Functional interface

1. It should contain exactly one abstract method (SAM: Single Abstract Method). This is also known as SAM interface.
2. It can contain any number of default and static methods. Restriction is only on abstract method.
3. It acts of a type for Lambda expression (Test2 t1 = () -> System.***out***.println("Hello World"); Here Test is Functional interface
4. It can be used to used to invoke Lambda expression (t1.m1();)

Why Functional interface should contain only one abstract method?

Let’s take an example’

**interface** Test {

**public** **void** m1();

**public** **void** m2();

}

In your method, if use below, then compiler doesn’t know whether to map the body to m1 or m2 method, so it should contain only one abstract method

Test t1 = () -> System.***out***.println("Hello World");

There will be an issue with mapping if we use multiple abstract methods.

What is advantage of @FunctionalInterface

@FunctionalInterface

**interface** Test {

**public** **void** add(**int** i, **int** j);

}

If someone tries to add an additional abstract method, then compiler throws the error.

If we don’t add @FunctionalInterface annotation, and someone accidentally adds an abstract method then all the classes that used the interface on lambda expression will get the error.