**Java Lambda Expression Syntax**

To create a lambda expression, we specify input parameters (if there are any) on the left side of the lambda operator ->, and place the expression or block of statements on the right side of lambda operator. For example, the lambda expression (x, y) -> x + y specifies that lambda expression takes two arguments x and y and returns the sum of these.

//Syntax of lambda expression

(parameter\_list) -> {function\_body}

**Lambda expression vs method in Java**

A method (or function) in Java has these main parts:  
1. Name  
2. Parameter list  
3. Body  
4. return type.

A lambda expression in Java has these main parts:  
Lambda expression **only has body and parameter list**.  
1. **No** name – function is anonymous so we don’t care about the name  
2. Parameter list  
3. Body – This is the main part of the function.  
4. **No** return type – The java 8 compiler is able to infer the return type by checking the code. you need not to mention it explicitly.

**Where to use the Lambdas in Java**

To use lambda expression, you need to either create your own functional interface or use the pre defined functional interface provided by Java. An interface with **only single abstract method** is called functional interface(or Single Abstract method interface), for example: Runnable, callable, ActionListener etc.

**Example 1: Java Lambda Expression with no parameter**

@FunctionalInterface

interface MyFunctionalInterface {

//A method with no parameter

public String sayHello();

}

public class Example {

public static void main(String args[]) {

// lambda expression

MyFunctionalInterface msg = () -> {

return "Hello";

};

System.out.println(msg.sayHello());

}

}

Output:

Hello

**Example 2: Java Lambda Expression with single parameter**

@FunctionalInterface

interface MyFunctionalInterface {

//A method with single parameter

public int incrementByFive(int a);

}

public class Example {

public static void main(String args[]) {

// lambda expression with single parameter num

MyFunctionalInterface f = (num) -> num+5;

System.out.println(f.incrementByFive(22));

}

}

Output:

27

**Example 3: Java Lambda Expression with Multiple Parameters**

interface StringConcat {

public String sconcat(String a, String b);

}

public class Example {

public static void main(String args[]) {

// lambda expression with multiple arguments

StringConcat s = (str1, str2) -> str1 + str2;

System.out.println("Result: "+s.sconcat("Hello ", "World"));

}

}

Output:

Result: Hello World

**Example 4: Iterating collections using foreach loop**

import java.util.\*;

public class Example{

    public static void main(String[] args) {

      List<String> list=new ArrayList<String>();

       list.add("Rick");

list.add("Negan");

  list.add("Daryl");

list.add("Glenn");

list.add("Carl");

  list.forEach(

// lambda expression

    (names)->System.out.println(names)

);

}

}

STATEMENTS IN C:

1. TYPE DECLARATION

2.INPUT/OUTPUT

3. ASSIGNMENT

4. CONTROL STATEMENTS ---

JUMB - goto , break, continue, return , exit

BRANCH - if , switch

LOOP - for , while , do while

OPERATORS IN C :

* Arithmetic Operators. ...
* Relational Operators. ...
* Logical Operators. ...
* Bitwise Operators. ...
* Assignment Operators. ...
* Conditional Operators. ...
* Special Operators.

Data types in C:

1. **Primary(Built-in) Data Types**:  
   *void*, *int*, *char*, *double* and *float*.
2. **Derived Data Types**:  
   *Array*, *Structure*, *Union ,functions* and *Pointers*.
3. **User Defined Data Types**: typedef, and *Enumeration*.

JAVA Basics:

JVM, Byte Code, Data types , Operators, Statements , Keywords,

Calss , Object , methods (taking and returning objects), Overloading, constructors

Inheritance- public, private,protected,static and final – super , this – overriding –

Abstract class/method - .. Interface – Exception , Threading- Networking.

Core 3 Packages……………

Lang … wrapper classes

System,Object, Class, Thread, Exception, String….

Util ….. Collections

Date, Vector,Enumeration……

IO …. strams

Applets