Chương 4
Lập trình mạng với JAVA

Giảng viên: Nguyễn Đức Thái
thai@cse.hcmut.edu.vn

Nội dung

- Giới thiệu ngôn ngữ Java
- Những ví dụ với ngôn ngữ Java
- Lập trình sockets với Java

Giới thiệu ngôn ngữ Java

Sơ lược về ngôn ngữ Java

- Java là một ngôn ngữ lập trình
- Người dùng có thể download bytecode Java từ Internet về và chạy trên máy của mình, rất đơn giản
- Chương trình java chạy trên những trang web, được gọi là Applets
- Để chạy được applets, bạn cần có trình duyệt (browser) với chức năng Java được mở (Java-enabled)
- Java ra đời năm 1991 khi một nhóm kỹ sư hãng Sun muốn thiết kế một ngôn ngữ máy tính đơn giản có thể được dùng cho thiết bị người dùng, như hộp chuyển mạch (switchbox) của cable TV
- 1996: Phiên bản đầu tiên của Java ra đời

Chương trình Java đơn giản

```
public class ClassName
{
  public static void main(String[] args)
  {
    program instructions
  }
}
```

```
public class FirstSample
{
  public static void main(String[] args)
  {
    System.out.println("Hello, World!");
  }
}
```

Comments

Example: FirstSample.java

```
1. /*
2.    This is the first sample program in Core Java Chapter 3
3.    Copyright (C) 1996...2000 Cay Horstmann and Gary Cornell
4. */
5.
6. public class FirstSample
7. {
8.    public static void main(String[] args)
9.    {
         // is this too cute?
10.       System.out.println("Hello, World!");
11.    }
12. }
```

Types trong ngôn ngữ Java

Types chính trong ngôn ngữ Java

- Integers
  - `byte` (1 byte, -128 to 127)
  - `short` (2 bytes)
  - `int` (4 bytes)
  - `long` (8 bytes)
- Floating-point types
  - `float` (4 bytes, 6-7 significant decimal digits)
  - `double` (8 bytes, 15 significant decimal digits)
- `char` (2 byte, Unicode) (ASCII 1 byte)
- `boolean` (true or false)

Biến & Hằng số

Biến trong ngôn ngữ Java

- Biến có thể được khai báo bất cứ nơi nào

```
for (int i = 0; i < 20; i++) {
    System.out.println("Hi");
    char ch = 'A';
}
double pi = 3.14159;
```

- Biến cục bộ (local variable)

```
public void someMethod()
{
    int x; // does not compile
}
```

- Instance variables of class automatically initialized.

Hằng số

- Biến dạng "read-only"
  - Dùng một lần và không thể thay đổi sau đó

```
public void someMethod()
{ final double pi = 3.14159;
   .. .. ..
  pi = 3.14; // illegal
}
```

  - Dùng **static final** để định nghĩa hằng số (constant)

```
public class Time {
   static final int MinHour = 0;
   static final int MaxHour = 23;
   private int hour, minute;
   // these properties are set to 0
   // unless overwritten by constructor
   … }
```

Phép tính

Operators trong ngôn ngữ Java

- Cơ bản là giống như trong ngôn ngữ C++

```
+, -, *, /, %, ++, --,
<, <=, >, >=, ==, !=,
 !, &&, ||
=, +=, -=, *=, /=, %=,
```

- User cannot "overload" operators; although + is overloaded to do string concatenation
  - In C++:
    - Int x=0; x += 1;
    - String& String::operator+=(const String &s)..

- Note that methods can be overloaded

Operators trong ngôn ngữ Java

- No Pointers!
  - Không tồn tại con trỏ (pointer) trong ngôn ngữ Java
  - Không tồn tại phép tính & và *
    - Trong C++:
      - `int *i = (int *) malloc( 3 * sizeof(int));`
      - `(i+1)* = 2;`
      - `int& y = &i;`
  - No pointer arithmetic
  - No function pointers

Array trong ngôn ngữ Java (1)

- Arrays là những đối tượng của lớp `java.lang.reflect.Array`
- Không thể dùng size khi khai báo array

```
int arr[3];     // không hợp lệ trong Java!
int arr[];      // hợp lệ
int[] arr;      // hợp lệ
```

- Arrays (as all objects) được cấp phát động (dynamically allocated)

```
int[] arr = new int[3];
```

- Trước khi cấp phát, biến array có giá trị `null`
- Tất cả phần tử bằng 0 khi array được cấp phát
- Destroyed automatically by garbage collector. No delete operator
- Shorthand to declare, allocate, and initialize

```
int[] arr = { 5, 10, 15, 20};
```

Array trong ngôn ngữ Java (2)

- Java array luôn biết độ dài của chính nó

```
int[] arr = {5, 20, 15, 10};
System.out.println("Length is " + arr.length);
```

- Phần tử của array được đánh số từ 0 to length-1, like C++
- Raises exception for "ArrayIndexOutOfBounds"
- Độ dài cố định khi được cấp phát (allocated), tạo array mới và copy vào để thay đổi độ dài

Array trong ngôn ngữ Java (3)

- Được dùng tương tự như những array bình thường

```
int sum(int[] arr)
{
   int i, sum = 0;
   for (i=0; i < arr.length; i++)
        sum += arr[i];
   return sum;
}
```

Array are references

- Arrays là đối tượng, vì thế, được hiện thực như references (reference is a pointer in disguise)

```
int[] arr = {5, 20, 15, 10};
int[] b = arr;
b[0] = 3;  // arr[0] also becomes 3!
```

- Array copying (`java.lang.System`)

```
System.arraycopy(from, fromIndex, to, toindex,
  count);
int[] c;
System.arraycopy(arr, 0, c, 0, 4);
```

- Array sorting (`java.util.Arrays`)

```
Arrays.sort(arr)  //use a tuned QuickSort
```

Luồng nhập xuất

- Luồng là một "dòng chảy" của dữ liệu được gắn với các thiết bị đầu ra.
- Có hai loại luồng:
  - Luồng nhập: gắn với các thiết bị nhập như bàn phím, máy scan, file,…
  - Luồng xuất: gắn với các thiết bị xuất như màn hình, máy in, file,…
- Việc xử lý vào ra thông qua luồng giúp cho lập trình viên không cần quan tâm đến bản chất của thiết bị vào ra.

Standard Streams

Có 3 standard stream

- Input: System.in
- Output: System.out
- Error: System.err

Những đối tượng này được định nghĩa tự đông.

System.out và System.err được định nghĩa như là những PrintStream Object.

Ví dụ

Chương trình java cho phép nhập vào một chuỗi bất kỳ từ bàn phím, sau đó in chuỗi vừa nhập ra màn hình.

Ví dụ

```
import java.io.*;

public class First {
    public static void main(String[] args) {
        System.out.print("Nhập 1 chuỗi bất kỳ: ");
        BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));
        String userName = null;
        try {
            userName = br.readLine();
        } catch (IOException ioe) {
            System.out.println("Có lỗi xảy ra");
             System.exit(1);
        }
        System.out.println("bạn vừa nhập chuỗi: " + userName);
    }
}
```

Các lệnh điều khiển

- Điều khiển rẽ nhánh:
  - Mệnh đề **if-else**
  - Mệnh đề **switch-case**
- Vòng lặp (Loops):
  - Vòng lặp **while**
  - Vòng lặp **do-while**
  - Vòng lặp **for**

Lệnh **if-else**

- Cú pháp

```
if (condition) {
    action1 statements;
}
Else {
    action2 statements;
}
```

Lệnh **switch-case**

- Cú pháp

```
switch (expression)
{
        case 'value1': action1 statement(s);
                       break;
        case 'value2': action2 statement(s);
                       break;
        :
        :
        case 'valueN': actionN statement(s);
                       break;
        default:  default_action statement(s);
}
```

Lệnh lặp **while**

- Cú pháp

```
while(condition)
{
        action statements;
        :
        :
}
```

Lệnh lặp **do-while**

- Cú pháp

```
do
{
        action statements;
        :
        :
} while(condition);
```

Vòng lặp **for**

- Cú pháp

```
for (initialization statements; condition; increment statements)
{
        action statements;
        :
        :
}
```

Những phương thức của lớp **String**

- charAt( )
- startsWith()
- endsWith( )
- copyValueOf( )
- toCharArray( )
- indexOf( )
- toUpperCase( )
- toLowerCase( )
- trim( )
- equals( )

Lớp java.lang.Math

- abs()
- ceil()
- floor()
- max()
- min()
- round()
- random()
- sqrt()
- sin()
- cos()
- tan()

Các kí tự định dạng xuất dữ liệu

Escape Sequence	Mô tả
\n	Xuống dòng mới
\r	Chuyển con trỏ đến đầu dòng hiện hành
\t	Chuyển con trỏ đến vị trí dừng Tab kế tiếp (ký tự Tab)
\\	In dấu \
\’	In dấu nháy đơn (’)
\”	In dấu nháy kép (”)

Thư viện ngôn ngữ Java (1)

- Các lớp được tập hợp thành từng gói (package)
- Thư viện ngôn ngữ Java tương tự như trong ngôn ngữ C
- java.lang
  - String, Math, Exception, Thread, Runtime, etc
- java.util
  - Vector, Stack, hashtable, Date, Tokenizer
- java.io
  - Varieties of input/output processing
- java.net
  - Networking, client/server sockets, URLs
- java.awt, javax.swing
  - Windows, buttons, drawing, images, events

Thư viện ngôn ngữ Java (2)

- java.security
  - Encryption, digital signature, message digest
- java.text
  - Formatting and parsing
- java.sql
  - Database connectivity
- java.rmi
  - Remote method invocation, distributed objects

Java: hostname và IP

- java.net.InetAddress class chuyển đổi giữa hostname và IP
  - `InetAddress tm = InetAddress.getByName("www.yahoo.com");`
  - `InetAddress tm= InetAddress.getByName("localhost"); //127.0.0.1`
  - `InetAddress tm = InetAddress.getLocalHost();`

- Nhận array of addresses (nếu có nhiều hơn một địa chỉ)
  - `InetAddress[] addrs;`
  - `addrs=InetAddress.getAllByName("www.yahoo.com");`
  - ```
    for (int i = 0; i < addr.length; i++)
      System.out.println(addrs[i].getHostAddress());
    ```
    ```

Java: port

- Nhiều dịch vụ khác nhau có thể chạy trên host
- Port là địa chỉ của một dịch vụ bên trong một host
- Nhiều dịch vụ thông dụng được gán ports:
  - FPT – port 21
  - Telnet – port 23
  - SMTP – port 25
  - HTTP – port 80
  - …

Java: giao thức

- Giao thức là tập hợp những quy tắc quy định cách giao tiếp giữa những thành phần khác nhau (machines)
- Một số giao thức:
  - HTTP: HyperText Transfer Protocol
  - FTP: File Transfer Protocol
  - SMTP: Simple Message Transfer Protocol
  - TCP: Transmission Control Protocol
  - UDP: User Datagram Protocol, good for, e.g., video delivery)

Lập trình mạng với Java

Những class của gói Java API: java.net

- InetAddress
- ServerSocket
- Socket
- URL
- URLConnection
- DatagramSocket

InetAddress Class

- Class dùng cho địa chỉ trên Internet (Internet Protocol)
- Use methods: `getLocalHost, getByName`, hoặc `getAllByName` *để tạo một* `InetAddress` instance:
  - `public static InetAddess InetAddress.getByName(String hostname)`
  - `public static InetAddess [] InetAddress.getAllByName(String hostname)`
  - `public static InetAddess InetAddress.getLocalHost()`
- Tìm địa chỉ IP hoặc hostname của một host:
  - `getHostAddress()`
  - `getHostName()`

Ví dụ: In địa chỉ IP của localhost

```
import java.net.*;
public class HostInfo {
    public static void main(String args[]) {
    HostInfo host = new HostInfo();
    host.init();
}
    public void init() {
    try {
    InetAddress myHost = InetAddress.getLocalHost();
    System.out.println(myHost.getHostAddress());
    System.out.println(myHost.getHostName());
    } catch (UnknownHostException ex) {
        System.err.println("Cannot find local host");
        }
   }
}
```

Ví dụ: In địa chỉ IP proxy.hcmut.edu.vn

```
import java.net.*;
class kku {
    public static void main (String args[]) {
      try {
              InetAddress[] addresses =
              InetAddress.getAllByName("proxy.hcmut.edu.vn");
              for (int i = 0; i < addresses.length; i++) {
                    System.out.println(addresses[i]);
              }
      }
      catch (UnknownHostException e) {
      System.out.println("Could not find proxy.hcmut.edu.vn");
      }
    }
}
```

Socket Class (1)

- Dùng để mô tả một *socket*
- Dùng để tạo một *socket*
  - **Socket**(InetAddress address, int port)
  - **Socket**(String host, int port)
  - **Socket**(InetAddress address, int port, InetAddress, localAddr, int localPort)
  - **Socket**(String host, int port, InetAddress, localAddr, int localPort)
  - **Socket**()

Socket Class (2)

- Dùng để lấy thông tin của socket
  - InetAddress **getInetAddress**()
  - int **getPort**()
  - InetAddress **getLocalAddress**()
  - int **getLocalPort**()
- Sử dụng Streams
  - public OutputStream **getOutputStream**() throws IOException
  - Trả về một output stream cho việc viết các byte đến socket này.
  - public InputStream **getInputStream()** throws IOException
  - Trả về một input stream cho việc đọc các byte từ socket này.

Giao tiếp với Web server (1)

```
import java.net.*;
import java.io.*;
public class getSocketInfo {
   public static void main(String[] args) {
     for (int i = 0; i < args.length; i++) {
         try {
         Socket theSocket = new Socket(args[i], 80);
         System.out.println("Connected to " +
         theSocket.getInetAddress() +
         " on port " + theSocket.getPort() + " from
           port " +
         theSocket.getLocalPort() + " of " +
         theSocket.getLocalAddress());
```

Giao tiếp với Web server (2)

```
        } catch (UnknownHostException e) {
            System.err.println("I can't find " + args[i]);
            } catch (SocketException e) {
                System.err.println("Could not connect to " +
                  args[i]);
            } catch (IOException e) {
                System.err.println(e);
            }
} // end for
} // end main
} // end getSocketInfo
```

ServerSocket Class (1)

- Class mô tả về *ServerSocket*
- Tạo một *ServerSocket*
  - **ServerSocket**(int port) throws IOException
  - **ServerSocket**(int port, int backlog) throws IOException
  - **ServerSocket**(int port, int backlog, InetAddress bindAddr) throws IOException

ServerSocket Class (2)

- Các phương thức trong ServerSocket
  - Socket **accept**() throws IOException: Lắng nghe một kết nối đến socket này và chấp nhận nó.
  - void **close**() throws IOException: Đóng socket
  - InetAddress **getInetAddress**(): Trả về địa chỉ cục bộ của socket
  - int **getLocalPort**(): Trả về port mà server đang lắng nghe
  - void **setSoTimeout**(int timeout) throws SocketException
  - Enable/disable SO_TIMEOUT với khai báo timeout (miliseconds)

Ví dụ: DateTime server (1)

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
   public final static int daytimePort = 5000;
   public static void main(String[] args) {
      ServerSocket theServer;
      Socket theConnection;
      PrintStream p;
      try {
         theServer = new ServerSocket(daytimePort);
```

Ví dụ: DateTime server (2)

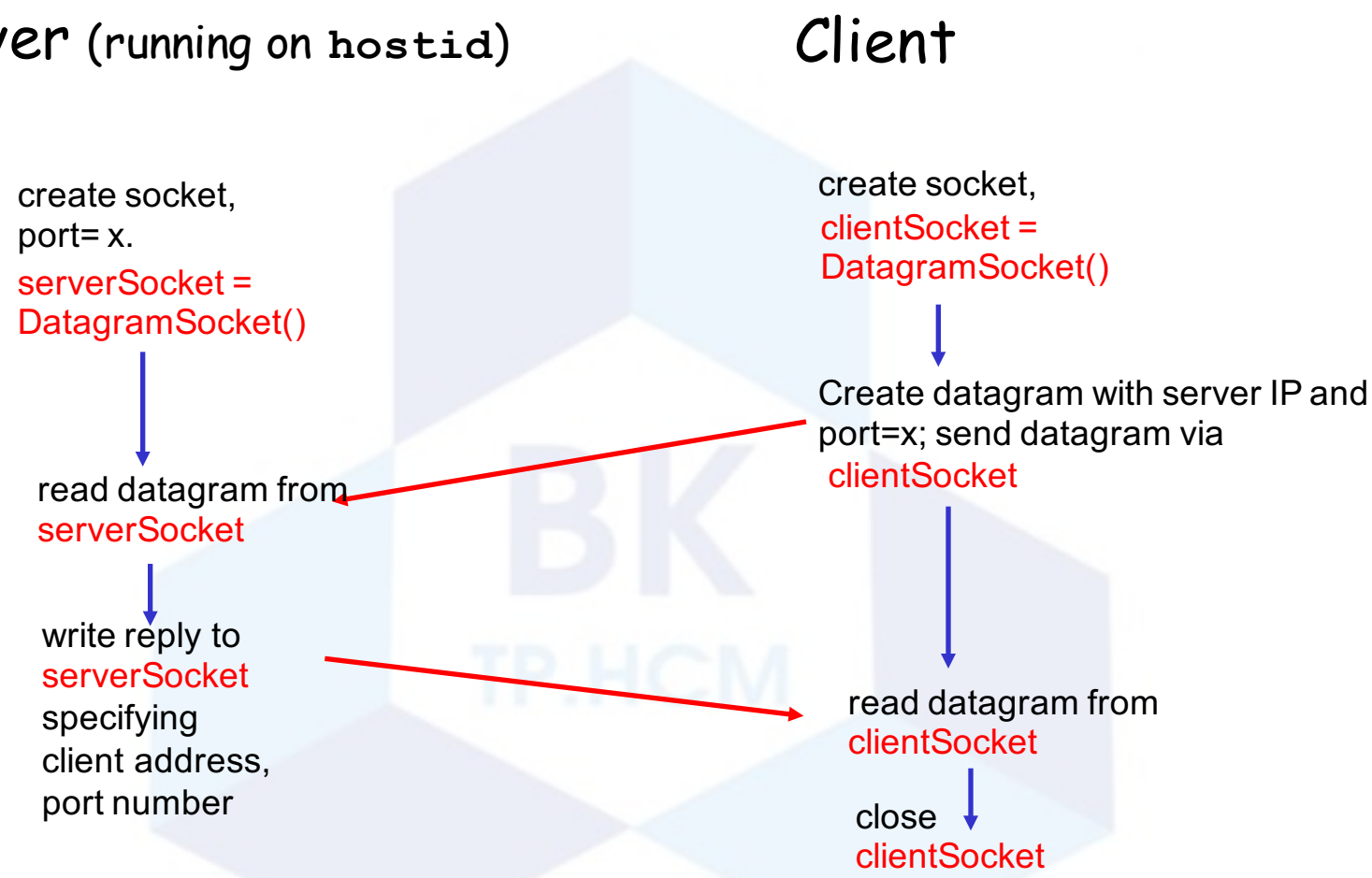
```
        while (true) {
            theConnection = theServer.accept();
            p = new
            PrintStream(theConnection.getOutputStream());
            p.println(new Date());
            theConnection.close();
            theServer.close();
        }

    } catch (IOException e) {
        System.err.println(e);
    }
}
}
```

Lập trình socket với UDP (1)

- Cung cấp cơ chế truyền không tin cậy giữa các nhóm các byte (datagrams) giữa client và server.
- Không cần thiết lập kết nối giữa client và server.
- Sender phải gửi kèm địa chỉ IP và port đích
- Server khi nhận dữ liệu sẽ phân tích địa chỉ của sender để truyền lại
- Có thể server chấp nhận nhiều client tại một thời điểm.

Lập trình socket với UDP (2)



Ví dụ: UDP Client (1)

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

class UDPClient{
   public static void main(String args[]) throws Exception
   {
     BufferedReader inFromUser =
       new BufferedReader(new InputStreamReader(System.in));

     DatagramSocket clientSocket = new DatagramSocket();

     InetAddress IPAddress = InetAddress.getByName("hostname");

     byte[] sendData = new byte[1024];
     byte[] receiveData = new byte[1024];

     String sentence = inFromUser.readLine();

     sendData = sentence.getBytes();
```

Create input stream

Create client socket

Translate hostname to IP address using DNS

Ví dụ: UDP Client (2)

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Read datagram from server

```
        DatagramPacket sendPacket =
          new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

        clientSocket.send(sendPacket);

        DatagramPacket receivePacket =
           new DatagramPacket(receiveData, receiveData.length);

        clientSocket.receive(receivePacket);

        String modifiedSentence =
            new String(receivePacket.getData());

        System.out.println("FROM SERVER:" + modifiedSentence);
        clientSocket.close();
        }
    }
```

Ví dụ: UDP server (1)

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

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

    DatagramSocket serverSocket = new DatagramSocket(9876);

    byte[] receiveData = new byte[1024];
    byte[] sendData  = new byte[1024];

    while(true)
     {

      DatagramPacket receivePacket =
        new DatagramPacket(receiveData, receiveData.length);

      serverSocket.receive(receivePacket);
```

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

Ví dụ: UDP server (2)

```
String sentence = new String(receivePacket.getData());

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

        String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket =
   new DatagramPacket(sendData, sendData.length, IPAddress,
                      port);

serverSocket.send(sendPacket);
    }
  }
}
```

Get IP addr port #, of sender

Create datagram to send to client

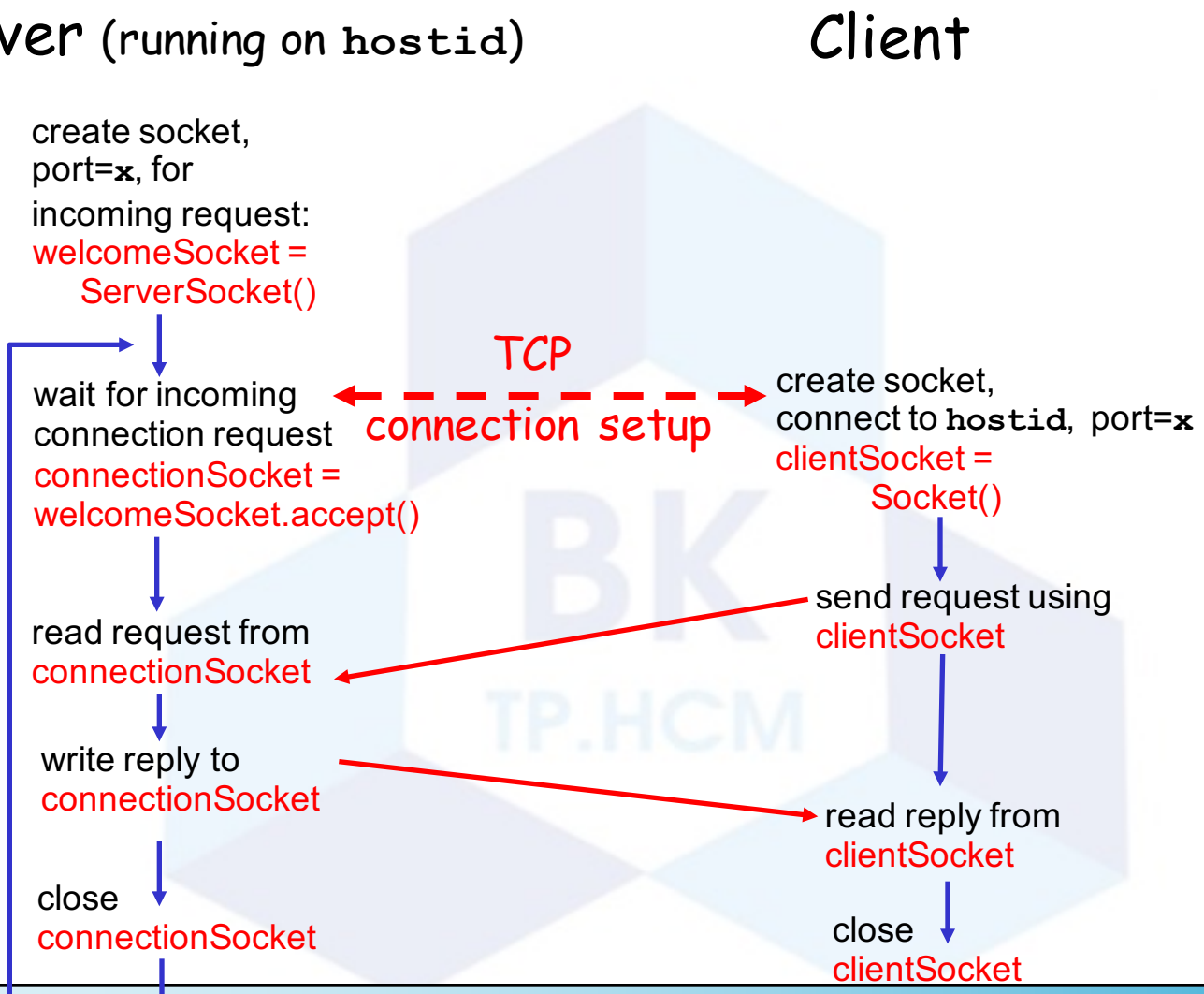
Write out datagram to socket

End of while loop, loop back and wait for another datagram

Lập trình socket với TCP

- **Server**
  - Server process phải chạy trước.
  - Server phải tạo một socket lắng nghe và chấp nhận các kết nối từ client.
- **Client**
  - Khởi tạo TCP socket.
  - Xác định IP address, port number của server.
  - Thiết lập kết nối đến server.
- Khi server nhận yêu cầu kết nối, nó sẽ chấp nhận yêu cầu và khởi tạo socket mới giao tiếp với client.
  - Có thể server chấp nhận nhiều client tại một thời điểm.

Lập trình socket với TCP



Ví dụ: TCP client

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream

Create client socket, connect to server

Create output stream attached to socket

Ví dụ: TCP client (tiếp theo)

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

Send line to server

```
outToServer.writeBytes(sentence + '\n');
```

Read line from server

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

    }
}
```

Ví dụ: TCP server (1)

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

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

       Socket connectionSocket = welcomeSocket.accept();

      BufferedReader inFromClient =
        new BufferedReader(new
        InputStreamReader(connectionSocket.getInputStream()));
```

Create welcoming socket at port 6789

Wait, on welcoming socket for contact by client

Create input stream, attached to socket

Ví dụ: TCP server (2)

Create output stream, attached to socket

Read in line from socket

Write out line to socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());

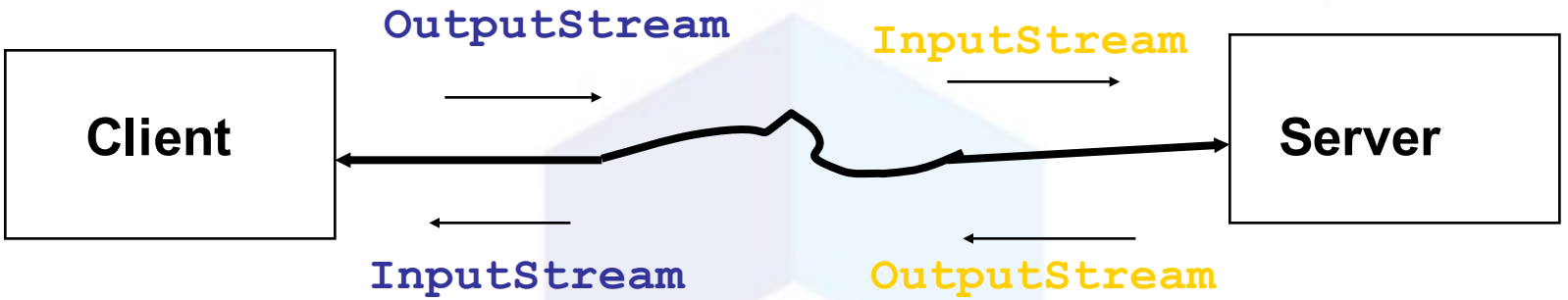
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';

outToClient.writeBytes(capitalizedSentence);
            }
        }
    }
```

End of while loop, loop back and wait for another client connection

Java: Lập trình sockets



- Viết một chương trình client như thế nào ?
- Viết một chương trình server như thế nào ?

Viết chương trình Client (1)

- Dùng **java.net.Socket**
  - Tạo một socket mới với hostname và port

```
Socket s = New Socket(String hostName,int portNumber);
```

  - Gọi `s.getOutputStream()` và `s.getInputStream()` để nhận streams dùng cho việc gởi và nhận thông tin
  - Cần nắm vững giao thức để giao tiếp
    - Cần biết làm thế nào gởi request đến server
    - Cần biết làm thế nào để hiểu những hồi âm từ server

Viết chương trình Client (2)

- SocketTest:

```
try
{
  Socket s = new Socket("cse.hcmut.edu.vn", 13);
   BufferedReader in = new BufferedReader
                (new InputStreamReader(
                 s.getInputStream() ));
          // read from in
 }
  catch (IOException e)
 {
   e.printStackTrace();
 }
```

Viết chương trình Server

- Viết chương trình Server dùng **java.net.ServerSocket**
  - Tạo ServerSocket với một port
    - `ServerSocket s = New ServerSocket( portNumber);`
  - Dùng accept() lắng nghe tại port vừa tạo
  - `accept()` returns a socket **incoming** khi một client gọi
    - `Socket incoming = s.accept();`
  - Gọi `incoming.getOutputStream()` và `incoming.getInputStream()` để có streams cho việc gởi và nhận thông tin

Viết chương trình Server – Ví dụ

Ví dụ: Echo server

```
ServerSocket s = new ServerSocket(8189);
Socket incoming = s.accept( );
BufferedReader in = new BufferedReader
   (new
   InputStreamReader(incoming.getInputStream())
   );
PrintWriter out = new PrintWriter
   (incoming.getOutputStream(), true
      /* autoFlush */ );
out.println( "Hello! Enter BYE to exit." );
            …
```

Multithread Server – Ví dụ (1)

Multithread server: bắt đầu một threat riêng biệt cho mỗi kết nối.

```
public class ThreadedEchoServer
{  public static void main(String[] args )
   {  int i = 1;
      try{ServerSocket s = new ServerSocket(8190);
      while (true)
      { Socket incoming = s.accept( );
        System.out.println("Spawning " + i);
        new ThreadedEchoHandler(incoming, i).start();
        i++;
      }
     } catch (Exception e) …. //ThreadedEchoServer.java
```

Multithread Server – Ví dụ (2)

```
class ThreadedEchoHandler extends Thread
{  public ThreadedEchoHandler(Socket i, int c)
   { incoming = i; counter = c; }
   public void run()
   {  try
      {  BufferedReader in = new BufferedReader
         (new InputStreamReader(incoming.getInputStream()));
         PrintWriter out = new PrintWriter
            (incoming.getOutputStream(), true /* autoFlush
  */);
         out.println( "Hello! Enter BYE to exit." );
      …
   private Socket incoming;
   private int counter; }
```

Giao tiếp với Web servers

Giao tiếp với Web servers (1)

- Mục đích giao tiếp với Web servers
  - Gởi / nhận thông tin
- Class `java.net.URL` tượng trưng (represent) cho một URL
  - Tạo một đối tượng Java mà tượng trưng cho một URL

```
URL url = new URL("http://www.cse.hcmut.edu.vn/index.html");
```

    - getHost(), getPath(), getPort(), getProtocol()
- java.net.URLConnection tượng trưng cho một liên kết giao tiếp giữa ứng dụng và một URL.
  - Constructor:
    - `URLConnection cnn = new` URLConnection(url)
  - Có thể nhận được từ URL:
    - `URLConnection cnn = url.openConnection();`

Giao tiếp với Web servers (2)

- Những bước làm việc với **`java.net.URLConnection`**
  - Set properties of connection:
    - **`setDoInPut(true)`** //default
    - **`setDoOutPut(true)`** for sending information to the server
    - **`...`**
  - Tạo kết nối: **`cnn.connect();`**
  - Query header information:
    - **`getContentType, getContentLength, getContentEncoding, getDate, getExpiration, getLastModified`**
  - **`getInputStream`** for reading and **`getOutputStream`** for writing
- API of the class has a more detailed description.

Giao tiếp với Web servers (3)

- Can directly open a stream for reading in URL class:
  - public final InputStream **openStream**() throws IOException

    url.opentStream()
    - Opens a connection to this URL and returns an InputStream for reading from that connection.
    - This method is a shorthand for: openConnection().getInputStream()

Lấy thông tin (1)

- URLConnectionTest.java

```
URL url = new URL(urlName);
URLConnection connection = url.openConnection();

connection.connect();

// print header fields
int n = 1;
String key;
while ((key = connection.getHeaderFieldKey(n)) != null)
{
   String value = connection.getHeaderField(n);
   System.out.println(key + ": " + value);
   n++;
}
```

Lấy thông tin (2)

```
// print convenience functions

        System.out.println("----------");
        System.out.println("getContentType: "
           + connection.getContentType() );

        System.out.println("getContentLength: "
           + connection.getContentLength() );

        System.out.println("getContentEncoding: "
           + connection.getContentEncoding() );
        ….
```

Lấy thông tin (3)

```
// print first ten lines of contents
     BufferedReader in = new BufferedReader(new
        InputStreamReader( connection.getInputStream()
  ));

     String line;
     n = 1;
     while ((line = in.readLine()) != null && n <= 10)
     {
            System.out.println(line);
            n++;
     }
     if (line != null) System.out.println(". . .");
```

Gởi thông tin

- Web servers receive information from clients using either GET or POST
  - GET requests are requests made by browsers when the user
    - types in a URL on the address line,
    - follows a link from a Web page, or
    - makes an HTML form that does not specify a METHOD or specifically use the GET method.
  - POST requests are generated when someone creates an HTML form that specifies METHOD="POST"
  - Examples:
    - http://maps.yahoo.com/py/maps.py: python,
    - <form action="/py/maps.py?Pyt=Tmap&YY=28457" method=GET> … </form>
    - http://www.census.gov/ipc/www/idbprint.html:
      - <form method=post action="/cgi-bin/ipc/idbsprd">

Tài Liệu Tham Khảo

- [1] Bộ Slides cũ môn Lập Trình Mạng, Khoa KH&KTMT, Trường Đại Học Bách Khoa TP.HCM