Работа в сети

Пакет java.net

Адреса в сети

- Классы
  - URI (uniform resource *identifier*)
  - URL (uniform resource *locator*)
  - InetAddress

Для того, чтобы извлечь реальную информацию, адресуемую данным URL, необходимо на основе URL создать объект URLConnection, воспользовавшись для этого методом openConnection()

Пример

```
URL hp = new
    URL("http://edu.mmcs.sfedu.ru/course/view.php?id=356");
URLConnection hpCon = hp.openConnection();
System.out.println("Date : " + hpCon.getDate ());
System.out.println("Type : " + hpCon.getContentType ());
System.out.println("Length: " + hpCon.getContentLength());
System.out.println( "HeaderField: " + hpCon.getHeaderField(0));
```

Сокеты

- Класс Socket

  Socket (String host, int port)

  InputStream getInputStream()

  OutputStream getOutputStream()

  Информация о текущем соединении

getInetAddress()

getPort()

getLocalPort()

Пример

```
try {
Socket s = new Socket("time-A.timefreq.bldrdoc.gov", 13);
  try {
        InputStream inStream = s.getInputStream();
        Scanner in = new Scanner(inStream);
        while (in.hasNextLine())  {
          String line = in.nextLine();
          System.out.println(line);
        }
      }  finally {  s.close(); }
} catch (IOException e) { //нет доступа к ресурсу}
```

Сокеты

- Класс ServerSocket

  ServerSocket(int port)

  ServerSocket(int port, int count)

Объект ServerSocket ожидает, пока к нему кто-нибудь не подключится.

При подключении он создает объект Socket

```
Socket socket = s.accept();
```

После подключения идет взаимодействие «сокет-сокет»

Клиент и сервер равноправны

Пример

```
try(ServerSocket s = new ServerSocket(8189);)
{
    System.out.println("Server start") ;
 // wait for client connection
    Socket incoming = s.accept();
    try {
      InputStream inStream = incoming.getInputStream();
      OutputStream outStream = incoming.getOutputStream();

      Scanner in = new Scanner(inStream);
      PrintWriter out = new PrintWriter(outStream, true );
```

Пример

```
 out.println("Hello! Enter BYE to exit.");
// echo client input
boolean done = false;
while (!done && in.hasNextLine()) {
   String line = in.nextLine();
   System.out.println("klient tell-"+line);
   out.println("Echo: " + line);
   if (line.trim().equals("BYE")) done = true;
 }
}finally  {  incoming.close();}
}   catch (IOException e)   {   e.printStackTrace();   }
```

Взаимодействие в отдельном потоке

```
class ServerToOne extends Thread {
//для одного клиента
 private Socket socket;
 private Scanner in;
 private PrintWriter out;
 public ServeToOne(Socket s) throws IOException {
  socket = s;
  in = . . .
  out = . . .
  start(); // Вызывает run()
 }
```

Что выполнить в потоке

```
public void run() {
   try {
     while (true) {
       String str = in.readLine();
       if (str.equals("END")) break;
       out.println("echo:“+str);
     }      System.out.println("closing...");
   } catch(IOException e) {  System.err.println("IO Exception");
   } finally {
     try {    socket.close();   } catch(IOException e) {
       System.err.println("Socket  not closed");}
}
}}
```

Многопоточный сервер

```
public class MultiServer {
  static final int PORT = 8189;
  public static void main(String[] args)  throws IOException {
    ServerSocket s = new ServerSocket(PORT);
    System.out.println("Server Started");
    try {
      while(true) {
        // Останавливает выполнение, до нового соединения:
        Socket socket = s.accept();
```

Многопоточный сервер

```
try {
        new ServerToOne (socket);
      } catch(IOException e) {
        // Если неудача - закрываем сокет,
        // в противном случае нить закроет его:
        socket.close();
      }
    }
  } finally {
    s.close();
  } }
}
```