〈 클래스 생성자, 접근한정자, 상속 정리〉

1. 클래스란? (참조 형식)
   - 객체를 만들기 위한 설계도
   - New 연산자에 의해 실제 기억공간 : 힙(heap) 동적 영역

   *객체
   -〉 클래스가 자료형으로 선언된 식별자(참조변수)
   -〉 객체 선언 및 생성
      - 클래스명 객체명 = new 클래스명();

   *인스턴스 -〉 New 연산자에 의해 실제 객체가 생성된 것

2. 클래스 선언 형식
   *클래스 선언 시 고려사항
   1) 사용목적(용도)
   2) 특성(성격) - 멤버변수(필드)
   3) 기능(일) - 멤버함수(메서드)
   4) 단계
      -〉 클래스 선언(힙 영역에 객체명을 잡고)
      -〉 클래스 생성(스택영역에 기억공간이 만들어짐)
         이때 생성되는 공간을 인스턴스라고 한다.
         클래스를 몇 개 선언하더라도, 기억공간만 여러 개 생성되고 클래스 내의 함수는 한번 생성됨.

   - 클래스는 다음과 같이 class 키워드를 이용해서 선언한다.

Colored By Color Scripter™
```
1 class classname
2 {
3    //data and method
4 }
```

예시)

Colored By Color Scripter™
```
1  class Car
2  {
3     //data
4     public string name;
5     public string color;
6
7     //method
8     public void enginestart()
9     {
10       Console.WriteLine("{0} : enginestart complete",name);
11    }
```

```
}
```

- name, color와 같이 '클래스 Car'에 선언된 변수들을 필드(Field)라고 한다.
- 필드, 메소드, 프로퍼티, 이벤트 등의 클래스 내에 선언되어 있는 요소들을 멤버(Member)라고 한다.

Car의 인스턴스

Colored By Color Scripter™

```
Car sonata = new Car();
sonata.color = "white";
sonata.name = "sonata";
sonata.enginestart();
Console.WriteLine("{0} : {1}, sonata.name, sonata.color);
```

```
Car sonata = new Car();
```

- 위 문장의 가장 끝에 있는 Car()는 생성자(Constructor)로 특별한 메소드이다.
- 생성자는 클래스의 이름과 동일한 이름을 가지며, 객체를 생성하는 역할을 한다.
- 다음의 선언문에서 sonata 자체에 메모리가 할당되는 것이 아니며 sonata는 참조로써 객체가 있는 곳을 가리킨다.

new 연산자와 생성자를 이용해서 힙에 객체를 생성하고, sonata는 생성자가 힙에 생성한 객체를 가리킨다.

Colored By Color Scripter™

```
using System;

namespace ClassTest
{
    class Car
    {
        //data
        public string name;
        public string color;

        //method
        public void enginestart()
        {
            Console.WriteLine("{0} : enginestart complete",name);
        }
    }

    class MainApp
    {
        static void Main(string[] args)
        {
            Car sonata = new Car();
```

```
            sonata.color = "white";
            sonata.name = "sonata";
            sonata.enginestart();
            Console.WriteLine("{0} : {1}, sonata.name, sonata.color);
        }
    }
}
```

3. 생성자

Colored By Color Scripter™

```
class classname
{
    한정자 클래스이름(매개변수)
    {
        //....
    }

    //필드
    //메소드
}
```

- 생성자는 클래스와 이름이 같고, 반환 형식이 없다(void가 없다).
- 인위적인 호출이 안됨. - 객체 생성시 자동으로 호출됨
- 생성자의 유일한 임무는 해당 형식(클래스)의 객체를 생성하는 것.
- 객체의 필드를 원하는 값으로 초기화하고 싶을 때, 이 작업을 할 수 있는 최적의 장소가 바로 객체를 생성하기 위해 호출하는 메소드인 생성자이다.
- 중복 선언 가능
- 상속되지 않는다.

예시)

Colored By Color Scripter™

```
class Car
{
    public Car()
    {
        name = "";
        color = "";
    }

    public Car(string _name, string _color)
    {
        name = _name;
        color = _color;
    }
```

```
14
15      public string name;
16      public string color;
17
18      //....
19  }
```

- 매개 변수가 있는 Car() 생성자는 생성자의 ()안에 필요한 매개 변수를 입력하면 된다.

4. 소멸자 (이해 안되서 넘어감)
   - 소멸자의 사용법은 CLR의 가비지 컬렉터(?)가 객체가 소멸되는 시점을 판단해서 소멸자를 호출해준다. (??????)
   - 클래스명과 함수명이 같음
   - 접근지정자, 리턴자료형 없음 클래스 명() {}
   - 객체가 소멸 될 때 자동으로 호출됨.
   - 상속 안됨.

5. 접근 한정자
   - 객체의 멤버에 .(점)을 찍어 접근 할 수 있는 지 없는지를 제어하는 것.

   (클래스의 사용자에게 필요한 최소의 기능만을 노출하고 내부를 감추는 것을 권유, 필드는 상수를 제외하고는 무조건 감추는 것이 좋다.)

   (왜인지는 모르겠음)
   - 접근한정자(Access modifier)는 감추고 싶은 것을 감추고 보여주고 싶은 것을 보여 줄 수 있도록 코드를 수식하며, 필드와 메소드를 비롯해 프로퍼티 등 모든 요소에 대해 사용할 수 있다.
   - C#에서 제공하는 접근 한정자는 모두 5가지.

접근 한정자	설 명
public	클래스의 내부 외부 모든 곳에서 접근할 수 있다.
protected	클래스의 외부에서는 접근할 수 없지만, 파생 클래스에서는 접근이 가능하다
private	클래스의 내부에서만 접근할 수 있다. 파생클래스에서 접근 불가
internal	같은 어셈블리에 있는 코드에 대해서만 public으로 접근할 수 있다. 다른 어셈블리에 있는 코드에서는 private와 같은 수준의 접근성을 가진다.
protected internal	같은 어셈블리에 있는 코드에 대해서만 public으로 접근할 수 있다. 다른 어셈블리에 있는 코드에서는 protected와 같은 수준의 접근성을 가진다.

6. 클래스의 상속
   - 클래스를 만들 때 완전히 새로운 데이터 멤버 및 함수를 작성하는 대신 새 클래스가 기존 클래스의 멤버를 상속해야 할 상황에서 클래스 상속.
   - 기존 클래스를 기본(상위)클래스, 상속받은 클래스를 파생(하위)클래스라고 함.
   - 클래스의 상속은 : (콜론)을 이용하여 이루어짐.

예시)

```
using System;
```

```
namespace InheritanceApplication {
   class Shpae {
      protected int width;
      protected int height;

      public void setWidth(int w) {
         width = w;
      }
      public void setHeight(int h) {
         height = h;
      }
   }

   class Rectangle : Shape {
      public int getArea() {
         return (width * height);
      }
   }

   class Inheritance {
      static void Main(string[] args) {
         Rectangle rect = new Rectangle();

         rect.setWidth(3);
         rect.setHeight(5);

         Console.WriteLine("AREA : {0}", rect.getArea());
      }
   }
}
```

〈찾아본 사이트〉

https://yeolco.tistory.com/143

https://coolpiz.tistory.com/entry/C%ED%81%B4%EB%9E%98%EC%8A%A4%EC%9D%98-%EC%9D%B4%ED%95%B4-32

https://coolpiz.tistory.com/entry/C%ED%81%B4%EB%9E%98%EC%8A%A4%EC%9D%98-%EC%9D%B4%ED%95%B431

이해정도(?)
- 줄글로 된 설명만 들을 때는 이해하기는 어려운 것 같고 위 사이트에서 예시 하나하나 들면서 같이 설명해주니까 좀 이해가 갑니다.

- 전에 3일동안 배울 때 중요하다고 강조되면서 들은 부분이라 아예 낯설지는 않다는 게 그나마 다행인 것 같네요

- 예시랑 같이 설명 듣고 계속 복습하다 보면 이해할 수 있을 거라고 생각합니다.

- 저 소멸자는 같이 안 배워서 사실 잘 이해가 안감.

- 접근 한정자도 5가지를 다 배우지 않아서 각각의 기능이 뭔지는 나와있지만 어떻게 사용해야 하는 건지 잘 감이 안옴. (public, private 빼고는 잘 모르겠음.)