

상속과 다형성(1)

HyoJoon Han
동국대학교
han6343@dongguk.edu



상속과 다형성 소개

상속

다형성

- 상속과 다형성 : C#에서 반복을 줄이기 위해 사용하는 방법
- 강아지와 고양이를 나타내는 클래스 만들기

코드 7-1 Dog 클래스

/7장/Inheritance

```
class Dog
{
    public int Age { get; set; }
    public string Color { get; set; }

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

    public void Eat() { Console.WriteLine("냠냠 먹습니다."); }
    public void Sleep() { Console.WriteLine("쿨쿨 잠을 잡니다."); }
    public void Bark() { Console.WriteLine("왈왈 짖습니다."); }
}
```

코드 7-2 Cat 클래스

/7장/Inheritance

```
class Cat
{
    public int Age { get; set; }

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

    public void Eat() { Console.WriteLine("냠냠 먹습니다."); }
    public void Sleep() { Console.WriteLine("쿨쿨 잠을 잡니다."); }
    public void Meow() { Console.WriteLine("냥냥 읊니다."); }
}
```

- Dog와 Cat 클래스의 인스턴스를 만들고 메서드 호출하기

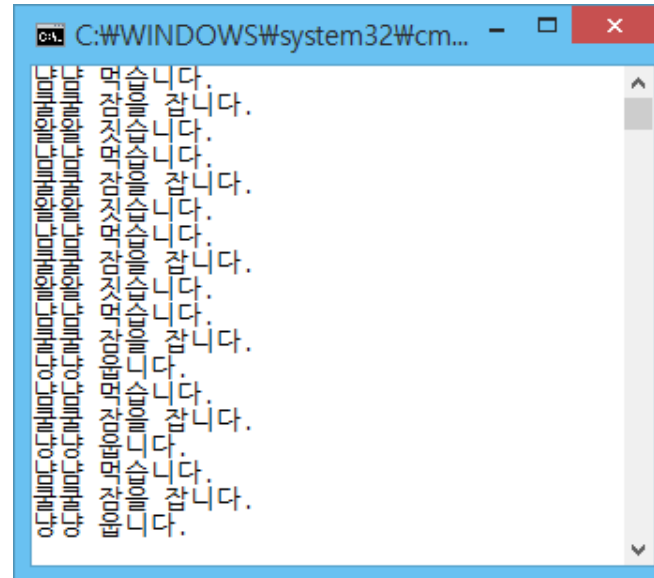
코드 7-3 Dog과 Cat 클래스의 인스턴스를 만들고 메서드 실행

/7장/Inheritance

```
static void Main(string[] args)
{
    List<Dog> Dogs = new List<Dog>() { new Dog(), new Dog(), new Dog() };
    List<Cat> Cats = new List<Cat>() { new Cat(), new Cat(), new Cat() };

    foreach (var item in Dogs)
    {
        item.Eat();
        item.Sleep();
        item.Bark();
    }

    foreach (var item in Cats)
    {
        item.Eat();
        item.Sleep();
        item.Meow();
    }
}
```



- 클래스 사이에 부모 자식 관계를 정의하는 작업
- 부모 클래스 하나는 자식을 여러 개 가질 수 있음
- 현재 Dog 클래스와 Cat 클래스의 구성

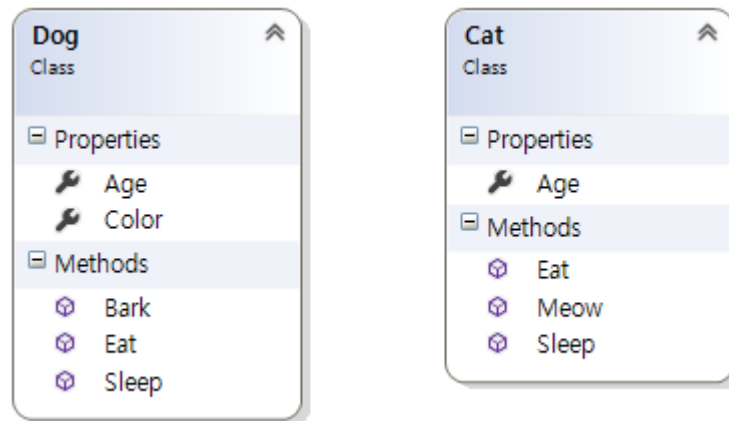
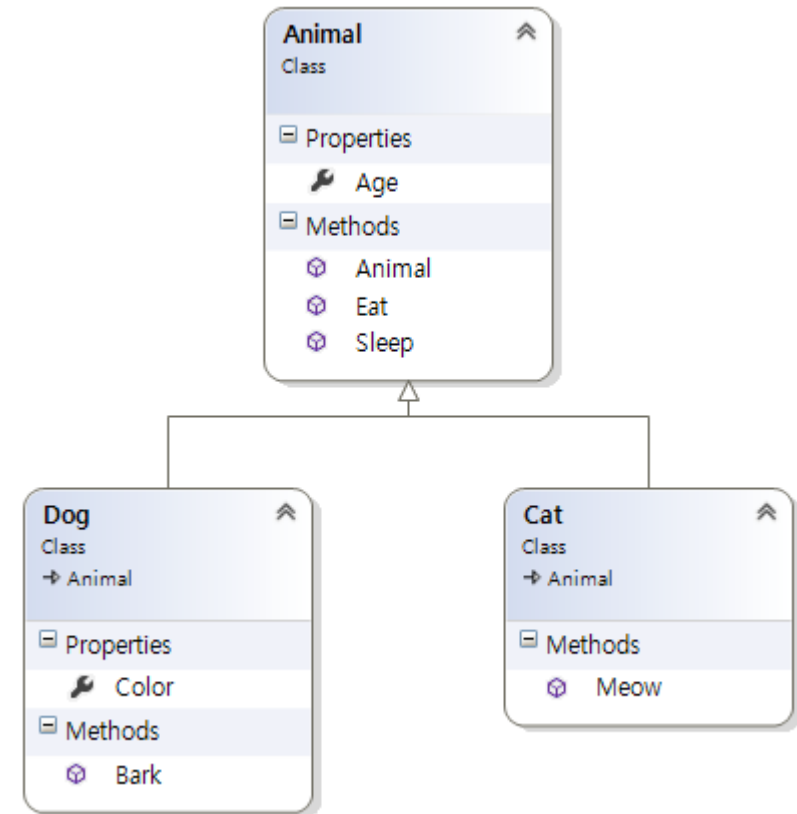


그림 7-2 분할되어 있는 클래스

- 부모 클래스(Animal)를 만드는 코드의 구성



- Animal 클래스 코드

코드 7-4 Animal 클래스

/7장/Inheritance

```
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 Dog와 Cat 클래스

/7장/Inheritance

```
class Dog : Animal ————— Animal 클래스의 상속을 받습니다.  
{  
    public string Color { get; set; }  
  
    public void Bark() { Console.WriteLine("왈왈 짖습니다."); }  
}  
  
class Cat : Animal ————— Animal 클래스의 상속을 받습니다.  
{  
    public void Meow() { Console.WriteLine("냥냥 읊니다."); }  
}
```

- 클래스의 부모 자식 관계가 형성되면 자식 클래스는 부모 클래스의 public 또는 protected 멤버에 접근 가능

- 부모의 모든 멤버 public 이므로 Dog 클래스의 인스턴스를 만들면 해당 인스턴스에서 자신의 멤버는 물론 부모의 멤버에 모두 접근 가능

```
Dog dog = new Dog();  
dog.
```

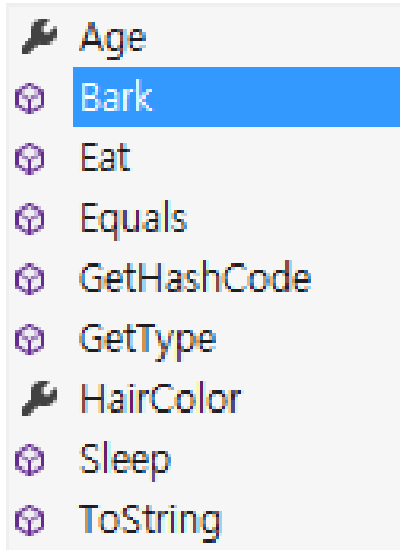


그림 7-4 Dog 클래스에서 접근할 수 있는 멤버 목록

- 다른 접근 제한자

표 7-1 C#의 접근 제한자

접근 제한자	내부 클래스	외부 클래스	파생 클래스	프로젝트
public	○	○	○	○
internal	○	○	○	
protected	○		○	
private	○			
protected internal	○	사용하는 클래스가 같은 어셈블리 안에 있을 때 접근 가능	○	

- base 키워드
 - 자식 클래스에서 부모 클래스에서 정의한 멤버의 사용

코드 7-6 부모에게서 상속받은 메서드 호출

/7장/Inheritance

```
class Animal
{
    public void Eat() { Console.WriteLine("냠냠 먹습니다."); }
    public void Sleep() { Console.WriteLine("쿨쿨 잠을 잡니다."); }
}

class Dog : Animal
{
    public void Test()
    {
        Eat();
        Sleep();
    }
}
```

부모에게서 상속받은 Eat() 메서드와 Sleep() 메서드를 호출합니다.

- 이름이 겹치는 등 특수한 이유로 부모의 메서드에 접근 불가할 경우 this 키워드와 같은 형태로 base 키워드 사용 (this : 자기 클래스를 나타내는 키워드, base : 부모 클래스를 나타내는 키워드)

```
class Dog : Animal
{
    public string Color { get; set; }

    public void Bark()
    {
        base.
    }
}
```

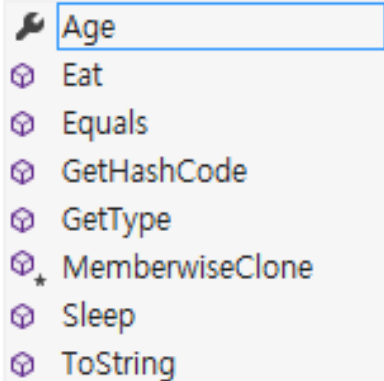


그림 7-5 부모를 나타내는 base 키워드

- protected 접근 제한자
 - private과 비슷하지만 상속한 클래스(파생 클래스)에서는 접근 가능

코드 7-7 세 가지 접근 제한자

/7장/ThreeModifiers

```
class Program
{
    class Animal
    {
        private void Private() { }
        protected void Protected() { }
        public void Public() { }

        public void TestA()
        {
            Private();
            Protected();
            Public();
        }
    }
}
```

자신의 클래스 내부에서는 모든 멤버를 사용할 수 있습니다.

```
class Dog : Animal
```

```
{
```

```
    public void TestB()
```

```
    {
```

```
        Protected();
```

```
        Public();
```

```
    }
```

```
}
```

```
static void Main(string[] args)
```

```
{
```

```
    Dog dog = new Dog();
```

```
    dog.Public();
```

```
}
```

```
}
```

상속받은 클래스에서는 private 접근 제한자가 붙은 멤버를 사용할 수 없습니다.

이외의 모든 장소에서는 public 접근 제한자가 붙은 멤버만 사용할 수 있습니다.

- 하나의 클래스가 여러 형태로 변환될 수 있는 성질
- 자식 클래스가 부모 클래스로 위장
- 예

```
Animal dog = new Dog();  
Animal cat = new Cat();
```

- 인스턴스 dog는 외관상으로 자료형 Animal이지만 실제 내부에는 Dog가 들어있음
- 외관상으로는 Animal 객체이므로 사용할 수 있는 멤버는 Animal 클래스의 멤버뿐임

dog.

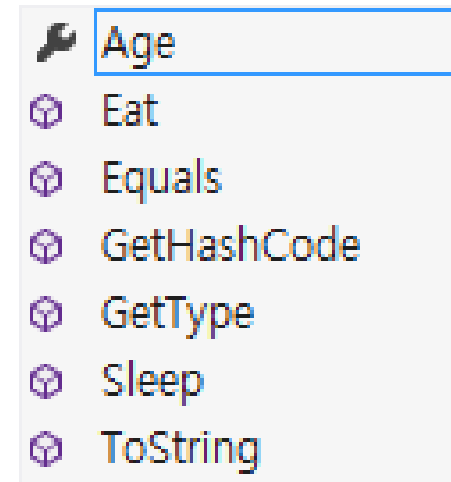


그림 7-6 부모로 위장한 자식은 부모의 멤버만 사용 가능

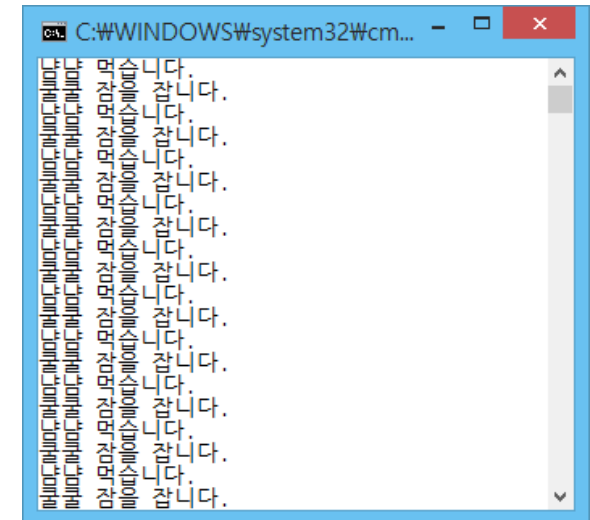
코드 7-9 다형성을 사용한 코드 중복 해결

```
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();
        item.Sleep();
    }
}
```

하나의 리스트를 사용합니다.

하나의 반복문을 사용합니다.



- 자식 클래스에 있는 메서드 사용 위해, 자식 클래스로 자료형 변환

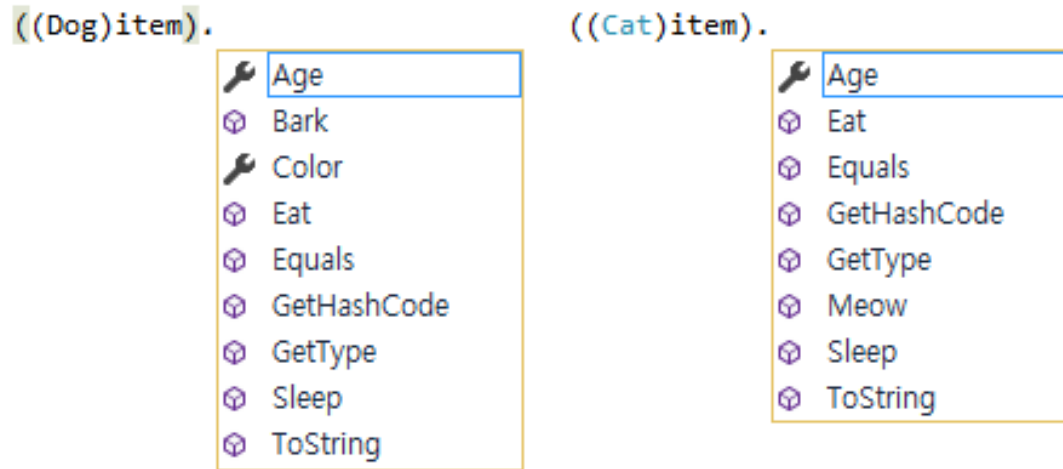
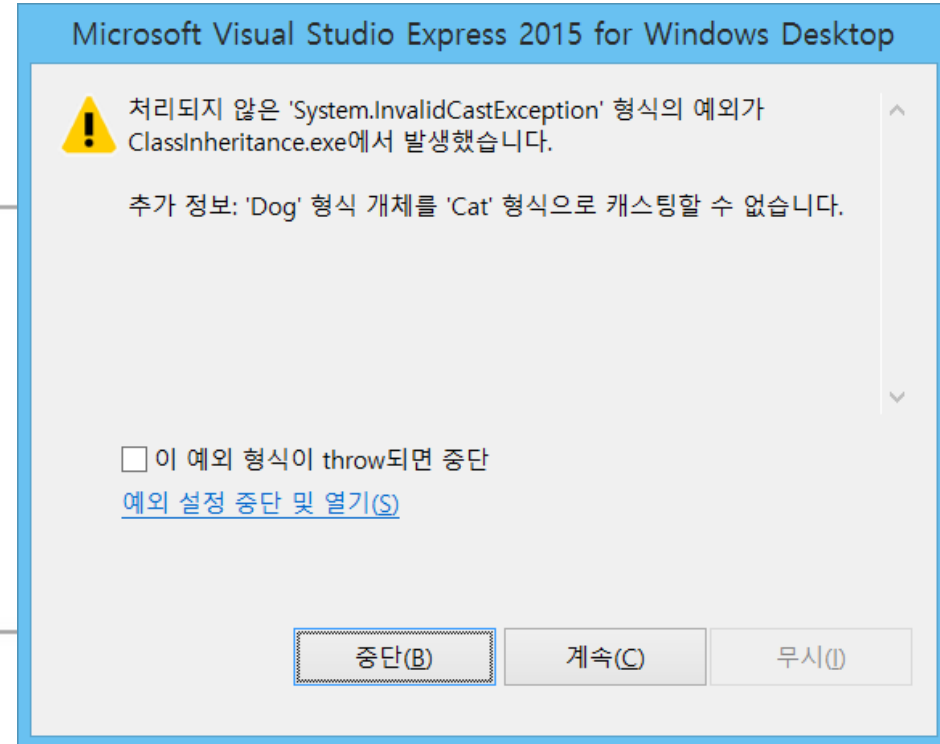


그림 7-8 위장을 해제하고 자신의 멤버를 사용

- 무작정 Cat 클래스로 변환해서 사용

코드 7-10 무작정 변환해서 메서드 호출

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



- 내부에 조건문 넣어 Dog 객체는 Dog 객체로 변환 Bark() 메서드 호출, Cat 객체는 Cat 객체로 변환 Meow() 메서드 호출이 필요

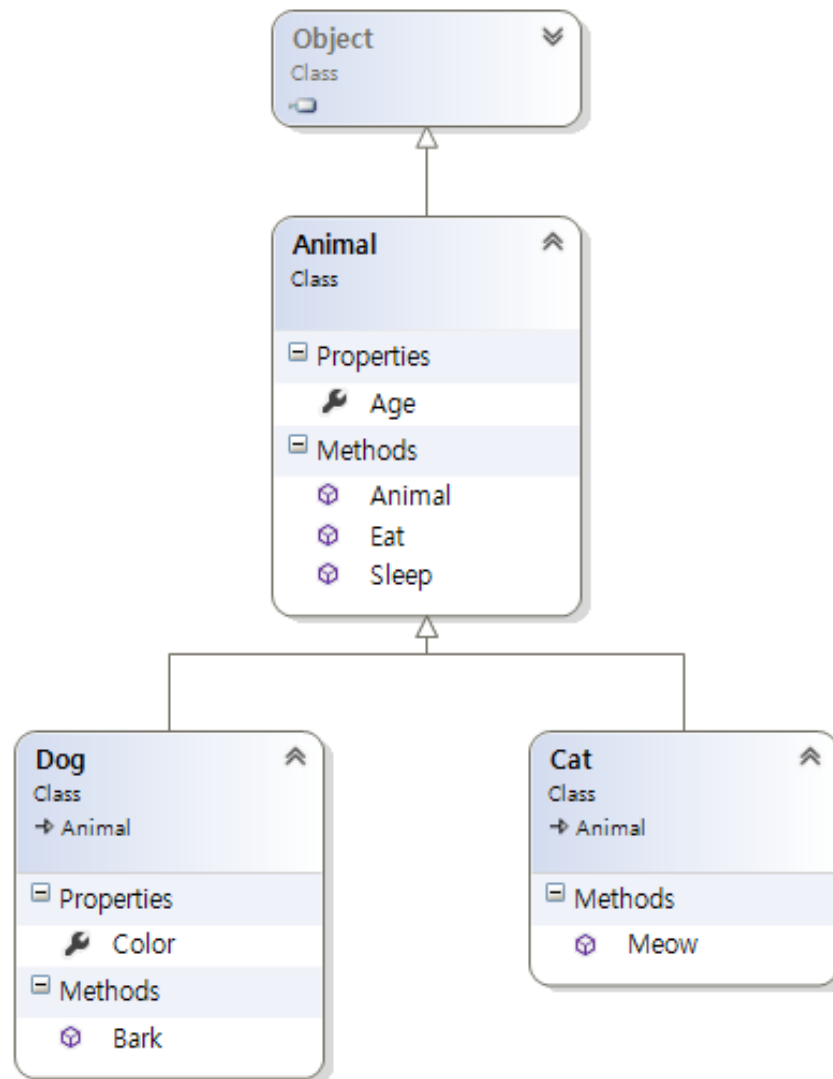
- 최상위 객체
 - C#에서 만드는 모든 객체는 Object라는 객체의 상속을 받게 됨

코드 7-11 Object 객체의 선언

```
class Object
{
    public Object();

    public virtual bool Equals(object obj);
    public static bool Equals(object objA, object objB);
    public virtual int GetHashCode();
    public Type GetType();
    protected object MemberwiseClone();
    public static bool ReferenceEquals(object objA, object objB);
    public virtual string ToString();
}
```

- 상속 관계



코드 7-12 object 객체의 다형성 예제(1)

/7장/CInheritance

```
List<Object> listOfObject = new List<Object>();  
listOfObject.Add(new Dog());  
listOfObject.Add(new Cat());
```

코드 7-13 object 객체의 다형성 예제(2)

/7장/CInheritance

```
List<Object> listOfObject = new List<Object>();  
listOfObject.Add(new Dog());  
listOfObject.Add(new Cat());  
listOfObject.Add(52);  
listOfObject.Add(521);  
listOfObject.Add(52.273f);  
listOfObject.Add(52.273);
```

- 코드 저장해 놓으세요 (다음 수업 때 이어서 사용)
- 실습 - 친구 관리 프로그램

1) 윈도우 폼 디자인

The screenshot shows a Windows Form titled "친구 관리 프로그램". The form contains the following elements:

- textBox1~4**: Four text input fields for "이름" (Name), "전화번호" (Phone Number), "주소" (Address), and "직업/전공" (Job/Major).
- button1**: A button labeled "친구추가" (Add Friend) at the bottom left.
- button2**: A button labeled "친구 정보 간단 확인" (Quickly check friend information) in the top right.
- button3**: A button labeled "친구 정보 확인" (Check friend information) in the top right, next to button2.
- textBox5**: A large empty text area on the right side of the form.
- Radio Buttons**: Two radio buttons at the bottom left labeled "○ 고등학교 친구" (High school friend) and "○ 대학교 친구" (University friend).

2) 클래스 추가

- Friend.cs , HighFriend.cs , UnivFriend.cs 3개의 클래스 추가

3) Friend.cs 코드 작성

```
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 string showData()
    {
        return "이름 : " + name + Environment.NewLine + "전화 : "
            + phoneNum + Environment.NewLine + "주소 : " + addr;
    }

    public string showBasicInfo()
    {
        return "";
    }
}
```

4) HighFriend.cs 코드 작성

```
class HighFriend : Friend
{
    private string work;

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

    public string showData()
    {
        string str = base.showData();
        str += Environment.NewLine + "직업 : " + work;
        return str;
    }

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

5) UnivFriend.cs 코드 작성

```
class UnivFriend : Friend
{
    string major;

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

    public string showData()
    {
        string str = base.showData();
        str += Environment.NewLine + "전공 : " + major;
        return str;
    }

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


6) 폼 코드 작성

```
public partial class Form2 : Form
{
    private List<Friend> myFriend = new List<Friend>();

    public Form2()
    {
        InitializeComponent();
    }
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    string name = textBox1.Text;
    string phoneNum = textBox2.Text;
    string addr = textBox3.Text;
    string majorOrJob = textBox4.Text;

    if (radioButton1.Checked)
    {
        HighFriend newFriend = new HighFriend(name, phoneNum, addr, majorOrJob);
        myFriend.Add(newFriend);
    }
    else
    {
        UnivFriend newFriend = new UnivFriend(name, phoneNum, addr, majorOrJob);
        myFriend.Add(newFriend);
    }

    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    MessageBox.Show("친구 추가 완료");
}
```

6) 폼 코드 작성

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

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

7) 결과 확인

친구 관리 프로그램

이름

전화번호

주소

직업/직종

☒ 고등학교 친구 ☐ 대학교 친구

친구추가

친구 정보 간단 확인

친구 정보 확인

이름 : 홍길동
연락처 : 010-0000-0000
주소 : 1212
직업/직종 : 고길동
직업/직종 : 010-1234-5678
수 : 13