Chapter 12: Exception Handling

Exercises :

1. When you use a function from a module, try to observe what type of exception the function can throw in the Scala documentation online. Try to write programs in such a way that you handle all of those exceptions.
2. Explore the benefits of using scala.util. {Try,Success,Failure} for exception handling.

Answers :

- Exception

An exception is an unwanted or unexpected event that occurs during the execution of a program which changes its normal flow. Exception handling is the mechanism to respond to and investigate the occurrence and cause of an exception.

Generally, exceptions are of two types - checked and unchecked. Scala only allows unchecked exceptions - this means that there is no way to know if a method throws an unhandled exception at compile-time
It is best practice in Scala to handle exceptions using a `try{...} catch{…}` block, similar to how it is used in Java, except that the `catch` block uses pattern matching to identify and handle exceptions.

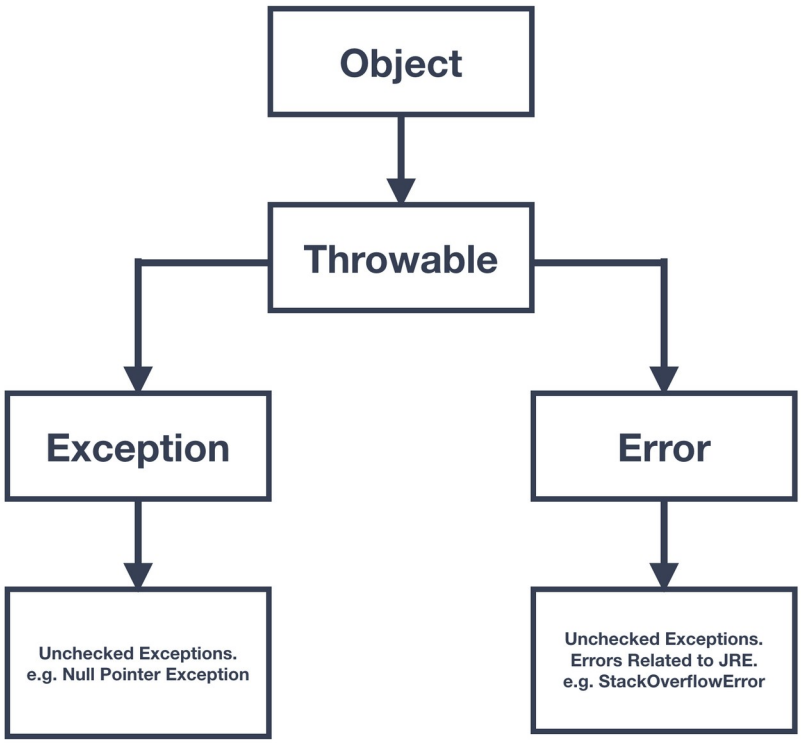

Figure: Scala Exceptions Hierarchy

As evident from the diagram shown above, one branch of the hierarchy under Throwable is headed by Exception, which is the class used for exceptional conditions that programs should catch. An example of an exception is NullPointerException.

Another branch is Error, which is used by the Java Virtual Machine (JVM) to indicate errors that are related to the Java runtime environment itself (JRE). An example of an error is StackOverflowError.

- Try/catch block

Scala allows handling exceptions using a single try/catch block. Exceptions can then be pattern matched using case blocks instead of providing a separate catch clause for each exception.

- Finaly block

An expression can be wrapped with the finally clause if some part of the code needs to be executed irrespective of what happens before and how the expression terminates. This can especially be useful to close resources like database connections.

- Custom exception

Scala allows developers to create their own custom exceptions. To declare a custom exception, the Exception class needs to be extended. In the custom class, the conditions to throw the exception and a message can be defined.

- The benefits of using scala.util. {Try,Success,Failure} for exception handling.

Try block – We put the actual code inside this block.

Success block – In this block, we get the output result in case the execution of the code is successful.
Failure block – In case of any exception, we land into this block with the error details.