**Лабораторная работа № 9.**

**Многопоточное программирование в языке Java.**

Язык Java делает сетевое программирование простым благодаря наличию специальных средств и классов. Рассмотрим некоторые виды сетевых приложений. Internet-приложения включают Web-браузер, e-mail, сетевые новости, передачу файлов и telnet. Основной используемый протокол – TCP/IP.

Приложения клиент/сервер используют компьютер, выполняющий специальную программу - сервер, которая предоставляет услуги другим программам - клиентам. Клиент – это программа, получающая услуги от сервера. Клиент-серверные приложения основаны на использовании верхнего уровня протоколов. На TCP/IP основаны следующие протоколы:

* HTTP – Hypertext Transfer Protocol (WWW);
* NNTP – Network News Transfer Protocol (группы новостей);
* SMTP – Simple Mail Transfer Protocol (посылка почты);
* POP3 – Post Office Protocol (чтение почты с сервера);
* FTP – File Transfer Protocol (протокол передачи файлов);
* TELNET – Удаленное управление компьютерами.

Каждый компьютер по протоколу TCP/IP имеет уникальный IP-адрес. Это 32-битовое число, обычно записываемое как четыре числа, разделенные точками, каждое из которых изменяется от 0 до 255. IP-адрес может быть временным и выделяться динамически для каждого подключения или быть постоянным, как для сервера. Обычно при подключении к компьютеру вместо числового IP адреса используются символьные имена (например - www.bsu.iba.by), называемые именами домена. Специальная программа DNS (Domain Name Sever) преобразует имя домена в числовой IP-адрес. Получить IP-адрес в программе можно с помощью объекта класса InetAddress из пакета java.net.

import java.net.\*;

public class MyLocal {

public static void main(String[] args) {

InetAddress myIP = null;

try {

myIP = InetAddress.getLocalHost();

} catch (UnknownHostException e) {

System.out.println("ошибка доступа ->" + e);

}

System.out.println("Мой IP ->" + myIP);

}

}

Следующая программа демонстрирует, как получить IP-адрес из имени домена с помощью сервера имен доменов (DNS), к которому обращается метод getByName().

import java.net.\*;

public class IPfromDNS {

public static void main(String[] args) {

InetAddress omgtu = null;

try {

omgtu = InetAddress.*getByName*("omgtu.ru");

} catch (UnknownHostException e) {

System.*out*.println("ошибка доступа ->" + e);

}

System.*out*.println("IP-адрес ->" + omgtu);

}

}

Будет выведено: IP-адрес –>omgtu.ru/195.69.204.35

Сокеты – это сетевые разъемы, через которые осуществляются двунаправленные поточные соединения между компьютерами. Сокет определяется номером порта и IP-адресом. При этом IP-адрес используется для идентификации компьютера, номер порта – для идентификации процесса, работающего на компьютере. Когда одно приложение знает сокет другого, создается сокетное соединение. Клиент пытается соединиться с сервером, инициализируя сокетное соединение. Сервер ждет, пока клиент не свяжется с ним. Первое сообщение, посылаемое клиентом на сервер, содержит сокет клиента. Сервер в свою очередь создает сокет, который будет использоваться для связи с клиентом, и посылает его клиенту с первым сообщением. После этого устанавливается коммуникационное соединение.

Сокетное соединение с сервером создается с помощью объекта класса Socket. При этом указывается IP-адрес сервера и номер порта (80 для HTTP). Если указано имя домена, то Java преобразует его с помощью DNS-сервера к IP-адресу:

try {

Socket socket = new Socket("localhost", 8030);

} catch (IOException e) {

System.out.println("ошибка: " + e);

}

Сервер ожидает сообщения клиента и должен быть запущен с указанием определенного порта. Объект класса ServerSocket создается с указанием кон-структору номера порта и ожидает сообщения клиента с помощью метода accept(), который возвращает сокет клиента:

Socket socket = null;

try {

ServerSocket server = new ServerSocket(8030);

socket = server.accept();

} catch (IOException e) {

System.out.println("ошибка: " + e);

}

Клиент и сервер после установления сокетного соединения могут получать данные из потока ввода и записывать данные в поток вывода с помощью методов getInputStrеam() и getOutputStrеam() или к PrintStream для того, чтобы про-грамма могла трактовать поток как выходные файлы.

В следующем примере для посылки клиенту строки "привет!" сервер вызывает метод getOutputStream() класса Socket. Клиент получает данные от сервера с помощью метода getInputStream(). Для разъединения клиента и сервера после завершения работы сокет закрывается с помощью метода close() класса Socket. В данном примере сервер посылает клиенту строку "привет!", после чего разрывает связь.

// передача клиенту строки : MyServerSocket.java import java.io.\*;

import java.net.\*;

public class MyServerSocket {

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

Socket s = null;

try {// посылка строки клиенту

ServerSocket server = new ServerSocket(8030);

s = server.accept();

PrintStream ps = new PrintStream(s.getOutputStream());

ps.println("привет!");

ps.flush();

s.close(); // разрыв соединения

} catch (IOException e) {

System.out.println("ошибка: " + e);

}

}

}

/\* получение клиентом строки : MyClientSocket.java \*/

import java.io.\*;

import java.net.\*;

public class MyClientSocket {

public static void main(String[] args) {

Socket socket = null;

try {// получение строки клиентом

socket = new Socket("имя\_компьютера", 8030);

BufferedReader dis = new BufferedReader(new InputStreamReader( socket.getInputStream()));

String msg = dis.readLine();

System.out.println(msg);

} catch (IOException e) {

System.out.println("ошибка: " + e);

}

}

}

Аналогично клиент может послать данные серверу через поток вывода с помощью метода getOutputStream(), а сервер может получать данные с помощью метода getInputStream().

Если необходимо протестировать подобный пример на одном компьютере, можно выступать одновременно в роли клиента и сервера, используя статические методы getLocalHost() класса InetAddress для получения динамического IP-адреса компьютера, который выделяется при входе в Internet.

**Варианты заданий**

Создать на основе сокетов клиент/серверное визуальное приложение

1. Клиент посылает через сервер сообщение другому клиенту.
2. Клиент посылает через сервер сообщение другому клиенту, выбранному из списка.
3. Чат. Клиент посылает через сервер сообщение, которое получают все клиенты. Список клиентов хранится на сервере в файле.
4. Клиент при обращении к серверу получает случайно выбранный сонет Шекспира из файла.
5. Сервер рассылает сообщения выбранным из списка клиентам. Список хранится в файле.
6. Сервер рассылает сообщения в определенное время определенным клиентам.
7. Сервер рассылает сообщения только тем клиентам, которые в настоящий момент находятся в on-line.
8. Чат. Сервер рассылает всем клиентам информацию о клиентах, вошедших в чат и покинувших его.
9. Клиент выбирает изображение из списка и пересылает его другому клиенту через сервер.
10. Игра по сети в «Морской бой».
11. Игра по сети в «21».
12. Игра по сети в классические «Крестики-нолики».
13. Игра по сети в «Го» («крестики-нолики» на безразмерном (большом) поле. Для победы необходимо выстроить пять в один ряд).