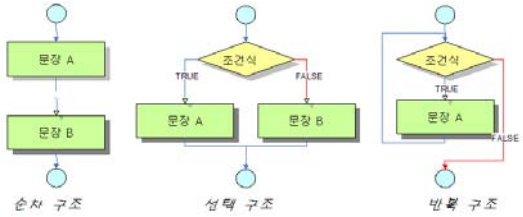
# 자바 기초문법
# 선택, 반복, 배열

514760-1
2016년 가을학기
9/22/2016
박경신

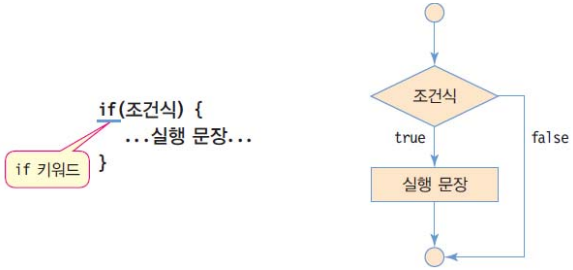
## Control Statement

- 제어문의 종류
  - 제어문이란 프로그램을 실행할 때는 논리적인 흐름이 필요한데, 이러한 문장의 논리적인 흐름을 통제해 주는것.
  - 조건문 - 조건식의 값에 따라 각각에 해당되는 명령문을 수행한다. 예) if 문, switch 문
  - 반복문 – 조건이 만족하는 동안 특정 명령문을 반복적으로 수행한다. 예) while 문, do 문, for 문, foreach 문
  - 점프문 – 제어권을 이동시킬 때 점프문을 사용한다. 예) label 문, break 문, continue 문




## If Statement

- 단순 if 문
  - if 다음의 괄호 안에는 조건식(논리형 변수나 논리 연산)
  - 조건식의 값
    - true인 경우, if문을 벗어나 다음 문장이 실행된다.
    - false의 경우에는 if 다음의 문장이 실행되지 않고 if 문을 빠져 나온다.
  - 실행문장이 단일 문장인 경우 둘러싸는 {, } 생략 가능




## Ex : 짝수 홀수 구별하기


*EvenOdd.java*

```
01 import java.util.Scanner;
02
03 public class EvenOdd {
04    public static void main(String[] args) {
05       // if 문을 사용하여 홀수와 짝수를 구별하는 프로그램
06       int number;
07       Scanner input = new Scanner(System.in);
08       System.out.print("정수를 입력하시오: ");
09       number = input.nextInt();
11       if (number % 2 == 0) {
12          System.out.println("입력된 정수는 짝수입니다.");
13       } else {
14          System.out.println("입력된 정수는 홀수입니다.");
15       }
16
17       System.out.println("프로그램이 종료되었습니다. ");
18    }
19 }
```

실행결과

```
정수를 입력하시오: 2
입력된 정수는 짝수입니다.
프로그램이 종료되었습니다.
```

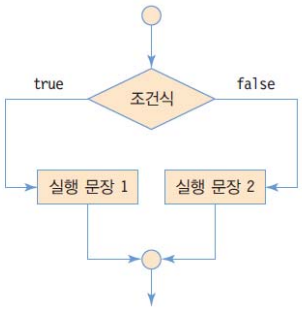
## If-Else Statement

- if-else 문
  - 조건식이 true면 실행문장1 실행 후 if-else문을 벗어남
  - false인 경우에 실행문장2 실행후, if-else문을 벗어남

```
if(조건식) {
    ...실행 문장 1...
}
else {
    ...실행 문장 2...
}
```

if 키워드

else 키워드




## Ex : 두수 중 큰 수 출력하기

Larger.java

```
01 import java.util.Scanner;
02
03 public class Larger {
04    public static void main(String[] args) {
05       int x, y, max;
06
07       Scanner input = new Scanner(System.in);
08       System.out.print("첫번째 정수: ");
09       x = input.nextInt();
10
11       System.out.print("두번째 정수: ");
12       y = input.nextInt();
13
14       if (x > y)
15          max = x;
16       else
17          max = y;
18
19       System.out.println("큰 수는 " + max);
20    }
21 }
```

조건에 따라 실행하는 문장이 하나이면 중괄호를 생략할 수 있다.

```
첫번째 정수: 10
두번째 정수: 20
큰 수는 20
```

## Ex : 성과급 계산

Bonus.java

```
01 import java.util.Scanner;
02
03 public class Bonus {
04    public static void main(String args[]) {
05       final int targetSales = 1000;
06       int mySales;
07       int bonus;
08       String result;
09
10       Scanner input = new Scanner(System.in);
11       System.out.print("실적을 입력하시오(단위: 만원): ");
12       mySales = input.nextInt();
13
14       if (mySales >= targetSales) {
15          result = "실적 달성";
16          bonus = (mySales - targetSales) / 10;
17       } else {
18          result = "실적 달성 못함";
19          bonus = 0;
20       }
21
22       System.out.println(result + "\n" + "보너스: " + bonus);
23    }
24 }
```

실적 목표가 상수로 정의되었다.

```
실적을 입력하시오(단위: 만원): 1200
실적 달성
보너스: 20
```

## If-Else Statement

- 다중 if-else 문
  - 실행문장이 다시 if문 또는 if-else문을 포함
  - else 문은 바로 전의 if문과 짝을 이룬다.
  - 조건문이 너무 많은 경우, switch 문 사용 권장

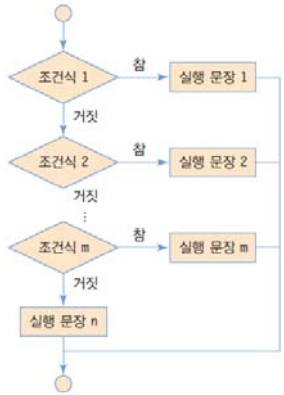
```
if(조건식 1) {
    실행 문장 1; // 조건식 1이 참인 경우
}
else if(조건식 2) {
    실행 문장 2; // 조건식 2가 참인 경우
}
else if(조건식 m) {
    ..... // 조건식 m이 참인 경우
}
else {
    실행 문장 n; // 앞의 모든 조건이 거짓인 경우
}
```

if 키워드

else if 키워드

else if 키워드

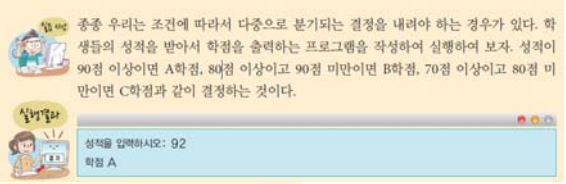
else 키워드



## Ex : 입력한 값에 따른 학점 분류

*Grading.java*

```
01 import java.util.Scanner;
02
03 public class Grading {
04    public static void main(String[] a
05       int score;
06
07       Scanner input = new Scanner(System.in);
08       System.out.print("성적을 입력하시오: ");
09       score = input.nextInt();
10       if (score >= 90)            // ①
11          System.out.println("학점 A");
12       else if (score >= 80)       // ②
13          System.out.println("학점 B");
14       else if (score >= 70)
15          System.out.println("학점 C");
16       else if (score >= 60)
17          System.out.println("학점 D");
18       else
19          System.out.println("학점 F");
20
21    }
22 }
```



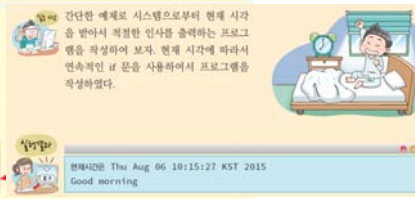

문장 ②에서 if( grade >= 80 && grade < 90)이라고 할 필요가 없음에 유의하라. grade가 90보다 크거나 같은 경우에는 앞의 문장 ①에서 이미 걸렀기 때문이다.

## Ex : 시간에 따라 인사말 출력

해답

*Welcome.java*

```
01 import java.util.Date;
02
03 public class Welcome {
04    public static void main(String args[])
05
06       Date date = new Date();
07       int currentHour = date.getHours();
08
09       System.out.println("현재시간은 "+ date);
10       if (currentHour < 11) {
11          System.out.println("Good morning");
12       } else if (currentHour < 15) {
13          System.out.println("Good afternoon");
14       } else if (currentHour < 20) {
15          System.out.println("Good evening");
16       } else {
17          System.out.println("Good night");
18       }
19    }
20 }
```




## Ternary Conditional Operator ?:

- 조건 연산자 ?:는 if-else로 바꿀 수 있음

```
i = a>b?a-b:b-a;
```

⇨

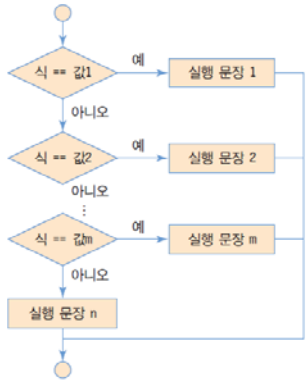
```
if (a>b)
      i = a – b;
else
      i = b – a;
```

## Switch Statement

- switch문은 식과 case 문의 값과 비교
  - case의 비교 값과 일치하면 해당 case문의 실행문장 수행
    - break를 만나면 switch문을 벗어남
  - case의 비교 값과 일치하는 것이 없으면 default 문 실행
- default문은 생략 가능

```
switch(식) {
   case 값1:
      실행 문장 1;
      break;
   case 값2:
      실행 문장 2;
      break;
   ...
   case 값m:
      실행 문장 m;
      break;
   default:
      실행 문장 n;
}
```

switch 키워드 / case 키워드 / break 키워드 / default 키워드



## Switch Statement

- switch문 내의 break문
  - break 문장을 만나면 switch문을 벗어남
  - 만일 case 문에 break문이 없다면, 다음 case문의 실행 문장으로 실행을 계속하게 되며, 언젠가 break를 만날 때까지 계속 내려감

```
char grade='A';
switch (grade) {
  case 'A':
    System.out.println("90 ~ 100점입니다.“);
    //break;
  case 'B':
    System.out.println("80 ~ 89점입니다.“);
    break;
  case 'C':
    System.out.println("70 ~ 79점입니다.“);
    break;
}
```

```
90 ~ 100점입니다.
80 ~ 89점입니다.
```

## Ex : 입력한 점수에 따른 학점 출력

Score2Grade.java

```
01 import java.util.*;
02
03 public class Score2Grade {
04   public static void main(String[] args) {
05     int score, number;
06     char grade;
07
08     Scanner scan = new Scanner(System.in);
09     System.out.print("성적을 입력하시오: ");
10     score = scan.nextInt();
11     number = score / 10;
12     switch (number) {
13     case 10:
14     case 9:
15       grade = 'A';
16       break;
17     case 8:
18       grade = 'B';
19       break;
20     case 7:
21       grade = 'C';
22       break;
23     case 6:
24       grade = 'D';
25       break;
26     default:
27       grade = 'F';
28       break;
29     }
30     System.out.print("학점: " + grade);
31   }
32 }
```

```
성적을 입력하시오: 92
학점: A
```

## Switch Statement

- case 문의 값의 특징
  - switch 문은 식의 결과 값을 case 문과 비교
  - 사용 가능한 case문의 비교 값
    - 정수 타입 리터럴, JDK 1.7부터는 문자열 리터럴도 허용

```
int a = 0;
int b = 1;
int c = 25;
switch(c%2) {
  case 1 : // 정수 리터럴
    ...;
    break;
  case 2: // 정수 리터럴
}
```

```
String s = "예“;
switch(s) {
  case "예" : // 문자열 리터럴
    ...;
    break;
  case "아니요“ : // 문자열 리터럴
    ...;
    break;
}
```

## Switch Statement

- 잘못된 case 문

```
switch(a) {
  case a :        // 오류. 변수 사용 안됨
  case a > 3 :    // 오류. 수식 안됨
  case a == 1 :   // 오류. 수식 안됨
}
```

## Ex : 월의 일수 출력


*StringSwitch.java*

```
01 import java.util.Scanner;
02
03 public class StringSwitch {
04    public static void main(String[] args) {
05       String month;
06
07       Scanner scan = new Scanner(System.in);
08       System.out.print("월의 이름을 입력하시오: ");
09       month = scan.next();
10
11       int monthNumber;
12       switch (month)
13       case "january":
14          monthNumber = 1;
15          break;
16       case "february":
17          monthNumber = 2;
18          break;
```



```
일수를 알고 싶은 월을 입력하시오:12
월의 날수는 31
```

```
일수를 알고 싶은 월을 입력하시오:6
월의 날수는 30
```

## Ex : 월의 일수 출력


*DaysInMoth.java*

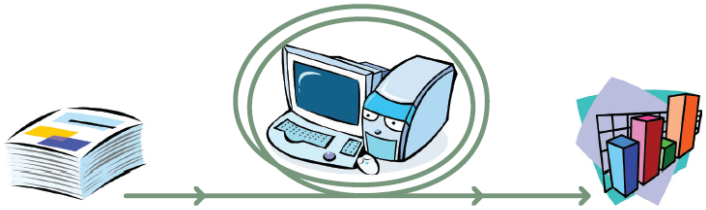
```
01 import java.util.*;
02
03 public class DaysInMonth {
04    public static void main(String[] args) {
05       int month;
06       int days = 0;
07
08       System.out.print("일수를 알고 싶은 월을 입력하시오:");
09       Scanner scan = new Scanner(System.in);
10
11       month = scan.nextInt();
12       switch (month) {
13       case 4:
14       case 6:
15       case 9:
16       case 11:
17          days = 30;
18          break;
19       case 2:
20          days = 28;
21          break;
22       default:
23          days = 31;
24          break;
25       }
26       System.out.println("월의 날수는 " + days);
27    }
28 }
```

고의적으로 break문을 생략하여서 여러 가지 경우를 동일한 문장으로 처리하고 있다.

## Loop

### Q) 반복 구조는 왜 필요한가?

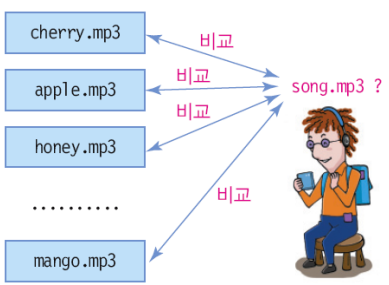
**A) 같은 처리 과정을 되풀이하는 것이 필요하기 때문이다. 학생 30명의 평균 성적을 구하려면 같은 과정을 30번 반복하여야 한다.**

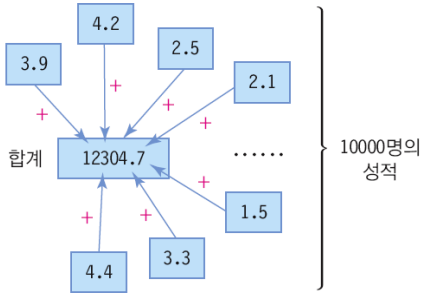


## Loop



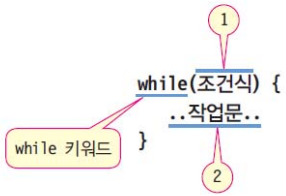
- 자바 반복문의 종류
  - while 문
  - for 문
  - do while 문

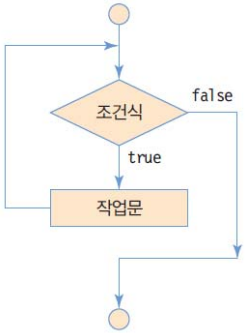


## While Statement

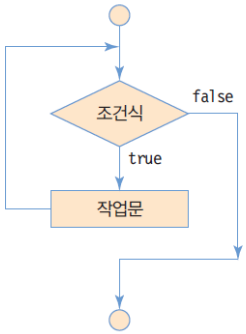
- 반복 조건이 true이면 반복, false 이면 반복 종료
- 반복 조건이 없으면 컴파일 오류
- 처음부터 반복 조건을 통과한 후 작업문 수행

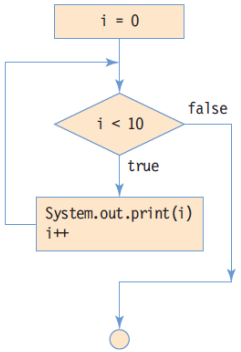



## While Statement

- While 문의 실행과정을 나타내는 순서도




```
i = 0;
while(i<10) {
    System.out.print(i);
    i++;
}
```

## Ex : 환영 메시지 출력


*WelcomeLoop.java*

```
01 public class WelcomeLoop {
02    public static void main(String[] args) {
03       int i = 0;
04       while (i < 5) {
05          System.out.println("환영합니다. ");
06          i++;
07       }
08       System.out.println("반복이 종료되었습니다.");
09    }
10 }
```


```
환영합니다.
환영합니다.
환영합니다.
환영합니다.
환영합니다.
반복이 종료되었습니다.
```

## Ex : 0~9 까지 출력


```
0 1 2 3 4 5 6 7 8 9
```


*LoopExmaple1.java*

```
01 public class LoopExample1 {
02    public static void main(String[] args) {
03       int i = 0;
04       while (i < 10) {
05          System.out.print(i+" ");
06          i++;
07       }
08    }
09 }
```

## Ex : 입력된 수의 평균 구하기

while문을 이용하여 키보드에서 숫자를 입력 받아 입력 받은 모든 수의 평균을 출력하는 프로그램을 작성해보자.
0이 입력되면 입력이 종료되고 평균을 구하여 출력한다.

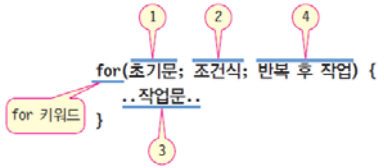
```
import java.util.Scanner;
public class WhileSample {
  public static void main (String[] args) {
    Scanner rd = new Scanner(System.in);
    int n = 0;
    double sum = 0;
    int i=0;
    while ((i = rd.nextInt()) != 0) {
      sum += i;
      n++;
    }
    System.out.println("입력된 수의 개수는 " + n
  + "개이며 평균은 " + sum / n + "입니다.");
  }
}
```

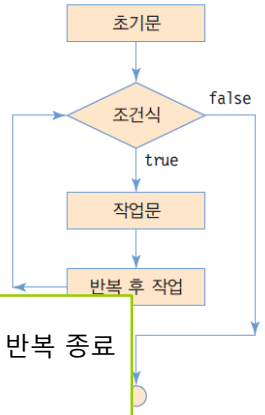
```
10
20
30
40
0
```

마지막 입력

입력된 수의 개수는 4개이며 평균은 25.0입니다.

## For Statement



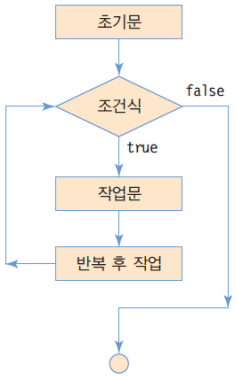

❶
- for 문이 실행한 후 오직 한번만 실행되는
- 콤마(',')로 구분하여 여러 문장 나열 가능
- 초기화할 일 없으면 비어둘 수 있음

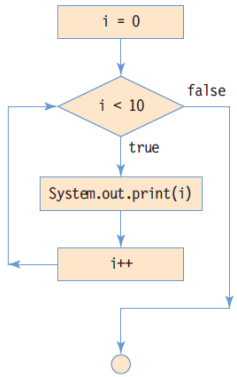
❷
- 논리형 변수나 논리 연산만 가능
- 반복 조건이 true이면 반복 계속, false이면 반복 종료
- 반복 조건이 true 상수인 경우, 무한 반복
- 반복 조건이 비어 있으면 true로 간주

❹
- 반복 작업 문장들의 실행 후 처리 작업
- 콤마(',')로 구분하여 여러 문장 나열 가능

## For Statement

- For 문의 실행 과정을 나타내는 순서도
  - 주로 정해진 횟수만큼 반복할 때 사용됨
  - 또는 정해진 범위를 반복할 때 활용됨




```
for(i=0; i<10; i++) {
    System.out.print(i);
}
```

## For Statement

- 0~9 까지 정수 출력

```
int i;
for(i = 0; i < 10; i++) {
  System.out.print(i);
}
```

```
int i;
for(i = 0; i < 10; i++)
  System.out.print(i);
```

- For-loop 안에 변수 선언 가능

```
for(int i = 0; i < 10; i++) // 변수 i는 for문을 벗어나서 사용할 수 없음
  System.out.print(i);
```

- 0~100까지 합 계산

```
int sum = 0;
for(int i = 0; i <= 100; i++)
  sum += i;
```

```
int sum;
for(int i = 0, sum=0; i <= 100; i++)
  sum += i;
```

```
int sum = 0;
for(int i = 100; i >= 0; i--)
  sum += i;
```

## For Statement

```
for(초기작업; true; 반복후작업) { // 반복 조건이 true이면 무한 반복
.....
}
```

```
for(초기작업; ; 반복후작업) { // 반복조건이 비어 있으면 true로 간주, 무한 반복
.....
}
```

```
// 초기 작업과 반복후작업은 ','로 분리하여 여러 문장 나열 가능
for(i=0; i<10; i++, System.out.println(i)) {
.....
}
```

```
// for문 내에 변수 선언
for(int i=0; i<10; i++) { // 변수 i는 for문 내에서만 사용 가능
.....
}
```

## Ex : 1부터 10까지 숫자의 합을 출력


```
1부터 10까지의 정수의 합 = 55
```


*Sum.java*

```
01 public class Sum {
02    public static void main(String[] args) {
03       int sum = 0;
04
05       for (int i = 1; i<= 10; i++)
06          sum += i;
07
08       System.out.printf("1부터 10까지의 정수의 합 = %d", sum);
09
10    }
11 }
```

for 문 안의 초기식에서 제어 변수를 선언할 수 있다.

## Ex : 1부터 10까지 숫자의 합을 출력

for문을 이용하여 1부터 10까지 덧셈을 표시하고 합을 구하시오.

```
public class ForSample {
  public static void main (String[] args) {
    int i, j;

    for (j=0,i=1; i <= 10; i++) {
      j += i;
      System.out.print(i);
      if(i==10) {
        System.out.print("=");
        System.out.print(j);
      }
      else
        System.out.print("+");
    }
  }
}
```

1+2+3+4+5+6+7+8+9+10=55

## Ex : Factorial 계산

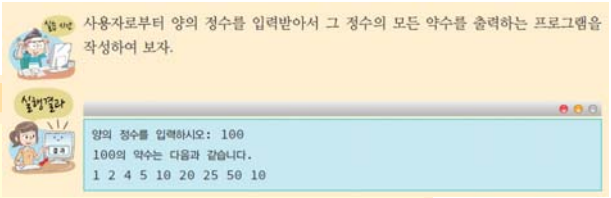

*Factorial.java*

```
01 import java.util.*;
02
03 public class Factorial {
04    public static void main(String[] args) {
05       long fact = 1;
06       int n;
07
08       System.out.printf("정수를 입력하시요:");
09       Scanner scan = new Scanner(System.in);
10       n = scan.nextInt();
11
12       for (int i = 1; i <= n; i++)
13          fact = fact * i;
14
15       System.out.printf("%d!은 %d입니다.", n, fact);
16
17    }
18 }
```

먼저 변수 fact를 long형으로 정의한다. 팩토리얼의 값은 생각보다 아주 커질 수 있다. 여기서 fact의 초기값은 반드시 1이어야 한다. 0이면 안 된다. 왜냐하면 팩토리얼은 정수를 전부 곱해서 계산하는 것이므로 초기값이 0이면 결과는 0이 되어 버린다. 따라서 반드시 1로 초기화를 시켜야 한다.


```
정수를 입력하시요:20
20!은 2432902008176640000입니다.
```

# Ex : 약수 계산



사용자로부터 양의 정수를 입력받아서 그 정수의 모든 약수를 출력하는 프로그램을 작성하여 보자.

*Divisor.java*

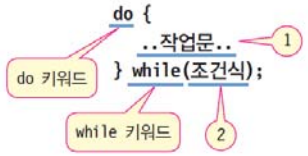
```
01 import java.util.Scanner;
02
03 public class Divisor {
04    public static void main(String[] args) {
05       Scanner scan = new Scanner(System.in);
06       System.out.print("양의 정수를 입력하시오: ");
07       int n = scan.nextInt();
08
09       System.out.println(n + "의 약수는 다음과 같습니다.");
10       for (int i = 1; i <= n; ++i) {
11          if (n % i == 0)
12             System.out.print(" " + i);
13       }
14    }
15 }
```

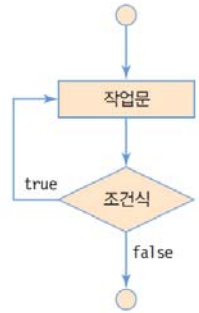
실행결과

```
양의 정수를 입력하시오: 100
100의 약수는 다음과 같습니다.
1 2 4 5 10 20 25 50 10
```

# Do-While Statement





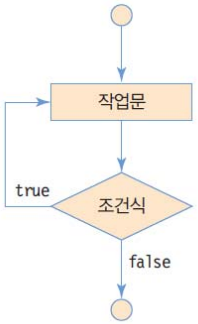



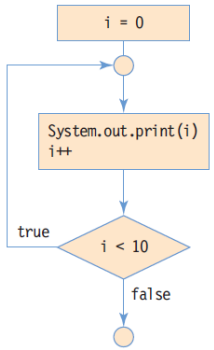

❶ • 무조건 최소 한번은 실행

❷ • 반복 조건이 true이면 반복, false이면 반복 종료
• 반복 조건이 없으며 컴파일 오류

# Do-While Statement

- do-while 문의 실행 과정을 나타내는 순서도




```
i = 0;
do {
    System.out.print(i);
    i++;
} while(i<10);
```

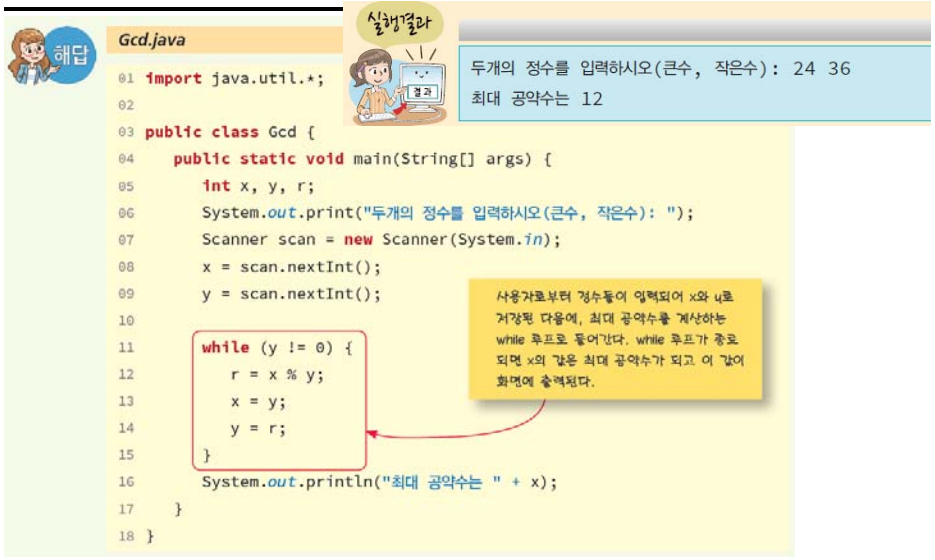
# Ex : 올바른 월 입력


*CheckInput.java*

```
01 import java.util.Scanner;
02
03 public class CheckInput {
04    public static void main(String args[]) {
05       Scanner input = new Scanner(System.in);
06       int month;
07       do {
08          System.out.print("올바른 월을 입력하시오 [1-12]: ");
09          month = input.nextInt();
10       } while (month < 1 || month > 12);
11    }
12    System.out.println("사용자가 입력한 월은 " + month);
13 }
```


```
올바른 월을 입력하시오 [1-12]: 13
올바른 월을 입력하시오 [1-12]: 14
올바른 월을 입력하시오 [1-12]: 0
올바른 월을 입력하시오 [1-12]: 1
사용자가 입력한 월은 1
```

## Ex : 최대공약수 출력




## Do-while-loop : a-z까지 출력

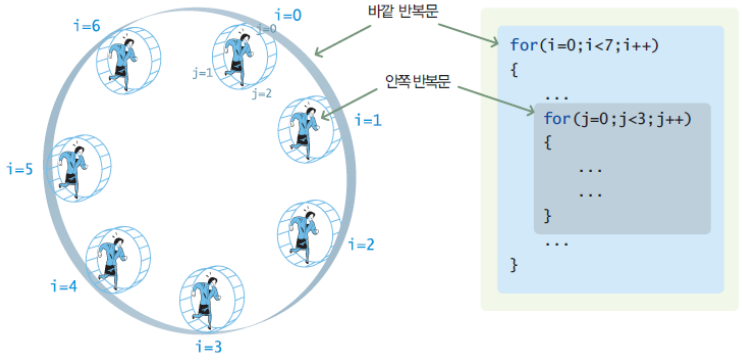
do-while문을 이용하여 'a'부터 'z'까지 출력하는 프로그램을 작성하시오.

```
public class DoWhileSample {
  public static void main (String[] args) {
    char a = 'a';

    do {
      System.out.print(a);
      a = (char) (a + 1);
    } while (a <= 'z');
  }
}
```

```
abcdefghijklmnopqrstuvwxyz
```

## Nested Loop

- 중첩 반복문(nested loop)은 반복문 안에 다른 반복문이 위치




## Nested Loop

- 중첩 반복
  - 반복문이 다른 반복문을 내포하는 구조
  - 이론적으로는 몇 번이고 중첩 반복 가능
  - 너무 많은 중첩 반복은 프로그램 구조를 복잡하게 하므로 2중 또는 3중 반복이 적당

```
for(i=0; i<100; i++) { // 100 개의 학교 성적을 모두 더한다.
    .....
    for(j=0; j<10000; j++) { // 10000 명의 학생
    성적을 모두 더한다.
        .....
        .....
    }
  .....
}
```

10000명의 학생이 있는 100개 대학의 모든 학생 성적의 합을 구할 때, for 문을 이용한 이중 중첩 구조

## Nested-Loop: 구구단

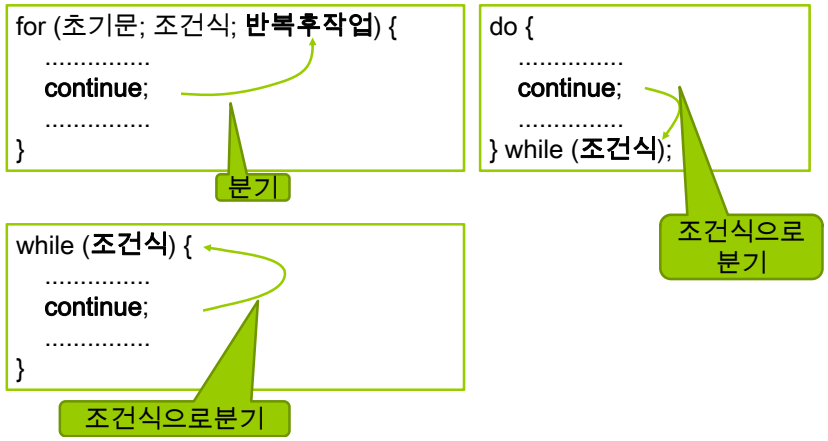
2중 중첩된 for문을 사용하여 구구단을 출력하는 프로그램을 작성하시오. 한 줄에 한 단씩 출력한다.

```
public class NestedLoop {
  public static void main (String[] args) {
    int i, j;

    for (i = 1; i < 10; i++,System.out.println()) {
      for (j = 1; j < 10; j++,System.out.print('\t')) {
        System.out.print(i + "*" + j + "=" + i*j);
      }
    }
  }
}
```

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

## Continue Statement

- continue 문
  - 반복문을 빠져 나가지 않으면서
  - 반복문 실행 도중 다음 반복을 진행




## Ex: 1부터 100까지 짝수의 합

for와 continue문을 사용하여 1부터 100까지 짝수의 합을 구해보자.

```
public class ContinueExample {
  public static void main (String[] args) {
    int sum = 0;
    for (int i = 1; i <= 100; i++) {
      if (i%2 == 1)
        continue;
      else
        sum += i;
    }
    System.out.println("1부터 100까지 짝수의 합은 " + sum);
  }
}
```

```
1부터 100까지 짝수의 합은 2550
```

## Break Statement

- break 문
  - 반복문 하나를 완전히 빠져 나갈 때 사용
  - break문은 하나의 반복문만 벗어남
    - 중첩 반복의 경우 안쪽 반복문의 break 문이 실행되면 안쪽 반복문만 벗어남

## Ex: 입력된 숫자 개수

while문과 break문을 사용하여 -1이 입력될 때까지 입력을 받고, 입력된 숫자의 개수를 출력하시오.

```
import java.util.Scanner;
public class BreakExample {
  public static void main (String[] args) {
    Scanner in = new Scanner(System.in);
    int num = 0;

    while (true) {
      if (in.nextInt() == -1)
        break;
      num++;
    }
    System.out.println("입력된 숫자 개수는 " + num);
  }
}
```

```
10
8
9
5
-1
입력된 숫자 개수는 4
```

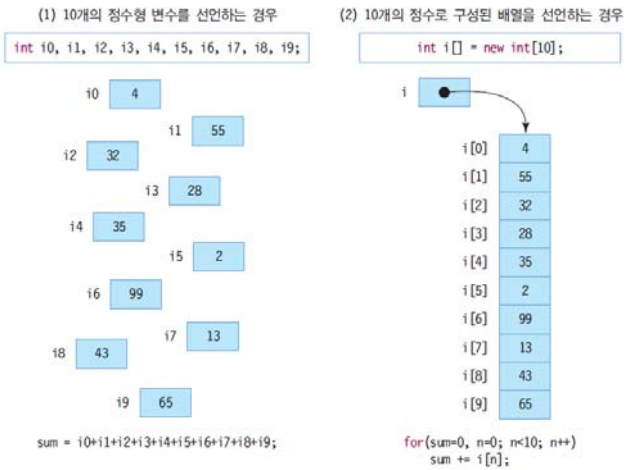
마지막 입력

## Array

- 배열(array)
  - 여러 개의 데이터를 같은 이름으로 활용할 수 있도록 해주는 자료 구조
    - 인덱스(Index, 순서 번호)와 인덱스에 대응하는 데이터들로 이루어진 자료 구조
    - 배열을 이용하면 한 번에 많은 메모리 공간 선언 가능
  - 배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간
    - 원소 데이터들이 순차적으로 저장됨
    - 인덱스를 이용하여 원소 데이터 접근
    - 반복문을 이용하여 처리하기에 적합한 자료 구조(주로 for 또는 for-each 반복문과 많이 사용됨)
  - 배열 인덱스
    - 0부터 시작
    - 인덱스는 배열의 시작 위치에서부터 데이터가 있는 상대 위치

## Array

- 자바 배열의 필요성과 모양




## Array

- 배열 선언과 배열 생성의 두 단계 필요
  - 배열 선언

```
int     intArray[];
char    charArray[];
```

또는

```
int[]   intArray;
char[]  charArray;
```

  - 배열 생성

```
intArray = new int[10];
charArray = new char[20];
```

또는

```
int intArray[] = new int[10];
char charArray[] = new char[20];
```

  - 선언과 초기화
    - 배열 생성과 값 초기화

```
// 총 10개의 정수 배열 생성 및 값 초기화
int intArray[] = {0,1,2,3,4,5,6,7,8,9};
```

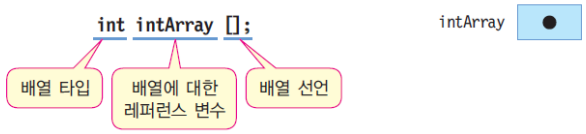
  - 잘못된 배열 선언

```
//int intArray[10]; // 컴파일 오류. 배열의 크기를 지정할 수 없음
```

# Array

## 배열 선언과 생성

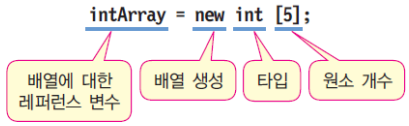
(1) 배열에 대한 레퍼런스 변수 intArray 선언

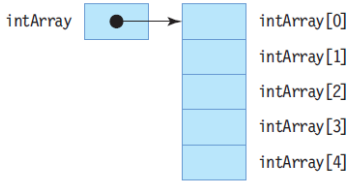






(2) 배열 생성

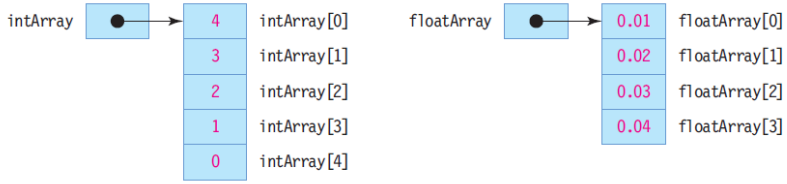



# Array

## 배열을 초기화하면서 생성한 결과

```
int intArray[] = {4, 3, 2, 1, 0};
float floatArray[] = {0.01, 0.02, 0.03, 0.04};
```

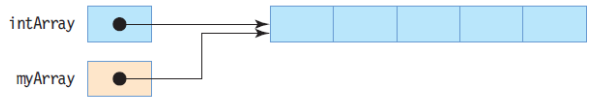



# Array

## 배열 참조

- 생성된 1개의 배열을 다수의 레퍼런스가 참조 가능

```
int intArray[] = new int[5];
int myArray[] = intArray;
```




```
intArray[1] = 2;
myArray[1] = 6;
```


# Array

## 배열 원소 접근

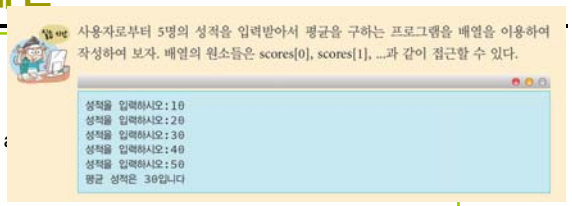
- 반드시 배열 생성 후 접근

```
int intArray []; // 배열 선언
intArray[4] = 8; // 오류, intArray 배열의 메모리가 할당되지 않았음
```

- 배열 변수명과 [] 사이에 원소의 인덱스를 적어 접근
  - 배열의 인덱스는 0부터 시작
  - 배열의 마지막 항목의 인덱스는 (배열 크기 – 1)

```
int[] intArray;
intArray = new int[10];

intArray[3]=6; // 배열에 값을 저장
int n = intArray[3]; // 배열로부터 값을 읽음
```

## Ex : 성적 평균 계산




```
import java.util.Scanner;

public class ArrayTest2 {
      public static void main(String[] a
          final int STUDENTS = 5;
          int total = 0;
          Scanner scan = new Scanner(System.in);
          int[] scores = new int[STUDENTS];

          for (int i = 0; i < scores.length; i++) {
                System.out.print("성적을 입력하시오:");
                scores[i] = scan.nextInt();
          }
          for (int i = 0; i < scores.length; i++)
                total += scores[i];
          System.out.println("평균 성적은" + total / STUDENTS + "입니다");
      }
}
```

## Ex : 입력 받은 수 중 최댓값 구하기

양수 5개를 입력 받아 배열에 저장하고, 제일 큰 수를 화면에 출력하는 프로그램을 작성하시오.

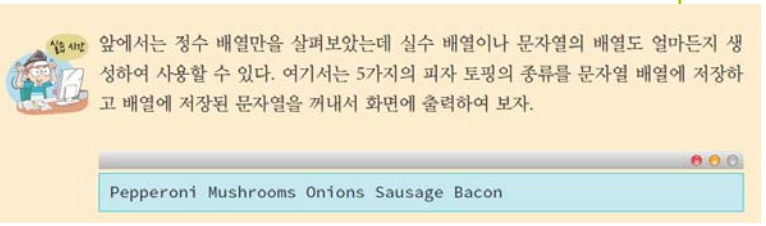
```
import java.util.Scanner;
public class ArrayAccess {
  public static void main (String[] args) {
    Scanner in = new Scanner(System.in);
    int intArray[] = new int[5];
    int max = 0;
    for (int i = 0; i < 5; i++) {
      System.out.print( " 숫자를 입력하
      intArray[i] = in.nextInt();
      if (intArray[i] > max)
        max = intArray[i];
    }
    System.out.print("입력된 수에서 가장 큰 수는 " + max + "입니다.");
  }
}
```

```
숫자를 입력하시오:1
숫자를 입력하시오:39
숫자를 입력하시오:78
숫자를 입력하시오:100
숫자를 입력하시오:99
입력된 수에서 가장 큰 수는
100입니다.
```

## Ex : 피자 토핑 출력

- 문자열 배열

```
public class PizzaTopping {
      public static void main(String[] args) {

          String[] toppings = { "Pepperoni", "Mushrooms", "Onions",
"Sausage", "Bacon" };

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




## Array

- 배열 인덱스
  - 인덱스는 0부터 시작하며 마지막 인덱스는 (배열 크기 -1)
  - 인덱스는 정수 타입만 가능

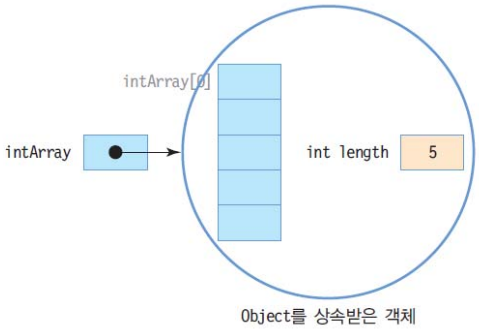
```
int intArray [] = new int[5]; // 인덱스는 0~4까지 가능
int n = intArray[-2];  // 실행 오류. -2는 인덱스로 적합하지 않음
int m = intArray[5];  // 실행 오류. 5는 인덱스의 범위(0~4)를 넘었음
```

- 배열의 크기
  - 배열의 크기는 배열 레퍼런스 변수를 선언할 때 결정되지 않음
    - 배열의 크기는 배열 생성 시에 결정되며, 나중에 바꿀 수 없음
  - 배열의 크기는 배열의 length 필드에 저장

```
int size = intArray.length;
```

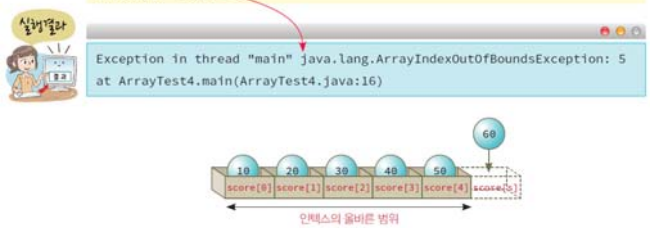
## Array

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

int size = intArray.length;
// size는 5
```




## Ex : 범위를 벗어난 배열의 접근

```
int[] scores = new int[5];
scores[0] = 10;
scores[1] = 20;
scores[2] = 30;
scores[3] = 40;
scores[4] = 50;
scores[5] = 60;
```



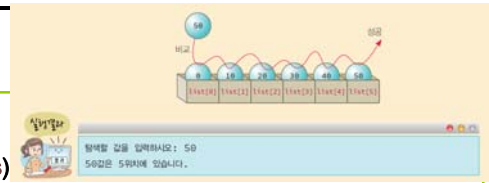

## Ex : 배열 원소의 평균 계산

배열의 length 필드를 이용하여 배열 크기만큼 키보드에서 정수를 입력 받고 평균을 구하는 프로그램을 작성하시오.

```
import java.util.Scanner;
public class ArrayLength {
  public static void main (String[] args) {
    Scanner in = new Scanner(System.in);
    int intArray[] = new int[5];
    double sum = 0;

    for (int i = 0; i < intArray.length; i++)
      intArray[i] = in.nextInt();

    for (int i = 0; i < intArray.length; i++)
      sum += intArray[i];
    System.out.print("배열 원소의 평균은 " + sum/intArray.length + "이다.");
  }
}
```

```
10
20
30
40
50
배열 원소의 평균은 30.0입니다.
```

## Ex : 순차 탐색 (Sequential Search)




```
import java.util.Scanner;
public class SeqSearch {
      public static void main(String[] args)
            int s[] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
            int value, index = -1;
            Scanner scan = new Scanner(System.in);
            System.out.print("탐색할 값을 입력하시오: ");
            value = scan.nextInt();
            for (int i = 0; i < s.length; i++) {
                  if (s[i] == value)
                        index = i;
            }
            if (index < s.length && index >= 0)
                  System.out.println("" + value + "값은 " + index + "위치에
있습니다.");
      }
}
```

## Array & For-each

- For-each 문
  - 배열이나 나열(enumeration)의 각 원소를 순차적으로 접근하는데 유용한 for 문

```
int[] num = { 1,2,3,4,5 };
int sum = 0;
// 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정
for (int k : num)
  sum += k;
System.out.println("합은 " + sum);
```

합은 15

```
String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" } ;
// 반복할 때마다 s는 names[0], names[1], ..., names[5] 로 설정
for (String s : names)
  System.out.print(s + " ");
```

사과 배 바나나 체리 딸기 포도

## 2D Array

- 2차원 배열 선언

```
int     intArray[][];
char    charArray[][];
float   floatArray[][];
```

또는

```
int[][]   intArray;
char[][]  charArray;
float[][] floatArray;
```

- 2차원 배열 생성

```
intArray = new int[2][5];
charArray = new char[5][5];
floatArray = new float[5][2];
```

또는

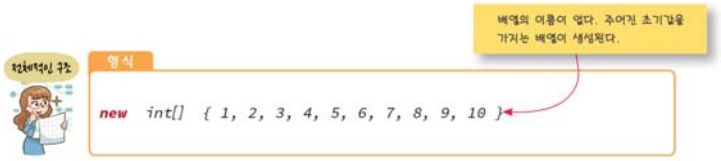
```
int     intArray[] = new int[2][5];
char    charArray[] = new char[5][5];
float   floatArray[] = new float[5][2];
```

- 2차원 배열 선언, 생성, 초기화

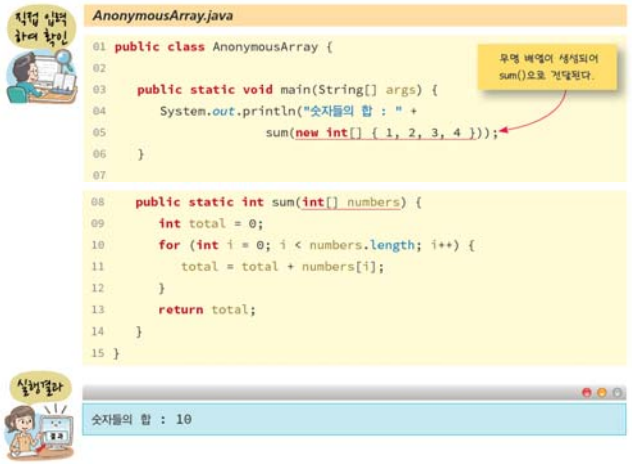
```
int intArray[][] = {{0,1,2},{3,4,5},{6,7,8}};
char charArray[][] = {{'a', 'b', 'c'},{'d', 'e', 'f'}};
float floatArray[][] = {{0.01, 0.02}, {0.03, 0.04}};
```

## Anonymous Array

- 자바에서는 배열의 이름을 지정하지 않고 단순히 초기값만으로 배열을 생성시킬 수 있음
- 무명 배열 (Anonymous array)는 즉시 배열을 만들어서 함수의 인수로 전달하고자 할 때 많이 사용됨



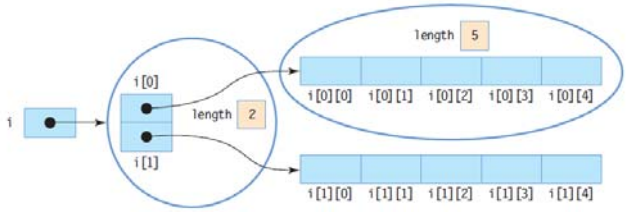

## Ex : 무명배열을 사용한 정수의 합 계산



## 2D Array

- 2차원 배열의 length
  - i.length -> 2차원 배열의 행의 개수로서 2
  - i[n].length는 n번째 행의 열의 개수
    - i[0].length -> 0번째 행의 열의 개수로서 5
    - i[1].length -> 1번째 행의 열의 개수로서 역시 5

```
int i[][] = new int[2][5];
int size1 = i.length; // 2
int size2 = i[0].length; // 5
int size3 = i[1].length; // 5
```




## Ex : 3년간 매출 총액과 평균 구하기

한 회사의 지난 3년간 분기별 매출의 총액과 연평균 매출을 구하는 프로그램을 작성하시오.

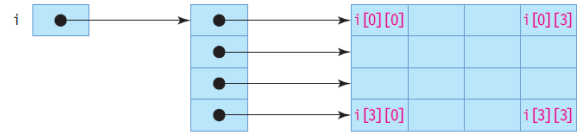
```
public class SalesRevenue {
  public static void main (String[] args) {
    int intArray[][] = {{90, 90, 110, 110},
                        {120, 110, 100, 110},
                        {120, 140, 130, 150}} ;
    double sum = 0;

    for (int i = 0; i < intArray.length; i++) // intArray.length=3
      for (int j = 0; j < intArray[i].length; j++) // intArray[i].length=4
        sum += intArray[i][j];

    System.out.println("지난 3년간 매출 총액은 " + sum +
      "이며 연평균  매출은 " +  sum/intArray.length + "입니다.");
  }
}
```

지난 3년간 매출 총액은 1380.0이며 연평균 매출은 460.0입니다.

## 비정방형 배열

- 정방형 배열
  - 각 행의 열의 개수가 같은 배열

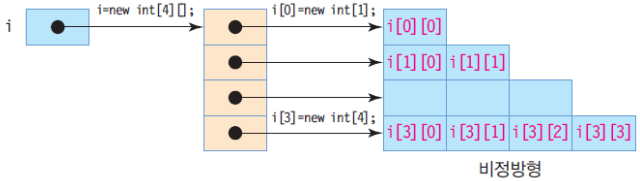
```
int i[][];
i = new int[4][4];
```




정방형

- 비정방형 배열
  - 각 행의 열의 개수가 다른 배열
  - 비정방형 배열의 생성

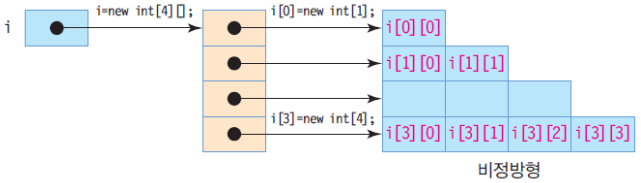
```
int i[][];
i = new int[4][];

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




비정방형

## 비정방형 배열의 length

```
int i[][];
i = new int[4][];

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

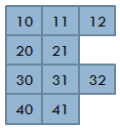



비정방형

- 비정방형 배열의 length
  - i.length -> 2차원 배열의 행의 개수로서 4
  - i[n].length는 n번째 행의 열의 개수
    - i[0].length -> 0번째 행의 열의 개수로서 1
    - i[1].length -> 1번째 행의 열의 개수로서 2
    - i[2].length -> 2번째 행의 열의 개수로서 3
    - i[3].length -> 3번째 행의 열의 개수로서 4

## Ex : 비 정방형 배열의 생성과 접근

다음 그림과 같은 비정방형 배열을 만들어 값을 초기화하고 출력하시오.

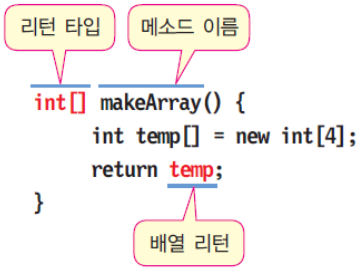



```
public class IrregularArray {
  public static void main (String[] args) {
    int intArray[][] = new int[4][];
    intArray[0] = new int[3];
    intArray[1] = new int[2];
    intArray[2] = new int[3];
    intArray[3] = new int[2];

    for (int i = 0; i < intArray.length; i++)
      for (int j = 0; j < intArray[i].length; j++)
        intArray[i][j] = (i+1)*10 + j;
    for (int i = 0; i < intArray.length; i++) {
      for (int j = 0; j < intArray[i].length; j++)
        System.out.print(intArray[i][j]+" ");
      System.out.println();
    }
  }
}
```

```
10 11 12
20 21
30 31 32
40 41
```

## 메소드에서 배열 리턴

- 메소드의 배열 리턴
  - 배열의 레퍼런스만 리턴
- 메소드의 리턴 타입
  - 메소드가 리턴하는 배열의 타입은 리턴 받는 배열 타입과 일치
  - 리턴 타입에 배열의 크기를 지정하지 않음



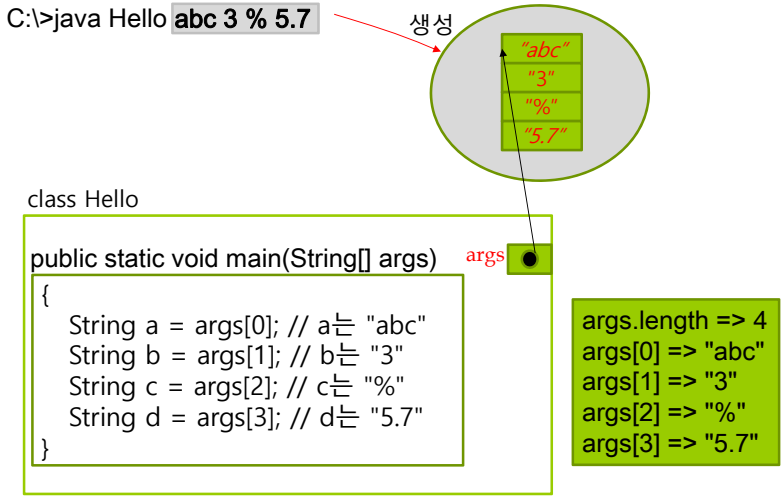

## Ex : 배열 리턴 예제

배열을 생성하고 각 원소 값을 출력하는 프로그램을 작성하시오. 배열 생성은 배열을 생성하여 각 원소의 인덱스 값으로 초기화하여 반환하는 메소드를 이용한다.

```
public class ReturnArray {
  static int[] makeArray() {
    int temp[] = new int[4];
    for (int i=0;i<temp.length;i++)
      temp[i] = i;
    return temp;
  }
  public static void main (String[] args) {
    int intArray [];
    intArray = makeArray();
    for (int i = 0; i < intArray.length; i++)
      System.out.println(intArray[i]);
  }
}
```

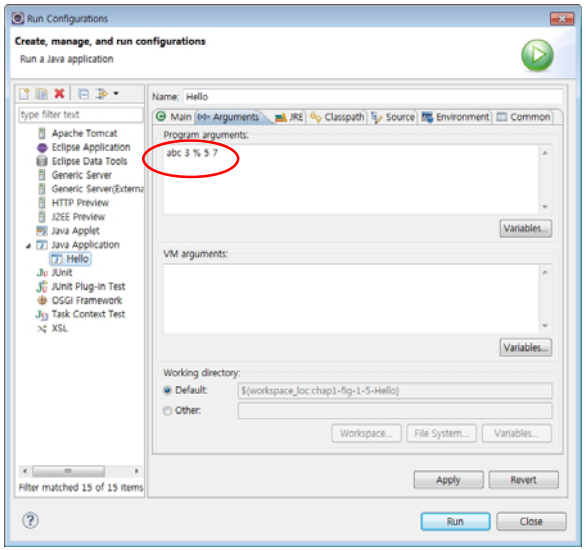
```
0
1
2
3
```

## main(string [] args) 메소드의 인자 전달



# 이클립스에서 main() 메소드의 인자전달

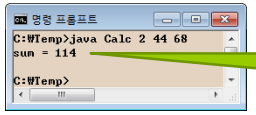
Run 메뉴의 Run Configurations 항목에서 main() 메소드의 인자 나열




# main()의 인자 이용 예

```
public class Calc {
  public static void main(String[] args) {
    int sum = 0;
    for(int i=0; i<args.length; i++) { // 명령행 인자의 개수만큼 반복
      int n = Integer.parseInt(args[i]); // 명령행 인자인 문자열을 정수로 변환
      sum += n; // 숫자를 합한다.
    }
    System.out.println("sum = " + sum);
  }
}
```

Integer.parseInt()는 매개변수로 주어진 문자열을 정수로 변환. Integer.parseInt("68") 은 정수 68 리턴



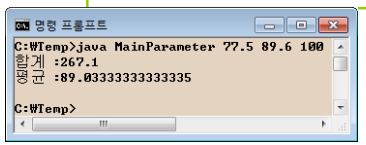

명령행 인자 2, 44, 68을 모두 합하여 114 출력

# Ex : main()의 인자들을 받아서 평균값을 계산하는 예제

여러 개의 실수를 main() 메소드 인자로 전달받아 평균값을 구하는 프로그램을 작성하시오.

```
public class MainParameter {
  public static void main (String[] args) {
    double sum = 0.0;

    for (int i=0; i<args.length; i++)
      sum += Double.parseDouble(args[i]);
    System.out.println("합계 :" + sum);
    System.out.println("평균 :" +
sum/args.length);
  }
}
```

Double.parseDouble()는 매개변수로 주어진 문자열을 실수로 변환. Double.parseDouble("77.5") 은 실수 77.5 리턴




# Ex : 정수가 아닌 문자열을 정수로 변환할 때 예외 발생

문자열을 정수로 변환할 때 발생하는 NumberFormatException을 처리하는 프로그램을 작성하라.

```
public class NumException {
  public static void main (String[] args) {
    String[] stringNumber = {"23", "12", "998", "3.141592"};
    try {
      for (int i = 0; i < stringNumber.length; i++) {
        int j = Integer.parseInt(stringNumber[i]);
        System.out.println("숫자로 변환된 값은 " + j);
      }
    }
    catch (NumberFormatException e) {
      System.out.println("정수로 변환할 수 없습니다.");
    }
  }
}
```

"3.141592"를 정수로 변환할 때 NumberFormatException 예외 발생

```
숫자로 변환된 값은 23
숫자로 변환된 값은 12
숫자로 변환된 값은 998
정수로 변환할 수 없습니다.
```

## LAB3

- 배열을 사용한 프로그램을 작성한다.
  - 프로젝트 디렉토리 안에 보고서 (1~2장)를 넣고 Lab3_학번_이름.zip 압축한 후 제출