

BÀI TẬP MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

(OBJECT ORIENTED PROGRAMMING EXERCISES)

HỆ: ĐẠI HỌC

MỤC LỤC

Chương 1. Tổng quan về cách tiếp cận hướng đối tượng.....	3
Chương 2. Những khái niệm cơ bản của lập trình hướng đối tượng.....	3
Chương 3. Giới thiệu về Java	3
Chương 4. Kế thừa và đa hình trên Java.....	36
Chương 5. Tập hợp trên Java.....	47
Chương 6. Lập trình Generics	50
Chương 7. ÔN TẬP VÀ KIỂM TRA THỰC HÀNH	70
Chương 8. Nhập xuất trên Java	71

Tuần 1. LÀM QUEN VỚI NGÔN NGỮ LẬP TRÌNH JAVA

Chương 1. Tổng quan về cách tiếp cận hướng đối tượng

Chương 2. Những khái niệm cơ bản của lập trình hướng đối tượng

Chương 3. Giới thiệu về Java

Mục tiêu:

- Làm quen với ngôn ngữ lập trình Java
- Làm quen với công cụ lập trình Java (Eclipse hoặc Jcreator/NetBeans)
- Hiểu được cấu trúc 1 chương trình Java, cách biên dịch và chạy chương trình dùng ngôn ngữ lập trình Java
- Hiểu và áp dụng được nhập xuất dữ liệu, các toán tử trong ngôn ngữ lập trình Java.
- Hiểu và áp dụng được các cấu trúc điều khiển, cấu trúc lặp trong ngôn ngữ lập trình Java

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: D:\MaSV_HoTen\Tuan01\ hoặc T:\MaSV_Hoten\Tuan01
- Máy tính phải được cài đặt sẵn JDK (Java Development Kit)
- Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)

PHẦN THIẾT LẬP MÔI TRƯỜNG LÀM VIỆC

Bài 1. Cấu trúc của 1 chương trình viết bằng ngôn ngữ lập trình Java

```
package packageName;                // 1. Khai báo tên gói nếu cần

import java.util.Scanner;            // 2. Khai báo thư viện có sẵn nếu cần dùng

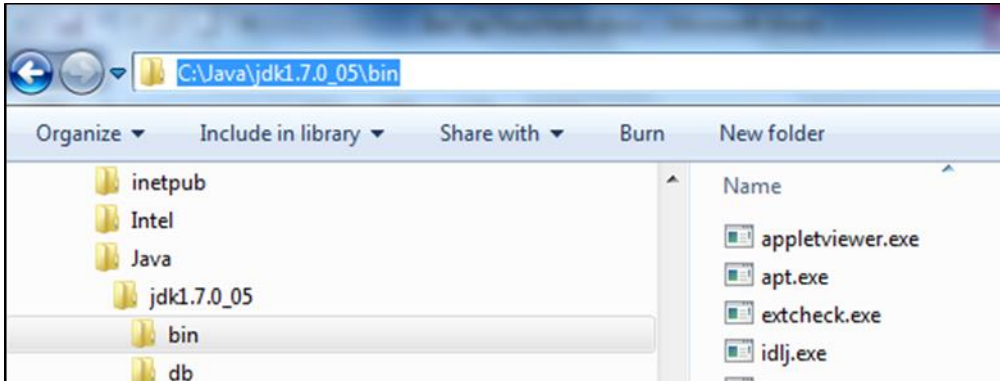
public class ClassName               // 3. Khai báo tên lớp
{
    /* các ghi chú liên quan */
    int var;                          // Khai báo biến của lớp
    public void methodName()         // 4. Khai báo tên phương thức và tham số
    {
        /* phần thân của phương thức */
        // Các lệnh thực hiện cho mục tiêu phương thức
    }
    public static void main(String[] args) // 5. Hàm chính để chạy
    {
        /* hàm chính */
    }
}
```

Bài 2. Thao tác biên dịch và chạy chương trình Java trên Console.

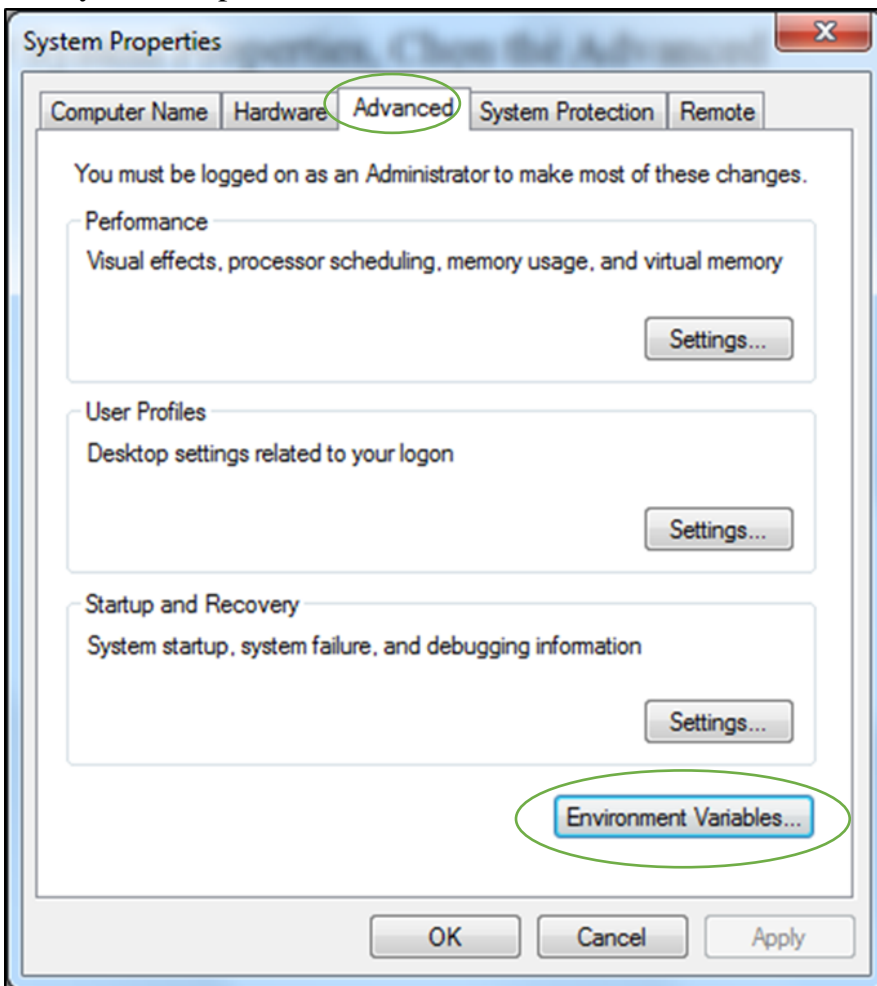
Dùng Notepad (hoặc Notepad++) soạn thảo code và dùng Console biên dịch → chạy chương trình (*javac* và *java* của JDK).

Thiết lập biến môi trường để chạy java ở cơ chế command -line

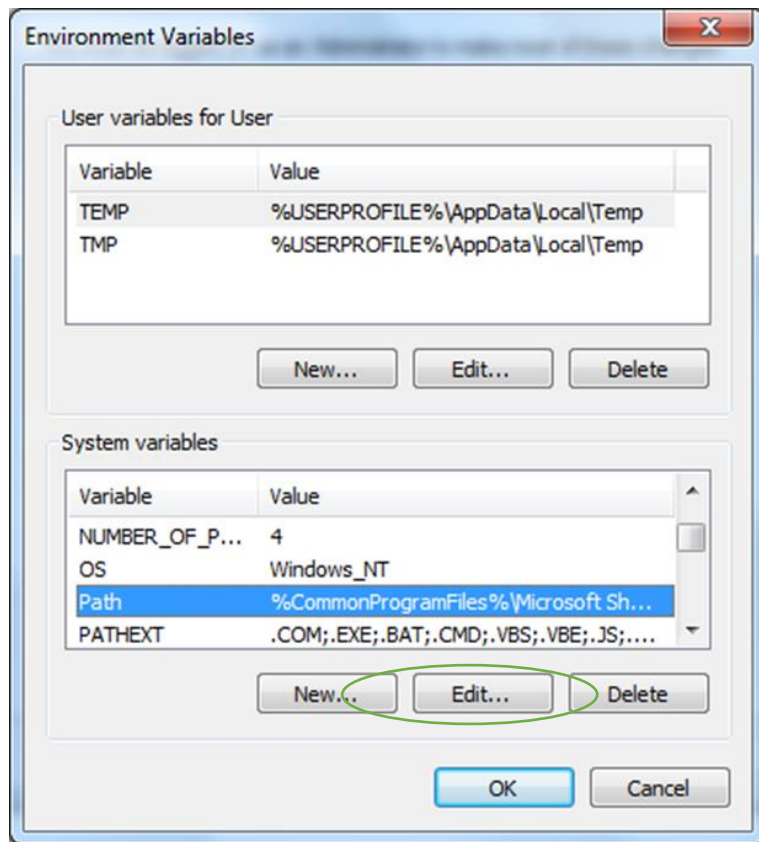
Khởi động Windows Explorer, copy đường dẫn tới thư mục bin khi cài đặt java



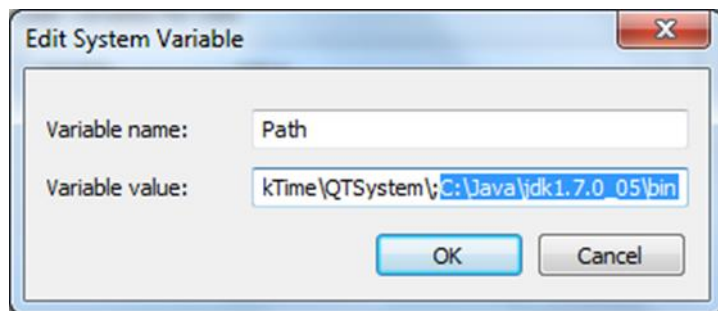
Mở System Properties, Chọn thẻ Advanced



Nhấn nút “Environment Variables...”

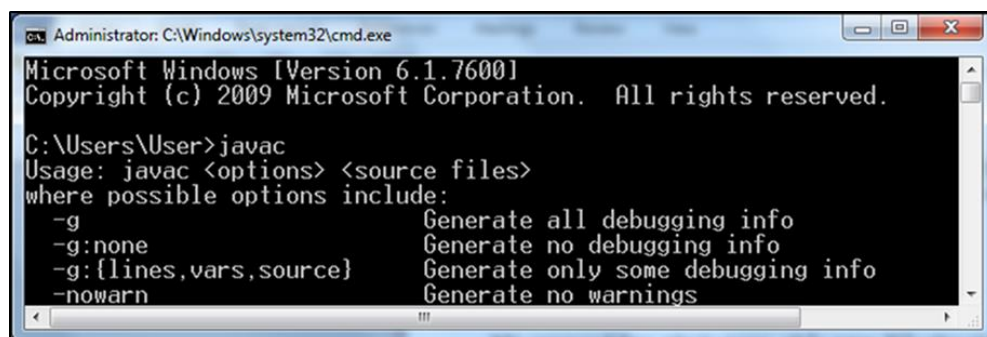


Tìm đến biến Path, và nhấn nút “Edit...”. Đưa con trỏ vào cuối dòng, gõ dấu ;. Rồi paste đường dẫn tới thư mục bin khi cài đặt java vào.

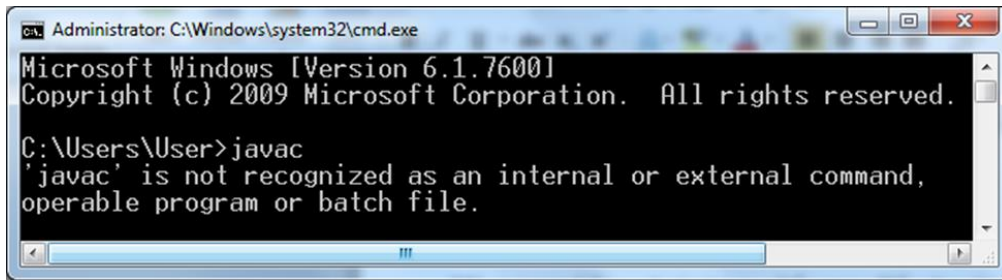


Nhấn nút OK liên tiếp để đóng các cửa sổ đã mở ra.

Mở command-line, gõ vào javac, nhấn enter. Nếu như cửa sổ chạy ra có dạng như hình là OK.



Nếu ra thông báo kiểu: 'javac' is not recognized as an internal or external command, operable program or batch file, như hình.



Thì coi như thiết lập sai, cần phải làm lại.

Hoặc thực hiện các bước sau:

Trong cửa sổ console thực hiện các bước

B1: Set Path=%PATH%;C:\Program Files\Java\Jdk1.5\bin

B2: Chuyển về thư mục *D:\MaSV_HoTen\Tuan01*

B3: Biên dịch (dùng lệnh *javac TenTapTin.java*, nếu quá trình biên dịch không có lỗi sẽ phát sinh tập tin *TenTapTin.class* trong cùng thư mục) và chạy chương trình (dùng lệnh *java TenTapTin*)

Bài 3. Thao tác tạo Project, tạo tập tin java, biên dịch tập tin dùng JCreator 5.0 hoặc Eclipse.

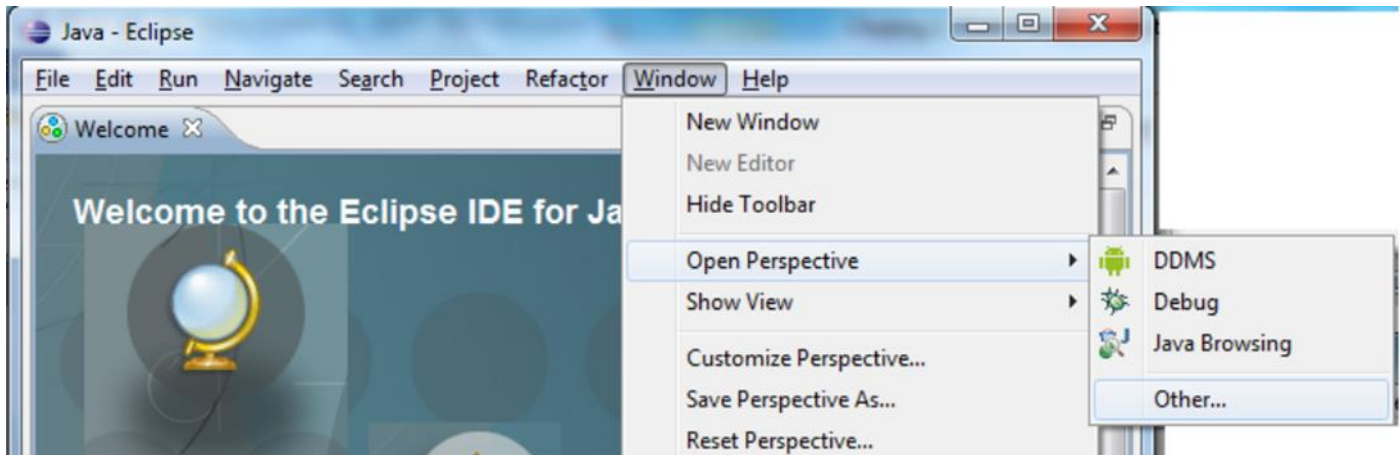
Sử dụng Eclipse IDE

A. Khởi động Eclipse

1. Khởi động Eclipse JSE



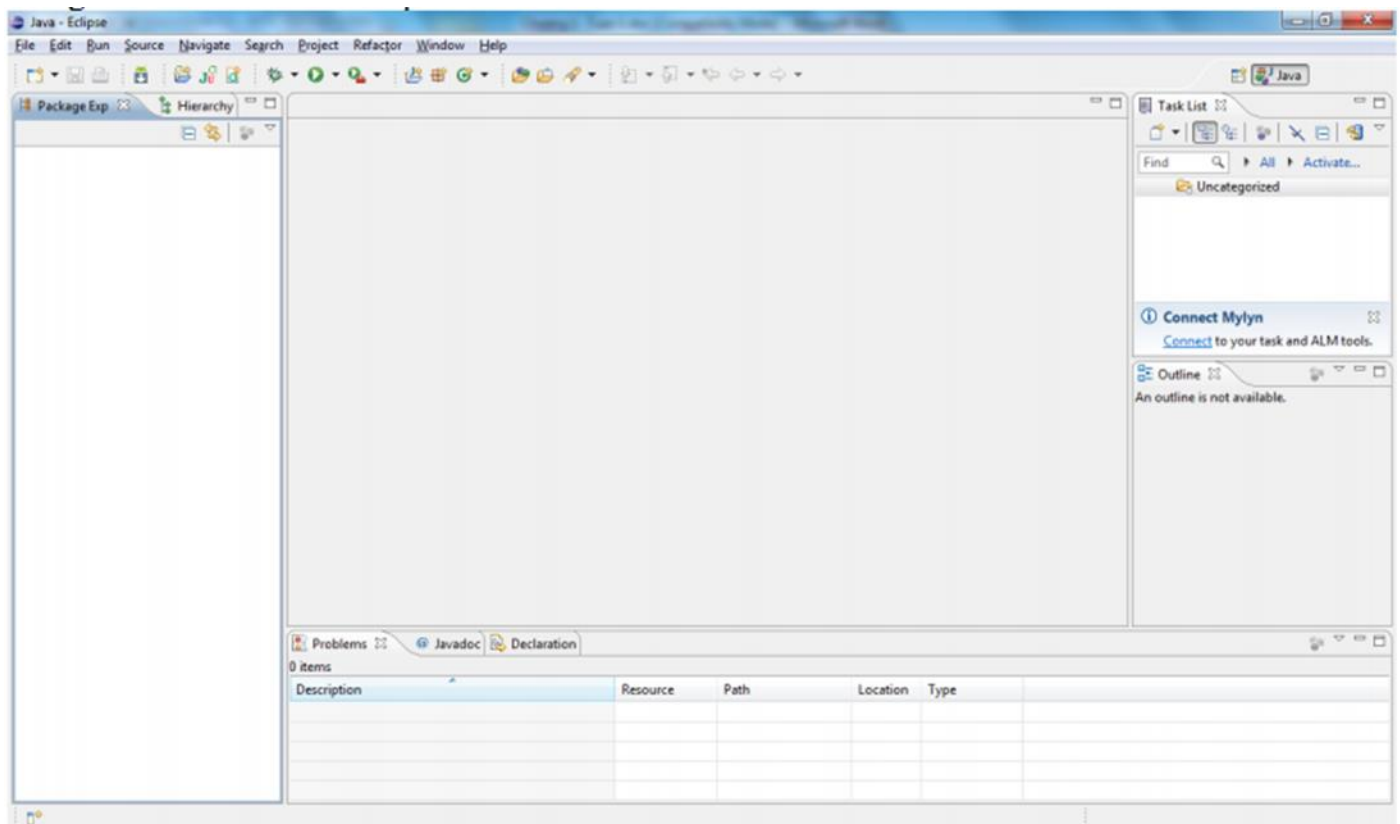
2. Chọn Perspective



Chọn Perspective Java(Default)

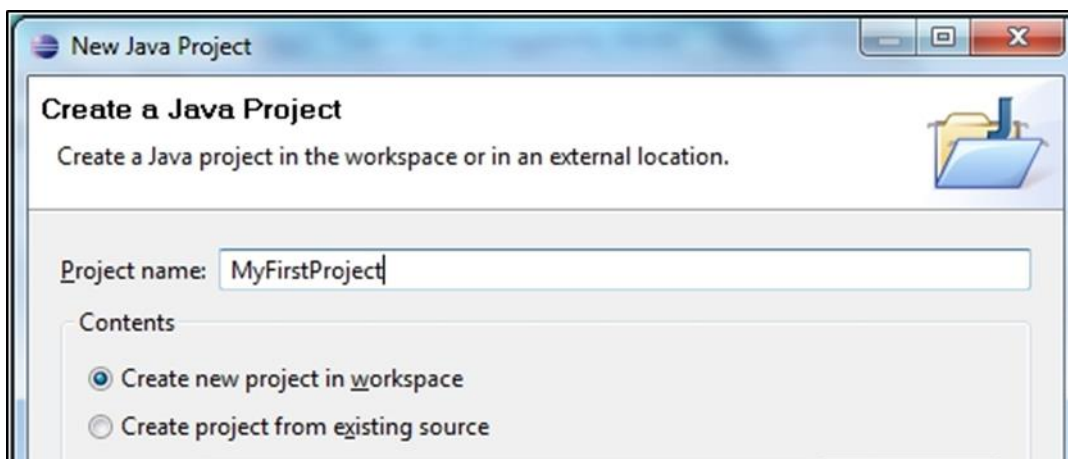
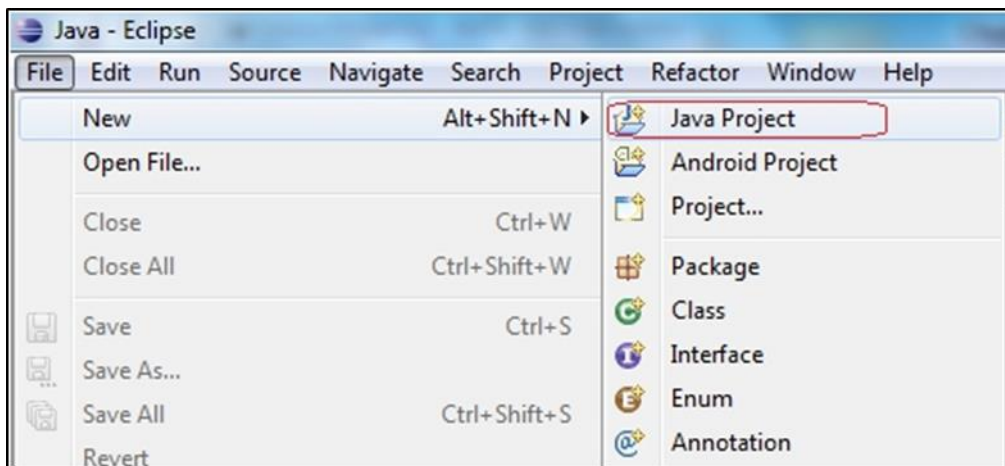


Đóng Welcome screen. Kết quả

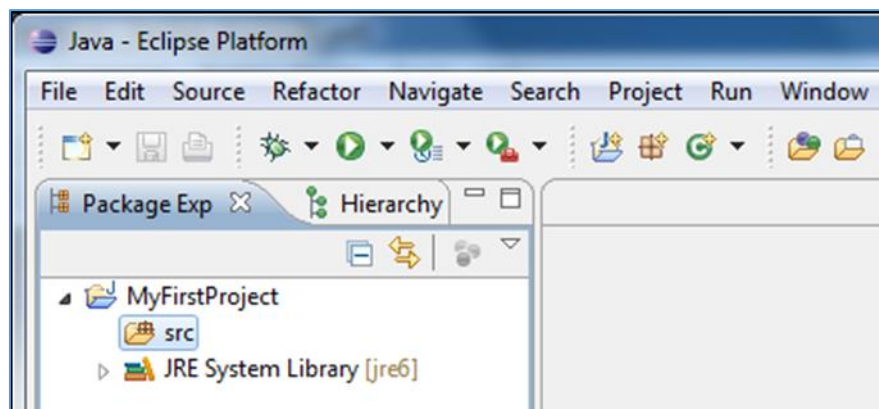


B. Tạo Project trong Eclipse

1. Tạo project mới: Menu File->New->Java Project



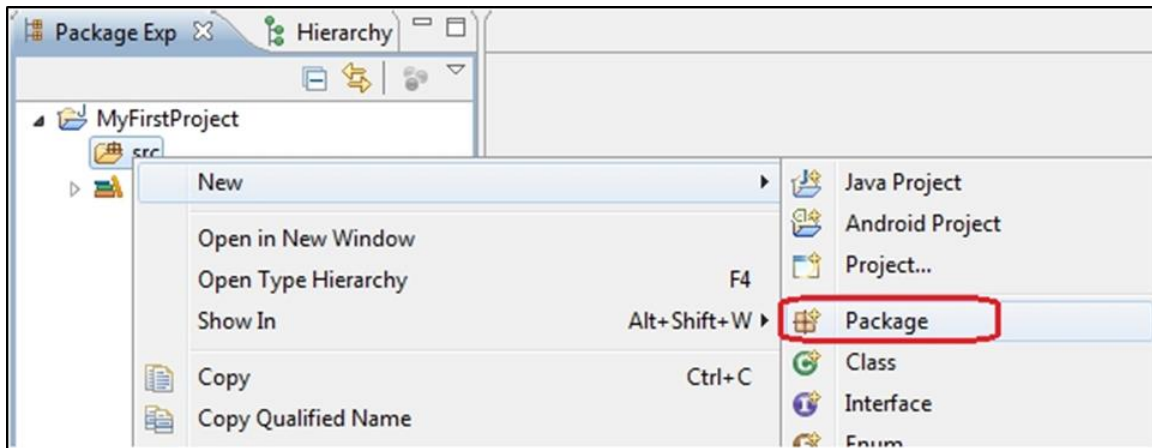
Nhấn Finish. Kết quả trong Project Explorer



2. Viết code:

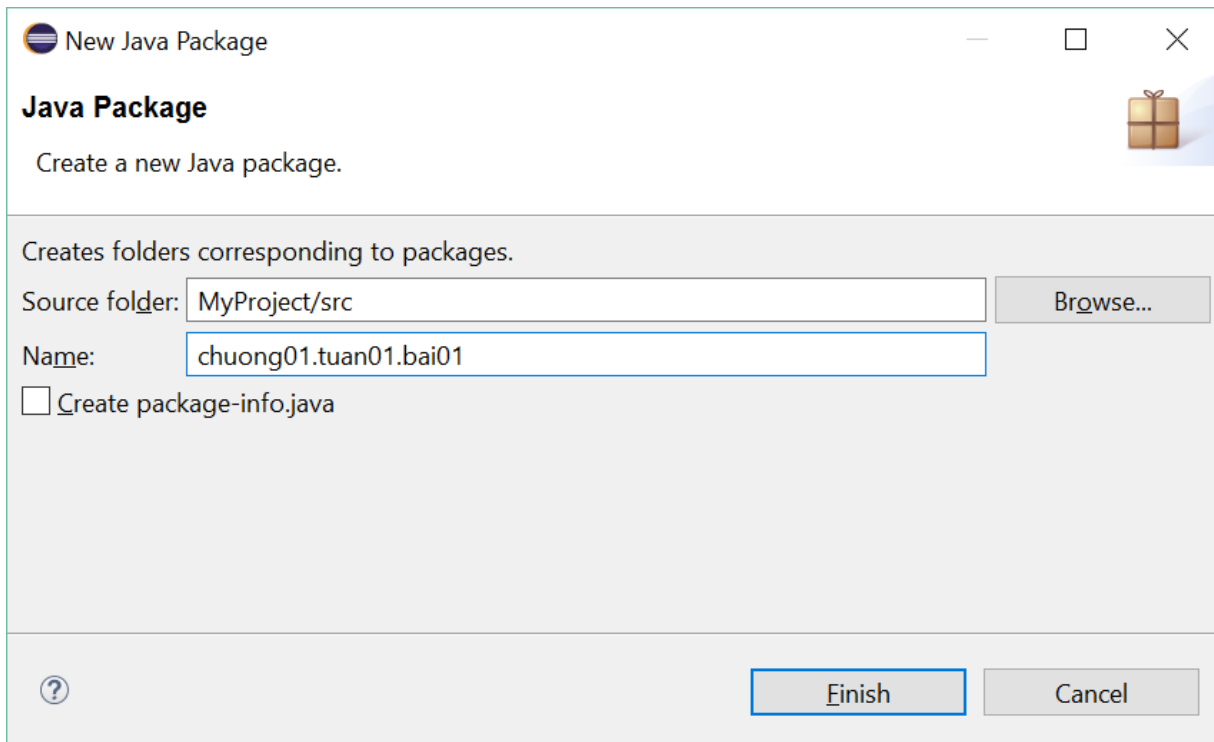
Lưu ý NÊN tạo các package để lưu trữ các lớp java. Package cho phép lưu trữ các class của ứng dụng theo nhóm (các lớp quan hệ gần thì lưu trong cùng package)

Mỗi ứng dụng có thể có 1 hoặc nhiều package. Mỗi package chứa một hoặc nhiều class

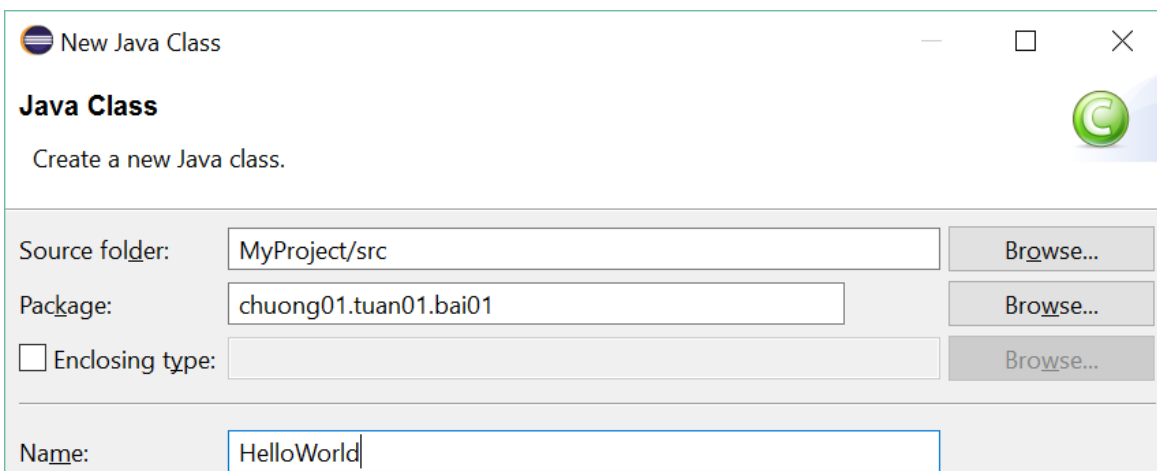
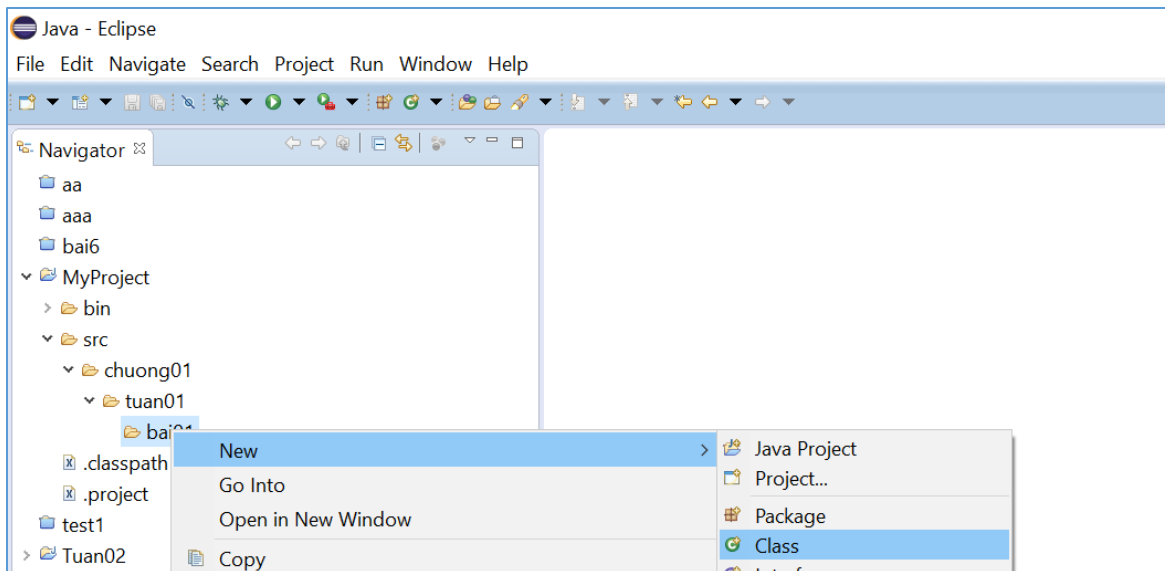


Đặt tên theo kiểu: a.b.c trong đó các ký tự là tên bất kỳ. Ví dụ: chuong01.tuan01.bai01 Điều đó có nghĩa là Eclipse sẽ tạo cho bạn 3 thư mục: chuong01\tuan01\.

Lưu ý: các gói luôn được đặt tên bằng chữ thường.

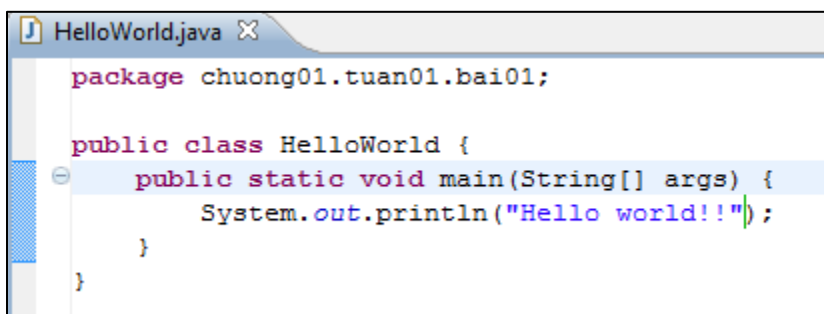


Tạo lớp mới bằng cách nhấn phải chuột lên package cần thêm lớp vào, chọn New → Class



Chú ý: Tên lớp luôn bắt đầu bằng 1 ký tự hoa. Đặt theo kiểu Title-Case

Bắt đầu viết code. Eclipse hỗ trợ cơ chế code completion rất tốt. Các bạn luôn nhớ phím Ctrl-SpaceBar để Eclipse hiện lên các suggestion.

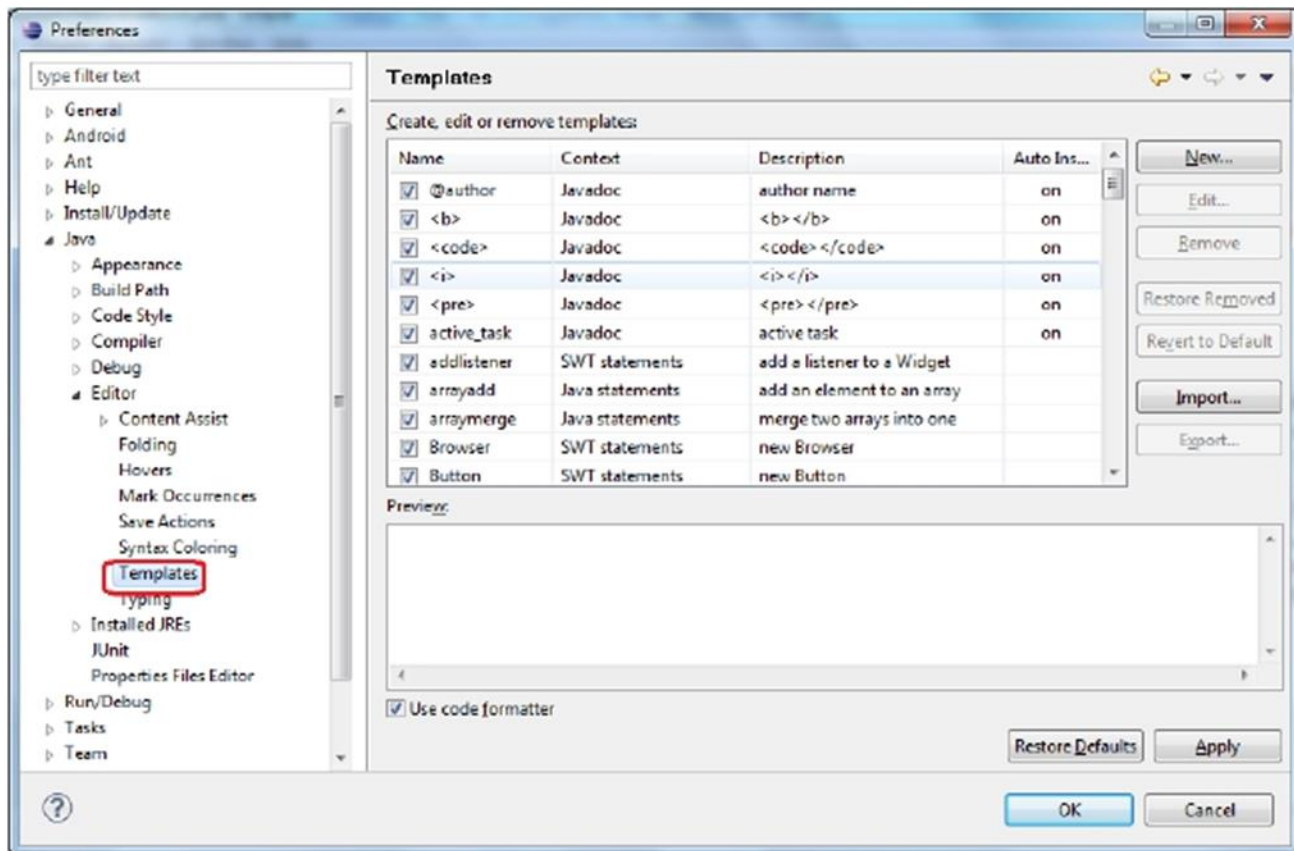


Một số editing template hay dùng:

Gõ sysout sau đó nhấn Ctrl-Spacebar sẽ cho `System.out.println();`

Gõ main sau đó nhấn Ctrl-Spacebar sẽ cho `public static void main(String[] args) {}`

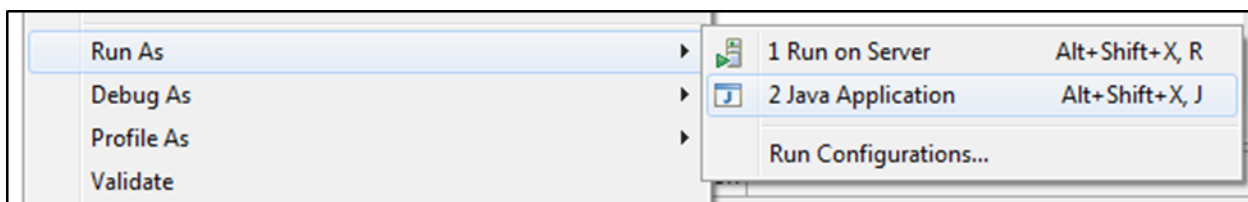
Để tìm hiểu thêm, vào menu Window->Reference



Các phím tắt hay dùng: Trong Eclipse, vào menu Help->Key Assist hoặc nhấn tổ hợp Ctrl-Shift-L để hiển thị.

3. Thực thi chương trình:

Nhấn chuột phải lên lớp cần chạy, chọn menu Run As-> Java Application.

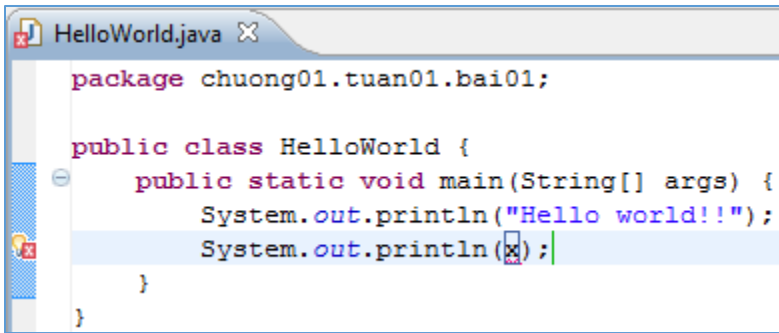


Hoặc nhấn F11 để chạy tập tin hiện tại, còn Ctrl+F11 biên dịch và chạy toàn bộ project.

Một vài vấn đề thường gặp:

1. Eclipse sẽ tự động biên dịch code và báo lỗi.

Nếu bạn có lỗi hay warning thì bên trái của dòng lỗi. Ví dụ như sau:



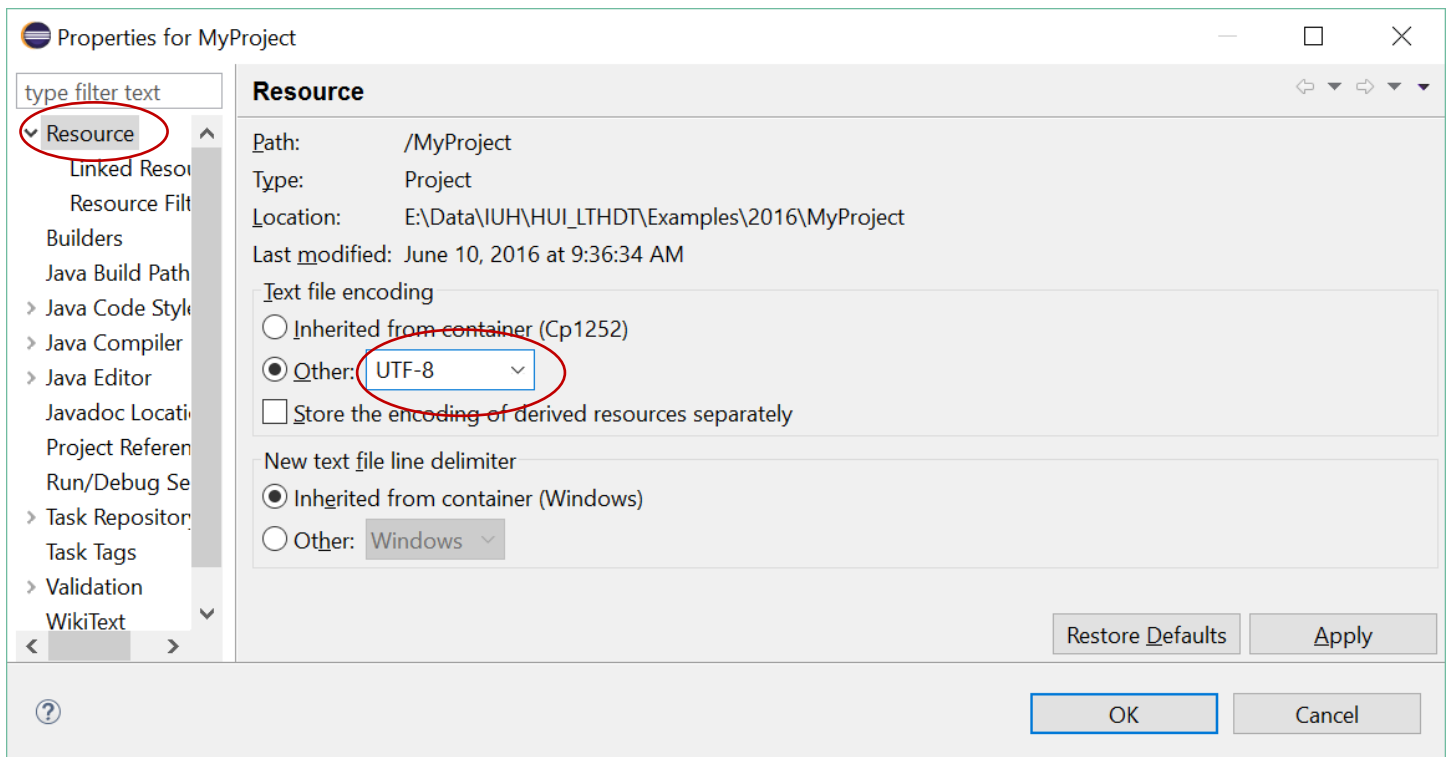
```
package chuong01.tuan01.bai01;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!!");
        System.out.println(x);
    }
}
```

2. Vấn đề gõ tiếng Việt(unicode) trong eclipse:

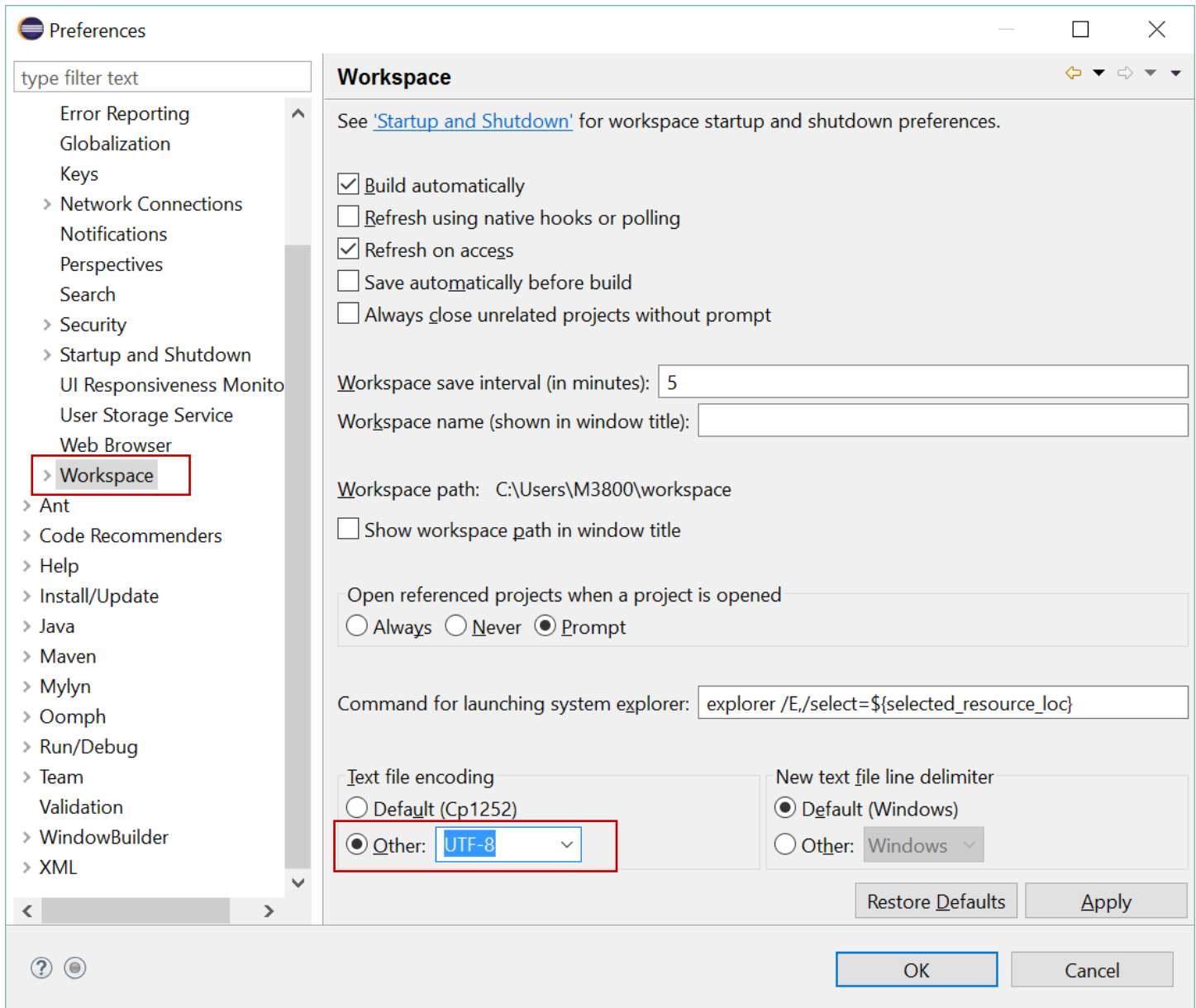
Java sử dụng bảng mã unicode nên việc gõ tiếng việt là OK. Để gõ được tiếng việt, đảm bảo là project của bạn phải được lưu với bảng mã UTF-8.

Cách làm như sau: Nhấn chuột phải lên Project, chọn Properties. Chọn mục resources như hình



Điều này cho phép project bạn chọn có sử dụng unicode.

Để cho tất cả từ project lúc thiết lập về sau sử dụng unicode (khởi mặc công mỗi project mỗi thiết lập), ta làm như sau: Vào menu Window->References, chọn mục General-> Workspace như hình

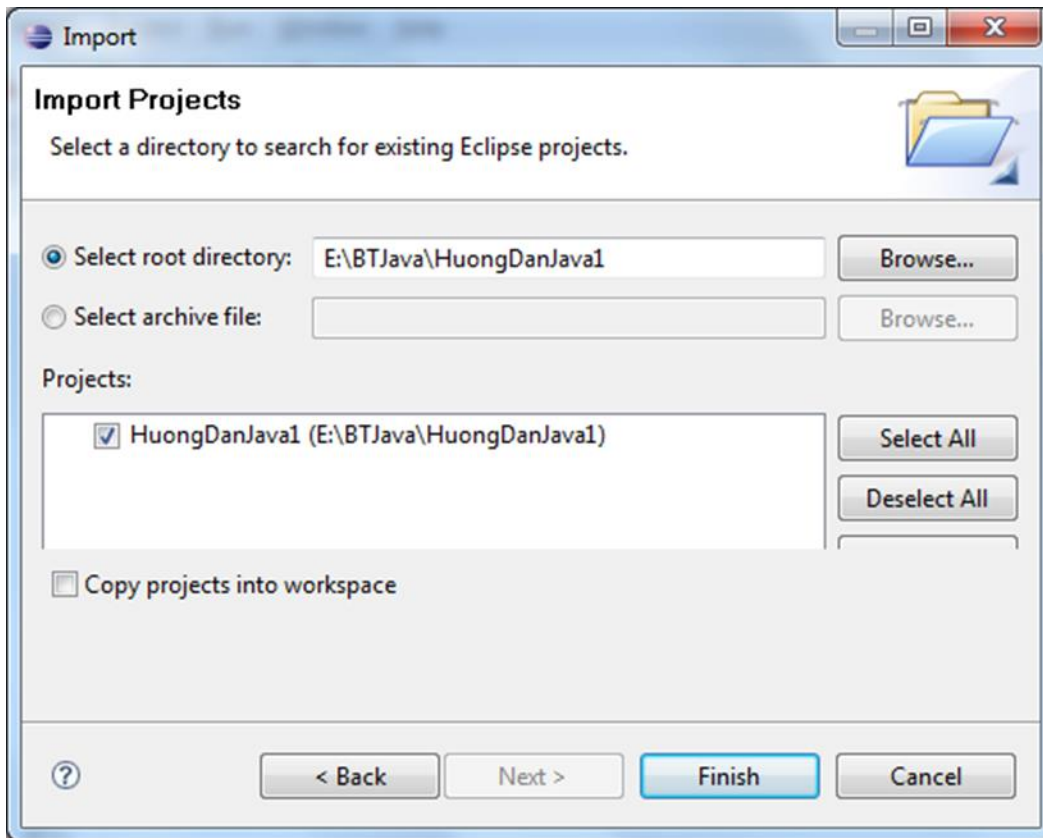


Nhấn Apply. Từ đây, bất cứ project nào tạo ra đều hỗ trợ Unicode.

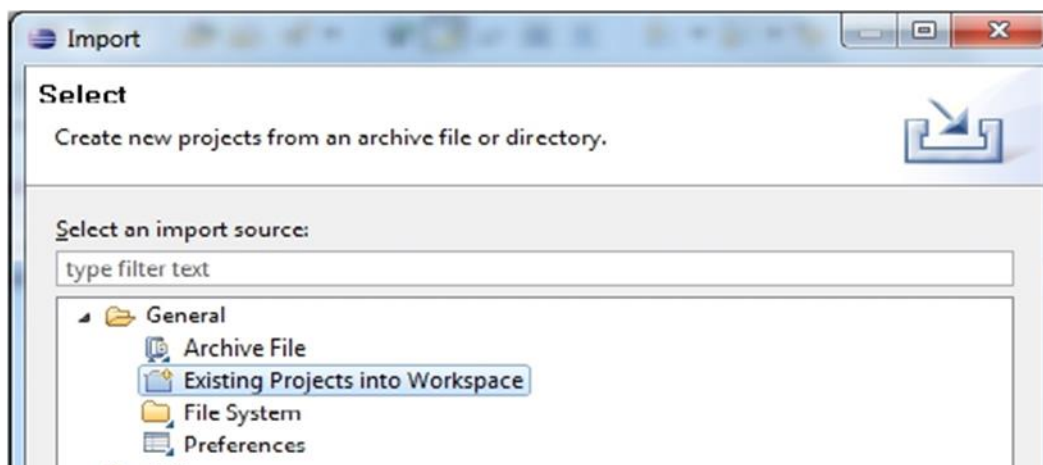
3. Mở Project trong Eclipse

Eclipse không hỗ trợ mở project trực tiếp nên bạn không có kiểu “double-click-for –open” thường thấy, mà bạn phải import project vào workspace như sau:

Vào menu File->Import rồi chọn như hình



Nhấn Next. Sau đó nhấn nút Browse để tìm đến thư mục chứa project.

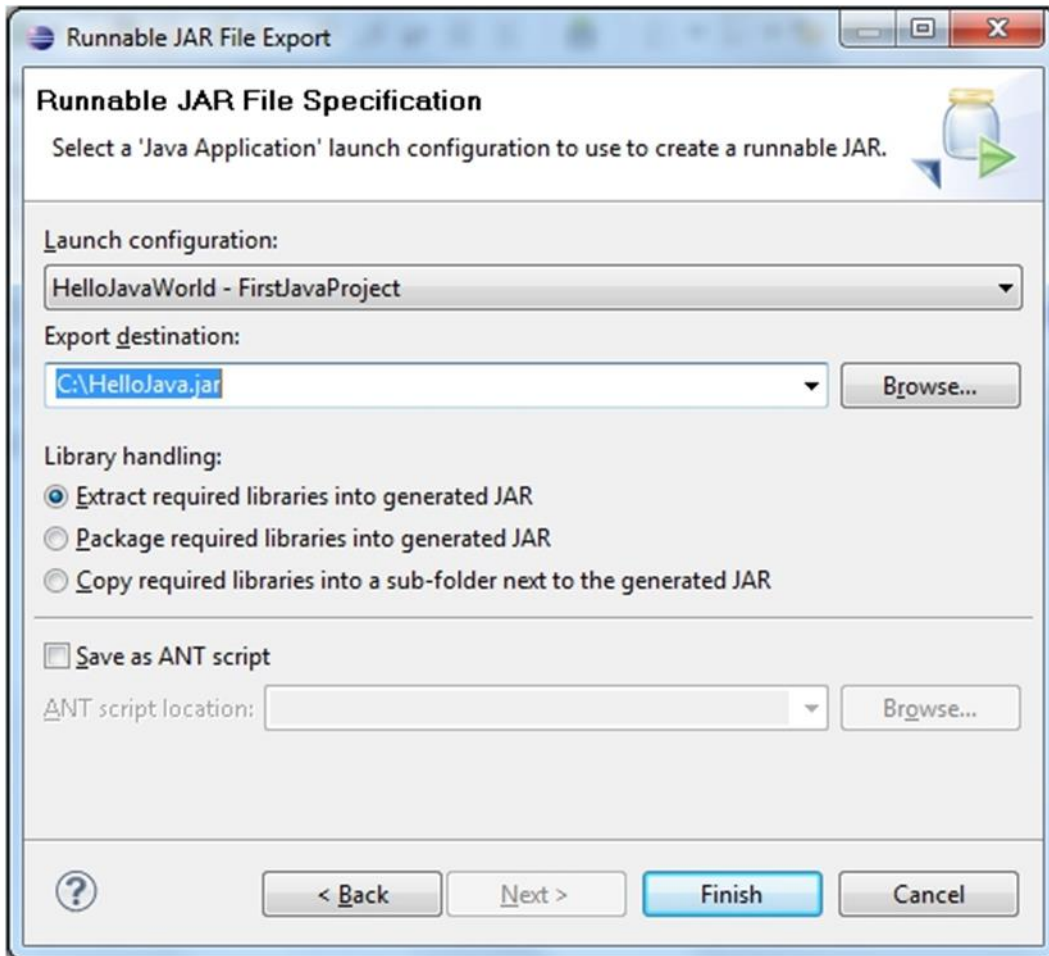


Chọn Project cần Import rồi nhấn Finish.

4. Export file jar tự chạy(executable jar file) trong eclipse

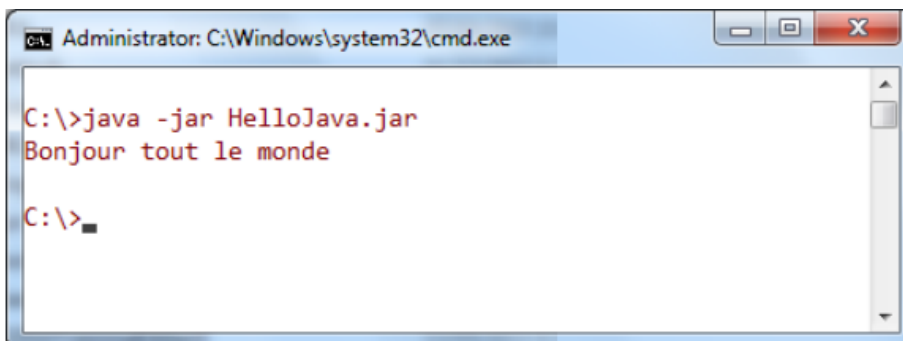
Nhấn chuột phải lên Project cần export, chọn Export.

Chọn Runnable JAR file như hình. Nhấn Next

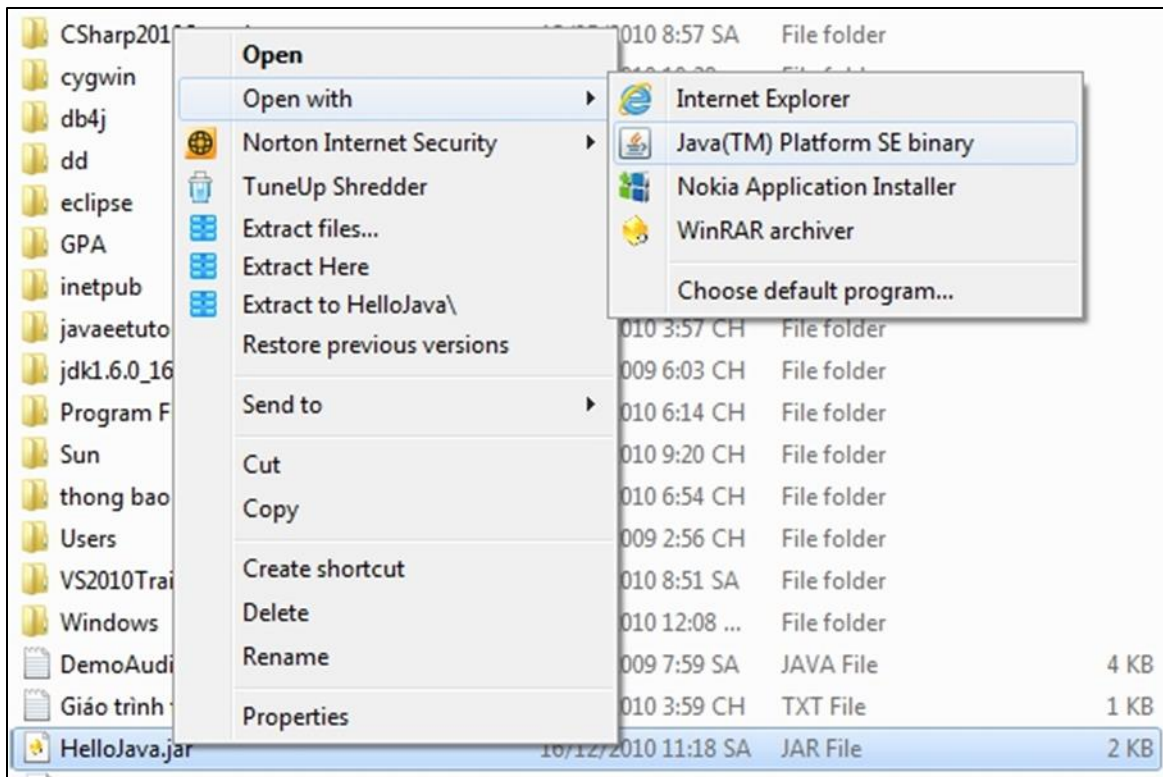


Chọn Launch configuration. Chọn thư mục chứa cũng như tên file jar. Nhấn Finish.

Thực thi jar file dưới dạng command-line:



Nếu Project của bạn ở cơ chế GUI thì bạn có thể mở file jar của bạn bằng Java Platform SE library như hình.



PHẦN LIÊN QUAN ĐẾN NHẬP XUẤT

Mục tiêu: Làm quen với việc nhập xuất trong Java

- `System.out.print()`
- `System.out.println()`
- `System.out.printf()`
- Sử dụng đối tượng `Scanner`

Bài bắt buộc: 4, 5

Bài 4. Viết chương trình xuất ra màn hình dòng chữ “Hello World!”

Lưu ý sử dụng xuất ra màn hình:

- `System.out.print()`: xuất ra màn hình một chuỗi hay một giá trị (không xuống dòng)
- `System.out.println()`: xuất ra màn hình một chuỗi hay một giá trị (có xuống dòng)
- `System.out.printf()`: xuất ra màn hình một chuỗi theo định dạng tương tự như ngôn ngữ lập trình C++

Cú pháp: `System.out.printf("format-string", [arg1, arg2, arg3, ...]);`

- `%d, %f, %c, %s`: số nguyên (byte, short, int, long), số thực (float, double)
- Dấu `-`: canh trái, mặc định canh phải.

- %0: fill số 0.
- Một số ký tự đặc biệt: \a, \b, \f, \n, \r, \t, \v, \\
- Độ chính xác của số thực: %5.3f: độ chính xác phần lẻ của số trong format-string là 3

System.out.printf("*format-string*" [, *arg1*, *arg2*, ...]);

Format String:

Composed of literals and format specifiers. Arguments are required only if there are format specifiers in the format string. Format specifiers include: flags, width, precision, and conversion characters in the following sequence:

% [flags] [width] [.precision] conversion-character (square brackets denote optional parameters)

Flags:

- : left-justify (default is to right-justify)
- + : output a plus (+) or minus (-) sign for a numerical value
- 0 : forces numerical values to be zero-padded (default is blank padding)
- , : comma grouping separator (for numbers > 1000)
- : space will display a minus sign if the number is negative or a space if it is positive

Width:

Specifies the field width for outputting the argument and represents the minimum number of characters to be written to the output. Include space for expected commas and a decimal point in the determination of the width for numerical values.

Precision:

Used to restrict the output depending on the conversion. It specifies the number of digits of precision when outputting floating-point values or the length of a substring to extract from a String. Numbers are rounded to the specified precision.

Conversion-Characters:

- d : decimal integer [byte, short, int, long]
- f : floating-point number [float, double]
- c : character Capital C will uppercase the letter
- s : String Capital S will uppercase all the letters in the string
- h : hashcode A hashcode is like an address. This is useful for printing a reference
- n : newline Platform specific newline character- use %n instead of \n for greater compatibility

Bài 5. Viết chương trình nhập vào tên của mình và xuất ra màn hình “Hello + Tên”.

Lưu ý:

- Để nhập dữ liệu từ bàn phím, dùng thư viện Scanner bằng cách
import java.util.Scanner;
- Khai báo đối tượng
Scanner sc=new Scanner(System.in);

- Dữ liệu nhập vào là số nguyên:
int a=sc.nextInt();
- Dữ liệu nhập vào là số thực:
double b=sc.nextDouble();
- Dữ liệu nhập vào là chuỗi:
String b=sc.nextLine();

PHẦN LIÊN QUAN ĐẾN CÁC TOÁN TỬ

Mục tiêu: Làm quen với việc sử dụng toán tử trong Java

- Toán tử số học
- Toán tử quan hệ
- Toán tử logic
- Toán tử điều kiện

Bài bắt buộc: 6, 7, 8

Bài 6. Cho đoạn chương trình sau:

Thao tác với toán tử số học

```
class ArithmeticDemo
{
    public static void main (String[] args)
    {
        int result = 1 + 2;
        result = result - 1;
        result = result * 2;
        result = result / 2;
        result = result + 8;
        result = result % 7;
        System.out.println("final result: " + result);
    }
}
```

Thao tác với toán tử nối chuỗi

```
class ConcatDemo
{
    public static void main(String[] args)
    {
        String firstString = "This is";
        String secondString = " a concatenated string.";
        String thirdString = firstString+secondString;
        System.out.println(thirdString);
    }
}
```

Thao tác với toán tử 1 ngôi

```
class UnaryDemo
{
    public static void main(String[] args)
    {
        int result = +1;
        System.out.println(result);

        result--;
        System.out.println(result);

        result++;
        System.out.println(result);

        result = -result;
        System.out.println(result);

        boolean success = false;
        System.out.println(success);
        System.out.println(!success);
    }
}
```

Thao tác với toán tử pre-increment và post-increment

```
class PrePostDemo
{
    public static void main(String[] args)
    {
        int i = 3;
        i++;
        System.out.println(i);
        ++i;
        System.out.println(i);
        System.out.println(++i);
        System.out.println(i++);
        System.out.println(--i);
        System.out.println(i--);
        System.out.println(i);
    }
}
```

Kết quả và giải thích?

Bài 7. Cho đoạn chương trình sau:

Thao tác với các toán tử quan hệ, toán tử so sánh trong ngôn ngữ lập trình Java.

```

class ComparisonDemo
{
    public static void main(String[] args)
    {
        int value1 = 1;
        int value2 = 2;

        System.out.println("value1 == value2: " + (value1 == value2) );
        System.out.println("value1 != value2: " + (value1 != value2) );
        System.out.println("value1 > value2: " + (value1 > value2) );
        System.out.println("value1 < value2: " + (value1 < value2) );
        System.out.println("value1 <= value2: " + (value1 <= value2) );
        System.out.println("(value1 <= value2) && (value1 == value2): "
            + ((value1 <= value2) && (value1 == value2)) );
        System.out.println("(value1 <= value2) || (value1 == value2) "
            + ((value1 <= value2) || (value1 == value2)) );
    }
}

```

Kết quả và giải thích?

Bài 8. Cho đoạn chương trình sau:

Thao tác với toán tử điều kiện

```

class ConditionalDemo
{
    public static void main(String[] args)
    {
        int value1 = 1; int value2 = 2; int result;
        boolean someCondition = true;
        result = someCondition ? value1 : value2;
        System.out.println(result);
    }
}

```

Mục tiêu: Làm quen với việc sử dụng các cấu trúc trong Java

- Cấu trúc **if, if .. else**
- Cấu trúc **switch**
- Cấu trúc lặp: **for, while, do..while**

Bài bắt buộc: 9, 10, 11, 14, 18, 20

PHẦN CẤU TRÚC LẶP VÀ CẤU TRÚC ĐIỀU KHIỂN

Bài 9. Cho biết ý nghĩa của các chương trình sau

Cấu trúc if-else, switch case

```
class IfElseDemo
{
    public static void main(String[] args)
    {
        int testscore = 76;
        char grade;
        if (testscore >= 90)
        {
            grade = 'A';
        }
        else if (testscore >= 80)
        {
            grade = 'B';
        }
        else if (testscore >= 70)
        {
            grade = 'C';
        }
        else if (testscore >= 60)
        {
            grade = 'D';
        }
        else {
            grade = 'F';
        }
        System.out.println("Grade = " + grade);
    }
}
```

```

class SwitchDemo
{
    public static void main(String[] args)
    {
        int month = 8;
        String monthString;
        switch (month)
        {
            case 1: monthString = "January"; break;
            case 2: monthString = "February"; break;
            case 3: monthString = "March"; break;
            case 4: monthString = "April"; break;
            case 5: monthString = "May"; break;
            case 6: monthString = "June"; break;
            case 7: monthString = "July"; break;
            case 8: monthString = "August"; break;
            case 9: monthString = "September"; break;
            case 10: monthString = "October"; break;
            case 11: monthString = "November"; break;
            case 12: monthString = "December"; break;
            default: monthString = "Invalid month"; break;
        }
        System.out.println(monthString);
    }
}

```

Cấu trúc while, do while và for

```

class ForDemo
{
    public static void main(String[] args)
    {
        for(int i=1; i<11; i++)
        {
            System.out.println("Count is: " + i);
        }
    }
}

```



```

class WhileDemo
{
    public static void main(String[] args)
    {
        int count = 1;
        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        }
    }
}

class DoWhileDemo
{
    public static void main(String[] args)
    {
        int count = 1;
        do {
            System.out.println("Count is: " + count);
            count++;
        } while (count < 11);
    }
}

```

Phát sinh ngẫu nhiên số sử dụng lớp Random. Lớp Random nằm trong gói java.util.* có 1 số phương thức:

Method	Produces
<code>boolean nextBoolean();</code>	A true or false value
<code>int nextInt()</code>	An integral value between Integer.MIN_VALUE and Integer.MAX_VALUE
<code>long nextLong()</code>	A long integral value between Long.MIN_VALUE and Long.MAX_VALUE
<code>float nextFloat()</code>	A decimal number between 0.0 (included) and 1.0 (excluded)
<code>double nextDouble()</code>	A decimal number between 0.0 (included) and 1.0 (excluded)

```

import java.util.Random;

public class RandomExercise {
    public static void main(String[] args) {
        Random rd = new Random();
        int n = rd.nextInt();
        System.out.println("Number: " + n);
    }
}

```

Phát sinh số ngẫu nhiên nằm trong một vùng (min, max)

```
int min = 65;  
int max = 80;  
Random r = new Random();  
int i1 = r.nextInt(max - min + 1) + min;
```

Lưu ý: Nếu `r.nextInt(max)` sẽ trả về giá trị giữa 0 và max.

- Bài 10.** Viết chương trình in ra tổng của 10 số chẵn đầu tiên (sử dụng vòng lặp for hoặc while)
- Bài 11.** Viết chương trình in ra những số lẻ từ 1 đến 99.
- Bài 12.** Viết chương trình xuất ra tổng các số là bội số của 7 (từ 1 đến 100)
- Bài 13.** Viết chương trình in ra tổng $1+2+3 \dots +n$ với n được nhập từ tham số command line
- Bài 14.** Viết chương trình in ra tổng $1+3+5 \dots +n$ nếu n là số chẵn, $2+4+6+ \dots n$ nếu n là số lẻ. Giá trị n được nhập vào từ tham số command line
- Bài 15.** Viết chương trình in ra giá trị lớn nhất và nhỏ nhất trong một dãy các giá trị user đã nhập vào từ tham số command line.
- Bài 16.** Viết chương trình giải phương trình bậc 1 với hệ số a, b được nhập vào bởi user từ tham số command line.
- Bài 17.** Viết chương trình đọc một giá trị nguyên từ bàn phím và in ra số đó là số chẵn, lẻ hoặc zero
- Bài 18.** Viết chương trình in ra bội số của 3 từ 300 đến 3.
- Bài 19.** Viết chương trình in ra số lần ký tự 'a' xuất hiện trong một chuỗi.
- Bài 20.** Viết hàm để đếm số lượng ký tự là số có trong chuỗi s . Chuỗi s được nhập từ bàn phím. Dùng mã ASCII để kiểm tra hoặc dùng class Character: Character.isDigit(ký tự) để kiểm ký tự có phải là số hay không.
- Bài 21.** Viết hàm tách chuỗi gốc thành chuỗi khác.

VD: chuỗi gốc $S = \text{"Bai Tap Mon Lap Trinh Java"}$, chuỗi sau khi tách là

Bai
Tap
Mon
Lap
Trinh
Java

- Bài 22.** Viết chương trình kiểm tra số nhập vào có phải là số nguyên tố hay không
- Bài 23.** Viết chương trình tìm USCLN của 2 số nhập vào.
- Bài 24.** Viết chương trình tính tổng N số nguyên.
- Bài 25.** Tính tổng các số nguyên tố nhỏ hơn N
- Bài 26.** Tính tổng N số nguyên tố đầu tiên
- Bài 27.** Viết chương trình nhập vào số nguyên n và thực hiện:

Tuần 2.

Tuần 3. **LỚP VÀ ĐỐI TƯỢNG**

Chương 1. Tổng quan về cách tiếp cận hướng đối tượng

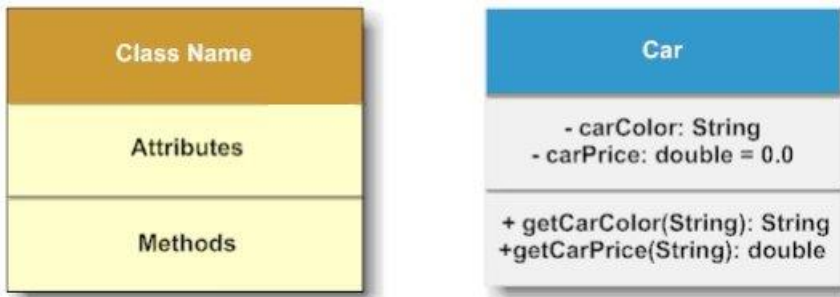
Chương 2. Những khái niệm cơ bản của lập trình hướng đối tượng

Chương 3. Giới thiệu về Java

Mục tiêu:

- Hiểu và áp dụng được cách viết các lớp với các thuộc tính, phương thức bằng ngôn ngữ lập trình Java
- Hiểu và áp dụng được cách sử dụng Object và Class Diagram (ở mức đơn giản trong UML)

Định nghĩa 1 lớp trong UML2



Cách truy xuất (visibility) gồm public, private và protected

public	+
private	-
protected	#
package	~

```
public class Car {
    private String carColor;
    private double carPrice = 0.0;

    public String getCarColor(String model) {
        return carColor;
    }

    public double getCarPrice(String model) {
        return carPrice;
    }
}
```

Lớp, đối tượng

Mục tiêu: Làm quen với lập trình hướng đối tượng đơn giản trong Java

- Thực hành viết lớp:
 - Định nghĩa getter, setter cho thuộc tính (kiểm tra ràng buộc trên thuộc tính nếu có)
 - Tạo Constructor có kiểm soát tham số truyền cho thuộc tính
 - Tạo cơ chế liên lạc giữa các lớp
- Ghi chú cho Lớp, Phương thức (*theo dạng document*)
- Tạo, sử dụng đối tượng

Bài bắt buộc: 1, 3, 3b, 5

Bài 1. Viết chương trình tính diện tích, chu vi hình chữ nhật.

- Hãy viết lớp **HìnhChuNhat** gồm có:
 - Attributes : chiều dài, chiều rộng.
 - Phương thức thiết lập (set), và lấy (get) thông tin chiều dài, chiều rộng.
 - Phương thức tính diện tích, chu vi.
 - Phương thức toString gồm các thông tin dài, rộng, diện tích, chu vi.
- Xây dựng lớp chứa hàm main cho phần kiểm nghiệm. Dài rộng có thể nhập từ bàn phím.

Bài 2. Viết chương trình OOP quản lý sinh viên đơn giản: Nhập, xuất thông tin, tính điểm TB.

- Viết lớp Sinh viên như sau:

Attributes (private):

- Mã sinh viên là số nguyên.
- Họ tên: chuỗi ký tự.
- Điểm LT, điểm TH : float

Constructor:

- Constructor mặc định (để khởi tạo đối tượng với các thông tin kiểu số là 0, kiểu chuỗi là chuỗi rỗng).
- Constructor thứ hai nhận đầy đủ thông tin để khởi tạo giá trị cho tất cả các biến instance.

Methods:

- Các getter và setter cho mỗi thuộc tính.
- Tính điểm trung bình.
- Phương thức toString để diễn tả đối tượng ở dạng chuỗi có định dạng

HD: Thông tin sinh viên in trên một dòng có định dạng. Sử dụng String.format(“chuỗi định dạng”, đối số 1, đối số 2,); Trong đó chuỗi định dạng giống c++, ví dụ:

“%-30s”: chuỗi, chiếm 30 ký tự, dấu trừ canh lề trái.

“%5.2f”: số thực, chiếm 5 ký tự, bao gồm 2 ký số lẻ.

Ký tự định dạng:

- s : chuỗi
- d: số nguyên (byte, short, int, long)
- f: số thực (float, double)
- b: boolean

- Xây dựng class chứa hàm main: tạo 3 đối tượng sinh viên sv1, sv2, sv3, trong đó:
 - sv1 chứa thông tin của chính mình (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
 - sv2 là thông tin người bạn thân nhất của em (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
 - sv3 tạo bằng constructor mặc định. Nhập các thông tin cho sv3 từ bàn phím rồi sau đó dùng các setter để gán vào cho các thuộc tính tương ứng.
 - In bảng danh sách sinh viên gồm 4 cột là MSSV, họ tên, điểm LT”, điểm TH, điểm TB (bảng có 3 dòng cho 3 sinh viên).

```
package iuh.oop.week01.exercise02;

/**
 * /iuh/oop/week01/exercise02/SinhVien.java
 * Biểu diễn cho 1 sinh viên trong trường. Với các thuộc tính gồm mã số sinh viên,
 * họ và tên của sinh viên, điểm thi phần lý thuyết và điểm thi thực hành
 *
 * @author VinhHien
 */
public class SinhVien {
    /**
     * Mã số của sinh viên
     */
    private int masv;
    /**
     * Họ và tên của sinh viên
     */
    private String hoten;
    /**
     * Điểm lý thuyết
     */
    private float diemlt;
    /**
     * Điểm thực hành
     */
    private float diemth;

    /**
     * Default constructor của lớp SinhVien
     */
    public SinhVien() {
        this(0, "no-name", 0.0f, 0.0f);
    }

    /**
     * Constructor đầy đủ của lớp SinhVien.
     * Dùng để tạo mới một sinh viên khi biết mã số sinh viên, họ và tên, điểm

```

lý thuyết, điểm thực hành

```
* @param masv là mã số sinh viên
* @param hoten là họ và tên của sinh viên
* @param diemlt là điểm lý thuyết của sinh viên
* @param diemth là điểm thực hành của sinh viên
*/
public SinhVien(int masv, String hoten, float diemlt, float diemth) {
    this.masv = masv;
    this.hoten = hoten;
    this.diemlt = diemlt;
    this.diemth = diemth;
}

/**
 * Lấy mã số sinh viên
 * @return the masv
 */
public int getMasv() {
    return masv;
}

/**Thiết lập mã số sinh viên
 * @param masv the masv to set
 */
public void setMasv(int masv) {
    this.masv = masv;
}

/**
 * Lấy thông tin họ và tên của sinh viên
 * @return the hoten
 */
public String getHoten() {
    return hoten;
}

/**
 * Thiết lập họ và tên sinh viên
 * @param hoten the hoten to set
 */
public void setHoten(String hoten) {
    this.hoten = hoten;
}

/**
 * Lấy điểm lý thuyết của sinh viên
 * @return the diemlt
 */
public float getDiemlt() {
```



```

        return diemlt;
    }

    /**
     * Thay đổi điểm lý thuyết cho sinh viên
     * @param diemlt the diemlt to set
     */
    public void setDiemlt(float diemlt) {
        this.diemlt = diemlt;
    }

    /**
     * Lấy điểm thực hành của sinh viên
     * @return the diemth
     */
    public float getDiemth() {
        return diemth;
    }

    /**
     * Thay đổi điểm thực hành cho sinh viên
     * @param diemth the diemth to set
     */
    public void setDiemth(float diemth) {
        this.diemth = diemth;
    }

    /**
     * Lấy điểm trung bình của sinh viên
     * @return the diemtb điểm trung bình
     */
    public float getDiemtb() {
        return (diemlt + diemth)/2;
    }

    /**
     * Biểu diễn đối tượng sinh viên ở dạng chuỗi
     * @return String
     */
    @Override
    public String toString() {
        return String.format("%-5s %-30s %10.2f %10.2f %10.2f", masv, hoten,
diemlt, diemth, getDiemtb());
    }
}

```

```
package iuh.oop.week01.exercise02;
```

```

import java.util.Scanner;

/**
 * Kiểm thử cho lớp {@link SinhVien}
 * @author VinhHien
 *
 */
public class SinhVienTest {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        //Tạo 2 đối tượng sv1 và sv2 bằng constructor đầy đủ tham số
        SinhVien sv1 = new SinhVien(11111, "Nguyễn Thanh An", 6.5f, 8.5f);
        SinhVien sv2 = new SinhVien(22222, "Lê Thị Bông", 7.5f, 8.0f);

        //Tạo đối tượng sv3 bằng default constructor
        SinhVien sv3 = new SinhVien();

        //Nhập dữ liệu của sinh viên sv từ bàn phím
        System.out.println("Nhập mã số của sinh viên sv3: ");
        int masv = sc.nextInt();
        sc.nextLine();
        System.out.println("Nhập họ và tên của sinh viên sv3: ");
        String hoten = sc.nextLine();
        System.out.println("Nhập điểm lý thuyết của sinh viên sv3: ");
        float diemlt = sc.nextFloat();
        System.out.println("Nhập điểm thực hành của sinh viên sv3: ");
        float diemth = sc.nextFloat();

        //Goi các phương thức để gán giá trị cho sv3
        sv3.setMasv(masv);
        sv3.setHoten(hoten);
        sv3.setDiemlt(diemlt);
        sv3.setDiemth(diemth);

        //In thông tin của 3 đối tượng sv1, sv2, sv3 ra cửa sổ console
        System.out.println(String.format("%-5s %-30s %10s %10s %10s", "masv",
"hoten", "diemlt", "diemth", "diemtb"));
        System.out.println(sv1);
        System.out.println(sv2);
        System.out.println(sv3);

        sc.close();
    }
}

```

```

Problems @ Javadoc Declaration Console
<terminated> SinhVienTest [Java Application] C:\Java\jre1.8.0_141\bin\javaw.exe (Aug 5, 2017, 2:00:05 PM)
Nhập mã số của sinh viên sv3:
33333
Nhập họ và tên của sinh viên sv3:
Nguyễn Hoàng Anh
Nhập điểm lý thuyết của sinh viên sv3:
5.0
Nhập điểm thực hành của sinh viên sv3:
9.0
masv    hoten                                diemlt    diemth    diemtb
11111   Nguyễn Thanh An                      6.50      8.50      7.50
22222   Lê Thị Bông                             7.50      8.00      7.75
33333   Nguyễn Hoàng Anh                        5.00      9.00      7.00

```

Bài 3. Sở giao thông cần theo dõi việc đăng ký xe của người dân. Dựa vào thông tin trị giá xe và dung tích xylanh của xe, sở giao thông cũng tính mức thuế phải đóng trước bạ khi mua xe như sau:

- Dưới 100cc, 1% trị giá xe.//50 10tr→ expected: 100000
- Từ 100 đến 200cc, 3% trị giá xe.//100 10tr→expected, 150
- Trên 200cc, 5% trị giá xe.//250 10tr →expected

Hãy thiết kế và cài đặt class Vehicle với các attributes và methods phù hợp. Class phải có các constructor và phải bảo đảm tính encapsulation.

Xây dựng class chứa hàm main. Hàm main in ra menu lựa chọn các công việc:

1. Nhập thông tin và tạo các đối tượng xe1, xe2, xe3
2. Xuất bảng kê khai tiền thuế trước bạ của các xe.
3. Thoát.

Mẫu kết xuất của chương trình:

Tên chủ xe	Loại xe	Dung tích	Trị giá	Thuế phải nộp
Nguyễn Thu Loan	Future Neo	100	35000000.00	1050000.00
Lê Minh Tính	Ford Ranger	3000	250000000.00	12500000.00
Nguyễn Minh Triết	Landscape	1500	1000000000.00	50000000.00

Bài 3b: Viết lớp **HangThucPham** mô tả một hàng hóa là hàng thực phẩm trong kho của một siêu thị, có các thuộc tính: mã hàng (*không cho phép sửa, không được để rỗng*), tên hàng (*không được để rỗng*), đơn giá (>0), ngày sản xuất và ngày hết hạn (*ngày không được để rỗng, ngày hết hạn phải sau ngày sản xuất*).

Ràng buộc chặt chẽ các ràng buộc trên các trường dữ liệu. Nếu dữ liệu không hợp lệ thì gán giá trị mặc định cho phép tương ứng của trường đó.

- Tạo 1 constructor có đầy đủ tham số, 1 constructor có tham số là mã hàng.
- Viết các phương thức setters/getters.
- Viết phương thức kiểm tra một hàng thực phẩm đã hết hạn chưa?

- Phương thức toString, trả về chuỗi chứa thông tin của hàng thực phẩm. Trong đó: Định dạng đơn giá có phân cách hàng nghìn. Định dạng kiểu ngày là dd/MM/yyyy.

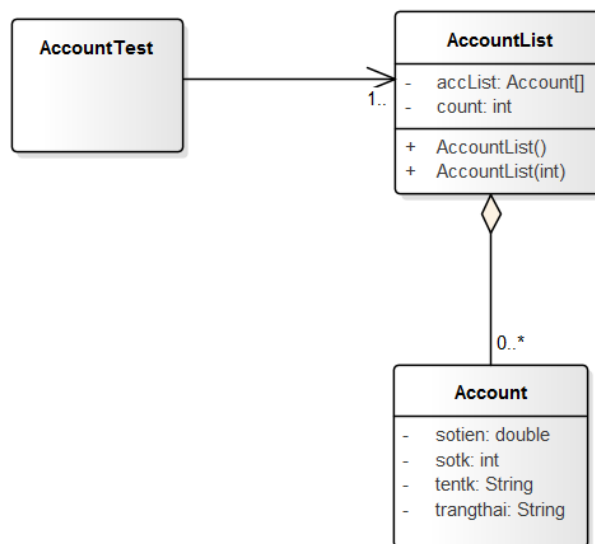
Viết lớp cho phần kiểm nghiệm.

Bài 4. Ngân hàng ABC muốn lưu trữ thông tin của mỗi tài khoản. Mỗi tài khoản chứa các thông tin:

- Số tài khoản (kiểu long),
- Tên tài khoản (kiểu chuỗi),
- Số tiền trong tài khoản (kiểu double)
- Trạng thái (Kiểu String): ghi nhận trạng thái hợp lệ của các thuộc tính của đối tượng (khởi tạo = "" trong các constructors)

(a). Thiết kế lớp Account để lưu trữ các thông tin, lớp bao gồm các phương thức sau:

- Constructor: Có 2 constructor (mặc định [sotk=999999, tênTK = "chưa xác định", số tiền = 50] và 3 tham số [số tk, tên tk, số tiền])
- Các phương thức get, set cho từng thuộc tính.
Phương thức set cho các thuộc tính phải kiểm tra tính hợp lệ của tham số (số TK > 0 và ≠ 999999, tên TK khác rỗng, số tiền >= 50). Trong trường hợp giá trị tham số không hợp lệ thì **gán giá trị mặc nhiên cho thuộc tính tương ứng, đồng thời gán chuỗi thông báo vào biến thuộc tính trạng thái**.
- Phương thức toString để trả về chuỗi chứa toàn bộ thông tin tài khoản, yêu cầu định dạng tiền tệ.



HD: Định dạng tiền tệ:

1. Tạo đối tượng Locale: Xác định ngôn ngữ và quốc gia áp dụng: `Locale local = new Locale("mã NN", "Ma quốc gia")`. Ví dụ: `Locale local = new Locale("vi", "vn")`

2. Tạo đối tượng NumberFormatter với tham số Locale ở trên: Ví dụ:

`NumberFormat formatter = NumberFormat.getCurrencyInstance(local);`

3. Dùng formatter để định dạng: ví dụ: `formatter(456953.12);`

[formatter sau khi định dạng sẽ trả về chuỗi số đã định dạng]

(b). Thêm các thông tin sau vào lớp Account

- Hằng số lãi suất có giá trị khởi tạo 0.035
- Constructor có 2 đối số: số tài khoản, tên tài khoản. Constructor này sẽ khởi tạo số tiền mặc định là 50.
- b.1 Phương thức nạp tiền vào tài khoản: Lấy số tiền hiện tại trong tài khoản + số tiền nạp vào
- b.2 Phương thức rút tiền: Lấy số tiền hiện tại trong tài khoản – (số tiền muốn rút)
- b.3 Phương thức chuyển khoản từ tài khoản này sang tài khoản khác
- b.4 Phương thức đáo hạn: $Số dư = Số dư + Số dư * Lãi suất$

Gợi ý: các phương thức b.1, b.2, b.3, phải kiểm tra số tiền nạp, rút, chuyển có hợp lệ hay không? (VD: tiền nhập vào <0, tiền rút nhiều hơn tiền trong tài khoản). Các phương thức nên có thông báo, cho biết việc thực hiện thành công hay bị lỗi do dữ liệu không hợp lệ.

(c) . Tạo lớp AccountList để lưu danh sách các tài khoản (dùng mảng):

- Khai báo một mảng kiểu Account: Account[] accList và biến int count (lưu số TK có trong mảng)
- Constructor mặc nhiên: khởi tạo mảng accList có 10 phần tử, **count=0**
- Constructor có tham số (**int n**): Khởi tạo mảng accList có n phần tử nếu n>0, ngược lại khởi tạo mảng accList có 10 phần tử; **count=0**
- Phương thức thêm 1 tài khoản vào danh sách, thêm thành công nếu kích thước mảng còn cho phép.
- Tìm TK theo số TK (Trả về TK cần tìm)
- Xóa tài khoản theo số TK
- Tính số lượng tài khoản có trong danh sách.
- In thông tin toàn bộ TK có trong danh sách.

(d). Tạo lớp AccountTest chứa hàm main để test các chức năng của lớp AccountList

- Khai báo biến thuộc lớp AccountList (AccountList list;)
- Tạo menu gồm các chức năng:
 1. Thêm tài khoản
 2. Số TK hiện có
 3. In thông tin tất cả TK
 4. Nạp tiền vào TK
 5. Rút tiền
 6. Chuyển tiền
 7. Kết thúc

Bài 5. a. Viết chương trình xây dựng đối tượng CD gồm có các thuộc tính sau:

- Mã CD là số nguyên (>0, mặc nhiên: 999999)
- Tựa CD: chuỗi ký tự (không được rỗng, tên mặc nhiên: “chưa xác định”)
- Số bài hát: số nguyên (>0)
- Giá thành: số thực (>0)
- Các thuộc tính khai báo private, định nghĩa các phương thức get/set cho từng thuộc tính và kiểm tra tính hợp lệ của dữ liệu
- Viết các constructor để khởi tạo đối tượng CD.
- Override phương thức toString của lớp Object.

b. Xây dựng lớp lưu danh sách các CD (dùng mảng).

- Phương thức thêm 1 CD vào danh sách, thêm thành công nếu không trùng mã CD và kích thước mảng còn cho phép.
- Tính số lượng CD có trong danh sách.
- Tính tổng giá thành của các CD.
- Phương thức sắp xếp danh sách giảm dần theo giá thành.

- Phương thức sắp xếp danh sách tăng dần theo tựa CD.
 - Phương thức trả thông tin của toàn bộ CD có trong danh sách.
- c. Viết lớp cho phần kiểm nghiệm. Dùng menu case thực hiện các chức năng theo yêu cầu.

Tuần 4.

Tuần 5. CÁC KHÁI NIỆM CƠ BẢN KẾ THỪA VÀ ĐA HÌNH

Chương 4. Kế thừa và đa hình trên Java

Mục tiêu:

- *Hiểu và áp dụng được cách viết code kế thừa và đa hình trên Java.*
- *Hiểu và áp dụng được cách sử dụng mô hình lớp với mô tả kế thừa.*
- *Bài bắt buộc: 1, 2, 3, 4, 5*

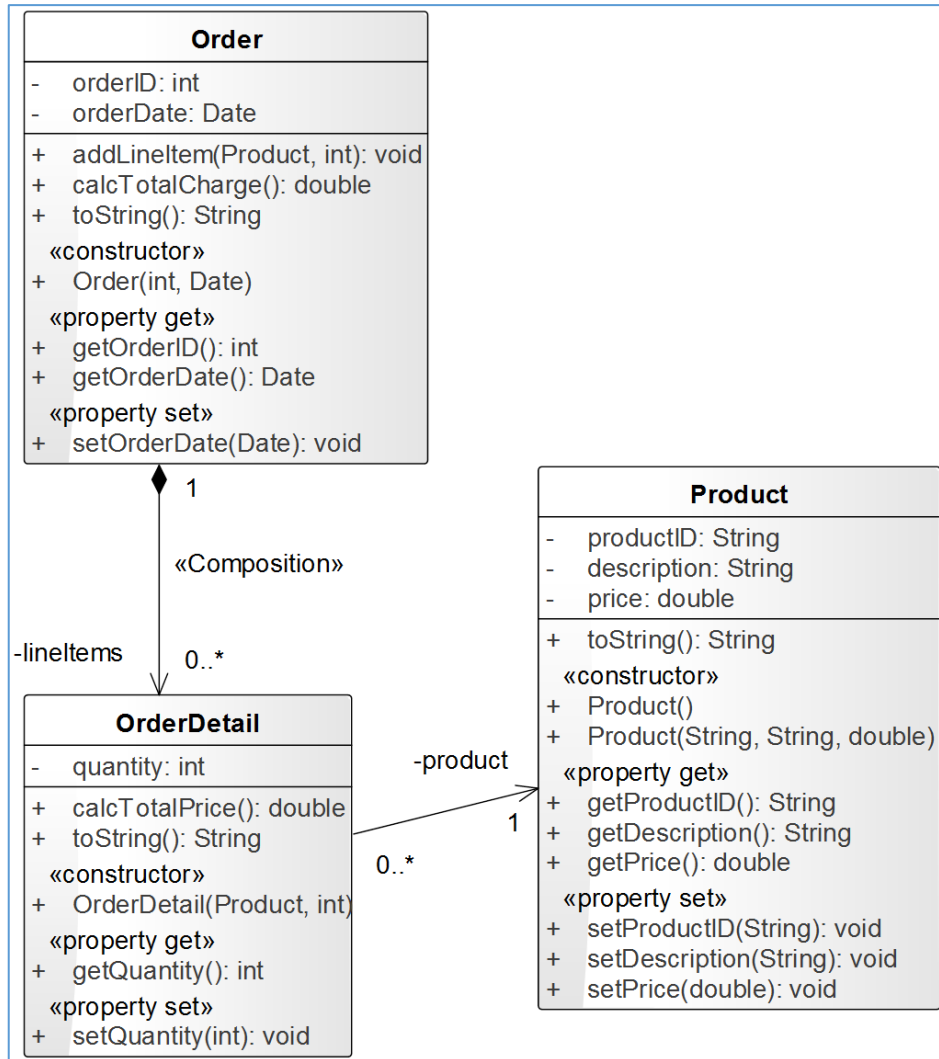
Bài 1. Một khách sạn X cần quản lý các hóa đơn của khách hàng thuê phòng. Hóa đơn có 2 loại: hóa đơn theo giờ, hóa đơn theo ngày. Thông tin chung của chi tiết hóa đơn là: Mã hóa đơn, ngày hóa đơn (ngày, tháng, năm), Tên khách hàng, mã phòng, đơn giá. Thông tin riêng của từng loại hóa đơn gồm:

- Hóa đơn theo giờ còn có số giờ thuê. Thành tiền = số giờ thuê * đơn giá. Nếu trường hợp số giờ > 24 tiếng và < 30 tiếng thì cũng chỉ tính 24 giờ. Nếu trường hợp số giờ là > 30 tiếng thì không dùng loại hóa đơn theo giờ.
- Hóa đơn theo ngày sẽ có số ngày thuê. Thành tiền = số ngày thuê * đơn giá. Nếu số ngày > 7 thì giảm 20% đơn giá cho những ngày còn lại.

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Nhập xuất danh sách các hóa đơn thuê phòng.
- Tính tổng số lượng cho từng loại thuê phòng.
- Tính trung bình thành tiền của hóa đơn thuê phòng trong tháng 9/2013

Bài 2. Hiện thực mô hình lớp sau bằng ngôn ngữ lập trình Java.



Trong đó:

- $\text{totalPrice} = \text{quantity} * \text{price}$

(thành tiền = số lượng * đơn giá sản phẩm)

- Tổng tiền hóa đơn (totalCharge) = $\sum_{count=0}^n$ thành tiền.
- Phương thức $\text{addLineItem}(\text{Product } p, \text{int } q) : \text{void}$, dùng để thêm một sản phẩm p với số lượng q vào hóa đơn.

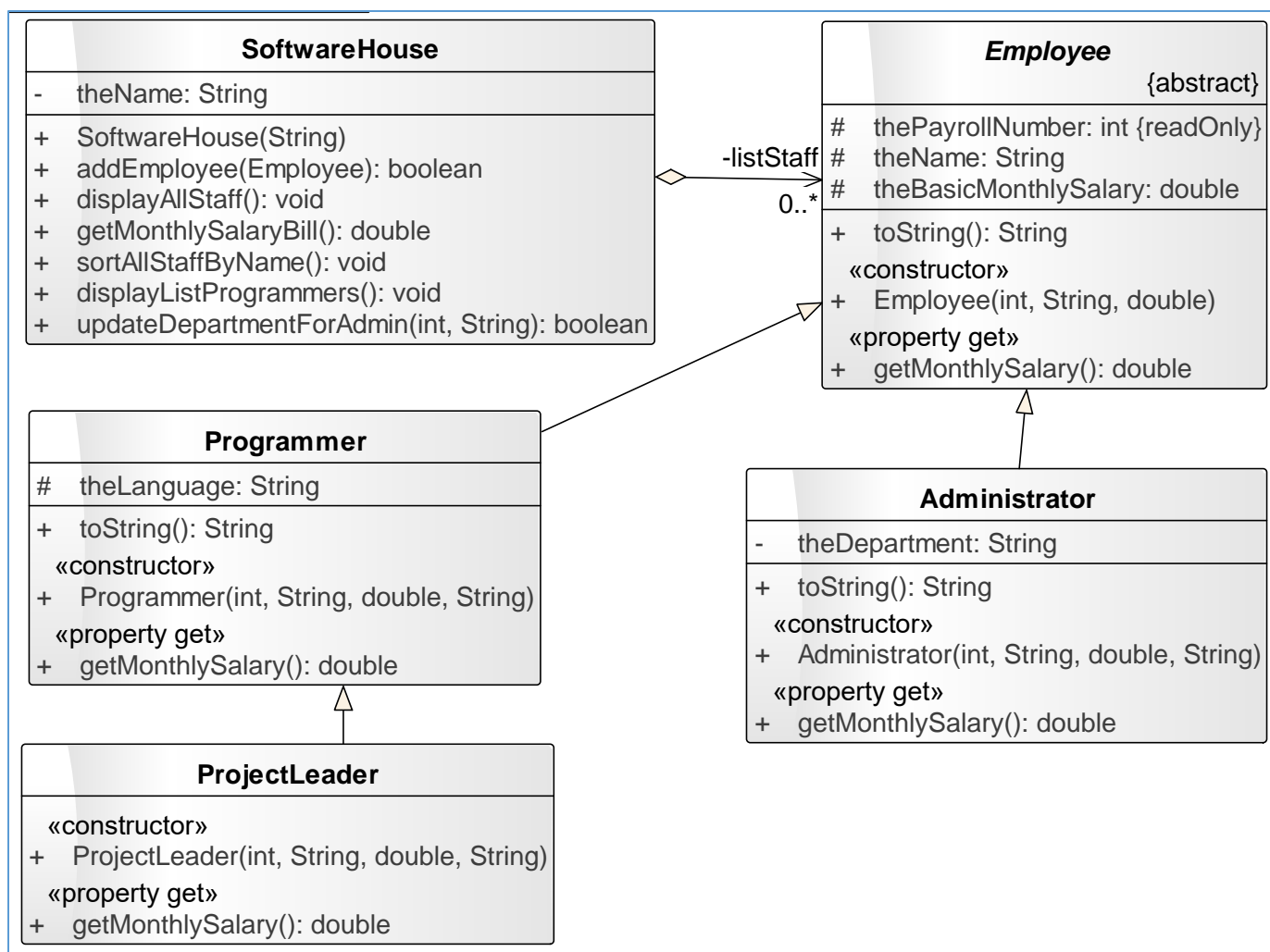
```

Problems | Javadoc | Declaration | Search | Console | Plug-in Dependencies | Bug Explorer | Bug Info | Error Log | Documentation
<terminated> Driver (15) [Java Application] C:\Java\jre1.8.0_25\bin\javaw.exe (Sep 10, 2015, 3:28:55 PM)
Mã HD: 1
Ngày lập hóa đơn: 10/09/2015
STT | Mã SP | Mô tả | Đơn giá | S Lượng | Thành tiền
1 | sp4 | Nước tương | 8,000 | 10 | 80,000 VND
2 | sp1 | Gạo | 18,000 | 5 | 90,000 VND
3 | sp3 | Đường | 10,000 | 1 | 10,000 VND
4 | sp1 | Gạo | 18,000 | 1 | 18,000 VND
Tổng tiền thanh toán: 198,000 VND

```

Phần viết lớp cho phần kiểm nghiệm, in ra màn hình theo mẫu:

Bài 3. Hiện thực mô hình lớp sau bằng ngôn ngữ lập trình Java.



Quản lý nhân viên trong một công ty phần mềm (*SoftwareHouse*). Công ty có nhiều nhân viên. Mỗi nhân viên (*Employee*) cần lưu trữ các thông tin: Mã số (*thePayrollNumber*), tên nhân viên (*theName*), lương cơ bản hằng tháng (*theBasicMonthlySalary*). Mã số của mỗi nhân viên chỉ được tạo 1 lần duy nhất và không được phép sửa.

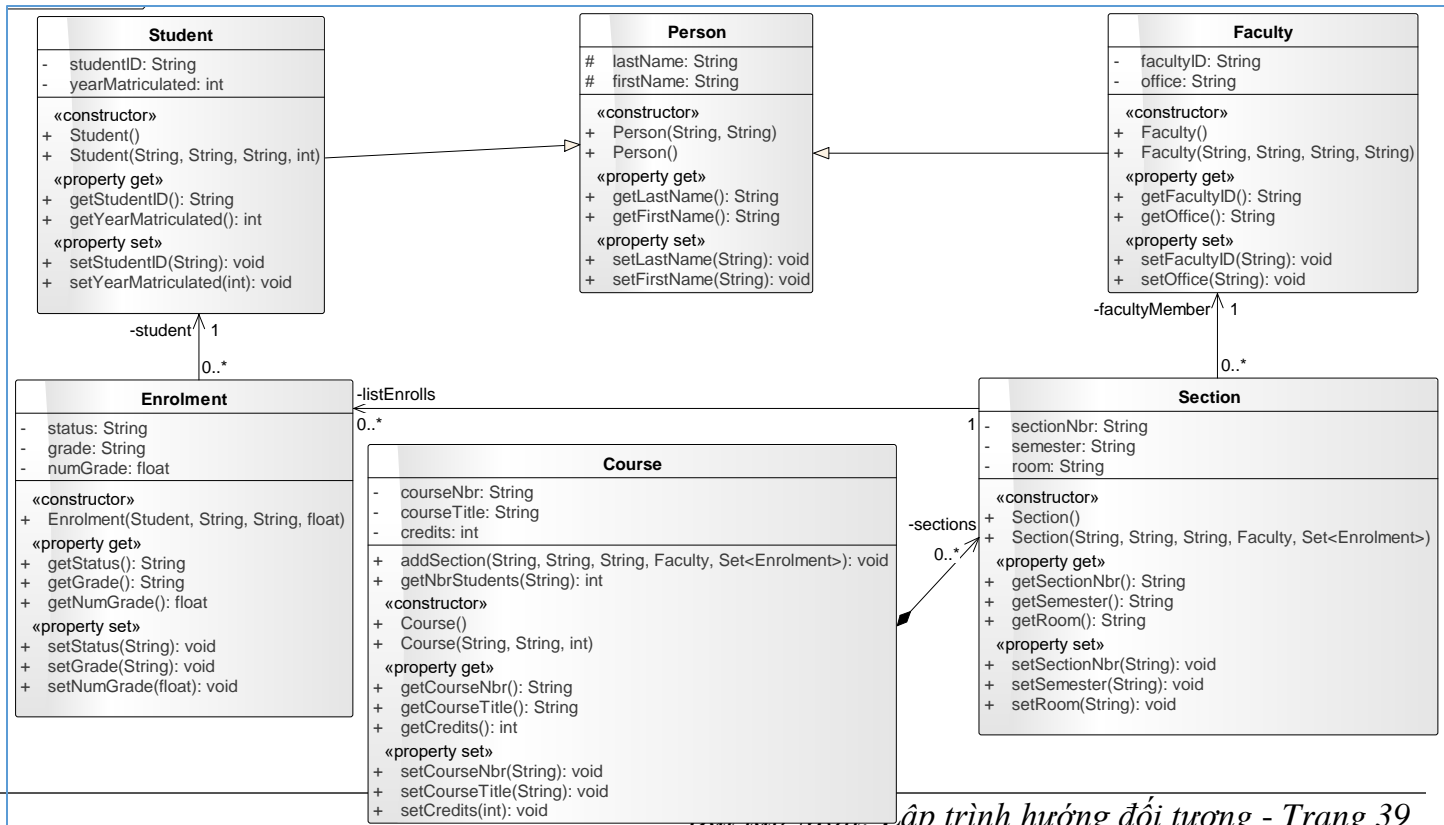
- Nhân viên trong công ty gồm: Lập trình viên (*Programmer*), Người quản lý (*Administrator*) và Người trưởng dự án (*ProjectLeader*)
 - Tạo các phương thức setters/getters cho các thuộc tính của các lớp.
 - Tiền lương hằng tháng (*getMonthlySalary*) là tiền lương cơ bản + phụ cấp.

Phụ cấp được tính như sau:

 - Lập trình viên: Nếu ngôn ngữ lập trình là “Java” thì phụ cấp 20% của lương cơ bản.
 - Người quản lý: Phụ cấp 40% của lương cơ bản
 - Người trưởng dự án: Phụ cấp 20% lương cơ bản.
- Công ty phần mềm có tên gọi (*theName*), và danh sách các nhân viên.
 - Constructor *SoftwareHouse(aName : String)*, tạo 1 công ty có tên là aName và khởi tạo danh sách nhân viên n phần tử. (Mỗi phần tử trong mảng là 1 nhân viên)
 - Phương thức *addEmployee(emp : Employee) : boolean*, dùng để thêm 1 nhân viên emp vào công ty. Thêm thành công nếu không trùng mã số và mảng chưa đầy.
 - Phương thức *displayAllStaff() : void*, hiển thị toàn bộ nhân viên trong công ty lên màn hình theo dạng cột, định dạng đơn vị tiền tệ là \$, phân cách hàng nghìn.
 - Phương thức *getMonthlySalaryBill() : double*, tính tổng tiền phải trả cho các nhân viên.
 - Phương thức *sortAllStaffByName() : void*, sắp xếp danh sách nhân viên theo tên.
 - Phương thức *displayListProgrammers() : void*, hiển thị danh sách các lập trình viên.
 - Phương thức *updateDepartmentForAdmin(aPayrollNo: int, deptNew: String) : boolean*, cập nhật nhân phòng ban là deptNew, cho người quản lý có mã số là aPayrollNo.

Chương trình chính tạo menu case để thực hiện từng chức năng của hệ thống.

Bài 4. Quản lý thông tin các khóa học và thông tin sinh viên tham gia đăng ký học các khóa học.



<pre> public class Person { private String <u>lastName</u>; private String <u>firstName</u>; //... } </pre>	<pre> public class Student extends Person { private int <u>yearMatriculated</u>; private String <u>studentID</u>; //... } </pre>
<pre> public class Faculty extends Person { private String <u>office</u>; private String <u>facultyID</u>; //... } </pre>	<pre> public class Section { private String <u>semester</u>; private String <u>sectionNbr</u>; private String <u>room</u>; private Faculty <u>facultyMember</u>; private Set<Enrolment> <u>listEnrolls</u> ; //... } </pre>
<pre> public class Enrolment { private String <u>status</u>; private String <u>grade</u>; private float <u>numGrade</u>; private Student <u>student</u>; //... } </pre>	<pre> public class Course { private String <u>courseNbr</u>; private String <u>courseTitle</u>; private int <u>credits</u>; private Set<Section> <u>sections</u>; public int getNbrStudents(String sectionNbr){ //body of the method } public void addSection(String sectionNbr, String semester, String room, Faculty facultyMember, Set<Enrolment> listEnrolls){ //body of the method } } </pre>

Viết lớp cho phần thử nghiệm:

1. Tạo danh sách giảng viên
2. Tạo danh sách sinh viên
3. Tạo khóa học.
4. Lập danh sách sinh viên tham gia vào 1 học phần của khóa học do 1 giảng viên giảng dạy.
5. In ra màn hình theo mẫu:

```
Problems @ Javadoc Declaration Search Console Plug-in Dependencies Bug Explorer Bug Info Error Log
<terminated> Driver (20) [Java Application] C:\Java\jre1.8.0_25\bin\javaw.exe (Sep 12, 2015, 12:23:44 AM)
Khóa học : [OOP - Lập trình hướng đối tượng (4TC)]
===== Thông tin học phần =====
Mã học phần: 0602
Học kỳ: I (2015 - 2016)
Phòng học: H5.01
Giảng viên: Lê Kim Khánh (Khoa : CNTT)
===== Danh sách sinh viên =====
Mã SV      Họ tên      Khóa năm      Điểm
140211     Hoàng Dũng   2014           8.5
140511     Trần Bình    2014           9.5
140811     Lê Huỳnh     2014           7.0
140611     Hồ Huyền     2013           5.5

Tổng số sinh viên : 4
```

Bài 5. Công ty du lịch V quản lý thông tin là các chuyến xe. Thông tin của 2 loại chuyến xe:

- Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.
- Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu.

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Viết chương trình quản lý các chuyến xe theo dạng cây thừa kế với các phương thức sau:
 - Nhập, xuất danh sách các chuyến xe (danh sách có thể dùng cấu trúc mảng).
 - Tính tổng doanh thu cho từng loại xe.

Bài 6. Thư viện X quản lý danh sách các loại sách . Thông tin về các loại sách:

- Sách giáo khoa: Mã_sách, ngày nhập (ngày, tháng, năm), đơn giá, số lượng, nhà xuất bản, tình trạng (mới, cũ).

Nếu tình trạng sách là mới thì: thành tiền = số lượng * đơn giá.

Nếu tình trạng sách là cũ thì: thành tiền = số lượng * đơn giá * 50%

- Sách tham khảo: Mã sách, ngày nhập (ngày, tháng, năm), đơn giá, số lượng, nhà xuất bản, thuế.
Thành tiền = số lượng * đơn giá + thuế

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Nhập xuất danh sách các loại sách.
- Tính tổng thành tiền cho từng loại.
- Tính trung bình cộng đơn giá của các sách tham khảo.
- Xuất ra các sách giáo khoa của nhà xuất bản X.

Bài 7. Xây dựng chương trình quản lý danh sách các giao dịch. Hệ thống quản lý 2 loại giao dịch:

- Giao dịch vàng: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, số lượng, loại vàng.
Thành tiền được tính như sau:

thành tiền = số lượng * đơn giá.

- Giao dịch tiền tệ: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), Đơn giá, số lượng, tỉ giá, loại tiền tệ có 3 loại: tiền Việt Nam, tiền USD, tiền Euro. Thành tiền được tính như sau:

- Nếu là tiền USD hoặc Euro thì: thành tiền = số lượng * đơn giá * tỉ giá
- Nếu là tiền VN thì: thành tiền = số lượng * đơn giá

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Nhập xuất danh sách các giao dịch.
- Tính tổng số lượng cho từng loại.
- Tính trung bình thành tiền của giao dịch tiền tệ.
- Xuất ra các giao dịch có đơn giá > 1 tỷ.

Bài 8. Xây dựng chương trình quản lý danh sách các giao dịch nhà đất. Thông tin bao gồm:

- Giao dịch đất: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, loại đất (loại A, B, C), diện tích.
 - Nếu là loại B, C thì: thành tiền = diện tích * đơn giá.
 - Nếu là loại A thì: thành tiền = diện tích * đơn giá * 1.5
- Giao dịch nhà: Mã giao dịch, ngày giao dịch (ngày, tháng, năm), đơn giá, loại nhà (cao cấp, thường), địa chỉ, diện tích.
 - Nếu là loại nhà cao cấp thì: thành tiền = diện tích * đơn giá.
 - Nếu là loại thường thì: thành tiền = diện tích * đơn giá * 90%

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Nhập xuất danh sách các giao dịch.
- Tính tổng số lượng cho từng loại.
- Tính trung bình thành tiền của giao dịch đất.
- Xuất ra các giao dịch của tháng 9 năm 2013.

Bài 9. Xây dựng chương trình quản lý danh sách hoá 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 hoá 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 hoá đơn (ngày, tháng, năm), quốc tịch, số lượng, đơn giá. Thành tiền được tính = số lượng * đơn giá.

Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Nhập xuất danh sách các hóa đơn khách hàng.
- Tính tổng số lượng cho từng loại khách hàng.
- Tính trung bình thành tiền của khách hàng người nước ngoài.
- Xuất ra các hoá đơn trong tháng 09 năm 2013 (của cả 2 loại khách hàng).

Bài 10. (Yêu cầu chung: Xác định access modifier (private, protected, public) cho từng thuộc tính/ phương thức mỗi lớp, cài đặt getter/setter, cài đặt constructor mặc định và constructor có thông số đầy đủ).

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.

Vậy hãy 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, đánh giá, và phương thức toString trả về thông tin hóa đơn cho khách hàng (gồm các thuộc tính của đối tượng).
- Lớp **Quản lý** 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).

Bài 11. Hàng hóa quản lý trong kho của một siêu thị gồm có hàng thực phẩm, hàng sành sứ và hàng điện máy.

Mỗi loại hàng đều có mã hàng (không được sửa, không được để trống), tên hàng (không được rỗng), số lượng tồn (≥ 0), đơn giá (> 0).

Hàng thực phẩm thì cần quan tâm đến thông tin ngày sản xuất, ngày hết hạn (ngày hết hạn phải sau hoặc là ngày sản xuất) và nhà cung cấp.

Hàng điện máy cần biết thời gian bảo hành bao nhiêu tháng (≥ 0), công suất bao nhiêu KW (> 0).

Hàng sành sứ thì cần biết thông tin về nhà sản xuất và ngày nhập kho.

Ngoài ra, người quản lý cần quan tâm đến số lượng tồn kho và các yếu tố khác của từng loại hàng hóa để đánh giá mức độ bán buôn, tiền VAT từng loại hàng hóa. Biết rằng VAT của hàng điện máy và sành sứ là 10%, VAT của hàng thực phẩm là 5%.

a) Dựa vào các thông tin trên, hãy xác định:

- Các lớp có thể có. Lớp nào là lớp trừu tượng (abstract class), lớp nào là lớp cụ thể.

- Các thuộc tính cho từng lớp.
 - Các phương thức cho từng lớp (phương thức nào là phương thức trừu tượng (*abstract method*), danh sách các tham số có thể có cho từng phương thức và kiểu trả về của phương thức).
 - Thiết kế mô hình class (*xây dựng cây thừa kế, các giao diện nếu có*).
- b) Dùng java IDE, tạo một project. Thực hiện cài đặt tường minh cho mỗi loại hàng cụ thể trên. Trong đó, để đánh giá mức độ bán buôn thì:
- Hàng điện máy, nếu số lượng tồn kho <3 thì được đánh giá là bán được.
 - Hàng thực phẩm, nếu vẫn còn tồn kho và bị hết hạn thì đánh giá là khó bán.
 - Hàng sành sứ, nếu số lượng tồn kho >50 và thời gian lưu kho >10 ngày thì đánh giá là bán chậm.
 - Các trường hợp còn lại xem như không đánh giá.
- Hãy viết lớp quản lý danh sách hàng hóa. Dùng Array để lưu trữ danh sách hàng hóa.
 - Tạo constructor khởi tạo danh sách.
 - Viết phương thức thêm một hàng hóa vào danh sách (*thêm thành công nếu không bị trùng mã hàng, thể hiện tính đa hình – polymorphism bằng cách cho phép lựa chọn nhập thông tin*)
 - Viết phương thức in toàn bộ danh sách các hàng hóa.
 - Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

Bài 12. Với 1 tập mini các loại xe trong thế giới thực cho bên dưới:



Yêu cầu quản lý:

- Thông tin từng loại xe.
- Tính tiền thuế cho từng chiếc xe dựa trên giá trị xe như sau:
 - **Xe đạp:** Không đóng thuế.
 - **Xe máy:** VAT=10% và thuế trước bạ 5%.
 - **Xe ô tô khách:** Thuế tiêu thụ đặc biệt 30% (số chỗ >=5); 50% (số chỗ <5), thuế VAT=10%, thuế trước bạ 20%.
 - **Xe ô tô tải:** VAT=10%, thuế trước bạ 2%.

Yêu cầu sinh viên:

- Dùng kiến thức mô hình hóa dữ liệu trong lập trình hướng đối tượng để xây dựng các lớp.
- Sử dụng 1 CASE tool (*Computer Aided Software Engineering*) để thiết kế mô hình cây phân cấp các lớp. Mối quan hệ giữa các lớp và các interface.

- Generate source code từ mô hình trên sang ngôn ngữ lập trình Java.
- Hiện thực tường minh chương trình.

Bài 13. Để quản lý thông tin về sinh viên của trường ĐHCN, thông tin về sinh viên được tổ chức như sau:

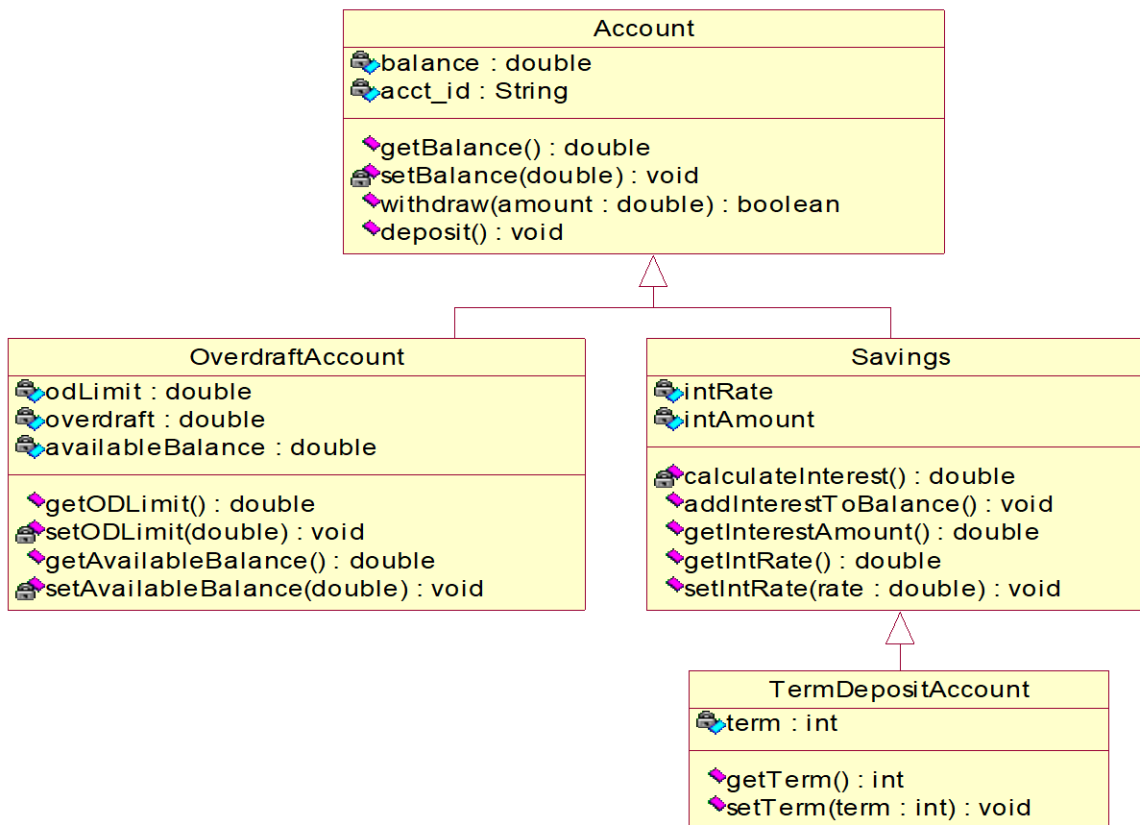
- Sinh viên trung cấp (trung cấp 2 năm)
 - Sinh viên cao đẳng
 - Sinh viên đại học
 - Sinh viên liên kết (Australia, US)
- a. Xác định các thông tin, dữ liệu chung, riêng.
 - b. Định nghĩa các lớp (các thuộc tính, và phương thức) và mô hình phân cấp các lớp
 - c. Viết lớp cho phép nhập và hiển thị thông tin về sinh viên.

Bài 14. Tạo lớp trừu tượng (*abstract class*) **Shape** với 3 phương thức trừu tượng **draw()** , **erase()**, and **move(int x, int y)** . Tạo các lớp con như liệt kê ở bảng dưới đây đồng thời ghi đè (overriding) các phương thức trừu tượng, các phương thức này in câu thông báo ra console.

Class	Superclass	Subclass
Shape	-	Circle, Quad, Triangle, Polygon
Circle	Shape	-
Quad	Shape	Rectangle
Rectangle	Quad	-
Triangle	Shape	-
Polygon	Shape	-

Viết lớp **Drawing** có phương thức **drawShape(Shape theShape)**, phương thức có tham số là đối tượng **Shape**. Trong phương thức gọi tới **draw()** của từng đối tượng Shape.

Bài 15. Cho mô hình như sau



- Sử dụng mô hình lớp ở trên cho cây phân cấp của các đối tượng Account, hãy tạo các phương thức để thi hành phân thiết kế này. Cung cấp các đoạn mã lệnh để thực thi các phương thức.
- Viết chương trình kiểm tra trong lớp BankingServices, tạo các đối tượng thuộc các lớp cụ thể và kiểm tra các phương thức.

Tuần 6.

Tuần 7. TẬP HỢP (COLLECTIONS)

Chương 5. Tập hợp trên Java

Mục tiêu:

- Hiểu và áp dụng được cách thao tác trên tập hợp
 - Khai báo và khởi tạo tập hợp
 - Các thao tác thêm, xóa, sửa, duyệt trên tập hợp
 - Các thuật toán sắp xếp, tìm kiếm trên tập hợp
- Bài tập bắt buộc 1, 2, 3

Bài 1. Công ty TrueLove:

- a) Công ty TrueLove cần lưu tên của các nhân viên của mình. Mỗi tháng một nhân viên sẽ được chọn ngẫu nhiên để nhận một quà tặng. Hãy dùng tuyển tập để viết chương trình quản lý danh sách nhân viên.
- b) Công ty TrueLove cần đặt tên cho sản phẩm mới, tên sản phẩm được chọn từ tên của nhân viên, vì vậy tên không được trùng, tên chỉ được dùng có 1 lần. Hãy dùng tuyển tập để viết chương trình cung cấp tên cho sản phẩm.
- c) Công ty TrueLove muốn dùng tên phổ biến nhất cho sản phẩm của họ, tên phổ biến là tên giống nhau nhiều nhất. Hãy dùng tuyển tập để viết chương trình cung cấp tên cho sản phẩm.
- d) Công ty TrueLove muốn cho nhân viên đi du lịch, chính sách được tạo ra là ưu tiên cho những người đăng ký trước. Hãy dùng tuyển tập để viết chương trình đăng ký du lịch.
- e) Công ty TrueLove muốn tạo danh sách các khách hàng theo thứ tự tăng dần theo doanh số. Hãy dùng tuyển tập để viết chương trình quản lý danh sách khách hàng.

Bài 2. Phòng học được quản lý trong một trường đại học gồm: phòng học lý thuyết, phòng máy tính và phòng thí nghiệm .

Mỗi phòng học đều có mã phòng, dãy nhà, diện tích, số bóng đèn.

Phòng học lý thuyết thì cần quan tâm xem có máy chiếu không.

Phòng máy tính thì cần biết là trang bị bao nhiêu máy tính.

Phòng thí nghiệm thì thêm thông tin chuyên ngành, sức chứa, có bồn rửa không (rửa dụng cụ thí nghiệm / rửa tay).

Ngoài ra, người quản lý cần phải xem xét phòng học có đạt chuẩn không.

Dùng java IDE, tạo một project được đặt tên theo quy định sau: TênLớp_TênSV_MSSV.

- Thực hiện cài đặt tường minh cho mỗi loại phòng cụ thể trên.
- Phòng học đạt chuẩn nếu: Tất cả các phòng đều phải đủ ánh sáng (trung bình $10m^2$ - 1 bóng đèn), và
 - Phòng lý thuyết, nếu có máy chiếu.
 - Phòng máy tính, nếu trung bình $1.5m^2$ đặt một máy.

- Phòng thí nghiệm, nếu có bồn rửa đi kèm.
- Hãy viết lớp quản lý danh sách phòng học. Dùng một List (ArrayList, LinkedList, Vector) để lưu trữ danh sách phòng học.
 - Tạo constructor khởi tạo danh sách.
 - Viết phương thức thêm một phòng học vào danh sách (*thêm thành công nếu không bị trùng mã phòng*).
 - Viết phương thức tìm kiếm một phòng học nào đó khi biết mã phòng.
 - Viết phương thức in toàn bộ danh sách các phòng học.
 - Viết các phương thức để in danh sách các phòng học đạt chuẩn.
 - Viết phương thức để sắp xếp danh sách tăng dần theo cột dãy nhà.
 - Viết phương thức để sắp xếp danh sách giảm dần theo cột diện tích.
 - Viết phương thức để sắp xếp danh sách tăng dần theo cột số bóng đèn.
 - Viết phương thức để cập nhật số máy tính cho một phòng máy tính nào đó khi biết mã phòng.
 - Viết phương thức để xóa một phòng học nào đó khi biết mã phòng. *Lưu ý khi test chương trình, khi xóa cần phải xác minh rằng có chắc chắn xóa không?*
 - Viết phương thức để in ra tổng số phòng học.
 - Viết các phương thức để in danh sách các phòng máy có 60 máy.
- Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

Bài 3. Quản lý khách hàng xếp hàng mua vé tại nhà ga. Thông tin lưu trữ cho khách hàng gồm: số CMND khách hàng (String), Tên khách hàng, Ga đến, giá tiền (double).

Hệ thống menu gồm các mục:

- Thêm một khách hàng mới vào hàng đợi mua vé.
- Bán một vé cho khách hàng. **Chỉ bán cho người đăng ký trước.**
- Hiện thị danh sách khách hàng.
- Hủy một khách hàng ra khỏi danh sách. (khách hàng không mua vé nữa).
- Thống kê tình hình bán vé
- Lưu danh sách vào file
- Hiện thị danh sách các ga đang chờ mua vé.
- Hiện thị danh sách các ga đang chờ mua vé và số vé tương ứng cho ga.

Lưu ý:

- Số khách hàng trong danh sách hiện tại là số khách đang chờ, nhưng chưa có vé. Khi một khách hàng đã mua vé, thì loại khách hàng này ra khỏi danh sách chờ mua vé.
- Việc mua vé phải có thứ tự: **ai vào trước thì mua vé trước (FIFO).**
- Mỗi khi khách hàng mua được vé phải lưu lại khách hàng này để dùng cho việc thống kê.
- Mỗi khi thêm một khách hàng mới, nếu Số CMND khách hàng đã có thì không tạo phần tử mới mà chỉ cập nhật lại ga và giá tiền đến cho khách hàng đó.
- Mục thống kê tình hình: cho biết còn bao nhiêu khách hàng chờ nhận vé, bao nhiêu khách hàng đã nhận vé, tổng số tiền đã thu về là bao nhiêu.

- Việc lưu danh sách: chỉ lưu các khách hàng chờ mua vé. Các khách hàng đã nhận vé xem như kết sổ trong ngày không cần lưu lại.
- Khi chương trình vừa được chạy, lập tức tự động nạp toàn bộ danh sách khách hàng từ file (cách khách hàng chưa có vé).
- Khi hiển thị danh sách các ga đến đang chờ mua vé, chỉ hiển thị tên ga đó một lần. (Ví dụ: giả sử 10 khách hàng nhưng đăng ký đi đến 2 ga, thì chỉ hiển thị 2 hàng).

Bài 4. Viết chương trình tạo 2 tập hợp các số nguyên (Set). Tính giao, hội, hiệu 2 tập trên, xuất ra kết quả tăng dần.

Hướng dẫn: - Dùng TreeSet

- a.addAll(b) → tập a hội tập b
- a.retainAll(b) → tập a giao tập b
- a.removeAll(b) → tập a trừ tập b

Bài 5. Sử dụng ArrayList để biểu diễn một vài chức năng của interface Collection. Chương trình thực hiện sử dụng 2 mảng Color trong ArrayLists và dùng Iterator để loại bỏ các phần tử trong mảng thứ 2 của tập hợp ArrayList từ mảng thứ nhất của tập hợp ArrayList.

Bài 6. Viết lớp mô tả các toán tử trên danh sách liên kết sử dụng LinkedList. Chương trình tạo 2 LinkedList chứa thông tin là các chuỗi String. Các phần tử của danh sách List này được đưa vào danh sách kia. Các chuỗi trong danh sách được chuyển sang chữ hoa, xoá các phần tử.

Bài 7. Viết chương trình tra cứu danh bạ điện thoại, sử dụng cấu trúc collection bất kỳ cho phù hợp để lưu trữ thông tin của danh bạ và dễ dàng thực hiện công việc:

- Tra cứu theo địa chỉ.
- Tra cứu theo số điện thoại, 1 địa chỉ có thể đăng kí nhiều số điện thoại cố định.

Tuần 8.

Tuần 9. LẬP TRÌNH GENERICS

Chương 6. Lập trình Generics

Mục tiêu:

- Hiểu và áp dụng được các kiểu Generic, phương thức Generic
- Hiểu và áp dụng được Generic theo lớp

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: D:\MaSV_HoTen\Tuan07-08\
- Máy tính phải được cài đặt sẵn JDK (Java Development Kit)
- Máy tính phải có sẵn phần mềm soạn thảo hỗ trợ cho lập trình hướng đối tượng dùng ngôn ngữ lập trình Java (Eclipse/JCreator/NetBeans)
- Trong cài đặt Generic sử dụng
 - E - Element (used extensively by the Java Collections Framework)*
 - K - Key*
 - N - Number*
 - T - Type, S, U, V etc. - 2nd, 3rd, 4th types*
 - V - Value*

Bài 1. Viết 1 lớp chung, constructor cho phép khởi tạo dữ liệu bất kỳ (xét trường hợp vừa là số vừa là chuỗi).

Hướng dẫn:

Tạo lớp Generic

```
class GenericClass<T>
{
    T ob;
    GenericClass(T o)
    {
        ob = o;
    }
    T getob()
    {
        return ob;
    }
    void showType()
    {
        System.out.println("Kieu cua T la " + ob.getClass().getName());
    }
}
```

Tạo lớp kiểm tra

```

public class GenericClassTest
{
    public static void main(String args[])
    {
        // Tao đối tượng cho GenericClass lưu trữ các số Integer.
        GenericClass<Integer> iOb = new GenericClass<Integer>(88);
        iOb.showType();

        // Không cần ép kiểu
        int v = iOb.getob();
        System.out.println("gia tri: " + v);

        // Tao đối tượng cho GenericClass lưu trữ các chuỗi String.
        GenericClass<String> strOb = new GenericClass<String>("Generics Test");
        strOb.showType();

        String str = strOb.getob();
        System.out.println("gia tri: " + str);
    }
}

```

Generic Term	Meaning
Set<E>	Generic Type , E is called formal parameter. Loại Generic, E được gọi là tham số hình thức.
Set<Integer>	Parametrized type , Integer is actual parameter here Tham số hóa loại, Integer là tham số thực.
<T extends Comparable>	Bounded type parameter
<T super Comparable>	Bounded type parameter
Set<?>	Unbounded wildcard
<? extends T>	Bounded wildcard type
<? Super T>	Bounded wildcards
Set	Raw type
<T extends Comparable<T>>	Recursive type bound

Bài 2. Tạo lớp Generic với 2 loại tham số.

Hướng dẫn:

Tạo lớp với 2 loại tham số

```

class TwoGenerics<T, V>
{
    T ob1;
    V ob2;
    TwoGenerics(T o1, V o2)
    {
        ob1 = o1;
        ob2 = o2;
    }
}

```

```

    }
    void showTypes()
    {
        System.out.println("Loai cua T la "
            + ob1.getClass().getName());
        System.out.println("Loai cua V la "
            + ob2.getClass().getName());
    }
    T getob1()
    {
        return ob1;
    }
    V getob2()
    {
        return ob2;
    }
}

```

Tạo lớp kiểm tra

```

public class TwoGenericsTest
{
    public static void main(String args[])
    {
        TwoGenerics<Integer, String> tgObj =
            new TwoGenerics<Integer, String>(88, "Generics");
        tgObj.showTypes();

        int v = tgObj.getob1();
        System.out.println("gia tri: " + v);

        String str = tgObj.getob2();
        System.out.println("gia tri: " + str);
    }
}

```

Bài 3. Thao tác với Generic lồng nhau (nested generic type). Tạo một danh sách List lưu trữ danh sách các danh sách (chuỗi).

Hướng dẫn:

```

import java.util.ArrayList;
import java.util.List;

public class ListOfLists
{
    public static void main(String[] args)
    {
        List<String> listOfStrings = new ArrayList<String>();
        listOfStrings.add("Hello again");
        List<List<String>> listOfLists =
            new ArrayList<List<String>>();
        listOfLists.add(listOfStrings);
        String s = listOfLists.get(0).get(0);
    }
}

```



```

        // tự thêm thông tin khác
        System.out.println(s); // kết quả "Hello again"
    }
}

```

Bài 4. Sử dụng interface Generic Comparable. Tạo một lớp Person bao gồm 2 thuộc tính họ và tên implements interface Comparable.

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;
    }
}

```

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 sắp xếp:");
        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 5. Thiết kế 1 lớp hoạt động như một là thư viện cho các loại Media sau: sách, video và báo chí.
Yêu cầu viết cả 2 cách: thông thường và Generic.

Hướng dẫn:

Cách viết thông thường

```

import java.util.List;
import java.util.ArrayList;

public class Library
{
    private List resources = new ArrayList();
    public void addMedia(Media x)
    {
        resources.add(x);
    }
    public Media retrieveLast()
    {
        int size = resources.size();
        if (size > 0)
        {
            return (Media)resources.get(size - 1);
        }
        return null;
    }
}

```

```

}

interface Media
{
}

interface Book extends Media
{
}

interface Video extends Media
{
}

interface Newspaper extends Media
{
}

```

Cách viết theo Generic

```

public class LibraryGeneric<E extends Media>
{
    private List<E> resources = new ArrayList<E>();
    public void addMedia(E x)
    {
        resources.add(x);
    }
    public E retrieveLast()
    {
        int size = resources.size();
        if (size > 0)
        {
            return resources.get(size - 1);
        }
        return null;
    }
}

```

Bài 6. Viết phương thức Generic tính max, min trong một tập hợp. Sử dụng cách viết có dùng wildcard.

Hướng dẫn:

```

import java.util.Arrays;
import java.util.Collection;
import java.util.Comparator;

class ComparatorsEx
{
    public static <T> T max(Collection<? extends T> coll,
        Comparator<? super T> cmp)
    {

```

```

T candidate = coll.iterator().next();
for (T elt : coll)
{
    if (cmp.compare(candidate, elt) < 0)
    {
        candidate = elt;
    }
}
return candidate;
}

public static <T extends Comparable<? super T>>
    T max(Collection<? extends T> coll)
{
    return max(coll, ComparatorsEx.<T> naturalOrder());
}

public static <T> T min(Collection<? extends T> coll,
    Comparator<? super T> cmp)
{
    return max(coll, reverseOrder(cmp));
}

public static <T extends Comparable<? super T>>
    T min(Collection<? extends T> coll)
{
    return max(coll, ComparatorsEx.<T> reverseOrder());
}

public static <T extends Comparable<? super T>>
    Comparator<T> naturalOrder()
{
    return new Comparator<T>()
    {
        public int compare(T o1, T o2)
        {
            return o1.compareTo(o2);
        }
    };
}

public static <T> Comparator<T> reverseOrder(final Comparator<T> cmp)
{
    return new Comparator<T>() {
        public int compare(T o1, T o2)
        {
            return cmp.compare(o2, o1);
        }
    };
}

public static <T extends Comparable<? super T>>
    Comparator<T> reverseOrder()
{
    return new Comparator<T>() {

```

```

        public int compare(T o1, T o2)
        {
            return o2.compareTo(o1);
        }
    };
}

public class ComparatorsExampleTest
{
    public static void main(String[] args)
    {
        Comparator<String> sizeOrder = new Comparator<String>() {
            public int compare(String s1, String s2) {
                return s1.length() < s2.length() ? -1 : s1.length() > s2.length() ? 1 :
s1.compareTo(s2);
            }
        };
        Collection<String> strings = Arrays.asList("AAA", "aaa", "CCC", "f");

        System.out.println(ComparatorsEx.max(strings));
        System.out.println(ComparatorsEx.min(strings));
        System.out.println(ComparatorsEx.max(strings, sizeOrder));
        System.out.println(ComparatorsEx.min(strings, sizeOrder));
    }
}

```

Bài 7. Viết phương thức Generic cho phép in ra mảng các phần tử, phương thức này cho phép in phần tử mảng của nhiều kiểu dữ liệu khác nhau.

Hướng dẫn:

```

public class PrintArrayGeneric
{
    // phương thức Generic
    public static <E> void printArray(E[] inputArray)
    {
        // hiển thị các phần tử mảng
        for (E element : inputArray)
            System.out.printf("%s ", element);

        System.out.println();
    }

    public static void main(String args[])
    {
        // 3 mảng dữ liệu khác nhau: Integer, Double, Character
        Integer[] integerArray = { 1, 2, 3, 4, 5, 6 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
        Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };

        System.out.println("Mang so nguyen:");
        printArray(integerArray); // tham số là mảng số nguyên
    }
}

```

```

        System.out.println("\nMang doubleArray co noi dung:");
        printArray(doubleArray); // tham số là mảng Double
        System.out.println("\nMang characterArray co noi dung:");
        printArray(characterArray); // tham số là mảng ký tự
    }
}

```

Bài 8. Sử dụng Generic tham số, dùng ký tự đại diện wildcard ?. Viết phương thức Generic cho phép in ra mảng các phần tử, phương thức sử dụng tham số Generic.

Hướng dẫn:

```

import java.util.ArrayList;
import java.util.List;

public class GenericParameter
{
    public static void printList (List<?> list)
    {
        for (Object element : list)
        {
            System.out.println(element);
        }
    }
    public static void main(String[] args)
    {
        List<String> list1 = new ArrayList<String>();

        list1.add ("Hello");
        list1.add ("World");
        printList (list1);

        List<Integer> list2 = new ArrayList<Integer>();
        list2.add(100);
        list2.add(200);
        printList(list2);
    }
}

```

Bài 9. Sử dụng bounded wildcard trong phương thức. Viết phương thức Generic cho phép tính trung bình các giá trị trong mảng.

Hướng dẫn:

Cú pháp: `GenericType<? extends upperBoundType>`

```

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)); // KQ?
}
}

```

Bài 10. 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 11. Sử dụng bounded type trong lớp và wildcard ?. Viết lớp Generic cho phép tính trung bình các giá trị trong mảng và so sánh giá trị trung bình.

Hướng dẫn:

Viết lớp GenericStats dùng bounded type và phương thức so sánh 2 đối tượng thuộc lớp GenericStats dùng wildcard.

```

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

    GenericStats(T[] obj)
    {
        nums = obj;
    }

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

    boolean sameAvg(GenericStats<?> ob)
    {
        if(average() == ob.average())
            return true;

        return false;
    }
}

```

Viết lớp kiểm tra

```

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

```



```

{
    Integer inums[] = { 1, 2, 3, 4, 5 };
    GenericStats<Integer> iob = new GenericStats<Integer>(inums);
    double v = iob.average();
    System.out.println("Trung binh cua iob " + v);

    Double dnums[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
    GenericStats<Double> dob = new GenericStats<Double>(dnums);
    double w = dob.average();
    System.out.println("Trung binh cua dob " + w);

    Float fnums[] = { 1.0F, 2.0F, 3.0F, 4.0F, 5.0F };
    GenericStats<Float> fob = new GenericStats<Float>(fnums);
    double x = fob.average();
    System.out.println("Trung binh cua fob " + x);

    System.out.print("Trung binh cua iob va dob ");
    if(iob.sameAvg(dob))
    {
        System.out.println("giiong nhau.");
    }
    else
    {
        System.out.println("khac nhau.");
    }

    System.out.print("Trung binh cua iob va fob ");
    if(iob.sameAvg(fob))
    {
        System.out.println("giiong nhau.");
    }
    else
    {
        System.out.println("khac nhau.");
    }
}
}

```

Bài 12. 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();
    }
}

```

Lưu ý: Thao tác ép kiểu trong Generic. Chỉ có thể ép kiểu một đối tượng thuộc lớp sang đối tượng của lớp khác trong trường hợp tương thích và tham số là như nhau.

```

class Gen<T>
{
    T ob;

    Gen(T o)
    {
        ob = o;
    }

    T getObject()
    {
        return ob;
    }
}

class Gen2<T> extends Gen<T>
{
    Gen2(T o)
    {
        super(o);
    }
}

```

```

    }
}

public class CastGeneric
{
    public static void main(String args[])
    {
        Gen<Integer> intObject = new Gen<Integer>(88);
        Gen2<Long> longObject = new Gen2<Long>(99L);

        //longObject = (Gen2<Long>)intObject;
    }
}

```

Bài 13. Thao tác với Generic Interface. Viết Interface với 2 phương thức max, min để tính max, min trong một tập hợp. Lớp thực thi interface phải định nghĩa chi tiết việc xử lý của 2 phương thức này.

Cú pháp chung

1. type-param-list is a comma-separated list of type parameters.
2. When a generic interface is implemented, you must specify the type arguments

```

interface interface-name<type-param-list> { // ...

class class-name<type-param-list>
    implements interface-name<type-param-list> {

```

```

interface MinMax<T extends Comparable<T>>
{
    T min();

    T max();
}

class MyClass<T extends Comparable<T>> implements MinMax<T>
{
    T[] vals;

    MyClass(T[] o)
    {
        vals = o;
    }

    public T min()
    {

```

```

        T v = vals[0];

        for (int i = 1; i < vals.length; i++)
            if (vals[i].compareTo(v) < 0)
                v = vals[i];

        return v;
    }

    public T max()
    {
        T v = vals[0];

        for (int i = 1; i < vals.length; i++)
            if (vals[i].compareTo(v) > 0)
                v = vals[i];

        return v;
    }
}

public class GenericInterfaceTest
{
    public static void main(String args[])
    {
        Integer inums[] = { 3, 6, 2, 8, 6 };
        Character chs[] = { 'A', 'r', 'V', 'w' };

        MyClass<Integer> iob = new MyClass<Integer>(inums);
        MyClass<Character> cob = new MyClass<Character>(chs);

        System.out.println("Max value in inums: " + iob.max());
        System.out.println("Min value in inums: " + iob.min());

        System.out.println("Max value in chs: " + cob.max());
        System.out.println("Min value in chs: " + cob.min());
    }
}

```

Bài 14. 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:

Lớp từ 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;

    public EVWordClass()
    {
        this("", "", "", "");
    }

    public EVWordClass(String word, String mean, String type, String notes)
    {
        this.word = word;
        this.mean = mean;
        this.type = type;
        this.notes = notes;
    }

    public String getMean()
    {
        return mean;
    }
    public void setMean(String mean)
    {
        this.mean = mean;
    }
    public String getNotes()
    {
        return notes;
    }
    public void setNotes(String notes)
    {
        this.notes = notes;
    }
    public String getType()
    {
        return type;
    }
    public void setType(String type)
    {
        this.type = type;
    }
    public String getWord() {
        return word;
    }
    public void setWord(String word) {
        this.word = word;
    }
}
```

```

}

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());
}
}

```

Lớp từ điển sử dụng Collections TreeMap

```

import java.util.TreeMap;
public class EVDictionary
{
    public TreeMap<String,EVWordClass> dic;

    // Tao TreeMap bao gom tu va
    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;
    }
}

```

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 15. Một quân bài trong bộ bài gồm hai thuộc tính loại bài (cơ, rô, chuồn, bích) và thứ tự quân bài(2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A). Dùng Generic.

Viết các lớp sau:

- Lớp mô tả quân bài
- Lớp tạo bộ bài 52 quân bài không trùng nhau với phương thức là xáo bài (dùng *shuffle()*)
- Lớp kiểm tra

Hướng dẫn:

Lớp mô tả quân bài

```
public class Card
{
    private int face;
    private int suit;
    public Card()
    {
        this.face = (int) (Math.random()*4);
        this.suit = (int) (Math.random()*13);
    }
}
```

```

}
public int getFace()
{
    return face;
}
public void setFace(int face)
{
    this.face = face;
}
public int getSuit()
{
    return suit;
}
public void setSuit(int suit)
{
    this.suit = suit;
}

public boolean equals(Object obj)
{
    Card c = (Card)obj;
    return (this.suit == c.getSuit() && this.face == c.getFace());
}
// Ham override

public String toString()
{
    String ret = "";
    switch(suit)
    {
        case 0: ret = "At";break;
        case 1: ret = "Hai";break;
        case 2: ret = "Ba";break;
        case 3: ret = "Bon";break;
        case 4: ret = "Nam";break;
        case 5: ret = "Sau";break;
        case 6: ret = "Bay";break;
        case 7: ret = "Tam";break;
        case 8: ret = "Chin";break;
        case 9: ret = "Muoi";break;
        case 10: ret = "J";break;
        case 11: ret = "Q";break;
        case 12: ret = "K";break;
    }
    switch(face)
    {
        case 0: ret += " Co";break;
        case 1: ret += " Ro";break;
        case 2: ret += " Chuon";break;
        case 3: ret += " Bich";break;
    }
    return ret;
}
}

```


Lớp mô tả bộ bài

```
import java.util.ArrayList;
import java.util.Collections;

public class Card_Pack
{
    private ArrayList<Card> pack;
    public Card_Pack()
    {
        pack = new ArrayList<Card>();
        int count =0;
        do
        {
            Card c = new Card();
            if(!pack.contains(c))
            {
                pack.add(c);
                count ++;
            }
        }while(count<52);
    }

    public void shuffleCardPack()
    {
        Collections.shuffle(pack);
    }
    public String toString()
    {
        String ret = "";
        for(Card c:pack)
            ret += c.toString() + "\n";
        return ret;
    }
}
```

Lớp kiểm tra chương trình

```
public class CardTesting
{
    public static void main(String[] args)
    {
        Card_Pack cp=new Card_Pack();
        System.out.println(cp);

        System.out.println("\nBAI SAU KHI XAO: \n");

        cp.shuffleCardPack();
        System.out.println(cp);
    }
}
```

Tuần 10.

Chương 7. ÔN TẬP VÀ KIỂM TRA THỰC HÀNH

Chương 8. Nhập xuất trên Java

Mục tiêu:

- Hiểu và áp dụng được đóng mở tập tin, thư mục
- Hiểu và áp dụng được các thao tác với tập tin, thư mục

Bài 1. Viết chương trình nhập vào một chuỗi và in ra chuỗi nghịch đảo của chuỗi nhập (Dùng BufferedReader và InputStreamReader).

Bài 2. Hiện thị nội dung của một file tên test.txt lưu tại D:\test.txt

Dùng BufferedInputStream thao tác đọc tập tin.

```
import java.io.*;

public class BufferedFileApp
{
    public static void main(String args[]) throws IOException
    {
        BufferedInputStream bStream = new BufferedInputStream(
            new FileInputStream("D:\\test.txt"));
        int ch=0;
        while ((ch=bStream.read()) != -1)
        {
            System.out.print((char)ch);
        }
        bStream.close();
    }
}
```

Bài 3. Copy nội dung một file text đến một file text khác. (Dùng BufferedInputStream/BufferedOutputStream Hoặc dùng FileInputStream/FileOutputStream)

```
import java.io.*;

public class CopyFileApp
{
    public static void main(String args[]) throws IOException
    {
        if (args.length!=2)
        {
            System.out.println("Usage : java CopyFileApp <SrcFile> <DestFile>");
            return;
        }

        String SourceFile=args[0]; // tập tin nguồn
        String DestFile =args[1]; // tập tin copy
        // Tạo bộ đệm đọc dữ liệu từ tập tin nguồn
        BufferedInputStream inFile = new BufferedInputStream(
```

```

        new FileInputStream(SourceFile));
// Lấy kích thước tập tin nguồn
int FileSize = inFile.available();
// Tạo bộ đệm ghi dữ liệu vào tập tin đích
BufferedOutputStream outFile = new BufferedOutputStream(
        new FileOutputStream(DestFile));

// Chuyển dữ liệu
int ch=0;
while ((ch=inFile.read()) != -1)
{
    outFile.write(ch);
}

System.out.println(FileSize + " bytes đã được copy xong.");

inFile.close();
outFile.close();
}
}

```

Bài 4. Dùng DataOutputStream và DataInputStream để ghi và đọc những kiểu dữ liệu khác nhau trên file.

Bài 5. Liệt kê danh sách các thư mục con và tập tin của 1 thư mục. Nếu thư mục, hiển thị thêm <DIR> phía trước của tên.

```

import java.io.*;
public class FileApp
{
    public static void main(String args[])
    {
        File curDir=new File("C:\\");
        String[] dirs=curDir.list();

        for (int i=0; i<dirs.length; i++)
        {
            File f=new File("C:\\"+dirs[i]);
            if (f.isDirectory())
            {
                System.out.println("<DIR>    "+dirs[i]);
            }
            else
            {
                System.out.println("        "+dirs[i]);
            }
        }
    }
}

```

Bài 6. Truy cập ngẫu nhiên trên file, viết chương trình ghi 6 số kiểu double xuống file, rồi đọc lên theo thứ tự ngẫu nhiên

Bài 7. Thực hiện đọc ghi đối tượng dùng ObjectOutputStream và ObjectInputStream.

```
import java.io.*;
import java.util.*;
public class ObjectWriteExApp
{
    public static void main(String args[])
        throws IOException
    {
        ObjectOutputStream oStream = new ObjectOutputStream(
            new FileOutputStream("Container.txt"));

        // ghi đối tượng String
        oStream.writeObject(new String("Hello World"));
        // ghi đối tượng Fruit
        oStream.writeObject(new Fruit("Orange",10));
        oStream.writeObject(new Fruit("Apple",5));
        // ghi đối tượng Date
        oStream.writeObject(new Date());
        System.out.println("Ghi 4 doi tuong vao tap tin Container.txt");
    }
}

class Fruit implements Serializable
{
    String name="";
    int weight=0;

    public Fruit(String n,int w){
        name =n;
        weight=w;
    }

    private void writeObject(ObjectOutputStream out)
        throws IOException{
        out.writeObject("X "+name);
        out.writeInt(weight-1);
    }

    private void readObject(ObjectInputStream in)
        throws IOException,ClassNotFoundException{
        name =(String)in.readObject();
        weight=in.readInt();
    }
    public String toString()
    {
        return (name + " " + weight+" g");
    }
}
```

Bài 8. Dùng BufferedReader đọc từng ký tự từ Console. Việc đọc kết thúc khi gặp dấu chấm (dấu chấm để kết thúc chương trình).

Bài 9. Dùng BufferedReader đọc chuỗi ký tự từ Console. Chương trình kết thúc khi gặp chuỗi đọc là chuỗi “stop”.

Bài 10. Định nghĩa lớp lưu trữ thông tin của sinh viên. Cho phép nhập dữ liệu sinh viên và lưu trữ thành file data.dat. Đọc dữ liệu từ tập tin, đưa vào mảng và hiển thị kết quả. Thông tin của SV bao gồm mã, họ tên, địa chỉ, số điện thoại và điểm trung bình của năm học vừa qua.