codingOn

값의 참조

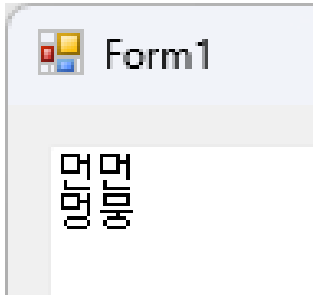
정적 메소드, 필드 static

- 인스턴스 생성 없이 사용 가능한 필드 또는 메소드
- Stack, Heap이 아닌 Static 영역에 할당 됨

```
3 references
class Dog
{
    public static string voice = "먼먼";
    1 reference
    public static string Vox() => voice;
}
```

```
1 reference
public Form1()
{
    InitializeComponent();

    textBox_print.Text += Dog.voice + "\r\n";
    Dog.voice = "멍뭉";
    textBox_print.Text += Dog.Vox() + "\r\n";
}
```



값의 참조 ref, out

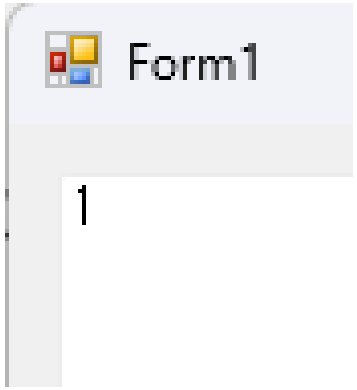
- 참조
  - 값을 복사하는 것이 아니라 값의 링크(주소값)을 전달하는 것

```
1 reference
public Form1()
{
    InitializeComponent();

    int num = 1;
    setThree(num);
    textBox_print.Text = num.ToString();
}

1 reference
void setThree(int x) => x = 3;
```

값이 복사됨



값이 복사(Copy)되었기 때문에
num 변수의 값은 변화가 없음

값의 참조 ref, out

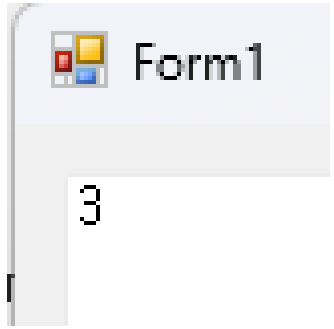
- 값을 복사하는 것이 아니라 값의 메모리 상의 주소값을 전달하기 때문에 스코프를 벗어나도 원래 값을 변경할 수 있음

```
1 reference
public Form1()
{
    InitializeComponent();

    int num = 1;
    setThree(ref num);
    textBox_print.Text = num.ToString();
}

1 reference
void setThree(ref int x) => x = 3;
```

값이 참조 됨



값이 참조(Reference)되었기 때문에
num 변수의 값이 바뀜

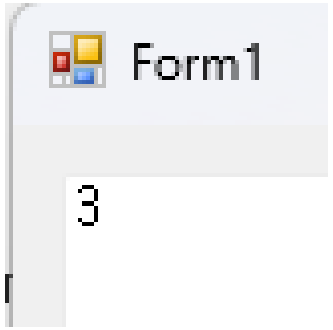
값의 참조 ref, out

```
1 reference
public Form1()
{
    InitializeComponent();

    int num = 1;
    setThree(out num);
    textBox_print.Text = num.ToString();
}

1 reference
void setThree(out int x) => x = 3;
```

out도 값의 참조(Reference)가 발생함



ref와 결과는 같음

값의 참조 ref, out

- 값 타입이 클 경우, 복사 비용이 크므로 참조를 통해 불필요한 복사를 피해서 메모리 효율이 향상하는 부가 효과 이점이 있음.
- 큰 사이즈의 클래스, 배열 등을 메소드에 전달 할 때 적극 이용
- ref
  - 메소드 밖에서 변수를 참조하여 가져옴
  - 따라서, ref 로 받아올 변수는 초기화가 필수 (비어있으면 안됨)

```
(local variable) int num

CS0165: Use of unassigned local variable 'num'
```

- out
  - 메소드 안에서 참조된 변수의 값을 바꿈
  - 따라서, 메소드 안에서 out 변수의 값을 반드시 바꿔줘야 함

```
void Form1.setThree(out int x)

CS0177: The out parameter 'x' must be assigned to before control
leaves the current method
```

실습. 값의 참조

- 다음 조건에 따라 void 형 메소드 2개를 작성할 것.

1. ref 키워드를 활용한 배열 채우기.

- 배열 생성 후, new로 생성된 배열을 ref로 메서드에 전달.
- 메서드 내부에서 배열의 각 요소를 **1부터 배열 길이만큼 순서대로** 채워 넣기. **(출력)**
- 예) 배열 길이가 5라면 [1, 2, 3, 4, 5] 로 채움.

```
[ref] 배열 값: 1 2 3 4 5
```

실습. 값의 참조

2. out 키워드를 활용한 배열 생성 및 채우기

- 초기화 되지 않은 배열 변수를 out으로 전달.
- 메서드의 두 번째 인자로 원하는 배열의 크기(int)를 함께 전달.
- 메서드 내부에서 해당 크기만큼의 배열을 생성한 뒤, 요소를 **1부터 크기 만큼 순서대로 채워 넣기. (출력)**
- 예) 크기 3이면 [1, 2, 3]

```
[out] 배열 값: 1 2 3
```

- **GitHub Repo. URL**과 **결과 창**을 슬랙 댓글로 제출

예외처리

예외처리란?

예외(Exception) 란?

- 프로그램 실행 도중 발생하는 **에러 상황**
- 예: 0으로 나누기, 없는 파일 열기, 배열 인덱스 초과 등

예외처리(Exception Handling)란?

- 프로그램 실행 중 예외가 발생했을 때,
  프로그램이 갑자기 멈추지 않도록 안전하게 처리하는 방법

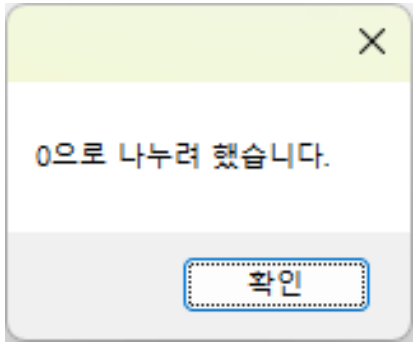
예외처리문

- try
  - 오류가 발생 할 가능성이 있는 소스코드를 작성
  - 예상 가능한 오류는 throw를 통해 의도적으로 catch를 작동시킴
- catch
  - try 안에서 오류 발생시 자동으로 catch로 넘어옴
  - 오류에 대한 내용이 적혀 있는 Exception 클래스 인스턴스를 전달 받을 수 있음
  - 원하는 종류의 오류 별로 각각 catch 문을 작성할 수 있음
- finally (선택)
  - 예외 발생 여부와 **상관없이 항상 실행**
  - 예외가 발생하고 catch로 잡지 못하더라도 finally는 실행

예외처리문

```
try
{
    // 만약 b가 0이라면 오류 발생
    double c = a / b;
}
catch // 어떤 오류든 일단 catch
{
    MessageBox.Show("알 수 없는 오류");
}
finally
{
    MessageBox.Show("정상 수행");
}
```

```
try
{
    // 만약 b가 0이라면 오류 발생
    double c = a / b;
}
// 오류 내용을 특정할 때는 일반 catch와 동시 사용 불가능
catch(Exception ex) // 오류 메시지 확인
{
    MessageBox.Show(ex.Message);
}
finally
{
    MessageBox.Show("정상 수행");
}
```

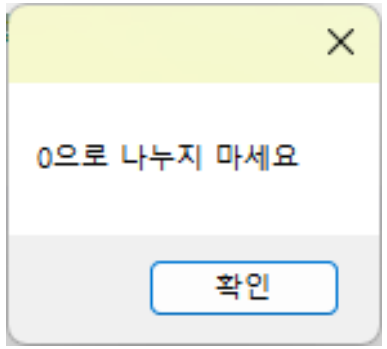


예외처리문

```
try
{
    // 만약 b가 0이라면 오류 발생
    double c = a / b;
}
// 오류 내용을 특정할 때는 일반 catch와 동시 사용 불가능
catch(DivideByZeroException)
{
    MessageBox.Show("0으로 나누려고 하셨군요?");
}
catch(Exception ex) // 0 나누기 이외의 오류를 catch
{
    MessageBox.Show(ex.Message);
}
finally
{
    MessageBox.Show("정상 수행");
}
```

```
try
{
    // 의도적으로 catch 발생 가능
    if (b == 0) throw new Exception("0으로 나누지 마세요");

    double c = a / b;
}
catch (DivideByZeroException)
{
    MessageBox.Show("0으로 나누려고 하셨군요?");
}
catch (Exception ex)
{
    // try에서 throw한 오류 메시지가 담겨있음
    MessageBox.Show(ex.Message);
}
finally
{
    MessageBox.Show("정상 수행");
}
```



* 예외 종류
https://parodev.tistory.com/8

codingOn

실습. 예외처리 – 회원가입 닉네임 검사.

- 사용자가 입력한 닉네임을 검사하는 프로그램 작성.
- 특정 조건을 만족하지 않으면 예외(throw)를 발생시키고, 예외에 따라 맞춤 메시지 출력하기.

요구사항

1. 사용자로부터 닉네임을 **한 줄 입력** 받기
2. 다음 조건을 검사하여, 해당하지 않을 경우 직접 예외 발생시키기.
   (다음 페이지에 계속..)

실습. 예외처리 – 회원가입 닉네임 검사.

검사 조건	예외 클래스	예외 메시지
닉네임이 비어 있음	Exception	"닉네임을 입력해주세요."
닉네임 길이가 2자 미만	Exception	"닉네임은 2글자 이상이어야 합니다."
닉네임에 admin 포함됨	Exception	"닉네임에 'admin'은 포함할 수 없습니다."

3. 조건에 해당하지 않으면 "닉네임 등록 완료!”를 출력.

4. 모든 예외는 try ~ catch로 처리, 예외 발생 시 ex.Message 출력.

5. 마지막에는 "프로그램을 종료합니다.”를 finally로 구현.

6. **각각의 오류 메시지 출력 캡쳐** 및 **Repo 주소** 슬랙 댓글에 업로드.