Why Java8 comes in first place ?

* To make our code more consise and readable.

By introducing lambda expression, method references our code become consise and more readable.

* Introduced Functional Programming Features

Java is a object oriented programming language but to have functional programming benefits java introduced features like lambda expression, Streams API, method reference.

* Enable parallel programming

Java 8 enables parallel programming, through Streams API, enabling more efficient utilization of multi-core processor.

Features of Java8

1. Lambda expression
2. Stream API
3. Date and time API
4. Base64 Encode and decode
5. Method reference and constructor reference
6. Default, static methods in interface
7. Functional interface
8. Optional class

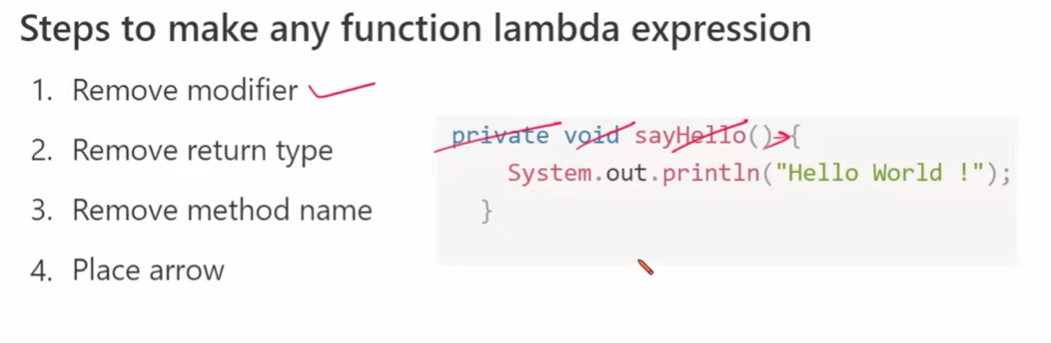
**Lambda Expression**

* It is an anonymous function

1. No name
2. No return type
3. No modifier

* lambda expression are used to implement functional interface.

(because functional interface has only one abstract method, so implementation given by lambda expression is of that abstract method only)



**Other Important Rules of Lambda Expression**

1. If the body has just one statement, then we can remove curly brackets also

(String str)-> return str.length();

2. If only one parameter remove small brackets

(str)-> str.length();

Converted to: str -> str.length();

3. Use type inference, means we can remove type of the parameters, (as lambda expression

is implementing the only abstract method present in interface so java compiler will know the

type by seeing the type of parameters in abstract method.

private void add(int a, int b){

System.out.println(a+b);

}

Converted to: (a, b) -> System.out.println(a+b);

4. No return keyword, if only return statement is there

private int getStringLength(String str){

return str.length();

}

Converted to: (str) -> str.length();

Benefits of lambda

* Reduces the lines of code
* Enables functional programming
* Facilitate Parallel execution
* Enables Streamlined Collection operations

**Functional Interface**

* If the interface has only one abstract method it is called functional interface.

Ex- Runnable ,Callable, Comparable etc

* Lambda expression is used to implement functional interface in very short manner.

How to implement functional interface using lambda expression see the code

**Stream API**

* Stream is an interface
* We can perform bulk operation on collection or group of objects(like array) using Stream.

How to get object of Stream interface(means its child class object) using Stream interface Static methods

**Predicate**

Predicate is a functional interface which has one abstract method called test…..

**boolean test(T t);**

which returns Boolean value true or false

Its implementation is given through lambda expression.

Why to use Predicate ?

If our function is returning boolean value, so instead of making our own class and function inside that class, use Predicate interface which is functional interface means, it has only abstract method called test….

And Functional interface can be easily implementated by using lambda expression, so give implementation to that abstract method, according to your problem.

So using Predicate interface, we have to write less code.

See Code in eclipse(PredicateFunctionalInterface)

**Method Reference**

What

Method Reference is used to refer an existing method without invoking them directly.(that why we write methodName parenthesis do not come)

Why

It helps us to avoid re-writing of function again as we can refer to existing method.

When to use

To give implementation of functional interface.