Java 8 Question

<https://docs.oracle.com/javase/8/docs/>

1.Java 8 come,

| **Java Releases** | **Release Date** |
| --- | --- |
| **Java 8 Release (OTN)** | **March 18, 2014** |

2.Java 8 made,

3. What are the new features released in Java 8?

1.Default and static method

2.Lambda Expression

3.Java 10 improvement

4.Concurrency API improvements

5.Java Time API

6.Collection Time improvement

7.Java Stream API

8.For Each method

4. What are the main benefits of new features introduced in Java

5. What is a Lambda expression in Java 8?

Ans-:lambda expression is a short block of code which takes in parameters and returns a value

6. What are the three main parts of a Lambda expression in Java?

7. What is the data type of a Lambda expression?

Lambda Expressions are **anonymous functions**. Lambda expressions basically express instances of functional interfaces An interface with a single abstract method is called a functional interface.

Its return type is a **parameter -> expression body**

8. Why did Oracle release a new version of Java like Java 8?

Java 8 includes features for **productivity, ease of use, improved polyglot programming, security and improved performance**.

9. What are the advantages of a lambda expression?

Fewer Lines of Code

Sequential and Parallel execution support by passing behavior as an argument in methods

Higher Efficiency

## **Example**

**import java.util.\*;**

**public class LambdaExpressionTest {**

**public static void main(String args[]) {**

**new LambdaExpressionTest().print();**

**}**

**public static void print() {**

**List<String> list = new ArrayList<String>();**

**list.add("Tutorials Point");**

**list.stream().forEach((String) -> { // lambda expression**

**System.out.println("The string is: " + list);**

**});**

**}**

**}**

10. What is a Functional interface in Java 8?

One abstract method is called Functional Interface. lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default methods.

11. Why do we need Functional interface in Java?

Java language provides support for using lambda expressions only with functional interfaces; **one abstract method is called Functional Interface.**

12. What are the differences between Collection and Stream API in Java 8?

**A stream does not store data**. An operation on a stream does not modify its source, but simply produces a result. Collections have a finite size, but streams do not.

13. What are the main uses of Stream APl in Java 8?

The stream API **allows you to perform operations on collections without external iteration**. In this case, we're performing a filter operation which will filter the input collection based on the condition specified.

14. What are the differences between Intermediate and Terminal Operations?

Intermediate operation will transform a stream into another stream, such as map(MapperFn) or filter(Predicate) Terminal operation will produce a result or side-effect, such as count() or forEach(Consumer)

15. What is a Spliterator in Java 8?

Spliterators can be used for **traversing the elements of a source one by one**. These sources could be an array, a Collection, an IO Channel or a generator function.

16. What is Type Inference in Java 8?

17. How does Internal Iteration work in Java 8?

Internal Iterators manage the iterations in the background. the elements of the Collection, rather than managing the iteration and making sure that all the elements are processed one-by-one.

**18. What are the main advantages of Internal Iterator over External Iterator?**

**19. Can we provide implementation of a method in a Java Interface?**

**20. Why do we need the Default method in a Java 8 Interface?**

**21. What is the purpose of a Static method in an Interface in Java8?**

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

Jb bhi hum define karty h interface or static method ko object class method will be generade compile error.

**22. What are the core ideas behind the Date/Time API of Java 82**

**23. What are the advantages of the new Date and Time APl in Java 8 over the old Date API?**

**24. What are the main differences between legacy Date/Time APl and Java 8 Date API?**

25. How can we get duration between two dates or time in Java 8?

26. What is the new method family introduced in Java 8 for processing of Arrays?

**import java.util.Arrays;**

**class Main {**

**public static void main(String[] args)**

**{**

**int intArray[] = {5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100};**

**//calculate sum using iterative method**

**int sum = 0;**

**for (int i = 0; i <intArray.length; i++)**

**sum += intArray[i];**

**System.out.println("Iteration approach : Average :" +**

**(sum / intArray.length));**

**// sum using streams**

**sum = Arrays.stream(intArray) // Step 1**

**.sum(); // Step 2**

**System.out.println("Streams approach : Average " +**

**(sum / intArray.length));**

**// print array elements from stream using forEach**

**System.out.println("Array elements : ");**

**Arrays.stream(intArray)**

**.forEach(e->System.out.print(e + " "));**

**}**

**}**

27. How does Java 8 solve the Diamond problem of Multiple Inheritance?

28. What are the differences between Predicate, Supplier and Consumer in Java 8?

**Predicate** is an anonymous function that accepts one argument and returns a result. **Supplier** is an anonymous function that accepts no argument and returns a result. **Consumer** is an anonymous function that accepts one argument and returns no result.

29. Is it possible to have a default method definition in an interface?

30. How does Java 8 support Multiple Inheritance?

The Java programming language supports multiple inheritance of type, which is **the ability of a class to impl**

31. How can you get the name of Parameter in Java by using reflection?

The getParameterTypes() method of Method class returns an array of Class objects that represents the parameter types, declared in the method at time of coding. The getParameterTypes() returns an array of length 0 if the method object takes no parameters.

32. What is Optional in Java 8?

Optional is a container object which may or may not contain a non-null value.

isPresent()

get()

33. How can we get current time by using the Date/Time API of Java 8?

34. Is it possible to define a static method in an Interface?

**Static Methods** in **Interface** are those methods, which are defined in the interface with the keyword static.

[Default Method in Interface](https://www.geeksforgeeks.org/default-methods-java/), the static method in an interface can be defined in the interface, but cannot be overridden in Implementation Classes. To use a static method, Interface name should be instantiated with it, as it is a part of the Interface only.

**35. How can we analyze the dependencies in Java classes and packages?**

1. Identify the artifact with dependencies you want to analyze.
2. Trace through any relationships defined on that artifact and identify the targets of the relationships. ...
3. If these "dependencies" also depend on other artifacts, then the selected artifact will also have an indirect dependency.

**36. What are the new JVM arguments introduced by Java 8?**

**37. What are the popular annotations introduced in Java 8?**

Java 8 has included two new features **repeating and type annotations**

**38. What is a StringJoiner in Java 8?**

public final class StringJoiner

extends [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)

StringJoiner is used to construct a sequence of characters separated by a delimiter and optionally starting with a supplied prefix and ending with a supplied suffix.

For example:

List<Integer> numbers = Arrays.asList(1, 2, 3, 4);

String commaSeparatedNumbers = numbers.stream()

.map(i -> i.toString())

.collect(Collectors.joining(", "));

<https://docs.oracle.com/javase/8/docs/api/java/util/StringJoiner.html>

**39. What is the type of a Lambda expression in Java 8?**

Lambda Expressions were added in Java 8. A lambda expression is **a short block of code which takes in parameters and returns a value**.

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Integer> numbers = new ArrayList<Integer>();

numbers.add(5);

numbers.add(9);

numbers.add(8);

numbers.add(1);

numbers.forEach( (n) -> { System.out.println(n); } );

}

}

**40. Main differences between an interface with default method and an abstract class.**

[**https://www.infoworld.com/article/2077421/abstract-classes-vs-interfaces-in-java.html**](https://www.infoworld.com/article/2077421/abstract-classes-vs-interfaces-in-java.html)