

Loop, Array, and Exception

Byeongjoon Noh

Dept. of AI/Bigdata, SCH Univ.

powernoh@sch.ac.kr

Contents

1. Loop
2. Nested Loop
3. Array
4. Exception handling

1. Loop

반복문 (Loop)

세 가지 형태의 반복문

- for문 →
- while문
- do-while문

1 2 4
`for(초기문; 조건식; 반복 후 작업) {`
 `.. 작업문 ..`
 3
`}`

```
// 0에서 9까지 출력  
for(int i=0; i<10; i++) {  
    System.out.print(i);  
}
```

0123456789

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

반복후 작업문에 콤마로 분리하여
2 문장 작성가능

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

for 문안에서만 사용되는
변수 i 선언 가능

```
for(초기문; true; 반복 후 작업) { // 무한반복  
    .....  
}
```

```
for(초기문; ; 반복 후 작업) { // 무한 반복  
    .....  
}
```

Example

Q. For문을 이용하여 1~10까지 자연수의 덧셈을 표시하고 합을 출력해보세요.

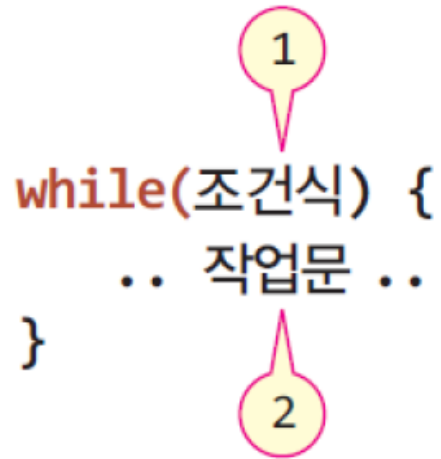
```
public class ForSample {  
    public static void main(String[] args) {  
        int i, sum=0;  
  
        for(i=1; i<=10; i++) { // 1~10까지 반복  
            sum += i;  
            System.out.print(i); // 더하는 수 출력  
  
            if(i<=9) // 1~9까지는 '+' 출력  
                System.out.print("+");  
            else { // i가 10인 경우  
                System.out.print("="); // '=' 출력하고  
                System.out.print(sum); // 덧셈 결과 출력  
            }  
        }  
    }  
}
```

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

while문

while문의 구성과 특징

- 조건식이 '참' 인 동안 반복 실행



A diagram illustrating the structure of a while loop. It shows the text `while(조건식) {` followed by `.. 작업문 ..` and then a closing brace `}`. A pink callout bubble with the number '1' points to the opening curly brace, and another pink callout bubble with the number '2' points to the closing curly brace.

```
while(조건식) {  
    .. 작업문 ..  
}
```

```
int i=0;  
while(i<10) { // 0에서 9까지 출력  
    System.out.print(i);  
    i++;  
}
```

0123456789

Example

Q. while문을 이용하여 정수 여러 개 입력받고 그 평균을 출력해보세요.
0이 입력되면 종료합니다.

```
import java.util.Scanner;
public class WhileSample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int count=0, n=0;
        double sum=0;

        System.out.println("정수를 입력하고 마지막에 0을 입력하세요.");
        while((n = scanner.nextInt()) != 0) { // 0이 입력되면 while 문 벗어남
            sum = sum + n;
            count++;
        }
        System.out.print("수의 개수는 " + count + "개이며 ");
        System.out.println("평균은 " + sum/count + "입니다.");

        scanner.close();
    }
}
```

정수를 입력하고 마지막에 0을 입력하세요.

10 30 -20 40 0

수의 개수는 4개이며 평균은 15.0입니다.

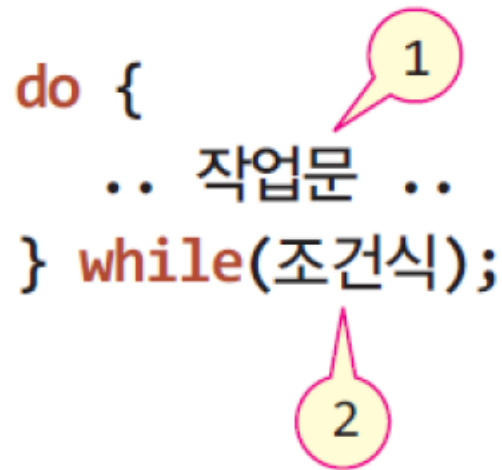
0은 마지막 입력을 뜻함

do-while문

do-while문의 구성과 특징

- 조건식이 '참' 인 동안 반복 실행
- 작업문은 한 번 반드시 실행

`do {`
 ..`작업문`..
`} while(조건식);`



```
int i=0;
do { // 0에서 9까지 출력
    System.out.print(i);
    i++;
} while(i<10);
```

0123456789

Example

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

```
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

중첩 반복

반복문이 다른 반복문을 내포하고 있는 구조

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

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

Example

2중 중첩된 for문을 이용하여 구구단을 출력해보세요.

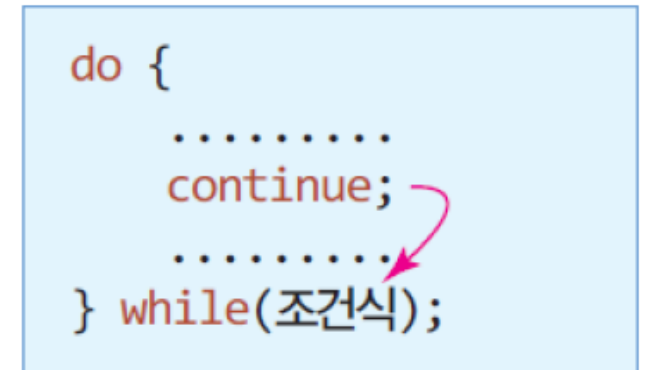
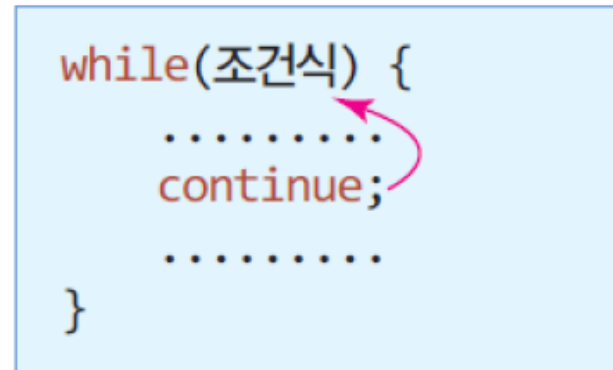
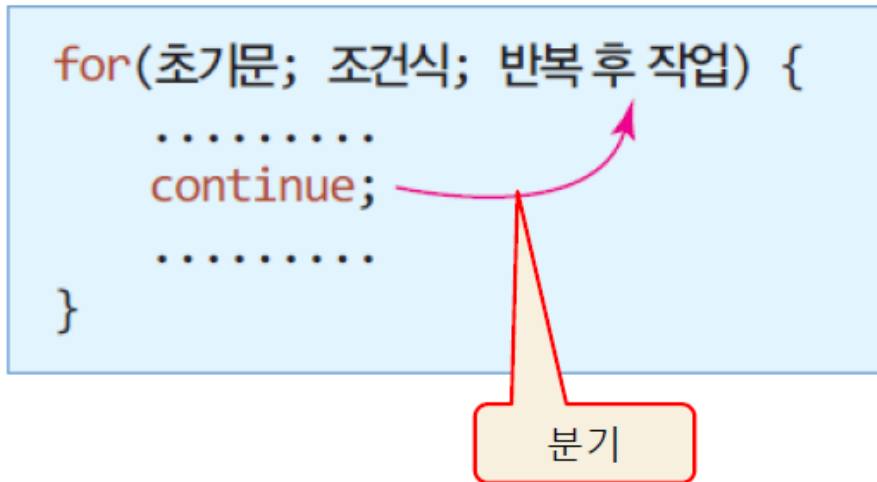
```
public class NestedLoop {  
    public static void main(String[] args) {  
  
        for(int i=1; i<10; i++) { // 단에 대한 반복. 1단에서 9단  
            for(int j=1; j<10; j++) { // 각 단의 곱셈  
                System.out.print(i + "*" + j + "=" + i*j); // 구구셈 출력  
                System.out.print('\t'); // 하나씩 탭으로 띄기  
            }  
            System.out.println(); // 한 단이 끝나면 다음 줄로 커서 이동  
        }  
    }  
}
```

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

continue문

Continue

- 반복문을 빠져나가지 않고, 다음 반복으로 제어 변경
- 반복문에서 continue 문에 의한 분기 예제



Example

Q. 정수 5개를 입력받고, 양수인 경우에만 그 합을 구하여 출력해보세요.

```
import java.util.Scanner;
public class ContinueExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("정수를 5개 입력하세요.");
        int sum=0;

        for(int i=0; i<5; i++) {
            int n=scanner.nextInt();
            if(n<=0) continue; // 0이나 음수인 경우 더하지 않고 다음 반복으로 진행
            else sum += n; // 양수인 경우 덧셈
        }
        System.out.println("양수의 합은 " + sum);

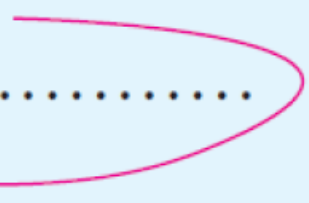
        scanner.close();
    }
}
```

break문

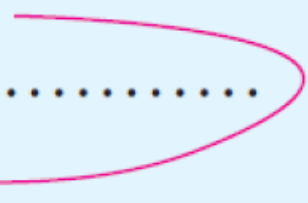
Break

- 반복문 “하나”를 즉시 벗어날 때 사용
- 중첩 반복의 경우 안쪽 반복문에서 실행되면, 안쪽 반복문만 벗어남

```
for(초기문; 조건식; 반복 후 작업) {  
    .....  
    break;  
    .....  
}  
.....
```

A pink arrow originates from the 'break;' statement and points to the closing curly brace of the for loop, indicating that the loop is exited immediately.

```
for(초기문; 조건식; 반복 후 작업) {  
    while(조건식) {  
        .....  
        break;  
        .....  
    }  
    .....  
}  
.....
```

A pink arrow originates from the 'break;' statement inside the while loop and points to the closing curly brace of the while loop, indicating that only the inner loop is exited. Another pink arrow originates from the closing curly brace of the while loop and points to the closing curly brace of the for loop, indicating that the outer loop continues.

Example

Q. “exit”이 입력되면 while문이 벗어나도록 break문을 활용해보세요.

```
import java.util.Scanner;
public class BreakExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("exit을 입력하면 종료합니다.");

        while(true) {
            System.out.print("> >");
            String text = scanner.nextLine();
            if(text.equals("exit")) // "exit"이 입력되면 반복 종료
                break; // while 문을 벗어남
        }

        System.out.println("종료합니다...");
        scanner.close();
    }
}
```

```
exit을 입력하면 종료합니다.
>>edit
>>exit
종료합니다...
```

2. Nested Loop

중첩 for loop

for loop 내에 1개 이상의 for loop가 존재하는 형태

```
// outer loop
for (int i = 1; i <= 5; ++i) {
    System.out.println("i = " + i);

    // inner loop
    for(int j = 1; j <=2; j++) {
        System.out.println("j = " + j);
    }
}
```

```
i = 1
j = 1
j = 2
i = 2
j = 1
j = 2
i = 3
j = 1
j = 2
i = 4
j = 1
j = 2
i = 5
j = 1
j = 2
```

Example 1 – Weeks and Days

```
int weeks = 3;
int days = 7;

// outer loop prints weeks
for (int i = 1; i <= weeks; ++i) {
    System.out.println("Week: " + i);

    // inner loop prints days
    for (int j = 1; j <= days; ++j) {
        System.out.println("    Day: " + j);
    }
}
```

```
Week: 1
    Day: 1
    Day: 2
    Day: 3
    Day: 4
    Day: 5
    Day: 6
    Day: 7
Week: 2
    Day: 1
    Day: 2
    Day: 3
    Day: 4
    Day: 5
    Day: 6
    Day: 7
Week: 3
    Day: 1
    Day: 2
    Day: 3
    Day: 4
    Day: 5
    Day: 6
    Day: 7
```

Example 2 - * Shape

```
final int MAXROWS = 4, MAXCOLS = 5;

for (int i = 1; i <= MAXROWS; i++) {
    for (int j = 1; j <= MAXCOLS; j++){
        System.out.print("*");
    }
    System.out.println();
}
```

```
*****
*****
*****
*****
```

Example 3 - * Shape

```
final int MAXROWS = 4, MAXCOLS = 5;

for (int i = 1; i <= MAXROWS; i++) {
    for (int j = 1; j <= MAXCOLS; j++){
        System.out.print((i-1)*MAXCOLS+j + "\t");
    }
    System.out.println();
}
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

3. Array

배열 (Array)

Index와 index에 대응하는 데이터들로 이루어진 자료 구조

- 한 번에 많은 메모리 공간 선언 가능

배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간

- Element들이 순차적으로 저장됨
- Index를 이용하여 element에 접근
- 반복문을 이용하여 처리하기에 적합한 자료 구조

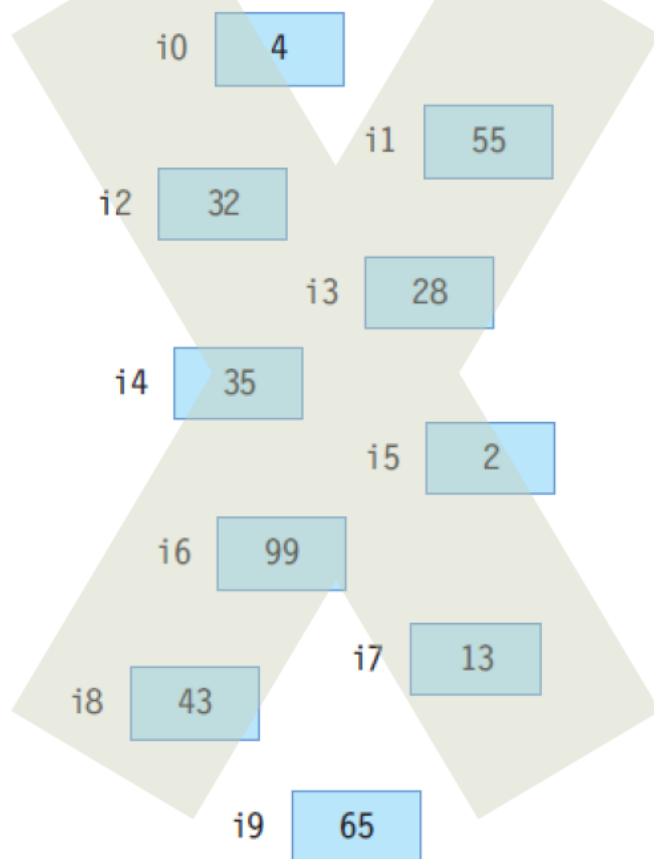
Index

- 0부터 시작!
- Index는 배열의 시작 위치에서부터 데이터가 있는 상대 위치

배열의 필요성

(1) 10개의 정수형 변수를 선언하는 경우

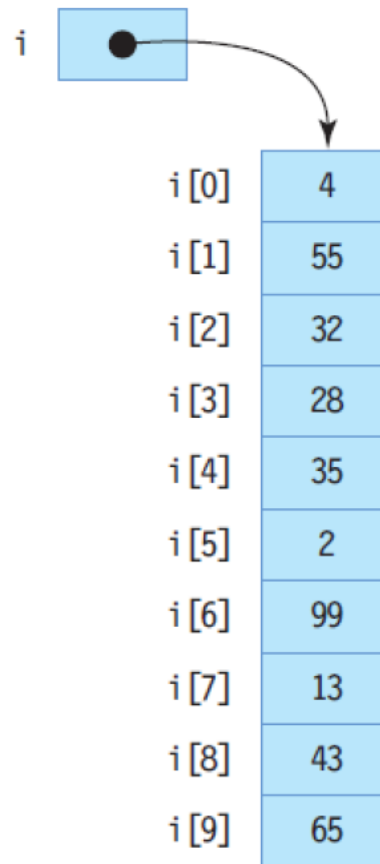
```
int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;
```



```
sum = i0+i1+i2+i3+i4+i5+i6+i7+i8+i9;
```

(2) 10개의 정수로 구성된 배열을 선언하는 경우

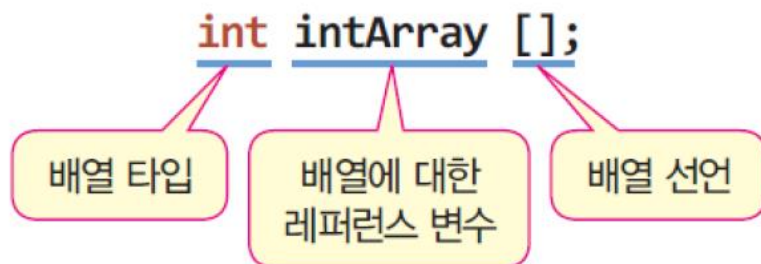
```
int i[] = new int[10];
```



```
for(sum=0, n=0; n<10; n++)  
    sum += i[n];
```

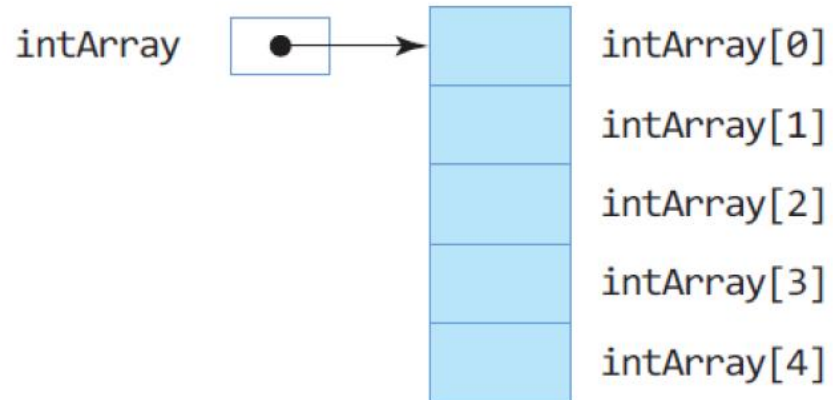
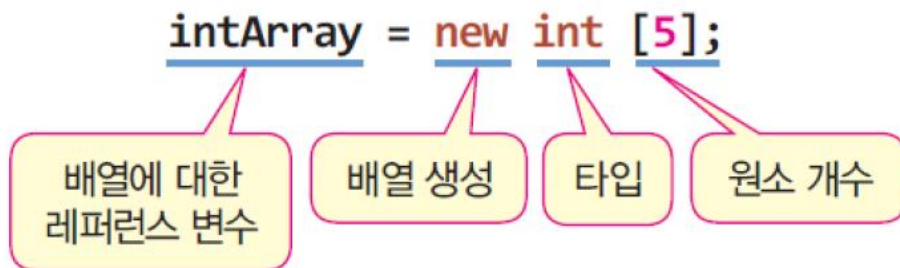
배열의 선언과 생성

(1) 배열에 대한 레퍼런스 변수 intArray 선언



intArray

(2) 배열 생성




배열의 선언과 생성

배열 선언

- 배열의 이름 선언 (배열 래퍼런스 변수 선언)

```
int intArray []; 또는  
int[] intArray;
```

 `int intArray [5];` // 크기 지정 안됨

배열 생성

- 배열 공간 할당 받는 과정

```
intArray = new int[5]; 또는  
int intArray[] = new int[5]; // 선언과 동시에 배열 생성
```

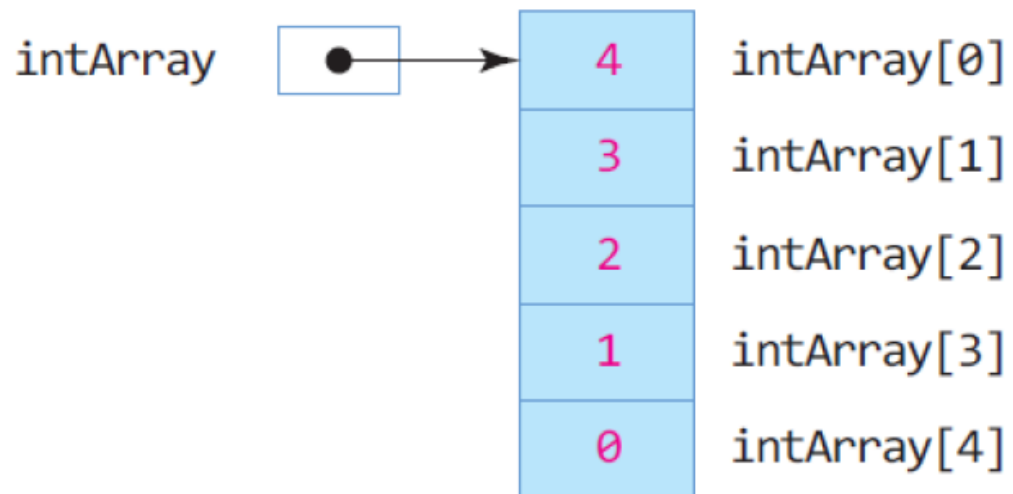
배열 초기화 ➔ 값 할당

```
int intArray[] = {4, 3, 2, 1, 0}; // 5개의 정수 배열 생성 및 값 초기화  
double doubleArray[] = {0.01, 0.02, 0.03, 0.04}; // 5개의 실수 배열 생성 및 값 초기화
```

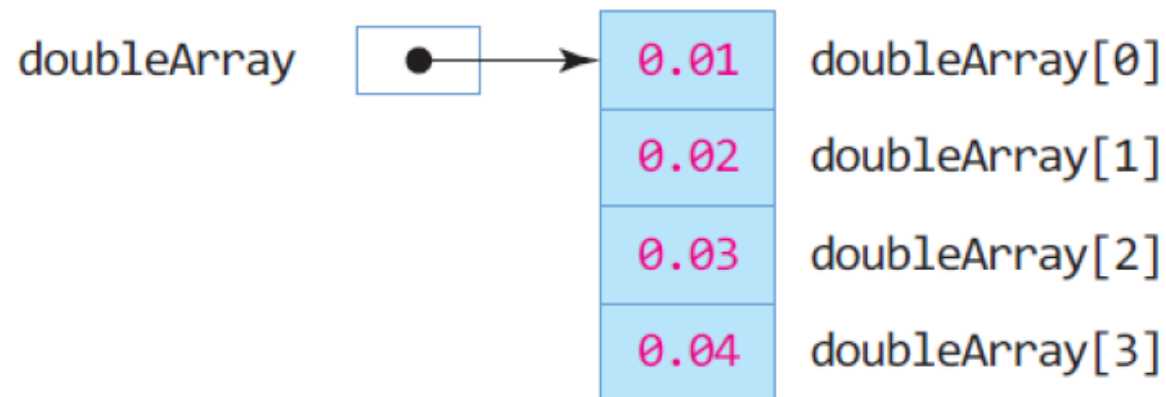
배열의 선언과 생성

배열을 초기화 하면서 생성한 결과

```
int intArray[] = {4, 3, 2, 1, 0};
```



```
double doubleArray[] = {0.01, 0.02, 0.03, 0.04};
```




배열 인덱스와 접근

Index

- 0 ~ (배열 크기 -1)

```
int intArray = new int[5];           // 인덱스는 0~4까지 가능
intArray[0] = 5;                      // 원소 0에 5 저장
intArray[3] = 6;                      // 원소 3에 6 저장
int n = intArray[3];                  // 원소 3의 값을 읽어 n에 저장
```


- Index 잘못 사용한 경우



```
int n = intArray[-2];           // 인덱스로 음수 사용 불가
int m = intArray[5];            // 5는 인덱스의 범위(0~4) 넘었음
```

- 반드시 배열 생성 후 접근

```
int intArray []; // 레퍼런스만 선언함
```



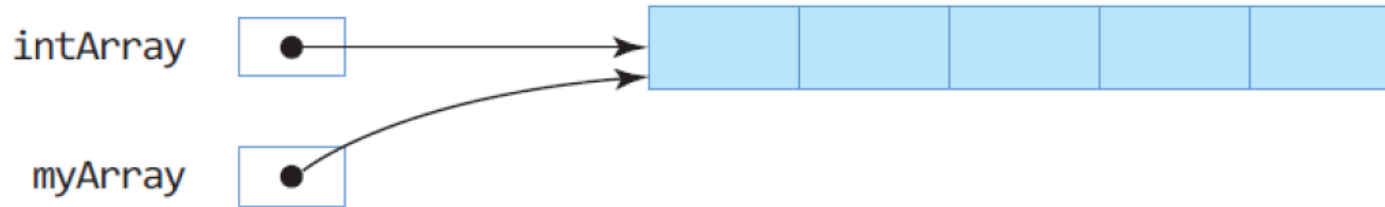
```
intArray[1] = 8; // 오류. 배열이 생성되어 있지 않음
```

레퍼런스 치환과 배열 공유

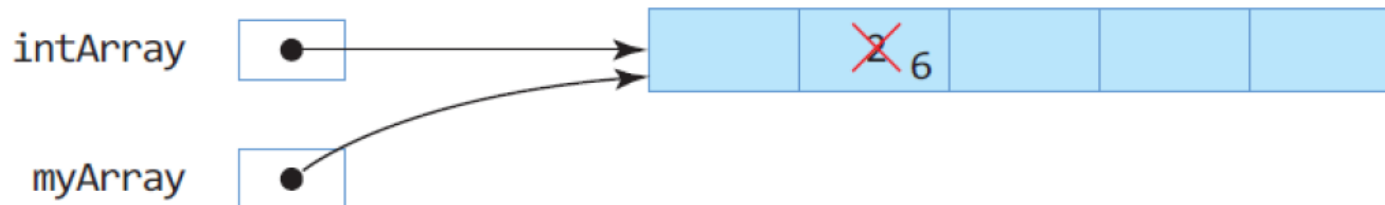
레퍼런스 치환으로 두 레퍼런스가 하나의 배열 공유

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

레퍼런스 치환으로 배열 공유



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



Example

Q. 양수 5개를 입력받아 배열에 저장하고, 제일 큰 수를 출력해보세요.

```
import java.util.Scanner;
public class ArrayAccess {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int intArray[];
        intArray = new int[5];
        int max=0; // 현재 가장 큰 수
        System.out.println("양수 5개를 입력하세요.");

        for(int i=0; i<5; i++) {
            intArray[i] = scanner.nextInt();    // 입력 받은 정수를 배열에 저장
            if(intArray[i] > max)
                max = intArray[i]; // max 변경
        }
        System.out.print("가장 큰 수는 " + max + "입니다.");

        scanner.close();
    }
}
```

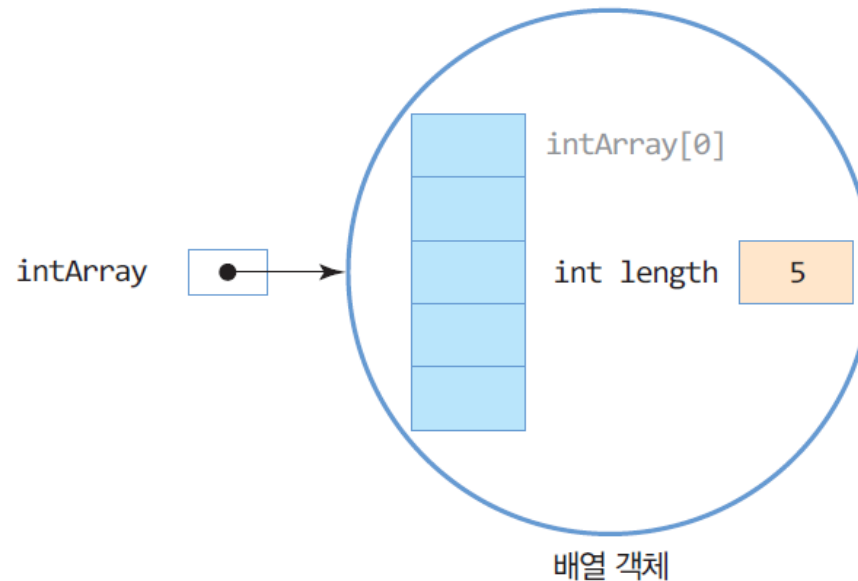
양수 5개를 입력하세요.
1 39 78 100 99
가장 큰 수는 100입니다..

배열의 크기

Java에서 배열은 객체로 처리됨

- 배열 객체의 length 필드 활용하여 배열의 크기를 handling

```
int intArray[];  
intArray = new int[5];  
  
int size = intArray.length;  
// size는 5
```



- 활용

```
for(int i=0; i<intArray.length; i++) // intArray 배열 크기만큼 루프를 돈다.  
    System.out.println(intArray[i]);
```

Example

Q. 배열의 length 필드를 이용하여 배열 크기만큼 정수를 입력받고 평균을 출력해보세요.

```
import java.util.Scanner;
public class ArrayLength {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("5개의 정수를 입력하세요.");
        int intArray[] = new int[5];

        double sum=0.0;
        for(int i=0; i<intArray.length; i++)
            intArray[i] = scanner.nextInt(); // 키보드에서 입력받은 정수 저장

        for(int i=0; i<intArray.length; i++)
            sum += intArray[i]; // 배열에 저장된 정수 값을 더하기

        System.out.print("평균은 " + sum/intArray.length);
        scanner.close();
    }
}
```

5개의 정수를 입력하세요.
2 3 4 5 9
평균은 4.6.

for-each문

배열이나 나열 (Enumeration)의 원소를 순차 접근하는데 유용함

1
for(변수 : 배열레퍼런스) {
 .. 반복작업문 ..
}

2

- 활용

```
int [] n = { 1,2,3,4,5 };  
int sum = 0;  
for (int k : n) {  
    sum += k;  
}
```

반복될 때마다 k는 n[0], n[1], ..., n[4]로 번갈아 설정

for 문으로 구성하면 다음과 같다.

```
for(int i=0; i<n.length; i++) {  
    int k = n[i];  
    sum += k;  
}
```


Example

Q. for-each문을 활용하여 int [] 배열의 합과 String[]배열의 문자열을 출력해보세요

```
public class foreachEx {  
    public static void main(String[] args) {  
        int [] n = { 1,2,3,4,5 };  
        int sum=0;  
        for(int k : n) { // k는 n[0], n[1], ..., n[4]로 반복  
            System.out.print(k + " "); // 반복되는 k 값 출력  
            sum += k;  
        }  
        System.out.println("합은 " + sum);  
  
        String f[] = { "사과", "배", "바나나", "체리", "딸기", "포도" };  
        for(String s : f) // s는 f[0], f[1], ..., f[5]로 반복  
            System.out.print(s + " ");  
    }  
}
```

1 2 3 4 5 합은 15
사과 배 바나나 체리 딸기 포도

2차원 배열

2차원 배열 선언

```
int intArray[][];    또는   int[][] intArray;
```

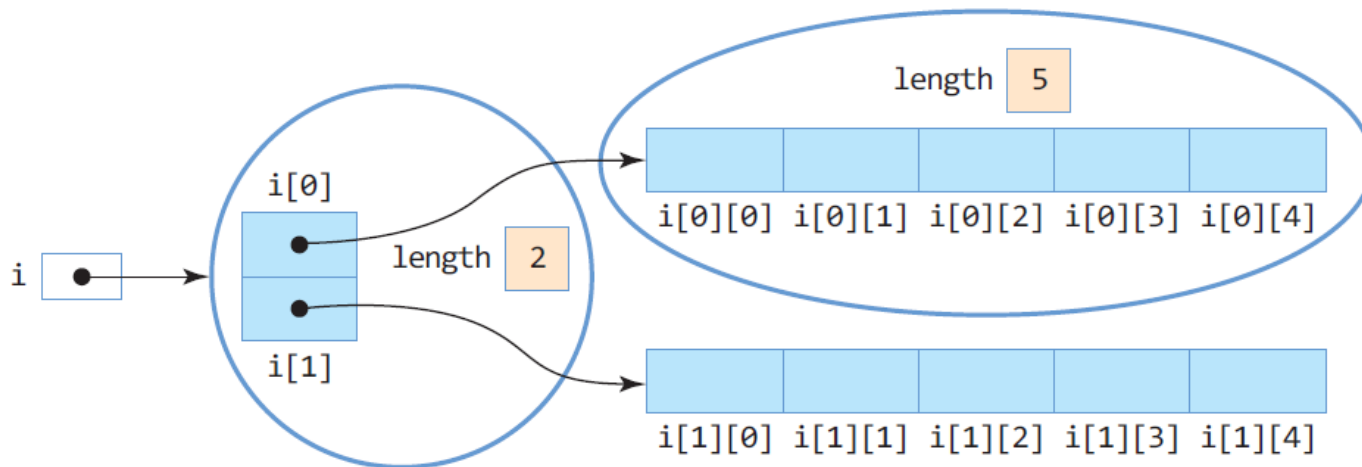
2차원 배열 생성

```
intArray = new int[2][5];
```

```
int intArray[] = new int[2][5]; // 배열 선언과 생성 동시
```

2차원 배열의 구조

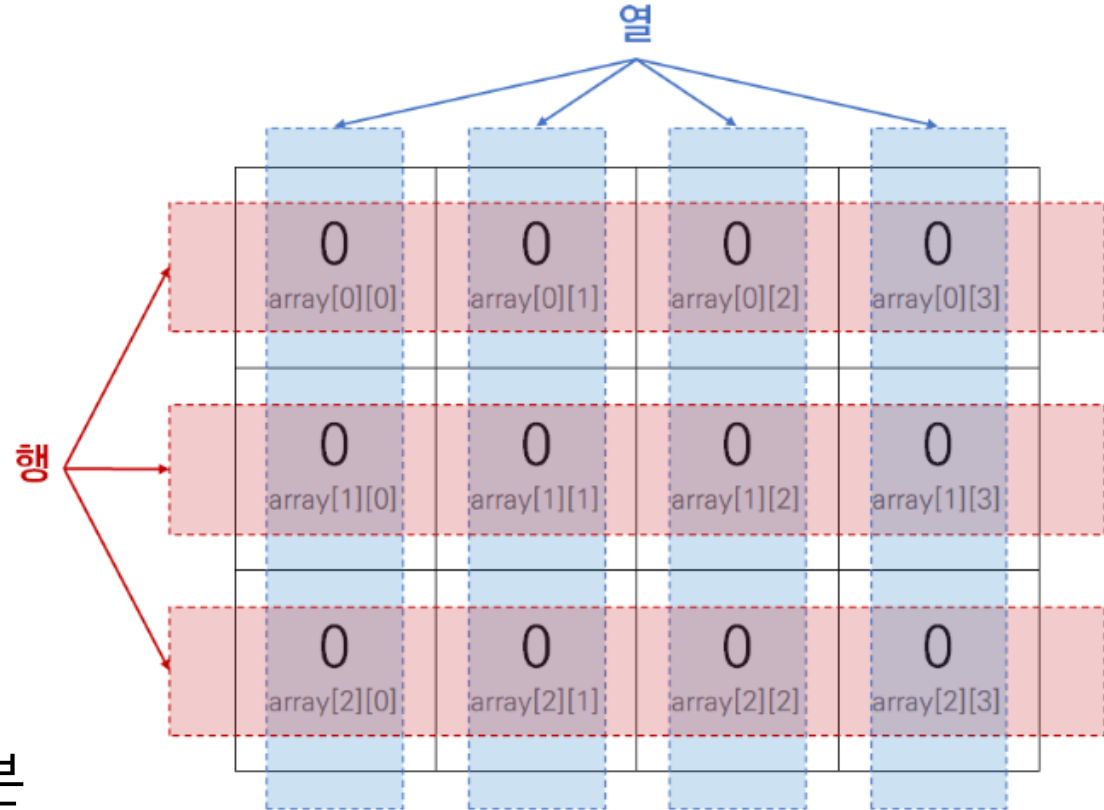
```
int i[][] = new int[2][5];  
int size1 = i.length; // 2  
int size2 = i[0].length; // 5  
int size3 = i[1].length; // 5
```



2차원 배열

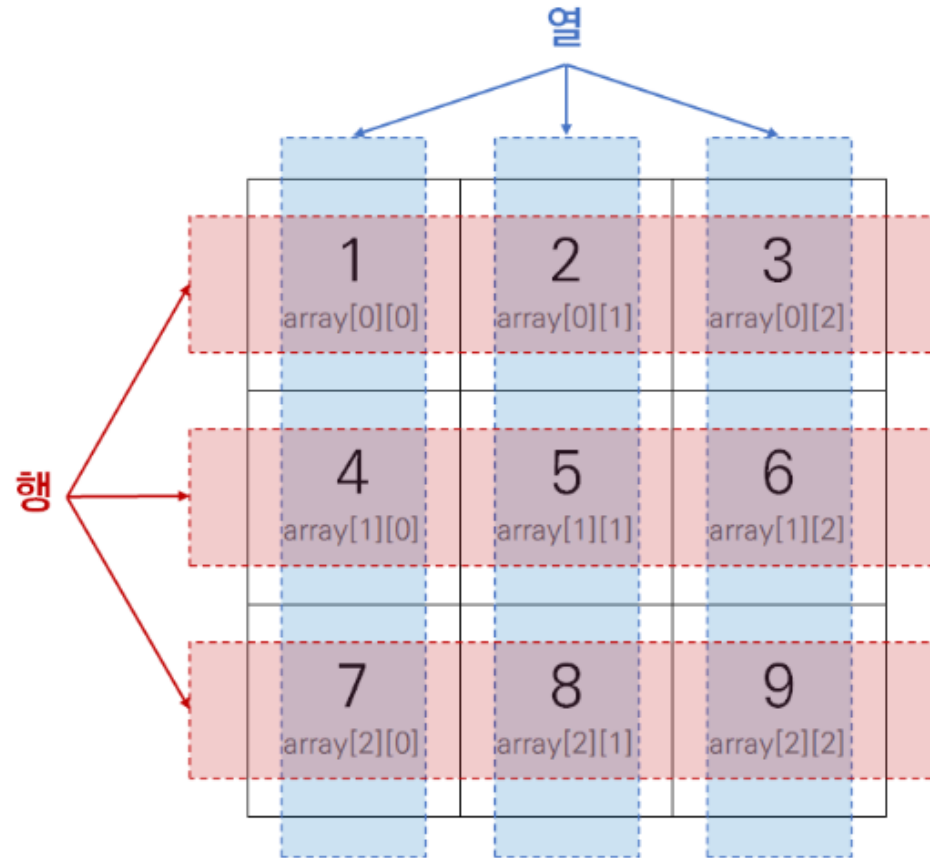
```
int[][] array = new int[3][4]
```

Array는 0으로 초기화되는 것이 기본



2차원 배열

```
int[][] array = new int[][] { { 1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```



2차원 배열 초기화

배열 선언과 동시에 초기화

```
int intArray[][] = { { 0, 1, 2},  
                    { 3, 4, 5},  
                    { 6, 7, 8} }; // 3x3 배열 생성
```

```
char charArray[][] = { {'a', 'b', 'c'}, {'d', 'e', 'f'} }; // 2x3 배열 생성
```

```
double doubleArray[][] = { {0.01, 0.02}, {0.03, 0.04} }; // 2x2 배열 생성
```

2차원 배열

2차원 배열의 length 필드

- 2차원 배열 arr2D[4][5]
- arr2D.length → 2차원 배열의 행 개수 4
- arr2D[n].length → n번째 행의 열의 개수 5
- arr2D[2].length → 3번째 행의 열의 개수 5

Example

Q. 2차원 배열에 학년별로 1, 2학기 성적을 저장하고, 4년 전체 평점 평균을 출력하세요.

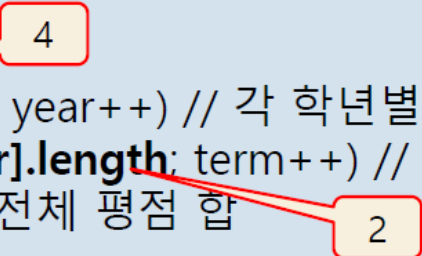
```
public class ScoreAverage {
    public static void main(String[] args) {
        double score[][] = { {3.3, 3.4}, // 1학년 1, 2학기 평점
                              {3.5, 3.6}, // 2학년 1, 2학기 평점
                              {3.7, 4.0}, // 3학년 1, 2학기 평점
                              {4.1, 4.2} }; // 4학년 1, 2학기 평점

        double sum=0;

        for(int year=0; year<score.length; year++) // 각 학년별로 반복
            for(int term=0; term<score[year].length; term++) // 각 학년의 학기별로 반복
                sum += score[year][term]; // 전체 평점 합

        int n=score.length; // 배열의 행 개수, 4
        int m=score[0].length; // 배열의 열 개수, 2

        System.out.println("4년 전체 평점 평균은 " + sum/(n*m));
    }
}
```



4년 전체 평점 평균은 3.725

2차원 배열 for-each 활용

```
int[][] array2D = { {79, 80, 81},  
                    {83, 65, 90},  
                    {80, 92, 100},  
                    {100, 92, 98} };
```

```
for (int[] innerArray : array2D) {  
    for (int ele : innerArray) {  
        System.out.println(ele);  
    }  
}
```


Example 4 – Matrix Multiplication

```
//creating two matrices
int x[][] = {{3,2,7},{1,6,2},{7,2,3}};
int y[][] = {{3,2,8},{2,4,6},{2,3,6}};

int z[][] = new int[3][3];

for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {

        z[i][j]=0;

        for(int k = 0; k < 3; k++) {
            z[i][j] += x[i][k] * y[k][j];
        }

        System.out.print(z[i][j] + "\t");
    }

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

27	35	78
19	32	56
31	31	86

4. Exception handling

예외 처리

예외 (Exception)

- 실행 중 오동작이나 결과에 악영향을 미치는 예상치 못한 상황 발생
- ➔ 실행 중 발생하는 에러를 예외로 처리

실행 중 예외가 발생하면

- Java platform에서 응용프로그램이 예외를 처리하도록 호출
 - 예외처리하지 않으면 강제 종료됨

예외 발생 경우

- 정수를 0으로 나누는 경우
- 배열의 크기보다 큰 index로 배열의 element를 접근하는 경우
- 정수를 읽는 코드가 실행될 때 사용자가 문자를 입력한 경우

Example

Q. 두 정수를 입력받아 나누는 프로그램에서, 사용자가 분모에 0을 입력하는 경우

```
import java.util.Scanner;
public class DivideByZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int dividend; // 나뉘는수
        int divisor; // 나눗수

        System.out.print("나뉘는수를 입력하십시오:");
        dividend = scanner.nextInt(); // 나뉘는수 입력
        System.out.print("나눗수를 입력하십시오:");
        divisor = scanner.nextInt(); // 나눗수 입력
        System.out.println(dividend + "를 " + divisor + "로 나누면 몫은 "
            + dividend/divisor + "입니다.");
    }
}
```

divisor가 0이므로
ArithmeticException예외 발생

나뉘는수를 입력하십시오:100

나눗수를 입력하십시오:0

Exception in thread "main" java.lang.ArithmeticException: / by zero
at DivideByZero.main(DivideByZero.java:13)

예외 처리 방법

예외 처리란

- 발생한 예외에 대해 개발자가 작성한 프로그램 코드에서 대응하는것
- try-catch-finally 문을 활용
- finally 블록은 생략 가능

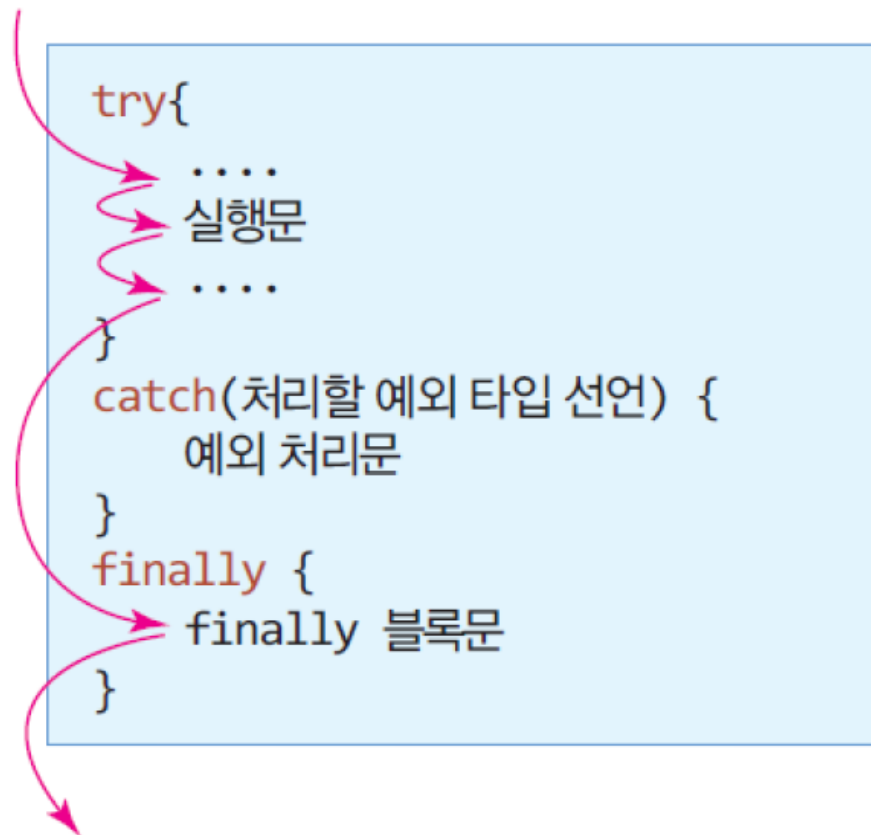
```
try {  
    예외가 발생할 가능성이 있는 실행문(try 블록)  
}  
catch (처리할 예외 타입 선언) {  
    예외 처리문(catch 블록)  
}  
finally {  
    예외 발생 여부와 상관없이 무조건 실행되는 문장(finally 블록)  
}
```

} 생략 가능

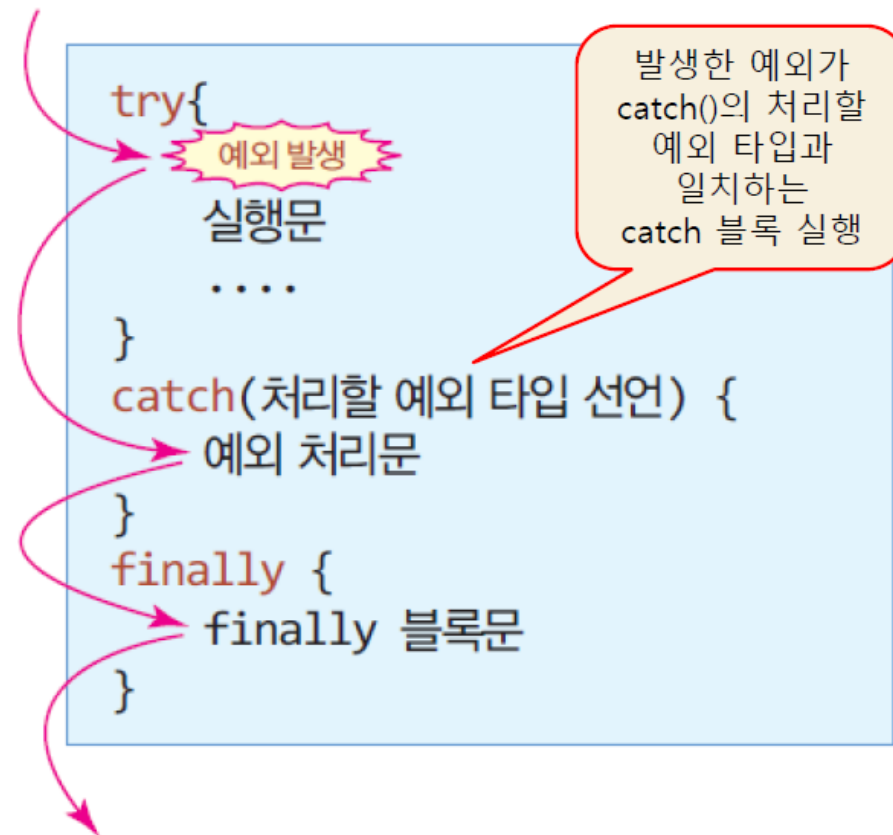
예외 처리 방법

예외 발생 시 제어 흐름

try 블록에서 예외가 발생하지 않은 정상적인 경우



try 블록에서 예외가 발생한 경우



Java 예외 클래스

Java platform은 응용프로그램이 실행 중 오류를 탐지할 수 있도록 다양한 예외를 클래스 형태로 제공

예외 타입(예외 클래스)	예외 발생 경우	패키지
ArithmeticException	정수를 0으로 나눌 때 발생	java.lang
NullPointerException	null 레퍼런스를 참조할 때 발생	java.lang
ClassCastException	변환할 수 없는 타입으로 객체를 변환할 때 발생	java.lang
OutOfMemoryError	메모리가 부족한 경우 발생	java.lang
ArrayIndexOutOfBoundsException	배열의 범위를 벗어난 접근 시 발생	java.lang
IllegalArgumentException	잘못된 인자 전달 시 발생	java.lang
IOException	입출력 동작 실패 또는 인터럽트 시 발생	java.io
NumberFormatException	문자열이 나타내는 숫자와 일치하지 않는 타입의 숫자로 변환 시 발생	java.lang
InputMismatchException	Scanner 클래스의 nextInt()를 호출하여 정수로 입력받고자 하였지만, 사용자가 'a' 등과 같이 문자를 입력한 경우	java.util

예외 클래스 활용

Ex) 배열의 범위를 벗어나 원소를 접근하는 예외 처리

- `ArrayIndexOutOfBoundsException`

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

```
try {  
    intArray[3] = 10; // 예외 발생하지 않음  
    intArray[6] = 5; // 예외 발생  
}
```

이 문장 실행 시
`ArrayIndexOutOfBoundsException`
예외 발생

```
catch(ArrayIndexOutOfBoundsException e) { // 객체 e에 예외 정보가 넘어옴  
    System.out.println("배열의 범위를 초과하여 원소를 접근하였습니다.");  
}
```


Example

Q. 두 정수를 입력받아 나누는 프로그램을 작성하세요.

단, 사용자가 분모에 0을 입력하는 경우 예외처리를 수행하세요.

```
import java.util.Scanner;
public class DevideByZeroHandling {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int dividend; // 나뉘는수
        int divisor; // 나눌수

        System.out.print("나뉘는수를 입력하시오:");
        dividend = scanner.nextInt(); // 나뉘는수 입력
        System.out.print("나눌수를 입력하시오:");
        divisor = scanner.nextInt(); // 나눌수 입력
        try {
            System.out.println(dividend+"를 " + divisor + "로 나누면 몫은 " + dividend/divisor + "입니다.");
        }
        catch(ArithmeticException e) { // ArithmeticException 예외 처리 코드
            System.out.println("0으로 나눌 수 없습니다!");
        }
        finally {
            scanner.close(); // 정상적이든 예외가 발생하든 최종적으로 scanner를 닫는다.
        }
    }
}
```

divisor가 0인 경우
ArithmeticException 예외 발생

나뉘는수를 입력하시오:100
나눌수를 입력하시오:0
0으로 나눌 수 없습니다.

ArithmeticException 예외가 발생해도
프로그램이 강제 종료되지 않고 정상 실행됨

Example

Q. 3개의 정수를 입력받아 합을 구하는 프로그램을 작성하세요. 단, 정수가 아닌 문자가 입력될 때는 예외처리를 통해 **다시 입력받도록 하세요.**

```
import java.util.Scanner;
import java.util.InputMismatchException;

public class InputException {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("정수 3개를 입력하세요");
        int sum=0, n=0;
        for(int i=0; i<3; i++) {
            System.out.print(i+">>");
            try {
                n = scanner.nextInt(); // 정수 입력
            }
            catch(InputMismatchException e) {
                System.out.println("정수가 아닙니다. 다시 입력하세요!");
                scanner.next(); // 입력 스트림에 있는 정수가 아닌 토큰을 버린다.
                i--; // 인덱스가 증가하지 않도록 미리 감소
                continue; // 다음 루프
            }
            sum += n; // 합하기
        }
        System.out.println("합은 " + sum);
        scanner.close();
    }
}
```

사용자가 문자를 입력하면
InputMismatchException 예외 발생

정수 3개를 입력하세요

0>>5

1>>R

정수가 아닙니다. 다시 입력하세요!

1>>4

2>>6

합은 15

End of slide