컴퓨터정보과
C# 프로그래밍

2021년도/1학기/7주차
장은미

클래스 생성

- 하나의 파일에 여러 개의 클래스 생성 p.218
- 서로 다른 파일에 클래스 생성 p.220
  - 파일당 클래스 하나를 생성하는 것이 좋음

클래스의 변수 (p.225)

- 클래스 안에 정의하는 변수
  - 인스턴스 변수 p.255
    - [접근제한자] 자료형 이름
    - public int name;
    - 초기화 : 자동으로 기본 값으로 가능 (지역 변수처럼 선언과 동시에 초기화도 가능)
    - 사용방식 : (인스턴스-값).(변수이름)
    - 인스턴스가 생성되어야 존재하는 변수 (인스턴스 개수만큼 존재)
    - 메모리 위치 : HEAP
  - 클래스 변수 p.228
    - [접근제한자] static 자료형 이름
    - private **static** int name;
    - 초기화: 자동으로 기본 값으로 가능 (지역 변수처럼 선언과 동시에 초기화도 가능)
    - 사용방법 : (클래스이름).(변수이름)
    - 인스턴스 없어도 항상 존재하는 변수 (클래스 당 하나만 생성 - 전역변수처럼 사용 가능)
    - 메모리 위치 : DATA
- 메소드 안에 정의하는 변수
  - 지역변수
    - 자동으로 초기화 안됨, 반드시 프로그래머가 초기화 할 수 있도록 코드를 작성해야 함.
    - 메모리 위치 : STACK

```cs
class TestClass
{
    public int iAge;
    public static int sAge;

    public void ITestMethod()
    {
        int lAge = 10;
        Console.WriteLine(iAge); //this.iAge
        Console.WriteLine(sAge); //TestClass.sAge
        Console.WriteLine(lAge);
    }

    public static void STestMethod()
    {
        int lAge = 20;
        //Console.WriteLine(iAge);
        Console.WriteLine(sAge);
        Console.WriteLine(lAge);
    }
}
```

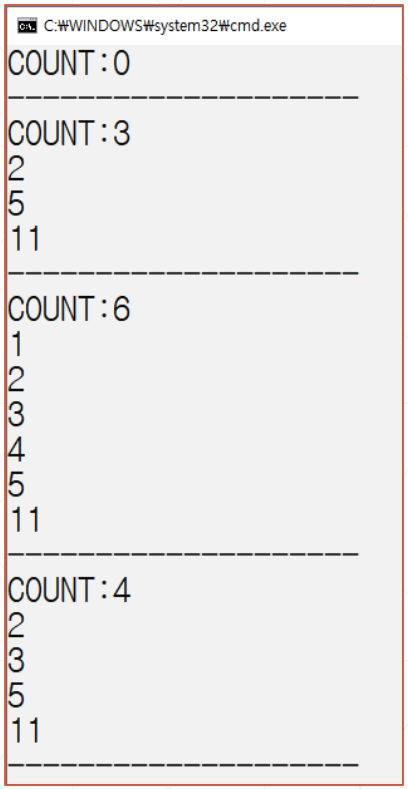
```cs
class Program
{
    static void Main(string[] args)
    {
        //Console.WriteLine(TestClass.iAge);
        Console.WriteLine(TestClass.sAge);

        TestClass t1 = new TestClass();
        TestClass t2 = new TestClass();

        Console.WriteLine("{0} / {1} ", t1.iAge, t2.iAge);
        //Console.WriteLine("{0} / {1} ", t1.sAge, t2.sAge);
        Console.WriteLine("{0} / {1} ", TestClass.sAge, TestClass.sAge);
        //Console.WriteLine("{0} / {1} ", TestClass.iAge, TestClass.iAge);

        TestClass.sAge = 100;
        t1.iAge = 200;
        t2.iAge = 300;

        Console.WriteLine("{2} / {0} / {1} ", t1.iAge, t2.iAge, TestClass.sAge);

        t1.ITestMethod();
        t2.ITestMethod();
        //t1.STestMethod();
        //t2.STestMethod();
        TestClass.STestMethod();
        //TestClass.ITestMethod();

    }
}
```

```
0
0 / 0
0 / 0
100 / 200 / 300
200
100
10
300
100
10
100
20
```

Class 활용 - List

- Random p.207
  - 인스턴스 생성 후 사용 가능 예) Random rand = new Random()
    - Next() : https://docs.microsoft.com/ko-kr/dotnet/api/system.random.next?view=net-5.0
    - NextDouble() : https://docs.microsoft.com/ko-kr/dotnet/api/system.random.nextdouble?view=net-5.0
- List<T> p.211
  - 대표적인 제네릭(일반화) 컬렉션
    - 배열 : 고정길이, 인덱스를 이용해 접근, 추가/삭제가 자유롭지 않음
    - List : 가변길이, 인덱스를 이용해 접근, 추가/삭제가 자유로움
  - 선언 형식 : List<T> name = new List<T>() ;
    - T는 List가 가질 수 있는 자료형 (을 알려주는 <>을 generic이라고 한다.)
    - Add(), Insert(), RemoveAt(), Remove()

```
static void Main(string[] args)
{
    List<int> numbers = new List<int>();

    Print(numbers);

    numbers.Add(2);
    numbers.Add(5);
    numbers.Add(11);
    Print(numbers);

    numbers.Insert(0, 1);
    numbers.Insert(2, 3);
    numbers.Insert(3, 4);
    Print(numbers);

    numbers.RemoveAt(0);
    numbers.Remove(4);
    Print(numbers);
}

static void Print(List<int> numbers)
{
    Console.WriteLine("COUNT:"+numbers.Count);
    for (int i=0; i < numbers.Count; i++)
    {
        Console.WriteLine(numbers[i]);
    }
    Console.WriteLine("--------------------");
}
```



메소드

- 기본 형태 p.265

```
[접근제한자] 반환형 메소드_이름([매개변수,…])
{
    [코드,…]
}
```

```
void test() {   }
int test() {  return 0; }
public int test() {  return 0; }
public void test(int a) {  return; }
public int test(int a, int b, double c) {  return 0; }
```

- 메소드 안에 정의하는 변수
  - 지역변수
    - 자동으로 초기화 안됨, 반드시 프로그래머가 초기화 할 수 있도록 코드를 작성해야 함.
    - 메모리 위치 : STACK

매개변수 , 반환형

- P.269
- 매개변수(parameter)
  - 메소드 호출 시, 전달하는 값을 메소드 내에서 저장하기 위한 공간
  - 매개변수는 0개 이상 정의 가능
  - 매개변수의 개수와 타입에 맞게, 호출하는 쪽에서 정확하게 값(인수, argument)을 전달해야함.
- 반환형
  - 메소드 실행 종료된 후, 호출자에게 넘겨주는 데이터의 타입
  - 반드시 타입 하나를 명시해야 함.
    - 넘겨줄 값이 없으면 void 표기

클래스 메소드

- P.272

```
class MATH
{
    public static int Add(int a, int b)
    {
        return a + b;
    }

    public int Sub(int a, int b)
    {
        return a - b;
    }
}

class Program
{
    static void Main(string[] args)
    {
        int a = 10;
        int b = 2;

        Console.WriteLine(MATH.Add(a, b));

        MATH m = new MATH();
        Console.WriteLine(m.Sub(a, b));
    }

}
```

메소드 오버로딩

- Console.WriteLine()
  - Console.WriteLine(1);
  - Console.WriteLine("1");
  - Console.WriteLine(1.1);
  - Console.WriteLine(true);

- 매개변수로 구별
  - 개수
  - 자료형

- 반환타입과 무관

```
class MATH
{
    public static int Add(int a, int b)
    {
        return a + b;
    }

    public static double Add(double a, double b)
    {
        return a + b;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(MATH.Add(1, 3));
        Console.WriteLine(MATH.Add(11.4, 3.2));
        Console.WriteLine(MATH.Add(11.4, 3));
    }
}
```

접근 제한자

- P.279
- [접근제한자] 자료형 변수이름
- [접근제한자] 반환형 메소드이름([매개변수]){ … }

- private
  - 기본 접근 제한자
  - 비공개 (내부에서만 접근 가능)
- public
  - 공개 (외부에서도 접근 가능)

생성자 & 종료자(소멸자)

- p.283 / p.288
- Constructor & Finalizer (Destructor)
- 규칙
  - 클래스 이름과 동일
  - 반환형 선언하지 않음

- this : 자기자신
- 기호상수 : p.291
  - const : 클래스 변수, 지역변수 (초기화 시점 : 선언시점)
  - readonly : 인스턴스 변수 (초기화 시점 : 선언시점, 생성자)

프로퍼티(Property, 속성)

- p.293

```
class Box
{
    public int width;
    public int height;

    public Box(int width, int height)
    {
        this.width = width;
        this.height = height;
    }

    public int Area()
    {
        return this.width + this.height;
    }
}
```

```
class Box
{
    private int _width;
    public int Width
    {
        get
        {
            return this._width;
        }
        set
        {
            if (value > 0) { this._width = value; }
            else { Console.WriteLine("0이상 입력"); }
        }
    }
    private int _height;
    public int Height
    {
        get
        {
            return this._height;
        }
        set
        {
            if (value > 0) { this._height = value; }
            else { Console.WriteLine("0이상 입력"); }
        }
    }

    public Box(int width, int height)
    {
        this.Width = width;
        this.Height = height;
    }

    public int Area()
    {
        return this.Width + this.Height;
    }
}
```

Bank - 0

- 프로젝트 이름 : Bank (Console 프로젝트)
- 은행프로그램
  - 통장정보
    - 계좌 번호
    - 계좌 명의자
    - 잔액
  - 통장 입금/출금/조회 기능

Bank – 1

```
 7    namespace Bank
 8    {
 9        class Account
10        {
11
12        }
13
14        class Program
15        {
16            static void Main(string[] args)
17            {
18            }
19        }
20    }
```

Bank – 2

```
 9    class Account
10    {
11        public string Number;   //계좌번호
12        public string Owner;    //명의자
13        public int Balance;     //잔액
14    }
```

Bank – 3

```
 9    class Account
10    {
11        public string Number;  //계좌번호
12        public string Owner;   //명의자
13        public int Balance;    //잔액
14
15        public bool Deposit(int amount)
16        {
17            throw new NotImplementedException();
18        }
19
20        public bool Withdraw(int amount)
21        {
22            throw new NotImplementedException();
23        }
24
25        public string AccountInfo()
26        {
27            throw new NotImplementedException();
28        }
29    }
```

입금 : 해야할 일은?

출금 : 해야할 일은?

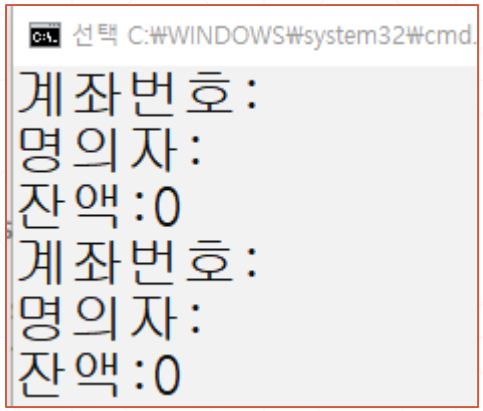
정보 : 해야할 일은?

Bank – 4

```
15      public bool Deposit(int amount)
16      {
17          this.Balance += amount;
18          return true;
19      }
20
21      public bool Withdraw(int amount)
22      {
23          if (this.Balance >= amount)
24          {
25              this.Balance -= amount;
26              return true;
27          }
28          else
29          {
30              return false;
31          }
32      }
33
34      public string AccountInfo()
35      {
36          StringBuilder builder = new StringBuilder();
37          builder.Append("계좌번호:").Append(this.Number).Append(Environment.NewLine);
38          builder.Append("명의자:").Append(this.Owner).Append(Environment.NewLine); ;
39          builder.Append("잔액:").Append(this.Balance);
40          return builder.ToString();
41      }
```

Bank – 5

```
44      class Program
45      {
46          static void Main(string[] args)
47          {
48              Account acc1 = new Account();
49              Account acc2 = new Account();
50
51              Console.WriteLine(acc1.AccountInfo());
52              Console.WriteLine(acc2.AccountInfo());
53          }
54      }
```



Bank - 6

```
46          static void Main(string[] args)
47          {
48              Account acc1 = new Account();
49              Account acc2 = new Account();
50
51              acc1.Number = "001-2345-06-789012";
52              acc1.Owner = "김인하";
53              acc1.Balance = 1000;
54
55              acc2.Number = "001-2345-06-234567";
56              acc2.Owner = "이인하";
57              acc2.Balance = 12000;
58
59              Console.WriteLine(acc1.AccountInfo());
60              Console.WriteLine(acc2.AccountInfo());
61          }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1000
계좌번호:001-2345-06-234567
명의자:이인하
잔액:12000
```

Bank - 7

```
46          static void Main(string[] args)
47          {
48              Account acc1 = new Account();
49              Account acc2 = new Account();
50
51              acc1.Number = "001-2345-06-789012";
52              acc1.Owner = "김인하";
53              acc1.Balance = 1000;
54
55              acc2.Number = "001-2345-06-234567";
56              acc2.Owner = "이인하";
57              acc2.Balance = 12000;
58
59              acc1.Deposit(200);
60              acc2.Deposit(3000);
61              acc1.Deposit(20000);
62
63              acc1.Withdraw(20000);
64              acc2.Withdraw(20000);
65
66              Console.WriteLine(acc1.AccountInfo());
67              Console.WriteLine(acc2.AccountInfo());
68          }
69      }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1200
계좌번호:001-2345-06-234567
명의자:이인하
잔액:15000
```

Bank – 8

```
 9    class Account
10    {
11        public string Number;  //계좌번호
12        public string Owner;   //명의자
13        public int Balance;    //잔액
14
15        public Account() { } //기본 생성자
16
17        public Account(string number, string owner, int balance)
18        {
19            this.Number = number;
20            this.Owner = owner;
21            this.Balance = balance;
22        }
23
24        public bool Deposit(int amount)...
29
30        public bool Withdraw(int amount)...
42
43        public string AccountInfo()...
51    }
```

Bank – 9

```
55          static void Main(string[] args)
56          {
57              Account acc1 = new Account("001-2345-06-789012", "김인하", 1000);
58              Account acc2 = new Account();
59
60              //acc1.Number = "001-2345-06-789012";
61              //acc1.Owner = "김인하";
62              //acc1.Balance = 1000;
63
64              acc2.Number = "001-2345-06-234567";
65              acc2.Owner = "이인하";
66              acc2.Balance = 12000;
67
68              acc1.Deposit(200);
69              acc2.Deposit(3000);
70              acc1.Deposit(20000);
71
72              acc1.Withdraw(20000);
73              acc2.Withdraw(20000);
74
75              Console.WriteLine(acc1.AccountInfo());
76              Console.WriteLine(acc2.AccountInfo());
77          }
78      }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1200
계좌번호:001-2345-06-234567
명의자:이인하
잔액:15000
```

(7과 동일)

Bank – 10

기본 생성자가 없을 경우
(테스트 후 다시 복구)

```
 9    class Account
10    {
11        public string Number;  //계좌번호
12        public string Owner;   //명의자
13        public int Balance;    //잔액
14
15        //public Account() { } //기본 생성자
16
17        public Account(string number, string owner, int balance)...
23
24        public bool Deposit(int amount)...
29
30        public bool Withdraw(int amount)...
42
43        public string AccountInfo()...
51    }
52
53    class Program
54    {
55        static void Main(string[] args)
56        {
57            Account acc1 = new Account("001-2345-06-789012", "김인하", 1000);
58            Account acc2 = new Account();
59
```

Bank – 11

```
 9    class Account
10    {
11        private string Number;  //계좌번호
12        private string Owner;   //명의자
13        private int Balance;    //잔액
14
15        //public Account() { } //기본 생성자
16
17        public Account(string number, string owner, int balance)...
23
24        public bool Deposit(int amount)...
29
30        public bool Withdraw(int amount)...
42
43        public string AccountInfo()...
51    }
```

```
53    class Program
54    {
55        static void Main(string[] args)
56        {
57            Account acc1 = new Account("001-2345-06
58            Account acc2 = new Account();
59
60            acc2.Number = "001-2345-06-234567";
61            acc2.Owner = "이인하";
62            acc2.Balance = 12000;
63
```

```
빌드 시작...
1>------ 빌드 시작: 프로젝트: Bank, 구성: Debug Any CPU ------
1>C:₩Users₩boong₩Google 드라이브₩[강의]2021-1₩컴정과₩수업내용₩7주차₩Bank₩Bank₩Program.cs(64,18,64,24): error CS0122: '보호 수준 때문에 'Account.Number'에 액세스할 수 없습니다.
1>C:₩Users₩boong₩Google 드라이브₩[강의]2021-1₩컴정과₩수업내용₩7주차₩Bank₩Bank₩Program.cs(65,18,65,23): error CS0122: '보호 수준 때문에 'Account.Owner'에 액세스할 수 없습니다.
1>C:₩Users₩boong₩Google 드라이브₩[강의]2021-1₩컴정과₩수업내용₩7주차₩Bank₩Bank₩Program.cs(66,18,66,25): error CS0122: '보호 수준 때문에 'Account.Balance'에 액세스할 수 없습니다.
========== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 ==========
```

Bank - 12

```
55          static void Main(string[] args)
56          {
57              Account acc1 = new Account("001-2345-06-789012", "김인하", 1000);
58              Account acc2 = new Account("001-2345-06-234567", "이인하", 12000);
59
60              acc1.Deposit(200);
61              acc2.Deposit(3000);
62              acc1.Deposit(20000);
63
64              acc1.Withdraw(20000);
65              acc2.Withdraw(20000);
66
67              Console.WriteLine(acc1.AccountInfo());
68              Console.WriteLine(acc2.AccountInfo());
69          }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1200
계좌번호:001-2345-06-234567
명의자:이인하
잔액:15000
```

(7,9와 동일)

Bank – 13

```
 9    class Account
10    {
11        private string Number;  //계좌번호
12        private string Owner;   //명의자
13        private int Balance;    //잔액
14
15        public Account() { } //기본 생성자
16
17        public Account(string number, string owner, int balance)
18        {
19            this.Number = number;
20            this.Owner = owner;
21            this.Balance = balance;
22        }
23
24        public Account(string number, string owner)
25        {
26            this.Number = number;
27            this.Owner = owner;
28            this.Balance = 0;
29        }
30
31        public bool Deposit(int amount)...
36
37        public bool Withdraw(int amount)...
49
50        public string AccountInfo()...
58    }
```

Bank – 14

```
62      static void Main(string[] args)
63      {
64          Account acc1 = new Account("001-2345-06-789012", "김인하", 1000);
65          Account acc2 = new Account("001-2345-06-234567", "이인하", 12000);
66          Account acc3 = new Account("002-0345-06-132539", "최인하");
67          Console.WriteLine(acc3.AccountInfo());
68
69          acc1.Deposit(200);
70          acc2.Deposit(3000);
71          acc1.Deposit(20000);
72
73          acc1.Withdraw(20000);
74          acc2.Withdraw(20000);
75
76          Console.WriteLine(acc1.AccountInfo());
77          Console.WriteLine(acc2.AccountInfo());
78      }
```

```
계좌번호:002-0345-06-132539
명의자:최인하
잔액:0
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1200
계좌번호:001-2345-06-234567
명의자:이인하
잔액:15000
```

Bank - 15

```
9   class Account
10  {
11      private string _number;  //계좌번호
12      private string _owner;   //명의자
13      private int _balance;    //잔액
14
15      public Account() { } //기본 생성자
16
17      public Account(string number, string owner, int balance)
18      {
19          this._number = number;
20          this._owner = owner;
21          this._balance = balance;
22      }
23
24      public Account(string number, string owner)
25      {
26          this._number = number;
27          this._owner = owner;
28          this._balance = 0;
29      }
30
31      public bool Deposit(int amount)
32      {
33          this._balance += amount;
34          return true;
35      }
36
37      public bool Withdraw(int amount)
38      {
39          if (this._balance >= amount)
40          {
41              this._balance -= amount;
42              return true;
43          }
44          else
45          {
46              return false;
47          }
48      }
49
50      public string AccountInfo()
51      {
52          StringBuilder builder = new StringBuilder();
53          builder.Append("계좌번호:").Append(this._number).Append(Environment.NewLine);
54          builder.Append("명의자:").Append(this._owner).Append(Environment.NewLine); ;
55          builder.Append("잔액:").Append(this._balance);
56          return builder.ToString();
57      }
58  }
```

Bank - 16

```
 9    class Account
10    {
11        private string _number;  //계좌번호
12        public string Number
13        {
14            get { return this._number; }
15        }
16
17        private string _owner;   //명의자
18        public string Owner
19        {
20            get { return this._owner; }
21        }
22
23        private int _balance;    //잔액
24        public int Balance
25        {
26            get { return this._balance; }
27        }
28
```

Bank - 17

```
112         static void Main(string[] args)
113         {
114             Account acc = new Account("001-2345-06-789012", "김인하", 1000);
115             Console.WriteLine(acc.AccountInfo());
116 
117             Console.Write("출금액:");
118             if(int.TryParse(Console.ReadLine(), out int money))
119             {
120                 if(acc.Balance >= money)
121                 {
122                     if (acc.Withdraw(money))
123                     {
124                         Console.WriteLine("출금완료");
125                         Console.WriteLine("잔액:{0}원", acc.Balance);
126                     }
127                     else
128                     {
129                         Console.WriteLine("잔액부족:{0}원", acc.Balance);
130                     }
131                 }
132                 else
133                 {
134                     Console.WriteLine("잔액부족:{0}원", acc.Balance);
135                 }
136             }
137 
138         }
139     }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1000
출금액:10
출금완료
잔액:990원
계속하려면 아무 키나 누르십시오 . . .
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:1000
출금액:12000
잔액부족:1000원
계속하려면 아무 키나 누르십시오 . . .
```

이 이후는 알면 좋으니 한 번 보세요.

Bank – 18

```
9   class TranscationInfo
10  {
11      DateTime _date;
12      public DateTime Date
13      {
14          get
15          {
16              return this._date;
17          }
18      }
19
20      uint _category; //0:deposit 1:withdraw
21      public uint Category
22      {
23          get
24          {
25              return this._category;
26          }
27      }
28      int _amount;
29      public int Amount
30      {
31          get
32          {
33              return this._amount;
34          }
35      }
36
37      public TranscationInfo(uint category, int amount)
38      {
39          this._date = DateTime.Now;
40          this._category = category;
41          this._amount = amount;
42      }
43  }
```

Bank - 19

```
45  class Account
46  {
47      private string _number;  //계좌번호
48      public string Number...
52
53      private string _owner;   //명의자
54      public string Owner...
58
59      private int _balance;    //잔액
60      public int Balance...
64
65      private List<TranscationInfo> transcationInfos = new List<TranscationInfo>();
66
67      public Account() { } //기본 생성자
68
```

```
83      public bool Deposit(int amount)
84      {
85          this._balance += amount;
86          transcationInfos.Add(new TranscationInfo(0, amount));
87          return true;
88      }
89
90      public bool Withdraw(int amount)
91      {
92          if (this._balance >= amount)
93          {
94              this._balance -= amount;
95              transcationInfos.Add(new TranscationInfo(1, amount));
96              return true;
97          }
98          else
99          {
100             return false;
101         }
102     }
```

Bank - 20

```
104         public string AccountInfo()
105         {
106             StringBuilder builder = new StringBuilder();
107             builder.Append("계좌번호:").Append(this._number).Append(Environment.NewLine);
108             builder.Append("명의자:").Append(this._owner).Append(Environment.NewLine);
109             builder.Append("잔액:").Append(this._balance).Append(Environment.NewLine);
110 
111             if (transcationInfos.Count > 0)
112             {
113                 builder.Append("최근 거래내역 (최대 10건)").Append(Environment.NewLine);
114                 int max = transcationInfos.Count > 10 ? transcationInfos.Count - 10 : 0;
115                 int j = 0;
116                 for (int i = transcationInfos.Count - 1 ; i >= max; i--)
117                 {
118                     builder.Append($"[{++j,2}] ");
119                     builder.Append($"{transcationInfos[i].Date:yyyy-MM-dd} ");
120                     builder.Append($"{(transcationInfos[i].Category == 0 ? "입금" : "출금")} ");
121                     builder.Append($"{transcationInfos[i].Amount,15}원");
122                     builder.Append(Environment.NewLine);
123                 }
124 
125                 builder.Append($"총 조회건수:{j}").Append(Environment.NewLine);
126             }
127 
128             return builder.ToString();
129         }
130     }
```

Bank – 21

```
132     class Program
133     {
134         static void Main(string[] args)
135         {
136             Account acc = new Account("001-2345-06-789012", "김인하", 1000);
137
138             acc.Withdraw(1000);
139             acc.Deposit(10);
140             acc.Deposit(200);
141             acc.Withdraw(300000);
142             acc.Withdraw(30);
143
144             Console.WriteLine(acc.AccountInfo());
145
146         }
147     }
```

```
계좌번호:001-2345-06-789012
명의자:김인하
잔액:180
최근 거래내역 (최대 10건)
[ 1] 2021-04-15 출금                30원
[ 2] 2021-04-15 입금               200원
[ 3] 2021-04-15 입금                10원
[ 4] 2021-04-15 출금              1000원
총 조회건수:4

계속하려면 아무 키나 누르십시오 . . .
```