методическое пособие разработанное для лабораторных работ по предмету “Основы сетевого программирования” часть 1 предназначено для бакалавров направления 5350100 – Телекоммуникационные технологии (“Телекоммуникации”)

Методическое пособие обсуждено на совещании заседании кафедры “АПОСУТ” №___ от ___ _____ 2022 года и рекомендовано для изучения в научно-методическом совете факультета.

Методическое пособие обсуждено на совещании факультета “Телекоммуникационные технологии” №___ от ____ _____ 2022 года и рекомендовано для изучения в научно-методическом совете университета.

Методическое пособие обсуждено на совещании научно-методического совета ТУИТ №___ от __ _____ 2022 года и рекомендовано для печати.

Авторы: О.Н.Джураев

Х.Х.Ахмедова

Ф.К.Тожиева

Рецензенты: DSc. У.Р.Хамдамов

PhD. Ж.Т.Усмонов

Ответственный редактор: DSc. С.С.Парсиев

Корректор: PhD. Ж.Б.Элов

Формат 60x84 1/16. Печ.лист 4,5.
Заказ №73. Тираж 10.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.