윈도우프로그래밍

11. 상속과 다형성(2)

2023. 5. 26

심미나

목 차

참고자료: 수업교재(C# 프로그래밍, 2판)

VI. 상속의 생성자

상속의 생성자

상속의 생성자

- 생성자
  - 인스턴스 초기화할 때 사용
  - 자식 인스턴스를 생성하면, 부모의 멤버 초기화 위해 부모 생성자도 자동으로 호출
- 상속 시 기본적인 생성자 호출 순서
  - 부모생성자 먼저 호출 후
  - 그 다음 자식 생성자 호출

```
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

자식 생성자

자식 인스턴스를 생성합니다.

실행 결과

```
부모 생성자
자식 생성자
```

상속의 생성자

상속의 생성자

- 부모 생성자 호출을 명시적으로 지정할 때
  - base 키워드를 사용한 생성자 지정
    - 생성자 메서드 뒤에 콜론 입력하고 base() 입력

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

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

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

base 키워드를 사용합니다.

상속의 생성자

상속의 생성자

- 매개변수가 있는 메서드를 호출하고 싶을 때
  - base 키워드를 사용한 생성자 지정

```
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");
    }
}
```

Parent(int param) 부모 생성자를 호출합니다.

Parent(string param) 부모 생성자를 호출합니다.

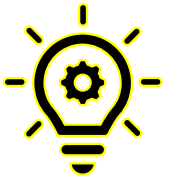
실행 결과

```
Parent(int param)
Child() : base(10)
Parent(string param)
Child(string input) : base(input)
```

윈도우프로

상속의 생성자

(참고) 클래스 변수 상속



- 클래스 변수는 상속되어도 공유
  - 클래스 변수 상속

```
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.

실행 결과

```
2
2
```

윈도우프로

VII. 섀도잉과 하이딩

섀도잉과 하이딩

섀도잉(Shadowing)

- 섀도잉은 특정한 영역에서 이름이 겹쳐 다른 변수를 가리는 것
  - 섀도잉의 예

```
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.

shadowing

실행 결과

```
20
```

섀도잉과 하이딩

하이딩(Hiding)

- 하이딩은 부모 클래스와 자식 클래스에 동일 이름으로 멤버 만들 때 발생
  - 변수 하이딩의 예

```
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);
    }
}
```

hiding

실행 결과

```
shadowing
```

섀도잉과 하이딩

하이딩(Hiding)

- 부모에 있는 int 자료형의 변수 사용할 때
  - 부모로 자료형을 변환하고 사용
    - 숨겨진 멤버를 찾는 방법

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

섀도잉과 하이딩

하이딩(Hiding)

- 메서드 하이딩
  - 메서드 하이딩의 예

```
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();
    }
}
```

hiding

섀도잉과 하이딩

하이딩(Hiding)

- 실행은 정상적이나 개발 환경에 경고 메시지 뜸
  - 경고 메시지

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

void Child.Method()

CS0108: 'Program.Child.Method()'은(는) 상속된 'Program.Parent.Method()' 멤버를 숨깁니다. 숨기려면 new 키워드를 사용하세요.

CA1822: 'Method' 멤버는 인스턴스 데이터에 액세스하지 않으며 static으로 표시할 수 있습니다.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

- 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.

VIII. 하이딩과 오버라이딩

하이딩과 오버라이딩

오버라이딩(Overriding)

- 오버라이딩은 부모 클래스의 메서드를 자식 클래스에서 재구현
  - 하이딩의 형태로 메서드 작성 후 앞에 virtual이라는 키워드 붙임
  - 하이딩은 멤버 전체(변수, 메서드 등)에서 발생
  - 오버라이딩은 메서드 관련만 발생

하이딩과 오버라이딩

new 메서드

- 하이딩한다는 표시를 위해 메서드 이름 앞에 new 키워드 붙임
  - new 메서드를 사용한 하이딩(계속)

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

하이딩과 오버라이딩

new 메서드

- 하이딩한다는 표시를 위해 메서드 이름 앞에 new 키워드 붙임

- new 메서드를 사용한 하이딩

public void Method()

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

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

new 키워드를 사용해 변수를 하이딩하겠다고 선언합니다.

new 키워드를 사용해 메서드를 하이딩하겠다고 선언합니다.

하이딩과 오버라이딩

virtual과 override 메서드

- virtual과 override 메서드를 사용한 오버라이딩(계속)

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

부모의 메서드에 virtual 키워드를 적용합니다.

하이딩과 오버라이딩

virtual과 override 메서드

- virtual과 override 메서드를 사용한 오버라이딩(계속)

```
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.

실행 결과

```
자식의 메서드
자식의 메서드
```

  - 오버라이딩하면 클래스형을 어떻게 변환해도 자식에서 다시 정의한 메서드 호출

하이딩과 오버라이딩

활용

- 하이딩 예(계속)

```
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.");
        }
    }
```

같은 이름을 재사용했습니다.

하이딩과 오버라이딩

활용

- 하이딩 예

```
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.

실행 결과

```
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
```

하이딩과 오버라이딩

활용

- 오버라이딩 예

```
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.

실행 결과

```
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
고양이 사료를 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
고양이 사료를 먹습니다.
강아지 사료를 먹습니다.
강아지 사료를 먹습니다.
```

하이딩과 오버라이딩

활용

- new 키워드를 사용하는 하이딩 예

```
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.

실행 결과

```
냠냠 먹습니다.
고양이 사료를 먹습니다.
고양이 사료를 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
고양이 사료를 먹습니다.
냠냠 먹습니다.
냠냠 먹습니다.
```

IX. 상속과 오버라이딩 제한

상속과 오버라이딩 제한

sealed 메서드

- 클래스 적용(상속 제한), 메서드 적용(오버라이딩 제한)에 사용
  - 상속 제한 오류의 예(sealed 클래스 오류)

```
class Program
{
    sealed class Parent          // sealed 클래스로 선언했습니다.
    {
        public void Test() { }
    }

    class Child : Parent         // 여기서 오류가 발생합니다.
    {
        public void Test() { }
    }

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

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

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

'Child': sealed 형식 'Parent'에서 파생될 수 없습니다.

상속과 오버라이딩 제한

sealed 메서드

- 클래스 적용(상속 제한), 메서드 적용(오버라이딩 제한)에 사용
  - 메서드 오버라이딩 제한 오류의 예(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.

상속과 오버라이딩 제한

abstract 키워드

- 무조건 상속, 또는 메서드 반드시 오버라이딩 할 때 사용
  - 상속 제한 오류의 예(abstract 클래스 오류)

```
class Program
{
    abstract 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();
    }
}
```

abstract 클래스로 선언했습니다.

여기서 오류가 발생합니다.

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

class Parent

오류:

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

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

class ClassBasic.Program.Parent

CS0144: 추상 형식 또는 인터페이스 'Program.Parent'의 인스턴스를 만들 수 없습니다.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

상속과 오버라이딩 제한

abstract 키워드

- 무조건 상속, 또는 메서드 반드시 오버라이딩 할 때 사용
  - 메서드 오버라이딩 제한 오류의 예(abstract 메서드 오류)

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

class Child : Parent
{

}
```

abstract 메서드를 선언하려면 반드시 abstract 클래스가 되어야 합니다.

abstract 메서드로 선언했습니다.

여기에서 오류가 발생합니다.

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

class ClassBasic.Program.Child

CS0534: 'Program.Child'은(는) 상속된 추상 멤버 'Program.Parent.Test()'을(를) 구현하지 않습니다.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

상속과 오버라이딩 제한

abstract 키워드

- abstract 메서드와 관련된 오류 해결

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

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

override 키워드를 사용해 오버라이딩해야 합니다.

X. 실습 및 과제

실습

실습 예제

- 실습코드 소개
  - (교재 p.351-373) 별도로 제공하는 실습코드(실습7-4 ~ 7-8)
    - 교재 7장은 예제가 구성되지 않고 부분 코드들만 다루고 있으므로 코드를 모두 종합하여 실습코드(실습7-4~7-8)를 별도 제공함
    - 다음 페이지의 실습코드를 바탕으로 자율적으로 실습할 것

실습

실습7-4

- (코드 7-18 ~ 7-20) (교재 p.351-353)
  - 상속 시 생성자 호출 순서 & base 키워드를 사용한 생성자

```
01  namespace ConstructorSequences
02  {
03      // 코드 7-18: 상속 때의 기본적인 생성자 호출 순서
04      namespace A
05      {
06          class Parent
07          {
08              public Parent()
09              {
10                  Console.WriteLine("부모 생성자");
11              }
12          }
13
14          class Child : Parent
15          {
16              public Child()
17              {
18                  Console.WriteLine("자식 생성자");
19              }
20          }
21      }
22
```

실습

실습7-4

- (코드 7-18 ~ 7-20) (교재 p.351-353)
  - 상속 시 생성자 호출 순서 & base 키워드를 사용한 생성자

```
23      // 코드 7-19: base 키워드를 사용한 생성자 지정
24      namespace B
25      {
26          class Parent
27          {
28              public Parent() { Console.WriteLine("부모 생성자"); }
29          }
30
31          class Child : Parent
32          {
33              public Child() : base()
34              {
35                  Console.WriteLine("자식 생성자");
36              }
37          }
38      }
39      // 코드 7-20: base 키워드를 사용한 생성자 지정(2)
40      namespace C
41      {
42          class Parent
43          {
44              public Parent() { Console.WriteLine("Parent()"); }
45              public Parent(int param) { Console.WriteLine("Parent(int param)"); }
46              public Parent(string param) { Console.WriteLine("Parent(string param)"); }
47          }
```

실습

실습7-4

- (코드 7-18 ~ 7-20) (교재 p.351-353)
  - 상속 시 생성자 호출 순서 & base 키워드를 사용한 생성자

```
48          class Child : Parent
49          {
50              public Child() : base(10)
51              {
52                  Console.WriteLine("Child() : base(10)");
53              }
54              public Child(string input) : base(input)
55              {
56                  Console.WriteLine("Child(string input) : base(input)");
57              }
58          }
59      }
60      class Program
61      {
62          static void Main(string[] args)
63          {   // 코드 7-18 호출
64              A.Child childA = new A.Child();
65              Console.WriteLine();
66              // 코드 7-19 호출
67              B.Child childB = new B.Child();
68              Console.WriteLine();
69              // 코드 7-20 호출
70              C.Child childC = new C.Child();
71              C.Child childD = new C.Child("string");
72          }
        }
```

실습

실습7-5

- (코드 7-22 ~ 7-25) (교재 p.357-359)
  - 섀도잉, 변수 & 메서드 하이딩

```
01  namespace ShadowAndHide
02  {
03      class Program
04      {
05          // 코드 7-22: 섀도잉
06          public static int number = 10;
07
08          // 코드 7-23: 변수 하이딩
09          // 코드 7-25: 메서드 하이딩
10          class Parent
11          {
12              public int variable = 273;
13              public void Method()
14              {
15                  Console.WriteLine("부모의 메서드");
16              }
17          }
18          class Child : Parent
19          {
20              public string variable = "shadowing";
21
22              public void Method()
23              {
24                  Console.WriteLine("자식의 메서드");
25              }
            }
```

실습

실습7-5

- (코드 7-22 ~ 7-25) (교재 p.357-359)
  - 섀도잉, 변수 & 메서드 하이딩

```
27
28          static void Main(string[] args)
29          {
30              // 코드 7-22 확인
31              int number = 20;
32              Console.WriteLine(number);
33              Console.WriteLine();
34
35              // 코드 7-23 확인
36              Child childA = new Child();
37              Console.WriteLine(childA.variable);
38
39              // 코드 7-24: 숨겨진 멤버를 찾는 방법
40              Child childB = new Child();
41              Console.WriteLine(((Parent)childB).variable);
42
43              // 코드 7-25: 확인
44              Child childC = new Child();
45              childC.Method();
46              ((Parent)childC).Method();
47          }
48      }
49  }
50
```

실습

실습7-6

- (코드 7-28) (교재 p.364-365)
  - 하이딩

```
01  namespace UsageOfHidding
02  {
03      // 코드 7-28: 하이딩
04      class Program
05      {
06          class Animal
07          {
08              public void Eat()
09              {
10                  Console.WriteLine("냠냠 먹습니다.");
11              }
12          }
13          class Dog : Animal
14          {
15              public void Eat()
16              {
17                  Console.WriteLine("강아지 사료를 먹습니다.");
18              }
19          }
20          class Cat : Animal
21          {
22              public void Eat()
23              {
24                  Console.WriteLine("고양이 사료를 먹습니다.");
25              }
            }
```

실습

실습7-6

- (코드 7-28) (교재 p.364-365)
  - 하이딩

```
27
28          static void Main(string[] args)
29          {
30              List<Animal> Animals = new List<Animal>()
31              {
32                  new Dog(), new Cat(), new Cat(), new Dog(),
33                  new Dog(), new Cat(), new Dog(), new Dog()
34              };
35
36              foreach (var item in Animals)
37              {
38                  item.Eat();
39              }
40          }
41      }
42  }
43
44
45
46
47
48
49
50
```

실습

실습7-7

- (코드 7-29) (교재 p.365-366)
  - 오버라이딩

```
01  namespace UsageOfOverriding
02  {
03      // 코드 7-29: 오버라이딩
04      class Program
05      {
06          class Animal
07          {
08              public virtual void Eat()
09              {
10                  Console.WriteLine("냠냠 먹습니다.");
11              }
12          }
13          class Dog : Animal
14          {
15              public override void Eat()
16              {
17                  Console.WriteLine("강아지 사료를 먹습니다.");
18              }
19          }
20          class Cat : Animal
21          {
22              public override void Eat()
23              {
24                  Console.WriteLine("고양이 사료를 먹습니다.");
25              }
            }
```

실습

실습7-7

- (코드 7-29) (교재 p.365-366)
  - 오버라이딩

```
01
02          static void Main(string[] args)
03          {
04              List<Animal> Animals = new List<Animal>()
05              {
06                  new Dog(), new Cat(), new Cat(), new Dog(),
07                  new Dog(), new Cat(), new Dog(), new Dog()
08              };
09
10              foreach (var item in Animals)
11              {
12                  item.Eat();
13              }
14          }
15      }
16  }
17
18
19
20
21
22
23
24
25
```

실습

실습7-8

- (코드 7-33 - 7-34) (교재 p.371-372)
  - abstract 클래스 오류 및 abstract 메서드

```
01  namespace Abstract
02  {
03      // 코드 7-33: abstract 클래스 오류
04      namespace A
05      {
06          abstract class Parent
07          {
08              public void Test() { }
09          }
10
11          class Child : Parent
12          {
13              public void Test() { }
14          }
15      }
16
17      // 코드 7-34: abstract 메서드
18      namespace B
19      {
20          abstract class Parent
21          {
22              public abstract void Test();
23          }
24
25
```

실습

실습7-8

- (코드 7-33 - 7-34) (교재 p.371-372)
  - abstract 클래스 오류 및 abstract 메서드

```
01
02          class Child : Parent
03          {
04              // 코드 7-35: abstract 메서드와 관련된 오류 해결
05              // public override void Test() { }
06          }
07      }
08
09      class Program
10      {
11          static void Main(string[] args)
12          {
13              // 코드 7-33 호출(여기에서 오류가 발생합니다)
14              A.Parent parent = new A.Parent();
15              A.Child child = new A.Child();
16
17              parent.Test();
18              child.Test();
19          }
20      }
21  }
22
23
24
25
```

과제

과제5

- 세부 과제
  - (교안) 11주차, 12주차 실습코드(실습7-2~7-8) 중 4개 선택
    - 각 예제의 코드를 작성하여 실행한 후, 실행결과 화면을 캡처한 이미지 파일을 제출

- 제출 시 주의사항
  - 실행결과 이미지 캡처 시, 소스코드와 이미지가 겹치지 않게 모두 보이도록 할 것
  - 각 소스코드의 마지막줄에 "학과, 분반, 학번, 이름" 출력문 삽입
    - Console.WriteLine("학과, 분반, 학번, 이름"); 추가하여 본인임을 증빙
  - 실행결과 이미지(.jpg 등)를 모두 압축하여 파일(.zip) 1개로 제출
    - 파일명 "과제5_분반_학번_이름.zip"으로 제출 (세부 파일명은 예제번호 등 구분되게 임의로 지정)
  - 제출기한 : 2023년 6월 1일 자정까지(기한 외 추가제출은 기본적으로 불인정)
    - 마감일에 한해 사이버캠퍼스 오류로 인한 메일제출 허용(mnshim@sungkyul.ac.kr) 그 외 불인정
    - 메일제목: "마감일 사이버캠퍼스 오류, 과제5(분반,성명)"

감사합니다

mnshim@sungkyul.ac.kr