Cú pháp Java cơ bản

Chúng ta có thể coi chương trình Java như một tập hợp các đối tượng mà có thể trao đổi lẫn nhau dùng các phương thức. Dưới đây chúng ta sẽ nêu ra một số định nghĩa cơ bản của lớp, đối tượng, phương thức cũng như biến trong Java:

- **Đối tượng -** Đối tượng có các trạng thái và hành vi. Ví dụ: Một con chó có các trạng thái của màu da, tên tuổi, thức ăn cũng như các hành vi như sủa, ăn, vẫy đuôi.
- **Lớp -** Một lớp có thể được định nghĩa như một bản thiết kế, mẫu mà có thể mô tả các trạng thái, hành vi của một đối tượng mà nó hỗ trợ.
- **Phương thức -** Một phương thức đơn giản là một hành vi. Một lớp có thể bao gồm nhiều phương thức. Trong mỗi phương thức có những phép toán logic, dữ liệu được xử lý và tất cả các hành động được thực thi.
- **Biến -** Mỗi đối tượng có một tập các biến duy nhất. Mỗi trạng thái của đối tượng được khởi tạo bởi các giá trị và gán với những biến.

Cú pháp cơ bản trong Java

Về chương trình Java, khi bạn đặt tên cho bất cứ thành phần nào, bạn cần tuân theo qui ước đặt tên của chúng. Qui ước đặt tên trong Java là một qui tắc bạn cần theo khi quyết định đặt tên nào cho định danh (Identifier) của mình, chẳng hạn như đặt tên cho lớp, package, biến, hằng, phương thức, … Nhưng nó không bắt buộc để bạn phải theo. Vì thế nó được gọi là qui ước chứ không phải là qui tắc

- Java là chương trình phân biệt chữ hoa chữ thường, điều đó có nghĩa là **VIETJACK**và **vietjack** mang những ý nghĩa khác nhau trong Java.
- **Tên Class -** Tất cả các tên Class trong Java nên viết hoa chữ cái đầu tiên và là một danh từ.

  Nếu không viết hoa, các trình IDE sẽ cảnh báo bạn (tất nhiên là Java vẫn chấp nhận nếu cố tình viết thường chữ cái đầu tiên).

  Ví dụ *class System*

- **Tên Interface -** Nên bắt đầu với chữ hoa và là một tính từ, ví dụ: Runnable, ActionListener ...
- **Tên phương thức -** Tất cả các tên phương thức nên bắt đầu với chữ thường và là một động từ.

  Ví dụ *public void actionPerformed()*
- **Tên biến -** Nên bắt đầu với chữ thường, ví dụ: firstName, orderNumber ...
- **Tên package -** Nên bắt đầu với chữ thường, ví dụ: java, lang, sql, util ...
- **Tên hằng -** Nên bắt đầu với chữ hoa, ví dụ: RED, YELLOW, MAX_PRIORITY ...
- **Tên file chương trình -** Tên file nên giống hệt tên class.

  Khi bạn lưu file, bạn nên sử dụng tên class và thêm hậu tố `.java`.

  Ví dụ: với tên class `MyfirstJavaProgram`, bạn nên lưu file dưới tên`*MyFirstJavaProgram.java*`
- **public static void main(String args[]) -** Chương trình Java bắt đầu bởi phương thức main() cho tất cả các chương trình J2SE.

CamelCase trong Qui ước đặt tên Java

Java theo cú pháp camelcase để đặt tên cho lớp, Interface, phương thức và biến. Nếu tên là tổ hợp của hai từ, thì từ thứ hai sẽ luôn bắt đầu với chữ hoa, ví dụ: actionPerformed(), firstName, ActionEvent, …

Định danh (Identifier) trong Java:

Tất cả các thành phần của Java đều yêu cầu tên. Tên được sử với các class, biến cũng như phương thức được gọi là Định danh (Identifier).

Trong Java, có vài điểm quan trọng dưới đây bạn phải ghi nhớ với Định danh (Identifier):

- Tất cả các idenfier nên bắt đầu với một chữ cái (A tới Z hoặc a tới z), ký tự ($) hoặc ký tự gạch dưới (_).
- Sau kí tự đầu tiên có thể là bất kỳ ký tự nào.

- Những key word trong Java không thể được sử dụng như một identifier.
- Các identifier phân biệt chữ hoa thường.
- Các trường hợp hợp lệ: tuoi, $ten, giatri, __1_giatri
- Các trường hợp không hợp lệ: 123abc, -hocphi

Modifier trong Java

Giống các ngôn ngữ khác, bạn có thể sửa đổi các lớp, phương thức, .., bởi sử dụng các Modifier. Trong Java, có hai loại Modifier:

- **Access Modifier:** Bao gồm default, public , protected, private
- **Non-access Modifier:** Bao gồm final, abstract, strictfp

Bạn sẽ hiểu rõ về các Modifier này trong các bài hướng dẫn tiếp theo.

Biến trong Java

Các loại biến trong Java như sau:

- Biến Local
- Biến của class (biến static)
- Biến đối tượng (không phải biến static)

Mảng trong Java

Mảng là đối tượng lưu trữ nhiều biến với chung một kiểu dữ liệu. Mặc dù vậy, một mảng bản thân nó cũng là một đối tượng trong bộ nhớ. Chúng ta cũng sẽ tìm hiểu các khởi tạo, khai báo đối tượng này trong các chương sắp tới.

Java Enums

Enums được giới thiệu bởi Java 5.0 Enums giới hạn số lượng các biến bởi cách định nghĩa trước. Các biến trong danh sách được liệt kê gọi là enums.

Với việc sử dụng enum sẽ có thể hạn chế số lượng các lỗi trong code.

Ví dụ, chúng ta giả sử một ứng dụng cho cửa hàng nước ép hoa quả, có thể giới hạn các kiểu cỡ cốc gồm có cỡ nhỏ, vừa và lớn. Điều này có thể đảm bảo giúp để mọi người khác không thể thêm các cỡ khác ngoài nhỏ, vừa, lớn.

Ví dụ:

```
class FreshJuice {

   enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
   FreshJuiceSize size;
}

public class FreshJuiceTest {

   public static void main(String args[]){
      FreshJuice juice = new FreshJuice();
      juice.size = FreshJuice. FreshJuiceSize.MEDIUM ;
      System.out.println("Size: " + juice.size);
   }
}
```

Chương trình sẽ in ra kết quả:

```
Size: MEDIUM
```

Ghi chú: Các Enums có thể được khai báo như là của riêng chính nó hoặc bên trong một lớp. Các phương thức, biến, constructor cũng có thể được định nghĩa bên trong các Enum.

Các từ khóa trong Java:

Danh sách dưới đây in ra những từ khóa được dành riêng trong Java. Những từ khóa dành riêng này không được sử dụng như một tên biến hoặc tên identifier.

abstract	assert	boolean	break
byte	case	catch	char

class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

Comment trong Java

Java hỗ trợ việc comment trên 1 dòng lệnh hoặc nhiều dòng lệnh tương tự như C và C++. Tất cả các ký tự trong các dòng comment đều được bỏ qua bởi Java compiler.

```
public class MyFirstJavaProgram{

   /* Chuong trinh in Hello World.
    * Day la vi du ve comment gom nhieu dong.
    */
```

```
    public static void main(String []args){
       // Vi du thu nhat ve comment co mot dong.
       /* Vi du thu hai ve comment co nhieu dong. */
       System.out.println("Hello World");
    }
}
```

Sử dụng dòng trống trong Java

Một dòng chỉ chứa khoảng trắng, có thể với một comment, được biết như là dòng trống, và Java hoàn toàn bỏ qua nó.

Tính kế thừa trong Java

Trong Java, các Class có thể được suy ra từ các Class khác. Về cơ bản, nếu bạn cần tạo một Class mới và ở đây đã là một Class mà có một số đoạn code bạn cần, thì khi đó nó là có thể để suy ra một class mới từ code đã tồn tại.

Khái niệm này cho bạn khả năng tái sử dụng các trường và các phương thức của class đang tồn tại mà không cần phải viết lại code trong class mới. Trong tình huống này, class đang tồn tại được gọi là superclass và class được suy ra được gọi là subclass.

Interface trong Java

Trong ngôn ngữ Java, một interface có thể được định nghĩa như là một contract giữa các đối tượng về cách giao tiếp với nhau. Các interface đóng vai trò thiết yếu khi nó đi với khái niệm về tính kế thừa.

Một interface định nghĩa các phương thức, các subclass nên sử dụng. Nhưng sự thực thi của các phương thức lại hoàn toàn là do các subclass.