Java 8 Features

Table of Contents

1. Java 8 New Features

1.1. Why Java 8

- To Simplify Programming
- To Utilize Functional Programming benefits in Java
- To Enable Parallel Processing

1.2. Lambda Expressions & Functional Interfaces

Lambda Expressions

- Lambda Expression is just an anonymous (nameless) function. That means the function which doesn’t have the name, return type and access modifiers.
- Lambda Expression also known as anonymous functions or closures.
- Examples

```
public void m1(){
    System.out.println("Hello");
}

// AS
()->System.out.println("Hello");

// ---------------------------------------------------------------

public void sum(int a, int b){
    System.out.println(a+b);
}

// AS
(int a, int b)-> System.out.println(a+b);

// AS - Type Inferance
(a, b)-> System.out.println(a+b);

// ---------------------------------------------------------------

public int getLenth(String s){
    return s.length();
}

// AS
(s) -> return s.length();

// AS
(s) -> s.length();
```

Lambdas Conclusion

1) A lambda expression can have zero or more number of parameters (arguments).

```
Ex:
() -> sop("hello");
(int a ) -> sop(a);
(inta, int b) -> return a+b;
```

2) Usually we can specify type of parameter. If the compiler expects the type based on the context then we can remove type. i.e., programmer is not required.

```
(int a, int b) -> sop(a+b);
(a,b) -> sop(a+b);
```

3) If multiple parameters present then these parameters should be separated with comma (,).

4) If zero number of parameters available then we have to use empty parameter [like ()].

5) If only one parameter is available and if the compiler can expect the type then we can remove the type and parenthesis also.

```
(int a) -> sop(a);
(a) -> sop(a);
A -> sop(a);
```

6) Similar to method body lambda expression body also can contain multiple statements. If more than one statements present then we have to enclose inside within curly braces. If one statement present then curly braces are optional.

7) Once we write lambda expression we can call that expression just like a method, for this functional interfaces are required.