Study java

Contents

[I. Kỹ thuật lập trình căn bản 1](#_Toc164235271)

[1) Các kiểu dữ liệu 1](#_Toc164235272)

[2) Các câu lệnh if, for, while, do-while, case switch… Cho ví dụ minh họa, sự khác nhau. 1](#_Toc164235273)

[II. Lập trình hướng đối tượng 2](#_Toc164235274)

[1. Class và object là gì? 2](#_Toc164235275)

[1. Lớp (Class) là gì? 2](#_Toc164235276)

[2. Đối tượng (Object) là gì? 2](#_Toc164235277)

[3. Sự khác biệt giữa class và object: 3](#_Toc164235278)

[2. Lập trình hướng đối tượng (OOP) là gì? 3](#_Toc164235279)

[a. Tính đa hình: 3](#_Toc164235280)

[b. Tính đóng gói 5](#_Toc164235281)

[c. Tính kế thừa 6](#_Toc164235282)

[d. Tính trừu tượng 6](#_Toc164235283)

[III. Xử lý ngoại lệ (Exeption) trong java 9](#_Toc164235284)

[1. Cơ chế try-catch 9](#_Toc164235285)

[2. Throw ngoại lệ 10](#_Toc164235286)

[3. Chuyển tiếp ngoại lệ 10](#_Toc164235287)

# Kỹ thuật lập trình căn bản

## Các kiểu dữ liệu

* Nguyên thủy: Byte, short, int, long, float, double, char, boolean
* Không nguyên thủy: String, Array, classes, interface, enum, wrapper

## Các câu lệnh if, for, while, do-while, case switch… Cho ví dụ minh họa, sự khác nhau.

* if - else:

int x = 10;

if(x < 100){

System.out.println(x)

}else{

System.out.println(x + “lớn hơn 100”);

}

* for:

for(int i = 0; i < 5; i++){

System.out.println(“Giá trị i: ” + i);

}

## Nhập trong java

### Scanner

* Đọc chuỗi trong màn console bằng phương thức nextLine()
* Đọc số từ màn console bằng:
  + nextInt()
  + nextDouble()
  + nextFloat()

### Buffered Reader

* Đọc chuỗi từ màn hình bằng realLine()
* Đọc số bằng:
  + Integer.parseInt()
  + Double.parseDouble()

## Các kiểu dữ liệu

| **Kiểu** | **Mô tả** | **Kích cỡ** | **Tối thiểu** | **Tối đa** | **Lớp bao** |
| --- | --- | --- | --- | --- | --- |
| byte | số nguyên một byte | 8 bit | -128 | 127 | Byte |
| short | số nguyên ngắn | 16 bit | -2^15 | 2^15-1 | Short |
| int | số nguyên | 32 bit | -2^31 | 2^31-1 | Integer |
| long | số nguyên dài | 64 bit | -2^63 | -2^63-1 | Long |
| float | kiểu thực với độ chính xác đơn | 32 bit | IEEE754 | IEEE754 | Float |
| double | Double-precision floating point | 64 bit | IEEE754 | IEEE754 | Double |
| char | kiểu kí tự | 16 bit | Unicode 0 | Unicode 2^16-1 | Character |
| boolean | kiểu logic | true hoặc false | - | - | Boolean |
| void | - | - | - | - | Void |

## Hàm và phương thức

* Hàm có kết quả trả về sẽ không có void
* Hàm không có kết quả trả về sẽ có void

## Ép kiểu trong java

* (kiểu dữ liệu) biểu thức
* Ví dụ:
  + (float) 3 \* 5;

## Chuyển kiểu dữ liệu

* Chuyển String sang int
  + int i = Integer.parseInt("200");
* Chuyển đổi int thành String trong java
  + String.valueOf() hoặc Integer.toString()

## Toán tử 3 ngôi

* Ví dụ: String result = a % 2 == 0 ? "a là số chẵn" : "a là số lẻ";

# Collections trong Java

## Collection là gì?

* Collection là gì: Trong lập trình Java, Collections (Tập hợp) là một khái niệm quan trọng và mạnh mẽ cho việc lưu trữ và quản lý một nhóm các đối tượng. Chúng ta có thể thực hiện nhiều thao tác trên các tập hợp như tìm kiếm, sắp xếp, chèn, xóa và nhiều thao tác khác
* Là một khung (framework) cung cấp kiến trúc để lưu trữ và thao tác trên các tập hợp đối tượng

## List trong java

* Trong Java, chúng ta phải import package là java.util.List để sử dụng được List.
  + ArrayList
    - Thêm phần tử vào ArrayList: add(“adb”)
    - Thêm phần tử ArrayList này qua ArrayList khác: addAll(mảng)
    - Truy cập tới phần tử trong ArrayList: mảng.get(0)
    - Thay đổi phần tử trong ArrayList: mảng.set(1, “bcv”);
    - Xóa phần tử mảng: mảng.remove(1);
    - Xóa tất cả phần tử: mảng.clear();
    - Kiểm tra độ dài của mảng: mảng.size();

## Set trong java

* Thêm phần từ set.add();
* Xóa phần tử set.remove(“”);
* Xóa tất cả trong set: set.clear();

# Lập trình hướng đối tượng

## Class và object là gì?

### Lớp (Class) là gì?

* Trong Java, một lớp là một thực thể xác định hành vi mà một đối tượng có và sẽ có. Nói cách khác, một lớp chỉ là một bản thiết kế hoặc một tập hợp các hướng dẫn để xây dựng các đặc tính của một đối tượng cụ thể sau này.
* **Cách tạo một lớp trong java:**

class <class\_name>{

field;

method;

}

Trong đó:

* Class: từ khóa để tạo một class
* <class\_name>: tên class bạn định tạo nó sẽ được viết liền và viết HOA chữ cái đầu tiên của từng từ
* Field: biến đối tượng
* Method: phương thức

### Đối tượng (Object) là gì?

* Một Object có thể chứa các thành phần như các phương thức (method) và thuộc tính (thuộc tính) để tạo ra các kiểu dữ liệu hữu ích. Đối tượng xác định hành vi của lớp. Khi bạn gửi tin nhắn đến một đối tượng, bạn bắt buộc phải gọi đối tượng hoặc thực hiện một trong các phương thức của nó.
* **Cách tạo một object:**

<class\_name> ReferenceVariable = new <class\_name>();

Trong đó:

* <class\_name>: kiểu dữ liệu của đối tượng
* ReferenceVariable: tên tham chiếu của đối tượng
* new: từ khóa dùng tạo đối tượng
* <class\_name>(): class dùng để tạo bạn đối tượng

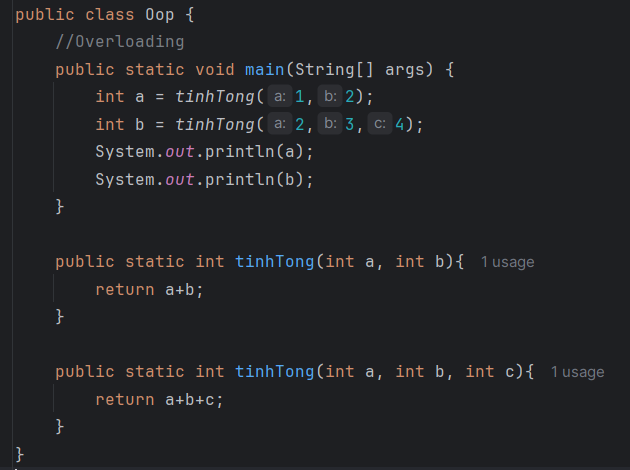
### Sự khác biệt giữa class và object:

|  |  |  |
| --- | --- | --- |
| STT | Lớp (Class) | Đối tượng (Object) |
| 1 | Là bản thiết kế hoặc khuôn mẫu mà từ đó đối tượng được tạo ra | Là một thể hiện của lớp |
| 2 | Là tập hợp những đối tượng có thuộc tính giống nhau | Là một thực thể trong thế giới thực như bút, máy tính, giường,… |
| 3 | Chỉ cần khai báo 1 lần | Khai báo bao nhiêu lần cũng được |
| 4 | Được khai báo với từ khóa Class | Đối tượng được khai báo với từ khóa “new” hoặc tên lớp |
| 5 | Lớp không được cấp phát bộ nhớ khi được tạo | Đối tượng được phân bộ nhớ mỗi khi khởi tạo |
| 6 | Thực thể logic | Thực thể vật lý |
| 7 | Như lớp động vật (Animal) | Đối tượng là một thực thể như con mèo (cat), con chó (dog) |

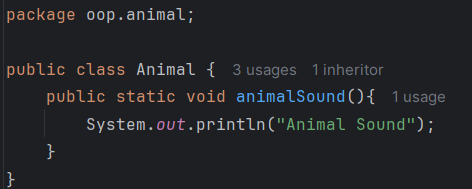
## Lập trình hướng đối tượng (OOP) là gì?

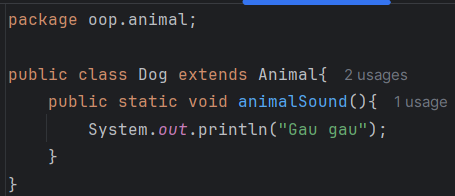
### Tính đa hình:

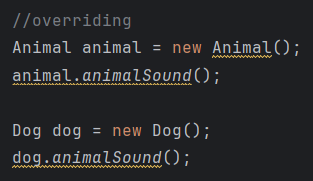
* OOP cho phép bạn sử dụng tính đa hình, nghĩa là bạn có thể sử dụng các đối tượng cùng một lớp một cách đa dạng.
  + Ví dụ, bạn có thể sử dụng một phương thức "di chuyển" cho nhiều loại đối tượng khác nhau, như “Người” và “Xe hơi”.
* **Overloading (Nạp chồng phương thức) là gì?**
  + Cho phép một class có nhiều phương thức cùng thêm nhưng khác nhau về tham số truyền vào, hoặc kiểu dữ liệu tham số:
  + Ví dụ:

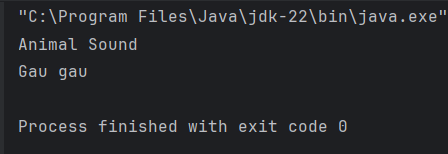


* **Overriding (Ghi đè phương thức) là gì?**
  + Overrding (tên đầy đủ là Method Overriding), được sử dụng trong trường hợp lớp con kế thừa từ lớp cha và muốn định nghĩa lại một phương thức đã có mặt ở lớp cha
  + Ví dụ:

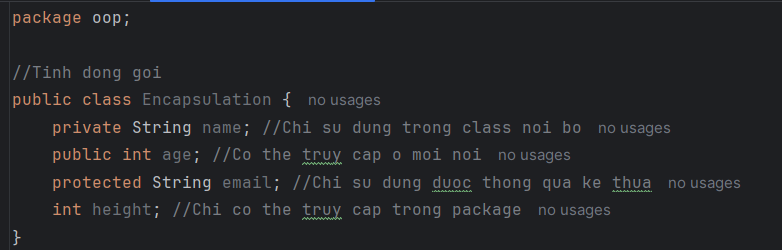






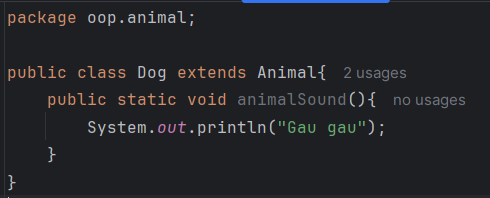


### Tính đóng gói



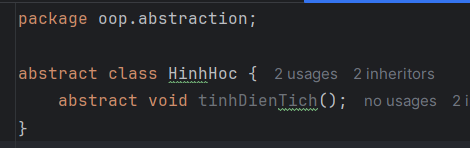
* private: chỉ sử dụng trong class nội bộ
* public: truy cập ở mọi nơi
* protected: sử dụng khi được kế thừa
* default: chỉ sử dụng trong packge

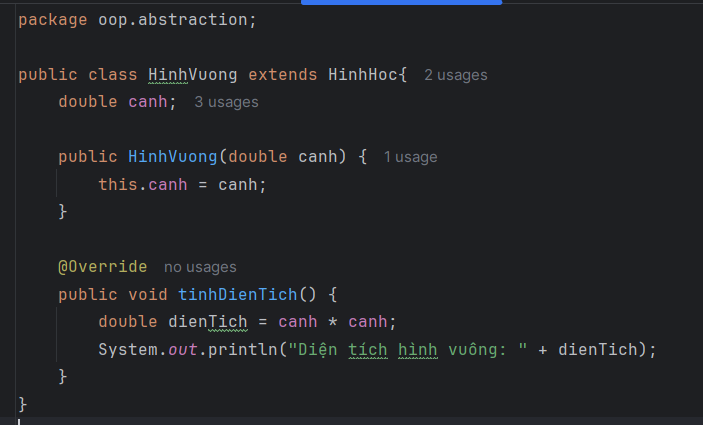
### Tính kế thừa

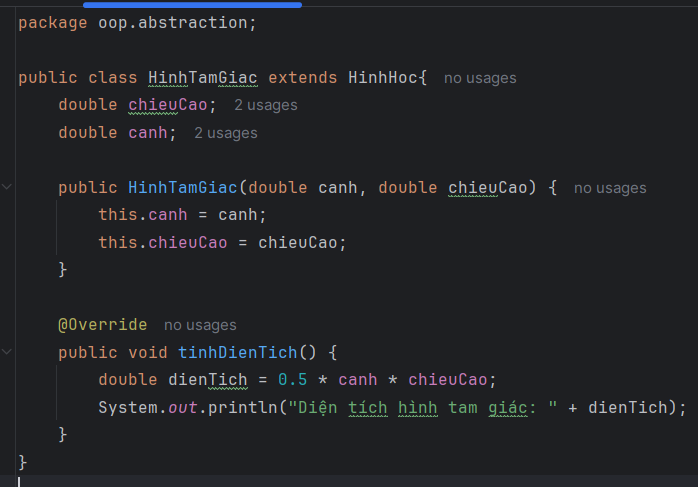
* Tính kế thừa giúp tái sử dụng mã nguồn và tạo ra cấu trúc phân cấp cho các lớp. Điều này cho phép bạn xây dựng các lớp mới trên cơ sở của các lớp đã có, tiết kiệm thời gian và công sức, đồng thời duy trì tính nhất quán trong mã nguồn. 

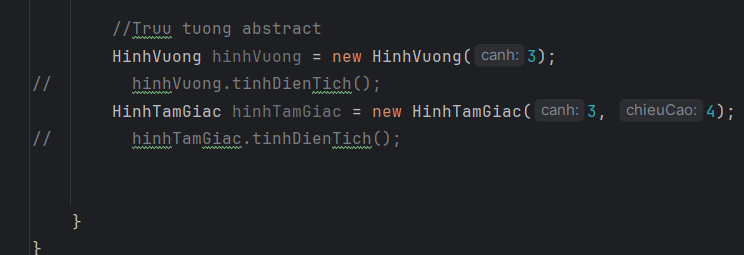
### Tính trừu tượng

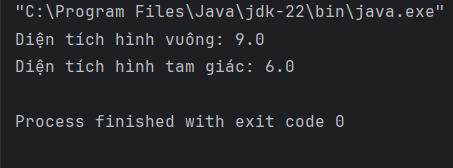
* Abstract nó cho phép bạn tạo ra các lớp và đối tượng trừu tượng, tập trung vào các tính năng quan trọng mà bạn quan tâm và che giấu chi tiết phức tạp bên trong
* Ví dụ:



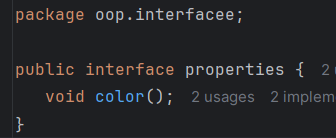


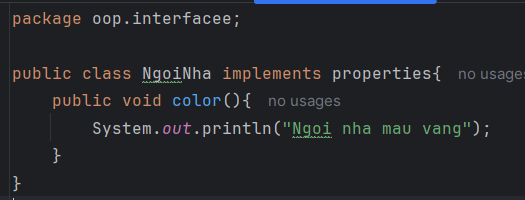


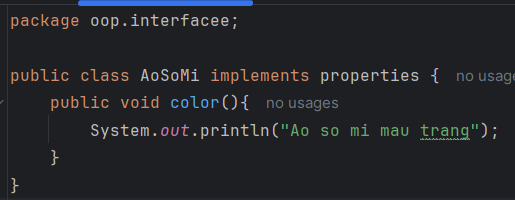


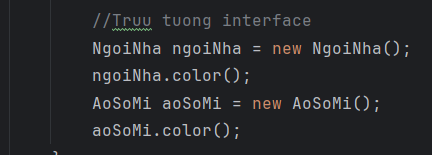


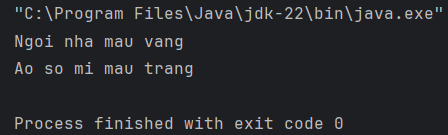
* Interface là một bản thiết kế của một lớp. Nó chỉ có các phương thức trừu tượng. Interface là một kỹ thuật để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java
* Ví dụ:











# Xử lý ngoại lệ (Exeption) trong java

* Exception là một tình trạng bất thường xảy ra trong quá trình thực thi chương trình, phá vỡ luồng chuẩn của chương trình.
* Checked Exception: Đây là các ngoại lệ mà chúng ta phải kiểm tra tại thời điểm biên dịch. Checked Exception thường xảy ra do người dùng và không thể dự đoán được bởi lập trình viên. Ví dụ, IOException, SQLException.
* Unchecked Exception: Đây là các ngoại lệ không cần kiểm tra tại thời điểm biên dịch và thường xảy ra do lỗi trong mã nguồn, chẳng hạn như NullPointerException, ArithmeticException.
* Error: Là các ngoại lệ không thường được đặt bẫy, và chúng thường xảy ra do lỗi vượt quá tầm kiểm soát của lập trình viên, ví dụ như OutOfMemoryError, StackOverflowError.

## Cơ chế try-catch

* Cú pháp của try-catch trong java

Try{

<mã lệnh có thể xảy ra ngoại lệ>

} catch (Exception\_Type e) {

<Xử lý ngoại lệ ở đây>

}

* try: Khối mã lệnh mà bạn muốn kiểm tra ngoại lệ.
* catch: Khối mã lệnh sẽ được thực thi nếu ngoại lệ xảy ra. Exception\_Type là kiểu ngoại lệ mà bạn muốn bắt.

## Throw ngoại lệ

* Từ khóa throw trong Java được sử dụng để ném ra một exception tường minh. Chúng ta có thể ném cả Checked Exception và Unchecked Exception trong Java bằng từ khóa throw. Từ khóa throw thường được sử dụng để ném các Custom Exception
* Cú pháp throw trong Java như sau:

throw exception;

* Ví dụ về việc sử dụng throw để ném ra một IOException:

throw new IOException("File không tồn tại");

## Chuyển tiếp ngoại lệ

* Trong Java, chuyển tiếp ngoại lệ (exception forwarding) là quá trình mà một phương thức không xử lý hoặc không có khả năng xử lý một ngoại lệ, nhưng nó chuyển tiếp (forward) ngoại lệ đó đến một phương thức gọi khác để xử lý.
* 