**Valid combination of try-catch-finally**

1)try followed by one catch,

2)try followed by multiple catch but the parent class exception must be caught after catching child class exceptions.

3)try followed by finally.

4)in between try and catch/finally you cannot have any statement.

5) between try/ catch and finally you cannot have any statement.

6)within try I can have try-catch-finally

7)within catch I can have try-catch-finally

8) within finally I can have try-catch-finally

9)curly braces are mandatory for all the try/catch/finally block.

**Difference between Exception and Error**

Throwable(Class)

Exception Error

Most of the cases Exceptions are caused by our program and these are recoverable. Ex:🡪FileNotFound Exception.

Most of the time Errors are not caused by our program , these are due to lack of system resources and Errors are non recoverable. Ex:🡪OutOfMemoryException.

**Difference between Checked and Unchecked Exceptions**

\*\*The exceptions which are checked by compiler for smooth execution of the program at compile time are **checked exception**. If we don’t handle these exceptions compile time error will come. It can be handled by either try catch block or adding throws clause in main.

\*\*The exceptions not checked by compiler are **unchecked exception.** In case of unchecked compiler will not check it is handled or not.

Throwable(Class)

Exception Error

Runtime Exception

\*\*Runtime Exception and its child classes and Error and its child classes are Unchecked Exception.

Other than this all are checked.

**Difference between fully Checked and partially Checked Exceptions**

\*\*A checked Exception is fully checked iff all its child classes are checked .

Ex: IOException ,InteruptedException.

\*\*A checked Exception is partially checked iff some of its child classes are unchecked .

Ex: Exception and Throwable.

\*\* Only two partially checked Exception in java 1) Throwable , 2)Exception.

\*\*Finally block always will get executed.

\*\*\*\*In Java SE 7 and later, a single catch block can handle more than one type of exception. This feature can reduce code duplication and lessen the temptation to catch an overly broad exception.

In the catch clause, specify the types of exceptions that block can handle, and separate each exception type with a vertical bar (|):

catch (IOException|SQLException ex) {

logger.log(ex);

throw ex;

}

**Note**: If a catch block handles more than one exception type, then the catch parameter is implicitly final. In this example, the catch parameter ex is final and therefore you cannot assign any values to it within the catchblock.

**Note:** If the JVM exits while the try or catch code is being executed, then the finally block may not execute. Likewise, if the thread executing the try or catch code is interrupted or killed, the finally block may not execute even though the application as a whole continues.

**TryWithResource**

The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement. Any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable, can be used as a resource.

The following example reads the first line from a file. It uses an instance of BufferedReader to read data from the file. BufferedReader is a resource that must be closed after the program is finished with it:

static String readFirstLineFromFile(String path) throws IOException {

**try (BufferedReader br =**

**new BufferedReader(new FileReader(path)))** {

return br.readLine();

}

}

**Note**: A try-with-resources statement can have catch and finally blocks just like an ordinary try statement. In a try-with-resources statement, any catch or finally block is run after the resources declared have been closed.

If an exception is thrown from the try block and one or more exceptions are thrown from the try-with-resources statement, then those exceptions thrown from the try-with-resources statement are suppresse

\*\* If try block returns a value still the finally block gets executed .If the return value of the try block is any value type or String type(which is immutable) then the return value doesn’t change but actual value gets changed however. If the return value of try block is reference variable and mutable then the return value also changed.

\*\*Important🡪 actually when the try block returns the value is put aside into the stack then finally gets executed (although the finally code will execute first then the try block will executed). So if the return value putted into the stack is any value type or immutable type like String then the finally block modification has no impact on return type but the actual value of the variable changes. Whereas for the reference type the returned value and the actual value both changes.