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, Ф.К.Тожиева, Х.Х.Ахмедова

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по выполнению лабораторных работ по предмету

ОСНОВЫ СЕТЕВОГО ПРОГРАММИРОВАНИЯ

ЧАСТЬ 2

Ташкент 2022

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

Методическое пособие по выполнению лабораторных работ по предмету "Основы сетевого программирования" часть 2 - Ташкент: ТУИТ. 2022. - 80 стр.

В методическом пособии представлены указания по созданию многопоточного приложения работающий в сети, создание сетевого приложения на основе JavaFX, распределенные клиент-серверные приложения, создание сетевой программы работающий с базами данных, создание сервлетов работающий на сервере, создание динамических веб-страниц. Оно включает название лабораторной работы, цель работы, теоретическую часть, задание, порядок выполнения работы и контрольные вопросы.

Методическое пособие предназначено для бакалавров направления 5350100 – Телекоммуникационные технологии ("Телекоммуникации").

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.

Вторая часть методического пособия по выполнению лабораторных работ по предмету «Основы сетевого программирования» по направлению «5350100 - Телекоммуникационные технологии («Телекоммуникации») состоит из указания по созданию многопоточного приложения работающий в сети, создание сетевого приложения на основе JavaFX, распределенные клиент-серверные приложения, создание сетевой программы работающий с базами данных, создание сервлетов работающий на сервере, создание динамических веб-страниц.

ЛАБОРАТОРНАЯ РАБОТА № 9

Тема: Создание сетевого приложения с использованием JavaFX

Цель работы:

Формирование практических навыков у студентов по созданию клиент-серверного сетевого приложения с графическим интерфейсом пользователя с использованием классов пакетов java.net.*, java.io.*, javafx.stage.*, javafx.application.*, javafx.event.* языка программирования Java.

Теоретическая часть:

JavaFX — это библиотека Java, используемая для создания многофункциональных интернет-приложений. Приложения, написанные с использованием этой библиотеки, могут работать на разных платформах. Приложения, разработанные с использованием JavaFX, могут работать на различных устройствах, таких как настольные компьютеры, мобильные телефоны, телевизоры, планшеты и т. д.

Для разработки приложений с графическим интерфейсом на языке программирования Java программисты используют такие библиотеки, как Advanced Windowing Toolkit(AWT) и Swing. После появления JavaFX эти Java-программисты теперь могут эффективно разрабатывать приложения с графическим интерфейсом с богатым контентом.

Для разработки клиентских приложений с богатыми функциями программисты привыкли полагаться на различные библиотеки для добавления таких функций, как мультимедиа, элементы управления пользовательским интерфейсом, веб, 2D и 3D и т.д. JavaFX включает все эти функции в одну библиотеку. В дополнение к этому, разработчики могут также получить доступ к существующим функциям библиотеки Java, такой как Swing.

JavaFX предоставляет богатый набор графических и мультимедийных API-интерфейсов, а также использует современный графический процессор с помощью аппаратно-ускоренной графики. JavaFX также предоставляет

интерфейсы, с помощью которых разработчики могут комбинировать графическую анимацию и управление пользовательским интерфейсом.

Можно использовать JavaFX с технологиями на основе JVM, такими как Java, Groovy и JRuby. Если разработчики выбирают JavaFX, нет необходимости изучать дополнительные технологии, поскольку предварительные знания любой из вышеупомянутых технологий будут достаточно хороши для разработки RIA с использованием JavaFX.

JavaFX предоставляет полный API с богатым набором классов и интерфейсов для создания приложений с графическим интерфейсом с богатой графикой. Важными пакетами этого API являются:

javafx.animation, javafx.application, javafx.css, javafx.event, javafx.geometry, javafx.stage, javafx.layout, javafx.beans, javafx.collections, javafx.util, javafx.concurrent, javafx.embed.swing, javafx.fxml, javafx.print, javafx.scene, javafx.scene.shape, javafx.scene.text, javafx.scene.effect, javafx.scene.input, javafx.scene.transform, javafx.scene.paint, javafx.scene.media, javafx.scene.web, javafx.scene.image, javafx.scene.control, javafx.scene.chart.

В JavaFX приложения с графическим интерфейсом были закодированы с использованием графа сцены. Граф сцены является отправной точкой при создании приложения с графическим интерфейсом. Он содержит прикладные примитивы (GUI), которые называются узлами.

Узел является визуальным / графическим объектом и может включать в себя:

- Геометрические (графические) объекты - (2D и 3D), например, круг, прямоугольник, многоугольник и т.д.;
- Элементы управления пользовательского интерфейса - Button, Checkbox, Choice box, Text Area и т.д.;;
- Контейнеры - Border Pane, Grid Pane, Flow Pane и т.д.;
- Элементы мультимедиа - объекты аудио, видео и изображения;

Stage (окно) содержит все объекты приложения JavaFX. Он представлен классом Stage пакета javafx.stage. Первичная стадия создается самой платформой. Созданный объект stage передается в качестве аргумента методу start () класса Application.

Stage имеет два параметра, определяющих его положение, а именно: ширина и высота . Он делится на область содержимого и украшения (строка заголовка и границы).

Есть пять типов доступных этапов:

- Decorated;
- Undecorated;
- Transparent;
- Unified;
- Utility.

Для отоброжения содержимого сцены вызываем метод show ().

Scene представляет физическое содержимое приложения JavaFX. Он содерит все содержимое графа сцены. Класс Scane пакета javafx.scene представляет объект сцены.

```
public class JavafxSample extends Application {
  @Override
  public void start(Stage primaryStage) throws Exception {
    /*
    Code for JavaFX application.
    (Stage, scene, scene graph)
    */
  }
  public static void main(String args[]){
    launch(args);
  } }
```

Первая задача клиента — установить связь с сервером. Объект Socket создается для установления соединения между сервером и клиентом. Сокет TCP выполняет следующие задачи для создания клиентского приложения:

Первой задачей клиента является связь с сервером. Сокетное соединение с сервером создается клиентом с помощью объекта

класса Socket. Сокет TCP выполняет следующие задачи для создания клиентского приложения:

1. Создать клиентский сокет, используя объект сокета.

2. Записовать в сокет и читать из него.

3. Завершить соединение.

Создание клиентского сокета.

Объект клиентского сокета создается с помощью конструктора класса Socket, который получает два параметра (IP-адрес и номер порта), прослушиваемые сервером.

Socket clientSocket = new Socket("127.0.0.1", 7777);

В предыдущем фрагменте кода IP-адрес 127.0.0.1 и порт 7777 определяют сокет, в котором сервер принимает запросы клиентов.

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 смог взаимодействовать с сервером, ему необходимо:

Два объекта объявляются для каждого из 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.

В клиентском приложении объявлен еще один объект класса BufferdReader для связи 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().

Создать TCP-сервер.

Процесс разработки 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, вам необходимо сделать следующее:

- Создать объект сокета для сервера ServerSocket;
- слушать запросы клиента после подключение

- запустить сервер;
- создание потока подключения для запросов клиентов.

Задание:

Каждый студент получает индивидуальное задание. В этом задании студент создает сетевое приложение TCP клиент-сервер на основе JavaFX. Индивидуальное задание для каждого студента приведено в таблице 9.1.

Таблица 9.1

Варианты заданий

№	Задания
1.	Сортировать число n в порядке возрастания
2.	Сортировать число n в порядке убывания
3.	Увеличить любое число до n уровней
4.	Вычислить квадратный корень из произвольного числа
5.	Найти наибольшее из любых n чисел
6.	Найти наименьшее из любых n чисел
7.	Найти объем произвольного куба
8.	Найти длину биссектрису треугольника
9.	Вычислить сумму любых n чисел
10.	Вычислить сумму положительных чисел из любых n чисел
11.	Рассчитать геометрию любых n чисел
12.	Выделить текст с двойным интервалом в любом тексте
13.	Выделить нечетные буквы в любом тексте
14.	Вычислить объем шара радиуса R
15.	Вычислить объем произвольного цилиндра
16.	Вычислить диагональ прямоугольника
17.	Найти поверхность шара радиуса R
18.	Вычислить площадь треугольника
19.	Отдельные пары произвольных n чисел
20.	Найти среднее арифметическое любых n чисел
21.	Найти среднюю геометрию любых n чисел
22.	Найти произвольный n факториал
23.	Найти площадь ромба
24.	Вычислить среднее арифметическое любых n чисел
25.	Найти площадь прямоугольника
26.	Вычислить геометрию любых n чисел
27.	Вычислить площадь треугольника
28.	Найти длину окружности произвольного радиуса R.
29.	Вычислить объем шара радиуса R
30.	Вычислить площадь поверхности произвольного радиуса R

Порядок выполнения работы:

Шаг 1. Работа начинается с загрузки среды NetBeans IDE. Для этого щелкните два раза ярлык среды NetBeans IDE на рабочем столе левой кнопкой мыши.

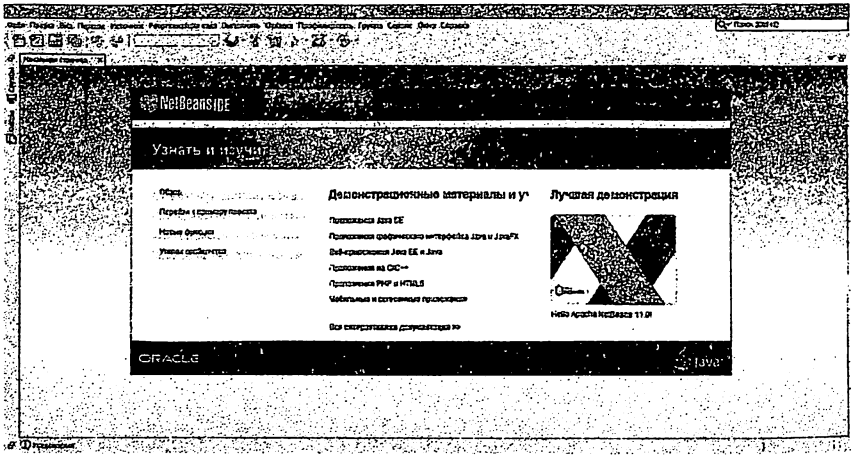


Рисунок 9.1. Главное окно среды NetBeans IDE

Шаг 2. Затем в окне, которое появляется при выборе раздела «Открыть проект» в меню «Файл», выберите проект «Имя ученика» и нажмите «Открыть проект».

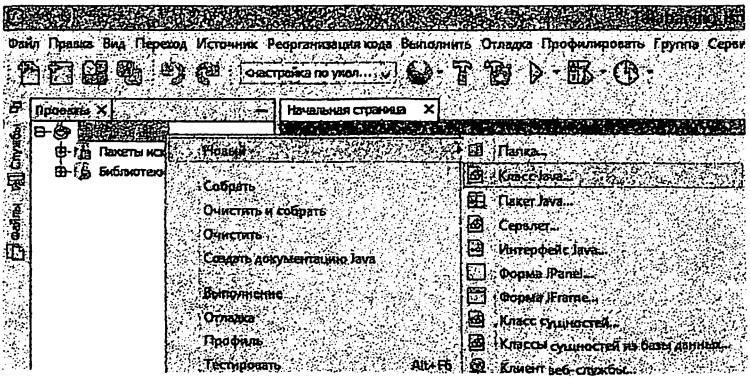


Рисунок 9.2. Создание нового класса среды NetBeans IDE

Шаг 3. Затем щелкните правой кнопкой мыши проект "Имя_студента" и выберите "Новый" → "Класс Java" из контекстного меню.

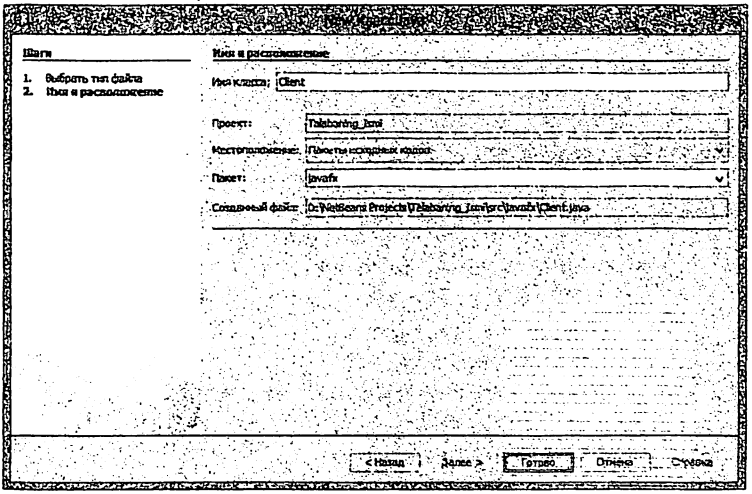


Рисунок 9.3. Окно «New Класс Java» среды NetBeans IDE

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

Шаг 5. Следующий код Java добавляется в файл Client.java внутри созданного пакета javafx.

```
package javafx;
import javafx.application.Application;
import javafx.event.*;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import java.io.*;
import java.net.*;
import static javafx.application.Application.launch;
public class Client extends Application {
   @Override
```

```
public void start(Stage stage) {
    Text text = new Text("Matnni kiriting");
    TextField textField = new TextField();
    Button button = new Button("Jo'natish");
    button.setOnAction(new EventHandler<ActionEvent>() {
      @Override
      public void handle(ActionEvent event) {
        try (Socket clientSocket = new Socket("localhost", 7777)) {
            DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
            BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            String a = textField.getText();
            outToServer.writeBytes(a + '\n');
            String b = inFromServer.readLine();
            textField.setText(b);          }
        . catch (IOException e){}      }    });
    GridPane gridPane = new GridPane();
    gridPane.setMinSize(400, 200);
    gridPane.setVgap(5);
    gridPane.setAlignment(Pos.CENTER);
    gridPane.add(text, 0, 1);
    gridPane.add(textField, 0, 2);
    gridPane.add(button, 0, 3);
    Scene scene = new Scene(gridPane);
    stage.setTitle("Client");
    stage.setScene(scene);
    stage.show();   }
  public static void main(String args[]) {
    launch(args);
  } }
```

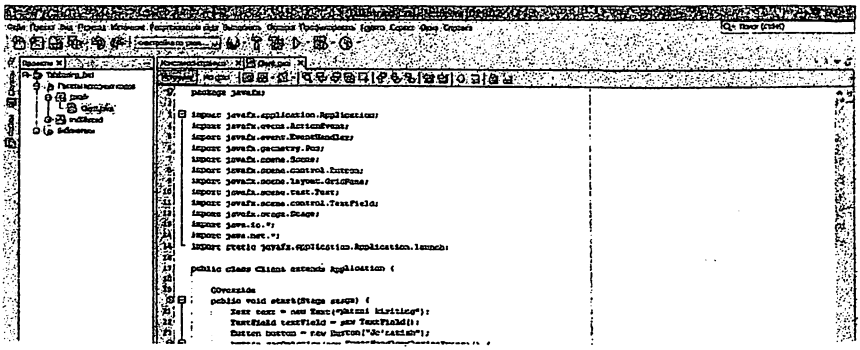


Рисунок 9.4. Файл Client.java в среде NetBeans IDE

Повторяя шаги 3 - 5 создаётся файл Server.java. Следующий код Java водится в файл Server.java.

```
package javafx;
import java.io.*;
import java.net.*;
class Server
{
  public static void main(String argv[]) throws Exception
  {
    String clientSentence;
    String capitalizedSentence;
    ServerSocket welcomeSocket = new ServerSocket(7777);
    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");
    } } }
```

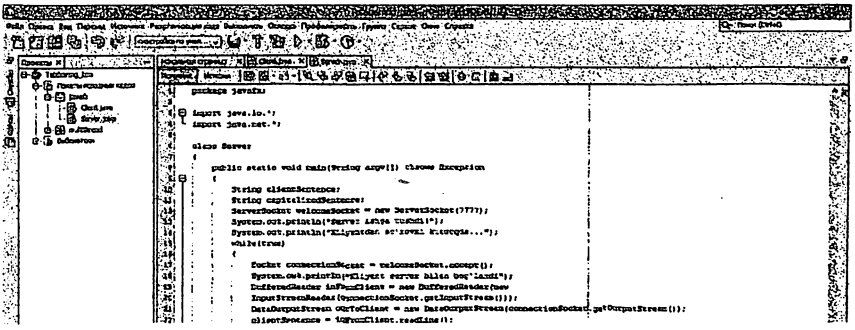


Рисунок 9.5. Файл Server.java в среде NetBeans IDE

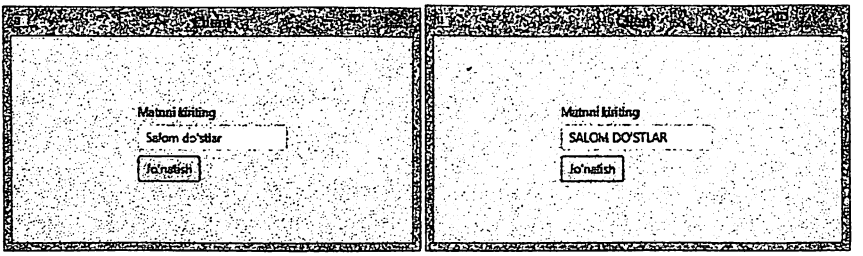


Рисунок 9.6. Результат клиента среды NetBeans IDE

Контрольные вопросы

1. Классы, методы пакета java.net.*
2. Классы, методы пакета java.io.*
3. Классы, методы пакета javafx.stage.*
4. Классы, методы пакета javafx.application.*
5. Классы, методы пакета javafx.event.*

ЛАБОРАТОРНАЯ РАБОТА № 10

Тема: Создание безопасных сокетов в сети

Цель работы:

Формирование практических навыков у студентов по созданию защищенных сокетов в сети с использованием классов пакетов javax.net.* и javax.net.ssl.* языка программирования Java.

Теоретическая часть:

Безопасные сокеты. AT&T позволила Агентству национальной безопасности США полностью подключить всех клиентов к интернет-трафику, копируя пакеты на устройства обработки данных, установленные в секретных комнатах коммутационных центров. Британская компания ввела в производство оптоволоконные кабели, по которым проходят телефонные звонки и интернет-трафик в мире. В Швеции Национальный институт Национальный обороны требует, чтобы владельцы оптоволоконных кабелей устанавливали оптоволоконное оборудование в своих зданиях, и это

небольшой пример правительства, которое слышит то, что мы знаем.

Как пользователь Интернета, мы должны защитить себя от наблюдателей. Сокеты могут быть зашифрованы, чтобы сделать подключение к Интернету более безопасным. Это позволяет транзакциям быть конфиденциальными, аутентифицированными и точными.

Однако шифрование является сложной проблемой. Для правильной реализации требуется детальное понимание не только математических алгоритмов, используемых для шифрования данных, но также ключей и протоколов, используемых для обмена зашифрованными данными. Даже небольшие ошибки могут открыть большую дыру в устройстве и сделать соединение опасным. Написание программного обеспечения для шифрования — это работа для лучших профессионалов. К счастью, неспециалисты обеспечивают безопасную связь, используя простые протоколы и алгоритмы, а также программное обеспечение, созданное экспертами. Когда вы заказываете что-то в интернет-магазине, процесс транзакции аутентифицируется с использованием зашифрованных и проверенных протоколов и алгоритмов. Разработчики, которые хотят писать программное обеспечение, которое подключается к интернет-магазинам, должны знать немного больше о протоколах и алгоритмах. Если вы хотите написать серверную программу, которая управляет интернет-магазином без использования других программ, вам нужно иметь больше опыта.

Протоколы Secure Sockets Layer (SSL) и Transport Layer Security (TLS) Java Secure Sockets Extension (JSSE) и связанные с ними алгоритмы могут использоваться для защиты сетевых подключений. SSL — это протокол безопасности, который позволяет веб-браузерам и другим клиентам TCP взаимодействовать с HTTP и другими серверами TCP на разных уровнях конфиденциальности и аутентификации.

Безопасная связь. Конфиденциальная связь через открытый канал, такой как Интернет, требует шифрования данных. Многие схемы шифрования на компьютере не ограничиваются текстом, а основаны на

общем ключе пароля. Простое текстовое сообщение в сочетании с битами ключа в соответствии с математическим алгоритмом создает зашифрованный текст. Увеличение количества бит затрудняет расшифровку сообщений.

В традиционном шифровании с закрытым ключом (или симметричном) один ключ используется для шифрования и расшифровки данных. Отправитель и получатель должны знать один ключ. Представьте, что Анвар хочет отправить секретное сообщение Сабиру. Сначала он отправляет ключ Сабиру, который тот использует для секретного обмена. Однако ключ нельзя зашифровать, поскольку ключа Сабира еще не существует, поэтому Анвар должен отправить ключ в незашифрованном виде. Элдор слушает связь между Анваром и Сабиром. Он получает ключ одновременно с Сабиром. С тех пор Анвар и Сабир использовали этот ключ, а Элдор слушаль их.

Шифрование с открытым ключом (или асимметричное) использует разные ключи для шифрования и дешифрования данных. Открытый (публичный ключ) доступен всем. Используется для шифрования данных при обращении браузера к серверу. Этот ключ может быть передан любому человеку. Если Анвар хочет отправить сообщение Сабиру, он просит у Сабира свой открытый ключ. Собир отправляет его по незашифрованному каналу связи. Анвар использует открытый ключ Сабира для шифрования сообщения и отправляет его ему. Если Элдор узнает от Собира открытый ключ, всё равно он не можеть получить сообщение.

Асимметричное шифрование также можно использовать для аутентификации и проверки целостности сообщений. Для этого Анвару пришлось зашифровать его с помощью закрытого ключа перед отправкой. Когда Сабир принимает это, Анвар расшифровывает его с помощью открытого ключа. После расшифровки Сабир понял, что сообщение пришло от Анвара. В конце концов, никто другой не мог создать должным образом зашифрованное сообщение с помощью своего открытого ключа. Сабир также знает, что сообщение не было изменено Элдором недобросовестно, или что оно не было изменено косвенной ошибкой программы или сетевым

вмешательством, поскольку такое изменение привело бы к нарушению дешифрования. Приложив немного больше усилий, Анвар может зашифровать сообщение дважды, первый раз с помощью своего закрытого ключа, а второй раз с помощью открытого ключа Сабира, что обеспечивает конфиденциальность, аутентификацию и целостность.

На практике шифрование с открытым ключом медленнее, чем шифрование с закрытым ключом. Поэтому вместо того, чтобы шифровать весь перевод открытым ключом Сабира, Анвар шифрует традиционный закрытый ключ и отправляет его Сабиру. Собир открывает его приватным ключом. Теперь Анвар и Сабир знают запертый ключ, но не Эльдор. Таким образом, Сабир и Анвар теперь могут использовать быстрое шифрование с закрытым ключом для конфиденциального общения без прослушивания Элдором.

Например, теория и практика шифрования и аутентификации, а также тот факт, что и алгоритмы, и протоколы полны мин и ловушек, поражающих криптографию. Очень легко разработать хороший алгоритм или протокол шифрования. И не всегда понятно, какие алгоритмы и протоколы хорошие, а какие плохие. К счастью, вам не нужно быть криптографом, чтобы использовать мощную криптографию в сетевых приложениях Java. JSSE защищает вас от низкоуровневых подробностей о том, как работают алгоритмы, подмена ключей, проверка сторон и шифрование данных. JSSE позволяет создавать клиентские и серверные сокеты, которые управляют паролями, необходимыми для безопасной связи. Расширение Java Secure Socket Extension разделено на четыре пакета:

javax.net.ssl - Абстрактные классы, определяющие API Java для безопасного сетевого подключения.

javax.net - Абстрактные классы, используемые вместо конструкторов для создания защищенных сокетов.

java.security.cert - Классы для управления сертификатами открытых ключей, необходимыми для SSL.

com.sun.net.ssl - Конкретные классы, реализующие алгоритмы и протоколы шифрования, используемые в JSSE. Технически они не являются частью стандарта JSSE.

Задание:

Студент получает индивидуальное задание на лабораторную работу. По заданию студенты создают безопасные сокеты в сети.

Порядок выполнения работы:

Шаг 1. Начнем с загрузки интегрированной среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе левой кнопкой мыши.

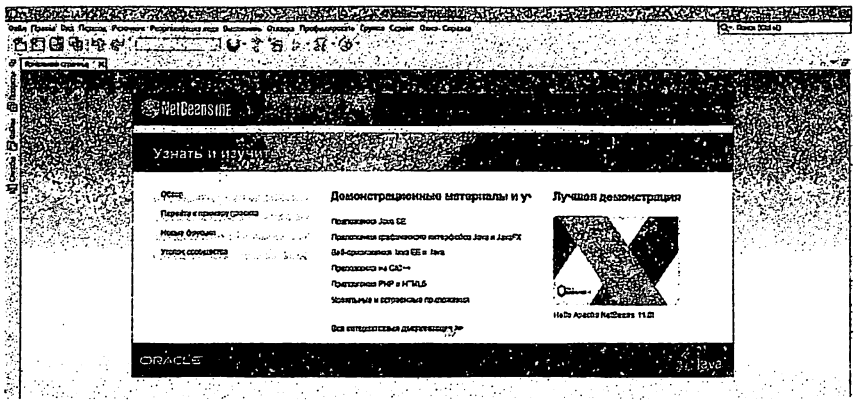


Рисунок 10.1. Главное окно среды NetBeans IDE

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

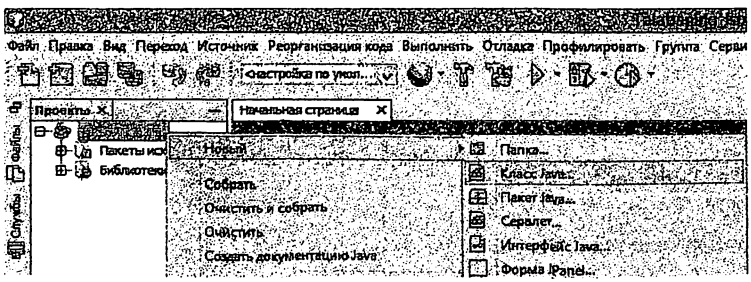


Рисунок 10.2. Создание нового класса в среде NetBeans IDE

Шаг 3. Затем щелкните правой кнопкой мыши над проектом «Имя_студента» и выберите «Новый» → «Класс Java» из контекстного меню.

В поле «Имя класса», введите «Client», в поле «Пакет» введите «ssl» и нажмите «Готово».

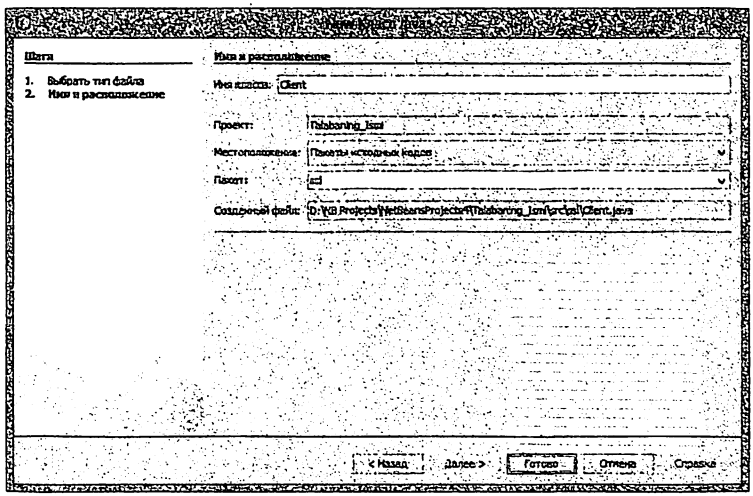


Рисунок 10.3. Окно «New Класс Java» среды NetBeans IDE

Шаг.4. Следующий код Java добавляется в файл Client.java

```
package ssl;

import java.io.*;

import javax.net.ssl.*;
import com.sun.net.ssl.*;
import com.sun.net.ssl.internal.ssl.Provider;
import java.security.Security;

public class Client {

    public static void main(String[] args) throws Exception{
        String strServerName = "localhost"; // SSL Server nomi
        int intSSLport = 4443; // SSL eshitadigan port raqami
        PrintWriter out = null;
  BufferedReader in = null;

        {
```

```
                // JSSE provayderni ro'yxatdan o'tkazish
                Security.addProvider(new Provider());
            }

            try {
                // Kliyent soketini yaratish
                SSLSocketFactory sslsocketfactory =
(SSLSocketFactory)SSLSocketFactory.getDefault();
                SSLSocket sslSocket =
(SSLSocket)sslsocketfactory.createSocket(strServerName,intSSLport);

        // Server bilan aloqa qiladigan oqimlarni yaratish
        out = new PrintWriter(sslSocket.getOutputStream(), true);
        in = new BufferedReader(new
InputStreamReader(sslSocket.getInputStream()));

                BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));
                String userInput = "Salom ";
                out.println(userInput);

                while ((userInput = stdIn.readLine()) != null) {
                   out.println(userInput);
                   System.out.println("echo: " + in.readLine());
                }

                    out.println(userInput);

                    // Closing the Streams and the Socket
                    out.close();
                    in.close();
                    stdIn.close();
                    sslSocket.close();
            }

            catch(Exception exp)
            {
                System.out.println(" Xatolik ro'y berdi .... " +exp);
                exp.printStackTrace();
            }
        }
}
```

Повторите шаги 3 и 4, чтобы создать файл Server.java.

```
package ssl;
import java.io.*;
import java.security.Security;
import java.security.PrivilegedActionException;

import javax.net.ssl.*;
import com.sun.net.ssl.*;
import com.sun.net.ssl.internal.ssl.Provider;

public class Server {

    public static void main(String[] args) throws Exception{

        int intSSLport = 4443; // Kliyentlarni tinglaydigan port raqami

        {
            // JSSE provayderni ro'yxatdan o'tkazish
            Security.addProvider(new Provider());

            //Specifying the Keystore details
            System.setProperty("javax.net.ssl.keyStore","testkeystore.ks");

    System.setProperty("javax.net.ssl.keyStorePassword","123456");
        }

        try {
                    // Server soketini yaratish
                    SSLServerSocketFactory sslServerSocketfactory =
(SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
                    SSLServerSocket sslServerSocket =
(SSLServerSocket)sslServerSocketfactory.createServerSocket(intSSLport);
                    SSLSocket sslSocket =
(SSLSocket)sslServerSocket.accept();

                    // Kliyent bilan aloqa qiladigan kiritish/chiqarish
oqimlarini yaratish
                    while(true)
                    {
                    PrintWriter out = new
PrintWriter(sslSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                                new InputStreamReader(
                                    sslSocket.getInputStream()));
            String inputLine, outputLine;
```

```
            while ((inputLine = in.readLine()) != null) {
                out.println(inputLine);
                System.out.println(inputLine);
            }
            // Oqimlar va soketni yopish
            out.close();
            in.close();
            sslSocket.close();
            sslServerSocket.close();}}
        catch(Exception exp)
          {
            PrivilegedActionException priexp = new
PrivilegedActionException(exp);
            System.out.println(" Priv exp --- " + priexp.getMessage());

            System.out.println(" Xatolik ro'y berdi .... " +exp);
            exp.printStackTrace();
        }}}
```

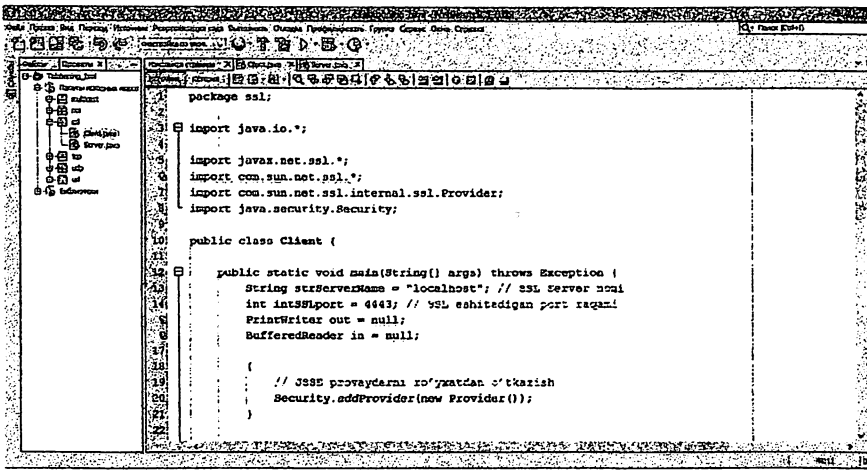


Рисунок 10.4. Среда NetBeans IDE

Контрольные вопросы

1. Библиотека безопасности Java
2. Java Secure Socket Extension (JSSE)
3. JSSE стандарт API
4. Создать SSLSocket & SSLEngine

5. Библиотека Java Security
6. Datagram Transport Layer Security
7. Transport Layer Security
8. Secure Socket Layer
9. Пакет javax.net
10. Пакет javax.net.ssl

ЛАБОРАТОРНАЯ РАБОТА № 11

Тема: Создание многопоточного сетевого приложения

Цель работы:

Формирование практических навыков у студентов по созданию многопоточного сетевого приложения с использованием классов пакетов java.net. *, java.io. * и java.util.* языка программирования Java.

Теоретическая часть:

Многопоточность в Java - это процесс одновременного выполнения нескольких потоков. Поток наименьшая единица процесса. Многопроцессорность и многопоточность используются для достижения многозадачности. Однако мы используем многопоточность, а не многопроцессорность, потому что потоки используют общую область памяти. В Java многопоточность используется в основном в играх и анимациях.

Многозадачность - это процесс одновременного выполнения нескольких задач. Многозадачность достигается двумя способами:

- многозадачность на основе процессов;
- многозадачность на основе потоков.

1. Многозадачность на основе процессов
   - у каждого процесса есть адрес в памяти;
   - процесс тяжелый;

- стоимость связи между процессами высока;
- для переключения с одного процесса на другой требуется некоторое время.

2. Многозадачность на основе потоков
   - потоки используют одно и то же адресное пространство;
   - поток легкий;
   - стоимость связи между потоками невысока.

Потоки независимы. Если в одном потоке возникает исключение, это не влияет на другие потоки. Он использует общую область памяти.

Как показано на рисунке 11.1 внутри процесса выполняется поток. Между потоками происходит переключение контекста. Внутри операционной системы может быть несколько процессов, и один процесс может иметь несколько потоков.

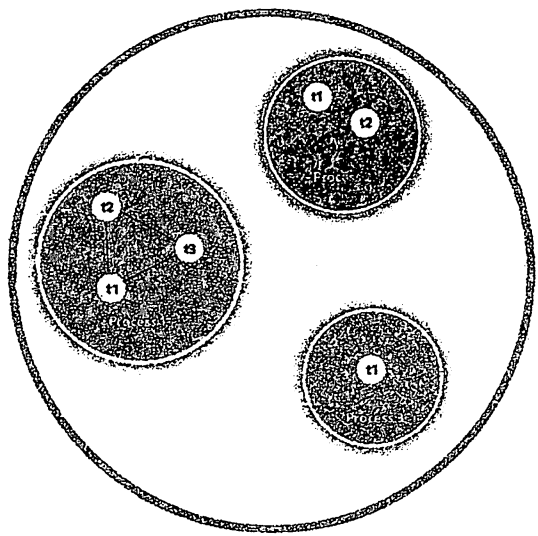


Рисунок 11.1. Потоки в Java

Жизненный цикл потока:

1. New.

2. Runnable.
3. Running.
4. Non-Runnable (Blocked).
5. Terminated.

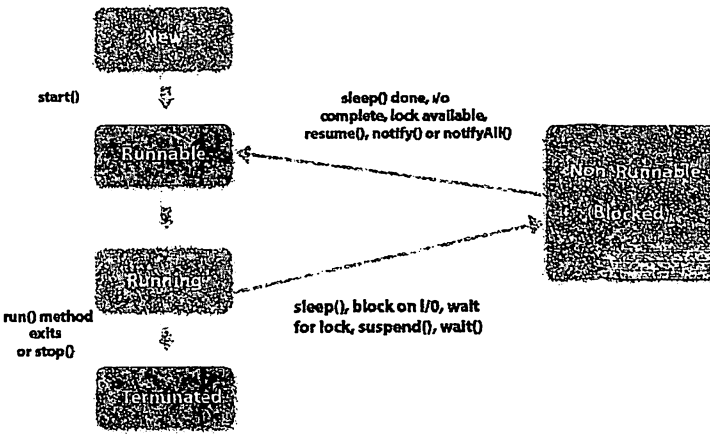


Рисунок 11.2. Жизненный цикл потока

Потоки создаются двумя способами:

1. Расширяя класс Thread
2. Путем реализации интерфейса Runnable

Создание потока с использованием класса Thread:

```
class Multi extends Thread{
public void run(){
System.out.println("Oqim ishga tushdi... ");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
}}
```

РЕЗУЛЬТАТ: трансляция началась ...

Создание потока на основе интерфейса Runnable:

```
class Multi3 implements Runnable{
public void run(){
System.out.println("Oqim ishga tushdi...");
}
 public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
} }
```

Метод Sleep в Java.

Метод sleep () класса потока Thread используется для перевода потока в спящий режим на некоторое время.

Синтаксис метода sleep ():

- public static void sleep(long milliseconds) throws InterruptedException
- public static void sleep(long milliseconds, int nanos) throws InterruptedException

Пример:

```
class TestSleepMethod1 extends Thread{
 public void run(){
 for(int i=1;i<5;i++){
   try{Thread.sleep(500);}
   catch(InterruptedException e){System.out.println(e);}
    System.out.println(i);   } }
 public static void main(String args[]){
  TestSleepMethod1 t1=new TestSleepMethod1();
  TestSleepMethod1 t2=new TestSleepMethod1();
   t1.start();
  t2.start();   } }
```

ThreadGroup в Java.

Java предоставляет несколько групп потоков в одном объекте.

Java thread group *java.lang.* Реализовано на основе класса *ThreadGroup.*

Таблица 11.1

Конструкторы класса ThreadGroup

Конструктор	Описание
ThreadGroup(String name)	Создает группу потоков с заданным именем.
ThreadGroup(ThreadGroup parent, String name)	Создает группу потоков с заданной родительской группой и именем.

Таблица 11.2

Основные методы класса ThreadGroup

Метод	Описание
int activeCount()	Возвращает количество потоков, работающих в текущей группе.
int activeGroupCount()	Возвращает количество активных групп в текущей группе потока.
void destroy()	Удалите текущую группу потока и все подгруппы.
String getName()	Вернуть текущее имя группы.
ThreadGroup getParent()	Вернуть родителя текущей группы.
void interrupt()	Остановить текущие групповые потоки.
void list()	Вывести данные текущей группы в стандартную консоль.

1. ThreadGroup tg1 = new ThreadGroup("Group A");
2. Thread t1 = new Thread(tg1,new MyRunnable(),"один");
3. Thread t2 = new Thread(tg1,new MyRunnable(),"два");
4. Thread t3 = new Thread(tg1,new MyRunnable(),"три");

Указанные выше 3 потока принадлежат одной группе. Вот имя группы потока tg1, класс, реализованный из интерфейса MyRunnable Runnable, и

имена потоков «один», «два» и «три».

Все потоки можно остановить в одной строке.

Thread.currentThread().getThreadGroup().interrupt();

ThreadGroup является примером

```
public class ThreadGroupDemo implements Runnable{
  public void run() {
      System.out.println(Thread.currentThread().getName());
  }
  public static void main(String[] args) {
    ThreadGroupDemo runnable = new ThreadGroupDemo();
      ThreadGroup tg1 = new ThreadGroup("Parent ThreadGroup");
      Thread t1 = new Thread(tg1, runnable,"один");
      t1.start();
      Thread t2 = new Thread(tg1, runnable,"два");
      t2.start();
      Thread t3 = new Thread(tg1, runnable,"три");
      t3.start();
      System.out.println("имя группы потока: "+tg1.getName());
     tg1.list();
  } }
```

Задание:

Студент получает индивидуальное задание на выполнение лабораторных работ. По заданию студент создаёт многопоточную сетевую программу. Индивидуальные задания для каждого студента приведены в таблице 11.3.

Таблица 11.3.

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.	Вычислить сумму положительных чисел из любых n чисел
19.	Вывести текст в обратном порядке
20.	Найти объем произвольного куба
21.	Выделять 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 NetBeans IDE.

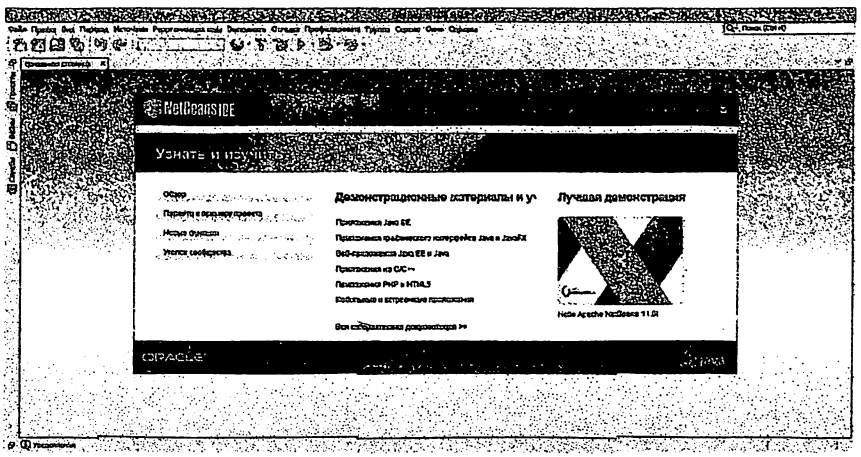


Рисунок 11.3. Главное окно среды NetBeans IDE

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

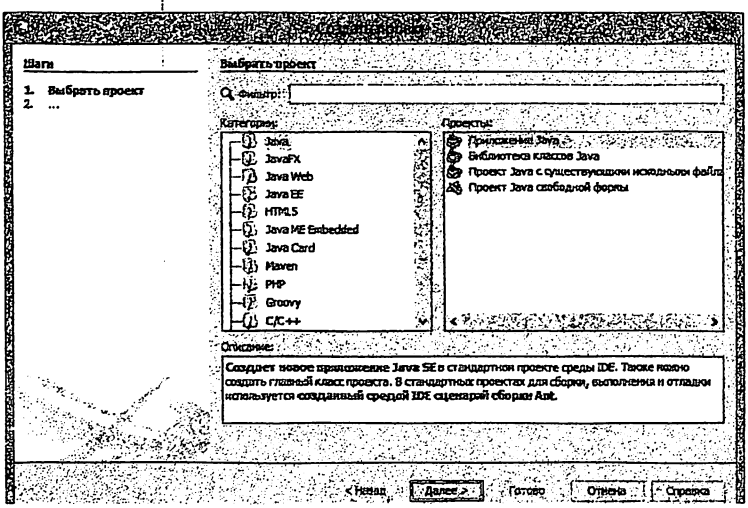


Рисунок 11.4. Окно "Создать проект" NetBeans IDE

Шаг 3. В поле «Имя проекта» этого окна введите Student_Name (имя студента) в качестве имени проекта, и снимите флажок «Создать главный класс» и нажмите «Готово».

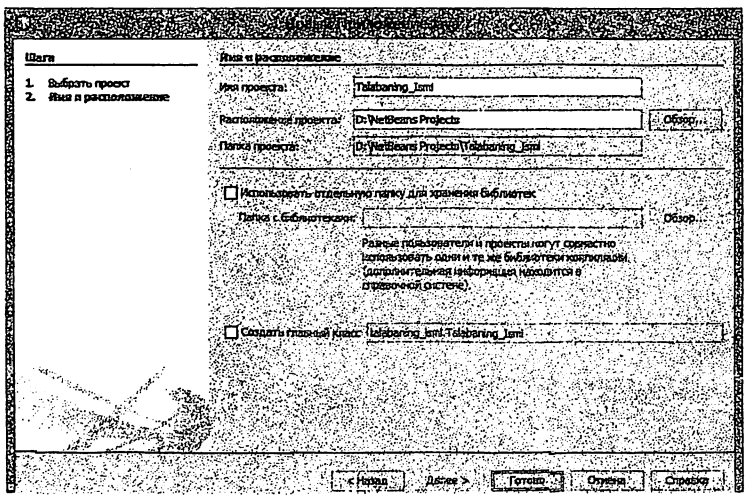


Рисунок 11.5. Окно «Новое приложение Java» среда NetBeans IDE

Шаг 4. Затем щелкните правой кнопкой мыши над созданным проектом и выберите «Новый»→ «Класс Java» из контекстного меню.

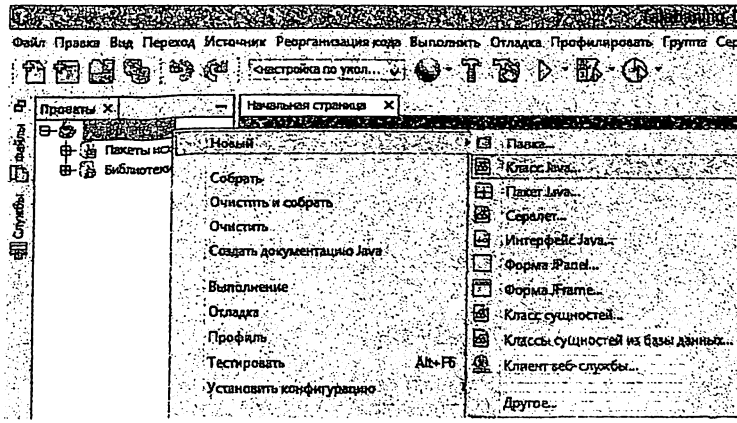


Рисунок 11.6. Создание нового класса в среде NetBeans IDE

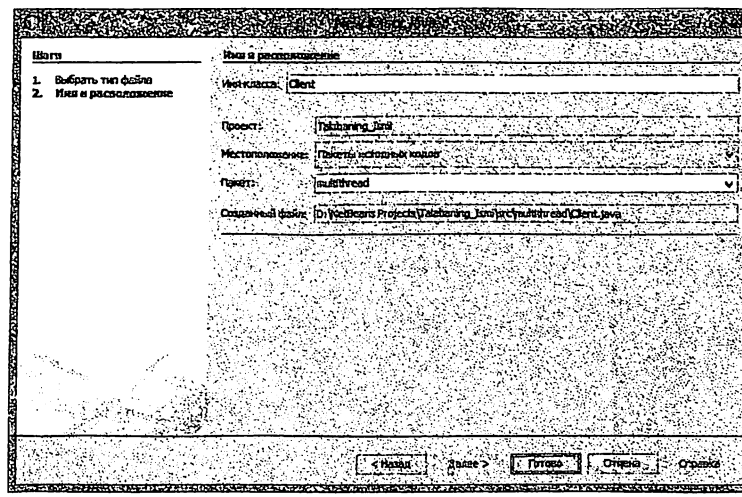


Рисунок 11.7. Окно «New Класс Java» среды NetBeans IDE

Шаг 5. В окне «New Класс Java» введите «Client» в поле «Имя класса», «multithread» в поле «Пакет» и нажмите «Готово».

Шаг 6. Следующий код Java вводится в созданный файл Client.java.

```
package multithread;
import java.io.*;
import java.net.*;
import java.util.*;
public class Client {
  private static InetAddress host;
  private static final int PORT = 1234;
  public static void main(String[] args) {
    try {
      host = InetAddress.getLocalHost();
    } catch (UnknownHostException uhEx) {
      System.out.println("\nID host topilmadi!\n");
      System.exit(1);
    }
    sendMessages();
  }
  private static void sendMessages() {
    Socket socket = null;
    try {
      socket = new Socket(host, PORT);
      Scanner networkInput = new Scanner(socket.getInputStream());
      PrintWriter networkOutput = new PrintWriter(socket.getOutputStream(),
true);
      Scanner userEntry = new Scanner(System.in);
      String message, response;
      do {
        System.out.print("\nXabarni kiriting (Chiqish uchun 'Q'): ");
        message = userEntry.nextLine();
        networkOutput.println(message);
        response = networkInput.nextLine();
```

```
            System.out.println("SERVER> " + response);
        } while (!message.equals("Q"));
    } catch (IOException ioEx) {
        ioEx.printStackTrace();
    } finally {
        try {
            System.out.println("\nBog'lanish uzildi...");
            socket.close();
        } catch (IOException ioEx) {
            System.out.println("Bog'lanish uzilmadi!");
            System.exit(1);
        }
    }
  }
}
```

4-5 шаги выполняются повторно и создается файл Server.java и вводится следующий код Java.

```
package multithread;
import java.io.*;
import java.net.*;
import java.util.*;
public class Server {
  private static ServerSocket serverSocket;
  private static final int PORT = 1234;
  public static void main(String[] args)
      throws IOException {
    try {
      serverSocket = new ServerSocket(PORT);
      System.out.println("\nServer soket ishga tushdi...\nKliyentni kutmoqda...");
    } catch (IOException ioEx) {
```

```
            System.out.println("\nPortni o'rnata olmadim!");
            System.exit(1);
        }
        do {
            Socket client = serverSocket.accept();
            System.out.println("Yangi kliyent qabul qilindi.");
            ClientHandler handler = new ClientHandler(client);
            handler.start();
        } while (true);
    }
}
class ClientHandler extends Thread {
    private Socket client;
    private Scanner input;
    private PrintWriter output;
    public ClientHandler(Socket socket) {
        client = socket;
        try {
            input = new Scanner(client.getInputStream());
            output = new PrintWriter(client.getOutputStream(), true);
        } catch (IOException ioEx) {
            ioEx.printStackTrace();
        }
    }
    public void run() {
        String received;
        do {
            received = input.nextLine();
            output.println("AKS-SADO: " + received);
            //Repeat above until 'QUIT' sent by client...
```

```
    } while (!received.equals("Q"));
    try {
      if (client != null) {
        System.out.println("Bog'lanish uzildi");
        client.close();
      }
    } catch (IOException ioEx) {
      System.out.println("Bog'lanish uzilmadi!");
    }
  }
}
```

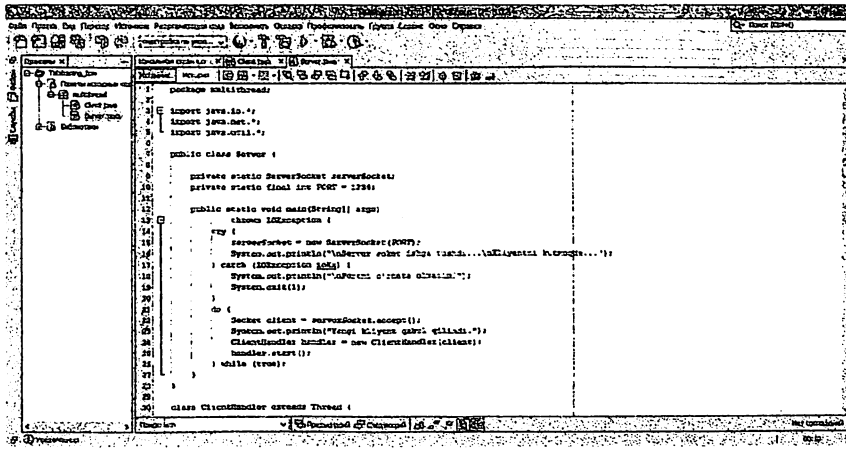


Рисунок 11.8. Файлы Client.java и Server.java в среде NetBeans IDE

Запуск сетевого приложения начинается с класса Server.java. Для этого щелкните правой кнопкой мыши файл Server.java и выберите «Выполнить файл» в появившемся контекстном меню. Запускается аналогичный файл Client.java.

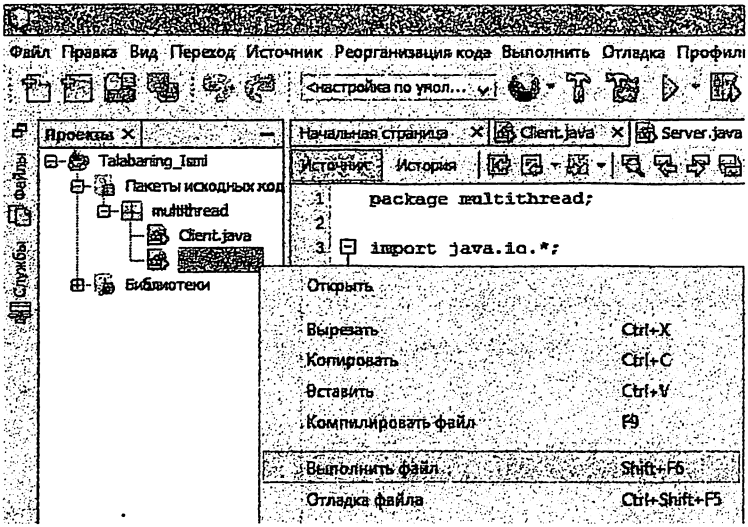


Рисунок 11.9. Запустить Server.java в среде NetBeans IDE

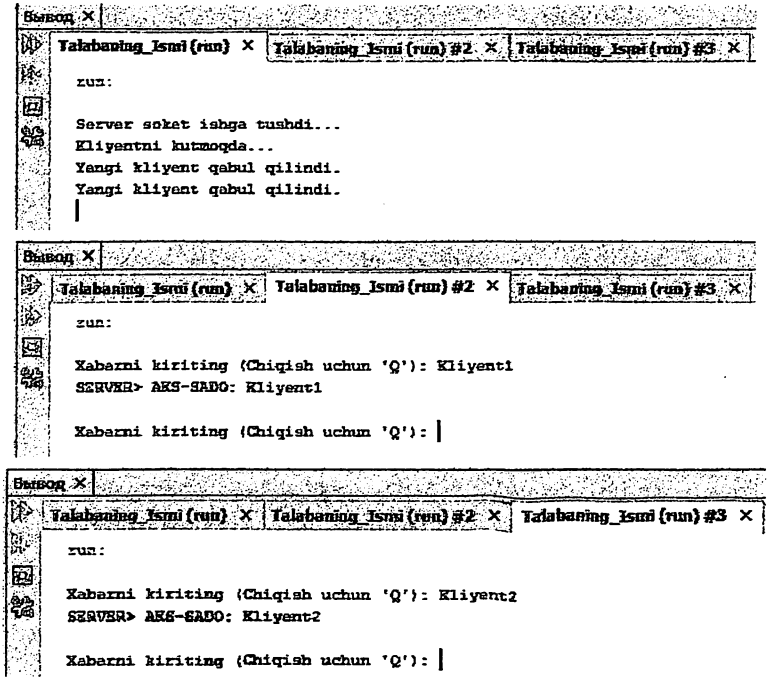


Рисунок 11.10. Окна результатов в среде NetBeans IDE

Контрольные вопросы

1. Классы и методы пакета java.net.*.
2. Классы и методы пакета java.io.*.
3. Классы и методы пакета java.util.*.
4. Работа класса Thread и его методы.
5. Работа интерфейса Runnable и его методы.
6. Работа метода sleep().
7. Работа класса ThreadGroup и его методы.
8. Работа класса InputStream и его методы.
9. Работа класса OutputStream и его методы.

ЛАБОРАТОРНАЯ РАБОТА № 12

Тема: Создание распределенных клиент-серверных приложений

Цель работы:

Формирование практических навыков у студентов по созданию распределенных клиент-серверных приложений с использованием классов пакета java.rmi.* языка программирования Java

Теоретическая часть:

RMI (Remote Method Invocation) – это API, который предоставляет механизм для создания распределенного приложения на Java. RMI позволяет объекту вызывать методы объекта, запущенного в другой JVM. Удаленный вызов метода обеспечивает удаленную связь между приложениями с использованием двух объектов.

RMI позволяет объектам вызывать управляемые методы на других виртуальных машинах Java (JVM). RMI использует два объекта, заглушку и каркас, для обеспечения удаленной связи между приложениями. RMI использует объекты-заглушки и каркасы для связи с удаленными объектами. Удаленные объекты — это объекты, которые извлекают методы из других

виртуальных машин в Java. Заглушка — это объект, который клиент выступает в качестве шлюза. Все исходящие запросы перенаправляются через него. Он находится на стороне клиента и участвует в удаленных объектах. Когда вызывающая сторона вызывает метод из объекта-заглушки, он выполняет следующие функции:

1 Начинает обмениваться данными с удаленной виртуальной машиной.
2 Записывает и передает параметры на удаленную виртуальную машину.
3 Пересчитывает значения.

И наконец, возвращает значение вызывающей стороне. Скелетон — это объект, который действует как шлюз на стороне сервера. Все входящие запросы перенаправляются через него. Когда Скелетон получает входящий запрос, он выполняет следующие функции:

1 Считывает параметры для удаленных методов.
2 Вызывает метод объекта реального расстояния, передает и записывает результат вызывающему объекту.

В приложении RMI клиент также взаимодействует с удаленным интерфейсом сервера. Клиентское приложение вызывает метод прокси-объекта, а RMI отправляет запросы к виртуальным машинам на языке Java. Затем возвращаемое значение отправляется обратно клиентскому приложению для прокси-объекта.

Java поддерживает распределение устройств управления на нескольких компьютерах посредством удаленного вызова метода. Этот набор распределенных объектов упрощает взаимодействие Java-приложений на нескольких машинах. Ниже мы суммируем шаги, необходимые для создания программы RMI. Вот четыре примера RMI:

1. Простой пример RMI, который возвращает серию сообщений от удаленного объекта.

2. Наглядный пример выполнения цифровой интеграции через удаленный объект.

3. Программа RMI соединяет удаленные объекты.

Этапы создания приложения RMI. Чтобы использовать RMI, нам нужно сделать две вещи: построить четыре класса и завершить пять этапов. Мы кратко объясним классы и этапы. При использовании RMI нам нужно построить четыре основных класса:

1. Интерфейс для удаленного объекта. Этот интерфейс используется как клиентом, так и сервером.
2. RMI-клиент – клиент ищет объект на удаленном сервере, а затем использует этот объект как локальный объект.
3. Реализация объекта – объект должен быть реализован на первом этапе интерфейса и используется сервером.
4. RMI-сервер – этот класс создает экземпляр для объекта и регистрирует объект.

Скомпилируйте и запустите систему.

1. Компиляция клиента и сервера – на этом этапе компилируется интерфейс удаленного объекта.
2. Создание клиентской заглушки и склетон сервера – клиентская заглушка и сервер поддерживают скелетонный метод вызова и обеспечивают независимое кодирование устройств. Клиентской системе нужен клиентский класс, интерфейс класса и клиентский класс-заглушка. Серверной системе нужен класс сервера, интерфейс удаленных объектов и скелетонный класс сервера.
3. Начать регистрацию RMI - Регистрация производится один раз и не для каждого удаленного объекта.
4. Запустить сервер – этот шаг выполняется на тех же устройствах.
5. Запустить клиента – этот этап выполняется на автономных устройствах.

Общая информация для приложений RMI.

В приложения RMI включены два отдельных приложения: серверное и клиентское. Как правило, серверное приложение создает некоторые

удаленные объекты, предоставляет на них ссылки и ожидает, пока клиенты вызовут методы для этих удаленных объектов. Типичное клиентское приложение предоставляет одно или несколько удаленных устройств на сервере, а затем вызывает их методы. RMI позволяет серверу и клиенту перемещать информацию туда и обратно. Эту программу иногда называют приложением распределенных объектов.

Приложениям с распределенными объектами необходимы:

Чтобы найти удаленные объекты: Приложения используют один из двух механизмов доступа к ресурсам, которые можно применять к удаленным объектам. Приложение регистрирует удаленный объект с помощью простой системы именования RMI.

Контакт с удаленным объектом: Информация о связи между удаленными объектами рассматривается RMI. Удаленная связь с программистом аналогична вызову стандартного метода Java.

- *Скачать байт-коды классов для объектов:*

Поскольку RMI позволяет удаленно перемещать устройства вызывающей стороны к объектам, RMI предоставляет необходимые механизмы для установки кода объекта, а также для передачи его данных. На следующем рисунке показано распределенное приложение RMI. Он используется для перечисления ресурсов, используемых на удаленных объектах. Это делается путем регистрации имени сервера, чтобы связать его с удаленным объектом. Клиент ищет удаленный объект по его имени. На рисунке система RMI также использует существующий веб-сервер от сервера к клиенту и от клиента к серверу для установки байт-кодов для объектов по мере необходимости.

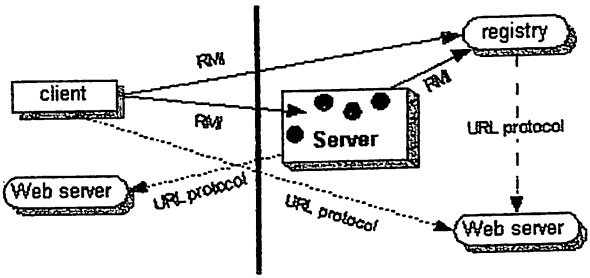


Рисунок 12.1. Интерфейс создания приложений RMI

Преимущества динамической загрузки кода:

Одной из особенностей RMI является возможность просто загрузить код класса объекта, если класс не обнаружен на виртуальной машине получателя. Типы и поведение объекта, которые ранее существовали на одной виртуальной машине, можно перенести на другую, удаленную виртуальную машину. RMI преобразует объекты в их фактический тип, поэтому поведение этих объектов не меняется при их отправке на другую виртуальную машину. Это позволяет удаленно добавлять новые типы в виртуальную машину, что позволяет приложению работать динамически.

Задание:

Студент получает индивидуальное задание на выполнение лабораторных работ. В этом задании студент создаст сетевое приложение клиент-сервер на основе распределенный системы.

Порядок выполнения работы:

Шаг 1. Начнем с загрузки интегрированной среды NetBeans IDE. Для этого дважды щелкните ярлык среды NetBeans IDE на рабочем столе левой кнопкой мыши.

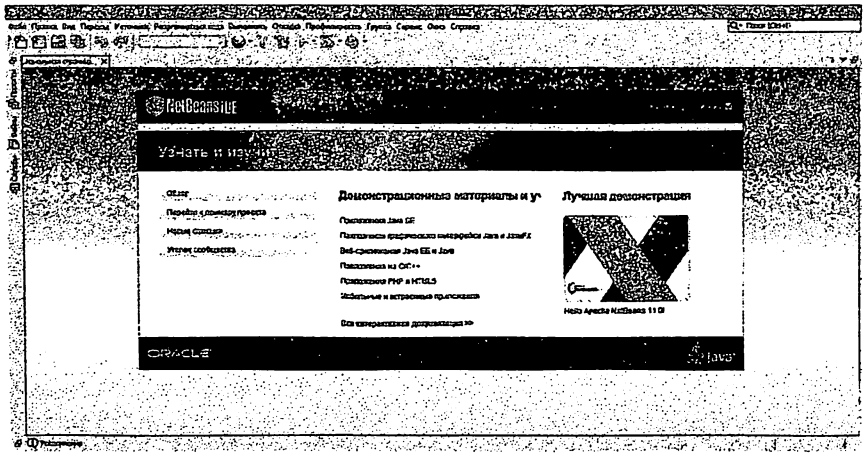


Рисунок 12.2. Главное окно среды NetBeans IDE

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

Шаг 3. Затем щелкните правой кнопкой мыши над проектом «Имя_студента» и выберите «Новый» → «Интерфейс Java» из контекстного меню.

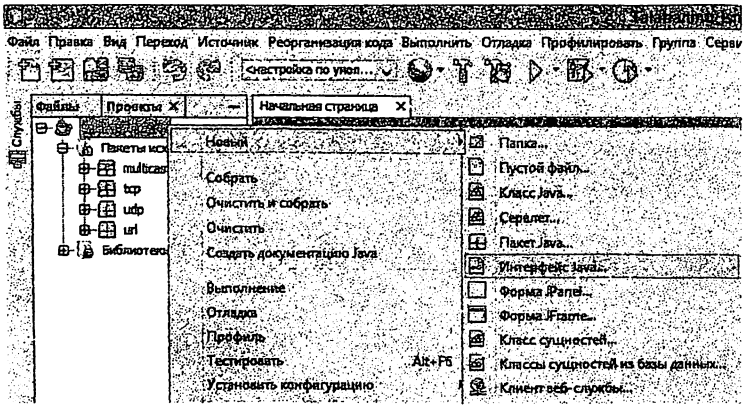


Рисунок 12.3. Создание нового класса в среде NetBeans IDE

Шаг 4. В окне «New Интерфейс Java» в поле «Имя класса» введите «BillingService», в поле «Пакет» введите «rmi» и нажмите кнопку «Готово».

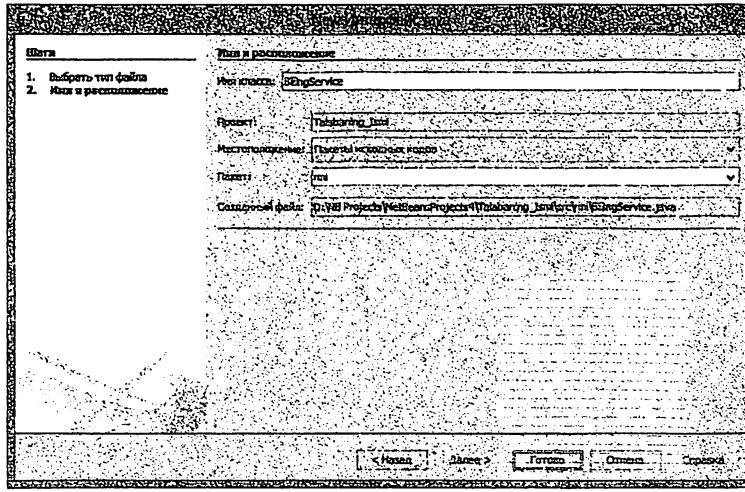


Рисунок 12.4. Окно «New Интерфейс Java» среды NetBeans IDE

Шаг 5. В созданный файл BillingService.java добавляется следующий код Java.

```
package rmi;

import java.rmi.*;

public interface BillingService extends Remote
{
    public void addNewCard(Card card) throws RemoteException;

    public void addMoney(Card card, double money) throws
RemoteException;

    public void subMoney(Card card, double money) throws
RemoteException;

    public double getCardBalance(Card card) throws RemoteException;
}
```

Шаг 6. Затем щелкните правой кнопкой мыши над проектом «Имя_студента» и выберите «Новый» → «Класс Java» из контекстного меню.

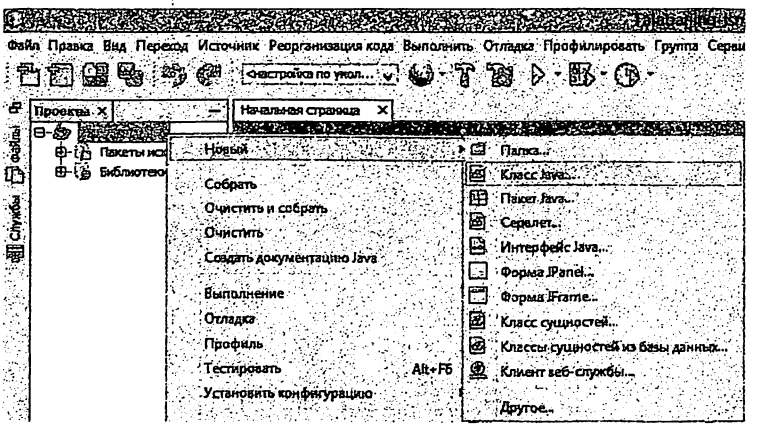


Рисунок 12.5. Создание нового класса в среде NetBeans IDE

Шаг 7. В окне «New класс Java» в поле «Имя класса» введите «Card», в поле «Пакет» введите «rmi» и нажмите «Готово».

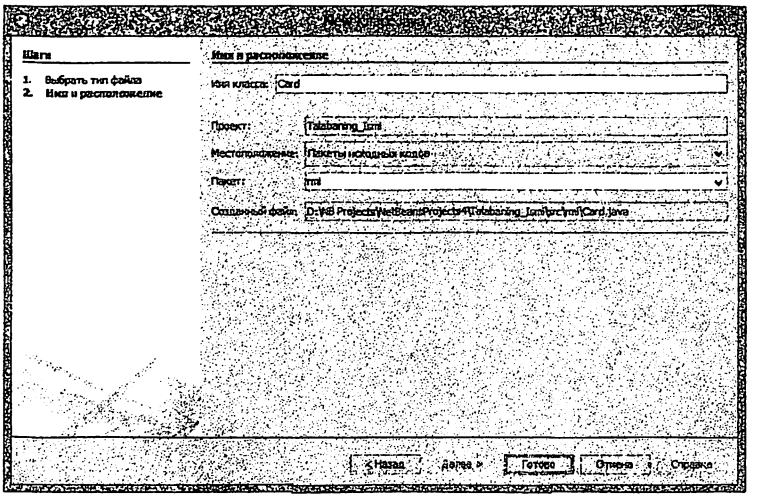


Рисунок 12.6. Окно «New Класс Java» среды NetBeans IDE

Шаг 8. В созданный файл Card.java добавляется следующий код Java.

```
package rmi;

import java.io.Serializable;

public class Card implements Serializable
{
	private static final long serialVersionUID = 1L;

	private String name ;
	private String number;
	private double money ;

	public Card(String name, String number, double money)
	{
		super();
		this.name   = name;
		this.number = number;
		this.money  = money;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getNumber() {
		return number;
	}
	public void setNumber(String number) {
		this.number = number;
	}
	public double getMoney() {
		return money;
	}
	public void setMoney(double money) {
		this.money = money;
	}
	@Override
	public boolean equals(Object card)
	{
		return
this.getNumber().equalsIgnoreCase(((Card)card).getNumber());
	}
```

```
}
```

Повторите шаги 6–8 выше, чтобы создать файлы BillingClient.java и BillingServiceImpl.java, и введите следующий код Java.

```
package rmi;

import java.net.MalformedURLException;
import java.rmi.*;

public class BillingClient
{
    private String localhost    = "127.0.0.1";
    private String RMI_HOSTNAME = "java.rmi.server.hostname";
    private String SERVICE_PATH = "rmi://localhost/BillingService";

    private String[][] CARDS    = {{"Salimov", "8600-0456-7890-5265"},
                                   {"Djurayev", "8600-6554-3210-5542"}};

    private double[]   MONEYS   = {135790.0, 24680.0};

    private Card createCard (final int idx)
    {
        return new Card(CARDS[idx][0], CARDS[idx][1], 0);
    }

    private void registerCards(BillingService bs)
    {
        for (int i = 0; i < CARDS.length; i++) {
            Card card = createCard (i);
            try {
                bs.addNewCard(card);
            } catch (RemoteException e) {
                System.err.println("RemoteException : " +
e.getMessage());
            }
        }
    }
    private void addMoney(BillingService bs)
    {
        for (int i = 0; i < CARDS.length; i++) {
            Card card = createCard (i);
            try {
                bs.addMoney(card, MONEYS[i]);
```

```
                } catch (RemoteException e) {
                    System.err.println("RemoteException : " +
e.getMessage());
                }
            }
        }
        private void getBalance(BillingService bs)
        {
            for (int i = 0; i < CARDS.length; i++) {
                Card card = createCard (i);
                try {
                    System.out.println("card : " + card.getNumber() +
                                       ", balance = " +
bs.getCardBalance(card));
                } catch (RemoteException e) {
                    System.err.println("RemoteException : " +
e.getMessage());
                }
            }
        }
        public BillingClient()
        {
            try {
                System.setProperty(RMI_HOSTNAME, localhost);

                String objectName = SERVICE_PATH;

                BillingService bs = (BillingService)
Naming.lookup(objectName);
                System.out.println("\nRegister cards ...");
                registerCards(bs);
                System.out.println("Add moneys ...");
                addMoney(bs);
                System.out.println("Get balance ...\n");
                getBalance(bs);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (RemoteException e) {
                e.printStackTrace();
            } catch (NotBoundException e) {
                e.printStackTrace();
                System.err.println("NotBoundException : " + e.getMessage());
            }
        }
```

```
    public static void main(String[] args)
    {
        new BillingClient();
        System.exit(0);
    }
}
```

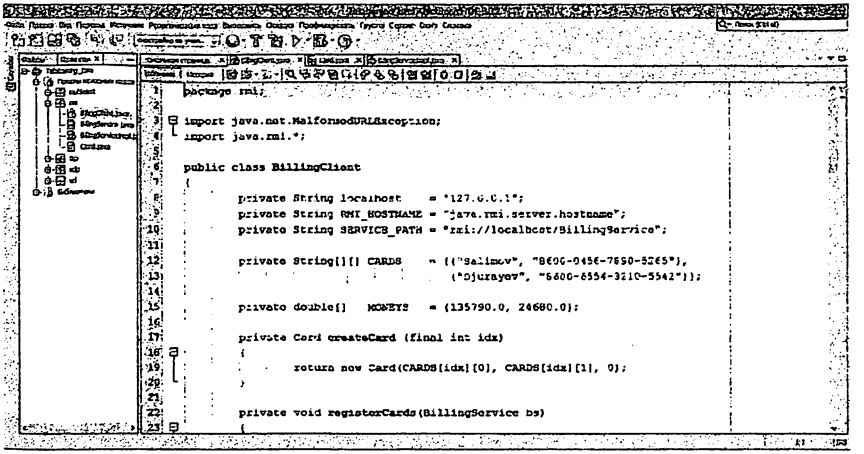


Рисунок 12.7. Среда NetBeans IDE

Контрольные вопросы

1. Common Object Request Broker Architecture (CORBA).
2. Object Management Group (OMG).
3. Remote Method Invocation (RMI).
4. Архитектура RMI.
5. Класс и методы пакета java.rmi.*
6. Класс и методы пакета java.rmi.server.*
7. Класс и методы пакета java.rmi.registry.*

ЛАБОРАТОРНАЯ РАБОТА № 13

Тема: Создание базы данных в сети

Цель работы:

Формирование практических навыков у студентов по созданию сетевого программного обеспечения с базой данных с использованием классов пакета java.sql.* языка программирования Java.

Теоретическая часть:

Для хранения и управления данными в более структурированном виде были внедрены базы данных- подмножества конкретной задачи реального мира, имеющие особое значение для определенной группы пользователей. Они работают под руководством системы управления базами данных (СУБД). Это набор программ, который обеспечивает процесс определения, конструирования и управления базами для различных приложений. В этой работе вы узнаете, как получить доступ к СУДБ с помощью Java.

JDBC (Java Database Connectivity) –популярный стандартизированный интерфейс прикладного программирования (API), который дает независимый от баз данных доступ к табличным данным, как правило, хранящимся в реляционных базах данных. Это позволяет установить связь с базой данных, обмениваться операторами SQL и обрабатывать результаты.

Классы JDBC и их назначение:

- *Statement* - определяет статический оператор SQL;
- *PreparedStatement* - определяет динамический оператор SQL;
- *CallableStatement* - выполняет сохраняемые процедуры в базе данных.

Создание экземпляра Statement.

Прежде, чем мы сможем использовать экземпляр Statement для выполнения SQL – запросов, нам необходимо создать такой экземпляр. Для этого используется метод Connection.createStatement(). В коде это выглядит таким образом:

```
try {
```

```
    statement =connection.createStatement();
  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    /*Do some job...*/
  }
```

После этго мы можем использовать наш экземпляр statement для выполнения SQL – запросов.

- Для этой цели интерфейс Statement имеет три метода, которые реализуются каждой конкретной реализацией JDBC драйвера:

- *boolean execute (String SQL)*

- Этот метод возвращает логическое значение *true,* если объект ResultSet может быть получен. В противном случае он возвращает *false.* Он используется для выполнения DDL SQL – запросов ил в случаях, когда мы используем динамический SQL.

- *int executeUpdate(String SQL)*

- Этот метода возвращает количество столбцов в таблице, на которое повлиял наш SQL – запрос. Мы используем этот метод для выполнения SQL – запросов, когда хотим получить количество задействованных столбцов, например, количество данных по определённому запросу.

- *ResultSet executeQuery (String SQL)*

Этот мтеод возвращает нам экземпляр ResultSet. Мы используем этот метод в случаях, когда мы рассчитываем получить множество объектов в результате выполнения нашего SQL – запроса. Например, при получении списка элементов, которые удовлетворяют определённым условиям.

Закрытие экземпляра Statement.

Когда мы закрываем наше соединение (Connection) для сохранения результатов в БД мы таким же образом закрываем и экремпляр Statement. Для этого мы используем метод close ().

Рассмотрим, как это выглядит в нашем коде:

```
Connection connection = null;
```

```
        Statement statement = null;
        Class.forName(JDBC_DRIVER);
    connection = DriverManager.getConnection(DATABASE_URL, USER,
PASSWORD);
        try {
          statement = connection.createStatement();
        } catch (SQLException e) {
          e.printStackTrace();
        } finally {
          if (statement != null) {
            statement.close();
          }
        }
```

Для понимания того, как это работает на практике, рассмотрим пример простого приложения, в котором мы попытаемся получить данные из БД.

Пример:

```
package jdbc;
import java.sql.*;
public class StatementDemo {
  static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
  static final String DATABASE_URL = "jdbc:mysql://localhost/talaba";
  static final String USER = "root";
  static final String PASSWORD = "admin";
  public static void main(String[] args) throws ClassNotFoundException,
SQLException {
    Connection connection = null;
    Statement statement = null;
    System.out.println("JDBC drayverni ro'yxatdan o'tkazish...");
    Class.forName(JDBC_DRIVER);
    System.out.println("Ma'lumotlar bazasiga bog'lanish...");
    connection = DriverManager.getConnection(DATABASE_URL, USER,
PASSWORD);
    System.out.println("So'rov yaratish...");
    statement = connection.createStatement();
    String sql = "SELECT * FROM developers";
    Boolean isRetrieved = statement.execute(sql);
    System.out.println("Ma'lumotlar olindi: " + isRetrieved);
    System.out.println("Olingan ma'lumotlarni ko'rsatish:");
    ResultSet resultSet = statement.executeQuery(sql);
    while (resultSet.next()) {
      int id = resultSet.getInt("id");
      String name = resultSet.getString("name");
      String specialty = resultSet.getString("specialty");
```

```
            int salary = resultSet.getInt("salary");
            System.out.println("id: " + id);
            System.out.println("Name: " + name);
            System.out.println("Specialty: " + specialty);
            System.out.println("Salary: " + salary);
            System.out.println("==================");
        }
        System.out.println("Bog'lanishni yopish...");
        try {
            resultSet.close();
            statement.close();
            connection.close();
        }finally {
            if(statement !=null){
                statement.close();
            }
            if(connection!=null){
                connection.close();
            }
        }
        System.out.println("Raxmat");
    }
}
```

Создание экземпляра PreparedStatement.

PreparedStatement наследует интерфейс Statement, что даёт нам определённые преимущества перед обычным Statement. В частности, мы получаем большую гибкость при динамической поддержке аргументов. Вот как выглядит создание экземпляра PreparedStatement на практике:

```
try {
    String SQL = "Update developers SET salary WHERE specialty = ?";
      preparedStatement = connection.prepareStatement(SQL);
    }catch (SQLException e){
      e.printStackTrace();
    }finally {
      /*do some job... */
    }
```

Все параметры, которые отмечены символом ? называются маркерами параметра. Это означает, что они должны быть переданы через параметры метода. Каждый параметр ссылается на свой порядковый номер в сигнатуре

метода. Т.е. первый маркер находится на первом месте, второй – на втором и т.д. В отличие от массивов, здесь отсчёт идёт с 1. Это связано с особенностями реляционной модели, на которой и основана работа реляционных БД. Для выполнения SQL – запросов используются методы с такими же названиями (execute (), executeQuery, executeUpdate), которые несколько модифицированы.

Закрытие экземпляра PreparedStatement.

Когда мы закрываем наше соединение (Connection) для сохранения результатов в БД мы таким же образом закрываем и экземпляр PreparedStatement. Для этого мы используем метод close(). Рассмотрим, как это выглядит в нашем коде:

```
try {
    String SQL = "Update developers SET salary WHERE specialty = ?";
      preparedStatement = connection.prepareStatement(SQL);
  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if (preparedStatement != null) {
      preparedStatement.close();
    }
  }
```

Создание экземпляра CallableStatement.

Экземпляр CallableStatement используется для выполнения процедур, непосредоственно в самой БД.

Задание: Студент получает индивидуальное задание в лабораторных работах. Для этой задачи студент создаёт сетевую программу работающуют с базой данных.

Порядок выполнения работ:

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.

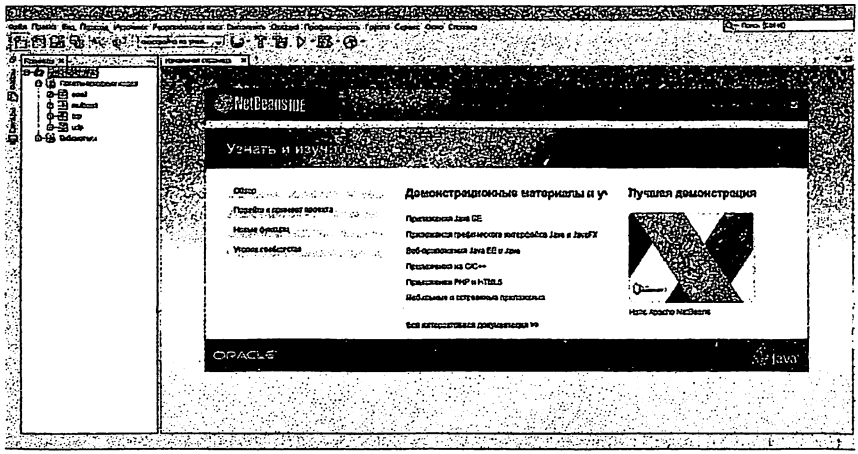


Рисунок 13.1. Главное окно среды NetBeans IDE

Шаг 2. Надо щелкнуть правой кнопкой мыши над проектом и выбрать «Новый» → «Класс Java».

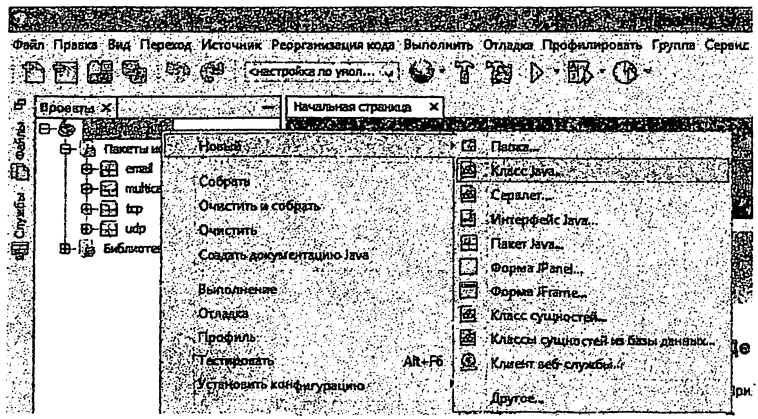


Рисунок 13.2. Создание нового класса Java

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

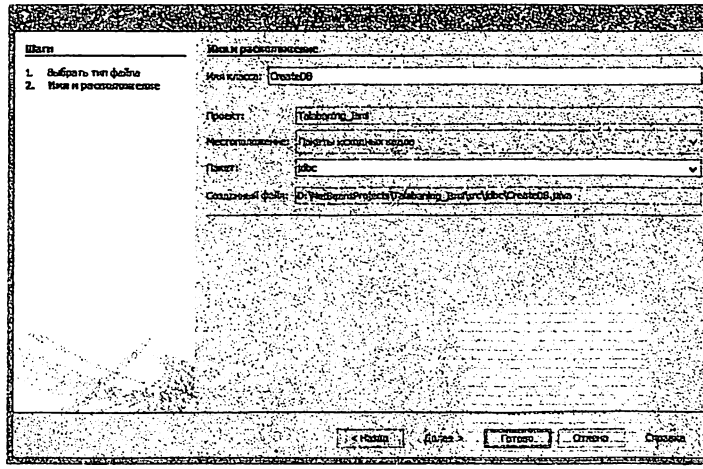


Рисунок 13.3. Окно «New Класс Java» среды NetBeans IDE

Шаг 4. Добавьте в свой класс следующие:

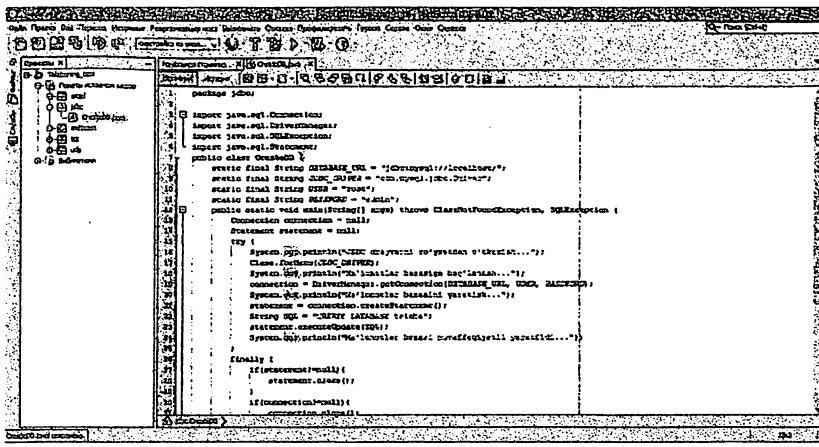


Рисунок 13.4. Создание базы данных

В Java коде необходимо выполнить следующее:

- Нужно импортировать пакет java.sql.*;
- Регистрация драйвера JDBC;

- Установить соединение;
- Создание новой базы данных;
- Запуск запроса.

код Java:

```
package jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class CreateDB {
   static final String DATABASE_URL = "jdbc:mysql://localhost/";
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String USER = "root";
   static final String PASSWORD = "admin";
   public static void main(String[] args) throws ClassNotFoundException,
SQLException {
      Connection connection = null;
      Statement statement = null;
      try {
         System.out.println("JDBC drayverni ro'yxatdan o'tkazish...");
         Class.forName(JDBC_DRIVER);
         System.out.println("Ma'lumotlar bazasiga bog'lanish...");
         connection = DriverManager.getConnection(DATABASE_URL, USER,
PASSWORD);
         System.out.println("Ma'lumotlar bazasini yaratish...");
         statement = connection.createStatement();
         String SQL = "CREATE DATABASE talaba";
         statement.executeUpdate(SQL);
         System.out.println("Ma'lumotlar bazasi muvaffaqiyatli yaratildi...");
      }
      finally {
         if(statement!=null){
            statement.close();
         }
         if(connection!=null){
            connection.close();
         }     }     }   }
```

Создание таблицы в базе данных

В среде NetBeans IDE файл CreateTable.java и InsertRecord.java создается как файл, созданный выше.

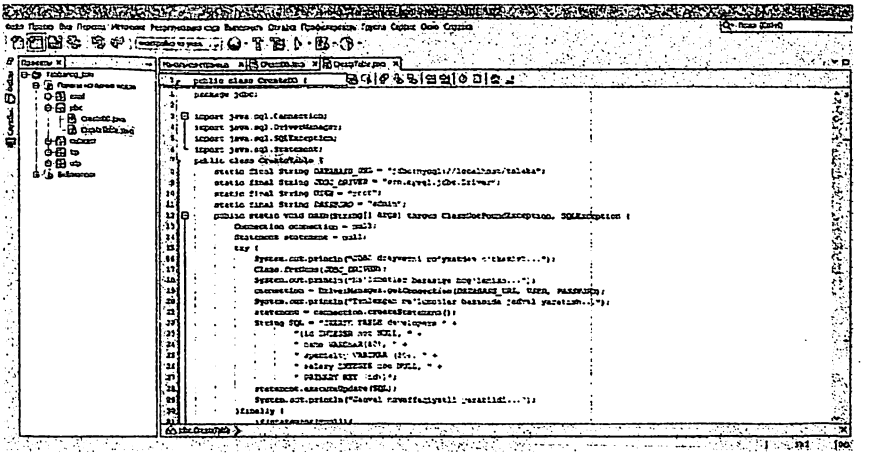


Рисунок 13.5. Создание таблицы

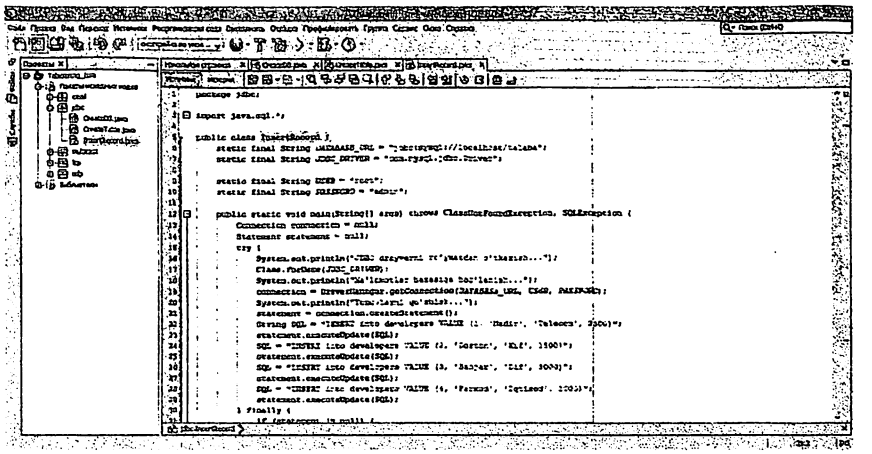


Рисунок 13.6. Добавление записей в таблицу

Контрольные вопросы

1. Java Database Connectivity
2. Интерфейс ResultSet
3. Драйверы JDBC
4. Основные компоненты JDBC и структура JDBC
5. Установки соединения с базой данных

ЛАБОРАТОРНАЯ РАБОТА № 14

Тема: Создание сервлетов на сервере

Цель работы:

Формирование практических навыков у студентов по созданию программы работающий с сервлетами с использованием классов пакетов javax.servlet.*, javax.servlet.http.* и java.io.* языка программирования Java.

Теоретическая часть:

Обработка запросов — это основа основ разработки веб-приложений Java. Чтобы отвечать на запросы из сети, веб-приложение Java сначала должно определить, какой код будет отвечать на URL-адрес запроса, а затем перенаправлять ответ. У каждого стека технологий есть способ выполнения обработки запроса-ответа. В Java для этой цели используется сервлеты. Servlet- это приложение, написанное на Java, которое работает на веб-сервере. Он может принимать данные от клиента, как правило через GET и POST-запросы, работать с cookie и параметрами сеанса.

На сервер приходит запрос от клиента, запрос содержит внутри себя URL и параметры. Сервер имеет специальный конфигурационный файл, который ему сообщает о том, какой сервлет надо выполнить в случае прихода определенного URL. Сервлет выполняется (там вы можете использовать параметры) и создает HTML-страницу, которая отсылается клиенту.

Сервер по сути является контейнером (теперь уже не визуальных компонентов), который загружает сервлеты, выполняет их, вызывая определенные методы и получив от них результат, отправляет его клиенту. Таким образом сервлет — это Java-класс, который наследуется обычно от класса HttpServlet и переопределяет часть методов:

doGet - если мы хотим, чтобы сервлет реагировал на GET запрос;

doPost - если мы хотим, чтобы сервлет реагировал на POST запрос;

doPut, doDelete - если мы хотим, чтобы сервлет реагировал на PUT и DELETE запрос (есть и такие в HTTP). Эти методы реализуются крайне редко, т.к. сами команды тоже очень редко встречаются;

init, destroy - для управления ресурсами в момент создания сервлета и в момент его уничтожения.

Servlet как прокси-сервер.

Для поддержки апплетов сервлеты могут выполнять функции их прокси- серверов. Это может быть важно, поскольку система безопасности Java позволяет апплетам соединяться только с сервером, с которого они были загружены. Если апплет нуждается в соединении с сервером баз данных, расположенном на другой машине, *servlet* может создать это соединение для апплета.

Временные и постоянные сервлеты.

Сервлеты могут запускаться и останавливаться для каждого клиентского запроса. Также они могут запускаться при старте Web-сервера и существовать до его остановки. Временные *сервлеты* загружаются по требованию и предлагают хороший способ сохранения ресурсов сервера для редко используемых функций. Постоянные сервлеты загружаются при старте Web- сервера и существуют до его остановки. Сервлеты устанавливаются как постоянные расширения для сервера в том случае, если затраты по их запуску очень велики (например, установка соединения с базой данных), если они предлагают постоянную функциональность на стороне сервера (например, служба RMI), или в случаях, когда они должны отвечать на запросы клиента как можно быстрее. Не существует специального кода для назначения *сервлета* постоянным или временным; это функция настройки Web-сервера.

Жизненный цикл сервлета, javax.servlet.Servlet.

Сервлеты выполняются на платформе Web-сервера как часть того же процесса, что и сам Web-сервер. Web-сервер отвечает за инициализацию,

вызов и уничтожение каждого экземпляра сервлета. Web-сервер взаимодействует с сервлетом через простой интерфейс: javax.servlet.Servlet.

Интерфейс javax.servlet.Servlet включает три главных методов:

- service();
- destroy();
- int().

Два вспомогательных метода:

- getServletConfig();
- getServletInfo().

Сходство между интерфейсами *сервлета* и апплета Java очевидны. Именно так и было спроектировано! Java сервлеты являются для Web-серверов тем же самым, чем являются апплеты для Web-браузеров. Апплет выполняется в Web- браузере, выполняя действия по его запросу через специальный интерфейс. Сервлет делает то же самое, работая на Web-сервере.

Интерфейс ServletRequest.

ServletRequest предоставляет клиентскую информацию о параметрах HTTP запроса сервлету, т.е. обеспечивает данные включая название параметра и значения, атрибуты, и входной поток. Эта информация передается в метод *service()*.

Следующий *servlet пример* показывает, как получить информацию из параметра *request* метода *service()*:

```
BufferedReader reader; String param1;
String param2;
public void service(ServletRequest request, ServletResponse response)
{
reader = request.getReader();
param1 = request.getParameter("First"); param2 =
request.getParameter("Second");
}
```

Дополнительная информация о запросе доступна сервлету через методы, основные из которых приведены в следующей таблице:

Таблица 14.1

Методы сервлета

getAttribute()	Возвращает значение указанного атрибута этого запроса.
getContentLength()	Размер запроса, если известен.
getContentType()	Возвращает тип MIME тела запроса.
getInputStream()	Возвращает InputStream для чтения двоичных данныхиз тела запроса.
GetParameterNames()	Возвращает массив строк с именами всех параметров.
getParameterValues()	Возвращает массив значений для указанного параметра.
getProtocol()	Возвращает протокол и версию для запроса как строку вида <protocol>/<major version>.<minor version>.
getReader()	Возвращает BufferedReader для получения текста из тела запроса.
getRealPath()	Возвращает реальный путь для Указанного виртуального пути.
getRemoteAddr()	IP-адрес клиента, пославшего данный запрос.
getRemoteHost()	Имя хоста клиентской машины, пославшего Данный запрос.
getScheme()	Возвращает схему, используемую в URL этого запроса (например, https, http, ftp, и т.д.).
getServerName()	Имя хоста сервера, принявшего данный запрос.
getServerPort()	Возвращает номер порта, используемого для приема этого запроса.

Задание:

Студенты создают сервлет используя варианты перечисленные в таблице 14.2.

Таблица 14.2

Варианты заданий

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

Порядок выполнения работы:

Шаг 1. Откройте NetBeans IDE.

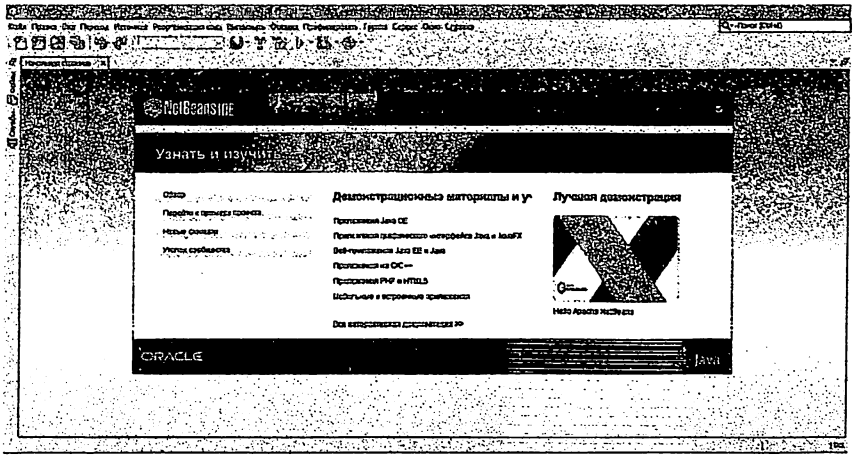


Рисунок 14.1. Среда NetBeans IDE

Шаг 2. В меню «Файл» выбираем «Создать проект». В появившемся окне «Создать проект» в разделе «Категории» выбираем «Java Web», из раздела «Проект» выбираем «Веб-Приложение» и нажимаем «Далее».

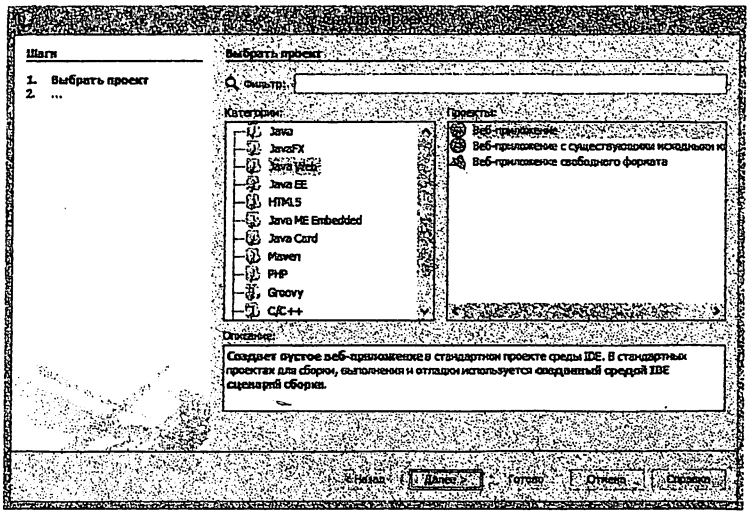


Рисунок 14.2. Окно «Создать проект» в среде NetBeans IDE

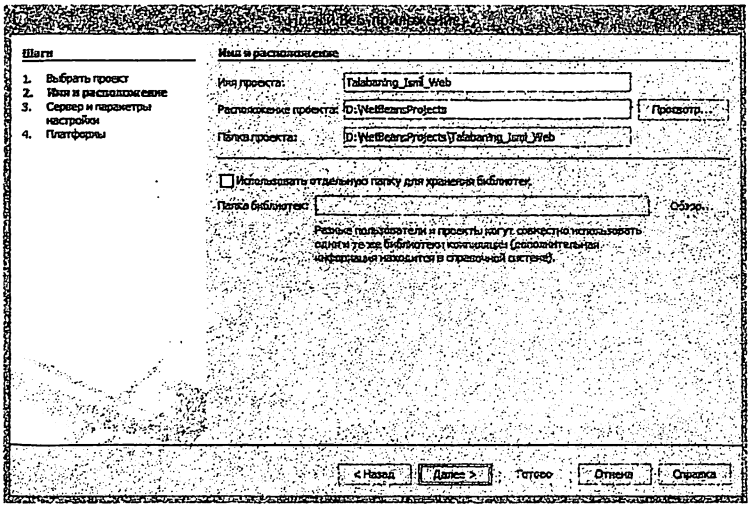


Рисунок 14.3. Окно «Новое веб-приложение» в среде NetBeans IDE

Шаг 3. В окне «Новое веб-приложение» введите «Talabaning_Ismi_Web» в поле «Имя проекта» и нажмите кнопку «Далее».

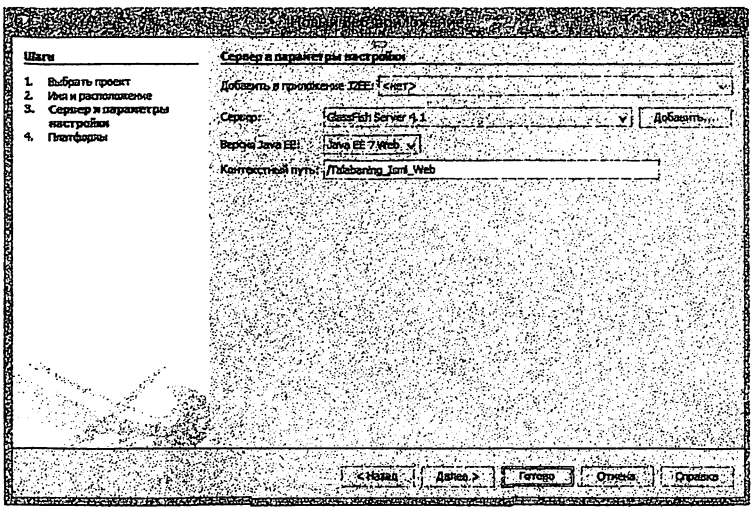


Рисунок 14.4. Выбор сервера

Шаг 4. Выбрать сервер и указать параметры:

- Выбор сервера: GlassFish 4.1
- Версия Java EE 7 Web.
- Нажать «Готово»

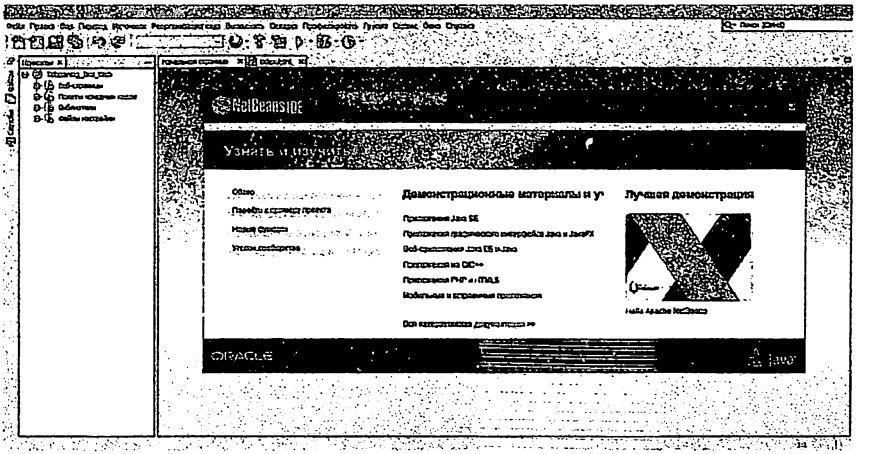


Рисунок 14.5. Среда NetBeans IDE

Шаг 5. Созданный проект и его компоненты появляется в окне слева. На панели проекта по нажатию правой кнопки мыши на компоненте проекта «Веб- страницы» в сплывающем меню выбрать «Новый» → «HTML...».

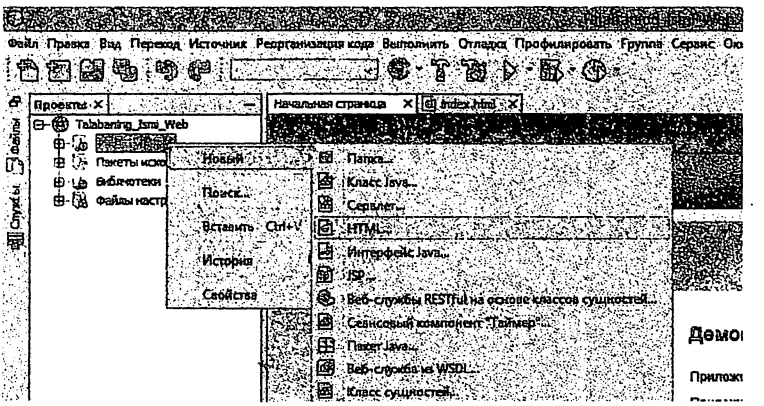


Рисунок 14.6. Среда NetBeans IDE

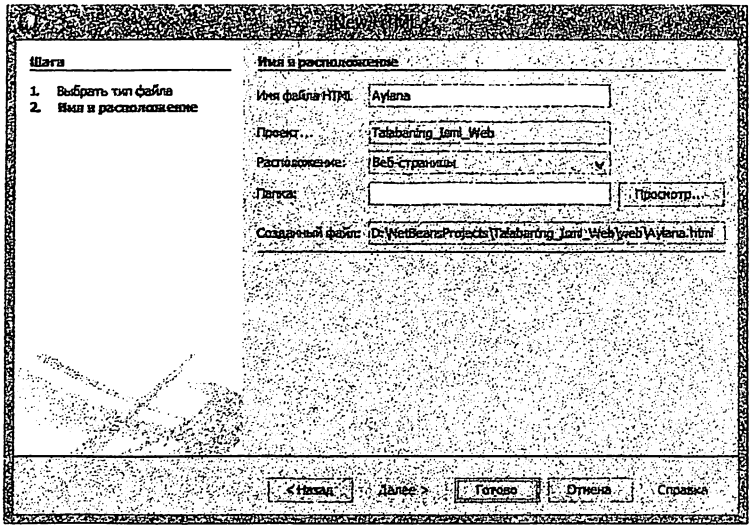


Рисунок 14.7. Окна New HTML

Шаг 6. Для имени файла HTML введите Krug(Aylana). Нажмите кнопку Готово. Теперь вводим HTML код.

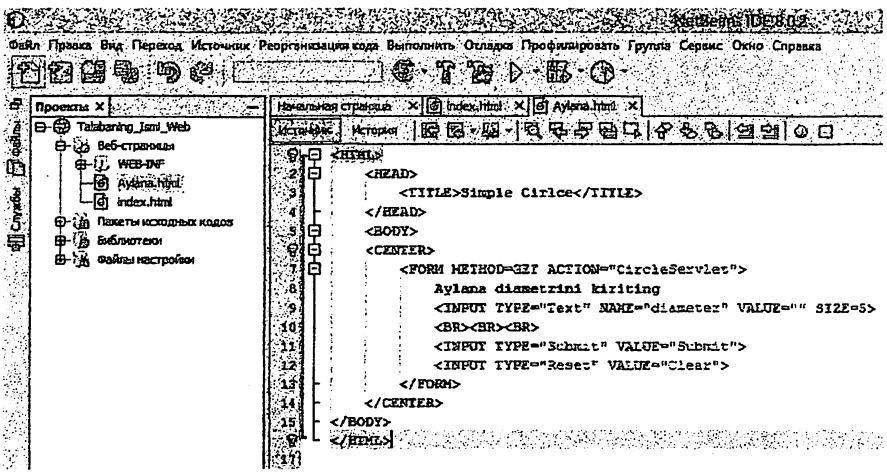


Рисунок 14.8. Файл Aylana.html

Шаг 7. Теперь создадим сервлет. На панели проекта по нажатию правой кнопки мыши на компоненте проекта «Пакеты исходных кодов» в сплывающем меню выбрать «Новый» → «Сервлет».

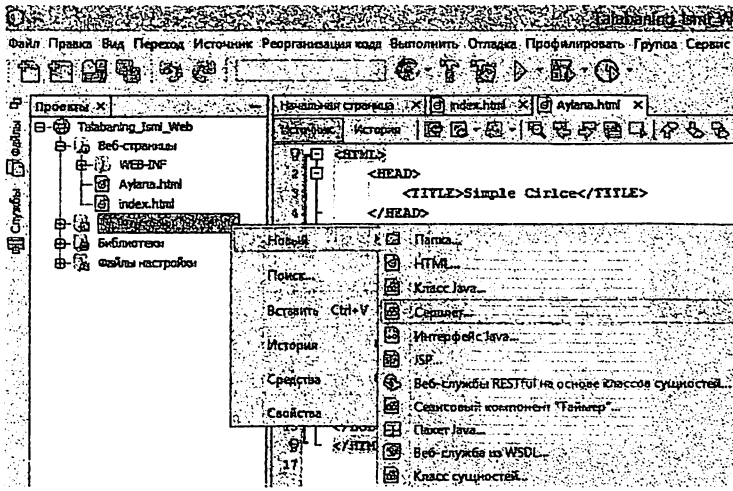


Рисунок 14.9. Создание сервлета

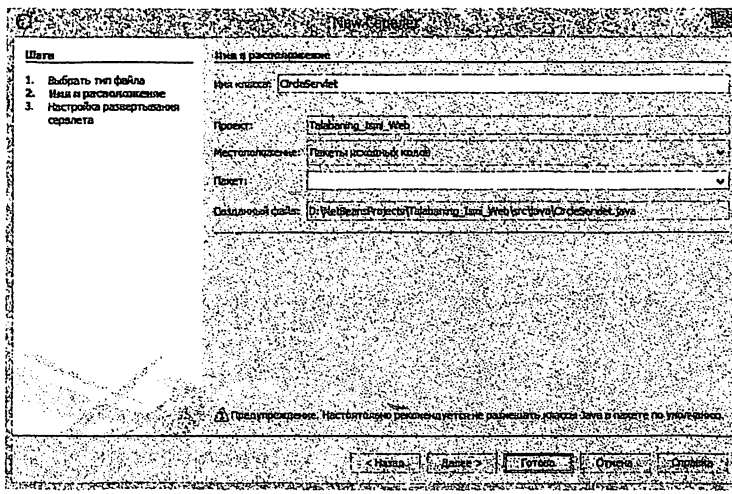
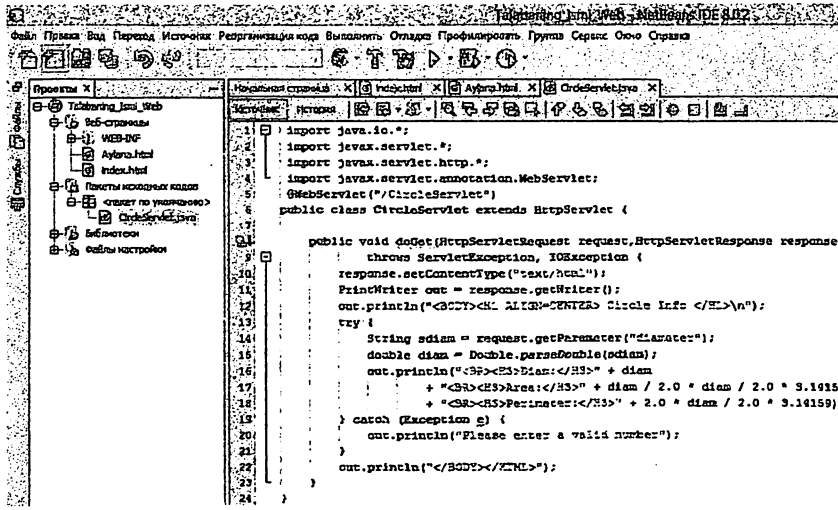


Рисунок 14.10. Создание сервлета

Шаг 8. Для имени класса введите «CircleServlet». Нажмите кнопку «Готово».



```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/CircleServlet")
public class CircleServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<BODY><H1 ALIGN=CENTER> Circle Info </H1>\n");
        try {
            String sdiam = request.getParameter("diameter");
            double diam = Double.parseDouble(sdiam);
            out.println("<B>Diam:</B>" + diam
                    + "<BR><B>Area:</B>" + diam / 2.0 * diam / 2.0 * 3.14159
                    + "<BR><B>Perimeter:</B>" + 2.0 * diam / 2.0 * 3.14159);
        } catch (Exception e) {
            out.println("Please enter a valid number");
        }
        out.println("</BODY></HTML>");
    }
}
```

Рис.14.11. Файл CircleServlet.java

Контрольные вопросы

1. Что такое сервлет?
2. Какие есть виды сервлета?
3. Как загрузить сервлет?
4. Какие библиотеки используется в сервлете?

ЛАБОРАТОРНАЯ РАБОТА № 15

Тема: Создание динамических веб-страниц

Цель работы:

Формирование практических навыков у студентов по созданию динамических веб-страниц с помощью JSP.

Теоретическая часть:

Java Server Pages (JSP) - это одна из технологий J2EE, которая представляет собой расширение технологии сервлетов для упрощения работы с Web-содержимым. Страницы JSP позволяет легко разделить Web-содержимое на статическую и динамическую часть, допускающую многократное использование ранее определенных компонентов.

Разработчики Java Server Pages могут использовать компоненты JavaBeans и создавать собственные библиотеки нестандартных тегов, которые инкапсулируют сложные динамические функциональные средства.

Спецификация Java Server Pages наследует и расширяет спецификацию сервлетов. Как и сервлеты, компоненты JSP относятся к компонентам Web и располагаются в Web-контейнере. Страницы JSP не зависят от конкретной реализации Web-контейнера, что обеспечивает возможность их повторного использования.

Технология Java Server Pages содержит четыре ключевых компонента:

1. *Директивы (directive)* представляют собой сообщения для контейнера JSP, дающим возможность определить параметры страницы, подключения других ресурсов, использовать собственные нестандартные библиотеки тегов.
2. *Действия actions* инкапсулируют функциональные возможности в предопределенных тегах, которые можно встраивать в JSP-страницу. JSP actions часто выполняются на основе информации, посылаемой на сервер в составе запроса от определенного клиента. Действия также могут создавать объекты Java для использования их в скриптлетах JSP.
3. *Скриптлеты scriptlets* позволяют вставлять код Java в страницы JSP, который взаимодействует с объектами страницы при обработке запроса.
4. *Библиотеки тегов (tag library)* являются составной частью механизма расширения Наличие данных с неизменяемой структурой определяют выбор программиста в принятии решения, какую технологию следует использовать: сервлеты или страницы JSP. Программисты

предпочитают использовать страницы JSP, если основная часть посылаемого клиенту содержимого представляет собой данные с неизменяемой структурой, и лишь небольшая часть содержимого генерируется динамически с помощью кода Java.

Сервлеты предпочтительнее использовать, если только небольшая часть содержимого, посылаемого клиенту, представляет собой данные с неизменяемой структурой. На самом деле отдельные сервлеты могут вообще не генерировать содержимого для клиента, выполняя определенную задачу в интересах клиента, а затем вызывают другие сервлеты или JSP-страницы, чтобы отправить ответ.

Необходимо заметить, что во многих случаях сервлеты и JSP-страницы являются взаимозамеяемыми. Подобно сервлетам, JSP-страницы обычно выполняются на стороне Web-сервера, который называют контейнером JSP.

Когда Web-сервер, поддерживающий технологию JSP, принимает первый запрос на JSP-страницу, контейнер JSP транслирует эту JSP-страницу в сервлет Java, который обслуживает текущий запрос и все последующие запросы к этой странице. Если при компиляции нового сервлета возникают ошибки, эти ошибки приводят к ошибкам на этапе компиляции. Контейнер JSP на этапе трансляции помещает операторы Java, которые реализует ответ JSP- страницы, в метод _jspService. Если сервлет компилируется без ошибок, контейнер JSP вызывает метод _jspService для обработки запроса.

JSP-страница может обработать запрос непосредственно, либо вызвать другие компоненты Web-приложения, чтобы содействовать обработке запроса.

Любые ошибки, которые возникают в процессе обработки, вызывают исключительную ситуацию в Web-сервере на этапе запроса.

Весь статический текст HTML, называемый в документации JSP шаблоном HTML (template HTML), сразу направляется в выходной поток.

Выходной поток страницы буферизуется. Буферизацию обеспечивает класс *JspWriter*, расширяющий класс Writer. Размер буфера по умолчанию ограничен до 8 Кбайт, но его можно изменить атрибутом buffer тега <%@ *page*>. Наличие буфера позволяет заносить заголовки ответа в выходной поток совместно с выводимым текстом. В буфере заголовки будут размещены перед текстом.

Таким образом, достаточно написать страницу JSP, сохранить ее в файле с расширением *jsp* и установить файл в контейнер, так же, как и страницу HTML, не заботясь о компиляции. При установке можно задать начальные параметры страницы JSP также, как и начальные параметры сервлета.

Жизненный цикл JSP.

Страница JSP обслуживает запросы, как сервлет. Следовательно, жизненный цикл и многие возможности страниц JSP (в частности, динамические аспекты) определяются технологией *Servlet* и многие обсуждения в этой главе ссылаются на функции, описанные на странице сервлетов.

Когда запрос отображается на страницу JSP, он обрабатывается специальным сервлетом, который сначала проверяет, не старше ли сервлет страницы JSP, чем сама страница JSP. Если это так, он переводит страницу JSP в класс сервлета и компилирует класс. При разработке Web-приложения одним из преимуществ страниц JSP перед сервлетами является то, что процесс построения (*компиляции страницы JSP в сервлет*) выполняется автоматически.

Трансляция и компиляция страницы JSP.

На фазе трансляции каждый тип данных в странице JSP интерпретируется отдельно. Шаблонные данные трансформируются в код, который будет помещать данные в поток, возвращающий данные клиенту. Элементы JSP трактуются следующим образом:

- директивы, используемые для управления тем, как Web-

контейнер переводит и выполняет страницу JSP;

- скриптовые элементы вставляются в класс сервлета страницы JSP;
- элементы в форме <jsp:XXX ... /> конвертируются в вызов метода для компонентов *JavaBeans* или вызовы API Java Servlet.

И фаза трансляции, и фаза компиляции могут порождать ошибки, которые будут выведены только, когда страница будет в первый раз запрошена. Если ошибка возникает при трансляции страницы (например, транслятор находит элемент JSP с неправильным форматом), сервер возвращает *ParseException*, и исходный файл класса сервлета будет пустым или незаконченным. Последняя незаконченная строка дает указатель на неправильный элемент JSP. Если ошибка возникает при компиляции страницы (например, синтаксическая ошибка в скриптлете), сервер возвращает *JasperException* и сообщение, которое включает в себя имя сервлета страницы JSP и строку, в которой произошла ошибка.

Таблица 5.1

Варианты заданий

№	Задания
1	Найти решение квадратного уравнения с произвольными коэффициентами
2	Сортировать n чисел в порядке возрастания
3	Сортировать n чисел в порядке убывания
4	Найти n степень числа m
5	Вычислить квадратный корень из произвольного числа n
6	Найти наибольшее из n чисел
7	Найти наименьшее из n чисел
8	Найти объем произвольного куба
9	Найти длину биссектрисы треугольника
10	Вычислить сумму любых n чисел
11	Вычислите сумму положительных чисел из любых n чисел
12	Выведите текст в обратном порядке
13	Найти объем произвольного куба
14	Выделяйте нечетные буквы в тексте
15	Вычислить объем шара R радиуса
16	Вычислить объем произвольного цилиндра

17	Вычислить диагональ прямоугольника
18	Найти поверхность шара радиуса R
19	Найти объем произвольного куба
20	Найти четные числа из n чисел
21	Найти среднее арифметическое n чисел
22	Найти среднее геометрическое n чисел
23	Найти n факториал
24	Найти площадь ромба
25	Рассчитать среднее арифметическое n чисел
26	Найти площадь прямоугольника
27	Рассчитать среднее геометрическое n чисел
28	Найдите площадь треугольника
29	Найти длину окружности произвольного радиуса R
30	Вычислить площадь поверхности шара произвольного радиуса R

Задание:

Студент получает индивидуальное задание в лабораторных работах. На этом задании студент создает динамические веб-страницы.

Порядок выполнения работы:

Шаг 1. Откройте NetBeans IDE. Созданный проект и его компоненты появляется в окне слева.

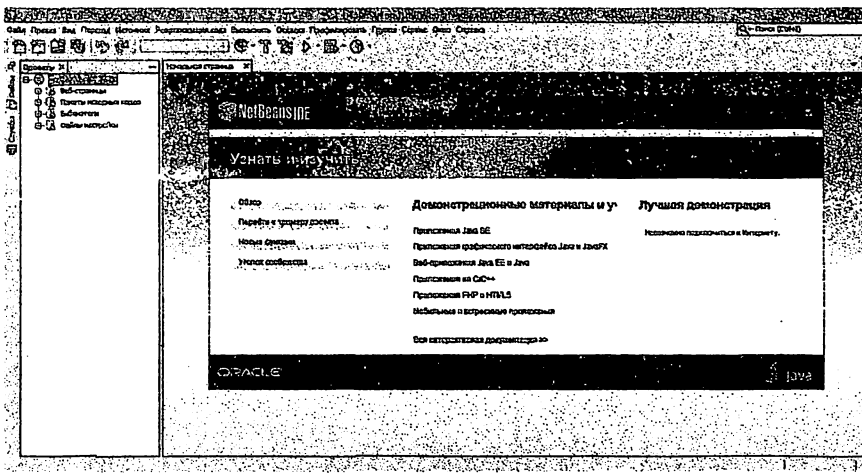


Рисунок 15.1. Среда NetBeans IDE

Шаг 2. На панели проекта по нажатию правой кнопки мыши на компоненте проекта «Веб-страницы» в сплывающем меню выбрать «Новый» → «HTML...».

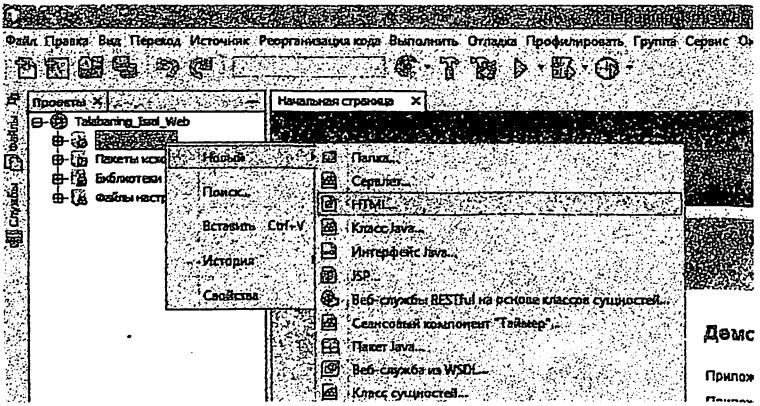


Рисунок 15.2. Среда NetBeans IDE

Шаг 3. Для имени файла «New HTML» введите Krug1.html (Aylana1.html). Нажмите кнопку «Готово». Теперь вводим HTML код.

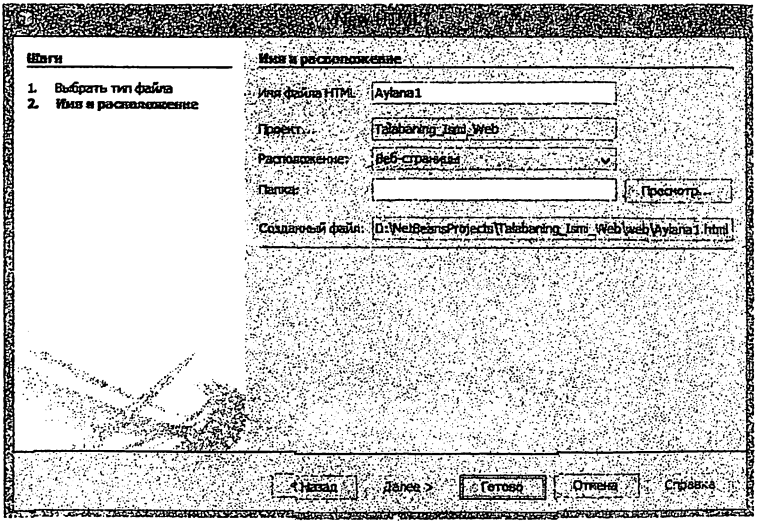


Рисунок 15.3. Окно New HTML

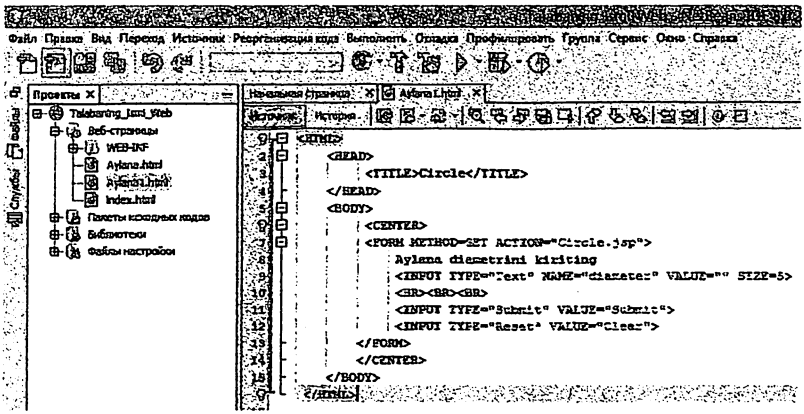


Рисунок 15.4. Файл Aylana1.html

Шаг 4. Теперь создадим JSP страницу. Для этого щелкните правой кнопкой мыши над «Веб-страницы», выберите «Новый» → «JSP ...» из контекстного меню.

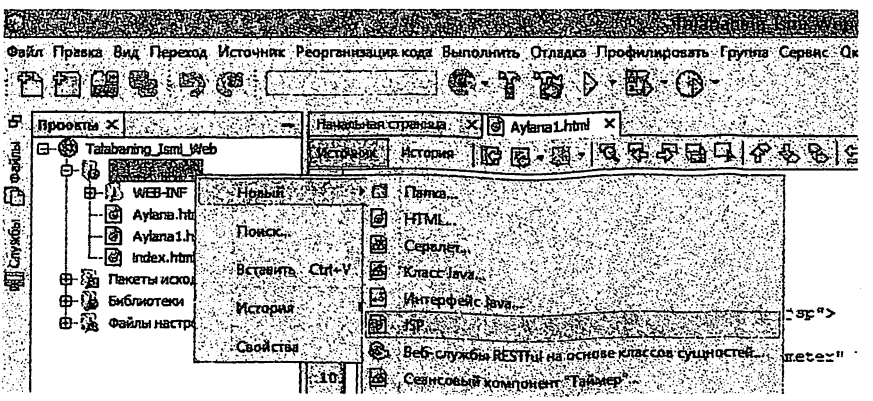


Рисунок 15.5. Процесс создания страницы JSP

Шаг 5. В окне «New JSP» введите в поле «Circle» и нажмите «Готово». Код JSP добавляется в созданный файл «Circle.jsp».

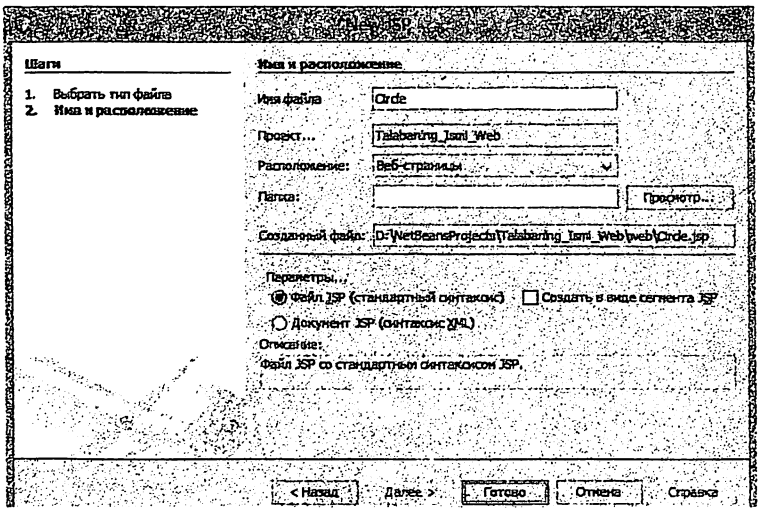


Рисунок 15.6. Процесс создания страницы JSP

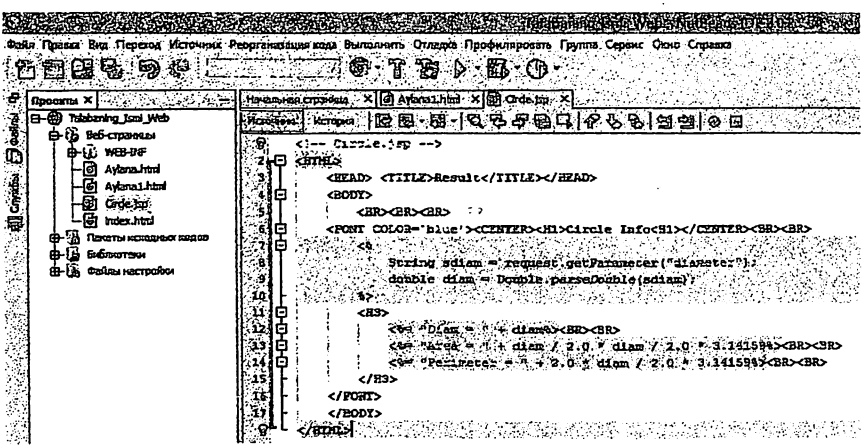


Рисунок 15.7. Файл Circle.jsp

Контрольные вопросы

1. Что такое JSP.
2. Каковы модели архитектуры JSP?
3. Объясните компиляцию и трансляцию страниц JSP.
4. Архитектура JSP.
5. JSP теги.

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. Alla Redko, Irina Fedortsov. JavaFX Working with JavaFX UI Components Release 8. 2014
10. David Reilly and Michael Reilly, Java Network Programming and Distributed Computing, Addison-Wesley (ISBN: 0-201-71037-4).
11. Java Servlet Programming 2nd Edition. Jason Hunter, William Crawford. 2001
12. 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.
13. https://www.javatpoint.com
14. https://www.tutorialspoint.com
15. https://www.w3schools.com

Методическое пособие разработанное для лабораторных работ по предмету “Основы сетевого программирования” часть 2 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

Ф.К.Тожиева

Х.Х.Ахмедова

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 5.
Заказ № 74. Тираж 10.
Отпечатано в «Редакционно издательском»
отделе при ТУИТ.
Ташкент ул. Амир Темур, 108.