**MỤC LỤC**

[CHƯƠNG 1. CẤU TRÚC JAVA 3](#_Toc202102178)

[1.1. Thành phần trong 1 file java 3](#_Toc202102179)

[1.1.1. Khai báo gói (Package Declaration - Tùy chọn) 3](#_Toc202102180)

[1.1.2. Import các lớp (Import Statements - Tùy chọn) 3](#_Toc202102181)

[1.1.3. Khai báo lớp (Class Declaration) 3](#_Toc202102182)

[1.1.4. Các biến thành viên/trường (Member Variables/Fields - Tùy chọn) 4](#_Toc202102183)

[1.1.5. Hàm tạo (Constructors - Tùy chọn) 4](#_Toc202102184)

[1.1.6. Các phương thức (Methods - Tùy chọn, trừ main) 4](#_Toc202102185)

[1.1.7. Phương thức main (Điểm bắt đầu thực thi) 5](#_Toc202102186)

[1.1.8. Tóm lại 5](#_Toc202102187)

[1.2. Những lưu ý 6](#_Toc202102188)

[1.3. THỨ TỰ THỰC HIỆN 7](#_Toc202102189)

[CHƯƠNG 2. BIẾN JAVA 8](#_Toc202102190)

[2.1. Kiểu dữ liệu 8](#_Toc202102191)

[2.1.1. Kiểu dữ liệu nguyên thủy (Primitive Data Types) 8](#_Toc202102192)

[2.1.2. Kiểu dữ liệu tham chiếu (Reference Data Types) 9](#_Toc202102193)

[CHƯƠNG 3. HÀM 10](#_Toc202102194)

[3.1.1. Điều kiện 10](#_Toc202102195)

[3.1.2. Loop 10](#_Toc202102196)

[3.1.3. Scanner 10](#_Toc202102197)

[CHƯƠNG 4. Bài tập 11](#_Toc202102198)

[4.1. Dạng Nhập xuất. 11](#_Toc202102199)

[4.1.1. Nhập xuất cơ bản 11](#_Toc202102200)

[4.1.2. Nhập xuất cơ bản 2 12](#_Toc202102201)

[4.1.3. Nhập xuất cơ bản 3 13](#_Toc202102202)

[4.1.4. Tổng chữ số 14](#_Toc202102203)

[4.1.5. Tính Tổng 15](#_Toc202102204)

[4.1.6. Dao nguoc 15](#_Toc202102205)

[4.1.7. Tính tổng 3. 16](#_Toc202102206)

[4.1.8. Nhập xuất. 17](#_Toc202102207)

[4.2. Dạng điều kiện 18](#_Toc202102208)

[4.3. Dạng vòng lặp 18](#_Toc202102209)

[4.4. dạng hàm 18](#_Toc202102210)

[CHƯƠNG 5. Thư viện util 18](#_Toc202102211)

[5.1. Scanner 18](#_Toc202102212)

[CHƯƠNG 6. CÁC Lớp CƠ BẢN 20](#_Toc202102213)

[6.1. System 20](#_Toc202102214)

[6.1.1. System.in 20](#_Toc202102215)

[6.1.2. System.out 20](#_Toc202102216)

[6.2. ép kiểu 20](#_Toc202102217)

# CẤU TRÚC JAVA

## Thành phần trong 1 file java

### Khai báo gói (Package Declaration - Tùy chọn)

Ví dụ: package fullstack101.backend.java;

**package** : Được sử dụng để nhóm các lớp liên quan lại với nhau. Nó giúp tổ chức mã nguồn và tránh xung đột tên lớp.

### Import các lớp (Import Statements - Tùy chọn)

Vd: import java.util.Scanner;

**import :** Được sử dụng để đưa các lớp từ các gói khác vào chương trình của bạn. Điều này cho phép bạn sử dụng các lớp đó mà không cần viết tên đầy đủ của gói.

Ví dụ: java.util.Scanner cho phép bạn sử dụng lớp Scanner để đọc dữ liệu nhập từ bàn phím.

### Khai báo lớp (Class Declaration)

**Cú pháp** : **public class TenLopCuaBan { ... }**

* Mỗi chương trình Java phải có ít nhất một lớp.
* **public**: Từ khóa truy cập, nghĩa là lớp này có thể được truy cập từ bất kỳ đâu.
* **class**: Từ khóa để khai báo một lớp.

Lưu ý:

|  |  |  |
| --- | --- | --- |
| Trường hợp | Có hợp lệ không? | Ghi chú |
| 1 lớp public, tên trùng tên file | Hợp lệ | Cách chuẩn nên dùng |
| Nhiều lớp, không có lớp nào public | Hợp lệ | Dùng trong chương trình nhỏ, test code |
| Nhiều lớp public trong 1 file | Sai | Lỗi biên dịch – chỉ 1 lớp public cho mỗi file |

Bảng 1.1 Lưu ý khi sử dụng class Java

### Các biến thành viên/trường (Member Variables/Fields - Tùy chọn)

int soNguyen = 10;

String ten = "Java";

* Đây là các biến được khai báo bên trong một lớp nhưng bên ngoài bất kỳ phương thức nào. Chúng đại diện cho trạng thái hoặc thuộc tính của đối tượng.
* **Nơi lưu trữ giá trị**.

### Hàm tạo (Constructors - Tùy chọn)

**Cú pháp:** **public TenLopCuaBan() { ... }**

Hàm tạo là một phương thức đặc biệt được sử dụng để khởi tạo các đối tượng của một lớp.

* Tên của hàm tạo phải giống hệt tên lớp.

Nó không có kiểu trả về (không có void hoặc int,...).

### Các phương thức (Methods - Tùy chọn, trừ main)

**Cú pháp: public void hienThiThongTin() { ... }**

Các phương thức là các khối mã thực hiện một nhiệm vụ cụ thể. Chúng đại diện cho hành vi của đối tượng.

* public: Từ khóa truy cập.
* void: Kiểu trả về (trong trường hợp này là không trả về gì). Có thể là int, String, v.v., tùy thuộc vào giá trị mà phương thức trả về.
* hienThiThongTin: Tên của phương thức.

### Phương thức main (Điểm bắt đầu thực thi)

* **Cú pháp: public static void main(String[] args) { ... }**

Đây là phương thức đặc biệt và **bắt buộc** trong mỗi ứng dụng Java độc lập (standalone application). Khi bạn chạy một chương trình Java, Java Virtual Machine (JVM) sẽ tìm và thực thi phương thức main này đầu tiên.

* **public:** Cho phép JVM truy cập phương thức này từ bên ngoài lớp.
* **static:** Cho phép phương thức main được gọi mà không cần tạo một đối tượng của lớp. Điều này là cần thiết vì JVM cần một điểm bắt đầu mà không cần một instance cụ thể.
* **void:** Phương thức main không trả về bất kỳ giá trị nào.
* **main:** Tên của phương thức. Tên này là cố định.
* **String[] args**: Đây là một mảng các chuỗi. Nó được sử dụng để nhận các đối số dòng lệnh khi bạn chạy chương trình từ terminal

### Tóm lại

|  |  |
| --- | --- |
| Thành phần | Giải thích đơn giản |
| class | Nơi chứa mọi thứ trong Java (giống như "cái hộp lớn") |
| main() | Điểm bắt đầu của chương trình |
| method | Hàm – tập hợp lệnh để làm việc cụ thể |
| variable | Biến – dùng để lưu trữ dữ liệu |
| import | Nhập các lớp từ thư viện bên ngoài |
| package | Tổ chức file theo nhóm |
| comment | Ghi chú trong mã (không ảnh hưởng khi chạy) |

## Những lưu ý

* **Tên tệp** và tên lớp : Tên tệp **.java** phải khớp chính xác với tên của lớp chứa phương thức main.

A computer screen shot of a code

AI-generated content may be incorrect.

* **Cách biên dịch và chạy:** Đảm bảo rằng bạn đang biên dịch và chạy đúng cách từ thư mục chứa tệp nguồn.

## THỨ TỰ THỰC HIỆN

A diagram of a computer code

AI-generated content may be incorrect.

# BIẾN JAVA

## Kiểu dữ liệu

**Java là một ngôn ngữ định kiểu tĩnh (statically-typed)**. Điều này có nghĩa là bạn **phải khai báo tường minh kiểu dữ liệu** cho mỗi biến trước khi sử dụng, và kiểu dữ liệu đó sẽ không thể thay đổi.

### Kiểu dữ liệu nguyên thủy (Primitive Data Types)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kiểu dữ liệu (Keyword) | Mô tả | Kích thước | Giá trị Mặc định | Phạm vi giá trị / Ví dụ |
| byte | Số nguyên rất nhỏ | 1 byte | 0 | -128 đến 127 |
| short | Số nguyên nhỏ | 2 bytes | 0 | -32,768 đến 32,767 |
| int | Số nguyên (phổ biến nhất) | 4 bytes | 0 | -2,147,483,648 đến 2,147,483,647 |
| long | Số nguyên lớn | 8 bytes | 0L | -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807 (Ví dụ: 12345L) |
| float | Số thực (độ chính xác đơn) | 4 bytes | 0.0f | Lưu số có dấu phẩy động (Ví dụ: 19.99f) |
| double | Số thực (độ chính xác kép, phổ biến nhất) | 8 bytes | 0.0d | Lưu số có dấu phẩy động (Ví dụ: 3.14159) |
| char | Một ký tự Unicode | 2 bytes | \u0000' | Một ký tự duy nhất (Ví dụ: 'A', '@') |
| boolean | Giá trị logic | ~1 bit | FALSE | Chỉ có thể là true hoặc false |

Bảng 2.1 Kiểu dữ liệu nguyên thủy

**Lưu ý:**

**Hậu tố L và f:** Đừng quên thêm L vào cuối số long và f vào cuối số float để trình biên dịch Java biết chính xác bạn đang muốn dùng kiểu dữ liệu nào.

### Kiểu dữ liệu tham chiếu (Reference Data Types)

Những kiểu dữ liệu này không lưu trữ giá trị thực tế của dữ liệu, thay vào đó chúng lưu trữ một **tham chiếu** (địa chỉ bộ nhớ) trỏ đến nơi đối tượng được lưu trữ (trên bộ nhớ heap).

|  |  |  |
| --- | --- | --- |
| Kiểu | Dùng để lưu | Ví dụ |
| String | Chuỗi văn bản | String name = "An"; |
| Array | Mảng nhiều giá trị có cùng kiểu dữ liệu nguyên thủy. | int[] nums = {1, 2, 3}; |
| Class/Object | Đối tượng lập trình hướng đối tượng | Student s = new Student(); |
| interface | Kiểu dữ liệu người dùng tự định nghĩa |  |
| List, Map | Cấu trúc dữ liệu nâng cao | List<String> names = new ArrayList<>(); |

Bảng 2.2 Kiểu dữ liệu tham chiếu.

**Lưu ý:**

* **Vì nó trỏ đến địa chỉ nên khi thay đổi giá trị của dữ liệu thì nó sẽ tạo ra một đối tượng mới**.
* **Mảng sẽ có số lượng phần tử cố định.**

Ví dụ: name = MinhQuan; name = name + 123; ⇨ name = MinhQuan123 (Đây là đối tượng mới).

# HÀM

### Điều kiện

### Loop

### Scanner

# Bài tập

## Dạng Nhập xuất.

### Nhập xuất cơ bản

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.1 Nhập xuất cơ bản

Nhập hai chuỗi : bằng đối tượng scanner thuộc lớp Scanner với phương thức next để không lấy khoảng trắng.

Sau đó dùng System.out.print: Để in ra chuỗi

import java.util.Scanner;

class Main {

public static void main(String arg[]){

Scanner scanner = new Scanner(System.in);

String number1 = scanner.next();

String number2 = scanner.next();

System.out.print(number2 +' '+ number1);

}

### Nhập xuất cơ bản 2

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.2 Nhập xuất cơ bản.

* Lời giải ta sẽ dùng System.out.println() : để in ra chuỗi số và nó sẽ xuống dòng.

public class Main {

public static void main(String Arg[]){

System.out.println(1);

System.out.println(12);

System.out.println(123);

System.out.println(1234);

System.out.println(12345);

};

};

### Nhập xuất cơ bản 3

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.3 Ép kiểu

* Nhập hai số ở dang chuỗi sau tính tổng bằng cách ép kiểu về số Nguyên
* Integer.paraseInt() : ép kiểu về số nguyên.   
  import java.util.Scanner;

public class Main {

public static void main(String arg[]){

Scanner scanner = new Scanner(System.in);

String a = scanner.next();

String b = scanner.next();

System.out.print( a + "+" + b + "=");

int c = Integer.parseInt(a) + Integer.parseInt(b);

System.out.print(c);

};

};

### Tổng chữ số

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.4 Tổng chữ số

Tách nhỏ chữ số bằng cách

Chữ số hàng đơn vị lấy bằng cách chia dư 10.

Chữ số hàng chục bằng chia hết cho 10.

import java.util.Scanner;

public class Main {

public static void main(String arg[]){

Scanner scanner = new Scanner(System.in);

int number = scanner.nextInt();

int sum = number % 10 + number / 10;

System.out.print(sum);

}

}

### Tính Tổng

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.5 Tính tổng 2 số a + b

* Ý tưởng dùng hàm Scanner
* Dùng phương thước nextInt() để nhập số.

### Dao nguoc

A white background with black text

AI-generated content may be incorrect.

Hình 4.6 Dao nguoc

* Ý tưởng. Dùng Scanner để nhập và in System.out.print() ngược lại.

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 4.7 Code dao nguoc

### Tính tổng 3.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.8 Tổng chữ số cuối cùng

Ta sẽ thực hiện lấy ký tự từ bàn phím bằng lớp Scanner kết hợp System.in để đọc byte.

Với phương thức nextInt() : đọc đầu vào như 1 kiểu số nguyên.

Lấy phần tử cuối cùng của số = số % 10. (dùng phép chia lấy dư)

import java.util.Scanner;

class Main {

public static void main(String arg[]) {

Scanner scanner = new Scanner(System.in);

int input1 = scanner.nextInt();

int input2 = scanner.nextInt();

int tong\_so\_cuoi = input1 % 10 + input2 % 10;

System.out.print(tong\_so\_cuoi);

}

};

### Nhập xuất.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4.9 Lấy chữ số sau chữ số thập phân

## Dạng điều kiện

## Dạng vòng lặp

## dạng hàm

# Thư viện util

## Scanner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nhóm Phương Thức | Tên Phương Thức | Kiểu Trả Về | Mô Tả | Ví Dụ Cơ Bản (Đầu vào: 123 Hello true) |
| Đọc Dữ Liệu | next() | String | Đọc từ tiếp theo (chuỗi không chứa dấu cách). | scanner.next() -> "123" |
|  | nextLine() | String | Đọc toàn bộ dòng tiếp theo (bao gồm cả dấu cách) cho đến khi gặp ký tự xuống dòng. | Nếu đầu vào là "Hello World", scanner.nextLine() -> "Hello World" |
|  | nextInt() | int | Đọc giá trị số nguyên (int) tiếp theo. | scanner.nextInt() -> 123 |
|  | nextDouble() | double | Đọc giá trị số thực (double) tiếp theo. | Nếu đầu vào là "3.14", scanner.nextDouble() -> 3.14 |
|  | nextBoolean() | boolean | Đọc giá trị boolean (true hoặc false) tiếp theo. | scanner.nextBoolean() -> true |
|  | nextByte() | byte | Đọc giá trị byte tiếp theo. | scanner.nextByte() -> (giá trị byte) |
|  | nextFloat() | float | Đọc giá trị số thực (float) tiếp theo. | scanner.nextFloat() -> (giá trị float) |
|  | nextLong() | long | Đọc giá trị số nguyên (long) tiếp theo. | scanner.nextLong() -> (giá trị long) |
|  | nextShort() | short | Đọc giá trị số nguyên (short) tiếp theo. | scanner.nextShort() -> (giá trị short) |
| Kiểm Tra Dữ Liệu | hasNext() | boolean | Trả về true nếu có một token (từ) khác trong đầu vào. | scanner.hasNext() -> true |
|  | hasNextLine() | boolean | Trả về true nếu có một dòng khác trong đầu vào. | scanner.hasNextLine() -> true |
|  | hasNextInt() | boolean | Trả về true nếu token tiếp theo là một giá trị int. | scanner.hasNextInt() -> true |
|  | hasNextBoolean() | boolean | Trả về true nếu token tiếp theo là một giá trị boolean. | scanner.hasNextBoolean() -> true |
|  | (Tương tự cho các kiểu dữ liệu khác như Double, Float, Byte, Long, Short) | boolean | Kiểm tra xem token tiếp theo có phải là kiểu dữ liệu tương ứng hay không. |  |
| Cấu Hình & Đóng | close() | void | Đóng Scanner và giải phóng các tài nguyên hệ thống liên quan. | scanner.close() |
|  | useDelimiter(String pattern) | Scanner | Đặt mẫu phân cách mà Scanner sẽ sử dụng để chia nhỏ đầu vào thành các token (mặc định là khoảng trắng). | scanner.useDelimiter(",") |
|  | useLocale(Locale locale) | Scanner | Đặt locale mà Scanner sử dụng để định dạng số. | scanner.useLocale(Locale.US) |
|  | reset() | Scanner | Đặt lại Scanner về trạng thái ban đầu, xóa mọi lỗi hoặc dữ liệu đang chờ xử lý. | scanner.reset() |

# CÁC Lớp CƠ BẢN

## System

### System.in

### System.out

## ép kiểu