## **Errors and Exceptions: An Overview**

Errors refer to the mistakes or faults that occur during program development or execution. If you don't find them and correct them, they cause a program to produce wrong results. In this [C# Tutorial](https://www.scholarhat.com/tutorial/csharp), we will explore more about Errors and Exceptions which will include**exception handling in c#, types of errors in c#, try-catch-finally blocks in c#,**and last **proper use of try-catch, and finally block.**

**System Namespace Exceptions**

1. **SystemException**: Base class for all built-in exceptions.
2. **ArgumentException**: Thrown when one of the arguments provided to a method is not valid.
   * **ArgumentNullException**: Thrown when a null reference is passed to a method that does not accept it.
   * **ArgumentOutOfRangeException**: Thrown when the value of an argument is outside the allowable range of values.
3. **ArithmeticException**: Base class for errors in an arithmetic, casting, or conversion operation.
   * **DivideByZeroException**: Thrown when there is an attempt to divide by zero.
   * **OverflowException**: Thrown when an arithmetic operation in a checked context overflows.
4. **ArrayTypeMismatchException**: Thrown when attempting to store an element of the wrong type within an array.
5. **BadImageFormatException**: Thrown when a DLL or executable file is not valid.
6. **FormatException**: Thrown when the format of an argument does not meet the parameter specifications.
7. **IndexOutOfRangeException**: Thrown when an array index is outside the bounds of the array.
8. **InvalidCastException**: Thrown when an invalid cast is performed.
9. **InvalidOperationException**: Thrown when a method call is invalid for the object's current state.
10. **NullReferenceException**: Thrown when there is an attempt to dereference a null object reference.
11. **OutOfMemoryException**: Thrown when there is not enough memory to continue the execution of a program.
12. **StackOverflowException**: Thrown when the execution stack overflows because it contains too many nested method calls.
13. **TypeInitializationException**: Thrown when a static constructor throws an exception and no catch clause exists to handle it.

**IO Namespace Exceptions**

1. **IOException**: Base class for exceptions thrown while accessing information using streams, files, and directories.
   * **DirectoryNotFoundException**: Thrown when part of a file or directory cannot be found.
   * **FileNotFoundException**: Thrown when an attempt to access a file that does not exist on disk fails.
   * **PathTooLongException**: Thrown when a path or fully qualified file name is longer than the system-defined maximum length.
   * **EndOfStreamException**: Thrown when reading is attempted past the end of a stream.

**Security Namespace Exceptions**

1. **SecurityException**: Thrown when a security error is detected.
2. **UnauthorizedAccessException**: Thrown when the operating system denies access because of an I/O error or a specific type of security error.

**Reflection Namespace Exceptions**

1. **AmbiguousMatchException**: Thrown when an attempt to bind to a member results in more than one member matching the criteria.
2. **TargetException**: Thrown when an attempt is made to invoke an invalid target.
3. **TargetInvocationException**: Thrown when an exception is thrown by the invoked method or constructor.
4. **TargetParameterCountException**: Thrown when the number of parameters for an invocation does not match the expected number.

**Serialization Namespace Exceptions**

1. **SerializationException**: Thrown when an error occurs during serialization or deserialization.

**Threading Namespace Exceptions**

1. **ThreadAbortException**: Thrown when a call is made to the Abort method to destroy a thread.
2. **ThreadInterruptedException**: Thrown when a thread is interrupted while it is in a waiting state.
3. **ThreadStateException**: Thrown when a thread is in an invalid state for the method call.

**Web Namespace Exceptions**

1. **WebException**: Thrown when an error occurs while accessing the network through a web request.

**XML Namespace Exceptions**

1. **XmlException**: Thrown when an error occurs while processing XML.

**Application Specific Exceptions**

1. **NotSupportedException**: Thrown when an invoked method is not supported or when there is an attempt to use an unsupported operation.
2. **KeyNotFoundException**: Thrown when the specified key for accessing a collection does not exist.
3. **TimeoutException**: Thrown when the time allotted for a process or operation expires.
4. **InvalidTimeZoneException**: Thrown when time zone information is invalid.

**Checked Exceptions**

Use the checked keyword to explicitly enable overflow checking for integral-type arithmetic operations and conversions. For this, just set the checked keyword.

Overflow checking can be enabled by compiler options, environment configuration, or use of the checked keyword.

res = checked(val + 10);

Let’s say the value of val is 2147483647 i.e. the max value of int type. The above will raise an error since it is checked. This enables overflow checking at runtime.

**Unchecked Exception**

Use the unchecked keyword to prevent overflow checking for integral-type arithmetic operations and conversions. For this, just set the unchecked keyword.

Here, the arithmetic overflow is ignored. Use it to prevent overflow checking.

res =unchecked(val + 10);

Let’s say the value of val is 2147483647. The above will not throw an error since the overflow checking is prevented using the unchecked keyword.