객체지향 프로그래밍

Object-Oriented Programming

실습 과제 해설

Problem 1

Problem 1. 구구단

사용자로부터 1~9의 정수를 입력 받아, 1 부터 9까지 곱한 수를 출력하는 프로그램을 작성하라.

출력 형식: X * Y = Z

[입출력 예시]

출력할 단을 입력하세요: 5

5 * 1 = 5

5 * 2 = 10

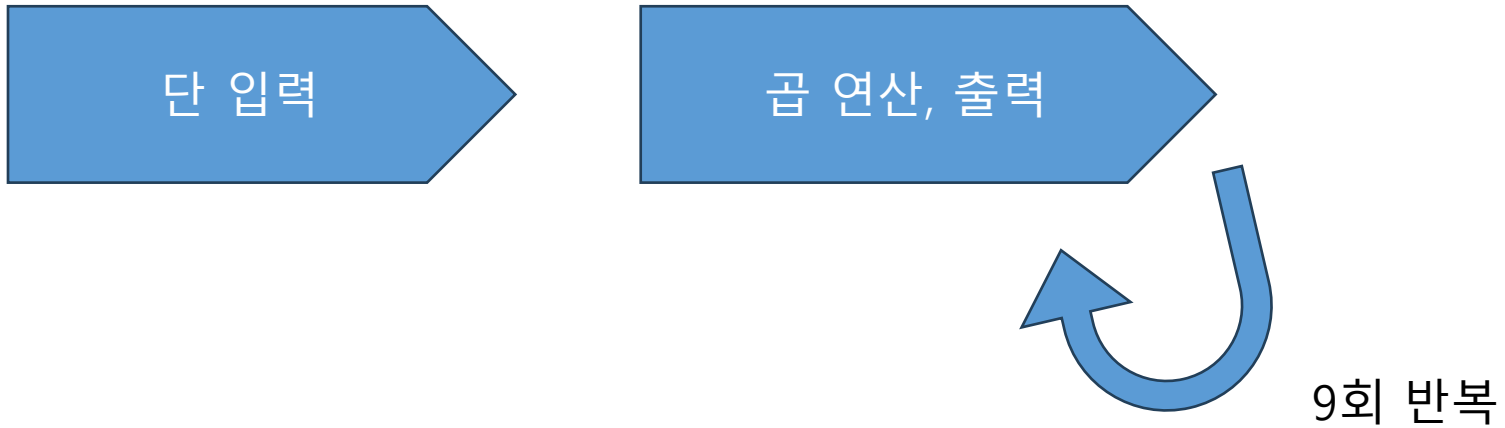
5 * 3 = 15

...

5 * 9 = 45

Problem 1

Process



Problem 1

Source code

```java
import java.util.Scanner;

public class Problem1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int multiplicand = scanner.nextInt();
        for (int i = 1; i < 10; i++) {
            String output = multiplicand + " * " + i + " = " + multiplicand * i;
            System.out.println(output);
        }
    }
}
```

Problem 2

Problem 2. 점수 관리

학생 수를 입력받고, 각 학생의 점수를 차례대로 입력받아 **총점, 평균, 최고점, 최저점**을 출력하는 프로그램을 작성하라. (각 입력은 100을 초과하지 않는 자연수)

[입출력 예시]

학생 수를 입력하세요: 4
1번째 학생 점수: 85
2번째 학생 점수: 100
3번째 학생 점수: 70
4번째 학생 점수: 90

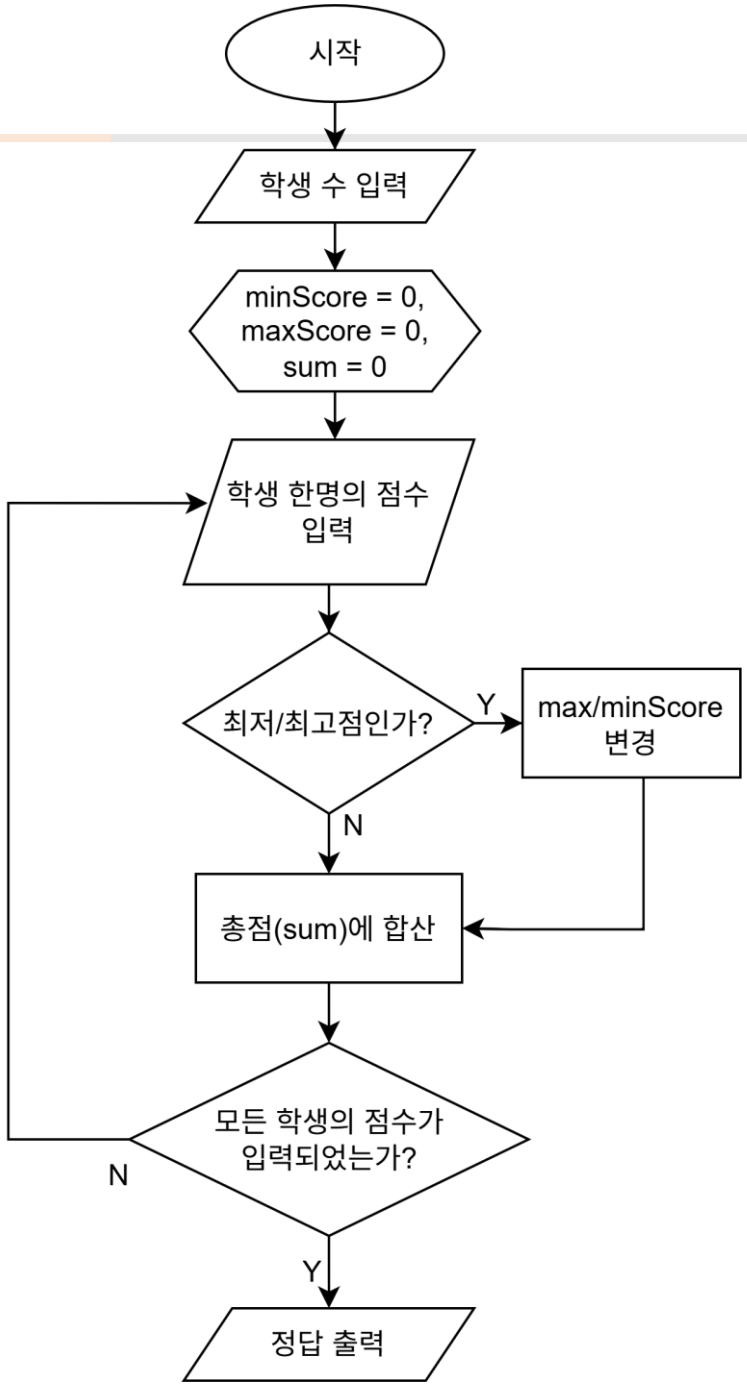
총점: 345, 평균: 86.25, 최고점: 100, 최저점: 70

Problem 2

Problem 2. 점수 관리

```java
import java.util.Scanner;

public class Problem2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("학생 수를 입력하세요: ");
        int count = scanner.nextInt();
        int maxScore = 0, minScore = 100, sum = 0;
        for (int i = 0; i < count; i++) {
            System.out.println(i + 1 + "번째 학생 점수: ");
            int score = scanner.nextInt();
            if (score > maxScore) {
                maxScore = score;
            }
            if (score < minScore) {
                minScore = score;
            }
            sum += score;
        }
        double avg = (double) sum / count;
        System.out.println("총점: " + sum + ", 평균: "
                + avg + ", 최고점: " + maxScore + ", 최저점: " + minScore);
    }
}
```



Problem 3

Problem 3

동호는 새악대로 T 통신사의 새 휴대폰 옴머나를 샀다.
새악대로 T 통신사는 동호에게 다음 두 가지 요금제 중 하나를 선택하라고 했다.

1. A 요금제
2. B 요금제

A 요금제는 30초 구간으로 10원씩 청구된다.
(29초 이하로 통화를 하면 10원이 청구되고, 30초부터 59초 사이로 통화를 하면 20원이 청구된다.)

B 요금제는 60초 구간으로 15원씩 청구된다.

(59초 이하로 통화를 하면 15원이 청구되고, 60초부터 119초 사이로 통화를 하면 30원이 청구된다.)

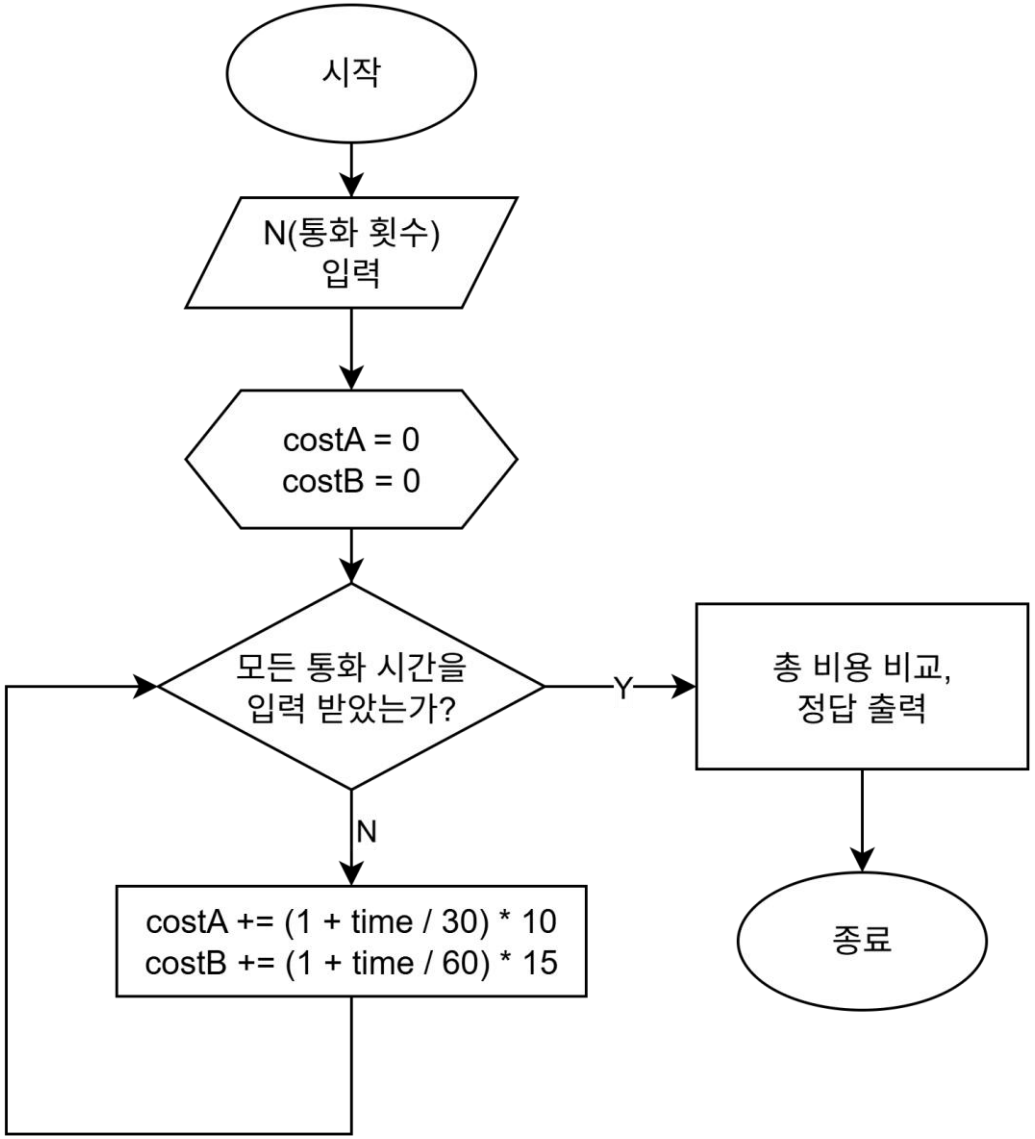
동호가 저번 달에 새악대로 T 통신사를 이용할 때 통화 시간 목록이 주어지면 어느 요금제를 사용 하는 것이 저렴한지 출력하는 프로그램을 작성하시오.

Problem 3

Problem 3

costA: A 통신사를 이용했을 때의 요금

costB: B 통신사를 이용했을 때의 요금



Problem 3

Problem 3

```java
import java.util.Scanner;

public class Problem3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int costA = 0;
        int costB = 0;
        for (int i = 0; i < n; i++) {
            int time = scanner.nextInt();
            costA += (1 + time / 30) * 10;
            costB += (1 + time / 60) * 15;
        }
        if(costA < costB) System.out.println("A " + costA);
        else if(costA == costB) System.out.println("A B " + costA);
        else System.out.println("B " + costB);
    }
}
```

Problem 4

Problem 4

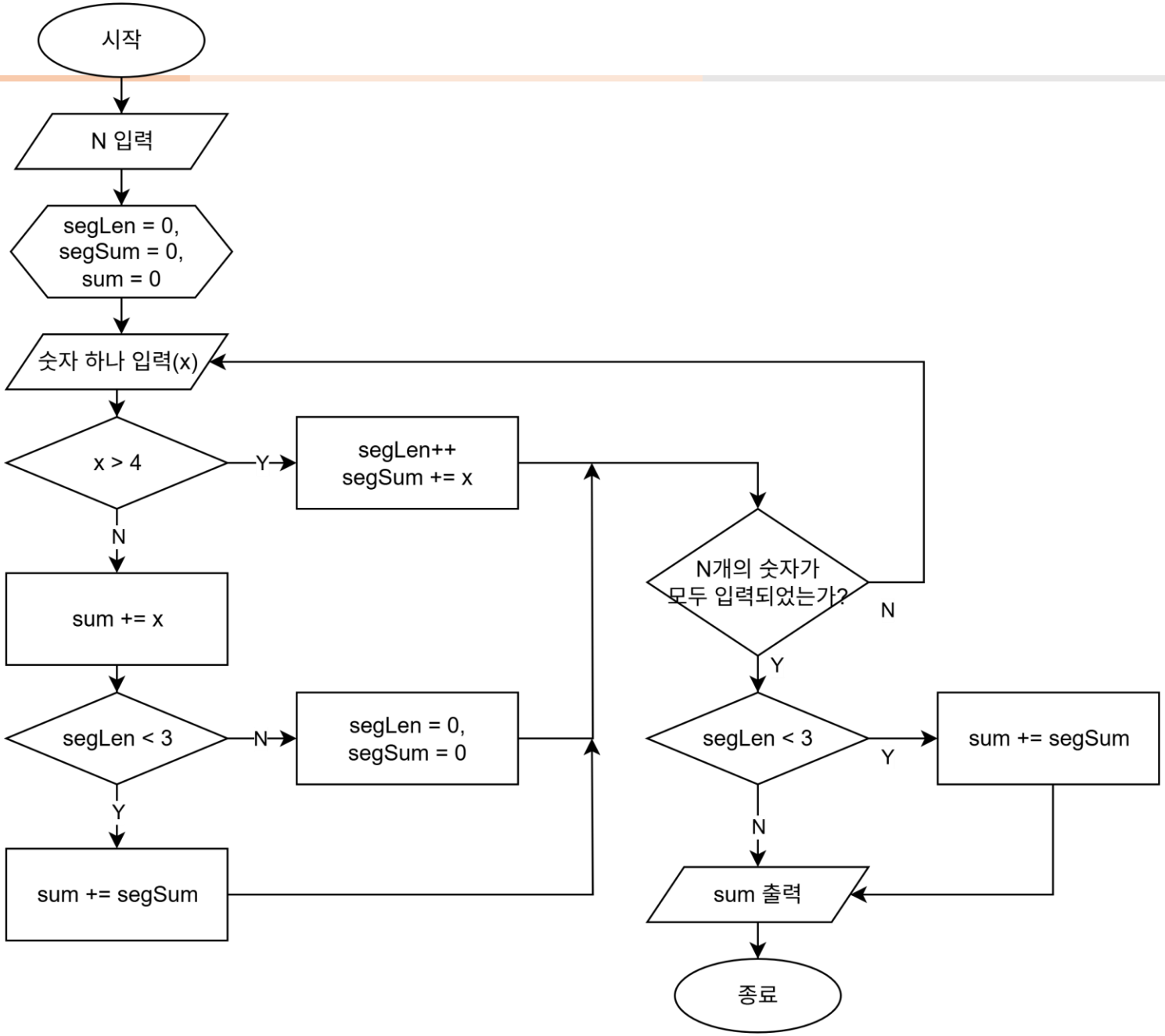
사용자로부터 정수 N을 입력 받는다. 이후 N개의 정수를 입력받아 순서대로 더하여 출력한다.

단, 5 이상의 정수가 3회 이상 연속으로 등장하는 구간은 합산에서 제외한다.

모든 입력은 자연수이며, 50을 초과하지 않는다.

Problem 4

Problem 4



Problem 4

Problem 4

```java
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt(), segmentLen = 0, segmentSum = 0, sum = 0;
    for (int i = 0; i < n; i++) {
        int x = scanner.nextInt();
        if(x >= 5) {
            segmentLen++;
            segmentSum += x;
            continue;
        }
        sum += x;
        if(segmentLen < 3) sum += segmentSum;
        segmentLen = 0;
        segmentSum = 0;
    }
    if(segmentLen < 3) sum += segmentSum;
    System.out.println(sum);
}
```

Chapter 05.
문자열, 배열, 열거 타입

문자열

문자열 리터럴

- 문자열 리터럴은 내부적으로 new String()을 호출해 생성한 객체이다.
- 그러나 내용이 같은 문자열 리터럴이라면 더 이상 새로운 String 객체를 생성하지 않은 채 기존 리터럴을 공유.
  따라서 s1과 s2는 동일한 String 객체를 가리킨다.

```
String 변수;      // String 타입의 변수 선언
변수 = "문자열";  // String 타입의 변수에 문자열 대입
```

```
String s1 = "안녕, 자바!";     // String 타입의 변수 선언과 초기화
String s2 = "안녕, 자바!";     // String 타입의 변수 선언과 초기화
```

문자열 리터럴이다.

문자열

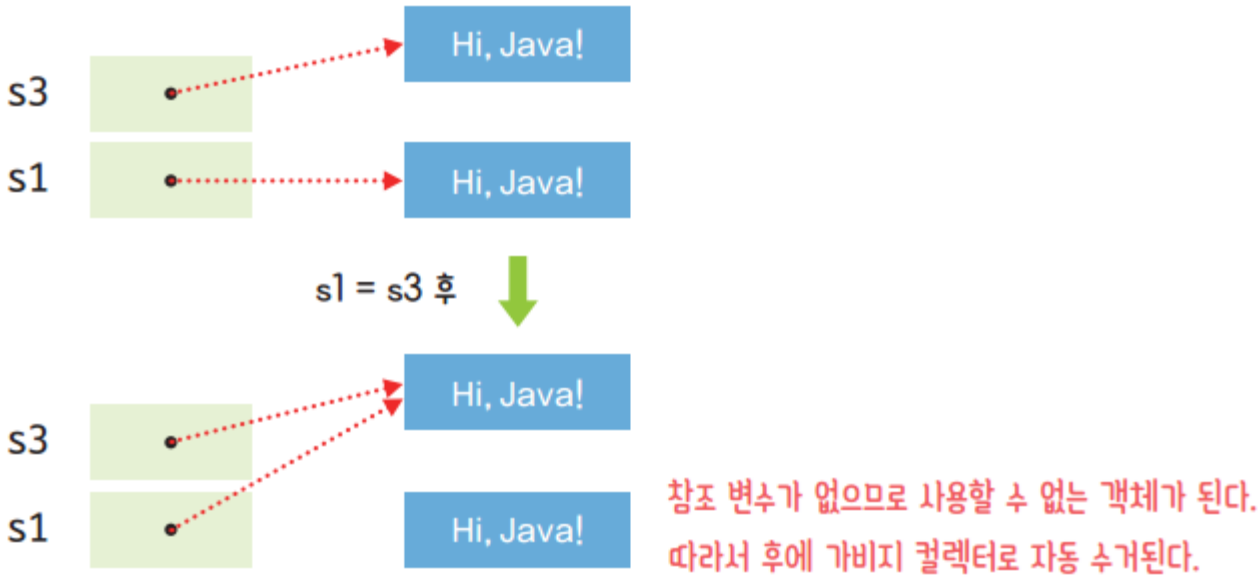
문자열의 비교

- ==와 != 연산자는 두 문자열의 내용을 비교하는 것이 아니라 동일한 객체인지 검사

문자열

Practice

- 예제 5-1: 문자열 객체 비교
  - chap05.String1Demo



문자열

문자열의 비교

- String 클래스에서 제공하는 문자열 비교 메서드

메서드	설명
int compareTo(String s)	문자열을 사전 순으로 비교해 정수 값을 반환한다.
int compareToIgnoreCase(String s)	대 · 소문자를 무시하고, 문자열을 사전 순으로 비교한다.
boolean equals(String s)	주어진 문자열 s와 현재 문자열을 비교한 후 true/false를 반환한다.
boolean equalsIgnoreCase(String s)	주어진 문자열 s와 현재 문자열을 대 · 소문자 구분 없이 비교한 후 true/false를 반환한다.

문자열

Practice

- 예제 5-2: 문자열 내용 비교
  - chap05.String2Demo

문자열

문자열 조작

- 문자열과 관련된 유용한 메서드들을 제공
  - 끝나는 문자열 조사
  - 시작 문자열 조사
  - 소문자로 변환
  - 대문자로 변환
  - 공백 제거
  - 문자열 길이

문자열

문자열 조작 – String 클래스의 인스턴스 메서드

메서드	설명
char charAt(int index)	index가 지정한 문자를 반환
String concat(String s)	주어진 문자열 s를 현재 문자열 뒤에 연결
boolean contains(String s)	문자열 s를 포함하는지 조사
boolean endsWith(String s)	끝나는 문자열이 s인지 조사
int indexOf(String s)	문자열 s가 나타난 위치를 반환
boolean isBlank( )	길이가 0 혹은 공백 있으면 true 반환(자바 11부터)
boolean isEmpty( )	길이가 0이면 true 반환
int length( )	길이를 반환
String repeat(int c)	c번 반복한 문자열을 반환(자바 11부터)
boolean startsWith(String s)	시작하는 문자열이 s인지 조사
String substring(int index)	index부터 시작하는 문자열의 일부를 반환
String toLowerCase( )	모두 소문자로 변환
String toUpperCase( )	모두 대문자로 변환
String trim( )	앞뒤에 있는 공백을 제거한 후 반환

문자열

문자열 조작 – String 클래스의 정적 메서드

정적 메서드	설명
String format( )	주어진 포맷에 맞춘 문자열을 반환
String join( )	주어진 구분자와 연결한 문자열을 반환(자바 8부터)
String valueOf( )	각종 기초 타입이나 객체를 문자열로 반환

문자열

Practice

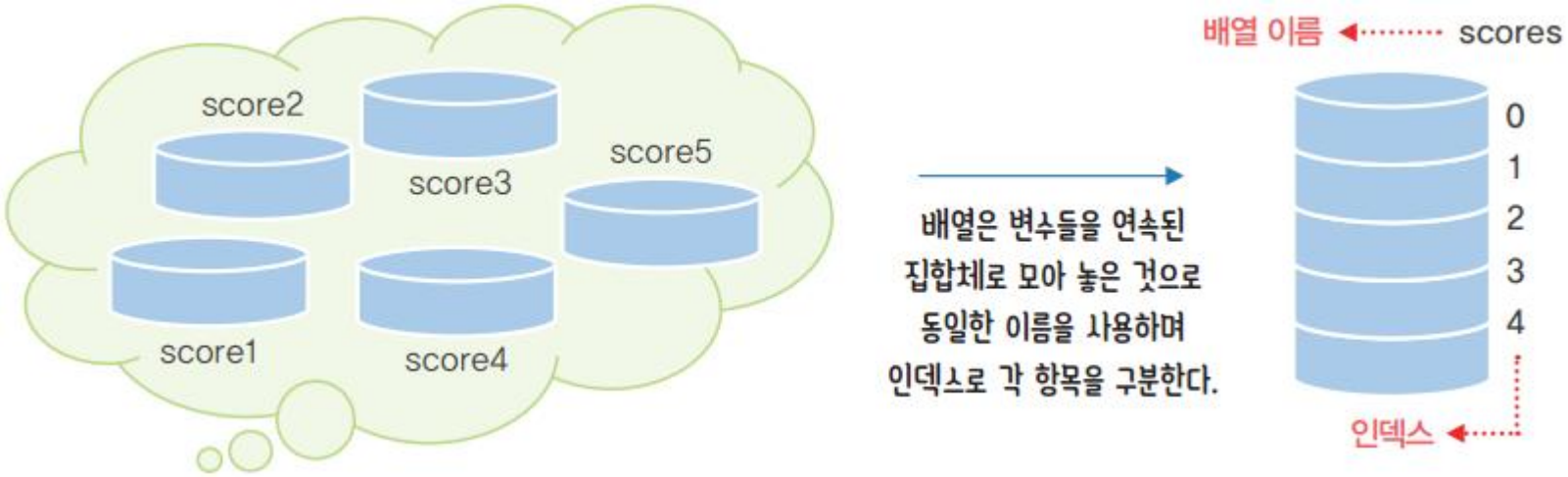
- 예제 5-3: String 클래스의 메서드 이용
  - chap05.String3Demo

- 예제 5-5: String 클래스의 정적 메서드 이용
  - chap05.String5Demo

배열

배열

- 동일 타입 데이터들의 순서 있는 집합



배열

배열의 필요성

```java
int score1 = 100;
int score2 = 90;
int score3 = 50;
int score4 = 95;
int score5 = 85;

int sum = score1;
sum += score2;
sum += score3;
sum += score4;
sum += score5;
double average = sum / 5;
```

(a) 배열을 사용하지 않을 때

```java
int[] scores = { 100, 90, 50, 95, 85 };
int sum = 0;

for (int i = 0; i < 5; i++)
  sum += scores[i];
double average = sum / 5;
```

(b) 배열을 사용할 때

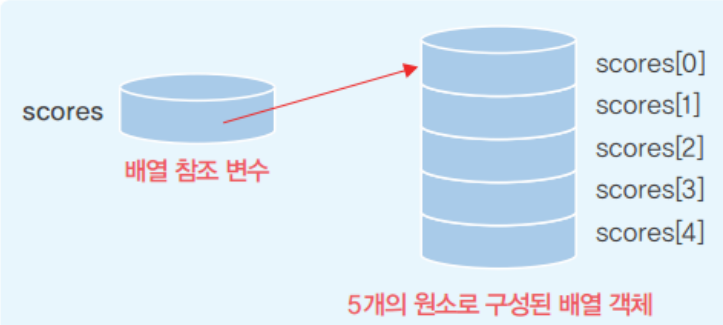
배열

배열의 선언과 생성

- Java에서 배열은 클래스와 마찬가지로 참조 타입
- 기초 타입과 참조 타입 둘 다 배열 원소가 될 수 있음
- new 키워드를 통해 배열을 생성

배열의 크기

```
scores = new int[5];
```




```
// 방법 ❶
int[] scores = { 100, 90, 50, 95, 85 };

// 방법 ❷
int[] scores = new int[] { 100, 90, 50, 95, 85 };

// 방법 ❸
int[] scores;
scores = new int[] { 100, 90, 50, 95, 85 };

// 방법 ❹
int[] scores;
scores = { 100, 90, 50, 95, 85 };
```

배열

배열 원소 접근

```
배열이름[인덱스];
```

배열의 크기

- 배열이 생성될 때 배열의 크기가 결정
- 배열의 length 필드가 배열의 크기를 나타냄.

배열

Practice

- 예제 5-6: 배열 실습
  - chap05.Array1Demo

배열

다차원 배열

- 배열을 원소로 갖는 배열

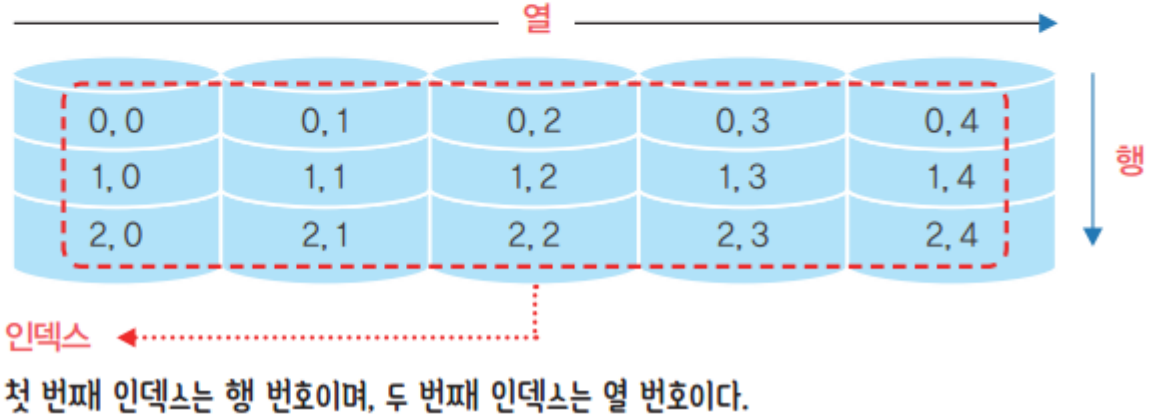
  - 예를 들어 학생 3명의 5과목 성적을 처리하는 정수 타입 2차원 배열(3행 × 5열)인 scores를 선언하고 생성해 보자.

```
int[][] scores = new int[3][5];
```

2개의 대괄호는 2차원 배열을 표시

행의 개수

열의 개수

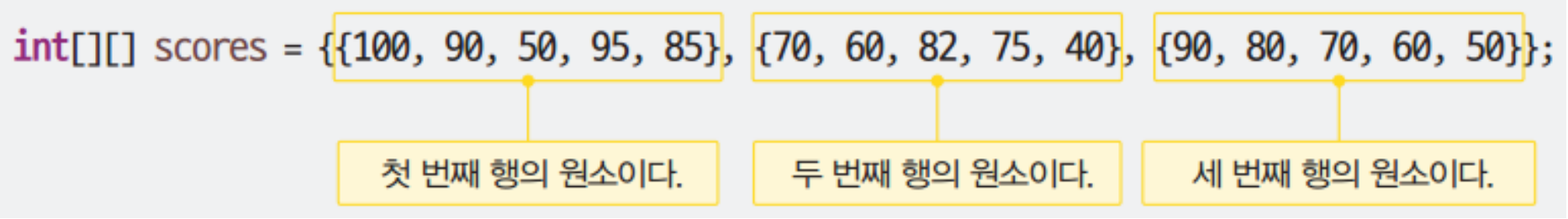


첫 번째 인덱스는 행 번호이며, 두 번째 인덱스는 열 번호이다.

배열

다차원 배열

- 선언과 초기화



배열

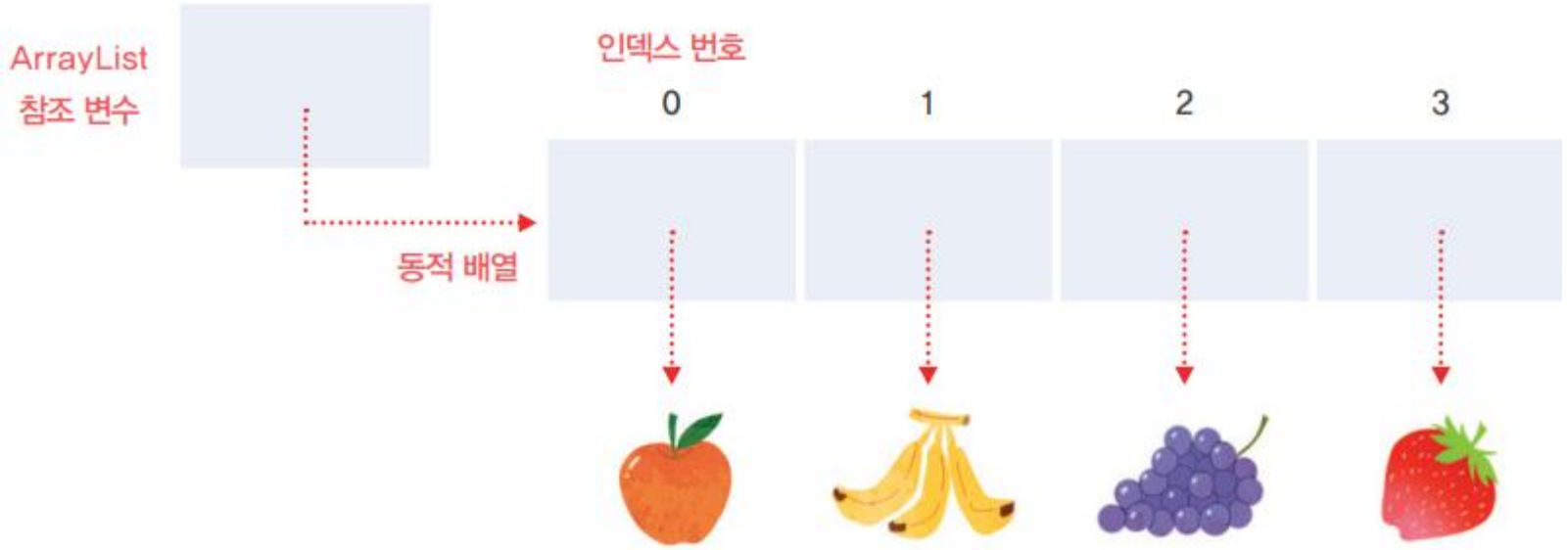
Practice

- 예제 5-7: 다차원 배열 실습
  - chap05.Array2Demo

배열

동적 배열

- 자바는 크기가 유동적인 배열을 지원하기 위하여 ArrayList 클래스를 제공



배열

동적 배열

- ArrayList 객체 생성

```
ArrayList<참조타입> 참조변수 = new ArrayList<>();
```

기초 타입의 동적 배열이라면 Integer, Long, Short, Float, Double, Charater, Boolean 등을 사용한다.

- ArrayList 원소 접근

참조변수.add(데이터)	데이터를 동적 배열에 원소로 추가
참조변수.remove(인덱스번호)	동적 배열에서 인덱스 번호의 원소를 제거
참조변수.get(인덱스번호)	동적 배열에서 인덱스 번호의 원소를 가져오기
참조변수.size()	동적 배열에 포함된 원소 개수

배열

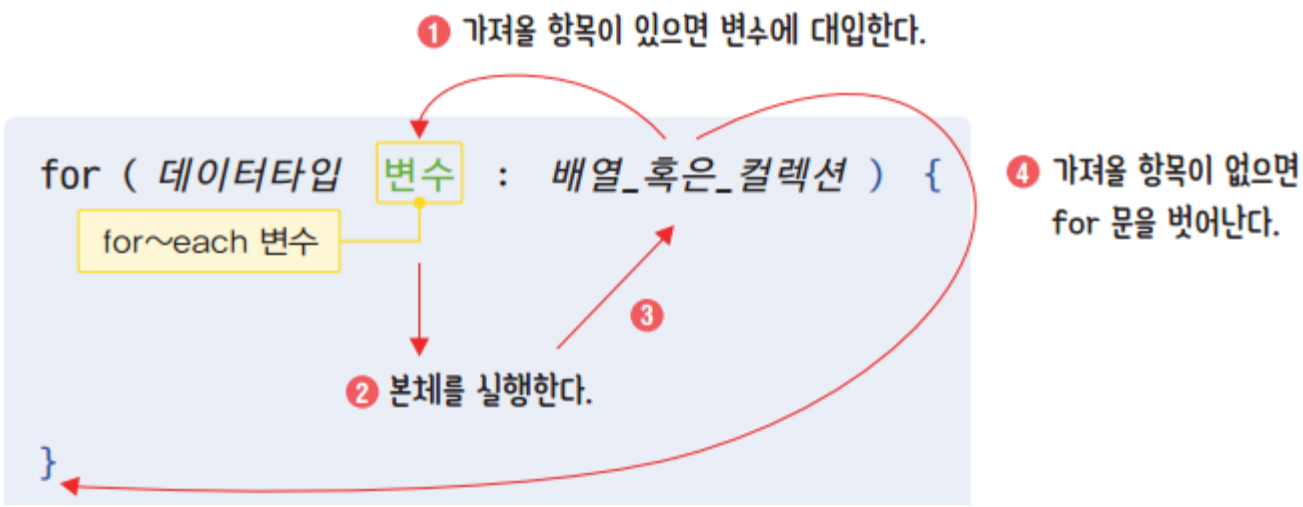
Practice

- 예제 5-7: ArrayList 실습
  - chap05.ArrayListDemo

배열 응용

for each 반복문

- for 문을 개선한 방식.
- 배열의 모든 원소를 순서대로 탐색.
- 특정 원소를 나타내기 위한 인덱스를 사용하지 않는다.



배열 응용

for each 반복문

```
20  public class ForEachDemo {
21      public static void main(String[] args) {
22          int[] one2five = {0, 1, 2, 3, 4};
23          int sum = 0;
24
25          for (int x = 0; x < one2five.length; x++)
26              one2five[x]++;
27
28          for (int x : one2five)
29              sum += x;
30
31          System.out.println("평균 = " + sum / 5.0);
32      }
33  }
```

final int x로 변경하면 x++에서 오류가 발생한다.

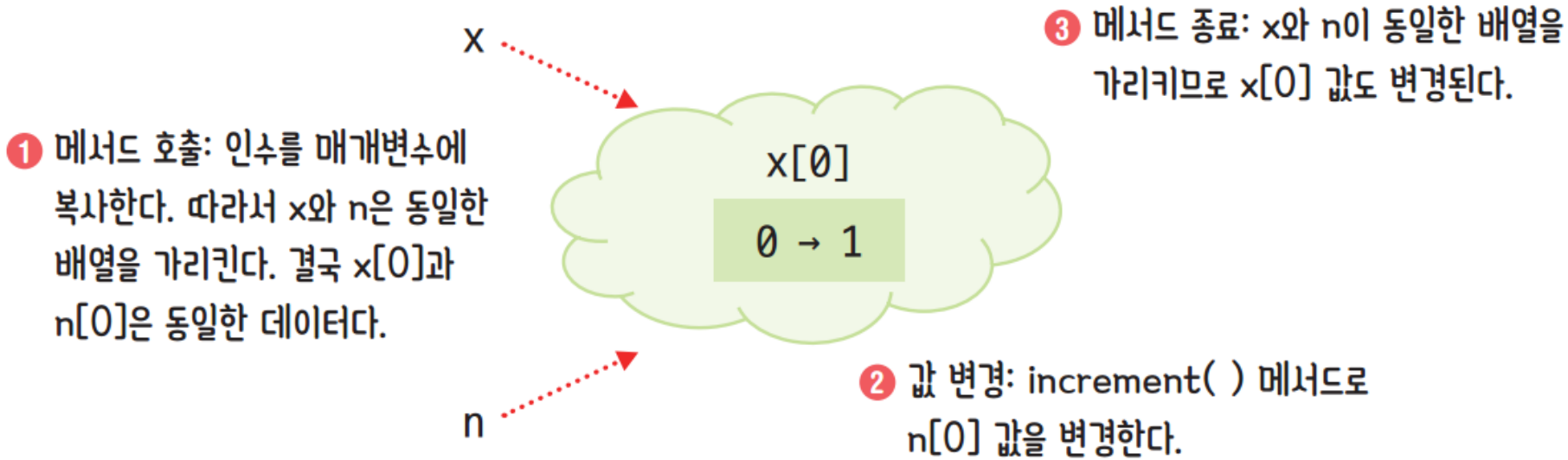
for 문은 특정 원소에 접근할 수 있지만 for-each 문은 할 수 없다.

final이 없어도 변수 x는 final int 타입이다.

배열 응용

메서드 인수로 배열 전달

- 배열은 참조 타입이므로, 배열 변수는 배열의 주소를 저장
- 배열 변수를 인수로 전달할 때 주소가 복사되어 전달
- 매서드 매개변수는 동일한 배열 객체를 가리키게 됨



배열

Practice

- 예제 5-10: 메서드 인수로 배열 전달
  - chap05.IncrementDemo

배열

가변 개수 매개변수

- 메서드에도 데이터 타입이 같은 가변 개수(variable length)의 인수를 전달 가능
- 한 개의 가변 개수 매개변수만 사용 가능하며 가변 개수 매개변수는 마지막에 위치
- 가변 개수 인수를 가진 메서드를 호출하면 내부적으로 배열을 생성하여 처리

```
public static void printSum(int... v) {
    int sum = 0;

    for (int i : v)
        sum += i;

    System.out.println(sum);
}
```

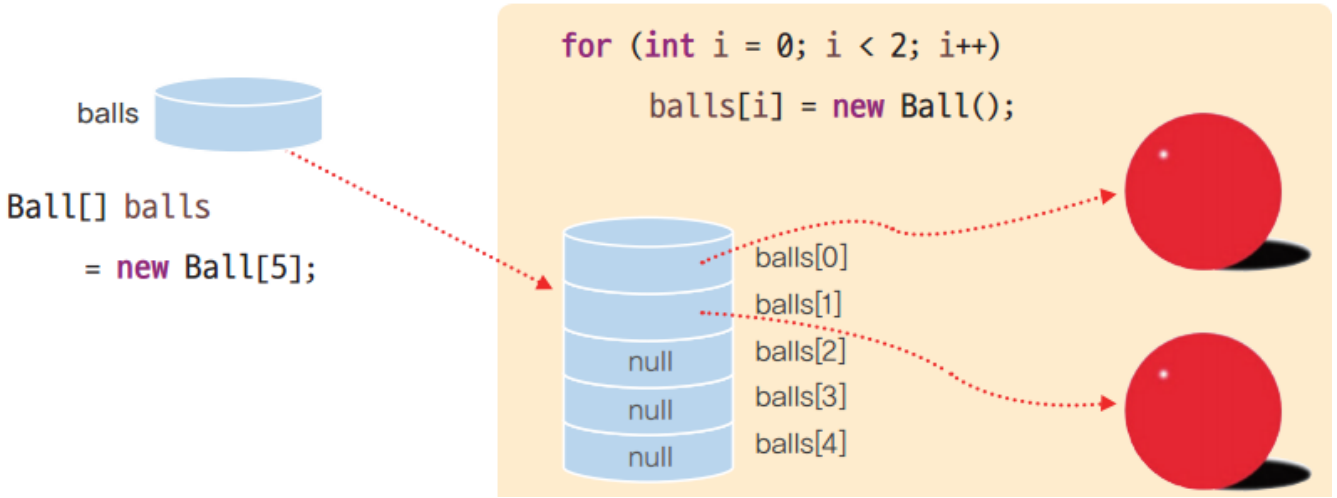
배열

객체의 배열

- 객체 배열은 객체를 참조하는 주소를 원소로 구성
- 예를 들어 Ball 클래스의 객체로 구성된 배열을 선언하고 초기화
- 초깃값은 null

```
Ball[] balls = new Ball[5];
```

5개의 Ball 객체를 생성하는 것이 아니라
5개의 Ball 객체를 참조할 변수를 준비



배열

Practice

- 예제 5-13: 객체의 배열 실습
  - chap05.CircleArrayDemo

- 예제 5-14: 객체 인수와 기초 타입 인수
  - chap05. ObjectArgumentDemo

열거 타입

열거 타입의 필요성

- 제한된 수의 일이나 사건 등에 대하여 숫자로 표현
  - 각 숫자에 대하여 부여된 의미를 개발자가 숙지 => 일이나 사건에 대한 경우의 수가 많다면 개발자 관점에서 불편
  - 부여되지 않은 의미 없는 숫자 => 컴파일러는 알 수 없다.
  - 출력 값이 의미 없는 숫자로 표현

- 제한된 사건에 대하여 숫자 대신에 상수를 정의해서 부여
  - 숫자에 부여된 의미를 개발자가 알 수 있지만, 여전히 나머지 문제가 미결

열거 타입

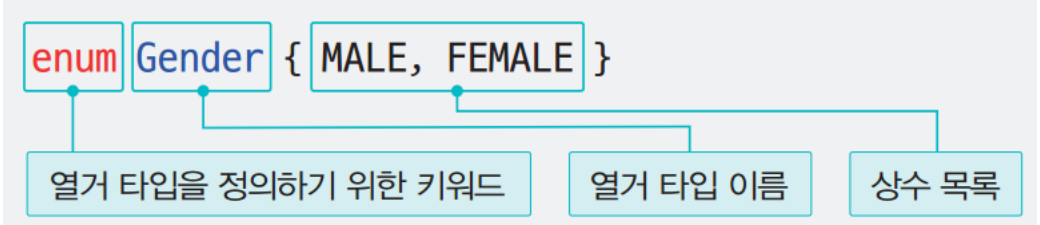
열거 타입

- 서로 연관된 사건들을 모아 상수로 정의한 java.lang.Enum클래스의 자식 클래스

- **선언**

```
enum 열거타입이름 { 상수목록 }
```

- **예시**



열거 타입

열거 타입과 응용

- 일종의 클래스 타입인 열거 타입 역시 생성자, 필드 및 메서드를 가질 수 있다.
- 열거 타입 상수는 생성자에 의한 인스턴스이다.
- 이때 생성자, 필드 및 메서드와 열거 타입 상수를 구분하기 위하여
  다음과 같이 열거 타입 상수 뒤에 반드시 세미콜론을 추가해야 한다.

```
enum 열거타입이름 {
    열거타입상수1, 열거타입상수2, . . . ;
    // 필드
    // 생성자
    // 메서드
}
```

필드, 생성자, 메서드가 있다면 서로 구분하는 데 필요하다.

필요하면 추가할 수 있는 선택사항이다.

열거 타입

Practice

- 예제 5-16: 열거 타입
  - chap05.EnumDemo

- 예제 5-17: 열거 타입 생성자, 메서드
  - chap05.EnumDemo

- 예제 5-18: switch에서 활용
  - chap05.SwitchDemo

Q & A