Types of Errors

1. **Compilation errors occur** in a program at the time of compilation. These errors occur due to the syntax mistakes under the program.

Ex- missing double quotes and terminators, typing wrong spelling for keywords, assigning wrong data to a variable, trying to create an object for abstract class and interface, etc.

2. Runtime errors- are occurred at the time of program execution. These errors occurred when we are entering wrong data into a variable, trying to open a file for which there is no permission, trying to connect to the database with the wrong user id and password, the wrong implementation of logic, missing required resources, etc

What is an Exception?

A runtime error is known as an exception. The exception will cause the abnormal termination of the program execution. So these errors (exceptions) are very dangerous because whenever the exception occurs in the programs, the program gets terminated abnormally on the same line where the error gets occurred without executing the next line of code.

Objects of exception classes are responsible for abnormal termination of the program whenever runtime errors (exceptions) occur. These exception classes are predefined under BCL (Base Class Libraries) where a separate class is provided for each and every different type of exception like

1. IndexOutOfRangeException
2. FormatException
3. NullReferenceException
4. DivideByZeroException
5. FileNotFoundException
6. SQLException,
7. OverFlowException, etc.

When an Exception is raised in C#, the program execution is terminated abnormally. That means the statements placed after the exception-causing statements are not executed but the statements placed before that exception-causing statement are executed by CLR.

What CLR does when an exception occurred in the program?

It creates the exception class object that is associated with that logical mistake (exception) and terminates the current method execution by throwing that exception object by using the “throw” keyword.

Exception Handling-

The process of catching the exception for converting the CLR given exception message to an end-user understandable message and for stopping the abnormal termination of the program whenever runtime errors are occurring is called Exception Handling

What is the procedure to Handle Exception?

1. Preparing the exception object that is appropriate to the current logical mistake.
2. Throwing that exception to the appropriate exception handler.
3. Catching that exception
4. Taking necessary actions against that exception

two methods to handle the exception in .NET

1. Logical Implementation
2. Try catch Implementation

To implement the try-catch implementation .NET framework provides three keywords

1. Try
2. Catch
3. finally

Try *keyword establishes a block in which we need to write the exception causing and its related statements. That means exception-causing statements must be placed in the try block so that we can handle and catch that exception for stopping abnormal termination and to display end-user understandable messages.*

Catch *block is used to catch the exception that is thrown from its corresponding try block. It has the logic to take necessary actions on that caught exception. The Catch block syntax in*

C# looks like a constructor. It does not take accessibility modifier, normal modifier, return type. It takes the only single parameter of type Exception. Inside catch block, we can write any statement which is legal in .NET including raising an exception.

Finally:

The keyword finally establishes a block that definitely executes statements placed in it. Statements that are placed in finally block are always going to be executed irrespective of the way the control is coming out from the try block either by completing normally or throwing an exception by catching or not catching.

When we implement multiple catch blocks in C#, then at any given point of time only one catch block going to be executed and other catch blocks will be ignored.

Is it possible to catch all exceptions using a single catch block ?

Yes, it is possible. We can catch all exceptions with a single catch block with the parameter “Exception”. The **Exception class is the superclass of all Exception classes** and hence it can handle all types of exceptions thrown in the try block.

What are System Exceptions?

An exception that is raised implicitly under a program by the exception manager because of some logical mistakes (some predefined conditions) is known as a system exception. For example:

1. DivideByZeroException
2. IndexOutOfRangeException
3. FormatExceptionetc

An exception that is raised explicitly under a program based on our own condition (i.e. user-defined condition) is known as the application exception.