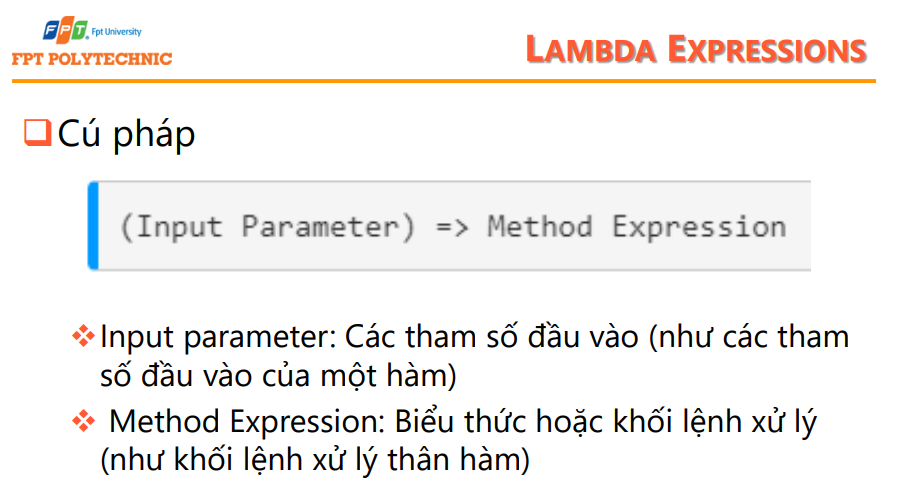
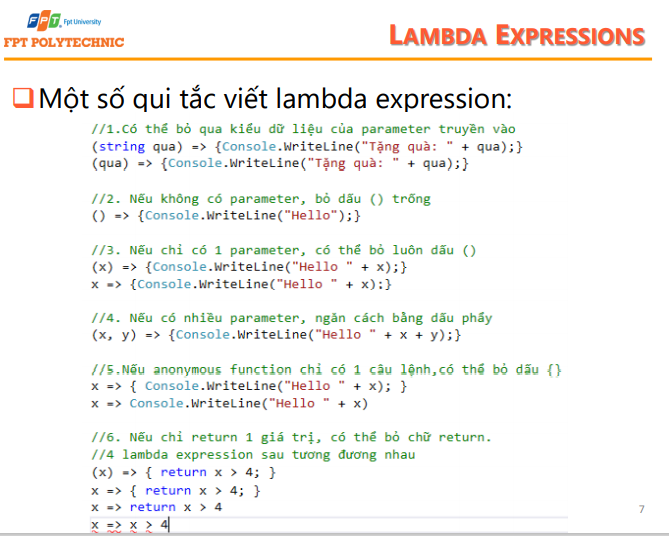
# Biểu thức Lambda - Một trong những tính năng hay nhất của Java 8





# 1. Lambda Expression là gì?

Lamda Expression là một hàm không có tên (unamed function) với các tham số (parameters) và nội dung thực thi (body). Nội dung thực thi của Lamda expression có thể là 1 khối lệnh hoặc 1 biểu thức. Dấu “->” tách biệt các tham số và nội dung thực thi.

Ví dụ:

(int x) -> x + 1 Có một tham số kiểu int và trả về giá trị tham số tăng lên 1

(int x, int y) -> x + y Có 2 tham số kiểu int và trả về tổng

(String msg) -> {System.out.println (msg);} Có một tham số String và in nó ra console.

msg -> System.out.println (msg) Có một tham số không cần xác định kiểu và in nó ra console. Nó giống hệt với đoạn code ở ví dụ trên.

() -> "Hi" Không có tham số và trả về một chuỗi.

(String str) -> str.length() Có một tham số String và trả về chiều dài của nó.

(int x, int y) -> {

int max = x > y ? x: y;

return max;

}

# Biểu thức Lambda(từ **Java 8) khi sắp xếp.**

Trước hết chúng ta xem lại cách **Java** phiên bản trước 8 sắp xếp một tập hợp có sẵn.

package org.o7planning.tutorial.j8.lambda;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class SortBefore8Example {

    public static void main(String[] args) {

        // Một danh sách các loại quả.

        List<String> fruits = Arrays.asList("Grapefruit", "Apple", "Durian", "Cherry");

        // Sử dụng phương thức tiện ích của Collections

        // để sắp xếp lại danh sách trên.

        // Cung cấp một Comparator (Bộ so sách).

        Collections.sort(fruits, new Comparator<String>() {

            @Override

            public int compare(String o1, String o2) {

                return o1.compareTo(o2);

            }

        });

        for (String fruit : fruits) {

            System.out.println(fruit);

        }

    }

}

# Dùng Biểu thức Lambda:

package org.o7planning.tutorial.j8.lambda;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class SortJava8Example {

    public static void main(String[] args) {

        // Một danh sách các loại quả.

        List<String> fruits = Arrays.asList("Grapefruit", "Apple", "Durian", "Cherry");

        // Sử dụng phương thức tiện ích của Collections để sắp xếp lại danh sách trên.

        // Cung cấp một Comparator (bộ so sánh) cho tham số thứ 2 của phương thức.

        // Vì Comparator là interface có duy nhất 1 phương thức.

        // ==> Có thể viết ngắn gọn với biểu thức Lambda.

        // Không cần viết tên interface,

        // Không cần viết tên phương thức.

**Collections.sort(fruits, (String o1, String o2) -> {**

**return o1.compareTo(o2);**

**});**

        for (String fruit : fruits) {

            System.out.println(fruit);

        }

    }

}

**Trong một khối lệnh nếu chỉ có một lệnh duy nhất, bạn có thể bỏ đi { }, khi đó đoạn code trên bạn có thể viết ngắn hơn:**

Collections.sort(fruits, (String o1, String o2) -> o1.compareTo(o2)  );

# Dùng Biểu thức Lambda khi sắp đối tượng:

## a. Sắp theo chuỗi

**//Y7: Sắp theo họ tên – phương thức cũ**

public void sapTheoHoTen() {

Comparator<NhanVien> com = new Comparator<NhanVien>() {

@Override

public int compare(NhanVien o1, NhanVien o2) {

return o1.getHoTen().compareTo(o2.getHoTen());

}

};

listNV.sort(com);

System.out.println("DSNV sắp theo họ tên là:");

xuatDSNV();

}

**//Y7: Dùng biểu thức lambda:**

public void sapTheoHoTen\_Lambda() {

Collections.sort(listNV, (NhanVien o1, NhanVien o2)

-> {

return o1.getHoTen().compareToIgnoreCase(o2.getHoTen());

});

System.out.println("DSNV sắp theo họ tên là:");

xuatDSNV();

}

## b. Sắp theo số:

**//Y8: Sắp theo thu nhập- phương thức cũ:**

public void sapTheoThuNhapGiamDan() {

Comparator<NhanVien> com = new Comparator<NhanVien>() {

@Override

public int compare(NhanVien o1, NhanVien o2) {

Double d1 = o1.getThuNhap();

Double d2 = o2.getThuNhap();

return d2.compareTo(d1);

}

};

listNV.sort(com);

}

**//Y8: Sắp theo thu nhập - Dùng biểu thức lambda:**

**//do thu nhập là double nên phải chuyển thành Double**

public void sapTheoThuNhapGiamDan\_Lambda() {

Collections.sort(listNV, (NhanVien o1, NhanVien o2) -> {

Double d1 = o1.getThuNhap();

Double d2 = o2.getThuNhap();

return d2.compareTo(d1);

});

}

Chúng ta có thể thấy biểu thức Lambda được đưa vào như là một tham số của phương thức addMessageListener().  
Ở đây, kí hiệu “->”chính là biểu thức Lambda mà chúng ta đang nói tới.

Đến đây phần nào các bạn cũng đã hiểu biểu thức Lamda là gì và được sử dụng như thế nào.

Tiếp theo chúng ta sẽ tìm hiểu về các truyền tham số vào biểu thức Lambda hay còn có thể hiểu là  c**ác tham số của biểu thức Lambda**

Cũng như phương thức, biểu thức Lambda có thể truyền vào bao nhiêu tham số tùy ý.

Các tham số của biểu thức Lambda cũng tương tự như Single Method Interface mà nó thay thế cho.

Biểu thức Lambda có thể không có tham số, một tham số hoặc nhiều tham số.

Trường hợp không có tham số:  
https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-4(1).png

Chú ý rằng, cặp ngoặc đơn () để thể hiện biểu thức Lambda không nhận tham số nào cả.

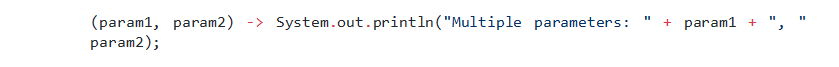
Trường hợp có một tham số.

https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-5.png

Tham số sẽ được đặt trong cặp ngoặc đơn ().

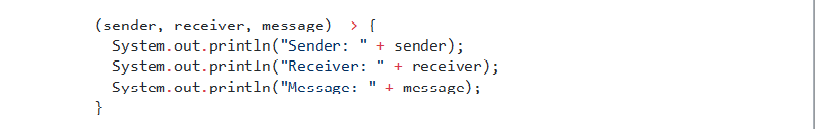
Đối với trường hợp một tham số chúng ta cũng có thể lược bỏ đi cặp ngoặc đơn này.Tham số sẽ được đặt trong cặp ngoặc đơn ().  
https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-6.png

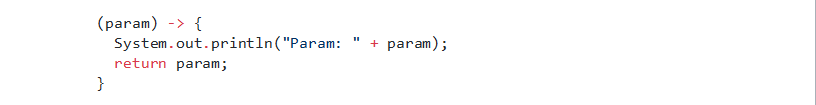
Trường hợp có nhiều tham số:

Trong trường hợp biểu thức Lambda nhận nhiều tham số, chúng ta phải đặt hết tham số trong cặp ngoặc đơn (). Chỉ trường hợp một tham số mới có thể lược bỏ cặp ngoặc đơn này.  
  
Ngoài ra tham số truyền vào biểu thức Lambda cũng có thể khai báo kiểu dữ liệu như sau:  
Viêc khai báo kiểu dữ liệu là cân thiết trong trường hợp trình biên dịch không thể tìm thấy giao diện chức năng nào phù hợp. Đừng lo lắng vì bạn không biết khi nào nên khai báo, khi nào không, bởi vì trình biên dịch sẽ nhắc nó cho bạn.  
Cách khai báo kiểu dữ liệu đơn giản như sau:  
https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-8.png

Nhìn các đoạn code ở trên chắc hẳn các bạn đều biết phần thân hàm của biểu thức Lambda nằm phía bên phải đấu “->”.Giờ chúng ta đã biết về cách dùng biểu thức Lambda cũng như về tham số truyền vào biểu thức Lambda. Vậy nếu giao diện chức năng phải thực hiện nhiều lệnh thì ta phải làm thế nào, chúng ta cùng tìm hiểu về phần thân hàm, **Lambda Function Body.**

Nhìn các đoạn code ở trên chắc hẳn các bạn đều biết phần thân hàm của biểu thức Lambda nằm phía bên phải đấu “->”.  
https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-9.pngNếu sử dụng nhiều lệnh trong phần thân hàm của biểu thức Lambda Expression, đơn giản chúng ta chỉ cần viết chúng trong cặp ngoặc nhọn {}.

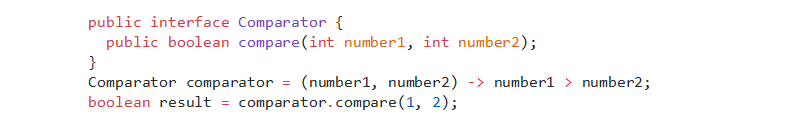
Ắt hẳn các bạn đang thắc mắc, nãy giờ Lambda Expression trả về **void,**vậy thì phải làm thế nào nếu muốn biểu thức Lambda trả về một giá trị nào đó. Đơn giản thôi, bạn có thể trả về giá trị trong thân hàm của biểu thức Lambda bằng lệnh **return**như bình thường.

Trong trường hợp này.

https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-12.pngBạn có thể viết ngắn gọn hơn bằng cách.

https://bachkhoa-aptech.edu.vn/upload/image/java-bieu-thuc-lambda-13.png

Ngoài ra biểu thức Lambda cũng được sử dụng như sau.



Chúng ta có thể hiểu, đoạn code đầu tiên ta khai báo giao diện chức năng. Đoạn code thứ 2, ta khai báo biểu thức Lambda. Đoạn cuối cùng chúng ta thấy biểu thức Lambda được gọi như thế nào.