→ Functional Interfaces ←

[•]

An interface that contains exactly one abstract method is known as functional interface.

[•]

Functional Interfaces introduced in Java 8 allows us to use a lambda expression to initiate the interface's method and avoid using lengthy codes for the anonymous class implementation.

Example →

```
// interface
@ Functional Interface

interface Sample
  {
    // abstract method
     int calculate (int val);
  }
```

[•]

→ We can write more than one default methods

→ We can write more than one static methods.

→ We can write java.lang.Object class methods.

[•]
The main intention of the functional interface is to developing functional programming in java.

- Functional Programming:
A paradigm that emphasizes pure functions, immutability and higher-order functions.

[•] To developing lambda expressions.

[•] To developing method references.

[•]
We should write

@FunctionalInterface annotations.

:: Lambda Expressions ::

- (int x) -> x+1 // Single declared type argum).
- (int x) -> { return x+1; } // Same as above
- (x) -> x+1 // Single inferred-type arguments, same as below.
- x -> x+1 // Parenthesis optional for single inferred-type case.

- `(String s) -> s.length()`
  // Single declared type argument.

- `(Thread t) -> { t.start(); }`
  // Single declared type argument.

- `s -> s.length()`
  // Single Inferred-type argument.

- `t -> { t.start(); }`
  // Single Inferred-type argument.

- `(int x, int y) -> x+y`
  // Multiple declared-type parameters.

- `(x, y) -> x+y`
  // Multiple Inferred-type parameters.