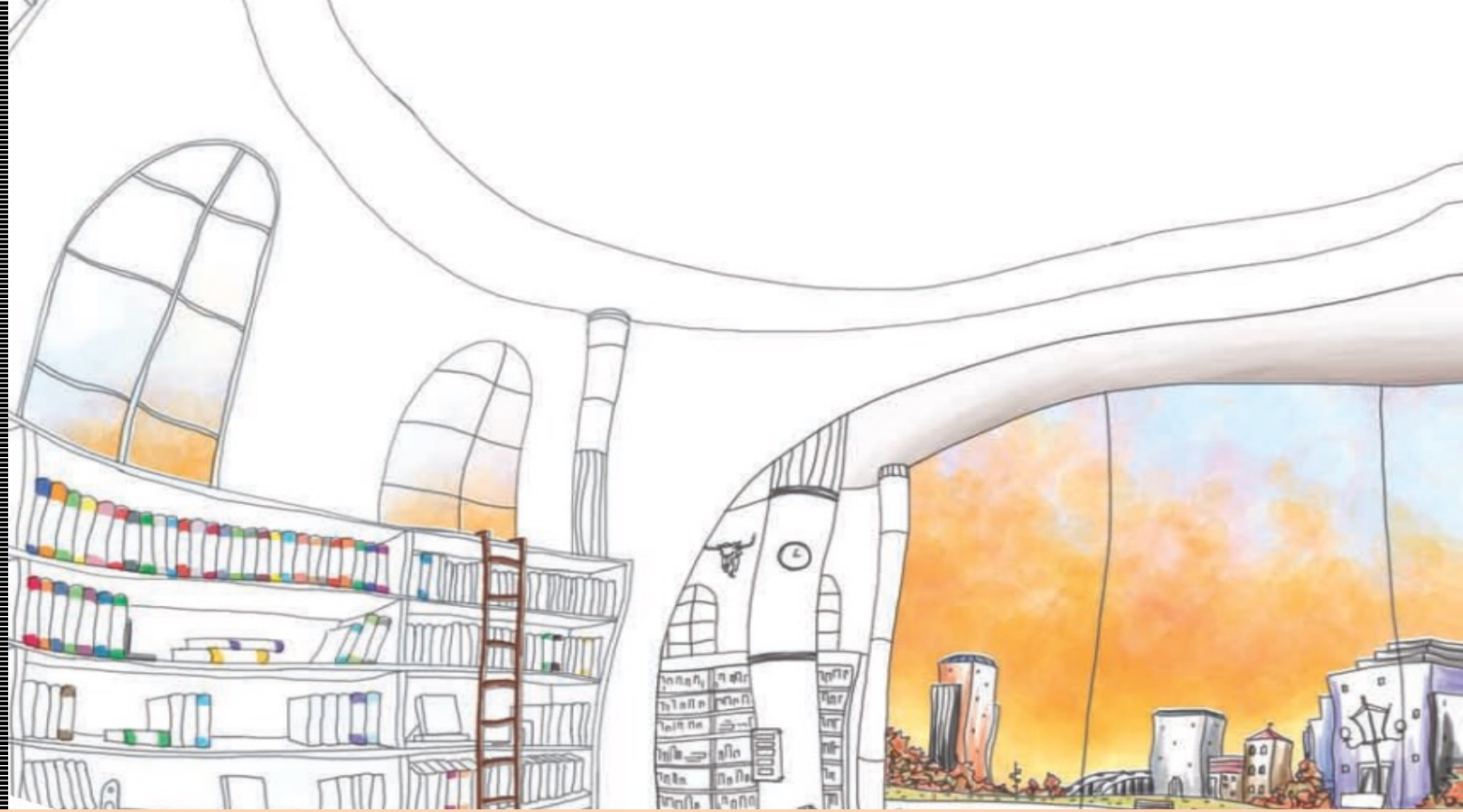


리플렉션과 애트리뷰트

이것이 C#이다

Contents

- 리플렉션
- 애트리뷰트

16.1 리플렉션

- **객체의 형식Type 정보를 들여다보는 기능**
  - 프로그램 실행 중에 객체의 형식 이름부터 프로퍼티 목록, 메소드 목록, 필드, 이벤트 목록 등 확인
- **형식의 이름만 있다면 동적으로 인스턴스를 만들고 인스턴스의 메소드 호출 가능**
- **새로운 데이터 형식을 동적으로 만들 수 있다.**
- **Object 형식의 GetType() 메소드**
  - 모든 형식을 들여다 볼 수 있는 장치

16.1.1 Object.GetType() 메소드와 Type 클래스(1)

❖ 모든 데이터 형식이 물려 받은 Object 형식의 메소드

- Equals(), GetHashCode(), GetType(), ReferenceEquals(),ToString()

❖ GetType() 메소드- Type 형식의 결과 반환

- Type 형식- .NET에서 사용되는 데이터 형식의 모든 정보 포함.

❖ Object.GetType() 메소드와 Type 형식을 사용하는 방법

```
int a = 0;

Type type = a.GetType();
FieldInfo[] fields = type.GetFields();

foreach (FieldInfo field in fields)
    Console.WriteLine("Type:{0}, Name:{1}", field.FieldType.Name, field.Name);
```

필드 목록 조회

16.1.1 Object.GetType() 메소드와 Type 클래스(2)

- 사용도가 높은 Type 형식의 메소드
- System.Reflection.BindingFlags 열거형

```
Type type = a.GetType();

// public 인스턴스 필드 조회
var fields1 = type.GetFields( BindingFlags.Public | BindingFlags.Instance );

// 비(非) public 인스턴스 필드 조회
var fields2 = type.GetFields( BindingFlags.NonPublic | BindingFlags.Instance );

// public 정적 필드 조회
var fields3 = type.GetFields( BindingFlags.Public | BindingFlags.Static );

// 비(非) public 정적 필드 조회
var fields4 = type.GetFields( BindingFlags.NonPublic | BindingFlags.Static);
```

- 데모 예제 - GetType

16.1.2 리플렉션을 이용해서 객체 생성하고 이용하기(1)

❖ 코드 안에서 런타임에 특정 형식의 인스턴스를 만드는 이점

- 프로그램이 동적으로 동작할 수 있도록 구성 가능

❖ System.Activator 클래스

- 동적 인스턴스 생성 Activator. CreateInstance()

```
List<int> list = Activator.CreateInstance<List<int>>();
```

❖ PropertyInfo 클래스

```
PropertyInfo name = type.GetProperty( "Name" );
PropertyInfo phone = type.GetProperty( "Phone" );

name.SetValue( profile, "박찬호", null );
phone.SetValue( profile, "997-5511", null );

Console.WriteLine("{0}, {1}",
            name.GetValue(profile, null),
            phone.GetValue(profile, null));
```

Type.GetProperties() 메소드는 그 형식의 모든 프로퍼티를 PropertyInfo 형식의 배열로 반환하지만, Type.GetProperty() 메소드는 특정 이름의 프로퍼티를 찾아 그 프로퍼티의 정보를 담은 하나의 PropertyInfo 객체만을 반환합니다.

16.1.2 리플렉션을 이용해서 객체 생성하고 이용하기(2)

❖ 리플렉션을 이용한 메소드 호출

- MethodInfo 클래스의 Invoke() 메소드

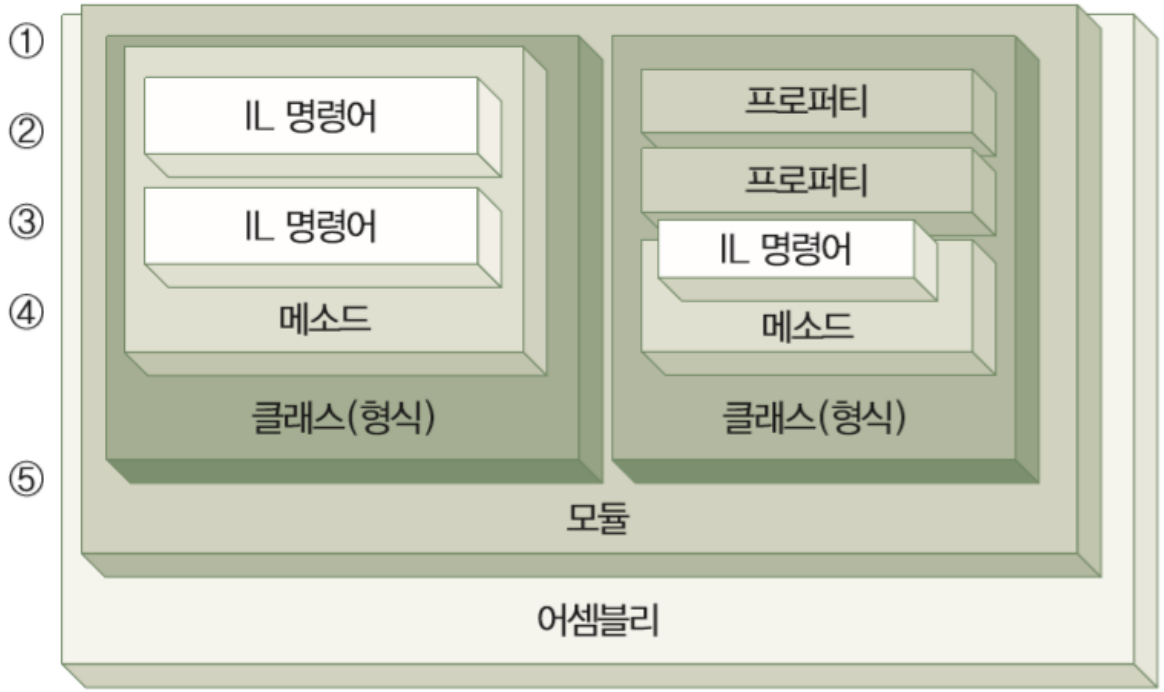
```
class Profile
{
    public string Name{ get; set; }
    public string Phone{ get; set; }
    public void Print()
    {
        Console.WriteLine( "{0}, {1}", Name, Phone );
    }
}
static void Main()
{
    Type type = typeof(Profile);
    Profile  profile = (Profile)Activator.CreateInstance( type );
    profile.Name = "류현진";
    profile.Phone = "010-1412-2222";

    MethodInfo method = type.GetMethod("Print");

    method.Invoke( profile, null );
}
```

null 매개 변수가 오는 자리에는 Invoke() 메소드가 호출할 메소드의 매개 변수가 와야 합니다. 여기에서는 Profile.Print() 메소드의 매개 변수가 없으므로 null 을 넘기는 것입니다.

❖ 데모 예제 - DynamicInstance

16.1.3 형식 내보내기

- **프로그램이 실행 중에 새 형식을 만들어 내어 CLR의 메모리에 “내보내는” 기능**
  - System.Reflection.Emit
  - →~Builder의 꼴의 이름을 갖는 클래스 제공(TypeBuilder, MethodBuilder 등)
- **클래스를 사용하는 요령**
- **.NET 프로그램의 계층 구조**



어 넣습니다.

니다.

PropertyBuilder 이용)를 만들어 넣

CPU가 실행할 IL 명령들을 넣습니다.

6.1.3 형식 내보내기(2)

❖ ~Builder와 ILGenerator를 이용해서 새 형식 만들어 보기

```
ILGenerator generator = newMethod.GetILGenerator();

generator.Emit(OpCodes.Ldc_I4, 1);

for (int i = 2; i <= 100; i++)
{
    generator.Emit(OpCodes.Ldc_I4, i);
    generator.Emit(OpCodes.Add);
}

generator.Emit(OpCodes.Ret); // 계산 스택에 담겨 있는 값을 반환합니다.
```

32비트 정수(1)를 계산 스택에 넣습니다.

32비트 정수(i)를 계산 스택에 넣습니다.

계산 후 계산 스택에 담겨 있는 두 개의 값을 꺼내서 더한 후, 그 결과를 다시 계산 스택에 넣습니다.

```
newType.CreateType();
```

```
object sum1To100 = Activator.CreateInstance(newType);
MethodInfo Calculate = sum1To100.GetType().GetMethod("Calculate");
Console.WriteLine(Calculate.Invoke(sum1To100, null));
```

❖ 데모 예제 - Emit

16.2 애트리뷰트

- **코드에 대한 부가 정보를 기록하고 읽을 수 있는 기능**
- **주석 vs. 애트리뷰트**
  - 주석 - 사람이 읽고 쓰는 정보
  - 애트리뷰트 - 사람이 작성하고 컴퓨터가 읽는 정보
- **시나리오 예**
  - 새 라이브러리를 배포하면서 이전 메소드의 보안 경고를 표시해야 하는 경우

16.2.1 애트리뷰트 사용하기

❖ 사용 형식

```
[ 애트리뷰트_이름( 애트리뷰트_매개 변수 ) ]
public void MyMethod()
{
    // …
}
```

❖ 사용 사례

- .NET 프레임워크에서 기본으로 제공하는 Obsolete 애트리뷰트 이용

오류 목록

0개의 오류 | 1개의 경고 | 0개의 메시지

		설명	파일	줄	열	프로젝트
⚠	1	'BasicAttribute.MyClass.OldMethod()'은(는) 사용되지 않습니다. 'OldMethod는 폐기되었습니다. NewMethod()를 이용하세요.'	MainApp.cs	25	13	BasicAttribute

❖ 데모 예제 - BasicAttribute

16.2.2 호출자 정보 애트리뷰트

- **메소드의 매개 변수에 사용됨.**
  - 메소드의 호출자 이름, 호출자 메소드가 정의되어 있는 소스 파일 경로, 소스 파일 내의 행 번호 파악
- **3가지 호출자 정보 애트리뷰트**
- **사용 사례**

```
public static class Trace
{
    public static void WriteLine(string message,
        [CallerFilePath] string file = "",
        [CallerLineNumber] int line = 0,
        [CallerMemberName] string member = "")
    {
        Console.WriteLine("{0}(Line:{1}) {2}: {3}", file, line, member, message);
    }
}
void SomeMethod()
{
    Trace.WriteLine("즐거운 프로그래밍!");
}
```

- **데모 예제 - CallerInfo**

16.2.3 내가 만드는 애트리뷰트

❖ System.Attribute 클래스로부터 상속

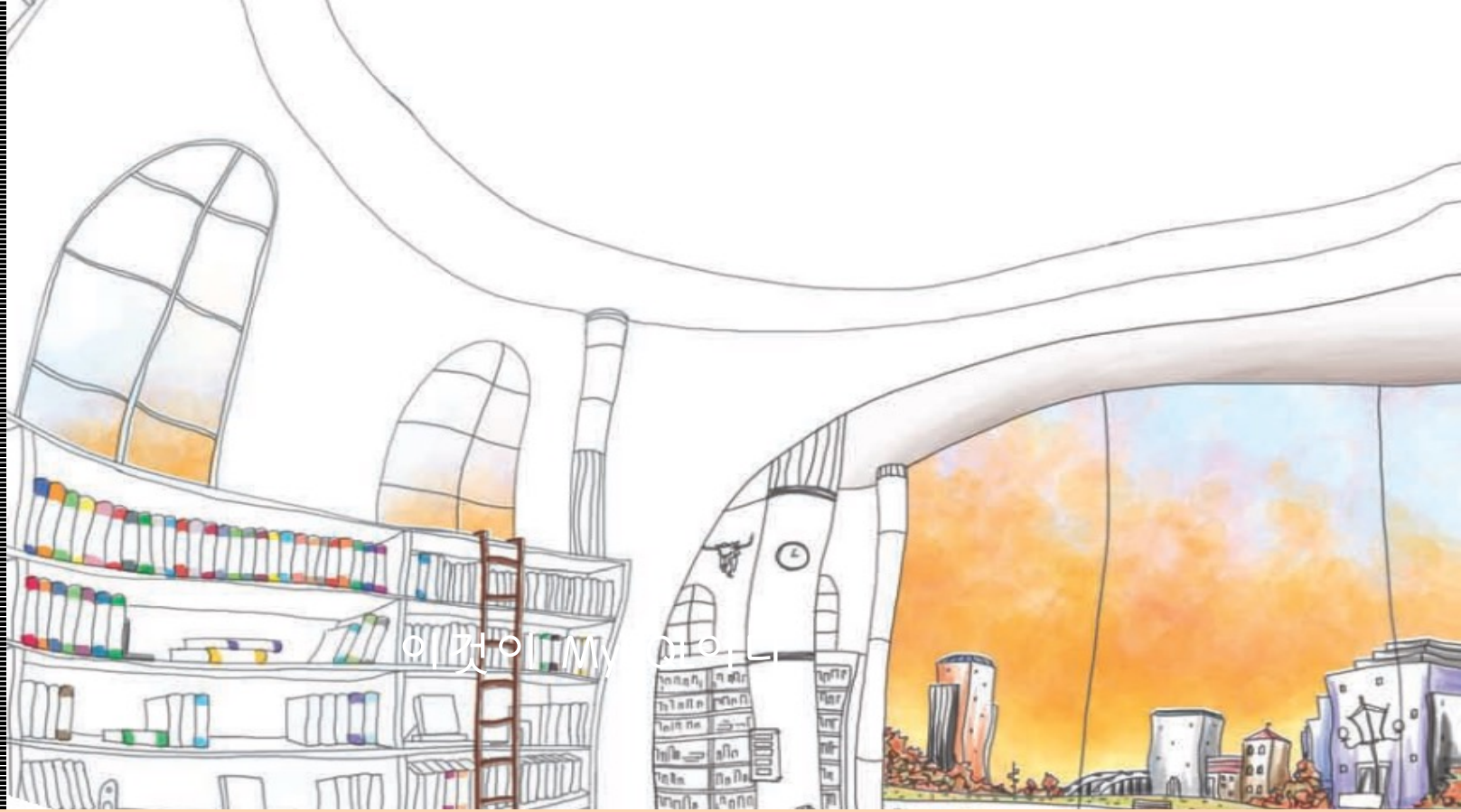
```
class History : System.Attribute
{
    //
}
```

❖ System.AttributeUsage

- 애트리뷰트의 애트리뷰트

```
[System.AttributeUsage(
    System.AttributeTargets.Class | System.AttributeTargets.Method,
    AllowMultiple=true)]
class History : System.Attribute
{
}
```

❖ 데모 예제 - Overloading



Thank You !

이것이 C#이다