05 - 07. 상수

목차

- 상수
- 열거형

상수

상수

- 우리가 의도하지 않은 동작(버그)이 발생하는 것은 데이터를 잘못 조작했기 때문이다.
  - (1) 식을 잘못 작성하는 경우 (2) 식은 올바르지만 조작 순서가 올바르지 않은 경우
- 즉, 수정이 가능한 데이터(가변 데이터)는 버그의 온상이라 할 수 있다.
- 따라서 수정이 불가능한 데이터(불변 데이터)를 최대한 활용하는 것은 강건한 프로그램을 만드는데 도움이 된다.
- 가변 데이터를 **변수(Variable)**라 하고, 불변 데이터를 **상수(Constant)**라 일컫는다.
- 또, 상수는 이해하기 어려운 리터럴[1]에 이름을 붙여 코드의 가독성을 높이기도 한다.[2]

[1]: 이런 리터럴을 **매직 넘버(Magic Number)**라고 한다.
[2]: 이런 상수를 **기호 상수(Symbolic Constant)**라 한다.

상수

- 상수는 const 한정자로 만들 수 있다.
  - **한정자(Modifier)**: 구성 요소에 제약을 거는 키워드
- const는 객체에 사용할 수 있고, 불리언, 문자열, 숫자에만 사용 가능하다.
- const 객체를 참조하는 곳은 모두 컴파일 타임에 실제 값으로 대체된다.
- const 한정자는 타입 앞에 붙이며, PascalCase로 명명한다.

```
const int SpeedOfLight = 299_792_458;
const string WelcomeMessage = "Greetings!";
const bool IsWindows = true;
```

열거형

열거형

- 프로그래밍을 하다 보면 연관된 여러 상수를 만들 때가 있다.[1]
- 예를 들어, 계절을 표현하고자 한다면 아래와 같이 나타낼 수 있다.

```
const int Spring = 0;
const int Summer = 1;
const int Autumn = 2;
const int Winter = 3;
int season = Spring;
```

- 이는 한계점이 존재한다.
  - 값의 유효 범위를 알기 어렵다.
  - 타입이 실제 값과 연관이 없어 가독성이 떨어진다.

[1]: 프로그래밍은 약속을 잘 해야 한다.

열거형

- 열거형(Enumeration)은 연관된 정수 타입의 상수를 묶을 수 있는 도구다.

```
enum Season
{
    Spring,
    Summer,
    Autumn,
    Winter
}
Season season = Season.Spring;
```

열거형

- 열거형 멤버의 값은 지정하지 않을 시, 자동으로 지정된다.
  - 첫 번째 멤버는 0이다.
  - 그 다음 멤버는 이전 멤버 값의 +1이다.
- **기반 타입(Underlying Type)**도 지정할 수 있으며, 기본값은 int다.[1]

```
enum ErrorCode : ushort
{
    None,
    Unknown = 1,
    ConnectionLost = 100,
    OutlierReading // ?
}
```

[1]: 물론 기반 타입을 명확한 이유가 없는 이상 바꾸지는 않는다.

열거형

- 열거형은 0과 암시적 변환이 일어나기에 주의해야 한다.
  - 따라서 버그 방지를 위해 0을 포함하는 것이 좋다. None, Unknown, Default 같은 상태를 이용한다.
- 기반 타입과는 명시적 변환이 가능하다.

```
public enum Season
{
    Spring = 1,
    Summer,
    Autumn,
    Winter
}

public class EnumConversionExample
{
    public static void Main()
    {
        Season a = Season.Autumn;
        Console.WriteLine($"Integral value of {a} is {(int)a}");

        Season b = (Season)1;
        Console.WriteLine(b);

        Season c = 0; // ! 암시적 변환
        Console.WriteLine(c);
    }
}
```

부록

더 나아가기

- 상수를 사용하는 것은 왜 중요한가요?
- Enum의 여러 메소드의 사용 방법을 정리해 봅시다.
- 일반적인 const int를 여러 개 선언하는 방식 대신 열거형(Enum)을 사용했을 때 얻을 수 있는 이점은 무엇인가요?
- 열거형의 멤버가 아닌 값을 명시적 변환을 하면 어떤 일이 생기나요? 확인해 보세요.
- 열거형은 비트 플래그로도 사용할 수 있는데, 문서를 읽어보고 어떻게 활용할 수 있는지 정리해 보세요.
- 열거형 포맷 문자열에는 어떤 것들이 있는지 여기를 참고하여 정리해 보세요.

참고자료

- Enum을 쓰자
- 버그를 유발하는 enum의 괴상한 기본 동작