**Q #1) List down the new features introduced in Java 8?**

**Answer:** **New features that are introduced in Java 8 are enlisted below:**

* Lambda Expressions
* Method References
* Optional Class
* Functional Interface
* Default methods
* Stream API

**Q #2) What are Functional Interfaces?**

**Answer:** Functional Interface is an interface that has only one abstract method. The implementation of these interfaces is provided using a Lambda Expression which means that to use the Lambda Expression, you need to create a new functional interface or you can use the predefined [functional interface of Java 8](https://www.softwaretestinghelp.com/java-8-interface-changes/).

The annotation used for creating a new Functional Interface is “**@FunctionalInterface**”.

**Q #3) What is an optional class?**

**Answer:** Optional class is a special wrapper class introduced in Java 8 which is used to avoid NullPointerExceptions. This final class is present under java.util package. NullPointerExceptions occurs when we fail to perform the Null checks.

**Q #4) What are the default methods?**

**Answer:** Default methods are the methods of the Interface which has a body. These methods, as the name suggests, use the default keywords. The use of these default methods is “Backward Compatibility” which means if JDK modifies any Interface (without default method) then the classes which implement this Interface will break.

On the other hand, if you add the default method in an Interface then you will be able to provide the default implementation. This won’t affect the implementing classes.

**Syntax:**

|  |
| --- |
| public interface questions{            default void print() {    System.out.println("www.softwaretestinghelp.com");               }      } |

**Q #5) What are the main characteristics of the Lambda Function?**

**Answer: Main characteristics of the Lambda Function are as follows:**

* A method that is defined as Lambda Expression can be passed as a parameter to another method.
* A method can exist standalone without belonging to a class.
* There is no need to declare the parameter type because the compiler can fetch the type from the parameter’s value.
* We can use parentheses when using multiple parameters but there is no need to have parenthesis when we use a single parameter.
* If the body of expression has a single statement then there is no need to include curly braces.

**Q #6) How can you create a Functional Interface?**

**Answer:** Although Java can identify a Functional Interface, you can define one with the annotation

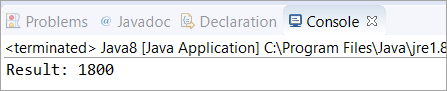
**@FunctionalInterface**

Once you have defined the functional interface, you can have only one abstract method. Since you have only one abstract method, you can write multiple static methods and default methods.

**Below is the programming example of FunctionalInterface written for multiplication of two numbers.**

|  |
| --- |
| @FunctionalInterface // annotation for functional interface  interface FuncInterface {        public int multiply(int a, int b);  }  public class Java8 {       public static void main(String args[]) {          FuncInterface Total = (a, b) -&gt; a \* b;          // simple operation of multiplication of 'a' and 'b'          System.out.println("Result: "+Total.multiply(30, 60));      }  } |

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/04/8th-Question.png)

**Q #7) What is Method Reference?**

**Answer:** In Java 8, a new feature was introduced known as Method Reference. This is used to refer to the method of functional interface. It can be used to replace Lambda Expression while referring to a method.

**For Example:** If the Lambda Expression looks like

num -> System.out.println(num)

Then the corresponding Method Reference would be,

System.out::println

where “::” is an operator that distinguishes class name from the method name.

**Q #8) What is a Predicate? State the difference between a Predicate and a Function?**

**Answer:** Predicate is a pre-defined Functional Interface. It is under java.util.function.Predicate package. It accepts only a single argument which is in the form as shown below,

**Predicate<T>**

| **Predicate** | **Function** |
| --- | --- |
| It has the return type as Boolean. | It has the return type as Object. |
| It is written in the form of **Predicate< T>** which accepts a single argument. | It is written in the form of **Function< T, R>** which also accepts a single argument. |
| It is a Functional Interface which is used to evaluate Lambda Expressions. This can be used as a target for a Method Reference. | It is also a Functional Interface which is used to evaluate Lambda Expressions. In Function< T, R>, T is for input type and R is for the result type. This can also be used as a target for a Lambda Expression and Method Reference. |

**Q #9) What is a Stream API? Why do we require the Stream API?**

**Answer:** Stream API is a new feature added in Java 8. It is a special class that is used for processing objects from a source such as Collection.

**We require the Stream API because,**

* It supports aggregate operations which makes the processing simple.
* It supports Functional-Style programming.
* It does faster processing. Hence, it is apt for better performance.
* It allows parallel operations.

**Q #10) What are static methods in Interfaces?**

Static methods, which contains method implementation is owned by the interface and is invoked using the name of the interface, it is suitable for defining the utility methods and cannot be overridden