What are functional interfaces?

An interface that has a single abstract method is called a functional interface.

While an interface can have one or more default methods, it should have only one abstract method to be called a functional interface.

Java 8 has defined the `java.util.function` package, containing lots of functional interfaces. Some of the functional interfaces defined in Java 8 are `Predicate`, `Consumer`, `Supplier`, `Function`, etc.

The functional interface is used by lambda expressions. In the next chapter, we will discuss lambdas and also see the usage of some of the functional interfaces in Java 8.

What is `@FunctionalInterface` annotation?

Any interface that has only one abstract method can be annotated with the `@FunctionalInterface` annotation.

This is not mandatory but if an interface is annotated with `@FunctionalInterface` annotation and someone tries to add another abstract method to the interface, then the compiler will throw an error. Below is an example of a functional interface.

```java
@FunctionalInterface
public interface Functional {
    void doSomething();

    default void foo() {
        System.out.println("foo");
    }
}

```

If we try to add one more abstract method in the above interface, the compiler shows an error. If an interface is annotated with `@FunctionalInterface` annotation but does not contain even a single abstract method, then also the compiler will complain.