**Exception Hendling**

1. Arrange the three classes String, StringBuffer and StringBuilder in the order of efficiency for String processing operations?
2. What is the base class for Error and Exception classes in Java?
3. What is Exception Handling in Java?
4. Can we re-throw an Exception in Java?
5. What is the concept of Exception Propagation?
6. When we override a method in a Child class, can we throw an
7. additional Exception that is not thrown by the Parent class method?
8. When is UnsupportedOperationException thrown in Java?
9. Why do we sometime get ConcurrentModificationException during
10. iteration?
11. How will you handle InterruptedException in Java?
12. Let say there is a method that throws NullPointerException in the
13. superclass. Can we override it with a method that throws
14. RuntimeException?
15. Can you catch an exception thrown by another thread in Java?
16. When do you use Exception or Error in Java? What is the difference
17. between these two?
18. What happens to the Exception object after the exception handling is
19. done?
20. How will you handle runtime exceptions in JSP?
21. How will you handle exceptions in Spring MVC Framework?
22. What is an exception in Java?
23. What is the difference between checked and unchecked exceptions in Java?
24. What is the purpose of the `try-catch` block?
25. Explain the `finally` block in Java.
26. How is an exception propagated through the call stack in Java?
27. Can you give an example of a custom exception class in Java?
28. What is the difference between `throw` and `throws` in Java?
29. throw` is used to explicitly throw an exception within a method, whereas `throws` is used in a method declaration to specify the exceptions that might be thrown by that method.
30. What is the `try-with-resources` statement in Java, and when is it typically used?
31. How can you prevent a specific piece of code from throwing an exception in Java?
32. What is the purpose of the `try` block without a `catch` block in Java?
33. What is the difference between `System.exit()` and throwing an exception to terminate a Java program?
34. Explain the differences between the `try-catch` and `try-finally` blocks in Java.
35. What is the purpose of the `throws` clause in a method signature?
36. What is the difference between `NullPointerException` and
37. What is the role of the `printStackTrace()` method in exception handling?
38. Explain the concept of chained exceptions in Java.
39. How can you handle multiple exceptions in a single `catch` block?
40. What is the difference between the `throw` and `throws` keywords in Java?
41. How can you create a custom unchecked exception in Java?
42. When should you use checked exceptions, and when should you use
43. What is the purpose of the try-with-resources statement in Java, and how does it improve resource management?
44. Explain the differences between sError and Exception` in Java. When should you catch or handle each of them?
45. How can you rethrow an exception in Java, and what is the purpose of doing so?
46. What is the difference between `Error` and `RuntimeException` in Java, and when should you catch or throw each of them?
47. What is the `try-with-resources` statement in Java, and how does it relate to the `AutoCloseable` interface?
48. How can you create a custom checked exception in Java, and when might it be useful to do so?
49. What is the purpose of the `try-catch-finally` block in Java, and in what scenarios is it commonly used?
50. Can you explain the `catch` or `throws` order of exception handling in multi-level exception hierarchies?
51. What is the difference between a compile-time error and a runtime exception in Java?
52. How do you handle exceptions when working with multi-threading or concurrency in Java?
53. What are the best practices for exception handling in Java, and how can you ensure robust and effective exception handling in your code?
54. Explain the concept of custom exception chaining and its advantages in software development.
55. How do you handle exceptions when working with I/O operations in Java, such as reading or writing to files?
56. Can you provide an example of the `try-catch-resource` pattern in Java, and explain its benefits in resource management?
57. What is the difference between the `finally` block and the `final` keyword in Java?
58. How can you create a custom exception class that includes additional information, such as custom error messages or error codes?
59. Explain the difference between the `throws` and `throw` keywords in Java, and when to use each of them.
60. How do you handle exceptions in Java when dealing with database operations, such as connecting to a database or executing SQL queries?
    1. hat are different scenarios causing “Exception in thread main”?
61. What is OutOfMemoryError in Java?
62. What is the difference between ClassNotFoundException and NoClassDefFoundError?
63. What do you understand by an unreachable catch block error?
64. Is it possible to throw checked exceptions from a static block?
65. What are different scenarios where “Exception in thread main” types of error could occur?

**ANS Exception**

1. Can we re-throw an Exception in Java?

Yes, you can re-throw an exception in Java using the `throw` statement. Re-throwing an exception allows you to catch an exception within a `catch` block, perform some processing or logging, and then throw the same or a different exception to be handled by an outer exception handler.

public class Example {

public static void main(String[] args) {

try {

doSomething();

} catch (Exception e) {

System.out.println("Caught an exception in the main method: " + e.getMessage());

}

} public static void doSomething() throws Exception {

try {

// Some code that may throw an exception

int result = 10 / 0; // This will throw an ArithmeticException

} catch (ArithmeticException ae) {

// Handle the exception locally

System.out.println("Caught an ArithmeticException: " + ae.getMessage());

// Re-throw the exception

throw ae;

}

}

}

**JDBC**

**SERVLET**

1. In which scenario, Container initializes multiple JSP/Servlet objects?
2. Java Design Patterns Which protocol does Browser and Servlet use to communicate with each other?
3. What is HTTP Tunneling?
4. Why do we use JSP instead of Servlet in Java
5. What is the difference between init-param and context-param?
6. In which scenario, Container initializes multiple JSP/Servlet objects?
7. What are the examples of Template
8. method design pattern in JDK?
9. What is DispatcherServlet?
10. What is a Servlet?
11. Explain the life cycle of a Servlet.
12. What is the difference between a Servlet and JSP?
13. How do you map a Servlet to a URL in web.xml?
14. What is the difference between the doGet() and doPost() methods in a Servlet?
15. What is the purpose of the ServletConfig and ServletContext objects?
16. How can you share data between Servlets?
17. Explain the differences between the forward and redirect methods in a Servlet.
18. What are Servlet Filters, and what is their purpose?
19. How can you handle exceptions in a Servlet?
20. Certainly, here are some more Servlet interview questions:
21. What is the purpose of the `init()` and `destroy()` methods in a Servlet?
22. What is a Servlet Container (or Servlet Engine)?
23. 1Explain the difference between `response.sendRedirect()` and `request.getRequestDispatcher().forward()` in servlets.
24. What is a session in Servlets, and how can you manage session data?
25. What is the purpose of the `ServletContextListener` and `HttpSessionListener` interfaces?
26. Explain the use of cookies in servlets and how to set and retrieve cookies.
27. What is the purpose of the `web.xml` deployment descriptor in a Servlet application?
28. Explain the difference between a web server and a servlet container.
29. How can you secure a Servlet application?
30. What is the purpose of the `@WebServlet` annotation in modern Servlet development?
31. Certainly, here are some more Servlet interview questions:
32. What is the purpose of the `doHead()` method in a Servlet, and when would you use it?
33. Explain the concept of Servlet thread safety.
34. How can you handle file uploads in a Servlet?
35. What is the purpose of the @MultipartConfig annotation in Servlet 3.0 and later versions?
36. How can you handle internationalization and localization in a Servlet application?
37. Explain the concept of a Servlet container's thread pool.
38. How can you handle session management in a clustered environment?
39. What is the difference between the `setContentType()` and `setCharacterEncoding()` methods in the `HttpServletResponse` class?
40. Explain the use of the `doFilter()` method in Servlet Filters.
41. What is the purpose of the `ServletContext` method `getRequestDispatcher()`?