Exception Handling in C#

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

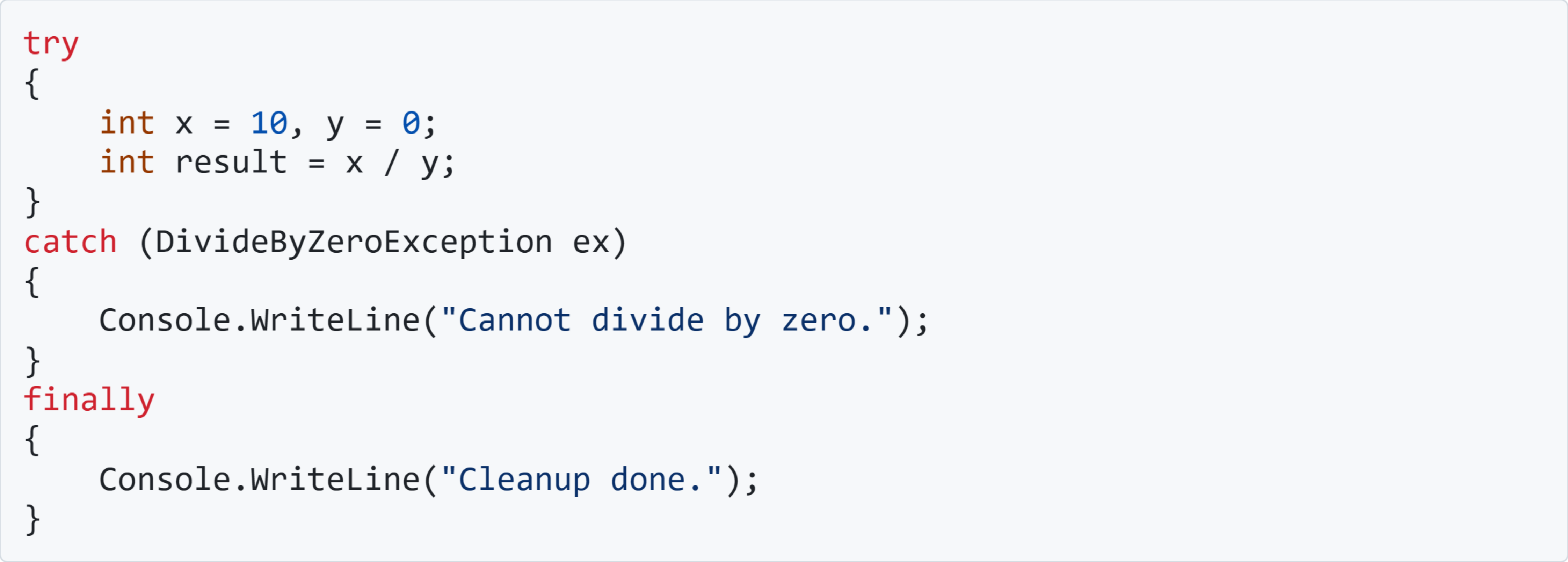
1. Introduction to Exception Handling

Exception Handling is a mechanism in C# to handle runtime errors gracefully, ensuring program flow continues or fails safely.

Goal: Prevent application crashes by managing exceptions in a structured way.

```
try
{
    // Risky code
}
catch (Exception ex)
{
    // Handle the exception
}
finally
{
    // Cleanup code
}
```

2



```
try
{
    int x = 10, y = 0;
    int result = x / y;
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Cannot divide by zero.");
}
finally
{
    Console.WriteLine("Cleanup done.");
}
```

3. try, catch, finally Blocks

`try` Block

- Contains code that may throw an exception.

`catch` Block

- Handles specific exceptions.

`finally` Block

- Executes regardless of an exception occurring or not. Used for cleanup.

4. Common Exceptions

`DivideByZeroException`

Occurs when dividing by zero.

```
int x = 10;
int y = 0;
int result = x / y; // Throws DivideByZeroException
```

`NullReferenceException`

Occurs when accessing members of a null object.

```
string str = null;
Console.WriteLine(str.Length); // Throws NullReferenceException
```

5. Catching Multiple Exception Types

```
try
{
    // Some code
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Math error: " + ex.Message);
}
catch (NullReferenceException ex)
{
    Console.WriteLine("Null reference: " + ex.Message);
}
catch (Exception ex)
{
    Console.WriteLine("General exception: " + ex.Message);
}
```

6. Exception Hierarchy and Flow

All exceptions derive from the base class `System.Exception`.

```
System.Object
   ↳ System.Exception
        ↳ System.SystemException
            ↳ System.DivideByZeroException
            ↳ System.NullReferenceException
```

Execution flow:

- If an exception is thrown, control moves to the first matching `catch` block.
- If no match, it propagates up the call stack.

7. throw vs throw ex

`throw ex`

- Resets the stack trace, losing the original error location.

`throw`

- Preserves original stack trace. Best practice.

```
try
{
    throw new Exception("Error");
}
catch (Exception ex)
{
    // throw ex; ❌ Stack trace reset
    throw;   //   Stack trace preserved
}
```

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com