using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace UsageOfHidding

{

// 코드 7-28: 하이딩

class Program

{

class Animal

{

public void Eat()

{

Console.WriteLine("냠냠 먹습니다.");

}

}

class Dog : Animal

{

public void Eat()

{

Console.WriteLine("강아지 사료를 먹습니다.");

}

}

class Cat : Animal

{

public void Eat()

{

Console.WriteLine("고양이 사료를 먹습니다.");

}

}

static void Main(string[] args)

{

List<Animal> Animals = new List<Animal>()

{

new Dog(), new Cat(), new Cat(), new Dog(),

new Dog(), new Cat(), new Dog(), new Dog()

};

foreach (var item in Animals)

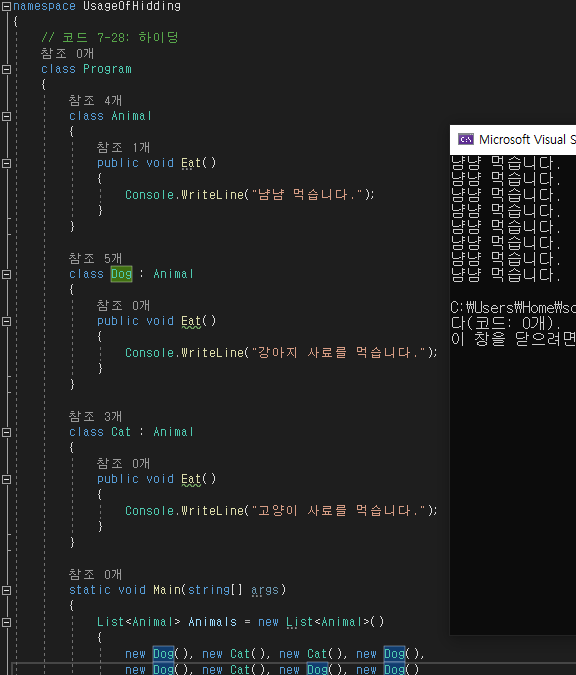
{

item.Eat();

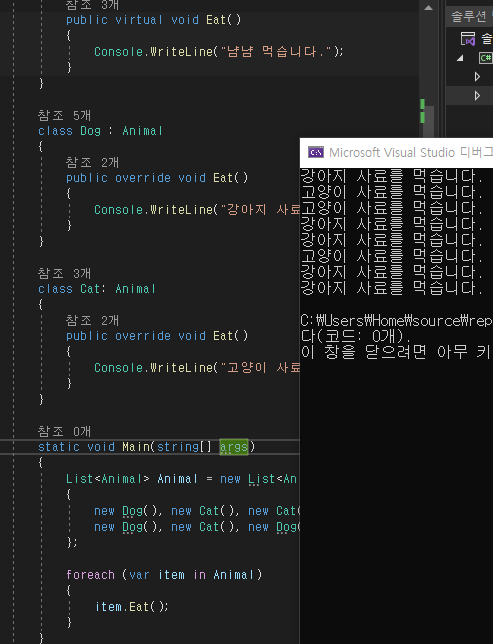
}

}

}



이유 : 오버라이딩 하지않았기 때문에 부모의 Eat이 실행된다



**부모클래스에 virtual**

**자식클래스에 override해서 오버라이딩 해주거나**

**New 로 부모클래스를 하이딩 해주면 원하는 값이 나온다**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BInheritance

{

// 코드 7-4: Animal 클래스

class Animal

{

public int Age { get; set; }

public Animal() { this.Age = 0; }

public void Eat() { Console.WriteLine("냠냠 먹습니다."); }

public void Sleep() { Console.WriteLine("쿨쿨 잠을 잡니다."); }

}

// 코드 7-5: Animal 클래스의 상속을 받는 Dog 클래스와 Cat 클래스

class Dog : Animal

{

public string Color { get; set; }

public void Bark() { Console.WriteLine("왈왈 짓습니다."); }

// 코드 7-6: 부모에게서 상속 받은 메서드 호출

public void Test()

{

Eat();

Sleep();

}

}

class Cat : Animal

{

public void Meow() { Console.WriteLine("냥냥 웁니다."); }

}

class Program

{

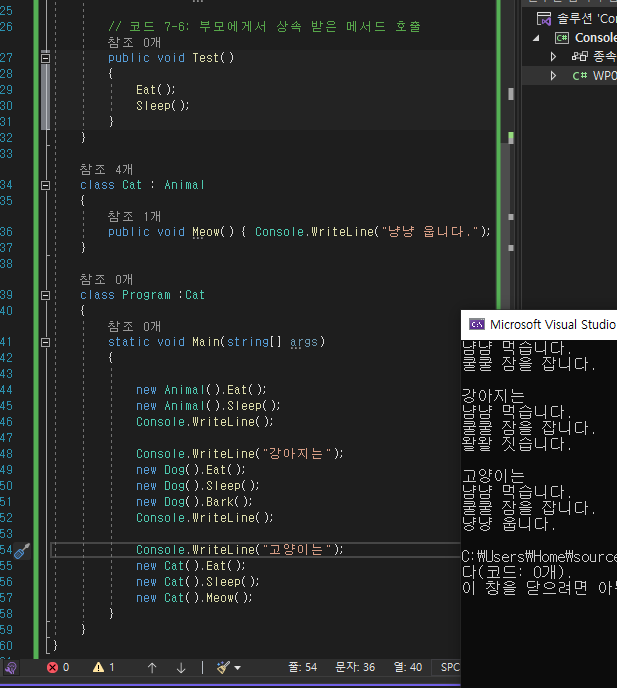
static void Main(string[] args)

{

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Sealed

{

// 코드 7-31: sealed 클래스 오류

namespace A

{

sealed class Parent

{

public void Test() { }

}

class Child : Parent

{

public void Test() { }

}

}

// 코드 7-32: sealed 메서드 오류

namespace B

{

class Parent

{

public virtual void Test() { }

}

class Child : Parent

{

sealed public override void Test() { }

}

class GrandChild : Child

{

public override void Test() { }

}

}

class Program

{

static void Main(string[] args)

{

// 코드 7-31 호출

A.Parent parent = new A.Parent();

A.Child child = new A.Child();

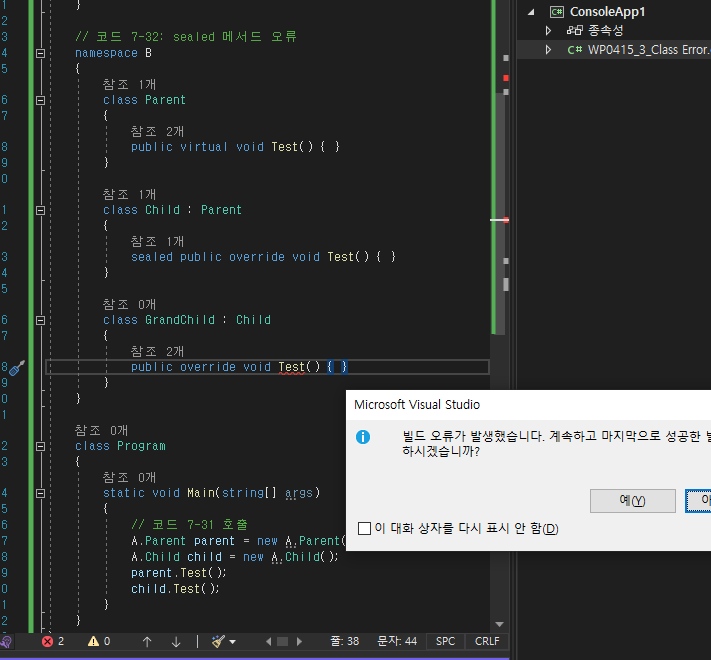
parent.Test();

child.Test();

}

}

}



**이유 : child의 Test()가 sealed 되어있어 오버라이딩이 불가능 하기 때문에 오류가 난다**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassVariableOnInheritance

{

class Program

{

class Parent

{

public static int counter = 0;

public void CountParent()

{

Parent.counter++;

}

}

class Child : Parent

{

public void CountChild()

{

Child.counter++;

}

}

static void Main(string[] args)

{

Parent parent = new Parent();

Child child = new Child();

parent.CountParent();

child.CountChild();

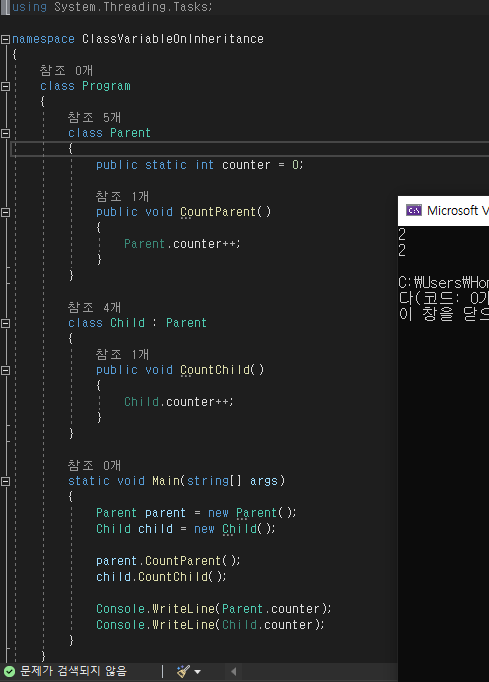
Console.WriteLine(Parent.counter);

Console.WriteLine(Child.counter);

}

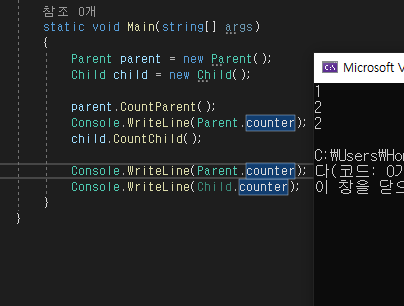
}

}



**이유 : counter가 static으로 선언되었기 때문에 2 / 2 로 출력된다**

**Countchild() 전에 Parent.counter를 출럭하면 1이 나온2다**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GenericBasic

{

class Wanted<T>

{

public T Value;

public Wanted(T value)

{

this.Value = value;

}

}

class Program

{

static void Main(string[] args)

{

Wanted<string> wantedString = new Wanted<string>("String");

Wanted<int> wantedInt = new Wanted<int>(52273);

Wanted<double> wantedDouble = new Wanted<double>(52.273);

Console.WriteLine(wantedString.Value);

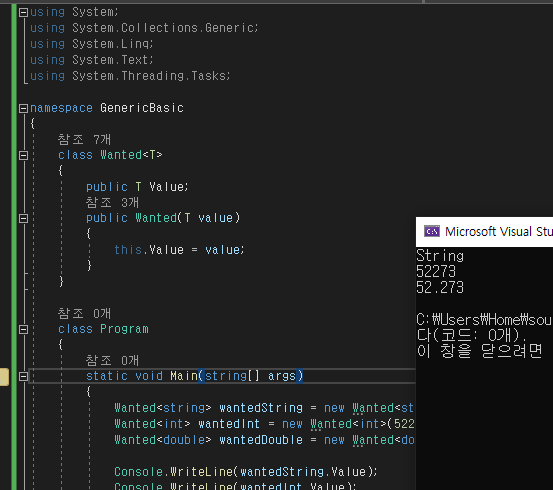
Console.WriteLine(wantedInt.Value);

Console.WriteLine(wantedDouble.Value);

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Generics

{

// 코드 8-2: 두 개 이상의 제네릭을 사용하는 클래스

class TestA<T, U>

{

}

// 코드 8-3: where 키워드

class TestB<T, U>

where T : class

where U : struct

{

}

class TestC<T, U>

where T : IComparable

where U : IComparable, IDisposable

{

}

class Program

{

static void Main(string[] args)

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace InterfaceBasic

{

interface IBasic

{

int TestInstanceMethod();

int TestProperty { get; set; }

}

class Program

{

class TestClass : IBasic

{

public int TestProperty

{

get

{

throw new NotImplementedException();

}

set

{

throw new NotImplementedException();

}

}

public int TestInstanceMethod()

{

throw new NotImplementedException();

}

}

static void Main(string[] args)

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConstructorOfStruct

{

class Program

{

struct Point

{

public int x;

public int y;

public Point(int x, int y)

{

this.x = x;

this.y = y;

}

}

static void Main(string[] args)

{

Point point = new Point();

Console.WriteLine(point.x);

Console.WriteLine(point.y);

}

}

}

