**Lambda Expression trong java 8**

1. **Biểu thức lambda là gì ?**

Lambda Expression(Biểu thức lambda) là một tính năng được giới thiệu trong java 8. Ta có thể hiểu biểu thức lambda là một hàm ẩn danh, nó không thuộc bất kì một class nào, nó giúp cho việc viết mã trở nên ngắn gọn hơn rất nhiều.

1. **Cú pháp của biểu thức lambda**

* Một biểu thức lambda là một hàm không tên, không có phạm vi truy cập, không có kiểu trả về

Biểu thức lambda gồm 3 thành phần sau:

**(argument-list) -> (body)**

* Argument-list: Có thể không có, có một hoặc nhiều tham số
* Arrow- token: là dấu “->” được dùng để liên kết danh sách tham số và phần thân
* body: chứa các câu lệnh và biểu thức cho biểu thức lambda.

Chúng ta có thể viết biểu thức lamda bằng nhiều cách khác nhau:

* Kiểu dữ liệu của tham số:

Việc khai bao kiểu dữ liệu cho tham số là không bắt buộc. Trình biên dịch sẽ tự suy luận ra kiểu dữ liệu từ giá trị của các tham số.

* Sử dụng cặp dấu ():

Đối với biểu thức lambda chỉ truyền vào một tham số duy nhất, ta có thể bỏ qua cặp dấu ngoặc nhọn.

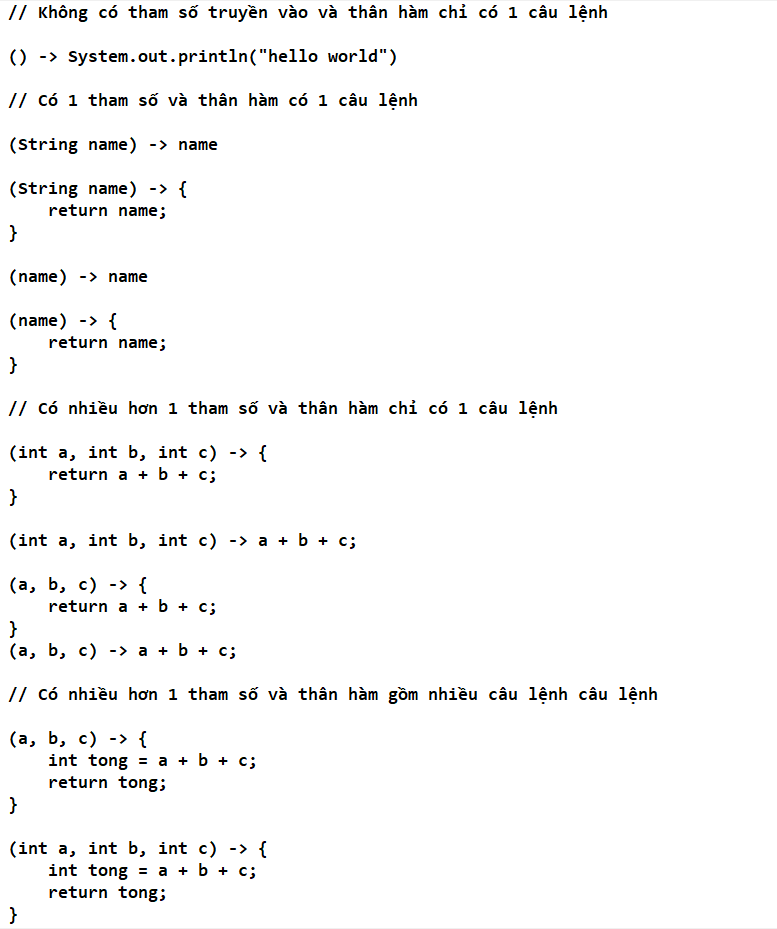
Đối với biểu thức lambda không có tham số hoặc có nhiều hơn một tham số việc sử dụng cặp dấu () là bắt buộc.

* Sử dụng cặp dấu {};

Trong trường hợp phần thân của biểu thức lambda chỉ có một câu lệnh duy nhất ta có thể không cần sử dụng đến cặp dấu {};

* Sử dụng return:

Nếu phần thân của biểu thức lambda chỉ có duy nhất 1 câu lệnh ta có thể không cần sử dụng từ khóa return.

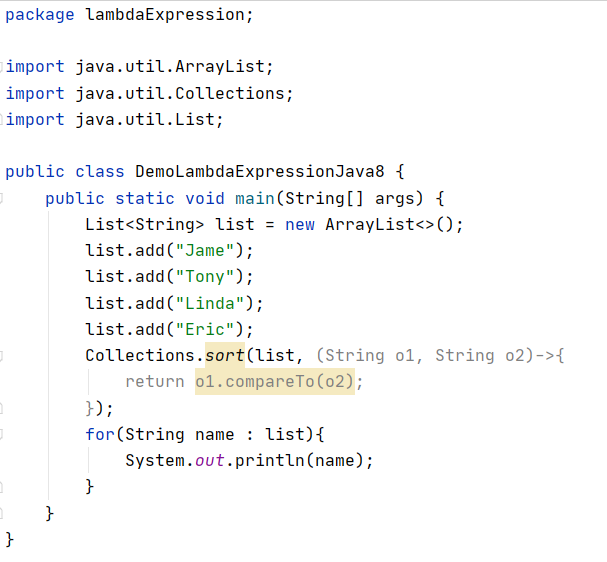


1. **Ví dụ sử dụng biểu thức Lambda**

**Ví dụ về sắp xếp một danh sách trước java 8**



**Sử dụng Lambda của Java 8 ta có thể viết lại như sau:**



Như các bạn có thể thấy việc sử dụng biểu thức lambda giúp code của chúng ta ngắn gọn hơn rất nhiều.

Trong một khối lệnh nếu chỉ có một lệnh duy nhất, ta có thể bỏ đi { }, khi đó đoạn code trên bạn có thể viết như sau:



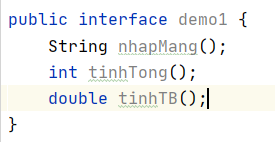
Trình biên dịch (Compiler) của Java có thể biết được kiểu phần tử trong tập hợp cần sắp xếp là kiểu gì, trong ví dụ này là kiểu String, và vì vậy bộ so sánh ( Comparator) chắc chắn là so sánh các kiểu dữ liệu String. Đoạn code trên có thể viết ngắn gọn hơn như sau:



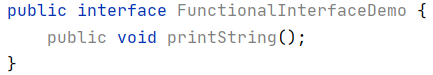
1. **Functional Interface**

Để có thể hiểu và áp dụng được biểu thức lambda. Chúng ta sẽ tìm hiểu qua một chút về Functional Interface.

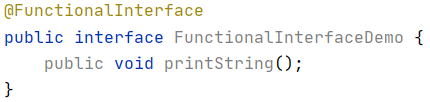
* Như chúng ta đã biết interface bao gồm các method(phương thức) abstract (trừu tượng) có nghĩa là nó không có thân.

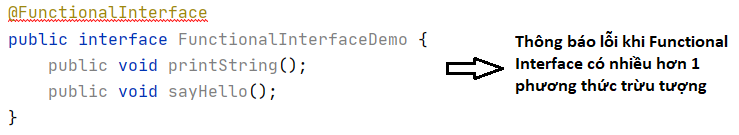


* Functional Interface là interface mà nó chỉ có duy nhất một method abstract.
* Phương thức abstract trong functional interface được gọi là functional method hoặc single abstract method(SAM).



* Để có thể dễ nhận biết được đâu là một functional interface java 8 cung cấp cho ta một annotation là @FunctionalInterface.
* Việc sử dụng annotation này là không bắt buộc nhưng nó sẽ giúp chúng ta dễ nhận biết một functional interface và trình biên dịch sẽ bắt lỗi nếu có nhiều hơn một abstract method được khai báo bên trong nó.





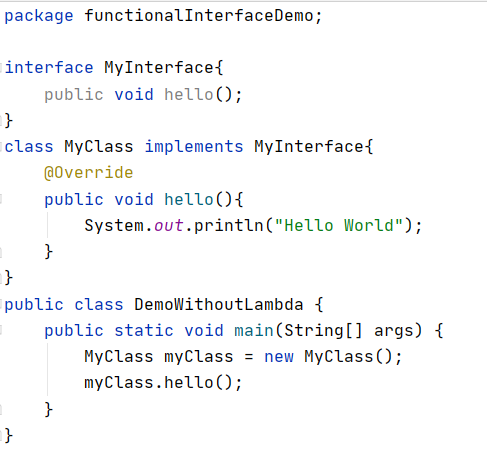
Một số interface trong java là functional interface:

1. Runnable: nó chỉ chứa phương thức run()
2. Comparable: nó chỉ chứa phương thức compareTo()
3. ActionListener: Nó chỉ chứa phương thức actionPerformed()
4. Callable: Nó chỉ chứa phương thức call()

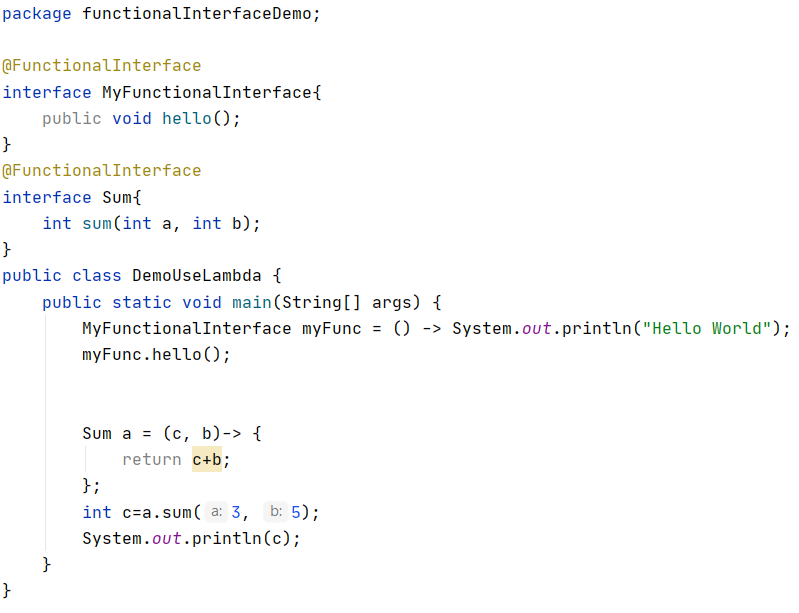
* Để sử dụng biểu thức lambda đầu tiên ta cần một functional interface

Ví dụ:

**Cách code trước Java 8**

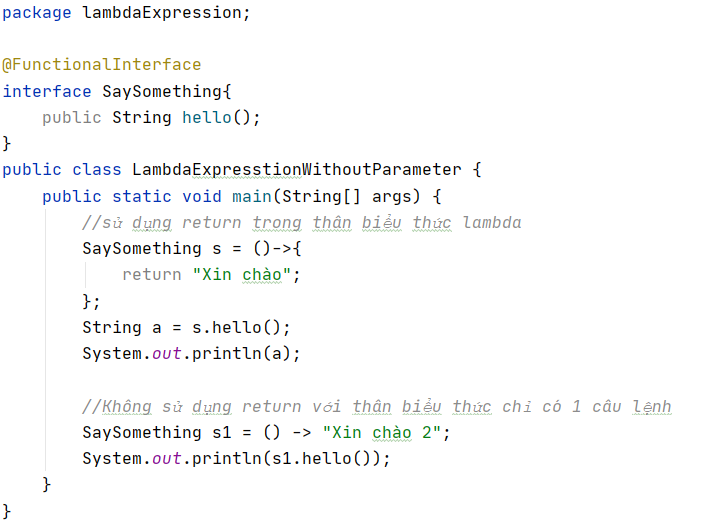


**Sử dụng biểu thức lambda**

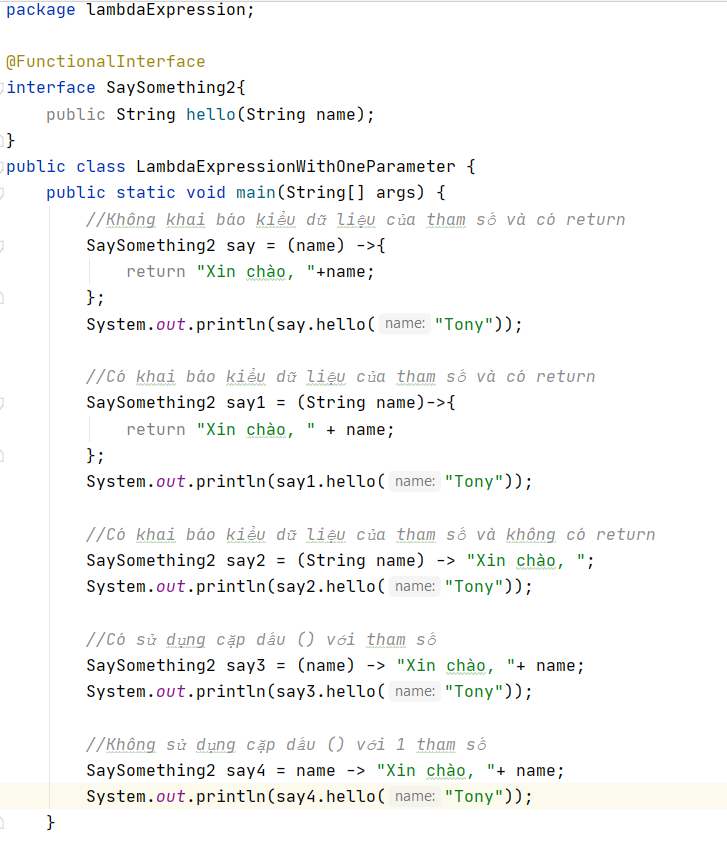


* Số lượng các tham số của biểu thức lambda sẽ phụ thuộc vào số lượng các tham số của phương thức trừu tượng của Functional Interface.

1. **Các ví dụ về biểu thức lambda**
   1. **Biểu thức lambda không tham số**



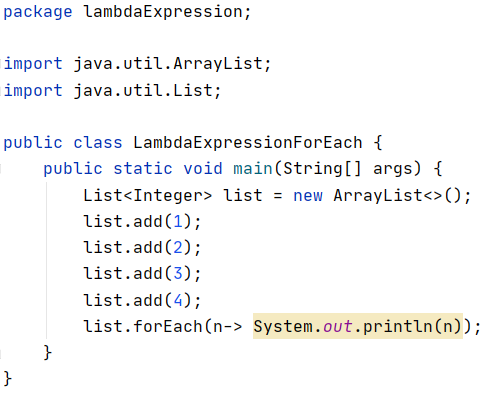
* 1. **Biểu thức lambda có 1 tham số**



* 1. **Biểu thức lambda có nhiều tham số**



* 1. **Biểu thức lambda với forEach()**



1. **Phạm vi truy cập biến trong biểu thức lambda**

Việc truy cập các biến phạm vi bên ngoài từ các biểu thức lambda rất giống với các đối tượng ẩn danh (anonymous objects). Bạn có thể truy cập bất kỳ biến final, static hoặc biến chỉ được gán một lần. Biểu thức Lambda throw một lỗi biên dịch, nếu một biến được gán một giá trị lần thứ hai.

* 1. **Truy cập biến local – Accessing local variables**

Chúng ta có thể truy cập các biến final và biến chỉ được gán một lần.



* 1. **Truy cập Accessing fields and static variables**

Chúng ta có thể truy cập và thay đổi các trường hoặc biến static

