Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

LTV CÔNG NGHỆ JAVA

Module 2 – Bài 1: ***Biểu thức Lambda***

Phòng LT – Mạng

http://csc.edu.vn/

2017

Nội dung

1. **Giới thiệu**

2. **Biểu thức Lambda**

Giới thiệu

- Java 8 đã được Oracle ra mắt vào ngày 25/3/2014 với rất nhiều cải tiến. Trong đó đáng chú ý là Java đã chính thức hỗ trợ biểu thức Lambda.
- Biểu thức Lambda là một trong những tính năng mới quan trọng trong Java 8.
- Biểu thức Lambda được sử dụng để viết viết code ngắn gọn hơn cho interface có chứa phương thức, gọi là giao diện chức năng (functional interface).
- Biểu thức Lambda cũng được sử dụng để lặp một mảng/ một bộ với mã rất đơn giản.
- Biểu thức Lambda định nghĩa phương thức giao diện chức năng và trả về thể hiện của giao diện này.

Nội dung

1. Giới thiệu

2. **Biểu thức Lambda**

Biểu thức Lambda

- Lambda là một phương thức không có tên (unamed function) với các tham số (parameters) và nội dung thực thi (body).
- Nội dung thực thi của Lambda expression có thể là 1 khối lệnh hoặc 1 biểu thức. Dấu "->" tách biệt các tham số và nội dung thực thi.
- **Cú pháp**

  public SomeType someMethod(args) { body }

  => Cú pháp theo biểu thức Lambda

  (args) -> body

Biểu thức Lambda

- Nó cho phép chúng ta xử lý chức năng (functionality) như một đối số của phương thức (thông qua function around), hoặc xử lý code như dữ liệu (data): những khái niệm mà các nhà phát triển chức năng rất quen thuộc.

- Trong dạng đơn giản nhất, lambda có thể được biểu diễn như một danh sách các dấu phẩy của các thông số.

Biểu thức Lambda

Ví dụ:

- Dạng 1: Phương thức không có tham số

```
() -> System.out.println("Your message");
```

- Dạng 2: Phương thức có tham số và có phần thân thực hiện công việc trả về kết quả.

```
(int a, int b) -> a+b; // giá trị a+b được trả
  về bởi phương thức

(String s) -> s + "Hello World";

list.forEach((Student s) ->
  System.out.println(s.getName()));
```

Biểu thức Lambda

- **Java có thể ngầm hiểu kiểu của tham số trong phương thức**
  - public SomeType someMethod(Type1var1, Type2 var2) {
    // method body
    }
  - ⇒(Type1var1, Type2 var2 …) -> { method body }
  - ⇒(var1, var2 …) -> { method body }
  - public SomeType someMethod(T1 var) {
    return(someValue);
    }
  - ⇒var -> someValue

Biểu thức Lambda

- Ví dụ

```
public interface Calculator {
    public int add(int a, int b);
}
public static void main(String[] args) {
   Calculator cal = (a, b) -> a + b;
       int res = cal.add(5, 6);
       System.out.println(res);
}
```

Biểu thức Lambda

Ví dụ

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);
for (Integer numer : numbers) {
        System.out.println(numer);
}
```

Một biểu thức Lambda sẽ có hai phần: tham số và hành vi cần thực hiện, chúng được cách nhau bởi một dấu ->

```
numbers.forEach(value -> System.out.println(value)); // biến value là một tham số còn System.out.println(value) là một hành vi (in dãy số)
```

Biểu thức Lambda

Ví dụ

```
Arrays.asList( "a", "b", "d" ).forEach( e ->
System.out.println( e ) ); // e được suy ra từ trình biên dịch compiler.

//Ta cũng có thể cung cấp kiểu đối số một cách tường minh, trong dấu ngoặc đơn.

Arrays.asList( "a", "b", "d" ).forEach( ( String e ) ->
System.out.println( e ) );


 Arrays.asList( "a", "b", "d" ).forEach( e -> {

    System.out.print( e );

 } );
```

Biểu thức Lambda

Ví dụ

- Lambda có thể reference tới thành viên của lớp và các biến cục bộ (chắc chắn rằng các biến này là kiểu final)

```
String separator = ",";
Arrays.asList( "a", "b", "d" ).forEach(
    ( String e ) -> System.out.print( e + separator )
);  // Hoặc
final String separator = ",";
Arrays.asList( "a", "b", "d" ).forEach(
    ( String e ) -> System.out.print( e + separator )
);
```

Biểu thức Lambda

Ví dụ

Lambda có thể trả về kết quả giá trị. Kiểu của return value sẽ được suy ra từ trình biên dịch compiler. Câu lệnh return không yêu cầu bắt buộc nếu lambda body chỉ chứa 1 dòng code.

```
Arrays.asList( "a", "b", "d" ).sort( ( e1, e2 ) ->
e1.compareTo( e2 ) );

//Tương đương

Arrays.asList( "a", "b", "d" ).sort( ( e1, e2 ) -> {
    int result = e1.compareTo( e2 );
    return result;
} );
```

Biểu thức Lambda

❑ Explicit và implicit lambda expression

- Một biểu thức lambda minh bạch (explicit): là biểu thức có khai báo kiểu của các tham số của nó một cách rõ ràng.
- Biểu thức lambda không minh bạch (implicit): Một biểu thức lambda không khai báo kiểu (type) của các tham số (parameter) của nó. Trình biên dịch sẽ suy ra các kiểu tham số đối với biểu thức lambda không minh bạch.

Biểu thức Lambda

Explicit và implicit lambda expression

Explicit	Implicit

public class Main {
    public static void main(String[] args) {
        MyIntegerCalculator myIntegerCalculator = (Integer s1) -> s1 * 2;
        System.out.println("Result: " + myIntegerCalculator.calcIt(5));
    }
}
interface MyIntegerCalculator {
    public Integer calcIt(Integer s1);
}
	public class Main {
	    public static void main(String[] args) {
	        MyIntegerCalculator myIntegerCalculator = (s1) -> s1 * 2;
	        System.out.println("Result: " + myIntegerCalculator.calcIt(5));
	    }
	}
	interface MyIntegerCalculator {
	    public Integer calcIt(Integer s1);
	}