

MỤC LỤC

MỤC LỤC	1
DANH MỤC THUẬT NGỮ TIẾNG ANH	3
DANH MỤC CHỮ VIẾT TẮT	5
MỤC LỤC HÌNH	7
LỜI NÓI ĐẦU	9
LAB 1. LẬP TRÌNH CƠ BẢN [1,2].....	11
LAB 2. LỚP - ĐỐI TƯỢNG - KẾ THỪA [1, 2, 7]	21
LAB 3. GIAO DIỆN, LỚP TRÙU TƯỢNG [7,13]	40
LAB 4. ARRAYLIST, LINKLIST, COLLECTION, GENERIC [1,7,13]	60
LAB 5. QUẢN LÝ THREAD [11,12]	82
LAB 6. LẬP TRÌNH CƠ SỞ DỮ LIỆU JDBC [3,5,8].....	106
LAB 7. ĐỊA CHỈ IP, GIAO TIẾP MẠNG (NIC)[6].....	144
LAB 8. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG TCP [4,6,9]	155
LAB 9. LẬP TRÌNH SERVER PHỤC VỤ NHIỀU CLIENT [4,6,10]	167
LAB 10. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG UDP [4, 6, 9,10].....	185
LAB 11. LẬP TRÌNH MULTICAST, URL, MAIL [4]	208
LAB 12. LẬP TRÌNH PHÂN TÁN VỚI RMI [5].....	222
TÀI LIỆU THAM KHẢO	247

DANH MỤC THUẬT NGỮ TIẾNG ANH

Từ	Nghĩa của từ
abstract	Trùu tượng
break	Dừng vòng lặp
catch	Từ khóa đầu của một khối bắt ngoại lệ
continue	Bỏ qua phần cuối vòng lặp, tiếp tục sang bước tiếp theo
default	Giá trị mặc định của phương thức switch()
extends	Ké thừa
final	Một hằng số, phương thức hay một lớp không được ghi đè
finally	Một phần của khối xử lý ngoại lệ try luôn được thực hiện
implements	Thực hiện giao diện
import	Khai báo một gói thư viện
instanceof	Kiểm tra một đối tượng là một thể hiện của lớp
interface	Giao diện
new	Tạo một đối tượng mới của lớp
null	Tham chiếu rỗng
package	Gói
private	Tiền tố chỉ được truy cập bởi phương thức của lớp
protected	Tiền tố được truy cập bởi phương thức của lớp, lớp con của và các lớp khác trong cùng một gói
public	Tiền tố có thể được truy cập bởi phương thức của tất cả các lớp
return	Trả về của một phương thức
super	Gọi phương thức của lớp cha
synchronized	Đồng bộ
this	Tham chiếu đến đối tượng hiện tại

DANH MỤC CHỮ VIẾT TẮT

Chữ viết tắt	Ý nghĩa
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
URL	Uniform Resource Locator
CSDL	Cơ Sở Dữ Liệu
JDBC	Java Database Connectivity
CNTT	Công Nghệ Thông Tin
HĐH	Hệ Điều Hành
MVC	Model-View-Control
DNS	Domain Name System
API	Application Programming Interface
FTP	File Transfer Protocol
JDK	Java Development Kit
GB	GigaByte
UCLN	Ước Chung Lớn Nhất
BCNN	Bội Chung Nhỏ Nhất
RAM	Random Access Memory
RMI	Remote Method Invocation
JVM	Java Virtual Machine
NIC	Network Interface Card
ĐH KTKT CN	Đại học Kinh tế Kỹ thuật Công nghiệp

MỤC LỤC HÌNH

Hình 1. Chèn code tự động	22
Hình 2. Tạo interface	40
Hình 3. Màn hình giao dịch ngân hàng	98
Hình 4. Thêm thư viện MYSQL JDBC Driver	107
Hình 5. Download JDBC Driver cho SQL	109
Hình 6. Một số màn hình quản lý thông tin Sinh viên	111
Hình 7. Một số màn hình quản lý Thư viện	143
Hình 8. Địa chỉ socket	156
Hình 9. Mô hình Client-Server chế độ hướng kết nối	163
Hình 10. Mô hình Client Server ở chế độ không kết nối.....	186
Hình 11. Tuần tự các bước thực hiện theo giao thức UDP	198
Hình 12. Sơ đồ lớp phía Client	199
Hình 13. Sơ đồ lớp phía Server	200
Hình 14. Các thành phần của URL	215
Hình 15. Kiến trúc RMI	222

LỜI NÓI ĐẦU

Ngày nay do nhu cầu thực tế và do sự phát triển mạnh mẽ của nhiều công nghệ tích hợp, dẫn đến các chương trình ứng dụng hầu hết đều có khả năng thực hiện trên môi trường mạng. Ngôn ngữ JAVA là ngôn ngữ phù hợp để viết các ứng dụng mạng. So với lập trình thông thường, lập trình mạng đòi hỏi người lập trình hiểu biết và có kỹ năng tốt để viết các chương trình giao tiếp và trao đổi dữ liệu giữa các máy tính với nhau.

Để hỗ trợ sinh viên chuyên ngành CNTT trong nhà trường tiếp cận với kỹ thuật lập trình mới này, tiếp theo cuốn tài liệu học tập lý thuyết “**Công nghệ JAVA**”, chúng tôi xây dựng cuốn “**Bài tập lập trình mạng**”, nhằm cung cấp cho sinh viên những kiến thức và kỹ thuật cơ bản nhất để phát triển các chương trình ứng dụng mạng, thông qua các dạng bài tập từ cơ bản đến nâng cao qua các chủ đề: lập trình cơ bản, lập trình hướng đối tượng, lập trình CSDL JDBC, lập trình mạng dùng socket, lập trình phân tán với RMI. Sinh viên sẽ thực hiện các bài thực hành này trên phòng máy nhà trường.

Nội dung cuốn tài liệu bao gồm 12 bài lab chia thành các chủ đề khác nhau. Trong mỗi chủ đề chúng tôi đưa ra tóm tắt lý thuyết, bài tập mẫu, sau đó là bài tập tương tự, và bài tập tổng hợp. Kết quả qua những bài lab, sinh viên được rèn và thành thạo các kỹ năng lập trình hướng đối tượng, lập trình CSDL, lập trình với giao thức truyền thông có sẵn và khả năng tích hợp trong các ứng dụng khác nhau, nhất là các giao thức truyền thông thời gian thực, từ đó sinh viên có thể viết được các phần mềm quản lý theo mô hình MVC, xây dựng được các ứng dụng mạng, các ứng dụng tích hợp và triệu gọi lẩn nhau trên mạng Intranet (mạng cục bộ), mạng Internet (mạng toàn cầu), các hệ thống xử lý truy xuất dữ liệu phân tán hoàn chỉnh. Nội dung biên soạn phù hợp với chuẩn đầu ra của ngành CNTT và ngành mạng máy tính và truyền thông dữ liệu về kỹ năng và kiến thức. Sau khi học xong học phần này sinh viên có thể viết phần mềm quản lý, truyền thông.

Chúng tôi xin chân thành cảm ơn Thầy Nguyễn Hoàng Chiến, phó chủ nhiệm khoa, phụ trách khoa CNTT trường ĐH KTKT CN cùng với các đồng nghiệp đã đóng góp ý kiến cho cuốn tài liệu này. Vì tài liệu được biên soạn lần đầu, chúng tôi đã cố gắng hoàn chỉnh, song không tránh khỏi thiếu sót. Rất mong nhận được sự góp ý của bạn đọc để tài liệu học tập được hoàn thiện hơn.

Xin trân trọng cảm ơn!

Nhóm tác giả

LAB 1. LẬP TRÌNH CƠ BẢN [1, 2]

A. MỤC TIÊU

Trang bị cho sinh viên kỹ năng lập trình cơ bản trong java: câu trúc **if**, vòng lặp **while...do, for..**, khai báo mảng, xử lý trên mảng.

B. NỘI DUNG

- Lệnh rẽ nhánh (**if**), lệnh lựa chọn (**switch**).
- Lệnh lặp, sử dụng mảng để thao tác trên các phần tử cùng kiểu.

C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HDH windows, RAM tối thiểu 1 GB.
- Phần mềm NETBEAN IDE 8.0, JDK 1.8.

D. KẾT QUẢ SAU KHI HOÀN THÀNH

Sinh viên thành thạo các câu lệnh : rẽ nhánh, lựa chọn, lệnh lặp, thao tác trên mảng, áp dụng giải các bài tập từ đơn giản đến phức tạp.

E. HƯỚNG DẪN CHI TIẾT

1. Câu trúc rẽ nhánh if...else

```
if (boolean_expr){ //Nếu boolean_expression là true
    //do task1
}
else { //Nếu boolean_expression là false
    //do task2
}
```

Bài 1. Tìm max, min của 2 số nguyên nhập vào từ bàn phím

Hướng dẫn:

```
public class Bai1 {
    public static void main(String[] args) {
        int a,b,max,min;
        Scanner sc = new Scanner(System.in); //Phím tắt CTRL + Space
        System.out.println("Nhập số nguyên a:");
        a = sc.nextInt();
        System.out.println("Nhập số nguyên b:");
        b = sc.nextInt();
        max = (a>b)?a:b;
        min = (a>b)?b:a;
    }
}
```

```

        System.out.println("Max la:"+max);
        System.out.println("Min la:"+min);
    }
}

```

Bài 2. Giải phương trình bậc 2 $ax^2 + bx + c = 0$

Hướng dẫn:

Sử dụng Scanner nhập 3 số nguyên a, b, c. Tính delta=b*b-4*a*c.

Sử dụng **if** kiểm tra từng trường hợp của delta.

```

if(delta<0)
    System.out.println("PT vo nghiem");
else if(delta==0){
    float x = (float) -b/(2*a);
    System.out.printf("PT co nghiem kep x1=x2=%,.2f",x);
}else{
    float x1 = (float) (-b-Math.sqrt(delta))/(2*a);
    float x2 = (float) (-b+Math.sqrt(delta))/(2*a);
    System.out.printf("PT co hai nghiem x1=%,.2f, x2=%,.2f",x1,x2);
}

```

2. Cấu trúc switch

```

switch (controlling_expr){
    case value1: //do task1
    break;
    case value2: //do task2
    break;
    ...
    case valueN: //do taskN
    break;
    default:
}

```

Bài 3. Viết chương trình java cho phép tạo và thực hiện theo menu sau:

1. Nhập vào một số nguyên dương n.
2. Tính tổng các số từ 1 đến n
3. Kiểm tra n có là số nguyên tố
4. Kiểm tra n có là số hoàn hảo.
5. Hiển thị số n thành tích các thừa số nguyên tố.

6. Thoát

(Hiển thị 1 số nguyên dương thành tích các thừa số nguyên tố: $n = 2^3 \times 3$)

Hướng dẫn:

Phương thức nhập: Dùng Scanner

Phương thức kiểm tra hoàn hảo

```
boolean sohh(int a){  
    int tong=0;  
    for (int i = 1; i < a; i++) {  
        if(a%i==0)  
            tong+=i;  
    }  
    if(tong==a)  
        return true;  
    else  
        return false;  
}
```

Phương thức hiển thị số hoàn hảo

```
public void hienThisHH(int a){  
    for (int i = 0; i < a; i++) {  
        if(sohh(i)==true)  
            System.out.print(i+" ");  
    }  
}
```

Phương thức kiểm tra nguyên tố

```
boolean songt(int a){  
    if (a < 2)  
        return false;  
    for (int i = 2; i <= Math.sqrt(a); i ++)  
    {  
        if (a%i==0)  
        {  
            return false;  
        }  
    }
```

```
    }  
    return true;  
}
```

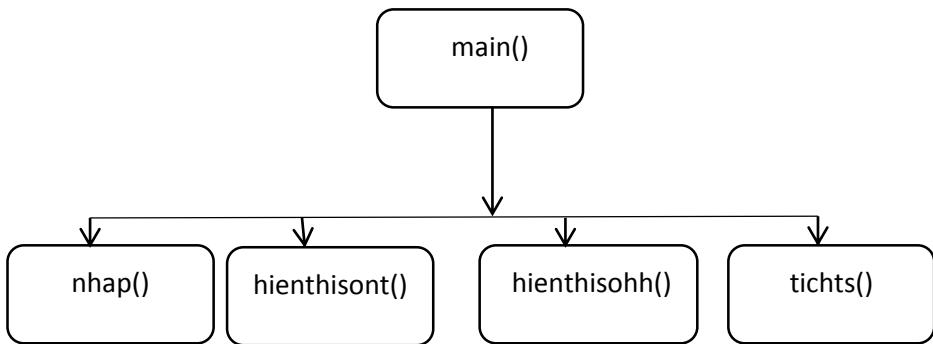
Phương thức hiển thị số nguyên tố

```
public void hienThisNT(int a){  
    for (int i = 1; i < a; i++) {  
        if(songt(i)==true){  
            System.out.print(i + " ");  
        }  
    }  
}
```

Phương thức phân tích thành thừa số

```
void tichthuaso(int a){  
    int i=2;  
    int dem=0;  
    System.out.println("Tich thua so");  
    while(a!=1){  
        if(a%i==0){  
            a/=i;  
            dem++;  
        }  
        else {  
            if(dem==0| |dem==1)  
                System.out.print(i+"*");  
            else  
                System.out.print(i+"*"+dem+"*");  
            i++;  
            dem=0;  
        }  
    }  
}
```

Viết phương thức main dùng **switch case** để tạo menu. Tổ chức chương trình theo sơ đồ sau:



3. Vòng lặp while, do...while, for

Dạng 1

```
while(boolean_expr){
//do something
}
```

Dạng 2

```
do {
//do something
} while(boolean_expr);
```

Bài 4. Tìm UCLN, BCNN của 2 số được nhập vào từ bàn phím.

Hướng dẫn:

```
public class Bai4{
    int a;
    int b;
    public int getA() {
        return a;
    }
    public int getB() {
        return b;
    }
    void nhap() {
        Nội dung phương thức nhập.
    }
    int USCLN(int a, int b) {
        while (a != b) {
            if (a > b) {
```

```

        a = a - b;
    } else {
        b = b - a;
    }
}

return a;
}

int BSCNN(int a,int b){
    return a*b/USCLN(a, b);
}

public static void main(String[] args) {
    Bai4 bai=new Bai4();
    bai.nhap();
    System.out.println("UCLN:" + bai.USCLN(bai.getA(), bai.getB()));
    System.out.println("BCNN :" + bai.BSCNN(bai.getA(), bai.getB()));
}
}

```

4. Mảng

Khai báo: DataType[] array = new DataType[size];

DataType[] array = {value1, value2,..., valueN};

Bài 5. Nhập dãy n phần tử. Xóa các phần tử có giá trị bằng x nhập từ bàn phím.

Hướng dẫn:

```

import java.util.Scanner;

public class Delete{
    public static void main(String[] args){
        int n, x, flag = 1, loc = 0;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter no. of elements you want in array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter all the elements:");
        for (int i = 0; i < n; i++){
            a[i] = s.nextInt();
        }
    }
}

```

```

System.out.print("Enter the element you want to delete:");
x = s.nextInt();
for (int i = 0; i < n; i++){
    if(a[i] == x){
        flag = 1;
        loc = i;
        break;
    }
    else{
        flag = 0;
    }
}
if(flag == 1){
    for(int i = loc+1; i < n; i++) {
        a[i-1] = a[i];
    }
    System.out.print("After Deleting:");
    for (int i = 0; i < n-2; i++){
        System.out.print(a[i]+",");
    }
    System.out.print(a[n-2]);
}
else{
    System.out.println("Element not found");
}
}
}

```

Bài 6. Nhập một mảng nguyên từ bàn phím

- a) Sắp xếp và in lại dãy đã sắp ra màn hình.
- b) In ra màn hình phần tử có giá trị nhỏ nhất
- c) Tính trung bình cộng các phần tử chia hết cho 3.

Hướng dẫn:

- Sử dụng Array.sort (mang) để sắp xếp, sau đó dùng vòng lặp **for** duyệt từng phần tử in ra màn hình.

- Cho số nhỏ nhất là số đầu tiên, sau đó so sánh số nhỏ nhất đó với các số còn lại nếu số đó được so sánh nhỏ hơn thì lấy số đó làm số nhỏ nhất.
Min=Math.min(Min,a[i])
- Duyệt mảng kiểm tra từng phần tử, nếu phần tử thứ i chia hết cho 3:
if(A[i] %3==0) thì thực hiện cộng vào tổng, tăng số đếm các số chia hết cho 3.
- Lấy tổng chia cho số đếm được kết quả

Bài 7. Nhập mảng có n phần tử các số nguyên.

- In ra các phần tử là số nguyên tố của mảng.
- In ra các phần tử là số hoàn hảo của mảng.
- In ra các phần tử là số chẵn.

Hướng dẫn:

Khai báo mảng n phần tử nguyên. Sau đó sử dụng các phương thức sau:

Phương thức kiểm tra hoàn hảo: tương tự các bài trước.

Phương thức kiểm tra chẵn/lẻ

```
public static boolean kiemTraChan(int n){
    if(n % 2 == 0){
        return true;
    }
    else{
        return false;
    }
}
```

Bài 8. Nhập mảng có n phần tử các số nguyên, sắp xếp mảng tăng dần bằng thuật toán chèn và hiển thị mảng đã sắp xếp ra màn hình.

Hướng dẫn:

```
import java.util.Arrays;

public class InsertionSort {
    void InsertionSort(int[] nums){
        for(int i = 1; i < nums.length; i++){
            int value = nums[i];
            int j = i - 1;
            while(j >= 0 && nums[j] > value){
                nums[j + 1] = nums[j];
                j = j - 1;
            }
        }
    }
}
```

```

        nums[j + 1] = value;
    }
}

public static void main(String args[]){
    InsertionSort ob = new InsertionSort();
    int nums[] = {7, -5, 3, 2, 1, 0, 45};
    System.out.println("Original Array:");
    System.out.println(Arrays.toString(nums));
    ob.InsertionSort(nums);
    System.out.println("Sorted Array");
    System.out.println(Arrays.toString(nums));
}
}

```

Bài 9.

Dãy số Fibonacci được định nghĩa như sau: $F_0 = 1$; $F_1 = 1$; $F_n = F_{n-1} + F_{n-2}$ với $n \geq 2$. Nhập một số tự nhiên n.

- a) Hãy viết chương trình tìm số Fibonacci thứ n.
- b) Hãy liệt kê các số Fibonacci nhỏ hơn n là số nguyên tố.

Hướng dẫn:

Viết phương thức ktnt (**int** n): tương tự các bài trên

Viết phương thức trả về số fibonacci thứ n

```

int fibonacci(int n) {
    if (n < 2) {
        return n;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}

```

Trong phương thức main(), nhập số nguyên n, gọi phương thức fibonacci(**int** n) để tìm số Fibonacci thứ n.

Sử dụng vòng lặp để in ra các số Fibonacci nhỏ hơn n và là số nguyên tố

```

for (int i = 0; i < n; i++){
    if (ktnt(i) == true) {
        for (int j = 0; j < n; j++) {
            if (i == fibonacci(j)) {

```

```
        System.out.print(i + " ");
    }
}
}
```

Bài 10. Viết chương trình tìm điểm trung bình của hai học sinh trong ba môn học.

Cho điểm của học sinh thứ nhất là 60, 55 và 70, điểm của học sinh thứ hai là 80, 60 và 41

Hướng dẫn: Lưu trữ điểm của hai học sinh trong một mảng 2 chiều có 2 hàng và 3 cột. Các hàng sẽ đại diện cho học sinh và các cột lưu điểm của học sinh.

```
class Array2DExample {  
    public static void main(String args[]) {  
        int[][] score = {  
            {60, 55, 70},  
            {80, 60, 41}  
        };  
  
        int[] sum = new int[2]; // tổng mảng  
        sum[0] = 0; // tổng của sinh viên 1  
        sum[1] = 0; // tổng của sinh viên 2  
  
        float[] avg = new float[2]; // trung bình mảng  
        for (int r = 0; r < 2; r++) {  
            for (int c = 0; c < 3; c++) {  
                sum[r] += score[r][c];  
            }  
        }  
  
        avg[0] = (float)sum[0] / 3; // trung bình của sinh viên 1  
        avg[1] = (float)sum[1] / 3; // trung bình của sinh viên 2  
        System.out.println("Average score of 1st student = " + avg[0]);  
        System.out.println("Average score of 2nd student = " + avg[1]);  
    }  
}
```

LAB 2. LỚP - ĐỐI TƯỢNG - KẾ THỪA [1, 2, 7]

A. MỤC TIÊU

Cung cấp cho sinh viên các kỹ thuật về lập trình hướng đối tượng trong JAVA:

- Biết cách đặc tả truy xuất các thành phần bên trong lớp.
- Khai thác các tính chất của lập trình hướng đối tượng (kế thừa, nạp chồng, ghi đè phương thức)
- Biết sử dụng mô hình lớp với mô tả kế thừa.
- Biết cách viết code kế thừa và đa hình trên Java.

B. NỘI DUNG

- Khai báo sử dụng lớp, đối tượng.
- Tạo mảng các đối tượng, giải quyết các bài toán quản lý cơ bản.
- Viết các chương trình vận dụng tính kế thừa.

C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB.
- Phần mềm NETBEAN IDE 8.0, JDK 1.8

D. KẾT QUẢ SAU KHI HOÀN THÀNH

- Xây dựng được ứng dụng với nhiều lớp được tổ chức theo sự phân cấp kế thừa
- Sử dụng lại những gì đã có ở một lớp cha
- Ghi đè hiệu chỉnh lại nội dung của một phương thức ở lớp con.

E. HƯỚNG DẪN CHI TIẾT

1. Lớp

- Tên lớp: Viết hoa
- Danh sách các thuộc tính (khai báo là **private**)
- Phương thức khởi tạo: Để khởi tạo đối tượng (phương thức tạo 0, 1, 2... tham số)
- Danh sách các phương thức được cài đặt thêm

Mô tả lớp:

```
package pack.name;  
Modifier class ClassName{  
    class's body  
}
```

Khai báo mảng đối tượng:

```
ClassName [ ] ob = new ClassName [size]
```

Bài 1.

Tạo **class** Product gồm các thuộc tính:

- Tên hàng hóa
- Nhà sản xuất

- Giá bán
 - + Tạo 2 constructor cho lớp này.
 - + Cài đặt phương thức nhập và hiển thị.
- Tạo **class** ProductMenu, khai báo phương thức main và tạo menu sau:

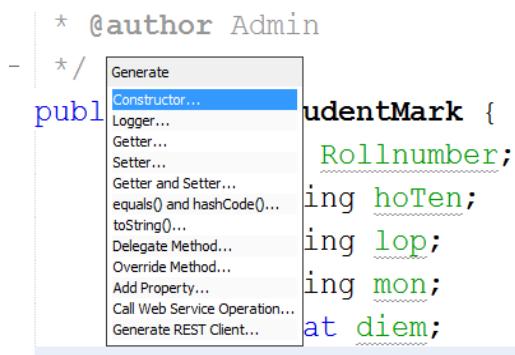
1. Nhập thông tin cho n sản phẩm
2. Hiển thị thông tin vừa nhập
3. Sắp xếp thông tin giảm dần theo giá và hiển thị
4. Thoát

Hướng dẫn:

Bước 1: Tạo lớp Product gồm các thuộc tính.

```
private String tenHangHoa;
private String nhaSanXuat;
private float giaBan;
```

Chèn tự động phương thức tạo, phương thức set, get. Sinh viên có thể dùng tổ hợp phím Alt+Insert để chèn code tự động.



Hình 1. Chèn code tự động

Viết phương thức nhập và phương thức xuất thông tin là các thuộc tính của lớp.

Bổ sung phương thức sắp xếp:

```
public void sort(Product[] b){
    for (int i = 0; i < b.length - 1; i++) {
        for (int j = i + 1; j < b.length; j++) {
            if (b[i].giaBan > b[j].giaBan) {
                Product tem = b[i];
                b[i] = b[j];
                b[j] = tem;
            }
        }
    }
}
```

Bước 2: Tạo lớp ProductMenu có nội dung như sau.

```

public class ProductMenu {
    static void menu() {
        Viết nội dung lựa chọn dùng switch tương ứng với đề bài
    }
    public static void main(String[] args) {
        int n = 0;
        Product a = new Product();
        Product[] product = null;
        do {
            menu();
            System.out.println("Nhập vào lựa chọn của bạn :");
            Scanner sc = new Scanner(System.in);
            n = Integer.parseInt(sc.nextLine());
            switch (n) {
                case 1: {
                    int m;
                    System.out.println("Nhập vào n :");
                    m = Integer.parseInt(sc.nextLine());
                    product = new Product[m];
                    for (int i = 0; i < m; i++) {
                        product[i] = new Product();
                        product[i].nhap();
                    }
                    break;
                }
                case 2: {
                    if (product == null) {
                        System.out.println("Bạn chưa nhập dữ liệu");
                    } else {
                        System.out.println("Dữ liệu bạn vừa nhập là :");
                        for (int i = 0; i < product.length; i++) {
                            System.out.println("Thông tin hàng hóa thứ " + (i + 1));
                            product[i].hienthi();
                        }
                    }
                    break;
                }
                case 3: {
                    if (product == null) {
                        System.out.println("Bạn chưa nhập dữ liệu");
                    }
                }
            }
        }
    }
}

```

```

        } else {
            a.sort(product);
            System.out.println("du lieu sau khi sap xep la :");
            for (int i = 0; i < product.length; i++) {
                System.out.println("thong tin hang hoa thu " + (i + 1));
                product[i].hienthi();
            }
        }
        break;
    }
    case 4:
        break;
    default: {
        System.out.println("khong co lua chon cua ban ");
        break;
    }
}
} while (n != 4);
}
}

```

Bài 2.

Cài đặt lớp Product gồm các thuộc tính (khai báo là **private**)

- **String** maHH;
- **String** tenHH;
- **float** soLuong;
- **float** gia1SP;

Cài đặt 2 constructors, các phương thức get/set.

Cài đặt phương thức input(), display().

Khai báo phương thức main và thực hiện như sau:

- Khai báo mảng có n phần tử kiểu Product.
- Gọi phương thức nhập thông tin cho các phần tử của mảng.
- Tìm ra sản phẩm nào có giá bán cao nhất.
- Sắp xếp theo thứ tự giảm dần của giá
- Tìm trong danh sách hàng hóa có mặt hàng “Sữa” hay không?

Hướng dẫn:

Bước 1: Tương tự , xây dựng lớp ProductBai2 gồm các thuộc tính, phương thức tạo, Phương thức set, get.

Bước 2: Xây dựng lớp TestProduct có phương thức main và một số code như sau:

Tìm mặt hàng có giá cao nhất.

```

ArrayList<ProductBai2> arrlist = new ArrayList<ProductBai2>();
float max = 0;
for (ProductBai2 pr : arrlist) {
    if (max < pr.getGia1SP()) {
        max = pr.getGia1SP();
    }
}
System.out.println("thong tin mat hang co gia cao nhat la :");
for (ProductBai2 pr : arrlist) {
    if (pr.getGia1SP() == max) {
        pr.hienthi();
    }
}

```

Sắp xếp theo giá: Sử dụng Collections.sort

```

Collections.sort(arrlist, new Comparator<ProductBai2>() {
    @Override
    public int compare(ProductBai2 pr1, ProductBai2 pr2) {
        if (pr1.getGia1SP() < pr2.getGia1SP()) {
            return 1;
        } else {
            if (pr1.getGia1SP() == pr2.getGia1SP()) {
                return 0;
            } else {
                return -1;
            }
        }
    }
);
System.out.println("danh sach duoc sap xep giam dan theo gia la:");
int i=1;
for (ProductBai2 pr : arrlist) {
    System.out.println("san pham thu :" +(i));
    pr.hienthi();
    i++;
}

```

Tìm kiếm mặt hàng tên là “Sữa”

```

for (ProductBai2 pr : arrlist)
{
    if (pr.getTenHH().equals("sữa") || pr.getTenHH().equals("SỮA"))

```

```
    pr.hienthi();
}
```

2. Kế thừa

- Lớp mới kế thừa những thành viên đã có trong lớp cũ
- Một lớp chỉ có thể kế thừa 1 lớp khác nhưng có thể thực thi nhiều giao diện.
- **extends** với lớp và **implements** với giao diện

Cú pháp

```
class Subclass extends Superclass{
//Subclass body
}
```

Lớp con có thể kế thừa được gì?

- Kế thừa được các thành viên (bao gồm dữ liệu và phương thức) được khai báo là **public** và **protected** của lớp cha.
- Không kế thừa được các thành viên **private**.

Khởi tạo đối tượng trong kế thừa

- Lớp con không kế thừa phương thức khởi tạo của lớp cha
- Lớp cha phải được khởi tạo trước lớp con
- Các phương thức khởi tạo của lớp con luôn gọi phương thức khởi tạo của lớp cha:
 - + Tự động gọi (không tường minh – không cần thể hiện bằng câu lệnh gọi): nếu lớp cha có phương thức khởi tạo **mặc định**
 - + Gọi trực tiếp (tường minh): nếu lớp cha có phương thức khởi tạo **khác mặc định**. Cú pháp: **super**(parameter List)

Bài 3. Cài đặt lớp Book gồm các thuộc tính

```
private String bookName;
private String bookAuthor;
private String producer;
private int yearPublishing;
private float price;
```

Cài đặt 2 constructors, các phương thức set/get cho các thuộc tính của lớp.

Cài đặt 2 phương thức input() và display để nhập và hiển thị các thuộc tính của lớp.

Cài đặt lớp UnetiBook kế thừa lớp Book và bổ sung thêm vào thuộc tính:

```
private String language;
private int semester;
```

Cài đặt 2 constructor trong đó sử dụng **super** để gọi đến constructor của lớp cha.

Cài đặt các phương thức get/set cho các thuộc tính bổ sung

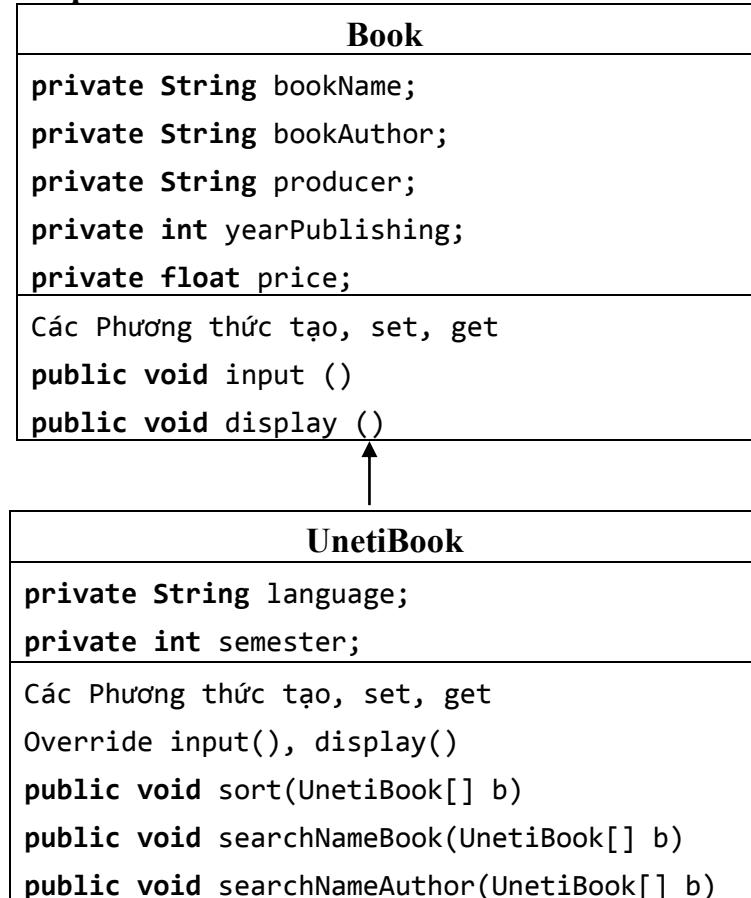
Override các phương thức input() và display().

Cài đặt lớp Test trong đó tạo menu và thực hiện theo các chức năng của menu:

1. Nhập thông tin n cuốn sách của Uneti
2. Hiển thị thông tin vừa nhập

3. Sắp xếp thông tin giảm dần theo năm xuất bản và hiển thị
 4. Tìm kiếm theo tên sách
 5. Tìm kiếm theo tên tác giả
 6. Thoát.

Hướng dẫn: Sơ đồ lớp



Bước 1: Tao lớp Book gồm các thuộc tính

```
public class Book {  
    private String bookName;  
    private String bookAuthor;  
    private String producer;  
    private int yearPublishing;
```

```
private float price;  
Sau đó chèn các phương thức tạo, set, get, phương thức input(), display()  
{
```

Bước 2: Tạo lớp UnetiBook kế thừa từ lớp Book

```
public class UnetiBook extends Book{  
    private String language; //Hai thuộc tính riêng của lớp UnetiBook  
    private int semester;
```

Bổ sung phương thức tạo, phương thức set, get cho các thuộc tính tương ứng của **class** này

Sau đó Override hai phương thức input () và display () của lớp Book như sau:

```
@Override  
    public void input(){  
        super.input();  
        Scanner sc=new Scanner(System.in);  
        System.out.println("nhap ngon ngu :");  
        language=sc.nextLine();  
        System.out.println("Nhập học kỳ :");  
        semester =Integer.parseInt(sc.nextLine());  
    }  
    @Override  
    public void display(){  
        super.display();  
        System.out.println("ngôn ngữ :" +language);  
        System.out.println("học kỳ :" +semester);  
    }
```

Viết phương thức sắp xếp sách theo năm xuất bản

```
public void sort(UnetiBook[] b){  
    for (int i = 0; i < b.length - 1; i++) {  
        for (int j = i + 1; j < b.length; j++) {  
            if (b[i].getYearPublishing() < b[j].getYearPublishing())  
            {  
                UnetiBook tem = b[i];  
                b[i] = b[j];  
                b[j] = tem;  
            }  
        }  
    }  
}
```

Phương thức tìm kiếm sách theo tên sách

```
public void searchNameBook(UnetiBook[] b){  
    Scanner sc=new Scanner(System.in);  
    System.out.println("nhập tên sách:");  
    String nameBook=sc.nextLine();  
    int dem=0;  
    System.out.println("thông tin sách bạn muốn tìm là :");  
    for (int i = 0; i < b.length; i++) {  
        if(b[i].getBookName().equals(nameBook)){  
            b[i].hienthi();  
        }  
    }  
}
```

```

        dem++;
    }
}
if(dem==0){
    System.out.println("khong co sach ban muon tim");
}
}

```

Phương thức tìm kiếm theo tên tác giả

```

public void searchNameAuthor(UnetiBook[] b){
    Viết tương tự Phương thức tìm kiếm theo tên sách.
    Sử dụng for để duyệt mảng.
    Sử dụng if(b[i].getBookName().equals(nameAuthor)) kiểm tra tên tác giả
}

```

Phương thức tạo menu() để sử dụng

```

void menu(){
    System.out.println("1 nhap thong tin n cuon sach ");
    System.out.println("2 hien thi thong tin vua nhap");
    System.out.println("3 sap xep giam dan theo nam xuat ban");
    System.out.println("4 tim kiem theo ten sach");
    System.out.println("5 tim kiem theo ten tac gia ");
    System.out.println("6 thoat");
}

```

Phương thức main() để chạy chương trình

```

public static void main(String[] args) {
    int n = 0;
    UnetiBook ab=new UnetiBook();
    UnetiBook[] ab1=null;
    do {
        ab.menu();
        System.out.println("Nhập vào lựa chọn của bạn :");
        Scanner sc = new Scanner(System.in);
        n = Integer.parseInt(sc.nextLine());
        switch (n) {
            case 1: {
                int m;
                System.out.println("Nhập vào n :");
                m = Integer.parseInt(sc.nextLine());
                ab1= new UnetiBook[m];
                for (int i = 0; i < m; i++) {

```

```

        ab1[i] = new UnetiBook();
        ab1[i].input();
    }
    break;
}
case 2: {
    if (ab1 == null) {
        System.out.println("ban chua nhap du lieu");
    } else {
        System.out.println("du lieu ban vua nhap la :");
        for (int i = 0; i < ab1.length; i++) {
            System.out.println("thong tin sach thu " + (i + 1));
            ab1[i].hienthi();
        }
    }
    break;
}
case 3: {
    if (ab1 == null) {
        System.out.println("ban chua nhap du lieu");
    } else {
        ab.sort(ab1);
        System.out.println("du lieu sau khi sap xep la :");
        for (int i = 0; i < ab1.length; i++) {
            System.out.println("thong tin hang hoa thu " + (i + 1));
            ab1[i].hienthi();
        }
    }
    break;
}
case 4: {
    ab.searchNameBook(ab1);
    break;
}
case 5: {
    ab.searchNameBook(ab1);
    break;
}
}

```

```

        case 6:
            break;
        default:{
            System.out.println("khong co lua chon cua ban ");
            break;
        }
    }
} while (n != 6);
}
}

```

Bài 4.

Tạo **class** Engine gồm các thuộc tính:

- engineId (Mã máy)
- engineName (Tên máy)
- manufacturer (Tên nhà sản xuất)
- yearMaking (Năm sản xuất)
- price (Giá bán)

+ Tạo 2 constructors

+ Tạo các phương thức get/set

+ Cài đặt phương thức input(), display()

Tạo **class** Mobile kế thừa lớp Engine ở trên và bổ sung thêm các thuộc tính:

- country (Nước sản xuất)

+ Tạo 2 constructors, trong đó constructor có tham số phải sử dụng từ khóa **super** để gọi đến constructor có tham số của lớp cha

+ Cài đặt các phương thức get/set cho thuộc tính bổ sung

+ Override phương thức input() và display() của lớp cha.

Tạo **class** Car kế thừa **class** Engine và bổ sung thêm thuộc tính:

- totalSeat (Số chỗ ngồi)
- speed (Tốc độ)

+ Tạo 2 constructors, trong đó constructor có tham số phải sử dụng từ khóa **super** để gọi đến constructor có tham số của lớp cha

+ Cài đặt các phương thức get/set cho thuộc tính bổ sung

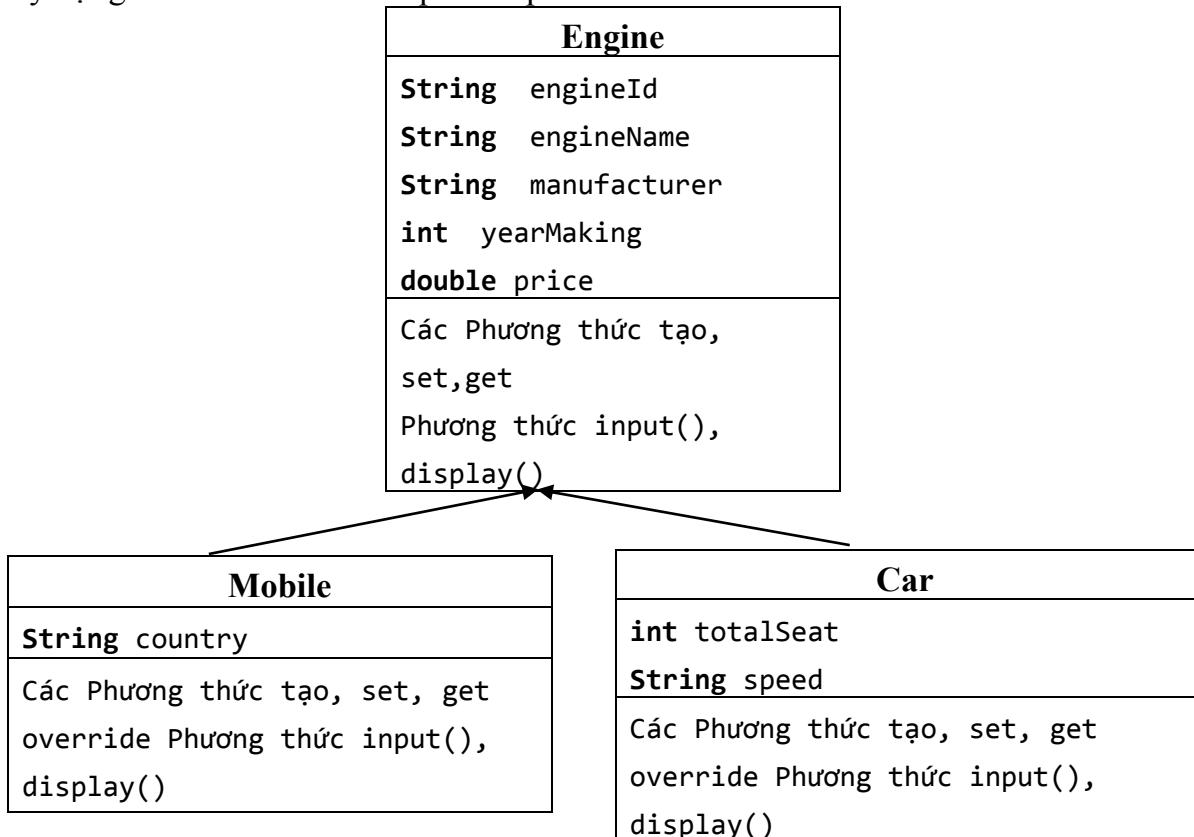
+ Override phương thức input() và display() của lớp cha.

Tạo **class** Manager trong đó có phương thức main.

1. Nhập vào thông tin cho n điện thoại
2. Nhập vào thông tin cho n ô tô
3. Hiển thị thông tin cả điện thoại và ô tô
4. Tìm kiếm thông tin theo tên nhà sản xuất.
5. Thoát

Hướng dẫn:

Xây dựng các **class** theo sơ đồ phân cấp kế thừa như sau:



Bài 5.

Một thư viện cần quản lý các tài liệu bao gồm Sách, Tạp chí, Báo

- + Mỗi tài liệu có các thuộc tính: Mã tài liệu, Tên nhà xuất bản, Số bản phát hành.
- + Các loại sách cần quản lý: Tên tác giả, Tên sách, số trang
- + Các tạp chí cần quản lý: Số phát hành, tháng phát hành
- + Các báo cần quản lý: ngày phát hành. (Date)

Xây dựng các lớp quản lý các loại tài liệu trên sao cho việc sử dụng lại được nhiều nhất.

Xây dựng lớp QuanLySach cài đặt các phương thức thực hiện các công việc sau:

- a. Nhập thông tin về các tài liệu
- b. Hiển thị thông tin về các tài liệu
- c. Tìm kiếm tài liệu theo loại
- d. Tìm kiếm tài liệu theo tên tác giả
- e. Tìm kiếm theo tên tác giả “gần đúng” .
- f. Hiển thị danh sách về loại tài liệu theo chỉ định của người sử dụng.

Hướng dẫn:

Bước 1: Xây dựng class TaiLieu

```

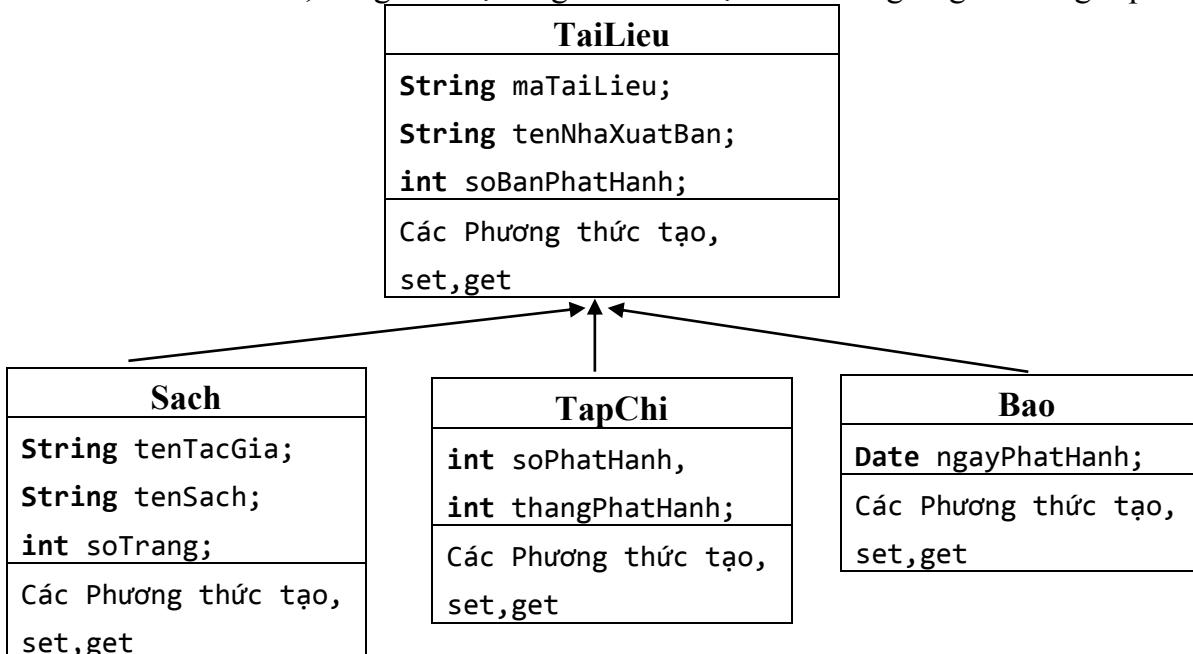
class TaiLieu
{
    String maTaiLieu;
    String tenNhaXuatBan;
}
    
```

```

    int soBanPhatHanh;
    Chèn tự động các Phương thức khởi tạo, set, get thuộc tính.
}

```

Sau khi xây dựng lớp TaiLieu, lần lượt tạo các lớp Sach, TapChi, Bao kế thừa lớp TaiLieu theo sơ đồ sau, đồng thời định nghĩa các thuộc tính tương ứng với từng lớp:



Bước 2: Định nghĩa lớp Sach kế thừa lớp TaiLieu

```

public class Sach extends TaiLieu{
    private String tenTacGia;
    private String tenSach;
    private int soTrang;
    Chèn tự động Phương thức tạo, các Phương thức set, get
}

```

Bước 3: Tượng tự tạo lớp TapChi kế thừa lớp TaiLieu

```

public class TapChi extends TaiLieu{
    private int soPhatHanh,thangPhatHanh;
    Chèn tự động các Phương thức tạo, set, get
}

```

Bước 4: Lớp Bao kế thừa lớp TaiLieu

```

public class Bao extends TaiLieu{
    private Date ngayPhatHanh;
    Chèn tự động Phương thức tạo, các Phương thức set, get tự động
}

```

Bước 5: Tạo lớp QuanLyTV để định nghĩa các phương thức nhằm cung cấp các chức năng cho hệ thống như sau:

```

public class QuanLyTV {
    private ArrayList<TaiLieu> taiLieus;
}

```

```

private Scanner reader;
public QuanLyTV() {
    taiLieu = new ArrayList<>();
    reader = new Scanner(System.in);
}
public Sach taoMoiSach(){
    Sach s = new Sach();
    System.out.println("Mã tài liệu: ");
    s.setMaTaiLieu(reader.nextLine());
    System.out.println("Tên nhà xuất bản: ");
    s.setTenNhaXuatBan(reader.nextLine());
    System.out.println("Số bản phát hành: ");
    s.setSoBanPhatHanh(Integer.parseInt(reader.nextLine()));
    System.out.println("Tên tác giả: ");
    s.setTenTacGia(reader.nextLine());
    System.out.println("Tên sách: ");
    s.setTenSach(reader.nextLine());
    System.out.println("Số trang: ");
    s.setSoTrang(Integer.parseInt(reader.nextLine()));
    return s;
}
public TapChi taoMoiTapChi(){
    TapChi tapChi = new TapChi();
    System.out.println("Mã tài liệu: ");
    tapChi.setMaTaiLieu(reader.nextLine());
    System.out.println("Tên nhà xuất bản: ");
    tapChi.setTenNhaXuatBan(reader.nextLine());
    System.out.println("Số bản phát hành: ");
    tapChi.setSoBanPhatHanh(Integer.parseInt(reader.nextLine()));
    System.out.println("Số phát hành: ");
    tapChi.setSoPhatHanh(Integer.parseInt(reader.nextLine()));
    System.out.println("Tháng phát hành: ");
    tapChi.setThangPhatHanh(Integer.parseInt(reader.nextLine()));
    return tapChi;
}
public Bao taoMoiBao(){
    Bao bao = new Bao();
    System.out.println("Mã tài liệu: ");
    bao.setMaTaiLieu(reader.nextLine());
    System.out.println("Tên nhà xuất bản: ");
}

```

```

        bao.setTenNhaXuatBan(reader.nextLine());
        System.out.println("Số bản phát hành: ");
        bao.setSoBanPhatHanh(Integer.parseInt(reader.nextLine()));
        System.out.println("Ngày phát hành: ");
        bao.setNgayPhatHanh(convertStringToDate(reader.nextLine()));
        return bao;
    }

    private Date convertStringToDate(String ddMMyyyy) {
        try {
            return new SimpleDateFormat("dd/MM/yyyy").parse(ddMMyyyy);
        } catch (ParseException ex) {
            Logger.getLogger(QuanLyTV.class.getName()).log(Level.SEVERE, null, ex);
        }
        return null;
    }

    public void nhapDanhSachTaiLieu(){
        System.out.println("Nhập 1 - Tạo mới sách");
        System.out.println("Nhập 2 - Tạo mới tạp chí");
        System.out.println("Nhập 3 - Tạo mới báo");
        System.out.println("Nhập 4 - Kết thúc");
        int selectedValue;
        do {
            System.out.println("Bạn chọn: ");
            selectedValue = Integer.parseInt(reader.nextLine());
            switch(selectedValue){
                case 1:
                    taiLieus.add(taoMoiSach());
                    break;
                case 2:
                    taiLieus.add(taoMoiTapChi());
                    break;
                case 3:
                    taiLieus.add(taoMoiBao());
                    break;
            }
        } while (selectedValue!=4);
    }

    private void xuatThongTin(TaiLieu taiLieu){
        System.out.println("Mã tài liệu: " + taiLieu.getMaTaiLieu());
        System.out.println("Tên nhà xuất bản: " taiLieu.getTenNhaXuatBan());
    }
}

```

```

System.out.println("Số bản phát hành:" + taiLieu.getSoBanPhatHanh());
    if (taiLieu instanceof Sach) {
        Sach sach = (Sach) taiLieu;
        System.out.println("Tên tác giả: " + sach.getTenTacGia());
        System.out.println("Tên sách: " + sach.getTenSach());
        System.out.println("Số trang: " + sach.getSoTrang());
    }
    else{
        if (taiLieu instanceof TapChi) {
            TapChi tapChi = (TapChi) taiLieu;
            System.out.println("Số phát hành: " + tapChi.getSoPhatHanh());
            System.out.println("Tháng phát hành: " + tapChi.getThangPhatHanh());
        }else{
            Bao bao = (Bao)taiLieu;
            System.out.println("Ngày phát hành"+
convertDateToString(bao.getNgayPhatHanh()));
        }
    }
}

private String convertDateToString(Date ngayPhatHanh) {
    return new SimpleDateFormat("dd/MM/yyyy").format(ngayPhatHanh);
}

public void xuatDanhSachTaiLieu(){
    for (TaiLieu taiLieu : taiLieus) {
        xuatThongTin(taiLieu);
    }
}

public void timTheoLoai(String loai){
    if (loai.equalsIgnoreCase("Sach")) {
        for (TaiLieu taiLieu : taiLieus) {
            if (taiLieu instanceof Sach) {
                xuatThongTin(taiLieu);
            }
        }
    } else {
        if (loai.equalsIgnoreCase("Tap Chi")) {
            for (TaiLieu taiLieu : taiLieus) {
                if (taiLieu instanceof TapChi) {
                    xuatThongTin(taiLieu);
                }
            }
        }
    }
}

```

```

        }
    } else {
        for (TaiLieu taiLieu : taiLieus) {
            if (taiLieu instanceof Bao) {
                xuatThongTin(taiLieu);
            }
        }
    }
}

public void timGanDungTheoTenSach(String str){
    for (TaiLieu taiLieu : taiLieus) {
        if (taiLieu instanceof Sach) {
            Sach sach = (Sach) taiLieu;
            if (sach.getTenSach().indexOf(str)!=-1) {
                xuatThongTin(taiLieu);
            }
        }
    }
}

public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    QuanLyTV QuanLyTV = new QuanLyTV();
    QuanLyTV.nhapDanhSachTaiLieu();
    QuanLyTV.xuatDanhSachTaiLieu();
    System.out.println("Nhập loại bạn muốn tìm: ");
    QuanLyTV.timTheoLoai(input.nextLine());
}
}

```

Bài 6. Xây dựng chương trình quản lý danh sách hóa đơn tiền điện của khách hàng. Thông tin bao gồm các loại khách hàng :

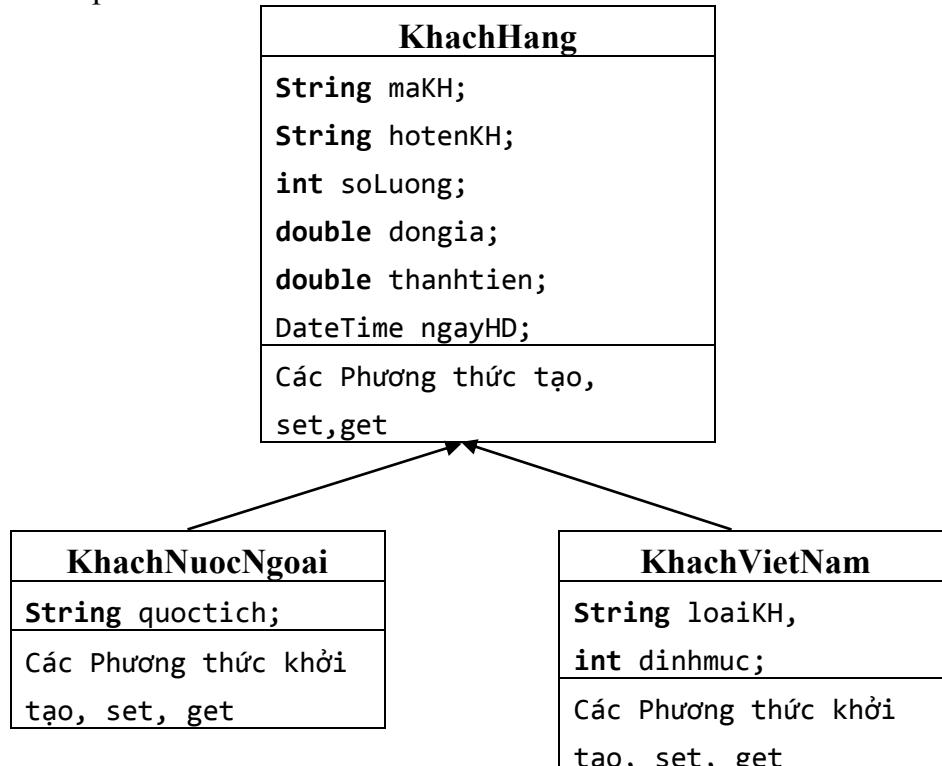
- Khách hàng Việt Nam: mã khách hàng, họ tên, ngày ra hóa đơn (ngày, tháng, năm), đối tượng khách hàng (sinh hoạt, kinh doanh, sản xuất): số lượng (số KW tiêu thụ), đơn giá, định mức. Thành tiền được tính như sau:
 - Nếu số lượng <= định mức thì: thành tiền = số lượng * đơn giá.
 - Ngược lại thì: thành tiền = số lượng * đơn giá * định mức + số lượng KW vượt định mức * Đơn giá * 2.5.
- Khách hàng nước ngoài: mã khách hàng, họ tên, ngày ra hóa đơn (ngày, tháng, năm), quốc tịch, số lượng, đơn giá. Thành tiền = số lượng * đơn giá.

Thực hiện các yêu cầu sau:

- a. Xây dựng các lớp tương ứng với sơ đồ kế thừa.
- b. Nhập xuất danh sách các hóa đơn khách hàng.
- c. Tính tổng số lượng điện tiêu thụ cho từng loại khách hàng.
- d. Tính trung bình thành tiền của khách hàng người nước ngoài.
- e. Xuất ra các hóa đơn trong tháng 09 năm 2013 (của 2 loại khách hàng).

Hướng dẫn:

Xây dựng các lớp theo sơ đồ kế thừa sau:



Bước 1. Xây dựng lớp Khách hàng bao gồm các thuộc tính chung cho cả Khách hàng nước ngoài và Khách hàng Việt Nam. Gồm các thuộc tính: mã khách hàng, số lượng, đơn giá, thành tiền, ngày của hóa đơn và họ tên khách hàng.

Bước 2. Xây dựng lớp Khách hàng nước ngoài thừa kế lớp Khách hàng bao gồm thuộc tính: *quoctich*.

Bước 3. Xây dựng lớp Khách hàng Việt Nam thừa kế lớp Khách hàng bao gồm thuộc tính: loại khách hàng, định mức.

Bước 4. Xây dựng lớp quản lý danh sách các khách hàng (dùng cấu trúc mảng)

Bước 5. Xây dựng lớp quản lý thông tin cho phép nhập xuất, tính trung bình thành tiền.

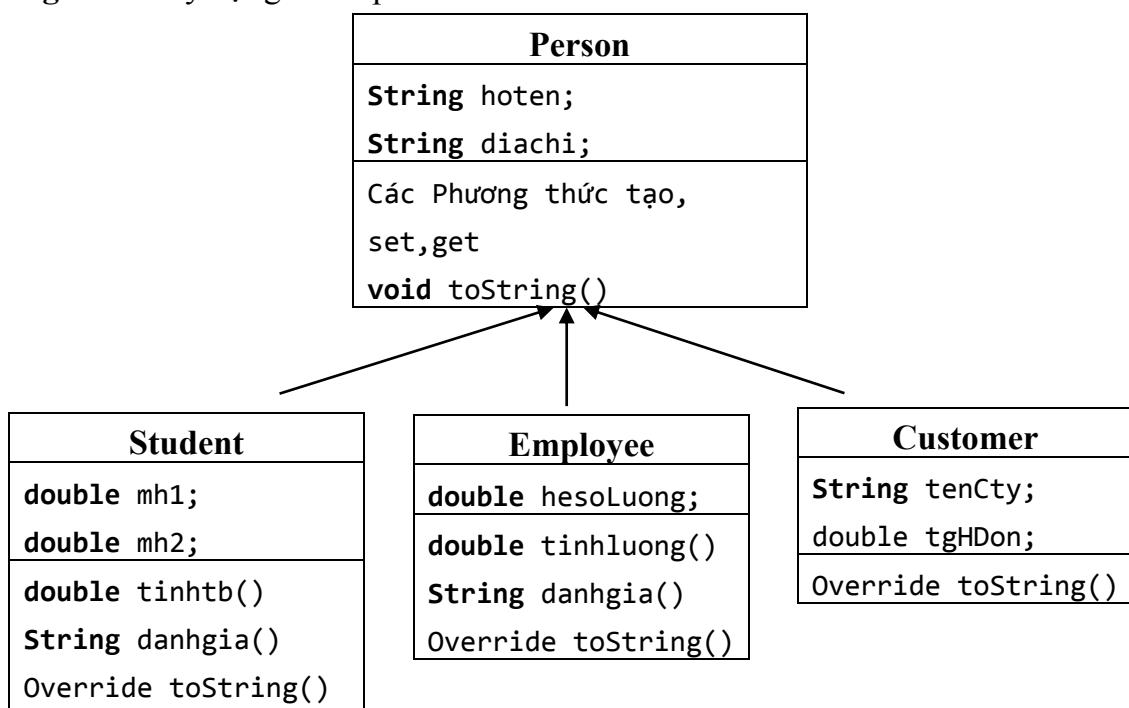
Bài 7. Tổng hợp

Giả sử cần xây dựng chương trình quản lý dùng cho một học viện nghiên cứu giảng dạy và ứng dụng. Đối tượng quản lý bao gồm các sinh viên đang theo học, các nhân viên đang làm việc tại học viện, các khách hàng đến giao dịch mua bán sản phẩm ứng dụng.

Dựa vào một số đặc tính của từng đối tượng, người quản lý cần đưa ra cách thức đánh giá khác nhau. Xây dựng các lớp sau:

- Lớp **Person**: bao gồm các thuộc tính *họ tên*, *địa chỉ*, phương thức *toString*.
- Các lớp **Student**, **Employee**, **Customer** (mô tả dưới đây) thừa kế lớp **Person**.
 - o Lớp **Student**: bao gồm các thuộc tính điểm môn học 1, điểm môn học 2, và các phương thức: tính điểm TB, đánh giá, overriding phương thức *toString* trả về bảng điểm sinh viên (gồm thông tin thuộc tính và điểm TB).
 - o Lớp **Employee**: bao gồm thuộc tính *heSoLuong*, và các phương thức: tính lương, đánh giá, overriding phương thức *toString* trả về bảng lương cho nhân viên (gồm thông tin thuộc tính đối tượng và tiền lương).
 - o Lớp **Customer**: bao gồm thuộc tính tên công ty, trị giá hóa đơn, phương thức *toString* trả về thông tin hóa đơn cho khách hàng (gồm các thuộc tính của đối tượng).
- Lớp có 1 biến danh sách để lưu các sinh viên, nhân viên, khách hàng (dùng 1 biến array Person), biến lưu tổng số người có trong danh sách, constructor mặc định khởi tạo array với dung lượng cho trước, phương thức thêm một người vào danh sách (thông số Person), xóa 1 người khỏi danh sách (nhận thông số là họ tên của người cần xóa), sắp xếp danh sách theo thứ tự họ tên, phương thức xuất danh sách. Khi danh sách đầy thì tự động tăng dung lượng dãy lên 50%.
- Viết lớp với phương thức main cho phần kiểm nghiệm. Giao tiếp với người dùng bằng menu (*thể hiện tính đa hình – polymorphism bằng cách cho phép lựa chọn nhập thông tin là sinh viên, nhân viên hay khách hàng*).

Hướng dẫn: Xây dựng các lớp theo sơ đồ kế thừa sau:



LAB 3. GIAO DIỆN, LỚP TRÙU TƯỢNG [7,13]

A. MỤC TIÊU

- Trang bị cho sinh viên kỹ năng lập trình với giao diện (**interface**), lớp trừu tượng (**abstract class**)
- Triển khai code **interface** và **abstract class** trong JAVA.

B. NỘI DUNG

- Sử dụng NETBEAN để tạo **interface**.
- Thực hành các bài toán dùng **interface**, lớp trừu tượng

C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB.
- NETBEAN IDE 8.0, JDK 1.8.

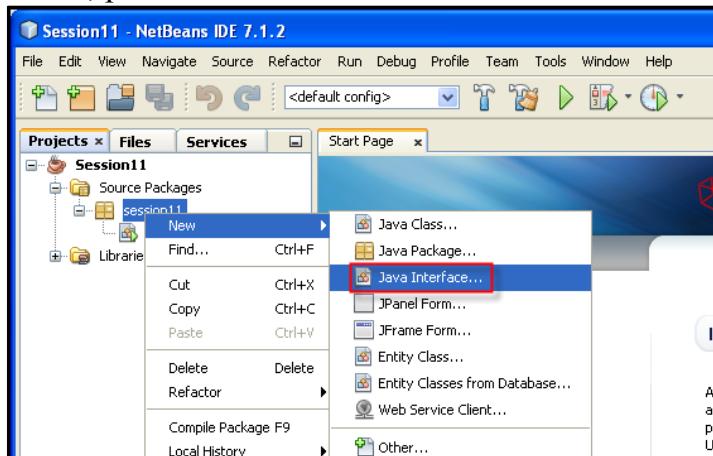
D. KẾT QUẢ SAU KHI HOÀN THÀNH

Sử dụng giao diện, lớp trừu tượng và phương thức trừu tượng giải quyết được các bài toán quản lý.

E. HƯỚNG DẪN CHI TIẾT

1. interface (giao diện)

- Khai báo các phương thức chung (không cài đặt cụ thể)
- Nếu có thuộc tính, phải được khai báo là **final**



Hình 2. Tạo interface

Bài 1.

- Tạo một giao diện IXe nằm trong gói xeco.info gồm có 2 phương thức:

```
public void nhap();  
public void hienthi();
```

- Tạo Lớp XeMay cũng nằm trong gói xeco.info thực thi giao diện IXe và có các thuộc tính sau:

```

String bienso;
String loaixe;
String mauxe;
float giatien;

```

Cài đặt Constructor, các phương thức set/get và Override các phương thức trong giao diện IXe.

- Tạo lớp XeMayHoaBinh nằm trong gói hoabinh.xemay kế thừa lớp XeMay. Bổ sung thêm thuộc tính:
 - n (kiểu int) : Dùng để nhập vào số lượng xe
 - Một mảng n phần tử kiểu XeMay dùng để lưu thông tin của n xe máy đang quản lý tại tỉnh Hòa Bình.

Cài đặt constructor, override tất cả các phương thức trong lớp XeMay. Phương thức nhập phải nhập vào số lượng xe và gọi nhập thông tin của n xe đó.

- Tạo lớp XeMayHaNoi nằm trong gói hanoi.xemay kế thừa lớp XeMay. Bổ sung thêm thuộc tính:
 - n (kiểu int): Dùng để nhập vào số lượng xe
 - Một mảng n phần tử kiểu XeMay dùng để lưu thông tin của n xe máy đang quản lý tại tỉnh Hà Nội.

Cài đặt constructor, override tất cả các phương thức trong lớp XeMay. Phương thức nhập phải nhập vào số lượng xe và gọi nhập thông tin của n xe đó.

Tạo lớp QuanLyChung nằm trong gói quanlychung.xemay, có menu như sau:

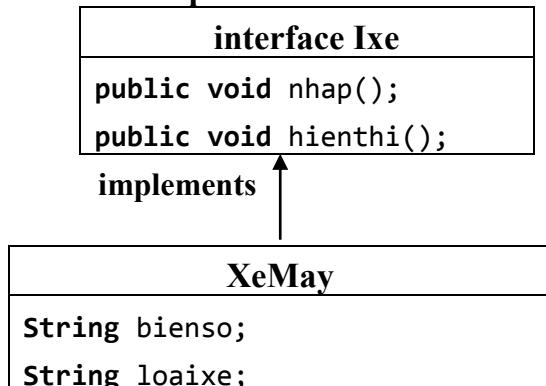
1. Nhập thông tin cho n xe máy tại tỉnh Hòa Bình.
2. Nhập thông tin cho n xe máy tại tỉnh Hà Nội.
3. Sắp xếp danh sách tăng theo biển số xe.
4. Tìm kiếm thông tin xe theo biển số xe.
5. Thống kê số lượng xe đang quản lý.
6. Thoát

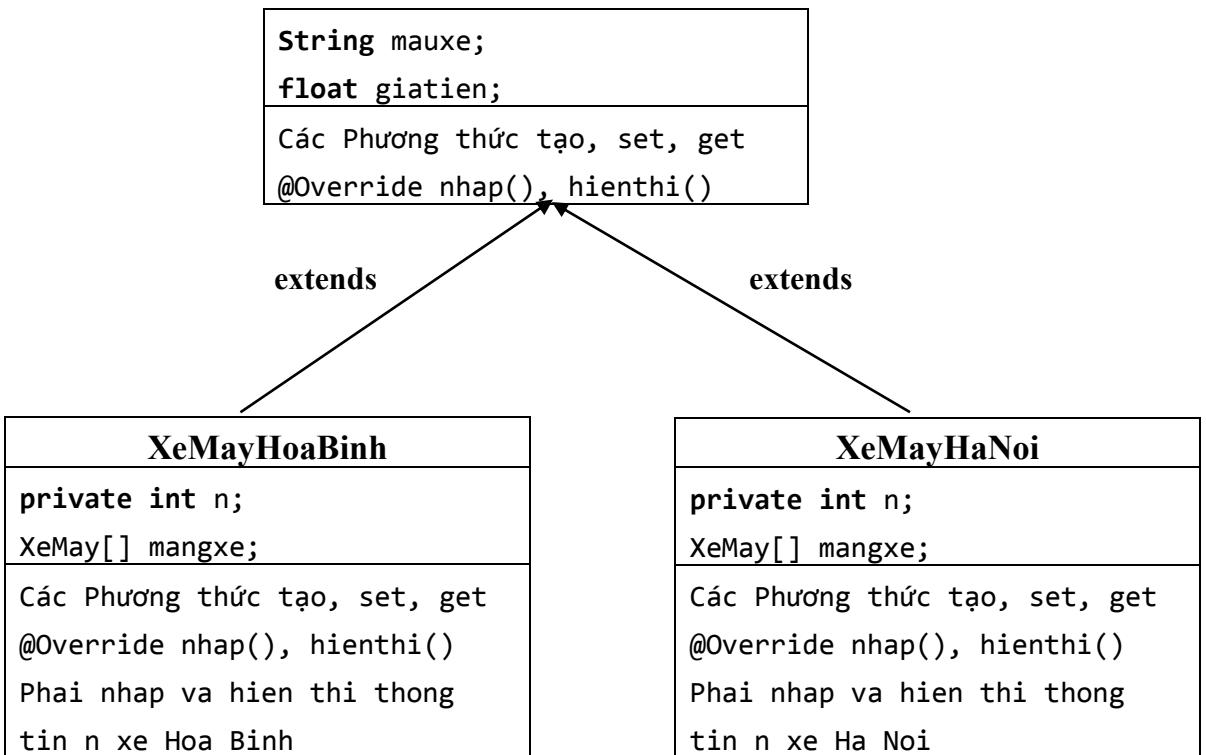
Yêu cầu:

Sau khi người sử dụng chọn chức năng 3, 4, 5 sẽ đưa ra thông báo nhập vào tinh cần thực hiện (Hòa Bình hoặc Hà Nội).

Hướng dẫn:

Sơ đồ phân cấp các interface và lớp





Bước 1: Tạo giao diện IXe

```

package bai2.xeco.info;
public interface IXe {
    public void nhap();
    public void hienthi();
}

```

Tạo lớp XeMay thực thi giao diện IXe, bổ sung các thuộc tính như đề bài

```

package bai2.xeco.info;
public class XeMay implements IXe{
    String bienso;
    String loaixe;
    String mauxe;
    float giatien;
    // Chèn các hàm tạo không tham số, 4 tham số vào đây
    // Sau đó thực hiện Override các phương thức nhap() và hienthi() để nhập
    // và hiển thị thông tin xe máy.
}

```

Bước 2: Tạo lớp XeMayHoaBinh kế thừa lớp XeMay

```

public class XeMayHoaBinh extends XeMay{
    private int n;
    XeMay[] mangxe;
}

```

```

public XeMayHoaBinh(String bienso, String loaixe, String mauxe,
float giatien, int n, XeMay[] mangxe) {
    super(bienso, loaixe, mauxe, giatien);
    this.n = n;
    this.mangxe = mangxe;
}
public XeMayHoaBinh() {
    super();
}
public int getN() {
    return n;
}
public void setN(int n) {
    this.n = n;
}
public XeMay[] getMangxe() {
    return mangxe;
}
public void setMangxe(XeMay[] mangxe) {
    this.mangxe = mangxe;
}
//Ghi đè phương thức nhap() và hienthi ()
@Override
public void nhap() {
    Scanner sc=new Scanner(System.in);
    System.out.println("Nhập vào số lượng xe máy ở hòa bình");
    n=Integer.parseInt(sc.nextLine());
    mangxe=new XeMay[n];
    for(int i=0;i<n;i++) {
        System.out.println("Nhập thông tin xe máy thứ :" +(i+1));
        mangxe[i]=new XeMay();
        mangxe[i].nhap();
    }
}
@Override
public void hienthi() {
    for(int i=0;i<n;i++) {
        System.out.println("Thông tin xe máy thứ :" +(i+1));
        mangxe[i].hienthi();
    }
}

```

```
    }  
}
```

Bước 3: Tương tự tạo lớp XeMayHaNoi kế thừa từ lớp XeMay, thực hiện giống lớp XeMayHoaBinh.

Bước 4: Tạo lớp QuanLyChung có hàm main và sử dụng

Bài 2.

- Tạo interface Imark trong package đặt tên uneti có hai phương thức:

```
public void input()  
public void display()
```

- Tạo một lớp đặt tên StudentUneti trong package uneti **implements interface IMark** và bổ sung các thuộc tính sau:

```
private String StuId;  
private String StuName;  
private String gender;  
private String birthday;  
private String nativePlace;
```

Tạo hai constructors cho **class** này.

Tạo phương thức set/get cho các thuộc tính.

Ghi đè các phương thức được thừa kế trong **interface**.

- Tạo lớp đặt tên StudentMark trong package uneti.mark **implements interface IMark** và thêm vào các thuộc tính sau:

```
private String StuId;  
private String className;  
private String subjectName;  
private int semester;  
private float mark;
```

Tạo hai phương thức tạo cho **class** này.

Tạo phương thức set/get cho các thuộc tính.

Ghi đè các phương thức được thừa kế trong **interface**.

Tạo **class** tên StudentMarkTotal trong package uneti.mark kế thừa lớp StudentMark và thêm vào các thuộc tính sau:

```
private int totalExamSubject;  
private float everageMark;
```

Tạo hai phương thức tạo cho **class** này.

Tạo phương thức set/get cho các thuộc tính .

Ghi đè các phương thức được kế thừa

Thêm phương thức tạo:

```
public void getTotalExamSubject(StudentMarkTotal [] list); //Tính tổng  
số môn thi
```

```
public void calculateEverageMark(StudentMarkTotal[] list); //Tính điểm  
trung bình các môn thi
```

Tạo class Manager trong package uneti.mark thực hiện tạo menu sau:

1. Nhập thông tin n sinh viên Uneti.
2. Nhập m điểm thi cho các sinh viên này
3. Sắp xếp sinh viên theo tên và hiển thị
4. Tìm thông tin điểm thi theo id của sinh viên
5. Exit

Chú ý:

Danh sách n sinh viên khai báo trong lớp StudentUneti

Danh sách n điểm thi được khai báo trong lớp StudentMarkTotal

Khi nhập điểm thi cho mỗi sinh viên, phải kiểm tra StuId có tồn tại trong danh sách thông tin sinh viên không.

Hướng dẫn:

Bước 1: Tạo interface Imark

Bước 2: Tạo lớp StudentUneti thực thi interface Imark

```
public class StudentUneti implements Imark{  
    private String stuId;  
    private String stuName;  
    private String gender;  
    private String birthday;  
    private String nativePlace;  
  
    //Chèn các Phương thức tạo không tham số, 5 tham số vào đây.  
    //Sau đó thực hiện Override các phương thức nhap() và hienthi() để nhập  
    //và hiển thị thông tin sinh viên.  
}
```

Tương tự tạo lớp StudentMark thực thi interface Imark

Bước 3: Tạo class StudentMarkTotal kế thừa lớp StudentMark

```
public class StudentMarkTotal extends StudentMark{  
    private int totalExamSubject;  
    private float everageMark;  
    public StudentMarkTotal(int totalExamSubject, float everageMark) {  
        super();  
        this.totalExamSubject = totalExamSubject;  
        this.everageMark = everageMark;  
    }  
    public StudentMarkTotal() {  
        super();  
    }
```

```

public int getTotalExamSubject() {
    return totalExamSubject;
}
public int getTotalExamSubject(StudentMarkTotal[] arr) {
    return arr.length;
}
public void calculateEverageMark(StudentMarkTotal[] arr) {
    for (int i = 0; i < arr.length; i++) {
        everageMark+=arr[i].getMark();
    }
}
public void setTotalExamSubject(int totalExamSubject) {
    this.totalExamSubject = totalExamSubject;
}
public float getEverageMark() {
    return everageMark;
}
public void setEverageMark(float everageMark) {
    this.everageMark = everageMark;
}
}

```

Bước 4: Tạo class Manager

```

public class Manager {
    public static void main(String[] args) {
        int n=0;
        StudentUneti[] std=null;
        StudentMark[] stdmark = null;
        do {
            Scanner sc=new Scanner(System.in);
            System.out.println("1 Nhập thông tin n sinh viên uneti");
            System.out.println("2 Nhập m điểm cho các sinh viên này");
            System.out.println("3 sắp xếp sinh viên theo tên và hiển thị ");
            System.out.println("4 Tìm thông tin điểm của sinh viên theo id");
            System.out.println("Nhập vào lựa chọn của bạn");
            n=Integer.parseInt(sc.nextLine());
            switch (n) {
                case 1:
                    int m;
                    System.out.println("nhập vào số sinh viên :");
                    m=Integer.parseInt(sc.nextLine());

```

```

        std=new StudentUneti[m];
        for (int i = 0; i < m; i++) {
            std[i]=new StudentUneti();
            int dem=0;
            do {
                System.out.println("Nhập vào thông tin sinh viên thứ "+(i+1));
                std[i].input();
                dem=0;
                for (int j = 0; j < i; j++) {
                    if(std[i].getStuId().equals(std[j].getStuId())&&std!=null) {
                        System.out.println("mã id trùng vui lòng nhập lại");
                        dem++;
                    }
                }
            }while(dem!=0);
        }
        break;
    case 2:
        if(std!=null) {
            int k;
            System.out.println("nhập vào số sinh viên :");
            k=Integer.parseInt(sc.nextLine());
            stdmark=new StudentMark[k];
            for (int i = 0; i < stdmark.length; i++) {
                stdmark[i]=new StudentMark();
                int dem=0;
                do {
                    System.out.println("Nhập vào thông tin môn học thứ "+(i+1));
                    stdmark[i].input();
                    dem=0;
                    for (int j = 0; j < std.length; j++) {
                        if(stdmark[i].getStuID().equals(std[j].getStuId())==false)
                            System.out.println("mã id sinh viên chưa có vui lòng nhập lại");
                        dem++;
                    }
                }while(dem!=0);
            }
        }
        else {
    
```

```

System.out.println("can nhap thong tin cho sinh vien truoc");
    }
    break;
case 3:
    if(std==null) {
        System.out.println("ban chua nhap du lieu ");
    }
    else {
        for (int i = 0; i < std.length- 1; i++) {
            for (int j = i + 1; j <std.length; j++) {
if(std[i].getStuName().compareTo(std[i].getStuName())>0)
{
                StudentUneti tg=std[i];
                std[i]= std[j];
                std[j]=tg;
}
        }
        for (int i = 0; i < std.length; i++) {
System.out.println("thong tin sinh vien thu "+(i+1));
            std[i].display();
        }
    }
    break;
case 4:
    if(std==null||stdmark==null) {
        System.out.println("ban chua nhap du lieu ");
    }
    else {
        String id;
        System.out.println("Mhap vao id ban muon tim:");
        id= sc.nextLine();
        System.out.println("Thong tin sinh vien ");
        for (int i = 0; i < std.length; i++) {
            if(id.equals(std[i].getStuId())){
                stdmark[i].display();
}
        }
        System.out.println("voi cac diem :");
        for (int i = 0; i < stdmark.length; i++) {

```

```

        if(id.equals(stdmark[i].getStuID())){
            stdmark[i].display();
        }
    }
    break;
case 5:
    break;
default:
    System.out.println("khong co lua chon cua ban");
    break;
}
}while(n!=5);
}
}

```

2. Lớp trừu tượng

- Lớp chứa phương thức trừu tượng bắt buộc phải khai báo là lớp trừu tượng.
- Phương thức trừu tượng không được phép định nghĩa cụ thể tại lớp cha
- Lớp con kế thừa lớp trừu tượng phải định nghĩa nội dung phương thức trừu tượng.

Bài 3.

Xây dựng lớp Employee có name và phương thức trừu tượng là earnings(). Xây dựng lớp Boss kế thừa từ Employee có cách tính lương là một khoản cố định hàng tháng. Xây dựng lớp PieceWorker có cách tính lương dựa trên số sản phẩm làm được, lương một sản phẩm là \$ 0.5. Xây dựng lớp CommissionWorker có cách tính lương là một khoản cố định + tiền hoa hồng trên số sản phẩm bán được, mỗi sản phẩm được \$ 0.1 hoa hồng. Viết chương trình có Phương thức main và sử dụng.

Hướng dẫn:

Bước 1: Tạo lớp trừu tượng Employee

```

public abstract class Employee {
private String name;
private float luong;
public abstract float earnings();
public abstract void hienthi();
public float getLuong() {
    return luong;
}
public void setLuong(float luong) {
    this.luong = luong;
}
public String getName() {
}

```

```

        return name;
    }

public void setName(String name) {
    this.name = name;
}

public void nhap() {
    Scanner sc=new Scanner(System.in);
    System.out.println("Nhập vào tên ");
    name=sc.nextLine();
    earnings();
}

public void display() {
    System.out.println("Name :" +name);
    hienthi();
}

}

```

Bước 2: Tạo lớp Boss kế thừa từ Employee

```

public class Boss extends Employee {
    private float luong;
    public float getLuong() {
        return luong;
    }
    public void setLuong(float luong) {
        this.luong = luong;
    }
    @Override
    public float earnings() {
        return luong=4500;
    }
    @Override
    public void hienthi() {
        System.out.println("luong :" +luong);
    }
}

```

Bước 3 : Xây dựng lớp PieceWorker kế thừa Employee

```

public class PieceWorker extends Employee {
    float soluongsanpham;
    float luong;
    @Override

```

```

public float earnings() {
    Scanner sc=new Scanner(System.in);
    System.out.println("Nhập vào số lương sản phẩm");
    soluongsanpham=Float.parseFloat(sc.nextLine());
    return luong=(float) (soluongsanpham*0.5);
}
@Override
public void hienthi() {
    System.out.println("luong :" +luong);
}
}

```

Bước 4: Xây dựng lớp CommissionWorker kế thừa Employee

```

public class CommissionWorker extends Employee{
float luong;
@Override
public float earnings() {
return luong=new Boss().earnings()+new PieceWorked().earnings();
}
@Override
public void hienthi() {
    System.out.println("luong :" +luong);
}
}

```

Bước 5: Tạo class Test có Phương thức main và sử dụng.

```

public class Test {
    public static void main(String[] args) {
        Employee[] arr=new Employee[3];
        System.out.println("Nhập Thông tin boss");
        arr[0]=new Boss();
        arr[0].nhap();
        System.out.println("Nhập Thông tin PieceWorked ");
        arr[1]=new PieceWorker();
        arr[1].nhap();
        System.out.println("Nhập Thông tin CommissionWorker ");
        arr[2]=new CommissionWorker();
        arr[2].nhap();
        System.out.println("Thông tin boss");
        arr[0].display();
        System.out.println("Thông tin PieceWorked");
    }
}

```

```

        arr[1].display();
        System.out.println("Thong tin CommissstionWorked");
        arr[2].display();
    }
}

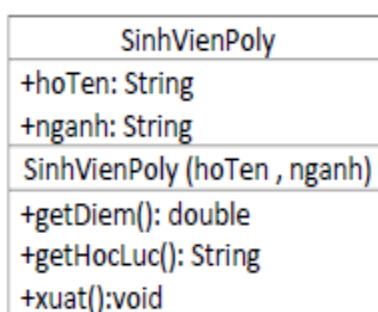
```

Bài 4.

Tạo lớp SinhVienPoly gồm hai thuộc tính họ tên và ngành cùng phương thức trừu tượng là getDiem(). Thêm phương thức getHocLuc() để xếp loại học lực. Lớp cũng bao gồm một phương thức xuất để xuất họ tên, ngành, điểm và học lực ra màn hình.

Hướng dẫn:

Xây dựng lớp SinhVienPoly có mô hình sau



Vì chưa biết sinh viên này học những môn nào nên chưa tính được điểm vì vậy phương thức getDiem() phải là trừu tượng.

Chú ý: Lớp SinhVienPoly phải là lớp trừu tượng vì có phương thức getDiem() là phương thức trừu tượng.

Phương thức getHocLuc() viết bình thường vẫn sử dụng phương thức getDiem() để lấy điểm của sinh viên. Học lực tính như sau:

Yếu: Nếu điểm <5

Trung Bình: Nếu $5 \leq \text{điểm} \leq 7$

Khá: Nếu $7 < \text{điểm} \leq 8$

Giỏi: Nếu điểm ≥ 8

Bài 5. Tạo lớp SinhVienIT và SinhVienBiz kế thừa từ lớp SinhVienPoly

SinhVienIT gồm các thuộc tính điểm java, html, css. Ghi đè phương thức getDiem() để tính điểm cho sinh viên IT theo công thức $(2*\text{java}+\text{html}+\text{css})/4$.

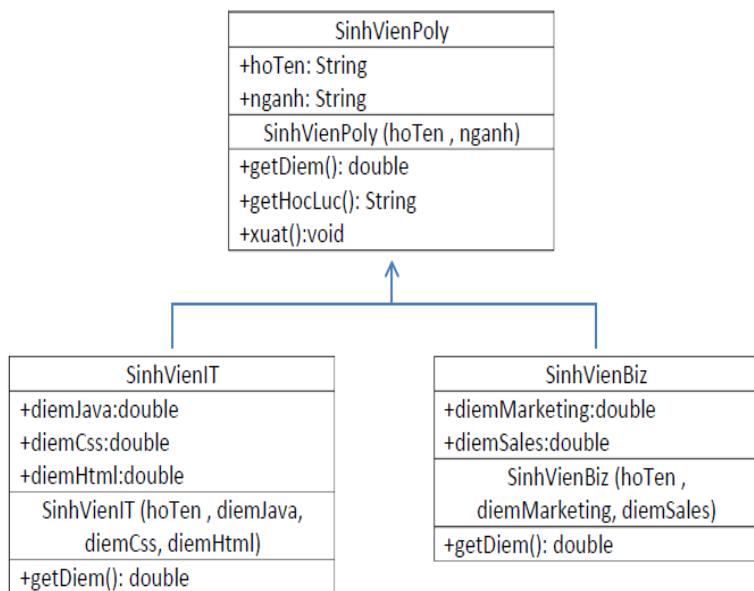
SinhVienBiz gồm các thuộc tính điểm marketing,sales. Ghi đè phương thức getDiem() để tính điểm cho sinh viên Biz theo công thức $(2*\text{marketing}+\text{sales})/3$.

Sau đó viết chương trình quản lý sinh viên gồm các chức năng sau:

1. Nhập danh sách sinh viên
2. Xuất thông tin sinh viên
3. Xuất danh sách sinh viên có học lực giỏi
4. Sắp xếp danh sách sinh viên theo điểm
5. Kết thúc.

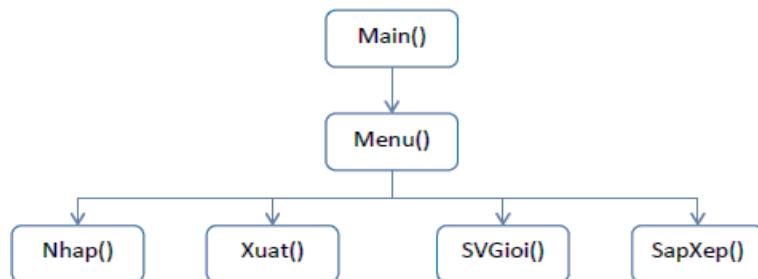
Hướng dẫn:

Bước 1: Tổ chức các lớp theo phân cấp kế thừa sau:



Ghi đè phương thức `getDiem ()` trên 2 lớp **SinhVienIT** và **SinhVienBiz** theo yêu cầu của đề để tính điểm cho sinh viên của các ngành.

Bước 2: Tổ chức chương trình theo các chức năng sau



Bài 6. Tổng hợp

Một trung tâm tin học cần quản lý giảng viên cơ hữu và giảng viên thỉnh giảng.

Giảng viên cơ hữu ký hợp đồng lao động lớn hơn một năm được hưởng thu nhập hàng tháng bao gồm lương thỏa thuận cố định và lương cộng thêm trong trường hợp vượt giờ quy định trong tháng (số giờ quy định trong tháng là 40)

Giảng viên tham gia giảng dạy thỉnh giảng ký hợp đồng lao động theo từng lớp học được hưởng thu nhập hàng tháng theo số giờ đứng lớp. Biết rằng mỗi giờ dạy có giá 200.000đ.

Thông tin giảng viên cơ hữu: tên giảng viên, email, địa chỉ, điện thoại, số giờ giảng dạy trong tháng, lương thỏa thuận và số giờ quy định chung trong tháng.

Thông tin giảng viên thỉnh giảng: tên giảng viên, email, địa chỉ, điện thoại, cơ quan làm việc, số giờ giảng dạy trong tháng

Xây dựng chương trình cho phép nhân viên trong trung tâm thực hiện các chức năng sau:

1. Nhập vào thông tin của giảng viên
2. Xuất danh sách toàn bộ giảng viên
3. Xuất danh sách giảng viên cơ hữu

4. Xuất danh sách giảng viên thỉnh giảng
5. Tính tổng số tiền lương của toàn bộ giảng viên
6. Tìm giảng viên có tổng lương cao nhất.

Hướng dẫn:

Bước 1: Tạo lớp trừu tượng GiangVien

Do chưa biết GiangVien thuộc loại nào nên có các phương thức trừu tượng sau: Nhập giảng viên, hiển thị, tính lương.

```

public abstract class GiangVien {
    private String tenGiangVien;
    private String email;
    private String diachi;
    private String dienThoai;
    private float tongLuong;
    private int soGioGiang;

    Chèn Phương thức tạo không tham số, có tham số set, get
    public abstract GiangVien inputq();
    public abstract void displayq();
    public abstract void tinhluong();
    public void input(){
        Sử dụng Scanner nhập dữ liệu
    }
}
public void display(){
    Hiển thị toàn bộ các thuộc tính. Dùng System.out.print
}
}

```

Bước 2: Tạo lớp GiangVienCoHuu kế thừa từ lớp GiangVien

```

public class GiangVienCoHuu extends GiangVien{
    private float luongThoaThuan;
    private int soGioQuyDinh=40;
    public GiangVienCoHuu() {
    }
    public GiangVienCoHuu(float luongThoaThuan, int soGioQuyDinh) {
        this.luongThoaThuan = luongThoaThuan;
        this.soGioQuyDinh = soGioQuyDinh;
    }
    public GiangVienCoHuu(float luongThoaThuan, int soGioQuyDinh,
    String tenGiangVien, String email, String diachi, String dienThoai, float
    tongLuong, int soGioGiang) {
        super(tenGiangVien, email, diachi, dienThoai, tongLuong, soGioGiang);
    }

```

```

        this.luongThoaThuan = luongThoaThuan;
        this.soGioQuyDinh = soGioQuyDinh;
    }
    @Override
    public GiangVien inputq() {
        super.input();
        Scanner sc=new Scanner(System.in);
        System.out.println("Nhập lương thỏa thuận:");
        this.luongThoaThuan=Float.parseFloat(sc.nextLine());
        return this;
    }
    @Override
    public void displayq() {
        super.display();
        System.out.println("lương thỏa thuận :" +luongThoaThuan);
        System.out.println("số giờ quy định :" +soGioQuyDinh);
    }
    @Override
    public void tinhluong() {
        float luong;
        luong=luongThoaThuan+(this.getSoGioGiang()-40)*200000;
        this.setTongLuong(luong);
    }
}

```

Bước 3: Tạo lớp GiangVienThinhGiang kế thừa GiangVien

```

public class GiangVienThinhGiang extends GiangVien{
    private String coQuanLamViec;
    @Override
    public GiangVien inputq() {
        super.input();
        Scanner sc=new Scanner(System.in);
        System.out.println("Nhập vào cơ quan làm việc :");
        coQuanLamViec=sc.nextLine();
        return this;
    }
    @Override
    public void displayq() {
        super.display();
        System.out.println("cơ quan làm việc :" +coQuanLamViec);
    }
}

```

```

@Override
public void tinhluong() {
    float luong;
    luong=this.getSoGioGiang()*200000;
    this.setTongLuong(luong);
}
}

```

Bước 4: Tạo lớp QuanLiGiaoVien có Phương thức main

```

public class QuanLiGiaoVien {
    public static void menu1(){
        System.out.println("1 Nhập thông tin giảng viên có hưu");
        System.out.println("2 nhập thông tin giảng viên thỉnh giảng");
        System.out.println("3 hiển thị toàn bộ thông tin");
        System.out.println("4 hiển thị thông tin giảng viên có hưu");
        System.out.println("5 hiển thị thông tin GV thing giang");
        System.out.println("6 tổng lương của toàn bộ giảng viên");
        System.out.println("7 lương cao nhất");
        System.out.println("8 thoát");
    }
    public static void main(String[] args) {
        int n = 0;
        ArrayList<GiangVien> list=new ArrayList<GiangVien>();
        do {
            menu1();
            System.out.println("Nhập vào lựa chọn của bạn :");
            Scanner sc = new Scanner(System.in);
            n = Integer.parseInt(sc.nextLine());
            switch (n) {
                case 1: {
                    System.out.println("nhập giảng viên có hưu :");
                    list.add(new GiangVienCoHuu().inputq());
                    break;
                }
                case 2: {
                    System.out.println("nhập giảng viên thỉnh giảng :");
                    list.add(new GiangVienThinhGiang().inputq());
                    break;
                }
                case 3: {
                    if(list==null){

```

```

        System.out.println("ban chua nhap du lieu");
    }
    else{
        System.out.println("thong tin giang vien vua nhap la :");
        int i=0;
        for (GiangVien e:list) {
            System.out.println("thong tin giang vien thu "+(i+1));
            i++;
            e.displayq();
        }
    }
    break;
}
case 4:
{
    if(list==null){
        System.out.println("ban chua nhap du lieu");
    }
    else{
        int dem=0;
        System.out.println("danh sach giang vien co huu");
        for (GiangVien i : list) {
            if (i instanceof GiangVienCoHuu) {
                i.displayq();
                dem++;
            }
        }
        if(dem==0){
            System.out.println("khong co san pham ban muon tim");
        }
    }
    break;
}
case 5:
{
    if(list==null){
        System.out.println("ban chua nhap du lieu");
    }
    else{
        int dem=0;

```

```

        System.out.println("danh sach giang vien thinh giang");
            for (GiangVien i : list) {
                if (i instanceof GiangVienThinhGiang) {
                    i.displayq();
                    dem++;
                }
            }
            if(dem==0){
                System.out.println("khong co san pham ban muon tim");
            }
        }
        break;
    }
case 6:{

    if(list==null){
        System.out.println("ban chua nhap du lieu");
    }
    else{
        float tong=0;
        System.out.println("tong luong cua toan bo giao vien la :");
        for (GiangVien i:list) {
            tong+=i.getTongLuong();
        }
        System.out.println(tong);
    }
    break;
}
case 7:{

    if(list==null){
        System.out.println("ban chua nhap du lieu");
    }
    else{
        System.out.println("giang vien co luong cao nhat");
        float max=0;
        for (GiangVien i:list) {
            if(max<i.getTongLuong()){
                max=i.getTongLuong();
            }
        }
        for (GiangVien i:list) {

```

```
        if(max==i.getTongLuong()){
            i.displayq();
        }
    }
}
case 8:
break;
default:{
    System.out.println("khong co lua chon cua ban ");
    break;
}
}
} while (n != 8);
}
}
```

LAB 4. ARRAYLIST, LINKLIST, COLLECTION, GENERIC [1, 7,13]

A. MỤC TIÊU

- Trang bị cho sinh viên cách thao tác trên ArrayList, LinkList.
- Khai thác trên tập hợp (collection):
 - a. Khai báo và khởi tạo tập hợp
 - b. Các thao tác thêm, xóa, sửa, duyệt trên tập hợp
 - c. Các thuật toán sắp xếp, tìm kiếm trên tập hợp
- Triển khai các kiểu Generic, phương thức Generic
- Áp dụng được Generic theo lớp.

B. NỘI DUNG

Thực hành các bài toán dùng ArrayList, LinkList, TreeSet, HashMap, Generic

C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB.
- NETBEAN IDE 8.0, JDK 1.8.

D. KẾT QUẢ SAU KHI HOÀN THÀNH

Sử dụng Array List, LinkList, TreeSet, HashMap áp dụng vào các bài toán.

Sử dụng Generic thực hiện các thuật toán tổng quát với các kiểu dữ liệu khác nhau.

E. HƯỚNG DẪN CHI TIẾT

1. ArrayList

Lớp ArrayList kế thừa AbstractList và triển khai List Interface. Hỗ trợ tạo mảng động có thể tăng kích cỡ nếu cần.

Khởi tạo ArrayList

```
ArrayList list = new ArrayList(); //non-generic - kiểu cũ  
ArrayList<String> list = new ArrayList<String>(); //generic - kiểu mới
```

Bài 1. Viết chương trình quản lý sinh viên dưới dạng console, yêu cầu sử dụng ArrayList và thực hiện các chức năng sau (thông tin sinh viên bao gồm : mã số sinh viên, họ tên, năm sinh, địa chỉ, lớp học):

- a) Cho phép thêm, sửa, xóa danh sách sinh viên
- b) Xuất ra số lượng sinh viên
- c) Xuất ra danh sách các sinh viên thuộc một lớp học bất kỳ nhập vào từ bàn phím.

Hướng dẫn:

Bước 1: Tạo Class Sinhvien

```
import java.util.Date;
```

```

import java.util.Date;
public class Sinhvien {
    private String Masv;
    private String Tensv;
    private Date Namsinh;
    private String Diachi;
    private String Lop;
    Chèn các phương thức khởi tạo, set, get các giá trị cho các thuộc tính
    public String toString() {
        return "Sinhvien [Masv=" + Masv + ", Tensv=" + Tensv + ", Namsinh="
        + Namsinh + ", Diachi=" + Diachi + ", Lop=" + Lop + "]";
    }
}

```

Bước 2: Tạo Class DanhSachSinhvien

```

public class DanhSachSinhvien {
    private ArrayList<Sinhvien> dsSv=new ArrayList<Sinhvien>();
    public boolean ktTrungma(String masv)
    {
        for(Sinhvien sv : dsSv)
        {
            if(sv.getMasv().equalsIgnoreCase(masv))
                return true;
        }
        return false;
    }
    public boolean addSinhvien(Sinhvien sv)
    {
        if(ktTrungma(sv.getMasv()))
            return false;
        return dsSv.add(sv);
    }
    public Sinhvien findSinhvien1(String masv)
    {
        for(Sinhvien sv : dsSv)

```

```

        {
            if(sv.getMasv().equalsIgnoreCase(masv))
                return sv;
        }
        return null;
    }

    public int findSinhvien2(String masv)
    {
        for(int i=0;i<dsSv.size();i++)
        {

            if(dsSv.get(i).getMasv().equalsIgnoreCase(masv))
                return i;
        }
        return -1;
    }

    public Sinhvien updateSinhvien(int index,Sinhvien sv)
    {
        return dsSv.set(index, sv);
    }

    public void removeSinhvien(String masv)
    {
        Sinhvien sv=findSinhvien1(masv);
        dsSv.remove(sv);
    }

    public String toString() {
        return dsSv.toString();
    }
}

```

Bước 3: Tạo Class TestSinhvien

```

public class TestSinhvien {
    public static void main(String[] args) {
        DanhSachSinhvien qlsv=new DanhSachSinhvien();
        Sinhvien teo=new Sinhvien();
    }
}

```

```

        teo.setMasv("113");
        teo.setTensv("Nguyễn Văn Tèo");
        qlsv.addSinhvien(teo);
        Sinhvien ty=new Sinhvien();
        ty.setMasv("114");
        ty.setTensv("Nguyễn Thị tí");
        qlsv.addSinhvien(ty);
        System.out.println(qlsv);
    }
}

```

Hướng dẫn: Để sắp xếp được các sinh viên theo mã số thì **class** Sinh viên phải **implements interface Comparable** đồng thời override phương thức **compareTo**.

```

public class Sinhvien implements Comparable<Sinhvien>{
    private String Masv;
    public String getMasv() {
        return Masv;
    }
    public int compareTo(Sinhvien o) {
        if(Masv.equalsIgnoreCase(o.getMasv()))
            return 0;
        return 1;
    }
}

```

Bước 4: Tạo Class DanhSachSinhVien

```

import java.util.ArrayList;
import java.util.Collections;
public class DanhSachSinhvien {
private ArrayList<Sinhvien>dsSv=new ArrayList<Sinhvien>();
    public void sort() {
        Collections.sort(dsSv);
    }
}

```

Muốn sắp xếp danh sách, gọi phương thức sort trong class trên.

```

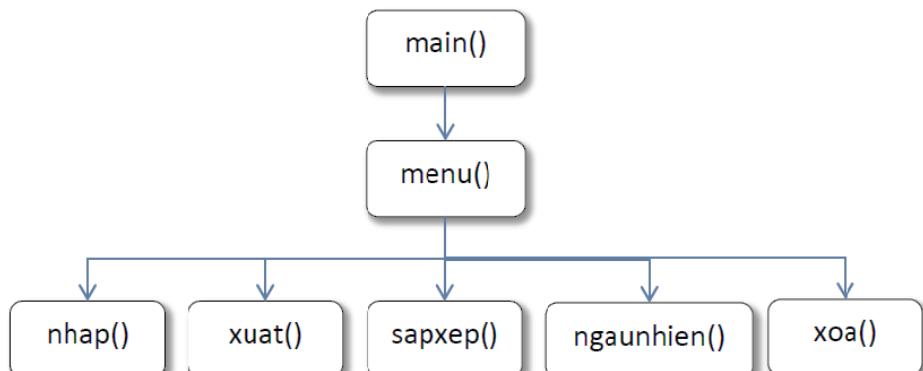
public class TestSinhvien {
    public static void main(String[] args) {
        DanhSachSinhvien qlsv=new DanhSachSinhvien();
        Sinhvien teo=new Sinhvien();
        teo.setMasv("113");
        teo.setTensv("Nguyễn Văn Tèo");
        qlsv.addSinhvien(teo);
        Sinhvien ty=new Sinhvien();
        ty.setMasv("114");
        ty.setTensv("Nguyễn Thị tí");
        qlsv.addSinhvien(ty);
        qlsv.sort();
        System.out.println(qlsv);
    }
}

```

Bài 2. Xây dựng ứng dụng quản lý sản phẩm (thông tin mỗi sản phẩm gồm tên và giá) theo menu sau.

1. Nhập danh sách sản phẩm từ bàn phím
2. Xuất danh sách vừa nhập.
3. Xuất danh sách ngẫu nhiên.
4. Sắp xếp giảm dần theo giá và xuất giá ra màn hình
5. Tìm và xóa sản phẩm theo tên nhập từ bàn phím.
6. Xuất giá trung bình của các sản phẩm

Hướng dẫn: Tổ chức ứng dụng theo sơ đồ sau



Tạo lớp SanPham có thuộc tính theo đề bài

Sử dụng ArrayList<SanPham> để duy trì danh sách số sản phẩm nhập từ bàn phím

Sử dụng vòng lặp while để nhập số lượng tùy ý.

```

while (true) {
    SanPham sp= new SanPham();
    Sp.nhap();
}

```

```

list.add (sp)
System.out.print ("Nhập thêm Y/N?");
if(Scanner.nextLine().equals("N"))
{
    break;
}

```

Sử dụng vòng lặp **for-each** để duyệt và xuất các phần tử của list ra màn hình.

Sử dụng phương thức Collections.shuffle(list) hoán đổi ngẫu nhiên các phần tử trong list.

Sử dụng Collections.sort(list, comp) để sắp xếp danh sách sản phẩm theo tiêu chí sắp xếp được định nghĩa như sau:

```

Comparator<SanPham> comp= new Comparator<SanPham>()
{
    @Override
    public int compare(SanPham o1, SanPham o2)
    {
        return o1.dongia-o2.dongia;
    }
}

```

Duyệt list, dùng list.remove() để xóa sản phẩm, dùng **break** để ngắt vòng lặp sau khi xóa.

Bài 3.

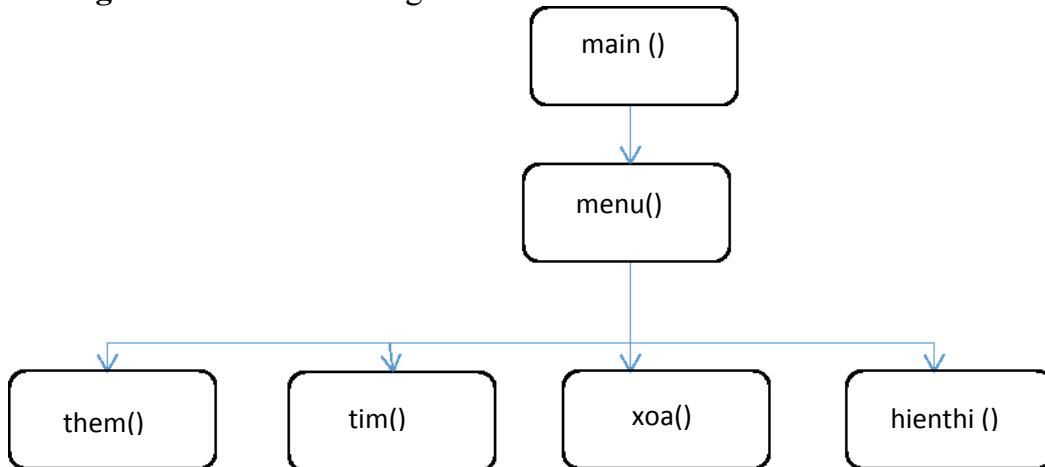
Xây dựng lớp EmployeeList sử dụng ArrayList. Lớp Employee gồm các trường (Mã NV, Tên NV, Chức vụ, Lương, Thời gian bắt đầu làm)

1. Khai báo 1 đối tượng ArrayList trong lớp EmployeeList
2. Xây dựng 1 constructor không tham số khởi tạo đối tượng ArrayList
3. Xây dựng 1 phương thức có tên là add không tham số cho phép người dùng nhập tên của nhân viên từ bàn phím và thêm vào ArrayList
4. Xây dựng 1 phương thức có tên là remove cho phép người dùng xóa nhân viên theo mã nhân viên ArrayList
5. Xây dựng phương thức hiển thị hiển thị tất cả tên các nhân viên trong ArrayList
6. Xây dựng phương thức tìm kiếm không tham số cho phép người dùng nhập tên 1 nhân viên từ bàn phím và tìm trong ArrayList xem có nhân viên đó không.

Xây dựng lớp test có menu như sau:

1. Thêm nhân viên vào danh sách
2. Tìm kiếm nhân viên theo tên
3. Xóa nhân viên ra khỏi danh sách
4. Hiển thị tất cả danh sách nhân viên
5. Thoát

Hướng dẫn: Tổ chức chương trình theo sơ đồ sau.



2. LinkedList

- Có thể chứa các phần tử trùng lặp.
- Duy trì thứ tự của phần tử được thêm vào.
- Có thể được sử dụng như list (danh sách), stack (ngăn xếp) hoặc queue (hàng đợi).

Khởi tạo LinkedList

```
LinkedList list = new LinkedList(); // non-generic - kiểu cũ  
LinkedList<String> list = new LinkedList<String>(); // generic - kiểu mới
```

Xây dựng lớp EmployeeList sử dụng LinkedList.

1. Khai báo 1 đối tượng LinkedList trong lớp EmployeeList
2. Xây dựng 1 constructor không tham số khởi tạo đối tượng LinkedList
3. Xây dựng 1 phương thức có tên là add không tham số cho phép người dùng nhập tên của nhân viên từ bàn phím và thêm vào LinkedList.
4. Xây dựng phương thức hiển thị hiển thị tất cả tên các nhân viên trong LinkedList
5. Xây dựng phương thức tìm kiếm không tham số cho phép người dùng nhập tên 1 nhân viên từ bàn phím và tìm trong LinkedList xem có nhân viên đó không.

Xây dựng lớp test có menu như sau:

1. Thêm employee vào danh sách
2. Tìm kiếm employee theo tên
3. Hiển thị danh sách
4. Exit

3. Collections (Tập hợp)

Để làm việc với các dữ liệu tập hợp khác nhau, API JAVA cung cấp lớp Collections trong package java.util.

Set **Interface** là một loại **Interface** Collection, các phần tử trong Set là duy nhất.

Để khai báo một Set cần dùng đến các **class** để triển khai nó: 2 loại phổ biến nhất là HashSet (các phần tử không được sắp xếp thứ tự) và TreeSet (thứ tự các phần tử trong Set được sắp xếp tăng dần).

Bài 4. Dùng TreeSet tạo tập hợp Student, mỗi Student gồm các thuộc tính: name, age, address. Sắp xếp tập Student theo tên tăng dần theo alphabet và in ra màn hình

Bước 1: Tạo class Student

```
import java.util.TreeSet;
class Student implements Comparable<Student> {
    private String name;
    private int age;
    private String address;
    public Student() {
    }
    Chèn Phương thức tạo, các phương thức set/get thuộc tính
    @Override
    public String toString() {
        return "Student@name=" + name + ",age=" + age + ",address=" + address;
    }
    @Override
    public int compareTo(Student student) {
        // sort student's name by ASC
        return this.getName().compareTo(student.getName());
    }
}
```

Bước 2: Tạo lớp TreeSetExample2 có Phương thức main

```
public class TreeSetExample2 {
    public static void main(String[] args) {
        // init treeSet
        TreeSet<Student> treeSet = new TreeSet<Student>();
        // create students object
        Student student1 = new Student("Cong", 17, "Hanoi");
        Student student2 = new Student("Dung", 16, "Haiphong");
        Student student3 = new Student("Ngon", 18, "Hanoi");
        Student student4 = new Student("Hanh", 19, "Danang");
        // add students object to treeSet
        treeSet.add(student1);
        treeSet.add(student2);
        treeSet.add(student3);
        treeSet.add(student4);
        treeSet.add(student1);
        // show treeSet
        for (Student student : treeSet) {
```

```

        System.out.println(student.toString());
    }
}
}

```

4. Map Interface (ánh xạ): Định nghĩa các ánh xạ từ các khóa (keys) vào các giá trị.

Lưu ý: Các khóa phải là duy nhất. Mỗi khóa được ánh xạ sang nhiều nhất 1 giá trị, được gọi là ánh xạ đơn.

Các lớp HashMap và HashTable

Hai lớp này cài đặt giao diện Map, cho phép tạo ra ánh xạ mới có thể rỗng hoặc có kích thước tùy ý.

Bài 5. Sử dụng HashMap tạo danh sách sản phẩm gồm có mã sản phẩm, tên sản phẩm. Thêm vào danh sách sản phẩm đó sản phẩm nhập từ bàn phím. In lại danh sách sản phẩm sau khi thêm.

```

public static void main(String[] args) {
    int soSanPham = 2;
    HashMap<String, String> hashMapProducts = new HashMap<>();
    Scanner Scanner = new Scanner(System.in);
    String maSanPham, tenSanPham;
    // thêm thông tin của 2 sản phẩm vào trong hashMapProducts
    // trong đó key là mã sản phẩm, còn value là tên của sản phẩm đó
    for (int i = 1; i <= soSanPham; i++) {
        System.out.println("Nhập thông tin của sản phẩm thứ " + i);
        System.out.println("Nhập mã sản phẩm: ");
        maSanPham = Scanner.nextLine();
        System.out.println("Nhập tên sản phẩm: ");
        tenSanPham = Scanner.nextLine();
        hashMapProducts.put(maSanPham, tenSanPham);
    }
    // hiển thị danh sách sản phẩm sử dụng Iterator
    System.out.println("Danh sách các sản phẩm vừa nhập: ");
    System.out.println("Mã sản phẩm\tTên sản phẩm");
    Iterator<Map.Entry<String, String>> iterator =
    hashMapProducts.entrySet().iterator();
    while (iterator.hasNext()) {
        // tạo 1 entry
        Map.Entry<String, String> entry = iterator.next();
        System.out.println(entry.getKey() + "\t\t" + entry.getValue());
    }
    // thêm 1 sản phẩm mới vào trong hashMapProducts

```

```

System.out.println("Nhập mã sản phẩm cần thêm: ");
String maSanPhamMoi = Scanner.nextLine();
if (hashMapProducts.containsKey(maSanPhamMoi)) {
    System.out.println("Mã sản phẩm = " + maSanPhamMoi + " đã tồn tại!");
} else {
    System.out.println("Nhập tên sản phẩm cần thêm: ");
    String tenSanPhamMoi = Scanner.nextLine();
    hashMapProducts.put(maSanPhamMoi, tenSanPhamMoi);
    soSanPham++;
    System.out.println("Danh sách các sản phẩm sau khi thêm: ");
    System.out.println("Số sản phẩm = " + soSanPham);
    System.out.println("Mã sản phẩm\tTên sản phẩm");
    iterator = hashMapProducts.entrySet().iterator();
    while (iterator.hasNext()) {
        // tạo 1 entry
        Map.Entry<String, String> entry = iterator.next();
        System.out.println(entry.getKey() + "\t" + entry.getValue());
    }
}
}

```

Bài 6.

Tạo lớp DoctorDetails trong package lifeline

1. Khai báo 4 biến lưu trữ Doctor code (**String**), Doctor Name(**String**), specialization of the Doctor (**String**), availability of the Doctor in hours(**int**).
2. Xây dựng 2 constructor, 1 không tham số và 1 có 4 tham số.
3. Viết đè phương thức **toString()** để hiển thị :

```

"\nDoctor code = " + docCode + "\nName = " + name + "\nSpecialization =
" + specialization + "\nAvailability = " + hours

```

Tạo lớp DoctorHash trong package lifeline demo về collection HashMap.

1. Khai báo 1 đối tượng static HashMap
2. Xây dựng 1 constructor không tham số khởi tạo hashMap
3. Xây dựng 1 phương thức add không tham số:
 - a. Khai báo 4 biến lưu trữ Doctor code (**String**), Doctor Name(**String**), specialization of the Doctor (**String**), availability of the Doctor in hours(**int**) và nhập các biến này từ bàn phím
 - b. Tạo 1 đối tượng DoctorDetails và đưa các thông số vào đối tượng. Sau đó thêm đối tượng này vào HashMap.

4. Phương thức tìm kiếm không tham số cho phép người dùng nhập Mã của doctor từ bàn phím và hiển thị thông tin đó ra nếu tìm thấy.
5. Phương thức hiển thị toàn bộ thông tin trong HashMap sử dụng iterator

Tạo lớp Test: xây dựng menu để thực hiện các chức năng trên

Hướng dẫn:

Bước 1: Tạo lớp DoctorDetails

```
public class DoctorDetails
{
    String code;
    String name;
    String specialization;
    int availability;

    Chèn phương thức khởi tạo không tham số và 4 tham số

    @Override
    public String toString()
    {
        return "Doctor Code:" +code+"\n"+ "Doctor
Name:" +name+ "\n" + "Specialization: " + specialization + "\n" +
"Availability: " + availability;
    }
}
```

Bước 2: Tạo class DoctorHash

```
public class DoctorHash
{
    static HashMap<String, DoctorDetails> elements;
    public DoctorHash()
    {
        elements = new HashMap<>();
    }
    public void Add()
    {
        Scanner Scanner = new Scanner(System.in);
        String code, name, specialization;
        int availability;
        System.out.print("Nhập code: ");
    }
}
```

```

        code = Scanner.nextLine();
        System.out.print("Nhập tên: ");
        name = Scanner.nextLine();
        System.out.print("Nhập chuyên môn: ");
        specialization = Scanner.nextLine();
        availability = InputNumberFormat("Nhập giờ làm việc: ");
        DoctorDetails doctorDetails = new DoctorDetails(code, name,
specialization, availability);
        elements.put(doctorDetails.code, doctorDetails);
    }
public void Search()
{
    Scanner Scanner = new Scanner(System.in);
    System.out.print("Nhập code cần tìm: ");
    String code = Scanner.nextLine();
    if(elements.containsKey(code)==false)
    {
        System.out.println("Không tìm thấy dữ liệu!");
    }else
    {
        System.out.println("Thông tin bác sĩ tìm thấy:");
        System.out.println(elements.get(code).toString());
    }
}
public void ShowAll()
{
    System.out.println("*****Thông tin tất cả bác sĩ*****");
    Iterator it = elements.entrySet().iterator();
    int index = 1;
    while(it.hasNext())
    {
        System.out.println("Bác sĩ " + index);
        HashMap.Entry entry = (HashMap.Entry)it.next();
        System.out.println(entry.getValue().toString() + "\n");
    }
}

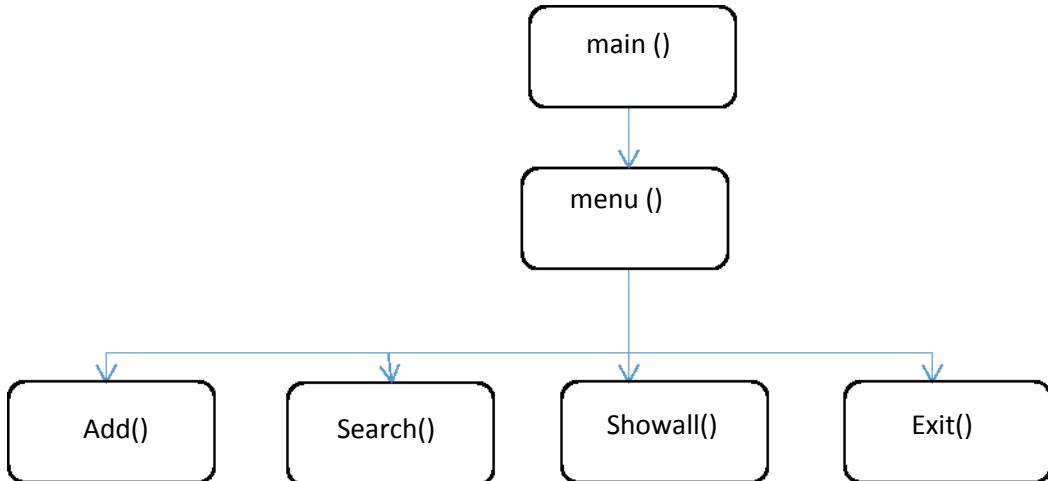
```

```

        index++;
    }
}

```

Xây dựng menu chương trình với các chức năng sau, sử dụng switch:



6. Lập trình Generic

Cho phép lập trình viên thực hiện các thuật toán tổng quát. Bằng cách dùng Generic nội dung đoạn code trở nên dễ hiểu và rõ ràng.

Bài 7. Sử dụng interface Generic Comparable. Tạo một lớp Person bao gồm 2 thuộc tính họ và tên implements interface Comparable.

Bước 1: Tạo lớp Person implements Generic Comparable

```

import java.util.Arrays;

class Person implements Comparable<Person>
{
    private String firstName;
    private String surname;
    public Person(String firstName, String surname)
    {
        this.firstName = firstName;
        this.surname = surname;
    }
    public String toString()
    {
        return firstName + " " + surname;
    }
    public int compareTo(Person person)
    {
        if (this.firstName < person.firstName)
            return -1;
        else if (this.firstName > person.firstName)
            return 1;
        else
            return 0;
    }
}

```

```

    {
        int result = surname.compareTo(person.surname);
        return result == 0 ? firstName.compareTo(((Person)
            person).firstName):result;
    }
}

```

Bước 2: Tạo lớp kiểm tra

```

public class PersonTest
{
    public static void main(String[] args)
    {
        Person[] authors = {
            new Person("D", "S"),
            new Person("J", "G"),
            new Person("T", "C"),
            new Person("C", "S"),
            new Person("P", "C"),
            new Person("B", "B") };

        Arrays.sort(authors); // Sắp xếp sử dụng phương thức Comparable
        System.out.println("\nSau khi sap xep:");
        for (Person author : authors)
        {
            System.out.println(author);
        }

        Person[] people = {
            new Person("C", "S"),
            new Person("N", "K"),
            new Person("T", "C"),
            new Person("C", "D") };

        int index = 0;
        System.out.println("\nTim kiem:");
        for (Person person : people)
        {
            index = Arrays.binarySearch(authors, person);
            if (index >= 0)
            {
                System.out.println(person + " tai vi tri index " + index);
            }
        }
    }
}

```

```

        else
        {
System.out.println(person +"khong tim thay. Gia tri tra ve: " + index);
        }
    }
}
}

```

Bài 8. Sử dụng bounded wildcard trong phương thức. Viết phương thức Generic cho phép tính trung bình các giá trị trong mảng.

Hướng dẫn:

```

import java.util.ArrayList;
import java.util.List;
public class BoundedWildcard
{
    public static double getAverage(List<? extends Number> numberList)
    {
        double total = 0.0;
        for (Number number : numberList)
        {
            total += number.doubleValue();
        }
        return total / numberList.size();
    }
    public static void main(String[] args)
    {
        List<Integer> IntegerList = new ArrayList<Integer>();
        IntegerList.add(3);
        IntegerList.add(30);
        IntegerList.add(300);
        System.out.println(getAverage(IntegerList)); // KQ?
        List<Double> doubleList = new ArrayList<Double>();
        doubleList.add(3.0);
        doubleList.add(33.0);
        System.out.println(getAverage(doubleList));
    }
}

```

Bài 9. Sử dụng bounded type trong lớp. Viết lớp Generic cho phép tính trung bình các giá trị trong mảng số (số nguyên/số thực).

Hướng dẫn:

```

class Stats<T extends Number>
{
    T[] nums;

    Stats(T[] o)
    {
        nums = o;
    }

    double average()
    {
        double sum = 0.0;
        for(int i=0; i < nums.length; i++) sum += nums[i].doubleValue();
        return sum / nums.length;
    }
}

public class BoundedType
{
    public static void main(String args[])
    {
        Integer inums[] = { 1, 2, 3, 4, 5 };
        Stats<Integer> iob = new Stats<Integer>(inums);
        double v = iob.average();
        System.out.println("Trung bình iob: " + v);
        Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
        Stats<Double> dob = new Stats<Double>(dnums);
        double w = dob.average();
        System.out.println("Trung bình dob: " + w);
    }
}

```

Bài 10. Viết phương thức override trong lớp Generic. Xét trường hợp lớp B thừa kế lớp A và viết override phương thức trong lớp A.

```

class Gen<T>
{
    T ob;
    Gen(T o)
    {

```

```

        ob = o;
    }

    T getObject()
    {
        System.out.println("Gen's getObject(): " );
        return ob;
    }
}

class Gen2<T> extends Gen<T>
{

    Gen2(T o)
    {
        super(o);
    }
    T getObject()
    {
        System.out.println("Gen2's getObject(): " );
        return ob;
    }
}

public class OverrideGenericMethods
{
    public static void main(String[] arg)
    {
        Gen<Integer> Intobject = new Gen<Integer>(88);
        Gen2<Long> longObject = new Gen2<Long>(99L);
        Intobject.getObject();
        longObject.getObject();
    }
}

```

Bài 11. Bài tập về Collection dùng TreeMap:

Viết 1 lớp mô tả 1 từ tiếng Anh bao gồm từ, nghĩa, loại từ, và phần ghi chú, lớp cần override phương thức `toString`, `equals` và phương thức so sánh 2 từ `compareTo` không phân biệt chữ thường hoa.

Lớp từ điển bao gồm các phương thức:

- Thêm từ điển mới
- Tra từ điển

- `toString()` để in ra tất cả các từ trong từ điển

Lớp kiểm tra cho phép nhập vào các từ và tra cứu các từ đó

Hướng dẫn:

Bước 1: Tạo lớp EVWordClass bao gồm các thuộc tính (từ, nghĩa, loại từ, ghi chú), phương thức get/set, constructors, phương thức so sánh equals, `toString`

```
public class EVWordClass implements Comparable
{
    private String word;
    private String mean;
    private String type;
    private String notes;

    Chèn tự động các Phương thức tạo, set,get các giá trị cho các thuộc
    tính.

    public boolean equals(Object obj)
    {
        EVWordClass w = (EVWordClass) obj;
        return word.equalsIgnoreCase(w.getWord());
    }

    public String toString()
    {
        return word + " ; " + type + " ; " + mean + " ; " + notes;
    }

    public int compareTo(Object o)
    {
        return
    this.word.compareToIgnoreCase(((EVWordClass)o).getWord());
    }
}
```

Bước 2: Tạo lớp từ điển sử dụng Collections TreeMap

```
import java.util.TreeMap;
public class EVDictionary
{
    public TreeMap<String, EVWordClass> dic;
    public EVDictionary()
    {
        dic = new TreeMap<String, EVWordClass>();
    }

    // Thêm từ mới vào từ điển
    public boolean addWord(EVWordClass word)
```

```

{
    if(dic.put(word.getWord().toLowerCase(),word) != null)
        return false;
    return true;
}
// Tra tu
public EVWordClass lookup(String word)
{
    return dic.get(word);
}
public String toString()
{
    String ret = "";
    for(EVWordClass w:dic.values()) ret += w.toString()+"\n";
    return ret;
}
}

```

Bước 3: Lớp kiểm tra chương trình

```

public class EVDictionaryTest
{
    public static void main(String[] args)
    {
        EVDictionary dic = new EVDictionary();
        for(int i=1; i<10; i++)
        {
            dic.addWord(new EVWordClass("Word" + i, "", "Tu thu " + i, ""));
        }
        System.out.println(dic);
        //Them tu
        EVWordClass w = new EVWordClass("Word2", "", "Tu thu ", "");
        if(!dic.addWord(w))
            System.out.println("Khong them duoc!");
        //Tra tu
        EVWordClass l = dic.lookup("word2");
        if(l != null)
            System.out.println(l.toString());
    }
}

```

```
    }  
}
```

Bài 12. Tổng hợp

1. Xây dựng lớp Employee trong package companydetail. Lớp này có 3 trường như sau:

- Biến **String** lưu trữ tên nhân viên
- Biến **String** lưu trữ mã nhân viên
- Biến **String** lưu trữ chức vụ của nhân viên

Xây dựng 2 constructor, 1 không tham số và 1 có 3 tham số.

Viết lại phương thức **toString()** để hiển thị thông tin chi tiết của nhân viên

2. Xây dựng lớp HREmployee trong package companydetail kế thừa từ lớp Employee

Lớp này có thêm 2 trường:

- Biến **String** lưu trữ tên phòng ban
- Biến **String** lưu trữ mã phòng ban

Xây dựng 2 constructor của lớp này.

Viết lại phương thức **toString()** để hiển thị chi tiết lớp này

3. Xây dựng tiếp lớp EmployeeDetail trong package companydetail. Lớp này hiển thị chi tiết các thông tin của lớp Employee và lớp HREmployee

Viết 1 phương thức **printCollection** có 1 tham số là Collection hiển thị chi tiết tất cả các nhân viên trong tập hợp này.

Viết tiếp 1 phương thức **printDerivedCollection** có 1 tham số là Collection cho phép nhập vào 1 tập hợp là các đối tượng kế thừa từ lớp Employee và hiển thị chi tiết các nhân viên trong lớp kế thừa Employee.

4. Tạo lớp Test

Khai báo 1 đối tượng thuộc lớp ArrayList có tham số kiểu là Employee. Thêm 2 Employee vào đối tượng này và hiển thị theo phương thức **printCollection** của lớp EmployeeDetail

Tương tự khai báo đối tượng ArrayList có tham số kiểu là HREmployee và gọi phương thức **printDerivedCollection** của lớp EmployeeDetail.

Hướng dẫn:

Bước 1: Tạo lớp Employee

```
public class Employee  
{  
    String ten, ma, chucVu;  
    Chèn tự động Phương thức tạo không, ba tham số.  
    static String StringFormat = "%1$-20s %2$-10s %3$-15s";  
    public String toString()  
    {  
        return String.format(StringFormat, ten, ma, chucVu);  
    }  
}
```

```
}
```

Bước 2: Tạo lớp HREmployee kế thừa từ Employee

```
public class HREmployee extends Employee
{
    String tenPhongBan, maPhongBan;
    Chèn tự động Phương thức tạo không tham số,5 tham số trong đó 2 tham số
    thuộc lớp HREmployee, 3 tham số còn lại thuộc lớp Employee.
    static String StringFormat = "%1$-20s %2$-10s";
    public String toString()
    {
        return super.toString() + String.format(StringFormat, tenPhongBan,
maPhongBan);
    }
}
```

Bước 3: Tạo lớp EmployeeDetail

```
public class EmployeeDetail
{
    public void printCollection(Collection c)
    {
        for (Object object : c)
        {
            System.out.println(object.toString());
        }
    }
    public void printDerivedCollection(Collection c)
    {
        for (Object object : c)
        {
            System.out.println(object.toString());
        }
    }
}
```

Bước 4:Tạo lớp Test

```
public class Test
{
    public static void main(String[] args)
    {
        EmployeeDetail employeeDetail = new EmployeeDetail();
        ArrayList<Employee> listEmployee = new ArrayList<>();
```

```
listEmployee.add(new Employee("Hùng", "EP1", "Quản lý"));
listEmployee.add(new Employee("An", "EP2", "Thư ký"));
employeeDetail.printCollection(listEmployee);
ArrayList<Employee> listHREmployee = new ArrayList<>();
listHREmployee.add(new HREmployee("Ban cỗ vấn", "B1", "Lương",
"EP3", "Quản lý"));
listHREmployee.add(new HREmployee("Ban tuyên giáo", "B2",
"Trang", "EP4", "Quản lý"));
employeeDetail.printDerivedCollection(listHREmployee);
}
}
```

LAB 5. QUẢN LÝ THREAD [11, 12]

A. MỤC TIÊU

- Hướng dẫn sinh viên cách tạo luồng.
- Khai thác giao tiếp giữa các luồng trong java.
- Khai thác sự phối hợp giữa các luồng: sử dụng từ khóa **synchronized, wait, notify**.

B. NỘI DUNG

- Tạo luồng.
- Quản lý luồng: Đồng bộ hóa luồng, mối quan hệ giữa các luồng.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

Viết chương trình mô phỏng đồng bộ luồng, áp dụng vào các bài toán cụ thể: Xử lý đồng bộ các giao dịch trong ngân hàng.

D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB.
- Phần mềm NETBEAN 8.0, JDK 1.8.

E. HƯỚNG DẪN

1. Tạo Thread

a) Tạo một lớp con kế thừa class Thread

```
public 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
```

b) Thực hiện interface Runnable

```
class B implements Runnable
{
    public void run()
    {
        //code for the new thread to execute
    }
}
B b = new B();           // Tạo đối tượng Runnable
Thread t = new Thread(b); // Tạo đối tượng luồng
t.start();                // Khởi động luồng
```

Một số phương của Thread

- **public static void sleep(long millis) throws InterruptedException**

Thread hiện tại ngừng hoạt động trong một thời gian millis.

- **public void start()**

- Tạo một đối tượng thread. JVM sẽ tự động gọi phương thức run của đối tượng

- Thread được bắt đầu hoạt động sau khi phương thức này được gọi thành công.

- **public final boolean isAlive()**

- Kiểm tra thread còn hiệu lực hay không (còn sống).

- **public final void join(long millis) throws InterruptedException**

- Đây là phương thức được gọi bởi một thread khác. Phương thức này làm cho thread gọi phải ngưng hoạt động và chờ trong một khoảng thời gian millis hoặc chờ cho đến khi thread gọi kết thúc thì mới tiếp tục hoạt động.

Đồng bộ luồng-Synchronization

Khi hai hay nhiều thread cùng sử dụng chung một biến hay một phương thức thì xảy ra vấn đề : Race condition.

- Race condition (điều kiện tranh chấp) : xảy ra khi có hai hay nhiều thread cùng chia sẻ dữ liệu, khi chúng cùng đọc, cùng ghi dữ liệu đó đồng thời.

- Sử dụng từ khoá synchronized cho phương thức hay đoạn mã cần bảo vệ.

public synchronized void protectedMethod(object obj)

Bài 1. Tạo một Thread kế thừa từ Thread class.

Hướng dẫn:

```
public class MyThread extends Thread
{
    public static void main(String[] args)
    {
        MyThread th = new MyThread();
        th.start();
        System.out.println("This is the main thread");
    }

    public void run()
    {
        while (true)
        {
            try
            {
                System.out.println("This is the child Thread");
                sleep(1000);
            } catch (InterruptedException ex)
            {

```

```
        ex.printStackTrace();
    }
}
}
```

Bài 2. Tao một Thread implements từ Runnable interface.

Hướng dẫn:

```
public class MyThread2 implements Runnable
{
    public static void main(String[] args)
    {
        MyThread2 myRunnable = new MyThread2();
        Thread thread = new Thread(myRunnable);
        thread.start();
        System.out.println("This is the main thread");
    }
    public void run()
    {
        while (true)
        {
            try
            {
                System.out.println("This is the child Thread");
                sleep(1000);
            } catch (InterruptedException ex)
            {
                ex.printStackTrace();
            }
        }
    }
}
```

Bài 3. Tao 2 Thread: Thread 1 – Cứ mỗi 2 giây sinh ra một số ngẫu nhiên từ 1 đến 20.

Thread 2: Lấy số ngẫu nhiên do Thread 1 sinh ra tính bình phương và hiển thị.

Hướng dẫn:

Bước 1: Tao interface Listener

```
public interface Listener {  
    public void addNumber(int number);  
}
```

Bước 2: Tạo 1 **class** Generator tạo các một số ngẫu nhiên 1->20, 1 **class** square để bình phương số mà generator vừa tạo ra. **Class** generator sẽ thông báo cho square khi nó tạo được 1 số ngẫu nhiên bằng callback.

Class Generator

```
public class Generator implements Runnable{
    Thread t;
    Listener listener;
    public Generator(Listener listener){
        this.listener = listener;
    }
    @Override
    public void run(){
        try {
            while(true) {
                int number = new Random().nextInt(20) + 1;
                System.out.println("Generator: " + number);
                // thông báo số vừa tạo ra cho listener
                listener.addNumber(number);
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    public void start(){
        if(t == null) {
            t = new Thread(this, "Generator");
            t.start();
        }
    }
}
```

Class Square

```
public class Square implements Runnable, Listener{
    Thread t;
    int number;
    boolean flag = false;
```

```

// nhận số được tạo ra từ Generator

@Override
public void addNumber(int number){
    this.number = number;
// flag = true khi nhận được 1 số mới tạo ra
    this.flag = true;
}

@Override
public void run(){
    try {
        while(true) {
            if(this.flag) {
                System.out.println("Square: " + number * number);
                this.flag = false;
            }
            Thread.sleep(1000);
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public void start(){
    if(t == null) {
        t = new Thread(this, "Square");
        t.start();
    }
}
}

```

Phương thức main

```

public class Demo {
    public static void main(String[] args) {
        Square square = new Square();
        square.start();
        Generator generator = new Generator(square);
    }
}

```

```

        generator.start();
    }
}

```

Bài 4.

- Tạo lớp ThreadUtils cài đặt giao diện Runnable, thực hiện chức năng tính tổng các số nguyên từ fromIndex đến toIndex, lưu trữ tổng kết quả tính được.
- Tạo lớp ThreadTest, khai báo một thê hiện của mảng các phần tử số nguyên, kích thước mảng là 10,000.
- Viết phương thức để gán các giá trị trị cho mảng số nguyên vừa khai báo ở trên, giá trị là các số nguyên sinh ngẫu nhiên từ 0 →? 10,000.
- Tạo hai tiến trình thực hiện chức năng tính tổng các giá trị của mảng vừa tạo. Tiến trình 1 thực hiện chức năng tính tổng các số nguyên từ index 0 →? 5000. Tiến trình hai thực hiện chức năng tính tổng các số nguyên từ 5001 →? 9999. Cộng tổng kết quả từ hai tiến trình rồi hiển thị ra màn hình.

Hướng dẫn:

Bước 1: Tạo ThreadUtils

```

public class ThreadUtils implements Runnable {
    static int arr[] = new int[1000];
    private int fromIndex;
    private int toIndex;
    private long total;
    public ThreadUtils(int fromIndex,int toIndex, int[] aaa)
    {
        this.fromIndex=fromIndex;
        this.toIndex=toIndex;
        this.arr = aaa;
        this.total = 0L;
    }
    public void run()
    {
        for(int i = fromIndex; i<=toIndex;i++)
            total += arr[i];
    }
    public long getSum() {

```

```
        return total;
    }
}
```

Bước 2: Tạo class ThreadTest

```
import java.util.Random;

public class ThreadTest {
    public static int[] mang = new int[1000];
    public static void gan()
    {
        for(int i =0; i<mang.length;i++)
        {
            int Random= new Random().nextInt(1000);
            mang[i]=Random;
        }
    }
    public static void main(String[] args)
    {
        gan();
        ThreadUtils tu1 = new ThreadUtils(0,500,mang);
        Thread t1 = new Thread(tu1);
        t1.start();
        try {
            t1.join();
        } catch(InterruptedException ex)
        {
        }
        long tong1 = tu1.getSum();
        ThreadUtils tu2 = new ThreadUtils(501, 999,mang);
        Thread t2 = new Thread(tu2);
        t2.start();
        try {
            t2.join();
        } catch(InterruptedException ex)
        {
```

```

        }
        long tong2 = tu2.getSum();
        System.out.println("Tong mang 1: " + tong1);
        System.out.println("Tong mang 2: " + tong2);
        System.out.println("Tong hai gia tri: " + tong1 + tong2);
    }
}

```

Bài 5. Sử dụng MultiThread và synchronized.Tạo hai luồng:

Luồng 1: Hiển thị các số nhỏ hơn 1000

Luồng 2: Dựa trên số hiển thị trong luồng 1, hiển thị thông tin tương ứng là “nguyên tố” hay “không là số nguyên tố”. Viết Phương thức main () minh họa hai luồng này.

Hướng dẫn:

Bước 1: Tạo class Thread1

```

public class Thread1 extends Thread{
    Number number;
    public Thread1(Number number) {
        this.number = number;
    }
    @Override
    public void run() {
        for (int i= 0; i < 1000; i++) {
            number.inSo();
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
            }
        }
    }
}

```

Bước 2: Tạo class Thread2

```

public class Thread2 extends Thread{
    Number number;
    public Thread2(Number number) {

```

```

    this.number = number;
}

@Override
public void run() {
    for (int i= 0; i < 1000; i++) {
        number.kiemtra();
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
        }
    }
}

```

Bước 3: Tạo class Number

```

public class Number {
    private int num = 1;
    private boolean flag = false;
    public synchronized void inSo(){
        while (flag) {
            try {
                wait();
            } catch (Exception e) {
            }
        }
        System.out.println("So: " + num);
        flag = true;
        notifyAll();
    }
    public synchronized void kiemtra(){
        while (!flag) {
            try {
                wait();
            } catch (Exception e) {
            }
        }
    }
}

```

```

        }

        if (laSoNgTo())
            System.out.println(num+ " la so nguyen to.");
        else
            System.out.println(num+ " khong phai la so nguyen to.");
        num++;
        flag = false;
        notifyAll();
    }

    private boolean laSoNgTo() {
        if (num == 1) return false;
        for (int i= 2; i < num; i++) {
            if (num % i == 0)
                return false;
        }
        return true;
    }
}

```

Bước 4: Tạo class Test

```

public class Test{
    public static void main(String[] args) {
        Number num = new Number();
        Thread1 t1 = new Thread1(num);
        Thread2 t2 = new Thread2(num);
        t1.start();
        t2.start();
    }
}

```

Bài 6.

Tạo ứng dụng java sử dụng Multithread và đồng bộ luồng sử dụng (synchronized)

Luồng 1: Hiển thị ngày, giờ hệ thống.

Luồng 2: Dựa trên số giây hiển thị trong luồng 1, hiển thị thông tin tương ứng là “giây là chẵn” hay “giây là lẻ”.

Viết phương thức main () minh họa hai luồng này. Luồng 1: now 12:20:22

Luồng 2: Giây là lẻ

Hướng dẫn:

Bước 1: Tạo class myObject

```
public class myObject {  
    boolean flag = false;  
    DateFormat df;  
    int second;  
    public synchronized void thoiGian() {  
        while (flag) {  
            try {  
                wait();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
        df = new SimpleDateFormat("HH:mm:ss");  
        String format = df.format(new Date());  
        System.out.println(format);  
        second = Calendar.getInstance().get(Calendar.SECOND);  
        flag = true;  
        notifyAll();  
    }  
    public synchronized void kiemtra() {  
        while (!flag) {  
            try {  
                wait();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
        if(second % 2 ==0 )  
            System.out.println("The "+ second +" is Even");  
        else  
            System.out.println("The "+ second +" is Odd");  
    }  
}
```

```
    flag = false;
    notifyAll();
}
```

Bước 2: Tạo Thread 1

```
public class Thread1 extends Thread{
    myObject obj;
    public Thread1(myObject obj) {
        this.obj = obj;
    }
    @Override
    public void run() {
        //myObject obj = new myObject();
        try {
            while (true) {
                obj.thoiGian();
                Thread.sleep(1000);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Bước 3: Tạo Thread2

```
public class Thread2 extends Thread{
    myObject obj;
    public Thread2(myObject obj) {
        this.obj = obj;
    }
    @Override
    public void run() {
        try {
            while (true) {

```

```

        obj.kiemtra();
        Thread.sleep(1000);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}
}

```

Bước 4: Tạo class Main

```

public class Main {
    public static void main(String[] args) {
        myObject obj = new myObject();
        Thread1 t1 = new Thread1(obj);
        Thread2 t2 = new Thread2(obj);
        t1.start();
        t2.start();
    }
}

```

Bài 7.

- Tạo class **Bank** gồm các trường và phương thức sau:

- Tạo 1 mảng: **double[] accounts**
- Tạo 1 constructors với 2 tham số: **int n**, và **double initBalance**, **n** là số phần tử mảng **accounts**, **initBalance** là giá trị ban đầu gán cho từng phần tử trong mảng **accounts**.
- Tạo phương thức tên **size()**, trả về kích thước của mảng
- Tạo phương thức tên **getTotalBalance()**, trả về tổng giá trị của mảng. Phương thức này được đồng bộ hoá.
- Tạo phương thức **transfer (int from, int to, double amount)** bắt lỗi **InterruptedException** thực hiện công việc sau:
 - Khi giá trị của phần tử **from** trong mảng nhỏ hơn **amount**, thread rơi vào trạng thái **wait**
 - In ra tên của thread hiện thời
 - Chuyển tiền với giá trị là **amount** từ **from** sang **to**. Mỗi lần chuyển tiền đều phải đưa ra thông báo. Ví dụ: Chuyển 200 từ account 12 sang account 24.
 - In ra tổng giá trị của tất cả các account, sử dụng **getTotalBalance()**.
 - Sau khi thực hiện xong phải đánh thức các thread khác.
 - Phương thức phải được đồng bộ hoá

- Tạo class **TransferMoney**, thực thi giao diện **Runnable**, gồm các trường và phương thức sau:

- Tạo 4 trường như sau: **Bank bank, int fromAcc, double maxAmount, int delay**.
- Tạo phương thức khởi tạo cho **bank, fromAcc** và **maxAmount**. Trường **delay** luôn có giá trị là 1000.
- Override phương thức **run()** để thực hiện công việc sau:
- Chuyển tiền từ account **fromAcc** sang account **toAcc** của **bank**, biết rằng **toAcc** là 1 account lấy ngẫu nhiên trong số các account còn lại của **bank**. Lượng tiền chuyển **amount** cũng là ngẫu nhiên, trong khoảng từ 0 tới **maxAmount**.
- Cho thread nghỉ 1 khoảng thời gian từ 0 đến 1 giây.

- Tạo class **SynchBank** để thực hiện 2 class trên như sau:

- Khởi tạo 1 **bank** có 100 account với giá trị ban đầu là 1000 usd.
- Chạy các thread **TransferMoney** đối với 100 account của **bank**

Hướng dẫn:

Bước 1: Tạo class Bank

```
public class Bank
{
    double[] accounts;
    public Bank(int n, double initBalance)
    {
        accounts = new double[n];
        for (int i = 0; i < accounts.length; i++)
        {
            accounts[i] = initBalance;
        }
    }
    public int size()
    {
        return accounts.length;
    }
    public synchronized double getTotalBalance()
    {
        double total = 0;
        for (int i = 0; i < accounts.length; i++)
        {
            total+=accounts[i];
        }
        return total;
    }
    public synchronized void transfer(int from, int to, double amount)
```

```

    {
        try
        {
            while(accounts[from] < amount)
            {
                System.out.println(Thread.currentThread().getName()+"đợi đủ tiền");
                wait();
            }
            accounts[from] -= amount;
            accounts[to] += amount;
            System.out.println("Chuyển " + amount + " từ account " + from + " sang
account " + to);
            System.out.println("Tổng tiền của các account: " +
getTotalBalance());
            notifyAll();
        }
        catch (InterruptedException ex)
        {
            InterruptedException("Giao dịch bị gián đoạn");
        }
    }
}

```

Bước 2: Tạo class TransferMoney

```

public class TransferMoney implements Runnable
{
    Bank bank;
    int fromAcc;
    double maxAmount;
    int delay = 1000;
    public TransferMoney(Bank bank, int fromAcc, double maxAmount) {
        this.bank = bank;
        this.fromAcc = fromAcc;
        this.maxAmount = maxAmount;

    }
    @Override
    public void run()
    {

```

```

Random rd = new Random();
int toAcc = 0;
double amount = 0;
try
{
    while (true)
    {
        do
        {
            toAcc = rd.nextInt(bank.size());
        }while (toAcc==fromAcc);
        amount = rd.nextInt((int)maxAmount);
        bank.transfer(fromAcc, toAcc, amount);
        Thread.sleep(rd.nextInt(delay));
    }
}
catch(InterruptedException ex)
{
    InterruptedException("Giao dịch chuyển tiền từ account " +
fromAcc + " sang account " + toAcc + " bị gián đoạn");
}
}
}

```

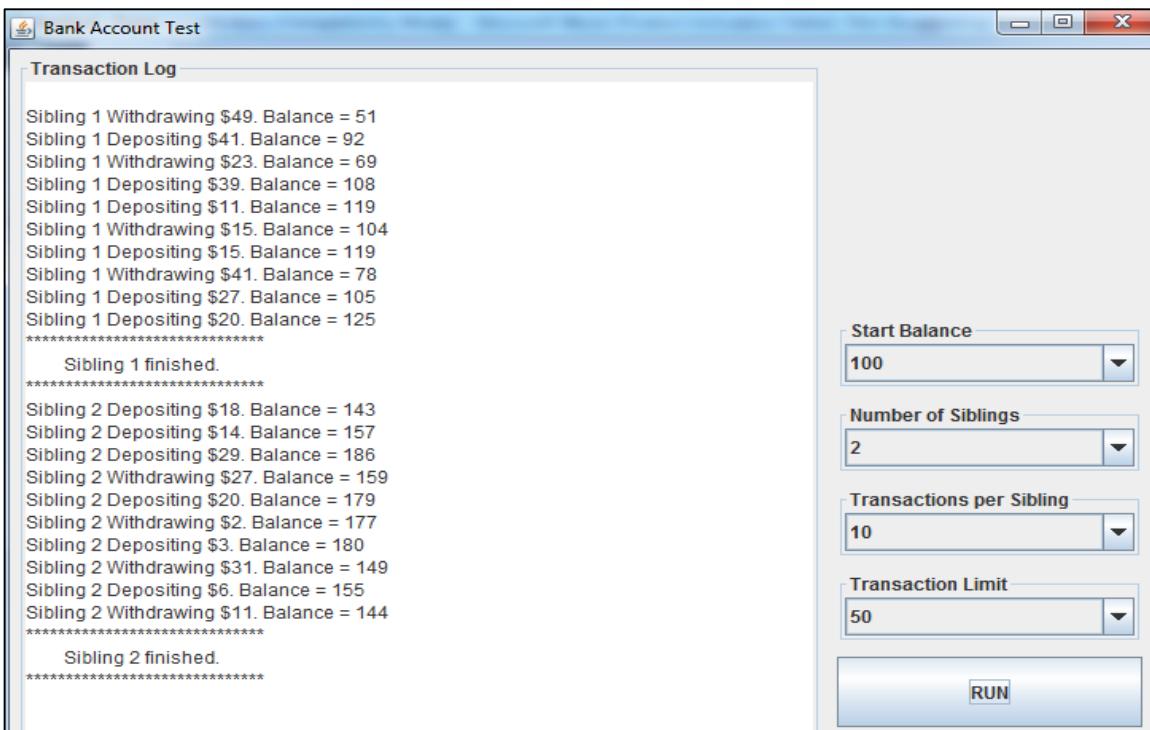
Bước 3. Tạo class SynchBank

```

public class SynchBank
{
    public static void main(String[] args) throws InterruptedException {
        Bank bank = new Bank(100, 1000);
        int size = bank.size();
        for (int i = 0; i < size; i++)
        {
            TransferMoney transferMoney = new TransferMoney(bank, i, 1000);
            Thread thread = new Thread(transferMoney);
            thread.start();
        }
    }
}

```

Bài 9. Tổng hợp Viết chương trình minh họa việc gửi và rút tiền tại ngân hàng (sử dụng mô hình MVC). Giao diện chương trình như sau:



Hình 3. Màn hình giao dịch ngân hàng

Yêu cầu: Một số anh chị em được cha mẹ cấp quyền truy cập vào tài khoản ngân hàng. Đối với mỗi lần chạy mô phỏng gồm các thông số:

- Số dư của tài khoản ngân hàng- số tiền ban đầu (start balance)
- Số lượng người dùng có quyền truy cập tài - số anh chị em (Number of Siblings)
- Đối với mỗi anh chị em, có số lần giao dịch giới hạn. (Transactions per Sibling).
- Giới hạn giao dịch: Transaction Limit.
- Mỗi lần giao dịch bao gồm hành động gửi và rút với số tiền xác định. Tất cả đều được tạo ngẫu nhiên.
- Mỗi anh chị em thực hiện danh sách các giao dịch của mình, với ràng buộc là số dư không thể nhỏ hơn 0

Kết quả của mỗi giao dịch được hiển thị trong nhật ký giao dịch.

Hướng dẫn:

Thiết kế kiến trúc phần mềm Model-View-Controller (MVC) trong đó:

- Model: Tài khoản ngân hàng
- View: Nhật ký giao dịch
- Control: code cho các chức năng trong combobox và button run

Bước 1: Tạo class BankAccount, lớp này kế thừa từ class Observable.

Lớp Observable được sử dụng để tạo các lớp con mà các phần khác của chương trình có thể quan sát. Khi một đối tượng của lớp con đó xảy ra một sự thay đổi, các lớp quan sát sẽ nhận được thông báo.

```
import java.util.Observable;
public class BankAccount extends Observable {
```

```

private int balance; //số dư ban đầu
//Phương thức thiết lập số dư

public void setBalance(int balance) {
    this.balance = balance;
    setChanged();
    notifyObServers(null);
}

//Phương thức nạp tiền vào tài khoản

public void deposit(int amount, BankAccountUser user) {
    log("\n" +user.getName()+" Depositing $" +amount);
    int newBalance = balance + amount;
    balance = balance + amount;
    log(". Balance = " + balance);
    checkFinished(user);
    assert(balance == newBalance);
}

//Phương thức rút tiền

public void withdraw(int amount, BankAccountUser user) {
    log("\n" +user.getName() + " Withdrawing $" + amount);
    if (amount > balance) {
        throw new RuntimeException("Amount (" +amount+ ") must not be
greater than " +balance+ ".");
    }
    int newBalance = balance - amount;
    balance = balance - amount;
    log(". Balance = " + balance);
    checkFinished(user);
    assert(balance == newBalance);
}

//Phương thức trả về số dư

public int getBalance() {
    return balance;
}
}

```

```

//Phương thức kiểm tra kết thúc
private void checkFinished(BankAccountUser user) {
    if ( user.isOneMore() ) {
        log("\n****\n " + user.getName() + " finished.\n*****");
        user.setFinished(true);
    }
}

//Phương thức theo dõi tín hiệu từ tài khoản bằng cách gửi đến tất cả các đối tượng
//đang quan sát.
private void log(String message) {
    setChanged();
    notifyObServers(message);
}

```

Bước 2: Tạo class BankAccountUser gồm các thuộc tính: name, account, list giao dịch

```

public class BankAccountUser {

    private String name;
    private BankAccount account;
    private List<Integer> transactions;
    private boolean oneMore;
    private boolean finished;

    public BankAccountUser(String name, BankAccount account,
List<Integer> transactions) {
        this.name = name;
        this.account = account;
        this.transactions = transactions;
        oneMore = false;
        finished = false;
    }

    public String getName() {
        return name;
    }

    public BankAccount getAccount() {
        return account;
    }
}

```

```

    }

    public boolean isFinished() {
        return finished;
    }

    public void setFinished(boolean finished) {
        this.finished = finished;
    }

    public boolean isOneMore() {
        return oneMore;
    }

    public void setOneMore(boolean oneMore) {
        this.oneMore = oneMore;
    }

    public void run() {
        Iterator<Integer> iter = transactions.iterator();
        while (iter.hasNext()) {
            int amount = iter.next();
            if ( !iter.hasNext() ) {
                oneMore = true;
            }
            if (amount > 0) {
                account.deposit(amount, this);
            }
            else if (amount < 0) {
                account.withdraw(Math.abs(amount), this);
            } // amount == 0 ignored
        }
    }
}

```

Bước 3: Tạo class BankAccountControl

```

public class BankAccountControl extends JComponent {
    private static Random generator = new Random();
    private BankAccount account;
    private BankAccountUser[] users;
}

```

```

private JComboBox balanceChoices;
private JComboBox usersChoices;
private JComboBox limitChoices;
private JComboBox transactionsChoices;
private JButton runButton;
private int startBalance;
private int numUsers;
private int amountLimit;
private int numTransactions;
public BankAccountControl(BankAccount account) {
    this.account = account;
    balanceChoices = makeComboBox("Start Balance", new int[] {100, 500,
1000, 10000});
    usersChoices = makeComboBox("Number of Siblings", new int[] {1, 2,
3, 4, 5});
    limitChoices = makeComboBox("Transaction Limit", new int[] {50,
100, 200, 500});
    transactionsChoices = makeComboBox("Transactions per Sibling", new
int[] {10, 25, 50, 100});
    runButton = new JButton("RUN");
    runButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            setup();
            run();
        }
    });
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(5, 1, 0, 10));
    panel.add(balanceChoices);
    panel.add(usersChoices);
    panel.add(transactionsChoices);
    panel.add(limitChoices);
    panel.add(runButton);
}

```

```

setLayout(new FlowLayout());
add(panel);
}

private JComboBox makeComboBox(String title, int[] options) {
    JComboBox comboBox = new JComboBox();
    comboBox.setPreferredSize(new Dimension(title.length()*8, 50));
    comboBox.setBorder(new TitledBorder(title));
    for (int i = 0; i < options.length; i++) {
        comboBox.addItem(options[i]);
    }
    return comboBox;
}

private void setup() {
    startBalance = (Integer) balanceChoices.getSelectedItem();
    numUsers = (Integer) usersChoices.getSelectedItem();
    amountLimit = (Integer) limitChoices.getSelectedItem();
    numTransactions = (Integer) transactionsChoices.getSelectedItem();
    account.setBalance(startBalance);
    users = new BankAccountUser[numUsers];
    for (int i = 0; i < users.length; i++) {
        users[i] = new BankAccountUser("Sibling " + (i+1), account,
makeTransactions());
    }
}

private void run() {
try {
    for (int i = 0; i < users.length; i++) {
        users[i].run();
    }
}
catch(RuntimeException ex) {
    JOptionPane.showMessageDialog(null, ex.getMessage());
}
}

```

```

private List<Integer> makeTransactions() {
    List<Integer> transactions = new ArrayList<Integer>();
    for (int i = 0; i < numTransactions; i++) {
        int amount = generator.nextInt(amountLimit) + 1;
    transactions.add(generator.nextBoolean() ? amount : -amount);
    }
    return transactions;
}
}

```

Bước 4: Tạo lớp BankAccountComponent

```

public class BankAccountComponent extends JPanel {
    public BankAccountComponent() {
        BankAccount account = new BankAccount();
        LogView view = new LogView("Transaction Log");
        account.addServer(view);
        BankAccountControl control = new BankAccountControl(account);
        setLayout(new FlowLayout());
        add(view);
        add(control);
    }
}

```

Bước 5: Tạo lớp BankAccountFrame

```

public class BankAccountFrame extends JFrame {
    public BankAccountFrame() {
        super("Bank Account Test");
        add(new BankAccountComponent());
        pack();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
    public static void main(String[] args) {
        new BankAccountFrame();
    }
}

```

```
}
```

Bước 6: Tạo class LogView

```
public class LogView extends JScrollPane implements ObServer {  
    public LogView(String title) {  
        super(transactionArea);  
        setBorder(new TitledBorder(title));  
    }  
    public void update(Observable account, Object message) {  
        if ( message == null ) {  
            transactionArea.setText("");  
        }  
        else {  
            transactionArea.append((String)message);  
        }  
    }  
    private static JTextArea transactionArea = new JTextArea(40,45);  
}
```

LAB 6. LẬP TRÌNH CƠ SỞ DỮ LIỆU JDBC [3,5,8]

A. MỤC TIÊU

- Trang bị cho sinh viên kỹ năng lập trình CSDL.
- Ôn tập lại các thao tác xử lý thêm, chèn, sửa xóa
- Kết hợp kiến thức tạo giao diện (JAVA SWING) lập trình ứng dụng Quản lý theo mô hình 3 lớp (MVC).

B. NỘI DUNG

- Thực hiện được việc tải và cài đặt JDBC driver cho project
- Thực hiện kết nối CSDL với các hệ Quản trị CSDL SQL Server, MySQL,...
- Thực hiện được việc kết nối và truy xuất (thêm, chèn, sửa, xóa) CSDL.
- Thực hiện được việc xử lý kết xuất kết quả truy xuất CSDL.

C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB
- Phần mềm SQL Server 2015, NETBEAN 8.0, JDK 1.8

D.KẾT QUẢ SAU KHI HOÀN THÀNH

- Hiểu được phương pháp kết nối CSDL trong Java bằng JDBC.
- Xây dựng ứng dụng kết nối CSDL.

E. HƯỚNG DẪN CHI TIẾT

Các bước chính trong kết nối CSDL MySQL với JDBC:

1. Nạp điều khiển driver:

Class.forName(driverString);

Trong đó **driverString** là chuỗi chỉ định tên của trình điều khiển cần nạp.

Ví dụ: Nạp điều khiển driver của CSDL MySQL

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

2. Kết nối CSDL: gọi phương thức DriverManager.getConnection(url,user,pass) để nhận về đối tượng Connection kết nối với CSDL, url có dạng:

jdbc:subprotocol:subname

Trong đó:

subprotocol: giao thức tương ứng với loại CSDL

subname: tên cầu nối ODBC thông qua đó ta có thể kết nối tới CSDL

Ví dụ :

```
url = "jdbc:mysql://localhost:3306/";  
user = "root";
```

```

    pass = "";
    myConnection = DriverManager.getConnection(url, user, pass);

```

3. Tạo đối tượng Statement: gọi phương thức createStatement() của đối tượng Connection. Đối tượng Statement dùng để thực hiện các câu truy vấn.

Ví dụ:

```

Statement stmt;
stmt = myConnection.createStatement();

```

4. Tạo truy vấn dữ liệu: Có 3 loại truy vấn:

executeQuery(strSQL) : dùng cho câu lệnh SELECT, kết quả trả về kiểu ResultSet

Ví dụ : ResultSet rs;

```
rs = stmt.executeQuery("Select * From TheLoai");
```

executeUpdate(strSQL) : dùng cập nhật dữ liệu như INSERT, UPDATE, DELETE

execute(strSQL) : dùng trong trường hợp không rõ kiểu truy vấn.

5. Đóng kết nối: gọi phương thức close() tương ứng để giải phóng vùng nhớ.

```

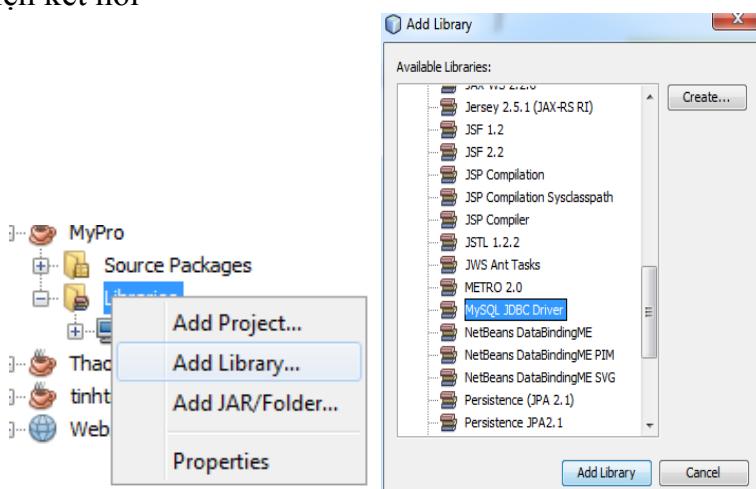
stmt.close();
rs.close();

```

Bài 1. Thực hiện kết nối CSDL MySQL

Mở MySQL, tạo bảng “taikhoan” gồm hai trường “ID” và “tên” trong “csdl1”. Nhập 5 bản ghi cho csdl. Thực hiện kết nối CSDL, in thông tin tài khoản ra màn hình

Bước 1: add thư viện kết nối



Hình 4. Thêm thư viện MySQL JDBC Driver

Bước 2:

Tạo tạo bảng “taikhoan” có hai trường ID kiểu int và tên kiểu char trong csdl1 bằng MySQL. Tạo file java KetNoiCSDL thực hiện các bước kết nối

```

public class KetNoiCSDL {
    Connection cn= null;
}

```

```

public KetNoiCSDL() throws SQLException {
    String url="jdbc:mysql://localhost:3306/cSDL1";
    this.cn=DriverManager.getConnection(url,"root","");
}

```

Bước 3: Tạo phương thức LayDL trả về kiểu ResultSet

```

public ResultSet LayDL (String tenbang) throws SQLException {
    ResultSet kq=null;
    Statement st=this.cn.createStatement();
    String sql= "select * from taikhoan";
    kq=st.executeQuery(sql);
    return kq;
}

```

Bước 4. Tạo phương thức main và sử dụng

```

public static void main(String[] args) throws SQLException
{
    KetNoiCSDL a= new KetNoiCSDL();
    ResultSet rs=a.LayDL("taikhoan");
    while(rs.next())
    {
        System.out.println(rs.getString(1));
    }
}

```

Bài 2. Thực hành thêm thư viện kết nối Sqlserver

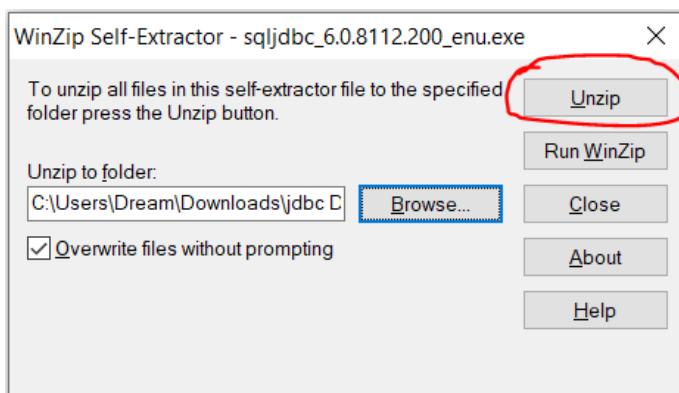
Bước 1: Tải driver từ microsoft:

<https://www.microsoft.com/en-us/download/details.aspx?id=54671>

Bước 2: Unzip file vừa tải vào folder bất kỳ

Name	Date modified	Type	Size
sqljdbc_6.0.8112.200_enus.exe	17/02/2018 5:06 CH	Application	2.308 KB

Giao diện unzip



Hình 5. Download JDBC Driver cho SQL

Folder thu được sau khi unzip

Name	Date modified	Type
sqljdbc_6.0	23/02/2019 10:49 ...	File folder

Bước 3: Mở folder vừa unzip được theo đường dẫn\sqljdbc_6.0\enu\auth\x64

Copy file file sqljdbc_auth.dll vào thư mục jdk của java theo đường dẫn:

Win 64bit: C:\Program Files\Java\jdk1.8.0_202\bin

Win 32bit: C:\Program Files (x86)\Java\jdk1.8.0_202\bin

(Phần gạch chân có thể khác tùy phiên bản jdk)

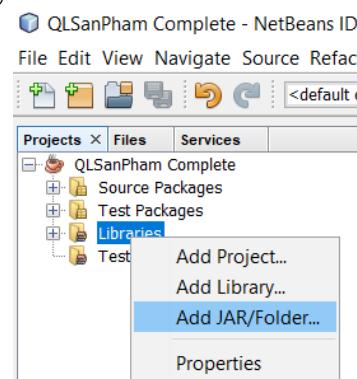
Bước 4: Mở folder unzip được ở trên theo đường dẫn\sqljdbc_6.0\enu\jre8

(Lưu ý: jre8 hoặc jre7 tùy phiên bản jre được cài đặt trên máy)

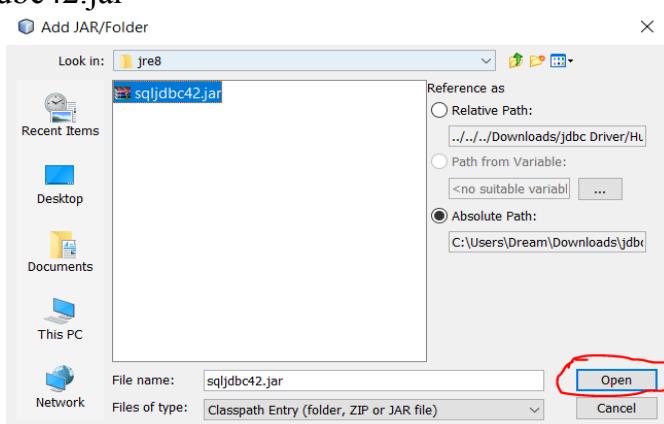
Trong đó có file: sqljdbc42.jar

Add file sqljdbc42.jar vào Library của project Netbean :

- Click chuột phải vào Library -> Add JAR/Folder

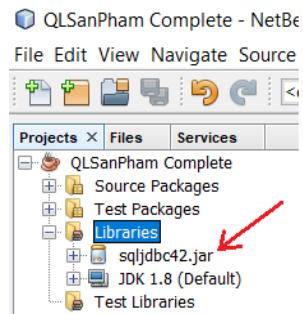


- Tìm đến file sqljdbc42.jar



Hình 6. Thêm sqljdbc42.jar

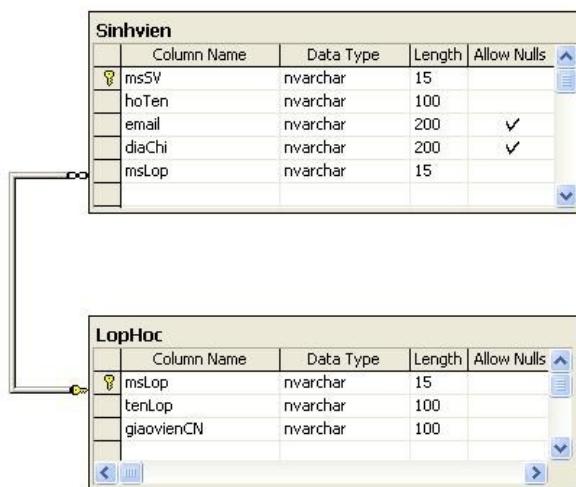
- Thêm thành công:



Để kết nối với SQL Server dùng chuỗi kết nối

```
String connString =
"jdbc:sqlserver://localhost:1433;integratedSecurity=true;
databaseName=" + dbName;
```

Bài 3. Cho cơ sở dữ liệu QLSV với cấu trúc bảng như sau:



- Kết nối với CSDL và hiển thị thông tin lên JFrame như hình sau:



- Khi nhấn nút Thêm hiện lên màn hình có nút Hủy và Lưu:



- ✓ Nhấn Hủy hoặc lưu, sau khi thực hiện sẽ khôi phục giao diện trở lại như ban đầu.
- ✓ Khi nhấn nút Sửa



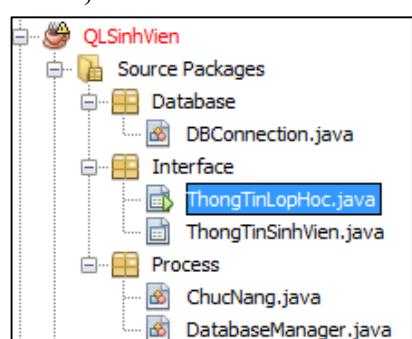
- ✓ Khi nhấn nút “Xem danh sách sinh viên lớp hiện tại”, 1 cửa sổ như sau xuất hiện hiển thị các sinh viên thuộc lớp hiện tại đang được chọn



Hình 6. Một số màn hình quản lý thông tin Sinh viên

Hướng dẫn:

Bước 1: Tạo project QLSinhVien, thiết kế theo mô hình MVC



Bước 2:

Tạo class DBConnection

```
public class DBConnection {
    String dbName = "QLSinhVien"; //tên cơ sở dữ liệu cần kết nối
    String connString =
    "jdbc:sqlserver://localhost:1433;integratedSecurity=true; databaseName="
```

```

+ dbName;

public Connection GetConnection()
{
    Connection conn;
    try {
        conn = DriverManager.getConnection(connString);
        return conn;
    } catch (SQLException ex) {
        logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE,null,ex);
        return null;
    }
}

public ResultSet GetData(String query)
{
    Connection conn = GetConnection();
    if(conn==null) //nếu không thể mở kết nối
    {
        CloseConnection(conn); //Đóng kết nối
        return null;
    }
    Statement stm;
    try
    {
        stm = conn.createStatement();
        ResultSet rs = stm.executeQuery(query);
        return rs;
    } catch (SQLException ex) {
        logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE,null,ex);
        CloseConnection(conn); //đóng kết nối
        return null;
    }
}

public boolean UpdateData(String query)
{

```

```

        Connection conn = GetConnection();

        if(conn==null)
            return false;

        Statement stm;

        try
        {
            stm = conn.createStatement();
            stm.executeUpdate(query);
            return true;
        } catch (SQLException ex) {
Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE,null,ex);
            return false;
        }
    }

    public void CloseConnection(Connection conn)
    {
        try
        {
            conn.close();
        } catch (SQLException ex) {
Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE,null,ex);
        }
    }
}

```

Trong package process tạo hai **class** ChucNang và DatabaseManager

```

public class ChucNang {

    public static final int NONE = -1; //chưa chọn
    public static final int ADD = 0;
    public static final int UPDATE = 1;
}

public class DatabaseManager {// thực hiện thêm, sửa, xóa trên database
    public static boolean ThemLopHoc(String msLop, String tenLop, String
gvcn)
    {

```

```

    DBConnection dbConn = new DBConnection();

    String qr = "Insert Into LopHoc Values('"+msLop+"',
N'"+tenLop+"', N'"+gvcn+"')";

    return dbConn.UpdateData(qr);

}

public static boolean SuaLopHoc(String msLop, String tenLop, String
gvcn)
{
    DBConnection dbConn = new DBConnection();

    String qr = "Update LopHoc Set tenlop = N'"+tenLop+"', giaoVienCN
= N'"+gvcn+"' Where mslop = '"+msLop+"'";

    return dbConn.UpdateData(qr);
}

public static boolean XoaLopHoc(String msLop)
{
    String qr = "Delete From LopHoc Where mslop = '"+msLop+"'";

    DBConnection dbConn = new DBConnection();

    return dbConn.UpdateData(qr);
}

public static boolean LopHocToTable(JTable jTable)
{
    try
    {
        DefaultTableModel dfTableModel = (DefaultTableModel)jTable.getModel();
        dfTableModel.setRowCount(0);
        DBConnection db = new DBConnection();
        ResultSet rs = db.GetData("Select * From LopHoc");
        String[] row = new String[3];
        while(rs.next())
        {
            row[0] = rs.getString(1);
            row[1] = rs.getString(2);
            row[2] = rs.getString(3);
            dfTableModel.addRow(row);
        }
    }
}

```

```

        }

        return true;
    }catch (Exception ex)
    {
        Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE,null,ex);
        return false;
    }
}

public static boolean ThemSinhVien(String mssv, String hoTen, String
email, String diaChi, String msLop) {
    String qr = "Insert Into SinhVien Values('"+mssv+"',
N'"+hoTen+"', N'"+email+"', N'"+diaChi+"', '"+msLop+"')";
    DBConnection dbConn = new DBConnection();
    return dbConn.UpdateData(qr)==true;
}

public static boolean SuaSinhVien(String mssv, String hoTen, String
email, String diaChi, String msLop) {
    String qr = "Update SinhVien Set hoten = N'"+hoTen+"', email =
N'"+email+"', diachi = N'"+diaChi+"', mslop = '"+msLop+"' Where masv =
'"+mssv+"'";
    DBConnection dbConn = new DBConnection();
    return dbConn.UpdateData(qr);
}

public static boolean XoaSinhVien(String masv) {
    String qr = "Delete From SinhVien Where masv = '"+masv+"' ";
    DBConnection dbConn = new DBConnection();
    return dbConn.UpdateData(qr);
}

```

//Lấy danh sách sinh viên theo lớp học

```

public static boolean SinhVienToTable_ByLopHoc(JTable jTable, String
msLop) {
    try {
        DefaultTableModel=(DefaultTableModel)jTable.getModel();
        dfTableModel.setRowCount(0);
    }
}

```

```

        DBConnection db = new DBConnection();

    ResultSet rs = db.GetData("Select * From SinhVien Where mslop =
'"+msLop+"''");

        String[] row = new String[4];

        while(rs.next()){

            row[0] = rs.getString(1);

            row[1] = rs.getString(2);

            row[2] = rs.getString(3);

            row[3] = rs.getString(4);

            dfTableModel.addRow(row);

        }

        return true;

    }catch (Exception ex){

        Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE, null,
ex);

        return false;

    }

}

public static int Count(String tableName, String columnName, String id)

{

    String qr = "";

    if(id.length()==0 || columnName.length()==0)

        qr = "Select COUNT(*) from "+tableName;

    else

        qr = "Select COUNT(*) from "+tableName+" Where "+columnName+" =
'"+id+"'";

    DBConnection dbConn = new DBConnection();

    ResultSet rs = dbConn.GetData(qr);

    try

    {

        if(rs.next())

        {

            int count = Integer.parseInt(rs.getString(1));

            return count;

        }

    }

}

```

```

        }
    }catch (Exception ex) {
        Logger.getLogger(DBConnection.class.getName()).log(Level.SEVERE, null,
ex);
    return -1;
}
return -1;
}
}

```

Trong package **Interface** tạo hai Jframe ThongTinLopHoc



```

public class ThongTinLopHoc extends javax.swing.JFrame {

    DefaultTableModel dfTableModel;
    int chucNangDaChon = ChucNang.NONE;
    public ThongTinLopHoc() {
        initComponents();
        dfTableModel = (DefaultTableModel)tblDSLop.getModel();
    }
    //Phương thức xử lý sự kiện khi lựa chọn ở bảng thay đổi
    void TblDSLop_SelectionChanged() {
        int row = tblDSLop.getSelectedRow();
        if(row>=0) {
            String msLop = (String)dfTableModel.getValueAt(row, 0);
            String tenLop = (String)dfTableModel.getValueAt(row, 1);
            String gvcn = (String)dfTableModel.getValueAt(row, 2);
        }
    }
}

```

```

        txtMSLop.setText(msLop.trim());
        txtTenLop.setText(tenLop.trim());
        txtGVCN.setText(gvcn.trim());
    }else {
        txtMSLop.setText("");
        txtTenLop.setText("");
        txtGVCN.setText("");
    }
    ReloadLblIndexTblDSLopHoc();
}

//Lấy dữ liệu cho bảng lớp học
void ReloadTaleLopHoc() {
    if(DatabaseManager.LopHocToTable(tblDSLop)==false)
        JOptionPane.showMessageDialog(null, "Lấy dữ liệu lớp học
có lỗi", "Có lỗi xảy ra", JOptionPane.ERROR_MESSAGE);
    ReloadLblIndexTblDSLopHoc();
}

//Lấy lại dữ liệu label hiện chỉ số hàng hiện tại
void ReloadLblIndexTblDSLopHoc() {
    int rowSelected = tblDSLop.getSelectedRow();
    int totalRow = tblDSLop.getRowCount();
    lblIndexTblLopHoc.setText((rowSelected+1) + "/" + totalRow);
}

void SwitchMode(int chucNang) {
    chucNangDaChon = chucNang;
    switch (chucNang) {
        case ChucNang.ADD:{
            boolean trangThai = true;
            txtMSLop.setEnabled(trangThai);
            txtTenLop.setEnabled(trangThai);
            txtGVCN.setEnabled(trangThai);
            txtMSLop.requestFocus();
            btnSave.setEnabled(trangThai);
            btnUpdate.setEnabled(!trangThai);
        }
    }
}

```

```

        btnDelete.setEnabled(!trangThai);

        txtMSLop.setText("");
        txtTenLop.setText("");
        txtGVCN.setText("");
        btnAdd.setText("Hủy");

        break;
    }

    case ChucNang.UPDATE: {
        boolean trangThai = true;

        txtTenLop.setEnabled(trangThai);
        txtGVCN.setEnabled(trangThai);
        txtTenLop.requestFocus();
        btnSave.setEnabled(trangThai);
        btnAdd.setEnabled(!trangThai);
        btnDelete.setEnabled(!trangThai);
        btnUpdate.setText("Hủy");

        break;
    }

    case ChucNang.NONE: {
        boolean trangThai = false;

        txtMSLop.setEnabled(trangThai);
        txtTenLop.setEnabled(trangThai);
        txtGVCN.setEnabled(trangThai);
        btnSave.setEnabled(trangThai);
        btnAdd.setEnabled(true);
        btnUpdate.setEnabled(true);
        btnDelete.setEnabled(true);
        btnAdd.setText("Thêm");
        btnUpdate.setText("Sửa");
    }
}

boolean CheckInput() {
    String msLop = txtMSLop.getText().trim();
}

```

```

String tenLop = txtTenLop.getText().trim();
String gvcn = txtGVCN.getText().trim();
if(msLop.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập mã số lớp", "Chưa
nhập mã số lớp", JOptionPane.WARNING_MESSAGE);
    txtMSLop.requestFocus();
    return false;
}
if(tenLop.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập tên lớp",
"Chưa nhập tên lớp", JOptionPane.WARNING_MESSAGE);
    txtTenLop.requestFocus();
    return false;
}
if(gvcn.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập giáo
viên chủ nhiệm", "Chưa nhập giáo viên chủ nhiệm",
JOptionPane.WARNING_MESSAGE);
    txtGVCN.requestFocus();
    return false;
}
return true;
}

void Exit() {
    int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn
thoát không?", "Thoát?", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);
    if(result == JOptionPane.CANCEL_OPTION)
        return;
    this.dispose();
}

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    tblDSLop.getSelectionModel().addListSelectionListener(new
ListSelectionListener()
{

```

```

    @Override
    public void valueChanged(ListSelectionEvent e) {
        TblDSLop_SelectionChanged();
    }
});

ReloadTaleLopHoc();
}

private void btnXemLopActionPerformed(java.awt.event.ActionEvent evt)
{
    String msLop = txtMSLop.getText().trim();
    String tenLop = txtTenLop.getText().trim();
    if(msLop.length()==0) {
        JOptionPane.showMessageDialog(null, "Chưa chọn lớp để xem",
        "Chưa chọn lớp", JOptionPane.WARNING_MESSAGE);
        return;
    }
    ThongTinSinhVien ttsv = new ThongTinSinhVien(msLop, tenLop);
    ttsv.setVisible(true);
}

private void formWindowClosing(java.awt.event.WindowEvent evt) {
    exit();
}

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    if(chucNangDaChon==ChucNang.NONE)
        SwitchMode(ChucNang.ADD);
    else
        SwitchMode(ChucNang.NONE);
}

private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    if(tblDSLop.getSelectedRow() == -1) {
        JOptionPane.showMessageDialog(null, "Chưa chọn lớp để sửa",
        "Chưa chọn lớp", JOptionPane.WARNING_MESSAGE);
        return;
    }
}

```

```

    if(chucNangDaChon==ChucNang.NONE)
        SwitchMode(ChucNang.UPDATE);
    else
        SwitchMode(ChucNang.NONE);
}

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt)
{
    int selectedRow = tblDSLop.getSelectedRow();
    if(selectedRow<0) {
        JOptionPane.showMessageDialog(null, "Bạn chưa chọn lớp
nào để xóa", "Chưa chọn lớp", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn xóa
lớp này không", "Xóa lớp?", JOptionPane.OK_CANCEL_OPTION,JOptionPane.QUESTION_MESSAGE);
    if(result == JOptionPane.CANCEL_OPTION)
        return;
    String msLop = (String)tblDSLop.getValueAt(selectedRow, 0);
    if(DatabaseManager.Count("SinhVien", "mslop", msLop)>0) {
        JOptionPane.showMessageDialog(null, "Đã có sinh viên
trong lớp này!", "Không thể xóa", JOptionPane.WARNING_MESSAGE);
        return;
    }
    if(DatabaseManager.XoaLopHoc(msLop))
    {
        btnAdd.requestFocus();
        SwitchMode(ChucNang.NONE);
        ReloadTaleLopHoc();
        JOptionPane.showMessageDialog(null, "Xóa thành công",
"Thành công", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    else
    {
}

```

```

        JOptionPane.showMessageDialog(null, "Xóa thất bại", "Có lỗi xảy
ra", JOptionPane.ERROR_MESSAGE);

    return;
}

private void btnFirstActionPerformed(java.awt.event.ActionEvent evt) {
    if(tblDSLop.getRowCount()>0)
        tblDSLop.getSelectionModel().setSelectionInterval(0, 0);
}

private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {
    int rowSelected = tblDSLop.getSelectedRow();
    if(rowSelected>0) {
        rowSelected--;
        tblDSLop.getSelectionModel().setSelectionInterval(rowSelected,
rowSelected);
    }
}

private void btnNextActionPerformed(java.awt.event.ActionEvent evt) {
    int rowSelected = tblDSLop.getSelectedRow();
    if(rowSelected<tblDSLop.getRowCount()-1)
    {
        rowSelected++;
        tblDSLop.getSelectionModel().setSelectionInterval(rowSelected,
rowSelected);
    }
}

private void btnLastActionPerformed(java.awt.event.ActionEvent evt)
{
    if(tblDSLop.getRowCount()>0) {
        int lastRowIndex = tblDSLop.getRowCount()-1;
        tblDSLop.getSelectionModel().setSelectionInterval(lastRowIndex,
lastRowIndex);
    }
}

```

```

private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String msLop = txtMSLop.getText().trim();
    String tenLop = txtTenLop.getText().trim();
    String gvcn = txtGVCN.getText().trim();
    if(chucNangDaChon==ChucNang.ADD) {
        if(CheckInput()==false)
            return;
        if(DatabaseManager.Count("LopHoc", "mslop", msLop)>0) {
            txtMSLop.requestFocus();
            JOptionPane.showMessageDialog(null, "Mã lớp bạn nhập
đã tồn tại trong csdl", "Trùng mã", JOptionPane.WARNING_MESSAGE);
            return;
        }
        if(DatabaseManager.ThemLopHoc(msLop, tenLop, gvcn)) {
            btnAdd.requestFocus();
            SwitchMode(ChucNang.NONE);
            ReloadTaleLopHoc();
            JOptionPane.showMessageDialog(null, "Thêm thành
công", "Thành công", JOptionPane.INFORMATION_MESSAGE);
            return;
        }else {
            JOptionPane.showMessageDialog(null, "Thêm thất
bại", "Có lỗi xảy ra", JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
    if(chucNangDaChon==ChucNang.UPDATE) {
        if(CheckInput()==false)
            return;
        if(DatabaseManager.SuaLopHoc(msLop, tenLop, gvcn)) {
            btnUpdate.requestFocus();
            SwitchMode(ChucNang.NONE);
            ReloadTaleLopHoc();
            JOptionPane.showMessageDialog(null, "Sửa thành công",

```

```

        "Thành công", JOptionPane.INFORMATION_MESSAGE);

    return;
} else
{
    JOptionPane.showMessageDialog(null, "Sửa thất bại", "Có
lỗi ", JOptionPane.ERROR_MESSAGE);
    return;
}
}

private void txtMSLopKeyTyped(java.awt.event.KeyEvent evt) {
    if ((txtMSLop.getText() + evt.getKeyChar()).length() > 15)
        evt.consume();
}

private void txtTenLopKeyTyped(java.awt.event.KeyEvent evt) {
    if ((txtTenLop.getText() + evt.getKeyChar()).length() > 50)
        evt.consume();
}

private void txtGVCNKeyTyped(java.awt.event.KeyEvent evt) {
    if ((txtGVCN.getText() + evt.getKeyChar()).length() > 50)
        evt.consume();
}

```

Bài 4. Xây dựng ứng dụng quản lý CSDL theo mô hình 3 lớp.

Thiết Kế & Cài đặt CSDL: QLSanpham, có 2 Table:

- LoaiSP(Maloai char(2), Tenloai nvarchar(20))
- SanPham(MaSP char(4), TenSP nvarchar(20), Dongia BigInt, Maloai char(2))

Nhập liệu như sau:

LoaiSP

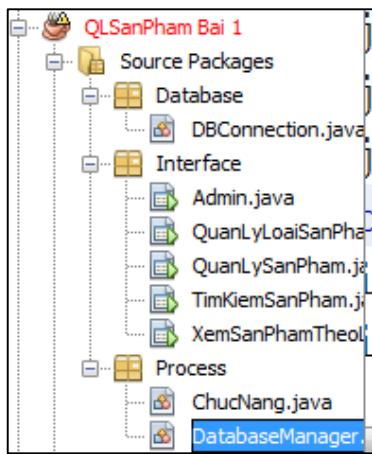
	Maloai	Tenloai
1	BK	Bánh keo
2	GK	Giải khát
3	MP	Mỹ phẩm

SanPham

	MaSP	TenSP	Dongia	Maloai
1	SP01	Bánh mì	10000	BK
2	SP02	Bánh bao	15000	BK
3	SP03	Coca c...	12000	GK
4	SP04	Pepsi	11000	GK
5	SP05	Kem ch...	85000	MP

Hướng dẫn:

Tạo Project: QLSanpham. Tạo 3 gói (Database, Interface, Process) tương ứng



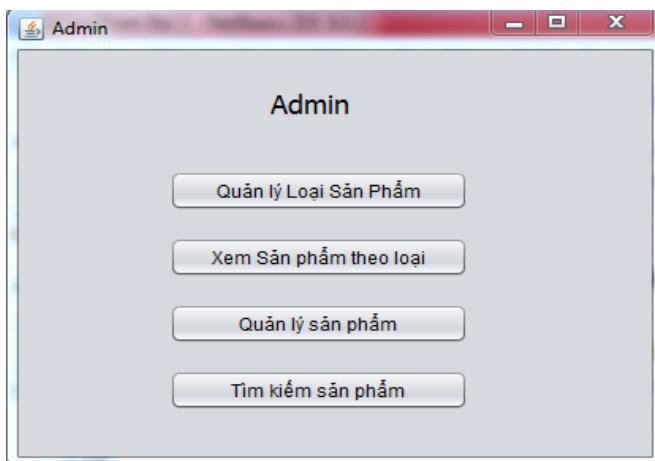
Trong packages Database tạo lớp DBConnection.java với nội dung tương tự file DBConnection.java trong bài tập trên; Thay tên cơ sở dữ liệu cần kết nối:

```

String dbName = "QLSanPham";
String connString =
"jdbc:sqlserver://localhost:1433;integratedSecurity=true;databaseName=" +
dbName;

```

Trong packages “Interface”, thiết kế màn hình ban đầu “Admin” gồm các chức năng:



Xử lý sự kiện khi bấm btnDanhMucLoaiSP mở form QuanLySanPham

```

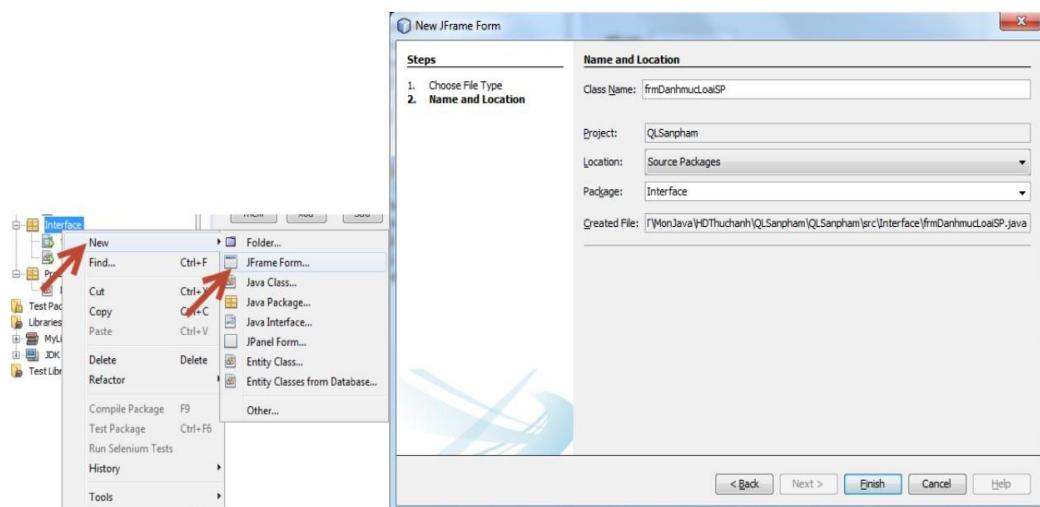
private void btnDanhMucLoaiSPActionPerformed(java.awt.event.ActionEvent evt) {
    QuanLyLoaiSanPham danhMucLoaiSP = new QuanLyLoaiSanPham();
    danhMucLoaiSP.admin = this;
    danhMucLoaiSP.setVisible(true);
    this.setVisible(false);
}

```

Tương tự viết sự kiện khi bấm btn XemSPTheoLoai hiện lên form XemSanPhamTheoLoai, bấm btnDanhMucSP hiện form QuanLySanPham, btnTimKiem hiện form TimKiemSanPham.

Thiết kế giao diện cho Form “QuanLySanPham”

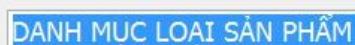
- Tạo mới JFrame Form trong Package Interface:



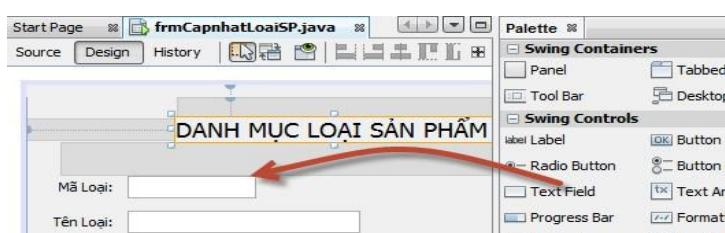
Kéo thả lần lượt các Control JLabel vào Form:



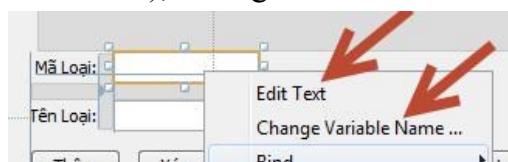
Double Click vào Control để cập nhật nhãn:



- Kéo thả lần lượt các Control JTextField vào Form:



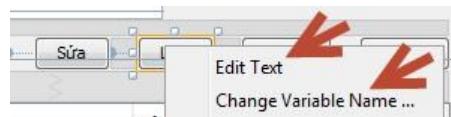
Click phải, chọn Edit Text (cập nhật nhãn), Change Variable Name (Đặt tên)



- Kéo thả lần lượt các Control JButton vào Form:



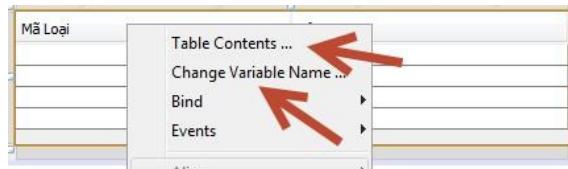
Đặt nhãn và tên cho các JButton (Tương tự JTextField và JLabel)



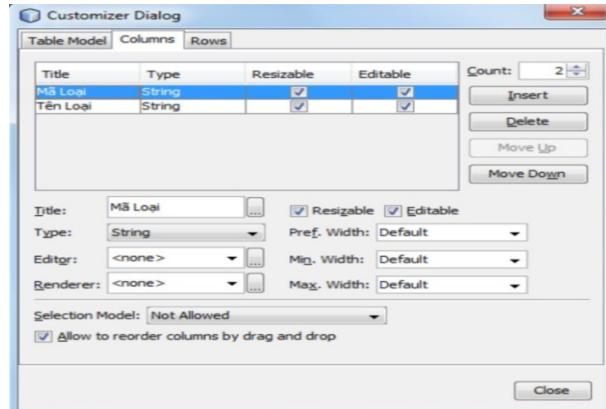
Kéo thả JTable vào Form:



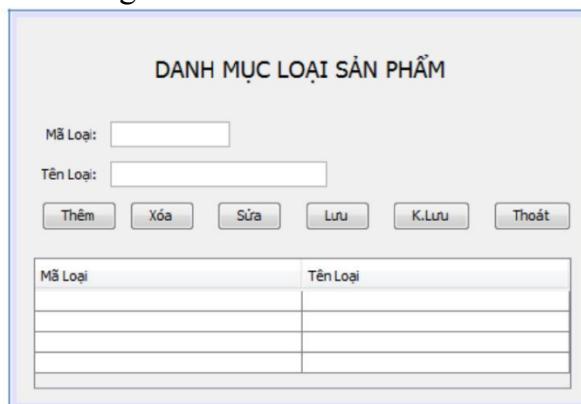
Click phải: Table Contents để điều chỉnh thiết kế, Change Variable Name (Đặt tên)



- Màn hình điều chỉnh thiết kế: Thực hiện thêm, xóa và cập nhật nhãn cho các tiêu đề cột



Mã nguồn file QuanLyLoaiSanPham.java, ngoài code tự sinh khi thiết kế giao diện dùng hình thức Design, cần bổ sung các code sau như sau:



```

public class QuanLyLoaiSanPham extends javax.swing.JFrame {

    DefaultTableModel dfTableModel;
    int chucNangDaChon = ChucNang.NONE;
    public Admin admin;
    public QuanLyLoaiSanPham() {
        initComponents();
        dfTableModel = (DefaultTableModel)tblLoaiSP.getModel();
    }
    //Phương thức load dữ liệu
    void ReloadLoaiSanPham() {
        if(DatabaseManager.LoaiSanPhamToTable(tblLoaiSP)==false) {
            JOptionPane.showMessageDialog(null, "Tải dữ liệu loại sản
phẩm có lỗi, vui lòng thử lại sau", "Lỗi CSDL",
JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
    //Phương thức thay đổi lựa chọn dòng dữ liệu trên bảng
    void TblLoaiSP_SelectionChanged() {
        int row = tblLoaiSP.getSelectedRow();
        if(row>=0) {
            String maLoai = (String)dfTableModel.getValueAt(row, 0);
            String tenLoai = (String)dfTableModel.getValueAt(row, 1);
            txtMaLoai.setText(maLoai);
            txtTenLoai.setText(tenLoai);
        }else {
            txtMaLoai.setText("");
            txtTenLoai.setText("");
        }
    }
    void SetStateControl(boolean trangThai) {
        btnAdd.setEnabled(trangThai);
        btnUpdate.setEnabled(trangThai);
    }
}

```

```

        btnDelete.setEnabled(trangThai);
        btnSave.setEnabled(!trangThai);
        btnDontSave.setEnabled(!trangThai);
        btnExit.setEnabled(trangThai);
        tblLoaiSP.setEnabled(trangThai); }

    }

    //Phương thức chọn chức năng
    void SwitchMode(int chucNang) {
        chucNangDaChon = chucNang;
        switch (chucNang) {
            case ChucNang.ADD: {
                SetStateControl(false);
                txtMaLoai.setEnabled(true);
                txtTenLoai.setEnabled(true);
                txtMaLoai.setText("");
                txtTenLoai.setText("");
                txtMaLoai.requestFocus();
                break;
            }
            case ChucNang.UPDATE:{

                SetStateControl(false);
                txtMaLoai.setEnabled(false);
                txtTenLoai.setEnabled(true);
                txtMaLoai.requestFocus();
                break;
            }
            case ChucNang.NONE: {
                SetStateControl(true);
                txtMaLoai.setEnabled(false);
                txtTenLoai.setEnabled(false);
                TblLoaiSP_SelectionChanged();
                break;
            }
        }
    }
}

```

```

    }

void Exit()
{
    int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn
thoát không?", "Thoát?", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);

    if(result == JOptionPane.CANCEL_OPTION)
        return;
    this.dispose();
    if(admin!=null)
        admin.setVisible(true);
}

boolean CheckInput() {
    String maLoai = txtMaLoai.getText().trim();
    String tenLoai = txtTenLoai.getText().trim();
    if(maLoai.length()==0) {
        JOptionPane.showMessageDialog(null, "Vui lòng nhập mã loại",
"Chưa nhập mã loại", JOptionPane.WARNING_MESSAGE);
        txtMaLoai.requestFocus();
        return false;
    }
    if(tenLoai.length()==0) {
        JOptionPane.showMessageDialog(null, "Vui lòng nhập tên loại",
"Chưa nhập tên loại", JOptionPane.WARNING_MESSAGE);
        txtTenLoai.requestFocus();
        return false;
    }
    return true;
}

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    ReloadLoaiSanPham();
    tblLoaiSP.getSelectionModel().addListSelectionListener(new
ListSelectionListener()
{
    @Override

```

```

public void valueChanged(ListSelectionEvent e) {
    TblLoaiSP_SelectionChanged();
}
});

}

private void txtMaLoaiKeyTyped(java.awt.event.KeyEvent evt) {
    if ((txtMaLoai.getText() + evt.getKeyChar()).length() > 2)
        evt.consume();
}

private void txtTenLoaiKeyTyped(java.awt.event.KeyEvent evt) {
    if ((txtTenLoai.getText() + evt.getKeyChar()).length() > 20)
        evt.consume();
}

//Code cho button Thêm

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    if(chucNangDaChon == ChucNang.NONE)
        SwitchMode(ChucNang.ADD);
    else
        SwitchMode(ChucNang.NONE);
}

private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    int rowSelected = tblLoaiSP.getSelectedRow();
    if(rowSelected<0) {
        JOptionPane.showMessageDialog(null, "Chưa chọn loại sản
phẩm nào để sửa", "Chưa chọn loại sản phẩm",
JOptionPane.WARNING_MESSAGE);
        return;
    }
    if(chucNangDaChon == ChucNang.NONE)
        SwitchMode(ChucNang.UPDATE);
    else
        SwitchMode(ChucNang.NONE);
}

private void btnDontSaveActionPerformed(java.awt.event.ActionEvent evt){

```

```

        SwitchMode(ChucNang.NONE);

    }

//Code cho chức năng xóa dữ liệu

private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt)
{
    int rowSelected = tblLoaiSP.getSelectedRow();
    if(rowSelected<0) {
        JOptionPane.showMessageDialog(null, "Bạn chưa chọn nhà xuất
bản", "Chưa chọn nhà xuất bản", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    int result = JOptionPane.showConfirmDialog(null, "Bạn muốn xóa
sản phẩm này không", "Xóa?", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);
    if(result == JOptionPane.CANCEL_OPTION)
        return;
    String maLoai = (String)tblLoaiSP.getValueAt(rowSelected, 0);
    if(DatabaseManager.Count("SanPham", "MaLoai", maLoai)>0)
    {
        JOptionPane.showMessageDialog(null, "Đã có sản phẩm nằm trong
loại sản phẩm này, không thể xóa!", "Không thể xóa",
JOptionPane.WARNING_MESSAGE);
        return;
    }
    if(DatabaseManager.XoaLoaiSanPham(maLoai))
    {
        ReloadLoaiSanPham();
        JOptionPane.showMessageDialog(null, "Xóa thành công", "Thành
công", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    else
    {
        JOptionPane.showMessageDialog(null, "Xóa thất bại", "Thất bại",
JOptionPane.ERROR_MESSAGE);
    }
}

```

```

        return;
    }

}

//Code cho chức năng Lưu

private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String maLoai = txtMaLoai.getText().trim();
    String tenLoai = txtTenLoai.getText().trim();
    if(chucNangDaChon == ChucNang.ADD) {
        if(CheckInput()==false)
            return;
        if(DatabaseManager.Count("LoaiSP", "MaLoai", maLoai)>0)
            JOptionPane.showMessageDialog(null, "Mã loại này đã tồn
tại trong csdl", "Trùng mã", JOptionPane.WARNING_MESSAGE);
        return;
    }
    if(DatabaseManager.ThemLoaiSanPham(maLoai, tenLoai)) {
        SwitchMode(ChucNang.NONE);
        ReloadLoaiSanPham();
        btnAdd.requestFocus();
        JOptionPane.showMessageDialog(null, "Thêm thành công", "Thành
công", JOptionPane.INFORMATION_MESSAGE);
        return;
    }else{
        JOptionPane.showMessageDialog(null, "Thêm thất bại", "Thất bại",
JOptionPane.ERROR_MESSAGE);
        return;
    }
}else if(chucNangDaChon==ChucNang.UPDATE) {
    if(CheckInput()==false)
        return;
    if(DatabaseManager.SuaLoaiSanPham(maLoai, tenLoai)) {
        SwitchMode(ChucNang.NONE);
        ReloadLoaiSanPham();
        btnUpdate.requestFocus();
    }
}

```

```

        JOptionPane.showMessageDialog(null, "Sửa thành công",
"Thành công", JOptionPane.INFORMATION_MESSAGE);

    return;
}

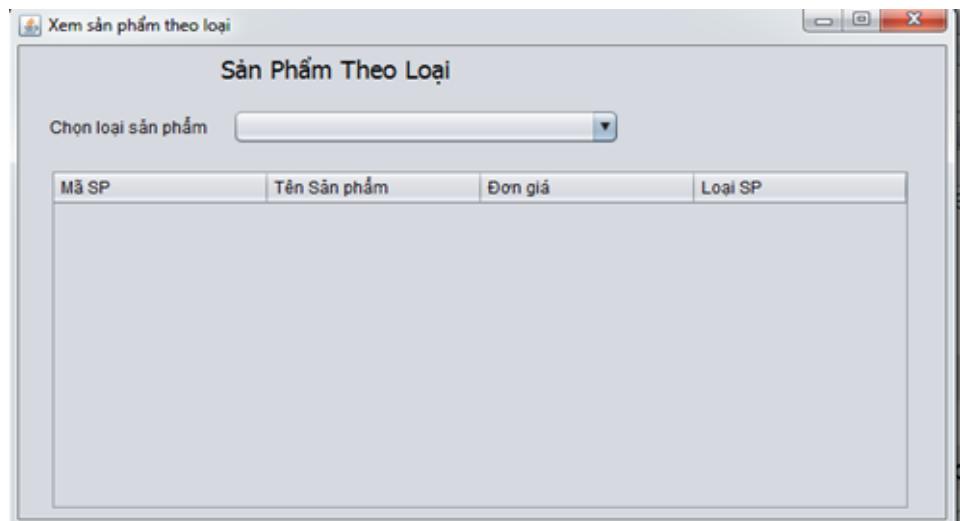
else
{
    btnAdd.requestFocus();

    JOptionPane.showMessageDialog(null, "Sửa thất bại", "Thất bại",
JOptionPane.ERROR_MESSAGE);

    return;
}
}

```

Tương tự thiết kế giao diện cho chức năng “Xem sản phẩm theo loại



Mã nguồn cho giao diện Sản Phẩm Theo Loại: Tạo Class XemSanPhamTheoLoai

```

public class XemSanPhamTheoLoai extends javax.swing.JFrame {

    public Admin admin;

    public XemSanPhamTheoLoai() {
        initComponents();
    }

    void ReloadSanPhamTheoLoaiDaChon() {
        String maLoai = (String)cbxLoaiSP.getSelectedItem();

        if(DatabaseManager.SanPhamToTable_ByLoaiSP(tblSanPham, maLoai)==false{
            JOptionPane.showMessageDialog(null, "Tải dữ liệu có lỗi, vui

```

```

lòng thử lại sau", "Lỗi CSDL", JOptionPane.ERROR_MESSAGE);

        return;
    }

}

void ReloadComboBoxLoaiSP() {
    if(DatabaseManager.LoaiSanPhamToComboBox(cbxLoaiSP)==false) {
        JOptionPane.showMessageDialog(null, "Tải dữ liệu loại sản
phẩm có lỗi, vui lòng thử lại", "Lỗi CSDL", JOptionPane.ERROR_MESSAGE);
        return;
    }
}

void Exit() {
int result = JOptionPane.showConfirmDialog(null, "Bạn muốn thoát không?", "Thoát?", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE);
    if(result == JOptionPane.CANCEL_OPTION) {
        return;
    }
    this.dispose();
    if(admin!=null)
        admin.setVisible(true);
}

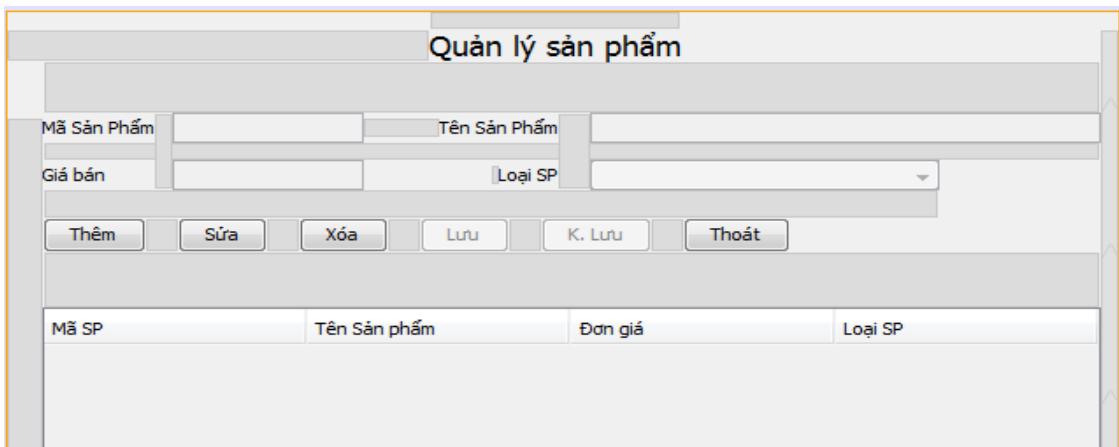
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    ReloadComboBoxLoaiSP();
}

private void cbxLoaiSPItemStateChanged(java.awt.event.ItemEvent evt) {
    ReloadSanPhamTheoLoaiDaChon();
}

private void formWindowClosing(java.awt.event.WindowEvent evt) {
    Exit();
}

```

Thiết kế giao diện cho form quản lý sản phẩm



Tạo class QuanLySanPham

```
public class QuanLySanPham extends javax.swing.JFrame {  
    DefaultTableModel tableView;  
    int chucNangDaChon = ChucNang.NONE;  
    public Admin admin;  
    public QuanLySanPham() {  
        initComponents();  
        tableView = (DefaultTableModel)tblSanPham.getModel();  
    }  
    void ReloadSanPham() {  
        if(DatabaseManager.SanPhamToTable(tblSanPham)==false) {  
            JOptionPane.showMessageDialog(null, "Tải dữ liệu sản phẩm có  
lỗi, vui lòng thử lại sau", "Lỗi CSDL", JOptionPane.ERROR_MESSAGE);  
            return;  
        }  
    }  
    void ReloadComboBoxLoaiSP() {  
        if(DatabaseManager.LoaiSanPhamToComboBox(cbxLoaiSP)==false) {  
            JOptionPane.showMessageDialog(null, "Tải dữ liệu loại sản phẩm  
có lỗi, vui lòng thử lại sau", "Lỗi CSDL", JOptionPane.ERROR_MESSAGE);  
            return;  
        }  
    }  
    void TblDanhMucSP_SelectionChanged() {  
        int row = tblSanPham.getSelectedRow();  
        if(row>=0) {  
    
```

```

        String maSP = (String)tableModel.getValueAt(row, 0);
        String tenSP = (String)tableModel.getValueAt(row, 1);
        String giaBan = (String)tableModel.getValueAt(row, 2);
        String loaiSP = (String)tableModel.getValueAt(row, 3);
        txtMaSP.setText(maSP);
        txtTenSP.setText(tenSP);
        txtPrice.setText(giaBan);
        cbxLoaiSP.setSelectedItem(loaiSP);

    }else {
        ClearTextField();
    }
}

void SetStateControl(boolean trangThai) {
    btnAdd.setEnabled(trangThai);
    btnUpdate.setEnabled(trangThai);
    btnDelete.setEnabled(trangThai);
    btnSave.setEnabled(!trangThai);
    btnNotSave.setEnabled(!trangThai);
    btnExit.setEnabled(trangThai);
    tblSanPham.setEnabled(trangThai);
}

void SetStateTextField(boolean trangThai) {
    txtMaSP.setEnabled(trangThai);
    txtTenSP.setEnabled(trangThai);
    txtPrice.setEnabled(trangThai);
    cbxLoaiSP.setEnabled(trangThai);
}

void ClearTextField() {
    txtMaSP.setText("");
    txtTenSP.setText("");
    txtPrice.setText("");
}

void SwitchMode(int chucNang) {
    chucNangDaChon = chucNang;
    switch (chucNang) {

```

```

        case ChucNang.ADD: {
            SetStateControl(false);
            SetStateTextField(true);
            ClearTextField();
            txtMaSP.requestFocus();
            break;
        }
        case ChucNang.UPDATE: {
            SetStateControl(false);
            SetStateTextField(true);
            txtMaSP.setEnabled(false);
            txtTenSP.requestFocus();
            break;
        }
        case ChucNang.NONE: {
            SetStateControl(true);
            SetStateTextField(false);
            TblDanhMucSP_SelectionChanged();
            break;
        }
    }
}

void Exit() {
    int result = JOptionPane.showConfirmDialog(null, "Bạn có muốn
thoát?", "Thoát?", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);
    if(result == JOptionPane.CANCEL_OPTION)
        return;
    this.dispose();
    if(admin!=null)
        admin.setVisible(true);
}
boolean CheckInput() {
    String maSP = txtMaSP.getText().trim();

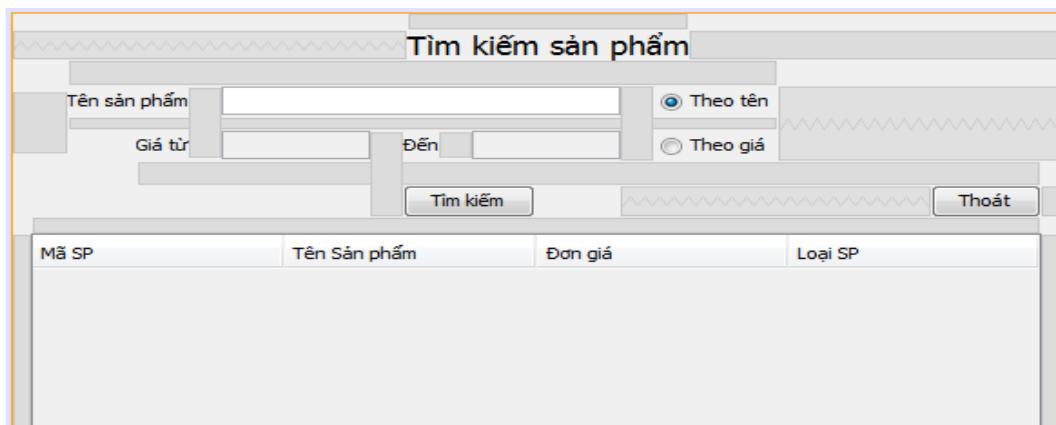
```

```

String tenSP = txtTenSP.getText().trim();
String price = txtPrice.getText().trim();
String maLoai = (String)cbxLoaiSP.getSelectedItem();
if(maSP.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập mã sản
phẩm", "Chưa nhập mã sản phẩm", JOptionPane.WARNING_MESSAGE);
    txtMaSP.requestFocus();
    return false;
}
if(tenSP.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập tên
sản phẩm", "Chưa nhập tên sản phẩm",
JOptionPane.WARNING_MESSAGE);
    txtTenSP.requestFocus();
    return false;
}
if(price.length()==0) {
    JOptionPane.showMessageDialog(null, "Vui lòng nhập giá bán",
"Chưa nhập giá bán", JOptionPane.WARNING_MESSAGE);
    txtPrice.requestFocus();
    return false; }
if(maLoai == null || maLoai.length() == 0) {
    JOptionPane.showMessageDialog(null, "Chưa có mã loại. Vui
lòng thêm ít nhất một mã loại vào CSDL", "Chưa có mã loại",
JOptionPane.WARNING_MESSAGE);
    btnExit.requestFocus();
    return false;
} return true;
}

```

Thiết kế giao diện Tìm kiếm sản phẩm như sau



Tạo class TimKiemSanPham, code cho chức năng tìm kiếm sản phẩm.

```

private void btnTimKiemActionPerformed(java.awt.event.ActionEvent evt) {
    if(rdbTheoTen.isSelected()==true)  {
        String tenSP  = txtTenSP.getText().trim();
        if(tenSP.length()==0){
            JOptionPane.showMessageDialog(null, "Vui lòng nhập từ khóa
tên sản phẩm", "Chưa nhập thông tin cần
thiết", JOptionPane.WARNING_MESSAGE);
            txtTenSP.requestFocus();
            return;
        }
        if(DatabaseManager.SearchSanPhamWithName(tblDanhMucSP, tenSP)==false
        {
            JOptionPane.showMessageDialog(null, "Tìm kiếm thất bại", "Lỗi
CSDL", JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
    if(rdbTheoGia.isSelected()==true)
    {
        String giaTu = txtGiaTu.getText().trim();
        String giaDen = txtGiaDen.getText().trim();
        if(giaTu.length()==0){
            JOptionPane.showMessageDialog(null, "Vui lòng nhập giá
từ", "Chưa nhập thông tin cần thiết", JOptionPane.WARNING_MESSAGE);
            txtGiaTu.requestFocus();
        }
    }
}

```

```

        }

        if(giaDen.length()==0){
            JOptionPane.showMessageDialog(null, "Vui lòng nhập giá đến", "Chưa
nhập thông tin cần thiết", JOptionPane.WARNING_MESSAGE);
            txtGiaDen.requestFocus();
            return;
        }

        if(DatabaseManager.SearchSanPhamWithPrice(tblDanhMucSP, giaTu,
giaDen)==false)
        {
            JOptionPane.showMessageDialog(null, "Tìm kiếm thất bại", "Lỗi
CSDL", JOptionPane.ERROR_MESSAGE);
        return ;
        }
    }
}

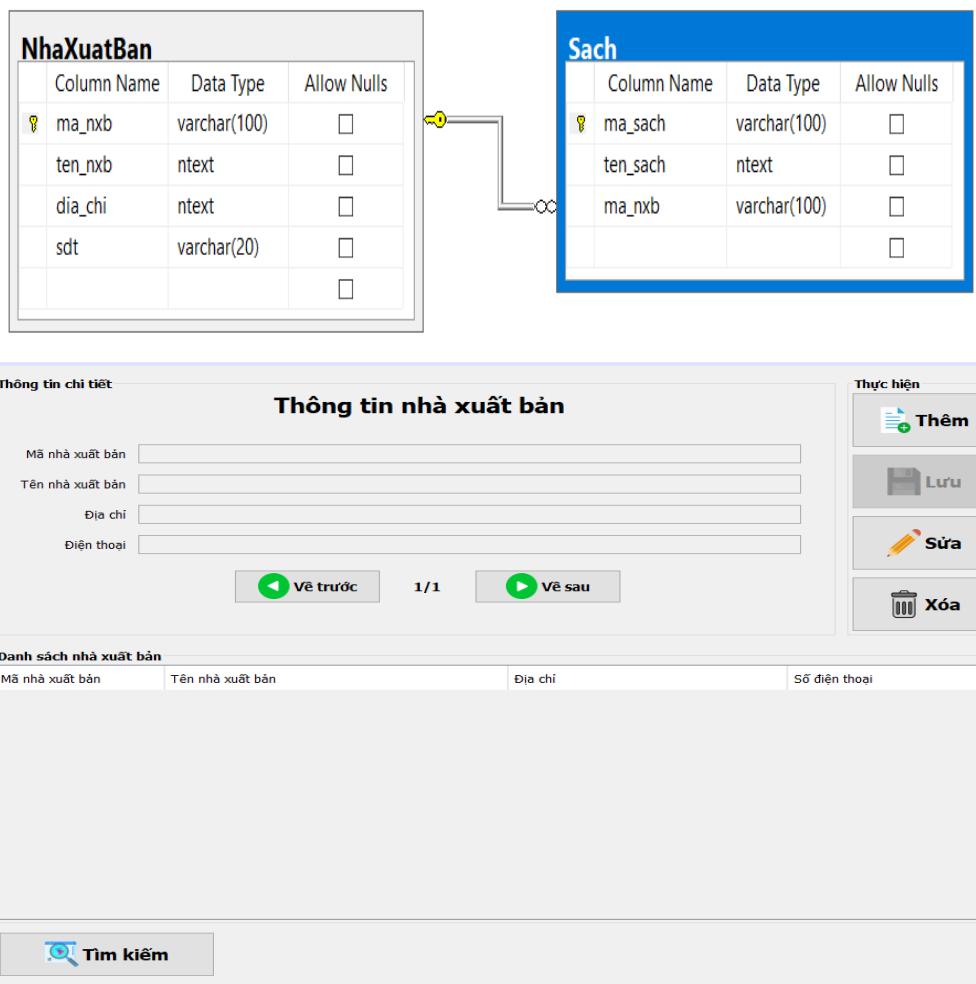
private void txtGiaTuKeyTyped(java.awt.event.KeyEvent evt) {
    try
    {
        String kyTuVuaNhap = evt.getKeyChar() + "";
        Integer.parseInt(kyTuVuaNhap);
    }catch(NumberFormatException ex) {
        evt.consume();
    }
}

private void txtGiaDenKeyTyped(java.awt.event.KeyEvent evt) {
    try {
        String kyTuVuaNhap = evt.getKeyChar() + "";
        Integer.parseInt(kyTuVuaNhap);
    }catch(NumberFormatException ex) {
        evt.consume();
    }
}
}

```

Bài 5. Tổng hợp

Xây dựng ứng dụng quản lý thư viện, thiết kế và cài đặt CSDL trong SQL



Khi thực hiện chức năng tìm kiếm hiện màn hình giao diện sau

Tim kiem

Nhập từ khóa mã nhà xuất bản **Tim kiem**

Mã sách	Tên sách	Nhà xuất bản

Hình 7. Một số màn hình quản lý Thư viện

Gợi ý: Các chức năng trong màn hình chính tương tự các bài đã làm, đối với chức năng tìm kiếm dùng khóa **like** và **%** trong câu truy vấn tìm kiếm tương đối.

LAB 7. ĐỊA CHỈ IP, GIAO TIẾP MẠNG (NIC)[6]

A. MỤC TIÊU

- Trang bị cho sinh viên các thao tác lập trình với địa chỉ IP sử dụng lớp InetAddress nằm trong gói java.net, lập trình với giao tiếp mạng sử dụng lớp NetworkInterface.

B. NỘI DUNG

- Lập trình với InetAddress
- Lập trình hiển thị các giao tiếp mạng và tham số đi kèm.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Sử dụng lớp InetAddress, sinh viên có thể tìm địa chỉ IP và hostname của máy cục bộ, lấy được địa chỉ của máy trạm thông qua tên miền cho trước.
- Sử dụng lớp NetworkInterface sinh viên có thể lấy được danh sách các giao tiếp mạng và các địa chỉ của chúng.

D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8.

E. HƯỚNG DẪN

1. Lớp InetAddress

Biểu diễn một địa chỉ Internet, bao gồm hai trường thông tin: hostName (một đối tượng kiểu **String**)- tên máy trạm và address (một số kiểu **int**)- địa chỉ IP của máy trạm đó.

Một số phương thức:

- getAddress() : Trả về địa chỉ IP chứa trong đối tượng InetAddress dạng mảng byte
- getAllByName(**String** host) : Trả về mảng địa chỉ của tất cả các máy trạm có cùng tên
- getHostName() : Trả về tên máy trạm chứa trong đối tượng.
- getLocalHost() : Trả về địa chỉ của máy cục bộ

Tạo đối tượng, sử dụng các phương thức tĩnh : getLocalHost(), getByName(**String**), getAllByName(**String**).

Tất cả các phương thức này đều thực hiện kết nối tới Server DNS cục bộ để biết được các thông tin trong đối tượng InetAddress.

Bài 1. In ra địa chỉ IP và hostname (tên máy trạm) của máy cục bộ

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

```

    {
        InetAddress address = InetAddress.getLocalHost();
        System.out.println("Hello. My name is " +
address.getHostName() + " and my IP address is " +
address.getHostAddress());
    }
    catch (UnknownHostException e)
    {
        System.out.println("I don't know my own name and address.");
    }
}
}

```

Bài 2. In ra địa chỉ các máy trạm trên mạng có cùng tên miền www.microsoft.com.

```

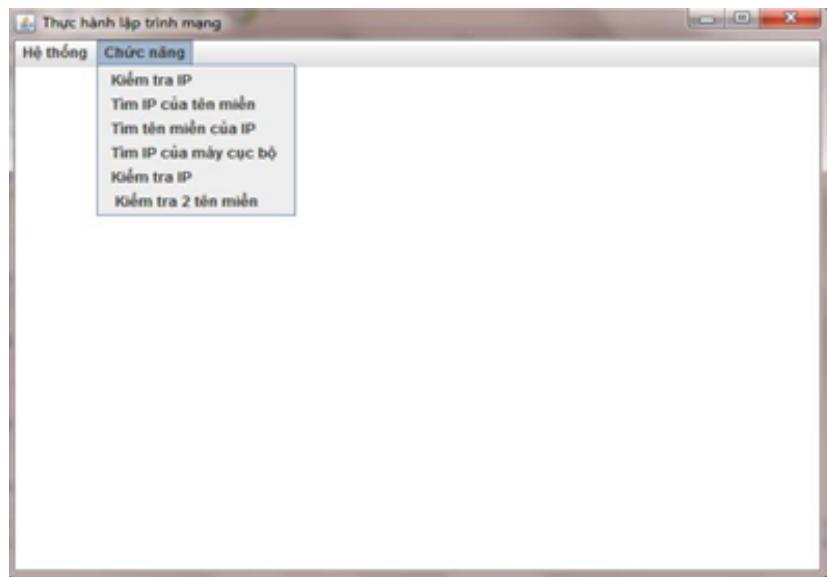
import java.net.*;
public class AllAddr
{
    public static void main(String[] args)
    {
        try
        {
            InetAddress[] addr = InetAddress.getAllByName("www.microsoft.com");
            for (int i = 0; i < addr.length; i++)
            {
                System.out.println (addr[i]);
            }
        }
        catch (UnknownHostException ex)
        {
            System.out.println("Could not find www.microsoft.com");
        }
    }
}

```

Dịch chạy chương trình trên máy tính có kết nối mạng Internet, kết quả trả về như sau:

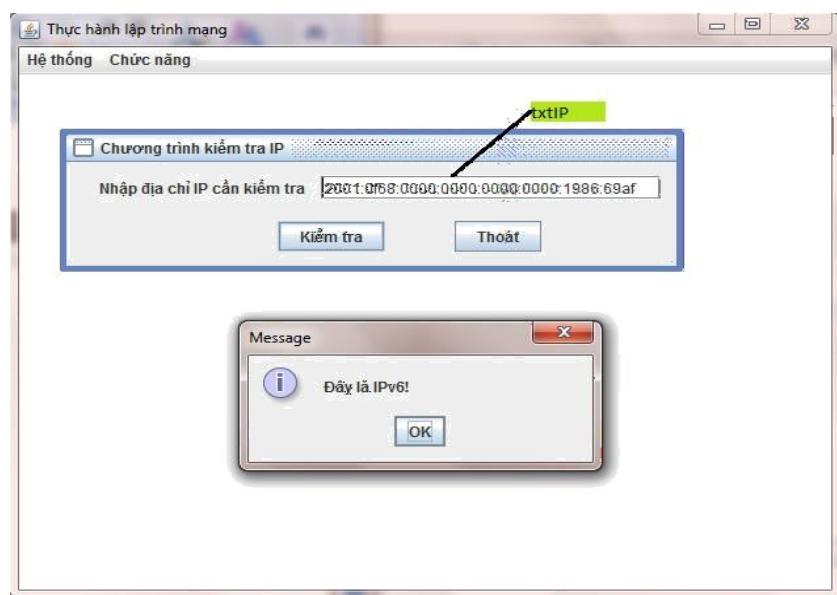
www.microsoft.com/63.211.66.123
www.microsoft.com/63.211.66.124
www.microsoft.com/63.211.66.131
www.microsoft.com/63.211.66.11

Bài 3. Tạo giao diện như hình sau: Thiết kế các chức năng tương ứng



a. Kiểm tra một tên miền có địa chỉ IP là phiên bản 4 hay phiên bản 6.

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button Kiểm tra

```
private void btnKiemTraActionPerformed(java.awt.event.ActionEvent evt) {  
    String IP = txtIP.getText();  
    try  
    {  
        InetAddress host = InetAddress.getByName(IP);  
        if(host != null)  
        {  
            if(IP.contains("."))  
                System.out.println("Đây là IPv4");  
            else  
                System.out.println("Đây là IPv6");  
        }  
    }  
}
```

```

        JOptionPane.showMessageDialog(null, "Đây là IPv4");

    else

        JOptionPane.showMessageDialog(null, "Đây là IPv6");

    }

else

{

JOptionPane.showMessageDialog(null, "Địa chỉ IP nhập sai!!!!");

}

}

catch (UnknownHostException ex)

{

    JOptionPane.showMessageDialog(null, "Địa chỉ IP nhập sai:

\n" + ex.toString());

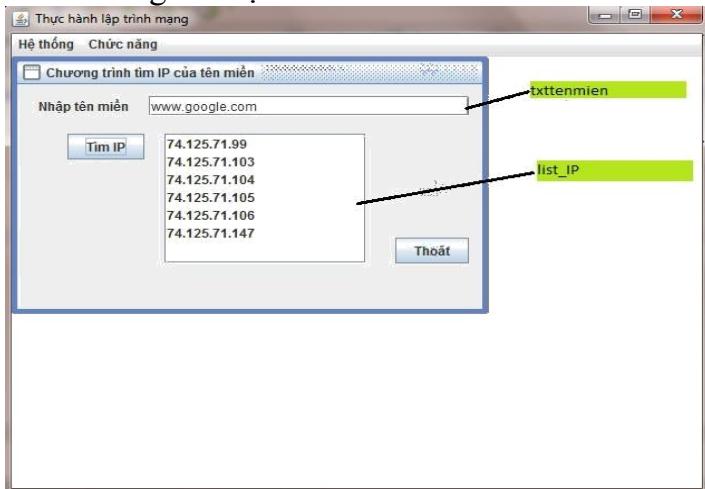
}

}

```

b. Nhập địa chỉ tên miền. Cho biết tất cả địa chỉ IP của tên miền tương ứng.

Bước 1: Tạo JFrameForm có giao diện như sau



Bước 2: Xử lý sự kiện cho button tìm IP

```

private void btnTimIPActionPerformed(java.awt.event.ActionEvent evt) {

    try
    {
        int i, j;
        InetAddress addr[] =
InetAddress.getAllByName(txtTenMien.getText());
        DefaultListModel dlm = new DefaultListModel();

```

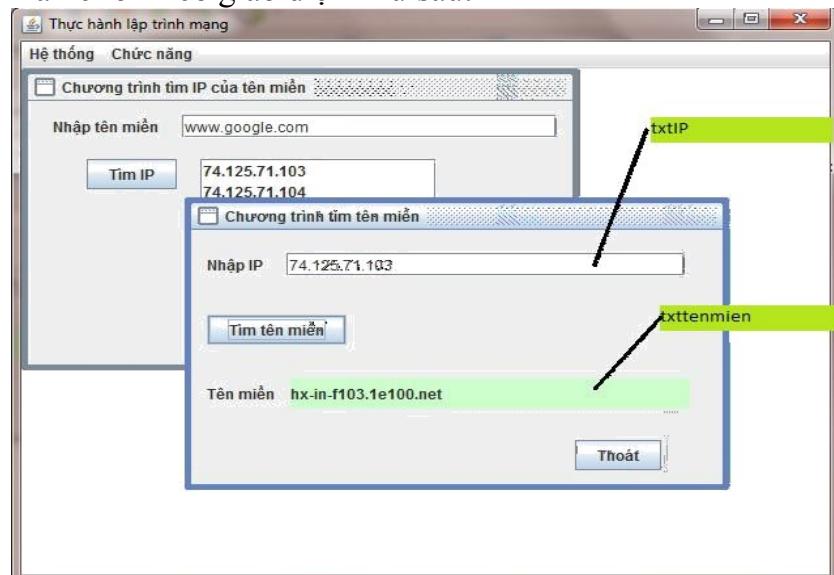
```

        for (i = 0; i < addr.length; i++)
        {
            byte[] ipAddr = addr[i].getAddress();
            String ipAddrStr = "";
            for (j = 0; j < ipAddr.length; j++)
            {
                if(j > 0)
                    ipAddrStr += ".";
                ipAddrStr += ipAddr[j]&0xFF;
            }
            dlm.addElement(ipAddrStr);
        }
        list_IP.setModel(dlm);
    }
    catch(UnknownHostException e)
    {
        JOptionPane.showMessageDialog(null,"Địa chỉ nhập sai!!!!");
    }
}

```

c. Viết chương trình nhập vào địa chỉ IP của một máy. Cho biết tên miền tương ứng.

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button tìm tên miền.

```

private void btnTimTenMienActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        InetAddress addr = InetAddress.getByName(txtIP.getText());
        String hostname = addr.getHostName();
    }
}

```

```

        txtTenMien.setText(hostname);
    }
    catch (UnknownHostException ex)
    {
        JOptionPane.showMessageDialog(null, "Bạn nhập sai tên miền");
    }
}

```

d. Viết chương trình cho biết địa chỉ IP của máy cục bộ.

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button Tìm IP.

```

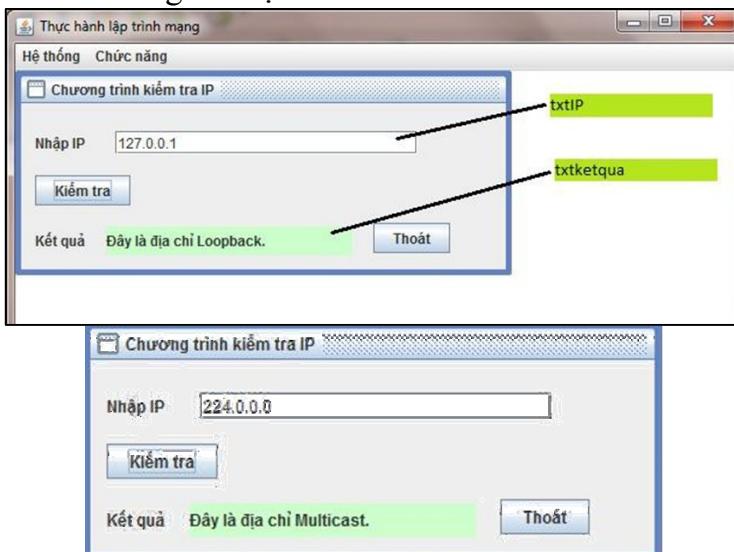
private void btnTimIPCuCBoActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        InetAddress addr = InetAddress.getLocalHost();
        byte[] ipAddr = addr.getAddress();
        String ipAddrStr = "";
        for (int i = 0; i < ipAddr.length; i++)
        {
            if(i > 0)
                ipAddrStr += ".";
            ipAddrStr += ipAddr[i]&0xFF;
        }
        txtIP.setText(ipAddrStr);
    }
    catch (UnknownHostException ex)
    {
        JOptionPane.showMessageDialog(null, "Lỗi!!!!");
    }
}

```

```
        }  
    }  
}
```

e. Nhập địa chỉ IP, kiểm tra đặc điểm của địa chỉ IP này có là một trong các dạng sau: **địa chỉ cục bộ, địa chỉ loopback, địa chỉ multicast**.

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button Kiểm tra.

```
private void btnKTxDacDiemIPActionPerformed(java.awt.event.ActionEvent evt)  
{  
    try  
    {  
        InetAddress add = InetAddress.getByName(txtIP.getText());  
        InetAddress localhost = InetAddress.getLocalHost();  
        if(add.equals(localhost))  
        {  
            txtKetQua.setText("Đây là địa chỉ localhost");  
        }  
        else if(add.isMulticastAddress())  
        {  
            txtKetQua.setText("Đây là địa chỉ Multicast");  
        }  
        else if(add.isLoopbackAddress())  
        {  
            txtKetQua.setText("Đây là địa chỉ Loopback");  
        }  
        else  
        {  
            txtKetQua.setText("Không thấy gì đặc biệt");  
        }  
    }  
}
```

```

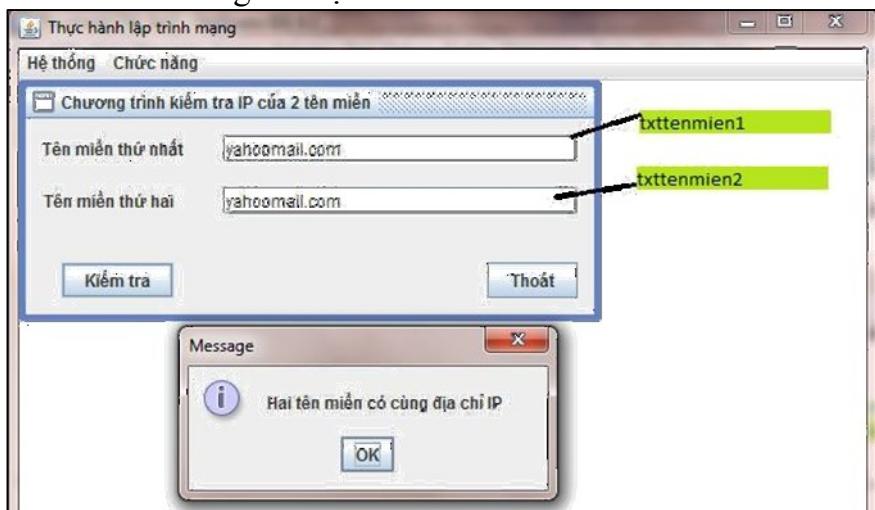
        }
    }

    catch (UnknownHostException ex)
    {
    }
}

```

f. Viết chương trình nhập vào 2 địa chỉ tên miền khác nhau. Kiểm tra xem hai tên miền này có cùng địa chỉ IP không?

Bước 1: Tạo JFrameForm có giao diện như sau:



Bước 2: Xử lý sự kiện cho button Kiểm tra.

```

private void btnKiemTraCungIPActionPerformed(java.awt.event.ActionEvent evt)
{
    InetAddress add1[], add2[];
    try
    {
        add1 = InetAddress.getAllByName(txtTenMien1.getText());
        add2 = InetAddress.getAllByName(txtTenMien2.getText());
        if(Arrays.equals(add1, add2))
            JOptionPane.showMessageDialog(null, "Hai tên miền cùng địa chỉ IP");
        Else
            JOptionPane.showMessageDialog(null, "Hai tên miền không cùng địa chỉ IP");
    }
    catch(UnknownHostException ex)
    {
        JOptionPane.showMessageDialog(null, ex.toString());
    }
}

```

2. Lớp NetworkInterface

Cung cấp các phương thức để liệt kê tất cả các địa chỉ cục bộ và tạo ra đối tượng InetAddress.

getByName():

Trả về đối tượng NetworkInterface biểu diễn một bộ giao tiếp mạng với tên cụ thể.

getByInetAddress():

Trả về đối tượng NetworkInterface biểu diễn giao tiếp mạng được gắn với với một địa chỉ IP cụ thể.

getNetworkInterfaces():

Trả về đối tượng java.util.Enumeration là một danh sách liệt kê tất cả các giao tiếp mạng có trên máy cục bộ

Bài 4 : Viết chương trình hiện danh sách tất cả các giao tiếp mạng trên máy cục bộ.

```
public class InterfaceLister {  
    public static void main(String[] args) throws Exception {  
        Enumeration interfaces = NetworkInterface.getNetworkInterfaces();  
        while (interfaces.hasMoreElements()) {  
            NetworkInterface ni = (NetworkInterface) interfaces.nextElement();  
            System.out.println(ni);  
        }  
    }  
}
```

Bài 5: Viết chương trình hiển thị tên của tất cả các giao tiếp mạng và giao tiếp con (nếu nó tồn tại) trên một máy.

```
public class ListNIFs {  
    public static void main(String args[]) throws SocketException {  
        Enumeration<NetworkInterface> nets =  
            NetworkInterface.getNetworkInterfaces();  
        for (NetworkInterface netIf : Collections.list(nets)) {  
            out.printf("Display name: %s\n", netIf.getDisplayName());  
            out.printf("Name: %s\n", netIf.getName());  
            displaySubInterfaces(netIf);  
            out.printf("\n");  
        }  
    }  
    static void displaySubInterfaces(NetworkInterface netIf) throws  
        SocketException {  
        Enumeration<NetworkInterface> subIfs = netIf.getSubInterfaces();  
        for (NetworkInterface subIf : Collections.list(subIfs)) {  
            out.printf("\tSub Interface Display name: %s\n",  
                subIf.getDisplayName());  
        }  
    }  
}
```

```

        subIf.getDisplayName());
        out.printf("\tSub Interface Name: %s\n", subIf.getName());
    }
}
}

```

Truy cập các tham số giao tiếp mạng

Người sử dụng có thể truy cập các tham số về giao tiếp mạng ngoài tên và địa chỉ IP gán cho nó. Chương trình có thể phát hiện giao tiếp mạng đang chạy với phương thức isUp(). Các phương thức sau chỉ thị kiểu giao tiếp mạng:

isLoopback(): chỉ thị giao tiếp mạng là một giao tiếp loopback.

isPointToPoint(): chỉ thị nếu giao tiếp là giao tiếp point-to-point.

isVirtual(): chỉ thị nếu giao tiếp là giao tiếp ảo (giao tiếp mềm).

supportsMulticast(): chỉ thị khi giao tiếp mạng hỗ trợ multicast.

getHardwareAddress(): trả về địa chỉ phần cứng vật lý của giao tiếp mạng, địa chỉ MAC

getMTU(): trả về đơn vị truyền cực đại(MTU) là kích cỡ gói tin lớn nhất.

Bài 6:

Viết chương trình hiển thị danh sách các giao tiếp mạng và các tham số đi kèm.

```

public static void main(String args[]) throws SocketException {
    Enumeration<NetworkInterface> nets
    =NetworkInterface.getNetworkInterfaces();
    for (NetworkInterface netint : Collections.list(nets))
        displayInterfaceInformation(netint);
    }

static void displayInterfaceInformation(NetworkInterface netint) throws
    SocketException {
    out.printf("Display name: %s \n", netint.getDisplayName());
    out.printf("Name: %s\n", netint.getName());
    Enumeration<InetAddress> inetAddresses = netint.getInetAddresses();
    for (InetAddress inetAddress : Collections.list(inetAddresses)) {
        out.printf("InetAddress: %s\n", inetAddress);
    }
    out.printf("Up? %s\n", netint.isUp());
    out.printf("Loopback? %s\n", netint.isLoopback());
    out.printf("PointToPoint? %s\n", netint.isPointToPoint());
    out.printf("Supports multicast? %s\n", netint.supportsMulticast());
    out.printf("Virtual? %s\n", netint.isVirtual());
    out.printf("Hardware address:
%s\n", Arrays.toString(netint.getHardwareAddress()));
    out.printf("MTU: %s\n", netint.getMTU());
}

```

```
    out.printf("\n");
}
}
}
```

LAB 8. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG TCP [4,6,9]

A. MỤC TIÊU

Trang bị cho sinh viên các kỹ thuật lập trình hướng kết nối:

- Sử dụng lớp Socket xây dựng ứng dụng kết nối gửi nhận dữ liệu phía Client theo cơ chế TCP.
- Sử dụng Server Socket, xây dựng ứng dụng Client-Server theo cơ chế TCP.

B. NỘI DUNG

- Viết các ứng dụng phía Client sử dụng **class** Socket.
- Viết các ứng dụng phía Server sử dụng **class** Server Socket.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Hiểu được nguyên lý lập trình mô phỏng các giao thức mạng.
- Viết được chương trình kết nối TCP thực hiện trao đổi dữ liệu giữa Client và Server.
- Xây dựng ứng dụng mạng theo mô hình 3 lớp.

D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8.

E. HƯỚNG DẪN

1. Giao thức hướng kết nối- TCP: sử dụng kết nối (ảo) để truyền thông.

Đặc điểm: Truyền thông theo kiểu điểm-điểm

Quá trình truyền thông được thực hiện qua 3 giai đoạn:

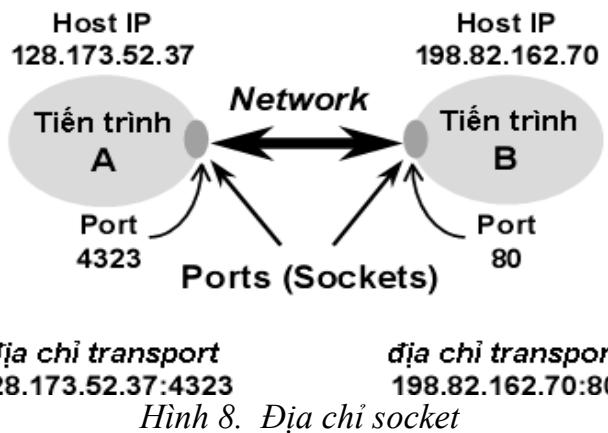
- Thiết lập kết nối
- Truyền dữ liệu kèm theo cơ chế kiểm soát chật chẽ
- Huỷ bỏ kết nối

2. Giao diện socket, địa chỉ socket

Socket là giao diện và đóng vai trò như một điểm cuối (end point) để truyền thông. Mỗi tiến trình khi muốn truyền thông bằng socket, đầu tiên phải tạo một socket và được gán một định danh duy nhất được gọi là **địa chỉ socket**.

Giao diện socket chia làm các loại sau:

- Stream socket: Cho phép truyền thông với giao thức TCP. TCP sử dụng một cặp stream socket để kết nối ứng dụng này với ứng dụng khác qua Internet.
 - Datagram socket: Cho phép truyền thông với giao thức UDP. UDP sử dụng một cặp datagram socket để kết nối từ một ứng dụng này tới một ứng dụng khác qua Internet
- Một địa chỉ socket là một tổ hợp gồm 2 địa chỉ: IP và cổng (port)- xác định một đầu mút cuối truyền thông. Nó chỉ ra tiến trình nào (port) chạy trên máy nào (IP).



Hình 8. Địa chỉ socket

3. Một số lớp lập trình với TCP

- a) Socket
- b) ServerSocket

3.1. Viết các ứng dụng kết nối phía Client sử dụng lớp Socket

Tạo đối tượng Socket phía Client.

Các phương thức tạo:

```
public Socket (String host, int port) throws UnknownHostException, IOException
public Socket (InetAddress address, int port) throws IOException
public Socket (String host, int port, InetAddress localaddr, int localPort)
```

Các phương thức:

```
public InputStream getInputStream(): Trả về luồng nhập của socket.
public OutputStream getOutputStream(): Trả về luồng xuất của socket.
public void close() throws IOException: Đóng socket
```

Bài 1. Viết ứng dụng phía Client truy xuất đến một trang web sử dụng class Socket

Hướng dẫn :

```
import java.io.*;
import java.net.Socket;
public class ConnectHttpUsingSocket {
    public static void main(String[] args) throws Exception{
        //Kết nối
        Socket socket=new Socket("localhost", 80);
        //Gửi yêu cầu
        OutputStream os = socket.getOutputStream();
        PrintWriter out=new PrintWriter(os,true);
        out.println("GET / HTTP/1.0");
        out.println();
        out.flush();
        //Nhận dữ liệu
        InputStream is = socket.getInputStream();
        int len=0;
```

```

byte []buffer=new byte[4086];
while((len=is.read(buffer))!=-1)
{
    String data=new String(buffer,0,len);
    System.out.println(data);
}
socket.close();
}
}

```

Bài 2.

Viết ứng dụng phía Client trả về chuỗi thời gian của một máy chủ đã cho, sử dụng dịch vụ ngày giờ chuẩn có sẵn trong cổng TCP 13. Các đối số là tên máy chủ và số cổng (số cổng mặc định là 13). Nếu chương trình chạy không có đối số sẽ in ngày giờ của localhost.

Hướng dẫn:

1. Xử lý đối số
2. Tạo kết nối mạng
3. Tạo luồng dữ liệu
4. Đọc chuỗi thời gian và in ra dạng chuẩn
5. Đóng kết nối

Tạo class DaytimeClient

```

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

public class DaytimeClient {
    static final int defaultPort = 13;
    public static void main(String[] args) {
        int portNumber;
        Socket ClientSocket;
        BufferedReader timeStream;
        String hostName;
        switch(args.length) {
            case 1: hostName = args[0];
                      portNumber = defaultPort;//daytimePort;
                      break;
            case 2: hostName = args[0];
                      portNumber = new Integer(args[1]).intValue();
                      break;
            default:
                      hostName = "localhost";
                      portNumber = defaultPort;
        }
        try {
            ClientSocket = new Socket(hostName, portNumber);
            timeStream = new BufferedReader(new InputStreamReader(ClientSocket.getInputStream()));
            String strTime = timeStream.readLine();
            System.out.println(strTime);
            ClientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

        }
        try {
            // Thực hiện kết nối tới Server
            ClientSocket = new Socket(hostName, portNumber);
            // Tạo luồng dữ liệu từ kết nối
            timeStream = new BufferedReader(new
                InputStreamReader(ClientSocket.getInputStream()));
            // Lấy dữ liệu từ Server
            String timeString = timeStream.readLine();
            System.out.println("It is " + timeString + " at " + hostName);
            // Đóng kết nối
            timeStream.close();
            ClientSocket.close();
        }
        catch (UnknownHostException e) {
            InterruptedException(" Unknown host error");
        }
        catch (ConnectException e) {
            System.out.println(" Service unavailable on port
"+portNumber+"of host "+hostName);
        }
        catch (IOException e) {
            InterruptedException(" Communication error occurred\r\n "+e);
        }
    }
}

```

Bài 3.

Viết ứng dụng đọc vào một chuỗi từ máy khách kết nối với máy chủ qua cổng TCP xác định. Nếu gặp dấu ‘.’: đóng kết nối và dừng lại, ngược lại sẽ gửi chuỗi đến máy chủ. Sau khi gửi đi, máy khách nhận trả lời từ máy chủ trả về là một chuỗi định dạng chuẩn. Các đối số đầu vào là tên máy chủ và số cổng (7).

Hướng dẫn:

1. Tạo đối số
2. Tạo kết nối mạng
3. Tạo luồng dữ liệu
4. Nhập vào mỗi chuỗi khác dấu “.” và gửi tới Server. Ngược lại đóng kết nối
5. Đọc xâu từ Server và in ra dạng chuẩn

Tạo class echoClient

```

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

```

```

public class echoClient {
    final static int defaultPort=7;
    public static void main(String[] args) {
        Socket ClientSocket;
        String hostName;
        int portNumber;
        BufferedReader theInput;
        PrintWriter theOutput;
        BufferedReader userInput;
        String inputString;
        switch(args.length) {
            case 1: hostName = args[0];
                      portNumber = defaultPort;
                      break;
            case 2: hostName = args[0];
                      portNumber = new Integer(args[1]).intValue();
                      break;
            default:
                      hostName = "localhost";
                      portNumber = defaultPort;
        }
        try {
            System.out.println("Client side Echo utility");
            ClientSocket = new Socket(hostName,portNumber);
            theInput = new BufferedReader(new
                InputStreamReader(ClientSocket.getInputStream()));
            theOutput = new
                PrintWriter(ClientSocket.getOutputStream());
            userInput = new BufferedReader(new
                InputStreamReader(System.in));
            System.out.println("Enter your text or put only '.' to quit.");
            while (true) {
                inputString = userInput.readLine();
                if (inputString.equals("."))
                    break;
                theOutput.println(inputString);
                System.out.println(theInput.readLine());
            }
        }
        catch (UnknownHostException e) {

```

```

        InterruptedException(" Unknown host error");
    }
    catch (ConnectException e) {
        System.out.println(" Service unavailable on port "
+ portNumber + " of host " + hostName);
    }
    catch (IOException e) {
        InterruptedException(" Communication error occurred\r\n" + e);
    }
}
}

```

3.2. Viết các ứng dụng kết nối phía Server sử dụng lớp ServerSocket

Tạo đối tượng socket phía Server và truyền thông với giao thức TCP. Sau khi tạo ra, Server đặt ở trạng thái lắng nghe chờ tín hiệu kết nối gửi tới từ Client.

Các phương thức tạo:

- **public ServerSocket(int port):** Tạo đối tượng ServerSocket với số cổng xác định bởi tham số port, đáp ứng cực đại tới 50 kết nối đồng thời.
- **public ServerSocket(int port, int queueLength):** Chỉ ra số kết nối cực đại mà socket có thể đáp ứng đồng thời bởi tham số queueLength.
- **public ServerSocket():** Tạo đối tượng ServerSocket nhưng không gắn kết thực sự socket với một cổng cụ thể nào, như vậy không chấp nhận bất cứ kết nối nào gửi tới. Sử dụng phương thức bind() để gắn địa chỉ.

Ví dụ:

```

ServerSocket ss = new ServerSocket( );
SocketAddress http = new InetSocketAddress(80);
ss.bind (http) ;

```

Các phương thức:

accept(): Đặt đối tượng ServerSocket ở trạng thái “nghe” tại số cổng xác định, chờ tín hiệu kết nối gửi đến từ Client.

close(): Đóng socket, giải phóng tài nguyên

Bài 4.

Viết ứng dụng phía máy chủ lắng nghe trên cổng TCP cho trước và gửi ngày theo định dạng chuỗi tới Client. Đối số chương trình là số cổng mặc định giá trị cổng là 13.

Hướng dẫn:

1. Tạo đối số
2. Tạo Server socket
3. Chờ yêu cầu từ Client
4. Tạo kết nối tới Client
5. Tạo luồng dữ liệu
6. Gửi chuỗi thời gian tới Client

7. Đóng kết nối mạng tới Client

Tạo class daytimeServer

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class daytimeServer {
    public final static int daytimePort = 13;
    public static void main(String[] args) {
        ServerSocket theServerSocket;
        Socket theConnectionSocket;
        PrintWriter out;
        try {
            theServerSocket = new ServerSocket(daytimePort);
            System.out.println("TimeServer ready at port "+daytimePort);
            try {
                while (true) {
                    theConnectionSocket = theServerSocket.accept();
                    System.out.println("Request arrived!");
                    out = new PrintWriter(theConnectionSocket.getOutputStream(), true);
                    out.println(new Date());
                    theConnectionSocket.close();
                }
            } catch (IOException e) {
                theServerSocket.close();
                InterruptedException(e);
            }
        } catch (IOException e) {
            InterruptedException(e);
        }
    }
}
```

Bài 5.

Viết ứng dụng phía máy chủ lắng nghe trên cổng TCP cho trước. Đọc chuỗi từ Client và gửi lại chuỗi cho Client như dịch vụ echo chuẩn. Đôi số chương trình là cổng số 7.

Hướng dẫn:

1. Khởi tạo đối số
2. Tạo Server socket
3. Chờ yêu cầu từ Client
4. Tạo kết nối tới Client

5. Tạo luồng dữ liệu
6. Đọc dữ liệu từ Client sau đó gửi dữ liệu trả lại Client
7. Đóng kết nối mạng
8. Quay lại bước 3

Tạo class echoServer

```

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

public class echoServer {
    public final static int echoPort = 7;
    public static void main(String[] args) {
        ServerSocket theServerSocket;
        Socket theConnectionSocket;
        BufferedReader in;
        PrintWriter out;
        try{
            theServerSocket = new ServerSocket(echoPort);
            System.out.println("EchoServer ready at port "+ echoPort);
            while (true) {
                theConnectionSocket = theServerSocket.accept();
                try {
                    System.out.println("Request arrived!");
                    in = new BufferedReader(new
InputStreamReader(theConnectionSocket.getInputStream()));
                    out = new PrintWriter(theConnectionSocket.getOutputStream(),true);
                    while(true) {
                        String readText = in.readLine();
                        out.println(readText);
                    }
                }
                catch (IOException e) {
                    theConnectionSocket.close();
                }
            }
        }
        catch (IOException e) {
            InterruptedException(e);
        }
    }
}

```

4. Lập trình Client- Server ở chế độ hướng kết nối

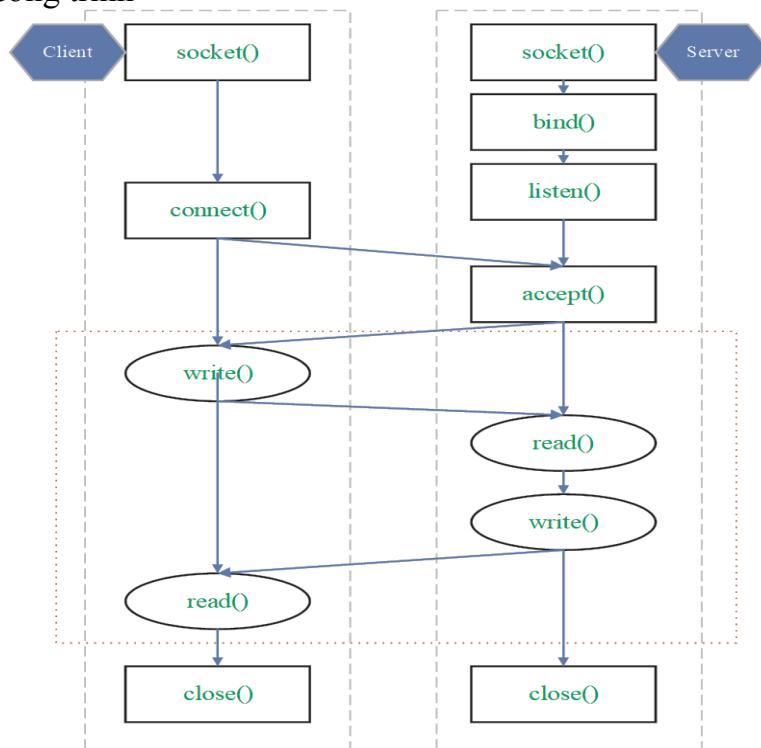
Để Client và Server có thể truyền thông được với nhau theo cơ chế hướng kết nối, mỗi phía phải thực hiện các thao tác sau:

a) Phía Server TCP

1. Tạo ServerSocket
2. Gọi thực thi phương thức accept() để chấp nhận thiết lập kết nối với Client => nhận được Socket giao tiếp với Client.
3. Lấy InputStream và OutputStream để nhận và gửi dữ liệu với Client.
4. Gửi và nhận dữ liệu với Client, sử dụng các phương thức read() và write() của các lớp InputStream và OutputStream.
5. Đóng Socket và ServerSocket
6. Kết thúc chương trình

b) Phía Client TCP

1. Tạo Socket kết nối đến Server
2. Lấy InputStream và OutputStream để nhận và gửi dữ liệu với Server.
3. Gửi và nhận dữ liệu với Server, sử dụng các phương thức read() và write() của các lớp đối tượng InputStream và OutputStream.
4. Đóng Socket
5. Kết thúc chương trình



Hình 9. Mô hình Client-Server ché độ hướng kết nối

Lưu ý:

- Chương trình Server luôn chạy trước chương trình Client.
- Một chương trình Server có thể phục vụ nhiều Client đồng thời

Bài 6. Viết chương trình:

- Client: Nhập vào từ bàn phím 2 số nguyên (a,b). Client chờ nhận kết quả từ Server để

in ra màn hình

- Server: Nhận 2 số nguyên mà Client vừa gửi, tính tổng và gửi kết quả cho Client

Hướng dẫn:

Tạo class Socket_tong2so_Client

```
import java.net.*  
  
public class Socket_tong2so_Client {  
    public static void main(String []args) throws IOException  
    {  
        System.out.println("Client dang ket noi voi Server... ");  
        String a,b,tong;  
        //tạo socket để kết nối tới Server, Localhost: Server mặc định  
        Socket ClientSocket = new Socket("Localhost", 1234);  
        //thông báo đã kết nối thành công  
        System.out.println("Da ket noi voi Server! ");  
        //tạo luồng nhập dữ liệu từ bàn phím  
        DataInputStream inFromUser = new DataInputStream(System.in);  
        //tạo luồng nhận dữ liệu từ Server  
        DataInputStream inFromServer =  
        DataInputStream(ClientSocket.getInputStream());  
        //tạo luồng gửi dữ liệu lên Server  
        DataOutputStream outToServer = new  
        DataOutputStream(ClientSocket.getOutputStream());  
        // nhập dữ liệu từ bàn phím  
        try {  
            System.out.println("\n Nhập a :");  
            a=inFromUser.readLine();  
            System.out.println("\n Nhập b :");  
            b=inFromUser.readLine();  
            // gửi lên Server  
            outToServer.writeBytes(a+'\n');  
            outToServer.writeBytes(b+'\n');  
        }catch(UnknownHostException e)  
        {  
            InterruptedException("Server Not Found");  
            System.exit(1);  
        }catch(IOException e)  
        {  
            InterruptedException("Cannot make a connection");  
        }  
        //nhận về từ Server
```

```

        tong=inFromServer.readLine();
        //in ra man hinh
        System.out.println("\n Ket qua :" +tong);
        //dong luong gui du lieu len Server
        outToServer.close();
        //dong luong nhan du lieu tu Server
        inFromServer.close();
        ClientSocket.close();
    }
}

```

Tạo class Socket_tong2so_Server

```

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

public class Socket_tong2so_Server {
    public static void main(String []args) throws IOException
    {
        System.out.println("Server dang khai dong... ");
        String so1, so2, so3;
        int tong;
        // tao Server socket
        ServerSocket Server = new ServerSocket(1234);
        System.out.println("Server da san sang! ");
        //tao 1 socket do ket noi tu Client toi Server
        Socket connectionSocket = Server.accept();
        //tao luong nhan du lieu tu Client
        DataInputStreaminFromClient=new
DataInputStream(connectionSocket.getInputStream());
        // tao luong gui du lieu toi Client
        DataOutputStreamoutToClient=new
DataOutputStream(connectionSocket.getOutputStream());
        // truyen du lieu tu Client vao 2 bien so1 va so2
        so1 = inFromClient.readLine();
        so2 = inFromClient.readLine();
        //ep so1 va so2 tu kieu String sang kieu Integer
        int a = Integer.parseInt(so1);
        int b = Integer.parseInt(so2);
        //tinh tong a + b
        tong = a + b;
        //ep tong sang kieu String
        so3 = String.valueOf(tong);
    }
}

```

```

//gui so3 ve Client
outToClient.writeBytes(so3+'\n');
//dong luong nhan du lieu tu Client
inFromClient.close();
//dong luong gui du lieu ve Client
outToClient.close();
//dong Server socket
Server.close();
}
}

```

Bài 7.

- Client: Nhập vào từ bàn phím một số nguyên a. Client chờ nhận kết quả từ Server để in ra màn hình
- Server: Nhận số nguyên Client vừa gửi, kiểm tra tính chẵn (lẻ, nguyên tố, hoàn hảo), gửi kết quả cho Client

Hướng dẫn:

Tạo file Server

1. Tạo Server socket
2. Tạo 1 socket kết nối từ Client tới Server
3. Tạo luồng nhận dữ liệu từ Client
4. Tạo luồng gửi dữ liệu tới Client
5. Truyền dữ liệu từ Client vào biến so
6. Ép so từ kiểu **String** sang kiểu **Integer**
7. Áp dụng phương thức kiểm tra nguyên tố, hoàn hảo, chẵn lẻ xử lý cho so.

Trong file Client

1. Tạo socket để kết nối tới Server, Locallhost: Server mặc định
2. Tạo luồng nhập dữ liệu từ bàn phím
3. Tạo luồng nhận dữ liệu từ Server
4. Tạo luồng gửi dữ liệu lên Server
5. Nhập dữ liệu từ bàn phím
6. Gửi lên Server
7. Nhận dữ liệu từ Server
8. Đóng luồng dữ liệu gửi lên Server
9. Đóng luồng nhận dữ liệu từ Server
10. Đóng socket Client

LAB 9. LẬP TRÌNH SERVER PHỤC VỤ NHIỀU CLIENT [4,6,10]

A. MỤC TIÊU

Trang bị cho sinh viên kỹ thuật lập trình các ứng dụng một Server có khả năng đáp ứng nhiều Client.

Kết hợp lập trình giao diện với java swing, xử lý đa luồng, lập trình Client-Server theo cơ chế TCP viết các ứng dụng đa luồng trên mạng.

B. NỘI DUNG

- Viết các ứng dụng Client-Server kết hợp sử dụng luồng.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Viết được chương trình kết nối TCP thực hiện trao đổi dữ liệu giữa Client và Server.
- Xây dựng ứng dụng chat LAN.

D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

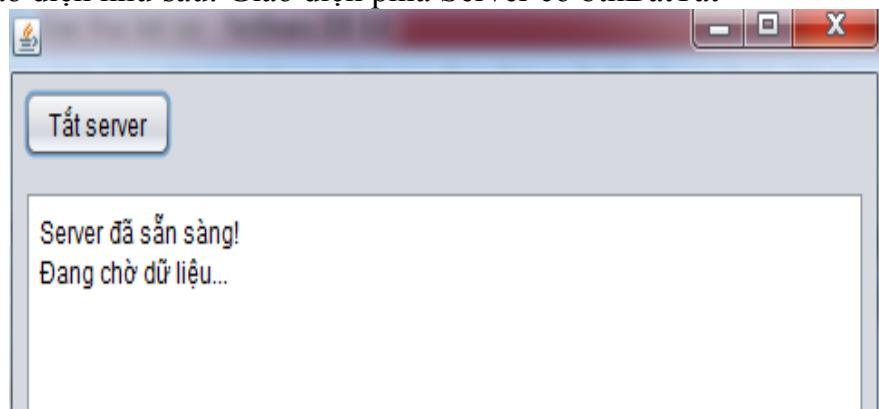
- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8.

E. HƯỚNG DẪN

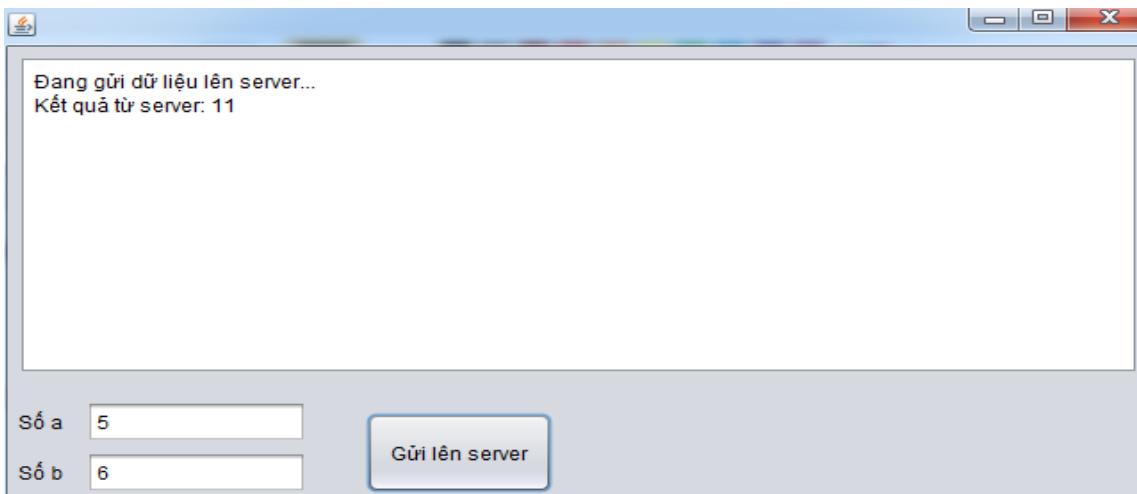
Để cài đặt chương trình Server TCP phục vụ nhiều Client đồng thời, trong lập trình mạng sử dụng kỹ thuật phổ biến đó là: Xây dựng chương trình đa luồng (đa tiến trình). Chương trình Server kiểu này sẽ sinh ra một luồng mới mỗi khi có một Client gửi yêu cầu tới đòi phục vụ.

Chạy chương trình Server trên một máy và Client trên máy khác nếu hệ thống có mạng. Nếu thử nghiệm trên một máy đơn ta có thể mô phỏng mô hình khách/chủ bằng cách mở hai cửa sổ dòng lệnh DOS.

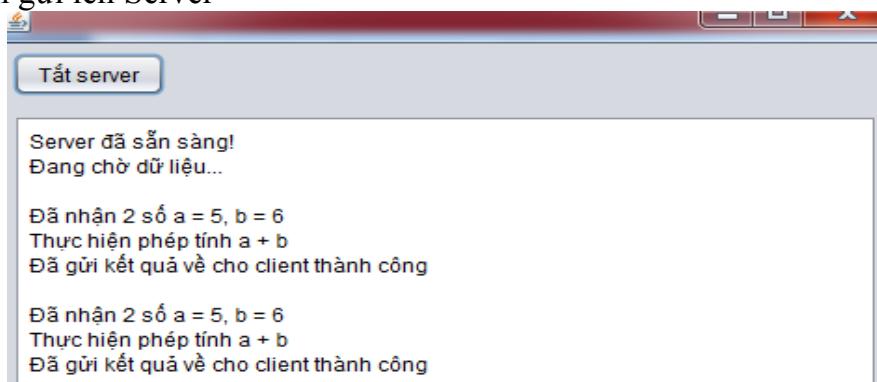
Bài 1. Viết chương trình minh họa giao tiếp Client-Server sử dụng TCP, xử lý tính tổng hai số có giao diện như sau: Giao diện phía Server có btnBatTat



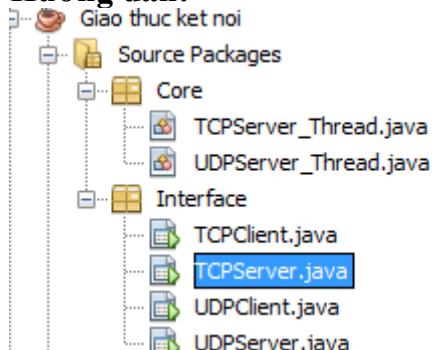
Phía Client: có btnSendToServer



Xử lý sự kiện gửi lên Server



Hướng dẫn:



Bước 1: Tạo file TCPServer

```

TCPserver_Thread mThreadServer;
boolean mTurnOn = false;
private void btnBatTatActionPerformed(java.awt.event.ActionEvent evt) {
    if(mTurnOn==false)
    {
        mThreadServer= new TCPserver_Thread(txStatus);
        mThreadServer.start();
        mTurnOn = true;
        btnBatTat.setText("Tắt Server");
    }
    else

```

```

    {
        mThreadServer.StopServer();
        mTurnOn = false;
        btnBatTat.setText("Bật Server");
        txaStatus.append("Đã tắt Server\n\n");
    }
}

```

Bước 2: File TCPClient

```

private void btnSendToServerActionPerformed(java.awt.event.ActionEvent evt) {
    txaStatus.append("Đang gửi dữ liệu lên Server...\n");
    try {
        Socket ClientSocket = new Socket("localhost", 1234);
        //tao luong nhan du lieu tu Server
        DataInputStream inFromServer = new
DataInputStream(ClientSocket.getInputStream());
        //tao luong gui du lieu len Server
        DataOutputStream outToServer = new
DataOutputStream(ClientSocket.getOutputStream());
        // nhap du lieu tu ban phim gui len Server
        outToServer.writeBytes(txtSoA.getText()+'\n');
        outToServer.writeBytes(txtSoB.getText()+'\n');
        txaStatus.append("Kết quả từ Server:"+inFromServer.readLine() + "\n\n");
        //dong luong gui du lieu len Server
        outToServer.close();
        //dong luong nhan du lieu tu Server
        inFromServer.close();
        //dong socket Client
        ClientSocket.close();
    }catch(UnknownHostException e)
    {
        txaStatus.append("Server Not Found\n\n");
    }catch(IOException e)
    {
        txaStatus.append("Cannot make a connection\n\n");
    }
}

private void SoABKeyTyped(java.awt.event.KeyEvent evt) {
    JTextField txt = (JTextField)evt.getSource();
}

```

```

String after = txt.getText() + evt.getKeyChar();
try
{
    Integer.parseInt(after);
}catch(NumberFormatException ex)
{
    evt.consume();
}
}

```

Bước 3: Tạo file TCPServer_Thread: Lớp này khởi động Server trong 1 thread khác để tránh giao diện bị treo khi Server đang đợi kết nối.

```

public class TCPServer_Thread extends Thread
{
    ServerSocket mServer;
    JTextArea mTxaStatus; //JTextArea để lưu status của Server
    public TCPServer_Thread(JTextArea txaStatus)
    {
        mTxaStatus = txaStatus;
    }
    @Override
    public void run()
    {
        try
        {
            String so1,so2,so3;
            int tong;
            mServer = new ServerSocket(1234);
mTxaStatus.append("Server đã sẵn sàng!\nĐang chờ dữ liệu...\n\n");
            while(true)
            {
                Socket connectionSocket = mServer.accept();
                DataInputStream inFromClient = new
DataInputStream(connectionSocket.getInputStream());
                DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
                so1 = inFromClient.readLine();
                so2 = inFromClient.readLine();
                mTxaStatus.append("Đã nhận 2 số a = " + so1 +", b = "+ so2 +"\n");
                int a = Integer.parseInt(so1);
                int b = Integer.parseInt(so2);
            }
        }
    }
}

```

```

        tong = a + b;
        so3 = String.valueOf(tong);
        mTxaStatus.append("Thực hiện phép tính a + b\n");
        outToClient.writeBytes(so3+'\n');
        mTxaStatus.append("Đã gửi kết quả về cho Client thành công\n\n");
    }
}
catch(Exception ex)
{
    Logger.getLogger(TCPServer_Thread.class.getName()).log(Level.SEVERE,
null, ex);
    mTxaStatus.append("Server đã xảy ra lỗi\n");
}
}

public void StopServer()
{
    super.stop();
    try {
        mServer.close();
    } catch (IOException ex) {
        Logger.getLogger(TCPServer_Thread.class.getName()).log(Level.SEVERE,
null, ex);
    }
}
}

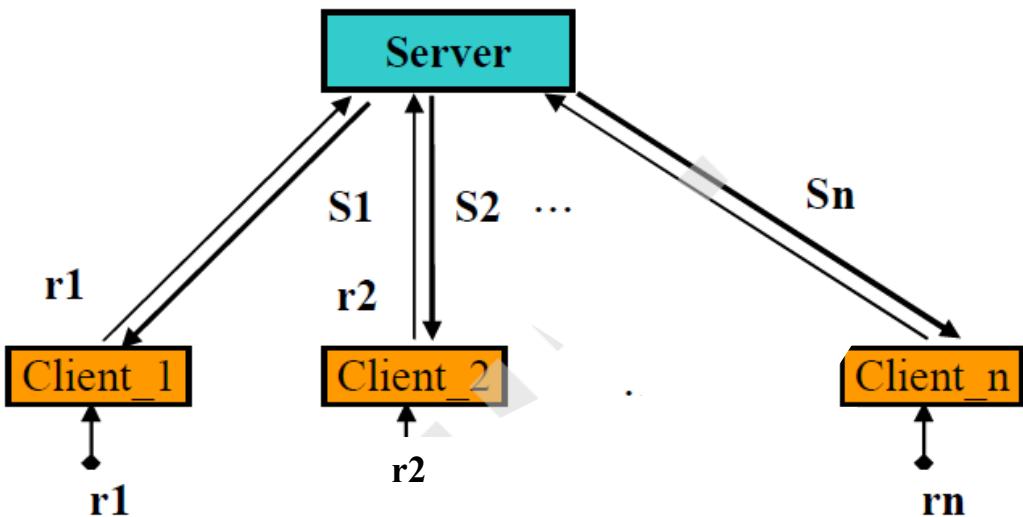
```

Bài 2.

Viết chương trình Server phục vụ nhiều Client đồng thời sử dụng giao thức truyền thông TCP. Chương trình cho phép nhận bán kính đường tròn gửi đến từ các Client, tính diện tích hình tròn, hiển thị tên, địa chỉ IP, số cổng, bán kính r, diện tích của Client tương ứng. Sau đó trả kết quả về cho Client.

Hướng dẫn:

Để viết chương trình này sử dụng kỹ thuật đa luồng trong Java. Mỗi khi một chương trình Client gửi yêu cầu kết nối đến Server, Server sẽ sinh ra một luồng mới để phục vụ kết nối đó. Sau khi phục vụ xong kết nối nào thì luồng đó được giải phóng. Mô hình xây dựng chương trình thể hiện như sau:



Bước 1: Tạo class areaClient

Chương trình Client thực hiện các công việc sau:

1. Gửi kết nối tới Server
2. Nhập bán kính r từ bàn phím
3. Gửi bán kính tới Server
4. Nhận kết quả trả về và hiển thị
5. Kết thúc chương trình

```

class areaClient{
public static void main(String[] args)
{
    Socket cl=null;
    BufferedReader inp=null;//luong nhap
    PrintWriter outp=null;//luong xuat
    BufferedReader key=null;//luong nhap tu ban phim
    String ipServer= "127.0.0.1";//Chuoi dia chi Server
    int portServer=3456; //dia chi cong Server
    String r; //Tao socket va ket noi toi Server
    try{
        cl=new Socket(ipServer,portServer);
        //tao luong nha/xuatp kieu ky tu cho socket
        inp=new BufferedReader(new InputStreamReader(cl.getInputStream()));
        outp=new PrintWriter(cl.getOutputStream(),true);
        //tao luong nhap tu ban phim
        key=new BufferedReader(new InputStreamReader(System.in));
        //Nhap ban kinh r tu ban phim
        System.out.print("r=");
        r=key.readLine().trim();
        //gui r toi Server
    }
}

```

```

        outp.println(r);
        //Nhan dien tich tra ve tu Server va hien thi
        System.out.println("Area:"+inp.readLine());
        //ket thuc chuong trinh
        if(inp!=null)
            inp.close();
        if(key!=null)
            key.close();
        if(outp!=null)
            outp.close();
        if(cl!=null)
            cl.close();
    }
    catch(IOException e)
    {
        System.out.println(e);
    }
}
}

```

Bước 2: Tạo chương trình phía Server

Chương trình Server phục vụ nhiều Client thực hiện các công việc sau:

1. Khởi tạo đối tượng ServerSocket và nghe tại số cổng 3456.
2. Thực hiện lặp lại các công việc sau:
3. Nhận kết nối mới, tạo socket mới
4. Phát sinh một luồng mới và nhận socket
5. Nhận bán kính gửi tới từ Client
6. Tính diện tích
7. Hiển thị số thứ tự luồng, tên, địa chỉ IP, số cổng, bán kính r, diện tích của Client
8. Gửi diện tích về cho Client
9. Kết thúc luồng

Tạo class NewThread

```

import java.io.*;
import java.net.*;
//Khai báo lớp NewThread cho phép tạo ra luồng mới
class NewThread extends Thread
{
    private int count;
    private Socket cl=null;
    private BufferedReader inp=null;//luong nhap
    private PrintWriter outp=null;//luong xuat
}

```

```

NewThread(Socket cl, int count)
{
    super();//Truy xuất cấu tử lớp Thread
    this.cl=cl;
    this.count=count;
    start();
}
//cai dat phuong thuc run-Luong moi
public void run()
{
    try{
        //tao luong nhap /xuat cho socket cl
        inp=new BufferedReader(new InputStreamReader(cl.getInputStream()));
        outp=new PrintWriter(cl.getOutputStream(),true);
        //Doc ban kinh gui toi tu Client
        double r=Double.parseDouble(inp.readLine().trim());
        // lay dia chi Client
        InetAddress addrClient=cl.getInetAddress();
        //lay so cong phia Client
        int portClient=cl.getPort();
        //Tinh dien tich
        double area=3.14*r*r;
        //Hien thi
        System.out.println("Luong thu:"+count, Client:"+addrClient.getHostName()+
        " , ip:"+addrClient.getHostAddress()+" , port:"+portClient+" ,
        r="+r+", area:"+area);
        //Gui dien tich ve cho Client tuong ung
        outp.println(area);
        //ket thuc luong
        inp.close();
        outp.close();
        cl.close();
    }
    catch(IOException e)
    {
        System.out.println(e);
    }
}
}

```

Bước 3: Tạo class AreaThreadServer

```

class AreaThreadServer{
    public static void main(String[] args)
    {
        //Khai bao bien
        int count;
        ServerSocket svr=null;
        Socket cl=null;
        int portServer=3456;
        try{
            svr=new ServerSocket(portServer);
            count=0;
            while(true){
                cl=svr.accept();
                new NewThread(cl, count);
                count++;
            }
        }
        catch(IOException e)
        {
            System.out.println(e);
        }
    }
}

```

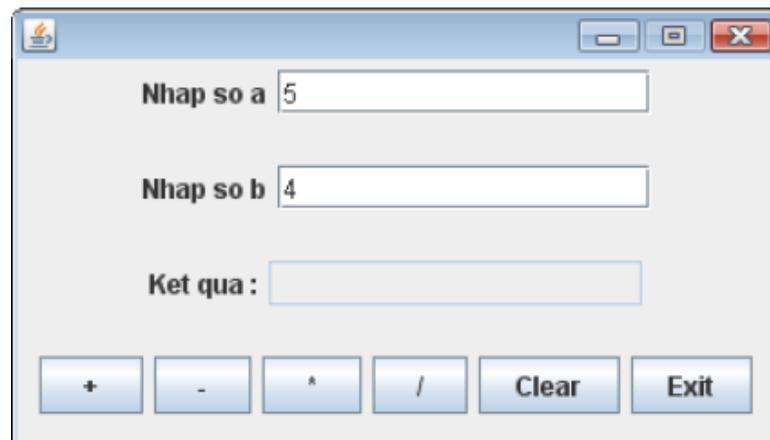
Bài 3.

Viết 1 chương trình Calculator_server nhận 1 biểu thức gồm 2 chữ số và 1 phép toán sau đó thực thi biểu thức này và gửi kết quả lại cho client.

Sửa chữa chương trình cho phép nhiều client kết nối cùng lúc.

Phía client, viết giao diện gồm 2 JTextField cho việc nhập số, 1 JLabel cho việc xuất kết quả. Các nút Cộng, trừ, nhân, chia, clear và thoát.

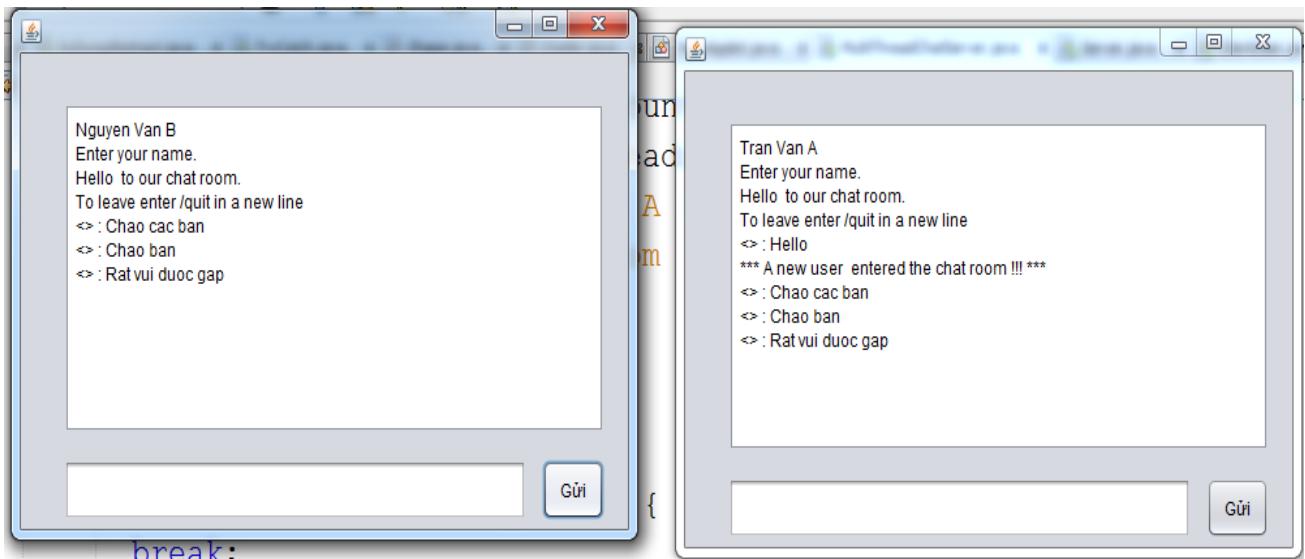
Giao diện client như sau:



Hướng dẫn: Sinh viên thực hiện tương tự bài 2.

Bài 4. Viết ứng dụng chat đa người dùng sử dụng TCP và luồng.

Yêu cầu: Giao diện các Client tham gia chat như sau:



Hướng dẫn: Kết hợp lập trình java swing và xử lý đa luồng trong java để giải quyết.

Bước 1: Tạo class MultiThreadChatServer sử dụng cổng 2222, đáp ứng đồng thời 10 Client. (file thực thi)

```
public class MultiThreadChatServer extends javax.swing.JFrame{  
    private static ServerSocket ServerSocket = null;  
    private static Socket ClientSocket = null;  
    private static final int maxClientsCount = 10;  
    private static final ClientThread[] threads = new  
    ClientThread[maxClientsCount];  
    public static void main(String args[]) {  
        int portNumber = 2222;  
        if (args.length < 1) {  
            System.out.println("Usage: java MultiThreadChatServer <portNumber>\n"  
                + "Now using port number=" + portNumber);  
        } else {  
            portNumber = Integer.valueOf(args[0]).intValue();  
        }  
        try {  
            ServerSocket = new ServerSocket(portNumber);  
        } catch (IOException e) {  
            System.out.println(e);  
        }  
        while (true) {
```

```
try {
    ClientSocket = ServerSocket.accept();
    int i = 0;
    for (i = 0; i < maxClientsCount; i++) {
        if (threads[i] == null) {
            threads[i] = new ClientThread(ClientSocket, threads)).start();
            break;
        }
    }
    if (i == maxClientsCount) {
        DataOutputStream os = new
DataOutputStream(ClientSocket.getOutputStream());
        os.writeUTF("Server too busy. Try later.");
        os.close();
        ClientSocket.close();
    }
} catch (IOException e) {
    System.out.println(e);
}
}
```

Bước 2: Tạo **class** ClientThread. Mỗi Client khi tham gia vào nhóm chat tương ứng với 1 thread.

```
class ClientThread extends Thread {  
    private DataInputStream is = null;  
    private DataOutputStream os = null;  
    private Socket ClientSocket = null;  
    private final ClientThread[] threads;  
    private int maxClientsCount;  
    public ClientThread(Socket ClientSocket, ClientThread[] threads) {  
        this.ClientSocket = ClientSocket;  
        this.threads = threads;  
        maxClientsCount = threads.length;  
    }  
    public void run() {  
        int maxClientsCount = this.maxClientsCount;  
        ClientThread[] threads = this.threads;  
        try {
```

```

is = new DataInputStream(ClientSocket.getInputStream());
os = new DataOutputStream(ClientSocket.getOutputStream());
os.writeUTF("Enter your name.");
String name = is.readUTF().trim();
os.writeUTF("Hello " + name
            + " to our chat room.\nTo leave enter /quit in a new line");
for (int i = 0; i < maxClientsCount; i++) {
    if (threads[i] != null && threads[i] != this) {
        threads[i].os.writeUTF("*** A new user " + name
                               + " entered the chat room !!! ***");
    }
}
while (true) {
    String line = is.readUTF();
    if (line.startsWith("/quit")) {
        break;
    }
    for (int i = 0; i < maxClientsCount; i++) {
        if (threads[i] != null) {
            System.out.println(line);
            threads[i].os.writeUTF("<" + name + "> : " + line);
        }
    }
}
for (int i = 0; i < maxClientsCount; i++) {
    if (threads[i] != null && threads[i] != this) {
        threads[i].os.writeUTF("*** The user " + name
                               + " is leaving the chat room !!! ***");
    }
}
os.writeUTF("*** Bye " + name + " ***");
for (int i = 0; i < maxClientsCount; i++) {
    if (threads[i] == this) {
        threads[i] = null;
    }
}
is.close();
os.close();
ClientSocket.close();
} catch (IOException e) {

```

```
    }
}
}
```

Bước 3: Tạo class Server (chạy trên Server)

```
public class Server extends javax.swing.JFrame {
    private static ServerSocket ServerSocket = null;
    private static Socket ClientSocket = null;
    private static final int maxClientsCount = 10;
    private static final ClientThread[] threads = new
ClientThread[maxClientsCount];
    public Server() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    private void initComponents() {
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE));
        layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
        pack();
    }
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
        javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
```

```

        logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Server.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Server().setVisible(true);
            int portNumber = 2222;
            if (args.length < 1) {
                System.out.println("Usage: java MultiThreadChatServer
<portNumber>\n" + "Now using port number=" + portNumber);
            } else {
                portNumber = Integer.valueOf(args[0]).intValue();
            }
            try {
                ServerSocket = new ServerSocket(portNumber);
            } catch (IOException e) {
                System.out.println(e);
            }
            while (true) {
                try {
                    ClientSocket = ServerSocket.accept();
                    for (i = 0; i < maxClientsCount; i++) {
                        if (threads[i] == null) {
                            (threads[i] = new ClientThread(ClientSocket, threads)).start();
                            break;
                        }
                    }
                    if (i == maxClientsCount) {
DataOutputStream os = new
DataOutputStream(ClientSocket.getOutputStream());
os.writeUTF("Server too busy. Try later.");
os.close();

```

```
        ClientSocket.close();
    }
} catch (IOException e) {
    System.out.println(e);
}
}
});
```

Bước 4: Tạo class ClientUser minh họa giao diện người dùng (chạy trên Client)

```
public class ClientUser extends javax.swing.JFrame implements Runnable{  
    private static Socket ClientSocket = null;  
    private static DataOutputStream os = null;  
    private static DataInputStream is = null;  
    private static BufferedReader inputLine = null;  
  
    public ClientUser() {  
        initComponents();  
    }  
  
    private void msg_sendActionPerformed(java.awt.event.ActionEvent evt) {  
        try{  
            os.writeUTF(msg_text.getText().trim());  
            msg_text.setText("");  
        }catch(Exception e){}  
    }  
  
    public static void main(String args[]) {  
        try {  
            for (javax.swing.UIManager.LookAndFeelInfo info :  
                javax.swing.UIManager.getInstalledLookAndFeels()) {  
                if ("Nimbus".equals(info.getName())) {  
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());  
                    break;  
                }  
            }  
        } catch (ClassNotFoundException ex) {  
            java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.  
logging.Level.SEVERE, null, ex);  
        } catch (InstantiationException ex) {  
            java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
```

```

        logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(ClientUser.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            int portNumber = 2222;
            String host = "localhost";
            try {
                ClientSocket = new Socket(host, portNumber);
                inputLine = new BufferedReader(new
InputStreamReader(System.in));
                os = new DataOutputStream(ClientSocket.getOutputStream());
                is = new DataInputStream(ClientSocket.getInputStream());
            } catch (UnknownHostException e) {
                System.err.println("Don't know about host " + host);
            } catch (IOException e) {
                System.err.println("Couldn't get I/O for connection to the host "+ host);
            }
            if (ClientSocket != null && os != null && is != null) {
                ClientUser c = new ClientUser();
                c.setVisible(true);
                new Thread(c).start();
            }
        }
    });
}
private javax.swing.JScrollPane jScrollPane1;
private static javax.swing.JTextArea msg_area;
private javax.swing.JButton msg_send;
private javax.swing.JTextField msg_text;
@Override
public void run() {
    String responseLine;
    try {
        while (true) {

```

```

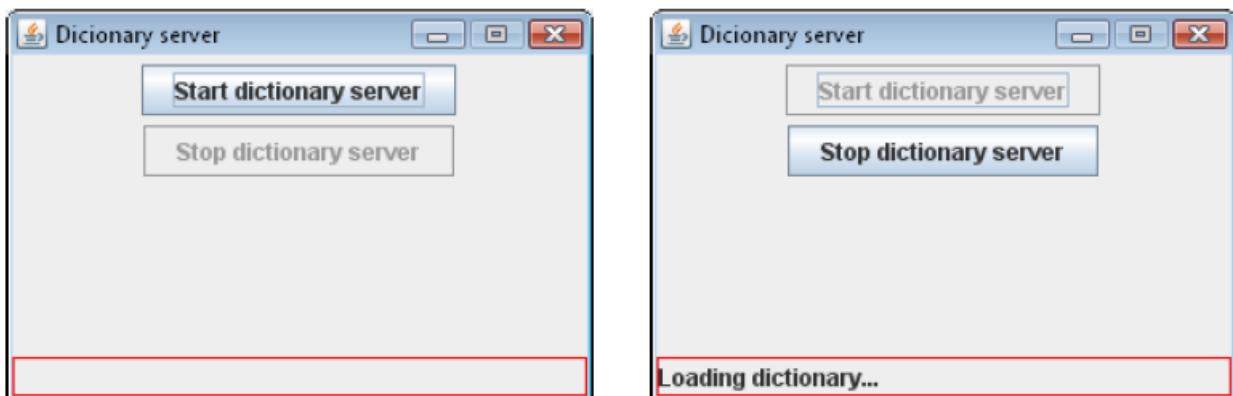
        responseLine = is.readUTF();
        System.out.println(responseLine);
        msg_area.setText(msg_area.getText().trim()+"\n"+responseLine);
    }
} catch (IOException e) {
    System.err.println("IOException: " + e);
}
}

```

Bài 5. Tổng hợp

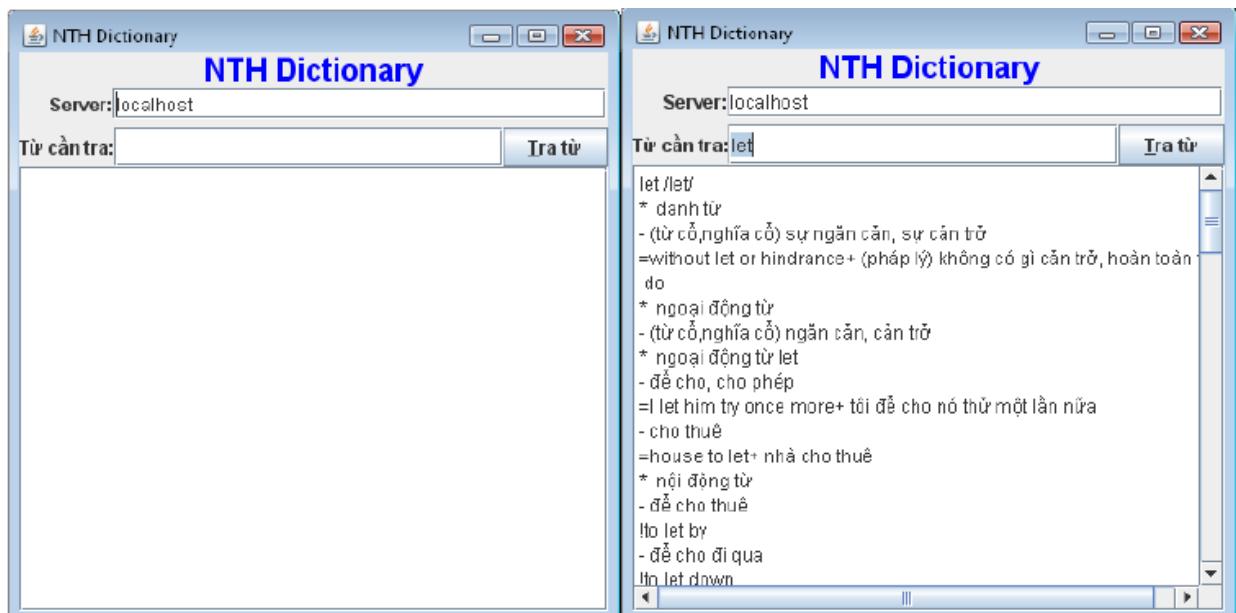
Viết chương trình từ điển cho phép tra từ qua mạng, việc tra từ này phải đảm bảo nhiều người có thể thực hiện cùng một lúc. Thiết kế chương trình gồm 2 phần: Phần server và client.

Phần server: Chạy trên server có giao diện như sau:



Khi người dùng start server từ điển, server này sẽ lắng nghe trên cổng 2520 và sẽ nhận đầu vào là từ cần tra sau đó thực hiện tra từ và trả kết quả về cho client hoặc là 1 đối tượng từ đã tra trong trường hợp tra cứu thành công, hoặc là null nếu từ không tồn tại.

Phần Client: được triển khai ở phía client, có giao diện như sau



Hướng dẫn:

Sử dụng đa luồng kết hợp với lập trình mạng sử dụng TCP, sinh viên triển khai tương tự bài 3.

LAB 10. LẬP TRÌNH CLIENT-SERVER SỬ DỤNG UDP [4, 6, 9,10]

A. MỤC TIÊU

Trang bị cho sinh viên kỹ thuật lập trình Client/Server sử dụng lớp DatagramSocket, DatagramPackage, để kết nối và trao đổi thông tin bằng giao thức UDP.

B. NỘI DUNG

Lập trình Client/Server sử dụng lớp DatagramSocket, DatagramPackage.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

Viết được các ứng dụng như: chat đa người dùng.

D. YÊU CẦU PHẦN CỨNG - PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8. MYSQL.

E. HƯỚNG DẪN

1. Giao thức hướng không kết nối – UDP: thực hiện truyền thông không cần kết nối (ảo)

Đặc điểm: Truyền thông theo kiểu điểm-đa điểm

- Quá trình truyền thông chỉ có một giai đoạn duy nhất là truyền dữ liệu, không có giai đoạn thiết lập cũng như huỷ bỏ kết nối.
- Dữ liệu truyền được tổ chức thành các tin gói tin độc lập, trong mỗi gói dữ liệu có chứa địa chỉ nhận.

2. Một số lớp dùng trong lập trình UDP

a. Lớp DatagramPacket cho phép tạo gói tin truyền thông với giao thức UDP.

Gói tin chứa 4 thành phần quan trọng: Địa chỉ, dữ liệu truyền thật sự, kích thước của gói tin và số hiệu cổng chứa trong gói tin.

Các phương thức tạo

public DatagramPacket(byte[] inBuffer, int length): Tạo gói tin nhận từ mạng.

Tham số:

- inBuffer: Bộ đệm nhập, chứa dữ liệu của gói tin nhận
- length: Kích cỡ của dữ liệu của gói tin nhận

public DatagramPacket(byte[] outBuffer, int length, InetAddress destination, int port):

Tạo gói tin gửi.

Tham số:

- outBuffer: Bộ đệm xuất chưa dữ liệu của gói tin gửi
- length: Kích cỡ dữ liệu của gói tin gửi (byte)
- destination: Địa chỉ nơi nhận gói tin.
- port: Số hiệu cổng đích, nơi nhận gói tin.

Các phương thức

- **public InetAddress getAddress():** Trả về đối tượng InetAddress của máy trạm từ xa chứa trong gói tin nhận.

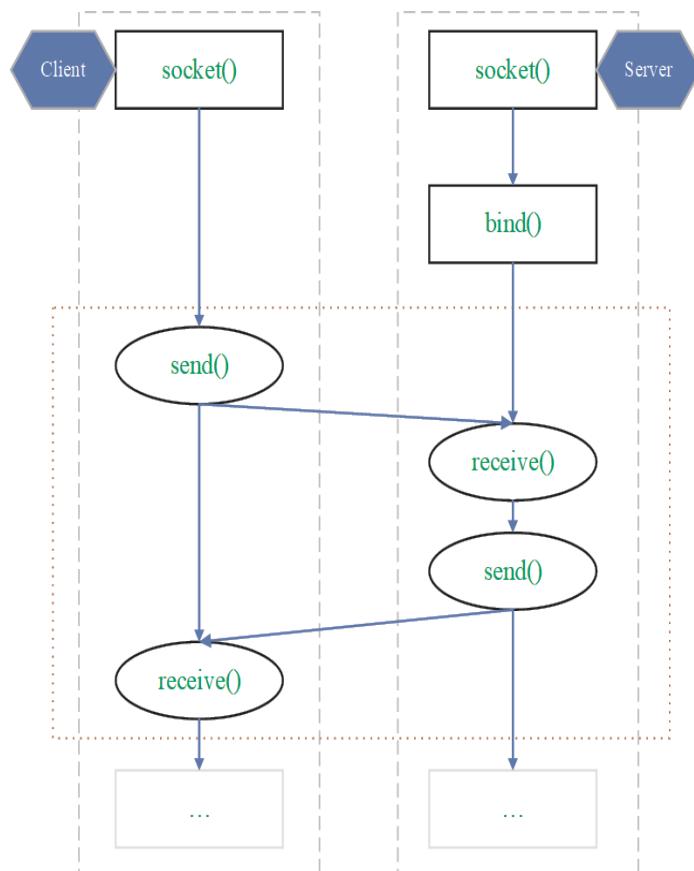
- **public int getPort()**: Trả về số hiệu cổng của máy trạm từ xa chứa trong gói tin.
- **public byte[] getData()** : Trả về dữ liệu chứa trong gói tin dưới dạng mảng byte.
- **public int getLength()**: Trả về kích cỡ của dữ liệu chứa trong gói tin

b. Lớp DatagramSocket

Tạo đối tượng socket truyền thông với giao thức UDP. Socket này cho phép gửi/nhận gói tin DatagramPacket.

Các phương thức tạo

- **public DatagramSocket()** Sử dụng phía Client tạo ra socket với số cổng nào đó.
- **public DatagramSocket(int port)** Sử dụng phía Server trong mô hình Client/Server, tạo socket với số cổng xác định và chờ nhận gói tin truyền tới.



Hình 10. Mô hình Client Server ở chế độ không kết nối

3. Các bước lập trình Client-Server sử dụng UDP

a) Phía Server

- 1.Tạo đối tượng DatagramSocket với số cổng xác định
- 2.Khai báo bộ đệm nhập /xuất inBuffer/outBuffer dạng mảng kiểu byte
- 3.Khai báo gói tin nhận gửi inData/outData là đối tượng DatagramPacket.
- 4.Thực hiện nhận/gửi gói tin với phương thức receive()/send()
- 5.Đóng socket, giải phóng tài nguyên, kết thúc chương trình, nếu không quay về bước 3.

b) Phía Client

1. Tạo đối tượng DatagramSocket với số cổng nào đó

2. Khai báo bộ đệm xuất/nhập outBuffer/inBuffer dạng mảng kiểu byte
3. Khai báo gói tin gửi/nhận outData/inData là đối tượng DatagramPacket.
4. Thực hiện gửi /nhận gói tin với phương thức send()/receive()
5. Đóng socket, giải phóng tài nguyên, kết thúc chương trình, nếu không quay về bước 3.

Bài 1.

Viết chương trình Client-Server minh họa sử dụng DatagramPacket. Phía Client gửi tin nhắn trong DatagramPacket đến máy chủ. Máy chủ lắng nghe trên cổng UDP. Nếu máy chủ nhận gói dữ liệu datagram, nó sẽ in thông điệp trong gói datagram để hiển thị.

Hướng dẫn

Bước 1: Tạo class Client

```

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

public class Client extends java.applet.Applet {
    String myHost;
    public void init() {
        myHost = getCodeBase().getHost();
    }
    public void sendMessage( String message ) {
        //phương thức gửi tin nhắn
        try {
            byte[] data = new byte[ message.length() ];
            message.getBytes(0, data.length, data, 0);
            InetAddress addr = InetAddress.getByName(myHost);
            DatagramPacket pack = new DatagramPacket(data, data.length, addr,1234);
            DatagramSocket ds = new DatagramSocket();
            ds.send(pack);
            ds.close();
        }
        catch (IOException e) {
            System.out.println( e );
        }
    }
    public void start() {
        sendMessage( "Bắt đầu!" );
    }
    public void stop() {
        sendMessage( "Kết thúc!" );
    }
}

```

Bước 2: Tạo class Server

```

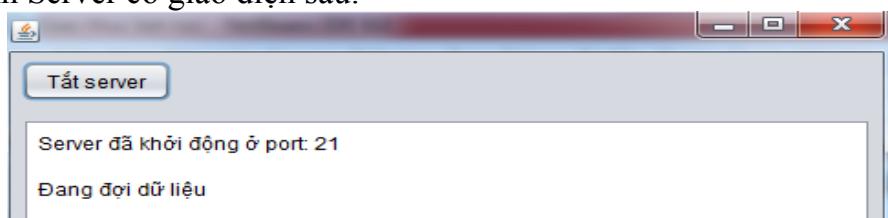
import java.net.*;
import java.io.*;
public class Server{
    public static void main( String argv[] ) throws IOException {
        DatagramSocket s = new DatagramSocket( 1234 );
        System.out.println(" Lắng nghe..");
        while (true) {
            DatagramPacket packet = new DatagramPacket( new byte[1024], 1024 );
            s.receive( packet );
            String message = new String(packet.getData(), 0, 0, packet.getLength());
            System.out.println( "Tín hiệu từ "+packet.getAddress().getHostName()+""
- "+message );
        }
    }
}

```

Bài 2. Viết chương trình Client-Server xử lý chuỗi, sử dụng giao thức UDP: Phía Client gửi một chuỗi lên Server, Server xử lý chuyển chữ hoa và đếm số từ trong xâu gửi lên.

Hướng dẫn:

Tạo màn hình Server có giao diện sau:



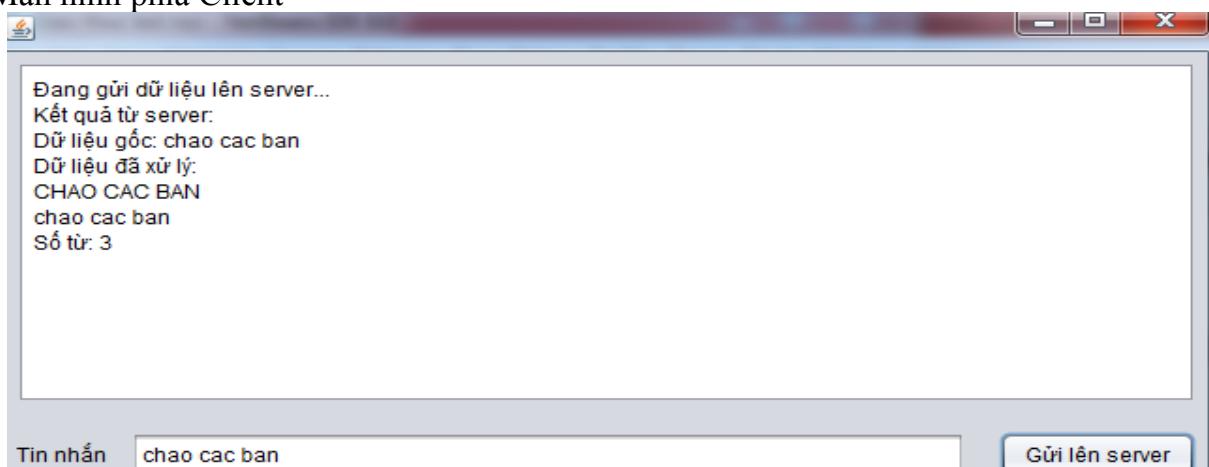
Bước 1: Tạo Class UDPServer kế thừa từ JFrame, có btnBatTat

UDPServer extends javax.swing.JFrame

Viết phương thức: **private void btnBatTatActionPerformed(java.awt.event.ActionEvent evt)**

//Xử lý sự kiện bật/tắt Server.

Màn hình phía Client



Bước 2: Tạo Class UDPClient có giao diện như trên viết phương thức xử lý sự kiện cho btnSendToServer

```
private void btnSendToServerActionPerformed(java.awt.event.ActionEvent evt) {  
    txaStatus.append("Đang gửi dữ liệu lên Server...\n");  
    int port = 21;  
    String hostName = "localhost";  
    byte[] outBuf = new byte[2048];  
    byte[] inBuf = new byte[2048];  
    try  
    {  
        //khai báo địa chỉ để kết nối  
        InetAddress address = InetAddress.getByName(hostName);  
        DatagramSocket socket = new DatagramSocket();  
        //chuyển đổi tin nhắn gửi đi sang byte  
        outBuf = txtTinNhan.getText().getBytes();  
        //Gửi dữ liệu đi  
        DatagramPacket outPacket = new DatagramPacket(outBuf, 0,  
outBuf.length, address, port);  
        socket.send(outPacket);  
        //Nhận dữ liệu về  
        DatagramPacket receivePacket = new DatagramPacket(inBuf, inBuf.length);  
        socket.receive(receivePacket);  
        String receiveMsg = new String(receivePacket.getData(),0,  
receivePacket.getLength());  
        txaStatus.append("Kết quả từ Server:\n" + receiveMsg);  
    } catch (IOException e)  
    {  
        e.printStackTrace();  
    }  
}
```

Bước 3: Tạo class UDPServer_Thread

```
public class UDPServer_Thread extends Thread  
{  
    DatagramSocket mServerSocket;  
    JTextArea mTxaStatus; //JTextArea để lưu status của Server  
    public UDPServer_Thread(JTextArea txaStatus)  
    {  
        mTxaStatus = txaStatus;  
    }
```

```

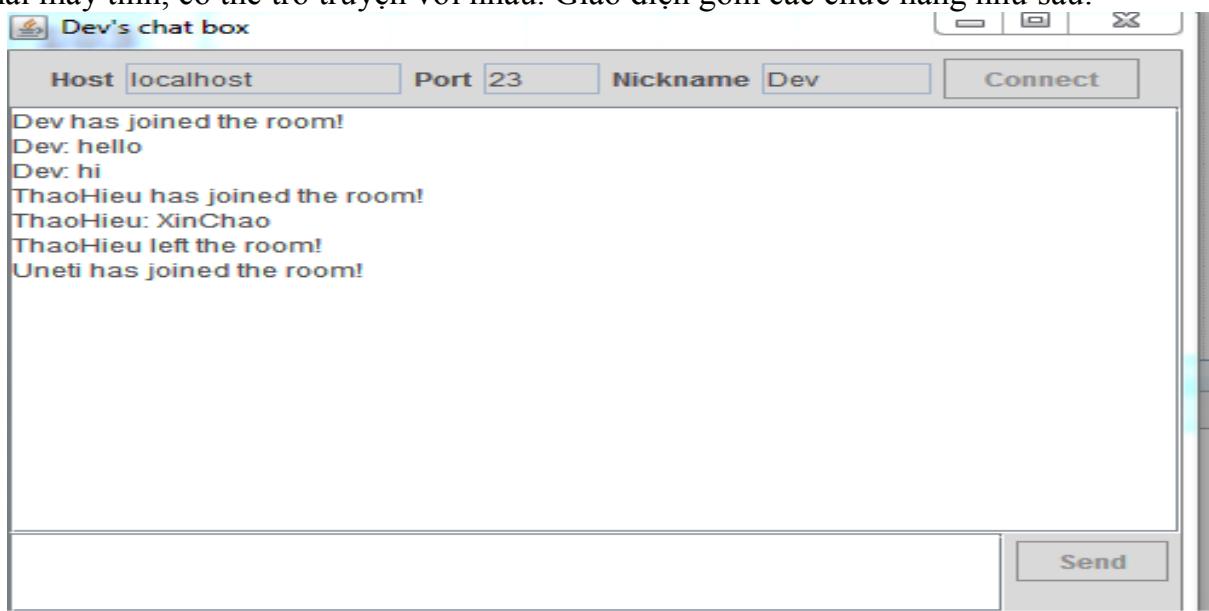
    @Override
    public void run()
    {
        int port = 21;
        try
        {
            mServerSocket = new DatagramSocket(port);
            mTxStatus.append("Server khởi động ở: "+port+"Đang đợi dữ liệu\n");
            byte[] buf = new byte[2048];
            DatagramPacket receivePacket = new DatagramPacket(buf, buf.length);
            while (true)
            {
                try
                {
                    mServerSocket.receive(receivePacket);
                    String ClientMsg = new String(receivePacket.getData(), 0,
                    receivePacket.getLength());
                    StringTokenizer st = new StringTokenizer(ClientMsg, " ");
                    //đếm số
                    từ dựa vào dấu space
                    int numWords = st.countTokens();
                    String returnMsg = "Dữ liệu gốc: " + ClientMsg + "\n"
                        + "Dữ liệu đã xử lý:\n" + ClientMsg.toUpperCase() + "\n"
                        + ClientMsg.toLowerCase() + "\n" + "Số từ: " + numWords + "\n\n";
                    mTxStatus.append(returnMsg);
                    DatagramPacket outPacket = new DatagramPacket(returnMsg.getBytes(),
                    returnMsg.getBytes().length, receivePacket.getAddress(),
                    receivePacket.getPort());
                    mServerSocket.send(outPacket);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    public void StopServer()
    {
        super.stop();
        mServerSocket.close();
    }

```

```
    }  
}
```

Bài 3. Viết chương trình tán gẫu (chat) theo chế độ không nối kết. Cho phép hai người ở hai máy tính, có thể trò chuyện với nhau. Giao diện gồm các chức năng như sau:



Bước 1: Tạo class ChatUDPServer

```
import java.io.IOException;  
import java.net.*  
import java.util.*;  
  
public class ChatUDPServer {  
    static int port = 23;//Mở cổng 23  
    static byte[] inBuf = new byte[2048];  
    static DatagramSocket socket;  
    static ArrayList<User> inUsers = new ArrayList<User>();  
    static class User {  
        private InetAddress address;  
        private int port;  
        public User(InetAddress address, int port) {  
            super();  
            this.address = address;  
            this.port = port;  
        }  
        public InetAddress getAddress() {  
            return address;  
        }  
        public void setAddress(InetAddress address) {  
            this.address = address;  
        }  
    }  
}
```

```

        public int getPort() {
            return port;
        }
        public void setPort(int port) {
            this.port = port;
        }
    }
    public static void main(String[] args) {
        try {
            socket = new DatagramSocket(port);
            System.out.println("Server started at port " + port);
            DatagramPacket inPacket = new DatagramPacket(inBuf, inBuf.length());
            while (true) {
                try {
                    socket.receive(inPacket);
                    boolean firstJoin = true;
                    User user = new User(inPacket.getAddress(), inPacket.getPort());
                    System.out.println("User port: " + user.getPort());
                    for (User u : inUsers) {
                        if (u.getPort() == user.getPort() &&
                            u.getAddress().equals(user.getAddress())) {
                            firstJoin = false;
                        }
                    }
                    System.out.println("inPackets length: " + inUsers.size());
                    String inMsg = new String(inPacket.getData(), 0, inPacket.getLength());
                    // Gửi tín hiệu tới phòng chat
                    StringTokenizer st = new StringTokenizer(inMsg, "\t");
                    String senderName = st.nextToken();
                    String msg = st.nextToken();
                    if (msg.equals("~leave")) {
                        // Người dùng rời phòng chat
                        for (User u : inUsers) {
                            if (u.getAddress().equals(user.getAddress()) && u.getPort() ==
                                user.getPort())
                                {
                                    user = u;
                                }
                        }
                        inUsers.remove(user);
                        sendMsg(senderName + " left the room! \n");
                    }
                }
            }
        }
    }
}

```

```

        } else {
            if (firstJoin) {
                inUsers.add(user);
                sendMsg(senderName + " has joined the room!\n");
                firstJoin = false;
            }
            else {
                String outMsg = senderName + ": " + msg + "\n";
                System.out.println(outMsg);
                sendMsg(outMsg);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
} catch (SocketException e) {
    e.printStackTrace();
}
}

private static void sendMsg(String outMsg) {
DatagramPacket outPacket = new DatagramPacket(outMsg.getBytes(),
outMsg.length(), null, 0);
for (User user : inUsers) {
    outPacket.setAddress(user.getAddress());
    outPacket.setPort(user.getPort());
    if (socket != null) {
        try {
            socket.send(outPacket);
System.out.println("Sent to chatroom: " +
outPacket.getAddress().toString() + " " + outPacket.getPort() + "\n" +
new String(outPacket.getData(), 0, outPacket.getLength()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}
}

```

Bước 2: Tao class UDPClient

```

import java.awt.*;
import java.io.IOException;
import java.net.*;
import javax.swing.*

public class ChatUDPCClient extends JFrame implements ActionListener {
    static String nickName = "Dev";
    static String hostName = "localhost";
    static int port = 23;
    static DatagramSocket socket;
    InetAddress address = null;
    static JTextArea chatInput = null;
    static JTextArea chatView = null;
    JTextField hostTF;
    JTextField portTF;
    JTextField nickNameTF;
    JButton sendBtn;
    JButton connBtn;
    static Thread getMsgThread;
    public ChatUDPCClient() {
        initFrame();
    }
    public static void main(String[] args) {
        new ChatUDPCClient();
        getMsgThread = new Thread(new MsgReceiver());
    }
    static class MsgReceiver implements Runnable {
        byte[] buf = new byte[2048];
        @Override
        public void run() {
            while (true) {
                DatagramPacket inPacket = new DatagramPacket(buf, buf.length);
                try {
                    socket.receive(inPacket);
                    if (inPacket.getLength() != 0) {
                }
                String inMsg = new String(inPacket.getData(), 0,inPacket.getLength());
                chatView.append(inMsg);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

        }
    }
}

// Phương thức tạo giao diện
private void initFrame() {
    setSize(500, 400);
    setMinimumSize(new Dimension(500, 400));
    setTitle(nickName + "'s chat box");
    Container container = getContentPane();
    JPanel panel = new JPanel(new BorderLayout());
    JPanel topPanel = new JPanel();
    JLabel lb1 = new JLabel("Host");
    hostTF = new JTextField(10);
    JLabel lb2 = new JLabel("Port");
    portTF = new JTextField(4);
    JLabel lb3 = new JLabel("Nickname");
    nickNameTF = new JTextField(6);
    connBtn = new JButton("Connect");
    connBtn.addActionListener(this);
    hostTF.setText(hostName);
    portTF.setText(port + "");
    nickNameTF.setText(nickName);
    topPanel.add(lb1);
    topPanel.add(hostTF);
    topPanel.add(lb2);
    topPanel.add(portTF);
    topPanel.add(lb3);
    topPanel.add(nickNameTF);
    topPanel.add(connBtn);
    panel.add(topPanel, "North");
    chatView = new JTextArea();
    chatView.setEditable(false);
    chatView.setLineWrap(true);
    JScrollPane conversationScrollView = new
JScrollPane(chatView);
    JScrollPane chatInputScrollView = new JScrollPane();
    panel.add(conversationScrollView, "Center");
    chatInput = new JTextArea(3, 10);
    chatInput.setLineWrap(true);
    chatInput.setEditable(false);
}

```

```

chatInput.getDocument().addDocumentListener(new DocumentListener()
{
    @Override
        public void removeUpdate(DocumentEvent e) {
            checkInput();
        }
    @Override
        public void insertUpdate(DocumentEvent e) {
            checkInput();
        }
});
chatInputScrollView.setViewportView(chatInput);
JPanel chatToolPanel = new JPanel(new BorderLayout());
chatToolPanel.add(chatInputScrollView, "Center");
sendBtn = new JButton("Send");
sendBtn.setEnabled(false);
sendBtn.addActionListener(this);
JPanel btnPanel = new JPanel();
btnPanel.add(sendBtn);
chatToolPanel.add(btnPanel, "East");
panel.add(chatToolPanel, "South");
container.add(panel);
setDefaultCloseOperation(EXIT_ON_CLOSE);
Runtime.getRuntime().addShutdownHook(new Thread(new Runnable() {
    public void run() {
        leave();
    }
})));
setVisible(true);
}
private void leave() {
    try {
        sendMsg(nickName + "\t" + "~leave");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
private void checkInput() {
    if (chatInput.getText().length() > 0) {
        sendBtn.setEnabled(true);
    }
}

```

```

        }
    else {
        sendBtn.setEnabled(false);
    }
}

@Override
public void actionPerformed(ActionEvent e) {
    String btnName = ((JButton) e.getSource()).getText();
    switch (btnName) {
    case "Connect":
        nickName = nickNameTF.getText();
        hostName = hostTF.getText();
        port = Integer.parseInt(portTF.getText());
        try {
            socket = new DatagramSocket();
            address = InetAddress.getByName(hostName);
            String msg = nickName + "\t" + "join";
            sendMsg(msg);
            getMsgThread.start();
            chatInput.setEditable(true);
            this.setTitle(nickName + "'s chat box");
            hostTF.setEditable(false);
            portTF.setEditable(false);
            nickNameTF.setEditable(false);
            connBtn.setEnabled(false);
        } catch (IOException e2) {
            e2.printStackTrace();
        }
        break;
    case "Send":
        String msg = nickName + "\t" + chatInput.getText();
        try {
            sendMsg(msg);
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        chatInput.setText("");
        break;
    default:
        break;
    }
}

```

```

        }
    }

    private void sendMsg(String msg) throws IOException {
        DatagramPacket outPacket = new DatagramPacket(msg.getBytes(),
msg.length(), address, port);
        socket.send(outPacket);
    }
}

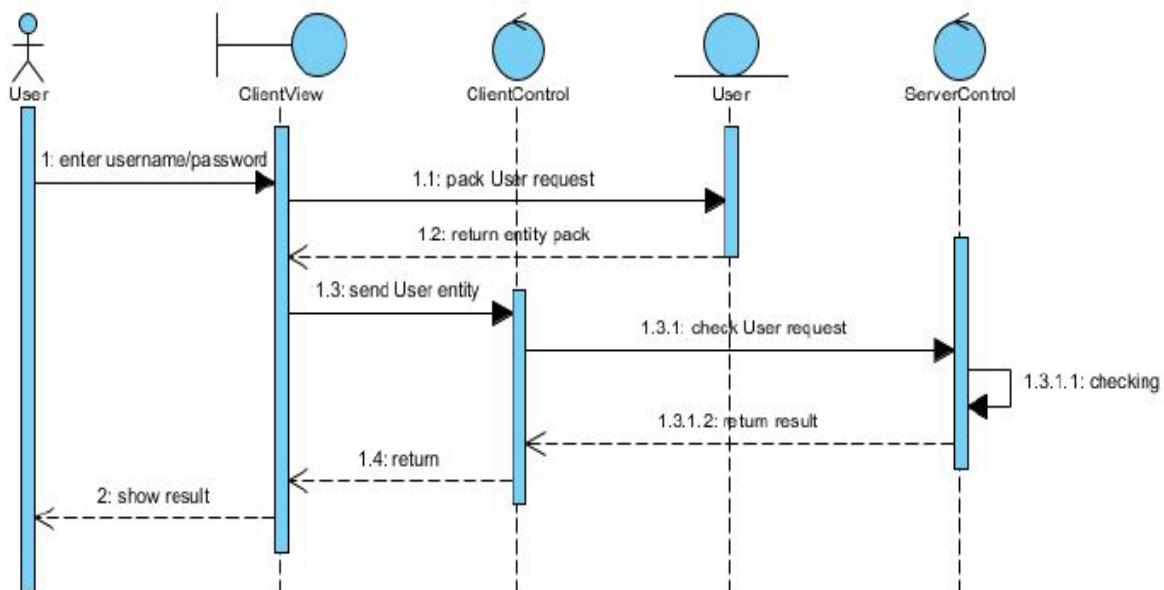
```

Bài 4. Kết hợp CSDL, xây dựng chương trình Login từ xa dùng giao thức UDP

Bài toán login từ xa dùng giao thức UDP đặt ra như sau:

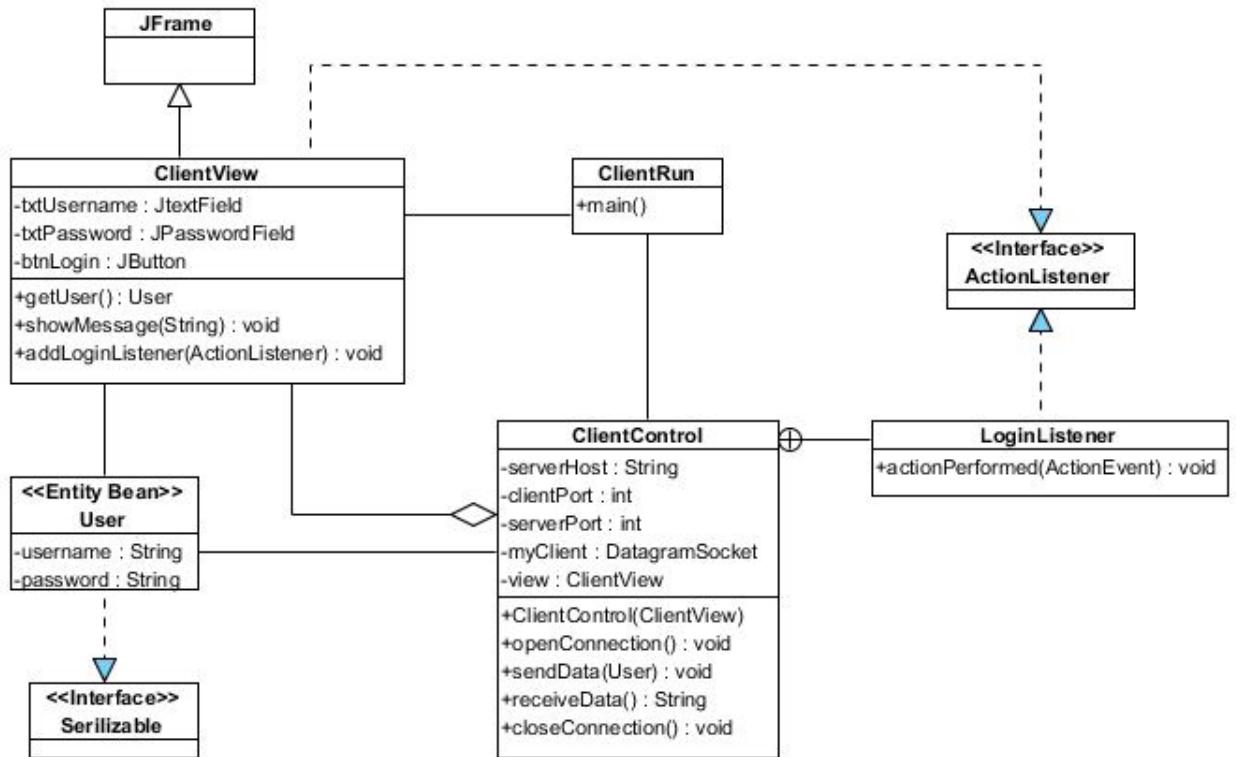
- CSDL được lưu trữ và quản lí trên Server UDP, trong đó có bảng users chứa ít nhất hai cột: cột username và cột password.
- Chương trình phía Client UDP hiện giao diện đồ họa, trong đó có một ô text để nhập username, một ô text để nhập password, và một nút nhấn Login.
- Khi nút Login được click, chương trình Client sẽ gửi thông tin đăng nhập (username/password) trên form giao diện, và gửi sang Server theo giao thức UDP
- Tại phía Server, mỗi khi nhận được thông tin đăng nhập gửi từ Client, sẽ tiến hành kiểm tra trong cơ sở dữ liệu xem có tài khoản nào trùng với thông tin đăng nhập nhận được hay không. Sau khi có kết quả kiểm tra (đăng nhập đúng, hoặc sai), Server UDP sẽ gửi kết quả này về cho Client tương ứng, theo giao thức UDP.
- Ở phía Client, sau khi nhận được kết quả đăng nhập (đăng nhập đúng, hoặc sai) từ Server, sẽ hiển thị thông báo tương ứng với kết quả nhận được: nếu đăng nhập đúng thì thông báo login thành công. Nếu đăng nhập sai thì thông báo username/password không đúng. Yêu cầu kiến trúc hệ thống được thiết kế theo mô hình MVC.

Hướng dẫn



Hình 11. Tuần tự các bước thực hiện theo giao thức UDP

Xây dựng sơ đồ lớp phía Client : được thiết kế theo mô hình MVC bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:



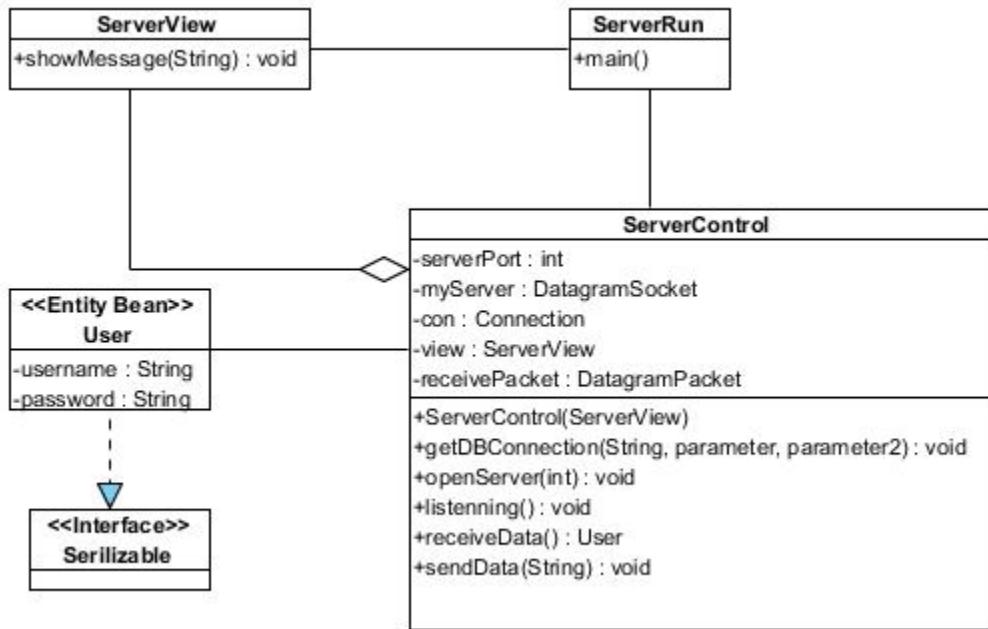
Hình 12. Sơ đồ lớp phía Client

Lớp User: tương ứng với thành phần model (M), bao gồm hai thuộc tính username và password, các hàm khởi tạo và các cặp getter/setter tương ứng với các thuộc tính.

Lớp ClientView: tương ứng với thành phần view (V), kế thừa từ lớp JFrame của Java, chứa các thuộc tính là các thành phần đồ họa bao gồm ô text nhập username, ô text nhập password, nút Login.

Lớp ClientControl: tương ứng với thành phần control (C), chứa một lớp nội tại là LoginListener. Khi nút Login trên tầng view click thì nó sẽ chuyển tiếp sự kiện xuống lớp này để xử lý.

Tất cả các xử lí đều gọi từ phương thức actionPerformed của lớp nội tại này, bao gồm: lấy thông tin trên form giao diện và gửi sang Server theo giao thức UDP, nhận kết quả đăng nhập từ Server về và yêu cầu form giao diện hiển thị.



Hình 13. Sơ đồ lớp phía Server

Sơ đồ lớp của phía Server được thiết kế theo mô hình MVC, bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:

Lớp User: Lớp thực thể, dùng chung thông nhất với lớp phía bên Client (M).

Lớp ServerView: Lớp tương ứng với thành phần view (V), là lớp dùng hiển thị các thông báo và trạng thái hoạt động bên Server UDP.

Lớp ServerControl: Tương ứng với thành phần control (C), đảm nhiệm vai trò xử lí của Server UDP, bao gồm: nhận thông tin đăng nhập từ phía các Client, kiểm tra trong CSDL xem các thông tin này đúng hay sai, sau đó gửi kết quả đăng nhập cho Client.

Sinh viên vận dụng các kiến thức đã học về lập trình mạng và kết nối CSDL tiến hành viết các lớp.

Bước 1: Tạo các lớp phía Client

Lớp User

```

package udp.Client;
import java.io.Serializable;
public class User implements Serializable{
    private String userName;
    private String password;
    public User(){
    }
    public User(String username, String password){
        this.userName = username;
        this.password = password;
    }
}

```

```

public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}

```

Lớp ClientView

```

public class ClientView extends JFrame implements ActionListener{
    private JTextField txtUsername;
    private JPasswordField txtPassword;
    private JButton btnLogin;
    public ClientView(){
        super("UDP Login MVC");
        txtUsername = new JTextField(15);
        txtPassword = new JPasswordField(15);
        txtPassword.setEchoChar('*');
        btnLogin = new JButton("Login");
        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());
        content.add(new JLabel("Username:"));
        content.add(txtUsername);
        content.add(new JLabel("Password:"));
        content.add(txtPassword);
        content.add(btnLogin);
        this.setContentPane(content);
        this.pack();
        this.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent e) {

```

```

    }
public User getUser(){
    User model = new User(txtUsername.getText(), txtPassword.getText());
    return model;
}
public void showMessage(String msg){
    JOptionPane.showMessageDialog(this, msg);
}
public void addLoginListener(ActionListener log) {
    btnLogin.addActionListener(log);
}
}

```

Lớp ClientControl

```

public class ClientControl {
    private ClientView view;
    private int ServerPort = 5555;
    private int ClientPort = 6666;
    private String ServerHost = "localhost";
    private DatagramSocket myClient;
    public ClientControl(ClientView view){
        this.view = view;
        this.view.addLoginListener(new LoginListener());
    }
    class LoginListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            openConnection();
            User user = view.getUser();
            sendData(user);
            String result = receiveData();
            if(result.equals("ok"))
                view.showMessage("Login succesfully!");
            else
                view.showMessage("Invalid username and/or password!");
                closeConnection();
        }
    }
    private void openConnection(){
        try {

```

```

        myClient = new DatagramSocket(ClientPort);
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

private void closeConnection(){
    try {
        myClient.close();
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

private void sendData(User user){
    try {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(user);
        oos.flush();
        InetAddress IPAddress = InetAddress.getByName(ServerHost);
        byte[] sendData = baos.toByteArray();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
            sendData.length, IPAddress, ServerPort);
        myClient.send(sendPacket);
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

private String receiveData(){
    String result = "";
    try {
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
        myClient.receive(receivePacket);
        ByteArrayInputStream bais = new ByteArrayInputStream(receiveData);
        ObjectInputStream ois = new ObjectInputStream(bais);
        result = (String)ois.readObject();
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

```

```
        }
        return result;
    }
}
```

Tạo lớp ClientRun

```
public class ClientRun {
    public static void main(String[] args) {
        ClientView view = new ClientView();
        ClientControl control = new ClientControl(view);
        view.setVisible(true);
    }
}
```

Bước 2: Các lớp phía Server

Tạo lớp ServerView

```
package udp.Server;
public class ServerView {
    public ServerView(){
    }
    public void showMessage(String msg){
        System.out.println(msg);
    }
}
```

Tạo lớp ServerControl

```
public class ServerControl {
    private ServerView view;
    private Connection con;
    private DatagramSocket myServer;
    private int ServerPort = 5555;
    private DatagramPacket receivePacket = null;
    public ServerControl(ServerView view){
        this.view = view;
        getDBConnection("usermanagement", "root", "12345678");
        openServer(ServerPort);
        view.showMessage("UDP Server is running...");
    }
    while(true){
        listenning();
    }
}
```

```

}

private void getDBConnection(String dbName, String username, String
password)
{
    String dbUrl = "jdbc:mysql://localhost:3306/" + dbName;
    String dbClass = "com.mysql.jdbc.Driver";
    try {
        Class.forName(dbClass);
        con = DriverManager.getConnection (dbUrl, username, password);
    }
    catch(Exception e) {
        view.showMessage(e.getStackTrace().toString());
    }
}

private void openServer(int portNumber){
try {
    myServer = new DatagramSocket(portNumber);
} catch(IOException e) {
    view.showMessage(e.toString());
}
}

private void listenning(){
    User user = receiveData();
    String result = "false";
    if(checkUser(user)){
        result = "ok";
    }
    sendData(result);
}

private void sendData(String result){
try {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(baos);
    oos.writeObject(result);
    oos.flush();

    InetAddress IPAddress = receivePacket.getAddress();
    int ClientPort = receivePacket.getPort();
    byte[] sendData = baos.toByteArray();
    DatagramPacket sendPacket = new DatagramPacket(sendData,

```

```

        sendData.length, IPAddress, ClientPort);
        myServer.send(sendPacket);
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
}

private User receiveData(){
    User user = null;
    try {
        byte[] receiveData = new byte[1024];
        receivePacket = new DatagramPacket(receiveData,receiveData.length);
        myServer.receive(receivePacket);
        ByteArrayInputStream bais = new ByteArrayInputStream(receiveData);
        ObjectInputStream ois = new ObjectInputStream(bais);
        user = (User)ois.readObject();
    } catch (Exception ex) {
        view.showMessage(ex.getStackTrace().toString());
    }
    return user;
}

private boolean checkUser(User user) {
    String query = "Select * FROM users WHERE username ='" +
    + user.getUserName() + "' AND password ='" + user.getPassword() + "'";
    try {
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next()) {
            return true;
        }
    } catch(Exception e) {
        view.showMessage(e.getStackTrace().toString());
    }
    return false;
}
}

```

Tạo lớp ServerRun

```

package udp.Server;
public class ServerRun {

```

```
public static void main(String[] args) {  
    ServerView view = new ServerView();  
    ServerControl control = new ServerControl(view);  
}  
}
```

Kết quả:



Login lỗi



Login thành công

LAB 11. LẬP TRÌNH MULTICAST, URL, MAIL [4]

C. MỤC TIÊU

Xây dựng ứng dụng Client/Server sử dụng lớp MulticastSocket để kết nối và trao đổi thông tin.

- Thực hiện được các thao tác lập trình java giao tiếp với URL, URLConnection sử dụng lớp URL, URLConnection nằm trong gói java.net.
- Thực hiện được kỹ thuật gửi email qua gmail sử dụng mail API

D. NỘI DUNG

- Lập trình các ứng dụng multicast.
- Lập trình với URL, URLConnection
- Gửi thư với mail API.

C. KẾT QUẢ SAU KHI HOÀN THÀNH

- Viết được các ứng dụng multicast: Chat, gửi ảnh đến một nhóm người dùng.
- Xử lý liên quan đến biểu diễn tài nguyên URL trên web.
- Viết được ứng dụng gửi thư hoàn chỉnh.

D. YÊU CẦU PHẦN CỨNG - PHẦN MỀM

- Máy tính cài HDH windows, RAM tối thiểu 1GB, có kết nối Internet.
- Phần mềm NETBEAN 8.0, JDK 1.8

E. HƯỚNG DẪN

Bài 1.

Viết chương trình chat multicast, giữa Client và Server sử dụng lớp MulticastSocket.

Hướng dẫn:

Bước 1: Tạo class MulticastChatServer

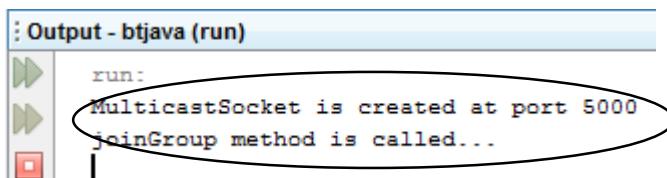
```
import java.net.*;  
  
public class MulticastChatServer {  
    public static void main(String args[]) throws Exception {  
        int portnumber = 5000;  
        if (args.length >= 1) {  
            portnumber = Integer.parseInt(args[0]);  
        }  
        // Create a MulticastSocket  
        MulticastSocket ServerMulticastSocket = new MulticastSocket(portnumber);  
        System.out.println("MulticastSocket is created at: " + portnumber);  
        // Determine the IP address of a host, given the host name  
        InetAddress group = InetAddress.getByName("225.4.5.6");  
        ServerMulticastSocket.joinGroup(group);  
        System.out.println("joinGroup method is called...");  
        boolean infinite = true;
```

```

        while (infinite) {
            byte buf[] = new byte[1024];
            DatagramPacket data = new DatagramPacket(buf, buf.length);
            ServerMulticastSocket.receive(data);
            String msg = new String(data.getData()).trim();
            System.out.println("Message received from Client = " + msg);
        }
        ServerMulticastSocket.close();
    }
}

```

Kết quả:



Bước 2. Tạo class MulticastChatClient

```

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

public class MulticastChatClient {
    public static void main(String args[]) throws Exception {
        int portnumber = 5000;
        if (args.length >= 1) {
            portnumber = Integer.parseInt(args[0]);
        }
        // Create a MulticastSocket
        MulticastSocket chatMulticastSocket = new
        MulticastSocket(portnumber);
        InetAddress group = InetAddress.getByName("225.4.5.6");
        // Joins a multicast group
        chatMulticastSocket.joinGroup(group);
        String msg = "";
        System.out.println("Type a message for the Server:");
        BufferedReader br =
        new BufferedReader(new InputStreamReader(System.in));
        msg = br.readLine();
        // Send the message to Multicast address
        DatagramPacket data = new DatagramPacket(msg.getBytes(), 0,
        msg.length(), group, portnumber);
        chatMulticastSocket.send(data);
        chatMulticastSocket.close();
    }
}

```

```
}
```

Chạy chương trình phía Client:

```
btjava (run) ✘ btjava (run) #2 ✘
```

```
run:  
Type a message for the server:  
Xin chao server  
BUILD SUCCESSFUL (total time: 20 seconds)
```

Khi đó phía Server nhận được thông tin như sau:

```
btjava (run) ✘ btjava (run) #2 ✘
```

```
run:  
MulticastSocket is created at port 5000  
joinGroup method is called...  
Message received from client = Xin chao server
```

Bài 2:

Viết chương trình broad cast thời gian và ngày tháng tới nhiều máy khách.

Hướng dẫn:

Bước 1: Tạo class BroadcastServer

```
public class BroadcastServer{  
    public static final int PORT = 1200;  
    public static void main(String args[]) throws Exception{  
        MulticastSocket socket;  
        DatagramPacket packet;  
        InetAddress address;  
        address = InetAddress.getByName();  
        socket = new MulticastSocket();  
        // join a Multicast group and send the group salutations  
        socket.joinGroup(address);  
        byte[] data = null;  
        for(;;){  
            Thread.sleep(1000);  
            System.out.println("Sending ");  
            String str = (new Date()).toString();  
            data = str.getBytes();  
            packet = new DatagramPacket(data,str.length(),address,PORT);  
            // Sends the packet  
            socket.send(packet);  
        } // for  
    } // main  
} // class BroadcastServer
```

Bước 2: Tạo class BroadcastClient

```
public class BroadcastClient{
```

```

public static final int PORT = 1200;
public static void main(String args[]) throws Exception{
    MulticastSocket socket;
    DatagramPacket packet;
    InetAddress address;
    address = InetAddress.getByName(args[0]);
    socket = new MulticastSocket(BroadcastServer.PORT);
    //join a Multicast group and send the group salutations
    socket.joinGroup(address);
    byte[] data = null;
    packet = new DatagramPacket(data,data.length);
    for(;;){
        // receive the packets
        socket.receive(packet);
        String str = new String(packet.getData());
        System.out.println(" Time signal received from"+
            packet.getAddress() + " Time is : " +str);
    } // for
} // main
} // class Broadcast

```

Bài 3: Viết chương trình multicast ảnh.

Bước 1: Tạo class MulticastImageSender

```

public class MulticastImageSender {
    private static final int TIMEOUT = 3000;
    private static final int MAXFILELEN = 65000; // File must fit in
single datagram
    public static void main(String[] args) throws IOException,
InterruptedException {
        if (args.length < 4) // Test for correct # of args
            throw new IllegalArgumentException(
                "Parameter(s): <Multicast Address> <Port> <TTL> <Image File>
[<Image File>...]");
        InetAddress multicastAddress = InetAddress.getByName(args[0]);
        int destPort = Integer.parseInt(args[1]); // Destination port of
multicast packets
        int TTL = Integer.parseInt(args[2]);
        // Create a UDP multicast socket with any available local port
        MulticastSocket socket = new MulticastSocket();
        socket.setTimeToLive(TTL); // Set the TTL
        for (int i=3; i < args.length; i++)

```

```

{
    RandomAccessFile file = new RandomAccessFile(args[i], "r");
    if (file.length() > MAXFILELEN)
        throw new IOException("File too big");
    byte [] fileBuffer = new byte[(int) file.length()];
    file.read(fileBuffer);
    file.close();
    // Create a datagram to send
    DatagramPacket sendPacket = new DatagramPacket(fileBuffer,
            fileBuffer.length, multicastAddress, destPort);
    socket.send(sendPacket); // Send the echo string
    System.out.println("Sent " + args[i] + " to " +
            sendPacket.getAddress().getHostAddress() +
            " on port " + sendPacket.getPort());
    Thread.sleep(TIMEOUT);
}
socket.close();
}
}

```

Bước 2: Tạo class MulticastImageReceiver

```

public class MulticastImageReceiver extends JFrame {
    private JLabel picture; // Label to contain image
    public MulticastImageReceiver() {
        super("Multicast Image Receiver"); // Set the window title
        setSize(300, 300); // Set the window size
        picture = new JLabel("No image", SwingConstants.CENTER);
        JScrollPane scrollPane = new JScrollPane(picture);
        getContentPane().add(scrollPane, "Center");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent evt) {
                System.exit(0);
            }
        });
    }
    public JLabel getPicture() {
        return picture;
    }
    public static void main(String[] args) throws IOException {
        if (args.length != 2) // Test for correct # of args
            throw new IllegalArgumentException("Parameter(s): <Multicast

```

```

Address> <Port>");

    final InetAddress multicastAddress = InetAddress.getByName(args[0]);
    if (!multicastAddress.isMulticastAddress())
        throw new IllegalArgumentException("Not a multicast address");
    int port = Integer.parseInt(args[1]);
    MulticastImageReceiver multicastImageReceiver = new
    MulticastImageReceiver();
    multicastImageReceiver.setVisible(true);
    new Thread(new MulticastImageReceiverThread(multicastImageReceiver,
multicastAddress, port,"No Image")).start();
}

}

class MulticastImageReceiverThread implements Runnable {
    private static final int MAXFILELEN = 65000; //
    private InetAddress multicastAddress;           // Sender multicast address
    private int port;                             // Sender port
    Runnable updateImage;
    String imageText;                           // Label text
    byte[] image = new byte[MAXFILELEN];         // Bytes of image
    boolean imageValid = false;
    public MulticastImageReceiverThread(final MulticastImageReceiver frame,
        InetAddress multicastAddress, int port, String initialImageText) {
        this.multicastAddress = multicastAddress;
        this.port = port;
        this.imageText = initialImageText;
        updateImage = new Runnable() {
            public void run() {
                JLabel picture = frame.getPicture();
                picture.setText(imageText);
                if (imageValid) {
                    ImageIcon newImage = new ImageIcon(image);
                    picture.setIcon(newImage);
                    picture.setPreferredSize(new Dimension(newImage.getIconWidth(),
                        newImage.getIconHeight()));
                } else
                    picture.setIcon(null);
                picture.revalidate();
            }
        };
    }
}

```

```

public void changeImage() {
    try {
        SwingUtilities.invokeAndWait(updateImage);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public void run() {
    DatagramPacket recvPacket = new DatagramPacket(image, MAXFILELEN);
    MulticastSocket socket;
    try {
        socket = new MulticastSocket(port);
        socket.joinGroup(multicastAddress); // Join the multicast group
    } catch (IOException e) {
        imageText = "Problem with multicast socket";
        imageValid = false;
        changeImage();
        return;
    }
    for (;;) {
        try {
            socket.receive(recvPacket); // Receive the image
        } catch (IOException e) {
            break; // Assume exception due to file closing
        }
        imageText = "";
        imageValid = true;
        changeImage();
        recvPacket.setLength(MAXFILELEN); // You have to reset this!!!
    }
}
}

```

Bài 4 .

Xây dựng chương trình multicast theo yêu cầu sau. Chức năng chương trình:

- Tham gia vào group của multicast
- Gửi dữ liệu đến địa chỉ multicast
- Nhận dữ liệu từ multicast
- Hiển thị lên màn hình
- Chỉ cần các client tham gia vào group của địa chỉ multicast này, khi có dữ liệu được gửi vào đó tất cả client đều nhận được.

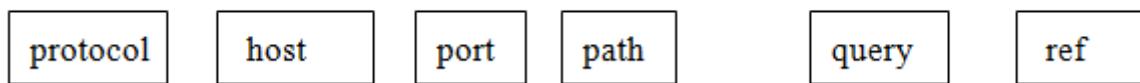
Gợi ý:

- Tham gia vào group của địa chỉ multicast
- Gửi dữ liệu đến địa chỉ multicast (Lúc này các client muốn nhận được thì phải tham gia vào group của multicast đó)

2. Lớp URL

Biểu diễn một tài nguyên trên Web. Một URL được phân chia thành các phần như sau:

https://www.gpcoder.com:80/java/index.html?page=1&order=desc#java-core



Hình 14. Các thành phần của URL

Các phương thức để truy cập các phần khác nhau của URL:

- **public String getPath()** : trả về path của URL.
- **public int getPort()** : trả về port của URL.
- **public String getProtocol()** : trả về protocol của URL.
- **public String getHost()** : trả về host của URL.
- **public String getRef()** : trả về phần reference của URL.
- **public URLConnection openConnection()**: mở một kết nối tới URL, cho phép một Client giao tiếp với tài nguyên.

Lớp URLConnection

- Lớp trừu tượng URLConnection là lớp **super class** của tất cả các lớp biểu diễn cho kết nối tương tác (2 chiều) giữa ứng dụng và một URL.
- Tạo một URLConnection từ một URL thực hiện qua bước sau :
 1. Xây dựng một đối tượng URL.
 2. Gọi phương thức **openConnection()** của đối tượng URL để tìm kiếm một đối tượng URLConnection cho URL đó.
 3. Cấu hình đối tượng URL.
 4. Đọc các trường header.
 5. Nhận một luồng nhập và đọc dữ liệu.
 6. Nhận một luồng xuất và ghi dữ liệu.
 7. Đóng liên kết

Bài 5. Phân tích các phần của một địa chỉ URL.

```
import java.net.URL;
public class TestURL
{
    public static void main(String[] args) throws Exception
    {
        URL url = new
        URL("https://www.packtpub.com:80/books/content/support");
```

```

        displayURL(url);
    }
    private static void displayURL(URL url)
    {
        System.out.println("URL: " + url);
        System.out.printf(" Protocol: %-32s Host: %-32s\n",
                           url.getProtocol(),url.getHost());
        System.out.printf(" Port: %-32d Path: %-32s\n",
                           url.getPort(),url.getPath());
        System.out.printf(" Authority: %-32s Query: %-32s\n",
                           url.getAuthority(),url.getQuery());
        System.out.println(" User Info: " + url.getUserInfo());
    }
}

```

Bài 6. Lấy dữ liệu từ 1 URL

```

public static void main(String[] args)
{
    try
    {
        DownloadResource dlr = new DownloadResource();
        URL url=new
URL("http://cdn2.tstatic.net/tribunnews/foto/bank/images/Shaila-
Sabt.jpg");
        String destinationFilePath="c:/a/girl.jpg";
        long bytes=dlr.download(url, destinationFilePath);
        System.out.printf("%d bytes downloaded",bytes);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public long download(URL url, String destinationFilePath) throws Exception
{
    long bytes=0;
    FileOutputStream fos= new FileOutputStream(destinationFilePath);
    int len=512;
    InputStream is=url.openStream();
    byte[] buffer=new byte[512];
    while(is.available()!=0)

```

```

{
    len=is.read(buffer);
    bytes+=len;
    fos.write(buffer,0,len);
}
fos.close();
return bytes;
}

```

Bài 7. Đọc thông tin, nội dung trực tiếp từ một URL.

Hướng dẫn: Tạo URL, gọi phương thức openStream() để lấy một luồng từ đó có thể đọc nội dung của URL.

```

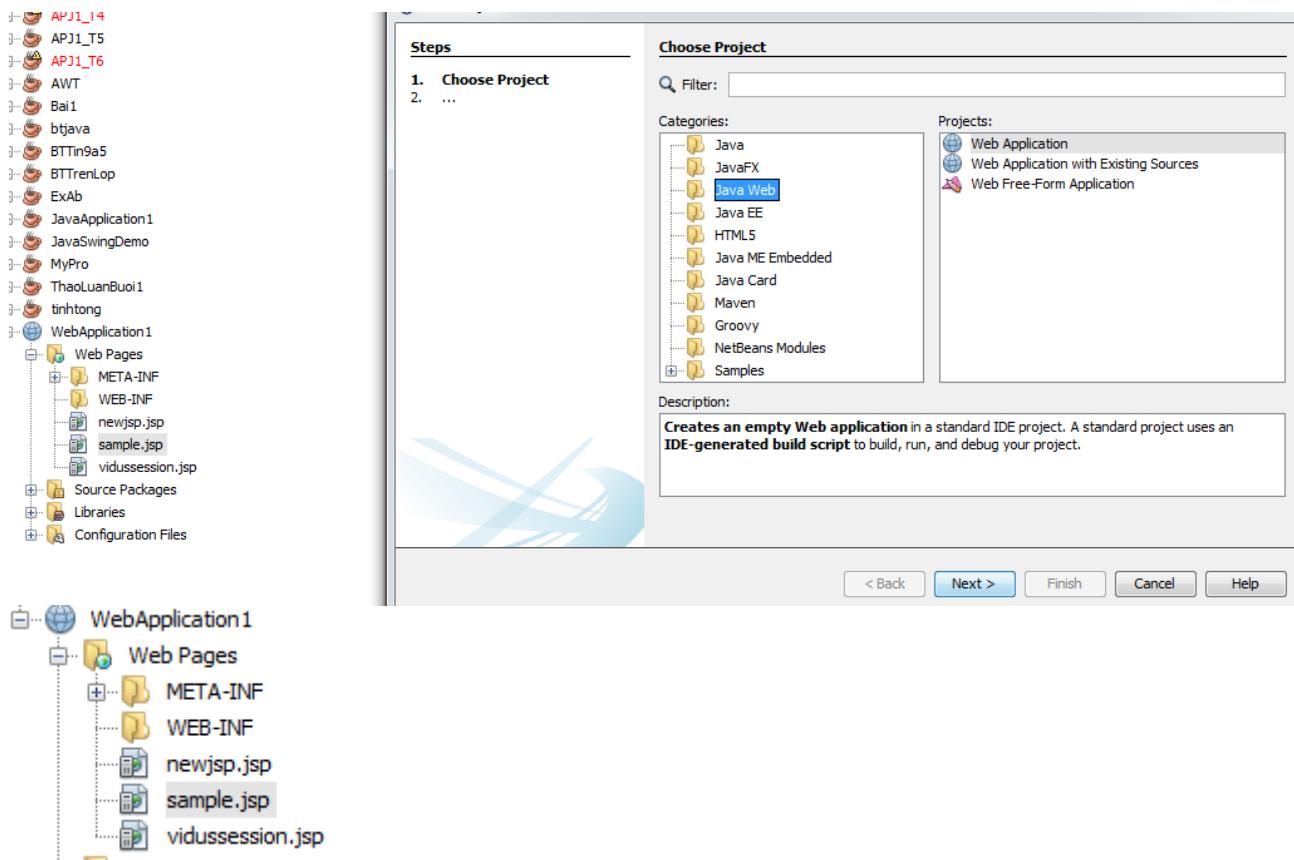
import java.net.*;
import java.io.*;
public class URLReader
{
    public static void main(String[] args) throws Exception
    {
        URL oracle = new URL("http://www.oracle.com/");
        BufferedReader in = new BufferedReader(
            new InputStreamReader(oracle.openStream()));
        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}

```

Bài 8. Viết chương trình gửi request đến một trang web xử lý sau đó nhận kết quả.

Hướng dẫn:

Bước 1: Vào NETBEAN tạo file sample.jsp có nội dung sau



Nội dung File sample.jsp

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <%
            String us=request.getParameter("user");
            String psw=request.getParameter("password");
            if(us.equals(psw))
                out.println("Welcome "+us);
            else
                out.println("Something wrong with your username & password");
        %>
    </body>
</html>

```

Bước 2: Mở trình duyệt thử với address sau

(i) localhost:8080/WebApplication1/sample.jsp?user=teo&password=teo

Bước 3: Viết code dùng URLConnection gửi dữ liệu đến web Server

```

import java.io.OutputStream;
import java.net.URL;
import java.netURLConnection;
import java.util.Scanner;
public class SendoutputSample
{
    public static void main(String[] args) throws Exception
    {
        String spec="http://localhost:8080/sample.jsp";
        URL url=new URL(spec);
        URLConnection urlcon=url.openConnection();
        urlcon.setDoInput(true);
        urlcon.setDoOutput(true);
        urlcon.connect();
        try(OutputStream os = urlcon.getOutputStream())
        {
            os.write("user=teo&password=teo".getBytes());
            os.flush();
        }
        try(Scanner in=new Scanner(urlcon.getInputStream()))
        {
            if (in.hasNextLine())
            {
                String line=in.nextLine();
                System.out.println(line);
            }
        }
    }
}

```

Thực thi, quan sát kết quả

Bước 4: Thay password trong dòng code gạch chân trên với password bằng một giá trị gì đó khác “teo”. Quan sát kết quả.

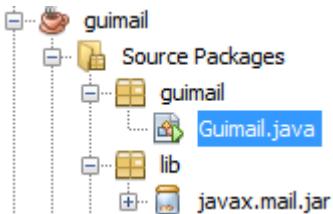
Bài 9. Viết chương trình gửi mail qua địa chỉ gmail cho trước.

Hướng dẫn:

Bước 1: Email phải kích hoạt tính năng POP:

- Để kích hoạt POP; Sau khi đăng nhập vào tài khoản gmail -> chọn Setting -> chọn Forwarding and POP/IMAP
- Chọn chức năng pop cho tất cả thư

Bước 2: import thư viện javax.mail.jar



Bước 3: Tạo phương thức sendmail với thuộc tính truyền vào tên email, password của email gửi, tên email nhận, nội dung gửi, tiêu đề.

```

import java.util.Properties;
import javax.mail.*;
public class Guimail
{
    public static void sendmail(String userName, String password, String
email2, String content, String title) throws AddressException,
MessagingException
    {
        //Tạo đối tượng để thiết lập các thuộc tính cho việc gửi mail
        Properties props = new Properties();
        props.put("mail.smtp.auth", true);
        props.put("mail.smtp.starttls.enable", true);
        props.put("mail.smtp.host", "smtp.gmail.com");
        // port mail thường là 587 hoặc 456
        props.put("mail.smtp.port", "587");
        // Tạo đối tượng Session (phiên làm việc với gmail)
        Session session = Session.getDefaultInstance(props, new
Authenticator()
        {
            protected PasswordAuthentication
getUserNamePasswordAuthentication()
            {
                //để gửi mail qua SMTP cần có tài khoản hợp lệ của google
                return new PasswordAuthentication(userName, password);
            }
        });
        // Tạo đối tượng chứa thông điệp cần gửi mail
        Message message=new MimeMessage(session);
        // Truyền địa chỉ muốn gửi thông điệp
        message.addRecipient(Message.RecipientType.TO, new
InternetAddress(email2));
        message.setSubject(content);
        message.setContent(title,"text/html");
    }
}

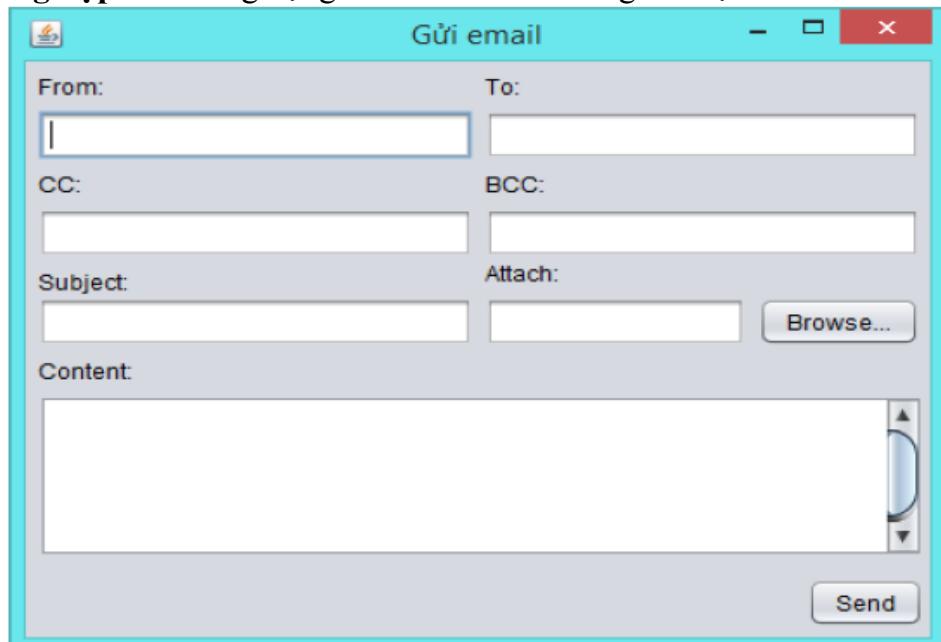
```

```

        Transport.send(message,userName,password);
    }
public static void main(String[] args) throws MessagingException
{
    try
    {
        String username1="congquyen2503";
        String pass="quyen2345";
        String username2="luongthaohieu@gmail.com";
        String content="test xem duoc khong ;";
        String title="mail from "+username1;
        sendmail(username1,pass,username2,content,title);
        System.out.println("gui thanh cong");
    }
    catch(Exception e)
    {
        System.out.println("gui that bai :" +e.getMessage());
        e.printStackTrace();
    }
}
}

```

Bài 10. Tổng hợp: Viết ứng dụng mail hoàn chỉnh có giao diện như sau:



Gợi ý:

Tạo giao diện, thực hiện chức năng gửi mail.

Chú ý hai thư viện gửi mail: mail.jar và activation.jar

LAB 12. LẬP TRÌNH PHÂN TÁN VỚI RMI [5]

A. MỤC TIÊU

- Trang bị cho sinh viên kỹ thuật lập trình phân tán và công nghệ RMI
- Ứng dụng công nghệ RMI để xây dựng các ứng dụng phân tán
- Chuyển tham số cho phương thức triệu gọi từ xa và nhận kết quả trả về từ phương thức triệu gọi từ xa.

B. NỘI DUNG

- Xây dựng giao diện từ xa
- Triển khai giao diện từ xa
- Cài đặt và đăng ký đối tượng từ xa
- Cài đặt chương trình trên máy khách, triệu gọi phương thức của đối tượng cài đặt từ xa

C. KẾT QUẢ SAU KHI HOÀN THÀNH

Viết chương trình ứng dụng phân tán với RMI, áp dụng vào các bài toán cụ thể: Xây dựng ứng dụng tra cứu thông tin sách phân tán, mô phỏng bài toán nạp, rút tiền ngân hàng, phát triển hệ thống ngân hàng phân tán.

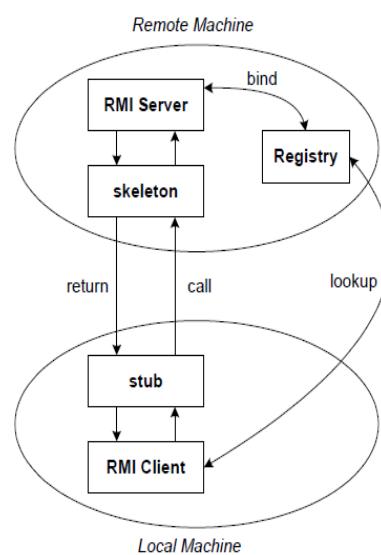
D. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

- Máy tính cài HĐH windows, RAM tối thiểu 1GB.
- Phần mềm NETBEAN 8.0, JDK 1.8, SQL Server 2015.

E. HƯỚNG DẪN

1. Tạo giao diện từ xa (Remote interface) cần tuân theo các bước sau:

- Giao diện từ xa phải là một giao diện **public**
- Phải được kế thừa từ giao diện **Remote**
- Tất cả các phương thức trong giao diện từ xa đều bung ra ngoại lệ **RemoteException**
- Nếu tham số truyền cho phương thức hoặc giá trị nhận về từ phương thức triệu gọi từ xa là một đối tượng thì đối tượng đó phải **implements** giao diện **Remote** hoặc giao diện **Serializable**



Hình 15. Kiến trúc RMI

Tạo lớp cài đặt giao diện từ xa

```
public class class_name extends UnicastRemoteObject implements  
interface_name {  
    Phương thức khởi tạo  
    public class_name() throws RemoteException {  
    }  
    Cài đặt xử lý cho tất cả các phương thức có trong giao diện  
    public datatype|void method_name([parameter list]) throws RemoteException  
    {  
        Viết xử lý cho phương thức  
    }  
}
```

Viết xử lý phía Client

```
Tạo đối tượng Registry  
Registry reg = LocateRegistry.getRegistry(Servername, portnumber);  
//portnumber nằm trong phạm vi 1024-65535  
Truy xuất remoteObject  
interface_name remoteObject = (interface_name)reg.lookup(registername);  
Gọi phương thức thông qua remoteObject  
remoteObject.method_name([parameter list])
```

Viết xử lý phía Server

```
Tạo đối tượng Registry  
Registry reg = LocateRegistry.createRegistry(portnumber);  
Đăng ký Remote Object  
class_name remoteObject = new class_name();  
reg.bind(registername, remoteObject);
```

Bài 1. Xây dựng chương trình gửi số thực a từ máy Client lên máy chủ Server (sử dụng cổng 1102). Server tính giá trị bình phương của số thực a vừa nhận được rồi gửi về cho Client. Kết quả hiển thị trên máy Client.

Hướng dẫn:

Bước 1: Tạo giao diện Calculator

```
package rmi_demo;  
import java.rmi.Remote;  
import java.rmi.RemoteException;  
public interface Calculator extends Remote {  
    // Khai báo phương thức tính bình phương  
    public double square(double a) throws RemoteException;  
}
```

Bước 2: Tạo lớp cài đặt interface (Sq_calculator)

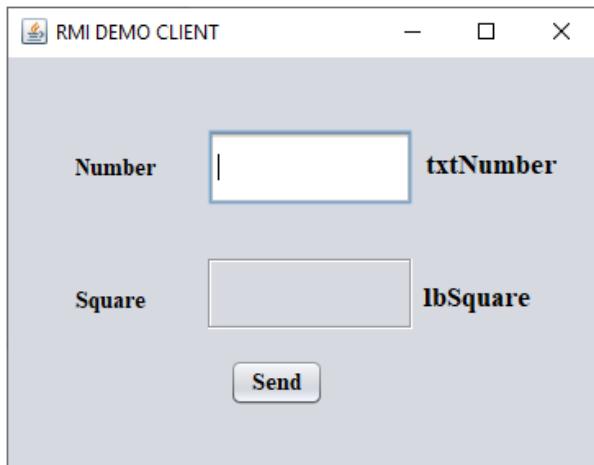
```

package rmi_demo;
import java.rmi.RemoteException;
import java.rmi.Server.UnicastRemoteObject;
public class Sq_calculator extends UnicastRemoteObject implements Calculator{
    // Khai báo phương thức khởi tạo
    public Sq_calculator() throws RemoteException{
    }
    // Viết xử lý cho phương thức tính bình phương
    public double square(double a) throws RemoteException {
        return a*a;
    }
}

```

Bước 3: Xây dựng lớp xử lý ở phía Client. Tạo File RMI_Client

- Xây dựng jFrame Form:



- Viết code xử lý sự kiện khi người sử dụng bấm nút Send:

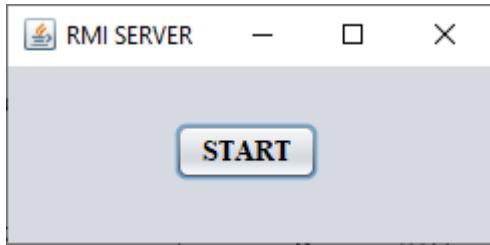
```

private void btSendActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // Gọi Server đang lắng nghe tại cổng 1102
        Registry reg = LocateRegistry.getRegistry("localhost", 1102);
        // Lấy đối tượng từ xa
        Calculator cal = (Calculator) reg.lookup("RMICalSer");
        // Gọi phương thức từ xa
        double result = cal.square(Double.parseDouble(txtNum.getText()));
        // Hiển thị kết quả
        lblResult.setText(String.valueOf(result));
    } catch (Exception e) {
    }
}

```

Bước 4. Viết chương trình xử lý phía Server:

- Xây dựng jFrame Form:



- Viết code xử lý khi người dùng nhấp chuột vào nút Start:

```
private void btStartActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        //Tạo đối tượng Registry  
        Registry reg = LocateRegistry.createRegistry(1102);  
        //Đăng ký Remote Object  
        Sq_calculator ci = new Sq_calculator();  
        reg.bind("RMICalSer", ci);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Bài 2.

Xây dựng chương trình tra cứu thông tin sách từ xa như sau: Client gửi thông tin mã isbn đến phía Server, Server tra cứu thông tin nếu có sách trùng mã isbn gửi đến thì hiện thông tin sách, nếu không có hiện thông báo không thấy. Ngoài ra phía Client thực hiện tra cứu thông tin sách sẽ hiện toàn thông tin các quyển sách ra màn hình.

Hướng dẫn:

Bước 1: Tạo interface RMInterface

```
public interface RMInterface extends Remote {  
    Book findBook(Book b) throws RemoteException;  
    ArrayList<Book> allBooks() throws RemoteException;  
}
```

Bước 2: Tạo Class Book

```
public class Book implements Serializable {  
    private static final long serialVersionUID = 1190476516911661470L;  
    private String title;  
    private String isbn;  
    private double cost;  
    public Book(String isbn) {  
        this.isbn = isbn;  
    }  
    public Book(String title, String isbn, double cost) {
```

```

        this.title = title;
        this.isbn = isbn;
        this.cost = cost;
    }
    public String getTitle() {
        return title;
    }
    public String getIsbn() {
        return isbn;
    }
    public double getCost() {
        return cost;
    }
    public String toString() {
        return "> " + this.title + " (" + this.cost + ")";
    }
}

```

Bước 3:

```

public class BookStore extends UnicastRemoteObject implements
RMInterface {
    private static final long serialVersionUID = 1L;
    private ArrayList<Book> bookList;
    protected BookStore(ArrayList<Book> list) throws RemoteException {
        super();
        this.bookList = list;
    }
    @Override
    public Book findBook(Book book) throws RemoteException {
        Predicate<Book> predicate = x -> x.getIsbn().equals(book.getIsbn());
        return
            bookList.stream().filter(predicate).findFirst().get();
    }
    @Override
    public ArrayList<Book> allBooks() throws RemoteException {
        return bookList;
    }
    private static ArrayList<Book> initializeList() {
        ArrayList<Book> list = new ArrayList<Book>();
        list.add(new Book("Head First Java, 2nd Edition", "978-0009205", 31.41));
    }
}

```

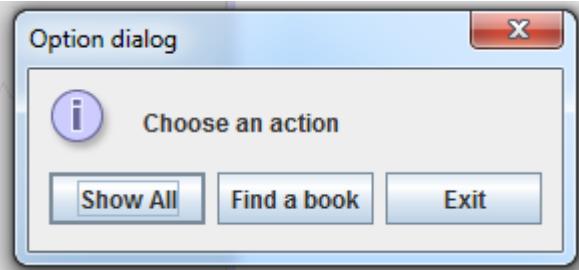
```

list.add(new Book("Java In A Nutshell", "978-059737", 10.90));
list.add(new Book("Java: The Complete Reference", "978-8082", 40.18));
list.add(new Book("Head First Servlets and JSP", "978-16680", 35.41));
list.add(new Book("Java Puzzlers", "978-0321336781", 39.99));
    return list;
}

public static void main(String[] args) {
    try {
        Registry reg=LocateRegistry.createRegistry(1122);
        BookStore bst = new BookStore(initializeList());
        reg.bind("abc", bst);
        System.out.println("Server ready");
    } catch (Exception e) {
        System.out.println("Server exception: " + e.getMessage());
    }
}
}

```

Bước 4: Tạo class Customer thực hiện phía Client. Giao diện phía Client gồm các chức năng sau:



```

public class Customer {
    private static RMIClient rmi;
    public static void main(String[] args) throws
        MalformedURLException, RemoteException, NotBoundException
    {
        Registry reg = LocateRegistry.getRegistry("localhost", 1122);
        // Lấy đối tượng từ xa
        RMIClient rmi = (RMIClient) reg.lookup("abc");
        // Gọi phương thức từ xa
        boolean findmore;
        do {
            String[] options = {"Show All", "Find a book", "Exit"};
            int choice = JOptionPane.showOptionDialog(null, "Choose an

```

```

        action", "Option dialog", JOptionPane.DEFAULT_OPTION,
                JOptionPane.INFORMATION_MESSAGE,
                null, options, options[0]);
        switch (choice) {
            case 0:
                ArrayList<Book> list = rmi.allBooks();
                StringBuilder message = new StringBuilder();
                list.forEach(x -> {
                    message.append(x.toString() + "\n");
                });
                JOptionPane.showMessageDialog(null, new String(message));
                break;
            case 1:
                String isbn = JOptionPane.showInputDialog("Type the isbn of the book you
                        want to find.");
                try {
                    Book response = rmi.findBook(new Book(isbn));
                    JOptionPane.showMessageDialog(null, "Title: " +response.getTitle() +
                            "\n" + "Cost: $" +response.getCost(),response.getIsbn(),
                    JOptionPane.INFORMATION_MESSAGE);
                } catch (NoSuchElementException ex) {
                    JOptionPane.showMessageDialog(null, "Not found");
                }
                break;
            default:
                System.exit(0);
                break;
        }
        findmore = (JOptionPane.showConfirmDialog(null, "Do you want to
        exit?", "Exit",JOptionPane.YES_NO_OPTION) == JOptionPane.NO_OPTION);
    } while (findmore);
}
}

```

Bài 3. Dùng RMI mô phỏng giao dịch gửi, rút tiền trong ngân hàng

Bước 1. Tạo giao diện Account có các phương thức sau:

```

import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Account extends Remote {
    public String getName() throws RemoteException;
}

```

```

    public float getBalance() throws RemoteException;
    public void withdraw(float amt) throws RemoteException;
    public void deposit(float amt) throws RemoteException;
    public void transfer(float amt, Account src) throws RemoteException;
}

```

Bước 2. Tạo lớp cài đặt interface trên (AccountImpl)

```

public class AccountImpl extends UnicastRemoteObject implements Account {
    private float balance = 0;
    private String name = "";
    // Constructor creates a new account with the given name
    public AccountImpl() throws RemoteException
    {

    }
    public AccountImpl(String aName) throws RemoteException {
        name = aName;
    }
    public String getName() throws RemoteException {
        return name;
    }
    public float getBalance() throws RemoteException {
        return balance;
    }
    //Withdraw some funds
    public void withdraw(float amt) throws RemoteException {
        balance -= amt;
        //Ensure balance never drops below zero
        balance = Math.max(balance, 0);
    }
    //Deposit some funds
    public void deposit(float amt) throws RemoteException {
        balance += amt;
    }
    //Transfer some funds from another (remote) account Into this one
    public void transfer(float amt, Account src) throws RemoteException
    {
        src.withdraw(amt);
        this.deposit(amt);
    }
}

```

Bước 3. Xây dựng lớp xử lý ở phía Client. Tạo File RMI_Client

- Xây dựng jFrame Form:



- Viết code xử lý sự kiện khi người sử dụng bấm nút NẠP TIỀN

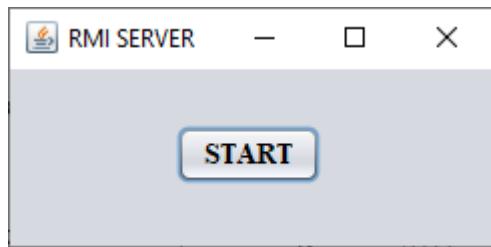
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        // Gọi Server đang lắng nghe tại cổng 1102  
        Registry reg = LocateRegistry.getRegistry("localhost", 1102);  
        // Lấy đối tượng từ xa  
        Account ac = (Account) reg.lookup("JhonAdams");  
        // Gọi phương thức từ xa  
        float sotien=Float.parseFloat(jTextField1.getText());  
        ac.deposit(sotien);  
        float balance = ac.getBalance();  
        jTextField2.setText(String.valueOf(balance));  
    }  
    catch (Exception e) {  
    }  
}
```

Viết code xử lý sự kiện khi người sử dụng bấm nút RÚT TIỀN

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        Registry reg = LocateRegistry.getRegistry("localhost", 1102);  
        Account ac = (Account) reg.lookup("JhonAdams");  
        float sotien=Float.parseFloat(jTextField1.getText());  
        ac.withdraw(sotien);  
        float balance = ac.getBalance();  
        jTextField2.setText(String.valueOf(balance));  
    }  
    catch (Exception e) {  
    }  
}
```

Bước 4. Viết chương trình xử lý phía Server:

- Xây dựng jFrame Form:



- Viết code xử lý khi người dùng nhấp chuột vào nút Start:

```
private void btStartActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        Registry reg = LocateRegistry.createRegistry(1102);  
        AccountImpl acct = new AccountImpl();  
        reg.bind("JhonAdams ", acct);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Bài 4. Kết hợp cơ sở dữ liệu và lập trình RMI, xây dựng chương trình Login từ xa

Bài toán login từ xa dùng RMI đặt ra như sau:

- CSDL được lưu trữ và quản lí trên Server RMI, trong đó có bảng users chứa ít nhất hai cột: cột username và cột password.
- Tại phía Server, có khai báo, định nghĩa, và đăng ký một đối tượng từ xa có phương thức kiểm tra đăng nhập, nó sẽ tiến hành kiểm tra trong CSDL xem có tài khoản nào trùng với thông tin đăng nhập nhận được hay không.
- Chương trình phía Client phải hiện giao diện đồ họa, trong đó có một ô text để nhập username, một ô text để nhập password, và một nút nhấn Login.
- Khi nút Login được click, chương trình Client sẽ triệu gọi làm kiểm tra login từ Server RMI, lấy thông tin đăng nhập (username/password) trên form giao diện để kiểm tra. Sau khi có kết quả kiểm tra (đăng nhập đúng, hoặc sai), Client sẽ hiển thị thông báo tương ứng với kết quả nhận được: nếu đăng nhập đúng thì thông báo login thành công. Nếu đăng nhập sai thì thông báo là username/password không đúng.
- Yêu cầu kiến trúc hệ thống ở cả hai phía Client và Server RMI đều được thiết kế theo mô hình MVC.

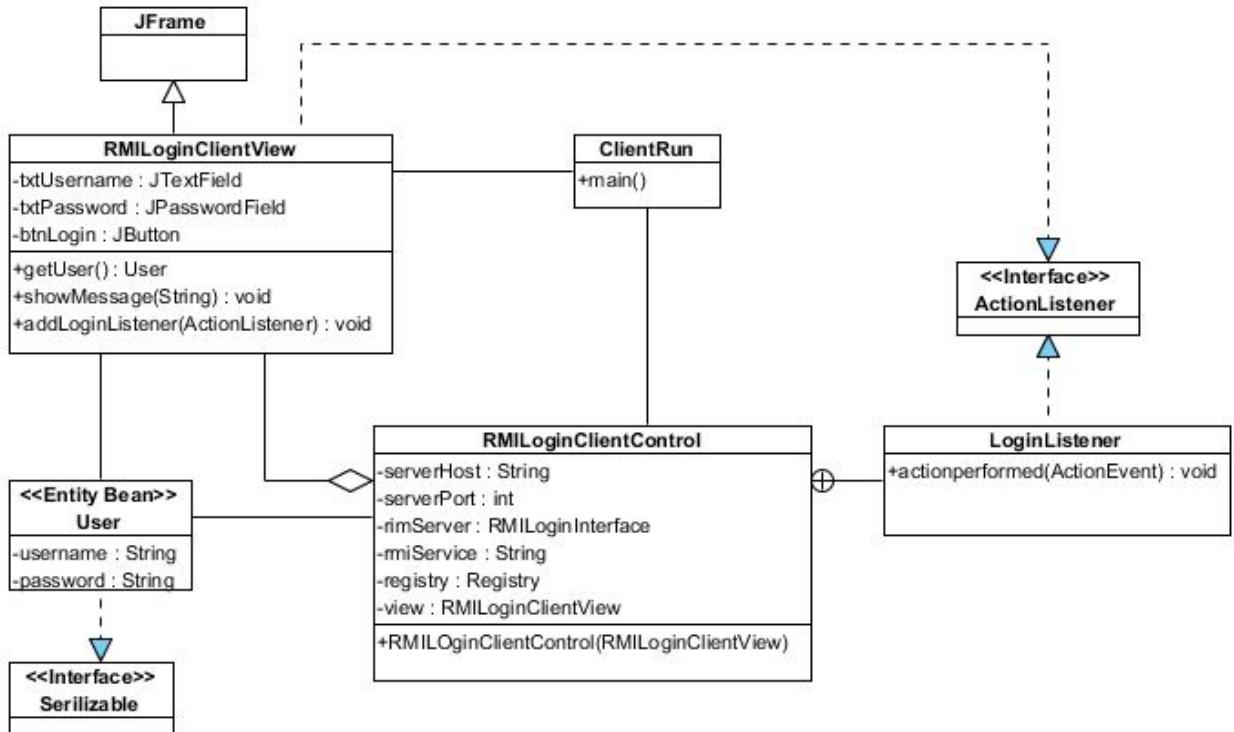
Gợi ý:

Sơ đồ lớp của phía client được thiết kế theo mô hình MVC bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:

- Lớp User: là lớp tương ứng với thành phần model (M), bao gồm hai thuộc tính username và password, các hàm khởi tạo và các cặp getter/setter tương ứng với các thuộc tính.

- Lớp RMILoginClientView: Lớp tương ứng với thành phần view (V), là lớp form nên kế thừa từ lớp JFrame của Java, nó chứa các thuộc tính là các thành phần đồ họa bao gồm ô text nhập username, ô text nhập password, nút nhấp Login.

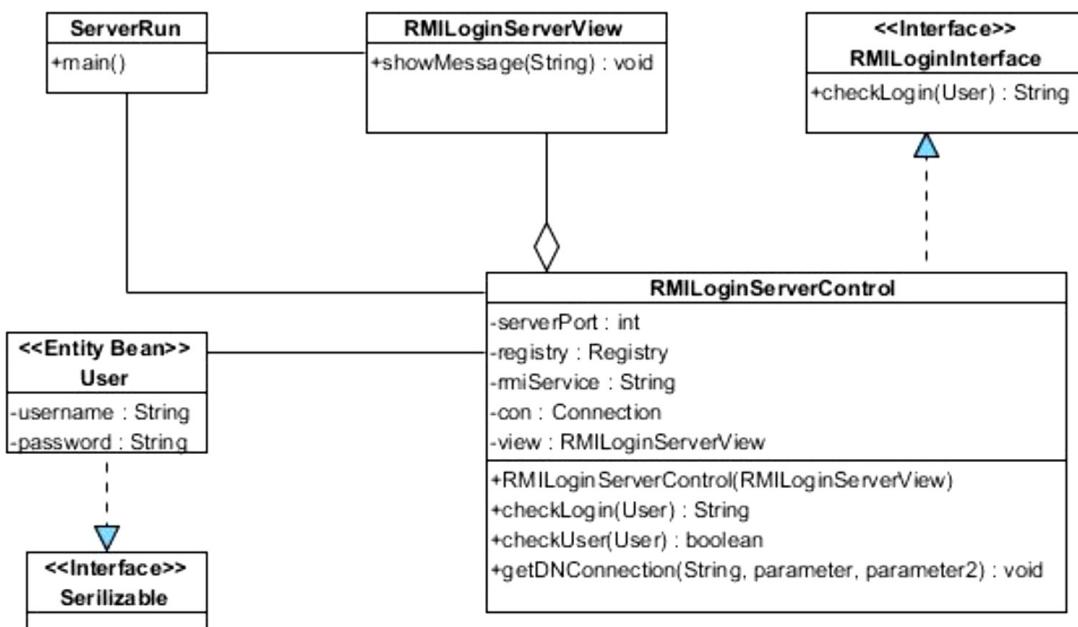
- Lớp RMILoginClientControl: Lớp tương ứng với thành phần control (C), nó chứa một lớp nội tại là LoginListener. Khi nút Login trên tầng view bị click thì sẽ chuyển tiếp sự kiện xuống lớp nội tại này để xử lý. Tất cả các xử lý đều gọi từ trong phương thức actionPerformed của lớp nội tại này, bao gồm: lấy thông tin trên form giao diện, triệu gọi thủ tục từ xa RMI về kiểm tra đăng nhập và yêu cầu form giao diện hiển thị.



Sơ đồ lớp phía RMI Client

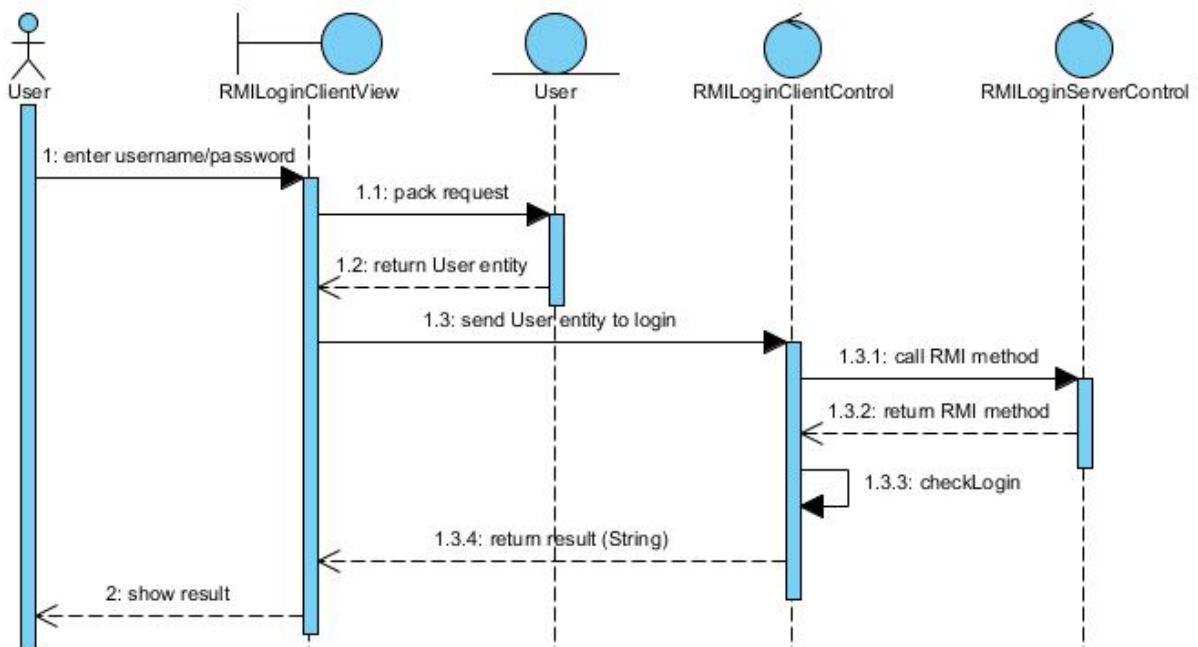
Sơ đồ lớp của phía server được thiết kế theo mô hình MVC bao gồm 3 lớp chính tương ứng với sơ đồ M-V-C như sau:

- Lớp User: là lớp thực thể, dùng chung thông nhất với lớp phía client.
- Lớp RMILoginServerView: Tương ứng với thành phần view (V), là lớp dùng hiển thị các thông báo và trạng thái hoạt động bên server RMI.
- Giao diện RMILoginInterface: Là giao diện (interface) khai báo đối tượng từ xa, trong đó nó khai báo thủ tục **checkLogin()**: Thủ tục nhận vào một tham số kiểu User, trả kết quả về dạng String.
- Lớp RMILoginServerControl: Tương ứng với thành phần control (C), nó đảm nhiệm vai trò xử lý của server RMI, trong đó định nghĩa cụ thể lại phương thức đã được khai báo trong RMILoginInterface, sau đó đăng ký bản thân nó vào server RMI để phục vụ các lời triệu gọi từ phía các client.



Sơ đồ lớp phía Server

Tuần tự các bước thực hiện



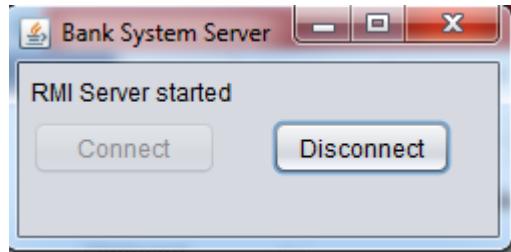
Tuần tự các bước xử lí như sau

- Ở phía client, người dùng nhập username/password và click vào giao diện của lớp RMILoginClientView
- Lớp RMILoginClientView sẽ đóng gói thông tin username/password trên form vào một đối tượng model User bằng phương thức getUser() và chuyển xuống cho lớp RMILoginClientControl xử lí.
- Lớp RMILoginClientControl sẽ triệu gọi làm checkLogin() từ phía server RMI
- Server trả về cho bên client một skeleton của phương thức checkLogin().
- Bên phía client, khi nhận được skeleton, nó gọi phương thức checkLogin() để kiểm tra thông tin đăng nhập.

6. Kết quả kiểm tra sẽ được lớp RMILoginClientControl sẽ chuyển cho lớp RMILoginClientView hiển thị bằng phương thức showMessage()

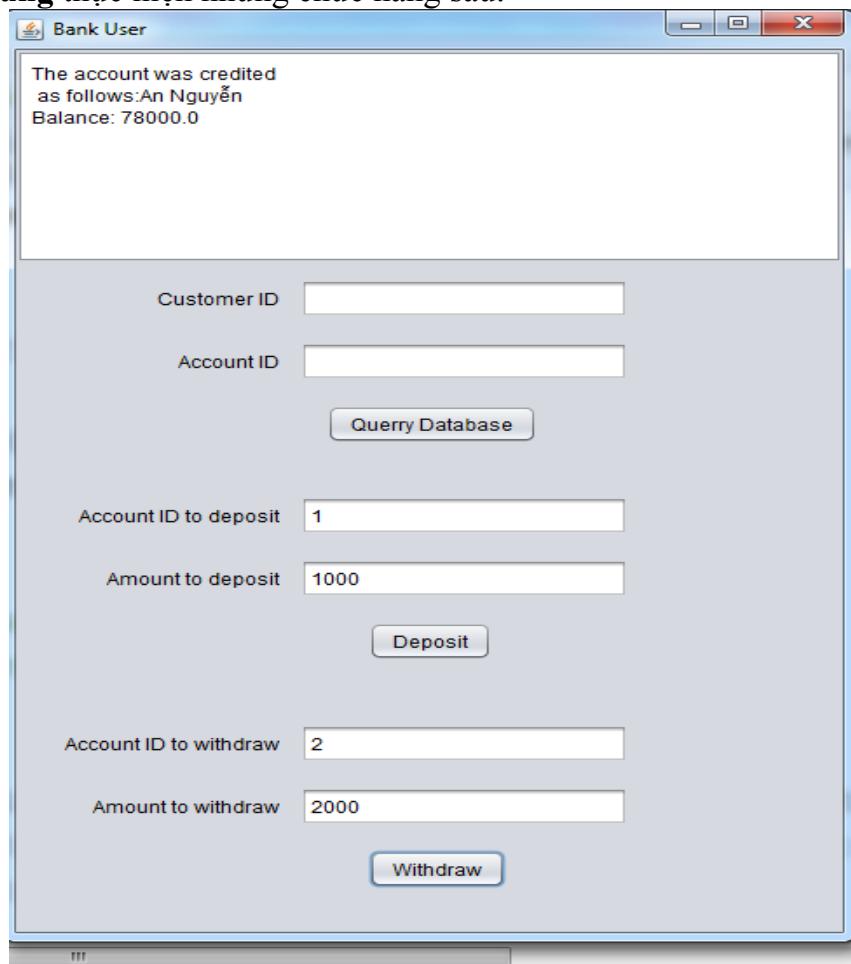
7. Lớp RMILoginClientView hiển thị kết quả đăng nhập lên cho người dùng. Sinh viên vận dụng các kiến thức đã học về lập trình RMI và kết nối CSDL tiến hành viết các lớp tương ứng.

Bài 5. Tổng hợp. Kết hợp kiến thức về RMI, database. Xây dựng hệ thống giao dịch ngân hàng phân tán.



Phía Server Bấm button Connect: Server kết nối với RMI.

Phía người dùng thực hiện những chức năng sau:



Bài toán phát triển hệ thống ngân hàng phân tán đặt ra như sau:

- Cơ sở dữ liệu được lưu trữ và quản lý trên Server, trong đó có 3 bảng:
Bảng Account lưu thông tin tài khoản gồm hai trường IDaccount và Balance

Column Name	Data Type	Allow Nulls
iDAccount	int	<input type="checkbox"/>
Balance	float	<input type="checkbox"/>

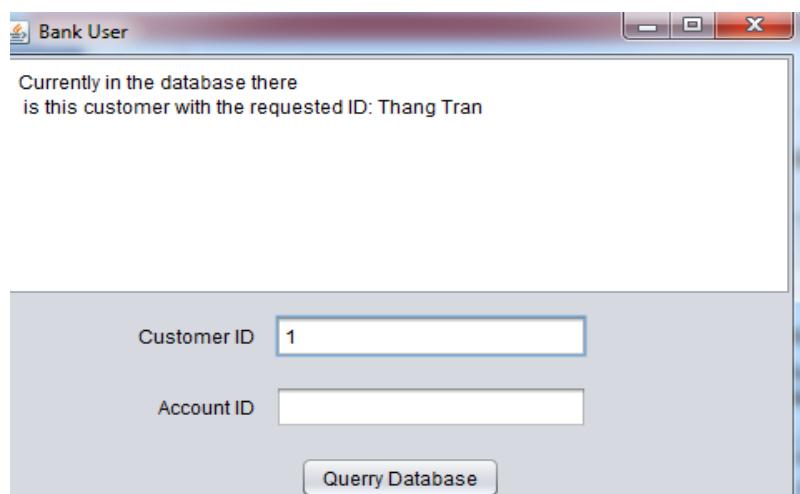
Bảng AccountCustomer lưu thông tin tài khoản khách hàng gồm 2 trường idAccount và IdCustomer

Column Name	Data Type	Allow Nulls
iDAccount	int	<input type="checkbox"/>
IdCustomer	int	<input type="checkbox"/>

Bảng Customer lưu thông tin như sau

Column Name	Data Type	Allow Nulls
idCustomer	int	<input type="checkbox"/>
Name	varchar(50)	<input checked="" type="checkbox"/>
SurName	varchar(50)	<input checked="" type="checkbox"/>

- Chương trình phía Client hiện giao diện đồ họa, trong đó có ô text để nhập CustomerID và AccountID, và một nút QueryDatabase.
- Khi nút QueryDatabase được click, chương trình Client sẽ gửi thông tin đăng nhập trên form giao diện, và gửi sang Server
- Tại phía Server, mỗi khi nhận được thông tin gửi từ Client, sẽ tiến hành kiểm tra trong CSDL xem có tài khoản nào trùng với thông tin nhận được hay không.
- Phía Client, sau khi nhận được kết quả từ Server, sẽ hiển thị thông báo tương ứng



Sau đó người dùng thực hiện các chức năng gửi tiền và rút tiền từ xa.

Hướng dẫn:

Bước 1: Tạo Interface Account

```
public interface Account extends Remote
{
    //Trả về ngân hàng quản lý tài khoản này
```

```

public BankManager getBankManager() throws RemoteException;
    //Trả về Client của tài khoản này
public Client getClient() throws RemoteException;
    //Lấy số dư tài khoản
public float getBalance() throws RemoteException;
    //Chỉnh sửa số dư tài khoản
public void setBalance(float bal) throws RemoteException;
    //phương thức rút tiền có kiểm tra số dư trước khi rút. Trả về amount
public long getCash (long amount) throws NoCashAvailableException,
RemoteException;
    //gửi tiền
public void deposit(float amount) throws RemoteException;
    //rút tiền
public void withdraw(float amount) throws RemoteException;
}

```

Bước 2: Tạo class AccountImpl thực hiện interface Account

```

public class AccountImpl implements Account, Serializable
{
    private BankManager bankManager;
    private Client Client;
    private float balance;
    private String accountNumber;

    public AccountImpl (BankManager bankManager, Client Client, String
accountNumber, float bal)
    {
        this.bankManager = bankManager;
        this.Client = Client;
        this.balance = bal;
        this.accountNumber = accountNumber;
    }

    public void deposit(float amount)
    {
        balance += amount;
    }
}

```

```
public void withdraw(float amount)
{
    balance -= amount;
}

public BankManager getBankManager() throws RemoteException
{
    return bankManager;
}

public Client getClient() throws RemoteException
{
    return Client;
}

public float getBalance() throws RemoteException
{
    return balance;
}

public void setBalance(float bal) throws RemoteException
{
    balance = bal;
}

public long getCash(long amount) throws NoCashAvailableException,
RemoteException
{
    if (amount > balance)
    {
        throw new NoCashAvailableException();
    }
    balance = balance - amount;
    return amount;
}
```

```
}
```

Bước 3: Tạo interface BankManager

```
public interface BankManager extends Remote
{
    //Phương thức lấy tài khoản theo mã
    public Account getAccount(String accountNumber) throws RemoteException;
    //Phương thức lấy Client theo tên
    public Client getClient(String ClientName) throws RemoteException;
    //phương thức lấy mã khách hàng theo mã tài khoản
    public int getCustomerId(int accountId) throws RemoteException;
    //phương thức gửi tiền vào tài khoản
    public void deposit(String idAccount, float Amount) throws
RemoteException;
    //phương thức rút tiền
    public void withdraw(String idAccount, float Amount) throws
RemoteException;
}
```

Bước 4: Tạo class BankManagerImpl thực hiện interface BankManager

```
public class BankManagerImpl extends UnicastRemoteObject implements
BankManager
{
    private Hashtable accounts;
    private Hashtable Clients;
    private Connection conn;
    private Statement s;
    private int CustomerID;
    public BankManagerImpl() throws RemoteException
    {
        super();
        initialize();
    }
    public Account getAccount(String accountNumber) throws RemoteException
    {
        AccountImpl account = (AccountImpl)accounts.get(accountNumber);
        return account;
    }
    public Client getClient(String ClientID) throws RemoteException
    {
        ClientImpl Client = (ClientImpl)Clients.get(ClientID);
```

```

    return Client;
    //Phương thức rút tiền
public void deposit(String idAccount, float amount) throws RemoteException
{
    Account theAccount = (Account)accounts.get(idAccount);
    theAccount.deposit(amount);
    accounts.remove(idAccount);
    accounts.put(idAccount, theAccount);
    try
    {
        Statement s = conn.createStatement();
        String sql = "Update account Set Balance ='" +
theAccount.getBalance() +"' where idAccount = '" + idAccount +"'";
        s.executeUpdate(sql);
        /// update in the database now
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
public void withdraw(String idAccount, float amount) throws
RemoteException
{
    Account theAccount = (Account)accounts.get(idAccount);
    theAccount.withdraw(amount);
    accounts.remove(idAccount);
    accounts.put(idAccount, theAccount);
    try
    {
        Statement s = conn.createStatement();
        String sql = "Update account Set Balance ='" +
theAccount.getBalance() +"' where idAccount = '" + idAccount +"'";
        s.executeUpdate(sql);
        /// update in the database now
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

```

```

}

public void getClientsFromDatabase()
{
    public void initialize() throws java.rmi.RemoteException
    {
        // Create the hashtables
        accounts = new Hashtable(20);
        Clients = new Hashtable(10);
        CreateConnection();
        getCustomersFromDatabase();
        getAccountsFromDatabase();
    }

    public boolean initializeConnection(String SERVER, String
DATABASE, String USER_ID, String PASSWORD) throws ClassNotFoundException,
SQLException
    {
        try
        {
            String connString = "jdbc:sqlserver://" + SERVER +
":1433;integratedSecurity=true;databaseName=" + DATABASE;
            conn = DriverManager.getConnection(connString);
            s = conn.createStatement();
            return true;
        }
        catch (SQLException e)
        {
            return false;
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return false;
        }
    }
}

public void CreateConnection()
{
    if(conn == null)
        try
        {
            initializeConnection("localhost", "QLNH", "root", "");
        }
}

```

```

        }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public int getCustomerId(int idAccount)
{
    ArrayList<Integer> ids = new ArrayList<Integer>();
try
{
    Statement s = conn.createStatement();
    String sql = "Select IdCustomer from accountCustomer where
idAccount ='" + idAccount + "'";
    ResultSet r = s.executeQuery(sql);
    while(r.next())
    {
        ids.add(r.getInt("IdCustomer"));
    }
}
catch (Exception ex)
{
    ex.printStackTrace();
}
return ids.get(0).intValue();
}

public void getCustomersFromDatabase()
{
try
{
    Statement s = conn.createStatement();
    String sql = "Select * from customer";
    ResultSet r = s.executeQuery(sql);
    while(r.next())
    {
        int idCustomer = r.getInt("IdCustomer");
        String name = r.getString("Name");
        String surname = r.getString("Surname");
        Client newClient = new ClientImpl(this, name + " " + surname);
        Clients.put(String.valueOf(idCustomer), newClient);
    }
}

```

```

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

public void getAccountsFromDatabase()
{
    System.out.println("-----");
    System.out.println("Reading accounts from the database:");
    try
    {
        int counter = 0;
        Statement s = conn.createStatement();
        Statement s1 = conn.createStatement();
        String sql = "Select * from accountcustomer";
        ResultSet r = s.executeQuery(sql);
        while(r.next())
        {
            int idCustomer = r.getInt("IdCustomer");
            int idAccount = r.getInt("idAccount");
            Client theClient = (ClientImpl)Clients.get(String.valueOf(idCustomer));
            Account newAccount = new
                AccountImpl(this,theClient,String.valueOf(idAccount),0);
            accounts.put(String.valueOf(idAccount), newAccount);
            System.out.println("Customer:" + newAccount.getClient().getName() + " "
- Account:" + String.valueOf(idAccount));
            counter++;
        }
        for(int i=1;i<=counter;i++)
        {
            if(accounts.containsKey(String.valueOf(i)))
            {
                sql = "Select Balance from account where idAccount = '" + i + "'";
                ResultSet r1 = s1.executeQuery(sql);
                r1.next();
                float balance = r1.getFloat("Balance");
                Account theAccount = (Account)accounts.get(String.valueOf(i));
                theAccount.setBalance(balance);
                accounts.remove(String.valueOf(i));
            }
        }
    }
}

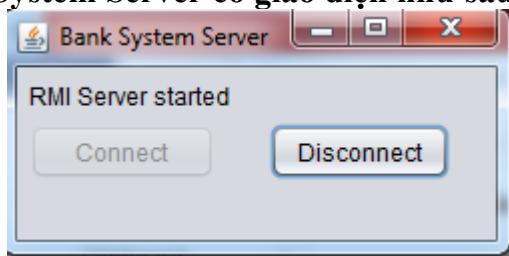
```

```

        accounts.put(String.valueOf(i),theAccount);
    }
}
s.close();
}
catch (Exception ex)
{
    ex.printStackTrace();
}

```

Bước 5:Tạo Form BankSystem Server có giao diện như sau



Thực hiện code cho button Connect và Disconnect

Phương thức ConnectServer

```

private void ConnectServer()
{.....
m_bankManager = new BankManagerImpl();
if(m_Registry == null)
    m_Registry = LocateRegistry.createRegistry(1234);
    m_Registry.rebind("BankSystem", m_bankManager);
    m_connected = true;
.....
}

```

Phương thức DisconnectServer()

```

private void DisconnectServer()
{
    m_bankManager = null;
    m_Registry.unbind("BankSystem");
    m_connected = false;
}

```

Về phía người dùng tạo interface Client

```

public interface Client extends Remote
{
    //Phương thức trả về ngân hàng quản lý Client này
    public BankManager getBankManager() throws RemoteException;
        //Phương thức trả về tên của Client
    public String getName() throws RemoteException;
}

```

Tạo ClientImpl thực hiện interface Client

```

public class ClientImpl implements Client, Serializable
{
    private BankManager bankManager;
    private String ClientName;
    public ClientImpl(BankManager bm, String name)
    {
        this.bankManager = bm;
        this.ClientName = name;
    }
    public BankManager getBankManager() throws RemoteException
    {
        return bankManager;
    }
    public String getName() throws RemoteException
    {
        return ClientName;
    }
}

```

Tạo Jframe BanhkUser có giao diện như yêu cầu đề bài:

```

public class BankUser extends javax.swing.JFrame {
    public BankUser() {
        initComponents();
    }
    // Code chức năng mở cửa sổ
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try
        {
            m_Registry = LocateRegistry.getRegistry("localhost" ,1234);
            m_bankManager = (BankManager) m_Registry.lookup("BankSystem");
        }
        catch (NotBoundException notBoundException)
    }
}

```

```

    {
        System.out.println("Not Bound: " + notBoundException);
    }
    catch (RemoteException remoteException)
    {
        System.out.println("Remote Exception: " + remoteException);
    }
}

```

Code cho button QueryDatabase

```

private void btnQrDatabaseActionPerformed(java.awt.event.ActionEvent evt)
{
    try
    {
        if(txtCustomerID.getText().length() > 0)
        {
            System.out.println("opps: " + txtCustomerID.getText() + "000");
            Client Client = m_bankManager.getClient(txtCustomerID.getText());
            txtCustomerID.setText("");
            System.out.println("Currently in the database there \n is this
customer with the requested ID: " + Client.getName());
            txa_printArea.setText("Currently in the database there \n is this
customer with the requested ID: " + Client.getName());
        }
        else
        {
            if(txtAccountID.getText().length() > 0)
            {
                Account account = m_bankManager.getAccount(txtAccountID.getText());
                txtAccountID.setText("");
                System.out.println("Currently in the database there is \n this
account with the requested ID: " + account.getClient().getName());
                txa_printArea.setText("Currently in the database there is \n this
account with the requested ID: " + account.getClient().getName() + "\n
Balance: " + account.getBalance());
            }
        }
    }
    catch (RemoteException remoteException)
    {
    }
}

```

Code cho button Deposit

```
private void btnDepositActionPerformed(java.awt.event.ActionEvent evt) {  
    try  
    {  
        if(txtAccIDtoDeposit.getText().length() > 0 &&  
txtAmountToDeposit.getText().length() > 0)  
        {  
            m_bankManager.deposit(txtAccIDtoDeposit.getText(),  
Float.parseFloat(txtAmountToDeposit.getText()));  
            Account account =  
m_bankManager.getAccount(txtAccIDtoDeposit.getText());  
            txa_printArea.setText("The account was credited \n as follows:" +  
account.getClient().getName() + "\nBalance: " + account.getBalance());  
        }  
    }  
    catch (RemoteException remoteException)  
    {  
        System.err.println(remoteException);  
    }  
}
```

Code cho button WithDraw

```
private void btnWithDrawActionPerformed(java.awt.event.ActionEvent evt) {  
    try  
    {  
        if(txtAccIDtoWithDraw.getText().length() > 0 &&  
txtAmountToWithDraw.getText().length() > 0)  
        {  
            m_bankManager.withdraw(txtAccIDtoWithDraw.getText(),  
Float.parseFloat(txtAmountToWithDraw.getText()));  
            Account account = m_bankManager.getAccount(txtAccIDtoWithDraw.getText());  
            txa_printArea.setText("The account was credited \n as follows:" +  
account.getClient().getName() + "\nBalance: " + account.getBalance());  
        }  
    }  
    catch (RemoteException remoteException)  
    {  
        System.err.println(remoteException);  
    }  
}
```

TÀI LIỆU THAM KHẢO

- [1]. Cay S. Horstmann, *Core Java Volum I - Fundamentals, Tenth Edition*, NewYork : Prentice Hall, 2016.
- [2]. Cay S. Horstmann. *Core Java Volum II - Advanced Features, Tenth Edition*, New York : Prentice Hall, 2017.
- [3]. Eng.haneen Ei-masry, *Java database connection*, Islamic University of Gaza Faculty of Engineering Department of Computer Engineering ECOM 4113: DataBase Lab, 2014.
- [4]. Angelos Stavrou, *Advanced Network Programming Lab using Java*, Network Security, ISA 656, Angelos Stavrou.
- [5]. Marenglen Biba, Ph.D, *Manual for Lab practices, Remote Method Invocation Three Tier Application with a Database Server*, Department of Comsputer Science, University of New York.
- [6]. Elliotte Rusty Harold, *Java Network Programming, Fourth Edition*, O'Reilly Media, 2013.
- [7]. Đoàn Văn Ban, *Lập trình hướng đối tượng với JAVA*, Nhà xuất bản Khoa học và Kỹ thuật, 2005.
- [8]. ThS. Dương Thành Phết, *Bài tập thực hành Chuyên đề 1 CNPM- Java*, Khoa CNTT- Trường ĐH Công nghệ TP.HCM.
- [9]. <https://www.oracle.com/technetwork/java/socket-140484.html#>
- [10]. https://personales.unican.es/corcuerp/java/Labs/LAB_22.htm
- [11]. <http://www.nrcmec.org/pdf/Manuals/CSE/student/2-2%20java16-17.pdf>
- [12]. <http://cse.mait.ac.in/pdf/LAB%20MANUAL/JAVA.pdf>
- [13]. https://www.academia.edu/35283541/Bài_tập_môn_lập_trình_hướng đối_tượng

