

제8장

모듈과 라이브러리

변영철

- 오픈 소스(Open Source)
- Know How가 아니라 Know Where
- 코드 암기(X)
- 마음에 드는 코드 덩어리를 나중에 쉽게 재사용할 수 있도록 코드 (클래스) 모듈화
- 모듈은 블랙박스 (내부가 깜깜, 몰라도 됨)
- 자동차, 자판기, TV, 스마트폰 (내부를 몰라도 잘 사용)
- 어떻게 동작하는지보다는 **어떻게 사용하는지**



이 장을 공부하면

- 프로그램에서 마음에 드는 코드 일부를 밖으로 뽑아내어 클래스 **부품(모듈)**으로 만들 수 있다.
- 내가 만든 클래스 모듈을 쉽게 **재사용**할 수 있다. (**라이브러리**)

1. 가장 간단한 프로그램 작성

- 새로운 프로젝트 My 생성
 - 빈 프로젝트(.NET Framework)
- 다음 코드 작성

```
class My
{
    public static void Main()
    {
        System.Console.WriteLine("Hello,World!");
    }
}
```

2. 코드 추상화

- 멤버 함수 Say로 추상화

```
class My
{
    public void Say()
    {
        System.Console.WriteLine("Hello,World!");
    }

    public static void Main()
    {
        Say(); //Error. Why?
    }
}
```

2. 코드 추상화

- 멤버 함수 Say로 추상화

```
class My
{
    public void Say()
    {
        System.Console.WriteLine("Hello,World!");
    }

    public static void Main()
    {
        My gildong = new My();
        gildong.Say();
    }
}
```

3. 새로운 클래스로 만들기

```
class Hello
{
    public void Say()
    {
        System.Console.WriteLine("Hello,World!");
    }
}
```

```
class My
{
    public static void Main()
    {
        Hello gildong = new Hello();
        gildong.Say();
    }
}
```

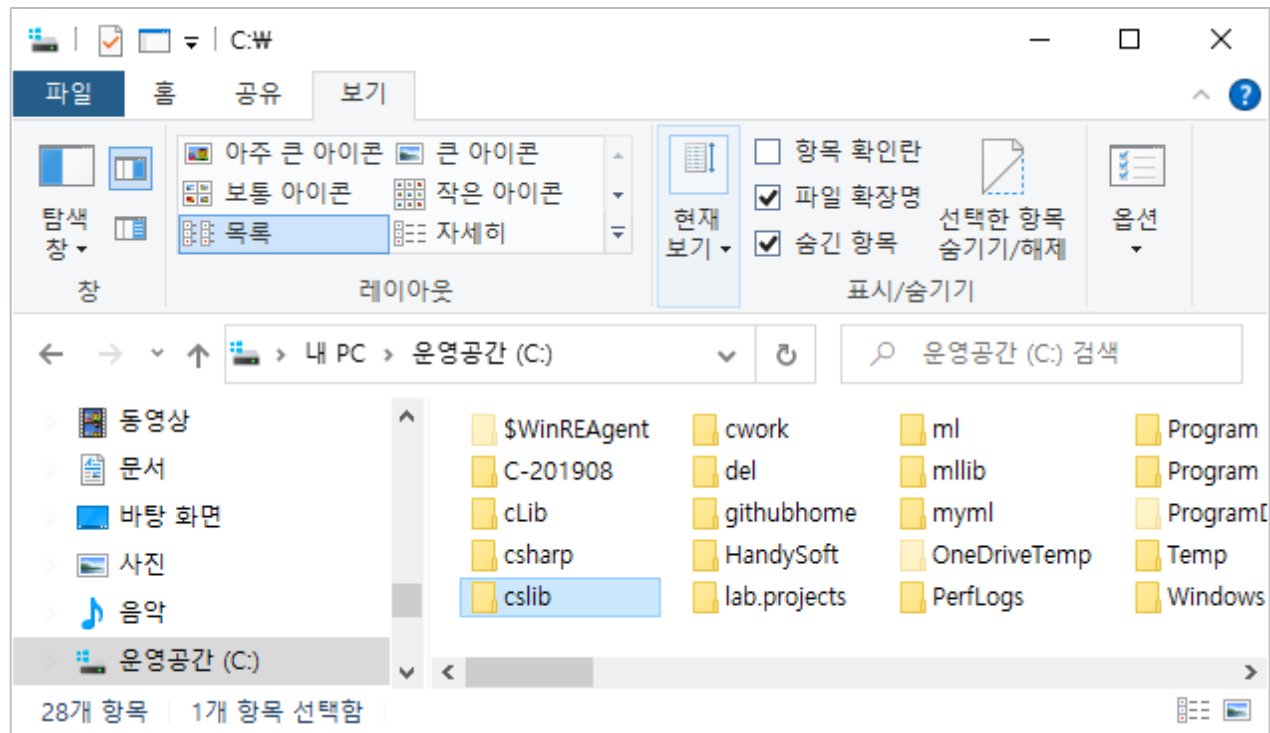
4. 새로운 파일로 이동

- 프로젝트 | 새 항목 추가
- ‘클래스’ 선택한 후 파일 이름 Hello.cs 입력
- 기존의 클래스 코드 삭제 후 Hello를 이동

```
class Hello
{
    public void Say()
    {
        System.Console.WriteLine("Hello,World!");
    }
}
```

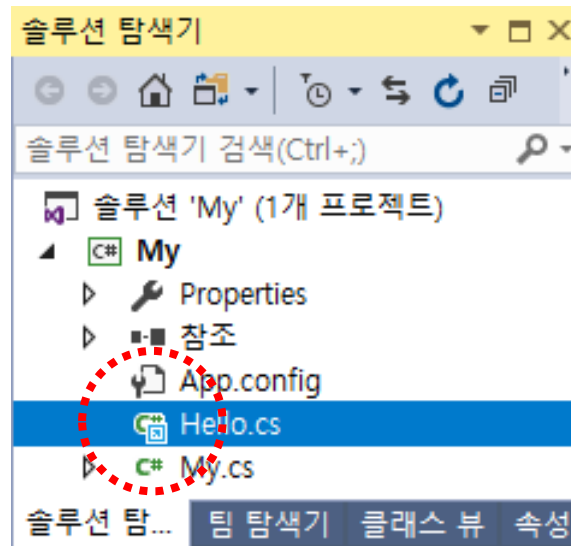
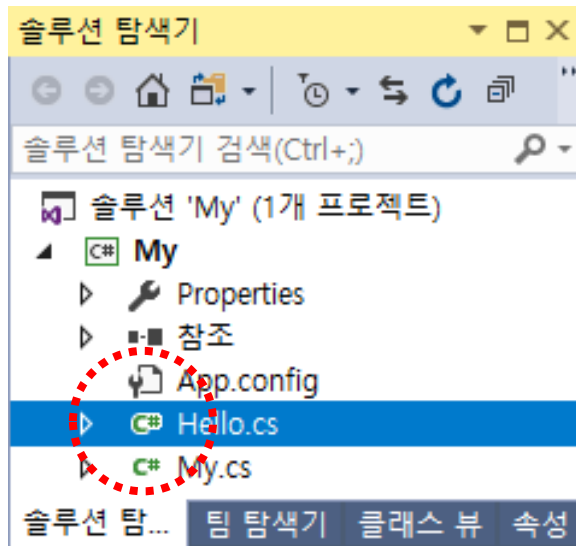

5. 파일을 외부 폴더로 옮기기

- C 루트에 라이브러리 **폴더**(cslib) **생성**
- 금방 만든 Hello.cs **파일**을 cslib 폴더로 **이동**



5. 파일을 외부 라이브러리 폴더로 옮기기

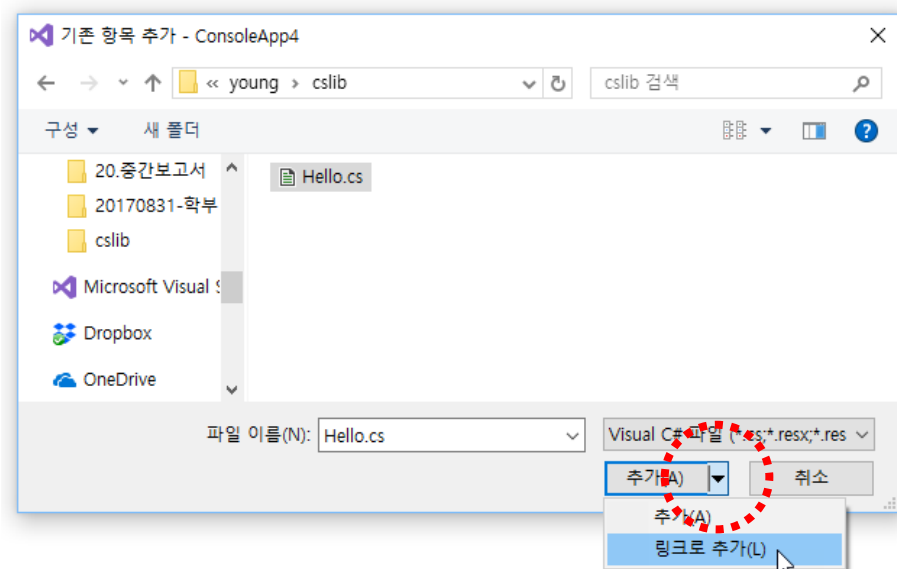
- 솔루션 탐색기에서 Hello.cs 파일 **삭제** (왜냐하면 파일이 cslib에 있으므로...)
- cslib의 Hello.cs 파일을 My 프로젝트에 **링크로 추가**



끝!

6. 클래스 모듈 이용하기

- 새로운 프로젝트 생성
- 프로젝트 | 기존 항목 추가...
 - clib 폴더로 이동하여 Hello.cpp 추가
 - 드롭다운 버튼 클릭 후 링크로 추가



6. 클래스 모듈 이용하기

```
class My
{
    public static void Main()
    {
        Hello gildong = new Hello();
        gildong.Say();
    }
}
```



사람

```
class My
{
    Hello gildong;
    public static void Main()
    {
        gildong = new Hello();
        gildong.Say();
    }
}
```

핸드폰

- 여러분 호주머니에 있는 스마트 폰은 여러분이 **소유**하는 것(**has-a**), 여러분의 멤버 변수
- 사람은 핸드폰을 **갖는다**.
- My는 Hello를 갖는다. My **has** a Hello.

7. 모듈화의 또 다른 예제 (Is-a, has-a)

새로운 '빈 프로젝트'
생성 후 다음 코드 작성

- 멤버는 멤버를 액세스할 수 있다.

```
class MainForm //gildong
{
    public int Font;
    public void CreateGraphics()
    {
    }
    public void MainForm_Click()
    {
        CreateGraphics();
        Font = 1;
        System.Console.WriteLine("Hello!");
    }
}
class My
{
    public static void Main()
    {
    }
}
```

Is-a

- 멤버는 멤버를 액세스할 수 있다.
- 눈에 보이는 멤버만이 멤버가 아니다. 상속 받았을 수도 있다.

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    public void MainForm_Click()
    {
        CreateGraphics();
        Font = 1;
        System.Console.WriteLine("Hello!");
    }
}

class My
{
    public static void Main()
    {
    }
}
```

MainForm is a
Form

- 함수로 추상화하고 싶은 부분

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    public void MainForm_Click()
    {
        CreateGraphics();
        Font = 1;
        System.Console.WriteLine("Hello!");
    }
}

class My
{
    public static void Main()
    {
    }
}
```

- 멤버는 멤버를 액세스 할 수 있다.
- 함수로 추상화하고 싶은 부분

this의 의미는?
 "여기에 있는,
 (길동이)자기
 자신이 가지고
 있는..."

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    public void MainForm_Click()
    {
        this.CreateGraphics();
        this.Font = 1;
        System.Console.WriteLine("Hello!");
    }
}

class My
{
    public static void Main()
    {
    }
}
```

“

이 코드를
Display 함수
 로 추상화해
 보자.


```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    public void Display()
    {
        this.CreateGraphics();
        this.Font = 1;
        System.Console.WriteLine("Hello!");
    }

    public void MainForm_Click()
    {
        Display();
    }
}
```

- 이 멤버 함수를 밖으로 꺼내어 두고 재사용하고 싶다.

새로운 클래스를
만든 후, 그 안으로
옮김.



```
class MyUtil
{
    public void Display() {
        this.CreateGraphics();
        this.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}
class MainForm : Form //gildong
{
    public void MainForm_Click()
    {
        Display();
    }
}
```

멤버가 아니니 오류!

has-a

클래스는 뭐
하라고 있는 것?



```
class MyUtil //util
{
    public void Display() {
        this.CreateGraphics();
        this.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}
class MainForm : Form //gildong
{
    public void MainForm_Click()
    {
        MyUtil util = new MyUtil();
        util.Display();
    }
}
```

has-a

```
class MyUtil //util
{
    public void Display() {
        this.CreateGraphics();
        this.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}
class MainForm : Form //gildong
{
    MyUtil util = new MyUtil();
    public void MainForm_Click()
    {
        util.Display();
    }
}
```

has-a

어떤 문제?

```
class MyUtil //util
{
    public void Display() {
        this.CreateGraphics();
        this.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    MyUtil util = new MyUtil();
    public void MainForm_Click()
    {
        util.Display();
    }
}
```

has-a

```
class MyUtil
{
    public void Display(Form mf) {
        mf.CreateGraphics();
        mf.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}
class MainForm : Form //gildong
{
    MyUtil util = new MyUtil();
    public void MainForm_Click()
    {
        util.Display(this);
    }
}
```

has-a

사람은 핸드폰을 갖는다.

MainForm 객체 gildong이는
MyUtil 객체 util을 갖는다.
(팀장과 팀원 관계?)

```
class MyUtil
{
    public void Display(Form mf) {
        mf.CreateGraphics();
        mf.Font = 1;
        Console.WriteLine("Hello!");
    }
}
```


```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
}

class MainForm : Form //gildong
{
    MyUtil util = new MyUtil();
    public void MainForm_Click()
    {
        util.Display(this);
    }
}
```

앞으로 필요할 때
사용할 모듈
(라이브러리)

8. Has-a 모듈화 실전 연습

- Windows Forms 앱 프로젝트 만들기
- 파일명 바꾸기
- 폼 윈도우 클릭 이벤트 핸들러 함수 추가
- 아래 코드 작성 후 MyUtil로 모듈화

```
private void MainForm_Click(object sender, EventArgs e)
{
    
    Graphics cheolsu = CreateGraphics();

    cheolsu.DrawString("Click!", Font,
        new SolidBrush(Color.Blue), 10, 10);
}
```


9. Is-a 모듈화로 구현하기

- MyUtil 클래스를 MainForm과 Form 사이에 넣기
- 코드 수정

10. 모듈(클래스) 만드는 방법 2가지

- 부모 클래스를 만든 후 코드 이동과 상속
 - is-a 관계
- 별도의 클래스를 만든 후 코드 이동
 - has-a 관계

11. 나만의 라이브러리, 왜 좋을까?

- 나중에 쉽게 코드를 재사용 할 수 있다.
- 코드가 어떻게 돌아가는지 잊어버려도 문제 없다(블랙박스).
- 쉽게 구현할 수 있으므로 프로그래밍이 재미 있다.

이 장을 공부하면

- 프로그램에서 마음에 드는 코드 일부를 밖으로 뽑아내어 클래스 **부품(모듈)**으로 만들 수 있다.
- 블랙 박스로서의 클래스 모듈을 이해할 수 있다.
- 내가 만든 클래스 모듈을 나중에 필요할 때 **재사용**할 수 있다. (**라이브러리**)