Trường Cao đẳng Công nghệ Thông tin TP.HCM

Khoa Công nghệ Thông tin – Điện tử

Chương 4:

LỚP VÀ ĐỐI TƯỢNG

Giảng viên: Hà Mỹ Trinh

Email: trinhhm@itc.edu.vn

Nội dung

1. Khái Niệm/ Định Nghĩa Lớp
2. Nạp Chồng Phương Thức
3. Hàm Tạo
4. Từ Khóa This
5. Tính Chất Tĩnh
6. Phương Thức Tĩnh
7. Mảng Đối Tượng
8. Package
9. Đặc Tả Truy Xuất
10. Encapsulation – Non Encapsulation
11. QUY TẮC ĐẶT TÊN Trong Java

1. Khái niệm (đối tượng (object))

Đối tượng (Object)

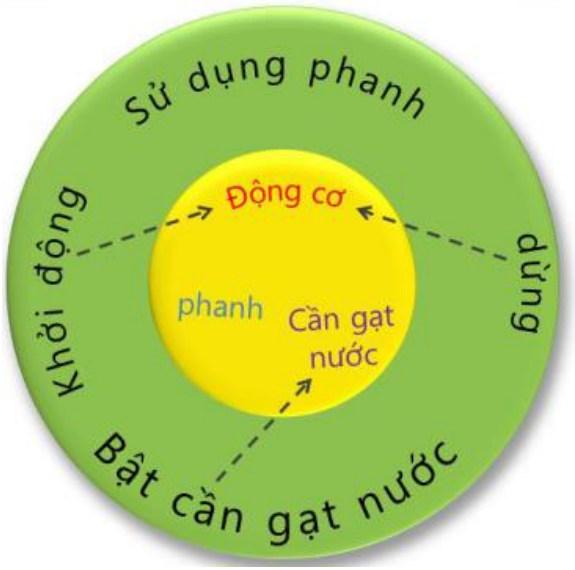
- Biểu diễn đối tượng trong thế giới thực
- Mỗi đối tượng được đặc trưng bởi các thuộc tính và các hành vi riêng của nó
- Thể hiện cụ thể của một lớp đối tượng

1. Khái niệm (đối tượng (object))

- Đối tượng (Object)
- Ví dụ:
  - Trong bài toán quản lý buôn bán xe hơi của một cửa hàng kinh doanh, mỗi chiếc xe đang có mặt trong cửa hàng được coi là một đối tượng. Chẳng hạn, một chiếc xe nhãn hiệu Ford, màu trắng, giá 5000$ là một đối tượng.
  - Trong bài toán quản lý nhân viên của một văn phòng, mỗi nhân viên trong văn phòng được coi là một đối tượng. Chẳng hạn, nhân viên tên là "Vinh", 25 tuổi làm ở phòng hành chính là một đối tượng.
- Mỗi đối tượng có một identity, state, và các behavior (hành vi) duy nhất:
- State = tập các data field (properties)
- Behavior = tập các method
- State xác định đối tượng, behavior xác định đối tượng làm cái gì.

1. Khái niệm (đặc điểm và hành vi)

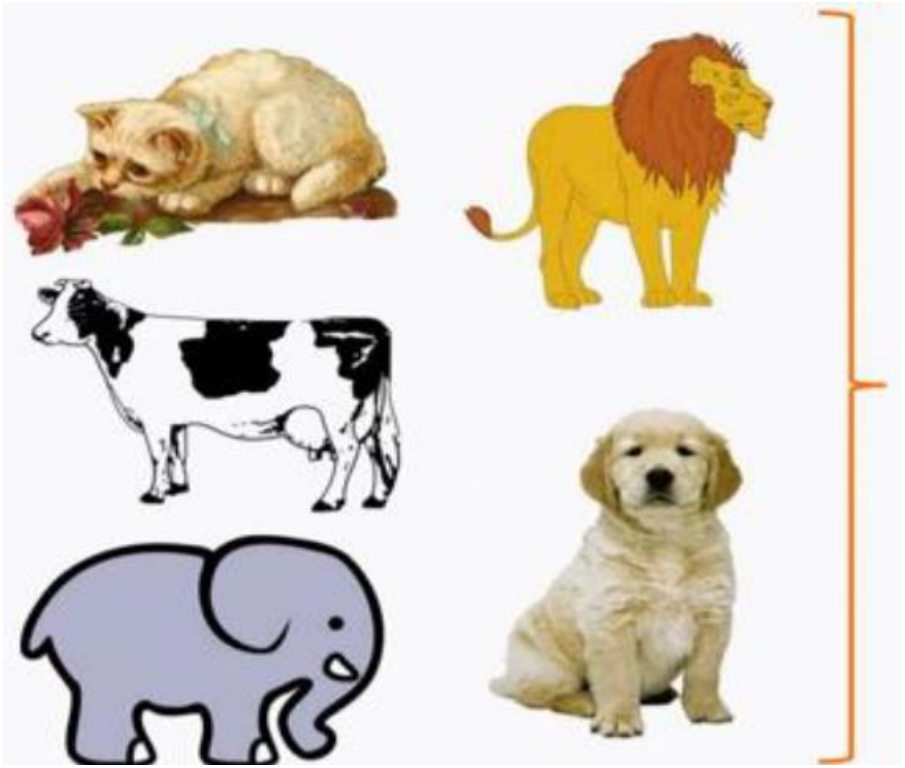
- Đặc điểm và hành vi của đối tượng:
- Đặc điểm
  - Hãng sản xuất
  - Model
  - Năm
  - Màu
- Hành vi (Ô tô có thể làm gì?)
  - Khởi động
  - Dừng
  - Phanh
  - Bật cần gạt nước



1.1 Khái niệm (lớp (class))

- Class là gì?

*Nhóm các **Xe ô-tô***

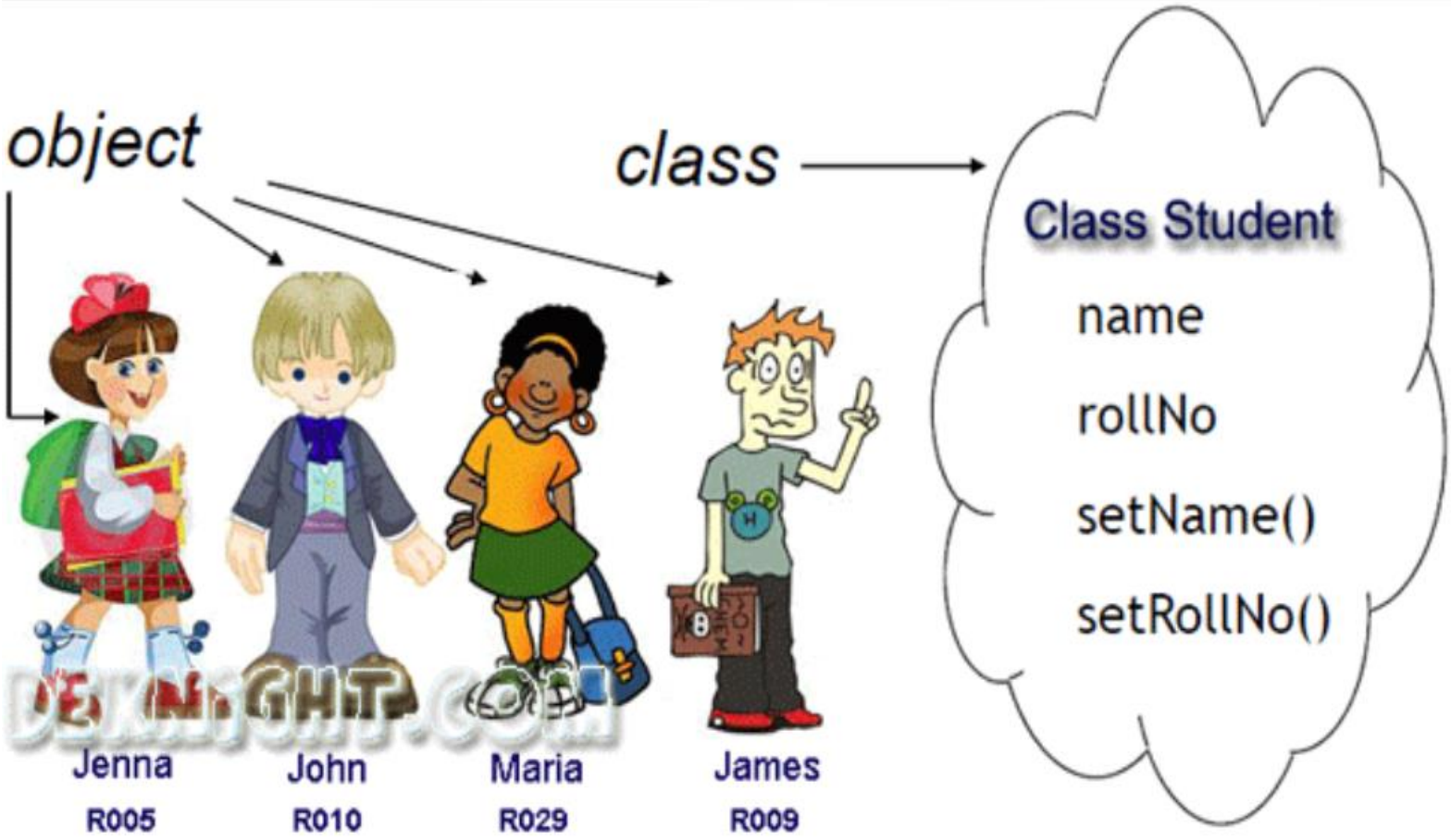


*Nhóm các **Động vật***

1.1 Khái niệm (lớp (class))

- Class là gì?
  - Lớp được xem như một khuôn mẫu (template) được sử dụng để mô tả các đối tượng cùng loại (Object).
  - Trong lớp bao gồm các thuộc tính của đối tượng (properties) và các phương thức (methods) tác động lên các thuộc tính.
  - Đối tượng được xây dựng từ lớp nên được gọi là thể hiện của lớp (class instance).

1.1 Khái niệm (lớp (class))



1.1 Khái niệm (lớp (class))

Lớp (Class)

XE MÁY

Màu sắc
Loại
Model
Thời gian sản xuất
Chi phí sản xuất
Số bánh xe

Tính giá bán()

Đối tượng (instance)

Piaggio

Trắng
Tay ga
Piaggio Fly 150
2012
2.000
2

Tính giá bán()

Lead

Đỏ
Tay ga
Honda Lead 2011
2011
1.700
2

Tính giá bán()

1.1 Khái niệm (lớp (class))

- Thuộc tính và phương thức:
- Thuộc tính (field)
  - Hãng sản xuất
  - Model
  - Năm
  - Màu

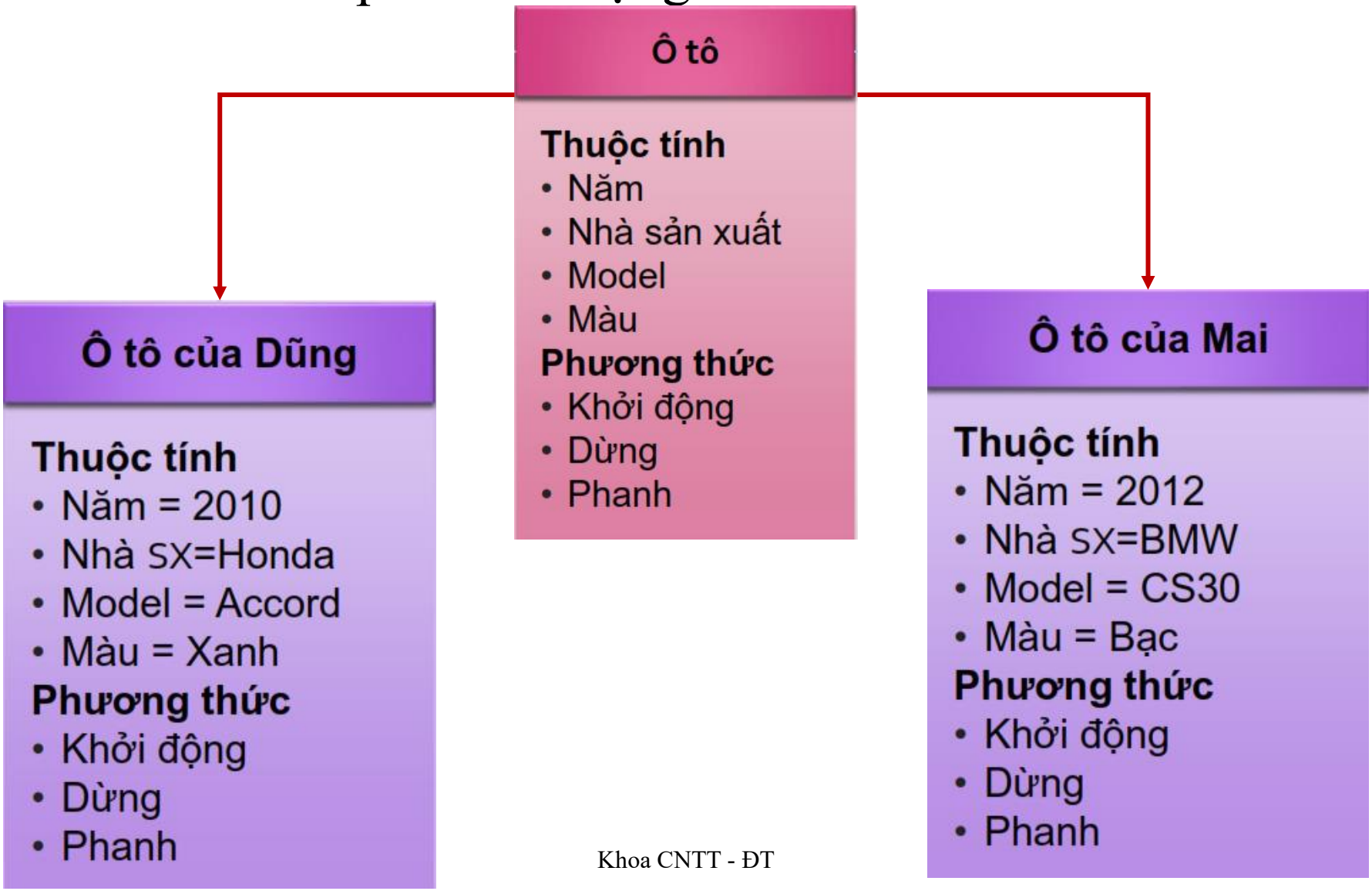
Danh từ

- Phương thức (method)
  - Khởi động()
  - Dừng()
  - Phanh()
  - Bật cần gạt nước()

Động từ

1.1 Khái niệm (lớp (class))

- Mô hình lớp và đối tượng:



1.2 Tính trừu tượng (Abstaction)

- Tính trừu tượng (Abstraction)
  - Abstraction là công việc lựa chọn các thuộc tính và hành vi của thực thể vừa đủ để mô tả thực thể đó trong một bối cảnh cụ thể mà không phải liệt kê tất cả các thuộc tính, hành vi của thực thể có.
  - Ví dụ: Mô tả một sinh viên ngành CNTT có rất nhiều thuộc tính và hành vi. Ở đây chúng ta chỉ sử dụng mã, họ và tên, điểm, ngành mà thôi, không cần thiết phải mô tả cao, nặng, hát, cười, nhảy cò cò…

1.3 Khai báo Lớp

```
class <ClassName>
{
    <kiểu dữ liệu> <field_1>;
    <kiểu dữ liệu> <field_2>;
    constructor
    method_1
    method_2
}
```

- class: là từ khóa của java
- ClassName: là tên lớp
- field_1, field_2: các thuộc tính (các biến, hay các thành phần dữ liệu của lớp)
- constructor: là phương thức xây dựng, khởi tạo đối tượng của lớp.
- method_1, method_2: là các phương thức (có thể gọi là hàm) thể hiện các thao tác xử lý, tác động lên các thuộc tính của lớp.

1.3 Khai báo Lớp

- Ví dụ lớp:

```
public class Employee{
    public String fullname;
    public double salary;

    public void input(){
        Scanner scanner = new Scanner(System.in);
        System.out.print(" >> Full Name: ");
        this.fullname = scanner.nextLine();

        System.out.print(" >> Salary: ");
        this.salary = scanner.nextDouble();
    }

    public void output(){
        System.out.println(this.fullname);
        System.out.println(this.salary);
    }
}
```

Trường

Phương thức

Lớp Employee có 2 thuộc tính là fullname và salary và 2 phương thức là input() và output()

1.4 Tạo đối tượng

- Khai báo
- Khởi tạo dữ liệu
- Cú pháp:
  - <Tên Lớp> tên đối tượng;
  - tên đối tượng = new <Tên Lớp>();
- Ví dụ:
  - BankAccount acc;
  - acc = new BankAccount();

- Có thể kết hợp khai báo và khởi tạo dữ liệu
- Cú pháp:
  - <Tên Lớp> tên đối tượng = new <Tên Lớp>();
- Ví dụ:
  - BankAccount acc = new BankAccount();

1.4 Tạo đối tượng

- Đoạn mã sau sử dụng lớp Employee để tạo một nhân viên sau đó gọi các phương thức của lớp.

```
public static void main(String[] args) {

    Employee emp = new Employee();
    emp.input();
    emp.output();
}
```

- Chú ý:
  - Toán tử new được sử dụng để tạo đối tượng
  - Biến emp chứa tham chiếu tới đối tượng
  - Sử dụng dấu chấm (.) để truy xuất các thành viên của lớp (trường và phương thức).

Truy cập đến thuộc tính và phương thức của lớp

- Sử dụng toán tử •
- Không cần thiết nếu truy cập từ trong cùng một lớp

```
BankAccount account = new BankAccount();
account.setOwner("Smith");
account.credit(1000.0);
System.out.println(account.getBalance());
...
```

BankAccount method

```
public void credit(double amount) {
    setBalance(getBalance() + amount);
}
```

1.5 Thuộc tính của lớp

- Vùng dữ liệu (fields) hay thuộc tính (properties) của lớp được khai báo bên trong lớp như sau:

```
class <ClassName>
{
    // khai báo những thuộc tính của lớp
    <tiền tố> <kiểu dữ liệu> field1;
    // …
}
```

- Để xác định quyền truy xuất của các đối tượng khác đối với vùng dữ liệu của một lớp người ta thường dùng 3 tiền tố sau:
  - public: có thể truy xuất từ tất cả các đối tượng khác
  - private: một lớp không thể truy xuất vùng private của một lớp khác.
  - protected: vùng protected của một lớp chỉ cho phép bản thân lớp đó và những lớp dẫn xuất từ lớp đó truy cập đến.

1.5 Thuộc tính của lớp

- Ví dụ:

```
public class xemay
{
    public String mausac;
    public String loai;
    public String model;
    private float chiphisx;
    protected int thoigiansx;
    public int sobanhxe;
}
```

- Thuộc tính "mausac", "loai", "model", "sobanhxe" có thể được truy cập đến từ tất cả các đối tượng khác.
- Thuộc tính "chiphisx" chỉ có thể truy cập được từ các đối tượng có kiểu "xemay" (trong cùng Class)
- Thuộc tính "thoigiansx" có thể truy cập được từ các đối tượng có kiểu "xemay" và các đối tượng của các lớp con dẫn xuất từ lớp "xemay"

1.5 Thuộc tính của lớp

- Lưu ý:
  - Thông thường để an toàn cho vùng dữ liệu của các đối tượng người ta tránh dùng tiền tố public, mà thường chọn tiền tố private để ngăn cản quyền truy cập đến vùng dữ liệu của một lớp từ các phương thức bên ngoài lớp đó.

1.5 Thuộc tính của lớp

- Ví dụ : Truy cập đến thuộc tính của lớp

```
public class Account {
    String name; //Account name
    long balance; //Balance

    void display(){
        System.out.println(…);
    }
    void deposit (long money){
        balance += money;
    }
}

Account acc1 = new Account();
acc1.name = "Vu T Huong Giang";
acc1.balance = "2000000";
Account acc2 = new Account();
acc2.name = "Nguyen Manh Tuan";
acc2.balance = "1000000";
```

1.6 Phương thức (Method) của lớp

- Phương thức là một mô-đun thực hiện một công việc cụ thể nào đó
  - Trong lớp Employee ở ví dụ lớp có 2 phương thức là input() và output()
- Phương thức có thể có một hoặc nhiều tham số
- Phương thức có thể có kiểu trả về hoặc void (không trả về gì cả)

1.6 Phương thức (Method) của lớp

- Hàm hay phương thức (method) trong Java là khối lệnh thực hiện các chức năng, các hành vi xử lý của lớp lên vùng dữ liệu

Khai báo phương thức:

```
<Tiền tố> <kiểu trả về> <Tên phương thức> (<danh sách đối số>)
{
    <khối lệnh>;
}
```

- Để xác định quyền truy xuất của các đối tượng khác đối với các phương thức của lớp người ta thường dùng các tiền tố sau:

  public, protected, private, static, final, abstract, synchronized

  - <kiểu trả về>: có thể là kiểu void, kiểu cơ sở hay một lớp.
  - <Tên phương thức>: đặt theo qui ước giống tên biến.
  - <danh sách thông số>: có thể rỗng

1.6 Phương thức (Method) của lớp

- **public**: phương thức có thể truy cập được từ bên ngoài lớp khai báo.
- **protected**: có thể truy cập được từ lớp khai báo và những lớp dẫn xuất từ nó.
- **private**: chỉ được truy cập bên trong bản thân lớp khai báo.
- **static**: phương thức lớp dùng chung cho tất cả các thể hiện của lớp, có nghĩa là phương thức đó có thể được thực hiện kể cả khi không có đối tượng của lớp chứa phương thức đó.
- **final**: phương thức có tiền tố này không được khai báo chồng ở các lớp dẫn xuất.
- **abstract**: phương thức không cần cài đặt (không có phần source code), sẽ được hiện thực trong các lớp dẫn xuất từ lớp này.
- **synchoronized**: dùng để ngăn các tác động của các đối tượng khác lên đối tượng đang xét trong khi đang đồng bộ hóa. Dùng trong lập trình miltithreads.

1.6 Phương thức (Method) của lớp

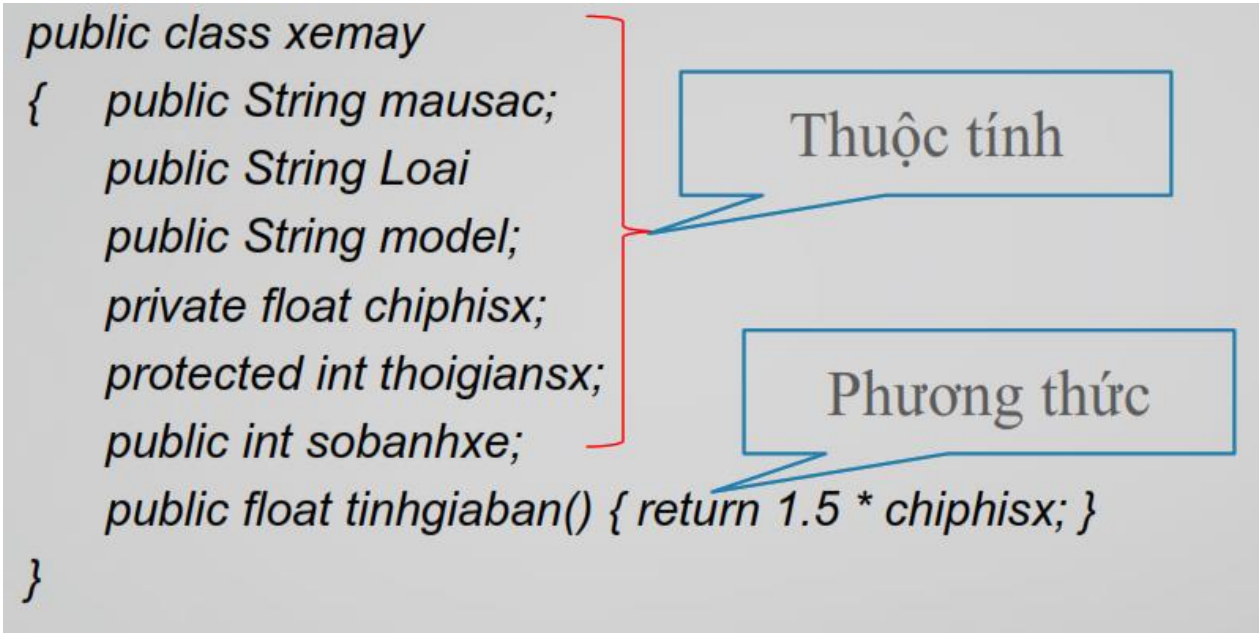
- Xem demo để hiểu rõ
  - **public, protected, private**
  - **final, abstract**

1.6 Phương thức (Method) của lớp

Phạm vi truy cập	public	protected	default	private
Cùng class	✓	✓	✓	✓
Cùng package	✓	✓	✓	✗
Khác package (kế thừa)	✓	✓	✗	✗
Khác package (không kế thừa)	✓	✗	✗	✗

1.6 Phương thức (Method) của lớp

- Ví dụ



- Lưu ý:
- Thông thường trong một lớp các phương thức nên được khai báo dùng từ khóa public, khác với vùng dữ liệu thường là dùng tiền tố private vì mục đích an toàn.
- Những biến nằm trong một phương thức của lớp là các biến cục bộ (local) và nên được khởi tạo sau khi khai báo.

1.6 Phương thức (Method) của lớp

- Ví dụ

```
public class Employee{
    public String fullname;
    public double salary;

    public void input(){…}
    public void output(){…}

    public void setInfo(String fullname, double salary) {
        this.fullname = fullname;
        this.salary = salary;
    }

    public double incomeTax(){
        if(this.salary < 5000000){
            return 0;
        }
        double tax = (this.salary - 5000000) * 10/100;
        return tax;
    }
}
```

Kiểu trả về là **void** nên thân phương thức **không chứa lệnh return giá trị**

Kiểu trả về là **double** nên thân phương thức phải chứa lệnh **return số thực**

1.6 Phương thức (Method) của lớp

- Ví dụ. Truy cập đến phương thức của lớp

```
public class Account {
    String name; //Account name
    long balance; //Balance

    void display(){
        System.out.println(…);
    }
    void deposit (long money){
        balance += money;
    }
}
```

```
// Class that uses
// methods of Account object
Account obj = new Account()
obj.display();
obj.deposit(1000000);
```

2. Nạp chồng phương thức (Overloading)

- Trong một lớp có thể có nhiều phương thức cùng tên nhưng khác nhau về tham số (kiểu, số lượng và thứ tự)

```
public class MyClass{
    void method(){...}
    void method(int x){...}
    void method(float x){...}
    void method(int x, double y){...}
}
```

- Trong lớp MyClass có 4 phương thức cùng tên là method nhưng khác nhau về tham số

2. Nạp chồng phương thức (Overloading)

- Ví dụ nạp chồng phương thức
- Xét trường hợp overload sau

```
class MayTinh{
    int tong(int a, int b){return a + b;}
    int tong(int a, int b, int c){return a + b + c;}
}
```

- Với lớp trên, bạn có thể sử dụng để tính tổng 2 hoặc 3 số nguyên

```
MayTinh mt = new MayTinh();
int t1 = mt.tong(5, 7);
int t2 = mt.tong(5, 7, 9);
```

3. Hàm tạo (Constructor)

- Constructor thật ra là một loại phương thức đặc biệt của lớp.
- Constructor dùng gọi tự động khi khởi tạo một thể hiện của lớp, có thể dung để khởi gán những giá trị mặc định.
- Các constructor **không có giá trị trả về**, và có thể có tham số hoặc không có tham số

```
public class ChuNhat {
    double dai, rong;
    public ChuNhat(){

    }
    public ChuNhat(double dai, double rong) {
        this.dai = dai;
        this.rong = rong;
    }
}
```

3. Hàm tạo (Constructor)

- Constructor phải có cùng tên với lớp và được gọi đến khi dùng từ khóa new.

```
public class test {
    Run | Debug
    public static void main(String[] args) {
        ChuNhat cn1 = new ChuNhat();
        ChuNhat cn2 = new ChuNhat(dai: 20, rong: 15);
        ChuNhat cn3 = new ChuNhat(dai: 50, rong: 25);
    }
}
```

3. Hàm tạo (Constructor)

- Nếu một lớp không có constructor thì Java sẽ cung cấp cho lớp một constructor mặc định (default constructor). Những thuộc tính, biến của lớp sẽ được khởi tạo bởi các giá trị mặc định (số: thường là giá trị 0, kiểu luận lý là giá trị false, kiểu đối tượng giá trị null, …)
- Lưu ý: thông thường để an toàn, dễ kiểm soát và làm chủ mã nguồn chương trình chúng ta nên khai báo một constructor cho lớp.

3. Hàm tạo (Constructor)

```
public class ChuNhat {
    double dai, rong;

}

public class test {
    Run | Debug
    public static void main(String[] args) {
        ChuNhat cn1 = new ChuNhat();
        ChuNhat cn2 = new ChuNhat(20, 15); //Lỗi
        ChuNhat cn3 = new ChuNhat(50, 25); //Lỗi
    }
}
```

3. Hàm tạo (Constructor)

- **Ví dụ**

Lớp

```
public class ChuNhat{
    double dai, rong;
    public ChuNhat(double dai, double rong){
        this.dai = dai;
        this.rong = rong;
    }
```

Đối tượng

```
ChuNhat cn1 = new ChuNhat(20, 15);
ChuNhat cn2 = new ChuNhat(50, 25);
```

```
public class ChuNhat{
    double dai, rong;
    public ChuNhat(double dai, double rong){
        this.dai = dai;
        this.rong = rong;
    }
    public ChuNhat(double canh){
        this.dai = canh;
        this.rong = canh;
    }
}
```

```
ChuNhat cn = new ChuNhat(20, 15);
ChuNhat vu= new ChuNhat(30);
```

4. Từ khóa this

- **this** được sử dụng để đại diện cho đối tượng hiện tại.
- **this** được sử dụng trong lớp để tham chiếu tới các thành viên của lớp (field và method)
- Sử dụng this.field để phân biệt field với các biến cục bộ hoặc tham số của phương thức

```
public class MyClass{
    int field;
    void method(int field){
      this.field = field;
    }
}
```

Trường

Tham số

```
class Foo {
  int i = 5;

  void setI(int
     this.i = i
  }
}
```

```
public class Circle{
  private double rd;

  public Circle(double r){
       this.rd = r;
    }
}
```

4. Từ khóa this

- Quan trọng khi hàm/phương thức thành phần thao tác trên hai hay nhiều đối tượng.
- Xóa đi sự nhập nhằng giữa một biến cục bộ, tham số với thành phần dữ liệu của lớp
- Không dùng bên trong các khối lệnh static

5. Tính chất tĩnh

- Có những trường hợp mà ta muốn tất cả các đối tượng thuộc một lớp chia sẻ một giá trị của một biến cụ thể thay vì phải từng đối tượng duy trì bản sao của chính mình của biến đó như là một thuộc tính. Ngôn ngữ Java đáp ứng nhu cầu này thông qua các thuộc tính tĩnh (static) có liên quan đến tất cả đối tượng của lớp đó hơn là với các cá nhân đối tượng riêng lẽ.
- Khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.
- Biến static có thể được sử dụng để tham chiếu thuộc tính chung của tất cả đối tượng (mà không là duy nhất cho mỗi đối tượng), ví dụ như tên công ty của nhân viên, tên trường học của các sinh viên, …

5. Tính chất tĩnh

- Ví dụ: Biến static trong java

```
//Chuong trinh vi du ve bien static trong Java

class Student8{
   int rollno;
   String name;
   static String college ="BachKhoa";

   Student8(int r,String n){
   rollno = r;
   name = n;
   }
 void display (){System.out.println(rollno+" "+name+" "+college);}

 public static void main(String args[]){
 Student8 s1 = new Student8(111,"Hoang");
 Student8 s2 = new Student8(222,"Thanh");

 s1.display();
 s2.display();
 }
}
```

5. Tính chất tĩnh

- Ví dụ: CT không sử dụng biến static

```
class Counter{
int count=0; //se lay bo nho (memory) khi bien instance duoc tao
 //Ket qua thuc hien chuong trinh hien ra 3 so 1 o 3 dong
Counter(){
count++;
System.out.println(count);
}

public static void main(String args[]){

Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();

 }
}
```

- Ví dụ: CT sử dụng biến static

```
class Counter2{
static int count=0; //se lay bo nho chi mot lan và giu lai gia tri cua no
 //ket qua thuc hien in ra 3 dong cac gia tri : 1,2,3
Counter2(){
count++;
System.out.println(count);
}

public static void main(String args[]){

Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();

 }
}
```

6. Phương thức tĩnh

- Nếu bạn áp dụng từ khóa static với bất cứ phương thức nào, thì phương thức đó được gọi là phương thức static.
- Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.
- Một phương thức static có thể được triệu hồi mà không cần tạo một instance của một lớp.
- Phương thức static có thể truy cập thành viên dữ liệu static và có thể thay đổi giá trị của nó.

6. Phương thức tĩnh

- Ví dụ:

```
class Student9{
      int rollno;
      String name;
      static String college = "BachKhoa";

      static void change(){
      college = "QuocGia";
      }

      Student9(int r, String n){
      rollno = r;
      name = n;
      }

      void display (){System.out.println(rollno+" "+name+" "+college);}

     public static void main(String args[]){
     Student9.change();

     Student9 s1 = new Student9 (111,"Hoang");
     Student9 s2 = new Student9 (222,"Thanh");
     Student9 s3 = new Student9 (333,"Nam");

     s1.display();
     s2.display();
     s3.display();
     }
}
```

6. Phương thức tĩnh

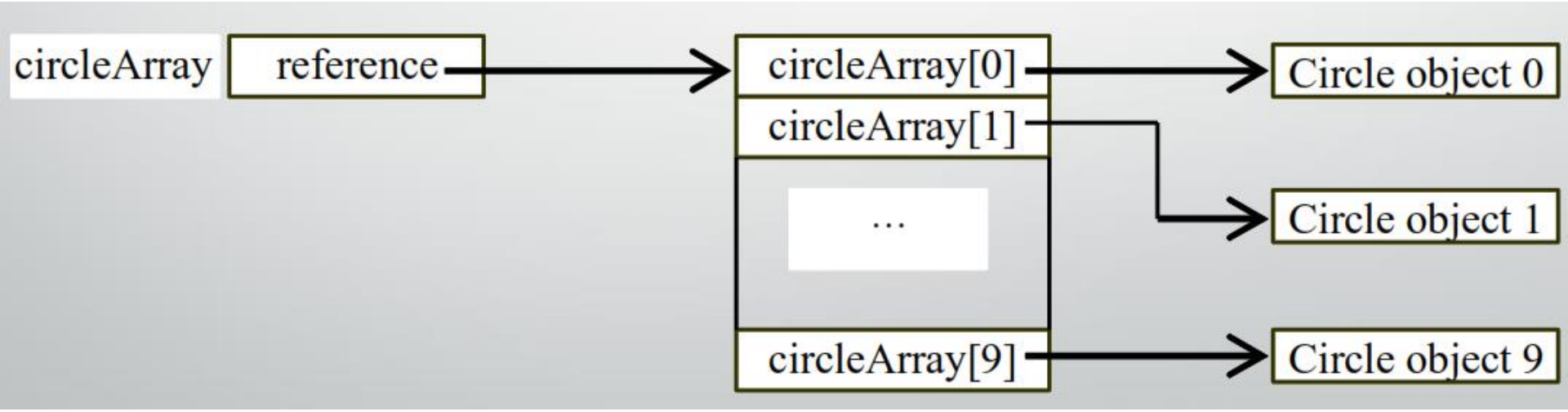
- Nếu bạn áp dụng từ khóa static với bất cứ phương thức nào, thì phương thức đó được gọi là phương thức static.
- Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.
- Một phương thức static có thể được triệu hồi mà không cần tạo một instance của một lớp.
- Phương thức static có thể truy cập thành viên dữ liệu static và có thể thay đổi giá trị của nó.

7. Mảng đối tượng

- Khai báo và tạo mảng các đối tượng:
  - Circle[] circleArray = new Circle[10];
- Khởi tạo:
  - for(int i=0; i<circleArray.length; i++){
    - circleArray[i] = new Circle();

7. Mảng đối tượng

- Circle[] circleArray = new Circle[10];
- Một mảng các đối tượng thực chất là mảng các biến tham chiếu. Do đó gọi circleArray[1].findArea() sẽ gọi 2 mức tham chiếu: circleArray tham chiếu toàn bộ mảng, circleArray[1] tham chiếu tới một đối tượng Circle.



8.Package

- Package được sử dụng để chia các class và interface thành từng gói khác nhau.
  - Việc làm này tương tự quản lý file trên ổ đĩa trong đó class (file) và package (folder)
- Ví dụ sau tạo lớp MyClass thuộc gói com.poly
  - package com.poly;
  - public class MyClass{…}
- Trong Java có rất nhiều gói được phân theo chức năng
  - java.util: chứa các lớp tiện ích
  - java.io: chứa các lớp vào/ra dữ liệu
  - java.lang: chứa các lớp thường dùng…

8. Import Package

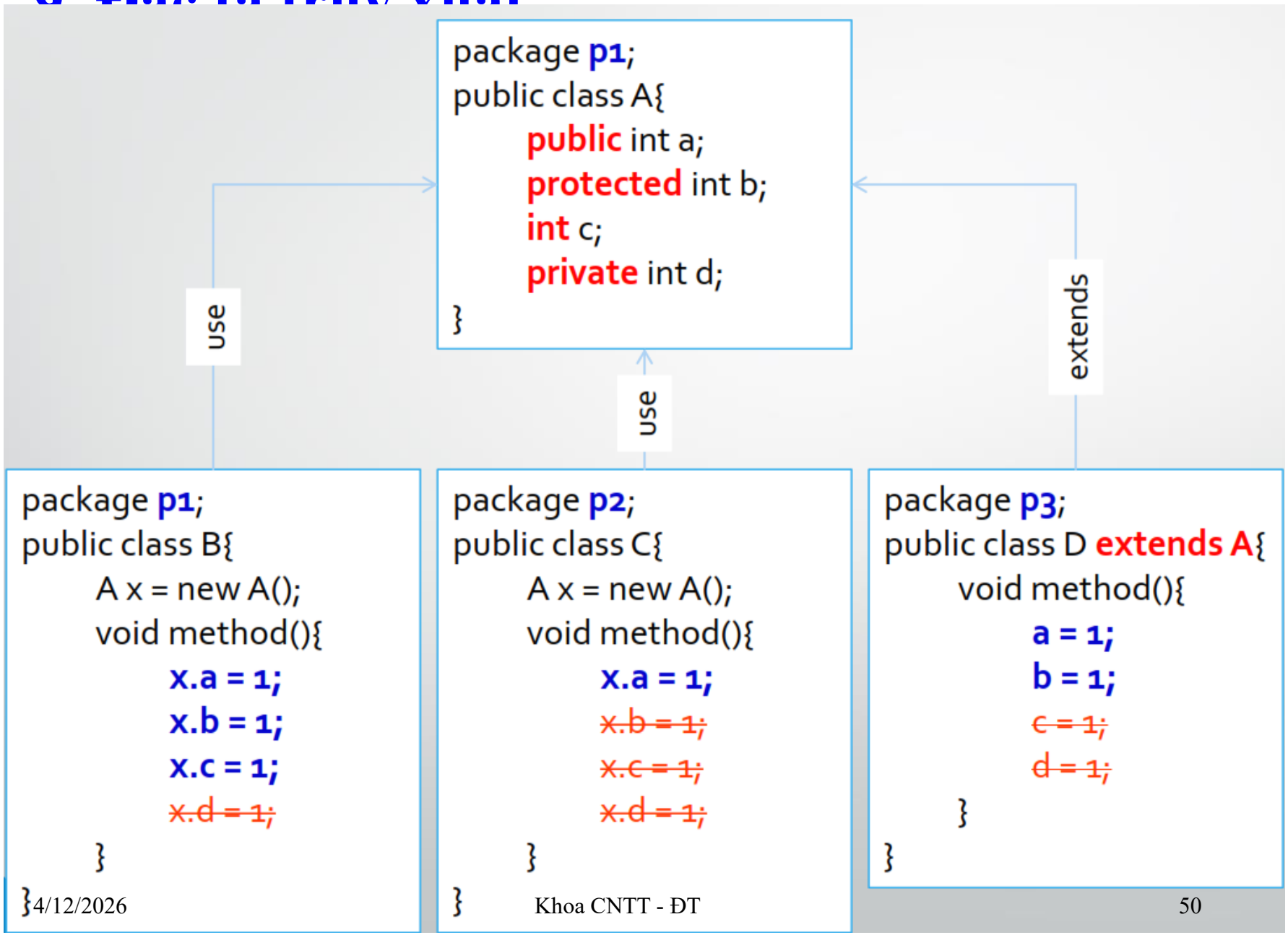
- Lệnh import được sử dụng để chỉ ra lớp đã được định nghĩa trong một package
- Các lớp trong gói java.lang và các lớp cùng định nghĩa trong cùng một gói với lớp sử dụng sẽ được import ngầm định

```
package com.polyhcm;
import com.poly.MyClass;
import java.util.Scanner;
public class HelloWorld{
    public static void main(String[] args){
        MyClass obj = new MyClass();
        Scanner scanner = new Scanner(System.in);
    }
}
```

9. Đặc tả truy xuất

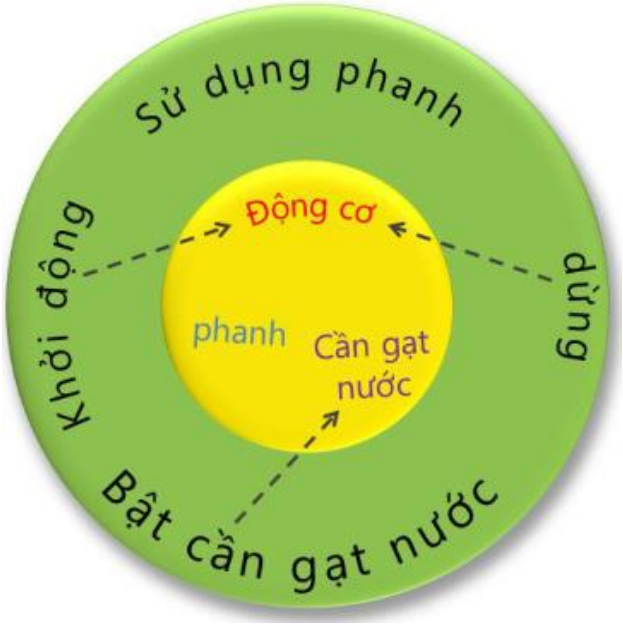
- Đặc tả truy xuất được sử dụng để định nghĩa khả năng cho phép truy xuất đến các thành viên của lớp. Trong java có 4 đặc tả khác nhau:
  - private: chỉ được phép sử dụng nội bộ trong class
  - public: công khai hoàn toàn
  - {default}:
    - Là public đối với các lớp truy xuất cùng gói
    - Là private với các lớp truy xuất khác gói.
  - protected: tương tự {default} nhưng cho phép kế thừa dù lớp con và cha khác gói.
- Mức độ che dấu tăng dần theo chiều mũi tên
  - public → protected → {default} → private

9. Đặc tả truy xuất



10.Encapsulation

- Encapsulation là tính che dấu trong hướng đối tượng.
  - Nên che dấu các trường dữ liệu
  - Sử dụng phương thức để truy xuất các trường dữ liệu
- Mục đích của che dấu
  - Bảo vệ dữ liệu
  - Tăng cường khả năng mở rộng



10. Encapsulation

- Để che dấu thông tin, sử dụng private cho các trường dữ liệu.
  - private double diem;
- Bổ sung các phương thức getter và setter để đọc ghi các trường đã che dấu

```
public void setDiem(double diem){
        this.diem = diem;
}
public String getDiem(){
        return this.diem;
}
```

10.Encapsulation

```
public class SinhVien{
    private String hoTen;
    private double diem;
    public void setHoTen(String hoTen){
        this.hoTen = hoTen;
    }
    public String getHoTen(){
        return this.hoTen;
    }
    public void setDiem(double diem){
        if(diem < 0 || > 10){
            System.out.println("Điểm không hợp lệ");
        }
        else{
            this.diem = diem;
        }
    }
    public String getDiem(){
        return this.diem;
    }
}
```

Chỉ cần thêm mã vào phương thức setDiem() để có những xử lý khi dữ liệu không hợp lệ

```
public class MyClass{
    public static void main(String[] args){
        SinhVien sv = new SinhVien();
        sv.setHoTen("Nguyễn Văn Tèo");
        sv.setDiem(20);
    }
}
```

10. Non - Encapsulation

- Giả sử định nghĩa lớp SinhVien và công khai hoTen và điểm như sau

```
public class SinhVien{
    public String hoTen;
    public double diem;
}
```

```
public class MyClass{
    public static void main(String[] args){
        SinhVien sv = new SinhVien();
        sv.hoTen = "Nguyễn Văn Tèo";
        sv.diem = 20.5;
    }
}
```

- Khi sử dụng người dùng có thể gán dữ liệu cho các trường một cách tùy tiện
- Điều gì sẽ xảy ra nếu điểm hợp lệ chỉ từ 0 đến 10

11. Quy tắc đặt tên trong java

- Tên (class, field, method, package, interface, variable) được đặt theo qui ước (mềm) như sau:
  - Tên package: toàn bộ ký tự thường và dấu chấm
    - java.util, com.poly
  - Tên class, interface: Các từ phải viết hoa ký tự đầu
    - class Employee{}, class SinhVien{}, class HinhChuNhat()
  - Tên field, method, variable: Các từ phải viết hoa ký tự đầu ngoại trừ từ đầu tiên phải viết thường
    - hoTen, diem, fullName, mark
    - setHoTen(), input(), setDiem()
- Tên class, field và variable sử dụng danh từ
- Tên phương thức sử dụng động từ


Q&A