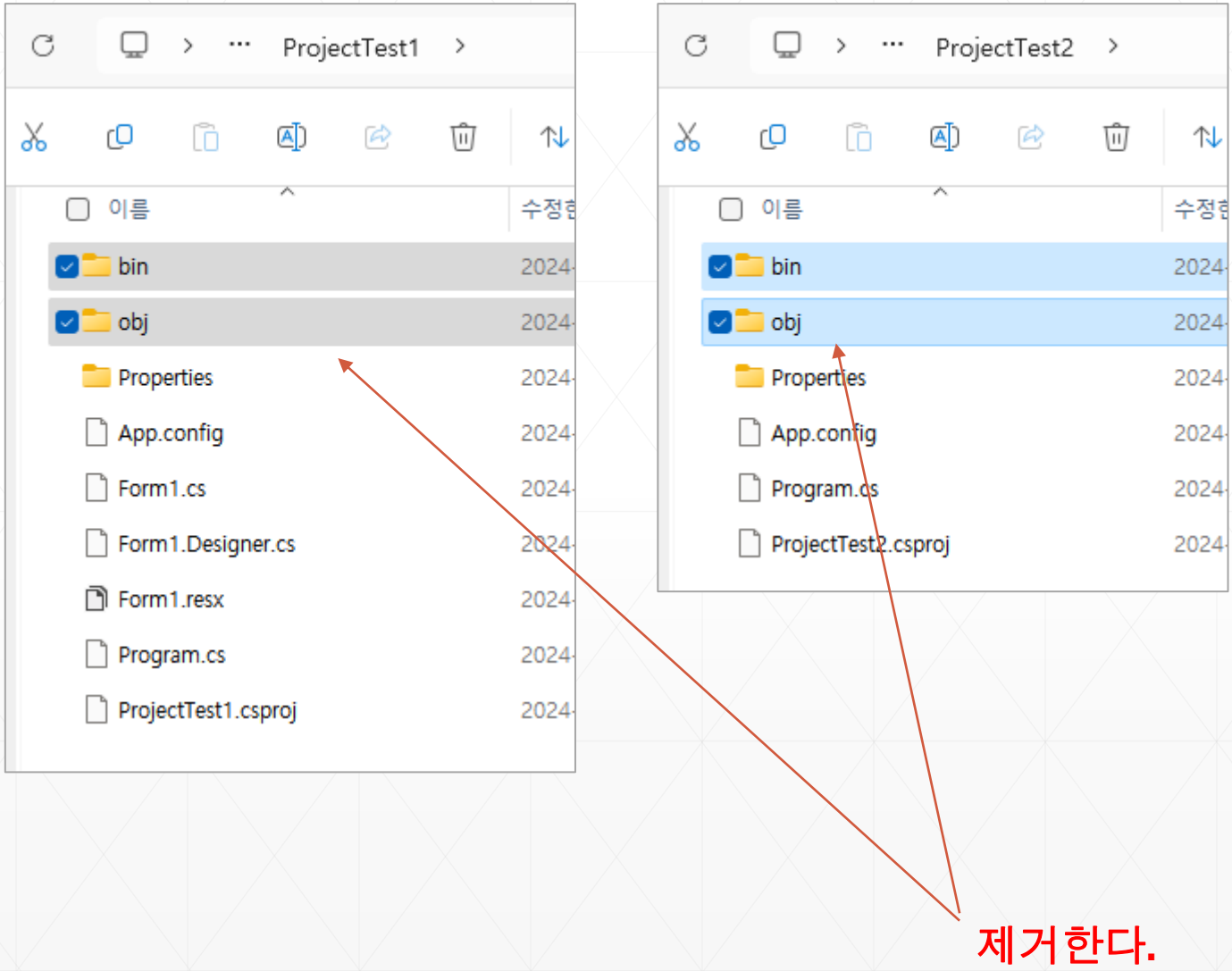
5주차 숙제

잘 읽고 제출해주세요.
수업 시간에 작성한 추가 페이지 있음

내용

- 뒤의 내용을 읽고 해당 사항을 수행해주세요.
- 바로 뒷 페이지에 나와있는 것과 같이 제출 직전에는 프로젝트 별로 생성되는 obj, bin 폴더를 모두 삭제한다.
- 그리고 앞 숙제와 마찬가지로 솔루션 폴더를 압축하여 제출한다.

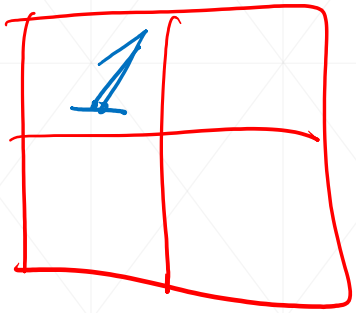
프로젝트에 있는 bin, obj는 제출 전 삭제



수업 시간에 진행한 배열비교

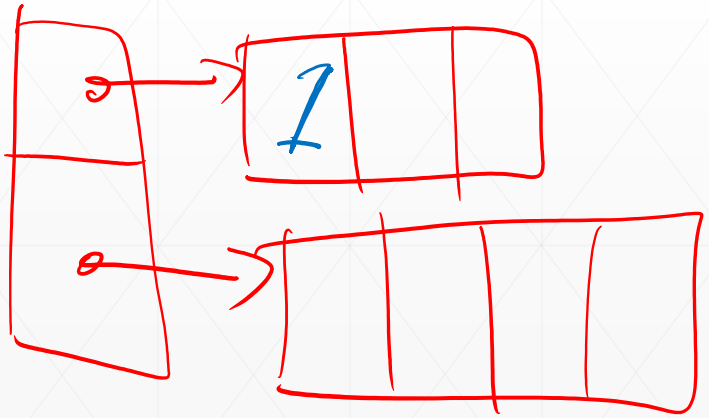
* 다차원 배열

```
int[,] a = new int[2,2];
a[0,0] = 1;
```

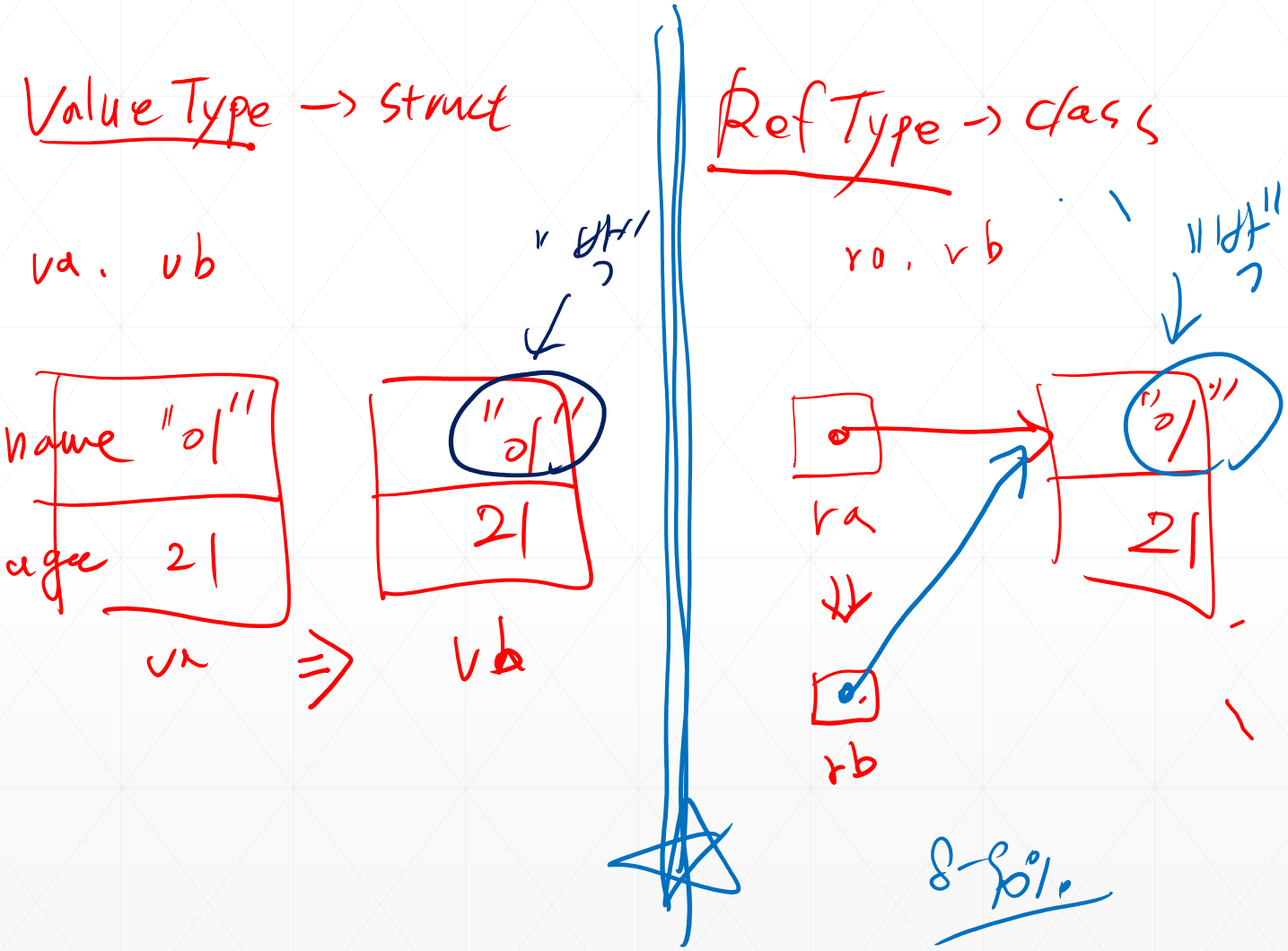


* 가변배열

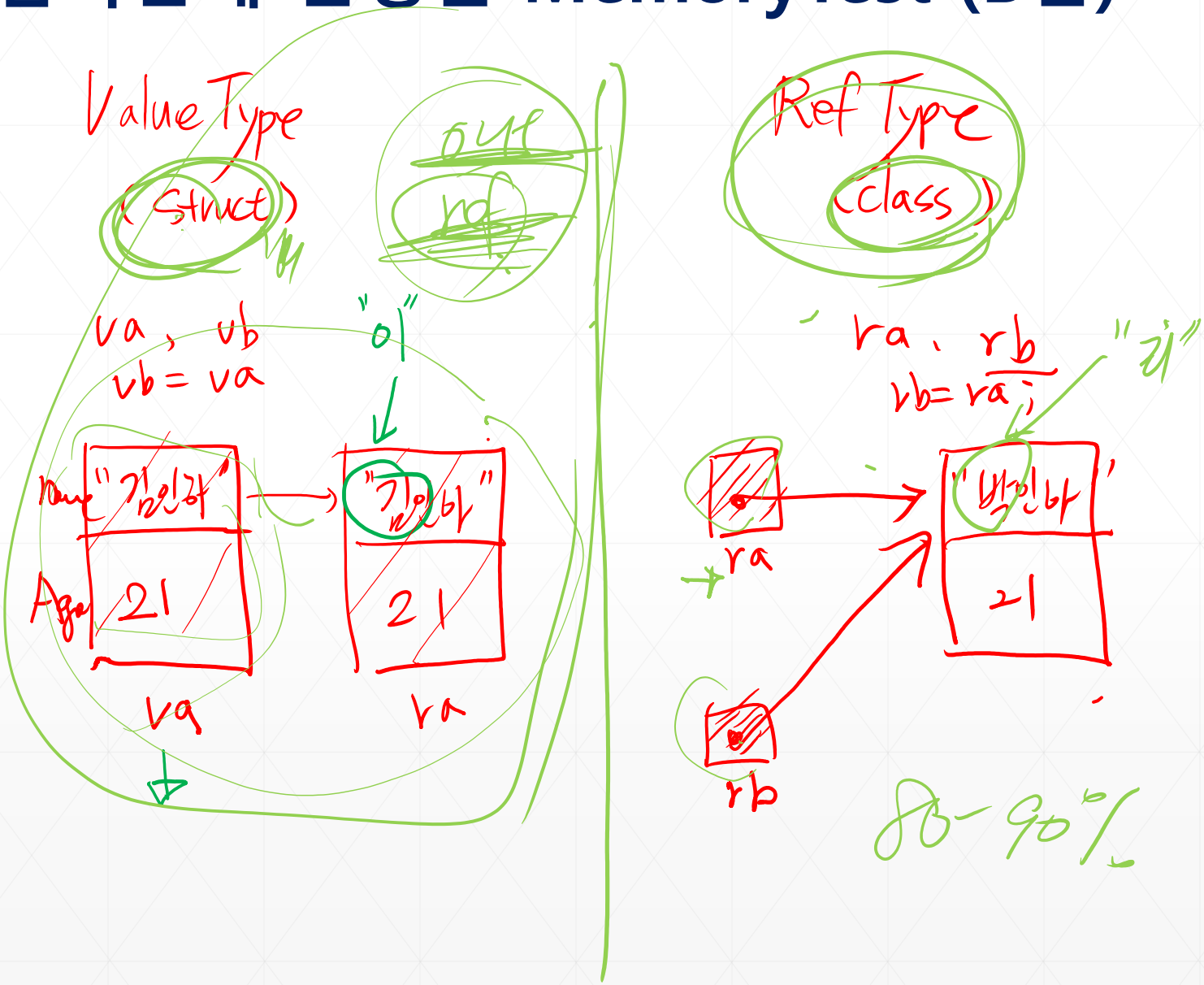
```
int[][] b = new int[2][];
b[0] = new int[3];
b[1] = new int[4];
b[0][0] = 1;
```



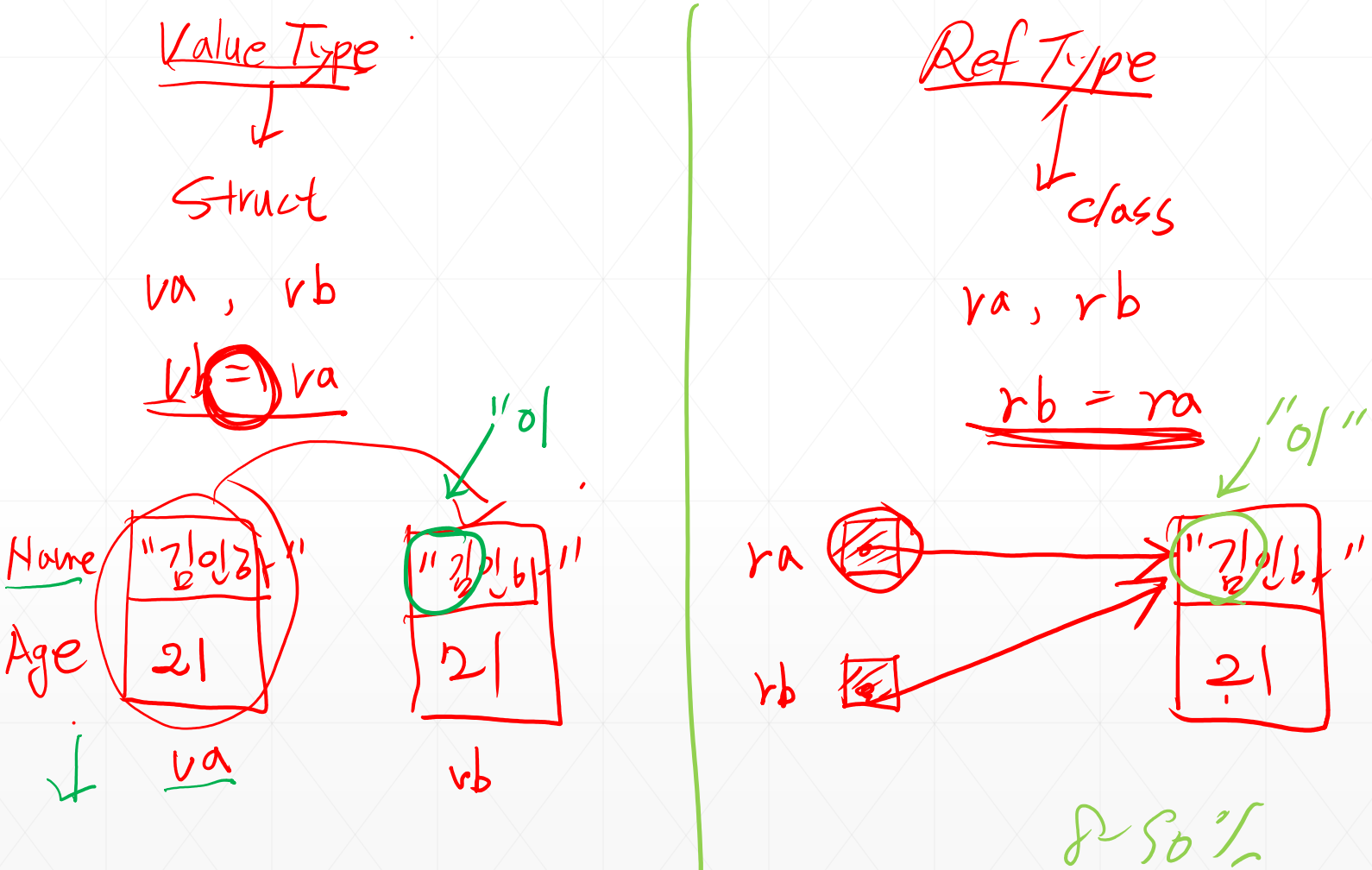
수업시간에 진행한 MemoryTest (A반)



수업시간에 진행한 MemoryTest (B반)



수업시간에 진행한 MemoryTest (C반)



CollectionExam

- JaggedArray()의 초기화 문장을 반복문으로 변경해보세요.

참고사항

- 뒤 숙제에 사용할 프로젝트의 기본 코드는 설명 이후 추가하였음.
  - 수업 자료를 볼 필요 없습니다.
- Proj 순서가 뒤섞인 것은 쉬운 순서대로 배치한 것입니다.

Proj6 (p.27~28)

- Program 클래스
  - Main() 메소드
    - 계좌 정보를 저장할 List를 생성한다.
      - 계좌 정보를 생성하면서 List에 추가한다.
      - index를 이용하여 각 계좌의 잔액을 증감한다.
    - for문을 통해서 생성한 List에 저장되어 있는 계좌 정보를 출력한다.
      - for문으로 변경하는 것을 연습해본다.

은행 계좌 관리

```
List<Account> accounts = new List<Account>();
accounts.Add(new Account("111 - 1111 - 1", "김인하"));
accounts.Add(new Account("111 - 1111 - 2", "김인하", 10000000));

accounts[0].AddBalance(10000);
accounts[1].SubBalance(100);

for(int i=0; i <accounts.Count; i++) {
    Console.WriteLine("{0}:{1}원", accounts[i].Number, accounts[i].Balance);
}
```

Account

```
class Account
{
    public string Number;
    public string Owner;
    public decimal Balance;

    public bool AddBalance(decimal money)
    {
        …
    }

    public bool SubBalance(decimal money)
    {
        …
    }

    public Account(string number, string owner)
        : this(number, owner, 0)
    {

    }

    public Account(string number, string owner
        , decimal balance)
    {
        Number = number;
        Owner = owner;
        Balance = balance;
    }
}
```

Proj3 (p.21~22)

- Program 클래스
  - Main() 메소드
    - List를 이용하여 Rect를 저장한다.
      - Rect를 생성하자마자 List에 추가한다.
      - Rect를 별도로 생성한 후에 List에 추가한다.
    - index를 이용하여 원하는 Rect인스턴스의 너비나 높이를 조절한다.
    - foreach를 통해 List의 요소(Rect 인스턴스)의 넓이(Area)를 출력한다.
      - for문으로 변경하는 것을 연습해본다.

면적 관리

```
List<Rect> rects = new List<Rect>();
rects.Add(new Rect(20.1, 30.5));

Rect rect2 = new Rect();
rect2.Width = 2;
rect2.Height = 3;
rects.Add(rect2);

rects[0].ChangeWidth(20.0);
rects[1].ChangeHeight(-3);

foreach(Rect rect in rects) {
    Console.WriteLine(rect.Area);
}
```

Rect

```
class Rect
{
    public double Width;
    public double Height;

    public double Area
    {
        get {
            return Width * Height;
        }
    }

    public bool ChangeWidth(double size)
    {
        if (Width + size < 0) {
            return false;
        }

        Width += size;
        return true;
    }

    public bool ChangeHeight(double size)
    {
        if (Height + size < 0) {
            return false;
        }
        Height += size;
        return true;
    }

    public Rect() { }

    public Rect(double width, double height)
    {
        Width = width;
        Height = height;
    }
}
```

Proj4 (p.23~24)

- Program 클래스
  - Main() 메소드
    - 회원 정보를 Dictionary에 저장한다.
      - key: 회원번호
      - value: Member 인스턴스
    - 회원번호는 중복되지 않도록 임으로 1,2,3,… 순으로 들어간다고 가정한다.
    - key(회원번호)를 통해서 해당 회원의 회원상태를 변경한다.
    - foreach를 통해 회원들의 회원 상태를 출력한다.
      - Key, Value를 어떻게 사용하는지 이해해본다.

회원 관리

```
Dictionary<int, Member> members = new Dictionary<int, Member>();
members[1] = new Member("김인하", 27, true);
members.Add(2, new Member("이인하", 22));

members[1].ChangeGrade();
members[2].ChangeGrade();

foreach (var member in members) {
    Console.WriteLine("등록번호:{0} {1}", member.Key, member.Value.Status);
}
```

Member

```
class Member
{
    public string Name;
    public int Age;
    public bool IsRegular;

    public string Status
    {
        get {
            string type = IsRegular ? "정회원" :
"준회원";
            return $"{Name} 회원은 {type} 입니다.";
        }
    }

    public void ChangeGrade()
    {
        IsRegular = !IsRegular;
    }
    public Member(string name, int age,
                  bool isRegualr = false)
    {
        Name = name;
        Age = age;
        IsRegular = isRegualr;
    }
}
```

Proj7 (p.29~30)

- Program 클래스
  - Main() 메소드
    - 직원 정보를 저장할 List를 생성한다.
    - 직원 정보 생성하자 마자 리스트에 추가하고, 추가한 후에 정보를 수정한다.
    - 직원정보를 별도의 변수를 통해 생성한 후, 정보를 수정한다. 그 이후에 리스트에 추가한다.
    - foreach를 통해서 직원의 정보를 출력한다.
      - emp.Salary:F0에서 F1, F2 으로 변경해 출력해본다.
      - for문으로 변경하는 것을 연습해 본다.

직원 관리

```
List<Employee> employees = new List<Employee>();
employees.Add(new Employee("김인하", "20240001"));
employees[0].Depart = "경영지원";
employees[0].Salary = 3_600_0000;

Employee emp2 = new Employee("이인하", "20200005");
emp2.Depart = "기술개발";
emp2.Salary = 4_5000_000;
employees.Add(emp2);

employees[0].ChangeSalary(5.6); //5.6% 상승

foreach(var emp in  employees) {
    Console.WriteLine($"{emp.Name} - {emp.Salary:F0}원");
}
```

Employee

```
class Employee
{
    public string Name;
    public string Number;
    public string Depart;
    public decimal Salary;

    public decimal MonthlySalary
    {
        get {
            return Salary / 12;
        }
    }

    public decimal ChangeSalary(double percent)
    {
        Salary *= (decimal)(1.0 + (percent / 100.0));
        return Salary;
    }

    public Employee(string name, string number)
    {
        Name = name;
        Number = number;
    }
}
```

Proj5 (p.25~26)

- Program 클래스
  - Main() 메소드
    - Team정보를 저장할 배열을 만든다.
      - 크기가 10인 이유는 현재 kbo의 구단 수가 10이기 때문이다.
    - 두 개 Team 정보를 만들어 배열에 저장한다.
    - for문을 통해서 배열에 저장되어 있는 팀의 정보를 출력한다.
      - for문을 통해서 출력 시 에러가 난다.
      - **왜 에러가 나며? 해결할 방안은 무엇이 있을 지 생각한다.**
        - 힌트: null
      - 위의 에러가 해결되면 foreach로 변경을 해본다.

경기 팀 관리

```
Team[] teams = new Team[10];
teams[0] = new Team("SSG", "인천");
teams[0].Coach = "이숭용";
teams[0].Level = 9;

teams[1] = new Team("삼성","박진만",3,"대구");

teams[0].IncreaseLevel(2);
teams[1].DecreaseLevel(2);

for(int i = 0; i < teams.Length; i++) {
    Console.WriteLine("{0}-{1}",teams[i].Name, teams[1].CurrentStatus);
    //왜 에러가 날까요?
    //처리할 수 있는 방법은?
}
```

Team

```
class Team
{
    public string Name;
    public string Coach;
    public int Level;
    public string Home;
    public static int LowerLevel = 10;

    public string CurrentStatus { … }

    public void IncreaseLevel(int value) { … }

    public void DecreaseLevel(int value) { … }
```

```
    public Team(string name, string home)
    {
        Name = name;
        Home = home;
    }

    public Team(string name, string coach,
       int level, string home) : this(name, home)
    {
        Coach = coach;
        Level = level;
    }
}
```

Proj1 (p.16~17)

- AddressBook 클래스
  - ToString() 메소드 추가하기
    - ToString()은 수업시간에 언급한 object의 메소드
      - AddressBook에서 ToString()을 다시 정의할 수 있음 (override 키워드 이용)
        - method overridng기법
    - StringBuilder 클래스
      - 문자열을 + 연산자로 계속 연결하면 쓸모없는 string instanc들이 메모리에 남게 됨.
        - 잦은 GC 동작을 유발
      - 이를 방지하기 위해 StringBuilder의 인스턴스를 생행 필요한 문자열을 넣은 후에 마지막에 문자열로 변경(ToString())함

- Program 클래스
  - Main() 메소드
    - 기존 만들어놓은 AddressBook 인스턴스를 배열에 넣고 이를 for문으로 출력
      - Console.WriteLine()는 문자열이 아닌 타입의 instance가 argument인 경우, 해당 instance의 ToString()을 호출해 문자열을 가져온 후에 Console에 출력한다.

주소록 관리 (AddressBook)

```
AddressBook addrBook1 = new AddressBook();
addrBook1.Name = "김인하";
addrBook1.Address = "인천 미추홀";
addrBook1.Phone = "010-1111-1111";
addrBook1.Group = "";

AddressBook addrBook2 = new AddressBook();
addrBook2.Name = "이인하";
addrBook2.Address = "인천 남미추홀";
addrBook2.Phone = "010-1111-1112";
addrBook2.Group = "친구";

AddressBook[] addressBooks = { addrBook1, addrBook2 };

for(int i=0; i < addressBooks.Length; i++) {
    Console.WriteLine(addressBooks[i]);
}
```

주소록 관리 (AddressBook)

```
class AddressBook
{
    // 기존 코드는 생략
    public override string ToString()
    {
        StringBuilder builder = new StringBuilder();

        builder.Append("이름:").Append(Name).Append(Environment.NewLine);
        builder.Append("주소:").Append(Address).Append(Environment.NewLine);
        builder.Append("전화:").Append(Phone).Append(Environment.NewLine);
        var group = string.IsNullOrEmpty(Group) ? "없음" : Group;
        builder.Append("그룹:").Append(group).Append(Environment.NewLine);

        return builder.ToString();
    }
}
```

Proj2 (p.18~20)

- Program 클래스
  - Main() 메소드
    - while문을 이용해서 3명의
      학번과 국어/영어/수학 점수를 입력 받아 Dictionary에 저장하는 코드
      - key: 학번
      - value : Score의 instance
      - 아래 조건을 만족하지 않으면 다시 입력하도록 한다.
        - 학번은 반드시 넣어야 하며(string.IsNullOrEmpty()) 반드시 다섯글자 이상이어야 함.
        - 학번은 중복되어서는 안된다.
        - 국어,영어,수학 점수는 모두 정수여야 한다.
    - foreach를 통해 Dictionary에 들어있는 key(학번)과 Score 인스턴스의 Average를 출력한다.

국어,영어,수학 점수 관리 (1)

```
Dictionary<string, Score> scores = new Dictionary<string, Score>();
int count = 0;
while(count < 3) {
    Console.Write("학번:");
    var number = Console.ReadLine();
    if(string.IsNullOrEmpty(number) || number.Length < 5) {
        continue;
    }

    if (scores.ContainsKey(number)) {
        continue;
    }

    Console.Write("국어:");
    if (false == int.TryParse(Console.ReadLine(), out int kor)) {
        continue;
    }
```

국어,영어,수학 점수 관리 (2)

```
    Console.Write("영어:");
    if (false == int.TryParse(Console.ReadLine(), out int eng)) {
        continue;
    }

    Console.Write("수학:");
    if (false == int.TryParse(Console.ReadLine(), out int mat)) {
        continue;
    }

    scores[number] = new Score(kor, eng, mat);

    count++;
}

foreach(var score in  scores) {
    Console.WriteLine("학번:{0} 평균:{1}", score.Key, score.Value.Average);
}
```

Score

```
class Score
{
    public int Kor;
    public int Eng;
    public int Mat;

    public int Average
    {
        get {
            return (Kor + Eng + Mat) / 3;
        }
    }

    public Score(int kor, int eng, int mat)
    {
        Kor = kor;
        Eng = eng;
        Mat = mat;
    }
}
```