https://smoothiecoding.kr/net-%ed%94%8c%eb%9e%ab%ed%8f%bc-clr/

.NET 플랫폼과 CLR (Common Language Runtime) – C# 초급 2

컴퓨터 언어는 필요에 의해서 발달합니다. 컴퓨터가 세상에 나온 이후 프레임워크의 중요한 방향성이 있는데 그것은 하나의 코드를 모든 시스템에 적용하는 것입니다. 자바의 슬로건이었던 write once run anywhere 입니다. 이것은 멋있는 말이지만 기존의 구조를 완전히 바꿔야만 가능합니다. 한번 작성해서 모든 시스템에서 동작하는 코드는 응용 프로그래머 레벨이고 그 코드가 여러 운영체제와 CPU에서 제대로 작동하도록 누군가 환경을 만들어 놔야 합니다. .NET 플랫폼에서 그 환경은 CLR (Common Language Runtime – 공통 언어 실행환경) 입니다.

CLR

Common Language Runtime – 공통 언어 런타임은 .NET 을 구성하는 핵심 요소입니다. 기존의 네이티브 프로그래밍과 비교해 보면 C언어는 소스 코드를 시스템에 맞춰서 오브젝트 코드로 컴파일 후 링커로 그 시스템에서만 작동하는 실행파일을 만듭니다. 처음 컴파일 할 때 소스 코드를 비롯한 모든 라이브러리를 네이티브 코드로 변환합니다. 메모리 관리는 직접해야 하고 일단 실행이 되면 CLR 처럼 중간의 개입이 없습니다.

반면 .NET 에서는 우선 C# 소스 코드를 컴파일하여 어셈블리를 만듭니다. 어셈블리는 소스코드와 네이티브 기계어의 중간적인 코드로써 CIL (Common Intermediate Language – 공통 중간 언어)와 타입 메타데이타, 보안 정보가 들어있습니다. 이것은 실행시간에 JIT 컴파일러에 의해서 운영체제와 하드웨어에 맞는 네이티브 코드로 변환되서 실행됩니다. 이 네이티브 코드를 런타임에서 책임지고 실행시키는 것이 CLR 입니다. CLR은 메모리 관리와 GC(가비지 콜렉션), 코드 검증, 예외 처리 등을 하며 C# 프로그램이 시작하고 끝날 때까지 관리합니다. 한번 실행된 코드는 캐시되기 때문에 반복실행 시 속도가 빨라집니다.

중간 코드인 CIL 로 변환하는 것은 자바 언어의 JVM의 바이트코드와 닮았습니다만 .NET은 C#을 비롯하여 VB .NET, C++ 등 약 30개의 CLI (Common Language Infrastructure) 언어가 하나의 CLR에서 함께 작동이 가능합니다. 이는 .NET의 넓은 통합과 확장성을 의미합니다. 다른 언어로 CLR에서 같은 BCL(Base Class Library)를 사용할 수 있습니다.

https://smoothiecoding.kr/csharp-hello-world-%ec%98%88%ec%a0%9c%eb%b6%84%ec%84%9d/

C# Hello World 예제의 분석 – C# 초급 3

첫번째 using System 은 System 네임스페이스를 사용하는 지시자입니다. 예제에 사용된 Console 은 System 네임스페이스에 포함된 type 입니다. 네임스페이스(namespace)는 우리말로 이름 공간으로 직역되는데 이것을 사용하는 근본적인 이유는 이름간의 충돌을 막기 위해서 입니다. 그래서 name(이름) 에 space(공간)을 주는 것 입니다. 간단한 원리지만 생소한 용어라서 초보자는 혼동하기 쉽습니다.

C#은 타입 안정성(type safety)을 중요시하는 언어입니다. 타입이 다르면 이름도 달라야 합니다. 그런데 수천개가 넘는 라이브러리의 이름이 중복된다면 충돌이 일어나게 됩니다. 예를 Console 이란 이름이 서버의 Console 일수도 있고 클라이언트의 Console 일수도 있습니다. 둘다 Console 이라고 객체를 생성하면 컴파일러는 어떤 것이 서버인지 클라이언트인지 알 수 없습니다. 그래서 앞에 네임스페이스를 붙여서 System.Console 을 서버의 Console 이라고 하고 Chrome.Console 을 클라이언트의 Console이라고 하면 이름이 충돌하는 문제가 해결됩니다.

두번째 namespace Smoothie 에도 네임스페이스가 나옵니다. 이것은 사용자의 커스텀 namespace 입니다. 이 안에서 생성하는 변수, 클래스, 인스턴스, 메소드 등 타입은 Smoothie 네임스페이스 안에서 사용가능합니다. 사용방법은 Smoothie.타입 입니다. 물론 여기서 생성한 타입을 다른 앱에서 사용하려면 위에서 처럼 using 지시자를 사용할 수 있습니다.

주의할 점은 using System을 사용하면 System 안에 있는 타입들이 다 들어오게 됩니다. Smoothie라는 네임스페이스 안에서 Console 이란 식별자를 재정의해서 사용할 수는 있으나 그렇게 되면 System.Console 은 우선순위에서 밀려서 사용할 수 없게 됩니다. 이 주제는 간단해 보이지만 변수의 영역(scope)과 객체지향프로그래밍과도 밀접한 관련이 있습니다. 아직 해당 주제를 학습하지 않았다면 조금 이해가 어려울 수도 있습니다.

세번째 class Program 입니다. class 키워드를 사용해서 클래스를 선언합니다. 클래스는 어떤 사물을 추상화 한 것 입니다. 뭐라도 클래스로 만들 수 있습니다… 이것은 해당 주제의 포스팅에서 자세히 다룰 것 입니다. C#은 객체지향 프로그래밍 언어로 프로그램을 클래스 단위로 작성하고 또 클래스 단위로 런타임에 로드하고 관리합니다. 소스 코드에는 보이지 않지만 클래스에는 여러가지 부가적인 정보가 딸려 있습니다. (버전, 메타데이타 등)

https://smoothiecoding.kr/csharp-identifier-keyword/

C# 식별자와 키워드 Identifier and Keyword – C# 초급 4

식별자 규칙

식별자 규칙은 컴파일러의 규칙입니다. 사람에게 가독성이 나쁜 코드도 컴파일만 되면 잘 돌아가지만 식별자의 규칙을 지키지 않으면 컴파일 에러가 납니다. 이것은 언어를 개발할 때 규칙을 정한 것 입니다.

* 영문 알파벳과 언더스코어 \_ 를 사용할 수 있습니다. (a~z, A~Z, \_)  
  – 사실 UTF-8 인코딩을 사용하는 현대의 컴파일러는 한글도 사용할 수 있습니다만, 아무래도 권장되지는 않습니다. 영어로만 코딩한다는 일종의 편견일 수도 있는데 현업에서 사용하기엔 무리가 있겠습니다. (개인 토이 프로젝트라면 말리지는 않음)
* 영문 대문자와 소문자를 구분합니다.  
  – 이것을 case-sensitive 라고 합니다. A와 a는 다른 문자입니다. 아스키코드가 다르니까 컴파일러에게는 다른 문자입니다.
* 숫자는 첫번째에 나올 수 없습니다.  
  – 예를 들어 2myAge 이건 안되고 myAge2 이건 됩니다. 숫자를 첫번째에 넣지 않는 것은 다른 언어의 컴파일러도 대부분 가진 규칙인데 소스코드의 숫자(아스키 코드)에는 리터럴 상수를 우선적으로 할당하는게 효율적이기 때문으로 보입니다. 또 가독성 측면에도 좋지 않겠지요.
* @는 식별자의 첫번째 문자로 사용가능합니다. 다른 위치에서는 허용되지 않습니다.  
  – @의 사용은 권장되지 않습니다.

https://smoothiecoding.kr/statement-block-csharp/

문(Statement)과 블록(Block) – C# 초급 5

문(Statement)

이렇듯 문은 변수를 정의하는 간단한 것 부터 수십개의 메소드를 체인하는 것 까지 범위가 넓습니다. 문(Statement)이 잘 이해가 안된다면 이런 특성을 이해하기 때문입니다. 문의 한계가 어디까지냐는 컴파일러 규칙과 사용자의 가독성 측면, 즉 유지보수 관점에서 볼 수 있습니다. 메소드 체이닝으로 한 줄에 쓸 수 있는 코드를 여러줄에 걸쳐 작성하는 프로그래머도 있습니다. 어느 것이 맞느냐는 사용자 관점에서 평가를 할 필요가 있습니다. 대체로 라이브러리와 프레임워크를 사용할 때는 문이 많이 줄어드는데 응용 프로그래머에게는 가리워진 부분들입니다. 실제로는 그 안에 여러가지 문이 있는데 인스턴스를 생성하거나 메소드 한 줄만으로 동작하도록 설계되어 있는 것 입니다.

블록(Block)

블록은 문(statement)이 모인 것 입니다. 중괄호({ } – curly bracket)로 영역을 표시합니다. 여러개의 문을 사용할 수 있다는 것은 더 복잡한 기능을 처리할 수 있다는 것 입니다. 또 가독성 면에서도 훨씬 유리하겠지요.

그리고 문을 사용하는 문법들이 많이 있습니다. for, while 루프 if 분기 class 등 여러가지가 있습니다. C#에서 블록은 특정 조건에 사용하는 규칙입니다. 그것들은 블록을 사용하는 해당 주제에서 다룹니다.

https://smoothiecoding.kr/csharp-console-writeline/

C# 기본 입출력 Console.WriteLine – C# 초급 6

콘솔(Console)

Console 클래스의 WriteLine 메소드는 콘솔창에 기본 출력을 합니다. 우선 콘솔이란 단어의 생소함을 어떻게 해봐야 겠습니다. 콘솔(Console) 영어의 뜻이 여러가지라 혼동하기 쉬운 단어입니다. 터미널(Terminal) 쉘(Shell) 과도 헷갈리는 단어이지요. 이 질문은 흔한 질문으로 유튜브 검색으로 봐도 많은 사람들이 혼동한다는 것을 알 수 있습니다.

콘솔은 CLI(명령 라인 인터페이스) 입출력을 표시하는 매개체입니다. 우리가 윈도우즈나 Mac 에서 사용하는 GUI 가 아니라 CLI 에서 보는 화면이 콘솔입니다. 물론 화면이 콘솔은 아닙니다. 콘솔은 사용자의 입력을 받고 출력하는 매개 역할을 합니다. 표준 입출력 개념을 알아야 하는데요. 컴퓨터의 입력 방법에는 여러가지가 있지만 표준은 키보드입니다. 마우스, 카메라, 마이크 이런 것들도 다 입력장치인데 기본은 키보드입니다. 콘솔은 키보드 입력을 받아서 모니터 스크린에 출력합니다.

String.Format(복합 형식 지정)

String.Format 메서드는 변수나 식의 값을 다른 문자열에 삽입하는 경우에 사용합니다. String 객체에 저장합니다. 보면 {0} {1} {2} 의 표시된 마커가 순서에 따라 후의 변수들로 대체됩니다. 문자열 결합 방식보다 정리가 잘 됩니다. 각 마커의 위치는 자유롭게 지정할 수 있고 여러번 사용될 수 있습니다. 마커의 인덱스인 { } 안의 숫자는 범위를 벗어나면 런타임 에러 (out of index range)를 발생시키므로 컴파일 시에 나타나지 않습니다. 필요한 경우 Exception 처리를 하도록 합니다.

문자열 보간 (String interpolation)

문자열 앞에 $ 가 나오면 보간된 문자열로 식별합니다. 위의 String.Format과는 작동방식이 차이가 있습니다. String.Format은 변수를 문자열 마커에 대체하는 방식입니다. 반면 분자열 보간은 $”문자열 {변수}” 처럼 변수를 직접 사용합니다. 더 쉽고 편리한 구문으로 이 기능은 C# 6부터 사용할 수 있습니다.

https://smoothiecoding.kr/csharp-comments/

C# 주석문 – C# 초급 7

싱글 라인 주석문

싱글 라인 주석문(한줄 주석문) 은 아래와 같이 사용합니다. 컴파일러는 // 다음의 텍스트는 그 다음 줄로 넘어가기 전까지 무시합니다. 한 줄을 다 사용할 수도 있고 세미콜론; 다음에 사용할 수도 있습니다.

비주얼 스튜디오의 디폴트 단축키는 Ctrl + K 다음에 Ctrl + C 입니다. 혹은 Ctrl + K를 누른 후 바로 C를 누르면 됩니다. 순서를 지켜야 적용됩니다. 해제는 Ctrl + K 다음에 Ctrl + U 혹은 Ctrl + K 를 누른 후 바로 U 를 누릅니다.

멀티 라인 주석문(구분자 주석문)

멀티라인 주석은 구분자 /\* \*/ 를 사용합니다. 구분자 안에 있는 모든 텍스트는 컴파일러가 무시합니다. 따라서 여러 줄에 걸쳐서 주석을 만들 수 있고 코드의 사이에 넣을 수도 있습니다.

비주얼 스튜디오 기본 단축키는 Ctrl + Shift + / 입니다. 주석문을 적용할 소스코드를 선택한 후에 단축키를 누릅니다. 해제도 마찬가지 입니다.

https://smoothiecoding.kr/csharp-type-concept/

C# 타입(Type) 개념, 변수, 데이터 – C# 초급 8

https://smoothiecoding.kr/csharp-classs-basics/

C# 클래스 기초 – C# 초급 9

그렇다면 클래스를 사용한다는 것은 무슨 말 일까요? 클래스는 일종의 빵틀같은 것이라고 했습니다. 설계도, 청사진이라고도 말합니다. 빵틀은 빵과 다릅니다. 빵틀이 있다고 해서 빵을 먹을 수는 없습니다. 자꾸 당연한 말을 하지만 당연한 것을 컴퓨터 안에서 이해하는 것이 생소할 수도 있습니다. 이제 빵을 찍어내고 구워서 먹어야겠습니다. 빵틀을 클래스라고 하면 빵틀에서 나온 빵은 인스턴스(instance) 혹은 객체(object) 라고 합니다. 그런데 클래스는 int 같은 기본 타입과는 다른 참조형 타입(reference type)입니다.

참조형 타입은 값을 직접 가지고 있는게 아니라 클래스의 참조, 즉 주소와 타입 정보를 가지고 있습니다.

앞서 만든 MyClass 변수를 선언해보겠습니다. 아래의 m1 은 MyClass 타입의 참조입니다. Main() 에서 선언하면 스택 메모리에 참조를 저장합니다.

참조 변수를 선언했다면 그 내용을 채울 MyClass 타입 데이터가 필요합니다. 참조 변수를 선언하면 초기화는 null 값이 들어갑니다. 이것으로는 아직 사용할 수가 없습니다.

다음 코드에서 MyClass 타입 인스턴스를 생성합니다. 클래스는 new 연산자로 힙메모리를 사용합니다. 인스턴스는 클래스라는 틀을 이용해서 실제 메모리에 생성한 객체입니다. m1 은 클래스 참조 변수이고 여기에 인스턴스를 할당해서 사용할 수 있습니다. m1 에서 public 으로 접근 가능한 변수와 메소드를 도트 연산자(.)로 사용합니다. [ex: m1.메소드( ) ] new 연산자는 C++ 부터 사용되던 방식인데 C언어에서 메모리를 할당하던 malloc 함수가 객체 지향 프로그래밍에 맞게 변화한 문법입니다. C++ 은 delete 명령으로 메모리를 직접 해제해야 하지만 C#에서는 GC(Garbage Collector)가 알아서 해제를 합니다.

https://smoothiecoding.kr/method-csharp-basic/

C# 메소드 기초 / Main 메소드 – C# 초급 10

Main 메소드 설명

- public static void Main(string[] arg)

public

위의 Program 클래스를 로드해서 사용하는 CLR의 입장에서 보면 Main 메소드는 private으로 접근합니다.

사실 Main을 public 으로 하는 것은 권장하지 않는데 CLR을 제외한 다른 클래스에서 접근 가능하기 때문입니다.

사실 이 부분은 같은 메소드 형태를 사용하는 자바와 유사하면서도 다른 것이 자바는 public을 기본으로 하고 C#은 public 이건 private이건 사용할 수 있습니다.

보통 앱을 만들 때는 Main 메소드를 다른 클래스에서 호출한다는 생각은 하지 않지만 가능은 하다는 것 입니다.

또 Main 함수의 재귀 호출도 가능합니다. 쓸일은 많이 없겠지만 일단 그렇습니다.

https://smoothiecoding.kr/csharp-local-variable/

지역 변수 – C# 초급 11