

C# 스터디

Value Object: VO

- VO는 데이터를 모을 수 있는 저장소
- Get/Set 속성을 이용해 변수 설정 및 값 수정
- 데이터 관리에 효과적

1. VO 클래스 생성 및 생성자

```
class BookVO
{
    private string id;    // 도서번호
    private string name;  // 도서명
    private string publisher; // 출판사명
    private string author; // 저자명
    private string price; // 가격
    private string quantity; // 수량

    public BookVO()
    {
        // 생성자
    }

    public BookVO(string id, string name, string publisher, string author, string price, string quantity)
    {
        this.id = id;
        this.name = name;
        this.publisher = publisher;
        this.price = price;
        this.quantity = quantity;
    }
}
```

1. VO get/set method, ToString()

```
public string Id    // get/set method
{
    get { return id; }
    set { id = value; }
}

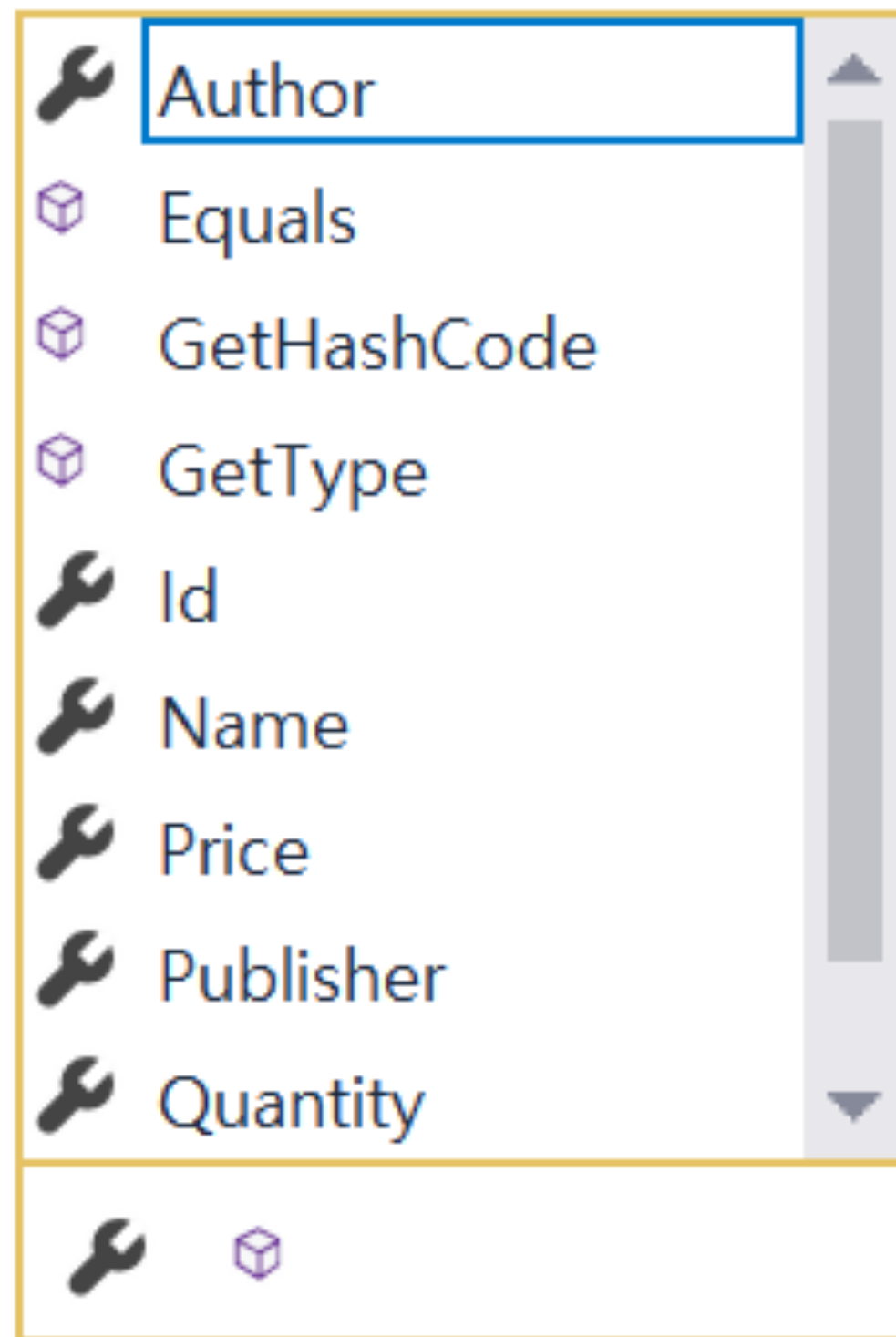
public string Name
{
    get { return name; }
    set { name = value; }
}

public string Publisher
{
    get { return publisher; }
    set { publisher = value; }
}
```

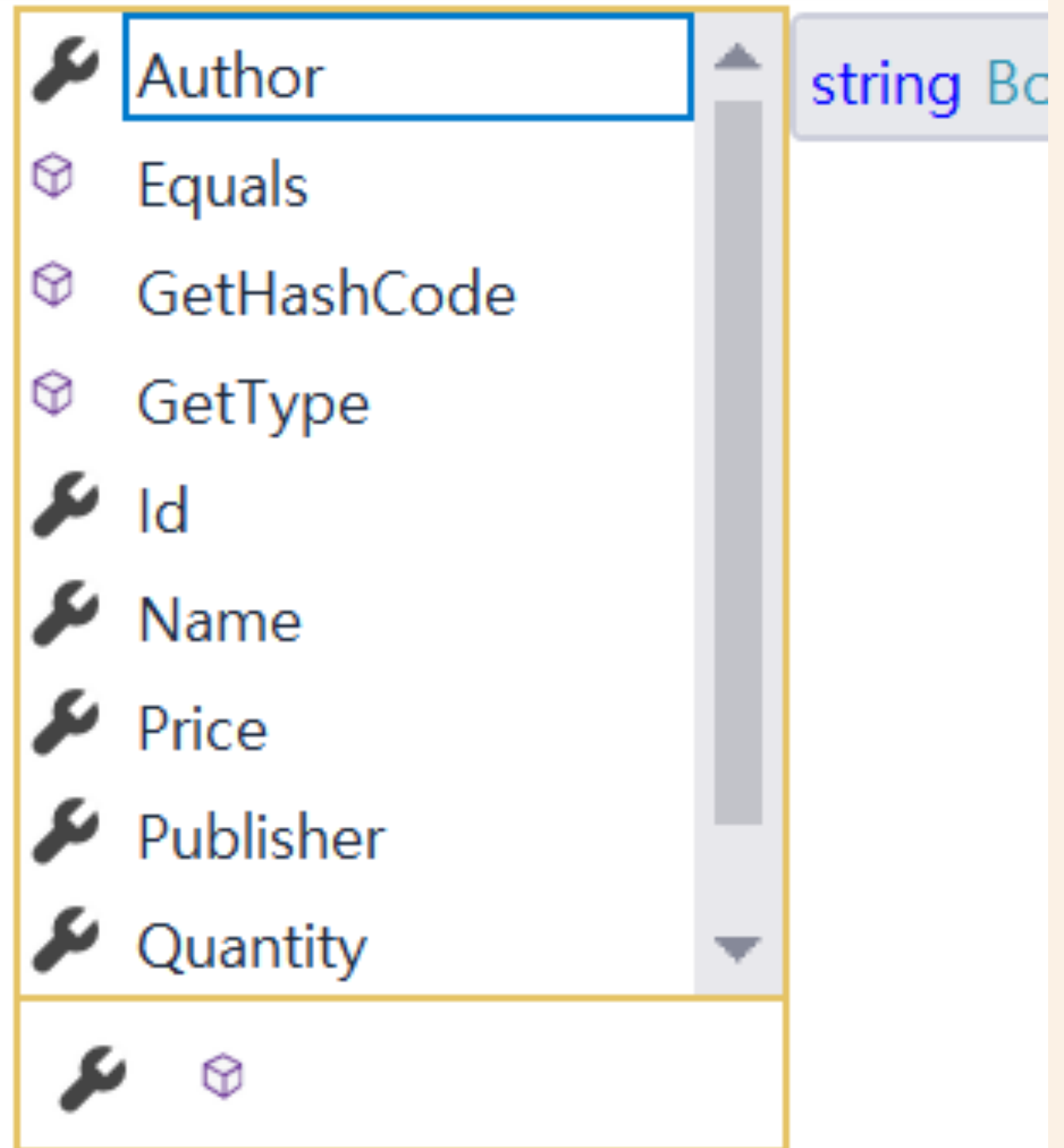
```
public override string ToString()
{
    return "Book id: " + id + ", Name: " + name + ", Author: " + author;
}
```

1. VO 사용 예시

```
BookVO bookVO = new BookVO();  
bookVO.~
```



```
List<BookVO> bookList = new List<BookVO>();  
bookList[0].Author = "Kim";  
bookList[1].~
```



상속(Inheritance)

- 부모 클래스가 자식 클래스의 기반이 됨
- 코드의 재사용성을 높이기 위해 사용



2. 상속 예시

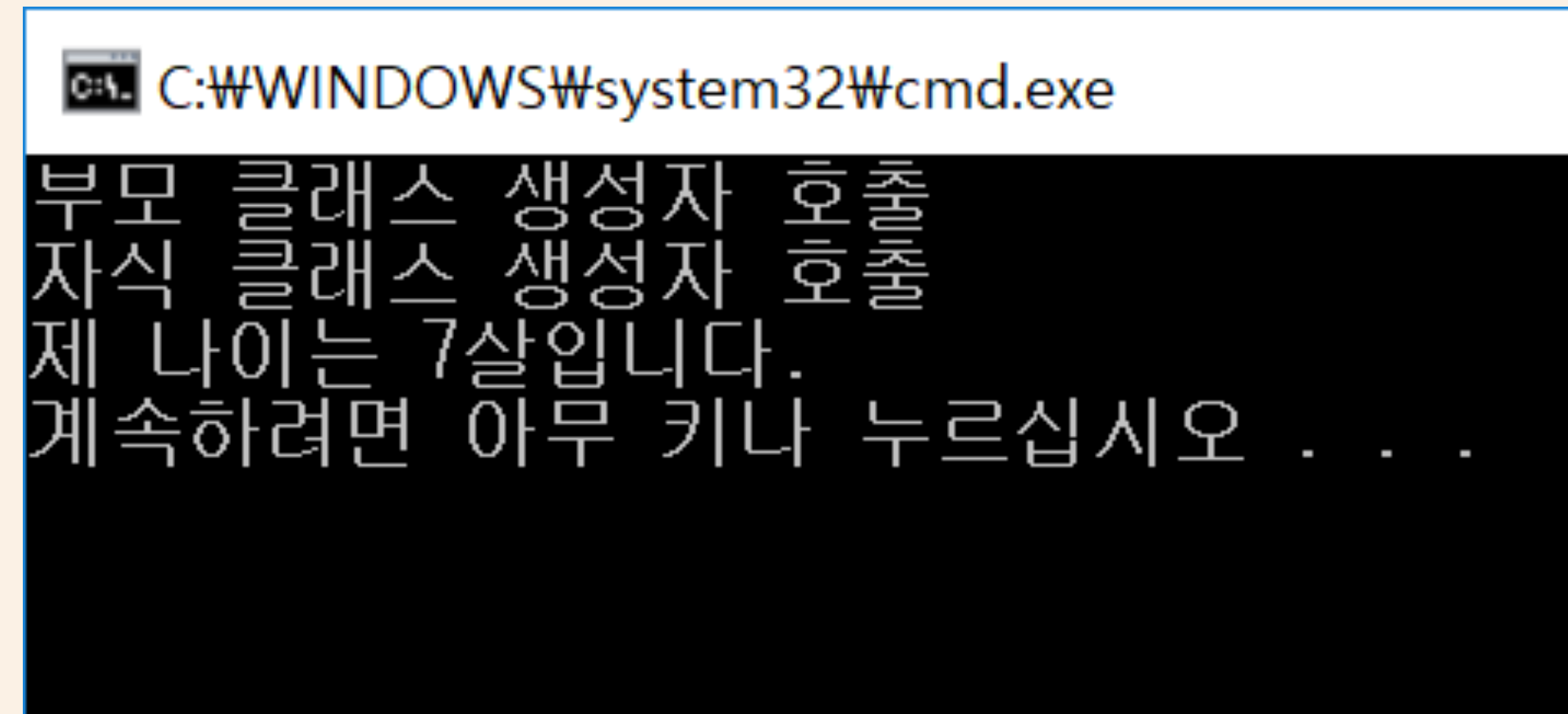
```
class Parent // 부모 클래스
{
    public int age; // private으로 선언된 멤버는 상속 불가

    public Parent()
    {
        Console.WriteLine("부모 클래스 생성자 호출");
    }

    public void DisplayAge()
    {
        Console.WriteLine("제 나이는 {0}살입니다.", age);
    }
}

class Child : Parent // 부모 클래스 상속
{
    public Child(int age)
    {
        this.age = age;
        Console.WriteLine("자식 클래스 생성자 호출");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Child child = new Child(7);
        child.DisplayAge();
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
부모 클래스 생성자 호출
자식 클래스 생성자 호출
제 나이는 7살입니다.
계속하려면 아무 키나 누르십시오 . . .
```

3. 싱글톤 패턴

```
class Ui

    private static Ui UiInstance = null;

    public static Ui Instance          //싱글톤 패턴으로 선언
    {
        get
        {
            if UiInstance == null)
                UiInstance = new Ui();
            return UiInstance;
        }
    }
}
```


4. Namespace

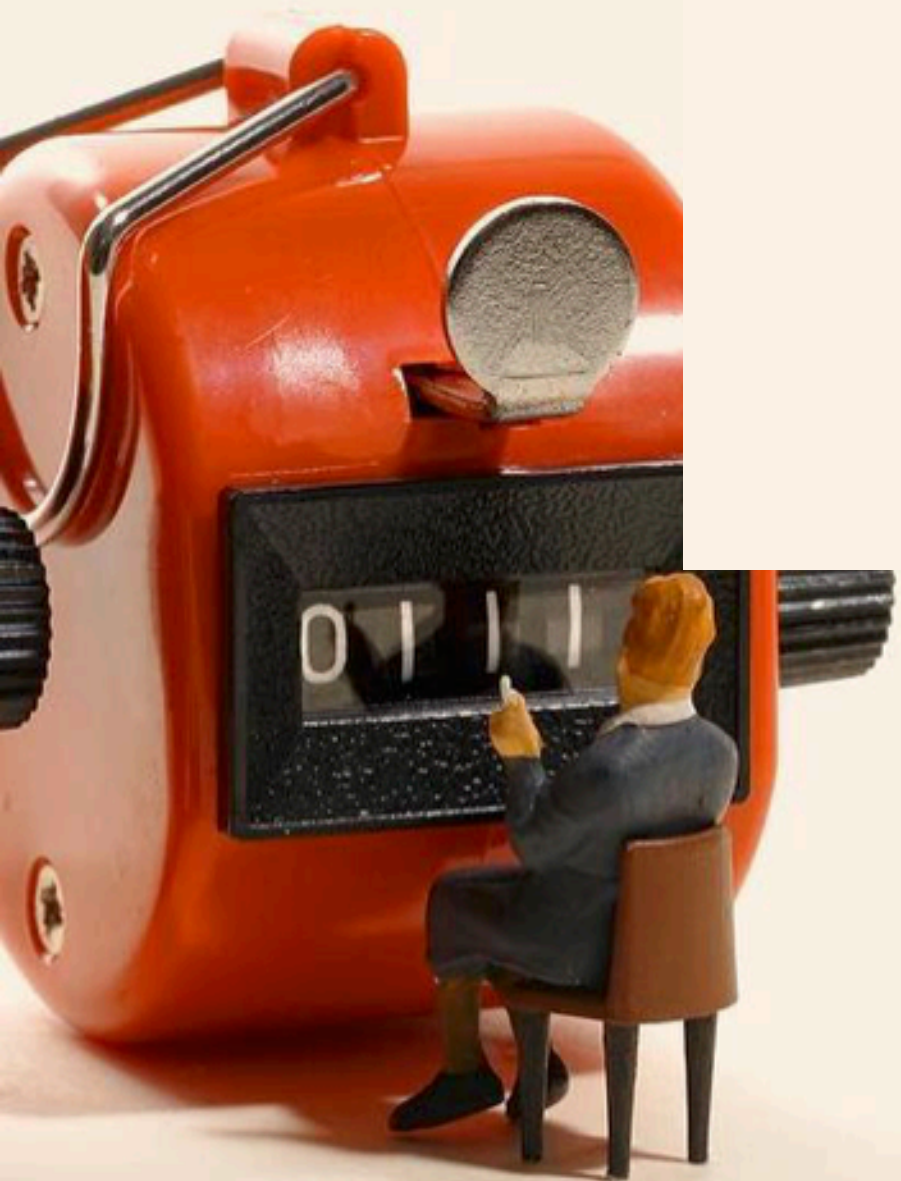
```
namespace Ex1
{
    class TipsExam
    {
        public static void Main()
        {
            Ex2.TipsExam data1 = new Ex2.TipsExam();
            data1.Test();
        }
    }
}
```

```
namespace Ex2
{
    class TipsExam
    {
        public void Test()
        {
            System.Console.WriteLine("Hello, World");
        }
    }
}
```

TipsExam 클래스가
1번 사람, 2번 사람 모두 같게 썼다.

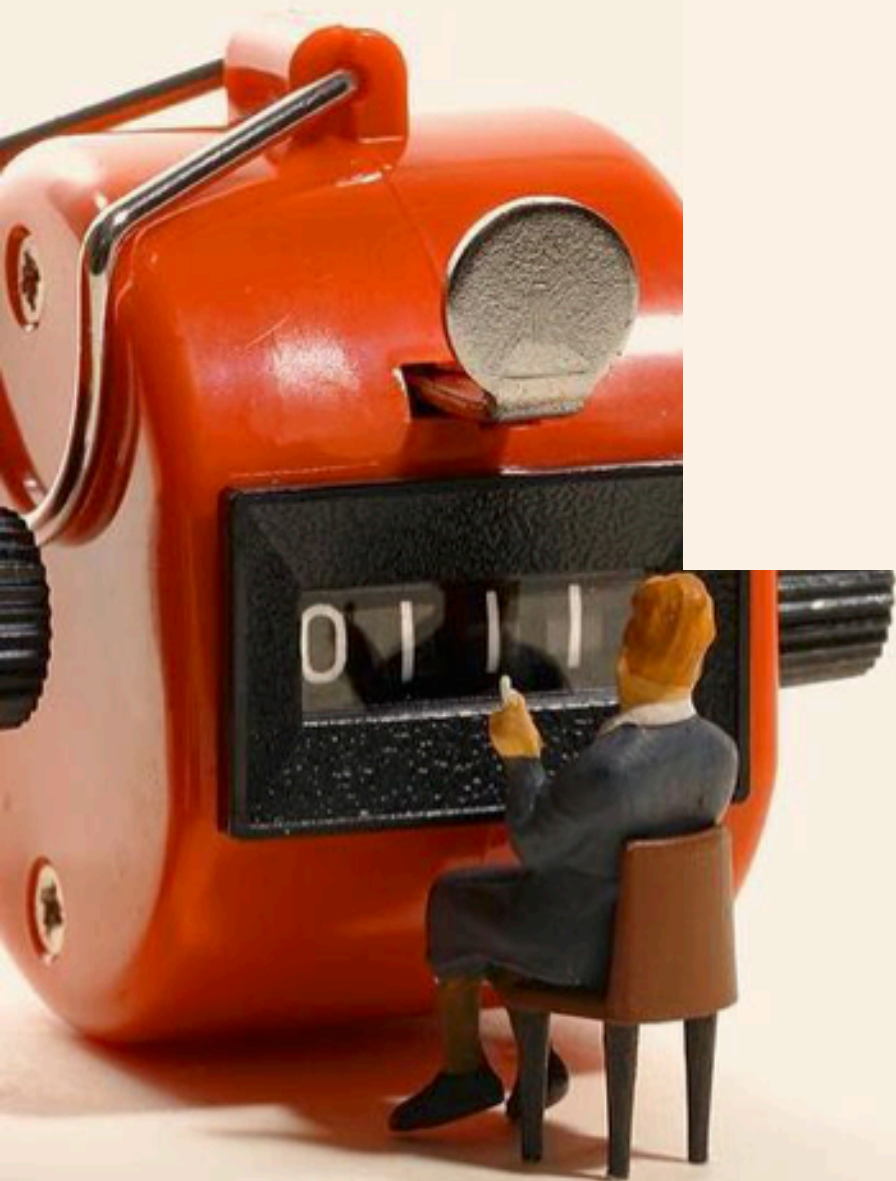
4. 기본 개념 설명

== vs equals



4. 기본 개념 설명

MVC 패턴



도서관리 프로그램



기능 명세

회원 관리

회원 등록(개인정보), 수정(주소/핸드폰 번호), 삭제, 검색, 출력(전체)

도서 관리

도서 등록, 수정(수량), 삭제, 검색(도서명/출판사/저자), 출력(전체)

도서 대여 및 반납

도서 대여 및 반납시간 추가

주의사항

- 클래스 및 UI 설계는 자유롭게
- 데이터가 많을수록 좋음
- 모든 예외 처리
- Git commit, push
- 과제는 반드시 완료
- 테스트