MCQ on Java 8 Features

Q1. Which Java 8 feature allows us to filter a collection based on a predicate?

(a) Lambda Expression

(b) Default Method

(c) Optional class

(d) Stream API

Q2. Which Java 8 feature allows us to pass behavior as a parameter to a method?

(a) Lambda expressions

(b) Optional class

(c) Stream API

(d) Functional Interface

Q3. What is the output of the following program?

```
List<String> names = Arrays.asList("ABC", "CAB", "BCA");
String result = names.stream().reduce("", (a, b) -> a + b.charAt(1));
System.out.println(result);
```

(a) "ABC"

(b) "ACB"

(c) "BAC"

(d) "CBA"

Q4. Which is the new method introduced in the String class in Java 8?

(a) offsetByCodePoints()

(b) matches()

(c) intern()

(d) join()

Q5. What will be the output of the following code snippet?

```
Optional num= Stream.of(9, 5, 8, 7, 4, 9, 2, 11, 10, 3)
                  .filter(n -> n > 10)
                  .filter(n -> n % 5 == 0)
                  .findFirst();
System.out.println(num);
```

(a) Optional[8]

(b) Optional.empty

(c) Optional[11]

(d) 11

Q6. Which one is the correct syntax for declaring a lambda expression in Java 8?

(a) (param1, param2) => expression

(b) (param1, param2) :: { expression }

(c) (param1, param2) -> expression

(d) (param1, param2) : expression

Q7. Which of the following interface is not a functional interface?

(a) An interface with a single abstract method

(b) An interface with an abstract method and a default method

(c) An interface with an abstract method and an equals() method

(d) An interface with an abstract method and a run() method

Q8. What is the new Date and Time API in Java 8?

(a) A replacement for the java.sql.Date and java.sql.Calendar classes

(b) A new API for working with Optional

(c) A replacement for the java.util.Date and java.util.Calendar classes

(d) A new API for working with Java Stream API

Q9. Which one among the following is the main feature introduced in Java 8?

(a) Annotations

(b) Generics

(c) Lambda expressions

(d) Enumerations

Q10. Which feature out of following is introduced in Java 8?

(a) Strings in switch statement

(b) Improved type inference: the diamond operator <>

(c) StringJoiner Class

(d) Private methods in Interfaces

Q11. What is the output of the following program?

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
int result= numbers.stream()
                .filter(n -> n % 2 == 0)
                .mapToInt(Integer::intValue)
                .sum();
System.out.println(result);
```

(a) 10

(b) 9

(c) 6

(d) 15

Q12. Mary is a Java developer. She is working in an assignment. She wants to use a predefined functional interface introduced in Java 8 that doesn’t take any input and it always returns some object. Which one among the following option should she use to complete her assignment?

(a) Consumer

(b) Pedicate

(c) Supplier

(d) BiConsumer

Q13. What is the purpose of default method in Java 8?

(a) A method that is automatically called when an object is created

(b) A method that is automatically called when an object is destroyed

(c) A method that is defined in an interface with a default implementation

(d) A method that is called by the garbage collector to free up memory

Q14. What is the method reference feature in Java 8?

(a) A way to create a new object using a constructor reference

(b) A way to call a static method using a method reference

(c) A way to call an instance method using a method reference

(d) All of the above

Q15. What is the Optional class in Java 8?

(a) A class that represents a value which can either be present or absent

(b) A class that represents an immutable collection

(c) A class that represents optional values to stream of elements

(d) A class as an alternative to functional interface

Q16. What is the new feature introduced in Java 8 in the context of concurrent programming?

(a) ExecutorService

(b) CompletableFuture

(c) ThreadLocal

(d) ReentrantLock

Q17. What is the purpose of the forEach() method in Java 8 Streams API?

(a) To apply an operation to each element of a stream

(b) To filter the elements of a stream

(c) To print the elements of a stream

(d) To map the elements of a stream to a different type

Q18. Robert is working in a Java project. In one of his assignment, he is using Stream API of Java 8 where he wants to transform each element to multiple values and also compress the resulting stream. Which of the following method of Stream API will be the best option for him?

(a) map()

(b) filter()

(c) flatMap()

(d) reduce()

Q19. What is the Stream API in Java 8?

(a) A new file I/O stream library

(b) A new concurrency library

(c) A new API for working with collections

(d) A new API for working with databases

Q20. In the context of Java 8 Stream API, assume that you have a requirement to convert a stream into an array or a collection. Which of the following method of the Stream interface will you apply on the stream to achieve that?

(a) toCollection()

(b) toArray()

(c) toList()

(d) collect()

Q21. What is the output of the following code?

```
Stream<String> stream = Stream.of("hello", "world", "java");
 stream.filter(s -> s.contains("o"))
      .map(String::toUpperCase)
      .forEach(System.out::print);
```

(a) HELLO WORLD JAVA

(b) hello world java

(c) HELLOWORLD

(d) HELLO WORLD

Q22. When should we use the reduce() method in Java 8 streams?

(a) When we wish to apply a transformation to each element of a stream

(b) When we wish to convert a nested stream into a single stream

(c) When we wish to produce one single result from a sequence of elements

(d) When we wish to filter the elements of a stream

Q23. What is the output of the following code?

```
Stream<Integer> stream = Stream.of(1, 2, 3, 4, 5);
stream.reduce((a, b) -> a + b)
     .ifPresent(System.out::println);
```

(a) 15

(b) 10

(c) 6

(d) 5

Q24. Which is the mandatory condition to define a functional interface in Java 8?

(a) The interface should have @functionalInterface annotation on top of it

(b) The interface should have only one method

(c) The interface should have only one abstract method

(d) The interface should at least have one method

Q25. What is the purpose of the parallelStream() method in Java 8?

(a) To filter the elements of a stream

(b) To apply an operation to each element of a stream

(c) To create a parallel stream from a sequential stream

(d) To create a sequential stream from a parallel stream

Q26. What is the output of the following code?

```
List<Integer> list = Arrays.asList(1, 15, 3, 10, 27);
 int sum = list.stream()
              .filter(i -> i % 3 == 0)
              .mapToInt(Integer::intValue)
              .sum();
 System.out.println(sum);
```

(a) 18

(b) 45

(c) 30

(d) 56

Q27. What is the purpose of the Optional class in Java 8?

(a) To provide optional values to a stream

(b) To store a reference to a value that may or may not be null

(c) To apply a transformation to each element of a stream

(d) To filter the elements of a stream

Q28. What is the output of the following code?

```
 List<String> list = Arrays.asList("banana", "apple", "cherry");
  list.stream()
     .sorted((s1, s2) -> s1.compareTo(s2))
     .forEach(System.out::println);
```

(a) apple, banana, cherry

(b) cherry, banana, apple

(c) banana, apple, cherry

(d) apple, cherry, banana

Q29. What is the purpose of the peek() method in Java 8 streams?

(a) To apply an operation to each element of a stream

(b) To filter the elements of a stream

(c) To convert a stream into an array or a collection

(d) To apply a transformation to each element of a stream

Q30. What is the output of the following code?

```
IntStream.range(1, 6)
        .mapToObj(i -> i + " ")
        .forEach(System.out::print);
```

(a) 1 2 3 4 5 6

(b) 2 3 4 5

(c) 1 2 3 4 5

(d) 1 2 3 4

Q31. Which of the following is a valid lambda expression in Java 8?

(a) () -> { System.*out*.println("Hello"); }

(b) (int x) -> { x++; }

(c) (String s, int x) -> { System.*out*.println(s + x); }

(d) All of the above

Q32. Which Java 8 feature allows us to iterate through a collection using lambda expressions?

(a) Stream API

(b) Method Reference

(c) Optional

(d) Annotation Processing

Q33. Which of the following is not a valid target type for a lambda expression in Java 8?

(a) An interface with a single abstract method

(b) A functional interface

(c) A class with a single abstract method

(d) All of the above are valid target types

Q34. What is the purpose of the Supplier interface in Java 8?

(a) To represent a method that takes no arguments and returns no value

(b) To represent a method that takes one argument and returns a value

(c) To represent a method that takes no arguments and returns a value

(d) To represent a method that takes one argument and returns no value

Q35. What is the output of the following code snippet?

```
List<String> words = Arrays.asList("Peris", "London", "New York",
"Sydney", "Washington");
String result = words.stream()
                   .filter(w -> w.length() > 10)
                   .findFirst()
                   .orElse("none");
System.out.println(result);
```

(a) none

(b) Washington

(c) New York

(d) An error is thrown

—-

For more questions, kindly visit Java 8 MCQ.

Answers With Explanations

A1: d). Stream API

Explanation: Stream API is a new feature introduced in Java 8 that allows us to filter a collection based on a predicate.

A2: a) Lambda expressions

Explanation: Lambda expressions allow us to pass behavior as a parameter to a method, which can be used to implement functional interfaces.

A3: c) “BAC”
Explanation: The code reduces the list of names to a single string by concatenating the second character of each name. The resulting string is “BAC”.

A4: d) join()
Explanation: The new join() method in the String class in Java 8 allows you to join a collection of strings into a single string with a specified delimiter. For examples of join() method, kindly visit Examples of join() method.
A5: b) Optional.empty
Explanation: Since 11 is not divisible by 5, the findFirst() will return an empty stream. When the stream is empty, the return value is an empty Optional which is represented by ‘Optional.empty’.

A6: c) (param1, param2) -> expression
Explanation: In Java 8, a lambda expression is declared using the syntax (param1, param2) -> expression, where param1 and param2 are the parameters of the lambda expression, and expression is the body of the lambda expression.
A7: d) An interface with an abstract method and a run() method
Explanation: In addition to single abstract method, we can also have any number of default & static methods inside a Functional Interface. Moreover, we don’t have any restrictions on including ‘Object’ class’ public methods inside Functional interfaces. Needless to say, they are equals(), hashCode(), notify(), notifyAll(), toString(), wait(), getClass().
A8: c) A replacement for the java.util.Date and java.util.Calendar classes
Explanation: The new Date and Time API in Java 8 provides a more comprehensive and flexible API for working with dates and times, with classes such as LocalDate, LocalTime, and ZonedDateTime.

A9: c) Lambda expressions
Explanation: Java 8 introduced lambda expressions, which allows you to write more concise and readable code by providing a way to represent anonymous functions.

A10: c) StringJoiner class

Explanation: StringJoiner is a new class added in Java 8 under java.util package. In fact, we can use it for joining Strings by making use of a delimiter, prefix, and suffix. For example on StringJoiner class, kindly visit a separate article on StringJoiner.
A11: c) 6
Explanation: The code filters the even numbers in the list, maps them to integers, and then sums them up using the sum() method. The even numbers in the list are 2 and 4, and their sum is 6.

A12: c) Supplier
Explanation: The Supplier functional interface in Java 8 represents a function that takes no arguments and returns an object.
A13: c) A method that is defined in an interface with a default implementation
Explanation: Default methods in Java 8 allow you to add new methods to interfaces without breaking backward compatibility, by providing a default implementation that can be overridden by implementing classes.

A14: d) All of the above
Explanation: Method reference in Java 8 allows you to reference a method or constructor by its name, without actually calling it, with various types such as constructor references, static method references, and instance method references.
A15: a) A class that represents an optional value
Explanation: The Optional class in Java 8 is used to represent an optional value, which can either be present or absent, to avoid NullPointerException and make code more readable.

A16: b) CompletableFuture
Explanation: CompletableFuture in Java 8 is a new feature for asynchronous and concurrent programming, which allows you to create and chain multiple asynchronous tasks with a fluent API.

A17: a) To apply an operation to each element of a stream
Explanation: The forEach() method in Java 8 streams is used to apply a specified operation to each element of a stream.

A18: c) flatMap()

Explanation: If you want to transform the elements of a stream in some way.
Use map() if you want to transform each element into a single value.
Use flatMap() if you want to transform each element to multiple values and also compress/flatten the resulting stream.
A19: c) A new API for working with collections
Explanation: The Stream API in Java 8 provides a new way to work with collections in a functional style, with operations such as filter, map, and reduce.
A20: d) collect()
Explanation: The collect() method of the 'Stream' interface is used to convert a stream into an array, a collection, or any other data structure.
The Collectors class provides various methods like toCollection(), toList(), toSet(), toMap(), and toConcurrentMap() to collect the result of Stream into List, Set, Map, and ConcurrentMap respectively. Please note that the collect() method doesn't belong to the Collectors class. It is defined in Stream interface and that's the reason you can call it on Stream after doing any filtering or mapping operations. However, it accepts a Collector to accumulate elements of Stream into a specified Collection.
A21: c) HELLOWORLD
Explanation: The code filters the elements in the stream that contain the letter 'o', converts them to uppercase strings, and prints them out using the forEach() method. The elements in the stream that contain the letter 'o' are "hello" and "world". Since 'print' (not 'println') method is used, it will print the two strings without any space or line break in between.

A22: c) When we wish to produce one single result from a sequence of elements
Explanation: The flatMap() method in Java 8 streams is used to convert a nested stream into a single stream. It flattens the nested stream by merging its elements into a single stream.

A23: a) 15
Explanation: The code reduces the stream to a single value by adding all its elements, and prints out the result using the ifPresent() method. The sum of the elements in the stream is 1 + 2 + 3 + 4 + 5 = 15.

A24: c) The interface should have only one abstract method

Explanation: A functional interface is an interface that has only one abstract method. We apply @FunctionalInterface annotation in order to tell compiler that it is a functional interface. When we try to add any additional abstract method in it, compiler will display a compilation error. On the other hand, in the absence of @FunctionalInterface annotation, compiler will consider it as a normal interface and doesn't display any compilation error.

A25: c) To create a parallel stream from a sequential stream
Explanation: The parallelStream() method in Java 8 is used to create a parallel stream from a sequential stream. It allows for the processing of elements in the stream to be done in parallel.

A26: b) 45
Explanation: The code filters the numbers which are divisible by 3 in the list, maps them to integers, and calculates the sum using the sum() method. The elements in the list which are divisible by 3 are 15, 3 and 27, and their sum is 45.

A27: b) To store a reference to a value that may or may not be null
Explanation: The Optional class in Java 8 is used to store a reference to a value that may or may not be null. It is used to avoid null pointer exceptions and to make the code more robust.

A28: a) apple, banana, cherry
Explanation: The code sorts the elements in the list in natural order using the sorted() method and prints them out using the forEach() method.

A29: a) To apply an operation to each element of a stream
Explanation: The peek() method in Java 8 streams is used to apply an operation to each element of a stream. It does not modify the stream, but allows for side effects such as logging or debugging.

A30: c) 1 2 3 4 5
Explanation: The code generates a stream of integers from 1 to 5 using the range() method, maps each integer to a string with a trailing space using the mapToObj() method, and prints out the strings using the forEach() method.
A31: a) () -> { System.out.println("Hello"); }
Explanation: This lambda expression takes no arguments and has a single statement that prints "Hello" to the console. The other options are invalid

because they either have an incorrect number of arguments or an incorrect syntax.

A32: a) Stream API
Explanation: Stream API is a new feature introduced in Java 8 that provides a functional way to iterate, filter, and transform collections using lambda expressions.

A33: c) A class with a single abstract method
Explanation: A lambda expression can only be used with a target type that is a functional interface or an interface with a single abstract method. A class is not a valid target type for a lambda expression.

A34: c) To represent a method that takes no arguments and returns a value
Explanation: The Supplier interface in Java 8 is a functional interface that represents a method that doesn't take any argument and returns a value. We use it when we need to get some value based on some operation like supply Random numbers, supply Random OTPs, supply Random Passwords etc.
A35: a) none
Explanation: The code creates a list of strings, and then creates a stream from the list. The stream is filtered to only include strings with a length greater than 10, and the first element of the resulting stream is retrieved using the findFirst() method. Since no element is found which is greater than 10 in length, the orElse() method returns the string "none".

---

1. Lambdas introduced in Java 8 allow us to treat -

a. ○ Data as code

b. ○ Code as data

c. ○ none

d. ○ all

2. The newly introduced Streams API is available in which package of java 8:

a. ○ java.io.streams

b. ○ java.io.stream

c. ○ java.util.streams

d. ○ java.util.stream

3. Which class can be used Instead of System.getCurrentTimeMillis() to get a date and time in Java 8 -

a. ○ Clock

b. ○ Date

c. ○ Time

d. ○ Timer

4. Lambda expressions in java 8 are based on

a. ○ Procedural programming

b. ○ Functional programming

c. ○ Data programming

d. ○ All

5. how many methods are there in functional interface in Java 8?

a. ○ 0

b. ○ 1

c. ○ 2

d. ○ 3

Q6 - Q15, 10 MEDIUM level difficulty questions 2 mark each. 2 * 10 = 20 marks

6. In Java 8 Interfaces, methods can be:

a. ○ default

b. ○ abstract

c. ○ all

d. ○ none

7. The Java 8 API with a sequence of elements which of these supports sequential and parallel aggregate operations -

a. ○ Hadoop

b. ○ Streams

c. ○ SequenceProgramming

d. ○ Big-data

8. Stream operations in java 8 can be divided into

a. ○ Terminal types

b. ○ Intermediate types

c. ○ All

d. ○ None

9. Which package contains Date/Time (JSR 310) API in Java 8 -

a. ○ java.time

b. ○ java.util.time

c. ○ java.timedate

d. ○ java.util.calendar

10. Nashorn the new JavaScript engine is an implementation of -

a. ○ javax.engine.Engine

b. ○ javax.script.Engine

c. ○ javax.javaScript.Engine

d. ○ javax.script.ScriptEngine

11. Which is new command line tool for the Nashhorn JavaScript engine in java 8 -

a. ○ jcs

b. ○ jfs

c. ○ jjs

d. ○ jss

12. Which of these should be used to show package-level and class-level dependencies of Class files in Java 8

a. ○ dep

b. ○ ideps

c. ○ jdep

d. ○ jdeps

13. Predicate is in which package in Java 8

a. ○ java.util.predicate

b. ○ java.util.object

c. ○ java.util.objects

d. ○ java.util.predict

14. Which of these represents a process that accepts one argument and produces a result in Java 8

a. ○ Function

b. ○ Process

c. ○ Method

d. ○ JavaFunctions

15. In java 8 Function is ?

a. ○ Class

b. ○ Interface

c. ○ Lambda Expression

d. ○ Object

Q16 - Q25, 10 HARD level difficulty questions 3 mark each. 3 * 10 = 30 marks

16. PermGen space has been replaced with which of these in Java 8 -

a. ○ PermSpace

b. ○ PermSpaceGen

c. ○ Metaspace

d. ○ MetaGenSpace

17. What can help us in avoiding NullPointeExceptions and null checks in java 8 -

a. ○ Optional

b. ○ Required

c. ○ NotNull

d. ○ NotRequired

18. code before Java 8 essentially used to be -

a. ○ Declarative

b. ○ Imperative

c. ○ Subjective

d. ○ None

19. In java 8, R apply(T t) is a method of-

a. ○ Function

b. ○ Process

c. ○ Predicate

d. ○ None

20. What is Predicate in Java 8 -

a. ○ method

b. ○ class

c. ○ Interface

d. ○ Framework

21. How sorting speed has been improved significantly on multi-core machines by using -

a. ○ Arrays.parallelSort

b. ○ Arrays.sort

c. Collection.parallelSort

d. Arrays.sortParallelly

22. Which is aggregate operation in Java 8

a. filter

b. map

c. forEach

d. All

23. Example of functional interfaces in Java 8 -

a. java.util.concurrent.Callable

b. java.lang.Runnable

c. All

d. None

24. Which method can be used to check null on an Optional variable in Java 8

a. isPresent()

b. isNullable()

c. isPresentable()

d. isNotNull()

25. We need to override which Predicate method in Java 8 -

a. predict(T t)

b. predictable(T t)

c. ○ testable(T t)

d. ○ test(T t)

Submit Quiz 1

Quiz 1 - Correct answers

EASY
1) b
2) d
3) a
4) b
5) b

MEDIUM
6) c
7) b
8) c
9) a
10) d
11) c
12) d
13) b
14) a
15) b

HARD
16) c
17) a
18) b

19) a
20) c
21) a
22) d
23) c
24) a
25) d**Q1 - Q5, 5 EASY level difficulty questions 1 mark each. 1 * 5 = 5 marks**

1. Functional interfaces can be annotated as

a. ○ @Function
b. ○ @FunctionalInterface
c. ○ @Functional
d. ○ @Interface

2. Functional interfaces method is

a. ○ static
b. ○ abstract
c. ○ final
d. ○ none

3. What needs to be implemented to use lambda expression

a. ○ Functional interface
b. ○ Functional class
c. ○ Functional method
d. ○ Functional object

4. Functional interface methods can be declared as

a. static

b. abstract

c. final

d. All

5. What will be output of following code -

a. compilation error

b. Runtime Exception

c. Compile and execute without any Exception or error

d. None

Q6 - Q15, 10 MEDIUM level difficulty questions 2 mark each. 2 * 10 = 20 marks

6. void accept(T t) is method of -

a. Consumer

b. Producer

c. Both

d. None

7. Which of these does Stream filter() operates on

a. Predicate

b. Interface

c. ○ Class

d. ○ Methods

8. Which of these does Stream map() operates on

a. ○ Class

b. ○ Interface

c. ○ Predicate

d. ○ Function

9. Which of these does forEach() operates on

a. ○ Methods

b. ○ Consumer

c. ○ Producer

d. ○ Predicate

10. A pipeline is a sequence of what operations in java 8

a. ○ multi-threading

b. ○ concurrent

c. ○ consequent

d. ○ stream

11. How can we obtain source of objects in java 8?

a. ○ Stream stream()

b. ○ Stream obtain()

c. ○ Stream obtainSource()

d. ○ All

12. What will be output of following program -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

13. What will be output of following program -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

14. What will be output of following code -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

15. What will be output of following code -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

Q16 - Q25, 10 HARD level difficulty questions 3 mark each. 3 * 10 = 30 marks

16. What will be output of following code -

a. ○ Strings

b. ○ Integers

c. ○ Characters

d. ○ Float

17. Lambda expressions are ___________ scoped

a. ○ Lexically

b. ○ Semantically

c. ○ Binary

d. ○ None

18. On which of these does annotations can be used on in Java 8

a. ○ Local variables

b. ○ Super classes

c. ○ Generic types

d. ○ All of these

19. What will be output of following code -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

20. What will be output of following code -

a. ○ compilation error

b. ○ compilation error

c. ○ Compile and execute without any Exception or error

d. ○ None

21. What will be output of following code -

a. ○ output may be 125

b. ○ output may be 695

c. ○ output may be 1511

d. ○ output may be 456

22. What will be output of following code -

a. Full Name: [none]

b.

c. compilation error

d. compilation error

23. What will be output of following code -

a. compilation error

b. Runtime Exception

c. Compile and execute without any Exception or error

d. None

24. What will be output of following code -

a. compilation error

b. Runtime Exception

c. Compile and execute without any Exception or error

d. None

25. What will be output of following code -

a. ○ compilation error

b. ○ Runtime Exception

c. ○ Compile and execute without any Exception or error

d. ○ None

Submit Quiz 2

Quiz 2 - Correct answers

EASY

1) b

2) b

3) a

4) d

5) c

MEDIUM

6) a

7) a

8) d

9) b

10) d

11) a

12) c

13) b

14) c

15) c

HARD

16) b
17) a
18) d
19) c
20) c
21) b
22) a
23) c
24) c
25) a