C# 8강. 클래스 class

프로그래밍 강사 : 이태성

목차

- 클래스 형식과 접근 한정자
- 객체 생성과 멤버 접근
- 생성자와 소멸자
- 정적 멤버 선언과 사용
- 클래스 참조
- this

01. 클래스 개요

클래스란?

- 변수와 메서드를 그룹화한 것
- 주요 개념
  - 그룹화를 위해 class 키워드와 형식 사용
  - new 연산자로 생성하여 사용
  - new를 사용하지 않으면 같은 클래스 참조할 수 있는 변수가 됨 (클래스 참조 변수)
  - 클래스를 new를 통해 생성하면 객체가 됨

클래스란?

- 인스턴스와 객체의 관계
  인스턴스는 본질
  객체는 실물인 형상

```
A test1 = new A();
A test2 = new A();
```

object　　instance

"같은 A 인스턴스를 가지는 test1과 test2 객체"

02. 클래스 형식과 접근 한정자

클래스 형식

- 클래스 형식과 구조체 형식은 유사

```
struct    struct_name          class class_name
{                                    {


}                                    }
```

xxx.멤버 변수, xxx.멤버 메서드

- 구조체 선언은 값형
  클래스 선언은 참조형

클래스 형식

- 형식

```
class   class_name
{
        variable 선언;
        method   선언;
}
```

member

클래스 형식

```
class Date
{
        int Year, Month, Day;
        void Print()
        {
                Console.WriteLine("{0} {1} {2}",
                                                Year,
Month, Day);
        }
}
```

접근 한정자(access modifier)

- 클래스 멤버에 대한 접근 권한 설정
- 접근 한정자
  private, protected, public, internal,
  protected internal
- 클래스 자체에 적용하는 접근 한정자
  public ➔ 명시적
  internal ➔ 암시적

접근 한정자(access modifier)

- 클래스 멤버에 대한 접근 권한 설정
- 접근 한정자
  private, protected, public, internal,
  protected internal
- 클래스 자체에 적용하는 접근 한정자
  public ➔ 명시적
  internal ➔ 암시적

접근 한정자(access modifier)

- 형식

```
[접근 한정자]  class  class_name
{
        [접근 한정자]  variable 선언;
        [접근 한정자]  method 선언;
 }
```

클래스 안에서 접근 한정자의 역할

- public
  - 클래스 내외부에서 모두 접근 가능
  - 상속관계에서도 위와 동일하게 접근 가능
- private
  - 클래스 또는 구조체 안에서만 멤버에 접근
  - 클래스 외부에서 사용 안됨

클래스 안에서 접근 한정자의 역할

- protected
  - 클래스 안에서는 private와 같은 역할
  - 하위 클래스 안에서는 상위 protected 멤버에 접근 가능
    (private와 동일한 역할)

소스 보기

어셈블리 안에서 접근 한정자의 역할

- internal
  - 같은 어셈블리(exe, dll) 안에 있을 때 만
    내외부에서 멤버에 접근 가능
  - 클래스 서두에 접근 한정자 생략은
    internal 생략한 것

```
class Test
{


 }
```

어셈블리 안에서 접근 한정자의 역할

- protected internal
  - 같은 어셈블리(exe, dll) 안에서는 internal과 동일
  - 서로 다른 어셈블리 안에서 상속관계일 때 하위 클래스는 상위 클래스의 protected internal 멤버에 접근할 수 있음
    외부에서만 접근할 수 없음

03. 객체 생성과 멤버 접근

객체 생성

- 객체 생성 키워드
  new
- 객체 해제는 가비지컬럭터에게

소스 보기

멤버 접근 및 사용

- 접근 한정자에 따라 접근 제한
- 객체의 멤버는 '.'과 함께 표기
  object_name.variable
  object_name.method

  소스 보기
- visual studio 의 인텔리센스 기능 활용

04. 생성자와 소멸자

개념

- 생성자 constructor
  객체를 생성할 때 자동으로 호출되는 메서드
- 소멸자 destructor
  객체가 소멸될 때 호출되는 메서드
- 호출되는 시기?
  생성자 ➔ new
  소멸자 ➔ 가비지컬렉터

생성자의 형식

- 기본 생성자
  - 클래스명과 같은 이름
  - 리턴형 없음
  - 접근 한정자는 public

소스 보기

생성자의 형식

- 생성자
  - 매개변수가 있음
  - 리턴형은 없음
  - 접근 한정자는 public

소스 보기

소멸자

- 소멸자 형식
  - 클래스명과 같음
  - 리턴형 없음
  - 접근 한정자 없음
  - ~ 시작함

```
class MyClass
{
        ~MyClass()
        {
        }
}
```

소멸자의 특징

- 소멸자는 클래스에만 있음
- 소멸자는 오직 한 개임
- 소멸자는 상속 또는 오버로딩 될 수 없음
- 소멸자는 임의로 호출 불가
- 소멸자는 접근 한정자나 매개변수 없음

소스 보기

05. 정적 멤버 선언과 사용

정적 멤버 선언과 사용

- 언어에 따른 static 역할 비교
  C : 변수 값 유지
  C++ : 객체 안에서 변수 공유
  C# : 객체를 생성하지 않고 멤버 사용

  ex) Console.WriteLine()

  소스 보기

정적 멤버 선언과 사용

- Main() 안에서 같은 멤버를 호출할 때

소스 보기

06. 클래스 참조

클래스 참조

- 클래스 형태
  - 선언한 클래스 ➔ 참조형 클래스 변수
    MyClass MyRefTest;

  - new 생성한 클래스 ➔ 객체
    MyClass MyTest = new MyClass();
    MyRefTest = MyTest; // 참조

소스 보기

07. this

this

- this는 객체 자신을 참조하는 키워드
- 사용 형식
  this.member
- 메소드의 매개변수와 멤버 변수명이 같을 때
  this.member_variable = 매개변수;
  ex) this.number = number;

소스 보기