응용소프트웨어실습 과제 Paper01

2018203048 박지홍

가비지컬렉터란?

C#에서는 CLR이 자동메모리를 관리한다. 이때, 가비지 컬렉션(Garbage Collection) 이라는 것이 프로그래머로 하여금 컴퓨터가 무한한 메모리를 가지고있는 것처럼 간주하고 코드를 작성할 수 있게한다. 즉 가비지라는 것은 더 이상 사용하지 않는 객체를 의미한다. 이 CLR 안에는 가비지 컬렉션을 담당하는 가비지컬렉터가 있다. 가비지 컬렉터는 객체 중에서 사용하는 것과 사용하지 않는 객체를 분리하여 사용하지 않는 것만 수거한다. 하지만 이 과정은 CPU와 메모리 자원을 소모하기 떄문에 최소한으로 사용할 수 있게 만들어 줘야한다.

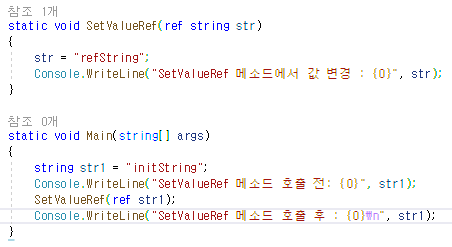
가비지컬렉터 동작 방식

할당된 메모리의 위치를 참조하는 객체를 루트라고 하는데 이 루트는 스택에 생성될 수도 있고 정적 필드처럼 힙에 생성될 수도 있다. 이때 .NET 어플리케이션이 실행되면 jit 컴파일러가 이 루트들을 목록으로 만들고, CLR은 이 루트 목록을 관리하며 상태를 갱신한다. 작업을 시작하기 전에 가비지컬렉터는 모든 객체가 쓰레기라고 가정하고 루트 목록 내의 어떤 루트도 메모리를 가리키지 않는다고 가정하는데 이 루트 목록을 순회하면서 각 루트가 참조하고 있는 힙 객체와의 관계 여부를 조사하고 루트가 참조하고 있는 힙의 객체가 또 다른 힙 객체를 참조하고 있다면 이 역시 해당 루트와 관계가 있는 것으로 판단하고 어떤 루트와도 관계가 없다면 쓰레기로 간주한다. 쓰레기 객체가 차지하고 있던 메모리는 비어있는 공간으로서 쓰레기의 인접 객체들을 이동시켜 채워 넣는다.

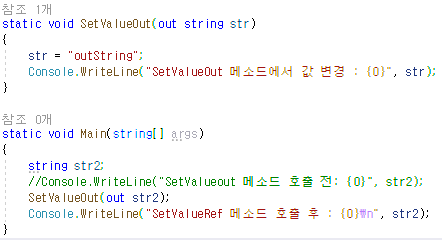
<ref 키워드와 out 키워드>

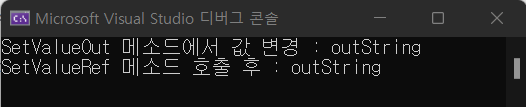
ref 키워드는 인수를 참조로 전달하는데 사용된다. 해당 매게 변수의 값이 메소드에서 변경되면 호출하는 메소드에 반영된다. 매개 변수가 ref로 전달되기 전에 초기화해야 하고 호출 메소드로 돌아가기 전에 매개 변수 값을 초기화할 필요는 없다. ref 매개 변수를 통한 값 전달은 호출된 메소드가 전달된 매개 변수의 값을 변경해야 할 때 유용하다. Ref를 사용하면 데이터를 양방향으로 전달 가능하다. (호출된 메소드 <-> 호출자 메소드)

out 키워드는 ref 키워드와 같은 인수를 전달하는데도 사용되지만 값을 할당하지 않고도 인수를 전달할 수 있다. 매개 변수가 전달되기 전에 초기화할 필요가 없다. 호출 메소드를 돌아가기 전에 매개 변수 값을 초기화해야 한다. out 매개 변수를 통한 매개 변수 선언은 메소드가 여러 값을 리턴할 때 유용하다. Out 사용할 때 데이터는 단방향 방식으로만 전달된다.(호출된 메소드 -> 호출자메소드)

텍스트이(가) 표시된 사진

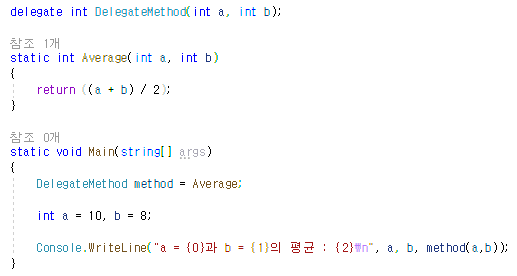
자동 생성된 설명

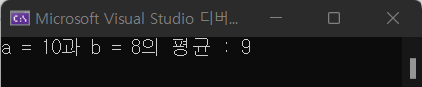




<대리자(Delegate)>

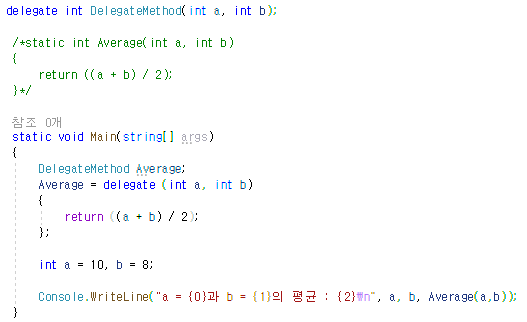
C언어의 함수 포인터의 역할을 함으로써 메소드의 위치를 가지고 있으면서 그 메소드를 대신 실행시켜주는 역할을 한다. 해당 메소드가 외무 어셈블리에 있거나 private 메소드이거나 런타임 도중 동적으로 바뀌는 경우 등 메소드를 직접 호출 할 수 없는 경우 대리자를 이용함으로써 해결한다.

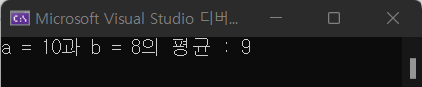




<익명 함수>

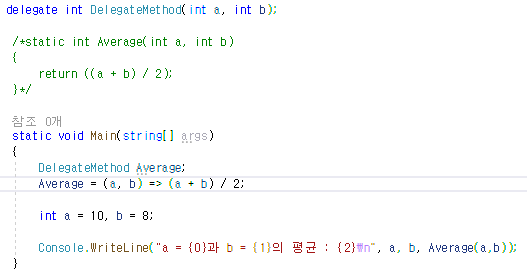
익명 함수는 메소드를 생성하지 않고도 위의 delegate를 생성할 수 있게 해준다. 미리 정의된 메소드를 delegate에 참조시키는 대신에 inline 형태로 메소드를 생성해서 delegate가 참조할 수 있도록 해준다.

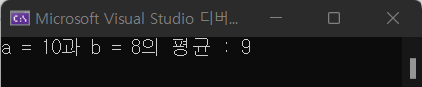




<람다식>

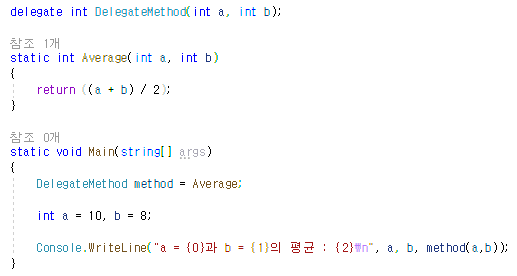
람다식은 위의 익명 함수를 쉽게 표한하는 식으로, delegate 타입 또는 expression tree 형태로 변환될 수 있다.

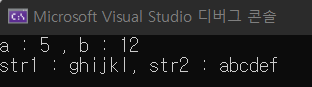




<제네릭(Generics)>

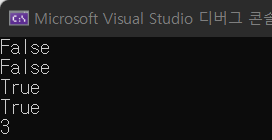
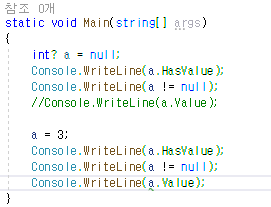
제네릭은 코드의 재사용, 알고리즘의 재사용을 최대화하기 위해서 사용하는 기능으로 기존의 클래스를 디자인 할 때가 아닌 클래스를 사용할 때 타입을 지정해준다. 하나의 클래스나 인스턴스에서 자신이 원하는 형식으로 선언하여 클래스나 인스턴스를 사용할 수 있다.





<Nullable 형식>

Nullable 형식은 비어있는 상태인 null을 나타낼 수 있는 형식이다. 비어있다는 말은 0을 넣거나 문자 자료형인 경우 “”을 넣어 아무런 문자를 할당하지 않은 경우가 아니라 그것들 조차 들어가있지 않은 상태를 의미한다. 이때, 변수 선언 시 아무런 값을 넣지 않음을 의미하지는 않는다. 변수 선언 시 아무런 값을 넣지 않았다 해도 이미 쓰레기값이 들어가 있기 때문에 null이 되었다고 할 수 없다. 선언 할 때 ‘데이터형식? 변수이름;’ 으로 선언한다.

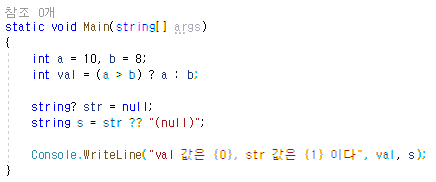


<?: 연산자, ?? 연산자>

?: 연산자와 ?? 연산자는 조건 연산자이다.

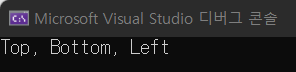
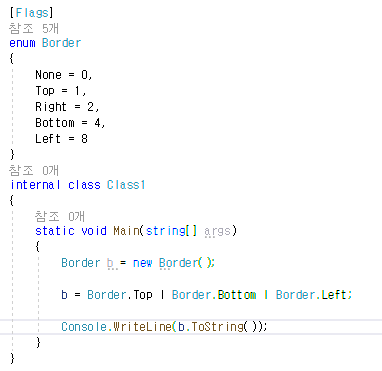
?: 예시) int val = (a > b) ? a : b; 는 a가 만약 b보다 크다면 val 값에 a를 대입하고 아니라면 val 값에 b를 대입한다.

?? 예시) string s = str ?? “null”; 는 변수 str이 null 이라면 문자열 s에 “null”이라는 문자열을 대입하고 아니라면 str의 값을 s에 대입한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

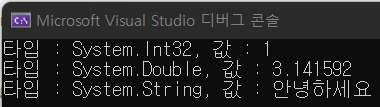
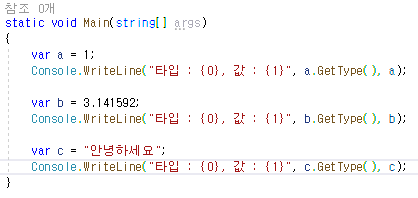
<FlagsAttribute를 갖는 enum 예제코드 작성>



Enum의 각 멤버들은 각 비트별로 구분되는 값들을 가질 수 있는데 이렇게 enum 타입이 비트 필드를 갖는다는 것을 표시하기 위해 [Flags]라는 Attribute를 지정할 수 있다. Flags 특성을 갖는 플래그 enum은 OR 연산자를 이용해서 enum 변수에 다중 값을 가질 수 있고 AND 연산자를 이용하여 enum 변수가 특정 멤버를 포함 하고 있는지 체크할 수 있다.

<var 형식>

Var 형식은 할당되는 데이터를 파악해 컴파일러가 자동으로 해당 변수의 형식을 지정한다. 그렇기 때문에 var 형식을 사용하려면 선언과 동시에 초기화를 해야 한다. 이때 var 형식은 지역 변수로만 사용할 수 있다. Var 형식이 초기화되는 데이터에 맞는 형식이 지정되면 이후에는 다른 형식의 값을 할당할 수 없다. 자동으로 지정된 형식을 확인하려면 GetType() 메소드를 사용한다.



<static 메소드 정의, [클래스이름].[메소드]방식으로 호출하는 예제코드 작성>

