

상속과 다형성(2)

HyoJoon Han
동국대학교
han6343@dongguk.edu



is 키워드

클래스 자료형 변환

상속의 생성자

새도잉과 하이딩

하이딩과 오버라이딩

상속과 오버라이딩 제한

- 특정 객체의 클래스 확인
- is 키워드 형태

변수 is 클래스

코드 7-14 is 키워드

/7장/Inheritance

```
static void Main(string[] args)
{
    List<Animal> Animals = new List<Animal>() { /* 생략 */ }

    foreach (var item in Animals)
    {
        item.Eat();
        item.Sleep();

        if (item is Dog) { } ————— 만약 변수 item이 Dog 객체라면
        if (item is Cat) { } ————— 만약 변수 item이 Cat 객체라면
    }
}
```

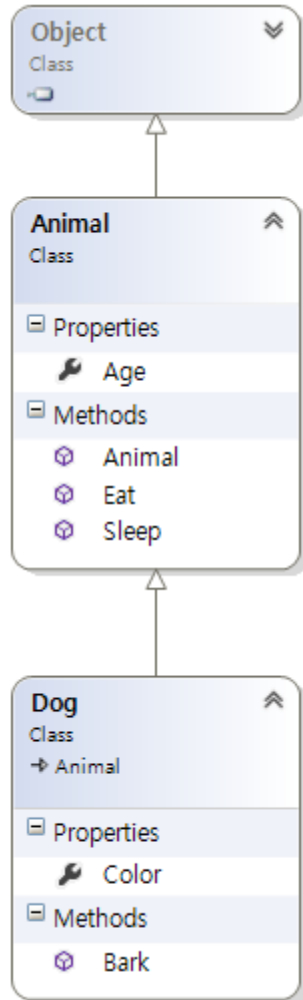


그림 7-11 Dog 클래스의 상속 관계

```
item is Dog
item is Animal
item is Object
```

- 일반적인 자료형 변환
 - 형태

(클래스) 변수

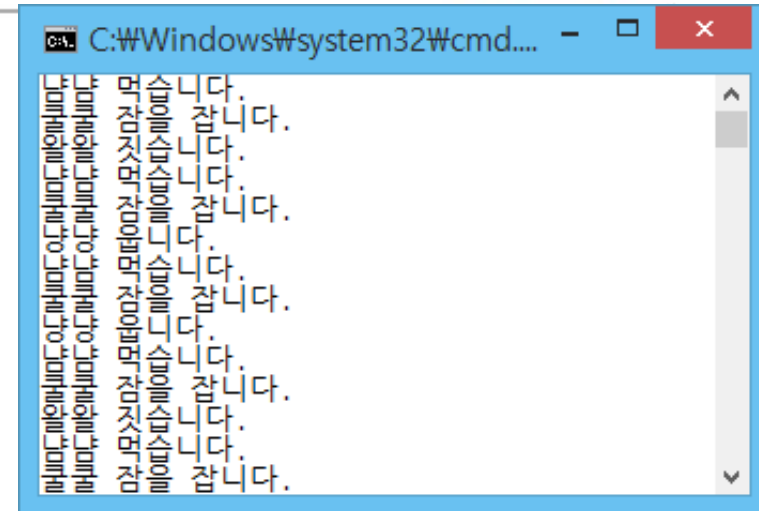
예

코드 7-15 일반적인 자료형 변환

```
foreach (var item in Animals)
{
    item.Eat();
    item.Sleep();

    if (item is Dog) { ((Dog)item).Bark(); }
    if (item is Cat) { ((Cat)item).Meow(); }
}
```

/7장/DInheritance



- as 키워드로 자료형 변환
 - 형태

변수 as 클래스

- 예

코드 7-17 as 키워드를 사용하는 경우의 일반적인 형태

/7장/DInheritance

```
foreach (var item in Animals)
{
    item.Eat();
    item.Sleep();

    var dog = item as Dog;
    if (dog != null) { dog.Bark(); }

    var cat = item as Cat;
    if (cat != null) { cat.Meow(); }
}
```

- 생성자 : 인스턴스 초기화 할 때 사용
- 자식 인스턴스 생성하면, 부모의 멤버 초기화 위해 부모 생성자도 자동으로 호출

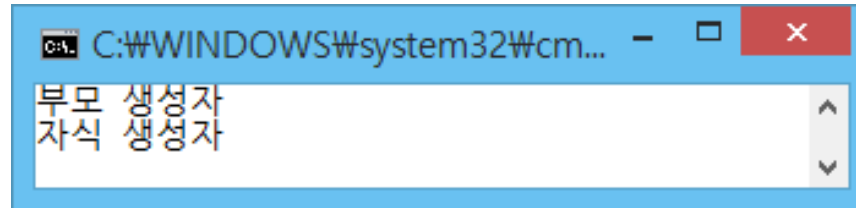
코드 7-18 상속했을 때 기본적인 생성자 호출 순서

/7장/ConstructorSequences

```
class Program
{
    class Parent
    {
        public Parent()
        {
            Console.WriteLine("부모 생성자");
        }
    }

    class Child : Parent
    {
        public Child()
        {
            Console.WriteLine("자식 생성자");
        }
    }

    static void Main(string[] args)
    {
        Child child = new Child();
    }
}
```



- 부모 생성자 호출을 명시적으로 지정할 때

코드 7-19 base 키워드를 사용한 생성자 지정(1)

/7장/ConstructorSequences

```
class Program
{
    class Parent
    {
        public Parent() { Console.WriteLine("부모 생성자"); }
    }

    class Child : Parent
    {
        public Child() : base()———— base 키워드를 사용합니다.
        {
            Console.WriteLine("자식 생성자");
        }
    }

    static void Main(string[] args)
    {
        Child child = new Child();
    }
}
```


- 매개변수가 있는 메서드를 호출하고 싶을 때

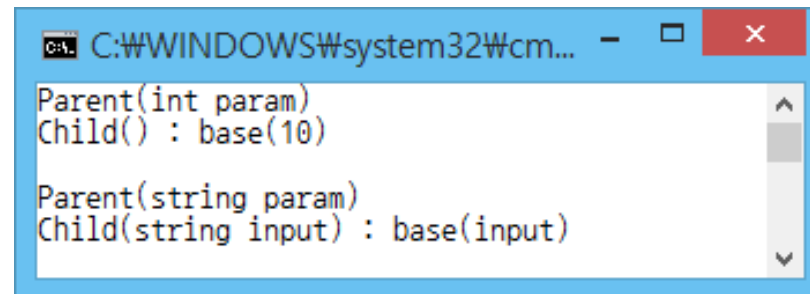
코드 7-20 base 키워드를 사용한 생성자 지정(2)

/7장/ConstructorSequences

```
class Program
{
    class Parent
    {
        public Parent() { Console.WriteLine("Parent()"); }
        public Parent(int param) { Console.WriteLine("Parent(int param)"); }
        public Parent(string param) { Console.WriteLine("Parent(string param)"); }
    }

    class Child : Parent
    {
        public Child() : base(10) { Console.WriteLine("Child() : base(10)"); }
        public Child(string input) : base(input) { Console.WriteLine("Child(string input) : base(input)"); }
    }

    static void Main(string[] args)
    {
        Child childA = new Child();
        Child childB = new Child("string");
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
Parent(int param)
Child() : base(10)

Parent(string param)
Child(string input) : base(input)
```

- 클래스 변수 상속

코드 7-21 클래스 변수 상속 /7장/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();

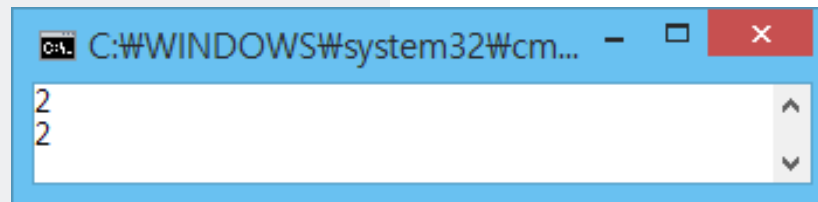
        parent.CountParent();
        child.CountChild();

        Console.WriteLine(Parent.counter);
        Console.WriteLine(Child.counter);
    }
}
```

클래스 변수 counter를 선언합니다.

Parent 클래스의 counter 변수를 증가시킵니다.

Child 클래스의 counter 변수를 증가시킵니다.



- 새도잉 : 특정한 영역에서 이름이 겹쳐 다른 변수 가리키는 것

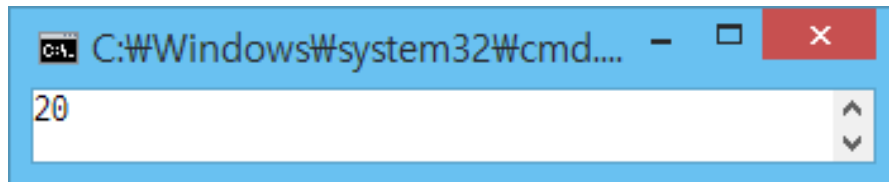
코드 7-22 새도잉

/7장/ShadowAndHide

```
class Program
{
    public static int number = 10;

    static void Main(string[] args)
    {
        int number = 20;
        Console.WriteLine(number);
    }
}
```

static 메서드 내부에서 사용할 수 있게 static 변수로 만들었습니다.



- 하이딩 : 부모 클래스와 자식 클래스에 동일 이름으로 멤버 만들 때 발생

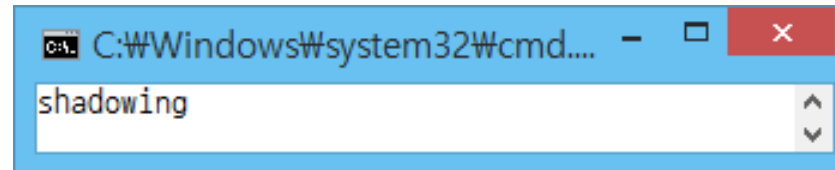
코드 7-23 변수 하이딩

/7장/ShadowAndHide

```
class Program
{
    class Parent
    {
        public int variable = 273;
    }

    class Child : Parent
    {
        public string variable = "shadowing";
    }

    static void Main(string[] args)
    {
        Child child = new Child();
        Console.WriteLine(child.variable);
    }
}
```



- 부모에 있는 int 자료형의 변수 사용할 때
 - 부모로 자료형을 변환하고 사용

코드 7-24 숨겨진 멤버를 찾는 방법

/7장/ShadowAndHide

```
static void Main(string[] args)
{
    Child child = new Child();
    Console.WriteLine(((Parent) child).variable);
}
```

- 메서드 하이딩 예

코드 7-25 메서드 하이딩

/7장/ShadowAndHide

```
class Program
{
    class Parent
    {
        public void Method()
        {
            Console.WriteLine("부모의 메서드");
        }
    }

    class Child : Parent
    {
        public void Method()
        {
            Console.WriteLine("자식의 메서드");
        }
    }

    static void Main(string[] args)
    {
        Child child = new Child();
        child.Method();
        ((Parent)child).Method();
    }
}
```

- 실행은 정상적이나 개발 환경에 경고 메시지 뜸

```
public class Child : Parent
{
    public void Method()
    {
        Console.
    }
}
```

'Child.Method()'은(는) 상속된 'Parent.Method()' 멤버를 숨깁니다. 숨기려면 new 키워드를 사용하십시오.

- 메서드는 변수와 다르게 충돌이 발생할 때 하이딩할지 오버라이딩할지 결정 가능

```
public new void Method()
{
    Console.WriteLine("자식의 메서드");
}
```

하이딩합니다.

```
public override void Method()
{
    Console.WriteLine("자식의 메서드");
}
```

오버라이딩합니다.

- 오버라이딩(overriding) : 부모 클래스의 메서드를 자식 클래스에서 재구현
 - 하이딩의 형태로 메서드 작성 후 앞에 virtual 이라는 키워드 붙임
 - 하이딩은 멤버 전체(변수, 메서드 등)에서 발생
 - 오버라이딩은 메서드 관련만 발생
- new 메서드
 - 하이딩 한다는 표시를 위해 메서드 이름 앞에 new 키워드 붙임

코드 7-26 new 메서드를 사용한 하이딩

/7장/NewMethods

```
class Program
{
    class Parent
    {
        public int variable = 273;
        public void Method()
        {
            Console.WriteLine("부모의 메서드");
        }
    }
}
```

```
class Child : Parent
{
    public new string variable = "hiding";
    public new void Method()
    {
        Console.WriteLine("자식의 메서드");
    }
}
```

new 키워드를 사용해 변수를 하이딩하겠다고 선언합니다.

new 키워드를 사용해 메서드를 하이딩하겠다고 선언합니다.

```
static void Main(string[] args)
{
    Child child = new Child();
    child.Method();
    ((Parent)child).Method();
}
```

코드 7-27 virtual과 override 메서드를 사용한 오버라이딩

/7장/OverrideMethods

```
class Program
{
    class Parent
    {
        public virtual void Method()
        {
            Console.WriteLine("부모의 메서드");
        }
    }
}
```

부모의 메서드에 virtual 키워드를 적용합니다.

```
class Child : Parent
{
    public override void Method()
    {
        Console.WriteLine("자식의 메서드");
    }
}

static void Main(string[] args)
{
    Child child = new Child();
    child.Method();
    ((Parent)child).Method();
}
}
```

자식의 메서드에 override 키워드를 적용합니다.

- virtual과 override 메서드
 - 예

- 활용 예(하이딩)

코드 7-28 하이딩

/7장/UsageOfHidding

```
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()
        {
```

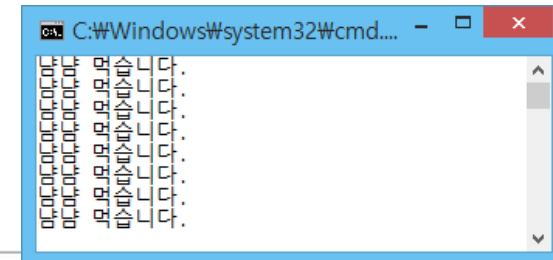
같은 이름을 재사용했습니다.

```
        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();
    }
}
```

Eat 메서드를 호출합니다.



- 활용 예(오버라이딩)

코드 7-29 오버라이딩

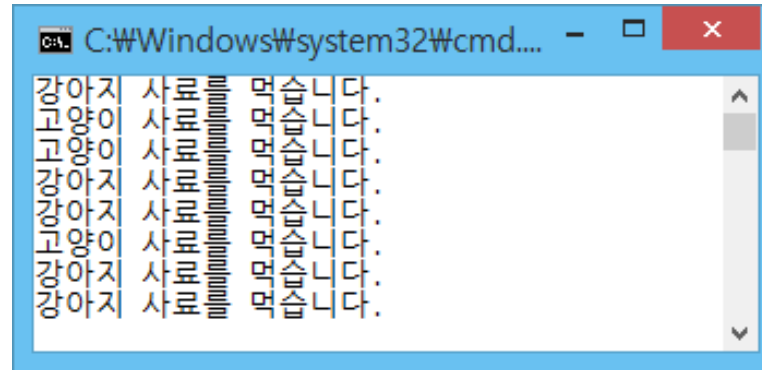
/7장/UsageOfOverriding

```
class Animal
{
    public virtual void Eat()
    {
        Console.WriteLine("냠냠 먹습니다.");
    }
}

class Dog : Animal
{
    public override void Eat()
    {
        Console.WriteLine("강아지 사료를 먹습니다.");
    }
}

class Cat : Animal
{
    public override void Eat()
    {
        Console.WriteLine("고양이 사료를 먹습니다.");
    }
}
```

오버라이딩합니다.



```
C:\Windows\system32\cmd...
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
고양이 사료를 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
```

- 활용 예(new 키워드를 사용하는 하이딩)

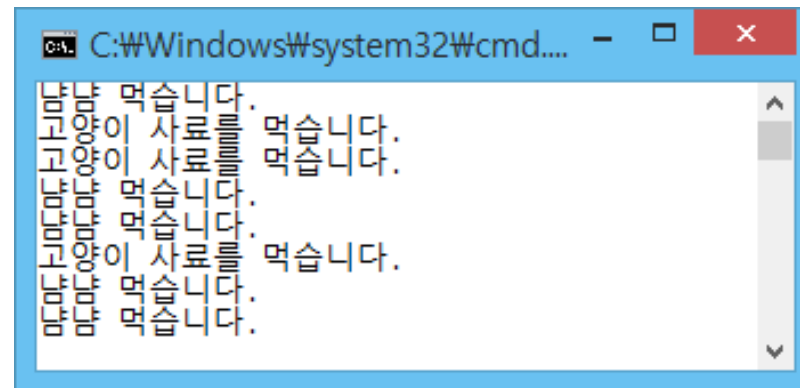
코드 7-30 new 키워드를 사용하는 경우

```
class Animal
{
    public virtual void Eat()
    {
        Console.WriteLine("냠냠 먹습니다.");
    }
}

class Dog : Animal
{
    public new void Eat()
    {
        Console.WriteLine("강아지 사료를 먹습니다.");
    }
}

class Cat : Animal
{
    public override void Eat()
    {
        Console.WriteLine("고양이 사료를 먹습니다.");
    }
}
```

하이딩으로 변경합니다.



```
C:\Windows\system32\cmd.exe
냠냠 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
고양이 사료를 먹습니다.
```

- sealed 메서드 : 클래스 적용(상속 제한), 메서드 적용(오버라이딩 제한)
 - 상속 제한 오류 예

코드 7-31 sealed 클래스 오류

/7장/Sealed

```
class Program
{
    sealed class Parent
    {
        public void Test() { }
    }

    class Child : Parent
    {
        public void Test() { }
    }

    static void Main(string[] args)
    {
        Parent parent = new Parent();
        Child child = new Child();

        parent.Test();
        child.Test();
    }
}
```

sealed 클래스로 선언했습니다.

여기서 오류가 발생합니다.

- 메서드 오버라이딩 제한 오류 예

코드 7-32 sealed 메서드 오류

```
class Parent
{
    public virtual void Test() { }
}

class Child : Parent
{
    sealed public override void Test() { }
}

class GrandChild : Child
{
    public override void Test() { }
```

sealed 메서드로 변경했습니다.

여기서 오류가 발생합니다.

```
class GrandChild : Child
{
    public override void Test() { }
}
```

'GrandChild.Test()': 상속된 'Child.Test()' 멤버가 sealed이므로 재정의할 수 없습니다.

- abstract 키워드 : 무조건 상속, 또는 메서드 반드시 오버라이딩
 - 상속 제한 오류 예

코드 7-33 abstract 클래스 오류

/7장/Abstract

```
class Program
{
    abstract class Parent
    {
        public void Test() { }
```

abstract 클래스로 선언했습니다.

```
    class Child : Parent
    {
        public void Test() { }
```

```
Parent parent = new Parent();
```

class Parent

오류:

'Parent' 추상 클래스 또는 인터페이스의 인스턴스를 만들 수 없습니다.

```
static void Main(string[] args)
{
    Parent parent = new Parent();
    Child child = new Child();
```

여기서 오류가 발생합니다.

```
    parent.Test();
    child.Test();
}
```


- 메서드 오버라이딩 제한 오류 예

코드 7-34 abstract 메서드

/7장/Abstract

```
abstract class Parent
{
    public abstract void Test();
}
```

abstract 메서드를 선언하려면 반드시 abstract 클래스가 되어야 합니다.

abstract 메서드로 선언했습니다.

```
class Child : Parent
{
}
```

여기에서 오류가 발생합니다.

```
class Child : Parent
{
}

'Child'은(는) 상속된 추상 멤버 'Parent.Test()'를(를) 구현하지 않습니다.
```

그림 7-26 abstract 메서드 오류

- abstract 메서드와 관련된 오류 해결

코드 7-35 abstract 메서드와 관련된 오류 해결

/7장/Abstract

```
abstract class Parent
{
    public abstract void Test();
}

class Child : Parent
{
    public override void Test() { }
```

override 키워드를 사용해 오버라이딩해야 합니다.

- 이전 실습 이어서
- 코드 변경 (Form.cs)

```
private void button2_Click(object sender, EventArgs e)
{
    textBox5.Text = "";
    foreach (Friend friend in myFriend)
    {
        if (friend is HighFriend)
        {
            textBox5.Text += ((HighFriend)friend).showBasicInfo() + Environment.NewLine;
        }
        else if (friend is UnivFriend)
        {
            textBox5.Text += ((UnivFriend)friend).showBasicInfo() + Environment.NewLine;
        }
    }
}
```

```
private void button3_Click(object sender, EventArgs e)
{
    textBox5.Text = "";
    foreach (Friend friend in myFriend)
    {
        if (friend is HighFriend)
        {
            textBox5.Text += ((HighFriend)friend).showData() + Environment.NewLine;
        }
        else if (friend is UnivFriend)
        {
            textBox5.Text += ((UnivFriend)friend).showData() + Environment.NewLine;
        }
    }
}
```

- 코드 변경 (Friend.cs)

```
abstract class Friend
{
    protected string name, phoneNum, addr;

    public Friend(string name, string phone, string addr)
    {
        this.name = name;
        this.phoneNum = phone;
        this.addr = addr;
    }

    public abstract string showData();

    public abstract string showBasicInfo();
}
```

- 코드 변경(HighFriend.cs)

```
class HighFriend : Friend
{
    string newLine = Environment.NewLine;
    private string work;

    public HighFriend(string name, string phone,
        string addr, string job)
        : base(name, phone, addr)
    {
        this.work = job;
    }

    public override string showData()
    {
        string str = "이름 : " + base.name + newLine;
        str += "전화 : " + base.phoneNum + newLine;
        str += "주소 : " + base.addr + newLine;
        str += "직업 : " + work + newLine;
        return str;
    }

    public override string showBasicInfo()
    {
        return "이름 : " + base.name + newLine
            + "전화 : " + base.phoneNum + newLine;
    }
}
```

- 코드 변경(UnivFriend.cs)

```
class UnivFriend : Friend
{
    string major;
    string newLine = Environment.NewLine;

    public UnivFriend(string name, string phone,
        string addr, string major)
        : base(name, phone, addr)
    {
        this.major = major;
    }

    public override string showData()
    {
        string str = "이름 : " + base.name + newLine;
        str += "전화 : " + base.phoneNum + newLine;
        str += "주소 : " + base.addr + newLine;
        str += "전공 : " + major + newLine;
        return str;
    }

    public override string showBasicInfo()
    {
        return "이름 : " + base.name + newLine
            + "전화 : " + base.phoneNum + newLine
            + "전공 : " + major + newLine;
    }
}
```

- 결과확인

친구 관리 프로그램

이름

전화번호

주소

직업/전공

☐ 고등학교 친구 ☒ 대학교 친구

친구추가

친구 정보 간단 확인

친구 정보 확인

이름 : 홍길동
전화 : 010-1234-5678
주소 : 서울 A
직업/전공 : 고길동
이메일 : 010-5487-8795
주소 : 서울
전공 : B