**TÌM HIỂU VỀ NGÔN NGỮ LẬP TRÌNH JAVA**

## Tổng quan vế ngôn ngữ lập trình java

### Java là gì ?

**Java** là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) [hướng đối tượng](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) (OOP) và dựa trên các lớp class. Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì [biên dịch](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch) [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) thành [mã máy](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_m%C3%A1y) hoặc [thông dịch](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_th%C3%B4ng_d%E1%BB%8Bch) mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành [bytecode](https://vi.wikipedia.org/wiki/Bytecode), bytecode sau đó sẽ được môi trường thực thi (runtime environment) chạy.

Trước đây, Java chạy chậm hơn những ngôn ngữ dịch thẳng ra mã máy như C và [C++](https://vi.wikipedia.org/wiki/C%2B%2B), nhưng sau này nhờ công nghệ "biên dịch tại chỗ" - Just in time compilation, khoảng cách này đã được thu hẹp, và trong một số trường hợp đặc biệt Java có thể chạy nhanh hơn. Java chạy nhanh hơn những ngôn ngữ thông dịch như [Python](https://vi.wikipedia.org/wiki/Python), [Perl](https://vi.wikipedia.org/wiki/Perl), [PHP](https://vi.wikipedia.org/wiki/PHP) gấp nhiều lần. Java chạy tương đương so với [C#](https://vi.wikipedia.org/wiki/C_th%C4%83ng), một ngôn ngữ khá tương đồng về mặt cú pháp và quá trình dịch/chạy.

[Cú pháp](https://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_h%E1%BB%8Dc) Java được vay mượn nhiều từ [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)) & [C++](https://vi.wikipedia.org/wiki/C%2B%2B) nhưng có cú pháp hướng đối tượng đơn giản hơn và ít tính năng xử lý cấp thấp hơn. Do đó việc viết một chương trình bằng Java dễ hơn, đơn giản hơn, đỡ tốn công sửa lỗi hơn.

Trong Java, hiện tượng rò rỉ bộ nhớ hầu như không xảy ra do bộ nhớ được quản lý bởi Java Virtual Machine (JVM) bằng cách tự động "dọn dẹp rác". Người lập trình không phải quan tâm đến việc cấp phát và xóa bộ nhớ như C, C++. Tuy nhiên khi sử dụng những tài nguyên mạng, file IO, database (nằm ngoài kiểm soát của JVM) mà người lập trình không đóng (close) các streams thì rò rỉ dữ liệu vẫn có thể xảy ra.

### 1.2.Lịch sử phát triển

Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun Microsystems năm 1991. Ban đầu ngôn ngữ này được gọi là Oak (có nghĩa là cây sồi; do bên ngoài cơ quan của ông Gosling có trồng nhiều loại cây này), họ dự định ngôn ngữ đó thay cho C++, nhưng các tính năng giống Objective C. Không nên lẫn lộn Java với JavaScript, hai ngôn ngữ đó chỉ giống tên và loại cú pháp như C. Công ty Sun Microsystems đang giữ bản quyền và phát triển Java thường xuyên.

Java được tạo ra với tiêu chí "Viết (code) một lần, thực thi khắp nơi" ("Write Once, Run Anywhere" (WORA)). Chương trình phần mềm viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua một môi trường thực thi với điều kiện có môi trường thực thi thích hợp hỗ trợ nền tảng đó. Môi trường thực thi của Sun Microsystems hiện hỗ trợ Sun Solaris, Linux, Mac OS, FreeBSD & Windows. Ngoài ra, một số công ty, tổ chức cũng như cá nhân khác cũng phát triển môi trường thực thi Java cho những hệ điều hành khác như BEA, IBM, HP.... Trong đó đáng nói đến nhất là IBM Java Platform hỗ trợ Windows, Linux, AIX & z/OS.

Java được sử dụng chủ yếu trên môi trường NetBeans và Oracle. Sau khi Oracle mua lại công ty Sun Microsystems năm 2009-2010, Oracle đã mô tả họ là "người quản lý công nghệ Java với cam kết không ngừng để bồi dưỡng một cộng đồng tham gia và minh bạch".[12]

### 1.3. Đặc điểm của ngôn ngữ lập trình Java

* **Tựa C++, hướng đối tượng hoàn toàn**

Trong quá trình tạo ra một ngôn ngữ mới phục vụ cho mục đích chạy được trên nhiều nền tảng, các kỹ sư của Sun MicroSystem muốn tạo ra một ngôn ngữ dễ học và quen thuộc với đa số người lập trình. Vì vậy họ đã sử dụng lại các cú pháp của C và C++.

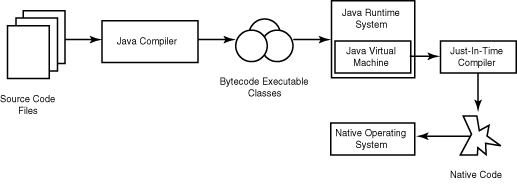
* **Độc lập phần cứng và hệ điều hành**

Một chương trình viết bằng ngôn ngữ Java có thể chạy tốt ở nhiều môi trường khác nhau. Gọi là khả năng "**cross-platform**”. Khả năng độc lập phần cứng và hệ điều hành được thể hiện ở 2 cấp độ là cấp độ **mã nguồn** và cấp độ **nhị phân**.

* **Ở cấp độ mã nguồn**: Kiểu dữ liệu trong Java nhất quán cho tất cả các hệ điều hành và phần cứng khác nhau. Java có riêng một bộ thư viện để hỗ trợ vấn đề này. Chương trình viết bằng ngôn ngữ Java có thể biên dịch trên nhiều loại máy khác nhau mà không gặp lỗi.
* **Ở cấp độ nhị phân**: Một mã biên dịch có thể chạy trên nhiều nền tảng khác nhau mà không cần dịch lại mã nguồn. Tuy nhiên cần có Java Virtual Machine để thông dịch đoạn mã này.
* **Ngôn ngữ thông dịch**

Ngôn ngữ lập trình Java thuộc loại ngôn ngữ **thông dịch**. Chính xác hơn, Java là loại ngôn ngữ vừa biên dịch vừa thông dịch. Cụ thể như sau

Khi viết mã, hệ thống tạo ra một tệp .java. Khi biên dịch mã nguồn của chương trình sẽ được biên dịch ra mã byte code. Máy ảo Java (Java Virtual Machine) sẽ thông dịch mã byte code này thành machine code (hay native code) khi nhận được yêu cầu chạy chương trình.



**Ưu điểm** : Phương pháp này giúp các đoạn mã viết bằng Java có thể chạy được trên nhiều nền tảng khác nhau. Với điều kiện là JVM có hỗ trợ chạy trên nền tảng này.

**Nhược điểm** : Cũng như các ngôn ngữ thông dịch khác, quá trình chạy các đoạn mã Java là chậm hơn các ngôn ngữ biên dịch khác (tuy nhiên vẫn ở trong một mức chấp nhận được).

* **Cơ chế thu gom rác tự động**

Khi tạo ra các đối tượng trong Java, JRE sẽ tự động cấp phát không gian bộ nhớ cho các đối tượng ở trên heap.

Với ngôn ngữ như C \ C++, bạn sẽ phải yêu cầu hủy vùng nhớ mà bạn đã cấp phát, để tránh việc thất thoát vùng nhớ. Tuy nhiên vì một lý do nào đó, bạn không hủy một vài vùng nhớ, dẫn đến việc thất thoát và làm giảm hiệu năng chương trình.

Ngôn ngữ lập trình Java hỗ trợ cho bạn điều đó, nghĩa là bạn không phải tự gọi hủy các vùng nhớ. Bộ thu dọn rác của Java sẽ theo vết các tài nguyên đã được cấp. Khi không có tham chiếu nào đến vùng nhớ, bộ thu dọn rác sẽ tiến hành thu hồi vùng nhớ đã được cấp phát.

* **Đa luồng**

Java hỗ trợ lập trình đa tiến trình (multithread) để thực thi các công việc đồng thời. Đồng thời cũng cung cấp giải pháp đồng bộ giữa các tiến trình (giải pháp sử dụng priority...).

* **Tính an toàn và bảo mật**

**Tính an toàn :**

Ngôn ngữ lập trình Java yêu cầu chặt chẽ về kiểu dữ liệu.

+ Dữ liệu phải được khai báo tường minh.

+ Không sử dụng con trỏ và các phép toán với con trỏ.

+ Java kiểm soát chặt chẽ việc truy nhập đến mảng, chuỗi. Không cho phép sử dụng các kỹ thuật tràn. Do đó các truy nhập sẽ không vượt quá kích thước của mảng hoặc chuỗi.

+ Quá trình cấp phát và giải phóng bộ nhớ được thực hiện tự động.

+ Cơ chế xử lý lỗi giúp việc xử lý và phục hồi lỗi dễ dàng hơn.

**Tính bảo mật :**

Java cung cấp một môi trường quản lý chương trình với nhiều mức khác nhau.

+ Mức 1 : Chỉ có thể truy xuất dữ liệu cũng như phương phức thông qua giao diện mà lớp cung cấp.

+ Mức 2 : Trình biên dịch kiểm soát các đoạn mã sao cho tuân thủ các quy tắc của ngôn ngữ lập trình Java trước khi thông dịch.

+ Mức 3 : Trình thông dịch sẽ kiểm tra mã byte code xem các đoạn mã này có đảm bảo được các quy định, quy tắc trước khi thực thi.

+ Mức 4: Java kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

### 1.4. các ứng dụng java

* **Java và ứng dụng console**

Ứng dụng Console là ứng dụng nhập xuất ở chế độ văn bản tương tự như màn hình Console của hệ điều hành MS-DOS. Lọai chương trình ứng dụng này thích hợp với những ai bước đầu làm quen với ngôn ngữ lập trình java. Các ứng dụng kiểu Console thường được dùng để minh họa các ví dụ cơ bản liên quan đến cú pháp ngôn ngữ, các thuật toán, và các chương trình ứng dụng không cần thiết đến giao diện người dùng đồ họa.

* **Java và ứng dụng applet**

Java Applet là loại ứng dụng có thể nhúng và chạy trong trang web của một trình duyệt web. Từ khi internet mới ra đời, Java Applet cung cấp một khả năng lập trình mạnh mẽ cho các trang web. Nhưng gần đây khi các chương trình duyệt web đã phát triển với khả năng lập trình bằng VB Script, Java Script, HTML, DHTML, XML, … cùng với sự canh tranh khốc liệt của Microsoft và Sun đã làm cho Java Applet lu mờ. Và cho đến bây giờ gần như các lập trình viên đều không còn “mặn mà” với Java Applet nữa. (trình duyệt IE đi kèm trong phiên bản Windows 2000 đã không còn mặc nhiên hỗ trợ thực thi một ứng dụng Java Applet). Hình bên dưới minh họa một chương trình java applet thực thi trong một trang web.

* **Java và phát triển ứng dụng Desktop dùng AWT và JFC**

Việc phát triển các chương trình ứng dụng có giao diện người dùng đồ họa trực quan giống như những chương trình được viết dùng ngôn ngữ lập trình VC++ hay Visual Basic đã được java giải quyết bằng thư viện AWT và JFC. JFC là thư viện rất phong phú và hỗ trợ mạnh mẽ hơn nhiều so với AWT. JFC giúp cho người lập trình có thể tạo ra một giao diện trực quan của bất kỳ ứng dụng nào

* **Java và phát triển ứng dụng Web**

Java hỗ trợ mạnh mẽ đối với việc phát triển các ứng dụng Web thông qua công nghệ J2EE (Java 2 Enterprise Edition). Công nghệ J2EE hoàn toàn có thể tạo ra các ứng dụng Web một cách hiệu quả không thua kém công nghệ .NET mà Microsft đang quảng cáo. Hiện nay có rất nhiều trang Web nổi tiếng ở Việt Nam cũng như khắp nơi trên thế giới được xây dựng và phát triển dựa trên nền công nghệ Java. Số ứng dụng Web được xây dựng dùng công nghệ Java chắc chắn không ai có thể biết được con số chính xác là bao nhiêu, nhưng chúng tôi đưa ra đây vài ví dụ để thấy rằng công nghệ Java của Sun là một “đối thủ đáng gờm” của Microsoft.

* **Java và phát triển các ứng dụng nhúng**

Java Sun đưa ra công nghệ J2ME (The Java 2 Platform, Micro Edition J2ME) hỗ trợ phát triển các chương trình, phần mềm nhúng. J2ME cung cấp một môi trường cho những chương trình ứng dụng có thể chạy được trên các thiết bị cá nhân như: điện thọai di động, máy tính bỏ túi PDA hay Palm, cũng như các thiết bị nhúng khác.

### 1.5. Dịch và thực thi một chương trình viết bằng Java

Việc xây dựng, dịch và thực thi một chương trình viết bằng ngôn ngữ lập trình java có thể tóm tắt qua các bước sau:

* Viết mã nguồn: dùng một chương trình soạn thảo nào đấy (NotePad hay Jcreator chẳng hạn) để viết mã nguồn và lưu lại với tên có đuôi “.java” .
* Biên dịch ra mã máy ảo: dùng trình biên dịch javac để biên dịch mã nguồn “.java” thành mã của máy ảo (java bytecode) có đuôi “.class” và lưu lên đĩa
* Thông dịch và thực thi: ứng dụng được load vào bộ nhớ, thông dịch và thực thi dùng trình thông dịch Java thông qua lệnh “java”.
  + Đưa mã java bytecode vào bộ nhớ: đây là bước “loading”. Chương trình phải được đặt vào trong bộ nhớ trước khi thực thi. “Loader” sẽ lấy các files chứa mã java bytecode có đuôi “.class” và nạp chúng vào bộ nhớ.
  + Kiểm tra mã java bytecode: trước khi trình thông dịch chuyển mã bytecode thành mã máy tương ứng để thực thi thì các mã bytecode phải được kiểm tra tính hợp lệ.
  + Thông dịch & thực thi: cuối cùng dưới sự điều khiển của CPU và trình thông dịch tại mỗi thời điểm sẽ có một mã bytecode được chuyển sang mã máy và thực thi.

### 1.6. Chương trình java đầu tiên

Viết trên notepad

/\*Viết chương trình in dòng HelloWorld lên màn hình Console\*/

class HelloWorldApp{

public static void main(String[] args)

{

//In dong chu “HelloWorld”

System.out.println(“HelloWorld”);

}

}

Lưu lại với tên HelloWorldApp.java

Việc biên dịch tập tin mã nguồn chương trình HelloWorldApp có thể thực hiện qua các bước cụ thể như sau:

* Mở cửa sổ Command Prompt.
* Chuyển đến thư mục chứa tập tin nguồn vừa tạo ra.
* Thực hiện câu lệnh: javac HelloWordApp.java

Nếu gặp thông báo lỗi “Bad Command of filename” hoặc “The name specified is not recognized as an internal or external command, operable program or batch file” có nghĩa là Windows không tìm được trình biên dịch javac. Để sửa lỗi này chúng ta cần cập nhật lại đường dẫn PATH của hệ thống. Ngược lại nếu thành công bạn sẽ có thêm tập tin HelloWordApp.class.

Cấu trúc chương trình HelloWordApp:

Phương thức main(): là điểm bắt đầu thực thi một ứng dụng. Mỗi ứng dụng Java phải chứa một phương thức main có dạng như sau: public static void main(String[] args) Phương thức main chứa ba bổ từ đặc tả sau:

* public chỉ ra rằng phương thức main có thể được gọi bỡi bất kỳ đối tượng nào.
* static chỉ ra rắng phương thức main là một phương thức lớp.
* void chỉ ra rằng phương thức main sẽ không trả về bất kỳ một giá trị nào.

## Hằng, biến, kiểu dữ lieuj, toán tử, biểu thức và các cấu trúc điều khiển trong java.

### Biến

Biến là vùng nhớ dùng để lưu trữ các giá trị của chương trình. Mỗi biến gắn liền với một kiểu dữ liệu và một định danh duy nhất gọi là tên biến.

* Tên biến thông thường là một chuỗi các ký tự (Unicode), ký số.
* Tên biến phải bắt đầu bằng một chữ cái, một dấu gạch dưới hay dấu dollar.
* Tên biến không được trùng với các từ khóa (xem phụ lục các từ khóa trong java).
* Tên biến không có khoảng trắng ở giữa tên.
* Trong java, biến có thể được khai báo ở bất kỳ nơi đâu trong chương trình.

Cách khai báo :

**<kiểu\_dữ\_liệu> <tên\_biến>;**

**<kiểu\_dữ\_liệu> <tên\_biến> = <giá\_trị>;**

Gán giá trị cho biến:

**<tên\_biến> = <giá\_trị>;**

**Biến toàn cục** : là biến có thể truy xuất ở khắp nơi trong chương trình, thường được khai báo dùng từ khóa public, hoặc đặt chúng trong một class.

**Biến cục bộ:** là biến chỉ có thể truy xuất trong khối lệnh nó khai báo.

### Các kiểu dữ liệu cơ sở

Ngôn ngữ lập trình java có 8 kiểu dữ liệu cơ sở: byte, short, int, long, float, double, boolean và char.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kiểu | Kích thước (bytes) | Giá trị min | Giá trị max | Giá trị mặc định |
| Byte | 1 | -256 | 255 | 0 |
| Short | 2 | -32768 | 32767 | 0 |
| Int | 4 | -231 | 231-1 | 0 |
| Long | 8 | -263 | 263-1 | 0L |
| float | 4 |  |  | 0.0f |
| double | 8 |  |  | 0.0d |

### Hằng

Hằng là một giá trị bất biến trong chương trình

- Tên hằng được đặt theo qui ước giống như tên biến.

- Hằng số nguyên: trường hợp giá trị hằng ở dạng long ta thêm vào cuối chuỗi số chữ “l” hay “L”. (ví dụ: 1L)

- Hằng số thực: truờng hợp giá trị hằng có kiểu float ta thêm tiếp vĩ ngữ “f” hay “F”, còn kiểu số double thì ta thêm tiếp vĩ ngữ “d” hay “D”.

- Hằng Boolean: java có 2 hằng boolean là true, false.

- Hằng ký tự: là một ký tự đơn nằm giữa nằm giữa 2 dấu ngoặc đơn.

- Hằng chuỗi: là tập hợp các ký tự được đặt giữa hai dấu nháy kép “”. Một hằng chuỗi không có ký tự nào là một hằng chuỗi rỗng.

**2.4. Lệnh, khối lệnh trong java**

Giống như trong ngôn ngữ C, các câu lệnh trong java kết thúc bằng một dấu chấm phẩy (;).

Một khối lệnh là đoạn chương trình gồm hai lệnh trở lên và được bắt đầu bằng dấu mở ngoặc nhọn ({) và kết thúc bằng dấu đóng ngoặc nhọc (}).

Bên trong một khối lệnh có thể chứa một hay nhiều lệnh hoặc chứa các khối lệnh khác.

{ // khối 1

{ // khối 2

lệnh 2.1

lệnh 2.2 …

} // kết thúc khối lệnh 2

lệnh 1.1

ệnh 1.2 …

} // kết thúc khối lệnh 1

{ // bắt đầu khối lệnh 3

// Các lệnh thuộc khối lệnh 3 // …

} // kết thúc thối lệnh 3

* 1. **. Toán tử và biểu thức**

**Toán tủ số học:**

|  |  |
| --- | --- |
| **Toán tử** | **Ý nghĩa** |
| **+** | **Cộng** |
| **\_** | **Trừ** |
| **\*** | **Nhan** |
| **/** | **Chia nguyên** |
| **%** | **Chia dư** |
| **++** | **Tăng 1** |
| **--** | **Giảm 1** |

**Toán tử trên bit**

|  |  |
| --- | --- |
| **Toán tử** | **Ý nghĩa** |
| **&** | **AND** |
| **|** | **OR** |
| **^** | **XOR** |
| **<<** | **Dịch trái** |
| **>>** | **Dịch phải** |
| **>>>** | **Dịch phải và điền 0 và bit trống** |
| **~** | **Bù bit** |

**Toán tủ quan hệ và logic**

Toán tử Ý nghĩa == So sánh bằng != So sánh khác > So sánh lớn hơn < So sánh nhỏ hơn >= So sánh lớn hơn hay bằng <= So sánh nhỏ hơn hay bằng

|  |  |
| --- | --- |
| **Toán tử** | **Ý nghĩa** |
| **==** | **Bằng** |
| **!=** | **Khác** |
| **>** | **Lớn hơn** |
| **<** | **Nhỏ hơn** |
| **>=** | **Lớn hơn hay bằng** |
| **<=** | **Nhỏ hơn hay bằng** |
| **&&** | **AND** |
| **||** | **OR** |
| **!** | **NOT** |

**Toán tử ép kiểu**

- Ép kiểu rộng (widening conversion): từ kiểu nhỏ sang kiểu lớn (không mất mát thông tin)

- Ép kiểu hẹp (narrow conversion): từ kiểu lớn sang kiểu nhỏ (có khả năng mất mát thông tin)

<tên biến>= (kiểu\_dữ\_liệu) <tên biến> ;

**Toán tử diều kiện**

**Cú pháp:**

<điều\_kiện> ? <biểu\_thức\_1> : <biểu\_thức\_2>

Nếu diều kiện đúng thì có giá trị hay thực hiện <biểu\_thức\_1> ngược lại là <biểu\_thức\_2>.

<điều kiện> : là một biểu thức logic

<biểu thức 1>, <biểu thức 2>: có thể là 2 giá trị, 2 biểu thức hay 2 hành động.

**2.6. cấu trúc điều khiển**

**Cấu trúc điều kiện if … else**

Dạng 1: if(<điều kiện>)

{ <khối lệnh>

}

Dạng 2:

if(<điều kiện>)

{ <khối lệnh 1>;

}

Else

{ <khối lệnh 2>;

}

**Cấu trúc switch … case**

Switch ( <biến>)

{

Case <giá trị 1>:

<khối lệnh 1>;

Break;

….

Case <giá trị n>:

<khối lệnh n>;

Break;

Default :

<khối lệnh default >;

}

**Cấu trúc lặp:**

**Dạng 1**: while(…)

while (điều\_kiện\_lặp)

{ khối \_lệnh; }

**Dạng 2**: do { … } while;

do { khối\_lệnh;

} while (điều\_kiện);

**Dạng 3:** for (…)

for (khởi\_tạo\_biến\_đếm;đk\_lặp;tăng\_biến) {

<khối lệnh> ;

}

**Cấu trúc lệnh nhảy:**

**Lệnh break:** trong cấu trúc switch chúng ta dùng câu lệnh break để thoát thỏi cấu trúc switch trong cùng chứa nó. Tương tự như vậy, trong cấu trúc lặp, câu lệnh break dùng để thóat khỏi cấu trúc lặp trong cùng chứa nó.

**Lệnh continue:** dùng để tiếp tục vòng lặp trong cùng chứa nó (ngược với break).

**Nhãn (label):** Không giống như C/C++, Java không hỗ trợ lệnh goto để nhảy đến 1 vị trí nào đó của chương trình. Java dùng kết hợp nhãn (label) với từ khóa break và continue để thay thế cho lệnh goto.

### 2.7. kiểu dữ liệu mảng

**Java có 2 kiểu dữ liệu:**

* Kiểu dữ liệu cơ sở
* Kiểu dữ liệu tham chiếu hay dẫn xuất. thường có kiểu :
  + Kiểu mảng
  + Kiểu lớp
  + Kiểu giao tiếp

Mảng là tập hợp nhiều phần tử có cùng tên, cùng kiểu dữ liệu và mỗi phần tử trong mảng được truy xuất thông qua chỉ số của nó trong mảng.

**Khai báo mảng:**

<kiểu dữ liệu > <tên mảng>[];

Hoặc <kiểu dữ liệu> [] <tên mảng>;

**Cấp phát bộ nhớ cho mảng:**

Để cấp phát bộ nhớ cho mảng trong Java ta cần dùng từ khóa new. (Tất cả trong Java đều thông qua các đối tượng). Chẳng hạn để cấp phát vùng nhớ cho mảng trong Java ta làm như sau:

int arrInt = new int[100];

**khởi tạo mảng:** Chúng ta có thể khởi tạo giá trị ban đầu cho các phần tử của mảng khi nó được khai báo.

Ví dụ: int arrInt[] = {1, 2, 3};

char arrChar[] = {‘a’, ‘b’, ‘c’};

String arrStrng[] = {“ABC”, “EFG”, ‘GHI’};

**Truy cập mảng:**

Chỉ số mảng trong Java bắt đầu tư 0. Vì vậy phần tử đầu tiên có chỉ số là 0, và phần tử thứ n có chỉ số là n-1. Các phần tử của mảng được truy xuất thông qua chỉ số của nó đặt giữa cặp dấu ngoặc vuông ([]).

Ví dụ: int arrInt[] = {1, 2, 3};

int x = arrInt[0]; // x sẽ có giá trị là 1.

int y = arrInt[1]; // y sẽ có giá trị là 2.

int z = arrInt[2]; // z sẽ có giá trị là 3.

## Hướng đối tượng trong java

### Class (lớp)

Chúng ta có thể xem lớp như một khuôn mẫu (template) của đối tượng (Object). Trong đó bao gồm dữ liệu của đối tượng (fields hay properties) và các phương thức(methods) tác động lên thành phần dữ liệu đó gọi là các phương thức của lớp.

Các đối tượng được xây dựng bởi các lớp nên được gọi là các thể hiện của lớp (class instance).

**Khai báo / định nghĩa 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 chúng ta đặt cho lớp

field\_1, field\_2: các thuộc tính, các biến, hay các thành phần dữ liệu của lớp.

constructor: là sự xây dựng, khởi tạo đối tượng lớp.

method\_1, method\_2: là các phương thức/hàm thể hiện các thao tác xử lý, tác động lên các thành phần dữ liệu của lớp.

**Tạo đối tượng của lớp**

ClassName objectName = new ClassName();

**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> field 1;**

**//…**

**}**

Để xác định quyền truy xuất của các đối tượng khác đối với vùng dữ liệu của 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 1 lớp khác. ·

protected: vùng protected của 1 lớp chỉ cho phép bản thân lớp đó và những lớp dẫn xuất từ lớp đó truy cập đến.

**phương thức 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: 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. 51 ·
* 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 ớ 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.

<kiểu trả vê> : 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

**Khởi tạo một đối tượng:**

Contructor 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 dùng để khởi gán những giá trị măc định. Các constructor không có giá trị trả về, và có thể có tham số hoặc không có tham số.

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

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, …)

### Đặc điểm hướng đối tượng trong java

**Đóng gói (encapsulation):**

Cơ chế đóng gói trong lập trình hướng đối tượng giúp cho các đối tượng dấu đi một phần các chi tiết cài đặt, cũng như phần dữ liệu cục bộ của nó, và chỉ công bố ra ngoài những gì cần công bố để trao đổi với các đối tượng khác. Hay chúng ta có thể nói đối tượng là một thành tố hỗ trợ tính đóng gói.

Đơn vị đóng gói cơ bản của ngôn ngữ java là class. Một class định nghĩa hình thức của một đối tượng. Một class định rõ những thành phần dữ liệu và các đoạn mã cài đặt các thao tác xử lý trên các đối tượng dữ liệu đó. Java dùng class để xây dựng những đối tượng. Những đối tượng là những thể hiện (instances) của một class.

Một lớp bao gồm thành phần dữ liệu và thành phần xử lý. Thành phần dữ liệu của một lớp thường bao gồm các biến thành viên và các biến thể hiện của lớp. Thành phần xử lý là các thao tác trên các thành phần dữ liệu, thường trong java người gọi là phương thức. Phương thức là một thuật ngữ hướng đối tượng trong java, trong C/C++ người ta thường dùng thuật ngữ là hàm.

**Tính đa hình (polymorphism):**

Tính đa hình cho phép cài đặt các lớp dẫn xuất khác nhau từ một lớp nguồn. Một đối tượng có thể có nhiều kiểu khác nhau gọi là tính đa hình.

**Tính kế thừa (inheritance):**

Một lớp con (subclass) có thể kế thừa tất cả những vùng dữ liệu và phương thức của một lớp khác (siêu lớp - superclass). Như vậy việc tạo một lớp mới từ một lớp đã biết sao cho các thành phần (fields và methods) của lớp cũ cũng sẽ thành các thành phần (fields và methods) của lớp mới. Khi đó ta gọi lớp mới là lớp dẫn xuất (derived class) từ lớp cũ (superclass). Có thể lớp cũ cũng là lớp được dẫn xuất từ một lớp nào đấy, nhưng đối với lớp mới vừa tạo thì lớp cũ đó là một lớp siêu lớp trực tiếp (immediate supperclass).

Dùng từ khóa extends để chỉ lớp dẫn xuất.

class A extends B

{

// …

}

### Gói (packages)

Việc đóng gói các lớp lại tạo thành một thư viện dùng chung gọi là package.

Một package có thể chứa một hay nhiều lớp bên trong, đồng thời cũng có thể chứa một package khác bên trong.

Để khai báo một lớp thuộc một gói nào đấy ta phải dùng từ khóa package.

Dòng khai báo gói phải là dòng đầu tiên trong tập tin khai báo lớp.

Các tập tin khai báo lớp trong cùng một gói phải được lưu trong cùng một thư mục.