Lập trình Socket với Java

Nội dung bài học

- Giới thiệu lập trình Java
- Giới thiệu gói java.net
- Lớp InetAddress
- Truyền tin với giao thức TCP
  - TCP Sockets
  - Ví dụ về máy chủ/khách TCP
- Truyền tin với giao thức UDP
  - Datagram Sockets
  - Ví dụ về máy chủ/khách UDP

Ngôn ngữ lập trình Java

- Phát triển bởi James Gosling và cộng sự (Sun Microsystems) từ năm 1991
- Mục tiêu ban đầu là tạo ra ngôn ngữ lập trình có thể chạy trên nhiều platform khác nhau (platform-independent)
  - Các hệ thống nhúng, mobile agent
- Phổ biến nhờ sự phát triển của World Wide Web
  - Cũng yêu cầu một ngôn ngữ lập trình độc lập với platform
- Có liên hệ gần gũi với C, C++
  - Không phải phiên bản mở rộng của C++

Đặc trưng của Java: Bytecode

- Chương trình biên dịch không tạo ra mã chạy (executable code) mà tạo ra bytecode
  - Chỉ chạy trên Java Virtual Machine (JVM)
- Chỉ cần JVM là chạy được bytecode
  - Không phụ thuộc vào platform
  - Bảo mật
    - JVM sẽ giới hạn phạm vi hoạt động của chương trình
- JVM được tối ưu hóa để chạy bytecode
  - Tăng tốc độ chạy chương trình

Đặc trưng của Java: Hướng đối tượng

- Encapsulation (đóng gói)
  - Đóng gói dữ liệu và code vào một blackbox (class)
  - *Class*
    - Variables
    - Methods
  - Variables và methodscó thể là *private* hay *public*
- Polymorphism (tính đa hình)
  - Một interface cho nhiều actions
- Tính thừa kế
  - Một lớp A có thể kế thừa các thuộc tính của lớp B

A simple java program

```
/*
      This is a simple Java program.
      Call this file Example.java.
*/
class Example {
     // A Java program begins with a call to main().
     public static void main(String args[]) {
       System.out.println("Hello World!");
     }
}
```

More about Java ...

- Có thể tìm trên Web, sách lập trình Java, ...

Các classes trong gói java.net

- Gói java.net chứa các classes cho phép thực hiện lập trình mạng
  - InetAddress:
    - Ánh xạ, chuyển đổi và trình diễn địa chỉ IP
  - ServerSocket:
    - Socket chờ phía máy chủ
  - Socket
    - Socket kết nối
  - DatagramPacket:
    - biểu diễn gói tin UDP
  - DatagramSocket:
    - giao diện socket gửi và nhận gói tin UDP
  - MulticastSocket:
    - giao diện socket gửi và nhận các gói tin multicast

Các classes trong gói java.net (2)

- URL
  - Biểu diễn tài nguyên mô tả bởi URL(Uniform Resource Locator)
- URLConnection
  - Biểu diễn kết nối giữa với máy chủ biểu diễn bởi URL
- URLEncoder & URLDecoder
  - Chuyển đổi biểu diễn dữ liệu với các mã dữ liệu khác nhau
- ContentHandler:
  - tự động cập nhật phần mềm xử lý các kiểu dữ liệu mới
  - Ít dùng

Exceptions in Java

- BindException
- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

Trình diễn địa chỉ IP - Lớp InetAddress

- Biểu diễn địa chỉ IP và tên miền
- Khởi tạo đối tượng InetAddress
  - public static InetAddress getByName(String host) throws UnknownHostException

    public static InetAddress[] getAllByName(String host) throws UnknownHostException

    public static InetAddress   getLocalHost() throws UnknownHostException
  - Kết nối đến chương trình DNS cục bộ để lấy thông tin
    - Có thể gây ra ngoại lệ nếu kết nối không được phép
    - Có thể tự động kết nối với bên ngoài
      - Có thể bị lợi dụng để truyền tin trái phép ra ngoài qua DNS

Trình diễn địa chỉ IP
- Lớp InetAddress (2)

- Trả về tên miền trong đối tượng InetAddress
  - public String getHostName()
- Trả về địa chỉ IP dạng chuỗi ký tự/chuỗi byte
  - public String getHostAddress()
  - public byte[] getAddress()
    - Dùng để xác định kiểu địa chỉ IP (IPv4/IPv6)
    - Cần chuyển đổi biểu diễn byte sang int

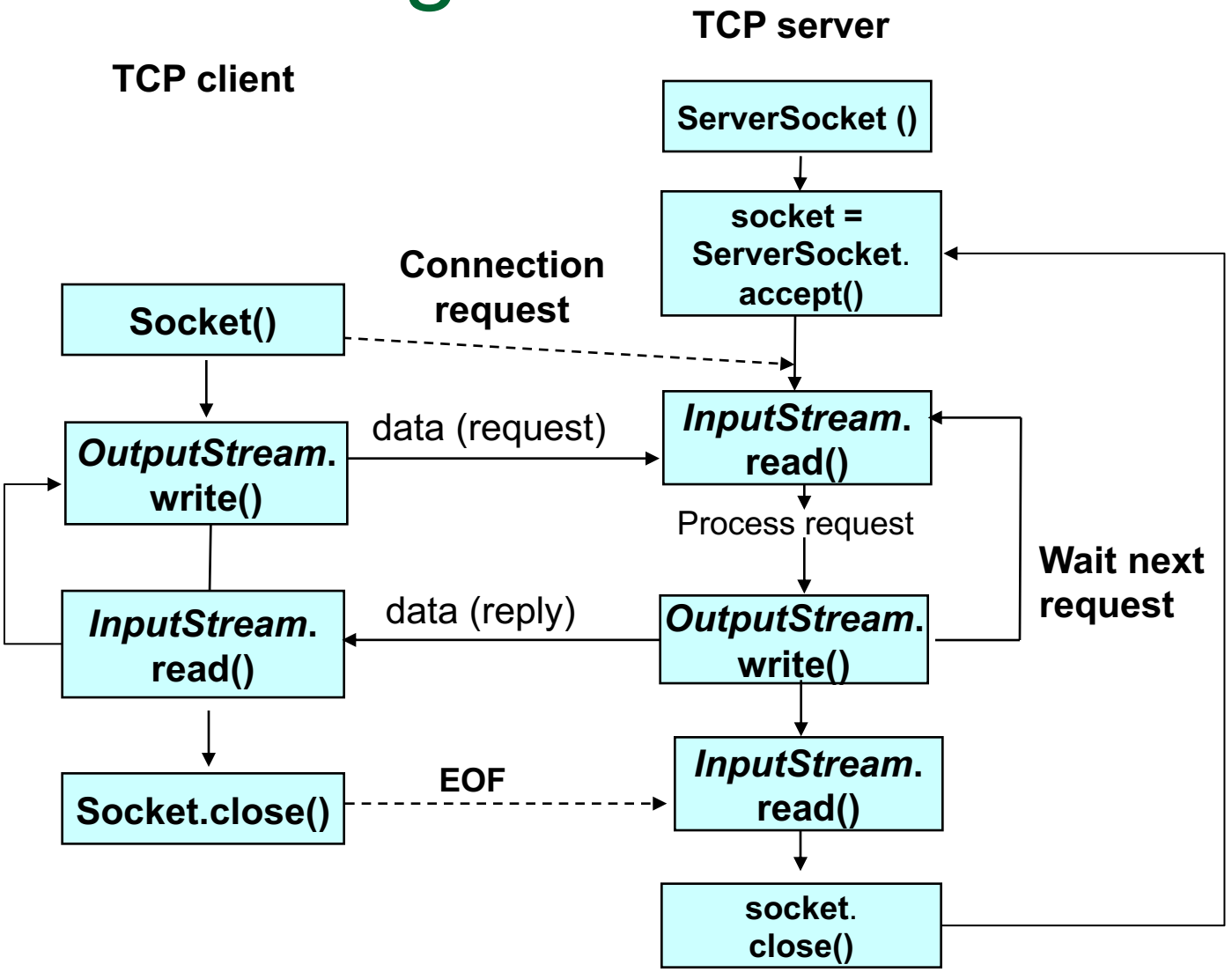
      int unsignedByte = signedByte < 0 ? signedByte + 256 : signedByte;
- Xác định kiểu địa chỉ
  - public boolean isMulticastAddress()
  - public boolean isLoopbackAddress( )
- Một số hàm khác
  - public int   hashCode()
  - public boolean equals(Object obj)
    - Đối tượng obj bằng một đối tượng InetAddress khi nó là một phiên bản (instance) InetAddress và có cùng địa chỉ IP
  - public String toString()

```
import java.net.*;
import java.io.*;

public class IPFinder
{
        public static void main(String[] args) throws IOException
        {
                String host;
                BufferedReader input =
                  new BufferedReader(
                        new InputStreamReader(System.in));

                System.out.print("\n\nEnter host name: ");
                host = input.readLine();  /*Đọc chuỗi ký tự nhập từ bàn phím*/
                try
                {
                        InetAddress address = InetAddress.getByName(host);
                        System.out.println("IP address: " + address.toString());
                }
        catch (UnknownHostException e)
        {
                        System.out.println("Could not find " + host);
                }
        }
}
```

Lấy địa chỉ của máy chủ

```
import java.net.*;

public class MyLocalIPAddress
{
    public static void main(String[] args)
     {
          try
          {
               InetAddress address = InetAddress.getLocalHost();
               System.out.println (address.toString());
          }
          catch (UnknownHostException e)
          {
              System.out.println("Could not find local address!");
          }
      }
}
```

Truyền tin với giao thức TCP



Lập trình máy khách TCP-
Lớp Java.net.Socket

Các bước thiết lập máy khách

- Tạo đối tượng Socket để thiết lập kết nối đến máy chủ sử dụng Socket()
- Thiết lập các dòng xuất/nhập dữ liệu
- Gửi và nhận dữ liệu
- Đóng kết nối close()

Lập trình máy khách TCP – Lớp Java.net.Socket

- Lớp cơ bản của Java để thực hiện truyền tin TCP giữa máy khách và máy chủ
  - Thiết lập hoặc ngắt kết nối và thiết lập các tùy chọn socket
- Kết nối được thiết lập khi khởi tạo đối tượng
  - Mỗi đối tượng Socket được gán với một máy chủ duy nhất

  public Socket(String host, int port) throws UnknownHostException, IOException

  public Socket(InetAddress address, int port) throws IOException

  public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException

  public Socket(InetAddress address, int port, InetAddress localAddress, int localPort) throws IOException

The Java.net.Socket Class –

Lấy thông tin về một Socket

- Lấy thông tin về địa chỉ máy tính kết nối đến

  public InetAddress getInetAddress( )

- Lấy thông tin cổng kết nối đến

  public int getPort( )

- Lấy thông tin về cổng kết nối cục bộ

  public int getLocalPort( )

- Lấy thông tin về địa chỉ kết nối cục bộ

  public InetAddress getLocalAddress( )

The Java.net.Socket Class – Xuất nhập dữ liệu socket

- Gửi và nhận dữ liệu socket được thực hiện thông qua dòng dữ liệu xuất/nhập
  - public InputStream getInputStream() throws IOException
    - trả về đối tượng InputStream
  - public OutputStream getOutputStream() throws IOException
    - trả về đối tượng OutputStream

Lớp InputStream

- Cung cấp các hàm đọc dữ liệu dạng byte

  public abstract int read( ) throws IOException

  public int read(byte[] input) throws IOException

  public int read(byte[] input, int offset, int length) throws IOException

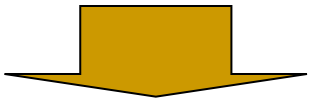
  public int available( ) throws IOException
- Giá trị trả về bằng -1 nếu đến cuối dòng dữ liệu
- Dữ liệu đọc dưới dạng signed byte
  - Cần chuyển đổi signed byte -> unsigned byte

    b = in.read( );

    int i = b >= 0 ? b : 256 + b;
- Thực hiện vòng lặp để đọc hết dữ liệu

Ví dụ về đọc dữ liệu

```
int bytesRead = 0;
int bytesToRead = 1024;
byte[] input = new byte[bytesToRead];
while (bytesRead < bytesToRead) {
    bytesRead += in.read(input, bytesRead, bytesToRead-bytesRead);
}
```



```
int bytesRead = 0;
int bytesToRead = 1024;
byte[] input = new byte[bytesToRead];
while (bytesRead < bytesToRead) {
   int result = in.read(input, bytesRead, bytesToRead - bytesRead);
   if (result == -1) break;
    bytesRead += result;
}
```

Lớp OutputStream

- Cung cấp các hàm ghi dữ liệu

  public abstract void write(int b) throws IOException

  public void write(byte[] data) throws IOException

  public void write(byte[] data, int offset, int length) throws IOException

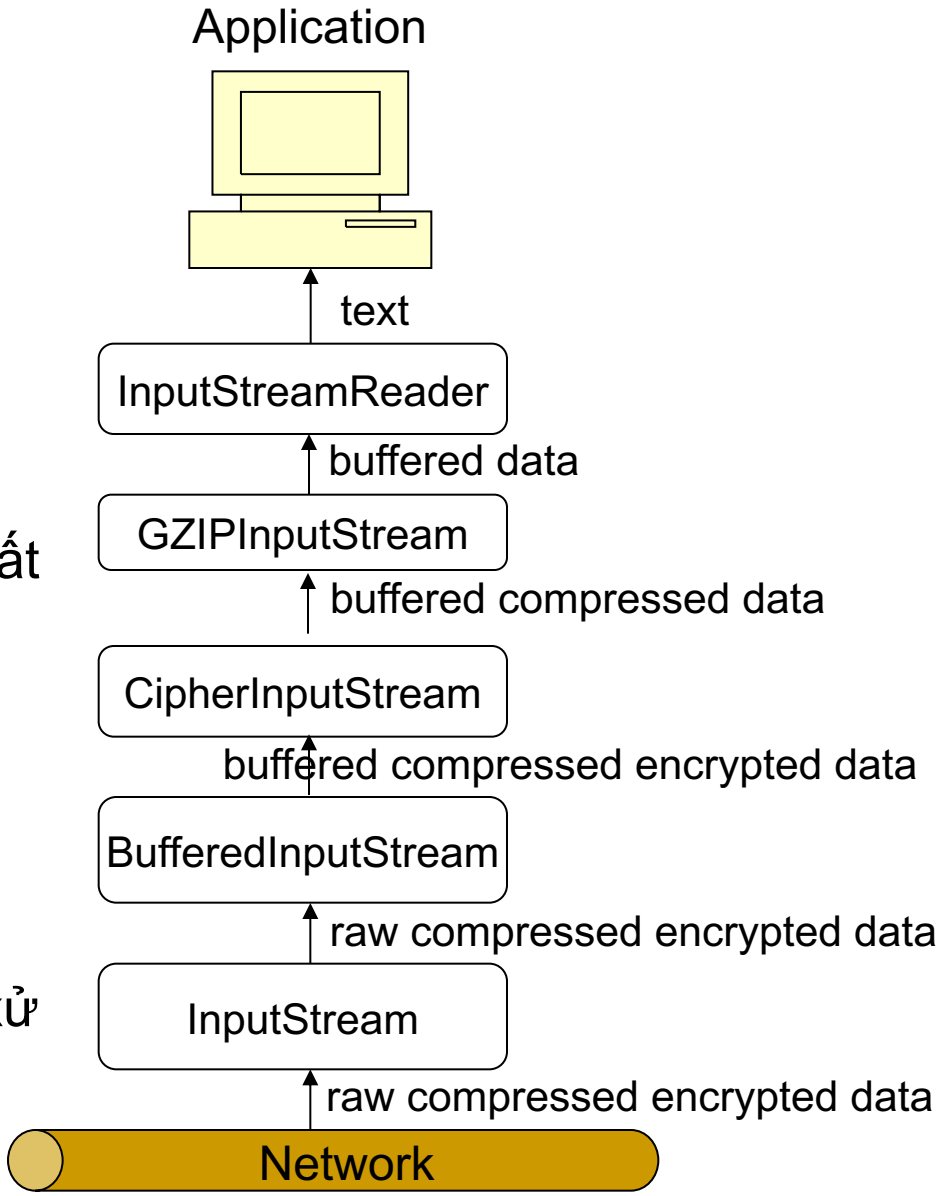
  public void flush( ) throws IOException

  public void close( ) throws IOException

- Ghi một dãy byte hiệu quả hơn ghi từng byte
- Nên sử dụng hàm flush() để lệnh ghi được thực hiện ngay

Filter Streams

- InputStream và OutputStream là các lớp xử lý dữ liệu thô (byte)
  - Cần chuyển đổi dữ liệu thô sang các định dạng dữ liệu nhất định và ngược lại
    - 7-bit ASCII, 8-bit Latin-1 hay UTF-8, định dạng zip
- Hai loại filter
  - filter stream: làm việc trên dữ liệu byte
  - readers and writers: làm việc xử lý dữ liệu text với các kiểu encoding khác nhau



The Java.net.Socket Class – Đóng socket

- public void close() throws IOException
- public void shutdownInput( ) throws IOException // Java 1.3
- public void shutdownOutput( ) throws IOException // Java 1.3
- public boolean isInputShutdown( ) // Java 1.4
- public boolean isOutputShutdown( ) // Java 1.4

Ví dụ: DaytimeClient.java

```
import java.net.*;
import java.io.*;

public class DaytimeClient {
      public static void main(String[] args) {
        String hostname;
        int    port;
        if (args.length > 0) {
              hostname = args[0];
              port = Integer.parseInt(args[1]);
        }
        else {
              hostname = "time.nist.gov";
              port     = 13;
        }
```

Example: DaytimeClient.java (2)

```
            try {
                        Socket theSocket = new Socket(hostname, port);
                InputStream timeStream = theSocket.getInputStream( );
                StringBuffer time = new StringBuffer( );
                int c;
                while ((c = timeStream.read( )) != -1)
                        time.append((char) c);
                String timeString = time.toString( ).trim( );
                System.out.println("It is " + timeString + " at " + hostname);
            }  // end try
            catch (UnknownHostException ex) {
                    System.err.println(ex);
            }
            catch (IOException ex) {
                    System.err.println(ex);
            }
        }  // end main
} // end DaytimeClient
```

Thiết lập tùy chọn socket

- TCP_NODELAY
  public void setTcpNoDelay(boolean on) throws SocketException
  public boolean getTcpNoDelay( ) throws SocketException
- SO_REUSEADDR // Java 1.4
  public void setReuseAddress(boolean on) throws SocketException
  public boolean getReuseAddress( ) throws SocketException
- SO_TIMEOUT
  public void setSoTimeout(int milliseconds) throws SocketException
  Public int getSoTimeout( ) throws SocketException
- SO_LINGER
  public void setSoLinger(boolean on, int seconds) throws SocketException
  public int getSoLinger( ) throws SocketException
- SO_SNDBUF/ SO_RCVBUF (Java 1.2 and later)
  public void setReceiveBufferSize(int size) throws SocketException,
  IllegalArgumentException
  public int getReceiveBufferSize( ) throws SocketException
- SO_KEEPALIVE (Java 1.3 and later)
  public void setKeepAlive(boolean on) throws SocketException
  public boolean getKeepAlive( ) throws SocketException

Lập trình máy chủ TCP-

Lớp Java.net.ServerSocket

- Các bước thiết lập máy chủ
  1. Tạo một đối tượng ServerSocket sử dụng hàm khởi tạo ServerSocket( )
  2. ServerSocket chờ kết nối từ phía máy khách bằng hàm *accept()*
     - Trả về một đối tượng Socket kết nối giữa máy khách và máy chủ
  3. Thiết lập các dòng xuất/nhập dữ liệu
  4. Gửi và nhận dữ liệu
  5. Đóng kết nối

Lớp Java.net.ServerSocket

- Có bốn hàm khởi tạo ServerSocket cho phép thiết lập cổng, kích thước hàng đợi của các yêu cầu kết nối và network interface gán cho tiến trình máy chủ
  - public ServerSocket(int port) throws BindException, IOException
  - public ServerSocket(int port, int backlog) throws BindException, IOException
  - public ServerSocket(int port, int backlog, InetAddress bindAddr) throws BindException, IOException
  - public ServerSocket( ) throws IOException // Java 1.4

Lớp Java.net.ServerSocket - Chấp nhận và đóng kết nối

- *public Socket accept() throws IOException*
  - Dừng thực hiện của tiến trình và đợi kết nối từ máy khách
  - Khi có một máy khách kết nối đến, hàm accept( ) sẽ trả về một đối tượng kiểu Socket
  - Chú ý xử lý các loại ngoại lệ khác nhau
- *public void close() throws IOException*
  - Đóng socket máy chủ và giải phóng cổng chờ

Ví dụ về máy chủ DaytimeServer

```
import java.net.*;
import java.io.*;
import java.util.Date;

public class DaytimeServer {
  public final static int DEFAULT_PORT = 13;
  public static void main(String[] args) {
      int port = DEFAULT_PORT;
      if (args.length > 0) {
          try {
                port = Integer.parseInt(args[0]);
                if (port < 0 || port >= 65536) {
                    System.out.println("Port must between 0 and 65535");
                    return;
                }
          }
          catch (NumberFormatException ex) {
                // use default port
          }
      }
```

```
        try {
            ServerSocket server = new ServerSocket(port);
            Socket connection = null;
            while (true) {
                try {
                    connection = server.accept( );
                    Writer out = new OutputStreamWriter(connection.getOutputStream( ));
                    Date now = new Date( );
                    out.write(now.toString( ) +"\r\n");
                    out.flush( );
                    connection.close( );
                }
                catch (IOException ex) {}
                finally {
                    try {
                        if (connection != null) connection.close( );
                    }
                    catch (IOException ex) {}
                }
            }  // end while
        }  // end try
        catch (IOException ex) {
            System.err.println(ex);
        } // end catch
    } // end main
} // end DaytimeServer
```

Lớp Java.net.ServerSocket – Lấy thông tin về socket

- public InetAddress getInetAddress( )
- public int getLocalPort( )

Lớp Java.net.ServerSocket – Tùy biến socket

- SO_TIMEOUT

  public void setSoTimeout(int timeout) throws SocketException

  public int getSoTimeout( ) throws IOException

- SO_REUSEADDR

  public void setReuseAddress(boolean on) throws SocketException

  public boolean getReuseAddress( ) throws SocketException

- SO_RCVBUF

  public void setReceiveBufferSize(int size) throws SocketException, IllegalArgumentException

  public int getReceiveBufferSize( ) throws SocketException

Truyền tin với giao thức UDP

Lớp java.net.DatagramPacket

- Biểu diễn các gói dữ liệu UDP
- Cung cấp các hàm
  - Lấy và thiết lập địa chỉ đích/nguồn từ/vào tiêu đề IP
  - Lấy và thiết lập cổng giao tiếp đích/nguồn
  - Nhận và thiết lập gói dữ liệu UDP

Hàm khởi tạo DatagramPacket

Với bên nhận:

```
DatagramPacket(byte[] buf, int len);
```

Với bên gửi:

```
DatagramPacket( byte[] buf, int len
   InetAddress a, int port);
```

Các hàm DatagramPacket

```
byte[] getData();
void setData(byte[] buf);

void setAddress(InetAddress a);
void setPort(int port);

InetAddress getAddress();
int getPort();
```

Địa chỉ đích

Có thể là địa chỉ/cổng nguồn/đích

Lớp DatagramSocket

- Tạo datagram socket để nhận DatagramPacket.
  - Không có phân biệt giữa socket máy khách và socket máy chủ
  - Một DatagramSocket có thể gửi cho nhiều địa chỉ đích khác nhau.
    - Địa chỉ đích được lưu tại DatagramPacket
- *public DatagramSocket() throws SocketException*

  public DatagramSocket(int port) throws SocketException

  public DatagramSocket(int port, InetAddress laddr) throws SocketException

Lớp DatagramSocket

– Gửi và nhận gói dữ liệu UDP

- public void send(DatagramPacket dp) throws IOException
  - Gửi gói dữ liệu UDP với đối tượng kiểu DatagramPacket được tạo ra
- public void receive(DatagramPacket dp) throws IOException
  - Nhận gói dữ liệu UDP và lưu lại tại đối tượng kiểu DatagramPacket được tạo ra từ trước
- public void close( )
  - Giải phóng cổng đang đựoc sử dụng bởi socket đó
- public int getLocalPort( )
  - Trả về số hiệu cổng mà socket đang sử dụng
- public InetAddress getLocalAddress( )
  - Trả về địa chỉ IP mà socket đang sử dụng

Điều khiển kết nối – với Java 1.2

- public void connect(InetAddress host, int port)
  - Gửi và nhận gói tin từ một địa chỉ IP và cổng được định trước
  - Không giống như kết nối TCP
- public void disconnect( )
- public int getPort( )
- public InetAddress getInetAddress( )
- public InetAddress getRemoteSocketAddress( ) // Java 1.4

Các bước thiết lập truyền tin UDP - MÁY CHỦ

1. Khởi tạo một đối tượng kiểu DatagramSocket
   DatagramSocket dgramSocket =
   new DatagramSocket(1234);
2. Tạo buffer cho dòng dữ liệu nhập
   byte[] buffer = new byte[256];
3. Tạo đối tượng kiểu *DatagramPacket* cho dòng dữ liệu nhập
   DatagramPacket inPacket =
   new DatagramPacket(buffer, buffer.length);
4. Chờ dòng dữ liệu nhập
   dgramSocket.receive(inPacket)

Các bước thiết lập truyền tin UDP - MÁY CHỦ(2)

5. Lấy địa chỉ và cổng của bên gửi từ gói tin nhận được
   InetAddress clientAddress = inPacket.getAddress();
   int clientPort = inPacket.getPort();
6. Lấy dữ liệu từ buffer
   string message =
   new String(inPacket.getData(), 0, inPacket.getLength());
7. Tạo gói dữ liệu UDP xuất

```
DatagramPacket outPacket =
                new DatagramPacket(
                  response.getBytes(), response.length(),
                                       clientAddress, clientPort);
```

8. Gửi gói dữ liệu
   dgramSocket.send(outPacket)
9. Đóng *DatagramSocket:*

```
dgramSocket.close();
```

Các bước thiết lập truyền tin UDP – Máy khách (1)

1. Tạo đối tượng kiểu DatagramSocket
   DatagramSocket dgramSocket = new DatagramSocket;
2. Tạo gói dữ liệu UDP xuất
   DatagramPacket outPacket = new DatagramPacket(
   message.getBytes(),
   message.length(),
   host, port);
3. Gửi gói dữ liệu
   dgramSocket.send(outPacket)
4. Tạo buffer cho dữ liệu nhập
   byte[] buffer = new byte[256];

Các bước thiết lập truyền tin UDP – Máy khách (2)

5. Tạo đối tượng kiểu *DatagramPacket* cho gói dữ liệu nhập

```
DatagramPacket inPacket =
        new DatagramPacket(buffer, buffer.length);
```

6. Nhận gói dữ liệu nhập

```
dgramSocket.receive(inPacket)
```

7. Lấy dữ liệu từ buffer

```
string response = new String(inPacket.getData(), 0,
                                    inPacket.getLength());
```

8. Đóng *DatagramSocket:*

```
dgramSocket.close();
```

Ví dụ về máy chủ UDP

```
import java.net.*;
import java.io.*;

public class UDPDiscardServer {
    public final static int DEFAULT_PORT = 9;
    public final static int MAX_PACKET_SIZE = 65507;
    public static void main(String[] args) {
        int port = DEFAULT_PORT;
        byte[] buffer = new byte[MAX_PACKET_SIZE];
        try {
            port = Integer.parseInt(args[0]);
        }
        catch (Exception ex) {
            // use default port
        }
```

Ví dụ về máy chủ UDP(2)

```
    try {
        DatagramSocket server = new DatagramSocket(port);
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
        while (true) {
            try {
                server.receive(packet);
                String s = new String(packet.getData( ), 0, packet.getLength( ));
                System.out.println(packet.getAddress( ) + " at port "
                                                    + packet.getPort( ) + " says " + s);
//              packet.setLength(buffer.length); // reset the length for the next packet
            }
            catch (IOException ex) {
                System.err.println(ex);
            }
        } // end while
    }  // end try
    catch (SocketException  ex) {
            System.err.println(ex);
    }  // end catch
  }  // end main
}
```

Ví dụ về máy kháchUDP

```
import java.net.*;
import java.io.*;

public class UDPDiscardClient {
      public final static int DEFAULT_PORT = 9;
      public static void main(String[] args) {
                String hostname;
                int port = DEFAULT_PORT;

                if (args.length > 0) {
                                hostname = args[0];
                                try {
                                                port = Integer.parseInt(args[1]);
                                }
                                catch (Exception ex) {
                                   // use default port
                                }
      }
          else {
                                hostname = "localhost";
                }
```

Ví dụ về máy khách UDP(2)

```
        try {
            InetAddress server = InetAddress.getByName(hostname);
            BufferedReader userInput   = new BufferedReader(new InputStreamReader(System.in));
            DatagramSocket theSocket = new DatagramSocket( );
            while (true) {
                String theLine = userInput.readLine( );
                if (theLine.equals(".")) break;
                byte[] data = theLine.getBytes( );
                DatagramPacket theOutput = new DatagramPacket(data, data.length, server, port);
                theSocket.send(theOutput);
            }  // end while
        }  // end try
        catch (UnknownHostException uhex) {
            System.err.println(uhex);
        }
        catch (SocketException socex) {
            System.err.println(socex);
        }
        catch (IOException ioex) {
            System.err.println(ioex);
        }
    }  // end main
}
```