FAQ's on Exception Handling

1. What are the different possible errors that we can encounter while compiling and executing the java program?

The different types of errors that we may encounter while compiling and executing a Java program are:

- Compile-time error
- Run-time errors
- Logical errors

2. What is an Exception?

An exception is an event that disrupts the normal flow of instructions during the execution of a program. It is a runtime error in java.

We can also say that the object-oriented representation of an abnormal event occurred during the program execution is known as "exception". I.e. Exception is an object.

3. What are the problems if the exception is raised?

- If an exception is raised, JVM terminates the application abnormally.
- If the program is terminated abnormally the following problems may occur:
  - Users work and data lost.
  - End user is not informed properly about what went wrong.
  - Resources allocated to the application cannot be released gracefully.

4. What is an exception handling?

It is a mechanism of dealing with exceptions and thereby preventing the dangerous effect caused by the exceptions.

5. How to handle exceptions in java applications?

There are two kinds of support from java to implement exception handling.

- Language support and
- API support

- As far as language support concern we have five keywords in java to implement exception handling:
    - try
    - catch
    - finally
    - throw
    - throws

To represent each abnormal scenario, there is a library exception class given in java API.

Example: ArithmeticException, NullPointerExcpetion, NumberFormatException, ClassNotFoundException, FileNotFoundException etc.

6. What are the types of exceptions?

There are two types of exceptions in java

- Checked exception
- Un-checked exception

7. What are checked exceptions?

Checked exceptions are those which the Java compiler forces you to catch. E.g. IOException.

8. What are runtime exceptions?

Runtime exceptions are those exceptions that are thrown at runtime because of either wrong input data or because of wrong business logic. These are not checked by the compiler at compile time.

9. What is the difference between error and an exception?

An error is an irrecoverable condition occurring at runtime, such as, OutOfMemory error. These are the JVM errors and you cannot repair them at runtime. While exceptions are the conditions that occur because of bad input etc. For example:

- FileNotFoundException will be thrown if the specified file does not exist.
- NullPointerException will take place if you try using a null reference.

In most of the cases it is possible to recover from an exception (probably by giving user a feedback for entering proper values, etc.)

10. How to create custom exceptions?

Your class should extend class Exception, or some more specific type thereof.

11. What should I do, if I want an object of my class to be thrown as an exception object?

Extend the class from Exception class. Or you can also extend your class from some more precise exception types.

12. What is the basic difference between the following approaches of exception handling?

1) 'try catch' block and
2) Specifying the candidate exceptions in the throws clause?

When should you use which approach?

In the first approach you, being a programmer of the method, yourself are dealing with the exception. This is fine, if you are in a best position to decide. If it is not the responsibility of the method to deal with its own exceptions, then do not use this approach.

In the second approach we are forcing the caller of the method to catch the exceptions that the method is likely to throw. This is often the approach library creator uses. They list the exception in the throws clause and we must catch them. You will find the same approach throughout the java libraries we use.

13. Is it necessary that each try block must be followed by a catch block?

It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block or by a finally block. And, whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

14. If I write return at the end of the try block, will the 'finally' block still execute?

Yes. Even if you write return as the last statement in the try block and no exception occurs, the 'finally' block will still execute and then returns the control.

15. If I write System.exit (0); at the end of the try block, will the 'finally' block still execute?

No. In this case the finally block will not execute because when you say **System.exit (0);** the control immediately goes out of the program, and thus 'finally' block never executes.

16. What is the use of 'Finally' block?

The "finally" statement identifies a block of code that cleans up regardless of whether an exception occurred within the try block or not. A "try" statement must be accompanied by at least one "catch" statement or a "finally" statement and may have multiple "catch" statements.

17. What is user-defined exception in java?

User-defined exceptions are the exceptions defined by the application developer, which are actually errors related to specific application. Application Developer can define the user defined exception by inheriting the Exception class as shown below. Using this class we can throw new exceptions.

Java Example:

```
public class noFundException extends Exception{
}
Throw an exception using a throw statement:
public  class  Fund { ...
 public Object getFunds() throws noFundException
    {
       if (Empty()) throw new noFundException(); ...
    }
}
User-defined exceptions should usually be checked.
```

18. By using one line of code how can one prove that the array is not null but empty?

Print `args.length`. It will print 0. That means it is empty. But, if it is null then it will thrown a NullPointerException.

19. What are Checked and UnChecked Exception?

A checked exception is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses. Making an exception checked, forces the client programmers to deal with the possibility that the exception will be thrown.

Example: IOException thrown by java.io.FileInputStream's read() method·

Unchecked exceptions are Runtime Exception and any of its subclasses. Class Error and its subclasses also are unchecked. With an unchecked exception, the compiler doesn't force client programmers either to catch the exception or declare it in a 'throws' clause. In fact, client programmers may not even know that the exception could be thrown.

Example: StringIndexOutOfBoundsException thrown by String's charAt() method· Checked exceptions must be caught at compile time. Runtime exceptions do not need to be. Errors often cannot be.

20. What are checked exceptions?

Checked exceptions are those which the Java compiler forces you to catch.
Example: IOException are checked exceptions.

21. 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?

You cannot do anything in this scenario. Java does not allow multiple inheritances and does not provide any exception interface too.

22. How does an exception permeate through the code?

An unhandled exception moves up the method stack in search of a matching. When an exception is thrown from a code that is wrapped in a 'try' block followed by one or more 'catch' blocks, a search is made for matching catch block. If a matching type is found then that block will be invoked. If a matching type is not found then the exception moves up the method stack and reaches the caller method.

Same procedure is repeated if the caller method is included in a 'try catch' block. This process continues until a 'catch' block handling the appropriate type of exception is found. If it does not find such a block then finally the program terminates.

23. What are the different ways to handle exceptions?

There are two ways to handle exceptions:

- By wrapping the desired code in a ‘try’ block followed by a ‘catch’ block to catch the exceptions. and
- List the desired exceptions in the ‘throws’ clause of the method and let the caller of the method handle those exceptions.

24. How does a try statement determine which catch clause should be used to handle an exception?

When an exception is thrown within the body of a try statement, the catch clauses of the ‘try’ statement are examined in the order in which they appear. The first ‘catch’ clause that is capable of handling the exception is executed. The remaining catch clauses are ignored.

25. What is the catch or declare rule for method declarations?

If a checked exception may be thrown within the body of a method, the method must either catch the exception or declare it in its throws clause.

26. What classes of exceptions may be caught by a catch clause?

A catch clause can catch any exception that may be assigned to the Throwable type. This includes the Error and Exception types.

27. Can an exception be re-thrown?

Yes, an exception can be re-thrown.

28. What class of exceptions are generated by the Java run-time system?

The Java runtime system generates Runtime Exception and Error exceptions.

29. What are the ‘final’, ‘finalize()’ and ‘finally’?

final: The ‘final’ keyword can be used for class, method and variables. A final class cannot be subclassed and it prevents other programmers from subclassing a secure class to invoke insecure methods. A final method can’t be overridden. A final variable can’t change from its initialized value.

finalize(): The ‘finalize()’ method is used just before an object is destroyed and can be called just prior to garbage collection.

finally: It is a key word used in exception handling, creates a block of code that will be executed after a try/catch block has completed and before the code following the try/catch block executes. The ‘finally’ block will execute whether or not an exception is thrown. For example, if a method opens a file upon exit, then you will not want the code that closes the file to be bypassed by the exception-handling mechanism. This ‘finally’ keyword is designed to address this contingency.