PUCMM

Pontificia Universidad Católica Madre y Maestra

Pontificia Universidad Católica Madre y Maestra Campus Santo Tomás de Aquino

Facultad de Ciencias e Ingeniería

CSD-1770-1843 – Programación I

Alicia Santos

Tarea 2

Dayán Paniagua Robles 2019-5025

Freddy Antonio Cruz Valerio 2019-5033

Miércoles 14 de julio de 2021

clasificación de excepciones

En primera instancia hablaremos acerca de la ArgumentNullException. Básicamente lo que maneja este tipo de excepción es que una variable que este siendo utilizada en nuestro código no pueda ser nula, ay que de ser así saltaría la excepción. Esto aplica principalmente para constructores parametrizados los cuales pueden trabajar con valores introducidos por el usuario mediante un Console.ReadLine. Dado que el si el usuario no introdujera ningún número este programa arrojaría un error de argumentación y seguido de esto se cerraría el programa es preferible implementar el ArgumentNullException y pedirle nuevamente al usuario que ingrese los datos, pero, esta vez de manera correcta.

Luego de esta tenemos la FormatException, está excepción se utiliza mayormente en los casos de conversiones con el método .Parse(). Dado que este busca se introduzca un tipo de dato especifico el cual casi siempre es un número, aunque también pueden ser letras. Un ejemplo de esto es cuando se posee un programa que dependiendo de los valores introducidos se realizan distintos tipos de cálculos matemáticos. Lógicamente el tipo de dato que se espera es int, float, double, long o alguna variante de estos, pero en ningún caso una cadena de caracteres o un booleano. Entonces lo que se hace es que al leer el dato con el Console.ReadLine y convertirlo de strig a un valor numérico, pues en esta misma línea de código se aplica la excepción para evitar el ingreso de este tipo de caracteres, evitando así el cierre inmediato y repentino de nuestro programa.

Después de esto tenemos a la ArgumentOutOfRangeException, la cual es utilizada en el caso de que tengamos un constructor o atributo al cual le asignemos valores. Dado que las variables de tipo numérico poseen un límite de caracteres, es decir, poseen un rango el cual les es imposible sobrepasar. por tanto, si a una de estas variables o atributos le asignáramos su valor cumbre + 1 nuestro programa simplemente terminaría cerrándose. Por lo cual para evitar este tipo de problema se utiliza esta excepción, explicándole así al usuario que hay un limite o rango preestablecido de datos o caracteres que puede admitir la variable de nuestro constructor o método. Un ejemplo de esto sería, en un programa que se construya para vender crédito para el móvil se solicite el número de teléfono, el cual posee diez caracteres y el tipo de variable que albergara este valor es tipo entero la cual solamente soporta diez siséis caracteres, por tanto, se debe de implementar la excepción por si en un caso aislado el usuario sobrepasa esta cifra el programa no se cierre de manera repentina y no sepa el porqué de esta acción.

De la mano de la anterior tenemos a la OverflowException, está es bastante parecida a la anteriormente mencionada debido a que ambas tratan el tema de cuando un parámetro sale de los limites preestablecidos. Como puede ser un int, double, long o cualquier otra de sus variantes. Pero este mayormente se utiliza para cuando una operación aritmética dentro de un método sobrepasa el limite que posee la variable que obtendrá el resultado de esta operación. Un ejemplo puede ser que tengamos una calculadora de cuadrados de números, la cual albergue el valor de el resultado en un numero entero, el cual no puede poseer más de diez siséis dígitos. Por lo tano si el usuario ingresa un numero suficiente grande el cuadrado de este sobrepasará los diez siséis dígitos, hará que se el programa explote y el usuario no sepa lo que pasó. Por lo cual se emplea esta excepción y se evitan este tipo de inconvenientes.

A continuación, tenemos la InvalidOperationException. Lo que busca este tipo de excepción es que al realizar un tipo de operación entre valores los cuales no se puede el programa no se cierra o explota. Estas operaciones pueden ir desde una conversión del tipo de dato con el Console.ReadLine.Parse() hasta a una simple operación aritmética como puede ser una suma o resta. Un ejemplo es que se posea un programa que calcule el área de un triángulo, el cual es base por altura entre dos. Entonces para esto se necesita que el usuario introduzca los datos por teclado, pero, por alguna razón el usuario introduce una letra u otro carácter en lugar de un digito entero o decimal. Lo que ocasiona lógicamente un error y seguidamente el cierre del programa. Para evitar esto utilizamos el InvalidOperationException y le pedimos de manera seguida al usuario que introduzca un valor valido esta vez

Por último, pero no por esto menos importante, tenemos, la ArgumentExcepcion, esta excepción se produce cuando uno de los argumentos para un método o constructor no es valido o admitido. Pongamos el ejemplo anterior en el cual se quería calcular el área de un triángulo y para esto se crea un método llamado área, al cual le pasaremos los parámetros de las medidas de los lados en valores enteros. Digamos que en lugar de esto el usuario digitase un numero decimal o cualquier otro carácter existente. Esto lo que ocasionaría seria un sierre de el programa sin proporcionarle ningún feedback al usuario de cual fue su error o mas bien cual es el error. Por lo cual para estos casos se implementa este tipo de excepciones y se mayormente se pone un bucle con la lectura de datos y las instrucciones al usuario de que debe de hacer para que este puede corregir su error y completar la ejecución del programa de manera eficaz.

Referencias

Manejo de excepciones. (s. f.). Microsoft Docs. Recuperado 14 de julio de 2021, de https://docs.microsoft.com/