Java 8 New features:

1. Lambda Expressions
2. Functional Interface
3. Defaut methods and static methods
4. Predefined Functional Interfaces

Predicate

Function

Consumer

Supplier

Etc..

1. Double colon (::) Operator

Method reference

Constructor reference

1. Streams
2. Date and Time API
3. Optional Class
4. Nashron Javascript Engine
5. Etc…

**1.Lambda Expressions**: It is an anonymousfunction. Anonymous means which doesn’t have name i.e nameless and doesn’t have return type and modifier.

**How to write Lambda expression:**

Example 1:

Basically we write below method to print Hello

Public void m1()

{

System.out.println(“Hello”);

}

By using Lambda expression we can write in one line

() -> System.out.println(“Hello”);

Here -> is a special symbol to represent Lambda expression

If multiple lines are there then we should use { }

**Example 2:**

Public void m1(int a,intb)

{

Syso(a+b);

}

Lamda expression for above method:

(a,b) -> Syso(a+b);

**Example 3**:

Public int squareIt(int n)

{

Return n\*n;

}

Lamda Expression for above metod

(int n) -> n\*n;

If we use curly braces we should use return statement (n) -> { return n\*n ;};

We don’t need to specify return type also. Compiler will automatically identifies the return type.

If we have only one parameter then () is option. So the lambda expression is n-> n\*n;

Finally the lambda expression is n-> n\*n;

**Example 4:**

Public void m1(String s)

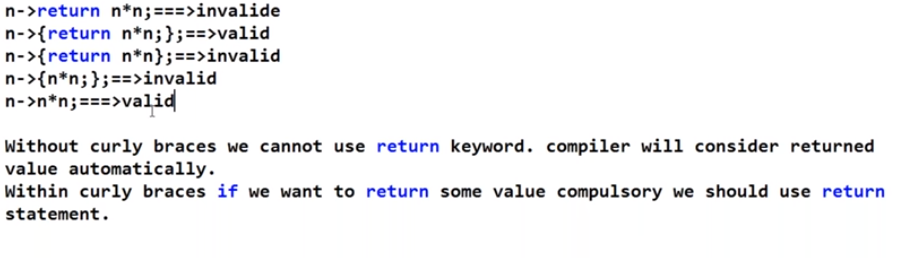
{

Return s.length();

}

Lambda expression is : s -> s.length();

**Conclusions for writing Lambda expressions**



**How to call Lambda expressions:** By using FI (Functional Interface)

**Functional Interface:** If an interface contains single abstract method then such type of interfaces are called functional interface.

**Use :** To invoke Lamda expressions

Below are functional interfaces which contains only one abstract method

Runnable 🡪 run()

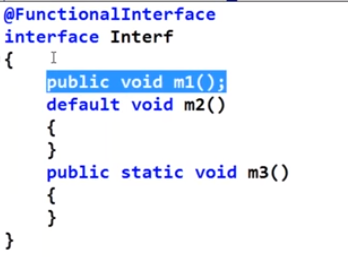
Comparable 🡪 compareTo()

Comparator 🡪 compare()

ActionListener 🡪 actionPerformed()

Callable 🡪 call()

@FunctionalInterface annotation is used explicitly to metion that it is functionalinterface. It is used mainly to validate syntax.

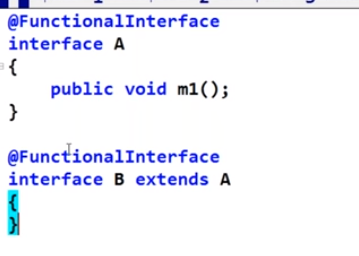


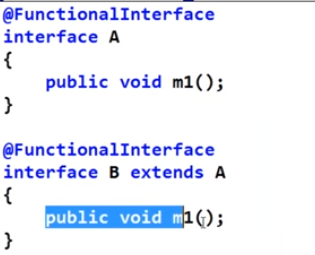
If we remove public void m1(); and try to compile we will get compile error because a functional interface should contain one abstract method.

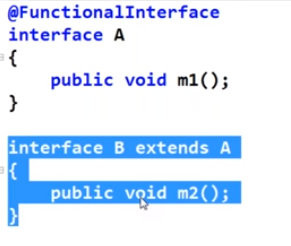


If we take more than 1 abstract method or 0 abstract method we will get compile time error.

Functional Interface wrt Inheritance:Below are valid syntaxes







Below is invalid because child interface b contains two abstract methods m1 and m2

