BÀI TẬP MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

(OJBECT ORIENTED PROGRAMMING EXERCISES)

HỆ: ĐẠI HỌC

MỤC LỤC

Module 0. LÀM QUEN VỚI ECLIPSE IDE

Mục tiêu:

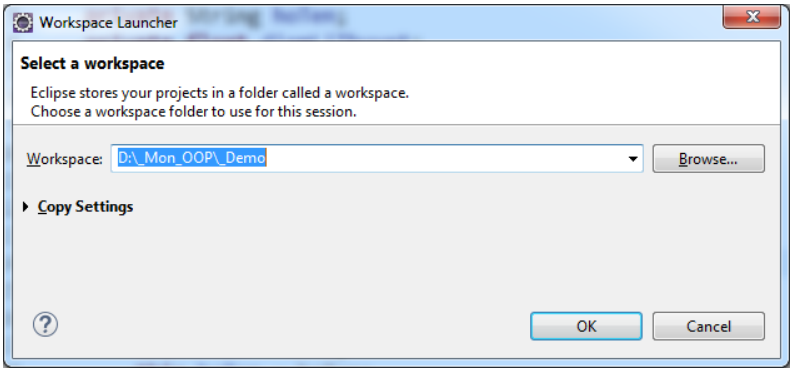
- *Làm quen với công cụ lập trình Java (Eclipse).*
- *Tạo workspace (nơi lưu project), tạo project Java, tạo package.*
- *Thay đổi workspace.*

Yêu cầu:

- *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).*

Bài 1. Khởi động Eclipse

Khi khởi động Eclipse, ở lần đầu tiên, Eclipse sẽ xuất hiện cửa sổ hỏi nơi lưu trữ các project (workspace). Ở những lần mở sau, Eclipse sẽ nhớ workspace này và sẽ tự mở. Có thể thay đổi workspace này.

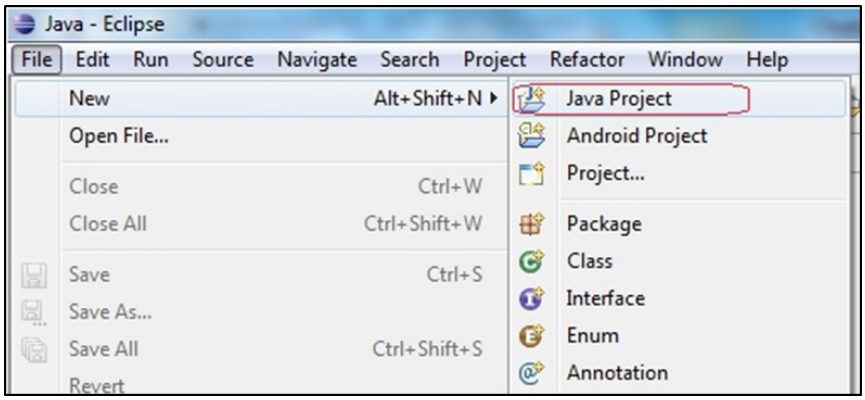


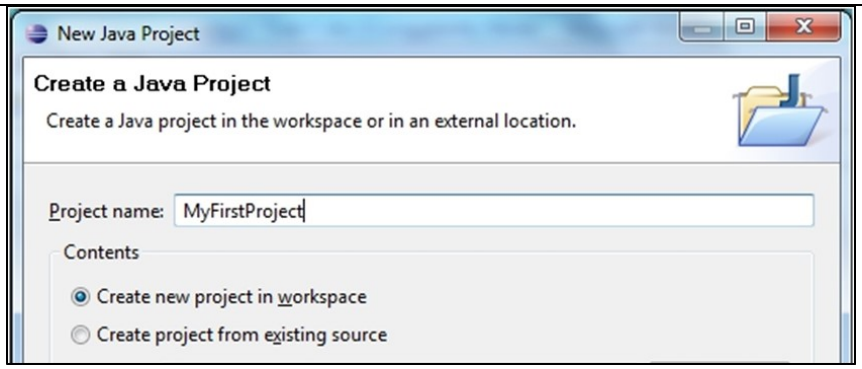
Bài 2. Thay đổi workspace

Vào File → Switch Workspace.

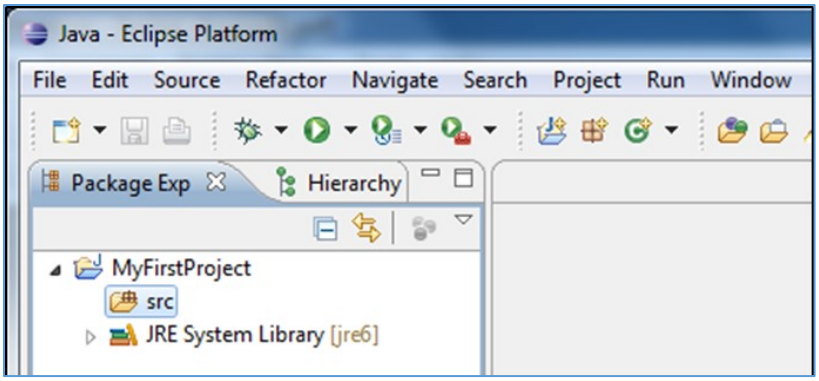
Bài 3. Tạo Project

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





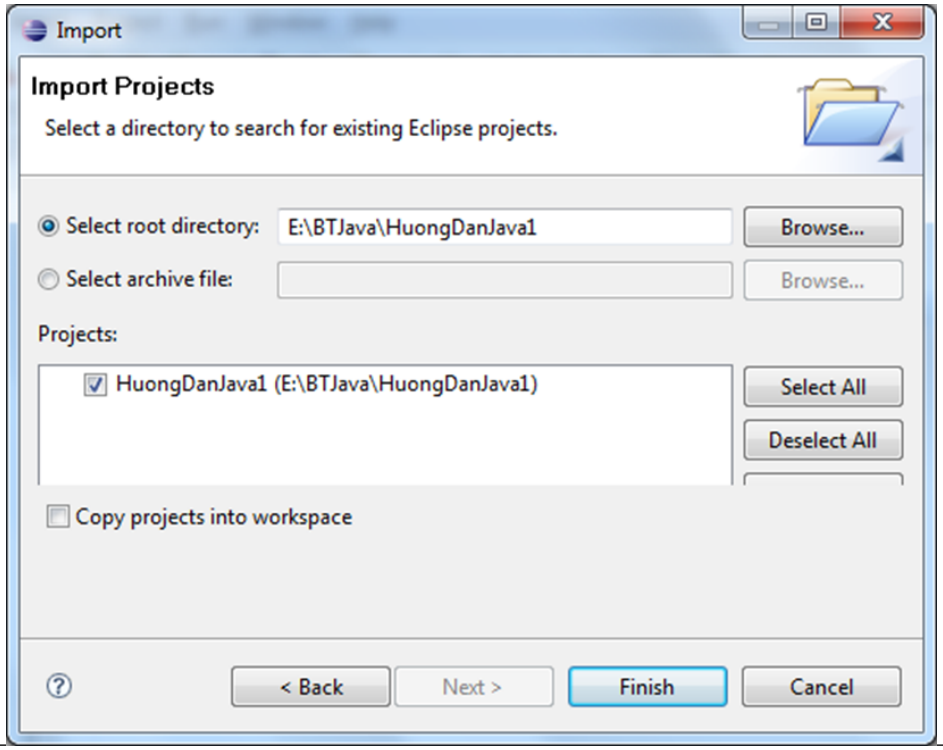
Nhấn Finish. Kết quả trong Project Explorer.



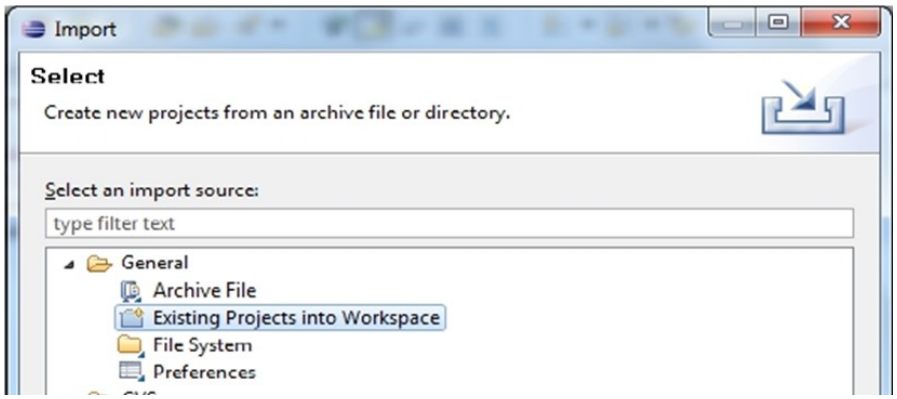
Bài 4. Mở Project

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.



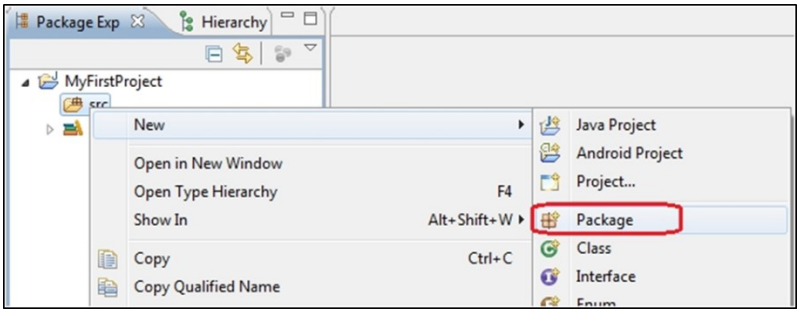
Chọn Project cần Import rồi nhấn Finish.

Hoặc mở cả workspace bằng cách khởi động Eclipse trước.

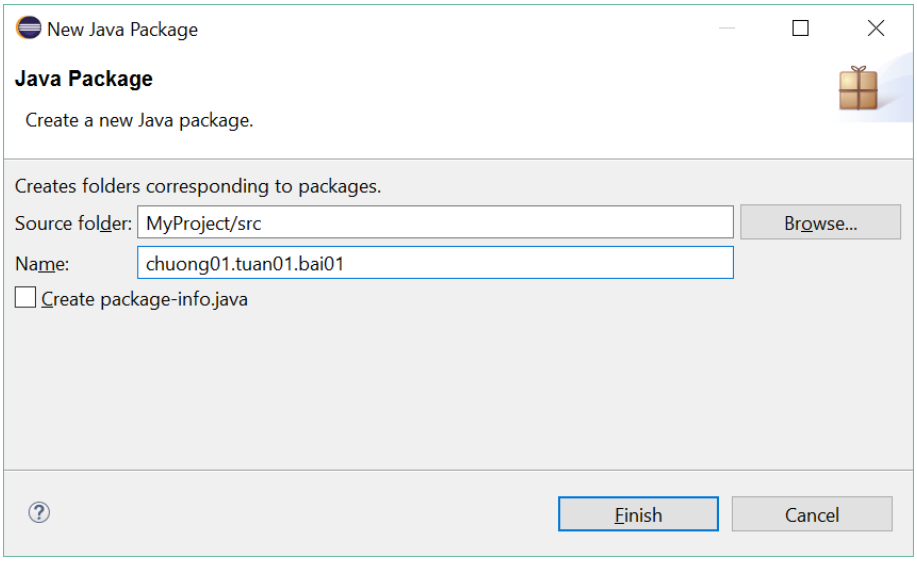
Bài 5. Tạo package

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).

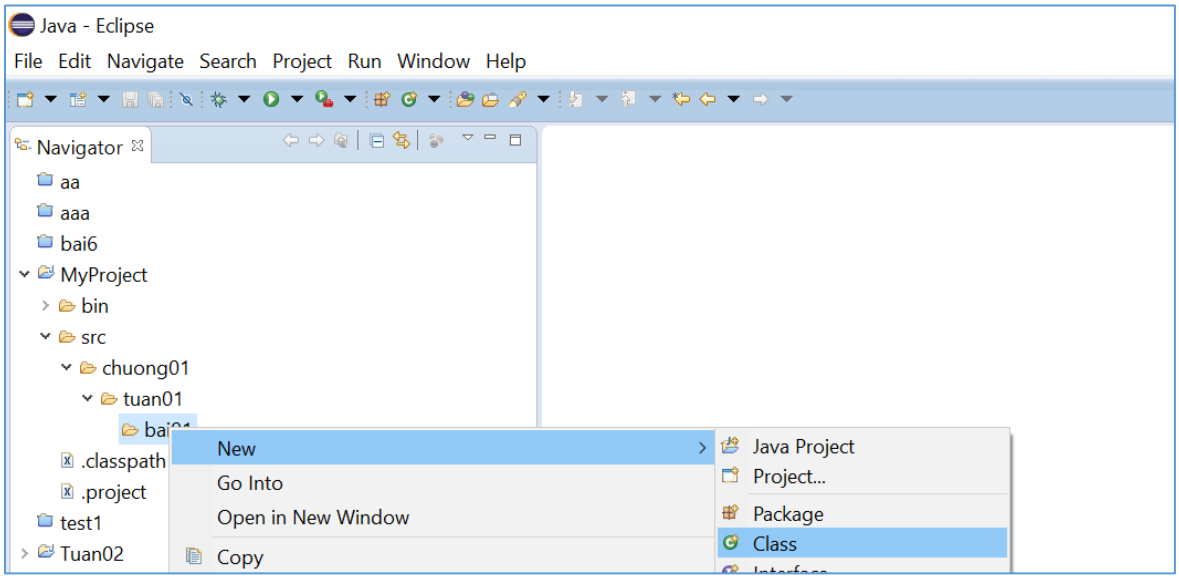
Mỗi ứng dụng có thể có 1 hoặc nhiều package. Mỗi package chứa một hoặc nhiều class.

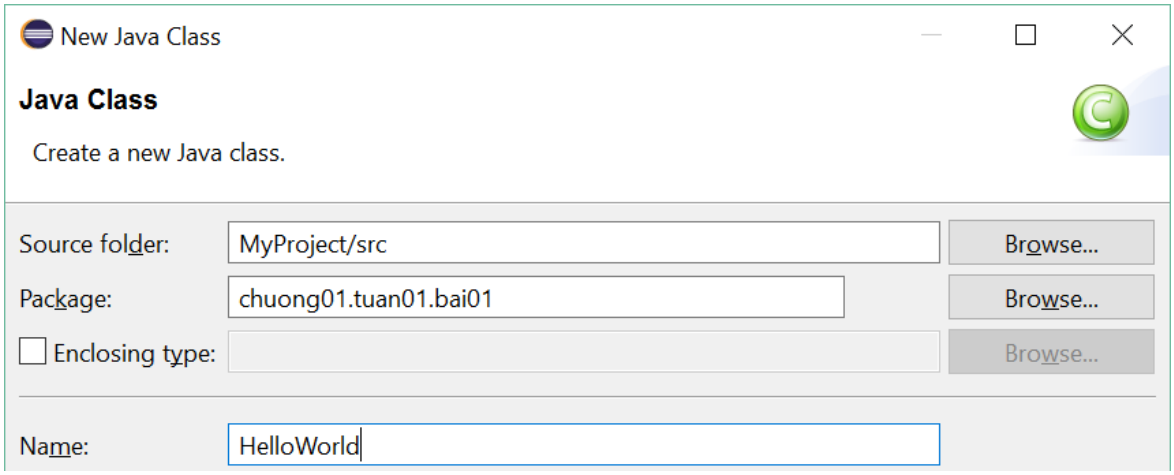


Đặt tên theo kiểu: a.b.c trong đó các ký tự là tên bất kỳ. Ví dụ: chuong01.tuan01.bai01 Điều đó có nghĩa là Eclipse sẽ tạo cho bạn 3 thư mục: chuong01\tuan01\. **Lưu ý: các gói luôn được đặt tên bằng chữ thường.**



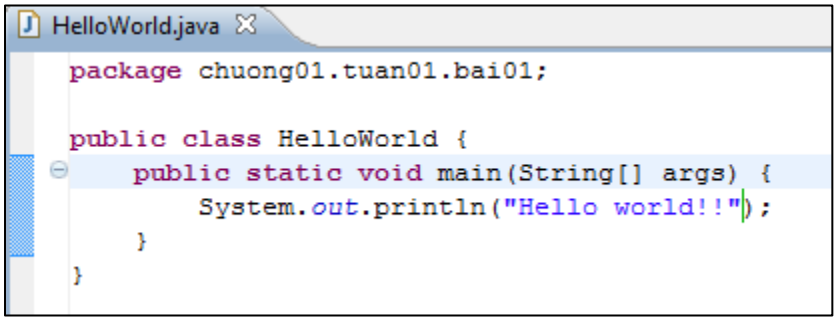
Tạo lớp mới bằng cách nhấn phải chuột lên package cần thêm lớp vào, chọn New → Class





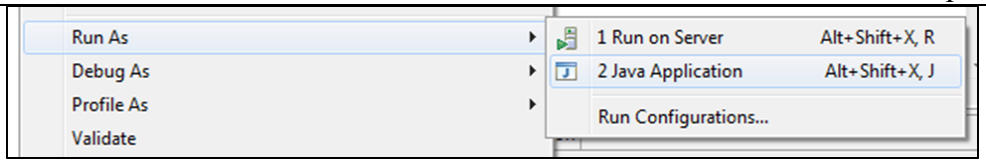
Chú ý: Tên lớp luôn bắt đầu bằng 1 ký tự hoa. Đặt theo kiểu Title-Case

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



Bài 6. Thực thi chương trình

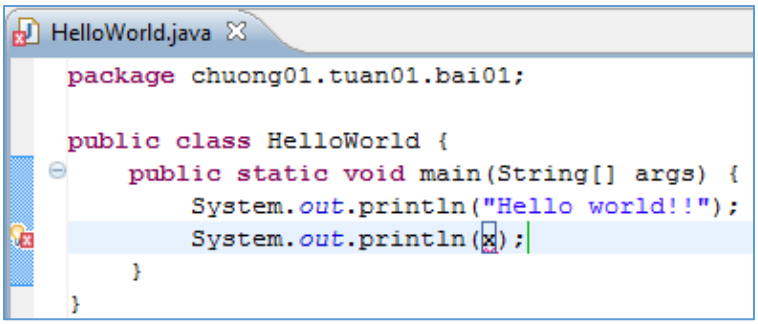
Nhấn chuột phải lên lớp cần chạy, chọn menu Run As-> Java Application.



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

Eclipse sẽ tự động biên dịch code và báo lỗi.

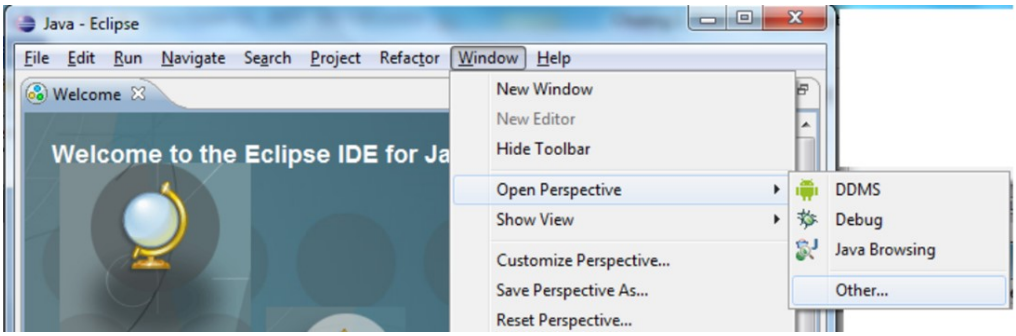
Nếu bạn có lỗi hay warning thì bên trái của dòng lỗi. Ví dụ như sau:



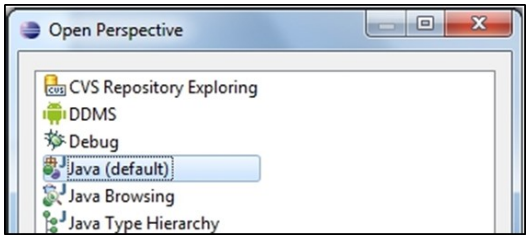
Bài 7. Chọn loại giao diện làm việc

1. Khởi động Eclipse IDE

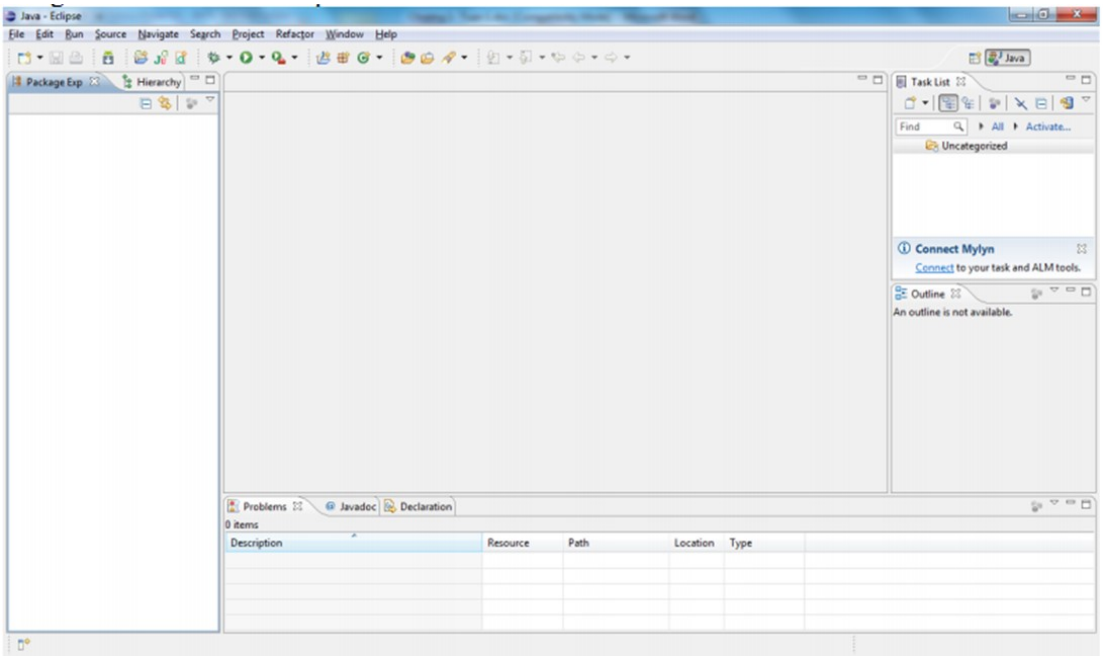
2. Chọn Windows\Open Perspective\Other



Chọn Perspective Java(Default)



Đóng Welcome screen. Kết quả



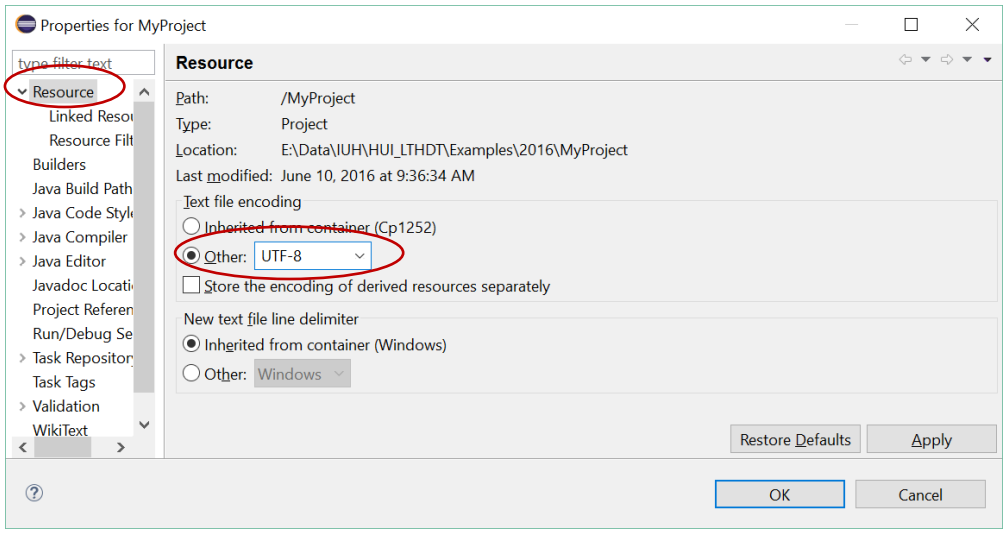
Bài 8. Đổi tên (project, package, class…)

Nhấn chọn tên cần đổi ở cửa sổ Package Explorer → F2 → đánh tên mới là xong.

Bài 9. Vấn đề gõ tiếng Việt (Unicode)

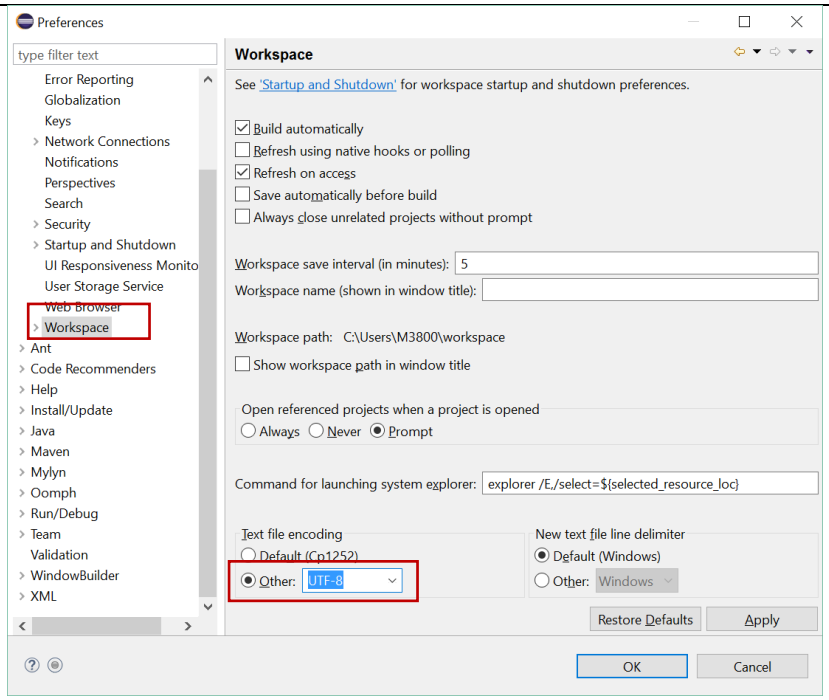
Java sử dụng bảng mã unicode nên việc gõ tiếng việt là OK. Để gõ được tiếng việt, đảm bảo là project của bạn phải được lưu với bảng mã UTF-8.

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 (khỏi mắc công mỗi Project mỗi thiết lập), ta làm như sau: Vào menu Window->References, chọn mục General-> Workspace như hình.

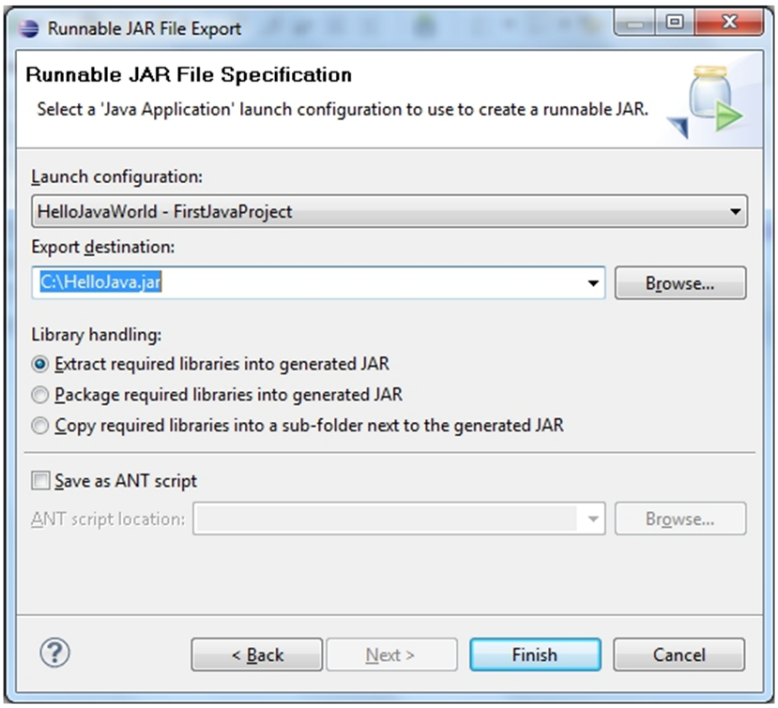


Nhấn Apply. Từ đây, bất cứ project nào tạo ra đều hỗ trợ Unicode.

Bài 10. Export file jar tự chạy (executable jar file)

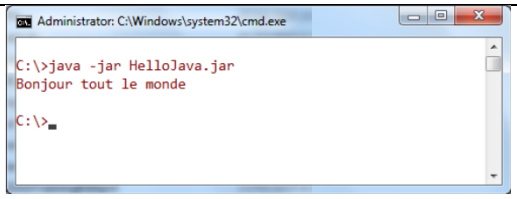
Nhấn chuột phải lên Project cần export, chọn Export.

Chọn Runnable JAR file như hình. Nhấn Next.

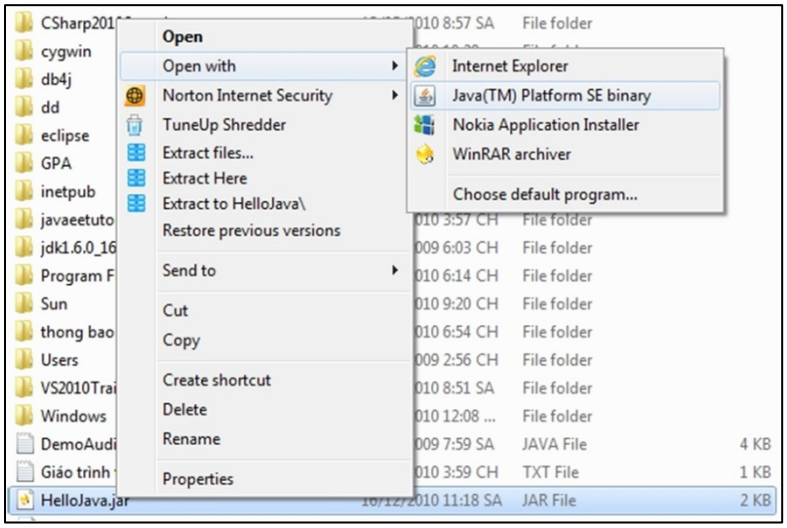


Chọn Lauch configuration. Chọn thư mục chứa cũng như tên file jar. Nhấn Finish.

Thực thi jar file dưới dạng command-line:



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

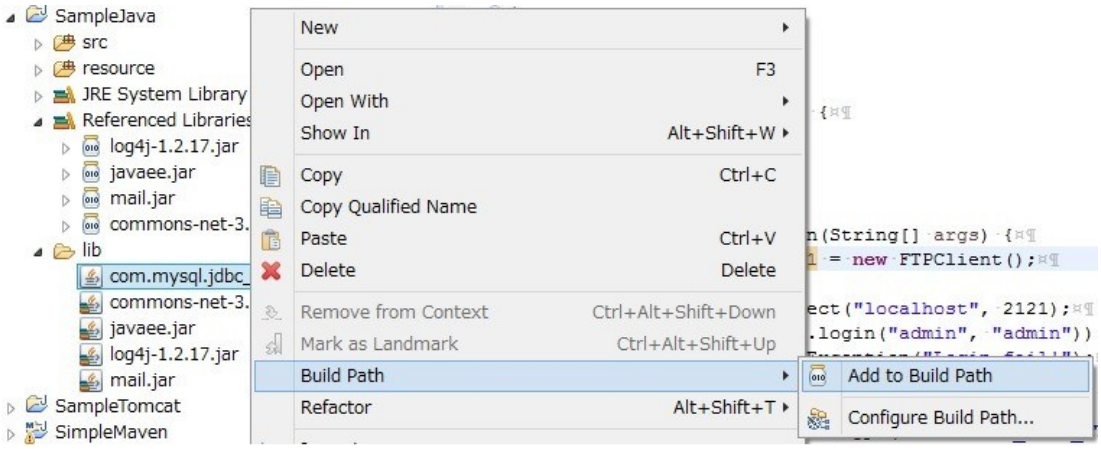


Bài 11. Thêm thư viện cho Project

1. Thêm thư viện jar

Để thêm thư viện Jar mình thường làm theo 2 bước sau

- Tạo thự mục chứa File jar (thường để tên là lib) -> copy file jar và thư viện đó.
- Thực hiện add jar file bằng cách : chọn chuột phải file jar -> Build Path -> Add to Build Path là xong (như hình bên dưới)

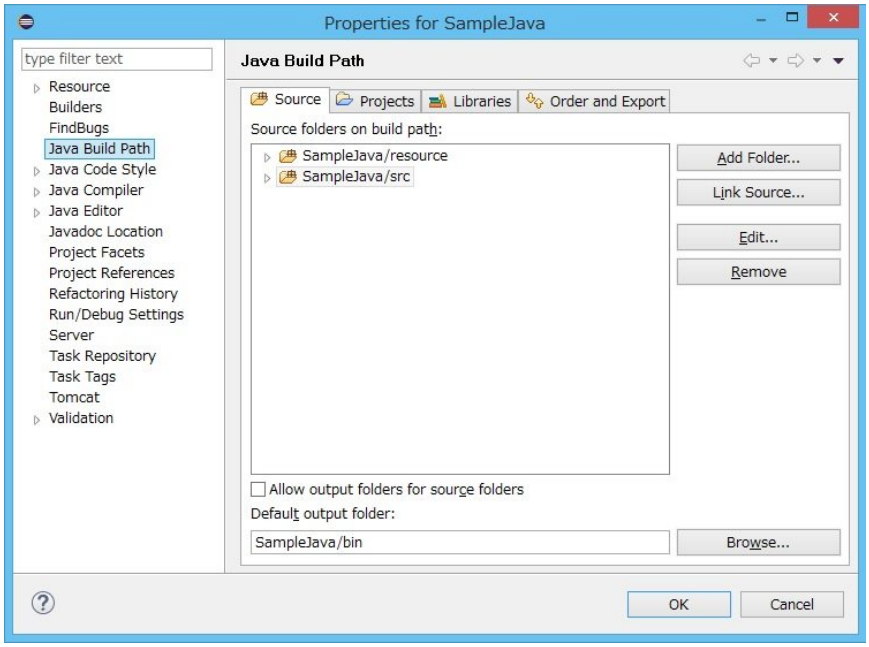


Bỏ thư viện jar

Vào phần Referenced Libraries -> chọn chuột phải vào File jar -> Build Path -> Remove from Build Path là xong.

2. Thêm thư viện ngoài , refer source code ...

Chọn Build Path -> Configure Build Path



Source : Phần này bạn có thể add một folder source hoặc là link đến thư mục source nào đó.

Project : Tương tự như source nhưng đây là refer sang một project cùng workspace.

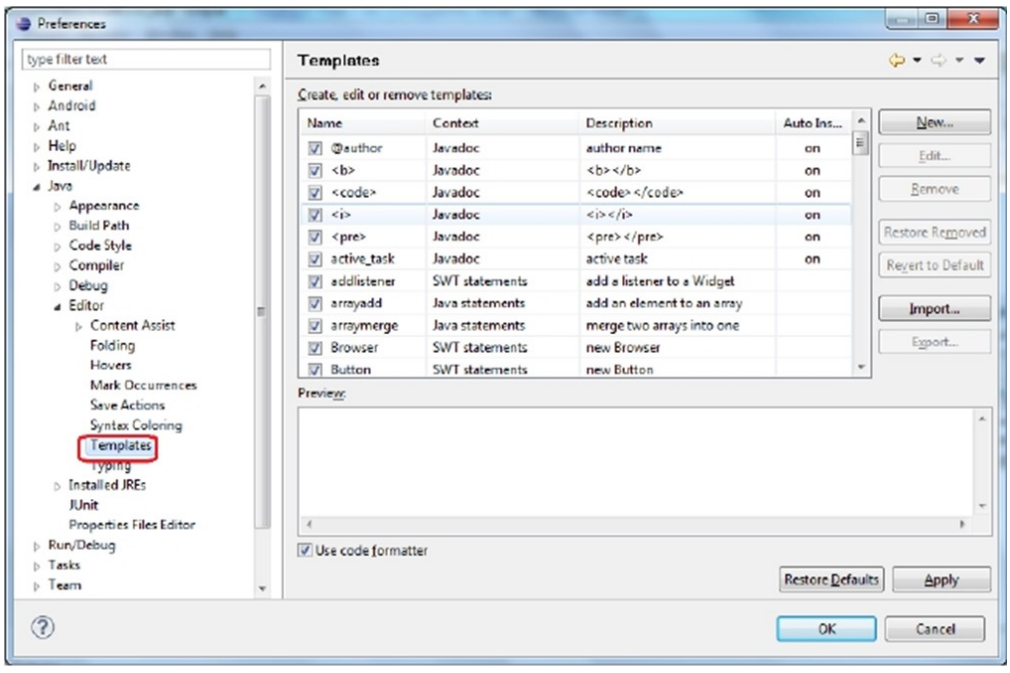
Bài 12. Phím tắt hay dùng

1. Một số editing template hay dùng:

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

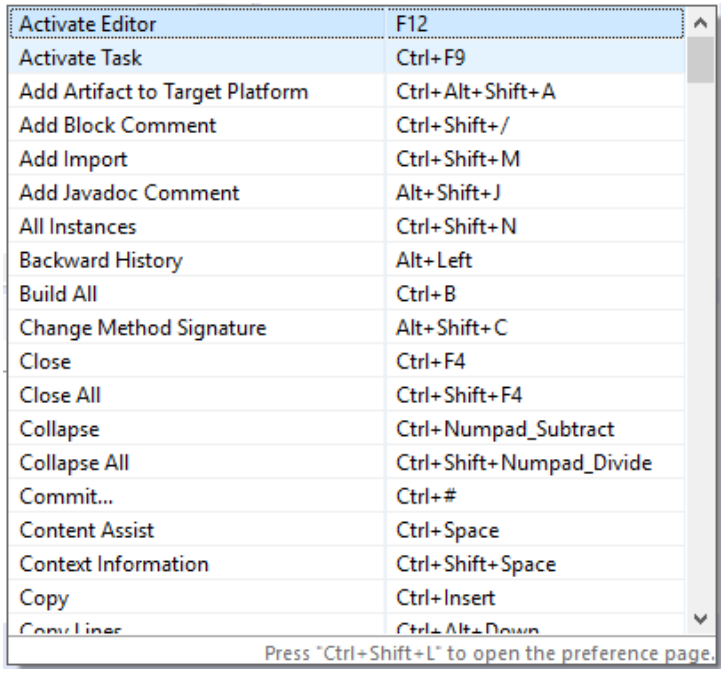
Gõ sysout sau đó nhấn Ctrl-Spacebar sẽ cho System.out.println();

Để tìm hiểu thêm, vào menu Window->Reference



2. Các phím tắt hay dùng:

Sử dụng phím tắt trong Eclipse sẽ giúp bạn thao tác nhanh hơn thay vì phải click chuột nhiều lần và trông chuyên nghiệp hơn. Trong Eclipse, vào menu Help->Key Assitst hoặc nhấn tổ hợp Ctrl-Shift-L để hiển thị.



Activate Editor	F12
Activate Task	Ctrl+F9
Add Artifact to Target Platform	Ctrl+Alt+Shift+A
Add Block Comment	Ctrl+Shift+/
Add Import	Ctrl+Shift+M
Add Javadoc Comment	Alt+Shift+J
All Instances	Ctrl+Shift+N
Backward History	Alt+Left
Build All	Ctrl+B
Change Method Signature	Alt+Shift+C
Close	Ctrl+F4
Close All	Ctrl+Shift+F4
Collapse	Ctrl+Numpad_Subtract
Collapse All	Ctrl+Shift+Numpad_Divide
Commit...	Ctrl+#
Content Assist	Ctrl+Space
Context Information	Ctrl+Shift+Space
Copy	Ctrl+Insert
Copy Lines	Ctrl+Alt+Down

Press 'Ctrl+Shift+L' to open the preference page.

Module 1. JAVA CĂN BẢN

Mục tiêu:

- *Làm quen với ngôn ngữ lập trình Java.*
- *Hiểu được cấu trúc chương trình Java, cách biên dịch và chạy chương trình dùng NNLT 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:

- *Bài làm lưu trong workspace có tên MaSV_HoTen.*
- *Tạo project tên Module01.*
- *Lần lượt tạo các package bai01, bai02, bai03 ứng với 3 phần của Module 1.*
- *Mỗi bài tập là một class với tên tương ứng, ví dụ Bai01, Bai02,…*

Nhắc lại: Cấu trúc của một 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
   {
         /* nội dung hàm chính */
   }
}
```

Bài 1. PHẦN LIÊN QUAN ĐẾN NHẬP XUẤT

Mục tiêu: *Làm quen với việc nhập xuất trong Java*

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

Bài bắt buộc: 1,2

1. Viết chương trình xuất ra màn hình dòng chữ "Hello World!"

Lưu ý sử dụng lệnh xuất ra màn hình:

- Sytstem.out.print(): xuất ra màn hình một chuỗi hay một giá trị (không xuống dòng).
- Sytstem.out.println(): xuất ra màn hình một chuỗi hay một giá trị (có xuống dòng).
- Sytstem.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: số nguyên (byte, short, int, long).
- %f: số thực (float, double).
- %c: ký tự (char).
- %s: chuỗi (string).
- %0: fill số 0.
- Độ chính xác của số thực: %5.3f: độ chính xác phần lẻ của số trong format-string là 3
- Dấu – dùng để canh trái, mặc định canh phải.
- Một số ký tự đặc biệt: \a, \b, \f, \n, \r, \t, \v, \\

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

2. Viết chương trình nhập vào tên của mình và xuất ra màn hình "Hello + Tên".

HD:

- 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();

Bài 2. 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

1. 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);
    }
}
```

2. 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);
    }
}
```

3. 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);
    }
}
```

4. 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?

5. 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?

6. 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);
    }
}
```

Bài 3. PHẦN CẤU TRÚC LẶP VÀ CẤU TRÚC ĐIỀU KHIỂN

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: 3,4,5,6,7,8,9,10 (yêu cầu viết phương thức để xử lý)

1. 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);
    }
}
```

2. 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
boolean nextBoolean();	A true or false value
int nextInt()	An integral value between **Integer.MIN_VALUE** and**Integer.MAX_VALUE**
long nextLong()	A long integral value between **Long.MIN_VALUE**and **Long.MAX_VALUE**
float nextFloat()	A decimal number between 0.0 (included) and 1.0 (excluded)
double nextDouble()	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.

3. Viết chương trình in ra tổng 1+3+5 …. +n nếu n là số chẵn, 2+4+6+ …. n nếu n là số lẻ.
4. Viết chương trình giải phương trình bậc 1.
5. Viết chương trình tìm USCLN của 2 số nhập vào.
6. Viết chương trình kiểm tra số nhập vào có phải là số nguyên tố hay không.
7. Tính tổng các số nguyên tố nhỏ hơn N
8. Tính tổng N số nguyên tố đầu tiên.
9. Viết chương trình in ra số lần kí tự ‘a’ xuất hiện trong một chuỗi.
10. Viết hàm tách chuỗi gốc thành chuỗi con.

    VD: chuỗi gốc S = “Bai Tap Mon Lap Trinh Java”, chuỗi sau khi tách là

    “Bai

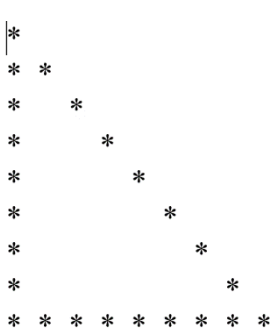
    Tap

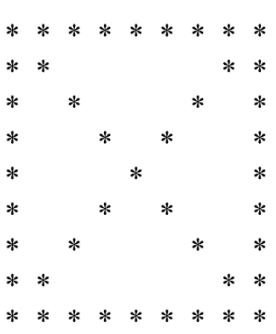
    Mon

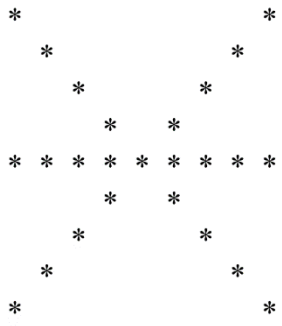
    Lap

    Trinh

    Java”
11. Viết hàm để đếm số lượng ký tự là số có trong chuỗi s. Chuỗi s được nhập từ bàn phím. *HD: Dùng mã ASCII để kiểm tra hoặc dùng class Character: Character.isDigit(ký tự) để kiểm ký tự có phải là số hay không.*
12. Viết chương trình in ra tổng của 10 số chẵn đầu tiên (sử dụng vòng lặp for hoặc while).
13. Viết chương trình in ra những số lẻ từ 1 đến 99.
14. Viết chương trình xuất ra tổng các số là bội số của 7 (từ 1 đến 100).
15. Viết chương trình in ra giá trị lớn nhất và nhỏ nhất trong một dãy các giá trị user đã nhập.
16. Viết chương trình đọc một giá trị nguyên từ bàn phím và in ra số đó là số chẵn, lẻ hoặc zero.
17. Viết chương trình in ra bội số của 3 từ 300 đến 3.
18. Viết chương trình nhập vào số nguyên n và thực hiện: Xuất ra màn hình n số đầu tiên của chuỗi Fibonaci (có hai giá trị đầu là 1 và 1).
19. Viết chương trình in ra những hình sau: (mỗi hình sử dụng những vòng lặp khác nhau).

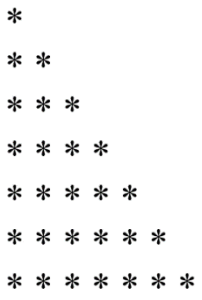






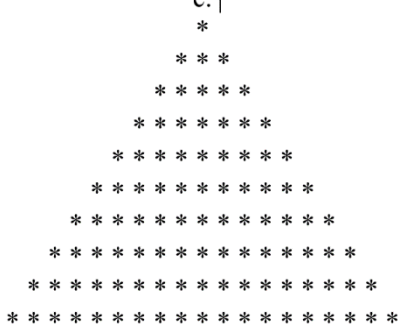
20. Viết chương trình nhập vào M và N, xuất ra các hình sau (dùng cấu trúc lặp):

a.

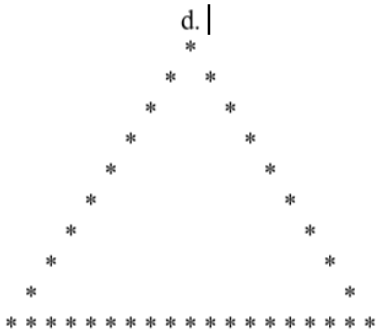


b

c.



d.



Module 2. CÁC KHÁI NIỆM CƠ BẢN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

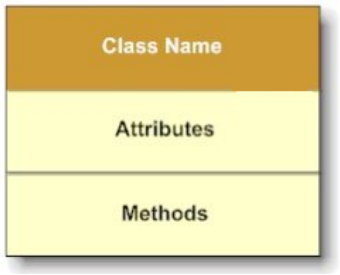
Mục tiêu:

- *Cài đặt lớp đối tượng.*
- *Tạo, sử dụng đối tượng.*
- *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).*

Yêu cầu:

- *Bài làm lưu trong workspace có tên MaSV_HoTen.*
- *Tạo project tên Module02.*
- *Mỗi bài làm trong một package có tên package là bai01, bai02,…*
- *Bài tập bắt buộc: 1, 3, 4, 5, 7, 8, 9.*
- *Viết documentation comment cho class.*

Định nghĩa một lớp trong UML:



Cách truy xuất (visibility):

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;
    }
}
```

Bài 1.

Biết lớp tọa độ có:

- Các thuộc tính: tọa độ x, tọa độ y và tên tọa độ.
- Các phương thức thiết lập (set), lấy (get) thông tin x, y và tên tọa độ.
- Các phương thức khởi tạo: một constructor mặc định và một construtor đầy đủ tham số.
- Phương thức toString trả về thông tin theo mẫu: tên tọa độ(x,y).

Yêu cầu:

a. Vẽ lược đồ lớp UML cho lớp tọa độ.
b. Cài đặt lớp tọa độ theo thiết kế.
c. Xây dựng lớp khác chứa hàm main cho phần kiểm nghiệm.

Bài 2.

a. Viết lớp Sinh viên như sau:

Attributes:

- Mã sinh viên (số nguyên>0),
- Họ tên (chuỗi, không được rỗng),
- Điểm LT, điểm TH (số thực, thuộc [0.0,10.0]).

Constructors:

- Constructor mặc định (gán giá trị mặc định),
- Constructor 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 (trung bình=(điểm LT + điểm TH)/2),
- Phương thức toString để diễn tả đối tượng ở dạng chuỗi có định dạng như hình kết quả.

Yêu cầu kiểm tra ràng buộc dữ liệu, gán giá trị mặc định khi dữ liệu không hợp lệ.

b. 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 bạn (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) như hình bên dưới.

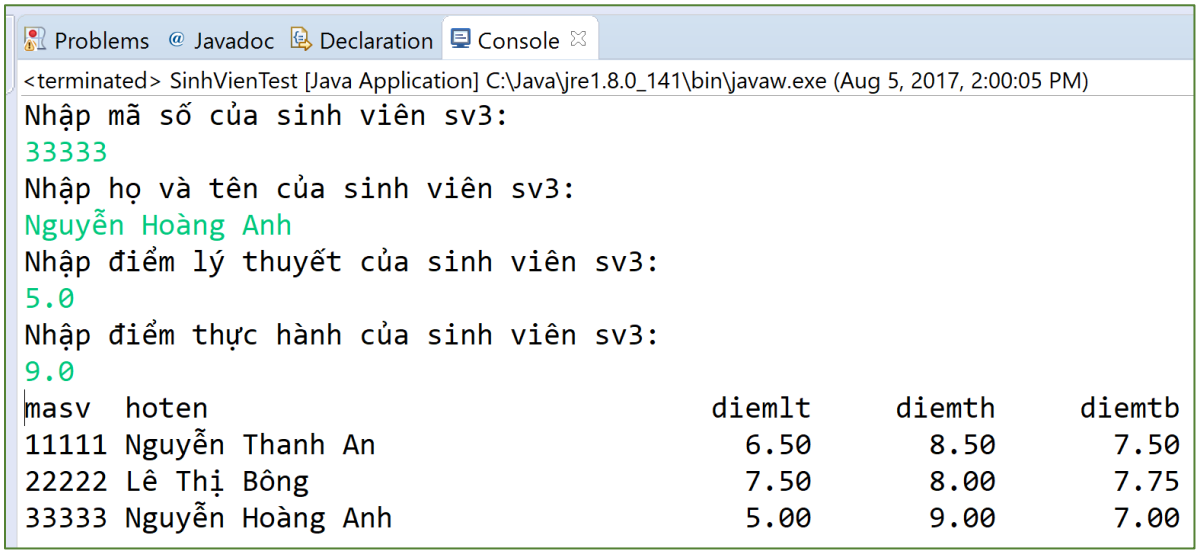
HD: Thông tin sinh viên in trên một dòng có định dạng. Sử dụng String.format("chuỗi định dạng", đối số 1, đối số 2, ….); Trong đó chuỗi định dạng giống c++, ví dụ:

"%-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



Kết quả thực thi chương trình.

HD: Tham khảo code ở phụ lục.

Bài 3.

a. Cài đặt lớp hình tam giác, biết tam giác có 3 cạnh ma, mb, mc:
   - Constructor đủ tham số: nếu giá trị truyền có số âm hoặc nếu 3 giá trị không lập thành hình tam giác thì gán 3 thuộc tính bằng 0.
   - Các phương thức getter/setter: nếu giá trị không hợp lệ thì không gán (giữ lại giá trị cũ).
   - Các phương thức tính chu vi, tính diện tích hình tam giác.
   - Phương thức trả về thông tin kiểu tam giác (thường, cân, đều, không phải tam giác).
   - Phương thức toString để diễn tả đối tượng ở dạng chuỗi gồm: thông tin 3 cạnh, kiểu tam giác, chu vi, diện tích.

b. Viết hàm main tạo 5 hình tam giác: 2 hình vi phạm ràng buộc, 3 hình là tam giác thường, cân, đều. Xuất thông tin các hình này theo dạng bảng.

HD:

- *Ba giá trị lập thành một hình tam giác khi và chỉ khi tổng hai cạnh bất kỳ luôn lớn hơn cạnh còn lại.*
- *Công thức tính diện tích tam giác:* $S = \sqrt{p(p-a)(p-b)(p-c)}$ *với* $p = \frac{a+b+c}{2}$.

Bài 4.

Cho mô tả bài toán:

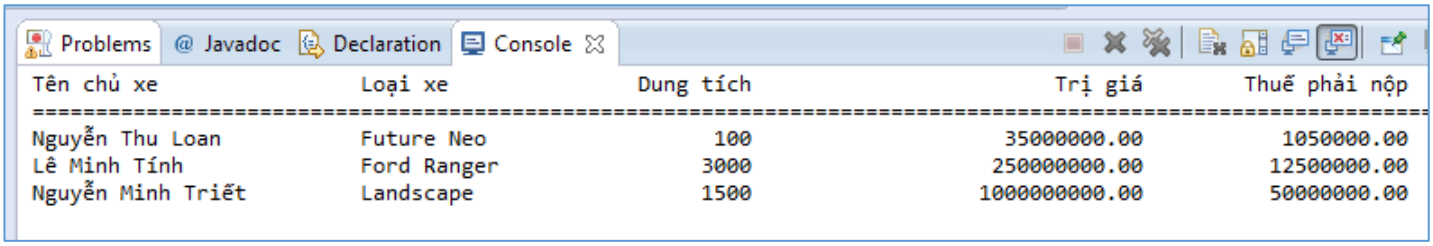
Sở giao thông cần theo dõi việc đăng ký xe (Vehicle) của người dân, biết mỗi xe cần lưu các thông tin là: *chủ xe, loại xe, trị giá xe* (>=0)*, dung tích xylanh* (>=0). Dựa vào thông tin *trị giá xe* và *dung tích xylanh*, sở giao thông cũng tính mức thuế phải đóng trước bạ khi mua xe như sau:

- Dưới 100cc, 1% trị giá xe.
- Từ 100 đến 200cc, 3% trị giá xe.
- Trên 200cc, 5% trị giá xe.

a. Hãy thiết kế và cài đặt class Vehicle với các attributes và methods phù hợp (có kiểm tra ràng buộc dữ liệu). Class phải có các constructor và phải bảo đảm tính encapsulation.

b. Xây dựng class chứa hàm main, có các công việc:

- Tạo 3 đối tượng Vehicle xe1, xe2, xe3. Dữ liệu được gán sẵn hoặc cho người dùng nhập.
- Xuất bảng kê khai tiền thuế trước bạ của các xe như mẫu:



Mẫu kết xuất của chương trình.

Bài 5. *

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, mặc định là "xxx")*, **đơn giá** *(>=0)*, **ngày sản xuất** (*phải trước ngày hiện tại, mặc định là ngày hiện tại*) và **ngày hết hạn** *(phải sau ngày sản xuất, mặc định là ngày sản xuất)*. Yêu cầu:

a. Khi gán giá trị, *nếu dữ liệu không hợp lệ thì gán giá trị mặc định* cho phép tương ứng của trường đó.

b. Viết các phương thức setters/getters (*có kiểm tra ràng buộc*) cho lớp HangThucPham.

c. Tạo một constructor có đầy đủ tham số; một constructor có tham số là mã hàng. *Nếu mã hàng rỗng thì phát sinh lỗi và không cho phép tạo đối tượng đó, các giá trị khác nếu không hợp lệ thì để mặc định.*

d. Viết phương thức kiểm tra một hàng thực phẩm đã hết hạn chưa.

e. Phương thức toString trả về chuỗi chứa thông tin của hàng thực phẩm, bao gồm thông tin đã hết hạn chưa. *Trong đó: định dạng đơn giá có phân cách hàng nghìn; định dạng kiểu ngày là dd/MM/yyyy.*

f. Viết lớp khác chứa hàm main: Tạo 3 đối tượng, trong đó có một đối tượng có ngày hết hạn trước ngày sản xuất. Xuất thông tin 3 mặt hàng này như mẫu:

```
Mã_Hàng Tên_Hàng           Đơn_Giá    Ngày_Sản_Xuất   Ngày_Hết_Hạn    Ghi_Chú
-------------------------------------------------------------------------------
001     Gạo          100,000.00VND   10/07/2018      10/07/2018
002     Mì             5,000.00VND   01/03/2018      01/09/2018
003     Nước          10,000.00VND   01/03/2017      01/03/2018      Hàng hết hạn
```

g. Kiểm thử lớp HangThucPham với các ràng buộc đã cho.

HD:

- ***Kiểm tra mã hàng:***

```
private void setMaHang(String maHang) throws Exception {
    if (!maHang.trim().equals(""))
        this.maHang = maHang;
    else
        throw new Exception("Lỗi: Mã hàng rỗng!");
}
```

- ***Kiểu ngày*** *dùng lớp định nghĩa sẵn LocalDate.*
- ***Kiểm tra ngày này trước hoặc sau ngày kia:*** *dùng phương thức isBefore, isAfter.*
- ***Kiểm tra hàng đã hết hạn chưa:***

```
public boolean hetHan() {
    return ngayHetHan.isBefore(LocalDate.now()) ? true : false;
}
```

- ***Định dạng số phân cách hàng nghìn:*** `DecimalFormat df = new DecimalFormat("#,##0.00");`
- ***Định dạng kiểu ngày là dd/MM/yyyy:*** `DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");`

Bài 6.

Cài đặt cho lớp tài khoản (Account) được thiết kế như sau:

Account
- accountNumber : long - name : String - balance : double - RATE = 0.035 : final double
<<constructors>> + Account() + Account(accNumber : long, name : String, balance : double) + Account(accNumber : long, name : String) <<property get>> + getAccountNumber() : long + getBalance() : double

<<other methods>>
+ deposit(amount : double) : boolean
+ withdraw(amount : double, fee : double) : boolean
+ addInterest() : void
+ transfer(acc2 : Account, amount : double): boolean
+ toString() : String

Mô tả:

- Ba constructor: **constructor mặc định** (số tài khoản = 999999, tên tài khoản = "chưa xác định", số dư = 50000); **constructor đầy đủ tham số**; **constructor có hai tham số**. *Trong trường hợp giá trị tham số không hợp lệ thì gán giá trị mặc định cho thuộc tính tương ứng (số tài khoản >0, tên tài khoản khác rỗng, số dư >=50000).*
- **deposit**(*amount* : double): cho phép gởi thêm số tiền *amount* vào tài khoản, hàm trả về *true* nếu gởi thành công (*amount*>0).
- **withdraw**(*amount* : double, *fee* : double): cho phép rút số tiền *amount* từ tài khoản, hàm trả về *true* nếu rút tiền thành công (*amount*>0 và *amount*+*fee* <= *balance*).
- **addInterest**(): tính tiền lãi, *balance* = *balance* + *balance* * *RATE*.
- **transfer**(*acc2* : Account, *amount* : double): chuyển một khoản tiền *amount* từ *account* này sang *account* kia, trả về true nếu chuyển thành công.
- **toString** trả về chuỗi chứa toàn bộ thông tin tài khoản, yêu cầu định dạng kiểu 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", "mã QG"). Ví dụ:

Locale local = new **Locale**("vi", "vn");

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

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

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

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

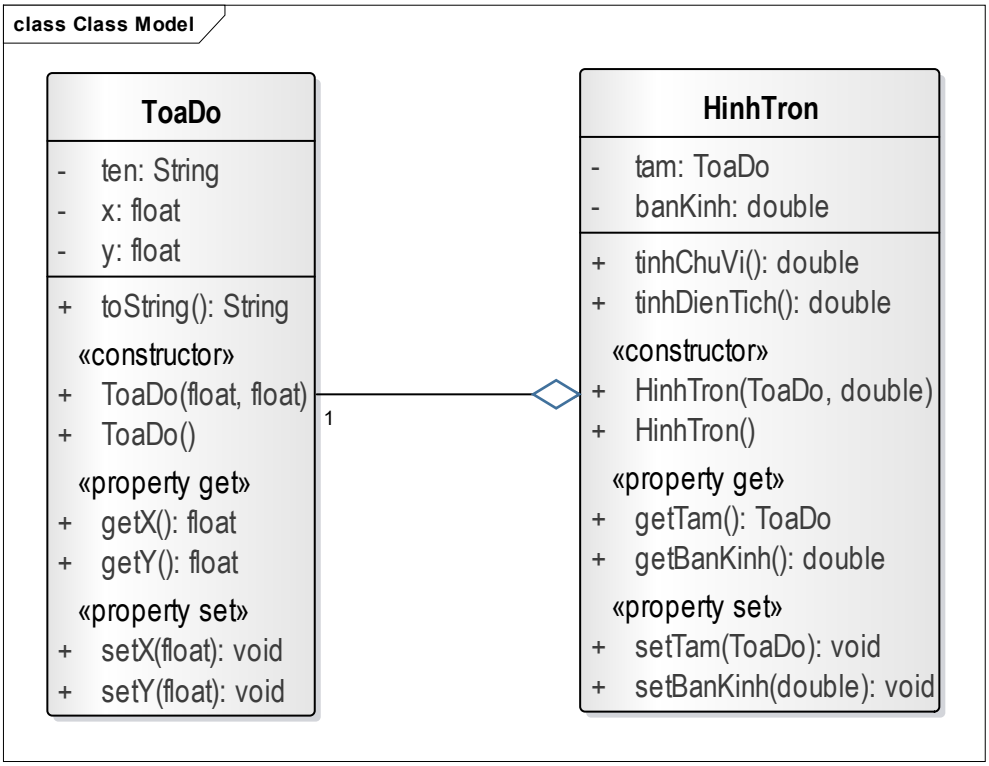
(c) . Test lớp Account như sau :

- Tạo 3 đối tượng acc1, acc2, acc3 với các giá trị name, accountNumber, balance lần lượt như sau: {"Ted Murphy", 72354, 102.56}; {"Jane Smith", 69713, 40.00}; {"Edward Demsey", 93757, 759.32}.
- Gởi thêm số tiền cho acc1 là 25.85, cho acc2 là 500.00.
- Rút khỏi acc2 số tiền là 430.75, mức phí 1.50.
- Tính tiền lãi cho acc3.
- Xuất thông tin của acc1, acc2, acc3 (kiểm tra lại kết quả).

- Chuyển tiền từ acc2 sang cho acc1 số tiền là 100.00.
- Xuất thông tin của acc1, acc2 (kiểm tra lại kết quả).

Bài 7. Cài đặt quan hệ Aggregation

Cài đặt cho mô hình lớp sau:



Yêu cầu kết xuất: Giả sử nhập vào hình tròn có tâm O với 2 tọa độ x,y là 5,5, có bán kính là 10.5 thì xuất ra dòng tương ứng là: "*Hình tròn có tâm O(5,5) với bán kính 10.5 có diện tích và chu vi lần lượt là 346.185 và 65.940.*"

Bài 8. Cài đặt lớp chứa danh sách

Lớp đối tượng CD gồm có các thuộc tính sau:

- Mã CD (là số nguyên >0, mặc định là 999999),
- Tựa CD (chuỗi không được rỗng, tên mặc định là "chưa xác định"),
- Số bài hát (số nguyên >0),
- Giá thành (số thực >0).

Yêu cầu (có vẽ mô hình lớp):

a. Viết lớp CD:

- 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.

- Viết phương thức toString trả về thông tin của một CD.

b. Xây dựng lớp **CDList** để lưu danh sách các CD (dùng mảng):

- Phương thức khởi tạo n phần tử cho lớp **CDList**.
- Phương thức thêm một 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 trả thông tin của toàn bộ CD có trong danh sách.
- Phương thức sắp xếp danh sách giảm dần theo giá thành.
- Phương thức sắp xếp danh sách tăng dần theo tựa CD.

c. Viết lớp cho phần kiểm nghiệm. Có thể dùng menu case thực hiện các chức năng theo yêu cầu.

Bài 9. Cài đặt lớp chứa danh sách

a. Cài đặt lớp **CongNhan**, biết:

- *mHo, mTen, mSoSP* lần lượt là các thuộc tính họ, tên và số sản phẩm (>0) của công nhân.
- Viết các phương thức getter/setter cho các thuộc tính của lớp.
- Viết các phương thức khởi tạo (mặc định và đầy đủ tham số).
- Viết phương thức *tinhLuong()* để tính lương cho công nhân, lương = số sản phẩm * đơn giá, với đơn giá tính theo từng cấp như sau:

Số sản phẩm	Đơn giá
Từ 1 – 199 sản phẩm	0.5
Từ sản phẩm thứ 200 - 399	0.55
Từ sản phẩm thứ 400 - 599	0.6
Từ sản phẩm thứ 600 trở lên	0.65

- Viết phương thức toString trả về thông tin của một công nhân.

b. Cài đặt lớp **DanhSachCongNhan**, có:

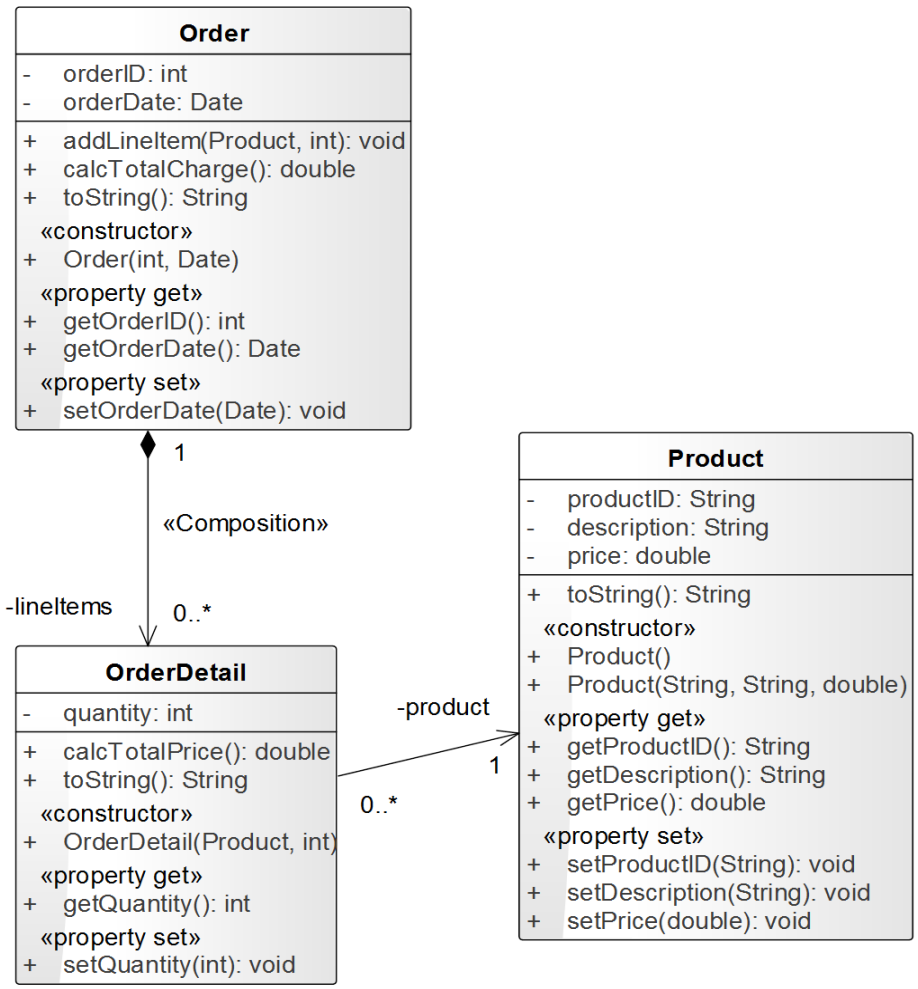
- Phương thức khởi tạo n phần tử cho lớp **DanhSachCongNhan**.
- Phương thức thêm một công nhân vào danh sách, thêm thành công nếu kích thước mảng còn cho phép.
- Phương thức xuất toàn bộ thông tin công nhân viên.
- Phương thức tính số lượng công nhân viên có trong danh sách.
- Phương thức xuất thông tin các công nhân làm trên 200 sản phẩm.
- Phương thức sắp xếp công nhân theo số sản phẩm giảm dần.

c. Viết hàm main sử dụng lớp **DanhSachCongNhan** theo yêu cầu sau:

Cho nhập thông tin một số công nhân vào chương trình, thực thi các yêu cầu trong câu b với các đối tượng này.

Bài 10. Cài đặt quan hệ Composition

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

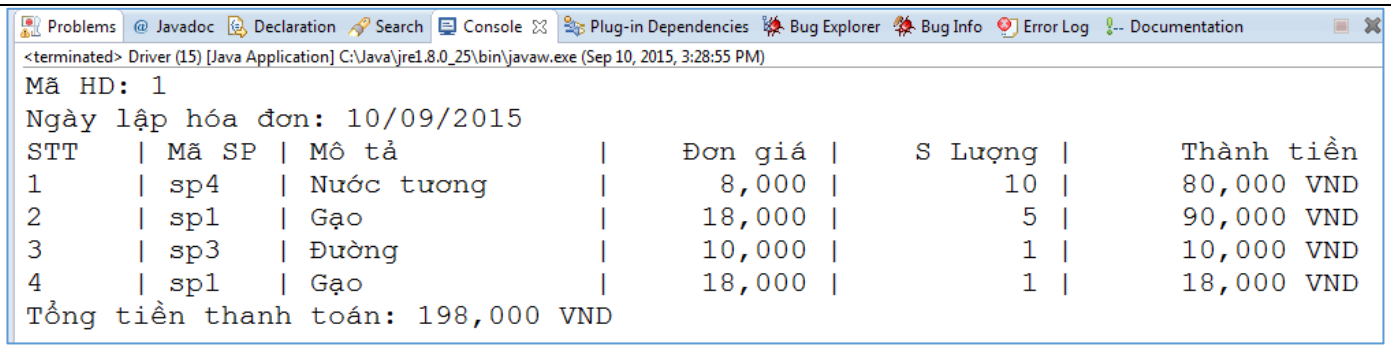


Trong đó:

- Tính thành tiền *(calcTotalPrice())* = quantity * price.
- Tính tổng tiền hóa đơn *(calcTotalCharge())* = $\sum_{count=0}^{n}$ thành tiền.
- Phương thức *addLineItem(Product p, int q): void*, để thêm một sản phẩm p với số lượng q vào hóa đơn.

```
public void addLineItem(Product p, int x) {
    lineItems.add(new OrderDetail(p, x));
}
```

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



Bài 11.

Xây dựng lớp HocVien để quản lý họ tên, năm sinh, điểm 5 môn học của các học viên trong lớp học. Cho biết bao nhiêu người trong lớp được làm luận văn tốt nghiệp, bao nhiêu người thi tốt nghiệp, bao nhiêu người phải thi lại là tên các môn thi lại.

Tiêu chuẩn để xét như sau: *làm luận văn* phải có điểm trung bình lớn hơn 7 trong đó không có môn nào dưới 5; *thi tốt nghiệp* khi điểm trung bình không lớn hơn 7 và điểm các môn không dưới 5; *thi lại* khi có môn dưới 5 điểm.

Bài 12.

Tại một trung tâm đào tạo, người quản lý muốn lưu trữ thông tin học viên của các khóa học, biết một khóa học có thể có tối đa 20 học viên. Thông tin học viên cần lưu trữ là: họ tên, địa chỉ, số điện thoại; thông tin khóa học là: tên khóa học, ngày mở khóa học, thời gian học, danh sách học viên. Ngoài ra người quản lý còn có nhu cầu muốn: tìm kiếm khóa học; biết một học viên đã từng học khóa học nào; các khóa học nào chưa kết thúc; các khóa học nào chưa bắt đầu…

Yêu cầu:

a. Vẽ mô hình lớp và cài đặt các lớp đó để quản lý thông tin học viên và khóa học như mô tả trên.
b. Viết hàm main để thực hiện theo nhu cầu của người quản lý.

Phụ lục

Code bài 2.

```
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, "", 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) {
		setMaSV(maSV);
		setHoTen(hoTen);
		setDiemLT(diemLT);
		setDiemLT(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) {
		if (maSV < 0)
			this.maSV = 0;
		else
			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) {
		if (diemLT >= 0 && diemLT <= 10)
			this.diemLT = diemLT;
		else
			this.diemLT = 0;
	}

	/**
	 * 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) {
		if (diemTH >= 0 && diemTH <= 10)
			this.diemTH = diemTH;
		else
			this.diemTH = 0;
	}

	/**
	 * 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();

                //Gọi 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();
            }
        }
```

Module 3. KẾ THỪA – ĐA HÌNH

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.*

Yêu cầu:

- *Bài làm lưu trong workspace có tên MaSV_HoTen.*
- *Tạo project tên Module03.*
- *Mỗi bài làm trong một package có tên package là bai01, bai02,…*
- *Viết documentation comment cho class.*
- *Bài tập bắt buộc: 1 → 8*

Bài 1.

Công ty du lịch X có quản lý thông tin các chuyến xe. Có 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:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.

b. Trong hàm main, tạo sẵn mỗi loại 2 chuyến xe vào danh sách (không cần cho nhập từ bàn phím). Xuất tổng doanh thu cho tất cả các chuyến xe và tổng doanh thu của từng loại chuyến xe.

Bài 2.

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

- Sách giáo khoa: Mã sách, ngày nhập, 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, đơn giá, số lượng, nhà xuất bản, thuế. Thành tiền = số lượng * đơn giá + thuế.

Thực hiện các yêu cầu sau:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.

b. Trong hàm main, tạo sẵn mỗi loại 3 cuốn sách vào danh sách (không cần cho nhập từ bàn phím), sau đó:

  - 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 K (cho nhập K).

Bài 3.

Viết 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, đơn giá, số lượng, loại vàng. Thành tiền tính như sau:

  Thành tiền = số lượng * đơn giá.

- Giao dịch tiền tệ: Mã giao dịch, ngày giao dịch, đơn giá, số lượng, tỉ giá, loại tiền tệ có 3 loại: tiền Việt Nam, tiền USD, tiền Euro. Thành tiền 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:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.

b. Trong hàm main, tạo sẵn mỗi loại 3 giao dịch vào danh sách (không cần cho nhập từ bàn phím), sau đó:

   - 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 4.

Viết chương trình quản lý danh sách các giao dịch nhà đất. Thông tin bao gồm:

- Giao dịch đất: Mã giao dịch, ngày giao dịch, đơn giá, loại đất (“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, 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:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.

b. Trong hàm main, tạo sẵn mỗi loại 3 giao dịch vào danh sách (không cần cho nhập từ bàn phím), sau đó:

   - 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 5.

Viết 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, 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, 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, quốc tịch, số lượng, đơn giá. Thành tiền được tính = số lượng*đơn giá.

Thực hiện các yêu cầu sau:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế.

b. Trong hàm main, tạo sẵn mỗi loại 3 khách hàng vào danh sách (không cần nhập từ bàn phím), sau đó:
   - 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 2018 (cùa cả 2 loại khách hàng).

Bài 6. *

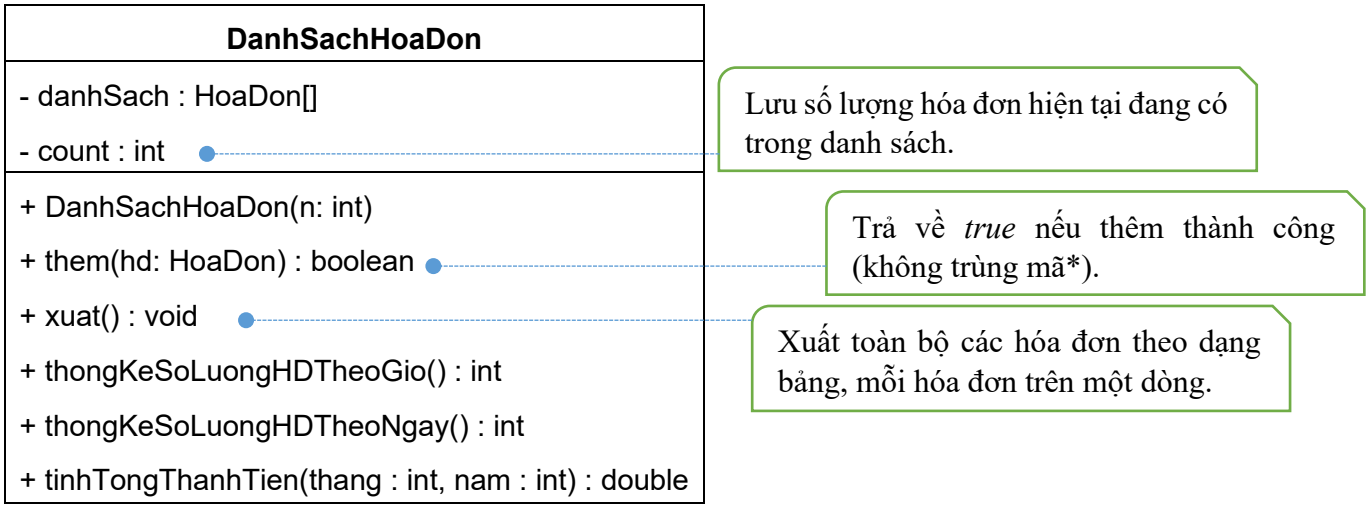
Một khách sạn X cần quản lý các hóa đơn của khách hàng thuê phòng. Hóa đơn có 2 loại: hóa đơn theo giờ, hóa đơn theo ngày. Thông tin chung của chi tiết hóa đơn là: *mã hóa đơn, ngày hóa đơn, tên khách hàng, mã phòng, đơn giá*. Thông tin riêng của từng loại hóa đơn gồm:

- 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 > 30 tiếng thì không dùng loại hóa đơn theo giờ (phát sinh ngoại lệ).
- 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:

a. Vẽ mô hình và cài đặt các lớp với chức năng thừa kế theo mô tả trên.

b. Cài đặt thêm lớp chứa danh sách các hóa đơn theo mô hình sau:



(*) Yêu cầu override phương thức equals của lớp Object để kiểm tra trùng mã.

c. Tạo menu case trong hàm main cho phép thực hiện các chức năng trong câu b.

Bài 7. *

Hàng hóa 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, mặc định là "xxx")*, **số lượng tồn** *(>=0)*, **đơn giá** *(>=0)*.

Hàng thực phẩm thì cần quan tâm đến thông tin: **nhà cung cấp**, **ngày sản xuất** (*phải trước ngày hiện tại, mặc định là ngày hiện tại*) và **ngày hết hạn** *(phải sau ngày sản xuất, mặc định là ngày sản xuất)*.

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 lớp (xây dựng cây thừa kế, các interface nếu có).

b) Thực hiện cài đặt 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*.

c) Hãy viết lớp quản lý **danh sách hàng hóa**. Dùng mảng để lưu trữ danh sách hàng hóa.

- Tạo constructor **khởi tạo** danh sách n phần tử.
- Viết phương thức **thêm** một hàng hóa vào danh sách *(thêm thành công nếu không bị trùng mã hàng)*.
- Viết phương thức **in toàn bộ** danh sách các hàng hóa.
- Viết các phương thức **in từng loại** hàng hóa.
- Viết phương thức **tìm kiếm** hàng hóa khi biết mã hàng (trả về hàng hóa tìm thấy).
- Viết phương thức **sắp xếp** hàng hóa theo tên hàng tăng dần.
- Viết phương thức **sắp xếp** hàng hóa theo số lượng tồn giảm dần.
- Viết phương thức **xuất** các hàng thực phẩm khó bán.

d) Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng trong câu c.

HD:

Dùng Arrays.sort và interface Comparator để sắp xếp.

Bài 8.

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.

Hãy xây dựng các lớp sau:

a. Lớp **Person**: bao gồm các thành phần *họ tên, địa chỉ,* phương thức *toString*.
b. Các lớp **Student, Employee, Customer** (theo mô tả bên dưới) thừa kế lớp **Person**.
   - Lớp ***Student:*** bao gồm các thuộc tính *điểm môn học 1, điểm môn học 2*; các phương thức: *tính điểm trung bình, đánh giá*, *toString* trả về bảng điểm sinh viên (gồm thông tin thuộc tính và điểm trung bình).
   - Lớp ***Employee:*** bao gồm thuộc tính *hệ số lương*; các phương thức: *tính lương, đánh giá*, *toString* trả về bảng lương cho nhân viên (gồm thông tin thuộc tính và tiền lương).
   - Lớp ***Customer:*** bao gồm thuộc tính *tên công ty, trị giá hóa đơn, đánh giá*; phương thức *toString* trả về thông tin hóa đơn cho khách hàng (gồm các thuộc tính của đối tượng).
c. Lớp **Management** lưu thông tin toàn bộ các sinh viên, nhân viên, khách hàng và tổng số người hiện tại có trong danh sách. Ngoài ra còn có các phương thức:
   - Constructor khởi tạo mảng với n phần tử.
   - Thêm một người vào danh sách.
   - Xóa một người khỏi danh sách (nhận thông số là họ tên của người cần xóa).
   - Sắp xếp danh sách theo thứ tự họ tên.
   - Xuất danh sách theo dạng bảng.
d. Viết lớp Test có hàm 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 9.

Tạo lớp trừu tượng **Shape** với 3 phương thức trừu tượng **draw()**, **erase( )** và **move(int** *x***, int** *y***)**. Tạo các lớp con như liệt kê ở bảng dưới đây đồng thời override các phương thức trừu tượng (các phương thức này chỉ in câu thông báo tương ứng 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. Thực thi phương thức này.

Bài 10.

Với một 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 loại xe dựa trên giá trị xe như sau:
  - **Xe đạp:** Không đóng thuế.
  - **Xe máy**: gồm VAT=10% và thuế trước bạ 5%.
  - **Xe ô tô khách**: gồm thuế tiêu thụ đặc biệt 30% (nếu số chỗ >=5), 50% (nếu số chỗ <5); VAT=10% và thuế trước bạ 20%.
  - **Xe ô tô tải:** gồm VAT=10% và thuế trước bạ 2%.

Yêu cầu sinh viên:

a. Thiết kế lược đồ lớp theo mô tả trên.
b. Cài đặt cho lược đồ đã thiết kế.

Bài 11.

Trường đại học X có nhiều cấp đào tạo, thông tin về sinh viên được tổ chức như sau:

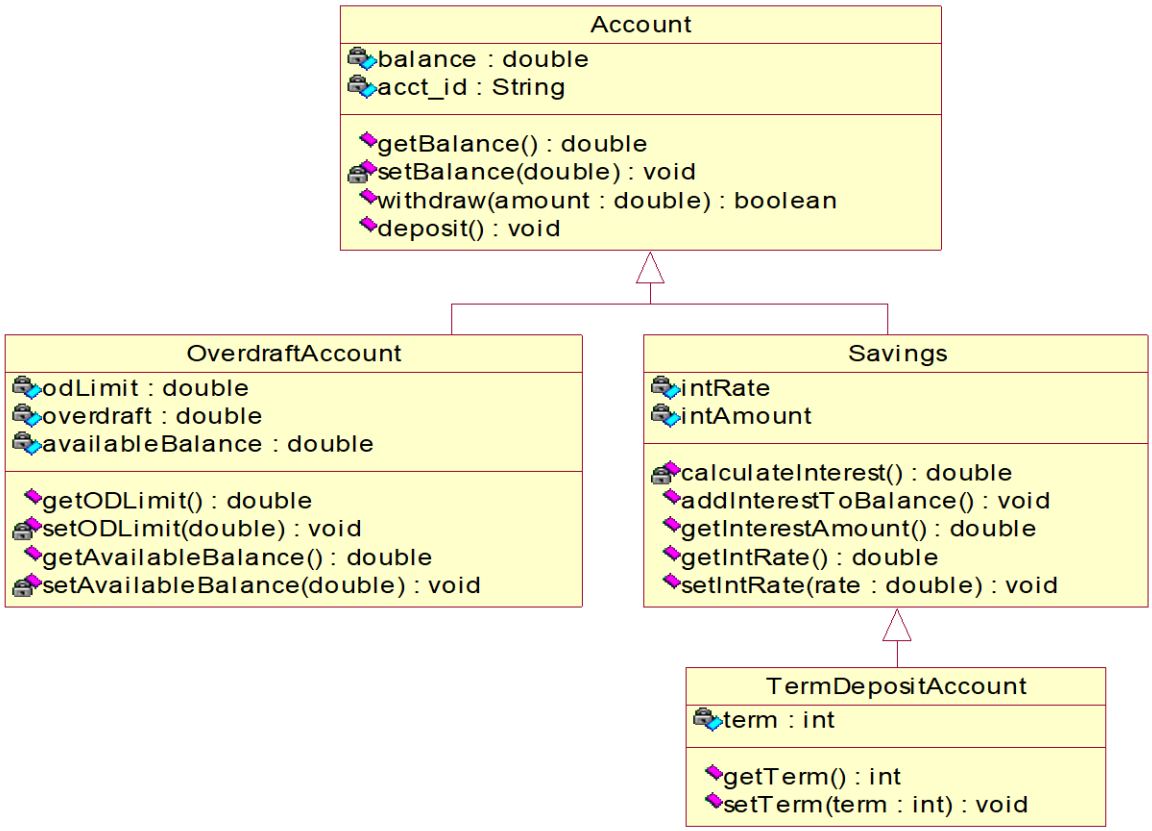
- Sinh viên trung cấp (trung cấp 2 năm)
- Sinh viên cao đẳng
- Sinh viên đại học
- Sinh viên liên kết (Austraulia, US)

Yêu cầu

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 12.

Cho mô hình sau:



a. Sử dụng mô hình lớp ở trên cho cây phân cấp của các đối tượng Account, hãy tạo các phương thức để thi hành phần thiết kế này. Cung cấp các đoạn mã lệnh để thực thi các phương thức.

b. Viết chương trình kiểm tra trong lớp BankingServices, tạo các đối tượng thuộc các lớp cụ thể và kiểm tra các phương thức.

Module 4. TẬP HỢP

Mục tiêu:

- *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*

Yêu cầu:

- *Bài làm lưu trong workspace có tên MaSV_HoTen.*
- *Tạo project tên Module04.*
- *Mỗi bài làm trong một package có tên package là bai01, bai02,…*
- *Bài tập bắt buộc: 1,2,3,4,5,6,7*
- *Viết documentation comment cho class.*

Tóm tắt bài học

Interface	Description	Concrete Classes
`Collection`	A basic interface that defines the normal operations that allow a collection of objects to be maintained or handled as a single unit.	
`Set`	The `Set` interface extends the `Collection` interface to represent its mathematical namesake: a set of **unique** elements.	`HashSet` `LinkedHashSet`
`SortedSet`	The `SortedSet` interface extends the `Set` interface to provide the required functionality for maintaining a set in which the elements are stored in some **sorted** order.	`TreeSet`

`List`	The `List` interface extends the `Collection` interface to maintain a **sequence** of elements that need not be unique.	`ArrayList`
		`Vector`
		`LinkedList`

`Map`	A basic interface that defines operations for maintaining mappings of **keys to values**.	`HashMap`
		`Hashtable`
		`LinkedHashMap`

`SortedMap`	Extends the `Map` interface for maps that maintain their mappings **sorted** in key order.	`TreeMap`

Bài 1.

Thực hiện các yêu cầu sau:

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 tập hợp 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 một lần. Hãy dùng tập hợp 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 tập hợp 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 tập hợp 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 tập hợp viết chương trình quản lý danh sách khách hàng.

Bài 2.

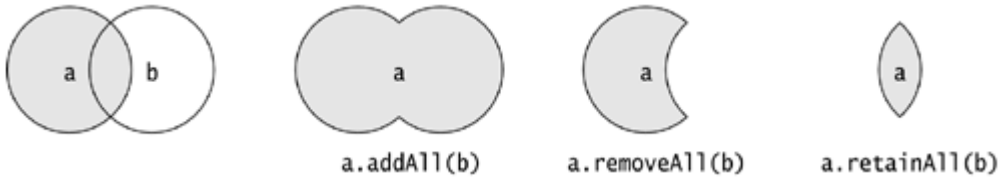
Viết phương thức nhận vào một chuỗi và trả về số ký tự duy nhất của chuỗi. Viết chương trình cho nhập vào nhiều chuỗi và xuất kết quả tương ứng.

Yêu cầu: Các chuỗi nhập vào có thể trùng nhau nên để tiết kiệm thời gian đếm ký tự, người ta muốn lưu lại các kết quả số ký tự của từng chuỗi để khi gặp phải chuỗi giống trước đó, chương trình chỉ truy xuất đến kết quả đã tính được. Lựa chọn sử dụng collection và map thích hợp.

Bài 3.

Viết chương trình tạo hai tập hợp số nguyên (Set). Tính giao, hội, hiệu hai tập trên, xuất kết quả tăng dần.

HD: Dùng TreeSet



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

Bài 4.

Tạo một ArrayList chứa các sinh viên, mỗi sinh viên gồm các thông tin mã, họ tên, năm sinh. Thêm vào danh sách 5 sinh viên tùy ý (không được trùng mã). Viết các phương thức: in danh sách sinh viên dạng bảng; thêm một sinh viên mới; xóa sinh viên khi biết mã; sửa thông tin sinh viên (không sửa mã); tìm kiếm sinh viên theo mã, theo tên, sắp xếp danh sách theo mã tăng dần.

Bài 5.

Tùy chọn sửa lại một trong các bài tập ở Module 3: lựa chọn sử dụng collection thích hợp để lưu trữ (không dùng mảng thông thường).

Bài 6. *

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*. Ngoài ra còn:

- 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* (để rửa dụng cụ thí nghiệm/rửa tay).

Thêm nữa, người quản lý cần phải xem xét phòng học *có đạt chuẩn không*. Phòng học đạt chuẩn nếu: tất cả các phòng đều phải đủ ánh sáng *(trung bình $10m^2$ - 1 bóng đèn)*, và:

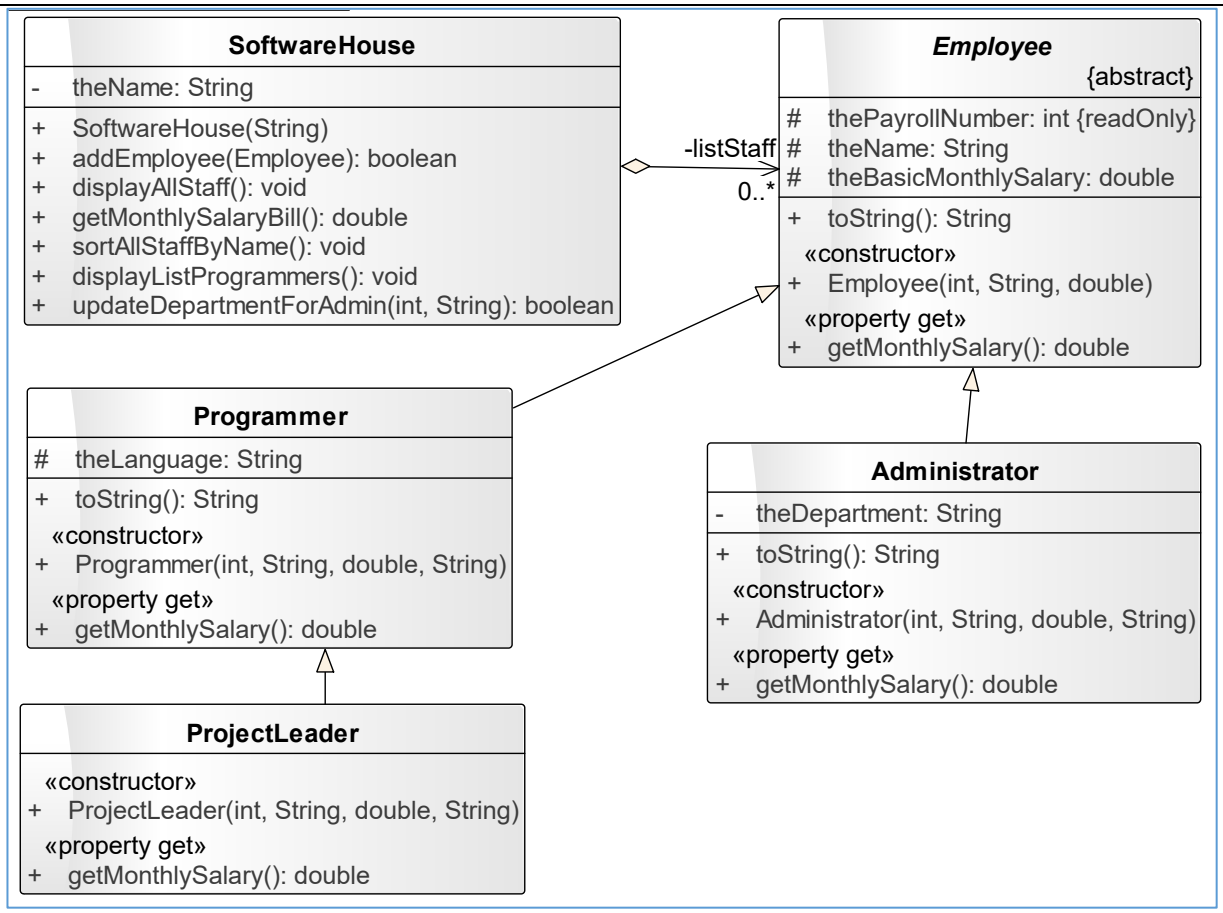
- Phòng lý thuyết: phải có máy chiếu.
- Phòng máy tính: trung bình $1.5m^2$ đặt một máy.
- Phòng thí nghiệm: phải có bồn rửa đi kèm.

Yêu cầu:

a. Thiết kế và thực hiện cài đặt tường minh cho mỗi loại phòng được mô tả trên.
b. Viết lớp quản lý danh sách phòng học. Yêu cầu dùng một List (ArrayList, LinkedList, Vector) để lưu trữ danh sách phòng học. Trong đó:
   - Tạo constructor **khởi tạo** danh sách.
   - Viết phương thức **thêm** một phòng học vào danh sách *(thêm được nếu không 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 dãy nhà**.
   - Viết phương thức để **sắp xếp danh sách giảm dần theo diện tích**.
   - Viết phương thức để **sắp xếp danh sách tăng dần theo 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 **tính tổng số phòng học**.
   - Viết các phương thức để **in danh sách các phòng máy có 60 máy**.
c. Tạo lớp cho phần thử nghiệm, với menu lựa chọn để thực hiện các chức năng theo yêu cầu.

Bài 7. *

Hiện thực theo mô hình lớp bên dưới bằng ngôn ngữ lập trình Java.



Đây là một phần ứng dụng 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 một lần duy nhất và không được phép sửa.

a. 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.

b. Công ty phần mềm *(SoftwareHouse)* có tên gọi *(theName)*, và danh sách các nhân viên. Tạo:
   a. Constructor *SofwareHouse(aName : String)*, tạo một công ty có tên là *aName* và khởi tạo danh sách nhân viên (mỗi phần tử trong mảng là một nhân viên).
   b. Phương thức *addEmployee(emp : Employee): boolean*, dùng để thêm một nhân viên *emp* vào công ty. Thêm thành công nếu không trùng mã số.
   c. 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.
   d. Phương thức *getMonthlySalaryBill(): double*, tính tổng tiền phải trả cho các nhân viên.
   e. Phương thức *sortAllStaffByName(): void*, sắp xếp danh sách nhân viên theo tên.

f. Phương thức *displayListProgrammers(): void*, hiển thị danh sách các lập trình viên.

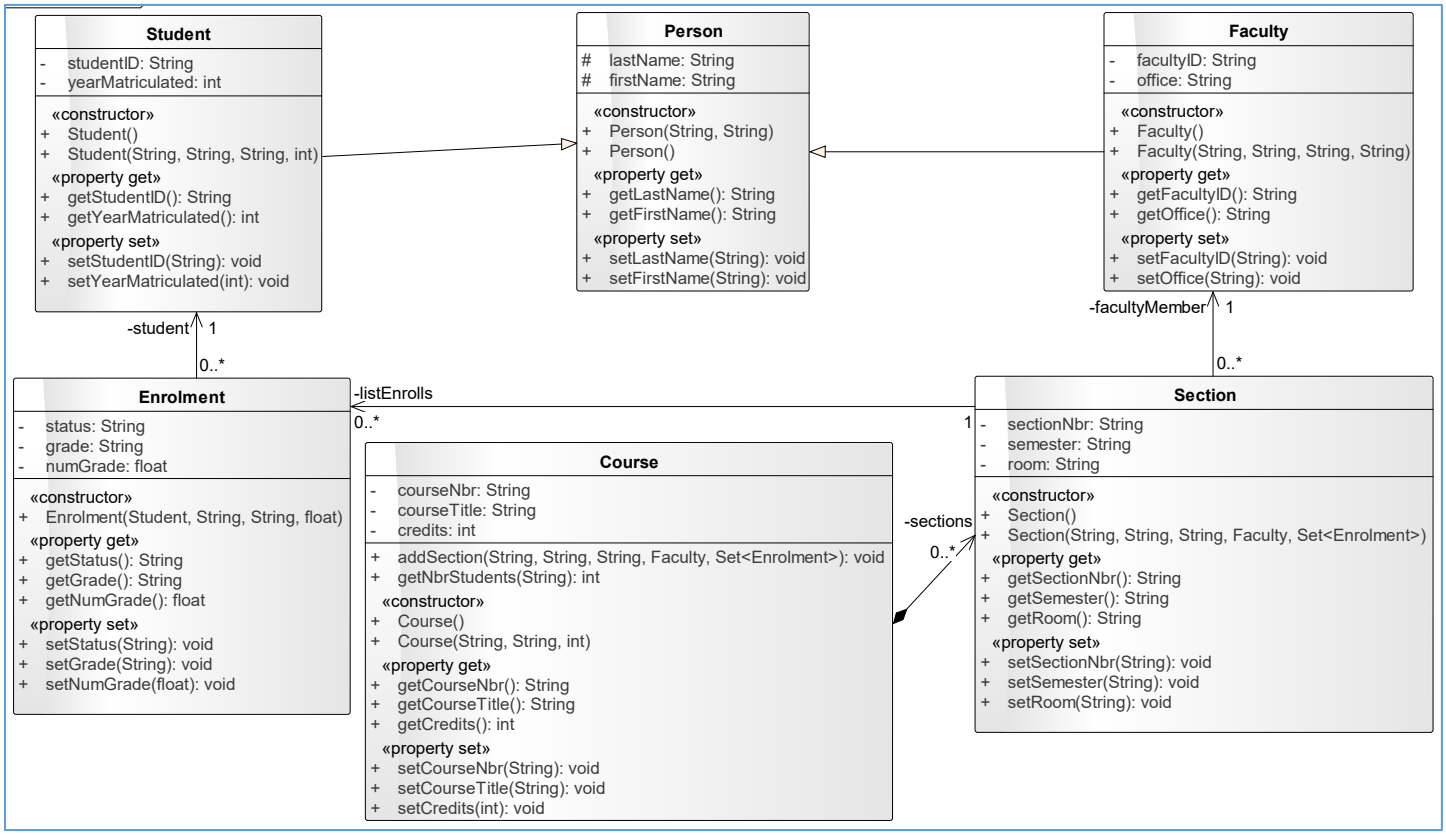
g. Phương thức *updateDepartmentForAdmin(aPayrollNo: int, deptNew: String): boolean*, cập nhật phòng ban là *deptNew* cho người quản lý có mã số là *aPayrollNo*, trả về *true* nếu cập nhật thành công.

h. Phương thức *deleteEmployee(id : int): boolean,* xóa nhân viên theo mã số, trả về *true* nếu xóa được.

c. Viết chương trình chính tạo menu case để thực hiện các chức năng trong câu b.

Bài 8.

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



Lớp **Person** *(người)* là lớp cơ sở. Lớp **Student** *(sinh viên)* và **Faculty** *(giảng viên)* là hai lớp dẫn xuất từ lớp này.

Mỗi học phần nhiều sinh viên đăng ký tham gia *(Enrolment)* học. Sinh viên tham gia học phần sẽ có một trạng thái *(status)*, một điểm chữ *(grade)* và một điểm số *(numGrade)*.

Trong đó, phương thức **addSection** dùng để thêm một học phần cho khóa học:

addSection(String sectionNbr, String semester, String room, Faculty facultyMember, Set<Enrolment> listEnrolls): void

HD: Mapping to java code

```
public class Person {
	private String lastName;
	private String firstName;
}
```

```
public class Student extends Person {
	private int yearMatriculated;
	private String studentID;
}
```

```
public class Faculty extends Person
{
	private String office;
	private String facultyID;
}
```

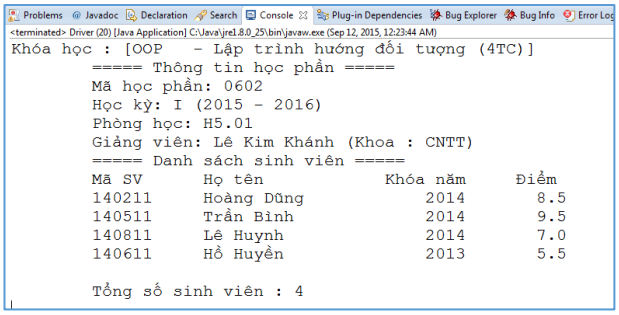
```
public class Section {
	private String semester;
	private String sectionNbr;
	private String room;
	private Faculty facultyMember;
	private Set<Enrolment> listEnrolls ;
}
```

```
public class Enrolment {
	private String status;
	private String grade;
	private float numGrade;
	private Student student;
}
```

```
public class Course {
	private String courseNbr;
	private String courseTitle;
	private int credits;
	private Set<Section> sections;
	public int getNbrStudents(String sectionNbr){}
	public void addSection(String sectionNbr,
String semester, String room, Faculty
facultyMember, Set<Enrolment> listEnrolls){}
}
```

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:



Bài 9.

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

Bài 10.

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

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

Bài 11.

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

Tạo 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.
- Hiển 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.
- Hiển thị danh sách các ga đang chờ mua vé.
- Hiển 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 khác 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 khác 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).

Module 5. LẬP TRÌNH GENERIC

Mục tiêu:

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

Yêu cầu:

- *Bài làm lưu trong workspace có tên MaSV_HoTen.*
- *Tạo project tên Module05.*
- *Mỗi bài làm trong một package có tên package là bai01, bai02,…*
- *Viết documentation comment cho class.*

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 một lớp chung có 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 hai 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); // ket qua "Hello again"
    }
}
```

Bài 4.

Sử dụng interface Generic Comparable. Tạo một lớp Person bao gồm hai 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 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 5.

Thiết kế một lớp hoạt động như một thư viện cho các loại Media sau: sách, video và báo chí. Yêu cầu viết cả hai 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("giong nhau.");
     }
     else
     {
         System.out.println("khac nhau.");
     }

     System.out.print("Trung binh cua iob va fob ");
     if(iob.sameAvg(fob))
```

```
        {
            System.out.println("giong 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 một từ tiếng Anh bao gồm từ, nghĩa, loại từ, và phần ghi chú, override phương thức *toString*, *equals* và phương thức so sánh hai 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).

Viết các lớp sau:

- Lớp mô tả quân bài
- Lớp tạo bộ bài 52 quân bài không trùng nhau với phương thức là xào 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());
 }
 @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);
 }
}
```