Q 1) 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 2) 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 3) 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.

Q 4) 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 5) What was wrong with the old date and time?

Answer: Enlisted below are the drawbacks of the old date and time:

Java.util.Date is mutable and is not thread-safe whereas the new Java 8 Date and Time API are thread-safe.

Java 8 Date and Time API meets the ISO standards whereas the old date and time were poorly designed.

It has introduced several API classes for a date like LocalDate, LocalTime, LocalDateTime, etc.

Talking about the performance between the two, Java 8 works faster than the old regime of date and time.

Q 6) What is the difference between the Collection API and Stream API?

Answer: The difference between the Stream API and the Collection API can be understood from the below table:

| Stream API | Collection API |
| --- | --- |
| It was introduced in Java 8 Standard Edition version. | It was introduced in Java version 1.2 |
| There is no use of the Iterator and Spliterators. | With the help of forEach, we can use the Iterator and Spliterators to iterate the elements and perform an action on each item or the element. |
| An infinite number of features can be stored. | A countable number of elements can be stored. |
| Consumption and Iteration of elements from the Stream object can be done only once. | Consumption and Iteration of elements from the Collection object can be done multiple times. |
| It is used to compute data. | It is used to store data. |

Q 7) 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.

Q 8) What is a SAM Interface?

Answer: Java 8 has introduced the concept of FunctionalInterface that can have only one abstract method. Since these Interfaces specify only one abstract method, they are sometimes called as SAM Interfaces. SAM stands for “Single Abstract Method”.

Q 9) 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.

Q 10) Explain the following Syntax

String:: Value of Expression

Answer: It is a static method reference to the *ValueOf* method of the String class. System.out::println is a static method reference to println method of out object of System class.

It returns the corresponding string representation of the argument that is passed. The argument can be Character, Integer, Boolean, and so on.

Q 11) 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 12) 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 13) What is the difference between Stream’s findFirst() and findAny()?

Answer: As the name suggests, the findFirst() method is used to find the first element from the stream whereas the findAny() method is used to find any element from the stream.

The findFirst() is predestinarianism in nature whereas the findAny() is non-deterministic. In programming, Deterministic means the output is based on the input or initial state of the system.

Q 14) What is the difference between Iterator and Spliterator?

Answer: Below is the differences between Iterator and Spliterator.

| Iterator | Spliterator |
| --- | --- |
| It was introduced in Java version 1.2 | It was introduced in Java SE 8 |
| It is used for Collection API. | It is used for Stream API. |
| Some of the iterate methods are next() and hasNext() which are used to iterate elements. | Spliterator method is tryAdvance(). |
| We need to call the iterator() method on Collection Object. | We need to call the spliterator() method on Stream Object. |
| Iterates only in sequential order. | Iterates in Parallel and sequential order. |

Q 15) What is the Consumer Functional Interface?

Answer: Consumer Functional Interface is also a single argument interface (like Predicate<T> and Function<T, R>). It comes under java.util.function.Consumer. This does not return any value.

In the below program, we have made use of the accept method to retrieve the value of the String object.

|  |
| --- |
| import java.util.function.Consumer;    public class Java8 {        public static void main(String[] args)            Consumer&lt;String&gt; str = str1 -&gt; System.out.println(str1);          str.accept("Saket");            /\* We have used accept() method to get the           value of the String Object           \*/      }  } |

Q 16) What is Nashorn in Java 8?

Answer: Nashorn in Java 8 is a Java-based engine for executing and evaluating JavaScript code.

Q 17) What is MetaSpace in Java 8?

Answer: In Java 8, a new feature was introduced to store classes. The area where all the classes that are stored in Java 8 are called MetaSpace. MetaSpace has replaced the PermGen.

Till Java 7, PermGen was used by Java Virtual Machine to store the classes. Since MetaSpace is dynamic as it can grow dynamically and it does not have any size limitation, Java 8 replaced PermGen with MetaSpace.

Q 18) What is the difference between Java 8 Internal and External Iteration?

Answer: The difference between Internal and External Iteration is enlisted below.

| Internal Iteration | External Iteration |
| --- | --- |
| It was introduced in Java 8 (JDK-8). | It was introduced and practiced in the previous version of Java (JDK-7, JDK-6 and so on). |
| It iterates internally on the aggregated objects such as Collection. | It iterates externally on the aggregated objects. |
| It supports the Functional programming style. | It supports the OOPS programming style. |
| Internal Iterator is passive. | External Iterator is active. |
| It is less erroneous and requires less coding. | It requires little more coding and it is more error-prone. |

Q 19) What is JJS?

Answer: JJS is a command-line tool used to execute JavaScript code at the console. In Java 8, JJS is the new executable which is a JavaScript engine.

Q 20) What is ChronoUnits in Java 8?

Answer: ChronoUnits is the enum that is introduced to replace the Integer values that are used in the old API for representing the month, day, etc.