임베디드 시스템 소프트웨어

3주

강 대 기

차례

- 매니지드 코드
- 매니지드 코드와 네이티브 코드 비교
- .NET 프레임워크 소개
- .NET 프레임워크
- 공통 언어 스펙 (CLS)
- 공통 자료형 시스템 (CTS)
- 실행 모델
- 어셈블리와 역어셈블리 과정
- 공통 언어 런타임 (CLR)

매니지드 코드(Managed Code)

- 윈도우폰7에서 사용하는 프로그래밍 언어는 C#
- C#은 매니지드 코드 또는 관리형 코드라고 함
- 대표적 특징
  - 컴파일, 링크 과정에서 생성된 실행파일 자체로 실행될 수 없음
    - 반대로 이렇게 실행될 수 있는 코드를 네이티브 코드(Native Code)라고 함
  - 가상 머신과 같은 시스템이 운영체제와 프로그램 코드 사이에서 프로그램의 실행을 도와 줌
    - .NET의 경우 .NET Framwork / .NET Compact Framework
- PC, 인터넷, 윈도우폰과 같은 임베디드 장치, XBox, 준 플레이어 등에서 전부 실행 가능함.

매니지드 코드와 네이티브 코드 비교

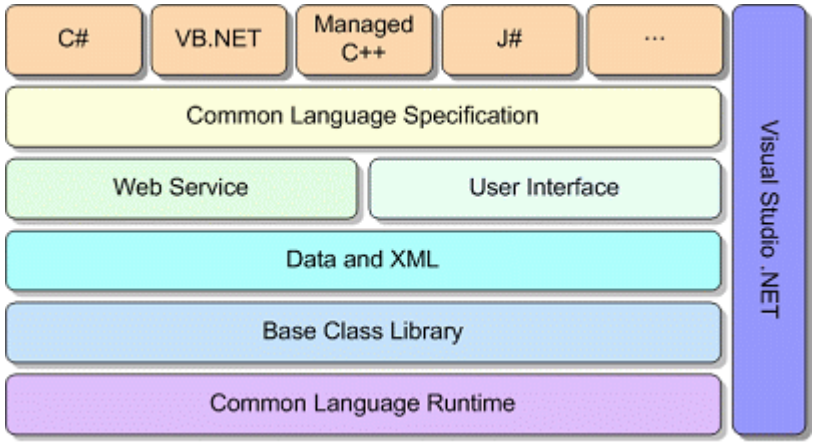
내용	네이티브 코드	매니지드 코드
대표언어	C/C++	Java, Python, C#
메모리 관리	수동 관리 프로그램 내부에서 직접 메모리 관리 메모리의 누수가 있을 수 있음	자동 관리 쓰레기 수집기(Garbage Collector)에 의해 자동적으로 관리됨
포인터	객체 포인터 사용 포인터 연산	객체 할당 및 사용
이식성	소스 코드 레벨의 이식성	실행 파일 레벨의 이식성
실행 조건	단독 실행 가능 •Win32 – 운영체제 기본 •MFC – MFC 관련 라이브러리의 도움으로 실행 가능	Java 는 가상 머신이 있어야 함 C#은 닷넷 콤팩트 프레임워크가 있어야, 실행 가능하며, 약 6MB 이상의 크기 요구

.NET 프레임워크 소개

- 공통 언어 스펙
  - CLS: Common Language Specification
- 공통 자료형 시스템
  - CTS: Common Type System
- 실행 모델
- 공통 언어 런타임
  - CLR: Common Language Runtime

.NET 프레임워크

- .NET 프레임워크(Framework)
  - 마이크로소프트사가 개발한 프로그램 개발 환경.
- .NET 프레임워크의 구조도



- 현재 지원 언어
  - C#, Visual Basic .NET, Managed C++, J#
  - CLS 만족하는 언어

공통 언어 스펙 (CLS)

- 언어의 상호 운용성을 지원하기 위한 스펙
- CLS
  - Common Language Specification
  - CLS를 만족하는 언어를 "공통 언어"라 부름
  - C#, Visual Basic .NET, J#, Managed C++
- CLS를 만족하면 서로 다른 언어에서 만들어진 라이브러리를 공유할 수 있다.
- 기본 클래스 라이브러리
  - .NET 프레임워크에서 제공
  - BCL : Base Class Library
  - CLS를 만족하면 BCL을 사용할 수 있다.

공통 자료형 시스템 (CTS)

- 다른 언어와 상호 운용성에 필요한 공통의 자료형
- CTS 형과 C# 자료형 관계

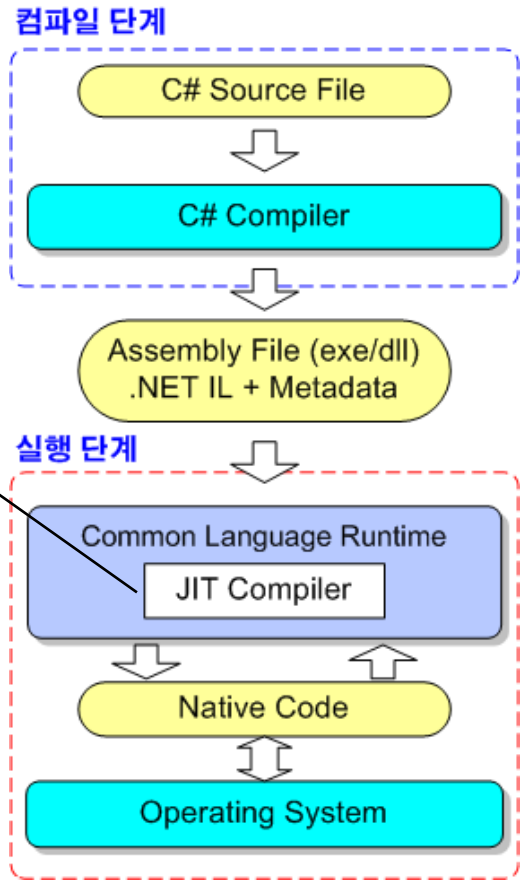
CTS 자료형	의 미	C# 자료형
System.Object	객체형	object
System.String	스트링형	string
System.Sbyte	부호있는 바이트형(-128 ~ 127)	sbyte
System.Byte	바이트형(0 ~ 255)	byte
System.Int16	16비트 정수형	short
System.Uint16	16비트 부호없는 정수형	ushort
System.Int32	32비트 정수형	int
System.Uint32	32비트 부호없는 정수형	uint
System.Int64	64비트 정수형	long
System.Uint64	64비트 부호없는 정수형	ulong
System.Char	문자형	char
System.Single	단일 정밀도 실수형	float
System.Double	이중 정밀도 실수형	double
System.Boolean	부울형	bool
System.Decimal	10진수형	decimal

실행 모델

- 컴파일 단계
- C# Compiler

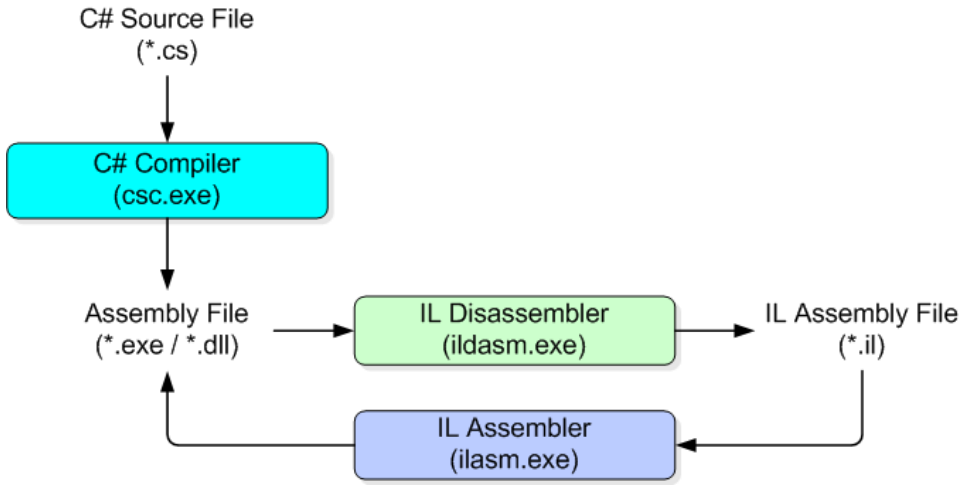
- 실행 단계
- Common Language Runtime

JIT 방법에 의해 실행



어셈블리와 역어셈블리 과정

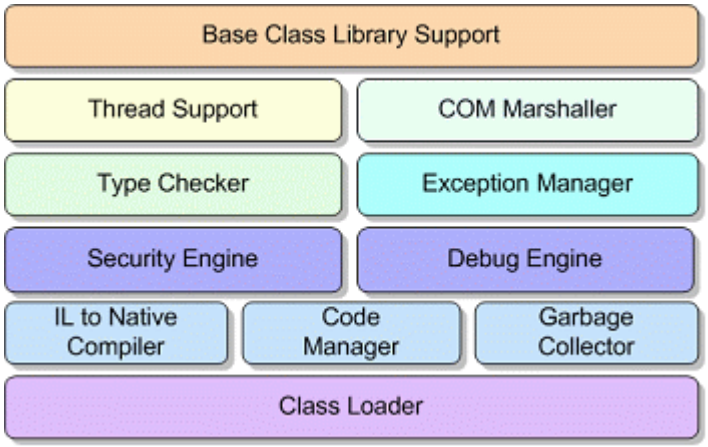
- 어셈블리 과정
  - *.il = *.exe or *.dll
- 역어셈블리 과정
  - *.exe or *.dll = *.il



- IL 파일
  - 텍스트 형태로 된 중간 언어 파일
  - 컴파일된 코드를 확인할 수 있음
  - 더 나아가 디버깅하는데 사용할 수 있음.

공통 언어 런타임 (CLR)

- .NET 프레임워크의 실행 시스템
- 자바의 JVM과 동일한 기능을 담당
- 실행 환경을 포함
  - 필수적인 실행 환경 3가지 컴포넌트
    - 메모리 관리기
    - 예외 처리기
    - 스레드 지원
- 공통 언어 런타임 컴포넌트



차례

- C# 소개
- 스마트폰/콘솔/윈폼 애플리케이션
- 콘솔 애플리케이션
- 윈폼 애플리케이션
- C# 프로그램 실행과정
- C# 활용분야
- C#의 자료형
- C#의 자료형
- 클래스
- 클래스의 설계
- 프로퍼티
- 연산자 중복

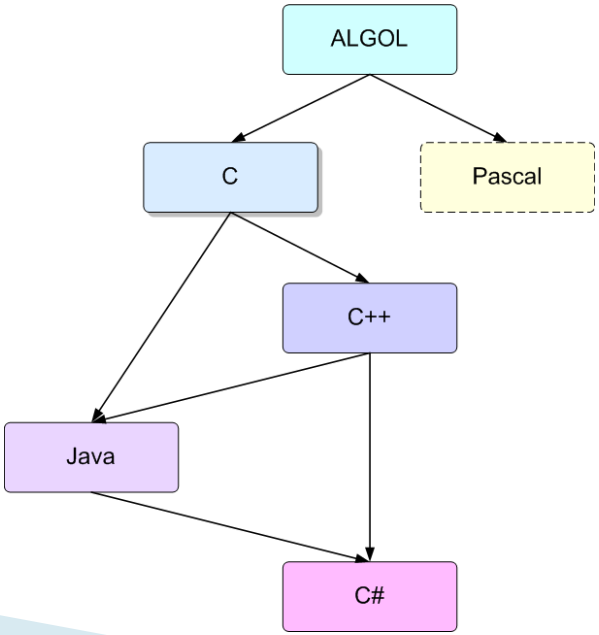
C# 소개 (1/2)

- C# 프로그래밍 언어
  - MS사의 앤더스 헬스버그(Anders Hejlsberg)가 고안
  - .NET에 최적화된 언어 (당연함)
  - 컴포넌트 지향 프로그래밍 언어
  - 자바의 단점을 보완했다고 '주장' (자바의 어떤 단점?)
  - 실행 방법: 자바: 인터프리테이션, C#: 컴파일 방법
  - 자바도 JIT 컴파일이 가능함
  - 자바 언어를 대체할 수 있는 언어라고 '주장'

- C#의 특징
  - 객체지향 언어: 자료 추상화
  - 델리게이트와 이벤트
  - 멀티스레드, 예외처리
  - 연산자 중복, 제네릭

C# 소개 (2/2)

- C 계열의 언어
  - C++와 자바로부터 영향을 받았음 (사실은 자바의 copy)
  - C: 연산자와 문장 등 기본적인 언어의 기능
  - C++: 객체지향 속성, 연산자 중복, 제네릭(Generic)
  - 자바: 예외처리와 스레드, (자바도 제네릭이 있음)
- C# 언어의 계통도



스마트폰/콘솔/윈폼 애플리케이션

- C# 개발 환경
  - SDK를 이용 – 편집기, 컴파일러, 실행엔진, 클래스 라이브러리
  - 통합개발 환경(IDE)
  - Visual Studio .NET

- 스마트폰 애플리케이션
  - 스마트 디바이스 환경에서 실행
  - GUI를 통해 입출력을 수행하며, 이벤트 처리 방식을 통해 실행됨
  - 통지(전화, 문자), 센서, 소셜 네트워크 서비스

- 콘솔 애플리케이션
  - 문자기반 명령어 프롬프트 환경에서 실행
  - 키보드를 통해 입력, 화면에 문자로 출력

- 윈폼 애플리케이션
  - 윈도우 폼 애플리케이션(Windows forms Application)의 약어
  - GUI를 통해서 입출력을 수행
  - 이벤트 처리 방식을 통해 실행

콘솔 애플리케이션

- 일반적인 응용 프로그램
- 예제 프로그램:

[HelloWorld.cs]

네임스페이스

```
using System;
class HelloWorld {
    public static void Main() {
        Console.WriteLine("안녕!");
    }
}
```

출력 메소드

실행 결과 :
안녕!

- 실행 과정

```
C:\Work>csc HelloWorld.cs
C:\Work>HelloWorld
안녕!
```

윈폼 애플리케이션

- 윈도우 응용 프로그램
- 예제 프로그램:

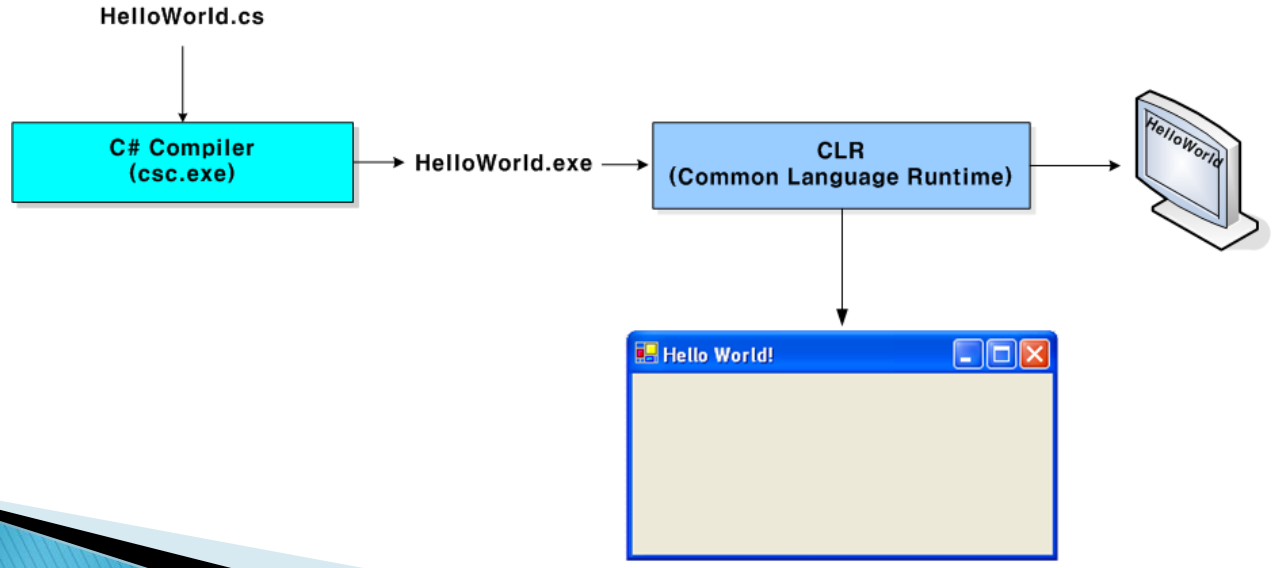
[WinFormApp.cs]

```
using System;
using System.Windows.Forms;
class WinFormApp : Form {
    WinFormApp() {
        Text = "Hello World!";
    }
    public static void Main() {
        Application.Run(new WinFormApp());
    }
}
```

실행 결과 :

C# 프로그램 실행과정

- 컴파일 과정
  - csc : C# compiler
- 실행 시스템
  - CLR – Common Language Runtime
- 실행 과정



C# 활용분야

- 윈도우폰7 응용 프로그램
- 데스크탑 용 윈도우 응용 프로그램
- 웹 프로그램

- 기존의 프로그램 환경
  - 콘솔 응용 프로그램 – C++
  - 윈도우 용 응용 프로그램 – MFC
  - 웹 용 응용 프로그램 – ASP

C#의 자료형 (1/4)

- 자료형
  - 변수나 상수가 가질 수 있는 값과 연산의 종류를 결정
  - C#의 자료형
    - 값형(value type)
    - 참조형(reference type)
  - 숫자형
    - 정수형
      - signed – sbyte, short, int, long
      - unsigned – byte, ushort, uint, ulong
    - 실수형 – float, double, decimal
  - 논리형 – true, false
- 연산자
  - 표준 C 언어와 유사
  - 형 검사 연산자(type testing operator)
    - is – 호환 가능한지를 검사
    - as – 지정한 형으로 변환

C#의 자료형 (1/4)

- 자료형
  - 변수나 상수가 가질 수 있는 값과 연산의 종류를 결정
  - C#의 자료형
    - 값형(value type)
    - 참조형(reference type)
  - 숫자형
    - 정수형
      - signed – sbyte, short, int, long
      - unsigned – byte, ushort, uint, ulong
    - 실수형 – float, double, decimal
  - 논리형 – true, false
- 연산자
  - 표준 C 언어와 유사
  - 형 검사 연산자(type testing operator)
    - is – 호환 가능한지를 검사
    - as – 지정한 형으로 변환

C#의 자료형 (2/4)

- 스트링
  - C#에서 스트링은 객체
  - System.String 클래스의 객체
  - C#의 string 형은 String 클래스의 alias
- 스트링 상수
  - 이중 인용부호(")로 묶인 문자들의 나열 (예 : "I am a string.")
- 스트링 초기화

```
string s = "Hello";
string s = new string("Hello");
```

- 스트링 연결
  - + 연산자 : concatenation operator

```
string s = "Hello";
s += " World";
=> s: Hello World
```

C#의 자료형 (3/4)

- 배열형
  - 같은 형을 갖는 여러 개의 값을 저장할 수 있는 자료구조.

- 배열 변수 선언
  - 배열을 가리키는 참조 변수

```
int[]     vector;
short[,]  matrix;
long[][]  arrayOfArray;
object[]  myArray1, myArray2;
```

C#의 자료형 (4/4)

- 배열 객체 생성
- new 연산자

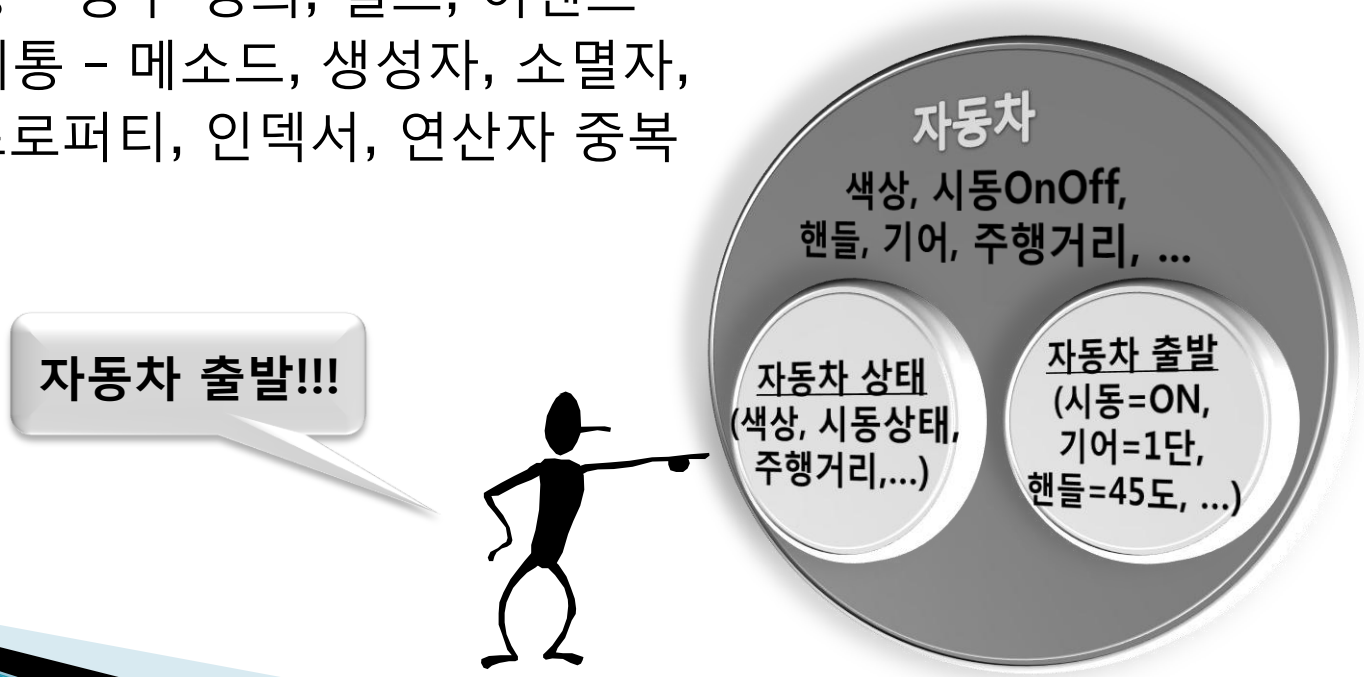
```
vector = new int[100];
matrix = new short[10,100];
myArray1 = new Point[3];
```

- 배열 사용

```
for (int i=0; i<vector.Length; i++)
        vector[i] = i;
```

클래스

- 클래스와 객체
- 프로그래밍 언어적인 측면에서
  - 객체 자료형 또는 객체 클래스
- 클래스의 구성 – 클래스 멤버
  - 필드 계통 – 상수 정의, 필드, 이벤트
  - 메소드 계통 – 메소드, 생성자, 소멸자, 프로퍼티, 인덱서, 연산자 중복



클래스의 설계

- 객체의 속성과 행위를 결정
- 속성 – 필드 계통
- 행위 – 메소드 계통

```
class CoffeeMaker {
    public bool  onState;
    public void StartCoffeeMaker() {
        if (onState == true)
            Console.WriteLine("The CoffeMaker is already on");
        else
            onState = true;
        Console.WriteLine("The CoffeMaker is now on");
    }
}
```

- CoffeeMaker 클래스를 이용하여 StartCoffeeMaker() 메소드를 호출하여 보시오.

프로퍼티 (1/2)

- 프로퍼티의 개념
  - 클래스의 private 필드를 형식적으로 다루는 일종의 메소드.
  - 값을 지정하는 셋-접근자와 값을 참조하는 겟-접근자로 구성.

- 프로퍼티의 정의 형태

```
[property-modifiers] returnType PropertyName {
    get {
        // get-accessor body
    }
    set {
        // set-accessor body
    }
}
```

프로퍼티 (2/2)

- 프로퍼티의 동작
  - 필드처럼 사용되지만, 메소드처럼 동작.
  - 배정문의 왼쪽에서 사용되면 셋-접근자 호출.
  - 배정문의 오른쪽에서 사용되면 겟-접근자 호출.
  - Value 지정어

연산자 중복

- 연산자 중복의 의미
  - 시스템에서 제공한 연산자를 재정의 하는 것.
  - 클래스만을 위한 연산자로써 자료 추상화가 가능.
  - 시스템에서 제공한 연산자처럼 사용 가능
  - 문법적인 규칙은 변경 불가(연산 순위나 결합 법칙 등).

- 연산자 중복 정의 형태

```
public static [extern] returnType operator op (parameter1 [, parameter2]) {
    // ... operator overloading body ...
}
```