Functional Interface(Java 8+)

Definition:

A functional interface in Java is an interface that contains only one abstract method. It can have any number of default, static, and private methods. These are the key to implementing lambda expressions and method references in Java 8 and beyond.

Syntax:

```
@FunctionalInterface
interface InterfaceName {
   void methodName();
}
```

Example:

```
@FunctionalInterface
interface MyFunctionalInterface {
   void sayHello();
}

public class Test {
   public static void main(String[] args) {
      MyFunctionalInterface greet = () -> System.out.println("Hello from Lambda!");
      greet.sayHello();
   }
}
```

Key Points:

- - A functional interface has exactly one abstract method.
- - They can have default or static methods.
- - Introduced in Java 8 for use with lambda expressions.
- - Commonly used in streams, collections, threading, and event handling.
- - The @FunctionalInterface annotation helps to avoid accidental addition of extra abstract methods.

Built-in Functional Interfaces (Java 8+):

Interface	Method	Description
Predicate<T>	test()	Takes an argument and returns boolean
Function<T, R>	apply()	Takes an argument and returns a result
Consumer<T>	accept()	Takes an argument and returns nothing
Supplier<T>	get()	Takes no argument, returns a result

Example using Predicate:

```
import java.util.function.Predicate;

public class PredicateTest {
    public static void main(String[] args) {
        Predicate<String> isLong = str -> str.length() > 5;
    System.out.println(isLong.test("Hello"));  // false
        System.out.println(isLong.test("Functional"));  // true
    }
}
```

Example using Function:

```
import java.util.function.Function;

public class FunctionTest {
    public static void main(String[] args) {
        Function<Integer, String> convert = i -> "Value: " + i;
        System.out.println(convert.apply(10));  // Value: 10
    }
}
```

Use Cases of Functional Interfaces:

- - Lambda expressions
- - Stream API operations (filter, map, forEach, etc.)
- - Callback functions
- - Asynchronous programming
- - Cleaner and more readable code