* **Exception**

1. **What is Exception in Java?**

Exception is an abnormal scenario that can happen during the execution of a program, which interrupts the normal execution of the program.

Exception can occur in different kind of scenarios user entered wrong data, hardware failure, network connection failure etc.

What happens actually in code when exception occurred?

Whenever any error occurs while executing a java statement, an exception object is created and then JRE tries to find exception handler to handle the exception. If suitable exception handler is found then the exception object is passed to the handler code to process the exception, known as catching the exception. If no handler is found then application throws the exception to runtime environment and JRE terminates the program.

1. **What are the Exception Handling Keywords in Java?**

There are four keywords used in java exception handling.

1. **try-catch:** try-catch block is used to handle exception in our code. try can contain that we may think can throw an exception and catch block can have code which handles the exception may be return an error code to user and inform him about the exception.

One try can have zero or more catch blocks. If catch block is not there then at least it should have finally block.

try-catch block can be nested also.

1. **finally:** Code that is to be mandatorily executed irrespective of whether exception occurs or not can be put in to finally block like closing the resources.

Finally block can be used only with try-catch block.

Finally block is optional.

If finally is not there then at least catch block should be there along with try

1. **throw:** Sometimes we explicitly want to create exception object and then throw it. Once we throw the exception it might be handled in some other method if there is no handling then will be throwed to runtime environment.
2. **throws:** This keyword is used in method signature indicates the method might throw the exceptions so that caller of the method is aware of it and can handle the exception if there is no handling at caller side then exception can be propagated to another method using same throws keyword.

We can provide multiple exceptions in throws keyword.

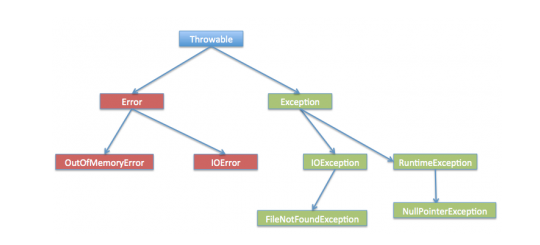
throws can be used with main() method also.

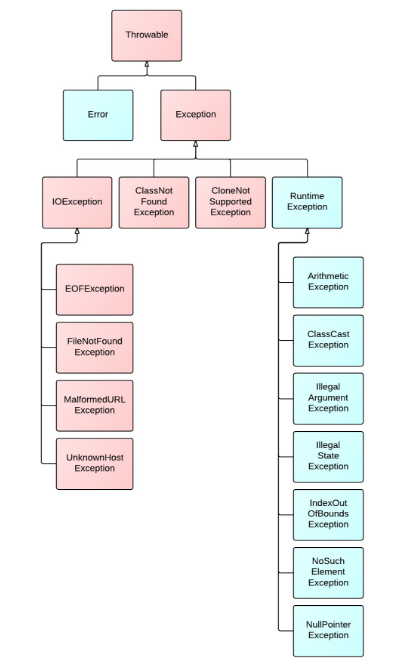
1. **What is difference between throw and throws keyword in Java?**

**throws** keyword is used with method signature to declare the exceptions that the method might throw.

**throw** keyword is used to interrupt the flow of program and handing over the exception object to runtime to handle it.

1. **Explain Java Exception Hierarchy?**

****

****

Throwable is the parent class of Java Exceptions Hierarchy and it has two child objects – Error and Exception. Exceptions are further divided into checked exceptions and runtime exception.

**Errors** are exceptional scenarios that are out of scope of application and it’s not possible to anticipate and recover from them, for example hardware failure, JVM crash or out of memory error.

**Checked Exceptions** / compile time exceptions are exceptional scenarios that we can anticipate in a program and try to recover from it.

For example FileNotFoundException. We should catch this exception and provide useful message to user and log it properly for debugging purpose. Exception is the parent class of all Checked Exceptions.

**Runtime Exceptions**/ uncheckedExceptions are caused by bad programming.

For example trying to retrieve an element from the Array. We should check the length of array first before trying to retrieve the element otherwise it might throw ArrayIndexOutOfBoundException at runtime. Runtime Exception is the parent class of all runtime exceptions.

Eg. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc

1. **What is the use of throwable in exception block? IMP**
2. **Difference between Exception and Error? IMP**

**Exceptions:** can be caught and dealt by the application

Ex: IOException.

**Error:** are the scenarios that are out of the scope of the application.It is not possible to anticipate them

**Ex:** hardware failure, JVM crash or out of memory error

1. **Which is final among error and exception? IMP**

1. **Method Overriding with exception handling? IMP**

Rules are as follows

**If the super class method does not declare an exception**

Subclass overridden method cannot declare the checked exception but it can declare unchecked exception.

**If the superclass method declares an exception**

Subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

If these rules are not followed then there will be compilation issue.

1. **Explain Java 7 ARM Feature and multi-catch block?**

<http://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-keywords>

1. **What is difference between Checked and Unchecked Exception in Java? IMP**
2. Checked Exceptions should be handled in the code using try-catch block or else main () method should use throws keyword

Unchecked Exceptions are not required to be handled in the program or to mention them in throws clause.

1. **Exception** is the super class of all checked exceptions whereas **RuntimeException** is the super class of all unchecked exceptions.
2. Checked exceptions are error scenarios that are not caused by program, for example FileNotFoundException in reading a file that is not present.

Unchecked exceptions are mostly caused by poor programming, for example NullPointerException when invoking a method on an object reference without making sure that it’s not null.

1. **Why we have 2 categories of checked and unchecked exceptions? IMP**
2. **How to create checked exception? IMP**

Checked exceptions are created by extending Exception class generally all exceptions in java are checked exceptions unless if they do not extend RuntimeException

1. **How to create unchecked exception in java? IMP**

Unchecked exceptions in java are created by extending RuntimeException.

1. **How to write custom exception in Java? IMP**

We can extend Exception class or any of its subclasses to create our custom exception class. The custom exception class can have its own variables and methods that we can use to pass error codes or other exception related information to the exception handler.

We can also extend Throwable if it is really required

Pending: Example code.

1. **What is the difference between ClassNotFoundException, NoClassDefinitionException (This question is also related to class loader)?**  **IMP**

**ClassNotFoundException:**

It occurs when an application tries to load a class at run time which is not updated in the classpath.

This is exception

It is thrown by the application itself. It is thrown by the methods like Class.forName(), loadClass() and findSystemClass().

**NoClassDefinitionException**

It occurs when class was there during compilation time but missing during runtime.

This is error.

It is thrown by the Java Runtime System.

1. **What is OutOfMemoryError in Java? IMP**

This error is thrown by JVM when it ran out of heap memory. We can fix this error by providing more memory to run the java application through java options.

$>java MyProgram -Xms1024m -Xmx1024m -XX:PermSize=64M -XX:MaxPermSize=256m

1. **How do you find out outOfMemory in java? How do you find out which code has problem? IMP** You can remember the RAS synchronization issue and how do you solved it
2. **What are different scenarios causing “Exception in thread main”?**

<http://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-keywords>

1. **What is difference between final, finally and finalize in Java?**

final and finally are keywords in java whereas finalize is a method.

**final** keyword can be used with class variables so that they can’t be reassigned, with class to avoid extending by classes and with methods to avoid overriding by subclasses.

**finally** keyword is used with try-catch block to provide statements that will always gets executed irrespective of whether exception arises or not, usually finally is used to close resource.

**finalize()** method is executed by Garbage Collector before the object is destroyed, it’s great way to make sure all the global resources are closed.

1. **What happens when exception is thrown by main method?**

When exception is thrown by main() method, Java Runtime terminates the program and print the exception message and stack trace in system console.

1. **Can we have an empty catch block?**

We can have an empty catch block but it’s the example of worst programming. We should never have empty catch block because if the exception is caught by that block, we will have no information about the exception and it will be a nightmare to debug it. There should be at least a logging statement to log the exception details in console or log files.

1. **Provide some Java Exception Handling Best Practices?**
2. Use Specific Exceptions for ease of debugging. Avoid Overly broad catch
3. Use custom exceptions to throw single type of exception from your application API.
4. Always log exception messages for debugging purposes.
5. Follow naming convention, always end with Exception.
6. Document the Exceptions Thrown by a method using @throws in javadoc.
7. Exceptions are costly, so throw it only when it makes sense. Else you can catch them and provide null or empty response.
8. Use multi-catch block for cleaner close.
9. Use Java 7 ARM feature to make sure resources are closed or use finally block to close them properly.
10. throw Exceptions Early (Fail-Fast) in the program.
11. Catch Exceptions late in the program, let the caller handle the exception.
12. **What is the base class for Error and Exception?**

Throwable

1. **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 a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method. Otherwise Exception will be thrown to respective caller.

1. **Can finally block be used without catch?**

Yes, by try block. Finally must be followed by either try or catch.

1. **Is there any case when finally will not be executed?**

finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort)

1. **Can an exception be rethrown?**

Yes

1. **What is the problem with below programs and how do we fix it?**

<http://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-keywords>

1. **What are important methods of Java Exception Class?**

Exception and all of its subclasses doesn’t provide any specific methods and all of the methods are defined in the base class Throwable.

**String getMessage()** – This method returns the message String of Throwable and the message can be provided while creating the exception through it’s constructor. This method provides basic information about the exception for example for FileNotFoundException, this contains the name of the file that could not be found.

**String getLocalizedMessage()** – It internally calls getMessage(), This method is provided so that subclasses can override it to provide locale specific message to the calling program.

**synchronized Throwable getCause()** – This method returns the cause of the exception or null id the cause is unknown.

**String toString()** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.

**void printStackTrace()** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as argument to write the stack trace information to the file or stream.