Взаимодействие компонентов на основе сетевых протоколов

Основные термины и понятия

Определение протокола

- *Сетевой протокол* – **это набор правил и соглашений, регламентирующих процесс передачи данных по компьютерным сетям**

Иерархия протоколов

Стеки протоколов

Модель OSI

Прикладной уровень (Application layer)

Уровень представления (Presentation layer)

Сеансовый уровень (Session layer)

Транспортный уровень (Transport layer)

Сетевой уровень (Network layer)

Канальный уровень (Data Link layer)

Физический уровень (Physical layer)

Обмен данными согласно модели OSI

Наиболее распространенные протоколы в соответствии с уровнями модели OSI

Прикладной: HTTP, gopher, Telnet, DNS, SMTP, SNMP, CMIP, FTP…

Представления: HTTP, ASN.1, XML-RPC, TDI, XDR, SNMP, FTP, Telnet…

Сеансовый: ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink…

Транспортный: TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX…

Сетевой: IP, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP…

Канальный: ARCnet, ATM, DTM, SLIP, SMDS, Ethernet…

Физический: RS-232, RS-422, RS-423, RS-449, RS-485, модификации стандарта Ethernet: 10BASE-T, 10BASE2, 10BASE5, 100BASE-T (включает 100BASE-TX, 100BASE-T4, 100BASE-FX), 1000BASE-T, 1000BASE-TX, 1000BASE-SX

Обмен данными на основе протоколов UDP и TCP/IP

Основные термины

- *Сетевой адрес* – **уникальный в рамках данной сети идентификатор, присваиваемый компьютеру в сети**
- ***URL*** **(Uniform Resource Locator) – идентификатор установленной формы, определяющий положение какого-либо ресурса (файла, документа) в сети www (World Wide Web). Обобщенная спецификация URL имеет следующий вид:**

  протокол:субпротокол//доменное_имя : порт/локальный_путь/имя_ресурса#раздел_документа
- *Сокет* - **специальная структура данных, содержащая сетевой адрес компьютера и номер порта на этом компьютере**
- *Порт программы* **– это специальный номер, который может быть сопоставлен каждой программе на компьютере**

Обмен данными на основе протоколов UDP и TCP/IP

Основные классы для представления адресов компьютеров и ресурсов:

- `InetAddress`
- `Inet4Address`
- `Inet6Address`
- `SocketAddress`
- `InetSocketAddress`
- `URL`
- `URI`

Обмен данными на основе протоколов UDP и TCP/IP

Основные методы класса InetAddress

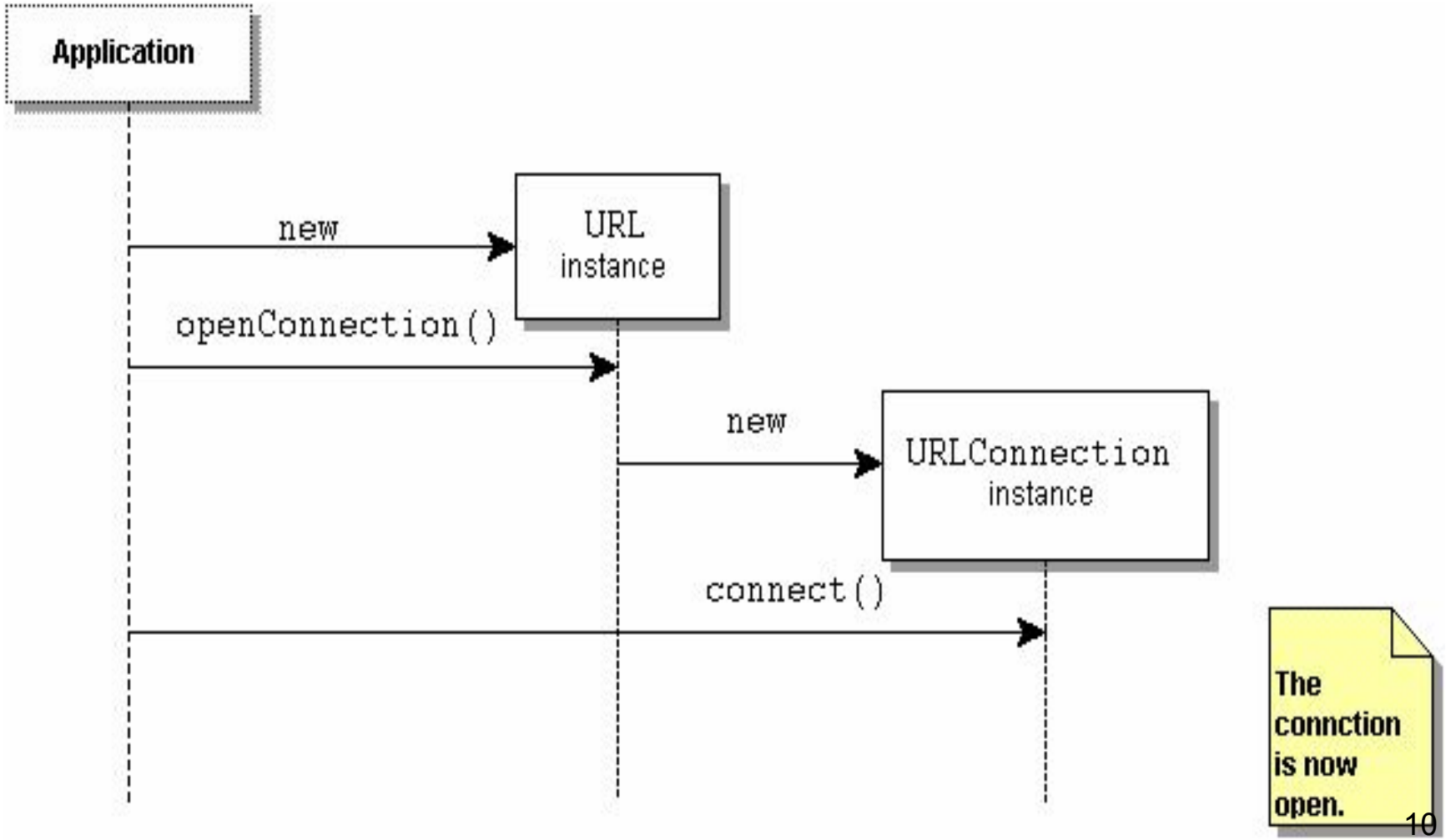
```
static InetAddress getByName (String host);
static InetAddress[] getAllByName (String host);
static InetAddress getByAddress (byte[] addr);
static InetAddress getLocalHost ();
String getCanonicalHostName ();
String getHostAddress();
boolean isLoopbackAddress ();
boolean isMulticastAddress ();
boolean isReachable (int timeout );
```

Обмен данными на основе протоколов UDP и TCP/IP

Основные методы класса URL:

```
URL (String spec);
URL (String protocol, String host, int port, String
   file);
String getProtocol ();
String getHost ();
String getPath ();
String getFile ();
int getPort ();
int getDefaultPort ();
InputStream openStream ();
Object getContent ();
```

Обмен данными на основе протоколов UDP и TCP/IP



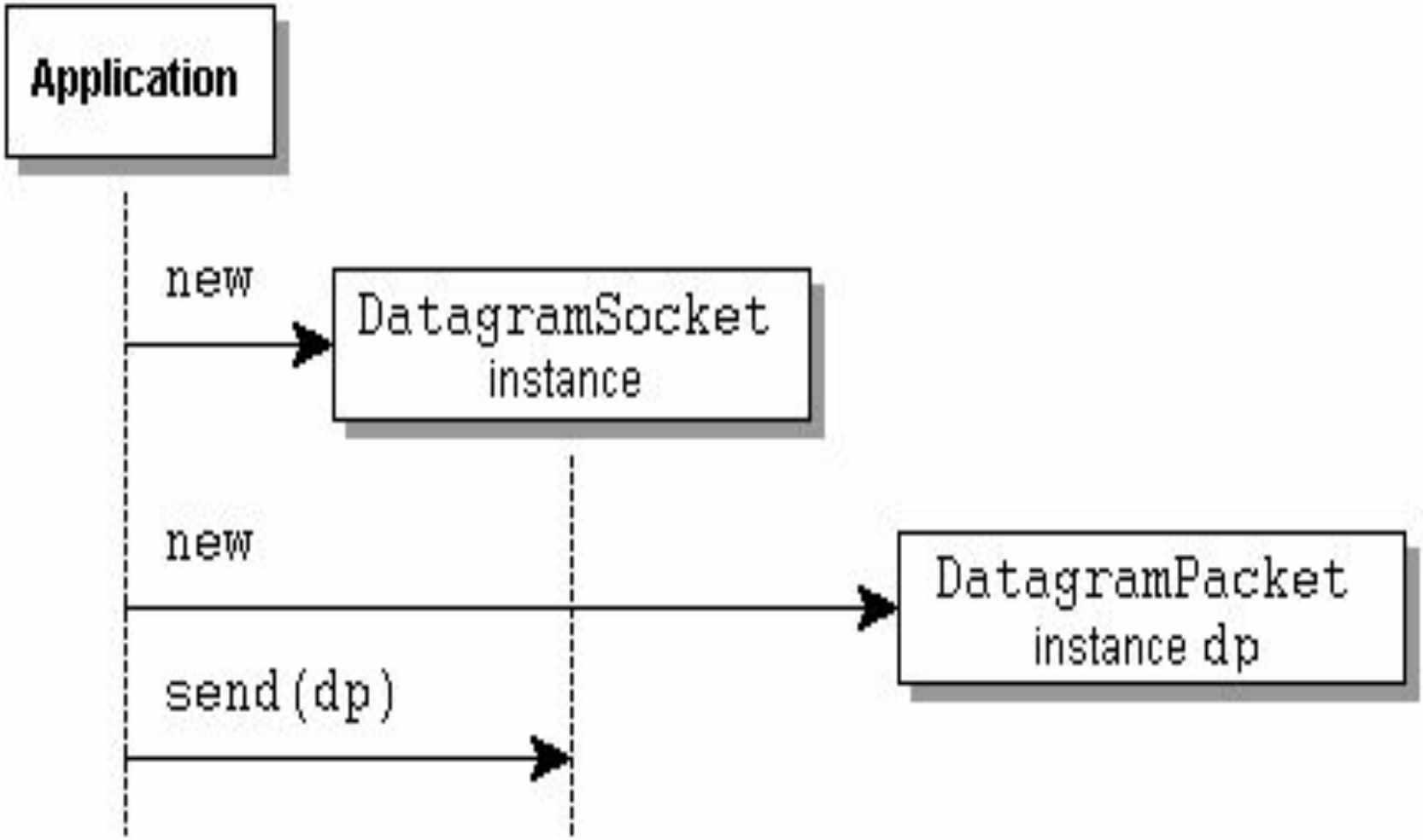
Обмен данными на основе протокола UDP

- **Обмен данными по протоколу UDP осуществляется с помощью специальных пакетов – *датаграмм* (datagram).**
- **В рамках протокола UDP постоянное соединение между компьютерами в сети не устанавливается.**
- **Протокол UDP не гарантирует доставки пакета адресату**
- **Протокол UDP не гарантирует сохранения порядка получения пакетов адресатом, поскольку каждый пакет может отправляться по индивидуальному маршруту**

Обмен данными на основе протокола UDP: Алгоритм обмена данными

- Создается экземпляр класса DatagramSocket с его привязкой к определенному локальному порту, а в случае такой необходимости и к локальному адресу.
- Для приема данных создается «пустой» экземпляра класса DatagramPacket с буфером заданного размера.
- Вызывается метод receive() класса DatagramSocket. После его завершения из пакета извлекаются данные, а также адрес и порт отправителя.
- Для отсылки данных создается другой экземпляр класса DatagramPacket и заполняется нужными данными. Кроме того, для данного пакета указывается адрес и порт назначения.
- Вызывается метод send() класса DatagramSocket.
- Сокет закрывается.

Обмен данными на основе протокола UDP: Диаграмма последовательности



Обмен данными на основе протокола UDP

Основные классы:

- `DatagramPacket`
- `DatagramSocket`

Обмен данными на основе протокола UDP

Основные методы класса `DatagramPacket`:

```
DatagramPacket(byte[] buf, int length)
DatagramPacket(byte[] buf, int length,
               InetAddress address, int port)
InetAddress getAddress()
int getPort()
SocketAddress getSocketAddress()
int getLength()
byte[] getData()
void setAddress(InetAddress iaddr)
void setData(byte[] buf)
void setPort(int port)
void setSocketAddress(SocketAddress address)
```

Обмен данными на основе протокола UDP

Основные методы класса DatagramSocket:

```
DatagramSocket()throws SocketException;
DatagramSocket (int port)
                throws SocketException;
void close();
boolean isClosed();
InetAddress getLocalAddress();
int getLocalPort();
void receive(DatagramPacket p)
                throws IOException;
void send(DatagramPacket p)
                throws IOException;
```

Обмен данными на основе протокола UDP: Пример кода отправки датаграммы

```
int port = 15679;
try {
  DatagramSocket udpSocket = new DatagramSocket (port);
  int bufferSize = 1024;
  String message = "UDP test message";
  byte[] buffer = message.getBytes ();

  DatagramPacket udpPacket = new DatagramPacket (buffer,
   buffer.length);
  udpPacket.setAddress (InetAddress.getByName
   ("localhost"));
  udpPacket.setPort (15678);
  udpSocket.send (udpPacket);
  udpSocket.close();
} catch (SocketException e) {
  System.out.println ("Socket exception occur");
} catch (IOException e) {
  System.out.println ("IOException occur");
}
```

Обмен данными на основе протокола UDP: Пример кода приема датаграммы

```
int Port = 15678;
try {
  DatagramSocket udpSocket = new DatagramSocket (Port);
  int bufferSize = 1024;
  byte[] buffer = new byte[BufferSize];

  DatagramPacket udpPacket = new DatagramPacket (buffer,
   buffer.length);
  UDPSocket.receive (UDPPacket);
  System.out.println ("Got packet from "
  + UDPPacket.getAddress () + ":" + UDPPacket.getPort ());
  System.out.println ("Packet contain: "
  + new String(UDPPacket.getData ()).trim ());
  udpSocket.close();
} catch (SocketException e) {
  System.out.println ("Socket exception occur");
} catch(IOException e) {
  System.out.println ("IOException occur");
}
```

Обмен данными на основе протокола TCP

- **Основные классы:**
  - Socket
  - ServerSocket

Обмен данными на основе протокола TCP

Алгоритм работы на стороне клиента:

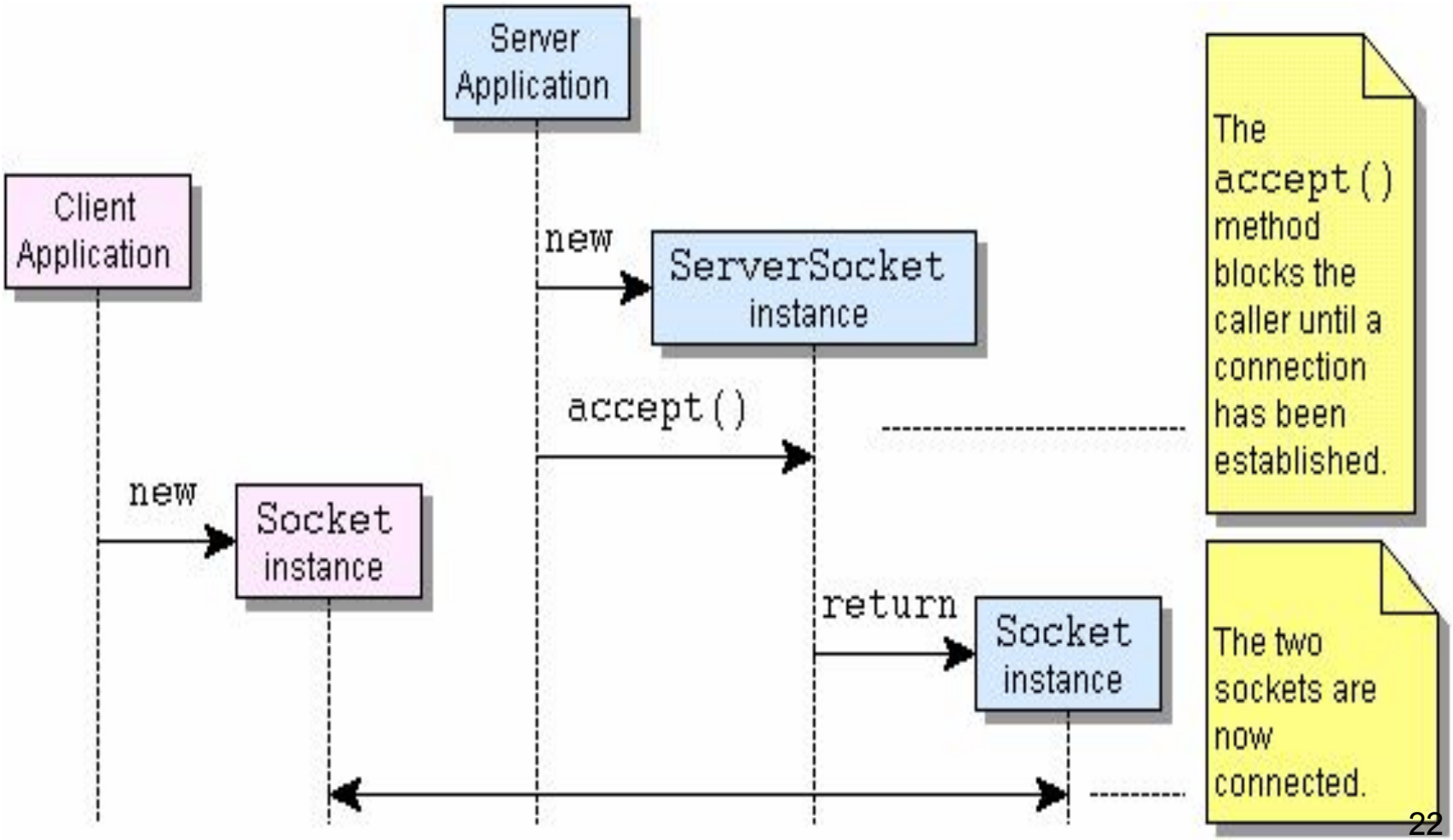
- Создать экземпляр класса Socket;
- Получить ссылки на входной и выходной потоки класса Socket;
- Произвести операции чтения из входного потока сокет;
- Произвести запись в выходной поток сокета;
- Закрыть входной и выходной потоки сокета;
- Закрыть сокет.

Обмен данными на основе протокола TCP

Алгоритм работы на стороне сервера:

- Создать экземпляр класса ServerSocket;
- Получить ссылку на экземпляр класса Socket с помощью метода accept();
- Получить ссылки на входной и выходной потоки класса Socket;
- Произвести операции чтения из входного потока сокета;
- Произвести запись в выходной поток сокета;
- Закрыть входные и выходные потоки сокета;
- Закрыть сокет, связанный с клиентом;
- Закрыть серверный сокет.

Обмен данными на основе протокола TCP



Обмен данными на основе протокола TCP: Класс Socket

Основные конструкторы класса:

```
Socket ();
Socket (InetAddress host, int port);
Socket (String host, int port);
Socket(InetAddress address, int port,
  InetAddress localAddr, int localPort);
Socket(String address, int port, InetAddress
  localAddr, int localPort);
protected Socket(SocketImpl impl)
```

Обмен данными на основе протокола TCP: Класс Socket

Методы для управления соединением:

```
void bind(SocketAddress bindpoint);
void connect(SocketAddress endpoint);
void connect(SocketAddress endpoint, int
  timeout);
void close() throws IOException
```

Обмен данными на основе протокола TCP: Класс Socket

Диагностические методы класса:

```
InetAddress getInetAddress();
int getPort();
InetAddress getLocalAddress();
int getLocalPort();
boolean isBound();
boolean isClosed();
boolean isConnected()
boolean isInputShutdown();
boolean isOutputShutdown();
```

Обмен данными на основе протокола TCP: Класс Socket

Методы для работы с входными и выходными потоками:

```
InputStream getInputStream() throws
  IOException;
OutputStream getOutputStream() throws
  IOException;
void shutdownInput()
                    throws IOException;
void shutdownOutput()
                     throws IOException;
```

Обмен данными на основе протокола TCP: класс ServerSocket

Основные конструкторы класса:

```
ServerSocket();
ServerSocket(int port);
ServerSocket(int port, int backlog,
  InetAddress bindAddr);
```

Обмен данными на основе протокола TCP: класс ServerSocket

Управление соединением:

```
Socket accept();
void bind(SocketAddress endpoint);
void close();
```

Обмен данными на основе протокола TCP: класс ServerSocket

Диагностические методы:

```
InetAddress getInetAddress();
int getLocalPort();
SocketAddress getLocalSocketAddress();
boolean isBound();
boolean isClosed();
```

Обмен данными на основе протокола TCP

Фрагмент кода реализации на стороне клиента:

```
final int DEFAULT_SERVER_PORT = 16789;
clsClientSocket = new Socket (InetAddress.getByName
   (strHost), DEFAULT_SERVER_PORT);
OutputStream out = clsClientSocket.getOutputStream ();
out.write ("Test message".getBytes());
InputStream in = clsClientSocket.getInputStream ();
byte[] message = new byte[1024];
int n = in.read ( message );
…
ClsClientSocket.shutdownInput ();
clsClientSocket.shutdouwOutput ();
clsClientSocket.close();
```

Обмен данными на основе протокола TCP

Фрагмент кода реализации на стороне сервера:

```
private final int DEFAULT_SERVER_PORT = 16789;
private final int DEFAULT_SERVER_CLIENTS_NUMBER = 124;

clsServerSocket = new ServerSocket (DEFAULT_SERVER_PORT,
   DEFAULT_SERVER_CLIENTS_NUMBER);
   Socket clsSocket = clsServerSocket.accept ();

   InputStream in =  clsSocket.getOutputStream ());

   OutputStrem out = clsSocket.getInputStream ());

    …
  clsSocket.shutdownInput();
  clsSocket.shutdownOutput();
  clsSocket.close();
  clsServerSocket.close();
```

Безопасность

Основные классы:

- для создания безопасного соединения:
  - SSLSocket
  - SSLServerSocket
- для работы с разрешениями:
  - SocketPermission
  - NetPermission
- для аутентификации:
  - Authenticator
  - PasswordAuthentication

Заключение

Заключительный обзор темы

Вопросы?