예외 처리하기

이것이 C#이다

Contents

- 예외에 대하여
- try~catch로 예외 받기
- System.Exception 클래스
- 예외 던지기
- try~catch와 finally
- 사용자 정의 예외 클래스 만들기
- 예외 필터하기

12.1 예외에 대하여

- ❖ 프로그래머가 생각하는 시나리오에서 벗어나는 사건 (예외)
  - → 프로그램의 오류나 다운으로 이어지지 않도록 처리 (예외 처리)
- ❖ 예외 발생 코드와 결과

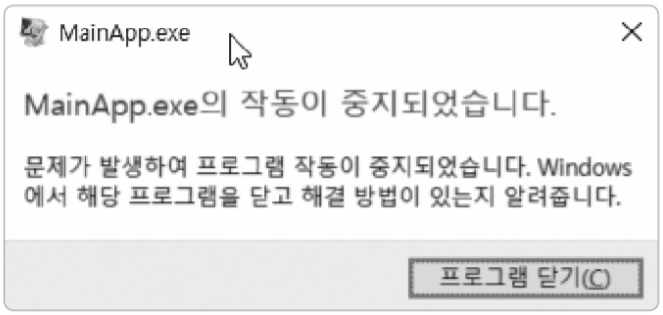
```
int[] arr = {1, 2, 3};

for (int i = 0; i < 5; i++)
{
    Console.WriteLine(arr[i]);
}
```

i가 "배열의 크기 –1"을 넘어서면 예외를 일으키고 종료됩니다. 이후의 코드들은 더 이상 실행되지 않습니다.

- ❖ 발생할 예외를 처리하지 않으면 → CLR이 처리

```
처리되지 않은 예외: System.IndexOutOfRangeException: 인덱스가 배열 범위를 벗어났습니다.
위치: KillingProgram.MainApp.Main(String[] args)
```



12.2 try~catch로 예외 받기

❖ 발생한 예외를 받아서 처리하는 방법

```
try
{
    // 실행하고자 하는 코드
}
catch( 예외_객체_1 )
{
    // 예외가 발생했을 때의 처리
}
catch ( 예외_객체_2 )
{
    // 예외가 발생했을 때의 처리
}
```

❖ 메커니즘

- try **절에서 원래 실행하고자 했던 코드 처리**
- **예외**(if **여러 개**)**가 던져지면** catch **블록**(**여러 개**)**에서 받아서 처리**
- Catch **예외 형식** = try **블록에서 던진 예외 형식**

❖ 데모 예제 – TryCatch

12.3 System.Exception 클래스

- **모든 예외 클래스의 조상**
- **예외 클래스 상속 관계**
  - **→ 모든 예외 클래스** System. Exception **형식으로 간주**
  - System.Exception **형식 하나로 모든 예외 처리 가능**
  - **→ 예외 상황에 따라 섬세한 예외 처리 곤란**
  - **→ 예상한 예외 외의 예외까지 받아 낼 수 있는 장치로 사용 (마지막)**

```
try
{
}
catch( IndexOutOfRangeException e )
{
    //...
}
catch( DivideByZeroException e )
{
    // ...
}
```

```
try
{
}
catch( Exception e )
{
    //...
}
```

12.4 예외 던지기 (1)

- 예외를 던지는 방법 – throw 문
- 메소드 내에서 예외를 던진 경우
  - 메소드를 호출한 try ~ catch 문에서 받음

```
static void DoSomething(int arg)
{
    if (arg < 10)
        Console.WriteLine("arg : {0}", arg);
    else
        throw new Exception("arg가 10보다 큽니다.");
}

static void Main()
{
    try
    {
        DoSomething(13);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
```

예외를 던졌지만 DoSomething() 메소드 안에서는 이 예외를 처리할 수 있는 코드가 없습니다. 이 예외는 DoSomething() 메소드의 호출자에게 던져집니다.

DoSomething() 메소드에서 던진 호출자의 try~catch 블록에서 받습니다.

- 데모 예제 – Throw

12.4 예외 던지기(2)

❖ C# 7.0의 변화

- 식에서 사용 가능
- 조건 연산자에서 사용 가능

a는 null이므로, b에 a를 할당하지 않고 throw 식이 실행됩니다.

```
int? a = null;
int b = a ?? throw new ArgumentNullException();
```

```
int[] array = new[] { 1, 2, 3 };
int index = 4;
int value = array[
    index >= 0 && index < 3
    ? index : throw new IndexOutOfRangeException()
    ];
```

❖ 데모 예제 – ThrowExpression

12.5 try~catch와 finally

- 예외가 try 블록의 중요 코드 실행을 못하게 한다면?
  - 예, try 블록의 끝에 데이터베이스의 커넥션을 닫는 코드
- 자원 해제 같은 뒷마무리를 담당하는 finally 절

```
try
{
    dbconn.Open(); // dbconn은 데이터베이스 커넥션
}
catch( XXXException e )
{
}
catch( YYYException e )
{
}
finally
{
    dbconn.Close();
}
```

- 데모 예제 - Finally

12.6 사용자 정의 예외 클래스 만들기

- Exception 클래스를 상속해서 만든다.

```
class MyException : Exception
{
    // ...
}
```

- 필요한 경우
  - 특별한 데이터를 담아서 예외 처리 루틴에 추가 정보 제공
  - 예외 상황을 더 잘 설명하고 싶을 때

- 데모 예제 - MyException

12.7 예외 필터하기

- catch 절이 처리할 예외 객체에 제약 사항을 명시해서 조건을 만족하는 예외 객체만 예외 처리 코드 실행 (6.0)
- catch() 문 뒤에 when 키워드를 이용해 제약 조건 기술

```
class FilterableException : Exception
{
    public int ErrorNo {get;set;}
}
try
{
    int num = GetNumber();
    if (num < 0 || num > 10)
        throw new FilterableException() { ErrorNo = num };
    else
        Console.WriteLine($"Output : {num}");
}
catch (FilterableException e) when (e.ErrorNo < 0)
{
    Console.WriteLine("Negative input is not allowed.");
}
```

- 데모 예제 - ExceptionFiltering

12.8 예외 처리 다시 생각해보기

❖ 예외 처리 기능이 없었던 시절

- 에러 코드 반환
- 메소드의 반환 값은 에러코드, 결과는 출력 전용 매개변수

```
// 피제수가 0이면 음수를 반환한다. 그렇지 않으면 몫을 반환한다.
static int Divide(int divisor, int dividend, out int result)
{
    if ( dividend == 0 )
    {
        result = 0;
        return -5;
    }
    else
    {
        result = divisor / dividend;
        return 0;
    }
}
```

❖ 예외 처리 → 오류를 종류별로 묶어서 처리 가능

❖ 데모 예제 – StackTrace

Thank You !

이것이 C#이다