TÀI LIỆU THỰC TẬP LẬP TRÌNH MẠNG: LAB 04

DANH MỤC THUẬT NGỮ TIẾNG ANH

Từ	Nghĩa của từ
abstract	Trừu 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	Kế 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

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 lẫn 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 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

Bước 2: Tạo Class DanhSachSinhvien

Bước 3: Tạo Class TestSinhvien

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.

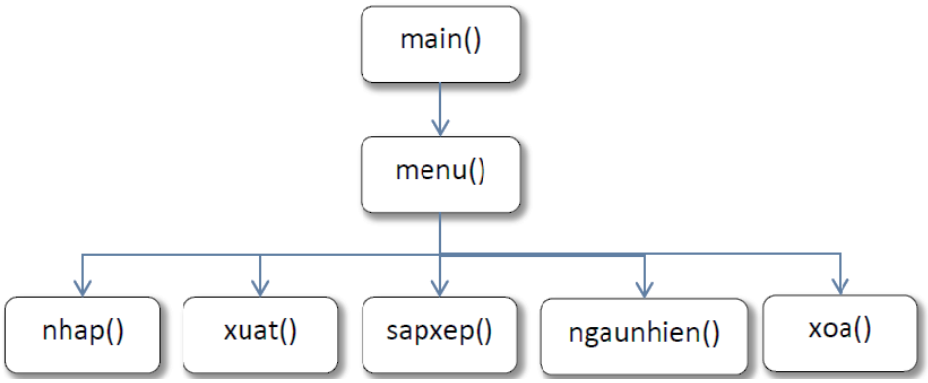
```
public class TestSinhvien {
     public static void main(String[] args) {
      DanhSachSinhvien qlsv=new DanhSachSinhvien();
      Sinhvien teo=new Sinhvien();
      teo.setMasv("113");
      teo.setTensv("Nguyễn Văn Tèo");
      qlsv.addSinhvien(teo);
      Sinhvien ty=new Sinhvien();
      ty.setMasv("114");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

```
        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 ("Nhap them 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.

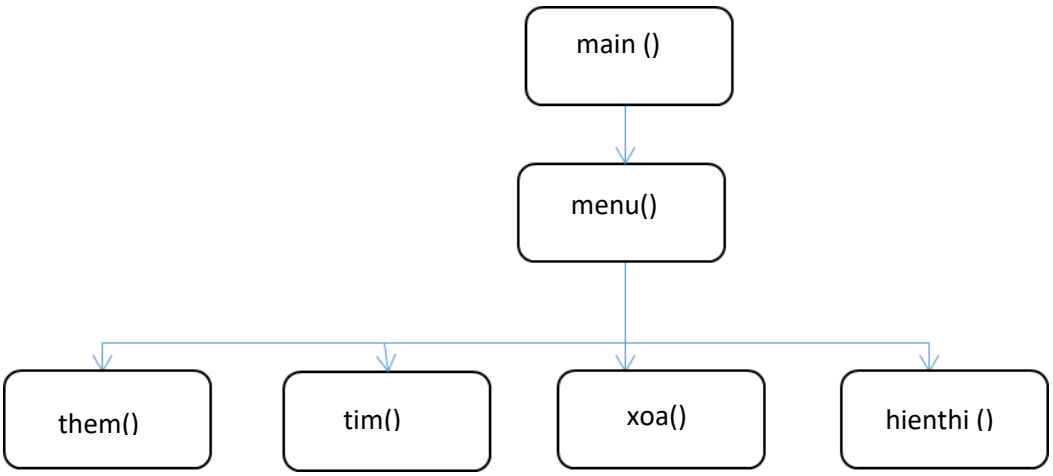
Xây dựng lớp EmployeeList sử dụng ArrayList. Lớp Employee gồm các trường (Mã NV, Tên NV, Chức vụ, Lương, Thời gian bắt đầu làm)

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 sanh 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\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:

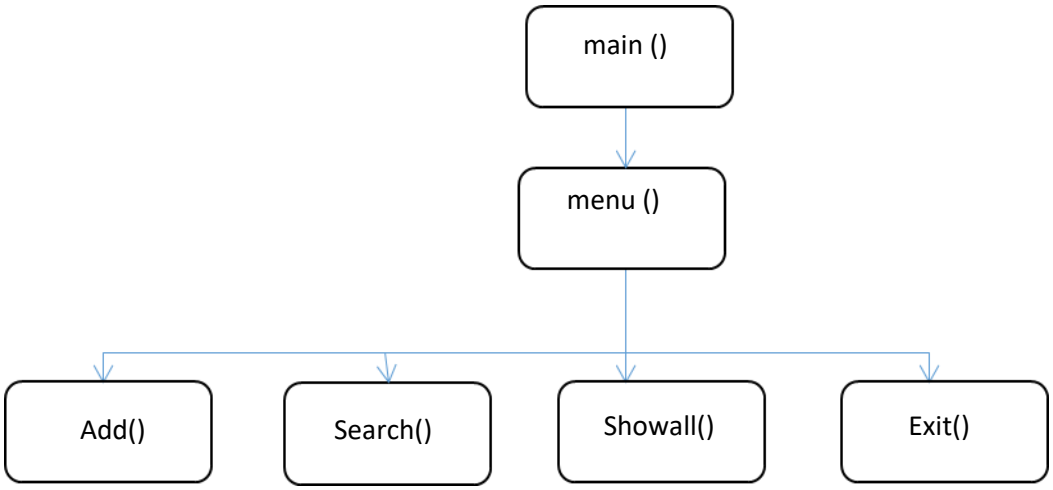
Bước 1: Tạo lớp DoctorDetails

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 sỹ tìm thấy:");
            System.out.println(elements.get(code).toString());
        }
    }
public void ShowAll()
    {
        System.out.println("*******Thông tin tất cả bác sỹ***********");
        Iterator it = elements.entrySet().iterator();
        int index = 1;
        while(it.hasNext())
        {
            System.out.println("Bác sỹ " + 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)
	{
		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("\Sau 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 binh 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 binh 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: Tạp 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>();
	}
	// Them tu moi vao tu dien
	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.

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