	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

Exception Handling In Java MCQ

Q#1. Which of the following is incorrect about try-with-resources in Java?

```
(a) try-with-resources was introduced in Java 7
(b) We don't need to use finally block, if we use try-with-resources

(c) The  try-with-resources statement ensures that each resource is closed at the e

(d) It increases the complexity of the code
```

Answer: d) It increases the complexity of the code

Explanation: The try-with-resources was introduced in Java 7. When we use try-with-resources, the resources will automatically be closed. Hence, we don't need to use finally block. It reduces the complexity of the code and even reduces the lines of code.

Q#2. Which of the following scenarios is best suited for utilizing the try-with-resources statement?

```
(a) Handling common runtime exceptions
(b) Implementing custom exception classes
(c) Working with IO operations involving streams
(d) Synchronizing multi-threaded operations
```

Answer: c) Working with IO operations involving streams

Explanation: From the mentioned options, try-with-resources statement will be best suited when working with IO operations involving streams, such as reading from or writing to files. It ensures that the streams are automatically closed after usage, reducing the risk of resource leaks and improving code reliability.

Q#3. What is the output of the following code snippet?

```
try {
      throw new RuntimeException("Error");
} catch (Exception e) {
        System.out.println(e.getMessage());
    }
```

```
    (a) Error
    (b) RuntimeException
    (c) null
    (d) The code will not compile.
```

Answer: a) Error

Explanation: The code explicitly throws a RuntimeException with the message "Error". The catch block catches the exception and prints the error message using getMessage().

Q#4. Which of the following statement(s) is/are correct about multi-catch statement?

```
    (a) A single  catch block can handle more than one type of exception.
    (b) A multi-catch statement is valid in Java 7 and later.
    (c) If a  catch block handles more than one exception type, then the  catch paramet
    (d) Alternatives in a multi-catch statement cannot be related by subclassing.
```

Answer: (a), (b), (c), (d)

Explanation: All statements are correct about a multi-catch statement since Java 7.

Q#5. Which of the following is the correct syntax for catching multiple exceptions in a single catch block?

```
    (a) catch (ExceptionType1 || ExceptionType2 || ExceptionType3 ex)
    (b) catch (ExceptionType1, ExceptionType2, ExceptionType3 ex)
```

```
(c) catch (ExceptionType1 && ExceptionType2 && ExceptionType3 ex)
(d) catch (ExceptionType1 | ExceptionType2 | ExceptionType3 ex)
```

Answer: d) catch (ExceptionType1 | ExceptionType2 | ExceptionType3 e)

Explanation: Multiple exceptions can be caught in a single catch block using the single pipe symbol (|) to separate the exception types.

Q#6. What is wrong with the following code snippet in the context of try-with-resources?

```
static String readFirstLineFromFile() throws IOException {

    try (FileReader fr = new FileReader("");
         BufferedReader br = new BufferedReader(fr)) {
        fr= new FileReader("xyz.txt");
        return br.readLine();
    }
}
```

```
(a) There is no catch block after the try block.
(b) The re-assignment of variable 'fr' is not allowed.
(c) Two resources can't be declared in a try block.
(d) The throws clause is not needed.
```

Answer: b) The re-assignment of variable ‘fr’ is not allowed.

Explanation: The resources declared in a try-with-resources context are final by default, hence we can’t re-assign them. There will be a compilation error at the same line.

Q#7. Which of the following statements is true about exceptions in the context of Java 7 and later versions?

```
(a) A try block must be followed by either a catch block or a finally block.
(b) In order to close the resources opened in try block, it is mandatory to includ
(c) Multiple types of exceptions can be handled by including multiple catch blocks
(d) It is not mandatory to include a catch block or finally block after a try bloc
```

Answer: d) It is not mandatory to include a catch block or finally block after a try block.

Explanation: Checked exceptions must be caught or declared to be thrown by the method that can potentially throw them.

Q#8. Which exception will be thrown by parseInt() method in Java?

```
(a) IntegerOutOfBoundException
(b) IntegerFormatException
(c) ArithmeticException
(d) NumberFormatException
```

Answer: d) NumberFormatException

Explanation: parseInt() method parses input into integer. This method will throw NumberFormatException.

Q#9. Which of the following exception must be either caught or declared to be thrown in Java?

```
(a) NullPointerException
(b) ArrayIndexOutOfBoundsException
(c) FileNotFoundException
(d) ArithmeticException
```

Answer: c) FileNotFoundException

Explanation: FileNotFoundException is a checked exception in Java, hence it must be either caught or declared to be thrown.

Q#10. What is the output of the following code snippet?

```
try {
    throw new NullPointerException();
} catch (RuntimeException e) {
    System.out.println("RuntimeException");
} catch (Exception e) {
    System.out.println("Exception");
}
```

```
(a) RuntimeException
(b) Exception
(c) NullPointerException
(d) The code will not compile.
```

Answer: a) RuntimeException

Explanation: The code explicitly throws a NullPointerException, which is a subclass of RuntimeException. Since the catch block for RuntimeException is defined first, it is executed.

Q#11. Which of the following is true about the catch block in Java?

```
(a) A catch block can catch multiple types of exceptions using the semicolon (;).
(b) A catch block can catch multiple types of exceptions using the logical AND ope
(c) A catch block can catch multiple types of exceptions using multiple catch stat
(d) A catch block can only catch one type of exception at a time.
```

Answer: c) A catch block can catch multiple types of exceptions using multiple catch statements.

Explanation: Traditionally, multiple types of exceptions can be caught in a catch block by using multiple catch statements, each catching a different exception type. In contrast, a multi-catch statement can catch multiple exceptions in a single catch block since Java 7.

Q#12. Which statement is used to catch and handle multiple exceptions in a single catch block?

```
(a) catch-all
(b) multi-catch
(c) exception-catch
(d) exception-all
```

Answer: b) multi-catch

Explanation: The multi-catch statement in Java allows catching and handling multiple exceptions in a single catch block.

Q#13. Which keyword will you use to specify that a method can potentially throw an exception?

```
(a) try
(b) catch
(c) throw
(d) throws
```

Answer: d) throws

Explanation: We use the throws keyword in a method declaration to state that the method can potentially throw one or more exceptions.

Q#14. What is the purpose of the finally block in exception handling?

```
(a) To catch and handle exceptions.
(b) To specify that a method can potentially throw an exception.
(c) To execute code regardless of whether an exception is thrown or not.
(d) To explicitly throw an exception.
```

Answer: c) To execute code regardless of whether an exception is thrown or not.

Explanation: The finally block is used to specify code that should be executed regardless of whether an exception is thrown or not.

Q#15. Which of the following statements is true about the catch block in exception handling?

```
(a) A try block can have multiple catch blocks.
(b) A catch block can have multiple try blocks.
(c) A catch block must always be followed by a finally block.
(d) A catch block cannot be used without a try block.
```

Answer: a) A try block can have multiple catch blocks.

Explanation: A try block can have multiple catch blocks to handle different types of exceptions.

Q#16. Which of the following statements is true about the finally block in exception handling?

```
(a) A finally block is always executed before a catch block.
(b) A finally block is always executed after a catch block.
(c) A finally block is only executed if an exception occurs.
(d) A finally block is optional and can be omitted.
```

Answer: b) A finally block is always executed after a catch block.

Explanation: A finally block is always executed after a catch block, regardless of whether an exception occurs or not.

Q#17. Which of the following is a subclass of the Exception class?

```
(a) RuntimeError
(b) Error
(c) Throwable
(d) StackOverflowError
```

Answer: a) RuntimeError

Explanation: RuntimeError is a subclass of the Exception class in Java.

Q#18. Which of the following statements is true about the finally block?

```
(a) The finally block is required for every try-catch statement.
(b) The finally block is optional and can be omitted.
(c) The finally block is executed only if an exception occurs.
(d) The finally block is executed only if a catch block is present.
```

Answer: b) The finally block is optional and can be omitted.

Explanation: The finally block is optional and can be omitted in exception handling.

Q#19. What is the output of the following code snippet?

```java
try {
    int[] array = new int[5];
    System.out.println(array[5]);
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("ArrayIndexOutOfBoundsException");
} finally {
    System.out.println("Finally block executed.");
}
```

```
(a) ArrayIndexOutOfBoundsException
     Finally block executed.
(b) ArrayIndexOutOfBoundsException
(c) Finally block executed.
(d) The code will not compile.
```

Answer: a) ArrayIndexOutOfBoundsException
Finally block executed.

Explanation: The code attempts to access an element at index 5 in the array which is not available. Hence it raises an ArrayIndexOutOfBoundsException. The catch block is executed, and then the finally block is executed.

Q#20. Which of the following is not a subclass of Throwable in Java?

```
(a) Checked exception
(b) Unchecked exception
(c) Fatal exception
(d) Error
```

Answer: c) Fatal exception

Explanation: “Fatal exception” is not a recognized subclass of Throwable in Java. Error is a direct subclass of Throwable.

Q#21. Which of the following is/are correct statement(s) about Unchecked Exceptions in Java?

```
(a) These exceptions occur during the execution of the program.
(b) They are also referred to as Runtime exceptions.
(c) These exceptions are generally ignored during the compilation process.
(d) They are checked while compiling the program.
```

Answer: (a), (b), (c)

Explanation: They are not checked while compiling the program.

Q#22. Which of the following is an incorrect statement about checked Exceptions in Java?

```
(a) Checked exceptions are called compile-time exceptions.
(b) They are subtypes of RuntimeException.
(c) These exceptions are checked at compile-time by the compiler.
(d) The IOException is a type of checked Exception.
```

Answer: b) They are subtypes of RuntimeException

Explanation: They are not subtypes of RuntimeException, but direct subtypes of Exception.

Q#23. In which of the below classes the printStackTrace() method defined?

```
(a) Exception.
(b) RuntimeException.
(c) Throwable.
(d) Error.
```

Answer: c) Throwable.

Explanation: The printStackTrace() method is defined in the Throwable class.

Q#24. Which of the following is a not a subclass of the Error class directly or indirectly?

```
(a) RuntimeError
(b) InternalError
(c) StackOverflowError
(d) OutOfMemoryError
```

Answer: a) RuntimeError

Explanation: RuntimeError is not a subclass of the Error class directly or indirectly in Java.

Q#25. What is the output of the following code snippet?

```
try {
    throw new Exception("Custom exception");
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

```
(a) Custom exception
(b) Exception
```

```
(c) null
(d) The code will not compile.
```

Answer: a) Custom exception

Explanation: The code explicitly throws an Exception with the message “Custom exception”. The catch block catches the exception and prints the error message using getMessage().

Q#26. Which of the following statements is true about custom exception classes in Java?

```
(a) Custom exception classes must extend  the Throwable class.
(b) Custom exception classes must extend  the RuntimeException class.
(c) Custom exception classes must extend the Exception class.
(d) Custom exception classes are not required to extend or implement any class or
```

Answer: a) Custom exception classes must extend the Exception class.

Explanation: In Java, custom exception classes must be subclasses of the Exception class or one of its subclasses.

Q#27. Which of the following is not a type of valid construct in exception handling?

```
(a) try-catch-finally
(b) try-finally
(c) catch-finally
(d) try-catch
```

Answer: c) catch-finally

Explanation: “catch-finally” is not a valid exception handling construct in Java. The correct construct is “try-catch” or “try-catch-finally” or “try-finally”.

Q#28. Which of the following is not a checked exception in Java?

```
(a) IOException
(b) FileNotFoundException
(c) NullPointerException
(d) ClassNotFoundException
```

Answer: c) NullPointerException

Explanation: NullPointerException is an unchecked exception in Java.

Q#29. What is the output of the following code snippet?

```
try {
    throw new Error("Fatal error");
} catch (Exception e) {
    System.out.println("Exception");
} catch (Error e) {
    System.out.println("Error");
}
```

```
(a) Exception
(b) Error
(c) Compiler error
(d) The code will not compile.
```

Answer: b) Error

Explanation: The code explicitly throws an Error with the message “Fatal error”. Since Error is a subclass of Throwable, it matches the catch block for Error, and “Error” is printed.

Q#30. Which of the following exceptions is not a subclass of the RuntimeException class?

```
(a) NullPointerException
(b) ArrayIndexOutOfBoundsException
```

```
(c) IOException
(d) ArithmeticException
```

Answer: c) IOException

Explanation: IOException is not a subclass of the RuntimeException class. It is a checked exception in Java.

Q#31. Which of the following statements is true about the try-with-resources statement in Java?

```
(a) It is used to handle multiple exceptions in a single catch block.
(b) It is used to specify that a method can potentially throw an exception.
(c) It is used to automatically close resources after usage.
(d) It is used to define custom exception classes.
```

Answer: c) It is used to automatically close resources after usage.

Explanation: The try-with-resources statement in Java is used to automatically close resources after usage, ensuring that resources are properly managed and released.

Q#32. Which of the following statements is true about handling exceptions in multi-threaded Java applications?

```
(a) Each thread should handle exceptions independently.
(b) Exceptions thrown by a thread cannot be caught by other threads.
(c) A separate exception handler should be defined for each thread.
(d) Exceptions in multi-threaded applications are handled automatically by the JVM
```

Answer: c) A separate exception handler should be defined for each thread.

Explanation: In multi-threaded Java applications, it is recommended to define a separate exception handler for each thread to handle exceptions specific to that thread.

Q#33. What is the output of the following code snippet?

```
try {
    throw new Exception("First Exception");
} catch (Exception e) {
    try {
        throw new Exception("Second Exception");
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
   }
}
```

```
(a) First Exception
(b) Second Exception
(c) First Exception followed by Second Exception
(d) Second Exception followed by First Exception
```

Answer: b) Second Exception

Explanation: The code throws the first exception, catches it, and then throws the second exception, which is caught and its message is printed.

Q#34. What is the output of the following code snippet?

```
try {
    throw new Error();
} catch (Throwable t) {
    System.out.println(t.getClass().getSimpleName());
}
```

```
(a) Error
(b) Throwable
```

```
(c) Exception
(d) The code will not compile.
```

Answer: a) Error

Explanation: The code explicitly throws an Error, which is caught by the catch block. The getClass().getSimpleName() method is used to retrieve the simple name of the caught exception's class.

Q#35. Which of the following is the correct syntax for using the try-with-resources statement?

```
(a) try [Resource r =new Resource()] { // code }
(b) try (Resource r = new Resource(); // code )
(c) try { Resource r = new Resource(); // code }
(d) try (Resource r = new Resource()) // code
```

Answer: d) try (Resource r = new Resource()) // code

Explanation: Option (d) is the correct syntax.

Q#36. Which of the following interfaces must be implemented by a resource in order to be used with the try-with-resources statement?

```
(a) Closeable
(b) AutoCloseable
(c) Resource
(d) Disposable
```

Answer: b) AutoCloseable

Explanation: Resources used with the try-with-resources statement must implement the AutoCloseable interface, which provides the close() method for releasing system resources held by the resource.

Q#37. Which method of AutoCloseable interface is called internally in the try-with-resources statement?

```
(a) clean()
(b) refresh()
(c) close()
(d) release()
```

Answer: c) close()

Explanation: The close() method in a resource class is responsible for releasing system resources held by the resource, such as closing file streams or network connections.

Q#38. Which of the following statements is true regarding the order of closing resources in a try-with-resources statement?

```
(a) Resources are closed in the order of declaration within the try block.
(b) Resources are closed in the reverse order of declaration within the try block.
(c) Resources are closed randomly.
(d) The order of closing resources does not matter.
```

Answer: b) Resources are closed in the reverse order of declaration within the try block.

Explanation: In a try-with-resources statement, resources are closed in the reverse order of their declaration within the try block. This ensures that resources are properly closed, even if an exception occurs.

Q#39. What happens if an exception is thrown both during resource initialization and within the try block of a try-with-resources statement?

```
(a) The exception thrown during resource initialization takes precedence.
(b) The exception thrown within the try block takes precedence.
(c) Both exceptions are caught and handled.
(d) Only the exception thrown within the try block is caught and handled.
```

Answer: a) The exception thrown during resource initialization takes precedence.

Explanation: If an exception is thrown both during resource initialization and within the try block, the exception thrown during resource initialization takes precedence. The exception thrown within the try block is added as a suppressed exception.

Q#40. What is the advantage of using the try-with-resources statement instead of a traditional try-catch-finally approach?

```
    (a) It reduces boilerplate code.
    (b) It ensures proper resource cleanup without explicitly writing a finally block.
    (c) It simplifies complex exception handling.
    (d) It improves the performance of exception handling significantly.
```

Answer: b) It ensures proper resource cleanup without explicitly writing a finally block.

Explanation: The try-with-resources statement ensures proper resource cleanup by automatically calling the close() method on the resource, eliminating the need to explicitly write a finally block. It improves code readability and reduces the risk of resource leaks.

Collection MCQ in Java with Answers Explained

Q#1. Which among the following has elements in insertion order?

```
A) HashMap
B) TreeMap
C) SortedMap
D) LinkedHashMap
```

Answer: D

Explanation: LinkedHashMap maintains the order of elements based on their insertion order. When iterating through a LinkedHashMap, the elements are returned in the order they were inserted. HashMap does not maintain any order, TreeMap maintains elements sorted according to their natural ordering or a specified comparator, and SortedMap is an interface that TreeMap implements to provide a sorted map, not necessarily in insertion order.

Q#2. A List in Java accesses any element from its ___?

```
A) key
B) index
C) value
D) position
```

Answer: B

Explanation: A List in Java accesses any element from its index. Lists are ordered collections that allow for precise control over where each element is inserted. They can be accessed by their integer index, which starts from 0 for the first element and increases consecutively.

Q#3. Which among the following allow duplicate elements?

```
A) ArrayList
B) HashSet
C) LinkedHashSet
D) TreeSet
```

Answer: A

Explanation: ArrayList allows duplicate elements. It is an ordered collection (also known as a sequence) and permits all elements, including duplicates. In contrast, Since HashSet, LinkedHashSet, and TreeSet are implementations of the Set interface, they do not allow duplicate elements.

Q#4. Which class in the collection framework has its implementation based on a balanced tree data structure?

```
A) HashMap
B) LinkedList
C) TreeMap
D) ArrayList
```

Answer: C

Explanation: TreeMap has its implementation based on a balanced tree data structure. Specifically, it uses a Red-Black tree to store its key-value pairs, which ensures that the map is sorted according to the natural ordering of its keys or by a specified comparator. This provides log(n) time complexity for basic operations such as get, put, and remove.

Q#5. Which one of these classes provides the features of maintaining insertion order and allowing null elements?

```
A) HashSet
B) LinkedHashSet
```

```
C) TreeSet
D) PriorityQueue
```

Answer: B

Explanation: LinkedHashSet provides the features of maintaining insertion order and allowing null elements. It combines the unique element property of a HashSet with the insertion order property of a LinkedList. HashSet does not maintain insertion order, TreeSet does not allow null elements and sorts elements according to their natural order, and PriorityQueue orders elements based on their priority and does not guarantee insertion order.

Q#6. Which one of the following statements is NOT true about the Collection interface?

```
A) The Collection interface is the root interface of the Java Collections Framework
B) The Collection interface extends the Iterable interface.
C) The Collection interface includes methods for accessing elements by index.
D) The Collection interface can be used to represent groups of objects.
```

Answer: C

Explanation: The Collection interface does not include methods for accessing elements by index. Methods for accessing elements by index are provided by the List interface, which is a subinterface of Collection. The Collection interface is the root interface of the Java Collections Framework, it extends the Iterable interface, and it can be used to represent groups of objects.

Q#7. Which of these interfaces must contain unique elements?

```
A) List
B) Queue
C) Set
D) Deque
```

Answer: C

Explanation: The Set interface must contain unique elements. This means no duplicates are allowed in a Set. In contrast, the List interface allows duplicates, the Queue interface does not enforce uniqueness, and the Deque (double-ended queue) also does not require elements to be unique.

Q#8. You need to store elements in a collection that guarantees no duplicates. Which collection should you use?

```
A) ArrayList
B) HashSet
C) LinkedList
D) PriorityQueue
```

Answer: B

Explanation: HashSet guarantees that no duplicates will be stored in the collection. It is an implementation of the Set interface, which enforces uniqueness of its elements. ArrayList and LinkedList allow duplicates, and PriorityQueue does not enforce uniqueness of its elements.

Q#9. Which collection type preserves the insertion order of key-value pairs?

```
A) HashMap
B) TreeMap
C) LinkedHashMap
D) SortedMap
```

Answer: C

Explanation: LinkedHashMap preserves the insertion order of key-value pairs. This means that when you iterate over the entries in a LinkedHashMap, they will appear in the order they were inserted. HashMap does not guarantee any specific order, TreeMap sorts the entries based on their keys, and SortedMap is an interface for maps that maintain a sorted order, typically implemented by TreeMap.

Q#10. Which of these is not an interface in the Collections Framework?

```
A) List
B) Set
C) ArrayList
D) None of the above
```

Answer: C

Explanation: ArrayList is not an interface in the Collections Framework; it is a class. List, and Setare interfaces in the Collections Framework. List is used for ordered collections, Set is used for collections that do not allow duplicates.

Q#11. Which of these interfaces declares core methods that all collections will have?

```
A) List
B) Set
C) Collection
D) Map
```

Answer: C

Explanation: The Collection interface declares core methods that all collections will have. It is the root interface in the Java Collections Framework hierarchy. Interfaces like List, Set, and Queue extend the Collection interface and add specific behaviors. Map, on the other hand, is not a subtype of Collection and defines a different type of data structure for storing key-value pairs.

Q#12. Which one of the methods below is not defined in the Collection interface?

```
A) add
B) remove
C) get
D) size
```

Answer: C

Explanation: The method get is not defined in the Collection interface. The Collection interface provides methods like add, remove, and size for adding elements, removing elements, and getting the size of the collection, respectively. However, accessing elements by index (like get(int index)) is specific to the List interface, which extends Collection.

Q#13. Which of the below does not implement the Map interface?

```
A) HashMap
B) TreeMap
```

```
C) LinkedHashMap
D) ArrayList
```

Answer: D

Explanation: ArrayList does not implement the Map interface. It is a class in Java that implements the List interface, which is used for ordered collections and allows duplicate elements. HashMap, TreeMap, and LinkedHashMap are all classes that implement the Map interface, which is used for storing key-value pairs and provides methods for accessing, inserting, and removing elements based on keys.

Q#14. Which of the following statements are true about ArrayList and Vector in Java?

```
A) Both ArrayList and Vector are thread-safe and implement the RandomAccess interfa
B) Both ArrayList and Vector have the same initial capacity and resize strategy.
C) ArrayList is non-synchronized and Vector is synchronized, making Vector thread-s
D) There is no significant difference between ArrayList and Vector; both offer the
```

Answer: C

Explanation: Both ArrayList and Vector are dynamic arrays that can store and access elements by index. However, they differ in thread-safety and resizing behavior:

- **Thread-safety:**
  - ArrayList: Not synchronized, meaning it's not thread-safe for concurrent access from multiple threads.
  - Vector: Synchronized, making it thread-safe but potentially slower for single-threaded operations due to synchronization overhead.
- **Resizing:**
  - ArrayList: When capacity is reached, it increases its size by 50% of the current size.
  - Vector: When capacity is reached, it doubles its size.

Q#15. Which of the following methods helps insert elements at a specific position in a collection within the Java Collection Framework?

```
A) add(element)
B) put(key, value)
```

```
C) addAll(elements)
D) get(index)
```

Answer: A

Explanation:

- add(element) is a general method available in most collection classes (like ArrayList, LinkedList, etc.) that allows inserting an element at the end of the collection. However, for targeted insertion at a specific position:
  - ArrayList and LinkedList provide the add(int index, element) method, where index specifies the desired insertion position.
- put(key, value) is used with Map implementations (like HashMap, TreeMap) to insert key-value pairs.
- addAll(elements) adds all elements from another collection to the current collection, not at a specific position.
- get(index) retrieves an element at a specific position but doesn't modify the collection by inserting elements.

Q#16. Which of the following methods is a new addition for Sets introduced in Java 9?

```
A) add(element)
B) contains(element)
C) isEmpty()
D) of(elements...)
```

Answer: D

Explanation:

- add(element) has existed in Sets since earlier Java versions and allows adding elements.
- contains(element) has also been available previously to check if an element exists in the Set.
- isEmpty() is another existing method to determine if the Set is empty.
- of(elements...) is a static factory method introduced in Java 9 for creating immutable Set objects. It allows concise initialization with a variable number of elements.

Q#17. Which of the following is an interface in the Java Collection Framework?

```
A) ArrayList
B) Collections
```

```
C) HashMap
D) Collection
```

Answer: D

Explanation:

- ArrayList, and HashMap are all concrete classes that implement specific interfaces within the Collection Framework.
- The Collections is a helper class that provides utility methods for working with collections (such as lists, sets, and maps).
- Collection is a fundamental interface in the Java Collection Framework that defines core operations for collections, such as adding, removing, and checking element existence. Many collection classes (like ArrayList, HashSet, etc.) inherit functionalities from the Collection interface.

Q#18. What is the relationship between HashSet and HashMap in the Java Collection Framework?

```
A) HashSet is a subclass of HashMap, and both store unique elements.
B) HashMap is a subclass of HashSet, and both store key-value pairs.
C) HashSet and HashMap are unrelated; HashSet stores unique elements, while HashMap
D) Both HashSet and HashMap use the same internal mechanism for storing elements bu
```

Answer: C

Explanation:

- HashSet and HashMap are distinct classes that utilize hashing for efficient element storage and retrieval.
- HashSet implements the Set interface and stores a collection of unique elements. It doesn't use key-value pairs.
- HashMap implements the Map interface and stores key-value pairs. Each key must be unique, and it allows efficient retrieval of values based on the key.
- While both leverage hashing, they cater to different data structures: HashSet for sets of unique elements and HashMap for key-value associations.

Q#19. Which of these classes provide an implementation of the Map interface?

```
A) ArrayList
B) TreeSet
C) HashMap
D) LinkedList
```

Answer: C

Explanation: HashMap provides an implementation of the Map interface. The Map interface is used for storing key-value pairs, and HashMap is one of its primary implementations. ArrayList and LinkedList implement the List interface, and TreeSet implements the Set interface.

Q#20. What is the difference between List and ArrayList in the Java Collection Framework?

```
A) List is a concrete class, while ArrayList is an interface.
B) ArrayList offers additional methods beyond the basic functionalities provided by
C) List allows duplicate elements, while ArrayList enforces uniqueness.
D) There's no significant difference; both List and ArrayList serve the same purpos
```

Answer: B

Explanation:

- List is an interface that defines core operations for ordered collections of elements, including adding, removing, accessing elements by index, and checking size. It allows for different implementations that can manage elements in various ways.
- ArrayList is a concrete class that implements the List interface. It provides a dynamic array-based implementation, offering functionalities outlined in the List interface and additional methods specific to ArrayList. These might include functionalities for managing the underlying array size, ensuring efficient resizing when capacity is reached.
- List defines the "what" (core functionalities for ordered collections).
- ArrayList defines the "how" (a specific implementation using dynamic arrays).

Q#21. Which of the following is true about LinkedHashSet in Java?

```
A) It does not allow null elements.
B) It maintains the insertion order of elements.
```

```
    C) It allows duplicate elements.
    D) It is synchronized by default.
```

Answer: B

Explanation: LinkedHashSet maintains the insertion order of elements. This means that when iterating through the LinkedHashSet, the elements will be returned in the order in which they were inserted. LinkedHashSet allows null elements and does not allow duplicate elements, similar to HashSet. It is also not synchronized by default, meaning it is not thread-safe without external synchronization.

Q#22. What are the difference between HashSet and TreeSet in the Java Collection Framework?

```
    A) Both HashSet and TreeSet allow duplicate elements and maintain insertion order.
    B) Both HashSet and TreeSet use hashing for element storage, offering constant time
    C) HashSet stores unique elements without a specific order, while TreeSet maintains
    D) HashSet is synchronized for thread-safe access, while TreeSet is not.
```

Answer: C

Explanation:

- **HashSet**:
  - Implements the Set interface.
  - Stores a collection of unique elements, ensuring no duplicates exist.
  - Employs hashing for efficient storage and retrieval (average constant time).
  - Doesn't maintain any specific order for elements (insertion order is not preserved).
- **TreeSet**:
  - Implements the SortedSet interface (a sub-interface of Set).
  - Stores a collection of unique elements, similar to HashSet.
  - Maintains elements in a naturally sorted order (ascending order by default).
  - Uses a tree-based data structure for efficient retrieval and ordered iteration. Sorting overhead might impact performance compared to HashSet for some operations.

In addition,

- **HashSet**: Focuses on efficient storage and retrieval of unique elements without a specific order.
- **TreeSet**: Prioritizes maintaining elements in a sorted order while ensuring uniqueness.

Q#23. What is the difference between ArrayList and LinkedList classes in the Collection Framework?

```
A) ArrayList uses a doubly linked list to store elements, while LinkedList uses a d
B) ArrayList is synchronized, while LinkedList is not synchronized.
C) ArrayList provides constant-time positional access, while LinkedList provides li
D) ArrayList allows duplicate elements, while LinkedList does not allow duplicate e
```

Answer: C

Explanation: The main difference between ArrayList and LinkedList is in their internal implementations and performance characteristics.

- **ArrayList** uses a dynamic array to store elements, which provides constant-time positional access (i.e., O(1) time complexity) because it allows direct access to any element using its index. This makes ArrayList more suitable for scenarios where frequent access to elements by index is required.
- **LinkedList** uses a doubly linked list to store elements, which provides linear-time positional access (i.e., O(n) time complexity) because it requires traversing the list from the beginning to reach a specific element by index. However, LinkedList is more efficient than ArrayList when it comes to insertions and deletions in the middle of the list, as these operations only require updating references rather than shifting elements.
- **ArrayList and LinkedList both** allow duplicate elements and are not synchronized by default.

Q#24. Which of the following concepts make extensive use of arrays?

```
A) Recursion
B) Linked Lists
C) Binary Trees
D) Sorting Algorithms
```

Answer: D

Explanation: Sorting algorithms make extensive use of arrays. Many sorting algorithms, such as QuickSort, MergeSort, and Bubble Sort, are designed to sort elements stored in arrays efficiently. Arrays provide a convenient structure for accessing and manipulating elements in a contiguous block of memory, which is essential for the operations performed by sorting algorithms. While recursion, linked lists, and binary trees can also use arrays, they do not rely on arrays as extensively as sorting algorithms do.

Q#25. Entries in a stack are 'ordered'. What is the meaning of this statement?

```
A) A collection of stacks is sorted.
B) Stack entries are stored in a FIFO (First In, First Out) manner.
C) Stack entries are stored in a LIFO (Last In, First Out) manner.
D) A stack keeps track of the number of entries it contains.
```

Answer: C

Explanation: The statement "entries in a stack are 'ordered'" means that stack entries are stored in a LIFO (Last In, First Out) manner. This means that the most recently added element is the first one to be removed. This ordering mechanism is fundamental to the stack data structure, where the push operation adds an element to the top of the stack and the pop operation removes the element from the top of the stack. Options A, B, and D do not correctly describe the ordering mechanism specific to stacks.

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