

영역 3 : 메소드 만들어 사용하기

영역 3에서는 프로그램을 제작할 때에 필수 능력이라고 할 수 있는 메소드 제작을 연습한다. 프로그램에서 처리해야 하는 작업들을 내용과 성격에 따라 서브 메소드로 제작하여 역할을 분담하는 기술은 반드시 터득해야 하는 기술이다. 영역 3의 연습을 마치고 나면 영역 1과 영역 2에서 습득한 프로그래밍의 제어구조 기술을 메소드에 골고루 적용할 수 있게 될 것이다.

메소드는 별도로 작업해야 하는 프로그램 구문들을 따로 정해놓은 것이며, 메소드를 호출할 때에 전달해야 하는 데이터가 있으면 이를 파라미터로 전달할 수 있다. 그리고 메소드의 동작이 마친 후에 결과 값을 돌려받아야 하는 경우에는 리턴 값을 명시해서 돌려줄 수 있다. 물론 메소드에 전달할 데이터가 없는 경우에는 파라미터 없이 메소드를 제작하면 되며, 리턴할 결과 값이 없는 경우에는 리턴 값이 없다고 명시하면 된다.

영역 3의 실습 문제에서는 사용해야 하는 변수를 알려주지 않는다. 여러분이 문제를 충분히 읽고 분석한 후, 적절하게 필요한 변수들을 선언하여 사용하기 바란다.

영역 3는 다음과 같이 3개의 단계로 구성된다.

Step I : 파라미터 또는 리턴값만 있는 메소드 만들기

Step J : 파라미터와 리턴 값이 모두 있는 메소드 만들기

Step K : 파일 사용하기

[Step 1] 파라미터 또는 리턴값만 있는 메소드 만들기

이번 단계에서는 클래스에 가장 기본적인 메소드들을 만드는 연습을 하려고 한다. 메소드를 호출하게 되면 뭔가 정해진 작업을 수행하기만 하면 메소드의 역할이 끝나는 경우이다. 이런 경우에는 메소드의 리턴 값이 없다고 말한다.

리턴값이 없는 메소드를 만드는 방법은 다음과 같다.

```
void 메소드명 ( 입력변수 ) {
    // 처리해야할 명령어
}
```

리턴값이 있는 메소드를 만드는 방법은 다음과 같다.

```
리턴값타입 메소드명 ( 입력변수 ) {

    // 처리해야할 명령어

    return 변수;
}
```

예를 들어, 다음 예제와 같이 화면에 구분선을 출력하는 경우가 여러 번 있다고 가정한다면, 이 작업이 필요할 때마다 해당 구문을 일일이 프로그램에 넣기 보다는 메소드로 만들어 호출하게 하면 프로그램을 읽거나 만들 때에 좀 더 편리하다. 소스에서 보는 바와 같이 메인 메소드 위에 메소드의 내용을 구현하였다.

메소드를 만들고 사용하는 방법은 다음과 같다.

1. 메소드를 만든다.
2. 사용하고자 하는 클래스를 객체화하여 인스턴스변수를 만든다.
3. 인스턴스 변수로 메소드에 접근한다.

```
import java.util.Random;
class Test8
{
    // 메소드 시작
    void print_header() {
        System.out.println("=====");
        System.out.println("생성된 로또 번호");
    }
}
```

```

        System.out.println("=====");
    }
    // 메소드 끝

    public static void main(String[] args)
    {
        int lotto[] = new int[6];
        Random r = new Random();

        for(int i = 0; i < lotto.length; i++){
            lotto[i] = r.nextInt(45) + 1;
        }

        Test8 obj = new Test8();           // 객체화 하여 인스턴스 변수 obj 생성
        obj.print_header();                 // 메소드 호출

        for(int i = 0; i < lotto.length; i++){
            System.out.print(lotto[i] + " ");
        }
        System.out.println("");
    }
}

```

```

C:\Windows\system32\cmd.exe
=====
생성된 로또 번호
=====
1 2 11 44 18 20
계속하려면 아무 키나 누르십시오 . . .

```

이번에는 파라미터가 있고 리턴 값이 없는 메소드를 만들어보자. 위의 예제에서 `print_header()` 메소드는 테두리를 등호('=')로 출력하고 있는데, 이 메소드에 테두리의 모양문자를 지정하는 값을 파라미터로 주어서 출력 모양을 바꿔보자. 그러면 메소드의 내용은 다음과 같이 바뀌어야 한다.

```
void print_header(char ch);           // 변경된 메소드 원형 정의, 파라미터로 ch가 추가되었다.
```

이렇게 바뀐 메소드를 사용하도록 변경된 프로그램은 다음과 같다.

```

import java.util.Random;

class Test9
{

```

```

void print_header(char ch) {// 메소드 시작
    for(int i=0;i <17; i++){
        System.out.print(ch);
    }
    System.out.println();
    System.out.println("생성된 로또 번호");

    for(int i=0;i <17; i++){
        System.out.print(ch);
    }
    System.out.println();
}

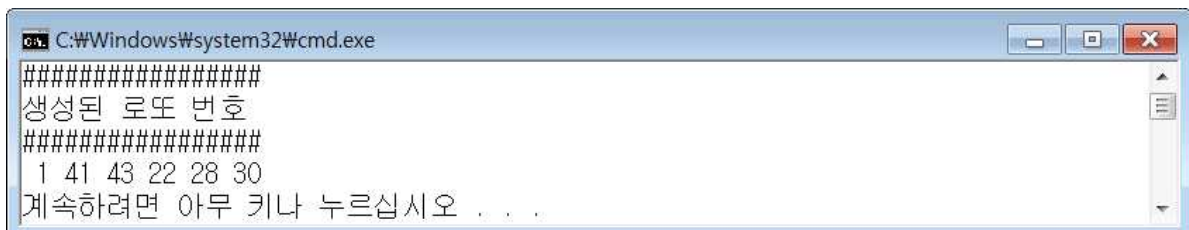
public static void main(String[] args)
{
    int lotto[] = new int[6];
    Random r = new Random();

    for(int i = 0; i <lotto.length; i++){
        lotto[i] = r.nextInt(45) + 1;
    }

    Test9 obj = new Test9();    // 원하는 객체 생성
    obj.print_header('#');      // 메소드 호출

    for(int i = 0; i <lotto.length; i++){
        System.out.print(lotto[i] + " ");
    }
    System.out.println("");
}
}

```



```

C:\Windows\system32\cmd.exe
#####
생성된 로또 번호
#####
1 41 43 22 28 30
계속하려면 아무 키나 누르십시오 . . .

```

이번에는 파라미터 없이 리턴 값만 돌려주는 메소드를 만들어보자. 예를 들어 입력받아야 하는 데이터가 바르게 입력되도록 점검해야 하는 일이 반복된다면 이 작업을 메소드로 제작해서 편리하게 불러 오면 된다. 성적을 처리하는 프로그램에서 점수를 입력받을 때에 반드시 0에서 100사이의 정

수를 입력받아야 하는데 이 범위에 맞는 수인지 검사하는 메소드를 만들어보자. 이 메소드를 사용하여 국어, 영어, 수학 점수를 입력받아 총점과 평균을 계산하는 프로그램은 다음과 같다.


```
import java.util.Random;
import java.util.Scanner;

class Test10
{
    int GetScore() { // 메소드 시작
        Scanner s = new Scanner(System.in);
        while(true){
            System.out.print("점수를 입력하시오. (0~100) :" );
            int num = s.nextInt();
            if(num >= 0 && num <= 100){
                return num;
            }
            System.out.println("잘못된 범위의 수입니다." );
        }
    }

    public static void main(String[] args)
    {
        String class_name[] = {"국어", "영어", "수학"};
        int score[] = {0,0,0};
        int sum = 0;

        Test10 ttt = new Test10();

        for(int i = 0; i < score.length; i++){
            System.out.print(class_name[i]);
            score[i] = ttt.GetScore();
            sum += score[i];
        }
        double average = sum / 3.0;
        System.out.println("3과목의 총점은 "+ sum + ", 평균은 "+ average + "입니다.");
    }
}
```



```
C:\Windows\system32\cmd.exe
국어 점수를 입력하십시오. (0~100) 120
잘못된 범위의 수입니다. 점수를 입력하십시오. (0~100) 95
영어 점수를 입력하십시오. (0~100) 85
수학 점수를 입력하십시오. (0~100) -50
잘못된 범위의 수입니다. 점수를 입력하십시오. (0~100) 70
3과목의 총점은 250, 평균은 83.3입니다.
계속하려면 아무 키나 누르십시오 . . .
```

○ 실습 문제

[I01] 메뉴판 보여주는 메소드 만들기

어떤 식당의 메뉴판을 보여주는 메소드를 만들어라. 메인 메소드에서 이 메소드를 호출하여 메뉴판을 보여준 다음 메뉴번호를 입력받아 해당 메뉴의 가격을 합산하되 메뉴선택 종료를 의미하는 5값을 입력 받을 때까지 계속 반복하여 메뉴를 선택하게 하고, 선택종료 후 선택한 메뉴의 총 합계금액을 출력하라.
단, 사용할 메뉴는 다음과 같다.

정의해야 할 메소드는 다음과 같다.

`void ShowMenu()` :

파라미터) 없음

리턴 값) 없음

수행내용) 메뉴판 출력

1. 피자(15,000원), 2. 스파게티(10,000원), 3. 샐러드(7,000원), 4. 음료수(2,000원), 5. 종료

```

C:\Windows\system32\cmd.exe
1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 1
현재까지의 주문 금액은 15000원입니다.

1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 1
현재까지의 주문 금액은 30000원입니다.

1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 2
현재까지의 주문 금액은 40000원입니다.

1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 3
현재까지의 주문 금액은 47000원입니다.

1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 4
현재까지의 주문 금액은 49000원입니다.

1. 피자(15,000원) 2. 스파게티(10,000원) 3. 샐러드(7,000원) 4. 음료수(2,000원)
메뉴를 선택해주세요.(종료 : 5) 5
현재까지의 주문 금액은 49000원입니다.

총 주문 금액은 49000원입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

[I02] 빈칸과 함께 특정 문자를 개수만큼 찍는 메소드 만들기

파라미터로 문자 하나와 숫자 두 개를 넘겨주면 한 줄에 첫 번째 숫자만큼 빈칸을 출력한 후, 바로 이어서 두 번째 숫자만큼 넘겨받은 문자를 화면에 출력하는 메소드를 만들어라. 그리고 사용자로부터 모양(문자 하나)과 높이, 여백을 입력받은 후, 이 메소드를 사용해서 입력한 크기만큼의 여백과 높이를 갖는 우직각 삼각형을 입력한 문자모양으로 화면에 출력시켜라.

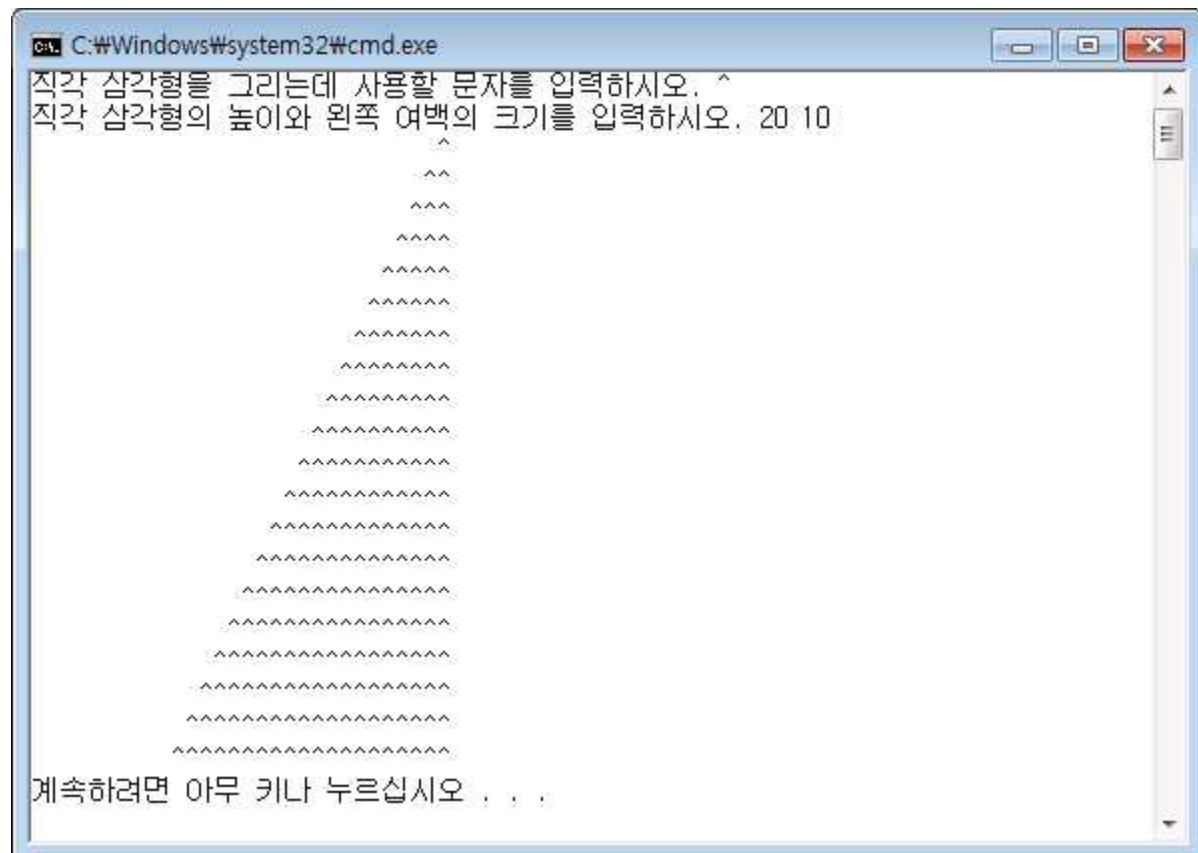
메소드 선언부는 다음과 같다.

```
void PrintCharWithBlank(int blanks, int size, char ch);
```

파라미터) blank : 빈칸의 개수, size : 출력할 문자의 개수, ch : 출력할 문자

리턴 값) 없음

수행내용) blanks 숫자만큼 빈칸 출력, size 개수만큼 ch 문자 출력 후 줄바꿈



[I03] 비만 판정

10명의 신장(cm단위)과 체중(kg단위)을 입력받으면서 AskBiman() 메소드를 통해 이들이 비만도를 출력하고 다음 기준에 따라 비만여부를 판정하여 출력하라.

비만도 수치 = 체중(kg) / (신장(m)의 제곱) 으로 계산한다. 이 때, 신장은 미터 단위로 환산해야 함을 유의하라.

비만도 수치에 따른 비만도 판정

1. 18.5 미만 : 저체중
2. 18.5 ~ 23 미만 : 정상체중
3. 23~25미만 : 과체중
4. 25~30미만 : 경도비만
5. 30이상 : 고도비만

메소드 선언부는 다음과 같다.

void AskBiman(int height, int weight) :

파라미터) height : 신장(cm), weight : 체중(kg)

리턴 값) 없음

수행내용) 비만도 계산 후 판정결과 출력

```

C:\Windows\system32\cmd.exe
1번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 176 80
경도 비만입니다.
2번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 182 99
경도 비만입니다.
3번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 168 54
정상입니다.
4번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 170 45
저체중입니다.
5번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 167 60
정상입니다.
6번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 176 98
고도 비만입니다.
7번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 152 60
경도 비만입니다.
8번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 172 80
경도 비만입니다.
9번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 167 70
경도 비만입니다.
10번째 사람의 신장(cm)과 체중(kg)을 입력하십시오. 168 75
경도 비만입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

[I04] 메뉴 번호 받아오는 메소드 만들기

어떤 식당의 메뉴판을 보여준 후에 메뉴번호를 입력받아 그 가격을 리턴하는 메소드를 만들어라. 메인 메소드에서 이 메소드를 호출하여 리턴 받은 가격을 합산하되 메뉴선택 종료로 의미하는 5를 리턴 받을 때까지 계속 반복하여 메뉴를 선택하게 하고, 선택종료 후 선택한 메뉴의 총 합계금액을 출력하라.

단, 사용할 메뉴는 다음과 같다.

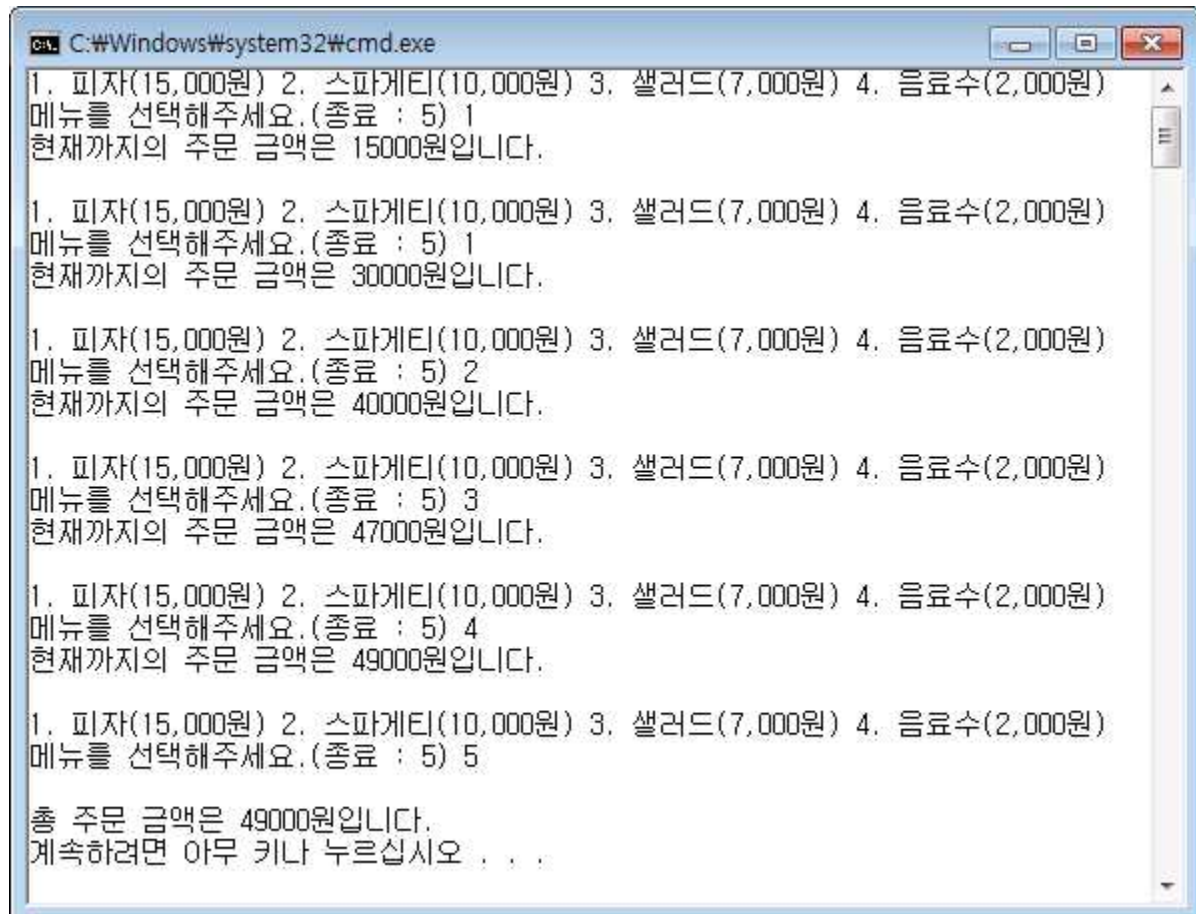
1. 피자(15,000원), 2. 스파게티(10,000원), 3. 샐러드(7,000원), 4. 음료수(2,000원), 5. 종료

메소드 선언부는 다음과 같다.

`int SelectMenu() ;`

파라미터) 없음

리턴 값) 1~4를 선택하면 선택한 메뉴의 가격, 5를 선택하면 -1



[I05] 최댓값 리턴하는 메소드 만들기

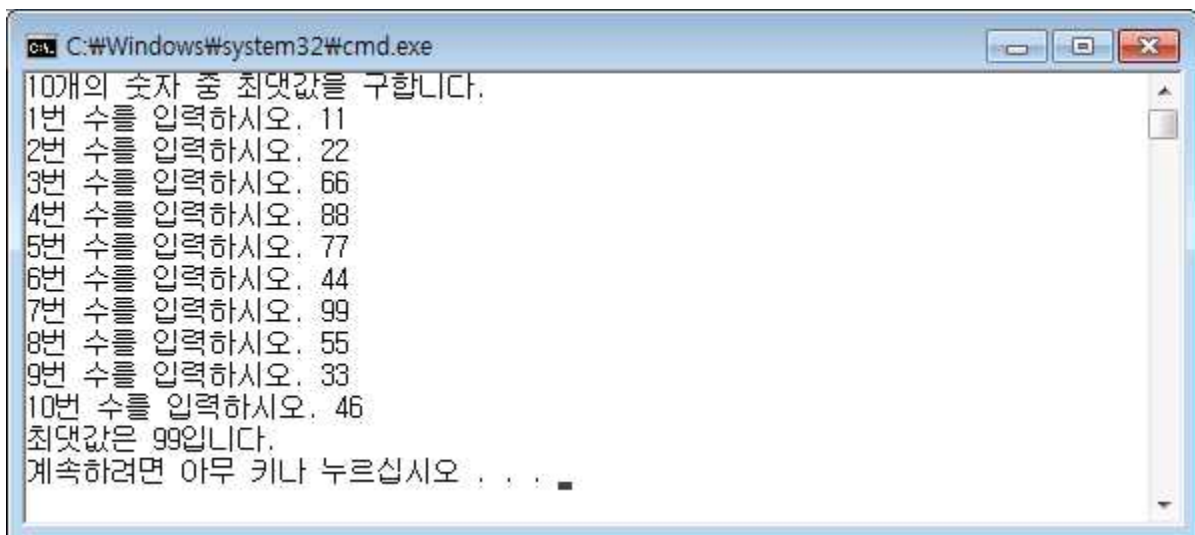
숫자 10개를 입력받아 이 중 최댓값을 찾아서 리턴하는 메소드 MaxOfTen()을 만들고, 이 메소드를 이용하여 10개의 숫자 중 최댓값을 출력하라.

메소드 선언부는 다음과 같다.

```
int MaxOfTen() ;
```

파라미터) 없음

리턴 값) 최댓값



```
C:\Windows\system32\cmd.exe
10개의 숫자 중 최댓값을 구합니다.
1번 수를 입력하시오. 11
2번 수를 입력하시오. 22
3번 수를 입력하시오. 66
4번 수를 입력하시오. 88
5번 수를 입력하시오. 77
6번 수를 입력하시오. 44
7번 수를 입력하시오. 99
8번 수를 입력하시오. 55
9번 수를 입력하시오. 33
10번 수를 입력하시오. 46
최댓값은 99입니다.
계속하려면 아무 키나 누르십시오 . . .
```

[I06] 임의의 숫자를 만들어 구간을 리턴하는 메소드 만들기

1부터 100사이의 임의의 숫자를 만들어서 대(70 이상), 중(40이상~70미만), 소(40미만) 셋 중에 하나를 판정하여 결과를 리턴하는 메소드 GetRandom()을 만들어라. 그리고 이 메소드를 이용해서 임의의 숫자 10개에 대해 대, 중, 소가 각각 몇 번씩 포함되어 있는지 개수를 출력하라.

메소드 선언부는 다음과 같다.

byte GetRandom();

파라미터) 없음

리턴 값) 임의로 만들어낸 숫자가 속하는 구간번호 0.대(70 이상), 1.중(40이상~70미만), 2.소(40미만)

```
C:\Windows\system32\cmd.exe
10개의 숫자를 생성합니다.
생성된 임의의 숫자는 50입니다.
생성된 임의의 숫자는 5입니다.
생성된 임의의 숫자는 87입니다.
생성된 임의의 숫자는 45입니다.
생성된 임의의 숫자는 13입니다.
생성된 임의의 숫자는 84입니다.
생성된 임의의 숫자는 57입니다.
생성된 임의의 숫자는 35입니다.
생성된 임의의 숫자는 87입니다.
생성된 임의의 숫자는 62입니다.

1. 대 (70 이상) : 3회 생성
2. 중 (40 이상) : 4회 생성
3. 소 (40 미만) : 3회 생성
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
10개의 숫자를 생성합니다.
생성된 임의의 숫자는 27입니다.
생성된 임의의 숫자는 98입니다.
생성된 임의의 숫자는 5입니다.
생성된 임의의 숫자는 54입니다.
생성된 임의의 숫자는 97입니다.
생성된 임의의 숫자는 1입니다.
생성된 임의의 숫자는 68입니다.
생성된 임의의 숫자는 30입니다.
생성된 임의의 숫자는 41입니다.
생성된 임의의 숫자는 94입니다.

1. 대 (70 이상) : 3회 생성
2. 중 (40 이상) : 3회 생성
3. 소 (40 미만) : 4회 생성
계속하려면 아무 키나 누르십시오 . . .
```

[107] 자판기에서 선택한 음료 가격을 리턴하는 메소드 만들기

자판기의 메뉴를 보여주고 선택하게 하여 선택된 음료의 가격을 리턴하는 메소드 `SelectCan()`을 만들어라. 그리고 이 메소드를 이용해서 자판기에서 음료를 반복해서 선택하게 하여 총 음료의 개수와 가격을 출력하라.

자판기의 음료 종류와 가격은 다음과 같다.

1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원) 4.홍차(500원) 5.코코아(600원)

메소드 선언부는 다음과 같다.

`int SelectCan() ;`

파라미터) 없음

리턴 값) 선택한 음료의 가격

```

C:\Windows\system32\cmd.exe
1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원)
4.홍차(500원) 5.코코아(600원)
메뉴를 선택해주세요 : 1
더 필요하십니까?(Y/N) Y
1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원)
4.홍차(500원) 5.코코아(600원)
메뉴를 선택해주세요 : 2
더 필요하십니까?(Y/N) Y
1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원)
4.홍차(500원) 5.코코아(600원)
메뉴를 선택해주세요 : 1
더 필요하십니까?(Y/N) Y
1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원)
4.홍차(500원) 5.코코아(600원)
메뉴를 선택해주세요 : 3
더 필요하십니까?(Y/N) Y
1.콜라(700원) 2.원두커피(300원) 3.레몬주스(1000원)
4.홍차(500원) 5.코코아(600원)
메뉴를 선택해주세요 : 4
더 필요하십니까?(Y/N) N
5개의 음료를 선택하여 총 3200원입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

[Step J] 파라미터와 리턴 값이 모두 있는 메소드 만들기

프로그램을 만들 때에 가장 일반적으로 사용하는 메소드의 형태는 파라미터와 리턴 값이 모두 있는 메소드이다. 메소드를 호출할 때에 메소드 실행에 필요한 값들을 파라미터로 전달하고, 실행이 끝나면 그 결과로 얻어내야 하는 값을 리턴받는 것이다. 이번 단계에서는 이런 형식의 메소드를 제작하는 연습을 하도록 한다.

앞의 [Step I]에서 만들어 본 **GetScore()** 메소드를 어떤 과목의 점수를 입력받아야 하는 지를 명시해서 호출하도록 변형해보도록 하자. 즉, 파라미터로 과목의 이름을 전달하는 것이다. 이를 위해서는 메소드의 선언부를 다음과 같이 바꾸어야 한다. 진하게 표시한 부분이 이전 소스에서 변경된 부분이다.

```
import java.util.Scanner;
class Test_J1
{
    int GetScore(String name) { // 파라미터로 과목명(name)을 전달받는다.
        Scanner s = new Scanner(System.in);
        while(true){
            System.out.print(name + " 점수를 입력하시오. (0~100) :" );
            int num = s.nextInt();
            if(num >= 0 && num <= 100){
                return num;
            }
            System.out.print("잘못된 범위의 수입입니다." );
        }
    }

    public static void main(String[] args)
    {
        String class_name[] = {"국어", "영어", "수학"};
        int score[] = {0,0,0};
        int sum = 0;

        Test_J1 obj = new Test_J1();

        for(int i = 0; i < score.length; i++){
            score[i] = obj.GetScore(class_name[i]);
            sum += score[i];
        }

        double average = sum / 3.0;
```

```

        System.out.println("3과목의 총점은 "+ sum + ", 평균은 "+ average + "입니다.");
    }
}

```

```

C:\Windows\system32\cmd.exe
국어 점수를 입력하시오. (0~100) 120
잘못된 범위의 수입니다. 국어 점수를 입력하시오. (0~100) 95
영어 점수를 입력하시오. (0~100) 85
수학 점수를 입력하시오. (0~100) 70
잘못된 범위의 수입니다. 수학 점수를 입력하시오. (0~100) 70
3과목의 총점은 250, 평균은 83.3입니다.
계속하려면 아무 키나 누르십시오 . . .

```

이번에는 메소드에서 자기 자신을 다시 호출하는 메소드, 즉 재귀 호출 메소드를 만들어보자. 메소드가 자기 자신을 다시 호출하는 방식이 왜 필요할까?

메소드를 제작하다보면 메소드의 구현 내용이 메소드 자신을 사용하도록 정의되는 경우가 있다. 예를 들어 계승(factorial)을 구하는 경우를 생각해보자. 계승이란 1부터 자기 자신의 수까지를 모두 곱하는 것이다. 즉 5의 계승은 $1*2*3*4*5$ 인 120이다. 먼저 반복문을 사용해서 계승을 구하는 메소드를 만들어보면 다음과 같다. 구현내용에서 보는 바와 같이 1부터 구하려는 수까지 모두 곱해나가면 되는 것이다. 아래 구문을 쉽게 이해할 수 있을 것이다.

```

import java.util.Scanner;
class Test_J2
{
    int FactorialNoRecur(int n) {           // 메소드 시작
        int result = 1;
        for (int i = 1; i < n+1 ; i++ )
        {
            result = result * i;
        }
        return result;
    }

    public static void main(String[] args)
    {
        System.out.println("계승을 구하려는 수를 입력하시오. ");
        Scanner s = new Scanner(System.in);
        int num = s.nextInt();

        Test_J2 obj = new Test_J2();
    }
}

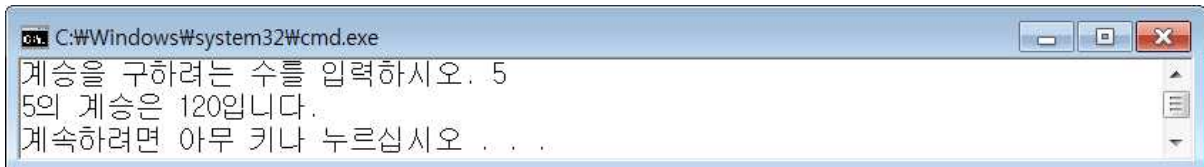
```



```

        System.out.println(num + "의 계승은 " + obj.FactorialNoRecur(num) + "입니
다.");
    }
}

```



그러면 이제 같은 작업을 수행하는 메소드를 재귀 호출 메소드로 만들어보도록 하자. 재귀 호출 메소드는 자신을 불러야 하기 때문에 계승의 원리를 다음과 같이 생각해야 한다.

$$\begin{aligned}
 n\text{의 계승} &= n * (n-1) * (n-2) * \dots * 2 * 1 \\
 &= n * \{(n-1) * (n-2) * \dots * 2 * 1\} \\
 &= n * \{(n-1)\text{의 계승}\}
 \end{aligned}$$

이렇게 해서 계승의 계산 원리 속에 자신보다 하나 적은 수의 계승을 넣을 수 있다. 물론 계승을 무한대로 재귀 호출 할 수 없으며, 기본이 되는 조건이 있어야 한다. 여기에서는 n 의 값이 1인 경우에 더 이상 자기 자신을 호출하지 않도록 해 준다. 즉 다음과 같이 n 의 계승을 재귀 호출의 형태로 정의할 수 있다.

$$\begin{aligned}
 n\text{의 계승} &= n * \{(n-1)\text{의 계승}\} && (n\text{이 } 1\text{보다 큰 경우}) \\
 \text{또는} &= 1 && (n\text{이 } 1\text{이하인 경우})
 \end{aligned}$$

이렇게 정의한 계승을 재귀 호출 메소드로 만들어 보면 다음과 같다.

```

import java.util.Scanner;
class Test_J3
{
    int FactorialNoRecur(int n) {           // 메소드 시작
        if (n <= 1) {
            return 1;
        } else {
            return n * FactorialNoRecur(n - 1);
        }
    }
}

```



```

public static void main(String[] args)
{
    System.out.println("계승을 구하려는 수를 입력하시오. ");
    Scanner s = new Scanner(System.in);
    int num = s.nextInt();

    Test_J3 obj = new Test_J3();

    System.out.println(num + "의 계승은 " + obj.FactorialNoRecur(num) + "입니
다.");
}
}

```

이렇게 재귀 호출 메소드를 만들 때에는 반드시 메소드 자기 자신을 사용하여 정의할 수 있어야 하며, 또한 더 이상의 재귀 호출이 일어나지 않는 기본 조건을 넣어 주어야 한다.

비슷한 예로 1부터 n 까지의 숫자들을 모두 더한 값을 알아내는 작업을 재귀 호출 메소드로 만든다면 다음 구문처럼 하면 된다.

```

int SumToOne(int n){
    if (n == 1)
        return 1;                // n이 1인 경우
    else
        return (n + SumToOne(n-1));    // n이 1보다 큰 경우
}

```

10개의 숫자가 들어 있는 리스트를 역순으로 출력하는 재귀 호출 메소드는 다음과 같다.

```

import java.util.Scanner;
class Test_J4
{
    void PrintReverse(int numbers[], int position) {                // 메소드 시작

        System.out.print (numbers[position] + " ");
        if (position > 0)
        {
            PrintReverse(numbers, position-1);
        }
    }

    public static void main(String[] args)
    {

```

```

        System.out.println("숫자 10개를 입력하시오 ");
        Scanner s = new Scanner(System.in);

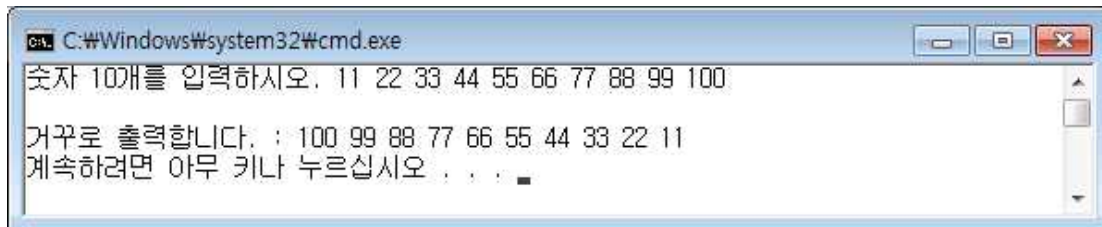
        int numlist[] = new int[10];

        for    (int i = 0; i < numlist.length; i++){
            numlist[i] = s.nextInt();
        }

        Test_J4 obj = new Test_J4();

        System.out.print("거꾸로 출력합니다. :" );
        obj.PrintReverse(numlist, 9);
        System.out.println("");
    }
}

```



```

C:\Windows\system32\cmd.exe
숫자 10개를 입력하시오, 11 22 33 44 55 66 77 88 99 100
거꾸로 출력합니다. : 100 99 88 77 66 55 44 33 22 11
계속하려면 아무 키나 누르십시오 . . .

```

○ 실습 문제

[J01] 나이 계산 및 연령대 판정

[G01] 문제를 참고하여 최대 100명까지 사람들의 2012보다 큰 년도가 입력되기 전까지 태어난 년도를 입력받도록 하라. 입력이 끝나면 AskAge() 메소드를 사용해서 지금까지 입력된 사람들의 나이를 모두 출력하고, 연령대 별로 각각 몇 명인지 출력하라.

AskAge()에서는 태어난 년도를 입력하면 나이를 출력한 후, 유아, 어린이, 청소년, 청년, 중년, 노년 여부를 판정하여 연령대 번호를 리턴한다.

단, 나이 = 2014 - 태어난 년도 + 1 로 계산하고 연령대 구분은 다음과 같이 판정한다.

7세 미만 : 유아, 7세 이상 ~ 13세 미만 : 어린이, 13세 이상 ~ 20세 미만 : 청소년,

20세 이상 ~ 30세 미만 : 청년, 30세 이상 ~ 60세 미만 : 중년, 60세 이상 : 노년

메소드의 선언부는 다음과 같다.

```
int AskAge(int birthyear);
```

파라미터) birthyear : 태어난 년도

리턴값) 계산한 나이에 따른 연령대 번호 (0.유아, 1.어린이, 2.청소년, 3.청년, 4.중년, 5.노년)

```
C:\Windows\system32\cmd.exe
1번째 사람의 태어난 년도를 입력하십시오. 1999
나이는 14 입니다.
2번째 사람의 태어난 년도를 입력하십시오. 2005
나이는 8 입니다.
3번째 사람의 태어난 년도를 입력하십시오. 1940
나이는 73 입니다.
4번째 사람의 태어난 년도를 입력하십시오. 1970
나이는 43 입니다.
5번째 사람의 태어난 년도를 입력하십시오. 1966
나이는 47 입니다.
6번째 사람의 태어난 년도를 입력하십시오. 1988
나이는 25 입니다.
7번째 사람의 태어난 년도를 입력하십시오. 2010
나이는 3 입니다.
8번째 사람의 태어난 년도를 입력하십시오. 1979
나이는 34 입니다.
9번째 사람의 태어난 년도를 입력하십시오. 1990
나이는 23 입니다.
10번째 사람의 태어난 년도를 입력하십시오. 1981
나이는 32 입니다.
11번째 사람의 태어난 년도를 입력하십시오. 2008
나이는 5 입니다.
12번째 사람의 태어난 년도를 입력하십시오. 2222
유아는 2명 입니다.
어린이는 1명 입니다.
청소년은 1명 입니다.
청년은 2명 입니다.
중년은 4명 입니다.
노년은 1명 입니다.
계속하려면 아무 키나 누르십시오 . . .
```

[J02] 심사점수 계산

심사점수를 10개를 입력받아 리스트에 저장한 후, 이 리스트를 파라미터로 하여 가장 큰 점수를 구하는 Max()와 가장 작은 점수를 구하는 Min()을 사용하여 10개의 점수 중 최대점수와 최소점수를 제외한 8개의 점수에 대한 평균을 계산하여 출력하라.

메소드의 선언부는 다음과 같다.

```
double Max(double num[]);
```

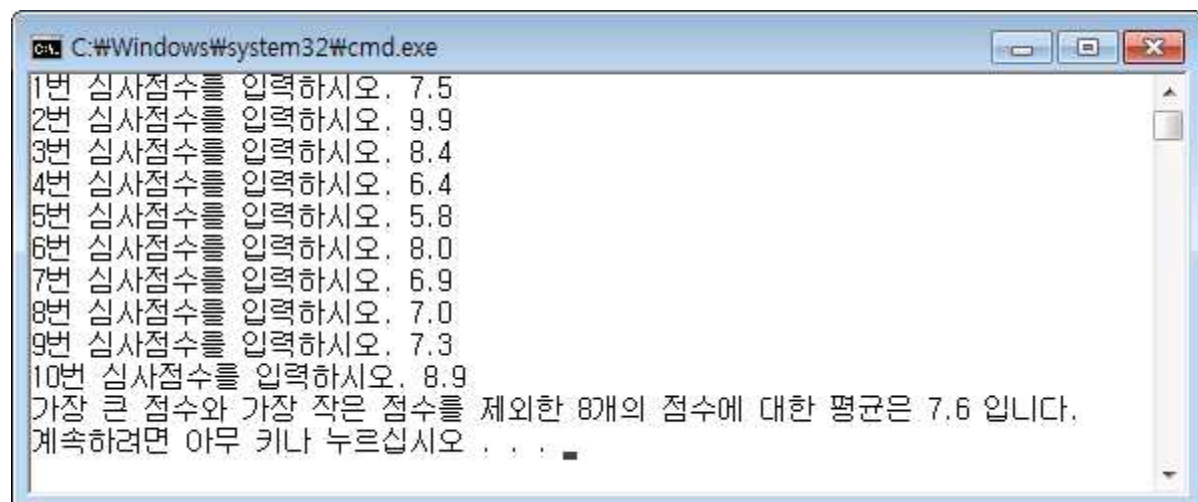
파라미터) num : 숫자 리스트

리턴값) 숫자 리스트에서 가장 큰 값

```
double Min(double num[]);
```

파라미터) num : 숫자 리스트

리턴값) 숫자 리스트에서 가장 작은 값



```
C:\Windows\system32\cmd.exe
1번 심사점수를 입력하십시오. 7.5
2번 심사점수를 입력하십시오. 9.9
3번 심사점수를 입력하십시오. 8.4
4번 심사점수를 입력하십시오. 6.4
5번 심사점수를 입력하십시오. 5.8
6번 심사점수를 입력하십시오. 8.0
7번 심사점수를 입력하십시오. 6.9
8번 심사점수를 입력하십시오. 7.0
9번 심사점수를 입력하십시오. 7.3
10번 심사점수를 입력하십시오. 8.9
가장 큰 점수와 가장 작은 점수를 제외한 8개의 점수에 대한 평균은 7.6 입니다.
계속하려면 아무 키나 누르십시오 . . .
```

[J03] 물의 온도 구간 개수 판정

물의 온도를 10회 입력받은 후, 이 물이 각각 어느 정도의 온수인지 AskWater()를 통해 판정하여 그 결과를 출력하라. 출력할 내용은 입력된 10개의 온도 값, 냉수 입력 횟수, 미온수 입력 횟수, 온수 입력 횟수, 끓는 물 입력 횟수를 각각 출력하라.

단, 온수의 판정 구간은 다음과 같이 판정한다.

0도 ~ 25도 미만 : 냉수

25도 ~ 40도 미만 : 미온수

40도 ~ 80도 미만 : 온수

80도 이상 : 끓는 물

메소드의 선언부는 다음과 같다.

byte AskWater(double degree);

파라미터) degree: 온도

리턴 값) 온도 판정 번호 (0.냉수, 1.미온수, 2.온수, 3.끓는 물)

```

C:\Windows\system32\cmd.exe
1번째 물의 온도를 입력하십시오. 15.5
2번째 물의 온도를 입력하십시오. 20.5
3번째 물의 온도를 입력하십시오. 27.7
4번째 물의 온도를 입력하십시오. 35.0
5번째 물의 온도를 입력하십시오. 40.5
6번째 물의 온도를 입력하십시오. 66.7
7번째 물의 온도를 입력하십시오. 90.1
8번째 물의 온도를 입력하십시오. 100.0
9번째 물의 온도를 입력하십시오. 77.7
10번째 물의 온도를 입력하십시오. 28.0
1번 물의 온도는 15.5도입니다.
2번 물의 온도는 20.5도입니다.
3번 물의 온도는 27.7도입니다.
4번 물의 온도는 35.0도입니다.
5번 물의 온도는 40.5도입니다.
6번 물의 온도는 66.7도입니다.
7번 물의 온도는 90.1도입니다.
8번 물의 온도는 100.0도입니다.
9번 물의 온도는 77.7도입니다.
10번 물의 온도는 28.0도입니다.
냉수 입력 횟수 2입니다.
미온수 입력 횟수는 3입니다.
온수 입력 횟수는 3입니다.
끓는 물 입력 횟수는 2입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

[J04] 연중 날짜 계산 메소드를 이용한 날짜 간격 세기

날짜 2개를 입력받은 후, 이 2개의 날짜 간격은 며칠인지 계산하여 출력하라. 단, 월과 일로 파라미터로 넘기면 이 날짜가 1년 중 몇 번째 날에 해당되는지 리턴하는 메소드 CalcDate() 메소드를 만들어 사용하라. 이 메소드에서 매 월의 날 수 계산 시 다음과 같이 매월의 날 수를 배열로 만들어 이를 이용하여 계산하라.
 int monthdays = {31,28,31,30,31,30,31,31,30,31,30,31}; // 1~12월의 날 수

메소드의 선언부는 다음과 같다.

```
int CalcDate(int month, int day);
```

파라미터) month : 월, day : 일

리턴값) 1년 중 해당 날짜가 몇 번째 날인지의 결과 값 (1~365)

```
C:\Windows\system32\cmd.exe
첫번째 날짜를 입력하십시오. (월 일) 1 1
두번째 날짜를 입력하십시오. (월 일) 5 5
두 날짜의 간격은 124일입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
첫번째 날짜를 입력하십시오. (월 일) 5 5
두번째 날짜를 입력하십시오. (월 일) 1 1
두 날짜의 간격은 124일입니다.
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
첫번째 날짜를 입력하십시오. (월 일) 13 1
두번째 날짜를 입력하십시오. (월 일) 2 30
잘못 입력하셨습니다.
계속하려면 아무 키나 누르십시오 . . .
```

[J05] 주차 관리 시스템

주차장에서 차량들의 주차 관리 시스템을 만들어라. 차량마다 주차를 시작한 시간을 시와 분으로 입력받고, 주차를 종료한 시간을 시와 분으로 입력받은 후, 이를 CalcParking() 메소드에 파라미터로 넘겨 주차요금을 리턴받도록 하라. 차량이 더 있는지 물어서 더 이상 차량이 없을 때까지 반복해서 요금을 계산하되 반복이 끝나면 지금까지 계산한 차량의 수량과 총 주차요금을 화면에 출력하라. 주차요금은 10분당 500원으로 한다.

메소드의 선언부는 다음과 같다.

`int CalcParking(int start_h, int start_m, int end_h, int end_m) :`

파라미터) `start_h` : 주차시작 시, `start_m` : 주차시작 분, `end_h` : 주차종료 시, `end_m` : 주차종료 분

리턴값) 주차시작 시간(시, 분)부터 종료 시간(시, 분)까지의 주차요금(원)



```

C:\Windows\system32\cmd.exe
1번 차량 주차 시작 시각 (시 분) : 10 30
1번 차량 주차 종료 시각 (시 분) : 11 15
주차요금 : 2500원
더 입력하시겠습니까?(Y/N) Y
2번 차량 주차 시작 시각 (시 분) : 9 10
2번 차량 주차 종료 시각 (시 분) : 15 10
주차요금 : 18000원
더 입력하시겠습니까?(Y/N) Y
3번 차량 주차 시작 시각 (시 분) : 12 10
3번 차량 주차 종료 시각 (시 분) : 14 55
주차요금 : 8500원
더 입력하시겠습니까?(Y/N) Y
4번 차량 주차 시작 시각 (시 분) : 11 00
4번 차량 주차 종료 시각 (시 분) : 11 05
주차요금 : 500원
더 입력하시겠습니까?(Y/N) N
주차차량 4대의 총 주차 요금은 29500원입니다.
계속하려면 아무 키나 누르십시오 . . .
    
```

[J06] 피보나치 수 구하기

n이 1부터 20까지 증가하는 경우 각각의 피보나치 수 `fibonacci(n)`을 출력하라. 피보나치 수는 다음과 같이 정의한다.

$$fibonacci(n) = \begin{cases} 1 & (n = 1 \text{ 또는 } n = 2 \text{인 경우}) \\ fibonacci(n-1) + fibonacci(n-2) & (n > 2 \text{인 경우}) \end{cases}$$

단, 메소드의 선언부는 다음과 같이 사용하라.

```
int fibonacci(int n) ;
```

```
C:\Windows\system32\cmd.exe
1부터 20까지 피보나치 수는 다음과 같습니다.
1번째 : 1
2번째 : 1
3번째 : 2
4번째 : 3
5번째 : 5
6번째 : 8
7번째 : 13
8번째 : 21
9번째 : 34
10번째 : 55
11번째 : 89
12번째 : 144
13번째 : 233
14번째 : 377
15번째 : 610
16번째 : 987
17번째 : 1597
18번째 : 2584
19번째 : 4181
20번째 : 6765
계속하려면 아무 키나 누르십시오 . . .
```


[J07] 2의 제곱수 구하기

반복해서 임의의 숫자 n 을 입력받은 후 2^n 을 계산하여 출력하되, 재귀메소드를 이용하여 계산하라. 이 때 사용할 재귀메소드 `poweroftwo()`의 정의는 다음과 같다.

$$\text{poweroftwo}(n) = \begin{cases} 1 & (n = 0 \text{인 경우}) \\ 2 \times \text{poweroftwo}(n-1) & (n > 0 \text{인 경우}) \end{cases}$$

단, 메소드의 선언부 다음과 같이 사용하라.

```
int poweroftwo(int n) ;
```

```
C:\Windows\system32\cmd.exe
숫자를 입력하십시오. (0. 종료) : 2
2의 2승은 : 4
숫자를 입력하십시오. (0. 종료) : 4
2의 4승은 : 16
숫자를 입력하십시오. (0. 종료) : 8
2의 8승은 : 256
숫자를 입력하십시오. (0. 종료) : 10
2의 10승은 : 1024
숫자를 입력하십시오. (0. 종료) : 16
2의 16승은 : 65536
숫자를 입력하십시오. (0. 종료) : 20
2의 20승은 : 1048576
숫자를 입력하십시오. (0. 종료) : 24
2의 24승은 : 16777216
숫자를 입력하십시오. (0. 종료) : 5
2의 5승은 : 32
숫자를 입력하십시오. (0. 종료) : 7
2의 7승은 : 128
숫자를 입력하십시오. (0. 종료) : 0
계속하려면 아무 키나 누르십시오 . . .
```

[J08] Ackermann 수 구하기

Ackermann's Function A 는 다음과 같이 정의된다. $A(i, j)$ 를 재귀 호출 메소드로 만들고, 이 메소드를 이용하여 $A(0,0)$ 에서 $A(3,3)$ 의 값을 구하라.

Ackermann's Function A

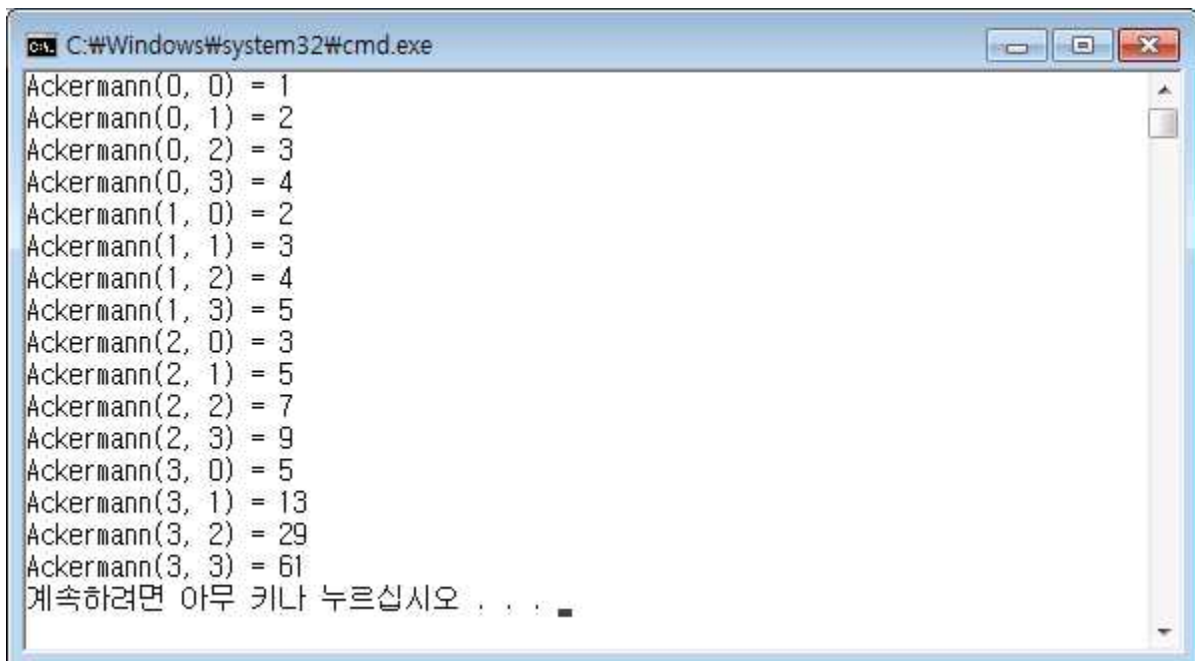
$A(0, j) = j+1$ $i = 0$ 이고, $j \geq 0$ 인 경우

$A(i, 0) = A(i-1, 1)$ $i > 0$ 이고 $j = 0$ 인 경우

$A(i, j) = A(i-1, A(i, j-1))$ i 와 j 모두 0보다 큰 경우

단, 메소드의 선언부는 다음과 같이 사용하라.

```
int Ackermann(int i, int j) ;
```



```
C:\Windows\system32\cmd.exe
Ackermann(0, 0) = 1
Ackermann(0, 1) = 2
Ackermann(0, 2) = 3
Ackermann(0, 3) = 4
Ackermann(1, 0) = 2
Ackermann(1, 1) = 3
Ackermann(1, 2) = 4
Ackermann(1, 3) = 5
Ackermann(2, 0) = 3
Ackermann(2, 1) = 5
Ackermann(2, 2) = 7
Ackermann(2, 3) = 9
Ackermann(3, 0) = 5
Ackermann(3, 1) = 13
Ackermann(3, 2) = 29
Ackermann(3, 3) = 61
계속하려면 아무 키나 누르십시오 . . .
```

[J09] pow() 메소드 만들기

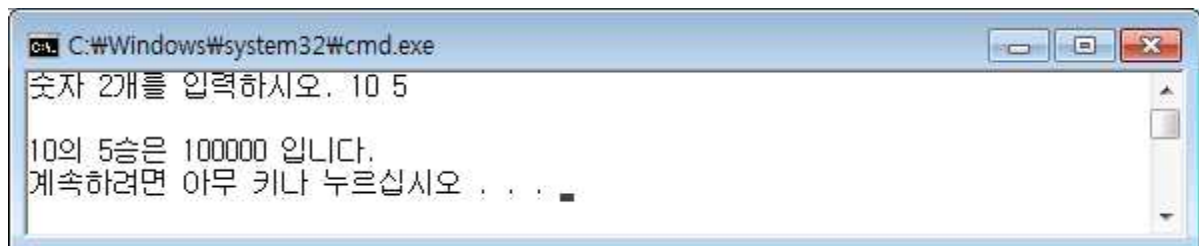
n^a 를 계산할 수 있는 math 모듈 내의 pow() 메소드와 같은 일을 하는 power() 메소드를 재귀 호출을 이용하여 만들어라. 그리고 숫자 2개(num1, num2)를 입력받아 $\text{num1}^{\text{num2}}$ 를 계산하라.

단, power() 메소드는 다음과 같이 정의된다.

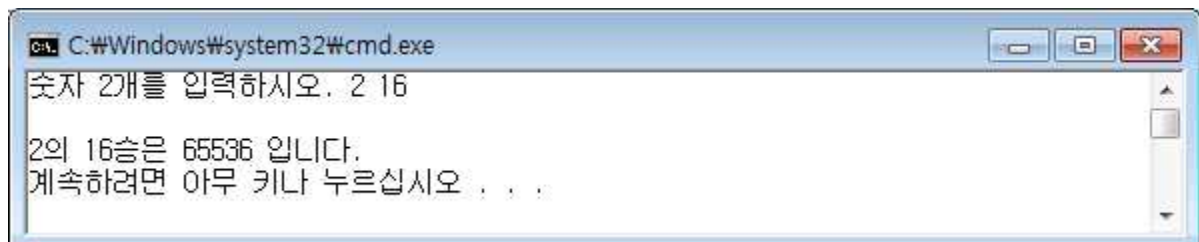
$$\text{power}(n, a) = \begin{cases} 1 & (a = 0 \text{인 경우}) \\ n & (a = 1 \text{인 경우}) \\ \text{power}(n, \frac{a}{2}) \times \text{power}(n, \frac{a}{2}) & (a > 1 \text{이고 짝수인 경우}) \\ \text{power}(n, \frac{a}{2}) \times \text{power}(n, \frac{a}{2}) \times n & (a > 1 \text{이고 홀수인 경우}) \end{cases}$$

단, 메소드의 선언부는 다음과 같이 사용하라.

```
int power( int num1, int num2 ) ;
```



```
C:\Windows\system32\cmd.exe
숫자 2개를 입력하십시오. 10 5
10의 5승은 100000 입니다.
계속하려면 아무 키나 누르십시오 . . .
```



```
C:\Windows\system32\cmd.exe
숫자 2개를 입력하십시오. 2 16
2의 16승은 65536 입니다.
계속하려면 아무 키나 누르십시오 . . .
```

[J10] 피보나치 수열로 황금비율 구하기

문제 [J06]에서 제작한 피보나치 메소드 `fibonacci(n)`를 사용하여 황금비율을 찾아내라. n 번째 황금비율이란 피보나치 수열의 연속적인 2개의 숫자의 비율($n+1$ 번째 수 / n 번째 수)로 정한다.
단, 계산한 비율이 직전의 비율에 비해 그 차이가 백만분의 1보다 작게 되면 멈추도록 하라.

```

C:\Windows\system32\cmd.exe
2번째 비율 (2 / 1) : 2.000000
3번째 비율 (3 / 2) : 1.500000
4번째 비율 (5 / 3) : 1.666667
5번째 비율 (8 / 5) : 1.600000
6번째 비율 (13 / 8) : 1.625000
7번째 비율 (21 / 13) : 1.615385
8번째 비율 (34 / 21) : 1.619048
9번째 비율 (55 / 34) : 1.617647
10번째 비율 (89 / 55) : 1.618182
11번째 비율 (144 / 89) : 1.617977
12번째 비율 (233 / 144) : 1.618056
13번째 비율 (377 / 233) : 1.618026
14번째 비율 (610 / 377) : 1.618037
15번째 비율 (987 / 610) : 1.618033
16번째 비율 (1597 / 987) : 1.618034
17번째 비율 (2584 / 1597) : 1.618034
계속하려면 아무 키나 누르십시오 . . .

```

[Step K] 여러 클래스 사용하기

이번 단계에서는 여러 클래스를 사용하여 데이터를 읽고 쓰는 방법을 연습하도록 하겠다. 클래스를 사용하기 위해서는 변수를 사용해야 하는데 한 사람의 정보에 다음과 같은 내용이 들어 있다고 가정한다.

```
홍길동
강원도 원주시 흥업면
25
```

위와 같은 정보를 담기 위하여 다음과 같은 클래스가 필요하다.(클래스 이름과 필드 이름은 임의로 정한다.)

```
class userInfo {
    String name; // 사용자 이름
    String address; // 주소
    int age;      // 나이
}
```

또한, 위의 클래스에 정보를 저장하기 위한 멤버 변수뿐만 아니라 정보를 물어 저장하거나 이미 저장된 정보들을 출력하기 위한 메소드를 포함할 수도 있다.

```
class userInfo {
    String name; // 사용자 이름
    String address; // 주소
    int age;      // 나이

    void input(){ // 정보를 사용자에게 입력받아 설정하는 메소드
        Scanner s = new Scanner(System.in);
        System.out.print("이름을 입력하세요 :");
        name = s.next();
        System.out.print("주소를 입력하세요 :");
        address = s.next();
        System.out.print("나이를 입력하세요 :");
        age = s.nextInt();
    }

    void print(){ // 설정된 정보를 출력함.
        System.out.println("이름은 : " + name);
    }
}
```

```

        System.out.println("주소는 :" + address);
        System.out.println("나이는 :" + age);
    }
}

```

실제로 위의 클래스를 사용하여 프로그램에서 사용자 정보를 입력 받기 위해서는 다음과 같이 main 메소드가 포함된 클래스에서 위 클래스 userInfo를 객체화 해서 사용하면 된다.

```

import java.util.Scanner;
class userInfo {
    String name;    // 사용자 이름
    String address; // 주소
    int age;        // 나이

    void input(){    // 정보를 사용자에게 입력받아 설정하는 메소드
        Scanner s = new Scanner(System.in);
        System.out.print("이름을 입력하세요 :");
        name = s.next();
        System.out.print("주소를 입력하세요 :");
        address = s.next();
        System.out.print("나이를 입력하세요 :");
        age = s.nextInt();
    }

    void print(){    // 설정된 정보를 출력함.
        System.out.println("이름은 :" + name);
        System.out.println("주소는 :" + address);
        System.out.println("나이는 :" + age);
    }
}

class TEST_K01{
    public static void main(String args[]){
        userInfo user1 = new userInfo();
        user1.input();
        user1.print();
    }
}

```

위의 소스를 실행하면, 한 파일에 두 개의 클래스 파일이 있으므로, 두 개의 .class 파일이 생성된다.

```

이름을 입력하세요 :홍길동
주소를 입력하세요 :경기도
나이를 입력하세요 :23
이름은 :홍길동
주소는 :경기도
나이는 :23
계속하려면 아무 키나 누르십시오 . . .

```

만약 여러 명의 정보를 입출력하기를 원한다면 어떻게 할 것인가? 그것은 배열을 사용하면 된다. 다섯명의 정보를 저장할 공간이 필요하다면 아래와 같이 코딩을 하면 된다. 그런데 배열을 만들 때 프리미티브 타입(primitive type)과 다른 점은 배열을 선언한 후, 각 클래스를 생성자로 다시 한 번 생성을 해 주어야만 된다.

```

userInfo user[] = new userInfo[5];
for(int i = 0; i < user.length; i++){
    user[i] = new userInfo();
}

```

위의 소스를 배열을 사용하는 경우에는 다음과 같이 코딩할 수 있다.

```

import java.util.Scanner;
class userInfo2 {
    String name; // 사용자 이름
    String address; // 주소
    int age; // 나이

    void input(){ // 정보를 사용자에게 입력받아 설정하는 메소드
        Scanner s = new Scanner(System.in);
        System.out.print("이름을 입력하세요 :");
        name = s.next();
        System.out.print("주소를 입력하세요 :");
        address = s.next();
        System.out.print("나이를 입력하세요 :");
        age = s.nextInt();
    }

    void print(){ // 설정된 정보를 출력함.
        System.out.println("이름은 : " + name);
        System.out.println("주소는 : " + address);
        System.out.println("나이는 : " + age);
    }
}

```

```

class TEST_K02{
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        int usercount = 0;
        userInfo2 user[] = new userInfo2[5];
        for(int i = 0; i < user.length; i++){
            user[i] = new userInfo2();
        }

        while(true){
            System.out.print("원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :");
            int menu = s.nextInt();
            if(menu == 1){
                user[usercount].input();
                usercount++;
            }else if(menu == 2){
                for(int i= 0; i < usercount; i++){
                    user[i].print();
                }
            }else if(menu == 3){
                break;
            }
        }
    }
}

```

실행한 결과는 다음과 같다.

```

원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :1
이름을 입력하세요 :홍길동
주소를 입력하세요 :경기도
나이를 입력하세요 :23
원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :2
이름은 :홍길동
주소는 :경기도
나이는 :23
원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :1
이름을 입력하세요 :성춘향
주소를 입력하세요 :서울시
나이를 입력하세요 :22
원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :2

```


이름은 :홍길동
주소는 :경기도
나이는 :23
이름은 :성춘향
주소는 :서울시
나이는 :22
원하는 메뉴를 입력하세요 (1:입력, 2:출력, 3:종료) :3
계속하려면 아무 키나 누르십시오 . . .

○ 실습 문제

[K01] 메뉴판 저장하기

다음 결과 예에서와 같이 메뉴 내용을 입력받은 후, 화면에 출력하는 프로그램을 제작하고 테스트하라.(멤버 변수와 메소드 및 파라미터, 리턴타입은 각자 결정함)

클래스명 : MenuPan

- 입력 메소드
- 출력 메소드

main 메소드 포함 클래스 명 : K01

결과 예)

메뉴의 번호, 메뉴명, 원산지, 가격을 입력하시오. 1 삼겹살 국내산 9000

=====

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000

[K02] 메뉴판 저장하기

K01에서 만든 MenuPan을 변경하지 않고, 다음에서와 같이 다섯 개의 메뉴를 입력할 수 있도록 프로그램을 제작하고 테스트하라.(멤버변수와 메소드 및 파라미터, 리턴타입은 각자 결정함)
 main 메소드 포함 클래스 명 : K02

결과 예)

등록하실 메뉴의 개수를 입력하세요 : 4

1번 메뉴의 번호, 메뉴명, 원산지, 가격을 입력하시오. 1 삼겹살 국내산 9000

2번 메뉴의 번호, 메뉴명, 원산지, 가격을 입력하시오. 2 갈비살 미국산 15000

3번 메뉴의 번호, 메뉴명, 원산지, 가격을 입력하시오. 3 꽃등심 호주산 30000

4번 메뉴의 번호, 메뉴명, 원산지, 가격을 입력하시오. 4 양념갈비 국내산 25000

=====

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000
2	갈비살	미국산	15000
3	꽃등심	호주산	30000
4	양념갈비	국내산	25000

[K03] 좌표 저장하기

화면과 같이 1개의 좌표 값(x,y)을 입력받아 point 클래스에 저장하라. 그리고 정보를 다시 읽어 들여서 이 좌표가 몇 사분면에 위치하는 지를 알아내는 메소드를 통해 각 사분면에 속하는 좌표의 갯수를 출력하라.

메소드 선언부는 다음과 같다.

클래스명 : point

- int get_area(double x, double y) ;

파라미터 : x좌표, y좌표

리턴값 : 좌표가 위치한 사분면 (1,2,3,4, 만일 x 또는 y가 0이면 리턴값은 0이다.)

main 메소드 포함 클래스 명 : K03

결과 예)

좌표의 X, Y 값을 입력하시오 --> 1.0 2.0
=====

1번째 좌표는 1사분면에 위치합니다.

1사분면의 좌표는 모두 1개입니다.

2사분면의 좌표는 모두 0개입니다.

3사분면의 좌표는 모두 0개입니다.

4사분면의 좌표는 모두 0개입니다.

[K04] 좌표 저장하기

K03에서 만든 point 클래스를 변경하지 않고, 화면과 같이 원하는 수만큼의 좌표 값(x,y)을 입력받아 이 좌표가 몇 사분면에 위치하는 지를 알아내는 메소드를 통해 각 사분면에 속하는 좌표의 갯수를 출력하라.

메소드 선언부는 다음과 같다.

main 메소드 포함 클래스 명 : K04

결과 예)

```
몇 개의 좌표를 입력하겠습니까? 5
1번째 좌표의 X, Y 값을 입력하십시오 --> 1.0  2.0
2번째 좌표의 X, Y 값을 입력하십시오 --> -1.0  2.0
3번째 좌표의 X, Y 값을 입력하십시오 --> -1.0  -2.0
4번째 좌표의 X, Y 값을 입력하십시오 --> 1.0  -2.0
5번째 좌표의 X, Y 값을 입력하십시오 --> 3.5  4.7
=====
1번째 좌표는 1사분면에 위치합니다.
2번째 좌표는 2사분면에 위치합니다.
3번째 좌표는 3사분면에 위치합니다.
4번째 좌표는 4사분면에 위치합니다.
5번째 좌표는 1사분면에 위치합니다.
1사분면의 좌표는 모두 2개입니다.
2사분면의 좌표는 모두 1개입니다.
3사분면의 좌표는 모두 1개입니다.
4사분면의 좌표는 모두 1개입니다.
```

[K05] 사용자 목록 저장하기

화면과 같이 아이디와 비밀번호를 저장하는 클래스를 생성하여 입력받는 메소드와 출력하는 메소드를 생성하라.

메소드 선언부는 다음과 같다.

클래스명 : UserInfo

void input(); //입력받는 메소드

void print(int index); //출력하는 메소드

main 메소드 포함 클래스 명 : K05

결과 예)

학생의 아이디, 비밀번호를 입력하시오 --> hong 1234

=====

등록된 학생 목록은 다음과 같습니다.

번호	아이디	비밀번호
1	hong	1234

[K06] 사용자 목록 저장하기

K05에서 만든 UserInfo 클래스를 변경하지 않고, 10명의 아이디와 비밀번호를 저장하는 클래스를 생성하여 입력받는 메소드와 출력하는 메소드를 생성하라.
main 메소드 포함 클래스 명 : K06

결과 예)

등록할 학생의 명수는? 5

1번 학생의 아이디, 비밀번호를 입력하시오 --> hong 1234
2번 학생의 아이디, 비밀번호를 입력하시오 --> kim 2345
3번 학생의 아이디, 비밀번호를 입력하시오 --> lee i123
4번 학생의 아이디, 비밀번호를 입력하시오 --> kkk ccdd23
5번 학생의 아이디, 비밀번호를 입력하시오 --> gogil 5588

=====

등록된 5명의 학생 목록은 다음과 같습니다.

번호	아이디	비밀번호
1	hong	1234
2	kim	2345
3	lee	i123
4	kkk	ccdd23
5	gogil	5588

[K07] 학생 점수 결과 저장하기

학생들의 국어, 영어, 수학 점수를 입력받아 총점과 평균을 계산하고 이 클래스를 이용하여 5명의 정보를 입력하고 출력할 수 있도록 프로그램을 완성하라.

클래스명 : Student

```
void input();           //입력받는 메소드
void print(int index);  //출력하는 메소드
```

결과 예)

등록할 학생의 명수는? 3

1번 학생의 국어, 영어, 수학점수를 입력하시오 --> 95 85 75

2번 학생의 국어, 영어, 수학점수를 입력하시오 --> 90 80 70

3번 학생의 국어, 영어, 수학점수를 입력하시오 --> 60 85 44

=====

등록된 3명의 학생 목록은 다음과 같습니다.

번호	국어	영어	수학	총점	평균	학점
1	95	85	75	255	85.0	B+
2	90	80	70	240	80.0	B
3	60	85	44	189	63.0	D

[Step L] 가변적인 배열 사용하기

이번 단계에서는 한 클래스를 여러 번 사용하기 위해서 개수가 정해진 배열 대신에 가변적으로 배열의 길이를 늘이거나 줄일 수 있는 ArrayList 클래스를 사용하려고 한다. 이 클래스는 가변적으로 배열의 길이를 추가하거나 삭제할 수 있으며, 또 다른 특징에는 다양한 객체를 원소로 가질 수도 있다는 점이다.

다음 예제를 보면 5명의 사용자 정보 객체를 배열로 생성한 예이다. 이 예에서 보면 이 배열은 최대 5명의 정보만 등록할 수 있으며, 중간에 정보를 삭제할 때에는 정보를 지우는 방식으로 해야만 한다.

```
import java.util.Scanner;
class UserInfoArray{
    String name ; // 사용자 이름
    int age ;      // 나이

    UserInfoArray(String n, int a){
        this.name = n;
        this.age = a;
    }
    void print(){
        System.out.println(this.name + "\t" + this.age);
    }
    public static void main(String arg[]){
        Scanner s = new Scanner(System.in);
        UserInfoArray u[] = new UserInfoArray[5];
        for(int I=0; I < u.length; I++){
            System.out.print("사용자 이름을 입력하세요.");
            String name = s.next();
            System.out.print("사용자 나이를 입력하세요.");
            int age = s.nextInt();
            u[I] = new UserInfoArray(name, age);
        }
        System.out.println("=====");
        System.out.println("등록하신 사용자는 다음과 같습니다.");
        for(int I=0; I < u.length; I++){
            u[I].print();
        }
    }
}
```

}

위와 같은 기능을 하면서 가변적인 배열이 되도록 ArrayList를 사용하려면 다음과 같은 단계를 적용해야 한다.

1. import java.util.ArrayList 추가;
2. ArrayList<데이터타입> 변수명 = new ArrayList<데이터타입>();

```
import java.util.*;           //ArrayList 클래스와 Scanner 사용하기 위한 패키지
class UserInfoArrayList{
    String name ; // 사용자 이름
    int age ;      // 나이

    UserInfoArrayList(String n, int a){
        this.name = n;
        this.age = a;
    }
    void print(){
        System.out.println(this.name + "\t" + this.age);
    }

    public static void main(String arg[]){
        Scanner s = new Scanner(System.in);

        //UserInfoArrayList를 요소로 갖는 ArrayList 객체화
        ArrayList<UserInfoArrayList> u = new ArrayList<UserInfoArrayList>() ;

        while(true){
            System.out.print("사용자 이름을 입력하세요.");
            String name = s.next();
            System.out.print("사용자 나이를 입력하세요.");
            int age = s.nextInt();
            u.add(new UserInfoArrayList(name, age)); //요소 추가

            System.out.print("계속 등록하시겠습니까?(Y/N)");
            String re = s.next();
            if(re.equals("N")) break;
        }

        System.out.println("=====");
        System.out.println("등록하신 사용자는 다음과 같습니다.");
    }
}
```

```

        for(int I=0; I < u.size(); I++){           //요소의 개수를 가져옴
            u.get(I).print();
        }
    }
}

```

위의 소스와 같이 ArrayList를 이용하여 사용자 정보를 개수에 제한 없이 추가하거나 수정, 삭제할 수 있다.

아래에는 ArrayList에서 사용할 수 있는 주요 메소드를 정리하고 있다. (더 자세히 알고 싶으면 Java API에서 ArrayList 클래스를 참조한다.)

메소드	설명
boolean add(E e)	리스트 마지막에 요소를 추가한다.
E get(int index)	리스트의 index 위치에 있는 요소를 리턴한다.
int size()	리스트에 있는 요소의 개수를 리턴한다.
E remove(int index)	리스트의 index 위치에 있는 요소를 삭제한다. 삭제된 요소를 리턴한다.

○ 실습 문제

[L01] 메뉴판 저장하기

다음과 같이 메뉴에 따라 데이터를 입력받거나, 삭제하거나, 리스트를 보여주는 프로그램을 객체지향으로 제작하라.(클래스이름, 멤버변수, 메소드 및 파라미터, 리턴타입은 각자 결정함)

결과 예)

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 1

메뉴의 메뉴명, 원산지, 가격을 입력하시오. 삼겹살 국내산 9000
입력되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 1

메뉴의 메뉴명, 원산지, 가격을 입력하시오. 갈비살 호주산 15000

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 4

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000
2	갈비살	호주산	15000

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 5

2개의 메뉴가 등록되어 있습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 4

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000
2	갈비살	호주산	15000

==> 삭제할 번호를 입력하세요 : 1

삭제되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 2

메뉴번호	메뉴이름	원산지	1인분가격
1	갈비살	호주산	15000

==> 수정할 번호를 입력하세요 : 1

==> 메뉴의 메뉴명, 원산지, 가격을 입력하시오. 갈비살 호주산 12000

수정되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 6

종료되었습니다.

[L02] 주차장 입출력 관리

다음과 같이 메뉴에 따라 데이터를 입력받거나, 삭제하거나, 리스트를 보여주는 프로그램을 객체지향으로 제작하라.(클래스이름, 멤버변수, 메소드 및 파라미터, 리턴타입은 각자 결정함)

- 입차할 때는 비어있는 자리부터 순서대로 입차한다.
 - 주차 비용은 처음 10분은 무료, 10분 추가할 때마다 500원씩 정한다.
- 단, 현재 날짜와 시간을 가져오기 위해서 SimpleDateFormat 클래스를 참조한다.
예)

```
SimpleDateFormat f = new SimpleDateFormat("YYYYMMdd", Locale.KOREA);
String today = f.format(new Date());
```

결과 예)

주차할 수 있는 차의 개수를 입력하세요 : 10

총 10대를 주차할 수 있습니다.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 1

입차할 차의 번호와 차종을 입력하세요. 3131 아반테

[1번] 2014년 2월 13일 12시 10분 입차 되었습니다.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 1

입차할 차의 번호와 차종을 입력하세요. 1212 소나타

[2번] 2014년 2월 13일 12시 20분 입차 되었습니다.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 3

[1번] 3131 아반테 2014-02-13 12:10

[2번] 1212 소나타 2014-02-13 12:20

8대 주차가능.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 2

[1번] 3131 아반테 2014-02-13 12:10

[2번] 1212 소나타 2014-02-13 12:20

==> 출차 번호를 입력하세요 : 1

==> 현재 시간은 [20140213 13:29]이므로 주차금액은 3500원입니다.

==> 정산되었습니다.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 3

[2번] 1212 소나타 2014-02-13 12:20
9대 주차가능.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 4

총 1대가 주차되어 있습니다.

[2번] 1212 소나타 2014-02-13 12:20

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 5

주차 가능한 자리는 9대입니다.

1) 입차 2) 출차 3) 리스트 4)입차 총개수 5)주차잔여개수 6)종료 ==> 6

종료되었습니다.

[L03] 식당 주문 관리

다음과 같이 메뉴에 따라 데이터를 입력받거나, 삭제하거나, 리스트를 보여주는 프로그램을 객체지향으로 제작하라.(클래스이름, 멤버변수, 메소드 및 파라미터, 리턴타입은 각자 결정함)

식사메뉴 :

스테이크 - 25000원

스파게티 - 15000원

파스타 - 17000원

- 멤버십 카드를 소지하면 식사 금액의 10퍼센트를 할인해 준다.
- 이렇게 계산된 식사 금액이 10만원 미만이면 봉사료 7%가 추가되고, 10만원 이상이면 봉사료 10%가 추가된다.
- 마지막으로 부가가치세 10%가 추가되면 총 식사금액이 결정된다.

결과 예)

1) 주문 2) 결제 3)리스트 4)종료 ==> 1

손님 수를 입력하세요 : 3

스테이크, 스파게티, 파스타의 개수를 입력하세요 : 1 1 1

멤버십 카드가 있습니까?(Y:있음) Y

들어오신 시간은 2014년 2월 13일 18:40입니다.

총 금액은 60380 원입니다.

1) 주문 2) 결제 3)리스트 4)종료 ==> 1

손님 수를 입력하세요 : 4

스테이크, 스파게티, 파스타의 개수를 입력하세요 : 1 1 1

멤버십 카드가 있습니까?(Y:있음) N

들어오신 시간은 2014년 2월 13일 19:40입니다.

총 금액은 67089 원입니다.

1) 주문 2) 결제 3)리스트 4)종료 ==> 3

번호 손님수 스테이크 스파게티 파스타 멤버십 입장시간

[1] 3 1 1 1 Y 2014년 2월 13일 18:40

[2] 4 1 1 1 N 2014년 2월 13일 18:40

1) 주문 2) 결제 3)리스트 4)종료 ==> 2


```
-----  
번호 손님수 스테이크 스파게티 파스타 멤버쉽 결제금액 입장시간  
[1] 3      1      1      1      Y      60380 2014년 2월 13일 18:40  
[2] 4      1      1      1      N      67089 2014년 2월 13일 18:40
```

==> 결제하실 번호를 입력하세요 : 1

==> 결제완료되었습니다.

```
-----  
1) 주문 2) 결제 3)리스트 4)종료 ==> 4  
-----
```

종료되었습니다.

[L04] 학생 정보 관리

다음과 같이 메뉴에 따라 데이터를 입력받거나, 삭제하거나, 리스트를 보여주는 프로그램을 객체지향으로 제작하라.(클래스이름, 멤버변수, 메소드 및 파라미터, 리턴타입은 각자 결정함)

학생정보

- 이름
- 학번
- 성별(W or M)
- 주소
- 전화번호
- 복학생여부(true or false)

결과 예)

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 1

학생이름, 학번, 성별을 입력하세요 : 홍길동 20131232 M

주소를 입력하세요 : 경기도 안산시

전화번호를 입력하세요 : 01012223333

복학생입니까?(Y/N) Y

총 1명이 등록되었습니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 1

학생이름, 학번, 성별을 입력하세요 : 성준향 201112323 W

주소를 입력하세요 : 서울시 강남구

전화번호를 입력하세요 : 01023237777

복학생입니까?(Y/N) N

총 2명이 등록되었습니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 4

이름	학번	성별	주소	전화번호	복학생
홍길동	20131232	M	경기도 안산시	01012223333	Y
성준향	201112323	W	서울시 강남구	01023237777	N

총 2명이 등록되었습니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 2

번호	이름	학번	성별	주소	전화번호	복학생
----	----	----	----	----	------	-----

```
1  홍길동  20131232      M      경기도 안산시  01012223333  Y
2  성춘향  201112323      W      서울시 강남구  01023237777  N
```

==> 수정하실 번호를 입력하세요 : 1

학생이름, 학번, 성별을 입력하세요 : 홍길영 20131232 M

주소를 입력하세요 : 경기도 용인시

전화번호를 입력하세요 : 01012223334

복학생입니까?(Y/N) Y

수정되었습니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 3

```
번호 이름      학번      성별      주소      전화번호      복학생
1  홍길영  20131232      M      경기도 용인시  01012223334  Y
2  성춘향  201112323      W      서울시 강남구  01023237777  N
```

==> 삭제하실 번호를 입력하세요 : 2

정말로 삭제하시겠습니까?(Y/N) Y

삭제되었습니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 5

검색하실 이름을 입력하세요 : 홍

검색되었습니다.

```
번호 이름      학번      성별      주소      전화번호      복학생
1  홍길영  20131232      M      경기도 용인시  01012223334  Y
```

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 6

```
번호 이름      학번      성별      주소      전화번호      복학생
1  홍길영  20131232      M      경기도 용인시  01012223334  Y
```

복학생은 1명입니다.

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 7

검색하실 이름을 입력하세요 : 용인

```
번호 이름      학번      성별      주소      전화번호      복학생
1  홍길영  20131232      M      경기도 용인시  01012223334  Y
```

1) 학생등록 2) 정보수정 3)삭제 4)리스트 5)이름검색 6)복학생리스트 7)주소검색 8)종료 ==> 8

종료되었습니다.

[Step M] 파일 사용하기

이번 단계에서는 java.io 패키지에 있는 클래스를 사용하여 기존의 파일이나 폴더에 대한 제어를 하는 법을 학습해 본다. 특히 File 클래스를 사용하여 자신이 원하는 폴더나 파일에 정보를 저장하거나 읽어올 수 있으며, 폴더의 파일 리스트등을 알 수 있다.

다음 예제를 보면 원하는 파일을 만들어서 파일에 있는 내용을 읽어오거나 현재 입력한 정보를 파일에 저장할 수 있는 메모장 예이다.

```
import java.util.Scanner;
import java.io.File;

class MyFile{
    String filename ;        // 파일명

    void save(){
        System.out.println("저장되었습니다.");
        System.out.println("저장되었습니다.");
    }
    public static void main(String arg[]){
        Scanner s = new Scanner(System.in);
        UserInfoArray u[] = new UserInfoArray[5];
        for(int I=0; I < u.length; I++){
            System.out.print("사용자 이름을 입력하세요.");
            String name = s.next();
            System.out.print("사용자 나이를 입력하세요.");
            int age = s.nextInt();
            u[I] = new UserInfoArray(name, age);
        }
        System.out.println("=====");
        System.out.println("등록하신 사용자는 다음과 같습니다.");
        for(int I=0; I < u.length; I++){
            u[I].print();
        }
    }
}
```

위와 같은 기능을 하면서 가변적인 배열이 되도록 ArrayList를 사용하려면 다음과 같은 단계를 적용해야 한다.

3. import java.util.ArrayList 추가;

4. ArrayList<데이터타입> 변수명 = new ArrayList<데이터타입>();

```
import java.util.*;           //ArrayList 클래스와 Scanner 사용하기 위한 패키지
class UserInfoArrayList{
    String name ; // 사용자 이름
    int age ;      // 나이

    UserInfoArrayList(String n, int a){
        this.name = n;
        this.age = a;
    }
    void print(){
        System.out.println(this.name + "\t" + this.age);
    }

    public static void main(String arg[]){
        Scanner s = new Scanner(System.in);

        //UserInfoArrayList를 요소로 갖는 ArrayList 객체화
        ArrayList<UserInfoArrayList> u = new ArrayList<UserInfoArrayList>() ;

        while(true){
            System.out.print("사용자 이름을 입력하세요.");
            String name = s.next();
            System.out.print("사용자 나이를 입력하세요.");
            int age = s.nextInt();
            u.add(new UserInfoArrayList(name, age)); //요소 추가

            System.out.print("계속 등록하시겠습니까?(Y/N)");
            String re = s.next();
            if(re.equals("N")) break;
        }

        System.out.println("=====");
        System.out.println("등록하신 사용자는 다음과 같습니다.");
        for(int I=0; I < u.size(); I++){           //요소의 개수를 가져옴
```

```

        u.get(I).print();
    }
}

```

위의 소스와 같이 ArrayList를 이용하여 사용자 정보를 개수에 제한 없이 추가하거나 수정, 삭제할 수 있다.

아래에는 ArrayList에서 사용할 수 있는 주요 메소드를 정리하고 있다. (더 자세히 알고 싶으면 Java API에서 ArrayList 클래스를 참조한다.)

메소드	설명
<code>boolean add(E e)</code>	리스트 마지막에 요소를 추가한다.
<code>E get(int index)</code>	리스트의 <code>index</code> 위치에 있는 요소를 리턴한다.
<code>int size()</code>	리스트에 있는 요소의 개수를 리턴한다.
<code>E remove(int index)</code>	리스트의 <code>index</code> 위치에 있는 요소를 삭제한다. 삭제된 요소를 리턴한다.

○ 실습 문제

[L01] 메뉴판 저장하기

다음과 같이 메뉴에 따라 데이터를 입력받거나, 삭제하거나, 리스트를 보여주는 프로그램을 객체지향으로 제작하라.(클래스이름, 멤버변수, 메소드 및 파라미터, 리턴타입은 각자 결정함)

결과 예)

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 1

 메뉴의 메뉴명, 원산지, 가격을 입력하시오. 삼겹살 국내산 9000
 입력되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 1

 메뉴의 메뉴명, 원산지, 가격을 입력하시오. 갈비살 호주산 15000

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 4

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000
2	갈비살	호주산	15000

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 5

 2개의 메뉴가 등록되어 있습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 4

메뉴번호	메뉴이름	원산지	1인분가격
1	삼겹살	국내산	9000
2	갈비살	호주산	15000

==> 삭제할 번호를 입력하세요 : 1

삭제되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 2

메뉴번호	메뉴이름	원산지	1인분가격
1	갈비살	호주산	15000

==> 수정할 번호를 입력하세요 : 1

==> 메뉴의 메뉴명, 원산지, 가격을 입력하시오. 갈비살 호주산 12000

수정되었습니다.

1) 추가 2) 수정 3) 삭제 4)리스트 5)총개수 6)종료 ==> 6

종료되었습니다