

C# 프로그램 기초

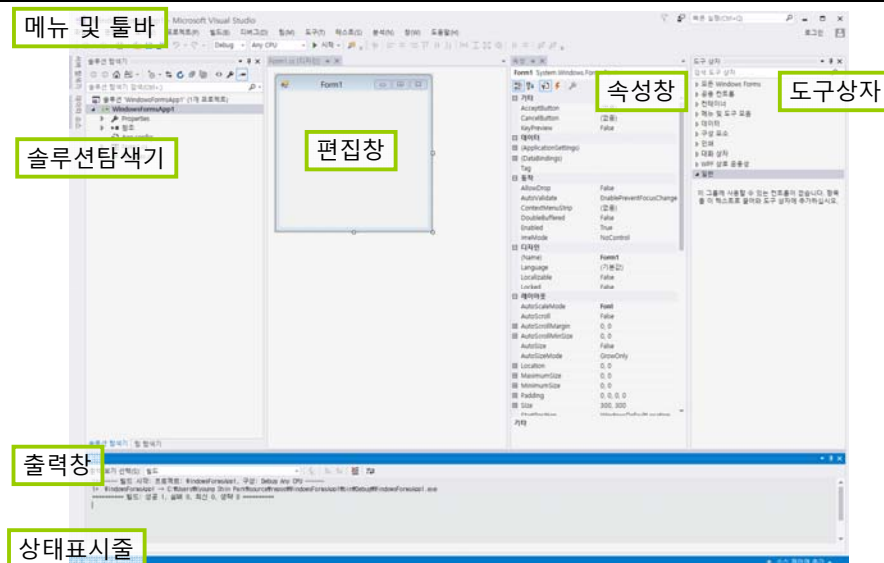
321190
2017년 가을학기
9/5/2017
박경신

Overview

- Visual Studio의 구성 요소
- C# 프로그램 기초
- C# 콘솔 프로그램 컴파일과 링크 및 실행

2

Visual Studio 2017 통합 개발 환경



Visual Studio

- The Editor
 - VB/C/C++/C# 코드를 작성하고 수정하기 위한 환경
- The Compiler
 - 소스 코드를 오브젝트 코드로 변환
- The Linker
 - 수행에 필요한 모듈들을 결합
- The Libraries
 - 미리 작성되어진 루틴들의 집합

4

Visual Studio

□ Project

- 프로그램을 구성하기 위한 모든 것을 담고 있는 것
- 프로젝트 폴더가 생성
- 프로젝트의 정보는 XML형태로 .csproj에 저장

□ Solution

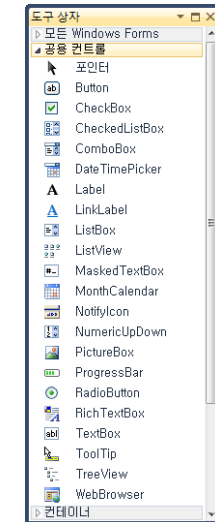
- 특정 문제를 해결하기 위한 모든 프로그램들과 다른 리소스들의 집합
- 하나 이상의 Project의 집합

5

Visual Studio

□ Toolbox (도구상자)

- 윈도우 컨트롤
- 웹폼 컨트롤
- 데이터 컨트롤
- 자주 사용되는 코드 등록 가능
- 컨트롤 추가 방법 - 컨트롤을
폼으로 드래그, 도구상자의
컨트롤 더블 클릭



Visual Studio

□ XML

- 데이터 교환의 표준으로 사용
- 데이터 교환 시 SOAP 이용
- XML과 XSD를 간단하게
작성할 수 있도록 지원

Visual Studio

□ Server Explorer (서버탐색기)

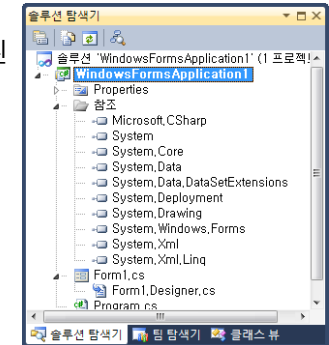
- 서버 상태 리소스를 한눈에
보임
- 현재 사용 중인 모듈과
프로세스의 리스트, 서비스
항목을 볼 수 있음

Visual Studio

- Code Editor (코드 편집기)
 - 코딩시 라인별 오류를 알려줌
 - 프로시저별로 코드 내용을 펼치거나 줄여서 볼 수 있음

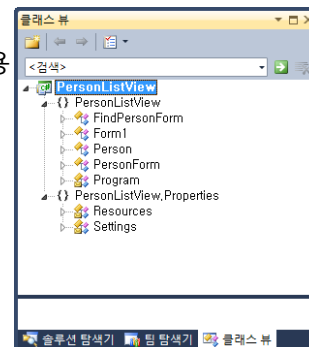
Visual Studio

- Solution Explorer (솔루션 탐색기)
 - 솔루션 – 프로젝트보다 큰 범위, 연관된 다수의 프로젝트 포함가능
 - 해당 솔루션을 구성하는 프로젝트들과 각각의 프로젝트를 구성하는 네임스페이스, 클래스, 소스 코드 등을 디렉토리화 하여 관리 가능
 - 팝업 메뉴를 통한 기능 지원



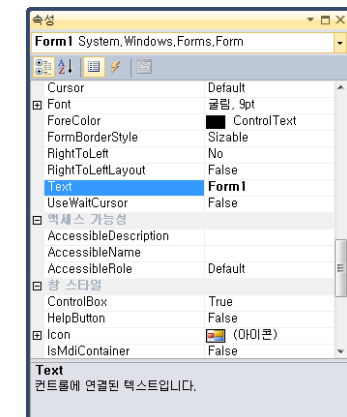
Visual Studio

- Class View (클래스 뷰)
 - 클래스에 대한 정보 표시, 편집 시 사용



Visual Studio

- Property Window (속성 창)
 - 컨트롤의 속성과 이벤트에 대한 설정 기능
 - 기본 내용에서 바뀐 부분은 볼드체로 표시
 - 컨트롤 더블 클릭 시 이벤트 핸들러에 자동으로 바인딩 기능 지원



Visual Studio

□ Help (도움말)

- 현재 개발자가 시행하고 있는 작업 내용을 자동으로 보여줌
- VS2010 부터 선호하는 웹 브라우저를 사용하여 온라인 또는 오프라인으로 문서를 볼 수 있으며, 최신 문서를 필요에 따라 다운로드하고 단순화된 목차를 사용하여 탐색가능

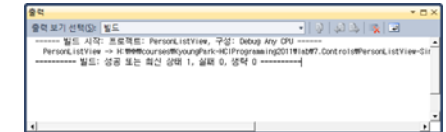


버튼에 대한 작업 시
동적으로 버튼에 대한
도움말을 보여줌 (F1)

Visual Studio

□ Task (작업 목록) & Output Window (출력 창)

- 프로젝트가 컴파일 될 때의 정보를 표시
- 직접 입력 및 VS.NET에 의한 자동 생성



C# 소개

- .NET CLR안에서 실행되는 언어 중 하나
- 다른 언어들의 장점을 도입하고 문제점을 제거하여 설계
- 데이터에 특정한 형식이 부여되면 그와 관계없는 형식으로 변환할 수 없음
- 형식의 안전을 위해 길고 장황한 코드를 작성필요
- .NET 코드라이브러리가 제공하는 모든 기능들을 완전하게 활용

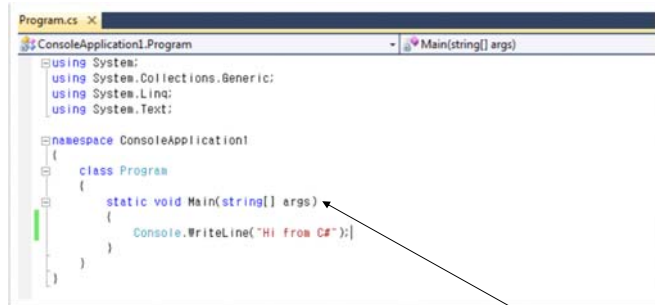
C# 소개

□ C#으로 작성할 수 있는 응용 프로그램

- Windows 응용 프로그램
- Web 응용 프로그램
- Web Service
- 데이터 베이스 액세스를 위한 응용 프로그램(ADO.NET)
- 네트워킹 구성 요소, 그래픽 출력, 복잡한 수학 연산 등을 위한 도구

C# 프로그램의 구조

- 객체 지향 언어, 클래스 단위의 프로그램 방식
- 반드시 하나 이상의 Main 메소드(method)를 갖는 클래스가 존재



Main() 메소드

C# 프로그램의 구조

- C# 프로그램의 구성
 - 프로그램 설명 (a.k.a header comments) (optional)
 - Library imports (optional)
 - 하나 이상의 클래스 (그리고 namespace)를 포함
 - 하나의 클래스(class)는 하나 이상의 메소드 (method)를 포함
 - 하나의 메소드(method)는 program statements을 포함

18

C# 프로그램의 구조

- 클래스

// comments about the class

class HelloWorld ← class header
일반적으로 클래스 이름은 대문자로 시작

{
class body
}

주석문(comments)은 어디에서나 사용가능

C# 프로그램의 구조

- class
 - C++나 Java의 클래스와 같은 개념
 - C# 소스 파일 하나 안에는 여러 개의 클래스가 존재 가능
 - 그러나 단일 클래스 하나를 C# 소스 파일 두 개에 연이어서 작성할 수 없음

C# 프로그램의 구조

□ 메소드

```
using System;
```

```
// comments about the class
```

```
class HelloWorld
```

```
{
```

```
    // comments about the method
```

```
    static void Main (string[] args)
```

```
    {
```

```
        Console.WriteLine("Hello World!");
```

```
        Console.WriteLine("HCI Programming II 수업입니다");
```

```
    }
```

```
}
```

C# 프로그램의 구조

□ namespace

- Java naming과 같은 개념
- 클래스들을 그룹핑한 단위를 네임스페이스라고 함
- 모든 .NET 라이브러리 코드는 네임스페이스로 정리되어있음
- 네임스페이스에 있는 코드를 참조하려면 반드시 해당 이름을 부르거나 (예, System.Console) 또는 명시적으로 import (예, using System;) 해야 함

```
using System;
namespace A {
    class MyClass {
        int value = 1;
        public void MyMethod() { Console.WriteLine(value); }
    }
}
namespace B {
    class MyClass {
        double value = 2.0;
        public void MyMethod() { Console.WriteLine(value); }
    }
}
```

```
namespace MyProgram {
    class Program {
        static void Main(string[] args)
        {
            A.MyClass a = new A.MyClass();
            a.MyMethod();
            B.MyClass b = new B.MyClass();
            b.MyMethod();
        }
    }
}
```

C# 프로그램의 구조

□ using

- Java에 import와 같은 것
- 다른 네임스페이스의 클래스를 접근하여 사용하고자 하면 using 문을 사용하여 해당 namespace를 접근함
- using문을 사용해서 System이란 네임스페이스를 열어서 Console.WriteLine의 사용이 가능해짐

```
using System;

class HelloWorld
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

```
class HelloWorld
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("Hello World!");
    }
}
```

C# 프로그램의 구조

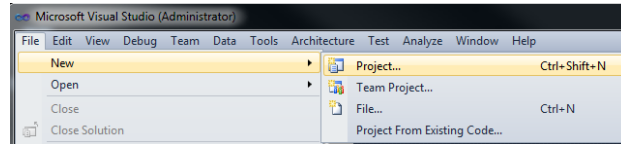
□ 닷넷의 namespace

- 네임스페이스의 기본적인 기능은 명칭의 선언 영역을 지정하여 명칭끼리의 충돌을 방지하는 것
- 닷넷 네임스페이스는 클래스를 기능별로 분류하는 역할로 함

Namespace	설명
System	타입, 메모리 관리 등 핵심 클래스
System.Collections	배열, 연결 리스트 등 컬렉션 클래스
System.IO	파일입출력 및 네트워크 관련 클래스
System.Windows.Forms	윈도우 폼과 컨트롤
System.Drawing	GDI+
System.Web	웹 개발에 관련된 클래스
System.Xml	XML 관련 클래스
System.Security	보완, 암호, 권한 관련 클래스

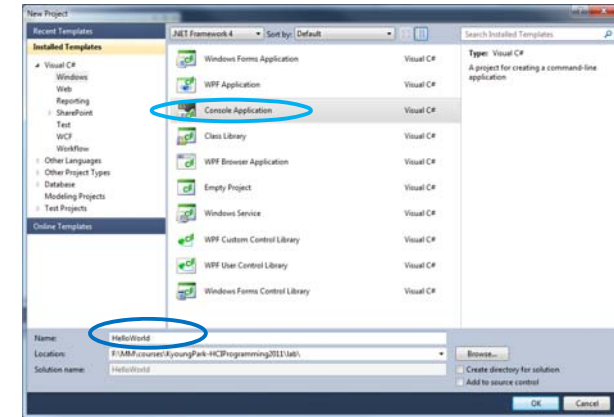
C# 콘솔 응용프로그램

- [파일->새로 만들기->프로젝트] 선택하여 콘솔 응용 프로그램 생성



C# 콘솔 응용프로그램

- Visual C# 프로젝트에서 콘솔 응용 프로그램 (Console Application) 선택
- HelloWorld 프로젝트 이름 입력

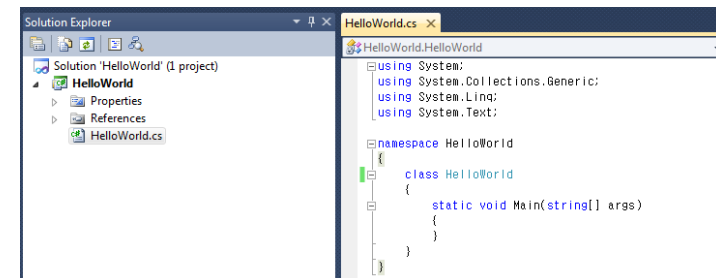


C# 콘솔 응용프로그램

- Program.cs의 이름을 HelloWorld.cs라고 변경
 - 오른쪽 마우스 이용하여 Program.cs파일을 선택한 후 속성 창에서 변경

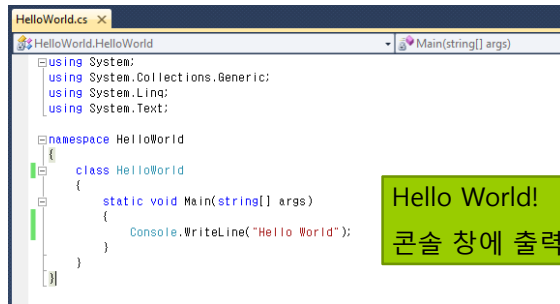
C# 콘솔 응용프로그램

- 기본으로 생성되는 코드



C# 콘솔 응용프로그램

□ 코드 추가

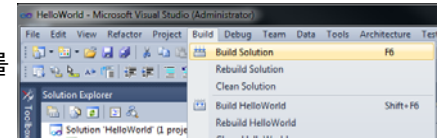


Hello World!
콘솔 창에 출력

C# 콘솔 응용프로그램

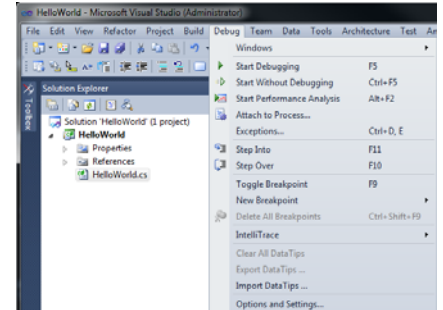
□ 컴파일과 실행

- 메뉴에서 [빌드->솔루션 빌드]를 선택하여 컴파일



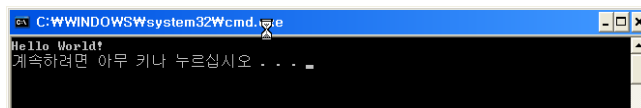
- [디버그->시작]을 선택하여 실행한다.

- 컴파일과 실행을 일괄적으로 처리하기 위해선 Ctrl+F5를 누르면 된다.



C# 콘솔 응용프로그램

□ HelloWorld의 결과 화면



C# 콘솔 응용프로그램

□ HelloWorld.cs 분석

- 자바와 같이 main() 메소드를 포함하고 있는 클래스 이름과 파일 이름이 같아야 할 필요 없음
- BCL(Base Class Library)중 **System** namespace 안에 정의된 클래스 사용 명시

```
using System;
```

- **HelloWorld** 란 이름의 **namespace**로 정의
유지보수와 프로그램 이해 이점

```
namespace HelloWorld  
{  
    ...//  
}
```


C# 콘솔 응용프로그램

- **class**란 키워드를 사용하여 HelloWorld 클래스 선언

```
class HelloWorld
{
    ...
}
```

- 메인 함수를 하나의 싱글 쓰레드 안에서 실행하기 위해 Attribute를 선언
 - Main() 메소드 안에서 멀티 쓰레드를 구현하기 위해서는 [MTAThread]로 설정

```
[STAThread]
```

C# 콘솔 응용프로그램

- **Main() 메소드는 응용프로그램의 시작점**

- static – 모든 클래스에서 공유하기 위한 멤버를 선언하는데 사용 객체를 생성하지 않고 시작이 가능
- string[] args – 매개변수, 명령 행의 명령어를 인자로 받음

```
static void Main(string[] args)
```

- Console 클래스는 콘솔 응용프로그램에 대한 표준 입출력 및 오류 스트림을 말함
 - Console 클래스의 WriteLine() 메소드를 이용하여 " " 내의 Hello World! 문자열 출력

```
Console.WriteLine("Hello World!");
```

C# 콘솔 응용프로그램

□ Write/WriteLine 메소드

- 스크린 모니터 상에 정보를 보여줌
- WriteLine은 캐리지 리턴 문자를 포함하여 출력 후에 다음 행으로 이동
- Write/WriteLine은 오버로드 되어있으므로 인자 형태로 숫자, 문자열 등 여러 형태가 가능
- {index [,alignment]:formatting}를 사용하여 인자를 출력
- Java에서와 같이 사용 – 예: "Test" + a

□ Read/ReadLine 메소드

- 키보드로 값을 입력받을 때 사용
- Read 메소드는 키보드로부터 하나의 문자를 입력받음
- ReadLine 메소드는 한 줄을 입력받음

C# 콘솔 응용프로그램

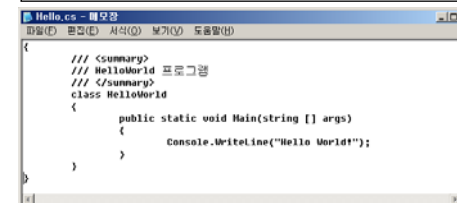
□ 주석문

- 컴파일 시 무시되며, 소스파일에서만 관리됨
- //를 사용하여 한 줄을 주석 처리함
- /* */를 사용하면 여러 라인을 주석처리 할 수 있음

□ XML 문서 만들기

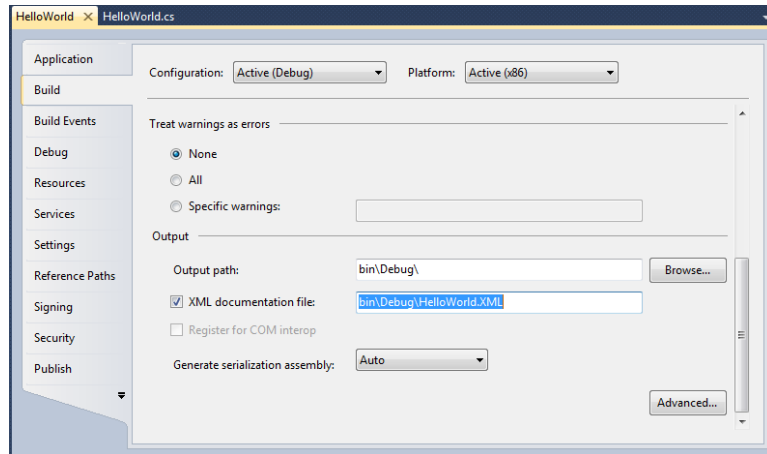
- 소스 상의 주석으로 입력했던 내용을 문서화할 때 사용
- ///로 XML 문서 부분을 표시해줌
- 컴파일 시 XML 문서옵션을 주어 XML문서를 만듦

```
~>csc.exe Hello.cs /doc:myComment.xml
```



XML 문서 만들기

Visual Studio에서 XML문서 만들기 설정



37

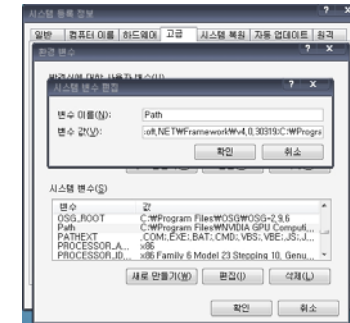
C# 콘솔 응용프로그램 메모장으로 작성하기

HelloWorld를 메모장으로 C# 프로그램 작성 및 실행

- 닷넷 프레임워크의 설치 및 환경변수의 PATH 경로확인
 - 아래 디렉토리 안에 C# 컴파일러인 csc.exe가 있는 지를 확인할 것

C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319

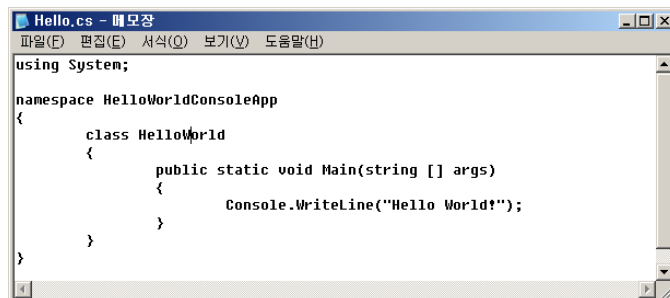
- 제어판 -> 시스템 -> 고급 -> 환경변수 -> 시스템변수 -> Path에 C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319를 추가



C# 콘솔 응용프로그램 메모장으로 작성하기

HelloWorld를 메모장으로 C# 프로그램 작성 및 실행

- 메모장에 C# 코드 작성 후 .cs라는 확장자로 저장

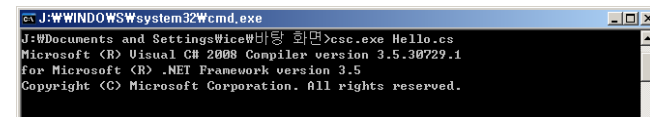


C# 콘솔 응용프로그램 메모장으로 작성하기

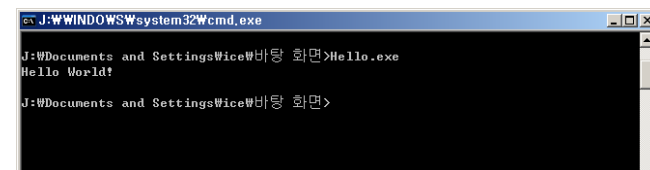
HelloWorld를 메모장으로 C# 프로그램 작성 및 실행

- 도스창에서 csc.exe 명령을 통해 컴파일

~>csc.exe HelloWorld.cs



- 생성된 실행파일(HelloWorld.exe) 실행



Console vs. Windows Application

□ 콘솔 응용프로그램 (Console Application)

- Visual component 지원하지 않음
- 텍스트 입력과 출력만 지원
- 도스창에서 실행

□ 윈도우 응용프로그램 (Windows Application)

- Graphical User Interfaces (GUI)를 통한 다양한 입출력 지원
- 메시지 박스
 - System.Windows.Forms 네임스페이스에 있음

```
using System;
using System.Windows.Forms;
class Welcome
{
    static void Main( string[] args )
    {
        MessageBox.Show( "WelcomeWntoWnC#Wnprogramming!" );
    }
}
```

