**Why need default methods**

The reason we have default methods in interfaces is to allow the developers to add new methods to the interfaces without affecting the classes that implements these interfaces.

**Static methods**

static method helps us in **providing security** by not allowing implementation classes to override them.

**Method References**

They're often used to create simple lambda expressions by referencing existing methods. ... Instance methods of particular objects. Instance methods of an arbitrary object of a particular type

**Predicate functional interface**

The Functional Interface **PREDICATE** is defined in the *java.util.function package*. It improves manageability of code, helps in unit-testing them separately, and contain some methods like:

  Predicate<Integer> lesserthan = i -> (i < 18);

        // Calling Predicate method

        System.out.println(lesserthan.test(10));

**Function**

 It represents a function which takes in one argument and produces a result. Hence this functional interface takes in 2 generics namely as follows:

apply()

andThen()

compose()

identity()

  Function<Integer, Double> half = a -> a / 2.0;

        // Applying the function to get the result

        System.out.println(half.apply(10));

**Consumer Functional Interface**

Represents an operation that accepts a single input argument and returns no result

method is accept(Object)..

**Suppiler Functional Interface**

Represents a supplier of results.

There is no requirement that a new or distinct result be returned each time the supplier is invoked.

This is a [functional interface](https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html) whose functional method is [get()](https://docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html#get--).