

컴퓨터정보과 C# 프로그래밍

7주차 이것저것

강의 순서

1. C# 환경설치 / C# 기본 구조
2. 클래스 기본(필드) + 변수, 자료형
3. 클래스 기본(메소드) + 연산자, 수식
4. 클래스 기본(메소드) + 제어문
5. 배열/리스트/딕셔너리
6. 클래스 기본: 접근제한자 (한정자) + 프로퍼티(속성)
7. 클래스 심화: 상속
8. 클래스 심화: 인터페이스/추상 클래스
9. 예외처리/일반화
10. 파일처리
11. UI (Winform or WPF)
12. LINQ/Delegate/Lambda/...

복습 및 6주차 추가요소

- 객체지향 네 가지 특징
 - 추상화
 - 상속
 - 다형성
 - 캡슐화
- 접근제한자
 - `private`
 - `public`
 - `protected`
 - `internal`
- Property (프로퍼티, 속성)
 - `getter, setter` 메소드의 새로운 C#의 문법
 - 생성은 메소드 , 사용은 변수
 - 자동 프로퍼티
- 프로세스의 메모리 구조
 - `Code`
 - `Data`
 - `Stack`
 - `Heap`

메모리~메모리~

메모리 구역 (추가 사항)

객체의 초기화 생명주기

- 정적 필드
 - 초기화 : 자동
 - 변수 선언 후 별도의 초기화 문장이 없어도 기본값으로 자동으로 초기화
 - 생명주기: Program 시작 ~ Program 종료
- 인스턴스 필드
 - 초기화 : 자동
 - 변수 선언 후 별도의 초기화 문장이 없어도 기본값으로 자동으로 초기화
 - 생명주기 : 객체 생성 ~ 객체 종료
- 지역 변수
 - 초기화 : 수동
 - 변수 선언 후 별도의 초기화 문장이 있어야 함
 - 지역(메소드) 시작 ~ 지역(메소드) 종료

생성자와 `this`

생성자와 this

- this : instance 자기 자신
- this() : instance 자기 자신의 생성자
- 예제

```
class Student
{
    private int age = 20;
    private string name;
    public Student(string name)
    {
        this.name = name;
    }

    public Student(string name, int age) : this(name)
    {
        this.age = age;
    }
}
```

```
Student s1 = new Student("김인하");
Student s2 = new Student("이인하", 21);
```

Project : MemoryTest2nd (SolWeek7)

```
class Test
{
    public static int sTest;    //초기화 하지 않으면?
    public int ITest;          //초기화 하지 않으면?
    public void LTest()
    {
        int localTest;        //초기화 하지 않으면?
        Console.WriteLine(localTest);
    }
}

internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(Test.sTest);

        Test t = new Test();
        Console.WriteLine(t.ITest);

        t.LTest();
    }
}
```

정적 필드는 언제 생성되는 가?
인스턴스 필드는 언제 생성되는 가?
지역 변수는 언제 생성되는가?

반대로 언제 사라지는 가?

class 생성 연습

그리고 DateTime

Project : Proj8

- Class : Bus
- Static Property
 - s_total (int) : public get/ private set
 - 현재 차고지에 있는 버스 대수
- Instance Property
 - Number (int) : public get/ private set
 - 버스 순번
 - InTime (DateTime) : public get/ private set
 - 버스가 차고지에 들어온 시간
 - OutTime (DateTime) : public get/ private set
 - 버스가 차고지에서 나간 시간
- Instance Method
 - SetInTime
 - 반환타입 : 없음
 - 매개변수 : 없음
 - 버스가 차고지에 들어올 때 실행한다. 현재 실행한 시간을 InTime에 넣는다. 그리고 s_total을 하나 증가한다.
- Instance Method
 - SetOutTime
 - 반환타입 : 없음
 - 매개변수 : 없음
 - 버스가 차고지에 나갈 때 실행한다. 현재 실행한 시간을 OutTime에 넣는다. 그리고 s_total을 하나 감소한다.
- Constructor
 - 매개변수 : 버스 순번 (int)
 - 버스 순번을 Number에 설정한다.
- 참고
 - DateTime.Now는 현재 시스템 시간을 가져오는 프로퍼티
 - DateTime 타입의 기본 값은 DateTime.MinValue (0001-01-01 00:00:00)

```

class Bus
{
    public static int s_total
    {
        get;
        private set;
    }

    public int Number
    {
        get;
        private set;
    }

    public DateTime InTime
    {
        get;
        private set;
    }

    public DateTime OutTime
    {
        get;
        private set;
    }
}

public Bus(int number)
{
    this.Number = number;
}

public void SetInTime()
{
    this.InTime = DateTime.Now;
    Bus.s_total++;
}

public void SetOutTime()
{
    this.OutTime = DateTime.Now;
    Bus.s_total--;
}
}

```

```

class Program
{
    static void Main(string[] args)
    {
        Bus[] buses = new Bus[3];
        for (int i = 0; i < buses.Length; i++) {
            buses[i] = new Bus(i);
        }

        Console.WriteLine("차고지 버스 대수:{0}대", Bus.s_total);

        buses[0].SetInTime();
        buses[1].SetInTime();
        buses[2].SetInTime();
        buses[2].SetOutTime();

        Console.WriteLine("차고지 버스 대수:{0}대", Bus.s_total);
    }
}

```

Project : Proj8

- Class : Bus
- Instance Property
 - IsExist (bool) : public get
 - 현재 차고지에 있는지 여부
차고지에 있으면 true, 없으면 false
(힌트 OutItem보다 InTime이 크다면 현재 차고지에 있음)

```
class Bus
{
    // 생략

    public bool IsExist
    {
        get {
            return this.InTime > this.OutTime;
        }
    }
}
```

```
class Program
```

```
{
```

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

```
    {
```

```
        Bus[] buses = new Bus[3];
```

```
        for (int i = 0; i < buses.Length; i++) {
```

```
            buses[i] = new Bus(i);
```

```
        }
```

```
        Console.WriteLine("차고지 버스 대수:{0}대", Bus.s_total);
```

```
        buses[0].SetInTime();
```

```
        Thread.Sleep(100);
```

```
        buses[1].SetInTime();
```

```
        Thread.Sleep(200);
```

```
        buses[2].SetInTime();
```

```
        Thread.Sleep(300);
```

```
        buses[2].SetOutTime();
```

```
        for(int i=0; i < buses.Length; i++) {
```

```
            if (buses[i] != null) {
```

```
                var result = buses[i].IsExist ? "있음" : "없음";
```

```
                Console.WriteLine($"{i}번: 차고지에 {result}");
```

```
            }
```

```
        }
```

```
        Console.WriteLine("차고지 버스 대수:{0}대", Bus.s_total);
```

```
    }
```

```
}
```

Project : Proj8

- Class : Bus
- Instance Property
 - ParkingTime (bool) : public get
 - 마지막 또는 현재 차고지에 주차되어 있는 시간을 반환.
 - 차고지에 있는 경우 현재 시간에서 들어온 시간을 뺀다.
차고지에 나온 경우는 최근 나온 시간에서 최근 들어온 시간을 뺀다.
 - (힌트) `DateTime - DateTime => TimeSpan`
 - 총 밀리초 : `double c = (a - b).TotalMilliseconds;`


```
class Bus
{
    // 생략

    public double ParkingTime
    {
        get {
            if (IsExist) {
                return (DateTime.Now - this.InTime).TotalMilliseconds;
            } else {
                return (OutTime - InTime).TotalMilliseconds;
            }
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        // 생략

        for (int i = 0; i < buses.Length; i++) {
            if (buses[i] != null) {
                var result = buses[i].IsExist ? "있음" : "없음";
                Console.WriteLine($"{i}번: 차고지에 {result}");
                var term = buses[i].ParkingTime;
                Console.WriteLine($"최근 차고지에 주차한 시간: {term:0.00} ms");
            }
        }

        Console.WriteLine("차고지 버스 대수:{0}대", Bus.s_total);
    }
}
```

DateTime & TimeSpan

날짜와 시간

DateTime 자료형

- Value Type (Struct로 생성)
 - 기본 값 DateTime.MinValue (0001-01-01 00:00:00.000)
- 생성의 다양한 형태
 - `DateTime dt1 = new DateTime(); //DateTime.MinValue;`
 - `DateTime dt2 = new DateTime(2024, 04, 17);`
 - `DateTime dt3 = new DateTime(2024, 04, 17, 9, 50, 10, 200); //yyyy, MM, dd, HH, mm, ss, fff`
 - `DateTime dt4 = DateTime.Now;`
- 주요 속성
 - `DateTime.MinValue` : DateTime의 최소값
 - `DateTime.Today` : 오늘 날짜
 - `DateTime.Now` : 현재 시스템 날짜/시간
 - `DateTime.Year / .Month / .Day / .Hour / .Minute / .Second`

DateTime 자료형

- 주요 인스턴스/정적 메소드
 - `DateTime newDate = dt1.AddDays(5);`
 - `.AddHours(), .AddMinutes()`
 - `TimeSpan diff = dt2.Subtract(dt1); //dt2 - dt1 ;`
 - `DateTime.Parse() / DateTime.TryParse() / DateTime.TryParseExact()`
 - `DateTime dt5 = DateTime.Parse("2024-04-17");`
 - `bool result1 = DateTime.TryParse("2024-04-17", out DateTime dt6);`
 - `bool result2 = DateTime.TryParseExact("2024-04-17", "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime dt7);`
 - `string str_datetime = dt7.ToString("yyyy-MM-dd HH:mm:ss.fff");`

TimeSpan 자료형

- 두 DateTime의 시간 간격을 나타내는데 사용
- 주요 속성
 - Days, Hours, Minutes, Seconds, Milliseconds
 - TotalDays, TotalHours, TotalMinutes, TotalSeconds, TotalMilliseconds
 - Ticks : **1/10,000,000초** (천만분의 일 초), 1tick == 100ns (nano seconds) , 1ms == 10000ticks
 - DateTime.MinValue를 나타내는 0001년 1월 1일 12:00:00 자정 이후 경과된 100ns 간격의 수를 나타냄.
- 주요 인스턴스 메소드
 - Add(TimeSpan), Subtract(TimeSpan)
 - CompareTo(TimeSpan)
 - Equals(TimeSpan)

은행

따라해보기

Level. 1

```
namespace Bank
```

```
{
```

```
    class Account
```

```
    {
```

```
    }
```

```
    class Bank
```

```
    {
```

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

```
        {
```

```
        }
```

```
    }
```

```
}
```


Level. 2

```
class Account
{
    public string  AccountNumber; //계좌번호
    public string  Owner;         //소유자
    public int     Balance;       //잔액
}
```

Level. 3

```
class Account
{
    public string    AccountNumber; //계좌번호
    public string    Owner;         //소유자
    public int       Balance;       //잔액

    public int Deposit(int amount) //입금
    {
        Balance += amount;
        return Balance;
    }

    public int Withdraw(int amount) // 출금
    {
        if (Balance >= amount)
            Balance -= amount;
        else
        {
            Console.WriteLine("[계좌번호] {0}", AccountNumber);
            Console.WriteLine("[잔고부족] {0}", Balance);
        }

        return Balance;
    }
}
```

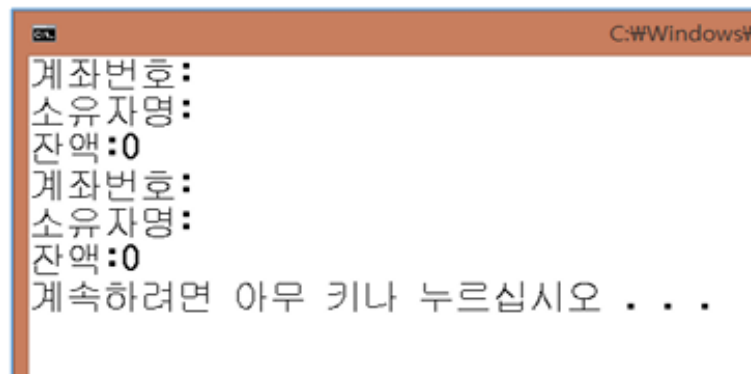
Level. 3

```
public void AccountInfo()    //계좌정보 출력
{
    Console.WriteLine("계좌번호:{0}", AccountNumber);
    Console.WriteLine("소유자명:{0}", Owner);
    Console.WriteLine("잔액:{0}", Balance);
}
```

Level. 4

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account();
        Account acc2 = new Account();

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```



C:\Windows

계좌번호:
소유자명:
잔액:0
계좌번호:
소유자명:
잔액:0
계속하려면 아무 키나 누르십시오 . . .

Level. 5

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account();
        Account acc2 = new Account();

        acc1.AccountNumber = "001-23456-02-1234";
        acc1.Owner          = "김인하";
        acc1.Balance         = 2000;

        acc2.AccountNumber = "001-23456-02-5678";
        acc2.Owner          = "김융합";
        acc2.Balance        = 12000;

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```



C:\W\Win

계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2000
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:12000
계속하려면 아무 키나 누르십시오 . .

Level. 6

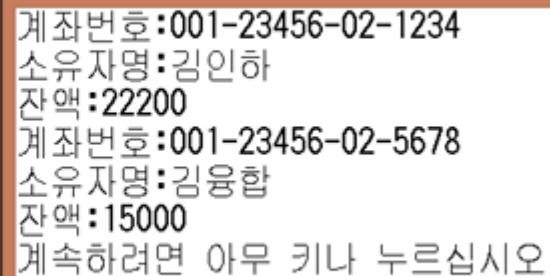
```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account();
        Account acc2 = new Account();

        acc1.AccountNumber = "001-23456-02-1234";
        acc1.Owner          = "김인하";
        acc1.Balance         = 2000;

        acc2.AccountNumber = "001-23456-02-5678";
        acc2.Owner          = "김융합";
        acc2.Balance        = 12000;

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```



계좌번호:001-23456-02-1234
소유자명:김인하
잔액:22200
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:15000
계속하려면 아무 키나 누르십시오

Level. 7

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account();
        Account acc2 = new Account();

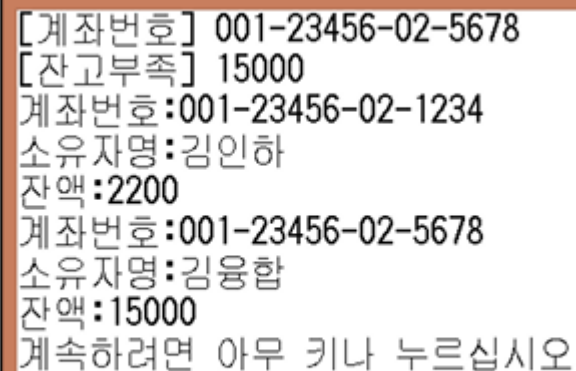
        acc1.AccountNumber = "001-23456-02-1234";
        acc1.Owner          = "김인하";
        acc1.Balance         = 2000;

        acc2.AccountNumber = "001-23456-02-5678";
        acc2.Owner          = "김융합";
        acc2.Balance         = 12000;

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.Withdraw(20000);
        acc2.Withdraw(20000);

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```



[계좌번호] 001-23456-02-5678
[잔고부족] 15000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:15000
계속하려면 아무 키나 누르십시오

Level. 8

```
Account acc1 = new Account(); //생성자
```

```
class Account
{
    public string  AccountNumber; //계좌번호
    public string  Owner;          //소유자
    public int     Balance;         //잔액

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}
    public void AccountInfo(){...}
}
```

```
class Account
{
    public string  AccountNumber; //계좌번호
    public string  Owner;          //소유자
    public int     Balance;         //잔액

    public Account() //기본 생성자
    {
    }

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}
    public void AccountInfo(){...}
}
```


Level. 8

```
class Account
{
    public string  AccountNumber; //계좌번호
    public string  Owner;         //소유자
    public int     Balance;       //잔액

    public Account() //기본 생성자
    {
    }

    public Account(string AccountNumber, string Owner, int Balance) //사용자 정의 생성자
    {
        this.AccountNumber = AccountNumber;
        this.Owner          = Owner;
        this.Balance        = Balance;
    }

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}
    public void AccountInfo(){...}
}
```

Level. 9

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account("001-23456-02-1234", "김인하", 2000);

        Account acc2 = new Account();

        acc1.AccountNumber = "001-23456-02-1234";
        acc1.Owner = "김인하";
        acc1.Balance = 2000;

        acc2.AccountNumber = "001-23456-02-5678";
        acc2.Owner = "김융합";
        acc2.Balance = 12000;

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.Withdraw(20000);
        acc2.Withdraw(20000);

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```

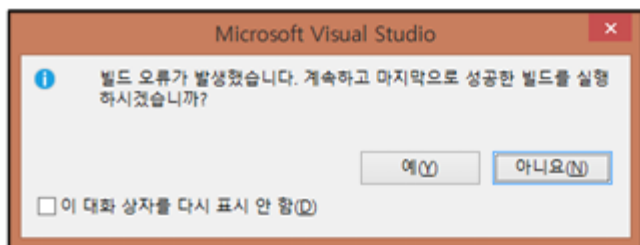
```
[계좌번호] 001-23456-02-5678
[잔고부족] 15000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:15000
계속하려면 아무 키나 누르십시오
```

Level. 10

```
class Account
{
    private string    AccountNumber; //계좌번호
    private string    Owner;         //소유자
    private int       Balance;       //잔액

    public Account(){...}
    public Account(string AccountNumber, string Owner, int Balance){...}

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}
    public void AccountInfo(){...}
}
```



- ❌ 2 보호 수준 때문에 'Bank.Account.Owner'에 액세스할 수 없습니다.
- ❌ 3 보호 수준 때문에 'Bank.Account.Balance'에 액세스할 수 없습니다.
- ❌ 1 보호 수준 때문에 'Bank.Account.AccountNumber'에 액세스할 수 없습니다.

Level. 10

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1
            = new Account( "001-23456-02-1234",
                           "김인하",
                           2000);
        Account acc2 = new Account();

        acc2.AccountNumber = "001-23456-02-5678";
        acc2.Owner          = "김용합";
        acc2.Balance         = 12000;

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.Withdraw(20000);
        acc2.Withdraw(20000);

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```

Level. 11

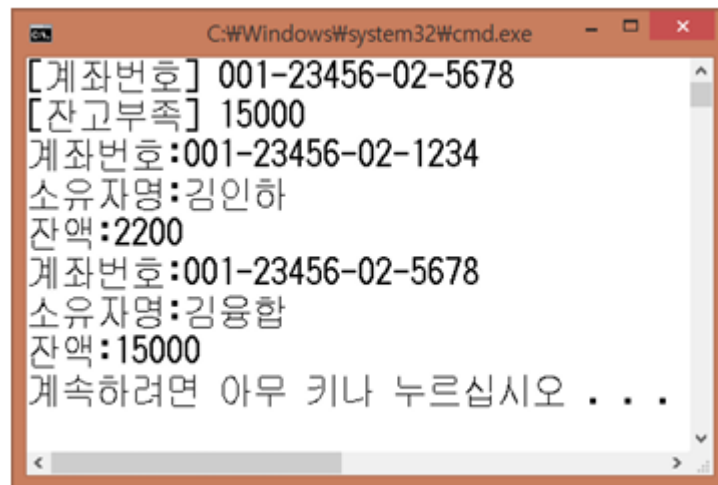
```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1
            = new Account( "001-23456-02-1234",
                           "김인하",
                           2000);

        Account acc2
            = new Account("001-23456-02-5678",
                           "김융합",
                           12000);

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.Withdraw(20000);
        acc2.Withdraw(20000);

        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```



```
C:\Windows\system32\cmd.exe
[계좌번호] 001-23456-02-5678
[잔고부족] 15000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:15000
계속하려면 아무 키나 누르십시오 . . .
```

Level. 12

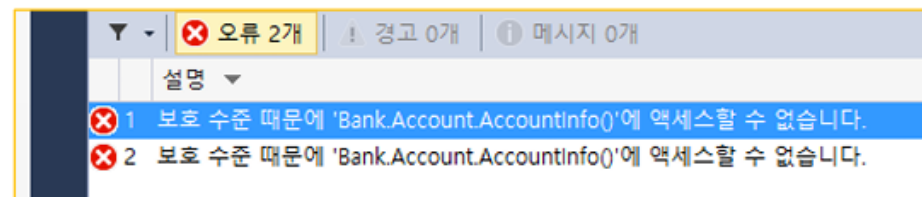
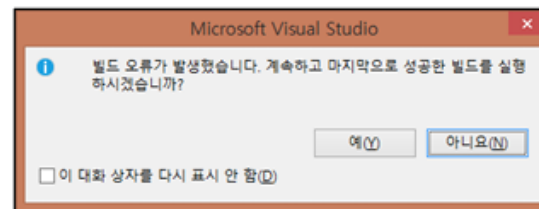
```

class Account
{
    private string  AccountNumber;  //계좌번호
    private string  Owner;           //소유자
    private int     Balance;          //잔액

    public Account(){...}
    public Account(string AccountNumber, string Owner, int Balance){...}

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}

    private void AccountInfo(){...}
}
  
```



Level. 12

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1
            = new Account( "001-23456-02-1234",
                           "김인하",
                           2000);

        Account acc2
            = new Account("001-23456-02-5678",
                           "김용합",
                           12000);

        acc1.Deposit(200);
        acc2.Deposit(3000);
        acc1.Deposit(20000);

        acc1.Withdraw(20000);
        acc2.Withdraw(20000);


        acc1.AccountInfo();
        acc2.AccountInfo();
    }
}
```

Level. 13

```
class Account
{
    private string    AccountNumber; //계좌번호
    private string    Owner;          //소유자
    private int       Balance;        //잔액

    public Account(){...}
    public Account(string AccountNumber, string Owner, int Balance){...}

    public int Deposit(int amount) //입금
    {
        Console.WriteLine("입금 처리=====");
        Console.WriteLine("입금액:{0}", amount);
        Balance += amount;
        this.AccountInfo();
        return Balance;
    }
}
```

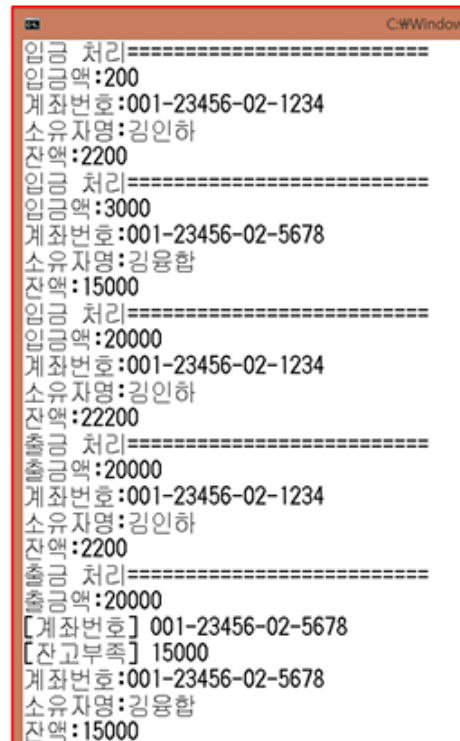


```
C:\Windows
입금 처리=====
입금액:200
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
입금 처리=====
입금액:3000
계좌번호:001-23456-02-5678
소유자명:김용함
잔액:15000
입금 처리=====
입금액:20000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:22200
출금 처리=====
출금액:20000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
출금 처리=====
출금액:20000
[계좌번호] 001-23456-02-5678
[잔고부족] 15000
계좌번호:001-23456-02-5678
소유자명:김용함
잔액:15000
```


Level. 13

```
public int Withdraw(int amount) // 출금
{
    Console.WriteLine("출금 처리=====");
    Console.WriteLine("출금액:{0}", amount);
    if (Balance >= amount)
        Balance -= amount;
    else
    {
        Console.WriteLine("[계좌번호] {0}", AccountNumber);
        Console.WriteLine("[잔고부족] {0}", Balance);
    }
    this.AccountInfo();
    return Balance;
}

private void AccountInfo(){...}
}
```



```
C:\Windows
입금 처리=====
입금액:200
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
입금 처리=====
입금액:3000
계좌번호:001-23456-02-5678
소유자명:김용함
잔액:15000
입금 처리=====
입금액:20000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:22200
출금 처리=====
출금액:20000
계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2200
출금 처리=====
출금액:20000
[계좌번호] 001-23456-02-5678
[잔고부족] 15000
계좌번호:001-23456-02-5678
소유자명:김용함
잔액:15000
```

Level. 14

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1
            = new Account( "001-23456-02-1234",
                           "김인하",
                           2000);

        Account acc2
            = new Account("001-23456-02-5678",
                           "김융합",
                           12000);

        Account acc3
            = new Account("001-23456-02-9012", "김씨샵");

        acc3.AccountInfo();
    }
}
```

// 계좌번호와, 이름만 있어도 계좌를 만들어 줄 수 있어야 한다.

// 그러기 위해서는 “계좌번호”, “이름”만 넣어도 생성할 수 있는 생성자를 추가해야 한다.

Level. 15

```
class Account
{
    private string  AccountNumber; //계좌번호
    private string  Owner;          //소유자
    private int     Balance;        //잔액

    public Account() //기본 생성자
    {
    }

    public Account(string AccountNumber, string Owner) //사용자 정의 생성자
    {
        this.AccountNumber = AccountNumber;
        this.Owner = Owner;
        //this.Balance = 0;
    }

    public Account(string AccountNumber, string Owner, int Balance) //사용자 정의 생성자
    {
        this.AccountNumber = AccountNumber;
        this.Owner = Owner;
        this.Balance = Balance;
    }

    public int Deposit(int amount){...}
    public int Withdraw(int amount){...}

    public void AccountInfo(){...}
}
```

계좌번호:001-23456-02-9012
소유자명:김씨샵
잔액:0
계속하려면 아무 키나 누르십시오

Level. 16

```
public Account(string AccountNumber, string Owner) //사용자 정의 생성자
{
    this.AccountNumber = AccountNumber;
    this.Owner = Owner;
    //this.Balance = 0;
}
```

```
public Account(string AccountNumber, string Owner, int Balance) //사용자 정의 생성자
{
    this.AccountNumber = AccountNumber;
    this.Owner = Owner;
    this.Balance = Balance;
}
```

```
public Account(string AccountNumber, string Owner)
{
    this.AccountNumber = AccountNumber;
    this.Owner = Owner;
}

public Account(string AccountNumber, string Owner, int Balance) : this(AccountNumber, Owner)
{
    this.Balance = Balance;
}
```

Level. 17

```
class Bank
{
    static void Main(string[] args)
    {
        Account acc1 = new Account( "001-23456-02-1234", "김인하", 2000);
        Account acc2 = new Account("001-23456-02-5678", "김융합", 12000);
        Account acc3 = new Account("001-23456-02-9012", "김씨삼");

        acc1.AccountInfo();
        acc2.AccountInfo();
        acc3.AccountInfo();
    }
}
```



계좌번호:001-23456-02-1234
소유자명:김인하
잔액:2000
계좌번호:001-23456-02-5678
소유자명:김융합
잔액:12000
계좌번호:001-23456-02-9012
소유자명:김씨삼
잔액:0
계속하려면 아무 키나 누르십시오

문제: Book 클래스 완성하기

■ 기본 제공 코드

```
class Book
{
    private string _title; //책 이름
    private DateTime _rentTime; //대여일
    private DateTime _returnTime; //반납일
    private string _name; //대여자
}
```

1. 책을 빌려줄 때 인스턴스를 생성하며, 생성시 정보는 아래와 같다.

빌리는 책 이름, 대여자, 빌리는 날짜를 설정한다.

2. 책을 반납할 때 호출하는 메소드를 생성한다.

메소드 이름: ReturnBook
반납일을 설정한다.
매개변수/반환타입 없음

3. 반납 여부를 알려주는 프로퍼티를 생성한다

프로퍼티 이름 : IsReturn
반환 타입 : bool

4. 모든 private 인스턴스 필드에 대한 public get property를 추가한다.

```
Book book = new Book("C# 프로그래밍", "김인하");
book.ReturnBook();
if(book.IsReturn)
    Console.WriteLine("대출중");
else
    Console.WriteLine("반납완료");
```

문제: Student 클래스 완성하기

- 기본 제공 코드

```
class Student
{
    private static int s_count; //현재 학생 총 명수
    private string _id; //학번
    private string _name; //이름
    private string _major; //전공
    private int _level; //학년
}
```

1. 학생 인스턴스를 생성시 아래 조건을 만족해야 한다.

학생의 이름/학년/전공을 입력 받는다.

현재 학생 총 명수를 증가시킨다.

학번은 S로 시작하고 s_Count를 8자리의 문자열로 변환해서 저장한다

예) string number = "S" + s_count.ToString("00000000");

2. 한 학생의 정보를 모두 출력할 수 있는 메소드를 생성한다.

메소드 이름: PrintStudentInfo

매개변수/반환타입 없음

예)

학년:1학년

학과:컴퓨터정보

학번:S00000001

이름:김인하

문제: Student 클래스 완성하기

3. 학년을 변경하는 메소드를 생성한다.

메소드 이름: ChangeGrade()
매개변수: 학년(int) / 반환타입 :bool
학년은 1~3학년 사이의 값을 가져야 하며, 그 외의 값이 들어오면 false를 반환한다.

4. 모든 인스턴스/정적 필드에 public get 프로퍼티를 생성한다.

5. 아래 인스턴스 필드에 대해서 public set 프로퍼티를 생성한다.
(개명, 전과로 인한 변경에 대비하기 위해서)

_name, _major

```
Student s = new Student("김인하",1, "컴퓨터정보");  
s.PrintStudentInfo();  
s.ChangeGrade(2);  
s.PrintStudentInfo();  
Console.WriteLine("총 인원수:{0}",Student.s_Count);  
s.SetMajor("정보통신");  
Console.WriteLine("{0}의 학과는{1} 입니다.",s.Name,s.Major);
```