TRƯỜNG ĐẠI HỌC DUY TÂN
Khoa Công Nghệ Thông tin
Bộ môn Kỹ thuật mạng
---0O0---

BÀI THỰC HÀNH

HỆ PHÂN TÁN

Biên soạn
ThS. Nguyễn Minh Nhật

Đà Nẵng, 08/2018

I. MỤC TIÊU THỰC HÀNH

- Giới thiệu cho sinh viên các kiến thức căn bản về kỹ thuật RMI, đây là kỹ thuật được sử dụng phổ biến để phát triển các ứng dụng phân tán trong java.
- Phát triển các ứng dụng phân tán khác nhau trên kỹ thuật RMI.

II. HỌC LIỆU

- Học liệu :
  - Đĩa USB
  - Phần mềm (JDK1.5.0 hoặc trở lên, Jcreator 4.0,NetBean 6.x)
  - Bài tập thực hành của giảng viên cung cấp
- Dụng cụ :
  - Projector, màn chiếu
  - Máy tính
  - Các phần mềm dạy học
  - Bảng, phấn
  - Hệ thống máy tính được kết nối mạng

III. NỘI DUNG

Gồm 8 Lab với các nội dung sau:

- **LAB_01 :** Xử lý dòng và File (Buổi 01)
- **LAB_02 :** Lập trình Socket và Multi Socket (Buổi 02)
- **LAB_03 :** Lập trình Threat và Multi Threat (Buổi 03)
- **LAB_04 :** Lập trình UDP (Buổi 04)
- **LAB_05 :** Lập trình TCP (Buổi 05)
- **LAB_06 :** Lập trình Databsae(Buổi 06)
- **LAB_07 :** Lập trình RMI (Buổi 07,08)
- **LAB_08 :** Xử lý các bài toán ứng phân tán cơ bản bằng kỹ thuật RMI (Buổi 09)
- **Kiểm tra thực hành**

VI.TÀI LIỆU THAM KHẢO

- **Tài liệu chính**

[1]. Tập bài giảng " *Hệ phân tán* ", của giảng viên Nguyễn Minh Nhật
[2]. Jie Wu, "*Distributed Systems Design*", Addison-Wesley, 2004

- **Tài liệu tham khảo**

[3]. S. Mullender ed., "*Distributed Systems*", 2nd ed., Addison-Wesley, 1993
[4]. G. Coulouris, J. Dollimore, T. Kinberg, *"Distributed systems : Concept and Design*"

[5]. Spiegel, A. (1998). Objects by value: Evaluating the trade-off. In *Proceedings Int. Conf. on Parallel and Distributed Computing and Networks (PDCN)*, pages 542- 548, Brisbane, Australia. IASTED, ACTA Press.
[]. Van Steen, M., Homburg, P., and Tanenbaum, A. (1999). Globe: A Wide-Area Distributed System. *IEEE Concurrency*, pages 70-78.
[6]. Waldo, J., Wyant, G., Wollrath, A., and Kendall, S. (1997). A Note on Distributed Computing. In Vitek, J. and Tschudin, C., editors, *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes in Computer Science*, pages 49-64. Springer-Verlag.

- **Tài liệu Internet**

[5]. http://java.sun.com

[6]. http://elegantjbeans.com/data_aware_bean.htm

[7]. http://www.comp.dit.ie/coleary/teaching/dt2494ds/notes/11/index.html

[8]. http://en.wikibooks.org/wiki/Distributed_Systems#Synchronization

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** Bài số : 01 Số giờ : 03giờ

GVHD : ThS.Nguyễn Minh Nhật

LAB 01

XỬ LÝ VÀO /RA TRÊN PC & HỆ THỐNG MẠNG

I.MỤC TIÊU

Củng cố một số kiên thức làm cơ sở cho thiết kế và lập trình phân tán với java, gồm các vấn đề như :

- Dòng và File
- Xử lý đọc, ghi trên dòng và File

II. NỘI DUNG

A. LÝ THUYẾT

Luồng : Luồng byte, luồng ký tự

a.Luồng Byte :

+ Lớp trừu tượng : InputStream và OutputStream

+ Các phương thức hỗ trợ

Lớp luồng byte	Ý nghĩa
BufferedInputStream	Luồng vào trên Bufered
BufferedOutputStream	Luồng ra trên Bufered
ByteArrayInputStream	Input stream đọc dữ liệu từ một mảng byte
ByteArrayOutputStream	Output Stream ghi dữ liệu từ một mảng
DataInputStream	Luồng nhập có những phương thức đọc những kiểu dữ liệu chuẩn trong java
DataOutputStream	Luồng xuất có những phương thức ghi những kiểu dữ liệu chuẩn trong java
FileInputStream	Luồng nhập cho phép đọc dữ liệu từ file
FileOutputStream	Luồng xuất cho phép ghi dữ liệu xuống file
FilterInputStream	Hiện thực lớp trừu tượng InputStream
FilterOutputStream	Hiện thực lớp trừu tượng OutputStream
InputStream	Lớp trừu tượng, là lớp cha của tất cả các lớp luồng nhập kiểu Byte
OutputStream	Lớp trừu tượng, là lớp cha của tất cả các lớp xuất nhập kiểu Byte
PipedInputStream	Luồng nhập byte kiểu ống (piped) thường phải được gắn với một luồng xuất kiểu ống.

PipedOutputStream	Luồng nhập byte kiểu ống (piped) thường phải được gắn với một luồng nhập kiểu ống để tạo nên một kết nối trao đổi dữ liệu kiểu ống.
PrintStream	Luồng xuất có chứa phương thức print() và prinln()
PushbackInputStream	Là một luồng nhập kiểu Byte mà hỗ trợ thao tác trả lại (push back) và phục hồi thao tác đọc một byte (unread)
RandomAccessFile	Hỗ trợ các thao tác đọc, ghi đối với file truy cập ngẫu nhiên·
SequenceInputStream	Là một luồng nhập được tạo nên bằng cách nối kết logic các luồng nhập khác.

Những phương thức định nghĩa trong lớp **InputStream** và **OutputStream**

Phương thức	*Ý nghĩa*
InputStream	
int available( )	- Trả về số lượng bytes có thể đọc được từ luồng nhập
void close( )	- Đóng luồng nhập và giải phóng tài nguyên hệ thống gắn với luồng. Không thành công sẽ ném ra một lỗi IOException
void mark(int numBytes)	- Đánh dấu ở vị trí hiện tại trong luồng nhập
boolean markSupported( )	- Kiểm tra xem luồng nhập có hỗ trợ phương thức mark() và reset() không.
int read( )	Đọc byte tiếp theo từ luồng nhập
int read(byte buffer[ ])	- Đọc buffer.length bytes và lưu vào trong vùng nhớ buffer. Kết quả trả về số bytes thật sự đọc được
int read(byte buffer[ ], int offset, int numBytes)	- Đọc numBytes bytes bắt đầu từ địa chỉ offset và lưu vào trong vùng nhớ buffer. Kết quả trả về số bytes thật sự đọc được
void reset( )	- Nhảy con trỏ đến vị trí được xác định bởi việc gọi hàm mark() lần sau cùng.
long skip(long numBytes)	- Nhảy qua numBytes dữ liệu từ luồng nhập
Output Stream	
void close( )	- Đóng luồng xuất và giải phóng tài nguyên hệ thống gắn với luồng. Không thành công sẽ ném ra một lỗi IOException
void flush( )	- Ép dữ liệu từ bộ đệm phải ghi ngay xuống luồng (nếu có)
void write(int b)	- Ghi byte dữ liệu chỉ định xuống luồng
void write(byte buffer[ ])	- Ghi buffer.length bytes dữ liệu từ mảng chỉ định

void write(byte buffer[ ], int offset, int numBytes)	xuống luồng - Ghi numBytes bytes dữ liệu từ vị trí offset của mảng chỉ định buffer xuống luồng

b.Luồng ký tự (Character Streams)

+ Lớp trừu tượng : Reader và Writer

+ Các phương thức hỗ trợ

Lớp luồng ký tự	Ý nghĩa
BufferedReader	Luồng nhập ký tự đọc dữ liệu vào một vùng đệm
BufferedWriter	Luồng xuất ký tự ghi dữ liệu tới một vùng đệm
CharArrayReader	Luồng nhập đọc dữ liệu từ một mảng ký tự
CharArrayWriter	Luồng xuất ghi dữ liệu tời một mảng ký tự
FileReader	Luồng nhập ký tự đọc dữ liệu từ file
FileWriter	Luồng xuất ký tự ghi dữ liệu đến file
FilterReader	Lớp đọc dữ liệu trung gian(lớp trừu tượng)
FilterWriter	Lớp xuất trung gian trừu tượng
InputStreamReader	Luồng nhập chuyển bytes thành các ký tự (*)
LineNumberReader	Luồng nhập đếm dòng
OutputStreamWriter	Luồng xuất chuyển những ký tự thành các bytes(*)
PipedReader	Luồng đọc dữ liệu bằng cơ chế đường ống
PipedWriter	Luồng ghi dữ liệu bằng cơ chế đường ống
PrintWriter	Luồng ghi văn bản ra thiết bị xuất (chứa phương thức print() và println())
PushbackReader	Luồng nhập cho phép đọc và khôi phục lại dữ liệu
Reader	Lớp nhập dữ liệu trừu tượng
StringReader	Luồng nhập đọc dữ liệu từ chuỗi
StringWriter	Luồng xuất ghi dữ liệu ra chuỗi
Writer	Lớp ghi dữ liệu trừu tượng

B.BÀI TẬP

1.1. Hãy đọc một mảng byte từ System.in, sau khi nhập từ bàn phím, xuất lên màn hình mảng này.

1.2. Sử dụng phương thức System.out.write() để xuất ký tự 'X' ra Console. Có nhận xét gì khi dung phương thức write() ?

1.3. Hãy đọc các ký tự từ console sử dụng **BufferedReader**, cho đến khi gặp dấu chấm thì ngừng lại. Xuất chuỗi đó lên màn hình.

1.4. Hãy đọc các chuỗi từ console sử dụng BufferedReader, nhưng đến chữ "stop" thì ngừng lại.

1.5. Hãy tạo một file text sử dụng BufferedReader

1.6. Hãy xuất ra Console dùng luồng ký tự có kiểu thay đổi

1.7. Hiển thị nội dung của một file tên test.txt lưu tại D:\test.txt
1.8. Copy nội dung một file text đến một file text khác

1.9. Dùng DataOutputStream và DataInputStream để ghi và đọc những kiểu dữ liệu khác nhau trên file.
1.10. Ghi 6 số kiểu double xuống file, rồi đọc lên theo thứ tự ngẫu nhiên.
1.11.Đọc những dòng văn bản nhập từ bàn phím và ghi chúng xuống file tên là "test.txt". Việc đọc và ghi kết thúc khi người dùng nhập vào chuỗi "stop".

HƯỚNG DẪN
1.1. Sử dụng phương thức read để đọc nội dung từ *system.in*

```
import java.io.*;
class ReadBytes
{
public static void main(String args[]) throws IOException
{
byte data[] = new byte[100];
System.out.print("Ban hay nhap mot so ky tu  ");
System.in.read(data); // Doc chuoi byte da nhap vao data
System.out.print(" Cac ky tu cua ban da nhap: ");
for(int i=0; i < data.length; i++)
System.out.print((char) data[i]);
 }
}
```

1.3. Nhập Console dùng luồng ký tự

Thường thì việc nhập dữ liệu từ Console dùng luồng ký tự thì thuận lợi hơn dùng luồng byte. Lớp tốt nhất để đọc dữ liệu nhập từ Console là lớp BufferedReader. Tuy nhiên chúng ta không thể xây dựng một lớp BufferedReader trực tiếp từ System.in. Thay vào đó chúng ta phải chuyển nó thành một luồng ký tự. Để làm điều này chúng ta dùng InputStreamReader chuyển bytes thành ký tự.

Để có được một đối tượng InputStreamReader gắn với System.in ta dùng constructor của InputStreamReader *:*

```
InputStreamReader(InputStream inputStream)
```

Tiếp theo dùng đối tượng InputStreamReader đã tạo ra để tạo ra một BufferedReader dùng constructor BufferedReader.
BufferedReader(Reader inputReader)

Ví dụ: Tạo một BufferedReader gắn với Keyboard

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

Sau khi thực hiện câu lệnh trên, br là một luồng ký tự gắn với Console thông qua System.in.

```
import java.io.*;
class ReadChars
{
public static void main(String args[]) throws IOException
{
char c;
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Nhap chuoi ky tu, gioi han dau cham.");
// Doc chuoi ky tu
 do
   {
c = (char)br.read();
      System.out.println(c);
    } while(c != '.');
 }
}
```

1.6. Trong ngôn ngữ java, bên cạnh việc dùng System.out để xuất dữ liệu ra Console (thường dùng để debug chương trình), chúng ta có thể dùng luồng PrintWriter đối với các chương trình "chuyên nghiệp". PrintWriter là một trong những lớp luồng ký tự. Việc dùng các lớp luồng ký tự để xuất dữ liệu ra Console thường được "ưa chuộng" hơn.

Để xuất dữ liệu ra Console dùng PrintWriter cần thiết phải chỉ định System.out cho luồng xuất.

Ví dụ: Tạo đối tượng PrintWriter để xuất dữ liệu ra Console

PrintWriter pw = new PrintWriter(System.out, true);

1.7. Đọc dữ liệu từ file

- Mở một file để đọc dữ liệu

FileInputStream(String fileName) throws FileNotFoundException

Nếu file không tồn tại: thì ném ra FileNotFoundException

- Đọc dữ liệu: dùng phương thức read()

int read( ) throws IOException: đọc từng byte từ file và trả về giá trị của byte đọc được. Trả về -1 khi hết file, và ném ra *IOException* khi có lỗi đọc.

- Đóng file: dùng phương thức close()

void close( ) throws IOException: sau khi làm việc xong cần đóng file để giải phóng tài nguyên hệ thống đã cấp phát cho file.

1.8. Ghi dữ liệu xuống file

- Mở một file để ghi dữ liệu

FileOutputStream(String fileName) throws FileNotFoundException

Nếu file không tạo được: thì ném ra FileNotFoundException

- Ghi dữ liệu xuống: dùng phương thức write()

void write(int byteval) throws IOException: ghi một byte xác định bởi tham số byteval xuống file, và ném ra IOException khi có lỗi ghi.

- Đóng file: dùng phương thức close()

  void close( ) throws IOException: sau khi làm việc xong cần đóng file để giải phóng tài nguyên hệ thống đã cấp phát cho file.

1.9.Đọc và ghi dữ liệu nhị phân

Để đọc và ghi những giá trị nhị phân của các kiểu dữ liệu trong java, chúng ta sử dụng DataInputStream và DataOutputStream.

DataOutputStream: hiện thực interface DataOuput. Interface DataOutput có các phương thức cho phép ghi tất cả những kiểu dữ liệu cơ sở của java đến luồng (theo định dạng nhị phân).

Phương thức	**Ý nghĩa**
void writeBoolean (boolean val)	Ghi xuống luồng một giá trị boolean được xác định bởi val.
void writeByte (int val)	Ghi xuống luồng một byte được xác định bởi val.
void writeChar (int val)	Ghi xuống luồng một Char được xác định bởi val.
void writeDouble (double val)	Ghi xuống luồng một giá trị Double được xác định bởi val.
void writeFloat (float val)	Ghi xuống luồng một giá trị float được xác định bởi val.
void writeInt (int val)	Ghi xuống luồng một giá trị int được xác định bởi val.
void writeLong (long val)	Ghi xuống luồng một giá trị long được xác định bởi val.
void writeShort (int val)	Ghi xuống luồng một giá trị short được xác định bởi val.

Contructor: DataOutputStream(OutputStream outputStream)
OutputStream: là luồng xuất dữ liệu. Để ghi dữ liệu ra file thì đối tượng outputStream có thể là FileOutputStream.

DataInputStream: hiện thực interface DataInput. Interface DataInput có các phương thức cho phép đọc tất cả những kiểu dữ liệu cơ sở của java (theo định dạng nhị phân).

Phương thức	**Ý nghĩa**
boolean readBoolean( )	Đọc một giá trị boolean
Byte readByte( )	Đọc một byte
char readChar( )	Đọc một Char
double readDouble( )	Đọc một giá trị Double

float readFloat( )	Đọc một giá trị float
int readInt( )	Đọc một giá trị int
Long readLong( )	Đọc một giá trị long
short readShort( )	Đọc một giá trị short

Contructor: DataInputStream(InputStream inputStream)
InputStream: là luồng nhập dữ liệu. Để đọ dữ liệu từ file thì đối tượng InputStream có thể là FileInputStream.

1.10. Bên cạnh việc xử lý xuất nhập trên file theo kiểu tuần tự thông qua các luồng, java cũng hỗ trợ truy cập ngẫu nhiên nội dung của một file nào đó dùng RandomAccessFile. RandomAccessFile không dẫn xuất từ InputStream hay OutputStream mà nó hiện thực các interface DataInput, DataOutput (có định nghĩa các phương thức I/O cơ bản). RandomAccessFile hỗ trợ vấn đề định vị con trỏ file bên trong một file dùng phương thức seek(long newPos).

1.12. Đọc/ghi File dùng luồng ký tự

Thông thường để đọc/ghi file người ta thường dùng luồng byte, nhưng đối với luồng ký tự chúng ta cũng có thể thực hiện được. Ưu điểm của việc dùng luồng ký tự là chúng thao tác trực tiếp trên các ký tự Unicode. Vì vậy luồng ký tự là chọn lựa tốt nhất khi cần lưu những văn bản Unicode.
Hai lớp luồng thường dùng cho việc đọc/ghi dữ liệu ký tự xuống file là FileReader và FileWriter.

---o0o---

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 03 Số giờ : 03 giờ

GVHD : ThS.Nguyễn Minh Nhật

LAB 02
LẬP TRINH THREAD & MULTI THREADS

I.MỤC TIÊU

Cũng cố một số kiên thức làm cơ sở cho thiết kế và lập trình phân tán với java, gồm các vấn đề như:

- Các bước xây dựng Thread
- Sử dụng Threat để giải các bài toán quen thuộc

II. NỘI DUNG

A. LÝ THUYẾT

1. Threat và multi thread

- Thread là gì ?

Là một dòng các điều khiển trong một process hay một ứng dụng. Luồng và đa luồng là một kĩ thuât cho phép CPU có thể thực hiện hay hay nhiều công việc đồng thời. Hiệu suất của CPU sẽ được tăng lên đáng kể nếu ta biết sử dụng hết công suất CPU đa luồng là kĩ thuật phổ biến giúp ta làm điều này.

- Multithread :

Với cơ chế multithreading ứng dụng của ta có thể thực thi đồng thời nhiều dòng lệnh cùng lúc. Có nghĩa là ta có thể làm nhiều công việc đồng thời trong cùng một ứng dụng của ta. Có thể hiểu một cách hết sức đơn giản : hệ điều hành với cơ chế đa nhiệm cho phép nhiều ứng dụng chạy cùng lúc. Thì với cơ chế đa luồng, mỗi ứng dụng của ta có thể thực hiện được nhiều công việc đồng thời.

- Đồng bộ hóa Thread

Đồng bộ có thể được sử dụng để kiểm soát các hoạt động của các threat. Để sử dụng đồng bộ hóa, sử dụng từ khóa **Synchronized** trước lớp tạo Thread.
Multithreading xảy ra không đồng bộ, có nghĩa là một thread thực thi độc lập với các thread khác. Theo đó, mỗi thread không phụ thuộc vào sự thực thi của các thread khác. Để bắt buộc, các xử lý chạy đồng bộ hóa phụ thuộc vào các xử lý khác. Đó là một xử lý chờ cho đến khi một xử lý khác kết thúc trước khi nó có thể
thực thi.

Thỉnh thoảng, việc thực thi của một thread có thể phụ thuộc vào việc thực thi của một thread khác. Giả sử bạn có hai thread – một tập hợp các thông tin đăng nhập và một cái khác kiểm tra mật khẩu và ID của người dùng. Thread login phải chờ thread validation

hoàn tất xử lý trước khi nó có thể nói cho người dùng việc đăng nhập có thành công hay không. Vì thế cả hai thread phải được thực thi đồng
bộ, không được không đồng bộ.

Java cho phép các thread đồng bộ hóa được định nghĩa bởi một method đồng bộ hoá. Một thread nằm trong một method đồng bộ hóa ngăn bất kỳ thread nào khác từ một phương thức đồng bộ hoá khác gọi trong cùng một đối tượng.

2. Các xây dựng và sử dụng Thread

Khi chạy ứng dụng trong java thì đã có một thread. Đây là thread chính, nó thực thi các dóng lệnh trong method: *public static void main*.Đây là một điểm nhập bắt buộc cho mọi ứng dụng độc lập.

Để tạo ra một thread khác ngoài thread chính trên, Java cung cấp cho chúng ta hai cách:

- Tạo ra một lớp con của lớp Thread (java.lang.Thread)
- Tạo ra một lớp thực hiện interface Runnable

- **Tạo luồng bằng phương pháp thừa kế (tạo ra lớp con lớp Thread)**
  + Khai báo

```
class A extends Thread {

public void run() {

... // code for the new thread to execute

}

}

...

A a = new A(); // create the thread object

a.start(); // start the new thread executing

...
```

Với cách này các dòng lệnh sẽ được đặt trong method run. Method này được override method nguyên thuỷ của lớp Thread. Sau đó ta sẽ tạo ra một đối tượng từ lớp của ta.Gọi phương thức start từ đối tượng đó. Lúc này thread của ta chính thức được tạo ra và phương thức start sẽ tự gọi method run của tavà thực thi các dòng lệnh mà đã đặc tả.

Chú ý: method start là method của hệ thống, nó có nhiệm vụ cấp phát bộ nhớ, tạo ra một thread và gọi hàm run của ta. Vì thế không nên override phương thức này. Điều này có thể dẫn đến không tạo được thread.

Ví dụ :

```
public class TwoThread extends Thread {
  public void run() {
    for (int i = 0; i < 10; i++) {
      System.out.println("New thread");
    }
  }

  public static void main(String[] args) {
    TwoThread tt = new TwoThread();
    tt.start();

    for (int i = 0; i < 10; i++) {
      System.out.println("Main thread");
    }
  }
}
```

Ví dụ 2

```
class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((int)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + getName());
    }
}

class TwoThreadsTest {
    public static void main (String[] args) {
        new SimpleThread("Jamaica").start();
```

```
        new SimpleThread("Fiji").start();
    }
}

class ThreeThreadsTest {
    public static void main (String[] args) {
        new SimpleThread("Jamaica").start();
        new SimpleThread("Fiji").start();
        new SimpleThread("Bora Bora").start();
    }
}
```

- **Hiện thực interface Runnable**

+ Khai báo

```
class B implements Runnable {

public void run() {

... // code for the new thread to execute

}

}

...

B b = new B(); // create the Runnable object

Thread t = new Thread(b); // create a thread object

t.start(); // start the new thread

...
```

Cũng giống như cách trên, dòng lệnh đặt trong method run (có thể gọi đến các phương thức khác, nhưng phải bắt đầu trong phương thức này). Sau đó tạo một đối tượng B từ lớp đã hiện thực interface Runnable, tạo thêm một đối tượng t của lớp Thread với thông số cho constructor là đối tượng B.

Sau đó khi gọi phương thức t.start() thì chính thức thread được tạo ra và phương thức run sẽ được triệu gọi một cách tự động.

Ví dụ 3 : Tạo Thread bằng giao diện Runnable thread

```
class MyThread implements Runnable {
  int count;
```

```
  MyThread() {
    count = 0;
  }
  public void run() {
    System.out.println("MyThread starting.");
    try {
      do {
        Thread.sleep(500);
        System.out.println("In MyThread, count is " + count);
        count++;
      } while (count < 5);
    } catch (InterruptedException exc) {
      System.out.println("MyThread interrupted.");
    }
    System.out.println("MyThread terminating.");
  }
}

class RunnableDemo {
  public static void main(String args[]) {
    System.out.println("Main thread starting.");
    MyThread mt = new MyThread();
    Thread newThrd = new Thread(mt);
    newThrd.start();
    do {
      System.out.println("In main thread.");
      try {
        Thread.sleep(250);
      } catch (InterruptedException exc) {
        System.out.println("Main thread interrupted.");
      }
    } while (mt.count != 5);

    System.out.println("Main thread ending.");
  }
}
```

Sự khác nhau giữa runable và thread

- Bản thân ngôn ngữ Java không hỗ trợ đa thừa kế. Chúng ta chỉ có thể *extends* từ một lớp duy nhất. Nhưng lại có thể *implements* cùng lúc nhiều interface. Khi mà lớp của ta đã [extends] một lớp nào đó rồi (vd : Applet), thì chỉ có thể *implements* Runnable để tạo ra Thread.

- Việc extends lớp Thread có thể dẫn đến rủi ro là bạn override các method start, stop, ... thì có thể làm cho việc tạo thread là không thể.

Với interface Runnable (cách thứ hai) : Khi muốn tạo ra một Thread. Chương trình sẽ trong sáng và dễ tìm lỗi hơn.

Method	Mô tả
getName()	Trả về tên của thread
getPriority()	Trả về quyền ưu tiên của thread.
isAlive()	Xác định thread nào đang chạy
join()	Tạm dừng cho đến khi thread kết thúc
run()	Danh mục cần thực hiện bên trong thread
sleep()	Suspends một thread. Method này cho phép bạn xác định khoảng thời gian mà thread được cho tạm dừng
start()	Bắt đầu thread.

III.

2.1. Tạo ra 1 thread và cho nó chạy cùng với thread của mang phương thức main(). Có nhận xét gì ?

2.2. Kiểm tra các trạng thái Suspend, resume, and stop của một tiến trình

2.3. Tạo ra 3 Threat có tên `"Ha Noi", "Da Nang" và "Sai Gon"`. Cho các Threat này xuất hiện 10 lần một cách ngẫu nhiên. Mỗi threat khi thực hiện xong thông báo đã thực hoàn thành công việc.

2.4. Tạo ra tiến trình, xuất ra 5 lần, mỗi lần xuất hiện các số từ 5 đến 1 trên màn hình.

2.5. Tạo lớp Demo cho phép đưa ra 4 tên threat cùng một threat bằng cách gọi constructor của lớp MyThread. Mỗi trong số này được coi như là một threat riêng biệt. Sau đó, threat chính tạm dừng 10 giây bằng cách gọi đến phương thức sleep (). Trong thời gian này, các threat tiếp tục thực thi. Khi thread chính "tỉnh lại", nó sẽ hiển thị thông báo rằng các thread chính là chấm dứt.

2.6. Tạo 2 Threat .

ThreatA : Ghi một mảng bytes vào Stream theo cơ chế Pipe

ThreatB : Đọc từ Stream các phần từ ở Stream ra màn hình

2.7.Tạo ra 5 threat, mỗi threat sau khi hoạt động được 5s, thì thông báo dừng, và threat thứ 2 hoạt động tiếp… quá trình trên thực hiện cho đến khi hệ thống đếm đủ 30s thì thu hồi tài nguyên.

2.8.Tạo ra 2 threat hiển thị chuỗi ra màn hinh nhận từ bàn phím. Kiểm tra trường hợp

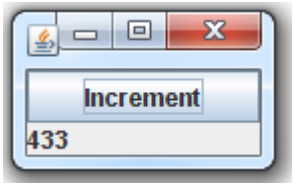
b. Không sử dụng đồng bộ hóa

a. Sử dụng đồng bộ hóa

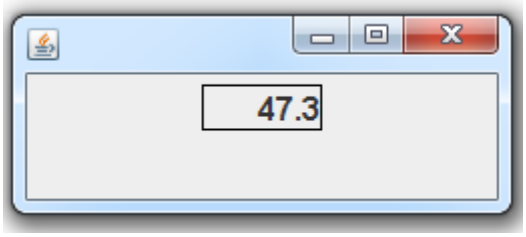
2.9. Tạo 2 threat, threat thứ nhất tạo chuỗi số nguyên ngẫu nhiên, sau đó threat thứ 2 thực hiện sắp xếp dãy số đó theo thuật toán nổi bọt và xuất kết quả ra màn hình.

2.10. Sử dụng đa luồng để đọc một ma trận với N hàng và M cột từ file text và xuất ra file khác với điều kiện là một phần tử trên ma trận này là trung bình cộng của các phần tử xung quanh.

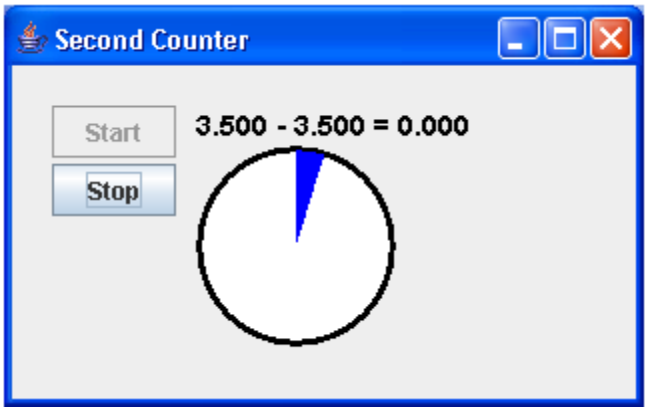
2.11.Sử dụng Thread hiển thị bộ đếm tăng dần ở Swing (Hình vẽ)



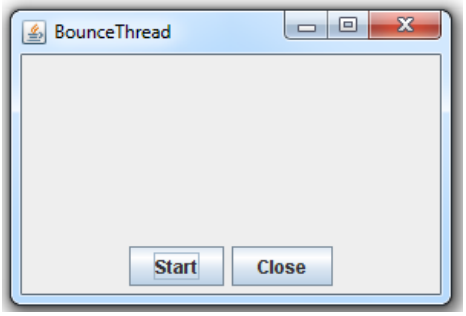
2.12. Sử dụng Thread bộ đếm tăng dần theo thời gian của đông hồ CMOS theo như hình dưới đây :

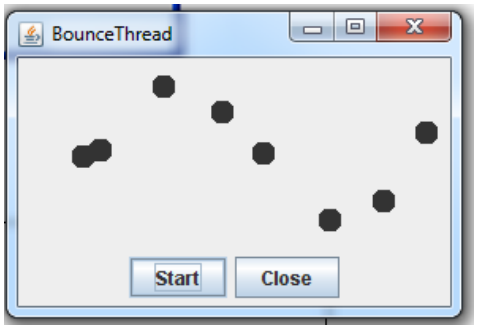


3.13. Sử dụng Thread hiển thị bộ đếm thời gian theo như hình dưới đây :



2.14. Sử dụng Thread để hiển thị sử chuyển động của trái ping pong khi va vào các mép của box và dừng lại tại 1 điểm cho trước (Xem hình vẽ)





2.15*. Sử dụng 3 thread, thread 1: Nhập 1 chuỗi từ file cho trước ở ổ đĩa, Thread 2 : Thực hiện sắp xếp tăng dần nữa dãy số đó theo thuật toán nổi bọt. Thread 3 : Thực hiện sắp xếp tăng dần nữa còn lại cũng theo thuật toán nổi bọt. Chương trình phương thức main() xuất xuất kết quả chuỗi sau khi sắp xếp ra màn hình.

HƯỚNG DẪN

UserTHread

2.1

```
public class TwoThread extends Thread{
      public void run() {
          for (int i = 0; i < 10; i++) {
            System.out.println("New thread");
          }
        }

        public static void main(String[] args) {
          TwoThread tt = new TwoThread();
          tt.start();

          for (int i = 0; i < 10; i++) {
            System.out.println("Main thread");
          }
        }
      }
```

2.2

```
public class TwoThreadAlive extends Thread {
  public void run() {
    for (int i = 0; i < 10; i++) {
      printMsg();
    }
  }

  public void printMsg() {
    Thread t = Thread.currentThread();
    String name = t.getName();
    System.out.println("name=" + name);
  }

  public static void main(String[] args) {
    TwoThreadAlive tt = new TwoThreadAlive();
    tt.setName("Thread");

    System.out.println("before start(), tt.isAlive()=" + tt.isAlive());
    tt.start();
    System.out.println("just after start(), tt.isAlive()=" + tt.isAlive());

    for (int i = 0; i < 10; i++) {
```

```
      tt.printMsg();
    }

    System.out.println("The end of main(), tt.isAlive()=" + tt.isAlive());
  }
}
```

2.3.

```
public class ThreeThreadsTest {

  public static void main(String[] args) {
    new SimpleThread("Ha Noi").start();
    new SimpleThread("Da Nang").start();
    new SimpleThread("Sai Gon").start();
  }
}

class SimpleThread extends Thread {
  public SimpleThread(String str) {
    super(str);
  }

  public void run() {
    for (int i = 1; i <= 10; i++) {
      System.out.println(i + " " + getName());
      try {
        sleep((long) (Math.random() * 1000));
      } catch (InterruptedException e) {
      }
    }
    System.out.println("Hoan thanh tien trinh! " + getName());
  }
}
```

2.4.

```
public class SleepingThread extends Thread {

  private int countDown = 5;

  private static int threadCount = 0;

  public SleepingThread() {
    super("" + ++threadCount);
    start();
  }

  public String toString() {
    return "#" + getName() + ": " + countDown;
  }
```

```
  public void run() {
    while (true) {
      System.out.println(this);
      if (--countDown == 0)
        return;
      try {
        sleep(1000);
      } catch (InterruptedException e) {
        throw new RuntimeException(e);
      }
    }
  }

  public static void main(String[] args) throws InterruptedException {
    for (int i = 0; i < 5; i++)
      new SleepingThread().join();
  }
}
```

2.5.

```
class Demo {
   public static void main (String args []) {
      new MyThread ("1");
      new MyThread ("2");
      new MyThread ("3");
      new MyThread ("4");
      try {
         Thread.sleep (10000);
      } catch (InterruptedException e) {
        System.out.println("Exception: Thread maininterrupted.");
      }
      System.out.println("Terminating thread: main thread.");
  }
}

class MyThread implements Runnable {
  String tName;
  Thread t;
  MyThread (String threadName) {
     tName = threadName;
     t = new Thread (this, tName);
     t.start();
  }
  public void run() {
     try {
        System.out.println("Thread: " + tName );
        Thread.sleep(2000);
     } catch (InterruptedException e ) {
       System.out.println("Exception: Thread "+ tName + " interrupted");
     }
```

```
    System.out.println("Terminating thread: " + tName );
  }
}
```

2.6

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PipedInputStream;
import java.io.PipedOutputStream;

public class PipedBytes extends Object {
  public static void writeStuff(OutputStream rawOut) {
    try {
      DataOutputStream out = new DataOutputStream(
          new BufferedOutputStream(rawOut));

      int[] data = { 82, 105, 99, 104, 97, 114, 100, 32, 72, 121, 100,
          101 };

      for (int i = 0; i < data.length; i++) {
        out.writeInt(data[i]);
      }
      out.flush();
      out.close();
    } catch (IOException x) {
      x.printStackTrace();
    }
  }

  public static void readStuff(InputStream rawIn) {
    try {
      DataInputStream in = new DataInputStream(new
BufferedInputStream(rawIn));
      boolean eof = false;
      while (!eof) {
        try {
          int i = in.readInt();
          System.out.println("Vua doc xong: " + i);
        } catch (EOFException eofx) {
          eof = true;
        }
      }

      System.out.println("Doc tat ca du lieu tu Pipe");
    } catch (IOException x) {
      x.printStackTrace();
    }
  }

public static void main(String[] args) {
```

```
    try {
      final PipedOutputStream out = new PipedOutputStream();

      final PipedInputStream in = new PipedInputStream(out);

      Runnable runA = new Runnable() {
        public void run() {
          writeStuff(out);
        }
      };

      Thread threadA = new Thread(runA, "threadA");
      threadA.start();

      Runnable runB = new Runnable() {
        public void run() {
          readStuff(in);
        }
      };

      Thread threadB = new Thread(runB, "threadB");
      threadB.start();
    } catch (IOException x) {
      x.printStackTrace();
    }
  }
}
```

2.8.

a.

```
class Parentheses {
   void display(String s) {
   System.out.print ("(" + s);
   try {
      Thread.sleep (1000);
   } catch (InterruptedException e) {
        System.out.println ("Interrupted");
   }
   System.out.println(")");
  }
}
class MyThread implements Runnable {
   String s1;
   Parentheses p1;
   Thread t;
   public MyThread (Parentheses p2, String s2) {
      p1= p2;
      s1= s2;
      t = new Thread(this);
      t.start();
   }
   public void run() {
     p1.display(s1);
   }
}
    public static void main (String args[]) {
     Parentheses p3 = new Parentheses();
```

```
      MyThread name1 = new MyThread(p3, "Bob");
      MyThread name2 = new MyThread(p3, "Mary");
      try {
         name1.t.join();
         name2.t.join();
      } catch (InterruptedException e ) {
           System.out.println( "Interrupted");
      }
   }
}
```

b.

```
class Parentheses {
   synchronized void display(String s) {
   System.out.print ("(" + s);
   try {
      Thread.sleep (1000);
   } catch (InterruptedException e) {
        System.out.println ("Interrupted");
   }
   System.out.println(")");
  }
}
```

2.11.

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities;

/**
 * @author NMN
 */
public class SwingThreading extends JFrame implements ActionListener {
  private JLabel counter;

  private int tickCounter = 0;

  private static SwingThreading edt;

  public SwingThreading() {
    super("Swing Threading");

    JButton freezer = new JButton("Increment");
    freezer.addActionListener(this);

    counter = new JLabel("0");
```

```
    add(freezer, BorderLayout.CENTER);
    add(counter, BorderLayout.SOUTH);

    pack();
    setLocationRelativeTo(null);

    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }

  public void actionPerformed(ActionEvent e) {
    incrementLabel();
  }

  private void incrementLabel() {
    tickCounter++;
    Runnable code = new Runnable() {
      public void run() {
        counter.setText(String.valueOf(tickCounter));
      }
    };

    if (SwingUtilities.isEventDispatchThread()) {
      code.run();
    } else {
      SwingUtilities.invokeLater(code);
    }
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        edt = new SwingThreading();
        edt.setVisible(true);

        new Thread(new Runnable() {
          public void run() {
            while (true) {
              try {
                Thread.sleep(300);
              } catch (InterruptedException e) {
              }
              edt.incrementLabel();
            }
          }
        }).start();
      }
    });
  }
}
```

2.12.

```
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.lang.reflect.InvocationTargetException;
import java.text.DecimalFormat;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;

public class DigitalTimer extends JLabel {
  private volatile String timeText;

  private Thread internalThread;

  private volatile boolean noStopRequested;

  public DigitalTimer() {
    setBorder(BorderFactory.createLineBorder(Color.black));
    setHorizontalAlignment(SwingConstants.RIGHT);
    setFont(new Font("SansSerif", Font.BOLD, 16));
    setText("00000.0"); // use to size component
    setMinimumSize(getPreferredSize());
    setPreferredSize(getPreferredSize());
    setSize(getPreferredSize());

    timeText = "0.0";
    setText(timeText);

    noStopRequested = true;
    Runnable r = new Runnable() {
      public void run() {
        try {
          runWork();
        } catch (Exception x) {
          x.printStackTrace();
        }
      }
    };

    internalThread = new Thread(r, "DigitalTimer");
    internalThread.start();
  }

  private void runWork() {
```

```
    long startTime = System.currentTimeMillis();
    int tenths = 0;
    long normalSleepTime = 100;
    long nextSleepTime = 100;
    DecimalFormat fmt = new DecimalFormat("0.0");

    Runnable updateText = new Runnable() {
      public void run() {
        setText(timeText);
      }
    };

    while (noStopRequested) {
      try {
        Thread.sleep(nextSleepTime);

        tenths++;
        long currTime = System.currentTimeMillis();
        long elapsedTime = currTime - startTime;

        nextSleepTime = normalSleepTime
            + ((tenths * 100) - elapsedTime);

        if (nextSleepTime < 0) {
          nextSleepTime = 0;
        }

        timeText = fmt.format(elapsedTime / 1000.0);
        SwingUtilities.invokeAndWait(updateText);
      } catch (InterruptedException ix) {
        // stop running
        return;
      } catch (InvocationTargetException x) {
        x.printStackTrace();
      }
    }
  }

  public void stopRequest() {
    noStopRequested = false;
    internalThread.interrupt();
  }

  public boolean isAlive() {
    return internalThread.isAlive();
  }

  public static void main(String[] args) {

    JFrame f = new JFrame();
```

```
    f.getContentPane().setLayout(new FlowLayout());
    f.getContentPane().add(new DigitalTimer());
    f.setSize(250, 100);
    f.setVisible(true);
  }
}
```

2.13.

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.text.DecimalFormat;

import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

public class SecondCounterMain extends JPanel {
  private SecondCounter sc = new SecondCounter();

  private JButton startB = new JButton("Start");

  private JButton stopB = new JButton("Stop");

  public SecondCounterMain() {
    stopB.setEnabled(false);
    startB.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        startB.setEnabled(false);

        Thread counterThread = new Thread(sc, "Counter");
        counterThread.start();

        stopB.setEnabled(true);
        stopB.requestFocus();
      }
    });

    stopB.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        stopB.setEnabled(false);
```

```
      sc.stopClock();
      startB.setEnabled(true);
      startB.requestFocus();
    }
  });

  JPanel innerButtonP = new JPanel();
  innerButtonP.setLayout(new GridLayout(0, 1, 0, 3));
  innerButtonP.add(startB);
  innerButtonP.add(stopB);

  JPanel buttonP = new JPanel();
  buttonP.setLayout(new BorderLayout());
  buttonP.add(innerButtonP, BorderLayout.NORTH);

  this.setLayout(new BorderLayout(10, 10));
  this.setBorder(new EmptyBorder(20, 20, 20, 20));
  this.add(buttonP, BorderLayout.WEST);
  this.add(sc, BorderLayout.CENTER);
}

public static void main(String[] args) {
  SecondCounterMain scm = new SecondCounterMain();

  JFrame f = new JFrame("Second Counter");
  f.setContentPane(scm);
  f.setSize(320, 200);
  f.setVisible(true);
  f.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      System.exit(0);
    }
  });
}

class SecondCounter extends JComponent implements Runnable {
  private volatile boolean keepRunning;

  private Font paintFont = new Font("SansSerif", Font.BOLD, 14);

  private volatile String timeMsg = "never started";

  private volatile int arcLen = 0;

  public SecondCounter() {
  }

  public void run() {
    runClock();
  }
```

```
    public void runClock() {
      DecimalFormat fmt = new DecimalFormat("0.000");
      long normalSleepTime = 100;
      long nextSleepTime = normalSleepTime;

      int counter = 0;
      long startTime = System.currentTimeMillis();
      keepRunning = true;

      while (keepRunning) {
        try {
          Thread.sleep(nextSleepTime);
        } catch (InterruptedException x) {
          // Bỏ qua
        }

        counter++;
        double counterSecs = counter / 10.0;
        double elapsedSecs = (System.currentTimeMillis() - startTime) /
1000.0;

        double diffSecs = counterSecs - elapsedSecs;

        nextSleepTime = normalSleepTime + ((long) (diffSecs * 1000.0));

        if (nextSleepTime < 0) {
          nextSleepTime = 0;
        }

        timeMsg = fmt.format(counterSecs) + " - "
            + fmt.format(elapsedSecs) + " = "
            + fmt.format(diffSecs);

        arcLen = (((int) counterSecs) % 60) * 360 / 60;
        repaint();
      }
    }

    public void stopClock() {
      keepRunning = false;
    }

    public void paint(Graphics g) {
      g.setColor(Color.black);
      g.setFont(paintFont);
      g.drawString(timeMsg, 0, 15);

      g.fillOval(0, 20, 100, 100); // Vẽ viền màu đen
```

```
      g.setColor(Color.white);
      g.fillOval(3, 23, 94, 94); // Màu trắng cho phần không sử dụng

      g.setColor(Color.blue); // Màu xanh cho phần sử dụng
      g.fillArc(2, 22, 96, 96, 90, -arcLen);
    }
  }
}
```

2.14.

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;

/**
   Hiển thị quả bóng chuyển động .
*/
public class BounceThread
{
   public static void main(String[] args)
   {
      JFrame frame = new BounceFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

/**
   Tạo lớp runnable để quả bóng chuyển động.
*/
class BallRunnable implements Runnable
{
   /**
      Constructs the runnable.
      aBall : quả bóng tung lên
      aPanel : Thành phần mà quả bóng khi tung lên
   */
   public BallRunnable(Ball aBall, Component aComponent)
   {
      ball = aBall;
      component = aComponent;
   }

   public void run()
   {
      try
      {
         for (int i = 1; i <= STEPS; i++)
         {
```

```
            ball.move(component.getBounds());
            component.repaint();
            Thread.sleep(DELAY);
         }
      }
      catch (InterruptedException e)
      {
      }
   }

   private Ball ball;
   private Component component;
   public static final int STEPS = 1000;
   public static final int DELAY = 5;
}

/**
   Một quả bóng di chuyển và bị trả lại khi chạm vào các cạnh của một
   hình chữ nhật
*/
class Ball
{
   /**
      Di chuyển bóng đến vị trí tiếp theo, đảo chiều
      nếu nó chạm một trong các cạnh
   */
   public void move(Rectangle2D bounds)
   {
      x += dx;
      y += dy;
      if (x < bounds.getMinX())
      {
         x = bounds.getMinX();
         dx = -dx;
      }
      if (x + XSIZE >= bounds.getMaxX())
      {
         x = bounds.getMaxX() - XSIZE;
         dx = -dx;
      }
      if (y < bounds.getMinY())
      {
         y = bounds.getMinY();
         dy = -dy;
      }
      if (y + YSIZE >= bounds.getMaxY())
      {
         y = bounds.getMaxY() - YSIZE;
         dy = -dy;
      }
```

```
   }

   /**
      Bắt hình dạng của quả bóng ở vị trí hiện tại.
   */
   public Ellipse2D getShape()
   {
      return new Ellipse2D.Double(x, y, XSIZE, YSIZE);
   }

   private static final int XSIZE = 15;
   private static final int YSIZE = 15;
   private double x = 0;
   private double y = 0;
   private double dx = 1;
   private double dy = 1;
}

/**
   vẽ panel các quả bóng .
*/
class BallPanel extends JPanel
{
   /**
      Thêm một quả bóng vào panel.
      b : thêm bóng
   */
   public void add(Ball b)
   {
      balls.add(b);
   }

   public void paintComponent(Graphics g)
   {
      super.paintComponent(g);
      Graphics2D g2 = (Graphics2D) g;
      for (Ball b : balls)
      {
         g2.fill(b.getShape());
      }
   }

   private ArrayList<Ball> balls = new ArrayList<Ball>();
}

/**
   Khung với bảng điều khiển và các nút.
*/
class BounceFrame extends JFrame
{
```

```
/**
   Xây dựng khung với bảng điều khiển cho hiển thị các
    bóng nảy và Start và nút Close
*/
public BounceFrame()
{
   setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
   setTitle("BounceThread");

   panel = new BallPanel();
   add(panel, BorderLayout.CENTER);
   JPanel buttonPanel = new JPanel();
   addButton(buttonPanel, "Start",
      new ActionListener()
      {
         public void actionPerformed(ActionEvent event)
         {
            addBall();
         }
      });

   addButton(buttonPanel, "Close",
      new ActionListener()
      {
         public void actionPerformed(ActionEvent event)
         {
            System.exit(0);
         }
      });
   add(buttonPanel, BorderLayout.SOUTH);
}

/**

   Thêm một nút vào một container.
    c:container
    title: tiêu đề nút
    listener:hành động cho nút
*/
public void addButton(Container c, String title, ActionListener listener)
{
   JButton button = new JButton(title);
   c.add(button);
   button.addActionListener(listener);
}

/**
   Thêm một quả bóng nảy vào khung và bắt đầu một chủ đề
    để làm cho nó bị trả
```

```
   */
   public void addBall()
   {
      Ball b = new Ball();
      panel.add(b);
      Runnable r = new BallRunnable(b, panel);
      Thread t = new Thread(r);
      t.start();
   }

   private BallPanel panel;
   public static final int DEFAULT_WIDTH = 450;
   public static final int DEFAULT_HEIGHT = 350;
   public static final int STEPS = 1000;
   public static final int DELAY = 3;
}
```

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 02 Số giờ : 06 (2 buổi)

GVHD: ThS.Nguyễn Minh Nhật

LAB 03

LẬP TRÌNH SOCKET, MULTI SOCKET

I.MỤC TIÊU

Thực hiện trao đổi thông điệp giữa các máy tính qua Socket.

II. NỘI DUNG

A.LÝ THUYẾT

1.Định nghĩa

Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (client) và một chương trình cung cấp dịch vụ (server) trên mạng LAN, WAN hay Internet và đôi lúc là giữa những quá trình ngay bên trong máy tính. Mỗi socket có thể được xem như một điểm cuối trong một kết nối. Một socket trên máy yêu cầu dịch vụ có địa chỉ mạng được cấp sẵn để "gọi" một socket trên máy cung cấp dịch vụ. Một khi socket đã được thiết lập phù hợp, hai máy tính có thể trao đổi dịch vụ và dữ liệu.

Những máy tính có Socket server đảm bảo tính trạng mở của cổng truyền thông, sẵn sàng để nhận bất kỳ cuộc gọi đến nào dù không định trước. Những máy yêu cầu dịch vụ thường xác định số hiệu cổng của server mong muốn bằng cách tìm nó trong cơ sở dữ liệu về Domain Name System

Với sự phát triển của Web, socket vẫn tiếp tục đóng vai trò quan trọng trong việc duy trì các luồng truyền thông trên Internet. Các ứng dụng có liên quan đến Internet đều viết ở lớp bên trên socket, ví dụ socket tích hợp một số phần của địa chỉ Website, trình duyệt web và công nghệ bảo mật Scure Socket Layer.

Trong lập trình Web hiện nay, Socket thực sự không cần thiết đối dù dùng Java, serlet, hay CGI,PHP,…vì không bao giờ mở được cổng một cách tường minh. Các socket vẫn tồn tại để kết nối người dùng với ứng dụng Web, nhưng các chi tiết của socket được ẩn trong những lớp sâu hơn để mọi người không phải động chạm đến.

Các lập trình viên có thể tránh được những khó khăn của việc tạo socket nhờ thư viện lớp các thế hệ mới, chẳng hạn Microsoft Foundation Clas Csocket và CsocketFile. Lập trình viên Uinix có thể dùng Socket++ v.v…

Trong java, hỗ trợ một sẵn lóp về socket là: Java.net.Socket, nó được dùng rộng rãi trong việc tạo ra các socket phía yêu cầu dịch vụ độc lập hệ thống, trong khi

java.net.ServerSocket có thể xây dựng một socket sẵn sàng cho việc nhận các yêu cầu từ máy yêu cầu dịch vụ. Với những công cụ này, các nhà phát triển có thể nhanh chóng tạo ra các socket mà không cần phải "sa lầy " trong các chi tiết lập trình.

Một số lớp cần thiết trong gói thư viện java.net:

- InetAddress : quản lý địa chỉ Internet
- Socket : tạo kết nối từ client đến server
- ServerSocket : tạo kết nối từ phía server đến client
- DatagramSocket: gửi nhận dữ liệu dưới dạng gói tin
- DatagramPackage: gói tin chứa dữ liệu gửi nhận sử dụng cho lớp DatagramSocket
- URL : địa chỉ định vị tài nguyên trên mạng

Lớp InetAddress

Các phương thức thường được sử dụng:

Sử dụng để quản lý địa chỉ host theo tên hay số

Phương thức	Ý nghĩa
static InetAddress getLocalHost()	trả về đối tượng InetAddress là địa chỉ của máy cục bộ(localhost)
static InetAddress getByName(String hostName)	trả về đối tượng InetAddress là địa chỉ của máy có tên là hostName.
byte[] getAddress()	trả về địa chỉ IP của đối tượng InetAddress dưới dạng chuỗi byte
string getHostAddress()	trả về địa chỉ IP của đối tượng InetAddress dưới dạng string
static InetAddress getLocalHost()	trả về đối tượng InetAddress là địa chỉ của máy cục bộ

2. Xây dựng chương trình Socket trong java

Một chương trình Socket bằng java được thực hiện như sau:

a/ **Server:**

- Lắng nghe và chập nhận kết nối từ cổng 9999.
- Cho phép nhiều client kết nối đến cùng một lúc.
- Khi client gửi đến 1 chuỗi thì:

+ Nếu chuỗi là "quit" thì ngắt kết nối với client.
+ Tiến hành đảo chuỗi.
+ Gửi chuỗi đã được đảo cho client.

b/ **Client:**

- Kết nối tới Server qua cổng 9999.
- Nhập chuỗi từ bàn phím.
- Gửi chuỗi tới server.
- Hiển thị chuỗi từ server gửi tới.

Để có thể hiểu sâu hơn về lập trình Socket, học viên sẽ thực hiện theo từng các bài tập cơ bản dưới đây.

B.BÀI TẬP

3.1. Hãy tạo đối tượng Socket dùng để truy cập vào port 80, địa chỉ mạng http//www.dtu.edu.vn.
3.2. Hãy hiển thị các thành phần của một socket, với socket tạo ở bài 3.1
3.3. Tìm kiếm các Server hoặc dịch vụ đang hoạt động trên các port từ 0 →1024 trên máy người dùng.
3.4. Thực hiện *ping* từ một máy đến một máy chủ, đưa ra câu "hello", nếu thành công thì trả "đang hoạt động ", ngược lại "tắt kết nối".
3.5. Thực hiện đọc dữ liệu là giá trị số từ một máy chủ
3.6. Hãy hiển thị nội được Client đọc vào Socket khi kết nối Cient và Server được thiết lập.
3.7. Sử dụng Socket để đọc và ghi vào Stream
3.8. Thực hiện ghi dòng chữ "I love you" vào Socket
3.9. Tạo kết nối Socket và package hiện thời
3.10. Thực hiện download trang Website từ Webserver
3.11. Thực hiện nhận mail từ Socket
3.12. Đọc một đối tượng từ Socket và lưu chúng
+ Đầu tiên, kết nối được chấp nhận, thực hiện đọc một số giá trị
+ Cho đến khi luồng dữ liệu được đóng lại. Một khi nó được đóng lại
+ Ghi các dữ liệu vào một tập tin trên đĩa.
+ Chương trình này luôn luôn ghi vào vào / tmp / Object.

3.13. Thực hiện băm dữ liệu lấy từ Socket với tiện ích **hash** ở gói java. util.zip.*.*
3.14. Server mở cổng 1234, chờ kết nối từ Client. Nếu Client kết nối thì Server sẽ gởi lời chào đến Client. Viết chương trình hiển thị lời chào này trên cả 2 màn hình Client và Server.
3.15. Client gởi cho Server một chuỗi nhập từ bàn phím, yêu cầu Server thực hiện đảo chuỗi và gởi kết quả trở về Client. Hiển thị kết quả trên màn hình Client.

***3.16.** Sử dụng socket để viết chương trình chat trên hai máy client và server. Khi ta chạy chương trình server thì máy server sẽ khởi tạo một socket và socket này ở trạng thái chờ kết nối (listen). Khi chương trình phía Client chạy sẽ kết nối tới socket mà Server đã tạo, nếu Server đồng ý thì Client và Server sẽ trao đổi thông tin được với nhau.
***3,17.** Viết chương trình giao tiếp qua Socket như sau:
- Server mở cổng 5000
Client gởi chuỗi bất kỳ cho Server (Ví dụ chuỗi S = "" đại học duy tân " . Sau khi nhận được chuỗi này, Server thực hiện:

HƯỚNG DẪN

3.1

//Tao mot socket voi cong va dia chi mac dinh

```
import java.net.InetAddress;
import java.net.Socket;

public class Main {

public static void main(String[] argv) throws Exception {
    InetAddress addr = InetAddress.getByName("www.dtu.edu.vn");
    int port = 80;
    Socket socket = new Socket(addr, port);
  }
}
```

3.2

//Hiển thị Sockets

```
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
class Whois {
  public static void main(String args[]) throws Exception {
    int c;
    Socket s = new Socket("internic.net", 43);
    InputStream in = s.getInputStream();
    OutputStream out = s.getOutputStream();
    String str = "asdfasdfasdf\n";
    byte buf[] = str.getBytes();
    out.write(buf);
    while ((c = in.read()) != -1) {
      System.out.print((char) c);
    }
    s.close();
  }
}
```

3.3.

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

public class lookForPorts {
public static void main(String[] args) {

    Socket theSocket;
    String host = "localhost";

    if (args.length > 0) {
      host = args[0];
    }
    for (int i = 0; i < 1024; i++) {
      try {
        System.out.println("Looking for "+ i);
```

```
        theSocket = new Socket(host, i);
        System.out.println("Day la mot server tren port " + i + " cua " +
host);
      }
      catch (UnknownHostException e) {
        System.err.println(" Khong ton tai server nao !");
        break;
      }
      catch (IOException e) {
      System.err.println(" Khong ket noi duoc!");

      }
    }
  }
}
```

3.4.

//Ping từ một server

```
import java.io.DataInputStream;
import java.io.PrintStream;
import java.net.Socket;

public class PingServer1 {
  public static void main(String[] argv) throws Exception {
    Socket t = new Socket("127.0.0.1", 7);
    DataInputStream dis = new DataInputStream(t.getInputStream());
    PrintStream ps = new PrintStream(t.getOutputStream());
    ps.println("Hello");
    String str = dis.readUTF();
    if (str.equals("Hello"))
      System.out.println("Alive!");
    else
      System.out.println("Dead");
    t.close();
  }
}
```

2.5.

//Doc tu server

```
import java.io.DataInputStream;
import java.io.InputStream;
import java.net.Socket;

class SocketDemo {
  public static void main(String args[]) throws Exception {
    String server = args[0];
    int port = Integer.parseInt(args[1]);
    Socket s = new Socket(server, port);

    InputStream is = s.getInputStream();
    DataInputStream dis = new DataInputStream(is);
    System.out.println(dis.readInt());

    s.close();
  }
```

```
}
```

2.6.

//Đang đọc nội dung từ một Socket

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

public class ReadSocket{
  public static void main(String[] argv) throws Exception {
    int port = 2000;
    ServerSocket srv = new ServerSocket(port);
```

// Chờ đợi để kết nối từ Client

```
    Socket socket = srv.accept();
      BufferedReader rd = new BufferedReader(new
      InputStreamReader(socket.getInputStream()));
    String str;
    while ((str = rd.readLine()) != null) {
      System.out.println(str);
    }
    rd.close();
  }
}
```

3.7.

//Sử dụng Socket để đọc và ghi vào stream

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

class SquareClient {
  private final static int BUFSIZE = 20;

  public static void main(String args[]) throws Exception {
    String server = args[0];
    int port = Integer.parseInt(args[1]);
    double value = Double.valueOf(args[2]).doubleValue();

    Socket s = new Socket(server, port);
    OutputStream os = s.getOutputStream();
    DataOutputStream dos = new DataOutputStream(os);
    dos.writeDouble(value);

    InputStream is = s.getInputStream();
    DataInputStream dis = new DataInputStream(is);
    value = dis.readDouble();

    System.out.println(value);
    s.close();
  }
}
```

3.8.

//Viết văn bản vào một Socket

```
import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class WritetoSocket {
  public static void main(String[] argv) throws Exception {
    int port = 2000;
    ServerSocket srv = new ServerSocket(port);

    // Cho ket noi tu Client
    Socket socket = srv.accept();
    BufferedWriter wr = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
    wr.write("I love you ");
    wr.flush();
  }
}
```

3.9.

//Socket connection and concurrent package

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class ExecutorHttpd {
  ExecutorService executor = Executors.newFixedThreadPool(3);

  public void start(int port) throws IOException {
    final ServerSocket ss = new ServerSocket(port);
    while (!executor.isShutdown())
      executor.submit(new TinyHttpdConnection(ss.accept()));
  }

  public void shutdown() throws InterruptedException {
    executor.shutdown();
    executor.awaitTermination(30, TimeUnit.SECONDS);
    executor.shutdownNow();
  }

  public static void main(String argv[]) throws Exception {
    new ExecutorHttpd().start(Integer.parseInt(argv[0]));
  }
}
```

```
class TinyHttpdConnection implements Runnable {
  Socket client;
  TinyHttpdConnection(Socket client) throws SocketException {
    this.client = client;
  }

  public void run() {
    try {
      BufferedReader in = new BufferedReader(new
InputStreamReader(client.getInputStream(),
          "8859_1"));
      OutputStream out = client.getOutputStream();
      PrintWriter pout = new PrintWriter(new OutputStreamWriter(out, "8859_1"),
true);
      String request = in.readLine();
      System.out.println("Request: " + request);

      byte[] data = "hello".getBytes();
      out.write(data, 0, data.length);
      out.flush();
      client.close();
    } catch (IOException e) {
      System.out.println("I/O error " + e);
    }
  }
}
```

3.10.

```
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.Socket;
import java.net.URL;
import java.net.UnknownHostException;

class Main{
  public String downloadWWWPage(URL pageURL) throws Exception {
    String host, file;
    host = pageURL.getHost();
    file = pageURL.getFile();

    InputStream pageStream = getWWWPageStream(host, file);
    if (pageStream == null) {
      return "";
    }
    DataInputStream in = new DataInputStream(pageStream);
    StringBuffer pageBuffer = new StringBuffer();
    String line;

    while ((line = in.readUTF()) != null) {
      pageBuffer.append(line);
    }
```

```
    in.close();
    return pageBuffer.toString();
  }

  public InputStream getWWWPageStream(String host, String file) throws
IOException,
      UnknownHostException {

    InetAddress webServer = InetAddress.getByName(host);

    Socket httpPipe = new Socket(webServer, 80);
    if (httpPipe == null) {
      System.out.println("Socket đến Web server tạo ra bị lỗi !");
      return null;
    }
    InputStream inn = httpPipe.getInputStream(); // get raw streams
    OutputStream outt = httpPipe.getOutputStream();
    DataInputStream in = new DataInputStream(inn); // turn into higher-level
ones
    PrintStream out = new PrintStream(outt);
    if (inn == null || outt == null) {
      System.out.println("Lỗi để mở streams đến socket.");
      return null;
    }
    out.println("GET " + file + " HTTP/1.0\n");

    String response;
    while ((response = in.readUTF()).length() > 0) {
      System.out.println(response);
    }
    return in;
  }
}
```

3.11.

//Nhận email bằng Socket

```
 import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;

public class POP3Demo {
  public static void main(String[] args) throws Exception {
    int POP3Port = 110;
    Socket client = new Socket("127.0.0.1", POP3Port);
    InputStream is = client.getInputStream();
    BufferedReader sockin = new BufferedReader(new InputStreamReader(is));
    OutputStream os = client.getOutputStream();
    PrintWriter sockout = new PrintWriter(os, true);
    String cmd = "user Smith";
    sockout.println(cmd);
    String reply = sockin.readLine();
    cmd = "pass ";
```

```
    sockout.println(cmd + "popPassword");
    reply = sockin.readLine();
    cmd = "stat";
    sockout.println(cmd);
    reply = sockin.readLine();
    if (reply == null)
      return;
    cmd = "retr 1";
    sockout.println(cmd);
    if (cmd.toLowerCase().startsWith("retr") && reply.charAt(0) == '+')
      do {
        reply = sockin.readLine();
        System.out.println("S:" + reply);
        if (reply != null && reply.length() > 0)
          if (reply.charAt(0) == '.')
            break;
      } while (true);
    cmd = "quit";
    sockout.println(cmd);
    client.close();
  }
}
```

3.12.

```
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

/**
 */
class WriteObjectsFromSocket {
  public static void main(String args[]) {
    Socket rs;
    ServerSocket s;
    int PortNumber = 2000; // Fixed port
    boolean acceptLoop = true;
    ReaderWriter readerwriter = null;
    System.out.println("Writ eFile From Socket Running");
    // ////////////////////////////////////////////////////////////
    // Mo mot Socket tren Server nay va lang nghe ket noi
    // Tim kiem dữ lieu neu chung ta goi den ơ day ma chung duoc yeu cau phan
hoi
    // ////////////////////////////////////////////////////////////
    try {
      System.out.println("Cho doi de chap nhan ket noi");
      s = new ServerSocket(PortNumber);
    } catch (Exception e) {
      System.out.println("Khong the mo cong " + PortNumber);
      return;
    }
    while (acceptLoop) {
      try {
        rs = s.accept(); // cho ket noi
      } catch (Exception e) {
```

```
        System.out.println("Cong truy cap loi" + PortNumber);
        return;
      }
      readerwriter = new ReaderWriter(rs); // Bat dau  thread de doc dong vao
    }
  }
}

/**
 * Quan ly bat dau viec ghi va doc file tu mot Client
 */
class ReaderWriter extends Thread {
  byte b[] = null;

  int inputByte = 0;

  String directory = null;

  String filename = null;

  DataInputStream is;

  FileOutputStream localFile = null;

  boolean writeloop = true;

  ReaderWriter(Socket rs) {
    // ////////////////////////////////////////////
    // Open a remote stream to the client
    // ////////////////////////////////////////////
    try {
      is = new DataInputStream(rs.getInputStream());
    } catch (Exception e) {
      System.out.println("Unable to get Stream");
      return;
    }
    // ////////////////////////////////////////////
    // Open the file that is to be written to
    // ////////////////////////////////////////////
    try {
      File theFile = new File("/tmp", "Objects");
      System.out.println("File co the duoc tao hoac viet de: " + theFile);
      localFile = new FileOutputStream(theFile);
    } catch (IOException e) {
      System.out.println("Mo bi loi " + e);
      return;
    }
    // ////////////////////////////////////////////
    // Look for all the double data constantly
    // ////////////////////////////////////////////
    while (writeloop) {
      // Look for data if we get here that is requests
      try {
        inputByte = is.readByte(); // Not the best way to do it
        // Consider looking for available bytes
        // and reading that amount.
      } catch (IOException e) {
```

```
        System.out.println("In the read loop:" + e);
        writeloop = false;
      }
      // ////////////////////////
      // We did get something
      // Write The Data
      // ////////////////////////
      try {
        localFile.write(inputByte);
      } catch (IOException e) {
        System.out.println("Viet bi loi");
        writeloop = false;
      }
    }
    // ////////////////////////////////
    // Close the connection and file
    // ////////////////////////////////
    try {
      rs.close();
      is.close();
      localFile.close();
    } catch (Exception e) {
      System.out.println("Khong the dong ");
    }
    System.out.println("Thread dang thoat ");
  }
}
```

3.14.

1/ Lập trình phía máy chủ

```
class Server {
   public static void main(String args[]) {
      String data = "Alo Xinh xin chao cac ban tu Da Nang!";
      try {
         ServerSocket srvr = new ServerSocket(1234);
         Socket skt = srvr.accept();
         System.out.print("May chu da duoc ket noi!\n");
         PrintWriter out = new PrintWriter(skt.getOutputStream(), true);
         System.out.print("Du lieu se gui di cho may tram: '" + data + "'\n");
         out.print(data);
         out.close();
         skt.close();
         srvr.close();
      }
      catch(Exception e) {
         System.out.print("He thong khong lam viec!\n");
      }
   }
}
```

2.Lập trình phía máy khách

```
class Client {
   public static void main(String args[]) {
      try {
         Socket skt = new Socket("localhost", 1234);
         BufferedReader in = new BufferedReader(new
```

```
            InputStreamReader(skt.getInputStream()));
         System.out.print("Du lieu da nhan: '");
         while (!in.ready()) {
         System.out.println(in.readLine()); // Doc mot dong va in no ra
man hinh
         System.out.print("'\n");
         in.close();
           }
         }
       catch(Exception e) {
         System.out.print("He thong khong lam viec!\n");
       }
   }
}
```

3.15.

1/ Lập trình phía máy chủ

```
import java.io.*;
import java.net.*;
import java.util.*;
public class threadServer extends Thread {
      Socket socket=null;
public threadServer(Socket socket){
      this.socket=socket;
}
public void run(){
try{
      PrintWriter out=new PrintWriter(socket.getOutputStream(),true);
      BufferedReader in=new BufferedReader(new
      InputStreamReader(socket.getInputStream()));
      String inLine;
      while(true){
            inLine=in.readLine();
            Calendar cal =new GregorianCalendar();
            if(inLine.equalsIgnoreCase("quit"))break;
            else {
                  out.println(daochuoi(inLine));
                        }
                     }
   }catch(IOException e){
      e.printStackTrace();
      }
   }
public String daochuoi(String st){
      int j=0;
      Char[] c=st.toCharArray();
      for(int i=st.length();i>=0;i--){
            c[j]=st.charAt(i);
            ++j;
   }
      String st2= String(c);
      return st2;
   }
}
```

2.Lập trình phía máy khách (Client.java)

```
import java.io.*;
import java.net.*;
public class client {
public static void main(String[] args)throws Exception{
Socket cSk=null;
PrintWriter out=null;
BufferedReader in=null;
try{
cSk=new Socket("127.0.0.1",9999) ; // khởi tạo một socket
// out : đưa chuỗi lên server
out=new PrintWriter(cSk.getOutputStream(),true);
// in: nhận chuỗi được gửi từ server
in=new BufferedReader(new InputStreamReader(cSk.getInputStream()));
}catch(IOException e){
e.printStackTrace();
}
String inLine;
// uIn: cho phép nhập một chuỗi từ bàn phím.
BufferedReader uIn=new BufferedReader(new InputStreamReader(System.in));
while(true){
inLine=uIn.readLine();// tiến hành đọc từ bàn phím và gán chuỗi đọc được cho inLine
if(inLine.equalsIgnoreCase("quit")){
break;
}
out.println(inLine);// gửi chuỗi lên server
out.flush();
System.out.println(in.readLine());// in ra màn hình chuỗi nhận về từ server
}
}
}
```

Hướng dẫn chạy :

Chạy file server.java trước sau đó là client.java. Bạn có thể dùng telnet thay cho client cũng được. Để dùng telnet: vào DOS đánh dòng lệnh telnet 127.0.0.1 9999. Bây giờ nhập vào một chuỗi sẽ nhận được ngay một chuỗi mới là đảo của chuỗi vừa nhập vào.

---o0o---

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua"

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 04 Số giờ : 03 (1 buổi)

GVHD :ThS.Nguyễn Minh Nhật

LAB 04
LẬP TRÌNH SOCKET VỚI UDP

I.MỤC TIÊU

Lập trình kết nối giữa 2 máy tính bằng socket với kỹ thuật UDP

II. NỘI DUNG

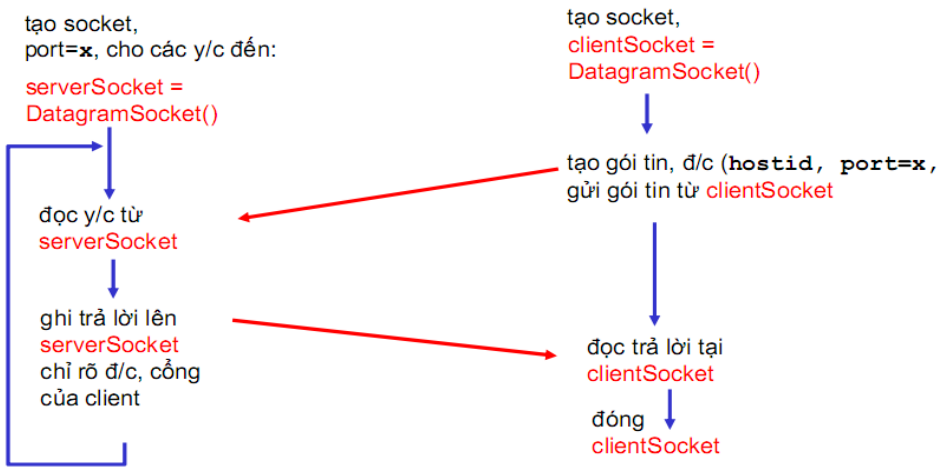
A.LÝ THUYẾT

1.Đặc điểm của lập trinh Socket với UDP

- Về lập trình
  - +Không có liên kết giữa client và Server theo cơ chế "bắt tay 3 bước".
  - + Bên gửi chỉ rõ địa chỉ và port của phía nhận trên mỗi gói tin
  - + Server sẽ tìm địa chỉ IP và số hiệu cổng tương ứng bên trong gói tin
- Về cơ chế truyền file
  - +Dữ liệu truyền theo dạng gói (package), do đó cần đóng gói trước khi gởi hoặc nhận
  - +Các gói tin có thể bị mất trên đường truyền hoặc đến không theo thứ tự.

2. Tương tác client/server qua UDP socket

Server (Máy hostID) **Client**



- **Các giai đoạn lập trình**

- Đối với Client

+ Tạo input Stream
+ Tạo Client Socket

- Đối với Server

+ Tạo datagram tại port cho trước
+ Tạo vùng đệm nhận datagram

+ Chuyển đổi hostname sang IP
+ Tạo datagram
+ Gởi datagram đến Server
+ Đọc datagram từ Server
+ Nhận datagram
+ Lấy địa chỉ IP, Port của Client
+Tạo datagram để gởi tới Client
+ Ghi datagram ra socket

B.BÀI TẬP

4.1. Xây dựng một DatagramPacket để nhận dữ liệu
4.2. Tìm port UDP cục bộ
4.3. Viết một UDP loại bỏ một Client
4.4. Viết một UDP loại bỏ một Server
4.5. Xây dựng UDP định thời gian của Client
4.6. Xây dựng lớp UDP Server
4.7. Mở rộng một tính năng UDP loại bỏ một Server
4.8. Xây dựng một UDP đăng nhập loại bỏ máy Server
4.9. Xây dựng UDP Echo cho máy Server
4.10. Xây dựng UDP Echo cho máy Client
4.11. Xây dựng lớp Threat gởi
4.12. Xây dựng lớp Threat nhận

HƯỚNG DẪN

4.1.

```
import java.net.*;
public class DatagramExample {
  public static void main(String[] args) {

    String s = "This is a test.";
    byte[] data = s.getBytes();
    try {
      InetAddress ia = InetAddress.getByName("www.dtu.edu.vn ");
      int port = 7;
      DatagramPacket dp
       = new DatagramPacket(data, data.length, ia, port);
      System.out.println("This packet is addressed to "
       + dp.getAddress() + " on port " + dp.getPort());
      System.out.println("There are " + dp.getLength()
       + " bytes of data in the packet");
      System.out.println(
        new String(dp.getData(), dp.getOffset(), dp.getLength()));
    }
    catch (UnknownHostException e) {
      System.err.println(e);
    }
  }
}
```

4.2.

```
import java.net.*;
public class UDPPortScanner {
  public static void main(String[] args) {
```

```
        for (int port = 1024; port <= 65535; port++) {
          try {
            // the next line will fail and drop into the catch block if
            // there is already a server running on port i
            DatagramSocket server = new DatagramSocket(port);
            server.close();
          }
          catch (SocketException ex) {
            System.out.println("There is a server on port " + port + ".");
          } // end try
        } // end for
      }
    }
```

4.3.

```
    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";
        }

        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 sex) {
```

```
            System.err.println(sex);
          }
          catch (IOException ioex) {
            System.err.println(ioex);
          }
       }  // end main
      }
```

4.4.

```
      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
          }

          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);
                // reset the length for the next packet
                packet.setLength(buffer.length);
              }
              catch (IOException ex) {
                System.err.println(ex);
              }
             } // end while
          }  // end try
          catch (SocketException  ex) {
            System.err.println(ex);
          }  // end catch

        }  // end main
      }
```

4.5.

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

```
public class UDPDaytimeServer extends UDPServer {

  public final static int DEFAULT_PORT = 13;

  public UDPDaytimeServer() throws SocketException {
    super(DEFAULT_PORT);
  }

  public void respond(DatagramPacket packet) {

    try {
      Date now = new Date();
      String response = now.toString() + "\r\n";
      byte[] data = response.getBytes("ASCII");
      DatagramPacket outgoing = new DatagramPacket(data,
       data.length, packet.getAddress(), packet.getPort());
      socket.send(outgoing);
    }
    catch (IOException ex) {
      System.err.println(ex);
    }
   }

  public static void main(String[] args) {

   try {
     UDPServer server = new UDPDaytimeServer();
     server.start();
   }
   catch (SocketException ex) {
     System.err.println(ex);
   }

  }

}
```

4.6.

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

      public abstract class UDPServer extends Thread {

        private int bufferSize; // in bytes
        protected DatagramSocket ds;

        public UDPServer(int port, int bufferSize)
         throws SocketException {
          this.bufferSize = bufferSize;
          this.ds = new DatagramSocket(port);
        }

        public UDPServer(int port) throws SocketException {
          this(port, 8192);
        }

        public void run() {
```

```
      byte[] buffer = new byte[bufferSize];
      while (true) {
        DatagramPacket incoming = new DatagramPacket(buffer,
    buffer.length);
        try {
          ds.receive(incoming);
          this.respond(incoming);
        }
        catch (IOException e) {
          System.err.println(e);
        }
      } // end while

    }  // end run

    public abstract void respond(DatagramPacket request);
  }
```

4.7.

```
  import java.net.*;

  public class FastUDPDiscardServer extends UDPServer {

    public final static int DEFAULT_PORT = 9;

    public FastUDPDiscardServer() throws SocketException {
      super(DEFAULT_PORT);
    }

    public void respond(DatagramPacket packet) {}

    public static void main(String[] args) {

     try {
       UDPServer server = new FastUDPDiscardServer();
       server.start();
     }
     catch (SocketException ex) {
       System.err.println(ex);
     }
     }
  }
```

4.8.

```
  import java.net.*;

  public class LoggingUDPDiscardServer extends UDPServer {

    public final static int DEFAULT_PORT = 9999;

    public LoggingUDPDiscardServer() throws SocketException {
      super(DEFAULT_PORT);
    }

    public void respond(DatagramPacket packet) {

      byte[] data = new byte[packet.getLength()];
      System.arraycopy(packet.getData(), 0, data, 0, packet.getLength());
      try {
```

```
            String s = new String(data, "8859_1");
            System.out.println(packet.getAddress() + " at port "
             + packet.getPort() + " says " + s);
          }
          catch (java.io.UnsupportedEncodingException ex) {
            // This shouldn't happen
          }
          }

        public static void main(String[] args) {

          try {
            UDPServer erver = new LoggingUDPDiscardServer();
            server.start();
          }
          catch (SocketException ex) {
            System.err.println(ex);
          }
        }
      }
```

4.9.

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

      public class UDPEchoServer extends UDPServer {

        public final static int DEFAULT_PORT = 7;

        public UDPEchoServer() throws SocketException {
          super(DEFAULT_PORT);
        }

        public void respond(DatagramPacket packet) {

          try {
            DatagramPacket outgoing = new DatagramPacket(packet.getData(),
             packet.getLength(), packet.getAddress(), packet.getPort());
            socket.send(outgoing);
          }
          catch (IOException ex) {
            System.err.println(ex);
          }
      }
        public static void main(String[] args) {

          try {
            UDPServer server = new UDPEchoServer();
            server.start();
          }
          catch (SocketException ex) {
            System.err.println(ex);
          }
        }
      }
```

4.10.

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

```
import java.io.*;

public class UDPEchoClient {

  public final static int DEFAULT_PORT = 7;
  public static void main(String[] args) {
    String hostname = "localhost";
    int port = DEFAULT_PORT;

    if (args.length > 0) {
      hostname = args[0];
    }

    try {
      InetAddress ia = InetAddress.getByName(hostname);
      Thread sender = new SenderThread(ia, DEFAULT_PORT);
      sender.start();
      Thread receiver = new ReceiverThread(sender.getSocket());
      receiver.start();
    }
    catch (UnknownHostException ex) {
      System.err.println(ex);
    }
    catch (SocketException ex) {
      System.err.println(ex);
    }
  }  // end main
}
```

4.11.

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

public class SenderThread extends Thread {

  private InetAddress server;
  private DatagramSocket socket;
  private boolean stopped = false;
  private int port;

  public SenderThread(InetAddress address, int port)
   throws SocketException {
    this.server = address;
    this.port = port;
    this.socket = new DatagramSocket();
    this.socket.connect(server, port);
  }

  public void halt() {
    this.stopped = true;
  }
  public DatagramSocket getSocket() {
    return this.socket;
  }
  public void run() {
    try {
      BufferedReader userInput
       = new BufferedReader(new InputStreamReader(System.in));
```

```
            while (true) {
              if (stopped) return;
              String theLine = userInput.readLine();
              if (theLine.equals(".")) break;
              byte[] data = theLine.getBytes();
              DatagramPacket output
               = new DatagramPacket(data, data.length, server, port);
              socket.send(output);
              Thread.yield();
            }
          }  // end try
          catch (IOException ex) {
            System.err.println(ex);
          }
        }  // end run
      }
```

4.12.

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

      class ReceiverThread extends Thread {
        DatagramSocket socket;
        private boolean stopped = false;
        public ReceiverThread(DatagramSocket ds) throws SocketException {
          this.socket = ds;
        }
        public void halt() {
          this.stopped = true;
        }
        public void run() {
          byte[] buffer = new byte[65507];
          while (true) {
            if (stopped) return;
            DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
            try {
              socket.receive(dp);
              String s = new String(dp.getData(), 0, dp.getLength());
              System.out.println(s);
              Thread.yield();
            }
            catch (IOException ex) {
              System.err.println(ex);
            }
          }
        }
      }
```

---o0o---

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 05 Số giờ : 03 (1 buổi)

GVHD: ThS. Nguyễn Minh Nhật

LAB 05
LẬP TRÌNH SOCKET VỚI TCP

I.MỤC TIÊU

Lập trình kết nối giữa 2 máy tính bằng socket với kỹ thuật TCP

II. NỘI DUNG

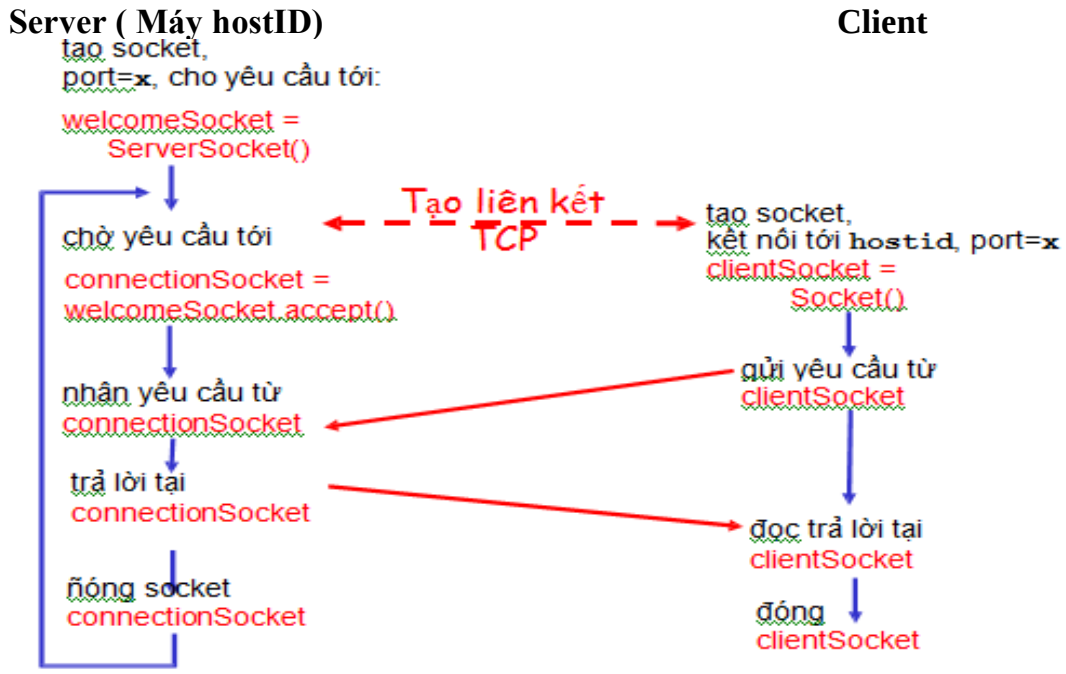
A.LÝ THUYẾT

A.LÝ THUYẾT

1.Socket và dịch vụ TCP

- **Socket:** Cửa giao tiếp giữa các tiến trình và giao thứcgiao vận (UCP hoặc TCP)
- Để kết nối qua Socket băng kỹ thuật TCP, dữ liệu truyền dạng dòng **bytes.**
- Đặc điểm kết nối TCP:
  +Dữ liệu truyền dạng dòng bytes
  +Tuân thủ theo cơ chế "bắt tay 3 bước" giữa Server và Client, tứ là truyền theo hướng kết nối
  +Dữ liệu truyền theo 1 kênh xác định trước
  +Ổn định và tin cậy
- Client phải gửi yêu cầu tới server
  + Tiến trình máy chủ phải đang được thực hiện
  + Máy chủ phải mở socket (cổng) để nhận yêu cầu từ client
- Client yêu cầu server bằng cách:
  + Tạo một socket TCP trên máy
  + Chỉ rõ IP address & port number của tiến trình máy chủ
  + Khi client tạo socket: client TCP tạo liên kết tới server TCP

2. Tương tác client/server qua TCP socket



B.BÀI TẬP

5.1. Lấy thông in từ Socket

5.2. Viết chương trình trao đổi thông điệp giữa 2 máy tính qua Socket TCP

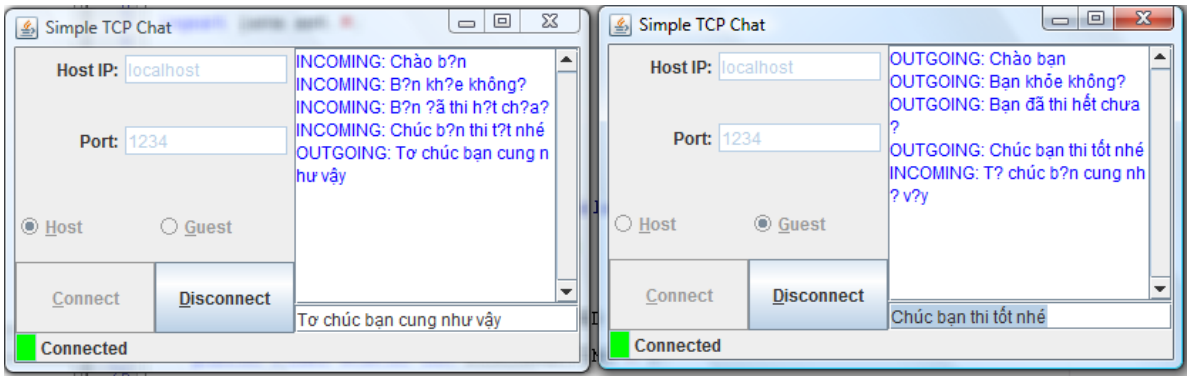
5.3. Thực hiện truyền file trên mạng qua giao thức ftp

5.4. Viết chương trình đọc một chuỗi từ Client gởi đến Server. Sau khi nhận được chuỗi Server thực hiện biến chữ thường thành chữ hoa và gởi kết quả trả cho Client

5.5. Xây dựng mộ ứng dụng three tier có một khách hàng kết nối đến máy chủ, với yêu cầu phản hồi từ cơ sở dữ liệu. Để thực hiện, máy chủ kết nối vào máy chủ dữ liệu và yêu cầu các truy vấn theo yêu cầu của khách hàng từ cơ sở dữ liệu. Sau đó các phản hồi lại cho Client dưới hình thức dữ liệu được cho vào máy chủ từ dữ liệu máy chủ, cuối cùng chuyển cho khách hàng.

5.6. Xây dựng mộ ứng dụng three tier có một khách hàng kết nối đến máy chủ, với yêu cầu phản hồi từ cơ sở dữ liệu. Để thực hiện, máy chủ kết nối vào máy chủ dữ liệu và yêu cầu các truy vấn theo yêu cầu của khách hàng từ cơ sở dữ liệu. Sau đó các phản hồi lại cho Client dưới hình thức dữ liệu được cho vào máy chủ từ dữ liệu máy chủ; cuối cùng chuyển cho khách hàng.

5.7. Xây dựng chương trình chat giữa 2 máy tính, sử dụng Socket TCP như hình bên dưới. Cả 2 chương trình đều viết trên 1 file. Tùy thuộc vào cách chọn các kết nối Host hoặc Guest mà chúng chuyển kết nối Server hoặc Client tương ứng.



***5.8.** Thiết kế và lập trình một hệ thống client – server thực hiện các chức năng trao đổi các bản tin dưới dạng chuỗi ký tự thông qua TCP socket. Hệ thống gồm các phần: chat client và chat server. Có 3 loại bản tin được trao đổi giữa client và server:

• **Connection setup**: Khi một người sử dụng đánh dòng lệnh remote-chat <địa chỉ IP server>:<địa chỉ port> <địa chỉ IP của client bị gọi>:<địa chỉ port>, một kết nối TCP sẽ được thiết lập giữa client đó và chat server. Sau khi kết nối TCP được thiết lập, client gửi bản tin connection setup, bao gồm các trường:

+ Trường nhận dạng bản tin: là một số nguyên 32 bit. Với bản tin connection setup thì trường này có giá trị 0.

+ Trường địa chỉ: dài 6 byte bao gồm địa chỉ IP của client bị gọi (4 byte) và địa chỉ port của client bị gọi (2 byte).

Khi nhận được bản tin connection setup, server sẽ thiết lập một kết nối TCP với client bị gọi với địa chỉ IP và địa chỉ port đã được chỉ ra trong bản tin connection setup.

• **Data exchange**: được sử dụng để trao đổi dữ liệu giữa client và server. Bao gồm các trường:

+Trường nhận dạng bản tin: là một số nguyên 32 bit. Với bản tin data exchange thì trường này có giá trị 1.

+ Trường độ dài dữ liệu: là một số nguyên chỉ ra đồ dài của bản tin text.

+ Trường dữ liệu: chứa bản tin text cần trao đổi.

Khi nhận được dữ liệu được gửi từ một client, chat server phải chuyển bản tin text đó đến client phía bên kia, cũng sử dụng bản tin data exchange.

• **Connection teardown**: khi người sử dụng bấm Ctrl-C thì bản tin connection teardown sẽ được gửi trước khi chương trình ở phía client kết thúc. Connection teardown gồm 1 trường:

+Trường nhận dạng bản tin: là một số nguyên 32 bit. Với bản tin connection teardown thì trường này có giá trị 2.

Khi nhận được bản tin này, chat server sẽ gửi bản tin đến client tương ứng, client sau khi nhận được connection teardown thì hủy bỏ kết nối TCP với server và kết thúc chương trình.

HƯỚNG DẪN

5.1.

```
import java.io.IOException;
```

```
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;

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());
      } // end try
      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);
      }
    }
  }
}
```

5.2

- Trên Server :

```
import java.lang.*;
import java.io.*;
import java.net.*;
class Server {
   public static void main(String args[]) {
      String data = "Xin chao cac ban cua may tram TCP Socket!";
      try {
         ServerSocket srvr = new ServerSocket(1234);
         Socket skt = srvr.accept();
         System.out.print("May chu da duoc ket noi!\n");
         PrintWriter out = new PrintWriter(skt.getOutputStream(), true);
         System.out.print("Du lieu se gui di cho may tram: '" + data + "'\n");
         out.print(data);
         out.close();
         skt.close();
         srvr.close();
      }
      catch(Exception e) {
         System.out.print("He thong khong lam viec!\n");
      }
   }
}
```

- Trên Client

```
import java.lang.*;
import java.io.*;
import java.net.*;
class Client {
```

```
   public static void main(String args[]) {
      try {
         Socket skt = new Socket("localhost", 1234);
         BufferedReader in = new BufferedReader(new
            InputStreamReader(skt.getInputStream()));
         System.out.print("Du lieu da nhan: '");
          while (!in.ready()) {}
         System.out.println(in.readLine()); // Doc mot dong va in no ra man
hinh
          System.out.print("'\n");
         in.close();
      }
      catch(Exception e) {
         System.out.print("He thong khong lam viec!\n");
      }
   }
}
```

5.3

```
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;


    public class ftp extends Frame {
    public static String hostAddress = "";
    public static int portNumber = 0, maxClients = 0;
    public static Vector sockets = null;
    public static ftp tp;
    public static String fileName = "",path = "";
    public static int check = 0;
    public static Socket connection = null;
    public static ObjectOutputStream out = null;
    public static ObjectInputStream in = null;


        public static void main (String [] args) throws IOException {
        BufferedReader stdin = new BufferedReader
        (new InputStreamReader (System.in));
        System.out.println("1. Connect to host");
        System.out.println("2. Wait for connections");
        System.out.println();
        System.out.print("Please make a choice: ");
        System.out.flush();
        int choice = Integer.parseInt(stdin.readLine());


            if (choice == 1) {
            System.out.print("Please type in the host address: ");
            System.out.flush();
            hostAddress = stdin.readLine();
            System.out.print("Please type in the host port number: ");
            System.out.flush();
            portNumber = Integer.parseInt(stdin.readLine());
```

```
        }

            if (choice == 2) {
            System.out.print("Give the port number to listen for requests: ");
            System.out.flush();
            portNumber = Integer.parseInt(stdin.readLine());
            System.out.print("Give the maximum number of clients for this
server: ");
            System.out.flush();
            maxClients = Integer.parseInt(stdin.readLine());
        }

        tp = new ftp(choice);
    }

    public Label l;
    public TextField tf;
    public Button browse;
    public Button send;
    public Button reset;

        public ftp (int c) {
        setTitle("Ezad's File Transfer Protocol");
        setSize(300 , 350);
        setLayout(null);
        addWindowListener(new WindowAdapter () { public void windowClosing
(WindowEvent e) { System.exit(0); } } );
        l = new Label("File:");
        add(l);
        l.setBounds(15,87,50,20);
        tf = new TextField("");
        add(tf);
        tf.setBounds(13,114,200,20);
        browse = new Button("Browse");
        browse.addActionListener(new buttonListener());
        add(browse);
        browse.setBounds(223,113,50,20);
        send = new Button("Send");
        send.addActionListener(new buttonListener());
        add(send);
        send.setBounds(64,168,50,20);
        reset = new Button("Reset");
        reset.addActionListener(new buttonListener());
        add(reset);
        reset.setBounds(120,168,50,20);
        show();

            if (c == 1) {
            check = 10;

                try {
                connection = new Socket (hostAddress,portNumber);
                out = new ObjectOutputStream(connection.getOutputStream());
                out.flush();
                in = new ObjectInputStream(connection.getInputStream());
                int flag = 0;
```

```
                    while (true) {
                    Object recieved = in.readObject();

                        switch (flag) {
                        case 0:

                            if (recieved.equals("sot")) {
                            flag++;
                        }

                       break;
                        case 1:
                        fileName = (String) recieved;
                        int option =
JOptionPane.showConfirmDialog(this,connection.getInetAddress().getHostName()+"
is sending you "+fileName+"!\nDo you want to recieve it?","Recieve
Confirm",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);


                            if (option == JOptionPane.YES_OPTION) {
                            flag++;

                            } else {

                            flag = 0;
                        }

                        break;
                        case 2:
                        byte[] b = (byte[])recieved;
                        FileOutputStream ff = new FileOutputStream(fileName);
                        //ObjectOutputStream o = new ObjectOutpu
                        //     tStream(ff);
                        //o.writeObject(b);
                        //o.flush();
                        //o.close();
                        ff.write(b);
                        flag = 0;
                        //out.writeObject("rcn");
                        JOptionPane.showMessageDialog(this,"File
Recieved!","Confirmation",JOptionPane.INFORMATION_MESSAGE);
                        break;
                    }

                    Thread.yield();
                }

             } catch (Exception e) {System.out.println(e);}

        }

           if (c == 2) {
            sockets = new Vector();
            check = 5;

                try {
                ServerSocket connect = new ServerSocket(portNumber,maxClients);
```

```
                    while (true) {
                    sockets.addElement(new ThreadedSocket(connect.accept()));
                    Thread.yield();
                }

            } catch (IOException ioe) {System.out.println(ioe);}

        }
    }

        public static String showDialog () {
        FileDialog fd = new FileDialog(new Frame(),"Select
File...",FileDialog.LOAD);
        fd.show();
        return fd.getDirectory()+fd.getFile();
    }

        private class buttonListener implements ActionListener {

            public void actionPerformed (ActionEvent e) {
            byte[] array = null;

                if (e.getSource() == browse) {
                path = showDialog();
                tf.setText(path);
                int index = path.lastIndexOf("\\");
                fileName = path.substring(index+1);
            }

                if (e.getSource() == send) {

                    //if (fileName != null && !fileName.equa

                        //     ls("")) {

                            try {
                            FileInputStream f = new FileInputStream (path);
                            int size = f.available();
                            array = new byte[size];
                            f.read(array,0,size);

                                if (check == 5) {

                                    for (int i=0;i<sockets.size();i++) {
                                    ThreadedSocket temp =
(ThreadedSocket)sockets.elementAt(i);
                                    temp.out.writeObject("sot");
                                    temp.out.flush();
                                    temp.out.writeObject(fileName);
                                    temp.out.flush();
                                    temp.out.writeObject(array);
                                    temp.out.flush();
                                }
                            }

                                if (check == 10) {
```

```
                                out.writeObject("sot");
                                out.flush();
                                out.writeObject(fileName);
                                out.flush();
                                out.writeObject(array);
                                out.flush();
                            }

                            JOptionPane.showMessageDialog(null,"File
Sent!","Confirmation",JOptionPane.INFORMATION_MESSAGE);
                        } catch (Exception ex) {}


                        //}
                    }
                        if (e.getSource() == reset) {
                        tf.setText("");
                    }
                }
            }
        }

            class ThreadedSocket extends Thread {
            public Socket socket;
            public ObjectInputStream in;
            public ObjectOutputStream out;


                public ThreadedSocket (Socket s) {
                socket = s;
                    try {
                    out = new ObjectOutputStream(socket.getOutputStream());
                    out.flush();
                    in = new ObjectInputStream(socket.getInputStream());
                } catch (Exception e) {System.out.println(e);}

                start();
            }

                public void run () {

                    try {
                    int flag = 0;
                    String fileName = "";

                       while (true) {
                        Object recieved = in.readObject();

                           switch (flag) {
                            case 0:
                                if (recieved.equals("sot")) {
                                flag++;
                            }
                            break;
                            case 1:
                            fileName = (String) recieved;
```

```
                            int option =
JOptionPane.showConfirmDialog(null,socket.getInetAddress().getHostName()+" is
sending you "+fileName+"!\nDo you want to recieve it?","Recieve
Confirm",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);

                                if (option == JOptionPane.YES_OPTION) {
                                flag++;

                                } else {

                                flag = 0;
                            }

                            break;
                            case 2:
                            byte[] b = (byte[])recieved;
                            FileOutputStream ff = new
FileOutputStream(fileName);
                            //ObjectOutputStream o = new ObjectOutpu
                            //     tStream(ff);
                            //o.writeObject(b);
                            //o.flush();
                            ff.write(b);
                            flag = 0;
                            //out.writeObject("rcn");
                            JOptionPane.showMessageDialog(null,"File
Recieved!","Confirmation",JOptionPane.INFORMATION_MESSAGE);
                            break;
                        }
                        Thread.yield();
                    }
                } catch (Exception e) {System.out.println(e);}
            }
        }
```

5.5.

Thực hiện theo trình tự sau :

B1. Tạo cơ sở dữ liệu **atif.mdb** (chứa 1Table Player)

B2. Tạo lớp DataServer để truy vấn vào CSDL của Server

Giả sử đoạn code như sau :

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

class DataServer
{
      static String query="";
      static ResultSet rs;
      static Statement st;
      static Connection con;
      static String result="";

      public static void main(String args[]) throws Exception
      {
            try
            {
            ServerSocket datasersoc = new ServerSocket(4445);
            Socket datasoc = datasersoc.accept();
```

```
      System.out.println("Hello"+(datasoc.getInetAddress().getHostName()));
            ObjectOutputStream oos = new
ObjectOutputStream(datasoc.getOutputStream());
            ObjectInputStream ois = new
ObjectInputStream(datasoc.getInputStream());

            query = ois.readObject()+"";

            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            String source = "jdbc:odbc:atif";
            con = DriverManager.getConnection(source);
            st = con.createStatement();
            rs = st.executeQuery(query);
            System.out.println(query);

            while(rs.next())
            {
                  result += "\n" +rs.getString("PlayerName")+ "
            "+rs.getInt("PlayerID");
            }

                  oos.writeObject(result);
            }
            catch(Exception e)
            {
                  System.out.println("Connection Closed"+e);
            }
      }
}
```

B3. Tạo lời chào đến Server, sau đó yêu cầu Server gởi tên của các khách hàng từ CSDL **atif.mdb**

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

class client
{
      public static void main(String arg[])
      {
            String query="";
            String result ="";


            try
            {

             Socket Soc = new Socket("127.0.0.1",4444);

             ObjectOutputStream oos = new
ObjectOutputStream(Soc.getOutputStream());

             ObjectInputStream ois = new
ObjectInputStream(Soc.getInputStream());
             BufferedInputStream in = new BufferedInputStream(ois);
```

```
             BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
             query = br.readLine();

             oos.writeObject(query);

             result = ois.readObject()+"";

             System.out.println(result);

      }
            catch(Exception e)
             {
                  System.out.println("Connection Closed");

             }
       }
   }
```

5.7.

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;
import java.awt.event.KeyEvent;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;


public class ChatTCP  implements Runnable {
   // Kh?i t?o các tr?ng thái k?t n?i
   public final static int NULL = 0;
   public final static int DISCONNECTED = 1;
   public final static int DISCONNECTING = 2;
   public final static int BEGIN_CONNECT = 3;
   public final static int CONNECTED = 4;
```

```
// Kh?i t?o m?t s? h?ng s?
public final static String statusMessages[] = {
   " Error! Could not connect!", " Disconnected",
   " Disconnecting...", " Connecting...", " Connected"
};
public final static ChatTCP tcpObj = new ChatTCP();
public final static String END_CHAT_SESSION =
   new Character((char)0).toString(); // k?t thúc phiên chat

// Thông tin k?t n?i
public static String hostIP = "localhost";
public static int port = 1234;
public static int connectionStatus = DISCONNECTED;
public static boolean isHost = true;
public static String statusString = statusMessages[connectionStatus];
public static StringBuffer toAppend = new StringBuffer("");
public static StringBuffer toSend = new StringBuffer("");

// Khai báo các thành ph?n giao di?n
public static JFrame mainFrame = null;
public static JTextArea chatText = null;
public static JTextField chatLine = null;
public static JPanel statusBar = null;
public static JLabel statusField = null;
public static JTextField statusColor = null;
public static JTextField ipField = null;
public static JTextField portField = null;
public static JRadioButton hostOption = null;
public static JRadioButton guestOption = null;
public static JButton connectButton = null;
public static JButton disconnectButton = null;

// Khai báo các thành ph?n TCP
public static ServerSocket hostServer = null;
public static Socket socket = null;
public static BufferedReader in = null;
public static PrintWriter out = null;

////////////////////////////////////////////////////////////////

private static JPanel initOptionsPane() {
   JPanel pane = null;
   ActionAdapter buttonListener = null;

// T?o JPanel
   JPanel optionsPane = new JPanel(new GridLayout(4, 1));
// IP
   pane = new JPanel(new FlowLayout(FlowLayout.RIGHT));
   pane.add(new JLabel("Host IP:"));
   ipField = new JTextField(10); ipField.setText(hostIP);
   ipField.setEnabled(false);
   ipField.addFocusListener(new FocusAdapter() {
         @Override
         public void focusLost(FocusEvent e) {
            ipField.selectAll();
            // khi ng?t k?t n?i m?i ???c s?a
```

```
            if (connectionStatus != DISCONNECTED) {
               changeStatusNTS(NULL, true);
            }
            else {
               hostIP = ipField.getText();
            }
         }
      });
      pane.add(ipField);
      optionsPane.add(pane);

      // C?ng
      pane = new JPanel(new FlowLayout(FlowLayout.RIGHT));
      pane.add(new JLabel("Port:"));
      portField = new JTextField(10); portField.setEditable(true);
      portField.setText((new Integer(port)).toString());
      portField.addFocusListener(new FocusAdapter() {
            @Override
            public void focusLost(FocusEvent e) {
               // ch? ???c s?a khi ?ã ng?t k?t n?i
               if (connectionStatus != DISCONNECTED) {
                  changeStatusNTS(NULL, true);
               }
               else {
                  int temp;
                  try {
                     temp = Integer.parseInt(portField.getText());
                     port = temp;
                  }
                  catch (NumberFormatException nfe) {
                     portField.setText((new Integer(port)).toString());
                     mainFrame.repaint();
                  }
               }
            }
});
      pane.add(portField);
      optionsPane.add(pane);

      // l?a ch?n host c?a Guest
      buttonListener = new ActionAdapter() {
            @Override
            public void actionPerformed(ActionEvent e) {
               if (connectionStatus != DISCONNECTED) {
                  changeStatusNTS(NULL, true);
               }
               else {
                  isHost = e.getActionCommand().equals("host");

                  // Không th? s?a host n?u ?ã ???c ch?n
                  if (isHost) {
                     ipField.setEnabled(false);
                     ipField.setText("localhost");
                     hostIP = "localhost";
                  }
                  else {
                     ipField.setEnabled(true);
```

```
                }
            }
        }
    };
    ButtonGroup bg = new ButtonGroup();
    hostOption = new JRadioButton("Host", true);
    hostOption.setMnemonic(KeyEvent.VK_H);
    hostOption.setActionCommand("host");
    hostOption.addActionListener(buttonListener);
    guestOption = new JRadioButton("Guest", false);
    guestOption.setMnemonic(KeyEvent.VK_G);
    guestOption.setActionCommand("guest");
    guestOption.addActionListener(buttonListener);
    bg.add(hostOption);
    bg.add(guestOption);
    pane = new JPanel(new GridLayout(1, 2));
    pane.add(hostOption);
    pane.add(guestOption);
    optionsPane.add(pane);

    // Nút k?t n?i và ng?t k?t n?i
    JPanel buttonPane = new JPanel(new GridLayout(1, 2));
    buttonListener = new ActionAdapter() {
          @Override
          public void actionPerformed(ActionEvent e) {
             // Yêu c?u kh?i t?o k?t n?i
             if (e.getActionCommand().equals("connect")) {
                changeStatusNTS(BEGIN_CONNECT, true);
             }
             // Ng?t k?t n?i
             else {
                changeStatusNTS(DISCONNECTING, true);
             }
          }
       };
    connectButton = new JButton("Connect");
    connectButton.setMnemonic(KeyEvent.VK_C);
    connectButton.setActionCommand("connect");
    connectButton.addActionListener(buttonListener);
    connectButton.setEnabled(true);
    disconnectButton = new JButton("Disconnect");
    disconnectButton.setMnemonic(KeyEvent.VK_D);
    disconnectButton.setActionCommand("disconnect");
    disconnectButton.addActionListener(buttonListener);
    disconnectButton.setEnabled(false);
    buttonPane.add(connectButton);
    buttonPane.add(disconnectButton);
    optionsPane.add(buttonPane);

    return optionsPane;
}
 ////////////////////////////////////////////////////////////////
// Kh?i t?o giao di?n
private static void initGUI() {
    // Cài ??t tr?ng thái
    statusField = new JLabel();
    statusField.setText(statusMessages[DISCONNECTED]);
```

```
    statusColor = new JTextField(1);
    statusColor.setBackground(Color.red);
    statusColor.setEditable(false);
    statusBar = new JPanel(new BorderLayout());
    statusBar.add(statusColor, BorderLayout.WEST);
    statusBar.add(statusField, BorderLayout.CENTER);
    JPanel optionsPane = initOptionsPane();
    // Cài ??t panel chat
    JPanel chatPane = new JPanel(new BorderLayout());
    chatText = new JTextArea(10, 20);
    chatText.setLineWrap(true);
    chatText.setEditable(false);
    chatText.setForeground(Color.blue);
    JScrollPane chatTextPane = new JScrollPane(chatText,
       JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
       JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    chatLine = new JTextField();
    chatLine.setEnabled(false);
    chatLine.addActionListener(new ActionAdapter() {
          @Override
          public void actionPerformed(ActionEvent e) {
             String s = chatLine.getText();
             if (!s.equals("")) {
                appendToChatBox("OUTGOING: " + s + "\n");
                chatLine.selectAll();
                // G?i tin chat
                sendString(s);
             }
          }
       });
    chatPane.add(chatLine, BorderLayout.SOUTH);
    chatPane.add(chatTextPane, BorderLayout.CENTER);
    chatPane.setPreferredSize(new Dimension(200, 200));

    // Cài ??t panel chính
    JPanel mainPane = new JPanel(new BorderLayout());
    mainPane.add(statusBar, BorderLayout.SOUTH);
    mainPane.add(optionsPane, BorderLayout.WEST);
    mainPane.add(chatPane, BorderLayout.CENTER);
    // Cài ??t Frame chính
    mainFrame = new JFrame("Simple TCP Chat");
    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainFrame.setContentPane(mainPane);
    mainFrame.setSize(mainFrame.getPreferredSize());
    mainFrame.setLocation(200, 200);
    mainFrame.pack();
    mainFrame.setVisible(true);
}
////////////////////////////////////////////////////////////////
// thay ??i tr?ng thái
private static void changeStatusTS(int newConnectStatus, boolean noError) {
    if (newConnectStatus != NULL) {
       connectionStatus = newConnectStatus;
    }
    // N?u có l?i thông báo ? ph?n tr?ng thái
    if (noError) {
       statusString = statusMessages[connectionStatus];
```

```
    }
    else {
       statusString = statusMessages[NULL];
    }
    SwingUtilities.invokeLater(tcpObj);
}
////////////////////////////////
private static void changeStatusNTS(int newConnectStatus, boolean noError) {

    if (newConnectStatus != NULL) {
       connectionStatus = newConnectStatus;
    }

    if (noError) {
       statusString = statusMessages[connectionStatus];
    }

    else {
       statusString = statusMessages[NULL];
    }
     tcpObj.run();
}
/////////////////////////////////////////////////////////////////
private static void appendToChatBox(String s) {
    synchronized (toAppend) {
       toAppend.append(s);
    }
}
 ////////////////////////////////////////////////////////////////
private static void sendString(String s) {
    synchronized (toSend) {
          StringBuffer append = toSend.append(s).append("\n");
    }
}
 ////////////////////////////////////////////////////////////////
private static void cleanUp() {
    try {
       if (hostServer != null) {
          hostServer.close();
          hostServer = null;
       }
    }
    catch (IOException e) { hostServer = null; }
    try {
       if (socket != null) {
          socket.close();
          socket = null;
       }
    }
    catch (IOException e) { socket = null; }

    try {
       if (in != null) {
          in.close();
          in = null;
       }
    }
```

```
    catch (IOException e) { in = null; }

    if (out != null) {
       out.close();
       out = null;
    }
 }
 ////////////////////////////////////////////////////////////////////
   public void run() {
    switch (connectionStatus) {
    case DISCONNECTED:
       connectButton.setEnabled(true);
       disconnectButton.setEnabled(false);
       ipField.setEnabled(true);
       portField.setEnabled(true);
       hostOption.setEnabled(true);
       guestOption.setEnabled(true);
       chatLine.setText(""); chatLine.setEnabled(false);
       statusColor.setBackground(Color.red);
       break;
    case DISCONNECTING:
       connectButton.setEnabled(false);
       disconnectButton.setEnabled(false);
       ipField.setEnabled(false);
       portField.setEnabled(false);
       hostOption.setEnabled(false);
       guestOption.setEnabled(false);
       chatLine.setEnabled(false);
       statusColor.setBackground(Color.orange);
       break;

    case CONNECTED:
       connectButton.setEnabled(false);
       disconnectButton.setEnabled(true);
       ipField.setEnabled(false);
       portField.setEnabled(false);
       hostOption.setEnabled(false);
       guestOption.setEnabled(false);
       chatLine.setEnabled(true);
       statusColor.setBackground(Color.green);
       break;
    case BEGIN_CONNECT:
       connectButton.setEnabled(false);
       disconnectButton.setEnabled(false);
       ipField.setEnabled(false);
       portField.setEnabled(false);
       hostOption.setEnabled(false);
       guestOption.setEnabled(false);
       chatLine.setEnabled(false);
       chatLine.grabFocus();
       statusColor.setBackground(Color.orange);
       break;
    }
    ipField.setText(hostIP);
    portField.setText((new Integer(port)).toString());
    hostOption.setSelected(isHost);
    guestOption.setSelected(!isHost);
```

```
   statusField.setText(statusString);
   chatText.append(toAppend.toString());
   toAppend.setLength(0);

   mainFrame.repaint();
}
////////////////////////////////////////////////////////////////
// hàm chính
public static void main(String args[]) {
   String s;
    initGUI();
    while (true) {
      try {
         Thread.sleep(10);
      }
      catch (InterruptedException e) {}

      switch (connectionStatus) {
      case BEGIN_CONNECT:
         try {
            // k?t n?i ??n host
            if (isHost) {
               hostServer = new ServerSocket(port);
               socket = hostServer.accept();
            }
             // k?t n?i ??n server
            else {
               socket = new Socket(hostIP, port);
            }
             in = new BufferedReader(new
               InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);
            changeStatusTS(CONNECTED, true);
         }
         // N?u l?i thì xu?t thông báo
         catch (IOException e) {
            cleanUp();
            changeStatusTS(DISCONNECTED, false);
         }
         break;

      case CONNECTED:
         try {
            // G?i d? li?u
            if (toSend.length() != 0) {
               out.print(toSend); out.flush();
               toSend.setLength(0);
               changeStatusTS(NULL, true);
            }
            // Nh?n d? li?u
            if (in.ready()) {
               s = in.readLine();
               if ((s != null) &&  (s.length() != 0)) {

                  if (s.equals(END_CHAT_SESSION)) {
                     changeStatusTS(DISCONNECTING, true);
                  }
```

```
                     // Nh?n tin
                    else {
                       appendToChatBox("INCOMING: " + s + "\n");
                       changeStatusTS(NULL, true);
                    }
                 }
              }
           }
           catch (IOException e) {
              cleanUp();
              changeStatusTS(DISCONNECTED, false);
           }
           break;

        case DISCONNECTING:
           //thông báo tr?ng thái ng?t k?t n?i
           out.print(END_CHAT_SESSION); out.flush();
            //xóa
           cleanUp();
           changeStatusTS(DISCONNECTED, true);
           break;

        default: break; // thoát
        }
     }
  }
}
//////////////////////////////////////////////////////////////////
class ActionAdapter implements ActionListener {
   public void actionPerformed(ActionEvent e) {}
}
```

---0O0---

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 06 Số giờ : 06 (2 buổi)

GVHD:Ths. Nguyễn Minh Nhật

LAB 06

LẬP TRÌNH RMI

I.MỤC TIÊU

- Tìm các kỹ thuật lập trinh phân tán với Java RMI
- Làm quen cách thức triển ứng dụng trên Java RMI

II. NỘI DUNG

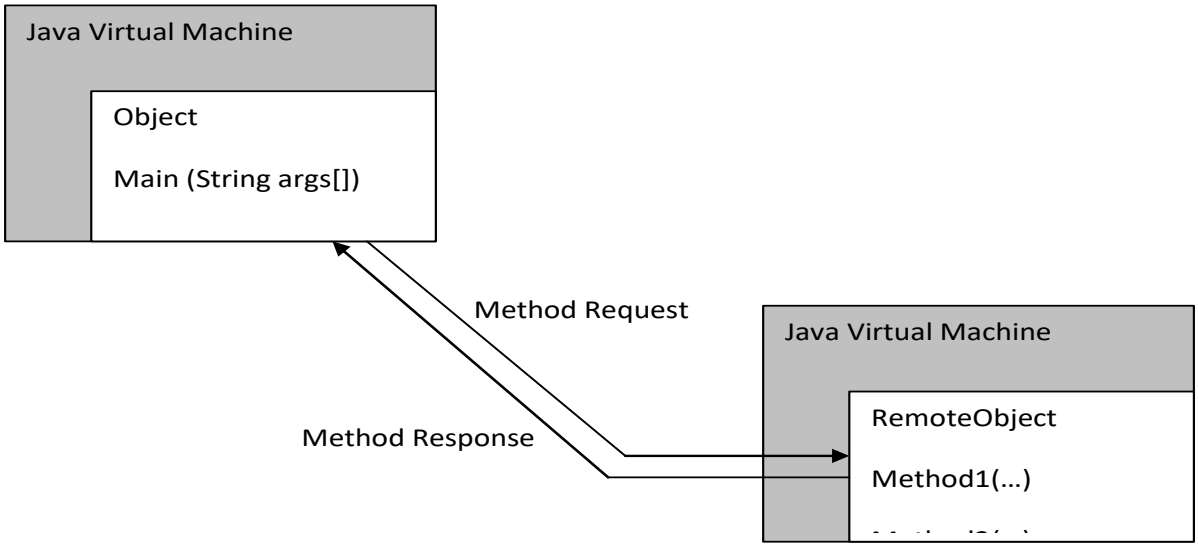
A.LÝ THUYẾT

1. Tổng quan về RMI

RMI là một công nghệ của các hệ thống phân tán nó cho phép một máy ảo Java (JVM) gọi những phương thức của đối tượng nằm trên máy ảo Java khác ở trong cùng một mạng.

1.1. RMI là gì

RMI là một công nghệ Java cho phép một JVM giao tiếp với một JVM khác và thi h



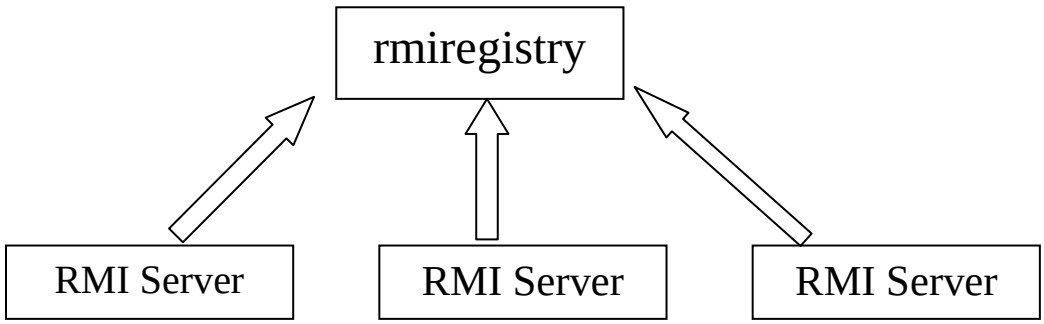
Hình 6.1. Lời triệu gọi từ xa được thi hành trên máy ảo Java ở xa

1.2. So sánh giữa RMI và RPC

Java là một ngôn ngữ nền tảng, dễ hiểu, nó cho phép những ứng dụng Java giao tiếp với những ứng dụng Java khác chạy trên bất kỳ môi trường phần cứng nào hỗ trợ một JVM. Sự khác nhau chủ yếu giữa RPC và RMI là RPC hỗ trợ nhiều ngôn ngữ, trong khi RMI chỉ hỗ trợ những ứng dụng viết trên Java.

1.3. RMI làm việc như thế nào?

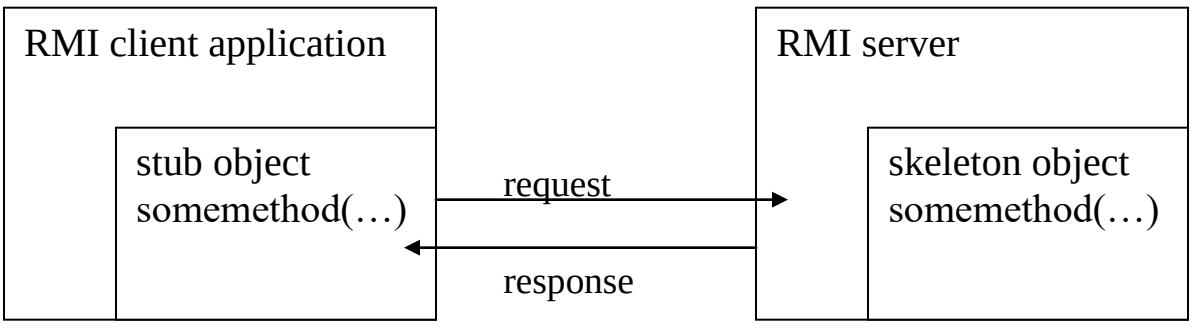
Các hệ thống sử dụng RMI cho việc truyền thông tiêu biểu được phân thành 2 loại: Clients và Servers. Server cung cấp dịch vụ RMI, và Client gọi các phương thức do Server cung cấp. RMI Server phải đăng ký một dịch vụ tìm kiếm, cho phép các Client tìm thấy thông tin Server cung cấp, hoặc chúng có thể tham chiếu tới dịch vụ khác. Một ứng dụng chạy nền cho RMI có tên là rmiregistry. Ứng dụng này chạy và xử lý độc lập với các chương trình RMI, nó cho phép các đối tượng trên Server đăng ký tên của mình. Mỗi lần một đối tượng được đăng ký xong, nó sẽ đợi sau đó thực hiện lời gọi từ phía Client.



Hình 6.2. Nhiều dịch vụ đăng ký với một bộ đăng ký

Các đối tượng trên Client sẽ gửi những thông điệp tới những phương thức ở xa. Trước khi một phương thức ở xa được thực thi Client phải có tham chiếu của nó trên Server. Điều đó được thực hiện bởi dịch vụ tìm kiếm trong bộ đăng ký RMI. Đối tượng trên Client yêu cầu một tên dịch vụ, và sẽ nhận được một URL. Nên nhớ những URL không phải cho HTTP, hầu hết các giao thức có thể đại diện sử dụng cú pháp của URL. Định dạng được sử dụng bởi RMI để đại diện cho một đối tượng tham chiếu từ xa như sau:

```
rmi://hostname:port/servicename
```



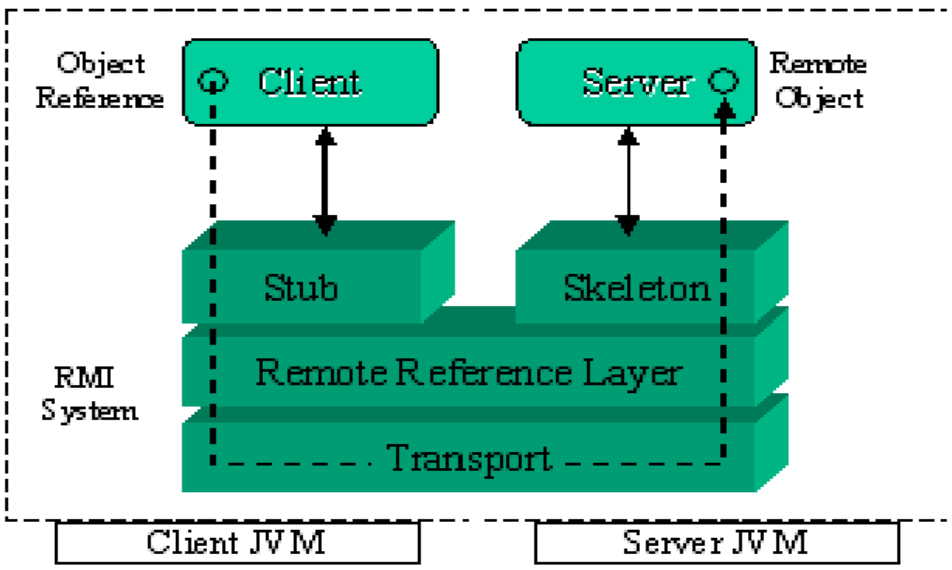
Hình 6.3. Đối tượng stub gọi đối tượng skeleton.

Trong đó hostname tên của của Server hoặc địa chỉ IP của Server, port số hiệu cổng cung cấp dịch vụ, servicename là một chuỗi mô tả dịch vụ. Những thông tin chi tiết của hoạt động mạng thì luôn trong suốt với người phát triển ứng dụng khi làm việc với các đối tượng ở xa, việc đó trở nên đơn giản như khi làm việc với đối tượng tại máy cục bộ. Điều này được thực hiện nhờ một phép chia thông minh của hệ thống RMI thành hai thành phần, một là stub và một là skeleton.

Tại RMI Server, đối tượng skeleton có nhiệm vụ lắng nghe những yêu cầu và chuyển các yêu cầu đó tới dịch vụ RMI. Sau khi người phát triển tạo ra một giao diện RMI, người đó còn phải định nghĩa cụ thể giao diện đó. Đối tượng được định nghĩa này sẽ được gọi là đối tượng skeleton.

1.4. Kiến trúc của chương trình RMI

Kiến trúc của một chương trình theo cơ chế RMI được mô tả như hình sau:



Hình 6.4. Kiến trúc chương trình kiểu RMI

Trong đó:

+ Server là chương trình cung cấp các đối tượng có thể được gọi từ xa.
+ Client là chương trình có tham chiếu đến các phương thức của các đối tượng ở xa trên Server.
+ Stub chứa các tham chiếu đến các phương thức ở xa trên Server.
+ Skeleton đón nhận các tham chiếu từ Stub để kích hoạt phương thức tương ứng trên Server.
+ Remote Reference Layer là hệ thống truyền thông của RMI.
+ Transport là tầng giao vận được dựa trên giao thức TCP/IP giữa các máy trong mạng.

Bằng cách sử dụng kiến trúc phân tầng như trên mà mỗi tầng có thể phân cấp hoặc thay thế mà không ảnh hưởng tới các tầng còn lại của hệ thống.

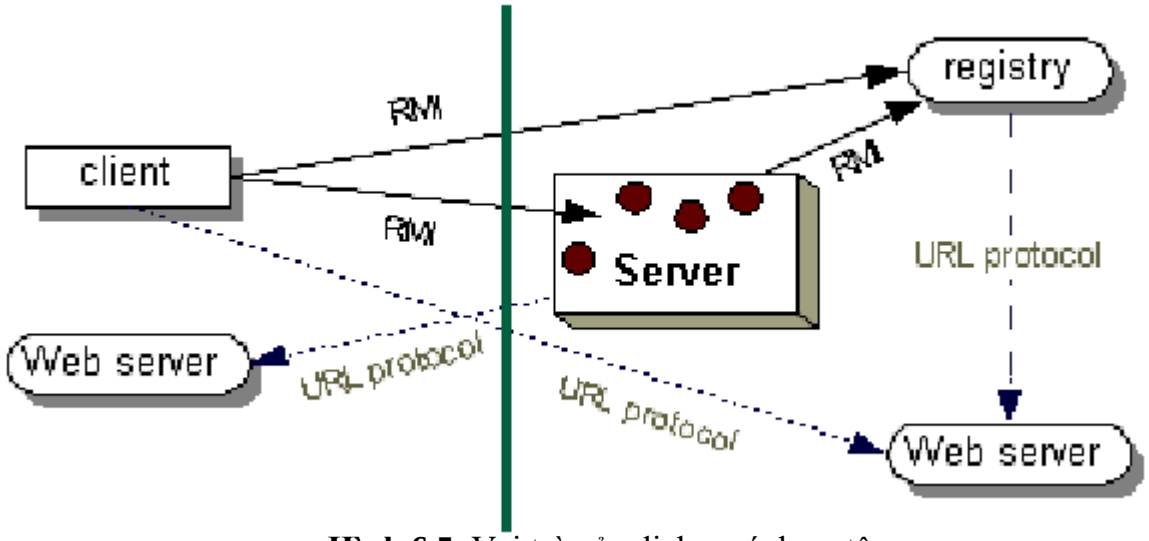
1.5. Các cơ chế liên quan trong một ứng dụng RMI

Trong một ứng dụng phân tán cần có các cơ chế sau:

+ Cơ chế định vị đối tượng ở xa (Locate remote objects)

+ Cơ chế giao tiếp với các đối tượng ở xa (Communicate with remote objects)

+ Tải các lớp danh bytecodes cho các lớp mà nó được chuyển tải qua lại giữa máy ảo (Load class bytecodes for objects that are passed around)



Hình 6.5. Vai trò của dịch vụ ánh xạ tên

Trong đó:

+ Server đăng ký tên cho đối tượng có thể được gọi từ xa của mình với dịch vụ ánh xạ tên (Registry Server).

+ Client tìm đối tượng ở xa thông qua tên đã đăng ký trên Registry Server (looks up) và tiếp đó gọi các phương thức ở xa.

+ Hình 3.5 cũng cho thấy cách thức mà hệ thống RMI sử dụng một WebServer sẵn có để truyền tải mã bytecodes của các lớp qua lại giữa Client và Server.

2. Cơ chế thực thi của một ứng dụng RMI

Tiến trình thực thi của một ứng dụng RMI diễn ra như sau:

Bước 1: Server tạo các đối tượng cho phép gọi từ xa cùng với Stub và Skeleton của chúng.

Bước 2: Server sử dụng lớp Naming để đăng ký tên cho một đối tượng từ xa (1).

Bước 3: Naming đăng ký Stub của đối tượng từ xa với Registry Server (2).

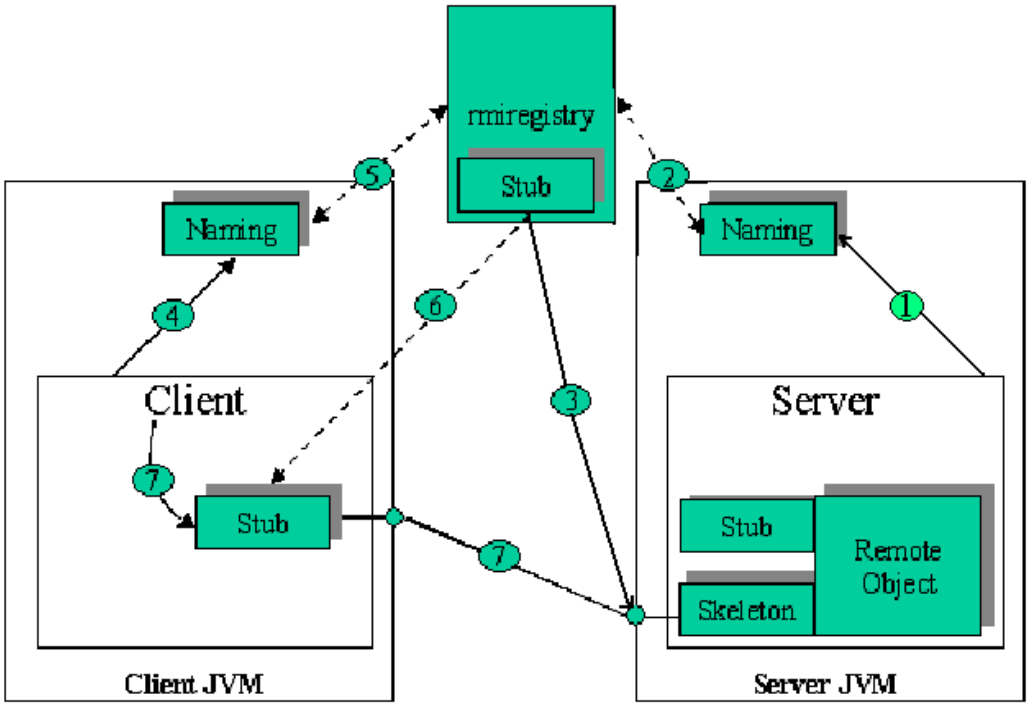
Bước 4: Registry Server sẵn sàng cung cấp tham chiếu đến đối tượng từ xa khi có yêu cầu (3).

+ Client yêu cầu Naming định vị đối tượng ở xa thông qua tên đã được đăng ký (phương thức lookup) với dịch vụ tên (4).

+ Naming tải Stub của đối tượng ở xa từ dịch vụ tên mà đối tượng đã đăng ký về Client (5).

+ Cài đặt đối tượng Stub và trả về tham chiếu đối tượng ở xa cho Client (6).

+ Client thực thi một lời gọi phương thức ở xa thông qua đối tượng Stub (7).



Hình 6.6. Tiến trình thực thi của ứng dụng RMI

2.1. Triển khai các ứng dụng với RMI

2.1.1. Các bước xây dựng chương trình RMI:

1. *Tạo 1 lớp giao diện.*Ví dụ: HelloInterface.java
2. *Tạo lớp hiện thực mô tả các phương thức của lớp giao diện.*
   Ví dụ:HelloImplement.java
3. *Xây dựng chương trình Server:*

- tạo đối tượng RemoteObject từ lớp Implement.
  đăng ký đối tượng với máy JVM :
  UnicastRemoteObject.exportObject(RemoteObject);
- đăng ký đối tượng với rmiregistry:Naming.bind("rmi://<IP address>/tên RemoteObject",RemoteObject);

4. *Xây dựng chương trình Client:*
   tạo một đối tượng Obj tham chiếu đến đối tượng từ xa thông qua:
   Naming.lookup("rmi../tênRemoteObject");
5. *Biên dịch tạolớp Stub,Skel:*
   rmic <tên lớp implement>
6. *Biên dịch chương trình Client,Server,..*
7. *Chạy chương trình:*
   - chạy rmiregistry
   - chạy server
   - chạy client

2.1.2.Các lớp, gói thường được sử dụng trong RMI

Trong kỹ thuật triệu gọi từ xa, một hệ thống RMI sử dụng rất nhiều gói và lớp của Java nhưng các lớp quan trọng nhất vẫn là:

+ java.rmi
+ java.rmi.activation
+ java.rmi.dgc
+ java.rmi.registry
+ java.rmi.server

Trong các gói trên thì quan trọng nhất là gói java.rmi với gói này lớp Naming đóng vai trò cực kỳ quan trọng, tất cả các phương thức của lớp này đều là phương thức tĩnh. Lớp này dùng để đăng ký hoặc khôi phục các tham chiếu đối tượng với bộ đăng ký rmiregistry. Gói này gồm các phương thức:

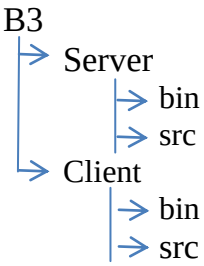
+ static void bind(String url, Remote Object).
+ static String[] list(String url.
+ static Remote lookup(String url).
+ static void rebind(String url, Remote Object).

B.BÀI TẬP

6.1. Client gởi Server 1 lời chào nhập từ bàn phím. Server trả lời.
Yêu cầu : Hiển thị lời chào của Server trên màn hình Client bằng kỹ thuật RMI.

6.2. Client gởi cho Server 1 số nguyên n từ bàn phím, yêu cầu Server yêu cầu thực hiện tính tổng : S = 1+2+ 3+...+ n
Yêu cầu : Hiển thị kết quả S trên màn hình Client bằng kỹ thuật RMI.

6.3. Lấy lại yêu cầu của bài **6.2**, tuy nhiên, khi chạy chương trình, ở mãy tính sẽ sinh ra thư mục B3 theo cấu trúc sau:



Trong đó :

- Thư mục *bin* của **Server** gồm các file sau :
  - CalculatorServer.class
  - Calculator.class
  - CalculatorImpl.class
  - CalculatorImpl_Stud.class
- Thư mục *bin* **Client** gồm các file sau :
  - CalculatorClient.class
  - Calculator.class

- CalculatorImpl.class
- CalculatorImpl_Stud.class

- Thư mục s**rc** chứa các file nguồn của java

6.5. Viết chương trình chat của 2 máy tính bằng kỹ thuật RMI. Quá trình chát sẽ dừng lại khi 1 trong 2 máy gõ vào từ " thoát ", không phân biệt từ hoa hay thường.

6.6. Viết chương trình sau theo kỹ thuật RMI, yêu cầu : Client gởi chuỗi bất kỳ cho Server nhập từ bàn phím (Ví dụ chuỗi S = " Đại học DuyTân "). Sau khi nhận được chuỗi này, Server thực hiện: Đếm số lần xuất hiện của các ký tự (Xét 2 trường hợp không phân biệt chữ hoa và phân biệt chữ hoa). Xuất kết quả trên màn hình Client.

6.7 Hệ thống chứng khoán được tổ chức tại 3 sàn giao dịch đặt tại các thành phố: Hà Nội, Sài Gòn và Đà Nẵng. Mỗi sàn giao dịch gồm 3 thông số: Ngày, giờ và chỉ số index của sàn. Viết chương trình hiển thị trên màn hình Client:

- Chỉ số chứng khoán của các sàn giao dịch nêu trên (Tên sàn, ngày, giờ và chỉ số Index của mỗi sàn) theo kỹ thuật RMI.

Hướng dẫn

1. Tạo lớp interface Hanoi
2. Tạo lớp HanoiImpl implements từ lớp interface Hanoi
3. Tạo lớp interface Time với phương thức triệu gọi về ngày, tháng, năm của Server kết nối
4. Tạo lớp TimeImpl implements từ lớp interface Time
5. Tạo lớp interface RateServer với phương thức triệu gọi về
+ Chỉ số Index tại sàn chứng khoán Hà Nội
+ Giờ tại sàn giao dịch Hà Nội
6. Tạo lớp RateServerImpl implements RateServer
7. Tạo lớp giao diện Server
8. Tạo lớp giao diện Client

6.8. Để thực hiện dãy tính: S = 25*(a+b) – 6*(3c- 2d), Client thực hiện gởi đến:

+ **Server 1**: các giá trị a và b, yêu cầu thực hiện tính tổng: a+b
+ **Server 2**: các giá trị c và d, yêu cầu thực hiện tính hiệu: 3c-2d

Yêu cầu:

Hiển thị kết quả của S trên màn hình Client theo kỹ thuật RMI trong các trường hợp sau :

1. Các giá trị a,b,c,d được nhập từ bàn phím trên cùng dòng, mỗi giá trị cách nhau bởi dấu ";"
2. Quá trình được thực hiện nhiều lần cho đến khi dòng gởi có dạng "stop" thì quá trình trên chấm dứt.

6.9. Client gởi cho Server 3 số a,b,c. Yêu cầu Server giải phương trình có dạng : $ax^2+bx+c = 0$.

Yêu cầu

Hiển thị nghiệm của phương trình trên ở màn hình Client trong các trường hợp sau :

a. a,b,c là 3 số ngẫu nhiên lấy từ đoạn [20,100].
b. a,b,c nhập từ bàn phím trên cùng 1 dòng cách nhau bởi dấu ";".

6.10. Client gởi cho Server 3 số a,b,c nhập từ bàn phím. Yêu cầu Server cho biết chúng có tạo thành tam giác hay không,nếu có thì đó là tam giác gì ?
Hiển thị nghiệm của phương trình trên ở màn hình Client trong các trường hợp sau :

a. a,b,c là 3 số ngẫu nhiên lấy từ đoạn [20,100].
b. a,b,c nhập từ bàn phím trên cùng 1 dòng cách nhau bởi dấu ";".

6.11. Client tạo một dãy Fibonacci F(n) với F(1) = a, F(2) = b, sau đó gởi F(1), F(2) và k(k>2) là số phần tử trong dãy cho Server. Tiếp đó, Client gởi một số nguyên p, yêu cầu Server cho biết p có tồn tại trong dãy không, nếu tồn tại thì nó ở số hạng thứ mấy trong dãy.

Yêu cầu :

Viết chương trình hiển thị kết quả trên màn hình Client theo kỹ thuật RMI trong các trường hợp sau :

1. Các giá trị a,b, k và p được nhập từ bàn phím trên cùng dòng, mỗi giá trị cách nhau bởi dấu ";"
2. Quá trình được thực hiện nhiều lần cho đến khi dòng gởi có dạng " stop" thì quá trình trên chấm dứt.
3. a,b,c,d, được nhập từ file input.txt (tạo sẵn trên máy) có dạng như sau :

File *Input.txt*

a	b	k	p
1	1	5	34
1	1	10	256
.....			
1	1	25	346

6.12. Viết 1 chương trình bằng kỹ thuật RMI, lấy ra thông tin của tất cả học viên có trong database (Giả sử học viên chỉ có id, name, address)

6.13. Xây dựng hệ thống quản lý đăng nhập từ xa sử dụng công nghệ RMI. Ngoài ra, hệ thống cho phép thay đổi pass word, user của khách hàng khi cần thiết,

HƯỚNG DẪN

6.1.

Tạo lớp giao tiếp

```
import java.rmi.*;
import java.math.BigInteger;

public interface Fibonacci extends Remote {

  public BigInteger getFibonacci(int n) throws RemoteException;
  public BigInteger getFibonacci(BigInteger n) throws RemoteException;
```

```
}
```

Tạo lớp tính toán

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.math.BigInteger;

public class FibonacciImpl extends UnicastRemoteObject implements Fibonacci {

  public FibonacciImpl() throws RemoteException {
    super();
  }

  public BigInteger getFibonacci(int n) throws RemoteException {
    return this.getFibonacci(new BigInteger(Long.toString(n)));
  }

  public BigInteger getFibonacci(BigInteger n) throws RemoteException {

    System.out.println("Calculating the " + n + "th Fibonacci number");
    BigInteger zero = new BigInteger("0");
    BigInteger one  = new BigInteger("1");

    if (n.equals(zero)) return one;
    if (n.equals(one)) return one;

    BigInteger i  = one;
    BigInteger low  = one;
    BigInteger high  = one;

    while (i.compareTo(n) == -1) {
      BigInteger temp = high;
      high = high.add(low);
      low = temp;
      i = i.add(one);
    }

    return high;

  }
}
```

Tạo lớp FibonacciServer

```
import java.net.*;
import java.rmi.*;

public class FibonacciServer {

  public static void main(String[] args) {

    try {
      FibonacciImpl f = new FibonacciImpl();
      Naming.rebind("fibonacci", f);
      System.out.println("Fibonacci Server ready.");
```

```
    }
     catch (RemoteException rex) {
      System.out.println("Exception in FibonacciImpl.main: " + rex);
    }
    catch (MalformedURLException ex) {
      System.out.println("MalformedURLException " + ex);
    }

  }
```

}

Tạo lớp FibonacciClient

```
import java.rmi.*;
import java.net.*;
import java.math.BigInteger;

public class FibonacciClient {

  public static void main(String args[]) {

    if (args.length == 0 || !args[0].startsWith("rmi:")) {
      System.err.println(
        "Usage: java FibonacciClient rmi://host.domain:port/fibonacci number");
      return;
    }

    try {
      Object o = Naming.lookup(args[0]);
      Fibonacci calculator = (Fibonacci) o;
      for (int i = 1; i < args.length; i++) {
        try {
          BigInteger index = new BigInteger(args[i]);
          BigInteger f = calculator.getFibonacci(index);
          System.out.println("The " + args[i] + "th Fibonacci number is "
           + f);
        }
        catch (NumberFormatException e) {
          System.err.println(args[i] + "is not an integer.");
        }
      }
    }
    catch (MalformedURLException ex) {
      System.err.println(args[0] + " is not a valid RMI URL");
    }
    catch (RemoteException ex) {
      System.err.println("Remote object threw exception " + ex);
    }
    catch (NotBoundException ex) {
      System.err.println(
       "Could not find the requested remote object on the server");
    }
  }
}
```

6.2.
Tạo lớp TinhToanInterface.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface TinhToanInterface extends Remote {
    public int tinhDienTich(int cr,int cc) throws RemoteException;
    public int tinhLuyThua(int cs,int sm) throws RemoteException;
}
```

Tạo lớp TinhToanServer.java

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
public class TinhToanServer extends UnicastRemoteObject implements
TinhToanInterface
{

    public TinhToanServer() throws RemoteException
    {

    }
    public int tinhDienTich(int cr, int cc) throws RemoteException
    {
        return cr*cc;
    }

    public int tinhLuyThua(int cs, int sm) throws RemoteException
    {
        int kq = 1;

        for(int i = 1; i <= sm;i++)
        {
            kq = kq*cs;
        }

        return kq;
    }

    public static void main(String args[])
    {

        try {
            TinhToanServer objServer = new TinhToanServer();
            Registry rg = LocateRegistry.createRegistry(123);
            try {
                Naming.rebind("rmi://localhost:123/tinhtoan",objServer);
            } catch (RemoteException ex) {
                ex.printStackTrace();
            } catch (MalformedURLException ex) {
                ex.printStackTrace();
```

```
            }
        } catch (RemoteException ex) {
            ex.printStackTrace();
        }
    }
}
```

Tạo lớp TinhToanClient.java

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class TinhToanClient extends javax.swing.JFrame {

    TinhToanInterface objProxyOfServer;

    /** Creates new form TinhToanClient */
    public TinhToanClient() {
        initComponents();
        try {
            objProxyOfServer = (TinhToanInterface)
Naming.lookup("rmi://localhost:123/tinhtoan");
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (RemoteException ex) {
            ex.printStackTrace();
        } catch (NotBoundException ex) {
            ex.printStackTrace();
        }

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-
BEGIN:initComponents
    private void initComponents() {
        lbValue1 = new javax.swing.JLabel();
        lbValue2 = new javax.swing.JLabel();
        lbResult = new javax.swing.JLabel();
        tfValue1 = new javax.swing.JTextField();
        tfValue2 = new javax.swing.JTextField();
        tfResult = new javax.swing.JTextField();
        btnSquare = new javax.swing.JButton();
        btnPower = new javax.swing.JButton();
        btnReset = new javax.swing.JButton();
        btnExit = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        lbValue1.setText("Value 1:");

        lbValue2.setText("Value 2:");
```

```
        lbResult.setText("Result");

        tfResult.setEditable(false);

        btnSquare.setText("square");
        btnSquare.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnSquareActionPerformed(evt);
            }
        });

        btnPower.setText("power");
        btnPower.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnPowerActionPerformed(evt);
            }
        });

        btnReset.setText("Reset");
        btnReset.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnResetActionPerformed(evt);
            }
        });

        btnExit.setText("Exit");
        btnExit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnExitActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(layout.createSequentialGroup()
                        .add(59, 59, 59)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                            .add(lbResult)
                            .add(lbValue2)
                            .add(lbValue1))
                        .add(28, 28, 28)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
                            .add(tfResult)
                            .add(tfValue2)
                            .add(tfValue1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 180, Short.MAX_VALUE)
                            .add(layout.createSequentialGroup()
                                .add(btnReset)
```

```
.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .add(btnExit)
                                .add(13, 13, 13))))
                    .add(layout.createSequentialGroup()
                        .add(134, 134, 134)
                        .add(btnSquare)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(btnPower)))
                .addContainerGap(94, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup()
                .add(51, 51, 51)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(lbValue1)
                    .add(tfValue1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(26, 26, 26)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(lbValue2)
                    .add(tfValue2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(22, 22, 22)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(btnPower)
                    .add(btnSquare))
                .add(29, 29, 29)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(lbResult)
                    .add(tfResult,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(29, 29, 29)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(btnReset)
                    .add(btnExit))
                .addContainerGap(37, Short.MAX_VALUE))
        );
        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void btnPowerActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnPowerActionPerformed
```

```
// TODO add your handling code here:
        String strValue1 = tfValue1.getText().trim();
        String strValue2 = tfValue2.getText().trim();

        int iValue1 = Integer.parseInt(strValue1);
        int iValue2 = Integer.parseInt(strValue2);
        try {

            int lt = objProxyOfServer.tinhLuyThua(iValue1,iValue2);
            String strResult = lt + "";
            tfResult.setText(strResult);

        } catch (RemoteException ex) {
            ex.printStackTrace();
        }

    }//GEN-LAST:event_btnPowerActionPerformed

    private void btnSquareActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnSquareActionPerformed
// TODO add your handling code here:
        String strValue1 = tfValue1.getText().trim();
        String strValue2 = tfValue2.getText().trim();

        int iValue1 = Integer.parseInt(strValue1);
        int iValue2 = Integer.parseInt(strValue2);
        try {

            int dt = objProxyOfServer.tinhDienTich(iValue1,iValue2);
            String strResult = dt + "";
            tfResult.setText(strResult);

        } catch (RemoteException ex) {
            ex.printStackTrace();
        }


    }//GEN-LAST:event_btnSquareActionPerformed

    private void btnResetActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnResetActionPerformed
// TODO add your handling code here:
        tfValue1.setText("");
        tfValue2.setText("");
    }//GEN-LAST:event_btnResetActionPerformed

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnExitActionPerformed
// TODO add your handling code here:
        System.exit(0);
    }//GEN-LAST:event_btnExitActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
            public void run() {
                new TinhToanClient().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnExit;
    private javax.swing.JButton btnPower;
    private javax.swing.JButton btnReset;
    private javax.swing.JButton btnSquare;
    private javax.swing.JLabel lbResult;
    private javax.swing.JLabel lbValue1;
    private javax.swing.JLabel lbValue2;
    private javax.swing.JTextField tfResult;
    private javax.swing.JTextField tfValue1;
    private javax.swing.JTextField tfValue2;
    // End of variables declaration//GEN-END:variables

}
```

6.4.

Bước 1 : thiết kế lớp student.class được dùng làm tham số chuyển qua mạng giữa client và server.

```
Public class student implements Serializable {
Public String ID;
Public String name;
Public String address;
}
```

Lưu ý: *Mục đích là sử dụng đối tượng student để di chuyển giữa client và server , cho nên lớp student phải khai báo Serializable (bạn không cần phải hiểu Serializable là gì ,mà chỉ cần biết đối tượng muốn chuyển được qua mạng thì phải được khai báo Serializable).*

Bước 2: Tạo ra 1 remote interface chứa các phương thức được gọi bởi client (client giao tiếp với server và gọi các phương thức được khai báo trong interface này)

```
Public inteface manageStudent extends Remote {
// lấy ra tất cả học viên
Public Vector getAllStudent() throws RemoteException;
}
```

Lưu ý : *interface phải khai báo public và các phương thức phải ném ra RemoteException;*

Bước 3 : Định nghĩa các phương thức trong interface managerStudent

```
import java.rmi.RemoteException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Vector;
```

```
public class manageStudentImpl implements manageStudent {

Connection con;

ResultSet result;

PreparedStatement stmt;

public Vector getAllStudent() throws RemoteException {
try {
Vector v = new Vector();

Class.forName("com.mysql.jdbc.Driver");

// giả sử có 1 database trong Sql Server tên students – trong đó có bảng
student và cài đặt ODBC với tên students
con = DriverManager.getConnection("jdbc:odbc:students", "sa","");

stmt = con.prepareStatement("select * from student");
result = stmt.executeQuery();

while (result.next()) {

student info = new student();
info.ID = result.getString(1);
info.name = result.getString(2);
info.address = result.getString(3);

v.addElement(info);
}
result.close();
con.close();
return v;

} catch (Exception e) {
// TODO Auto-generated catch block
System.out.println("\n Loi lấy tất cả thông tin học viên :" + e.getMessage());
}
return null;
}
}
```

Bước 4 : Thiết kế chương trình cài đặt đối tượng manageStudentImpl trên Server

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class SetupServer {
public static void main(String[] args) {

try {
manageStudent server = new manageStudentImpl();
```

//thông báo khả năng giao tiếp được từ xa của đối tượng server với máy ảo java

```
UnicastRemoteObject.exportObject(server);
```

```
//đăng ký đối tượng server với dịch vụ rmiregistry với tên students
Naming.rebind("rmi://127.0.0.1/students", server);

System.out.println("Server began....");
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Bước 5 : Thiết kế chương trình cài đặt client trên các máy client gọi đến server

```
import java.rmi.Naming;
import java.util.Vector;

public class SetupClient {

public static void main(String[] args) {

try {
// yêu cầu rmiregistry truy tìm đối tượng manageStduent
manageStudent server = (manageStudent)
Naming.lookup("rmi://127.0.0.1/students");

// gọi đến server lấy về tất cả thông tin học viên từ server
Vector v = server.getAllStudent();

for(int i=0;i<v.size();i++) {

      student st = (student)v.elementAt(i);
System.out.println( st.ID + " , " + st.name + " , " + st.address);
}

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

Để chạy được chương trình :

- trước tiên, biên dịch các file ra thành file class.
- Chạy câu lệnh : rmic manageStudentImpl
- Tiếp tục chạy câu lệnh : start rmiregistry
- Chạy : java SetupServer
- Chạy : java SetutpClient

6.10

Hướng dẫn

1. Tạo lớp interface Hanoi

2. Tạo lớp HanoiImpl implements từ lớp interface Hanoi

3. Tạo lớp interface Time với phương thức triệu gọi về ngày, tháng, năm của Server kết nối
4. Tạo lớp TimeImpl implements từ lớp interface Time
5. Tạo lớp interface RateServer với phương thức triệu gọi về
+ Chỉ số Index tại sàn chứng khoán Hà Nội
+ Giờ tại sàn giao dịch Hà Nội
6. Tạo lớp RateServerImpl implements RateServer
7. Tạo lớp giao diện Server
8. Tạo lớp giao diện Client

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."

TRƯỜNG ĐẠI HỌC DUY TÂN KHOA CÔNG NGHỆ THÔNG TIN BỘ MÔN KỸ THUẬT MẠNG	**HỆ PHÂN TÁN** **(Distributed System)**	**BÀI THỰC HÀNH** LAB số : 07 Số giờ : 06 (2 buổi)

GVHD :Ths.Nguyễn Minh Nhật

LAB 07

LẬP TRÌNH ỨNG DỤNG PHÂN TÁN

I.MỤC TIÊU

- Tìm hiểu các bài toán ứng dụng phân tán tiêu biểu như : bài toán bãi đỗ xe, bài toán người sản xuất, người tiêu thu v.v..
- Ứng dụng các nguyên lý về vòng tròn ảo, bầu cử, tranh chấp, tạo bản sao v.v…để giải quyết

II. NỘI DUNG

A.LÝ THUYẾT

Sinh viên xem lại chương 7 của bài giảng *" Hệ thống phân tán"* do giáo viên cung cấp.

B. BÀI TẬP

Yêu cầu :

Để thực hiện, mỗi lớp chia thành các nhóm. Mỗi nhóm từ 4-5 sinh viên sẽ bốc xăm để chọn đề tài. Sau đó, các nhóm tiến hành thực hiện trong vòng 6 tuần. Các nhóm sẽ báo cáo kết quả thực hiện tại lớp trong vòng 2 buổi, theo lịch trình của GVHD bố trí.

1.Kịch bản 01(i)

Xây dựng 3 server kết nối với nhau theo nguyên lý vòng tròn ảo (Java). Yêu cầu của bài toán:

- Khi Server(i) nhận thông điệp từ Server(j) nó sẽ gửi thông điệp đến Server(i+1), đồng thời gửi thông tin phản hồi lại cho Server(j) là đã nhận được thông điệp.
- Xây dựng chương trình Client cho phép đăng ký tài nguyên và nhận phản hồi khi hoàn tất giao dịch, chương trình tự động đăng ký tài nguyên với số lần đăng ký là 30.
- Trong màn hình Server hiển thị nội dung của các thông điệp đến, thông điệp gửi đi và nội dung các thông điệp.

2.Kịch bản 02(ii)

Xây dựng 3 server kết nối với nhau theo phương thức multicast.
Yêu cầu của bài toán :

1. Khi Serveri nhận thông điệp từ Client, nó sẽ chuyển thông điệp đến tất cả các Server còn lại trong hệ, sau khi nhận phản hồi từ các Server này, dữ liệu sẽ được chuyển đến tất cả các Server cho đến khi kết thúc thông tin.
   Xây dựng chương trình Client cho phép chuyển tài nguyên và nhận phản hồi khi hoàn tất tài nguyên.

2.Trong màn hình Server/Client hiển thị thông tin tài nguyên.

Bài tập 7.1

Từ (i). Dựa vào sự di chuyển thông điệp trên các server, các anh chị hãy xây dựng trật tự toàn phần các thông điệp dựa trên thuật toán của Lamport cho phép loại trừ tương hỗ nhờ dấu. Hệ thống có 4 vòng tròn khẳng định, vòng tròn đầu tiên để lấy giá trị Lamport sẽ thiết kế chương trình theo đa luồng (MultiThread) để thiết kế đồng hồ chung. Các giá trị trong trường điều khiển tại giá trị của Lamport phải được thường xuyên cập nhật khi di chuyển trong vòng tròn.

Bài tập 7.2

Từ (i). chuyển trên vòng tròn sẽ thực hiện và đảm nhiệm một nhiệm vụ cụ thể.

Vòng 1, khóa trường dữ liệu.
Vòng 2, tạo bảng tạm.
Vòng 3, cập nhật bảng chính.
Vòng 4, kiểm tra đồng bộ tiến trình.

Viết chương trình giám sát màn hình để kiểm tra trạng thái của việc sản xuất và tiêu thụ. Xây dựng chương trình mô phỏng bãi đỗ xe Ô tô.(Sinh viên tự tìm hiểu thêm về mô hình bài toán bãi đỗ xe)

Bài tập 7.3

Từ (i). Trên cơ sở kiến thức về công tơ sự kiện, các anh chị hãy viết chương trình mô phỏng xác định tập các sự kiện có trước qua bài toán người sản xuất - người tiêu thụ.
Mỗi vòng di

Bài tập 7.4

Từ (i). Trên cơ sở kiến thức về đăng ký tour du lịch, các anh chị hãy viết chương trình mô phỏng quá trình đăng ký và xác nhận đăng ký tour du lịch qua phương thức giao dịch lồng. (Nguyên lý các hệ cơ sở dữ liệu phân tán)
Mỗi thông tin đăng ký trên 1 Server đảm nhiệm một nhiệm vụ cụ thể, thiết kế 4 Server với thông tin, dữ liệu vê tour du lịch :

Server 1 : thông tin về các hành trình.
Server 2 : thông tin phòng khách sạn.
Server 3 : thông tin về dịch vụ xe.
Server 4 : các dịch vụ ăn uống, mua sắm.

Bài tập 7.5

Từ (i)/(ii). Các anh chị hãy viết chương trình đăng ký (mượn/trả) sách trong thư viện.

Yêu cầu:
1. Xây dựng giao diện kiểm soát các thông điệp được di chuyển trên tập server.
2. Đưa ra trật tự tổng quát các thông điệp di chuyển trong vòng tròn ảo.
3. Giao diện thể hiện đăng ký sách.

Bài tập 7.6

Từ (i)/(ii), Các anh chị hãy viết chương trình đăng ký vé tàu trực tuyến. Trong đó có sử dụng nội dung thông điệp để cập nhật và giá trị đồng hồ Lamport để tránh trường hợp tương tranh (đăng ký trùng tài nguyên)

Yêu cầu:
1. Xây dựng giao diện kiểm soát các thông điệp được di chuyển trên tập server.
2. Đưa ra trật tự tổng quát các thông điệp di chuyển trong vòng tròn ảo.
3. Giao diện thể hiện đăng ký vé tàu từ client.

Bài tập 7.7

Từ (i)/(ii). Các anh chị hãy viết chương trình đăng ký tín chỉ.

Yêu cầu:
1. Xây dựng giao diện kiểm soát các thông điệp được di chuyển trên tập server.
2. Đưa ra trật tự tổng quát các thông điệp di chuyển trong vòng tròn ảo.
3. Giao diện thể hiện đăng ký tín chỉ.

Bài tập 7.8

Từ (ii). Các anh chị hãy viết chương trình truyền file theo cơ chế sau :
Dữ liệu của file từ Client sẽ được đưa vào ma trận (nxm), mỗi hàng sẽ là một vector có hướng và chuyển đến các Server trong tập đa Server.
Một Client bất kỳ nhận file từ một Server bất kỳ sẽ nhận thông tin ban đầu về file và tiếp nhận dữ liệu từ các Server trong hệ.

Bài tập 7.9

Từ (i). Với một bộ tuần tự tuần hoàn trên vòng tròn ảo sẽ đặt ra nhiều vấn đề khi bị sự cố ở một trạm nào đó.

Trạm bị sự cố không thể tự phục vụ các số mà nó đã rút được. Trong trường hợp đó, cần phải có một giải thuật có tính chất mặc định để sắp xếp lại.Khi một trạm có Jeton lại bị sự cố, nó giữ luôn jeton đó. Người hàng xóm bên phải phải tái sinh jeton mới. Nếu trạm sự cố đã rút số, phép toán này là nguyên nhân lờ đi người bên phải. Hay trạm trước khi sự cố đã có thể phục vụ số được rút.

Một trạm vào lại trong mạng cần phải tìm một giá trị thích hợp của jeton. Các anh chị hãy viết chương trình xử lý sự cố khi một server bị sự cố rời khỏi hệ thống mạng và yêu cầu vào lại vòng tròn sau khi đã khắc phục xong.

---o0o---

"Mọi sự thành công của ngày hôm nay là sự nổ lực từ ngày hôm qua."