모듈과 라이브러리

제주대학교 컴퓨터공학과
변영철 교수
(ycb@jejunu.ac.kr)

이 장을 공부하면

- 내가 작성한 프로그램에서 일부를 클래스 부품(라이브러리)으로 만들 수 있다.
- 내가 만든 라이브러리를 쉽게 이용(재사용)할 수 있다.

1. 가장 간단한 프로그램 작성

- 새로운 프로젝트 My 생성
- 기존의 코드를 삭제한 후 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();
    }
}
```

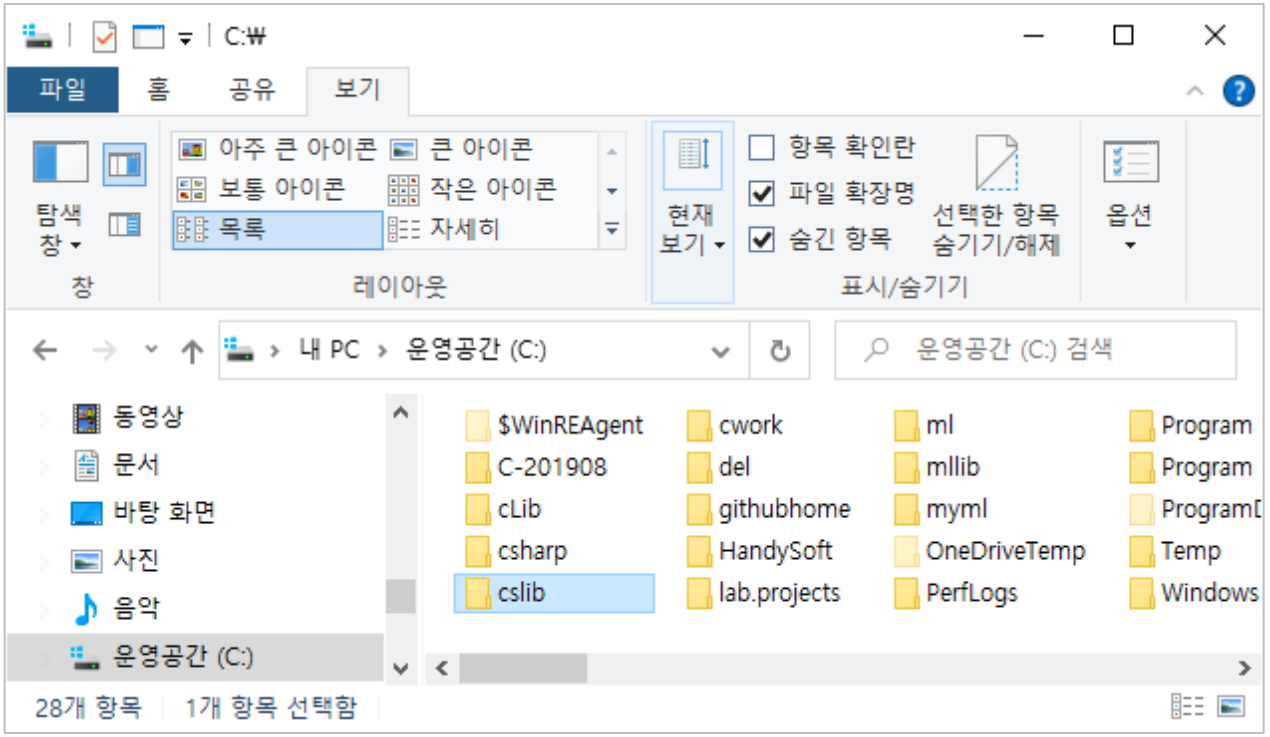
4. 새로운 파일로 이동

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

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

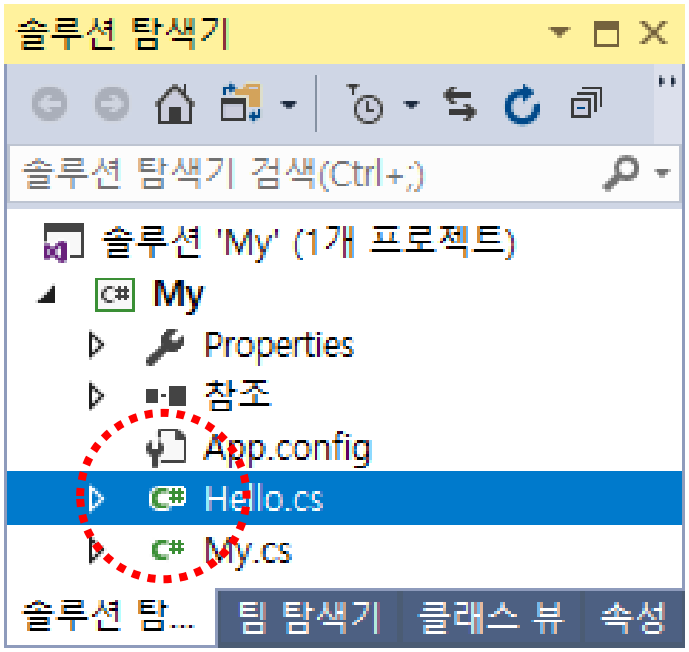
5. 라이브러리 폴더에 모아두기

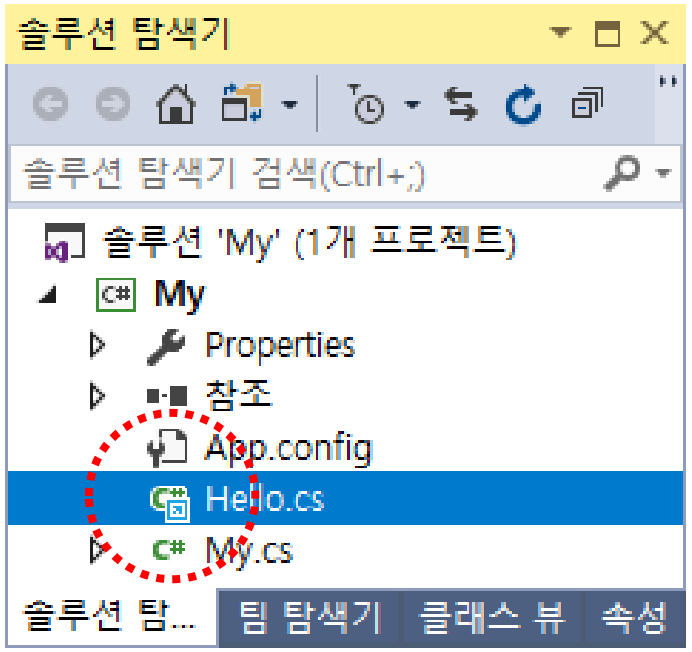
- C 루트에 라이브러리 폴더(cslib) 생성
- 금방 만든 Hello.cs 파일을 cslib 폴더로 이동



5. 라이브러리 폴더에 모아두기

- 솔루션 탐색기에서 Hello.cs 파일 **삭제** (왜냐하면 파일을 cslib로 이동했으므로...)
- cslib의 Hello.cs 파일을 My 프로젝트에 **링크로 추가**

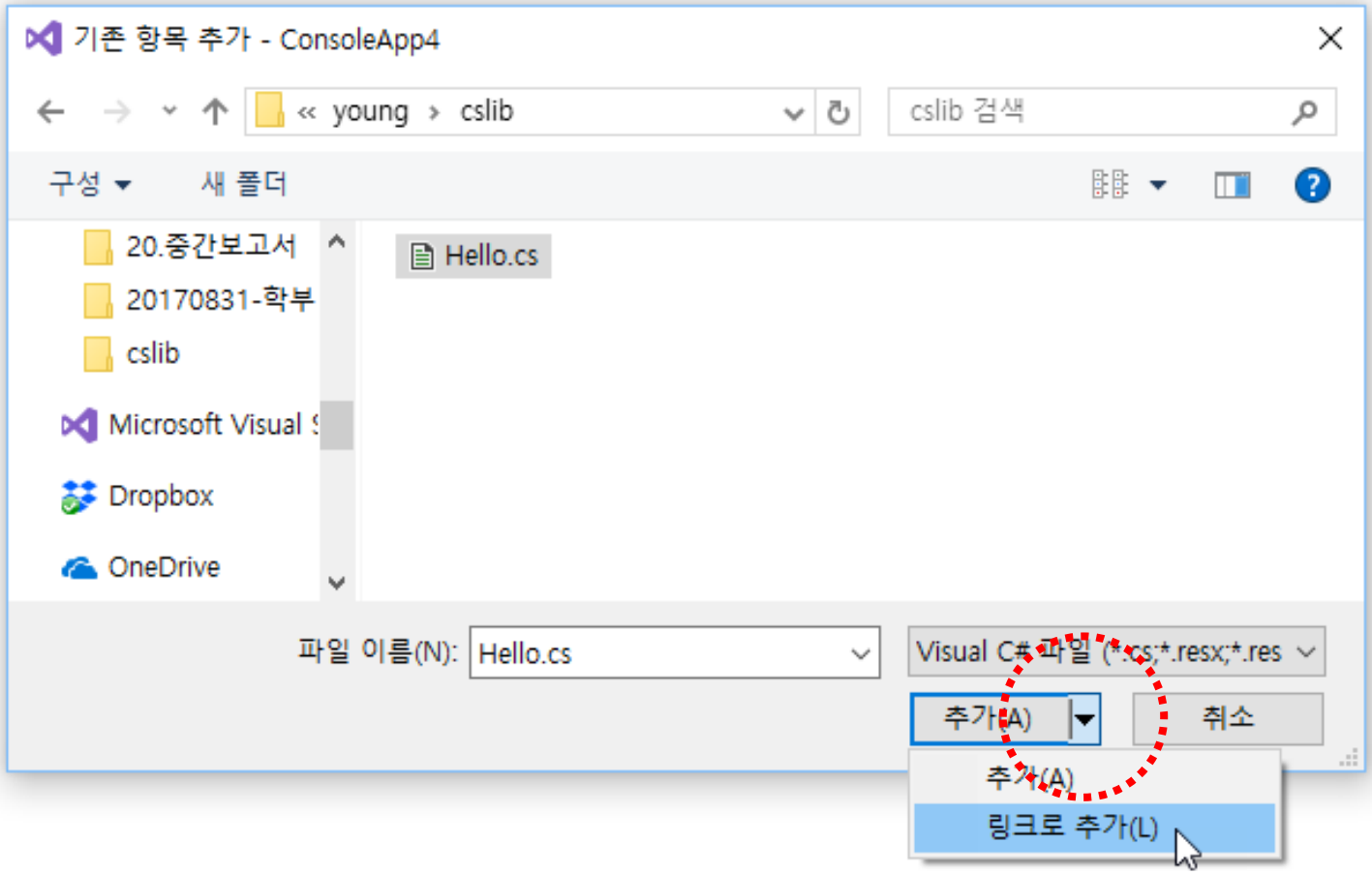




끝!

6. 라이브러리 이용하기

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


기존 항목 추가 - ConsoleApp4
« young › cslib
cslib 검색
구성
새 폴더
20.중간보고서
20170831-학부
cslib
Microsoft Visual S
Dropbox
OneDrive
Hello.cs
파일 이름(N):
Hello.cs
Visual C# 파일 (*.cs;*.resx;*.res
추가(A)
취소
추가(A)
링크로 추가(L)

독립된 클래스로 구현
has-a

새로운 '콘솔 응용' 프로젝트 생성 후 오른쪽 코드 작성

- **멤버는 멤버를 액세스 할 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()
    {
    }
}
```

```
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.**

- **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.
- 추상화하고 싶은 부분

```
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()
    {
    }
}
```

코드추상화

- 이 멤버 함수를 밖으로 꺼내어 두고 두고 재사용하고 싶다.

```
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();
    }
}
```

다른 클래스로

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

다른 클래스로

클래스는 뭐
하라고 있는 것?

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

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

다른 클래스로

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

어떤 문제?

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

다른 클래스로

```
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
{
    public void MainForm_Click()
    {
            MyUtil util = new MyUtil();
            util.Display(this);
    }
}
```

has-a 관계

사람은 핸드폰을
갖는다.

MainForm객체는
MyUtil객체를 갖는다.

```
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
{
    MyUtil util = new MyUtil();
    public void MainForm_Click()
    {
        util.Display(this);
    }
}
```

베이스 클래스로 구현
is-a

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

- 멤버는 멤버를 액세스 할 수 있다.
- 눈에 보이는 멤버만이 멤버가 아니다.

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

코드 추상화

- **클래스 끼워 넣기!**

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

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

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
};
class MyUtil : Form
{
    public void Display() {
        CreateGraphics();
        Font = 1;
        System.Console.WriteLine("Hello!");
    }
}
class MainForm : MyUtil
{
    public void MainForm_Click()
    {
        Display();
    }
}
```

코드를 위로 옮겼다!

is-a 관계

사람은 동물이다.

아랫것은 위의 것이다.
MainForm은 MyForm이다.

```
class Form
{
    public int Font;
    public void CreateGraphics()
    {
    }
};
class MyForm : Form
{
    public void Display() {
        CreateGraphics();
        Font = 1;
        System.Console.WriteLine("Hello!");
    }
}
class MainForm : MyForm
{
    public void MainForm_Click()
    {
        Display();
    }
}
```

따라서, 모듈(클래스) 만드는 방법 2가지

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

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

- 내가 작성한 프로그램에서 일부를 클래스 부품(라이브러리)으로 만들 수 있다.
- 내가 만든 라이브러리를 쉽게 이용(재사용)할 수 있다.