Explore More

Subcription : Premium CDAC NOTES & MATERIAL @99

Contact to Join
Premium Group

Click to Join
Telegram Group

For More E-Notes

Join Our Community to stay Updated

TAP ON THE ICONS TO JOIN!

	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql

51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9cIHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

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", "Washi
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. What is a lambda expression in Java?

a) A way to define anonymous methods

b) A method with no return type

c) A method that is only available in interfaces

d) A class that implements an interface

Click to View Answer and Explanation

Answer:

a) A way to define anonymous methods

Explanation:

Lambda expressions in Java are a way to define anonymous methods (functions) that can be passed as arguments, stored in variables, or returned as values.

2. Which Java version introduced lambda expressions?

a) Java 6

b) Java 7

c) Java 8

d) Java 9

Click to View Answer and Explanation

Answer:

c) Java 8

Explanation:

Lambda expressions were introduced in Java 8, bringing functional programming features to the language.

3. What is the Stream API in Java?

a) A feature for handling files in Java

b) A way to process sequences of elements in a functional style

c) A method for sorting arrays

d) An interface for input/output operations

Click to View Answer and Explanation

Answer:

b) A way to process sequences of elements in a functional style

Explanation:

The Stream API in Java allows processing sequences of elements in a functional style, enabling operations like filtering, mapping, and reducing.

4. Which method is used to create a stream from a collection in Java?

a) stream()

b) createStream()

c) asStream()

d) toStream()

Click to View Answer and Explanation

Answer:

a) stream()

Explanation:

The stream() method is used to create a stream from a collection, such as a List or Set.

5. What does the Optional class in Java represent?

a) A way to represent null values

b) A container that may or may not contain a value

c) A collection of elements

d) A method that returns multiple values

Click to View Answer and Explanation

Answer:

b) A container that may or may not contain a value

Explanation:

The Optional class in Java is a container object which may or may not contain a non-null value. It helps in avoiding null pointer exceptions.

6. Which functional interface is commonly used with lambda expressions in Java?

a) Runnable

b) Callable

c) Function

d) Serializable

Click to View Answer and Explanation

Answer:

c) Function

Explanation:

The Function interface is a common functional interface used with lambda expressions in Java. It represents a function that takes one argument and returns a result.

7. What is the syntax for a basic lambda expression in Java?

a) (parameters) -> expression

b) (expression) -> parameters

c) {expression} -> parameters

d) (parameters) => expression

Click to View Answer and Explanation

Answer:

a) (parameters) -> expression

Explanation:

The basic syntax for a lambda expression in Java is (parameters) -> expression. It can also have a block of code if needed.

8. Can lambda expressions have multiple parameters?

a) Yes, they can have multiple parameters

b) No, they can only have one parameter

c) Yes, but only if they return a value

d) No, they cannot have any parameters

Click to View Answer and Explanation

Answer:

a) Yes, they can have multiple parameters

Explanation:

Lambda expressions in Java can have multiple parameters, enclosed in parentheses and separated by commas.

9. What does the Stream API's filter() method do?

a) It filters elements of a stream based on a predicate

b) It maps each element to a collection

c) It reduces elements of a stream

d) It sorts elements in a stream

Click to View Answer and Explanation

Answer:

a) It filters elements of a stream based on a predicate

Explanation:

The filter() method in the Stream API is used to filter elements of a stream based on a given predicate.

10. What is the purpose of the forEach() method in the Stream API?

a) To count the elements in the stream

b) To iterate over each element in the stream and perform an action

c) To remove duplicates from the stream

d) To sort the elements in the stream

Click to View Answer and Explanation

Answer:

b) To iterate over each element in the stream and perform an action

Explanation:

The forEach() method is a terminal operation that iterates over each element in the stream and performs the specified action.

11. What is a default method in an interface in Java?

a) A method with no implementation

b) A method that must be overridden

c) A method with a default implementation in the interface

d) A method that is called first in a class

Click to View Answer and Explanation

Answer:

c) A method with a default implementation in the interface

Explanation:

A default method in an interface is a method with a default implementation. It allows interfaces to evolve without breaking existing implementations.

12. Which method is used to check if a stream is empty?

a) isEmpty()

b) noneMatch()

c) count()

d) allMatch()

Click to View Answer and Explanation

Answer:

b) noneMatch()

Explanation:

The noneMatch() method checks if no elements in the stream match a given predicate, which can be used to check if the stream is empty.

13. What is the purpose of the Predicate functional interface in Java?

a) To return an object

b) To perform an action on an object

c) To test a condition on an object

d) To create an object

Click to View Answer and Explanation

Answer:

c) To test a condition on an object

Explanation:

The Predicate functional interface in Java is used to test a condition on an object and returns a boolean value.

14. Can lambda expressions be used with the Comparator interface?

a) Yes, lambda expressions can be used with Comparator

b) No, lambda expressions cannot be used with Comparator

c) Yes, but only with Integer objects

d) No, they are only used with Function interface

Click to View Answer and Explanation

Answer:

a) Yes, lambda expressions can be used with Comparator

Explanation:

Lambda expressions can be used with the Comparator interface to simplify comparison logic.

15. What is the use of the Optional class in Java?

a) To handle collections of data

b) To represent the presence or absence of a value

c) To throw exceptions

d) To perform arithmetic operations

Click to View Answer and Explanation

Answer:

b) To represent the presence or absence of a value

Explanation:

The Optional class in Java is used to represent the presence or absence of a value, helping to avoid null pointer exceptions.

16. Which method is used to retrieve a value from an Optional if it is present?

a) getValue()

b) retrieve()

c) get()

d) fetch()

Click to View Answer and Explanation

Answer:

c) get()

Explanation:

The get() method is used to retrieve a value from an Optional if it is present.

17. What is the difference between map() and flatMap() in the Stream API?

a) map() transforms elements, flatMap() flattens nested structures

b) map() filters elements, flatMap() maps them

c) map() reduces elements, flatMap() sorts them

d) There is no difference between map() and flatMap()

Click to View Answer and Explanation

Answer:

a) map() transforms elements, flatMap() flattens nested structures

Explanation:

The map() method transforms elements in a stream, while flatMap() flattens nested structures such as streams of streams into a single stream.

18. Can streams be parallelized in Java?

a) Yes, using the parallelStream() method

b) No, streams cannot be parallelized

c) Yes, but only for numeric streams

d) No, only arrays can be parallelized

Click to View Answer and Explanation

Answer:

a) Yes, using the parallelStream() method

Explanation:

Streams can be parallelized in Java using the parallelStream() method, which processes elements concurrently on multiple cores.

19. What is the Collectors class in the Stream API used for?

a) To collect elements from a stream into a collection

b) To map elements in a stream

c) To filter elements in a stream

d) To sort elements in a stream

Click to View Answer and Explanation

Answer:

a) To collect elements from a stream into a collection

Explanation:

The Collectors class in the Stream API is used to collect elements from a stream into a collection, such as a List or Set.

20. Which method in the Optional class is used to return a default value if the value is absent?

a) orElse()

b) ifPresent()

c) getOrDefault()

d) fetch()

Click to View Answer and Explanation

Answer:

a) orElse()

Explanation:

The orElse() method in the Optional class returns a default value if the value is absent.

21. Which new date and time API was introduced in Java 8?

a) java.util.Date

b) java.sql.Date

c) java.time

d) java.date

Click to View Answer and Explanation

Answer:

c) java.time

Explanation:

The java.time package introduced in Java 8 provides a new date and time API that is more comprehensive and user-friendly compared to the older java.util.Date and java.sql.Date classes.

22. What does the map() method do in the Stream API?

a) It transforms each element in the stream

b) It maps elements to a new collection

c) It reduces elements to a single value

d) It filters elements in the stream

Click to View Answer and Explanation

Answer:

a) It transforms each element in the stream

Explanation:

The map() method in the Stream API transforms each element in the stream using a given function, producing a new stream of transformed elements.

23. Which method is used to check if all elements in a stream match a given predicate?

a) allMatch()

b) anyMatch()

c) noneMatch()

d) matchesAll()

Click to View Answer and Explanation

Answer:

a) allMatch()

Explanation:

The allMatch() method is used to check if all elements in a stream match the given predicate.

24. Can streams in Java be reused?

a) Yes, streams can be reused multiple times

b) No, streams cannot be reused once they are consumed

c) Yes, but only parallel streams

d) Yes, if they are collected first

Click to View Answer and Explanation

Answer:

b) No, streams cannot be reused once they are consumed

Explanation:

Streams in Java cannot be reused once they are consumed. After a terminal operation is performed, the stream is considered closed and cannot be used again.

25. Which method in the Stream API is used to remove duplicate elements?

a) filter()

b) distinct()

c) unique()

d) removeDuplicates()

Click to View Answer and Explanation

Answer:

b) distinct()

Explanation:

The distinct() method in the Stream API is used to remove duplicate elements from a stream, ensuring that each element appears only once.

26. Which method is used to sort elements in a stream?

a) sort()

b) order()

c) sorted()

d) arrange()

Click to View Answer and Explanation

Answer:

c) sorted()

Explanation:

The sorted() method in the Stream API is used to sort the elements in a stream either in natural order or using a custom comparator.

27. Which method in the Stream API is used to combine elements of a stream into a single result?

a) map()

b) reduce()

c) combine()

d) collect()

Click to View Answer and Explanation

Answer:

b) reduce()

Explanation:

The reduce() method in the Stream API is used to combine elements of a stream into a single result using an accumulator function.

28. What does the findFirst() method do in a stream?

a) It finds the first element in the stream and returns it

b) It finds the first element matching a condition

c) It returns the first element or throws an exception

d) It returns the first non-null element

Click to View Answer and Explanation

Answer:

a) It finds the first element in the stream and returns it

Explanation:

The findFirst() method in the Stream API finds the first element in a stream and returns it wrapped in an Optional if it exists.

29. Which method is used to get the count of elements in a stream?

a) size()

b) count()

c) getCount()

d) length()

Click to View Answer and Explanation

Answer:

b) count()

Explanation:

The count() method in the Stream API returns the number of elements in the stream.

30. Can you perform parallel processing with the Stream API in Java?

a) Yes, using parallelStream()

b) No, the Stream API does not support parallel processing

c) Yes, but only with List collections

d) No, only arrays support parallel processing

Click to View Answer and Explanation

Answer:

a) Yes, using parallelStream()

Explanation:

You can perform parallel processing with the Stream API in Java using the parallelStream() method, which processes elements concurrently on multiple cores.

31. What is the difference between findFirst() and findAny() in the Stream API?

a) findFirst() returns the first element, findAny() returns any element

b) findFirst() is for sequential streams, findAny() is for parallel streams

c) findFirst() returns a non-null element, findAny() returns any element

d) There is no difference

Click to View Answer and Explanation

Answer:

a) findFirst() returns the first element, findAny() returns any element

Explanation:

findFirst() returns the first element in a stream, while findAny() returns any element in the stream, potentially more efficient in parallel streams.

32. Which method is used to skip a given number of elements in a stream?

a) drop()

b) skip()

c) omit()

d) jump()

Click to View Answer and Explanation

Answer:

b) skip()

Explanation:

The skip() method in the Stream API skips the first n elements and returns a stream containing the remaining elements.

33. Which new interface was introduced in Java 8 to represent a function that takes no arguments and returns a result?

a) Function

b) Supplier

c) Consumer

d) Predicate

Click to View Answer and Explanation

Answer:

b) Supplier

Explanation:

The Supplier interface was introduced in Java 8 to represent a function that takes no arguments and returns a result.

34. Can lambda expressions be used with the Runnable interface?

a) Yes, lambda expressions can be used with Runnable

b) No, lambda expressions cannot be used with Runnable

c) Yes, but only for single-threaded applications

d) No, lambda expressions are only used with functional interfaces

Click to View Answer and Explanation

Answer:

a) Yes, lambda expressions can be used with Runnable

Explanation:

Lambda expressions can be used with the Runnable interface to simplify the syntax of creating threads or tasks in Java.

35. What is the main purpose of the Stream API in Java?

a) To perform I/O operations

b) To handle exceptions

c) To process collections of data in a functional style

d) To create multi-threaded applications

Click to View Answer and Explanation

Answer:

c) To process collections of data in a functional style

Explanation:

The Stream API in Java is designed to process collections of data in a functional style, enabling operations like filtering, mapping, and reducing.

36. What is the purpose of the peek() method in the Stream API?

a) To modify the elements of a stream

b) To view elements of a stream without modifying them

c) To remove elements from a stream

d) To sort the elements of a stream

Click to View Answer and Explanation

Answer:

b) To view elements of a stream without modifying them

Explanation:

The peek() method is used to view elements of a stream without modifying them. It is often used for debugging purposes.

37. Which method is used to limit the number of elements in a stream?

a) limit()

b) reduce()

c) cap()

d) filter()

Click to View Answer and Explanation

Answer:

a) limit()

Explanation:

The limit() method is used to limit the number of elements in a stream to the specified size.

38. What is the difference between a stream() and a parallelStream() in Java?

a) stream() is faster than parallelStream()

b) parallelStream() processes elements in parallel

c) stream() uses more memory

d) parallelStream() is only for arrays

Click to View Answer and Explanation

Answer:

b) parallelStream() processes elements in parallel

Explanation:

The parallelStream() method processes elements in parallel, potentially improving performance on multi-core processors.

39. Which method is used to find the first element in a stream?

a) findFirst()

b) firstElement()

c) getFirst()

d) find()

Click to View Answer and Explanation

Answer:

a) findFirst()

Explanation:

The findFirst() method is used to find the first element in a stream and returns an Optional containing the element if it exists.

40. Can lambda expressions be used to create threads in Java?

a) Yes, using the Runnable interface

b) Yes, using the Callable interface

c) No, lambda expressions cannot be used to create threads

d) Yes, using the Supplier interface

Click to View Answer and Explanation

Answer:

a) Yes, using the Runnable interface

Explanation:

Lambda expressions can be used to create threads in Java by passing them to the Runnable interface, which is then passed to a Thread.

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project