

Chương 3: Hướng đối tượng trong Java



ThS. Trần Văn Hữu



Nội dung

- ❖ Các khái niệm cơ bản về lớp, đối tượng.
- ❖ Lớp và đối tượng trong java.
- ❖ Tính đóng gói.
- ❖ Tính kế thừa.
- ❖ Tính đa hình.
- ❖ Interface.



❖ Đối tượng (object): *Trong thế giới thực*, khái niệm đối tượng có thể xem như một thực thể: người, vật, bảng dữ liệu,...

- Cơ sở cho việc cài đặt trên máy tính
- Mỗi đối tượng có định danh, thuộc tính, hành vi

❖ Ví dụ: Đối tượng sinh viên

MSSV: “TH0701001”;

Tên sinh viên: “Nguyễn Văn A”



Các khái niệm cơ bản

❖ **Lớp (class)**: Là khuôn mẫu (**template**) để sinh ra đối tượng.

Ví dụ: lớp các đối tượng **Sinhvien**

- Sinh viên “Nguyễn Văn A”, mã số TH0701001 → 1 đối tượng thuộc lớp **Sinhvien**
- Sinh viên “Nguyễn Văn B”, mã số TH0701002 → là 1 đối tượng thuộc lớp **Sinhvien**

❖ **Đối tượng (object) của lớp**: Một đối tượng cụ thể thuộc 1 lớp, **1 thể hiện** cụ thể của 1 lớp đó.



Khai báo lớp

❖ Khai báo lớp

```
class <ClassName>
```

```
{
```

```
    <danh sách thuộc tính>
```

```
    <các khởi tạo>
```

```
    <danh sách các phương thức>
```

```
}
```



Thành phần dữ liệu

- ❖ Thuộc tính: Các đặc điểm mang giá trị của đối tượng, là vùng dữ liệu được khai báo bên trong lớp

```
class <ClassName>  
{  
    <Tiền tố> <kiểu dữ liệu> <tên thuộc tính>;  
}
```

- Kiểm soát truy cập đối với thuộc tính?



Thành phần xử lý

- ❖ **Phương thức:** Chức năng xử lý, hành vi của các đối tượng.

```
class <ClassName>
```

```
{
```

```
...
```

```
<Tiền tố> <kiểu trả về> <tên phương thức>(<các đối số>)
```

```
{
```

```
...
```

```
}
```

```
}
```

Phạm vi truy xuất

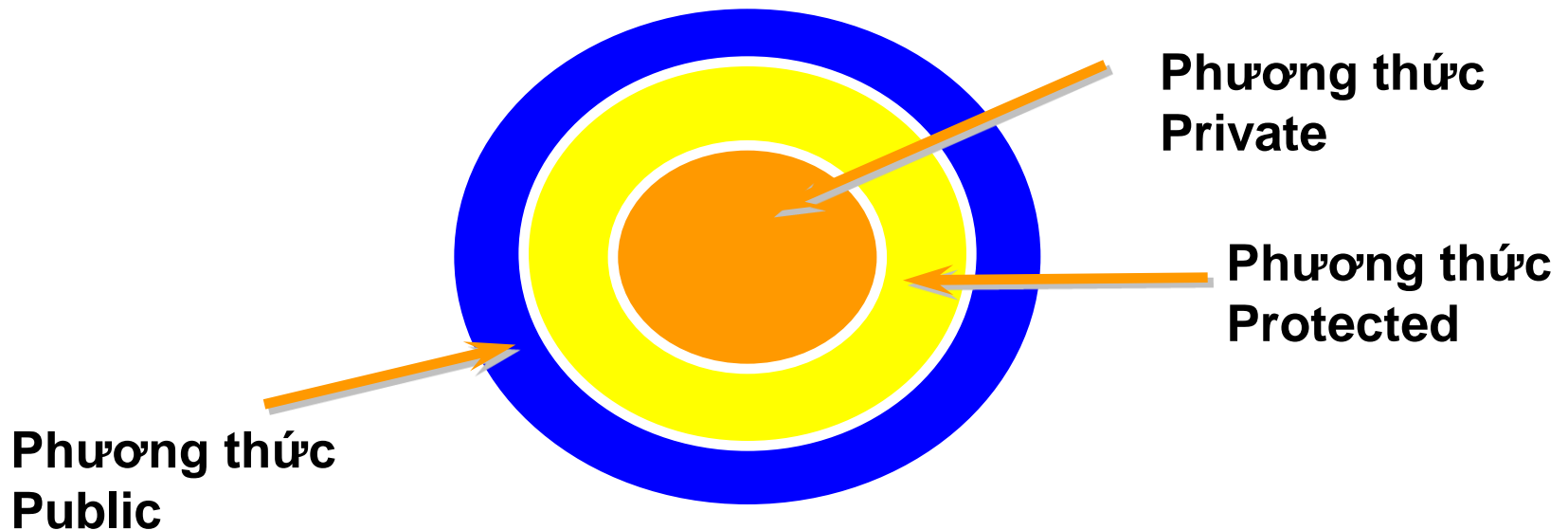
❖ Phạm vi truy xuất của các thành phần:

- Public, protected, *default*, private

+ Thuộc tính/Phương thức **public**

Thuộc tính/Phương thức **protected**

- Thuộc tính/Phương thức **private**



Ví dụ

❖ Ví dụ 1:

```
class Sinhvien {  
    //Danh sách thuộc tính  
    String maSv, tenSv, dcLienlac;  
    int     tuoi;  
    ...  
    //Danh sách các khởi tạo  
    Sinhvien(){ }  
    Sinhvien (...) { ...}  
    ...  
    //Danh sách các phương thức  
    public void capnhatSV (...) { ...}  
    public void xemThongTinSV() { ...}  
    ...  
}
```

Ví dụ

...

//Tạo đối tượng mới thuộc lớp Sinhvien

Sinhvien sv = **new** Sinhvien();

...

//Gán giá trị cho thuộc tính của đối tượng

sv.maSv = "TH0601001" ;

sv.tenSv = "Nguyen Van A";

sv.tuoi = "20";

sv.dcLienlac = "KP6, Linh Trung, Thu Duc";

...

//Gọi thực hiện phương thức

sv.xemThongTinSV();

Ví dụ

❖ Ví dụ 2:

```
class Sinhvien {  
    //Danh sách thuộc tính  
    private String    maSv;  
    private String    tenSv, dcLienlac;  
    private int       tuoi;  
    ...  
}  
...  
Sinhvien sv = new Sinhvien();  
sv.maSv = "TH0601001";  
sv.tenSv = "Nguyen Van A";  
...
```





Phương thức khởi tạo

❖ Khởi tạo (constructor):

- Dùng để khởi tạo giá trị cho các thuộc tính của đối tượng.
- Cùng tên với tên lớp.
- Không có giá trị trả về.
- Có thể có tham số hoặc không.



Phương thức khởi tạo

❖ Ví dụ 1

```
class Sinhvien
{
    ...
    // Không có định nghĩa constructor nào
}
...
// Dùng constructor mặc định
Sinhvien sv = new Sinhvien();
```

Phương thức khởi tạo

❖ Ví dụ 2:

```
class Sinhvien
```

```
{
```

```
...
```

```
//không có constructor mặc định
```

```
Sinhvien(<các đối số>) {...}
```

```
}
```

```
...
```

```
Sinhvien sv = new Sinhvien();
```

Lỗi

```
class Sinhvien
```

```
{
```

```
...
```

```
//khai báo constructor mặc định
```

```
Sinhvien(){ }
```

```
Sinhvien(<các đối số>) {...}
```

```
}
```

```
...
```

```
Sinhvien sv = new Sinhvien();
```



Nạp chồng phương thức

❖ Phương thức khai báo chồng (overloading method)

- Ví dụ:

```
class Sinhvien{  
    ...  
    public void xemThongTinSV() {///...}  
    public void xemThongTinSV(String psMaSv){  
        ///...  
    }  
}
```



Tham chiếu *this*

❖ Tham chiếu *this*: Một biến ẩn tồn tại trong tất cả các lớp, *this* được sử dụng trong khi chạy và **tham chiếu đến bản thân lớp chứa nó**.

❖ Ví dụ:

```
class Sinhvien {  
    String  maSv, tenSv, dcLienlac;  
    int     tuoi;  
    ...  
    public void xemThongTinSV() {  
        System.out.println(this.maSv);  
        System.out.println(this.tenSv);  
        ...  
    }  
}
```




Thành viên tĩnh (static)

- ❖ Đối với class, **static** dùng để khai báo thành viên dữ liệu tĩnh:
 - Thuộc tính khai báo là static
 - Phương thức khai báo là static
 - Có thể được **truy cập thông qua bất kỳ đối tượng nào** của lớp đó, hoặc **thông qua tên lớp**.
- ❖ Cả 2 trường hợp này đều được **cấp phát 1 lần và duy nhất và ra đời trước đối tượng**.
- ❖ => => Các phương thức **static** không thể gọi các thuộc tính và phương thức **không static**.



Tính đóng gói

❖ **Đóng gói:** Nhóm những gì có liên quan với nhau thành một và có thể sử dụng một tên để gọi.

Ví dụ:

- Các phương thức đóng gói các câu lệnh.
- Đối tượng đóng gói dữ liệu và các hành vi/phương thức liên quan.



Tính đóng gói

- ❖ Đóng gói dùng để **che dấu** một phần hoặc tất cả thông tin, chi tiết cài đặt bên trong với bên ngoài.
- ❖ Ví dụ: khai báo các lớp thuộc cùng gói trong java

package <tên gói>;

// khai báo trước khi khai báo lớp

class <tên lớp>

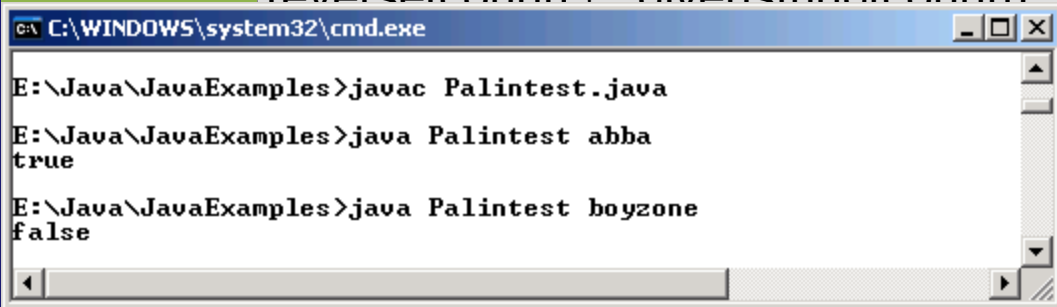
{ ... }

Creating packages in Java

```
package mypackage;
import mypackage.*;
class Palintest
{
    public static void main(String[] args)
    {
        Palindrome objPalindrome = new Palindrome();
        System.out.println(objPalindrome.test(args[0]));
    }
}
```

Output

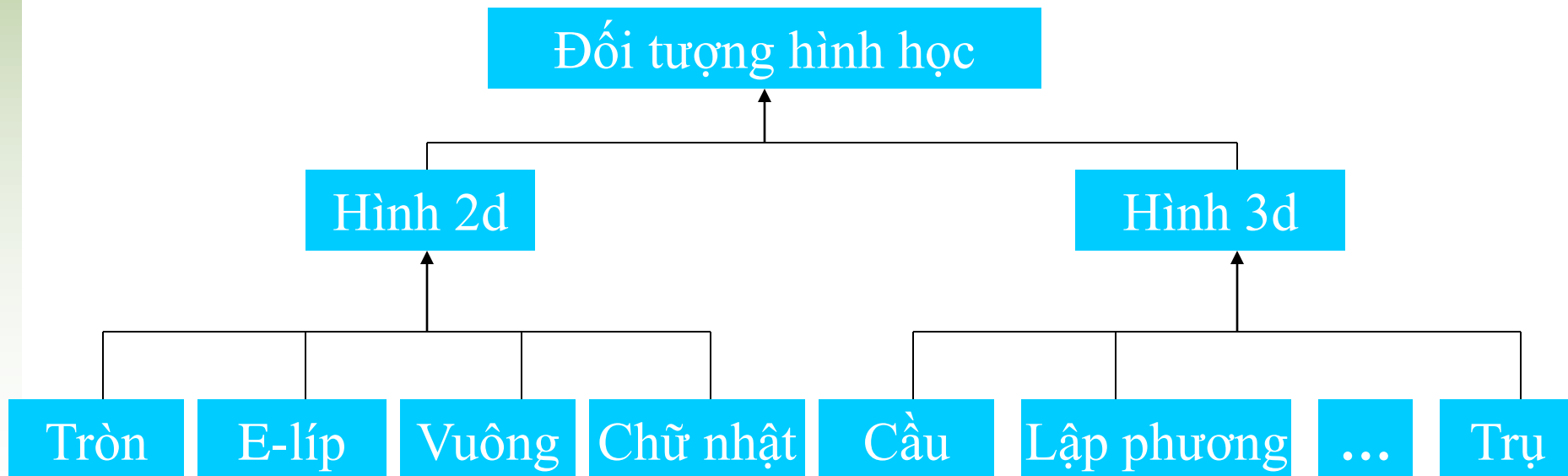
```
{count = 0;count < str.length();count++){
(reverse[count] != givenstring[count])
```



```
C:\WINDOWS\system32\cmd.exe
E:\Java\JavaExamples>javac Palintest.java
E:\Java\JavaExamples>java Palintest abba
true
E:\Java\JavaExamples>java Palintest boyzone
false
```

Tính kế thừa

- ❖ Thừa hưởng các thuộc tính và phương thức đã có.
- ❖ Bổ sung, chi tiết hóa cho phù hợp với mục đích sử dụng mới
 - Thuộc tính: Thêm mới
 - Phương thức: Thêm mới hay hiệu chỉnh



Tính kế thừa

- ❖ Lớp dẫn xuất/lớp con (SubClass)
- ❖ Lớp cơ sở/lớp cha (SuperClass)
- ❖ Lớp con có thể truy xuất các thuộc tính, phương thức của lớp cha (**public, protected, default**)
- ❖ Dùng từ khóa **extends**.

Ví dụ: **class** *nguoi* { ...

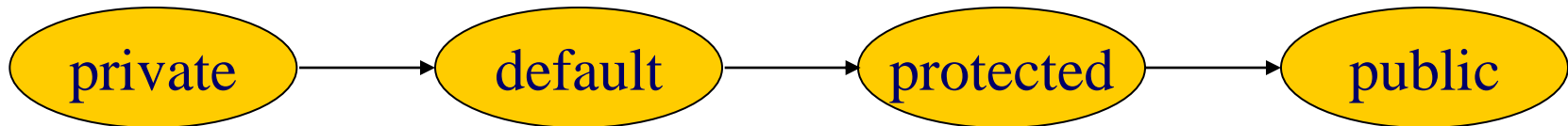
}

class *sinhvien* **extends** *nguoi* { ...

}

❖ Phương thức định nghĩa lại (Overriding Method)

- Được định nghĩa lại trong lớp con
- Có tên, kiểu trả về & các đối số giống với phương thức của lớp cha
- Có kiểu, phạm vi truy cập *“lớn hơn hoặc bằng”* phương thức trong lớp cha



Tính kế thừa

❖ Ví dụ:

```
abstract class Hinhhoc { ...  
    public float tinhdientich() {  
        return 0;  
    }  
    ...  
}  
  
class HinhVuong extends Hinhhoc {  
    private int canh;  
    public float tinhdientich() {  
        return canh*canh;  
    }  
    ...  
}
```

Chỉ có thể **public** do phương thức *tinhdientich()* của lớp cha là **public**

Tính kế thừa

```
class HìnhChuNhat extends HìnhHoc{  
    private int cd;  
    private int cr;  
    public float tinhdientich() {  
        return cd*cr;  
    }  
    ...  
}
```

Chỉ có thể **public** do phương thức *tinhdientich()* của lớp cha là **public**



Tính đa hình

❖ **Tính đa hình:** việc một đối tượng có thể có nhiều “kiểu” hiện hành khác nhau được gọi là tính đa hình.



Tính đa hình

```
class A{
    public void method(){
        System.out.println("method of A"); }
}

class B extends A {
    public void method(){
        System.out.println("method of B"); }
}

class C extends A {
    public void method(){
        System.out.println("method of C"); }
}
```

```
// Câu lệnh trong main
A a = new A();
a.method();
```

```
a = new B();
a.method();
```

```
C c = new C();
a = c;
a.method();
```

```
// Kết quả màn hình
method of A
method of B
method of C
```



Phương thức final, Lớp final

❖ Phương thức final:

- Không được khai báo chồng với các lớp dẫn xuất (không được ghi đè ở lớp con)

❖ Lớp final:

- Lớp không cho phép các lớp khác dẫn xuất từ nó hay lớp final không thể có lớp con.
- Định nghĩa: từ khóa **final**

```
public final class A{
```

```
    ...
```

```
}
```



Phương thức trừu tượng

- ❖ Là phương thức không có phần cài đặt, sẽ được cài đặt cụ thể trong các lớp dẫn xuất.
- ❖ Dùng từ khóa **abstract**.

```
public abstract class Shape
{
    static final double PI = 3.14159;
    public abstract double getArea();
    public abstract double getVolume();
}
```



Lớp trừu tượng

- ❖ Lớp trừu tượng chỉ được dùng làm **lớp cha** cho các lớp khác, nó **không có các thể hiện** (instance).
- ❖ Lớp trừu tượng khai báo các thuộc tính, phương thức chung cho các lớp con của nó.
- ❖ Ví dụ có thể thiết kế lớp Hình tròn, Hình vuông... kế thừa từ **lớp trừu tượng Hình**. Lớp Hình có thuộc tính là tên hình, các phương thức tính diện tích, chu vi...
- ❖ Lớp trừu tượng (abstract) thường **có ít nhất một phương thức trừu tượng**, là phương thức **không có cài đặt**.
 - **public abstract void draw();**



Lớp trừu tượng

- ❖ Khai báo lớp trừu tượng
 - `public abstract class ClassName {...}`
- ❖ Các lớp con của một lớp cha trừu tượng phải cài đặt tất cả các phương thức trừu tượng.
- ❖ Không thể tạo các đối tượng của một lớp trừu tượng nhưng có thể khai báo biến thuộc kiểu lớp trừu tượng để tham chiếu đến các đối tượng thuộc lớp con của nó.



Lớp trừu tượng

```
public abstract class Shape{
    static final double PI = 3.14159;
    public abstract double getArea();
    public abstract double getVolume();
}

class Circle extends Shape{
    double radius;
    public double getArea()    { return PI*radius*radius; }
    public double getVolume() { return 0; }
}

class Cube extends Shape{
    double a, b, c;
    public double getArea()    { return 2*(a*b+b*c+c*a); }
    public double getVolume() { return a*b*c; }
}
```




Tham chiếu super

- ❖ Là một biến **tham chiếu** đến đối tượng **có kiểu là lớp cha** của lớp hiện tại.
- ❖ Tham chiếu **super** được dùng để truy cập đến các thành viên của lớp cha đã bị che bởi lớp con và constructor của lớp cha.



Khối vô danh

- ❖ Trong java ta có thể đặt một khối lệnh không thuộc một phương thức nào.
- ❖ Khi đó khối lệnh này được gọi là khối vô danh, khối vô danh này được java gọi thực thi khi một đối tượng được tạo ra, **các khối vô danh được gọi trước cả hàm tạo.**
- ❖ Khối vô danh phải đặt trong cặp { }

//bắt đầu khối vô danh

{

System.out.println ("khởi tạo đầu thu 1 ");

}//kết thúc khối vô danh



Khởi khởi đầu tĩnh

❖ Khởi khởi đầu tĩnh

- Là một khối lệnh bên ngoài tất cả các phương thức, kể cả hàm tạo
- Trước khối lệnh này ta đặt từ khoá **static**, từ khoá này báo cho java biết đây là khối khởi đầu tĩnh, khối này chỉ được gọi 1 lần khi đối tượng đầu tiên của lớp này được tạo ra
- Khối khởi đầu tĩnh này cũng được java gọi tự động trước bất cứ hàm tạo nào, thông thường ta sử dụng khối khởi đầu tĩnh để khởi đầu các thuộc tính tĩnh.



Giao tiếp – interface

- ❖ Interface được java đưa ra với hai mục đích chính:
 - Để tạo ra lớp cơ sở thuần ảo (không có bất kỳ phương thức nào được cài đặt)
 - Thực hiện hành vi tương tự như kế thừa bội.
- ❖ Để tạo ra một interface, ta dùng từ khoá **interface**
- ❖ Để triển khai một interface dùng từ khóa **implements**.
- ❖ Nếu một lớp triển khai nhiều interface?



Giao tiếp – interface

❖ Một số chú ý:

- Các trường trong interface là **static** và **final**
- Tất cả các thành phần của một giao diện tự động là **public** do vậy ta không cần phải thêm bổ từ này vào.
- Ta không được phép thêm các bổ từ khác như **private**, **protected** trước các khai báo trong interface.
- Một interface có thể **thừa kế một interface** khác.
- Một lớp có thể **cài đặt một hay nhiều interface** nhưng chỉ có thể thừa kế (**extends**) từ một lớp.



Giao tiếp – interface

- ❖ Ví dụ: Định nghĩa một interface Shape trong tập tin shape.java

```
public interface Shape{  
    //Tính diện tích  
    public abstract double area();  
    //Tính thể tích  
    public abstract double volume();  
    //Trả về tên của shape  
    public abstract String getName();  
}
```



Giao tiếp – interface

//Lớp Point cài đặt/hiện thực interface tên shape.

//Định nghĩa lớp Point trong tập tin Point.java

```
public class Point extends Object implements Shape {
```

```
    protected int x, y;    //Tọa độ x, y của 1 điểm
```

```
    //Constructor không tham số.
```

```
    public Point() {  
        setPoint( 0, 0 );
```

```
    }
```

```
    //Constructor có tham số.
```

```
    public Point(int xCoordinate, int yCoordinate) {  
        setPoint( xCoordinate, yCoordinate );
```

```
    }
```



Giao tiếp – interface

//Gán tọa độ x, y cho 1 điểm

```
public void setPoint( int xCoordinate, int yCoordinate ) {  
    x = xCoordinate;  
    y = yCoordinate;  
}
```

//Lấy tọa độ x của 1 điểm

```
public int getX() {  
    return x;  
}
```

//Lấy tọa độ y của 1 điểm

```
public int getY() {  
    return y;  
}
```




Giao tiếp – interface

//Thể hiện tọa độ của 1 điểm dưới dạng chuỗi

```
public String toString() {  
    return "[" + x + ", " + y + "];"  
}
```

//Tính diện tích

```
public double area() {  
    return 0.0;  
}
```

//Tính thể tích

```
public double volume() {  
    return 0.0;  
}
```



Giao tiếp – interface

```
//Trả về tên của đối tượng shape  
public String getName() {  
    return "Point";  
}  
} //End class Point
```



Giao tiếp – interface

❖ Một số chú ý:

- Lớp triển khai interface phải thực thi tất cả các phương thức được khai báo trong interface, nếu như lớp đó không triển khai, hoặc triển khai không hết thì nó phải được khai báo là **abstract**
- Interface cũng là một lớp trừu tượng do vậy ta không thể tạo thể hiện của interface
- Một interface có thể mở rộng một interface khác, bằng hình thức kế thừa



Phạm vi truy cập

Phạm vi	Nội bộ Class	Trong Package	Lớp con	Toàn cục
private	YES			
(default)	YES	YES		
protected	YES	YES	YES	
public	YES	YES	YES	YES



Một số lưu ý

- ❖ Java không có toán tử phạm vi (scope) ::
- ❖ Java không có hủy tử (destructor), nó chỉ có phương thức finalize() được gọi bởi Garbage Collector.
- ❖ Java không có template.
- ❖ Java không có quá tải toán tử (operator overloading).



Bài tập

1. Xây dựng lớp Candidate (Thí sinh) gồm các thuộc tính: mã, tên, ngày tháng năm sinh, điểm thi Toán, Văn, Anh và các phương thức cần thiết.
2. Xây dựng lớp TestCandidate để kiểm tra lớp trên:
 - Nhập vào n thí sinh (n do người dùng nhập)
 - In ra thông tin về các thí sinh có tổng điểm lớn hơn 15

3. Xét phần mềm quản lý nhân sự. Giả sử Công ty có hai loại nhân viên: nhân viên văn phòng và nhân viên sản xuất. Viết chương trình quản lý và tính lương cho từng nhân viên của công ty:

- Mỗi nhân viên cần quản lý các thông tin sau: Họ tên, ngày sinh, lương
- Công ty cần tính lương cho nhân viên như sau:
 - Đối với nhân viên sản xuất:
 - $\text{Lương} = \text{lương căn bản} + \text{số sản phẩm} * 5.000$
 - Đối với nhân viên văn phòng:
 - $\text{lương} = \text{số ngày làm việc} * 100.000$



- ❖ Viết chương trình đọc ghi file danh sách khách hàng gồm các thông tin : Mã khách hàng và tên khách hàng.



Hỏi & đáp

