**Explain different types of errors in C#.**

When we write and execute our code in the .NET framework then there is possible of two types of error occurrences they are

* Compilation errors
* Runtime errors

**What is a Compilation Error in C#?**

* An error that occurs in a program at the time of compilation is known as the compilation error. These errors occur due to syntax mistakes under the program.
* These errors occur by typing the wrong syntax like 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.
* That means this error occurs due to the poor understanding of the programming language. These errors can be identified by the programmer and can be rectified before the execution of the program only. So these errors do not cause any harm to the program execution.

##### ****What is Runtime Error in C#?****

The errors which are occurred at the time of program execution are called as a runtime error. These errors are like entering wrong data into a variable, trying to open a file for which there is no permission, trying to connect to the database with wrong user id and password, the wrong implementation of logic, missing of required resources, etc.

The Runtime errors are dangerous because whenever they occur in the program, the program terminates abnormally on the same line where the error gets occurred without executing the next line of code.

**What is an Exception?**

A runtime error is known as an exception. An exception cannot be identified and rectified by the programmer. The exception will cause abnormal termination of the program execution.

So these errors (exceptions) are very dangerous because whenever an 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

**What happens if an exception is raised in the program?**

Program execution is terminated abnormally. It means statements placed after exception causing statement are not executed but the statements placed before that exception causing statement are executed by CLR.

**What is Exception Handling in C#?**

The process of catching the exception for converting CLR given exception message to end-user understandable message or for stopping the abnormal termination of the program whenever runtime errors are occurring is called exception handling. Once we handle an exception under a program we will be getting following advantages

1. We can stop the abnormal termination
2. We can perform any corrective action that may resolve the problem occurring due to abnormal termination.
3. Displaying a user-friendly error message, so that the client can resolve the problem provided if it is under his control.

**What is the Exception Handling Procedure?**

Exception handling is a 4 steps procedure

1. Preparing the exception objects 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

**How can we handle the exception in .NET?**

There are three methods to handle the exception in .NET

1. Logical implementation
2. Try catch implementation
3. On error go to implementation

**Explain the difference between Error and Exception in C#?**

Exceptions are those which can be handled at the runtime whereas errors cannot be handled.

An exception is an object of a type deriving from the System.Exception class. The exception is thrown by the CLR (Common Language Runtime) when errors occur that are nonfatal and recoverable by user programs. It is meant to give you an opportunity to do something with a throw statement to transfer control to a catch clause in a try block.

The error is something that most of the time we cannot handle it. Errors are the unchecked exception and the developer is not required to do anything with these. Errors normally tend to signal the end of our program, it typically cannot be recovered from and should cause us to exit from the current program. It should not be caught or handled.

All the Errors are Exceptions but the reverse is not true. In general, Errors are which nobody can control or guess when it happened on the other hand Exception can be guessed and can be handled.

**What is the difference between System exceptions and Application exceptions?**

System exceptions are derived directly from a base class System.SystemException. A System-level Exception is normally thrown when a nonrecoverable error has occurred.

Application exceptions can be user-defined exceptions thrown by the applications. If you are designing an application that needs to create its own exceptions class, you are advised to derive custom exceptions from the System.ApplicationException class. It is typically thrown when a recoverable error has occurred.

**Explain about try-catch implementation.**

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

1. Try
2. Catch
3. finally

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

Catch block syntax looks like a constructor. It does not take accessibility modifier, normal modifier, return type. It takes a single parameter of type Exception.

Inside catch block, we can write any statement which is legal in .NET including raising an exception. If the catch block is used without an exception class then it is known as a generic catch block. If the catch block is used with exception class then it is known as a specific catch block.

**Finally:**Finally establishes a block that definitely executes statements placed in it. Statements that are placed in finally block are always 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.

**What is the difference between the “throw” and “throw ex” in .NET?**

throw re-throws the exception that was caught, and preserves the stack trace. throw ex-throws the same exception, but resets the stack trace to that method. Unless you want to reset the stack trace (i.e. to shield public callers from the internal workings of your library), the throw is generally the better choice, since you can see where the exception originated.