Lesson:

Lambda Expression

List of Concepts Involved:

- What is Lambda Expression
- Different ways to create Lambda Expression
- Lambda Expression exercises

Lambda Expression

- Lambda calculus is a big change in the mathematical world which was introduced in 1930.
- Because of the benefits of Lambda calculus, slowly these concepts started being used in the programming world.
- "LISP" is the first programming which uses Lambda Expression.
- The other languages which uses lambda expressions are:
  C#.Net
  C Objective
  C
  C++
  Python
  Ruby etc.
  and finally in java also.
- The Main Objective of λLambda Expression is to bring benefits of functional programming into java.

What is Lambda Expression (λ):

- 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.

Ex: 1

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

Equivalent lambda expressions

```
() →System.out.println("hello");
```

Ex:2

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

Equivalent lambda expressions

```
(a,b) →System.out.println(a+b);
```

- If the type of the parameter can be decided by the compiler automatically based on the context then we can remove types also.
- The above Lambda expression we can rewrite as (a,b)->System.out.println (a+b);

Ex: 3

```
public String str(String str) {
        return str
}
```

Equivalent lambda expressions

```
(str) →str
```

Conclusions:

- A lambda expression can have zero or more parameters(arguments).
- Usually we can specify the type of parameter.If the compiler expects the type based on the context then we can remove type. i.e., a programmer is not required.
- If multiple parameters are present then these parameters should be separated with comma(,).
- If there are zero number of parameters available then we have to use empty parameter [like ()].
- If only one parameter is available and if the compiler can expect the type then we can remove the type and parentheses also.
- 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 is present then curly braces are optional.
- Once we write a lambda expression we can call that expression just like a method, for this functional interfaces are required.

Functional Interfaces:

If an interface contains only one abstract method, such types of interfaces are called functional interfaces and the method is called functional method or single abstract method(SAM).

- **Runnable** It contains only run() method
- **Comparable** It contains only compareTo() method
- **ActionListener** It contains only actionPerformed()
- **Callable** It contains only call()method

Inside the functional interface in addition to the single Abstract method(SAM) we write any number of default and static methods.

Ex:

```
interface Interf {
       public abstract void m1();
       default void m2() {
              System.out.println (“hello”);
       }
}
```

In Java 8 ,SunMicroSystem introduced @FunctionalInterface annotation to specify that the interface is FunctionalInterface.

Ex:

```
@FunctionalInterface
Interface Interf {
	public void m1();
}
```

InsideFunctionalInterface we can take only one abstract method,if we take more than one abstract method then the compiler raise an error message that is called we will get compilation error.

Ex:

```
@FunctionalInterface {
	public void m1(); //this code gives compilation error.
	public void m2();
}
```