Exception Handling

1. **What are exceptions?**
   Exception refers to such mistakes that would occur during the execution time of an app due to faulty inputs given by the user.

2. **What happens if exceptions are not handled?**
   It would result in abrupt termination of the app due to the exception object reaching the default exception handler.

3. **Does C language support exception handling?**
   No.

4. **Does C++ language support exception handling?**
   Yes. But not robust.

5. **Does Java language support exception handling?**
   Yes.

6. **What is meant by exception handling?**
   It refers to the process of managing the exception object in a manner that it does not result in abrupt termination of an app.

7. **Are exceptions detected during compilation time?**
   No.

8. **Are exceptions detected during execution time?**
   Yes.

9. **What is an error?**
   Error refers to such mistakes that would occur during execution time due to faulty coding by the programmer. Errors cannot be handled using try and catch.

10. **What is the difference between exception and error?**

Exception	Error
Occurs due to faulty inputs given by the user.	Occurs due to faulty coding by the developer.
Can be handled using try and catch	Cannot be handled using try and catch
Abrupt termination of an app can be prevented	Abrupt termination of an app cannot be prevented
Examples include ArithmeticException, NegativeArraySizeException, ArrayIndexOutOfBoundsException etc.,	Examples include StackOverFlowError, OutOfMemoryError etc.,

11. **Why are exceptions caused normally in a program?**
Due to faulty inputs given by the user.

12. **Why do errors occur normally in a program?**
Due to faulty coding of an app by the developer.

13. **What is the difference between compile time error and run time error?** Compilation time errors are syntax errors which can be detected by the compiler.
Eg. missing semicolon, undefined symbol etc.

Run time errors are such errors which cannot be detected by the compiler. They would be detected during execution time.
Eg. StackOverFlowError , OutOfMemoryError etc.

14. **Which construct is used in Java for exception handling**?
try-catch-throw-throws-finally.

15. **What is a role of try keyword in Java?**

Such statements which could possibly result in an exception would be placed inside the try block.

16. **What is a role of catch keyword in Java?**
    If any exception object is generated in the try block, then it would be caught by the catch block.

17. **What is a role of throw keyword in Java?**
    It is used to re-throw an exception object and hence forcefully enables a handled exception object to trickle down the stack hierarchy.

18. **What is a role of finally keyword in Java?**
    The most critical statements that have to be compulsory executed such as statements releasing resources present at the end of a method must be enclosed within the finally block.

19. **What is a role of throws keyword in Java?**
    It is used to duck an exception. It is also used to alert the caller of the method that this method could possibly throw an exception object.

20. **Does Java language provide any exception handler?**
    Yes. It provides default exception handler.

21. **If Java language provides exception handler then why should programmer handle exceptions?**
    Because the default exception handler provided by java ensures only system safety. It does not ensure termination of connections and release of acquired resources. Hence programmer should handle an exception.

22. **Who creates the exception object?**
    The method in which a run time mistake has occurred would create an exception object.

23. **What would be present in an exception object?**
    i) The type of exception
    ii) Line no. on which exception occurred

iii) Stack trace

24. **Whom is the exception object given to soon after its creation?**
Exception object would be given to Run Time System(RTS).

25. **What is run time system?**
It is a part of JVM which deals with exception handling.

26. **What is a role of run time system in exception handling?**
It receives the exception object from the method and verifies if a user defined exception handler is present. If the user defined exception handler is present then the exception object would be handed over to it. Otherwise the exception object would be handed over to the default exception handler.

27. **Can we have multiple catches for a single try block?**
Yes.

28. **Why should an Exception type reference be placed at the bottom of multiple catch blocks?**
In order to ensure that if the specific catch blocks present at the top of the catch hierarchy fails to handle an exception, then the generic catch present at the bottom would be able to handle it.
By doing this,
    1) The exception object is prevented from reaching the default exception handler and hence prevents abrupt termination.
    2) By using log4j tool in the generic catch block, a log report can be sent to the developer which would be useful to upgrade the app.

29. **How is it that Exception reference can catch (refer to) all types of exceptions?**
Because, in the Exception hierarchy it is a parent type. In java parent type reference can refer to any child object. Hence the Exception type reference can catch any of the child class objects.

30. **What are checked exceptions?**

They are such exceptions for which java compiler forces to handle it using a try catch block.
Eg: SQLException, ClassNotFoundException etc.,

31. **What are unchecked exceptions?**
They are such exceptions for which java compiler does not force the programmer to handle it using try catch.
Eg: ArrayOutOfBoundsException, ArithmeticException etc.,

32. **Why should we not have exception type reference at the top of multiple catch blocks?**
Because if it is placed at the top of the catch hierarchy, it would catch all exception that are generated by the try block without giving any opportunity for the specific catch blocks present below in the hierarchy to catch the exception object.

33. **What is meant by ducking?**
It refers to the process in which a method would pass an exception object without handling it. "throws" keyword is used for ducking.

34. **What is meant by rethrowing?**
It is the process of a method passing the exception object down the stack hierarchy after handling it. "throw" and "throws" keywords are used for rethrowing.

35. **What is the problem associated with throw keyword? How is it resolved?** Disadvantage of "throw" keyword is that statements below throw keyword would not get executed. This problem can be overcome by using "finally" block.

36. **Is using throws compulsory in exception handling?**
Not upto JDK 1.7. However, from JDK 1.8 onwards if a method ducks or re-throws an exception object, then it has been made compulsory to use "throws" keyword.

37. **How does the run time system respond in case of multiple method calls?**
    If the exception handler is not present in the method that generated the exception object, then the Run Time System would not send the exception object directly to the Default Exception Handler (DEH). Rather, the exception object would trickle down
    the stack hierarchy till a handler is found in one of the methods (activation record). If no handler is found then the exception object would be handed over to Default Exception Handler.

38. **Can you give an example for an exception?**
    ArithmeticException, ArrayIndexOutOfBoundsException, NegativeArraySizeException, ArrayStoreException etc.,

39. **Can you give an example for an error?**
    StackOverFlowError, OutOfMemoryError etc.,

40. **Is there any way in Java by which the finally block would not get executed?**
    Yes. By using System.exit(0).

41. **When should a method duck an exception object?**
    It is decided by the designer of the project depending upon the specific project requirements.

42. **When should a method rethrow an exception object?**
    It is decided by the designer of the project depending upon the specific project requirements.

43. **Can we manually create exception objects?**
    Yes.

44. **Is a try-catch combination valid?**
    Yes.

45. **Is a try-catch-finally combination valid?**
    Yes.

46. **Is a try-finally combination valid?**
    Yes.

47. **Is a catch-finally combination valid?**
    No.

48. **Can we have a try block without any other block?**
    No.

49. **Can we have a catch block without any other block?**
    No.

50. **Can we have a finally block without any other block?**
    No.

51. **Can we have try-catch-finally within a try block?**
    Yes.

52. **Can we have try-catch-finally within a catch block?**
    Yes.

53. **Can we have try-catch-finally within a finally block?**
    Yes.

54. **Name a few checked exceptions?**
    FileNotFoundException, ClassNotFoundException, SQLException etc.,

55. **Name a few unchecked exceptions**
    ArithmeticException, ArrayIndexOutOfBoundsException, NegativeArraySizeException, ArrayStoreException etc.,

56. **Which class is the superclass of Exception?**
    Throwable.

57. **Which class is the superclass of Error?**
    Throwable.

58. **What is the role of getMessage() in exception handling?**
It enables programmer to create custom exception.

59. **What is the role of printStackTrace() in exception handling?**
It gives the developer an idea about the method in which exception occurred and also the subsequent propagation of the exception object.

60. **Differentiate between throw and throws?**

throw	throws
Used to rethrow an Exception object.	Used for ducking and also to rethrow an exception object.
It is associated with the Exception type reference.	It is associated with the signature of the method.
It would be present within the body of the method.	It would be present in the signature of the method.

61. **What is meant by LSP?**
LSP stands for Liskov Substitution Principle. It speaks about the rules associated with Exceptions and overriding.

62. **What is try region also called as?**
Guarded region.

63. **What is catch region also called as?**
Handling region.

64. **What is finally region also called as?**
Cleanup region.

65. **What are the rules of exception that must be followed during inheritance?**
LSP rules. (Eg: Refer class notes)

66. **Can the base class method and derived class method throw different types of exception?**
Yes. Provided
1) Is-a relationship exists between them
2) Even though Is-a relationship does not exists, it is possible provided the exceptions being thrown are of Runtime Exceptions (unchecked Exceptions).

67. **If the base class method throws an exception is it mandatory for the derived class method also to throw an exception?**
No.

68. **How do we create custom exceptions?**

i) By sub classing Exception class ,
ii)By overriding public String getMessage() to provide suitable diagnostic message.

69. **If I want an object of my class to be thrown as an exception object then what should I do?**
If I want an object of my class to be thrown as Exception object, then a class should be made as subclass of an Exception class.

70. **If my class already extends from some other class what should I do if I want an instance of my class to be thrown as an exception object?**
Not possible, because multiple inheritance is not permitted in java.

71. **What are the different ways to manage exceptions?**
i) Handling an Exception,
ii) Re-throwing an Exception,
iii) Ducking an Exception.

72. **If I write return at the end of the try block, will the finally block still execute?** Yes.

73. **What is the difference between final, finally and finalize()?**
final - used to create constants and to prevent inheritance
finally - is an exception handing control construct in which most

critical statements are placed.

finalize() - finalize() method would be called by garbage collector in java to perform clean up operations on an object before it is removed from the memory.