컴퓨터정보과
C# 프로그래밍

13주차 가계부 연습

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/Lamda/…

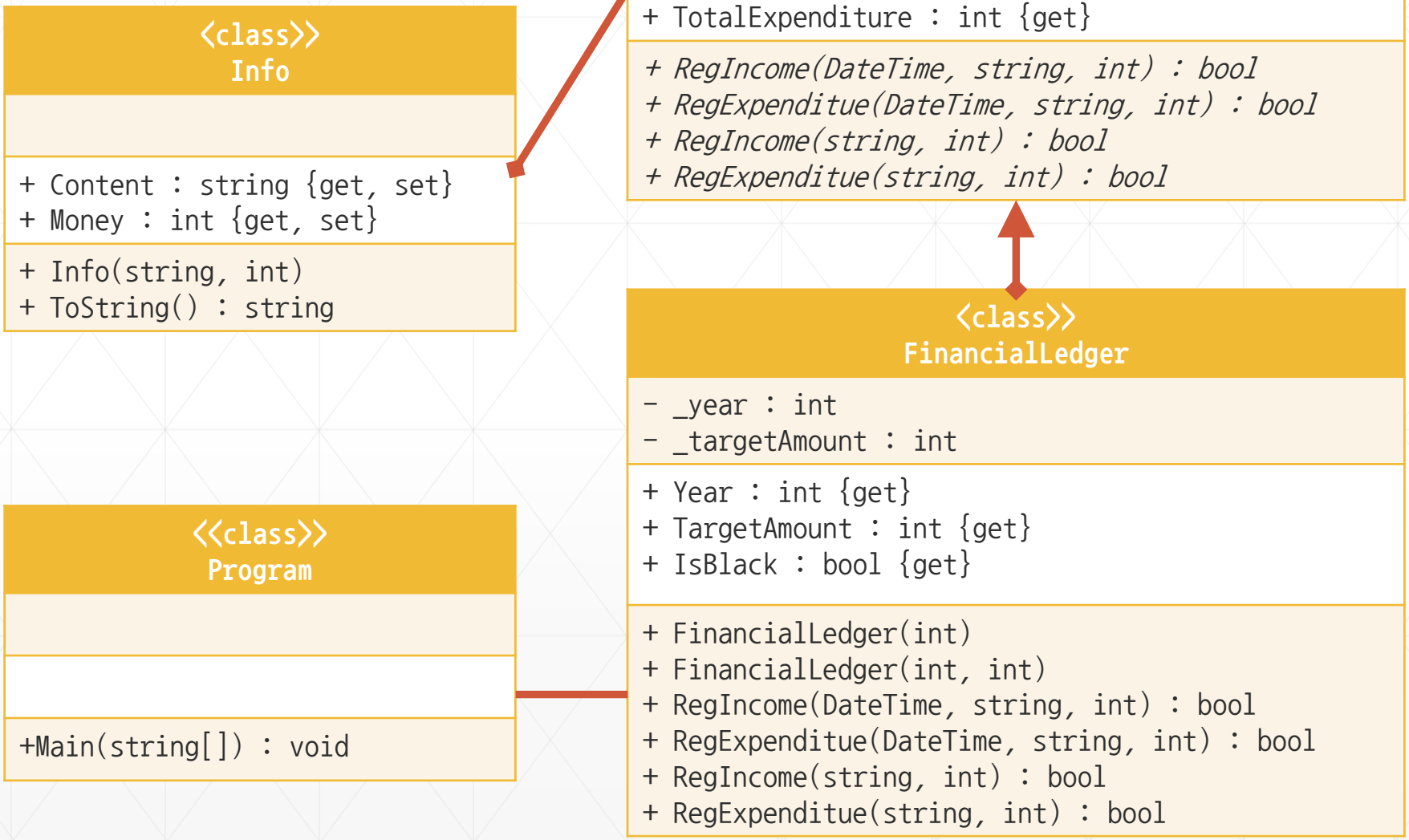
연습문제

가계부 (FinancialLeger)

기본 정보

- Solution Name : Sol_반_학번 (예:Sol_A_12345678)
- Project Name : Proj_FinacialLedger_학번 (예: Proj_FinacialLedger_12345678)
  - Template : Console App(.NET Framework)
- 내용 : 가계부
  - 수입, 지출을 날짜별로 기입하고 해당 정보를 통해 원하는 자료를 얻는다.
- 주의사항
  - 모든 클래스는 별도의 파일로 생성한다.

class diagram



<<class>> Info : 수입/지출 건당 정보

- Property
  - Content : 수입/지출 내역
  - Money : 수입/지출 금액
- Constructor
  - Constructor 1 : 수입/지출 내역과 금액을 받아서 초기화 한다.
- Method
  - ToString ()
    - override
    - 수입/지출 내역과 금액 사이에 ‘,’을 넣은 문자열을 생성하여 반환한다.

《class》 Finacial : 수입/지출내역 (1)

- Field
  - _incomes : 〈수입 발생 날짜시간, 수입 건당 정보〉 수입 정보 목록
  - _expenditures : 〈지출 발생 날짜시간, 수입 건당 정보〉 지출 정보 목록
- Property
  - TotalIncome : 해당 연도의 전체 수입액
  - TotalExpenditures : 해당 연도의 전체 지출액

<<class>> Finacial : 수입/지출내역 (2)

- Constructor
  - 없음
- Method
  - RegIncome 1: 수입 발생 날짜, 내역, 금액을 받아서 수입정보 목록에 설정한다.
    - 단 금액이 0이상이어야 등록 가능
  - RegIncome 2: 수출 내역, 금액을 받아서 실행 시점을 기준으로 수입정보 목록에 설정한다.
    - 단 금액이 0이상이어야 등록 가능
  - RegExpenditue 1: 지출 발생 날짜, 내역, 금액을 받아서 지출정보 목록에 설정한다.
    - 단 금액이 0이상이어야 등록 가능
  - RegExpenditue2: 지출 내역, 금액을 받아서 실행 시점을 기준으로 수입정보 목록에 설정한다.
    - 단 금액이 0이상이어야 등록 가능

<<class>> FinacialLedger : 가계부 (1)

- Field
  - _year : 연도
  - _targetAmount : 해당 연도 흑자 목표액
- Property
  - Year : _year 필드
  - TargetAmount : _targetAmount 필드
  - IsBlack : 흑자 여부
    - 목표 흑자액이 지정되지 않은 경우 전체 수입액이 전체 지출액보다 큰 경우
    - 목표 흑자액이 있는 경우 전체 수입액과 총 지출액의 차이가 해당 연도 흑자 목표액 이상인 경우
    - 위 두 가지를 제외 하고 모두 적자이다.

<<class>> FinacialLedger : 가계부 (2)

- Constructor
  - Constructor 1 : 연도를 전달받아 설정한다.
  - Constructor 2 : 연도와 흑자 목표액를 전달받아 설정한다.
- Method
  - RegIncome 1: 수입 발생 날짜, 내역, 금액을 받아서 수입정보 목록에 설정한다.
    - 단, 목표 연도와 다른 날짜정보가 들어오면 설정하지 않는다.
  - RegIncome 2: 수출 내역, 금액을 받아서 실행 시점을 기준으로 수입정보 목록에 설정한다.
    - 단, 목표 연도와 다른 날짜정보가 들어오면 설정하지 않는다.
  - RegExpenditue 1: 지출 발생 날짜, 내역, 금액을 받아서 지출정보 목록에 설정한다.
    - 단, 목표 연도와 다른 날짜정보가 들어오면 설정하지 않는다.
  - RegExpenditue2: 지출 내역, 금액을 받아서 실행 시점을 기준으로 수입정보 목록에 설정한다.
    - 단, 목표 연도와 다른 날짜정보가 들어오면 설정하지 않는다.

Program.cs 예제

```
internal class Program
{
    참조 0개
    static void Main(string[] args)
    {
        Dictionary<string, Dictionary<int, FinancialLedger>> ledgers
            = new Dictionary<string, Dictionary<int, FinancialLedger>>();

        ledgers["kim"] = new Dictionary<int, FinancialLedger>();
        ledgers["kim"][2024] = new FinancialLedger(2024, 10_000_000);
        ledgers["kim"][2023] = new FinancialLedger(2023);
        ledgers["lee"] = new Dictionary<int, FinancialLedger>();
        ledgers["lee"][2024] = new FinancialLedger(2024);

        ledgers["kim"][2024].RegIncome("월급", 3_000_000);
        ledgers["lee"][2024].RegIncome("월급", 3_000_000);
        ledgers["lee"][2024].RegExpenditue("과자", 30_000);

        foreach (var person in ledgers) {
            Console.WriteLine(person.Key);
            foreach (var ledger in person.Value) {
                Console.WriteLine($"Year: {ledger.Key}");
                Console.WriteLine($"Total IN:{ledger.Value.TotalIncome}");
                Console.WriteLine($"Total OUT:{ledger.Value.TotalExpenditure}");
                var isBlack = ledger.Value.IsBlack ? "YES" : "NO";
                Console.WriteLine($"BLACK:{isBlack}");
            }
            Console.WriteLine();
        }
    }
}
```

읽기전용 Dictionary 추가

```
참조 1개
internal class Financial
{
    protected Dictionary<DateTime, Info> _incomes = new Dictionary<DateTime, Info>();
    protected Dictionary<DateTime, Info> _expenditures = new Dictionary<DateTime, Info>();

    참조 1개
    public IReadOnlyDictionary<DateTime, Info> Incomes
    {
        get {
            return _incomes;
        }
    }

    참조 1개
    public IReadOnlyDictionary<DateTime, Info> Expenditures
    {
        get {
            return _expenditures;
        }
    }
```

Program.cs 변경

```
static void Main(string[] args)
{
    Dictionary<string, Dictionary<int, FinancialLedger>> ledgers
        = new Dictionary<string, Dictionary<int, FinancialLedger>>();

    if (false == ReadFile(ledgers)) {

        ledgers["kim"] = new Dictionary<int, FinancialLedger>();
        ledgers["kim"][2024] = new FinancialLedger(2024, 10_000_000);
        ledgers["kim"][2023] = new FinancialLedger(2023);
        ledgers["lee"] = new Dictionary<int, FinancialLedger>();
        ledgers["lee"][2024] = new FinancialLedger(2024);

        ledgers["kim"][2024].RegIncome("월급", 3_000_000);
        ledgers["lee"][2024].RegIncome("월급", 3_000_000);
        ledgers["lee"][2024].RegExpenditue("과자", 30_000);
    }

    foreach (var person in ledgers) {
        Console.WriteLine(person.Key);
        foreach (var ledger in person.Value) {
            Console.WriteLine($"Year: {ledger.Key}");
            Console.WriteLine($"Total IN:{ledger.Value.TotalIncome}");
            Console.WriteLine($"Total OUT:{ledger.Value.TotalExpenditure}");
            var isBlack = ledger.Value.IsBlack ? "YES" : "NO";
            Console.WriteLine($"BLACK:{isBlack}");
        }
        Console.WriteLine();
    }


    WriteFile(ledgers);
}
```

WriteFile() 메소드

```
private static void WriteFile(Dictionary<string, Dictionary<int, FinancialLedger>> ledgers)
{
    if (false == Directory.Exists(path)) {
        Directory.CreateDirectory(path);
    }

    if(ledgers == null) { return; }

    foreach (var person in ledgers) {
        var name = person.Key;
        foreach (var ledger in person.Value) {
            var year = ledger.Key;
            var filename1 = Path.Combine(path, $"{name}_{year}_in.txt");
            using(var sw = new StreamWriter(new FileStream(filename1, FileMode.Create))) {
                foreach (var data in ledger.Value.Incomes) {
                    var date = data.Key.ToString("yyyyMMd HHmmssfff");
                    var content = data.Value.ToString();
                    sw.WriteLine($"{date},{content}");

                }
            }

            var filename2 = Path.Combine(path, $"{name}_{year}_out.txt");
            using (var sw = new StreamWriter(new FileStream(filename2, FileMode.Create))) {
                foreach (var data in ledger.Value.Expenditures) {
                    var date = data.Key.ToString("yyyyMMdd HHmmssfff");
                    var content = data.Value.ToString();
                    sw.WriteLine($"{date},{content}");

                }
            }
        }
    }

    Console.WriteLine("파일쓰기");
}
```

```
20240529 091409687,월급,3000000
```