



목차

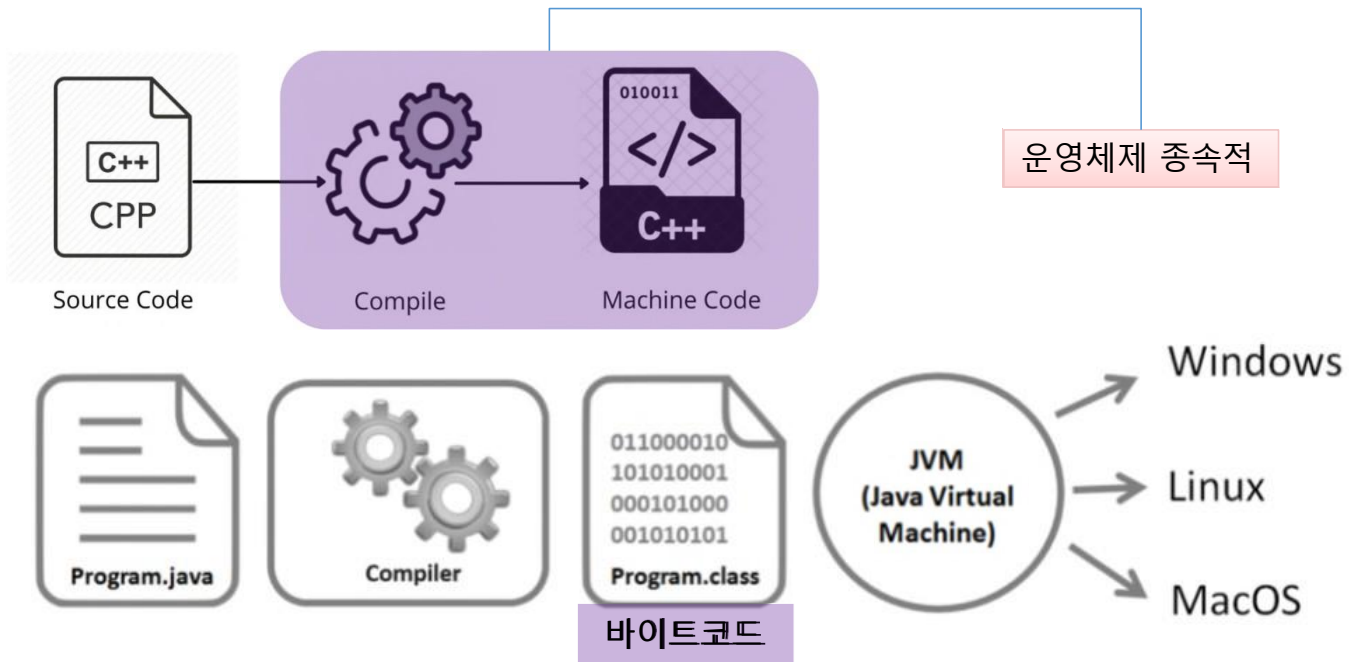
1. Java 기본 지식 복습 및 확인

- 자바의 특징
- 변수와 데이터 타입
- 연산자
- 조건문
- 반복문

Java의 특징, 변수, 데이터 타입

❖ 기본 문제 #01

- Write Once, Run Anywhere 하면 떠오르는 키워드는?



❖ 기본 문제 #02

- 자바의 특징 중 하나로 더 이상 사용하지 않는 메모리를 자동으로 정리하는 기능을 무엇이라고 하는가?

```
String hello = new String("Hello");
```

```
System.out.println(hello);
```

```
hello = null;
```

0x100

hello

Hello

Garbage Collection



❖ 기본 문제 #03

- 자바의 객체지향 특징을 4가지 적으시오.
 - OOP is A.P.I.E
 - Abstraction, Polymorphism, Inheritance, Encapsulation

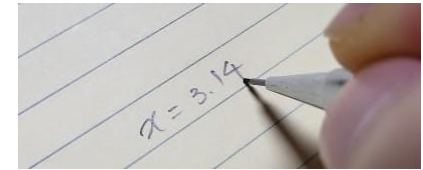


JAVA 기본 및 응용

Confidential

❖ Variable(變數) 이란?

- 수학에서는 변하는 수 x
- 컴퓨터에서는 메모리 공간(그릇)
- 메모리 공간에 값(value)을 할당(assign) 후 사용
- 공간의 크기는 타입별로 달라짐



❖ Type(形) 이란?

- 데이터의 종류
- Primitive Type (기본형)
 - ◆ 미리 정해진 크기의 Memory Size 로 표현
 - ◆ 변수 자체에 값 저장
- Reference Type(참조형)
 - ◆ 크기가 미리 정해질 수 없는 데이터의 표현
 - ◆ 변수에는 실제 값을 참조할 수 있는 주소만 저장

나이

10

우리집



int 나이 = 10

집 우리집 = 서울 특별시 중랑구 ~~

❖ 기본 문제 #04

- Java 의 Primitive Type 을 모두 기술하세요.

구분	Type	bit 수	값
논리형	boolean		true / false
정수형	byte	8	$-2^7 \sim 2^7-1$ (-128 ~ 127)
	short	16	$-2^{15} \sim 2^{15}-1$ (-32768 ~ 32767)
	int	32	$-2^{31} \sim 2^{31}-1$ (-2147483648 ~ 2147483647, 20억 쯤?)
	long	64	$-2^{63} \sim 2^{63}-1$ (-9223372036854775808 ~ 9223372036854775807)
실수형	float	32	<code>float f = 0.1234567890123456789f; // 0.12345679</code>
	double	64	<code>double d = 0.1234567890123456789; // 0.12345678901234568</code>
문자형	char	16	<code>\u0000 ~ \uffff</code> (0 ~ $2^{16}-1$)

파란색의 의미는
무엇일까요?

❖ 문자형

- 내부적으로 ascii 와 unicode로 지정된 값 사용
- 기억할 ASCII 코드

이진법	십진법	문자
0110000	48	'0'
0110001	49	'1'
0110010	50	'2'
1000001	65	'A'
1100001	97	'a'

'b'는 십진수로
얼마일까요??

❖ 기본 문제 #06

- 자바 기본형 데이터 타입(Primitive type)의 개수는 8개이다. 참조형 데이터 타입(Reference type)의 개수는 최대 몇 개까지 사용할 수 있을까?

- 기본 데이터 타입은 byte, short, int, long, float, double, boolean, char 8가지
- 참조형은 나머지 모든 데이터 타입(String, int [], Home 과 같은 사용자 정의 타입 ...)

❖ 기본 문제 #07

● 다음 코드의 실행 결과는?

```
public static void main(String[] args) {  
    int i1 = Integer.MAX_VALUE;  
  
    int i2 = i1+1;  
  
    System.out.println(i2);  
}
```

[illegible]

- 정수 계산 시 overflow 주의
- 필요한 수의 크기를 고려해서 int 또는 long 등 타입 선택

Confidential



❖ 기본 문제 #08

● 다음 코드의 실행 결과는?

```
public static void main(String[] args) {  
    float f1 = 2.0f;  
    float f2 = 1.1f;  
    float f3 = f1 - f2;  
    System.out.println(f3);
```

0.9

```
    double d1 = 2.0;  
    double d2 = 1.1;  
    double d3 = d1 - d2;  
    System.out.println(d3);
```

0.8999999999999999

```
    System.out.println(( (int)(d1*100) - (int)(d2*100))/100.0);
```

0.9

```
    BigDecimal b1 = new BigDecimal("2.0");  
    BigDecimal b2 = new BigDecimal("1.1");  
    System.out.println("BigDecimal을 이용한 빼기 : " + b1.subtract(b2));
```

BigDecimal을 이용한 빼기 : 0.9

```
}
```

실수의 연산은 정확하지 않다.
- 유효 자리수를 이용한 반올림 처리

❖ 형 변환(形, Type casting)이란?

- 변수의 타입을 다른 타입으로 변환하는 것
 - ◆ char \leftrightarrow int
- primitive는 primitive끼리, reference는 reference끼리 형 변환 가능
 - ◆ boolean은 다른 기본 타입과 호환되지 않음
 - ◆ 기본 타입과 참조형의 형 변환을 위해서 Wrapper 클래스 사용
- 형 변환 방법
 - ◆ 형 변환 연산자(괄호) 사용

```
double d = 100.5;  
int result = (int)d;
```

```
result = ?  
d = ?
```

す

● 묵시적 형 변환(promotion)

```
byte b = 10;  
int i = (int)b;  
int i2 = b;
```

```
int i = 300;  
byte b = (byte)i;
```

-
- ```
graph LR; byte --> short; short --> int; int --> long; long --> float; float --> double; char --> int;
```

- 14

## ❖ 기본 문제 #09

### ● 다음 코드의 실행 결과는?

```
public static void main(String[] args) {
 int i1 = Integer.MAX_VALUE;
 int i2 = i1 + 1;
 System.out.println(i2);

 long l1 = i1 + 1;
 System.out.println(l1);

 long l2 = (long) (i1 + 1);
 System.out.println(l2);

 long l3 = (long) i1 + 1;
 System.out.println(l3);

 int i3 = 1000000 * 1000000 / 100000;
 int i4 = 1000000 / 100000 * 100000;
 System.out.println(i3 + " : " + i4);
}
```

-2147483648

-2147483648

-2147483648

2147483648

-7273 : 1000000

### ❖ 기본 문제 #10

#### ● 다음 코드의 실행 결과는?

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

 int k = 66;
 char c = (char) k;
 System.out.println(c);

 c = 'A';
 k = c;
 System.out.println(k);

 int i = 10 / 3;
 System.out.println(i);

 float f = 10 / 3;
 System.out.println(f);

 float f2 = 10f / 3f;
 System.out.println(f2);

 double d = 10d / 3d;
 System.out.println(d);

 System.out.println((10 / 3) * 3);
}
```

```
B
65
3
3.0
3.3333333
3.3333333333333335
9
```



함께가요 미래로!  
Enabling People

# 연산자

### ❖ 연산자란?

- 어떤 기능을 수행하는 기호(+, -, \*, / ...)
- 연산자 종류와 우선순위 및 결합 방향

| 연산기호                                   | 결합방향 | 우선순위 |
|----------------------------------------|------|------|
| ( ), .                                 |      |      |
| ++ -- +(부호) -(부호) ~ ! (type) : 형변환     | ←    | 높음   |
| * / %                                  | →    |      |
| + (덧셈) - (뺄셈)                          | →    |      |
| << >> >>>                              | →    |      |
| < > <= >= instanceof                   | →    |      |
| == !=                                  | →    |      |
| &                                      | →    |      |
| ^                                      | →    |      |
|                                        | →    |      |
| &&                                     | →    |      |
|                                        | →    |      |
| ? :                                    | →    |      |
| = *= /= %= += -= <<= >>= >>>= &= ^=  = | ←    | 낮음   |

연산자 우선순위가 같을 경우  
→ 연산 진행 방향에 의해 결정

3 \* 4 \* 5  
1 → 2

x = y = 3  
← 2 1

### ❖ 기본 문제 #11

#### ● 다음 코드의 실행 결과는?

```
public static void main(String[] args) {
 byte b1 = 10;
 byte b2 = 20;
 byte b3 = b1 + b2;

 int i1 = 10;
 long l1 = 20;
 int i2 = i1 + l1;

 float f1 = 10.0;
 float f2 = f1 + 20.0;
}
```

Type mismatch: cannot convert from int to byte

Type mismatch: cannot convert from long to int

Type mismatch: cannot convert from double to float

- 산술 이항 연산자는 연산 전에 피 연산자의 타입을 일치시킨다.
- 피연산자의 크기가 4byte(int) 미만이면 int로 변경한 후 연산 진행
- 두 개의 피연산자 중 큰 타입으로 형 변환 후 연산 진행

### ❖ 기본 문제 #12

#### ● 다음 중 잘못된 코드들과 그 원인은?

```
public static void main(String[] args) {
 byte b1 = 10;
 b1 = b1 + 1;

 b1 += 1;
 System.out.println(b1);

 byte b2 = ++b1;
 System.out.println(b2);

 byte b3 = (byte) (b2 + 1);
 System.out.println(b3);
}
```

Type mismatch: cannot convert from int to byte

### ❖ 기본 문제 #13

#### ● 다음 코드의 ( 1 ) ~ ( 4 ) 의 실행 결과는?

```
public static void main(String[] args) {
 int i = 10;

 System.out.println((i--) % 2); // (1)

 System.out.println(--i); // (2)

 System.out.println(i++); // (3)

 System.out.println(++(i - 2)); // (4)
}
```

```
0
8
8
// 컴파일 에러
```

❌ Invalid argument to operation ++/--

++이나 --는 변수에만 사용할 수 있다.  
i-2는 값!!

## ❖ 비트 논리 연산자 종류

### ● 2진수 형태로 변환 후 연산

| 연산자 | 연산자 기능                                          | 결합 방향 |
|-----|-------------------------------------------------|-------|
| &   | 두 피 연산자의 비트 값이 모두 1인 경우만 1, 나머지는 0<br>ex) a & b | →     |
|     | 두 피 연산자의 비트 값이 모두 0인 경우만 0, 나머지는 1<br>ex) a   b | →     |
| ^   | 두 피 연산자의 비트 값이 서로 다르면 1, 같으면 0<br>ex) a ^ b     | →     |
| ~   | 피 연산자의 모든 비트를 반전시킴 → 1의 보수<br>ex) ~a            | ←     |

a: 10 진수 3 → 이진수 0 0 0 0 0 0 1 1

b: 10 진수 5 → 이진수 0 0 0 0 0 1 0 1

a & b : 0 0 0 0 0 0 0 1

a | b : 0 0 0 0 0 1 1 1

a ^ b : 0 0 0 0 0 1 1 0

### ❖ 비트 이동연산자(쉬프트 연산자)

| 연산자 | 연산자 기능                                                                            | 결합 방향 |
|-----|-----------------------------------------------------------------------------------|-------|
| <<  | 앞의 피 연산자 비트 열을 뒤 피 연산자 만큼 왼쪽으로 이동하고 이동에 따른 빈 공간은 0으로 채움 ex) $a \ll 2$             | →     |
| >>  | 앞의 피 연산자 비트 열을 뒤 피 연산자 만큼 오른쪽으로 이동하고 이동에 따른 빈 공간은 음수는 1, 양수는 0으로 채움 ex) $a \gg 2$ | →     |
| >>> | 앞의 피 연산자 비트 열을 뒤 피 연산자 만큼 오른쪽으로 이동하고 이동에 따른 빈 공간은 0으로 채움 ex) $a \ggg 2$           | →     |

int a=10;

0 ... 0 0 0 0 1 0 1 0

2 진수 a

10 진수 a

2 진수 b

10 진수 b

int b = a<<1

2 진수 d

10 진수 d

int d = a>>1

부호비트

$$\begin{array}{cccc}
 1 & 0 & 1 & 0 \\
 2^3 * 1 + 2^2 * 0 + 2^1 * 1 + 2^0 * 0 = 8 + 2 = 10
 \end{array}$$

$$\begin{array}{ccccc}
 1 & 0 & 1 & 0 & 0 \\
 2^4 * 1 + 2^3 * 0 + 2^2 * 1 + 2^1 * 0 + 2^0 * 0 = 16 + 4 = 20
 \end{array}$$

\*2와 동일

$$\begin{array}{ccc}
 1 & 0 & 1 \\
 2^2 * 1 + 2^1 * 0 + 2^0 * 1 = 4 + 1 = 5
 \end{array}$$

/2와 동일

- \*2, /2 에 비해 속도가 빠름

### ❖ 논리 연산자 - &, | vs &&, ||

#### ● 논리 연산자

| 연산자 | 연산자 기능                                                                                                                                                    | 결합 방향 |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| &   | 두 개의 피연산자가 모두 true인 경우 true Ex) $a > 0 \ \& \ b > 0$                                                                                                      | →     |
|     | 두 개의 피 연산자가 하나라도 true이면 true Ex) $a > 0 \   \ b < 0$                                                                                                      | →     |
| !   | 단항 연산자로 피 연산자의 값이 false이면 true, true이면 false로 변경 Ex) $!a$                                                                                                 | ←     |
| ^   | 두 피 연산자가 서로 다를 경우만 true, 같으면 false Ex) $\text{true} \wedge \text{false} \rightarrow \text{true}, \text{true} \wedge \text{true} \rightarrow \text{false}$ | →     |

|   |   |   |
|---|---|---|
| & | T | F |
| T | T | F |
| F | F | F |

|   |   |   |
|---|---|---|
|   | T | F |
| T | T | T |
| F | T | F |

#### ● Short circuit 연산자

| 연산자 | 연산자 기능                                                                       | 결합 방향 |
|-----|------------------------------------------------------------------------------|-------|
| &&  | &와 동일한 의미이나 앞의 피 연산자가 false이면 뒤의 피 연산자를 검사하지 않는다. ex) $a > 0 \ \&\& \ b > 0$ | →     |
|     | 와 동일한 의미이나 앞의 피 연산자가 true이면 뒤의 피 연산자를 검사하지 않는다. ex) $a > 0 \    \ b < 0$     | →     |



### ❖ 기본 문제 #15

#### ● 다음 코드의 실행 결과는?

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

```
 int a = 10;
```

```
 int b = 20;
```

```
 System.out.println((a > b) & (b > 0));
```

false

```
 System.out.println((a += 10) > 15 | (b -= 10) > 15);
```

true

```
 System.out.println("a = " + a + ", b = " + b);
```

a = 20, b = 10

```
 a = 10;
```

```
 b = 20;
```

```
 System.out.println((a += 10) > 15 || (b -= 10) > 15);
```

true

```
 System.out.println("a = " + a + ", b = " + b);
```

a = 20, b = 20

```
}
```

## ❖ Random 수 구현하기

| 구분                                                      | 코드                                                 |                     |
|---------------------------------------------------------|----------------------------------------------------|---------------------|
| <b>Math.random()</b>                                    | <code>double d = Math.random()</code>              | $0.0 \leq d < 1.0$  |
|                                                         | <code>double d = Math.random() * N</code>          | $0.0 \leq d < N.0$  |
|                                                         | <code>int i = (int) (Math.random() * N)</code>     | $0 \leq i \leq N-1$ |
|                                                         | <code>int i = (int) (Math.random() * N) + 1</code> | $1 \leq i \leq N$   |
| <b>Random rand = new Random();<br/>rand.nextInt(N);</b> | <code>int i = rand.nextInt(N)</code>               | $0 \leq i < N$      |
|                                                         | <code>int i = rand.nextInt(N) + 1</code>           | $1 \leq i \leq N$   |

### ❖ 기본 문제 #16

- 주사위를 던져서 나올 수 있는 경우의 수(1~6)를 시뮬레이션 하려고 한다. 무작위 수를 이용하여 코드를 작성하세요.

```
public static void main(String[] args) {
 int N = 6;

 int num = (int)(Math.random() *N) +1;
 System.out.println(num);

 Random rand = new Random();
 num = rand.nextInt(N)+1;
 System.out.println(num);
}
```

# 조건문

## ❖ 조건문 ( Conditional Statement )

if (        )

논리형

boolean **b**;

비교식

**x >= y**

Method  
Call

**isEven()**

switch (        )

정수호환

byte, short, char, int **x**;

Enum

**Day.MONDAY**

Class  
Object

**Byte, Short, Character,  
Integer, String**

Method  
Call

**getNumber()**

### ❖ 조건문

- 주사위를 던져서 나오는 숫자에 따라 동작하는 코드를 작성하시오

```
int N = 6;
int result = (int) (Math.random()*N) + 1;

if(result == 1) {
 // do something
}else if(result == 2) {
 // do something
}else if(result == 3) {
 // do something
}else if(result == 4) {
 // do something
}else if(result == 5) {
 // do something
}else {
 // do something
}
```

```
int N = 6;
int result = (int) (Math.random()*N) + 1;

switch(result) {
 case 1 : // do something
 break;
 case 2 : // do something
 break;
 case 3 : // do something
 break;
 case 4 : // do something
 break;
 case 5 : // do something
 break;
 case 6 : // do something
 break;
 default : // do something
}
}
```

### ❖ 기본 문제 #17

- 다음의 조건 문에서 잘못 작성된 것을 고르시오.(논리적 오류 포함)

```
int a = 20;
String grade = null;
```

```
if (a >= 19) {
 grade = "성인";
} else if (a >= 13) {
 grade = "청소년";
} else if (a >= 6) {
 grade = "아동";
} else {
 grade = "유아";
}
```

```
if (a >= 6) {
 grade = "아동";
} else if (a >= 13) {
 grade = "청소년";
} else if (a >= 19) {
 grade = "성인";
} else {
 grade = "유아";
}
```

```
if (a >= 19)
 grade = "성인";
else if (a >= 13)
 grade = "청소년";
else if (a >= 6)
 grade = "아동";
else
 grade = "유아";
```

```
if (a >= 19)
 grade = "성인";
 System.out.println("SSAFY 지원 가능");
else if (a >= 13)
 grade = "청소년";
else if (a >= 6)
 grade = "아동";
else
 grade = "유아";
```

### ❖ 기본 문제 #18

- 다음 표현을 3항 연산자를 이용하는 형태로 처리하세요.

```
public static void main(String[] args) {
 int age = 10;
 String status = null;
 if (age >= 19) {
 status = "충분히 성장했다.";
 } else {
 status = "아직 어리다.";
 }

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

```
status = age >= 19 ? "충분히 성장했다." : "아직 어리다.";
```



### ❖ switch - break, default

switch ( value )

```
case 1 : ...;
case 2 : ...; break;
case 3 : ...;
case 4 : ...;
case 5 : ...; break
default :
```

if value == 1?

if value == 3?

if value == 0?

### ❖ 기본 문제 #19

- 다음 코드의 Local Variables 중 switch() X 에 사용할 수 없는 것은?

```
public static void main(String[] args) {
 int I = 3;
 byte B = 3;
 short S = 3;
 char C = 'C';
 double D = 3.0d;
 String str = "STR";

 switch(X) {
 }
}
```

```
double D = 3.0d;
```

### ❖ 기본 문제 #20

#### ● 다음 코드의 실행 결과는?

```
public static void main(String[] args) {
 int num = 3;

 switch(num) {
 case 1 : System.out.println(num);
 case 2 : System.out.println(num);
 case 3 : System.out.println(num);
 case 4 : System.out.println(num);
 case 5 : break;
 case 6 : break;
 default : System.out.println(num);
 }
}
```

3  
3

```
int month = 3;
int day = -1;
switch (month) {
 case 2:
 day = 29; break;
 case 4:
 case 6:
 case 9:
 case 11:
 day = 30; break;
 default:
 day = 31;
}
```

### ❖ 기본 문제 #21

- 다음 코드의 오류가 발생한다. 그 이유는?

```
public static void main(String[] args) {
 char C = 'A';

 switch(C) {
 case 'A' : // do nothing
 break;
 case 'B' : // do nothing
 break;
 case 65 : // do nothing
 break;
 }
}
```

❌ Duplicate case

### ❖ 기본 문제 #22-1

- 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, "A"를 그렇지 않으면 "B"를 출력하는 코드를 작성하시오

```
public static void main(String[] args) {
 int N = 6;
 Random rand = new Random();
 int num1 = rand.nextInt(N)+1;
 int num2 = rand.nextInt(N)+1;
 String result = null;
 // TODO: your code here
```

Nested If 사용

```
 // END:
 System.out.printf("%d, %d --> %s\n", num1, num2, result);
}
```

```
boolean isNum1Even = num1%2==0;
boolean isNum2Even = num2%2==0;

if(isNum1Even) {
 if(isNum2Even) {
 System.out.println("A");
 }else {
 System.out.println("B");
 }
}else {
 if(!isNum2Even) {
 System.out.println("A");
 }else {
 System.out.println("B");
 }
}
```

### ❖ 기본 문제 #22-2

- 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, "A"를 그렇지 않으면 "B" 를 출력하는 코드를 작성하시오

```
boolean isNum1Even = num1%2==0;
boolean isNum2Even = num2%2==0;

result = isNum1Even==isNum2Even?"A":"B";
```

3항 연산자의 활용

### ❖ 기본 문제 #22-3

- 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, "A"를 그렇지 않으면 "B" 를 출력하는 코드를 작성하시오

```
result = (num1 + num2)%2==0?"A":"B";
```

수학적 개념에 의한 개선

# 반복문



## ❖ 반복문 for 구성

변수 초기화

`int i = 0;`

반복 조건

`i < 10;`

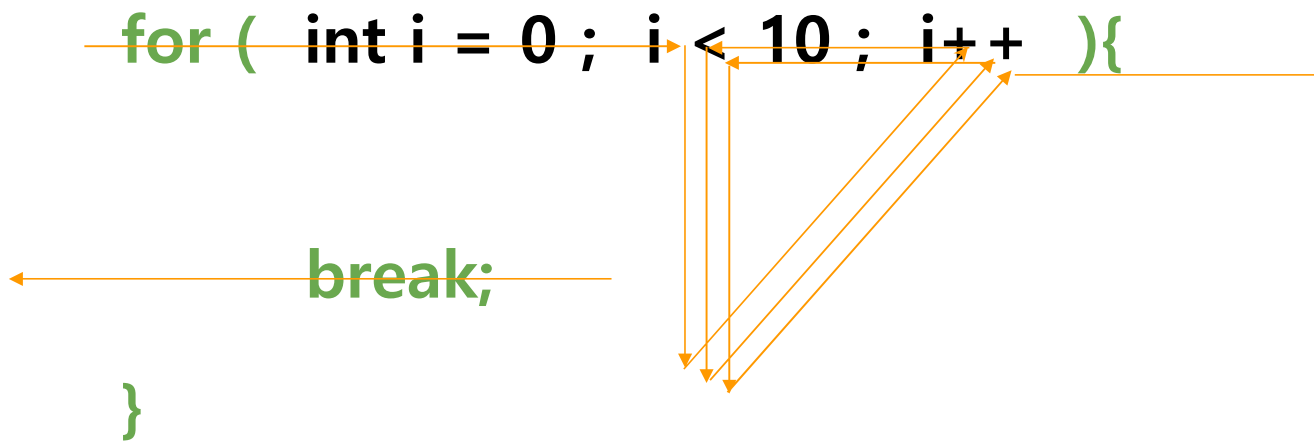
증감식

`i++`

```
for (_____ ; _____ ; _____) {
 실행문
}
```

## ❖ 반복문 for 실행 및 종료

```
for (int i = 0 ; i < 10 ; i++){
 break;
}
```



### ❖ 기본 문제 #23

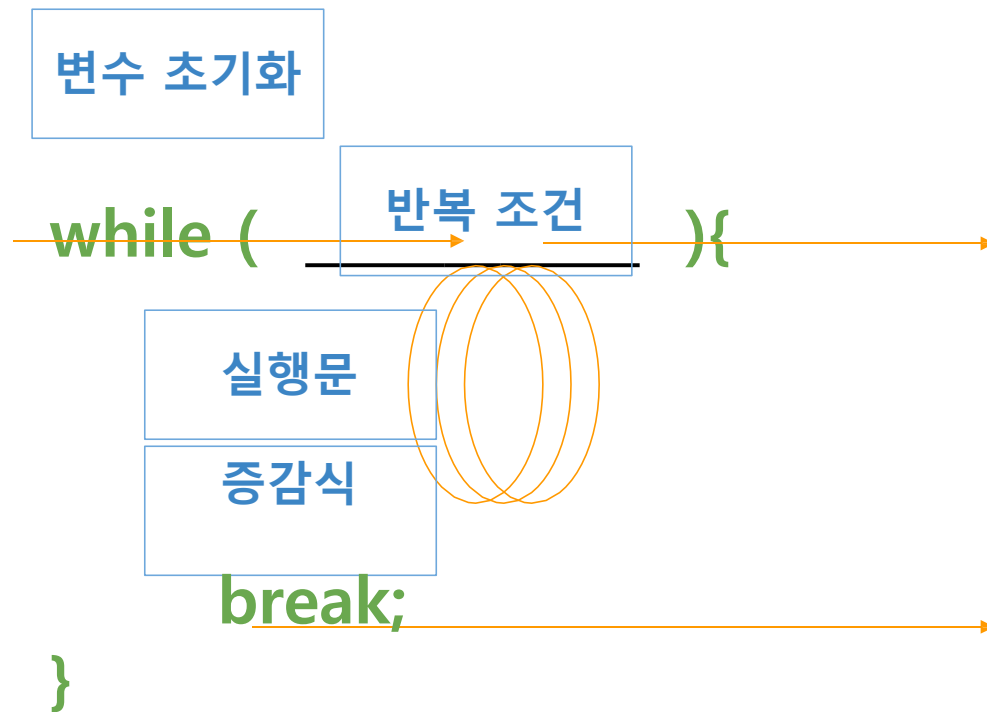
- 프로그램은 반복적으로 어떤 작업을 수행해야 할 수 있다.  
100 번 주사위를 던진 결과의 합과 평균값(실수)을 출력하는 코드를 for 문을 이용하여 구현하시오.

```
public static void main(String[] args) {
 int sum = 0;
 int cnt = 100;
 double avg = 0;
 Random rand = new Random();

 for (int i = 0; i < cnt; i++) {
 sum += rand.nextInt(6) + 1;
 }
 avg = 1.0*sum/cnt;

 System.out.printf("sum: %d, avg: %f\n", sum, avg);
}
```

## ❖ 반복문 while 실행 및 종료



## ❖ 기본 문제 #24

- 기본 문제 #23 을 while 을 이용하여 만드시오.

```
public static void main(String[] args) {
 int sum = 0;
 int cnt = 100;
 double avg = 0;
 Random rand = new Random();
 int i = 0;

 while (i < cnt) {
 sum += rand.nextInt(6) + 1;
 i++;
 }
 avg = 1.0*sum/cnt;

 System.out.printf("sum: %d, avg: %f\n", sum, avg);
}
```

## ❖ for vs while

|       |                                                                                                                            |
|-------|----------------------------------------------------------------------------------------------------------------------------|
| for   | <ul style="list-style-type: none"><li>- 초기값, 조건식, 증감식의 위치가 명확</li><li>- 반복 회수 예측이 가능한 반복</li><li>- index 의 증감 활용</li></ul> |
| while | <ul style="list-style-type: none"><li>- 반복 회수 예측이 가변적인 반복</li><li>- index 보다는 break, continue 활용</li></ul>                 |

앞으로 365일!  
싸피에서 열심히  
공부하자!



싸피 끝날 때까지  
열심히 공부하자!



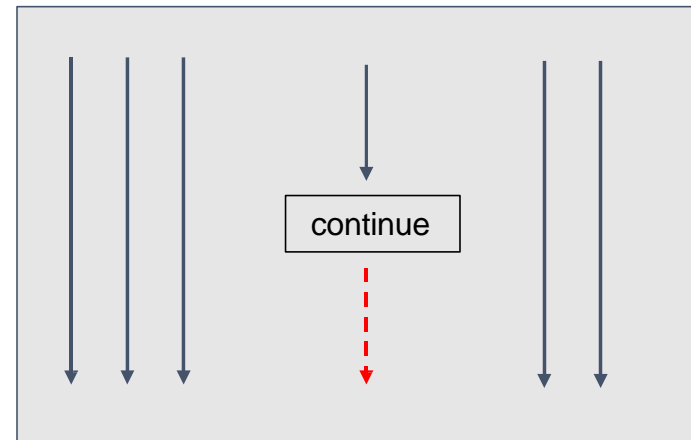
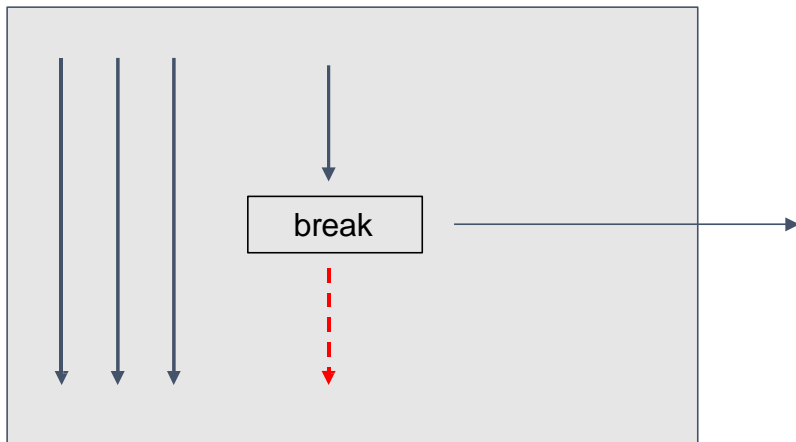
### ❖ 기본 문제 #25

- 프로그램은 반복문을 수행하는 도중, 특정 조건에 따라, **반복문을 중단**하거나, 그 **조건**의 경우에만 **수행하지 않고**, 계속 반복문을 이어서 진행하는 경우가 있다. 이에 해당하는 Java Keyword 는?

1. exit, continue
2. stop, skip
3. break, skip
4. break, continue

4. break, continue

## ❖ break vs continue





## ❖ 기본 문제 #26

### ● 다음 코드의 결과는?

```
public static void main(String[] args) {
 for(int i=1; i<10; i++) {
 for(int j=1; j<10; j++) {
 if(j==5) break;
 if(j==3) continue;
 System.out.print(i*j+"Wt");
 }
 System.out.println();
 }
}
```

|       |   |    |    |    |
|-------|---|----|----|----|
| i \ j | 1 | 2  | 3  | 4  |
| 1     | 1 | 2  | 3  | 4  |
| 2     | 2 | 4  | 6  | 8  |
| 3     | 3 | 6  | 9  | 12 |
| 4     | 4 | 8  | 12 | 16 |
| 5     | 5 | 10 | 15 | 20 |
| 6     | 6 | 12 | 18 | 24 |
| 7     | 7 | 14 | 21 | 28 |
| 8     | 8 | 16 | 24 | 32 |
| 9     | 9 | 18 | 27 | 36 |

### ❖ 기본 문제 #27

#### ● 다음 코드의 결과는?

```
public static void main(String[] args) {
 outer: for(int i=1; i<10; i++) {
 for(int j=1; j<10; j++) {
 if(j==5) break outer;
 if(j==3) continue outer;
 System.out.print(i*j+" ");
 }
 System.out.println();
 }
}
```

1 2 **2** **4** **3** **6** **4** **8** **5** **10** 6 12 7 14 8 16 9 18

## ❖ 기본 문제 #28

- for 반복문을 사용하여 # 문자가 아래와 같이 출력되도록 코드를 작성하세요.

```


#####
```

```
for(int i=0; i<5; i++) {
 for(int j=0; j<=i; j++) {
 System.out.print("#");
 }
 System.out.println();
}
```

### ❖ 기본 문제 #29

- for 반복문을 사용하여 # 문자가 아래와 같이 출력되도록 코드를 작성하세요.

```


#
```

```
for(int i=5; i>0; i--) {
 for(int j=0; j<i; j++) {
 System.out.print("#");
 }
 System.out.println();
}
```

```
for(int i=0; i<5; i++) {
 for(int j=5-i; j>0; j--) {
 System.out.print("#");
 }
 System.out.println();
}
```

### ❖ 기본 문제 #30

- for 반복문을 사용하여 # 문자가 아래와 같이 출력되도록 코드를 작성하세요.

```


#
```

```
for(int i=0; i<5; i++) {
 for(int j=0; j<i; j++) {
 System.out.print(" ");
 }
 for(int j=9-2*i; j>0; j--) {
 System.out.print("#");
 }
 System.out.println();
}
```

함께가요 미래로!  
Enabling People

# 배열

## ❖ 기본 문제 #01

- 주사위를 5 번 던져서 각각의 결과 값을 저장한 후, 필요할 때 사용 하려고 한다.
- 각각의 값을 출력하는 코드를 작성 하시오. 단, Array 를 사용하지 않고 작성하세요.

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

 int N = 6;
 Random rand = new Random();

 int result1 = rand.nextInt(N) + 1;
 int result2 = rand.nextInt(N) + 1;
 int result3 = rand.nextInt(N) + 1;
 int result4 = rand.nextInt(N) + 1;
 int result5 = rand.nextInt(N) + 1;

 System.out.println(result1);
 System.out.println(result2);
 System.out.println(result3);
 System.out.println(result4);
 System.out.println(result5);
}
```

6  
6  
4  
1  
1

## ❖ 동일한 타입의 변수를 여러 개 사용하면..

- 변수의 수 증가
- 코드의 길이 증가
- 반복문 적용 불가
- 변수의 수가 동적으로 결정될 경우, 사용 불가

## ❖ 배열(Array) 로 동일 타입 변수 묶어서 사용하기

- 배열이란? **동일한 타입의 데이터 0개 이상**을 하나의 **연속된 메모리 공간에서 관리**하는 것
- 요소에 접근하는 속도가 매우 빠르다.
- 한번 생성하면 크기 변경 불가



## ❖ 기본 문제 #02



- 왜 계란을 한 판으로 포장할까?
- 계란 판에 메추리알, 타조알을 담을 수 있을까?
- 계란 한 판은 30개 → 31개를 저장하려면?
- 처음 계란에서 3번째 계란을 가져오려면?
- 마지막 그림에서 관리되는 대상은 무엇일까?

## ❖ Array 만들기 #1

- 타입 [] 변수명; 또는 타입 변수명 []

- ◆ int [] points; String [] names;

- 변수의 타입과 저장하는 데이터의 타입?

|        |     |
|--------|-----|
| int    | a   |
| int [] | arr |

a의 타입?

arr의 타입?

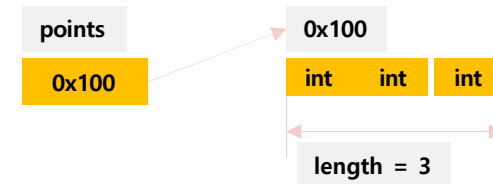
arr이 저장하는 데이터의 타입?

### ❖ 배열의 생성과 초기화

#### ● 생성

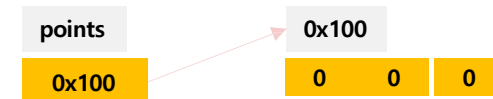
◆ new keyword와 함께 배열의 데이터 타입 및 크기 지정 : `new data_type[ length ]`

- `new int [3];` int타입의 자료 3개를 저장할 수 있는 배열을 메모리에 생성
- `points = new int [3];` 생성된 배열을 points라는 변수에 할당
- points는 메모리에 있는 배열을 가리키는 reference 타입 변수



#### ● 초기화

◆ 배열을 생성과 동시에 자료형에 대한 default 초기화 진행



| 자료형              | 기본값      | 비고           |
|------------------|----------|--------------|
| boolean          | false    |              |
| char             | '\u0000' | 공백문자         |
| byte, short, int | 0        |              |
| long             | 0L       |              |
| float            | 0.0f     |              |
| double           | 0.0      |              |
| 참조형 변수           | null     | 아무것도 참조하지 않음 |

## ❖ 배열의 사용

- 배열은 index 번호를 가지고 각 요소에 접근 가능
  - ◆ index 번호는 0부터 시작
  - ◆ 배열의 길이 : 배열이름.length 로 배열의 크기 조회 가능

```
int [] points = new int[3];
System.out.printf("배열의 크기: %d\n", points.length);

points[0] = 1;
points[1] = 'A'; //묵시적 형 변환
//points[2] = 1.5; //double 할당 불가

System.out.printf("0 번째 요소: %d\n", points[0]);
System.out.printf("1 번째 요소: %d\n", points[1]);
System.out.printf("2 번째 요소: %d\n", points[2]);
```

## ❖ 기본 문제 #03

- 기본 문제 #01을 배열을 이용해서 처리하시오.

```
int N = 6;
Random rand = new Random();

int [] resultArray = new int [5];

for(int i=0; i<resultArray.length; i++) {
 resultArray[i] = rand.nextInt(N)+1;
}

for(int i=0; i<resultArray.length; i++) {
 System.out.printf("%d번째 - %d\n",i, resultArray[i]);
}
```

### ❖ 기본 문제 #04

- char [ ]을 이용해 String "SS" 의 각 문자를 저장하고 출력하는 코드를 작성하시오.

```
String org = "SS";
char[] chars = new char[org.length()];

for (int i = 0; i < chars.length; i++) {
 chars[i] = org.charAt(i);
}

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

```
// API의 활용
chars = org.toCharArray();
for (int i = 0; i < chars.length; i++) {
 System.out.print(chars[i]);
}
```

배열의 길이를 몇으로 해야 할까요?

문자열에서 문자를 가져오려면요?



### ❖ 기본 문제 #05

- String "1234567890" 의 자리 별 수를 1차원 배열에 저장하고 배열을 순회해서 그 합을 출력하시오.

```
String org = "1234567890";

char [] nums = org.toCharArray();

int sum = 0;

for(int i=0; i<nums.length; i++) {
 sum+=nums[i]-'0';
}

System.out.printf("sum: %d\n", sum);
```

문자에서 숫자를  
구할 수 있어요?

## ❖ Array 출력을 편리하게

- for 문을 통한 출력대신 **Arrays.toString()**

### toString

```
public static String toString(char[] a)
```

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (a comma followed by a space). Elements are converted to strings as by `String.valueOf(char)`. Returns "null" if a is null.

#### Parameters:

a - the array whose string representation to return

#### Returns:

a string representation of a

#### Since:

1.5



## ❖ Array 만들기 #2

- 생성과 동시에 할당한 값으로 초기화

- ◆ `int [] b = {1, 3, 5, 6, 8};`

- ◆ `int [] c = new int [] {1, 3, 5, 6, 8};`

- 선언 후 생성 시 초기화 주의

- ◆ `int [] points;`

- `points = {1, 3, 5, 6, 8};`                      // 컴파일 오류

- ◆ `int [] points;`

- `points = new int [] {1, 3, 5, 6, 8};`    // 선언할 때는 배열의 크기를 알 수 없을 때

```
public static void main(String[] args) {
 String [] strs1 = new String [3];
 strs1[0] = "Hello";
 strs1[1] = "Java";
 strs1[2] = "World";
 String [] strs2 = {"Hello", "Java", "World"};
 if(args.length==0){
 args = new String[] {"Hello", "Java", "World"};
 }
 for(int i=0; i<strs1.length; i++){
 System.out.printf("strs1: %s, strs2: %s, args: %s %n", strs1[i], strs2[i], args[i]);
 }
}
```

# JAVA 기본 및 응용

**Confidential**

## ❖ 배열의 생성과 메모리 사용 과정

```
int [] points = new int[3];
```

배열 선언 : `int [] points`

배열 생성: `new int[3];`  
메모리에 연속된 공간 차지 → 크기 변경 불가!  
Type에 대한 default 초기화

참조 값 할당 : `points = new int[3];`

요소에 값 할당 :  
`points[0] = 1;`  
`points[1] = 'A';`

points

null

points

null

[0]

[1]

[2]

0

0

0

0x100

int 타입의 데이터 3개를 담을 수 있는 메모리 공간 확보

points

0x100

[0]

[1]

[2]

0

0

0

0x100

배열의 주소를 변수에 할당하여 참조케 함

points

0x100

[0]

[1]

[2]

1

65

0

0x100

### ❖ for-each with Array

- 가독성이 개선된 반복문으로, 배열 및 Collections 에서 사용
- index 대신 직접 요소( elements )에 접근하는 변수를 제공
- naturally read only ( copied value )
- 사용

```
int intArray [] = { 1, 3, 5, 7, 9 };
```

```
for(int x : intArray){
 System.out.println(x);
}
```

```
for(int i=0; i<intArray.length; i++){
 int x = intArray[i];
 System.out.println(x);
}
```

Index를 사용하지 않아도 되지만,  
사용할 수 없다. → 용도에 따라 사용

## ❖ 기본 문제 #07

- SAFFY 의 어떤 반 학생들의 이름이 아래와 같다. 아래 이름을 Array 로 순차적으로 저장하고, for-each로 출력하는 코드를 작성하시오.

|   |     |
|---|-----|
| 1 | 홍길동 |
| 2 | 임꺽정 |
| 3 | 장길산 |
| 4 | 이몽룡 |

```
String[] students = {"홍길동", "임꺽정", "장길산", "이몽룡"};

for (String student : students) {
 System.out.println(student);
}
```

### ❖ 기본 문제 #08

- 기본 문제 #07 코드에 이어서 '임꺽정' 와 '장길산' 의 순서를 바꾸고 다시 for-each 로 이름을 출력하는 코드를 작성하시오.

|   |     |
|---|-----|
| 1 | 홍길동 |
| 2 | 임꺽정 |
| 3 | 장길산 |
| 4 | 이몽룡 |



```
String temp = students[1];
students[1] = students[2];
students[2] = temp;
```

```
for (String student : students) {
 System.out.println(student);
}
```

홍길동  
장길산  
임꺽정  
이몽룡

### ❖ 기본 문제 #09

- 국,영,수 3과목에 대한 시험 점수를 아래와 같은 배열에서 관리중이다.
- `int [] scores = {90, 80, 100};`
- 새롭게 사회 과목을 추가로 관리해야 할 때 적절하지 못한 코드는?

```
int[] scores = {90, 80, 100};
// #1
scores[3] = 95;
// #2
scores = new int[] {90, 80, 100, 95};
// #3
scores = {90, 80, 100, 95};
// #4
int[] scores2 = new int[4];
System.arraycopy(scores, 0, scores2, 0, scores.length);
scores2[3] = 95;
// #5
scores = Arrays.copyOf(scores, 5);
```

java.lang.ArrayIndexOutOfBoundsException: 3

✖ Array constants can only be used in initializers

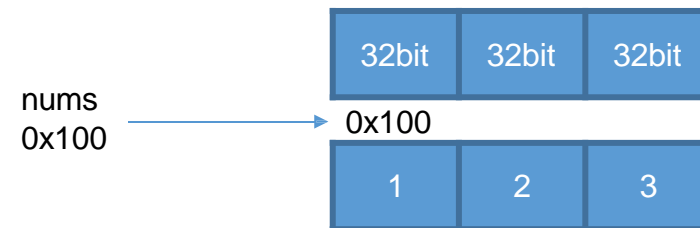
## ❖ Array is Immutable

- 최초 메모리 할당 이후, 변경할 수 없음.
- 개별 요소는 다른 값으로 변경이 가능하나, 삭제할 수는 없음.
- 크기를 늘리거나 줄일 수 없음.

```
int [] nums = {1,2,3};

nums[1]=100;

nums = new int[] {1,2,3,4};
```



## ❖ System.arraycopy

### ● api 제공하는 배열 복사 method

#### arraycopy

```
public static void arraycopy(Object src,
 int srcPos,
 Object dest,
 int destPos,
 int length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. A subsequence of array components are copied from the source array referenced by `src` to the destination array referenced by `dest`. The number of components copied is equal to the `length` argument. The components at positions `srcPos` through `srcPos+length-1` in the source array are copied into positions `destPos` through `destPos+length-1`, respectively, of the destination array.



## ❖ 기본 문제 #10

- 절대 값이 1000을 넘지 않는 정수로 구성된 intArray 배열이 아래와 같이 주어졌을 때, 최대값, 최소값을 출력하는 프로그램을 작성하세요.
- `int[] intArray = { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };`

```
int[] intArray = { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };
```

```
int min = 1000;
int max = 0;
for(int num: intArray) {
 if(num>max) {
 max = num;
 }
 if(num<min) {
 min = num;
 }
}
```

```
System.out.printf("min: %d, max: %d\n", min, max);
```

min: 3, max: 435

### ❖ 기본 문제 #11

- 아래와 같이 intArray 배열이 주어졌을 때, 평균(1)을 구하고 평균과 차이가 가장 큰 값(2)과 가장 작은 값(3) 을 출력하는 프로그램을 작성하세요.

◆ 1은 실수로 소수점 셋째자리에서 반올림해서 출력하고 2, 3의 값이 여러 개일 경우는 아무 값이나 출력한다.

- `int[] intArray = { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };`

```
int sum = 0;
for (int num : intArray) {
 sum += num;
}
double avg = 1.0 * sum / intArray.length;
double max = Double.MIN_VALUE;
double min = Double.MAX_VALUE;
int maxIdx = -1;
int minIdx = -1;

for (int i = 0; i < intArray.length; i++) {
 if (Math.abs(intArray[i] - avg) > max) {
 max = Math.abs(intArray[i] - avg);
 maxIdx = i;
 }
 . . .
}
```

```
System.out.printf("avg: %.2f, maxDiff: %d, minDiff: %d\n", avg, intArray[maxIdx], intArray[minIdx]);
```

뭔가 이상한걸?

avg:76.73, maxDiff: 435, minDiff:54

### ❖ 기본 문제 #12

- intArray 배열이 아래와 같이 주어졌을 때, 각 숫자가 몇 번 사용 되었는지 숫자별로 사용 횟수를 출력 하세요. 사용 안된 숫자는 0으로 출력 한다. (배열의 요소는  $0 \leq x < 10$ 이다.)
- `int[] intArray = { 3, 7, 2, 5, 7, 7, 9, 2, 8, 1, 1, 5, 3 };`

```
public static void main(String[] args) {
 int[] intArray = {3, 7, 2, 5, 7, 7, 9, 2, 8, 1, 1, 5, 3};
 int[] used = new int[10];
```

```
 for(int num:intArray) {
 used[num]++;
 }
```

```
 System.out.println(Arrays.toString(used));
}
```

```
[0, 2, 2, 2, 0, 2, 0, 3, 1, 1]
```

배열 숫자가 used 배열의 index 로 활용  
Used 요소의 default value : 0

### ❖ 기본 문제 #13

- 1~20 까지의 숫자를 사용한 intArray 배열이 아래와 같이 주어졌을 때, 사용되지 않은 숫자를 출력 하세요.
- `int[] intArray = { 1, 3, 4, 7, 8, 10, 12, 15, 16, 17, 18 };`

```
public static void main(String[] args) {
 int[] intArray = {1, 3, 4, 7, 8, 10, 12, 15, 16, 17, 18};
 int[] used = new int[21];

 for(int num:intArray) {
 used[num]++;
 }

 for(int i=1; i<used.length; i++) {
 if(used[i]==0) {
 System.out.print(i+" ");
 }
 }
}
```

2 5 6 9 11 13 14 19 20

# 다차원 배열

## ❖ 기본 문제 #14

❖ 다음은 4x3 의 2차원 배열을 만드는 방법이다. 올바르지 않은 것은?

```
int intArray[][] = new int [4][3];
```

```
int [] intArray2[] = new int [4][3];
```

```
int [][] intArray3 = new int [4][3];
```

```
int [][] intArray4 = new int [4]{1,2,3};
```

```
int [][] intArray5 = new int[][] {{1,2,3},{1,2,3},{1,2,3},{1,2,3}};
```

```
int [][] intArray6 = {{1,2,3},{1,2,3},{1,2,3},{1,2,3}};
```

## ❖ 2차원 Array 만들기 #1

### ● int Type 기준으로 4x3 배열 (Array) 만들기

| 선언                  | 생성                        | 할당                  |
|---------------------|---------------------------|---------------------|
| int [][] intArray;  | intArray = new int[4][3]; | intArray[0][2] = 3; |
| int intArray [][];  |                           |                     |
| int [] intArray []; |                           |                     |

int

int

int []

[]

[]

a

arr

arr2

a의 타입?

arr의 타입?

arr이 저장하는 데이터의 타입?

arr2의 타입?

arr2에 저장하는 데이터의 타입?

## ❖ 2차원 Array 만들기 #2

- int Type 기준으로 4x3 배열 (Array) 과 값을 동시에 만들기

선언, 생성, 할당 동시에

```
int [][] intArray = { { 0, 1, 2 }, { 0, 1, 2 }, { 0, 1, 2 }, { 0, 1, 2 } }; // int intArray [], int [] intArray []
```

- {} 안에, 와 {} 을 이용해서 선언과 동시에 값을 할당



## ❖ 2차원 Array 만들기 #3

- int Type 기준으로 4x? 배열 (Array) 만들기

1,2차 선언 / 1차 생성

```
int [][] intArray = new int[4][]; // int intArray [], int [] intArray []
```

- 1차 Array 만 생성 후, 필요에 따라 2차 배열을 생성함

2차 생성

```
intArray[1] = new int[2];
intArray[0] = new int[4];
intArray[2] = {1,2,3}; (X)
```

## ❖ 다차원 배열

### ● 2차원 배열의 메모리 사용 단계

```
int a = 10;
```

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

```
int [][] arr2 = new int[2][];
```

```
arr2[0] = new int [3];
```

```
arr2[1] = new int [3];
```

```
arr2[1][1] = 100;
```

**a**



**arr**



**arr2**



## ❖ 기본 문제 #15

- 아래 표와 같이 4x3 = 12 개의 영역으로 나누어 'A', 'B', 'C' 로 등급을 나누려고 한다. 이를 Array를 이용하여 표현하고, 표 대로 출력하는 코드를 작성하시오.

|   |   |   |
|---|---|---|
| C | A | A |
| C | C | B |
| B | A | B |
| C | C | C |

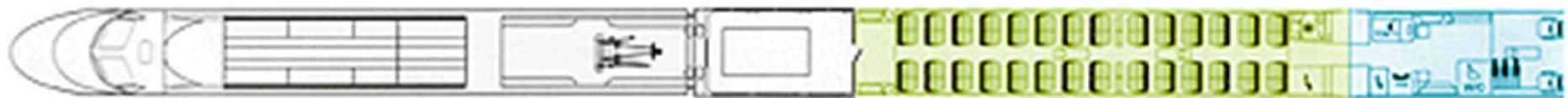
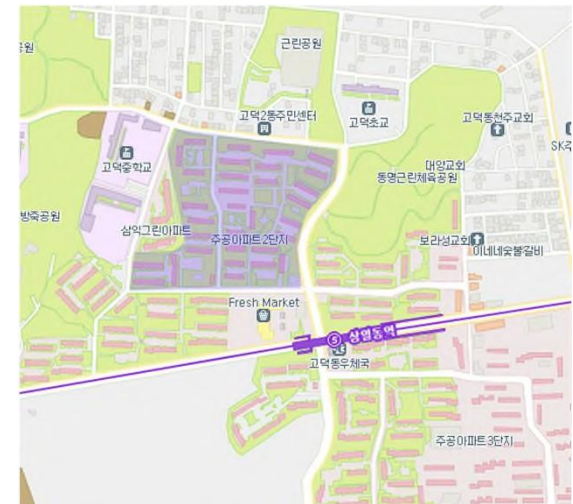
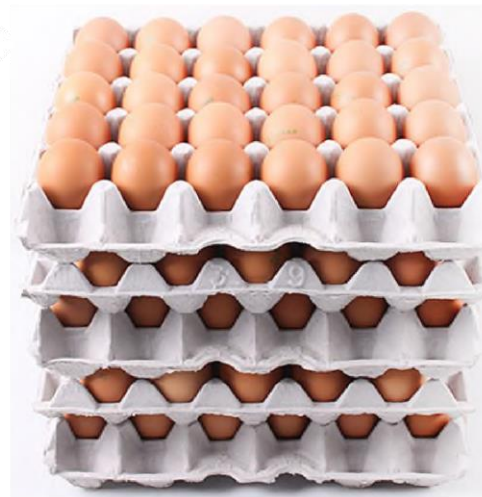
```
public static void main(String[] args) {
 char[][] grid = {{'C', 'A', 'A'},
 {'C', 'C', 'B'},
 {'B', 'A', 'B'},
 {'C', 'C', 'C'}};
 for(char [] chars: grid) {
 for(char c: chars) {
 System.out.print(c);
 }
 System.out.println();
 }
}
```

```
CAA
CCB
BAC
CCC
```

# JAVA 기본 및 응용

❖ 다음은 무엇에 대한 몇 차원 배열일까?

**Confidential**



### ❖ 기본 문제 #16 배열의 타입

### ❖ 다음과 같은 변수의 선언 및 할당이 있을 때

```
char c = 'A';
char [] chars = new char [3];
char[] [] chars2 = new char [4][3];
```

- 각 변수의 타입을 이야기 하시오.
- chars2는 char를 저장할 수 있다? 없다?
- chars2[0]가 참조하는 배열에는 몇 개의 char를 저장할 수 있는가?
- int의 100차원 배열을 정의해본다면?

char, char [], char [][]

chars는 char[]을 저장한다.

3개

int 99차원 배열을 저장하는 배열

## ❖ 기본 문제 #17

- 다음은 5x5 의 2차원 배열을 나타낸다. 각 항목의 숫자 중 3의 배수의 수와 합을 구하는 코드를 작성하시오.

|    |    |    |    |    |
|----|----|----|----|----|
| 2  | 3  | 1  | 4  | 7  |
| 8  | 13 | 3  | 33 | 1  |
| 7  | 4  | 5  | 80 | 12 |
| 17 | 9  | 11 | 5  | 4  |
| 4  | 5  | 91 | 27 | 7  |

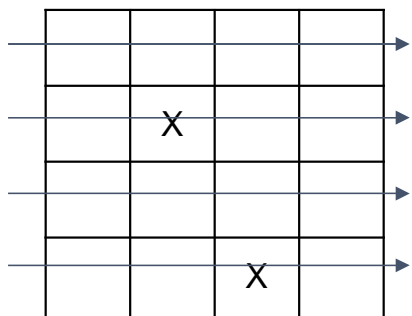
```
public static void main(String[] args) {

 int[][] grid = {
 {2, 3, 1, 4, 7}, {8, 13, 3, 33, 1},
 {7, 4, 5, 80, 12}, {17, 9, 11, 5, 4},
 {4, 5, 91, 27, 7}
 };

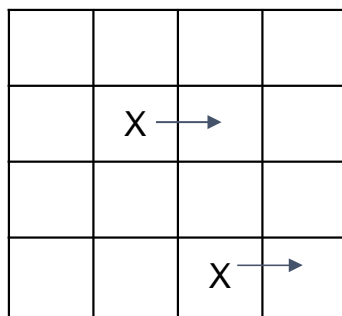
 int count = 0;
 int sum = 0;
 for(int [] row: grid) {
 for(int num:row) {
 if(num%3==0) {
 count++;
 sum+=num;
 }
 }
 }
 System.out.printf("개수: %d, 총합: %d\n", count, sum);
}
```

개수: 6, 총합: 87

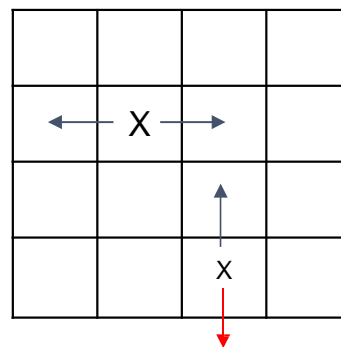
## ❖ Array 순회 / 탐색



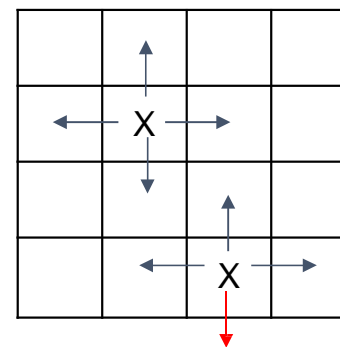
전체 중 X를 만나면



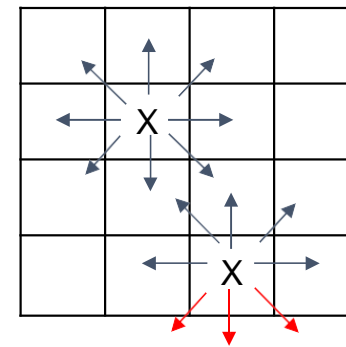
X가 움직이면서



X 주변 탐색  
좌/우  
상/하



X 주변 탐색  
4방



X 주변 탐색  
8방

특정 좌표로부터 주변을 탐색하는 경우, 배열의 범위를 벗어나지 않기 위한 코드 필요

## ❖ 기본 문제 #18

- 배열은 index 로 요소( element )에 접근한다. 만약, index 의 범위가 벗어나면 발생하는 예외( Exception )은?

```
java.lang.ArrayIndexOutOfBoundsException
```



## ❖ 기본 문제 #19

- 다음은 4x4 의 2차원 배열을 나타낸다. x 로 표시된 항목의 좌우 숫자의 합을 구하는 코드를 작성하시오.

|   |   |   |   |
|---|---|---|---|
| 2 | 3 | 1 | 4 |
| 1 | X | 3 | 2 |
| 3 | 4 | X | X |
| X | 4 | 1 | 5 |

```
int sum = 0;
for (int r = 0; r < 4; r++)
 for (int c = 0; c < 4; c++)
 if (grid[r][c] == 'X') {
 // 왼쪽 살펴보기
 if (c - 1 >= 0 && grid[r][c - 1] != 'X')
 sum += grid[r][c - 1] - '0';
 // 오른쪽 살펴보기
 if (c + 1 < 4 && grid[r][c + 1] != 'X')
 sum += grid[r][c + 1] - '0';
 }
System.out.println(sum);
```

## ❖ 기본 문제 #20

- 다음은 4x4 의 2차원 배열을 나타낸다. X 로 표시된 항목의 상하좌우 숫자의 합을 구하는 코드를 작성하시오. 단, 합을 구할 때, 이미 사용된 숫자는 다시 사용하지 않음

|   |   |   |   |
|---|---|---|---|
| 2 | 3 | 1 | 4 |
| 1 | X | 3 | 2 |
| 3 | 4 | X | X |
| X | 4 | 1 | 5 |

위의 X 와 우측의 X 에 모두 인접 하지만, 한번만 더함.

```
int[][] deltas = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};

int sum = 0;
for (int r = 0; r < 4; r++)
 for (int c = 0; c < 4; c++)
 if (grid[r][c] == 'X') {
 for (int d = 0; d < 4; d++) {
 int nr = r + deltas[d][0];
 int nc = c + deltas[d][1];
 if (nr >= 0 && nr < 4 && nc >= 0 && nc < 4 && grid[nr][nc] != 'X') {
 sum += grid[nr][nc] - '0';
 grid[nr][nc] = '0';
 }
 }
 }
System.out.println(sum);
```