Array and String

Java Programming

Byeongjoon Noh

powernoh@sch.ac.kr

SCH SOON CHUN HYANG UNIVERSITY

Most of the slides are available on Senseable AI Lab homepage: https://sailab.space/courses/

Contents

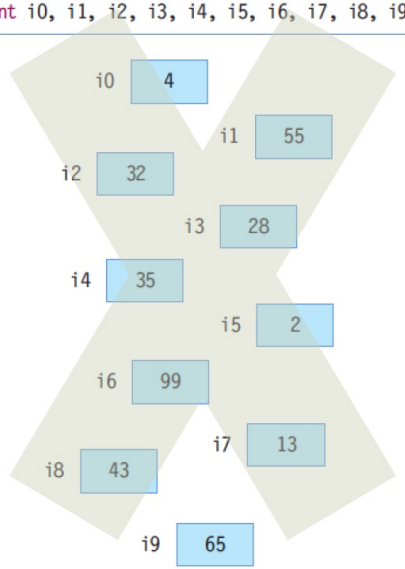
1. Introduction on array

What is array?

- Array are a fundamental data structure
  - allowing to store multiple values of the same type in a single variable
  - in the consecutive memory address
  - provide a means to store fixed size sequential collections of elements

(1) 10개의 정수형 변수를 선언하는 경우

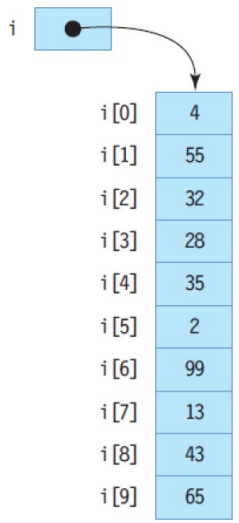
```
int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9;
```



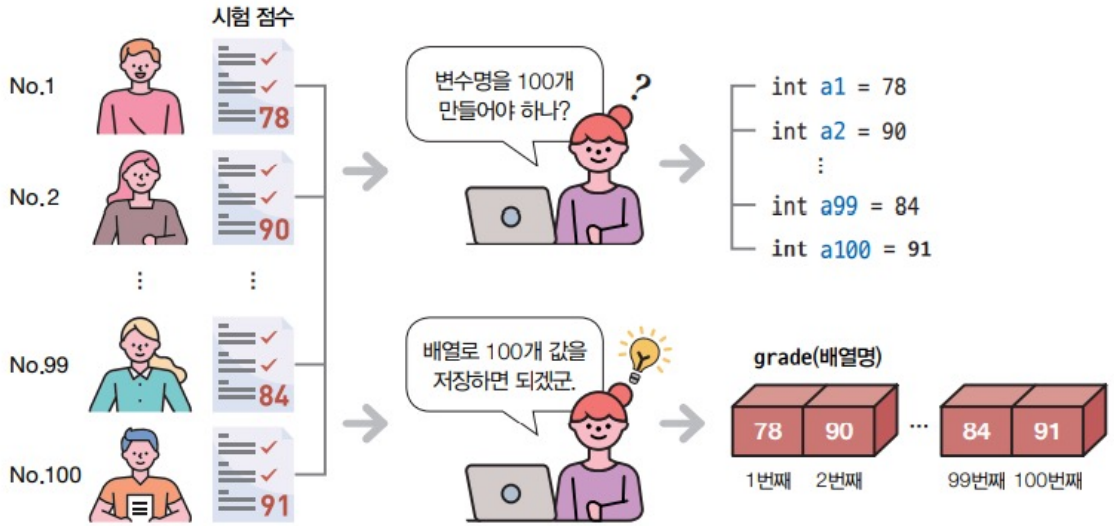

```
sum = i0+i1+i2+i3+i4+i5+i6+i7+i8+i9;
```

(2) 10개의 정수로 구성된 배열을 선언하는 경우

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




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



Declaring and initializing arrays

- Array is essentially a container object holds a fixed number of values of a single type
  - the length of an array is established when the array is created, and then, its size is fixed
  - syntax – array declaration

```
dataType[] arrayName; // or dataType arrayName[];
```

    - '*datatype*': the type of element the array (e.g., int, double, String, etc.)
    - '*arrayName*': the identifier that will be used to refer to the array
  - syntax – memory allocation (instantiation, 인스턴스화)

```
arrayName = new dataType[arraySize];
```

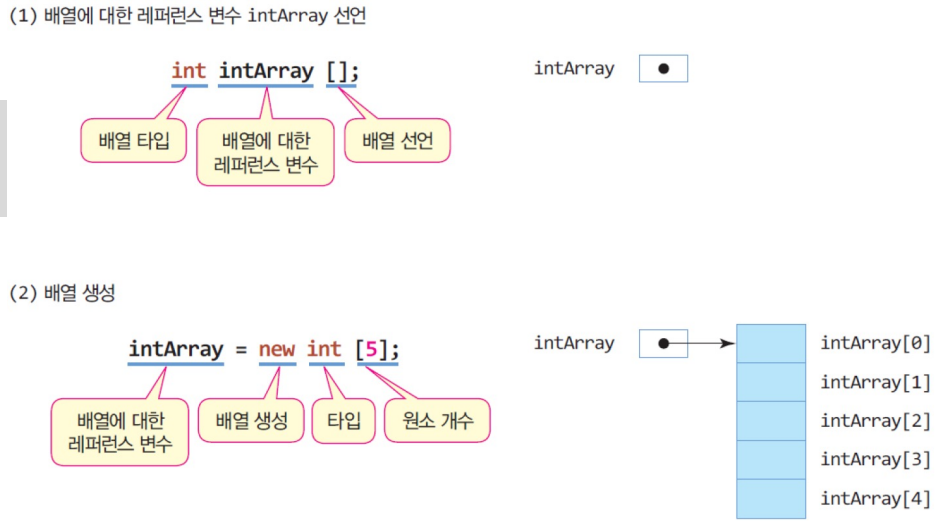
    - '*arraySize*': the length of the array
    - '*new*': keyword to allocate memory for the array
  - syntax – alternatively, declaration and memory allocation in a single line

```
dataType[] arrayName = new dataType[arraySize];
```

Declaring and initializing arrays

- Examples

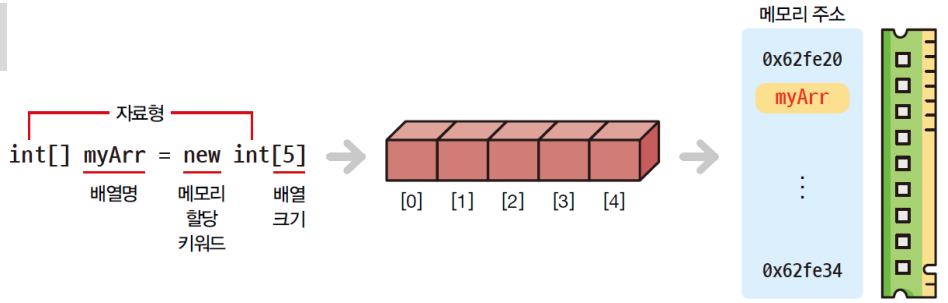
```
int intArray []; // or int[] intArray;
intArray = new int [5];
```



- Alternatively, array declaration and initialization in a single line

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

  - created an integer array with length = 5



Declaring and initializing arrays

- More examples

```
int[] myIntArray = new int[10];
String[] myStringArray = new String[5];
float fArray[] = new float [100];
```

- Wrong examples

```
String[] string = new int [10];
int iArray[] = new float[15];
```

  - the types should be matched

Declaring and initializing arrays

- Then, the values can be assigned

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

intArray[0] = 10; // Assigns 10 to the first element
intArray[1] = 20; // Assigns 20 to the second element
intArray[2] = 30; // Assigns 30 to the third element
intArray[3] = 40; // Assigns 40 to the fourth element
intArray[4] = 50; // Assigns 50 to the fifth element
```

  - in Java, array is zero-based indexing rule
    - array indices start at 0, the last is N-1
      - N is the length of the array
  - alternative; declaring and initializing the array in a single line (a.k.a. initialization)

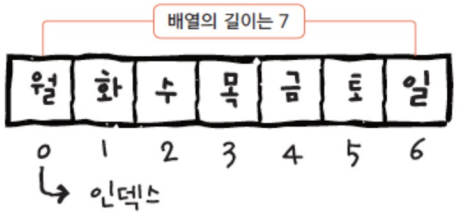
```
int[] myIntArray = {1, 2, 3, 4, 5};
String[] myStringArray = {"Hello", "World", "In", "Java"};
```

    - the most common usage

Declaring and initializing arrays

- More examples for array declaration and initialization

```
int[] odds = {1, 3, 5, 7, 9};
String[] weeks = {"월", "화", "수", "목", "금", "토", "일"};
double[] dArray = {0.1, 0.3, 1.0, 2.3, 4.5, 7.112, -4.32};
float[] fArray = {0.1f, 0.3f, 1.0f, 2.3f, 4.5f, 7.112f, -4.32f};
```



  - should use float literal with explicit 'f' when we use float arrays in Java

- Wrong examples

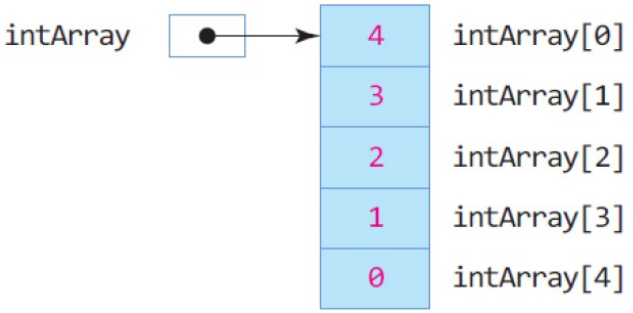
```
int intArray[];
intArray[1] = 8;
```

  - it just declared a reference; not fixed array size
    - in this case, a computer has no idea for array size !

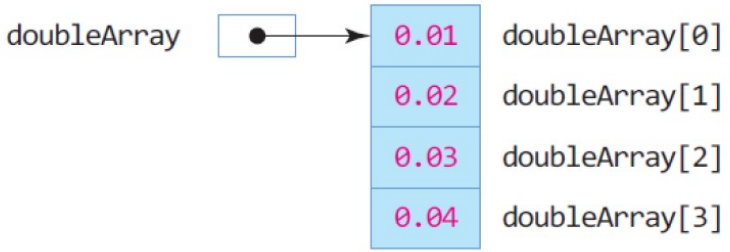
Declaring and initializing arrays

- Detailed procedure of the array declaration and initialization at the same time

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




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



Accessing elements in an array

- Accessing elements in an array involves specifying the index of the element
  - can retrieve or modify the element
  - zero-based indexing rule
- Example
  - accessing single elements

```
int[] numbers = {10, 20, 30, 40, 50};

// Accessing the first element
int firstElement = numbers[0];

// Accessing the third element
int thirdElement = numbers[2];

System.out.println("First Element: " + firstElement + ", Third Element: " + thirdElement);
```

```
First Element: 10, Third Element: 30
```

Accessing elements in an array

- Accessing elements by using loop statement

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

for (int i = 0; i < 5; i++)
  System.out.println(i + "-th element : "+ myArr[i]);
```

```
0-th element : 10
1-th element : 20
2-th element : 30
3-th element : 40
4-th element : 50
```

Accessing elements in an array

- Accessing elements by using loop statement
  - calculating sum and average in array

```
double[] gradeArr = {90, 70.5, 80, 79, 82.5, 50, 70, 90.2, 89.5, 89.7};
double sum = 0.0;

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

double average = sum / gradeArr.length;
System.out.println("Sum: "+ sum);
System.out.format("Average: %.2f", average);
```

```
합계: 791.4000000000001
평균: 79.14
```

Accessing elements in an array

- Wrong example for index usage

```
int[] numbers = {10, 20, 30, 40, 50};
int n = intArrary[-2];
int m = intArray[5];
```

  - index must be a positive number and less than the size of array
  - Error code: 'ArrayIndexOutOfBoundsException'

Usage of accessing elements

- Find the maximum number in array

```
int[] arr = {90, 70, 80, 79, 82, 16, 19, 99, 89, 87};

int max = intArray[0];

for(int i = 1; i < arr.length; i++) {
  if (arr[i] > max) {
    max = intArray[i];
  }
}

System.out.println("The max is " + max);
```

```
The max is 99
```

Usage of accessing elements

- Accessing elements by user input

```
String[] myArr;
myArr = new String[3];

Scanner s = new Scanner(System.in);

System.out.println("Enter the three strings:");

for (int i = 0; i < 3; i++) {
  myArr[i] = s.nextLine();
}

for (int i = 0; i < 3; i++)
  System.out.print(myArr[i]+" ");
```

```
Enter the three strings:
apple
banana
orange
apple banana orange
```

Usage of accessing elements

- Find the maximum number in array by user input

```
System.out.print("Enter the number of elements in the array: ");
int n = scanner.nextInt();
int[] arr = new int[n];

System.out.println("Enter the array elements:");
for (int i = 0; i < n; i++) {
  arr[i] = scanner.nextInt();
}

int max = arr[0];

for (int i = 1; i < arr.length; i++) {
  if (arr[i] > max) {
    max = arr[i];
  }
}

System.out.println("The max is " + max);
```

```
Enter the number of elements in the array: 5 95 8 -71 49 53 -4 91
Enter the array elements:
The max is 95
```

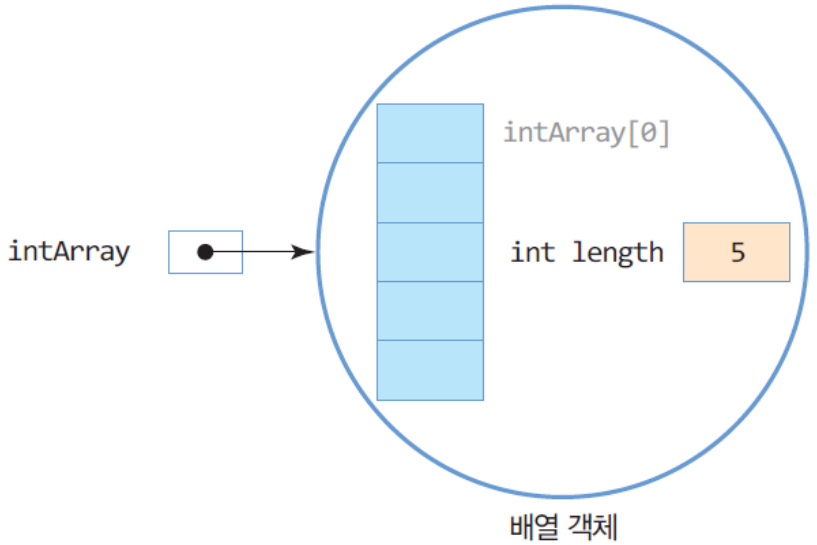
Array properties and methods

- In Java, array have properties and methods that can be used to manipulate and interact with the array
  - the primary property
    - 'length': getting the size of an array

```
int[] intArray = {10, 20, 30, 40, 50};
int size = intArray.length;
System.out.println("Array size: " + size);
```

```
Array size: 5
```

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

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



Array properties and methods

- Examples using 'length' property
  - print all elements in array

```
int[] iArray = {5, 10, 15, 20, 25};
for(int i = 0; i < numbers.length; i++)
  System.out.println(i + "-th array is " + iArray[i]);
```

```
0-th array is 5
1-th array is 10
2-th array is 15
3-th array is 20
4-th array is 25
```

  - calculating the average value of an array

```
int[] numbers = {5, 10, 15, 20, 25};
int sum = 0;

for(int i = 0; i < numbers.length; i++) {
  sum += numbers[i];
}
double average = (double) sum / numbers.length;
System.out.println("The average is: " + average);
```

Examples and practices for array

- 주어진 array에서 값이 60인 인덱스를 찾아서 출력하는 프로그램을 작성해보세요.
  - File path and name: Chap05Example/ArrayPractice01.java
  - input and output examples
    - given array:

```
int[] arr = {10, 20, 30, 50, 3, 60, -3};
```

    - output:

```
5
```

Examples and practices for array

- 사용자로부터 크기가 7인 배열을 한 개와 인덱스를 하나 입력받고, 해당 인덱스의 값을 1000으로 변경하는 프로그램을 작성해보세요,
  - File path and name: Chap05Example/ArrayPractice02.java
  - inputs and outputs

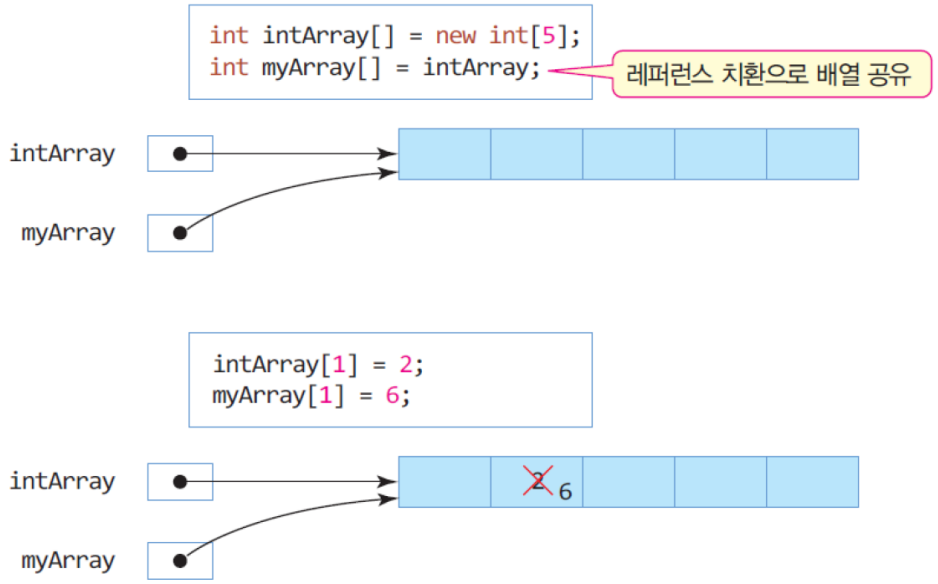
```
Origin array: 10 20 30 50 3 60 -3
Enter the index you wish to change: 5
New array: 10 20 30 50 3 1000 -3
```

Note: Array reference assignment and sharing

- Concept of the reference assignment and sharing
  - one array can be manipulated and accessed through different reference variables
    - when you assign one array reference variable to another, both variables refer to the same array in the memory

```
int[] intAray = {1, 2, 3, 4, 5};
int[] myArray = intAray;

intAray[2] = 2;
intAray[2] = 6;

System.out.println("intArray: " + intAray[2]);
System.out.println("myArray: " + myArray[2]);
```

```
intArray: 6
myArray: 6
```



Multidimensional arrays

- Multidimensional arrays: arrays of arrays
  - each element of the array itself can be an array
  - allows for the creation of complex data structures like matrices or tables
- Concept
  - single-dimensional array
    - a linear array containing elements of the same type, accessed by a single index
  - multidimensional array
    - array of array, where each element is accessed by multiple indices
    - the most common form: two-dimensional array
      - Java supports arrays with more dimensions

Multidimensional arrays

- Declaration and value assignment (initialization) in multidimensional array
  - declaration

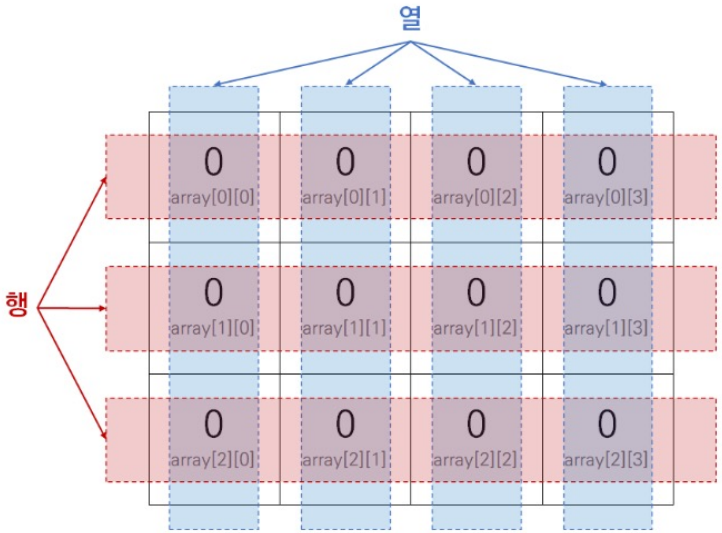
```
int[][] array; // Declares a 2-dimensional array of integers
int[][][] array3D; // Declares a 3-dimensional array of integers
```

    - for integer-typed array
  - memory allocation (instantiation)

```
array = new int[3][4]; // 3 rows and 4 columns
array3D = new int[3][10][20]; // 3 depths, 10 rows, and 20 columns
```

    - 기본값은 0
      - array ➔ { {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0} }
  - declaration and memory allocation in a single line

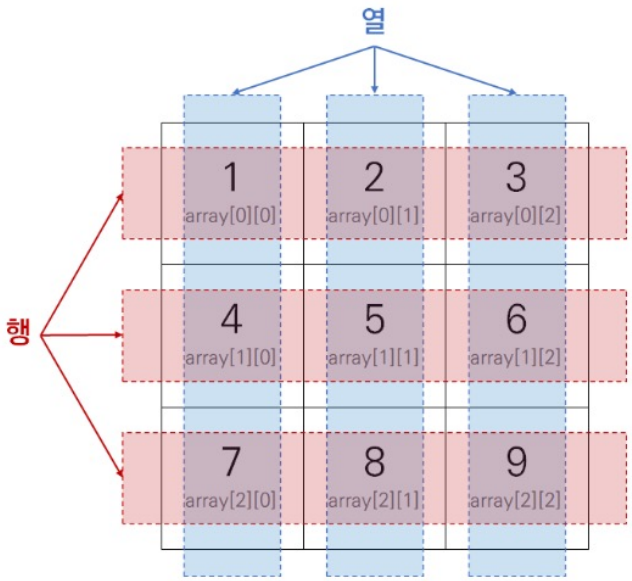
```
int[][] array = new int[3][4];
```



Multidimensional arrays

- Declaration and value assignment (initialization) in multidimensional array
  - the most common usage for 2-d array in a single line

```
int[][] array = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }; // 3x3 matrix
```



- Basic examples
  - declaring and instantiating

```
double[][] matrix = new double[3][3];
```

  - initializing at declaration

```
String[][] names = {
  {"John", "Doe"},
  {"Jane", "Doe"}
};
```

Multidimensional arrays

- More examples for 2-d array declaration and initialization

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

```
char charArray[][] = {{'a', 'b', 'c'}, {'d', 'e', 'f'}};
```

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

Quiz

- Which of the following is not correct as a way to declare an array?

```
int[][] intArray; // (a)

int intArray[][]; // (b)

int[] intArray;   // (c)

int intArray[];   // (d)
```

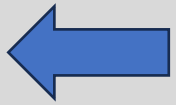
Multidimensional arrays

- Accessing the elements
  - use ‘row index’ and ‘column index’

```
int[][] matrix = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} };
System.out.println(matrix[0][2]);
```

  - nested for loop statement is nice tool

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
for (int i=0; i< matrix.length; i++) {
  for (int j=0; j< matrix[i].length; j++) {
    System.out.println(matrix[i][j]);
  }
}
```



Pay attention to using the ‘length’ property

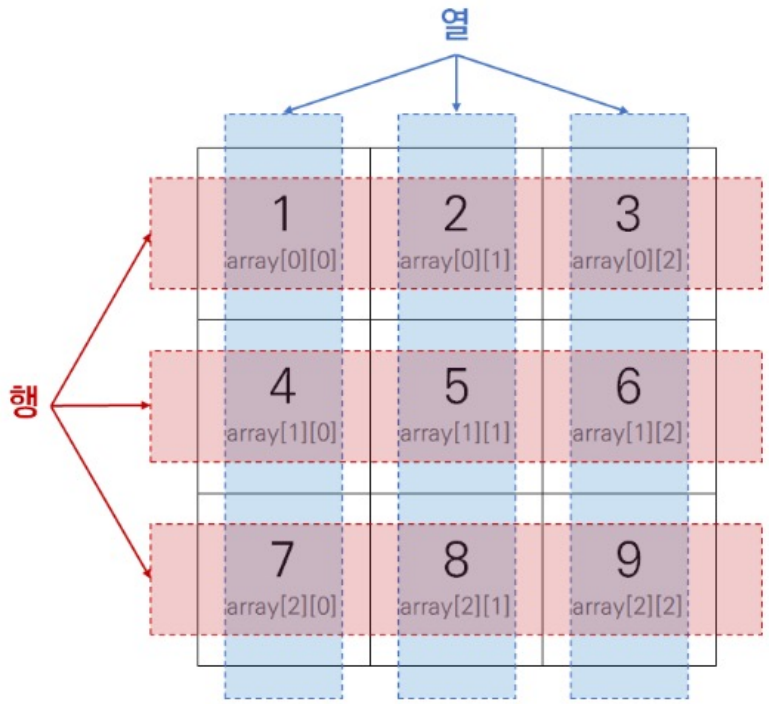
```
1
2
3
4
5
6
7
8
9
```

Multidimensional arrays

- Accessing the elements
  - tips for nice representation as matrix format

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
for (int i=0; i< matrix.length; i++) {
  for (int j=0; j< matrix[i].length; j++) {
    System.out.print(matrix[i][j]+ " ");
  }
  System.out.println();
}
```

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



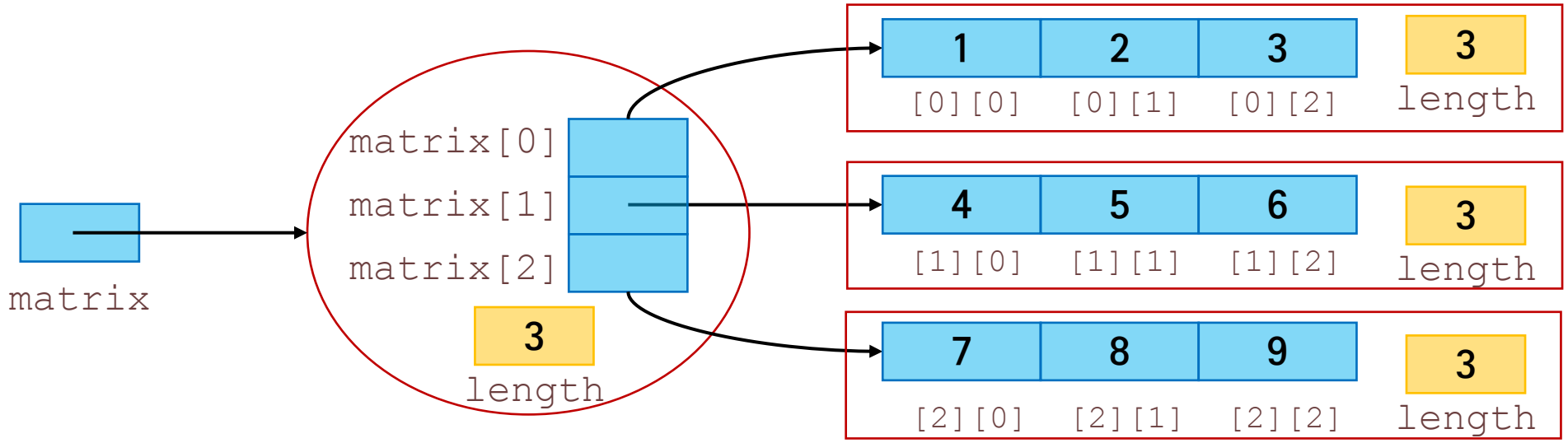
  - what is the value of `matrix[0]`? expect
    - {1, 2, 3}? or other result

Multidimensional arrays

- Practical results of the single index in 2-d array

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }; // 3x3 matrix
for (int i=0; i< matrix.length; i++) {
    System.out.print(matrix[i]+ " ");
  }
  System.out.println();
}
```

```
[I@3d012ddd
[I@626b2d4a
[I@5e91993f
```



Usage of multidimensional array

- 2차원 배열에 학년별로 1, 2학기 성적을 저장하고, 4년 전체 평점 평균을 출력하는 프로그램

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

  double sum = 0;

  for(int year=0; year<score.length; year++) {
    for(int term=0; term<score[year].length; term++) {
      sum += score[year][term];
    }
  }

  System.out.println("4년 전체 평균은 " + sum/(score.length * score[0].length));
}
```

Usage of multidimensional array

- 두 행렬의 덧셈을 수행하는 프로그램

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

// Resultant matrix, where the result will be stored
int[][] resultMatrix = new int[3][3];
for(int i=0; i<firstMatrix.length; i++) {
  for(int j=0; j<firstMatrix[i].length; j++) {
    resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
  }
}

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

```
10 10 10
10 10 10
10 10 10
```

Usage of multidimensional array

- 학생들의 성적을 입력받고 평균을 계산하는 프로그램

```
double[][] marks = new double[2][3];
Scanner s = new Scanner(System.in);

for (int i = 0; i < 2; i++) {
  System.out.println("Student ID" + (i + 1));

  System.out.print("Korean score : ");
  marks[i][0] = s.nextDouble();

  System.out.print("Math score : ");
  marks[i][1] = s.nextDouble();

  marks[i][2] = (marks[i][0] + marks[i][1])/2;
}

for (int i = 0; i < 2; i++) {
  System.out.println("Student ID" + (i + 1));
  System.out.print("Korean" + ":" + marks[i][0] + " ");
  System.out.print("Math" + ":" + marks[i][1] + " ");
  System.out.println("Avg." + ":" + marks[i][2] + " ");
}
```

```
Student ID1
Korean score : 95
Math score : 23
Student ID2
Korean score : 74
Math score : 100
Student ID1
Korean:95.0 Math:23.0 Avg.:59.0
Student ID2
Korean:74.0 Math:100.0 Avg.:87.0
```

Note: for-each statement

- for-each statement
  - the enhanced ‘for’ loop to make it easier to iterate over arrays and collections
  - less error-prone
    - not need to manage start and end conditions as in traditional ‘for’ loop
  - syntax

```
for (Type element : arr) {
  // Use element here
}
```

Note: for-each statement

- Example of for-each statement
  - iterating over an array of integers

```
int[] numbers = {1, 2, 3, 4, 5};
for (int number : numbers) {
  System.out.println(number);
}
```

```
1
2
3
4
5
```

Note: for-each statement

- Example of for-each statement
  - multiplying elements in an array

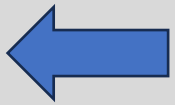
```
int[] factors = {2, 4, 6, 8, 10};
for (int factor : factors) {
  System.out.println(factor * 2);
}
```

```
4
8
12
16
20
```

Note: for-each statement

- Example of for-each statement
  - iterating over each element in a 2D array

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

for (int[] row : grid) {
  for (int element : row) {
    System.out.println(element);
  }
}
```



Pay attention to using the outer loop

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

Note: for-each statement

- Example of for-each statement
  - printing a 2D array in matrix form

```
char[][] board = {
  {'X', 'O', 'X'},
  {'O', 'X', 'O'},
  {'X', 'O', 'X'}
};
for (char[] row : board) {
  for (char cell : row) {
    System.out.print(cell + " ");
  }
  System.out.println(); // new line after each row
}
```

```
X O X
O X O
X O X
```

Note: for-each statement

- Example of for-each statement
  - counting occurrences of a value in a 2D array

```
int[][] numbers = {
  {1, 1, 2},
  {3, 1, 2},
  {4, 1, 6}
};
int count = 0;
int toFind = 1;
for (int[] row : numbers) {
  for (int num : row) {
    if (num == toFind) {
      count++;
    }
  }
}
System.out.println("Number of 1s: " + count);
```

```
Number of 1s: 4
```

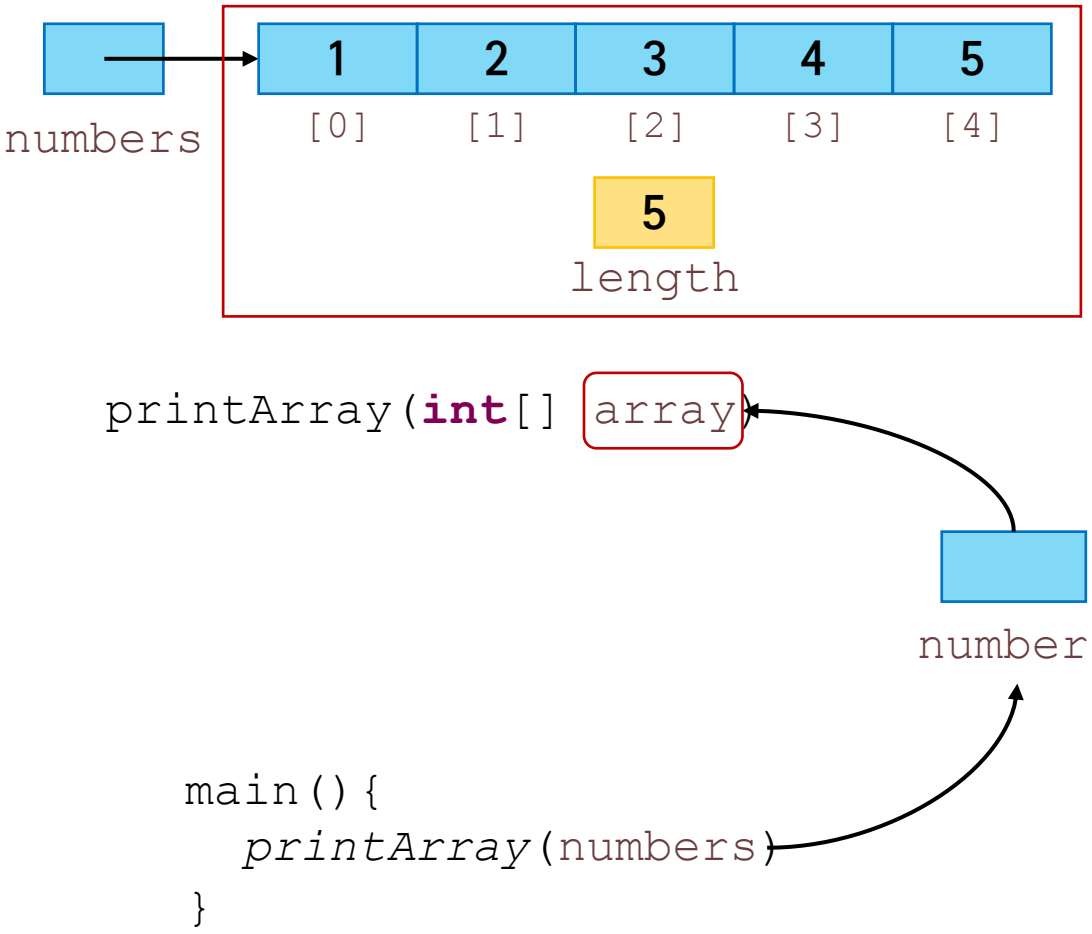
Arrays as parameters in the method

- Arrays can be passed to methods as "reference"
  - not the actual array itself

```
public static void printArray(int[] array) {
  for (int element : array) {
    System.out.print(element + " ");
  }
  System.out.println();
}
```

```
public static void main(String[] args) {
  int[] numbers = {1, 2, 3, 4, 5};
  System.out.println("Array :");
  printArray(numbers);
}
```

```
Array :
1 2 3 4 5
```



array의 reference가 method에 전달 ➔ 함수 내에서 값을 조작하면 실제 그 값이 바뀜

➔ call by reference (레퍼런스에 의한 호출)

Recall: Call by value

- main 메소드 내 x 값은 외부 함수에서 바뀌지 않음

  ➔ call by value (값에 의한 호출)

```
public static void printX(int x) {
  System.out.println("X in printX method = " + x);
  x++;
  System.out.println("X in printX method = " + x);
}

public static void main(String[] args) {
  int x = 10;
  System.out.println("X in main method = " + x);
  printX(x);
  x = 50;
  System.out.println("X in main method = " + x);
}
```

```
X in main method = 10
X in printX method = 10
X in printX method = 11
X in main method = 50
```

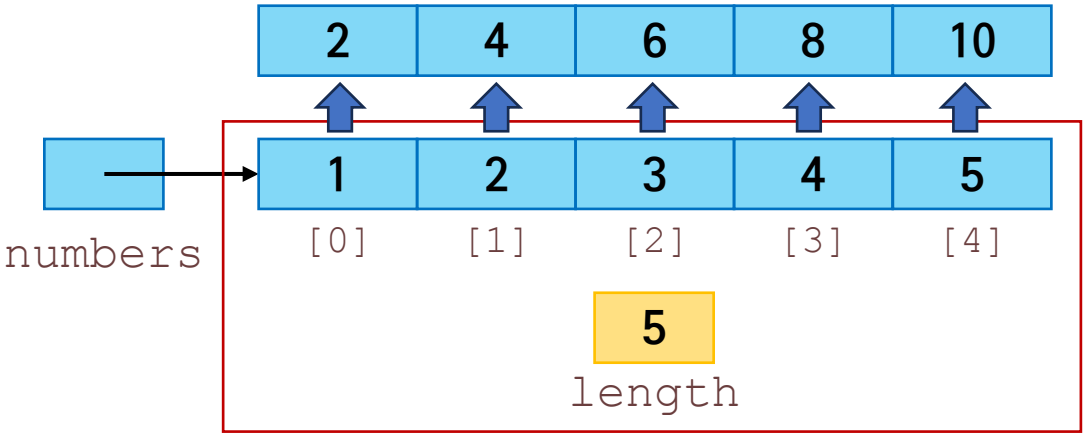
Arrays as parameters in the method

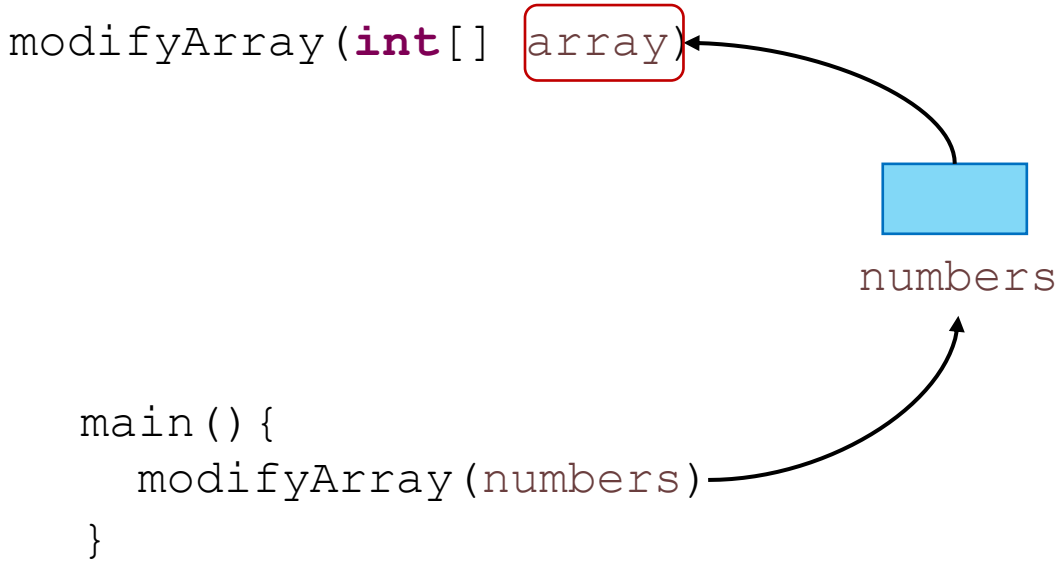
- Arrays can be passed to methods as "reference"
  - not the actual array itself

```
public static void modifyArray(int[] arr) {
  for (int i = 0; i < arr.length; i++) {
    arr[i] *= 2;
  }
}
```

```
public static void main(String[] args) {
  int[] numbers = {1, 2, 3, 4, 5};
  System.out.println("Before modification:");
  printArray(numbers);
  modifyArray(numbers);
  System.out.println("After modification:");
  printArray(numbers);
}
```

```
Before modification:
1 2 3 4 5
After modification:
2 4 6 8 10
```





Returning array in the method

- Array is also returned in the method as reference

```
public static int[] generateArray(int size) {
  int[] array = new int[size];
  for (int i = 0; i < array.length; i++) {
    array[i] = i * 2;
  }
  return array;
}
```

```
public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  System.out.print("Enter the size of array: ");
  int size = scanner.nextInt();
  int[] generatedArray = generateArray(size);
  System.out.println("Generated Array:");
  for (int value : generatedArray) {
    System.out.print(value + " ");
  }
}
```

```
Enter the size of array: 10
Generated Array:
0 2 4 6 8 10 12 14 16 18
```

2. Sort

What is sort?

- Sorting (정렬)
  - 데이터의 집합을 어떤 기준의 대소관계를 따져 일정한 순서로 줄지어 세우는 것
    - 데이터의 집합: 1, 5, 6, 7, 2, 4, 9, 8, 3
    - 일정한 순서: 오름차순/내림차순 등
      - 오름차순: 1, 2, 3, 4, 5, 6, 7, 8, 9
      - 내림차순: 9, 8, 7, 6, 5, 4, 3, 2, 1

- Sorting algorithms
  - Bubble sort, Selection sort, Insertion sort, Merge sort, Quick sort, Heap sort, Radix sort, etc.
  - 상황별로 유리한 정렬 알고리즘이 존재함

Swap

- 두 변수 (혹은 element)의 값을 서로 바꾸는 것

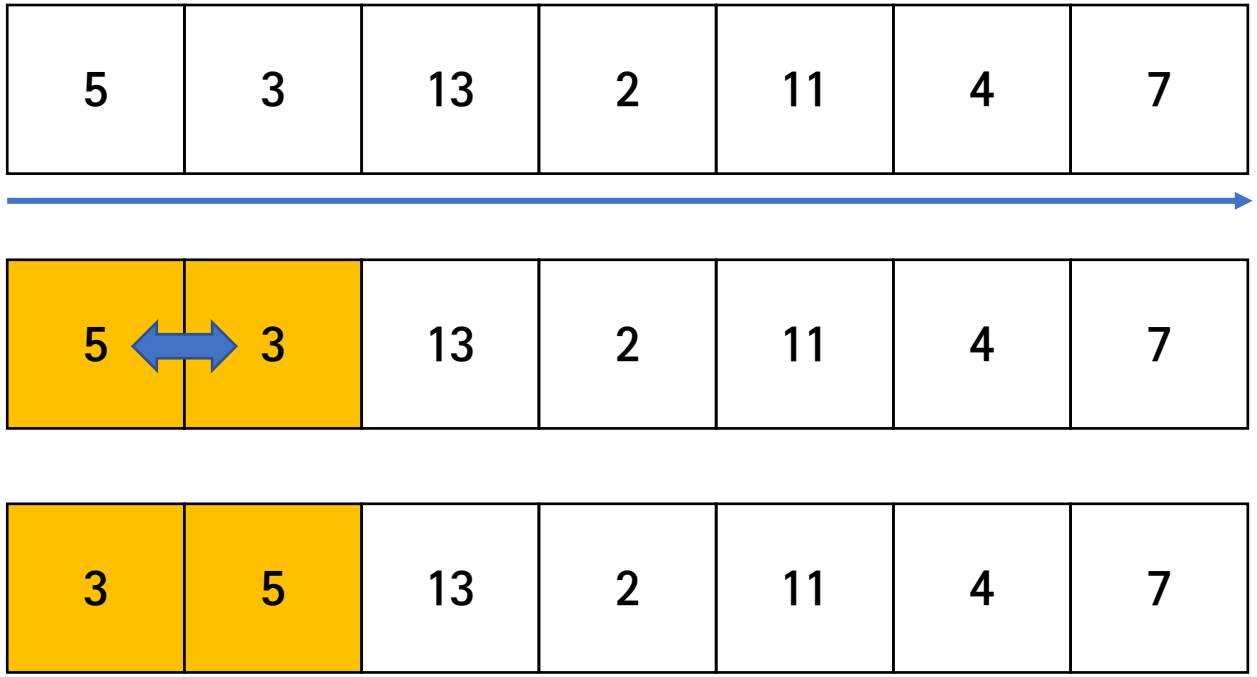
```
int a = 10;
int b = 5;
int temp;

temp = a;
a = b;
b = temp;
```

Bubble sort

- Concept
  - 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘
  - ➔ 인접한 원소를 비교하여 크기가 순서대로 되어있지 않으면 서로 위치를 교환 (swap)

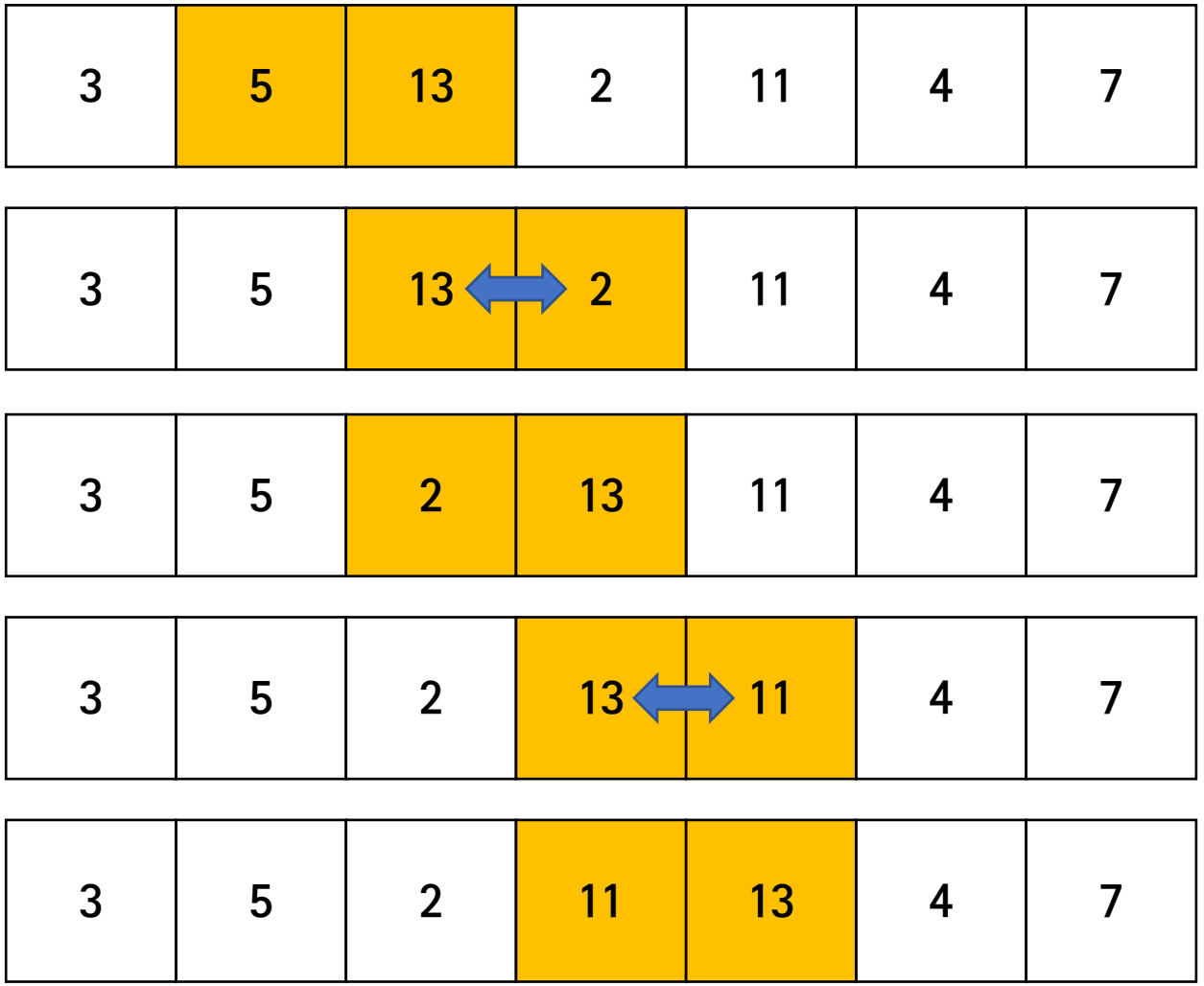
Bubble sort

- Algorithm (오름차순 정렬 기준)
  - Input array: `int[] array = {5, 3, 13, 2, 11, 4, 7};`



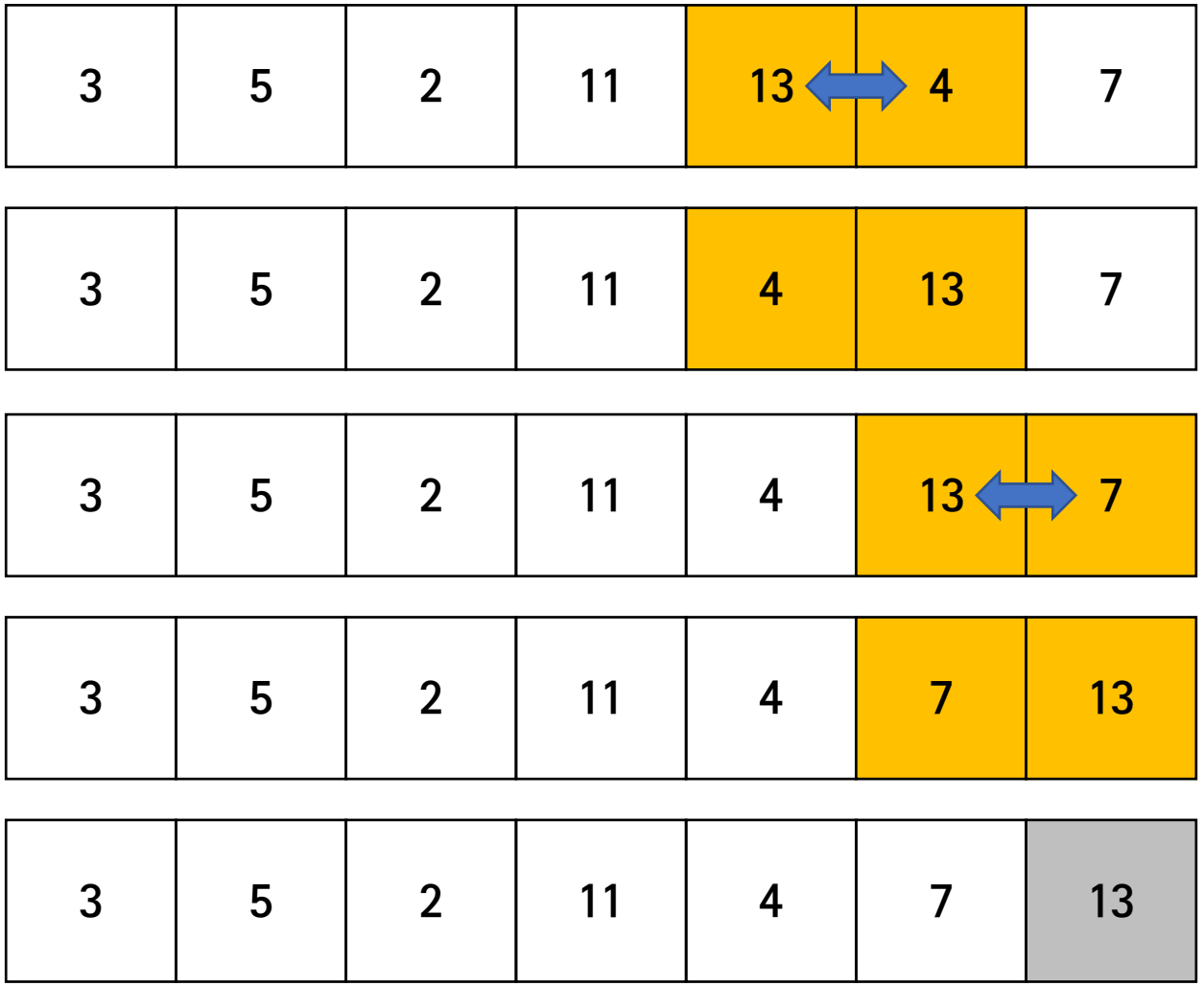
Bubble sort

- Algorithm (오름차순 정렬 기준)



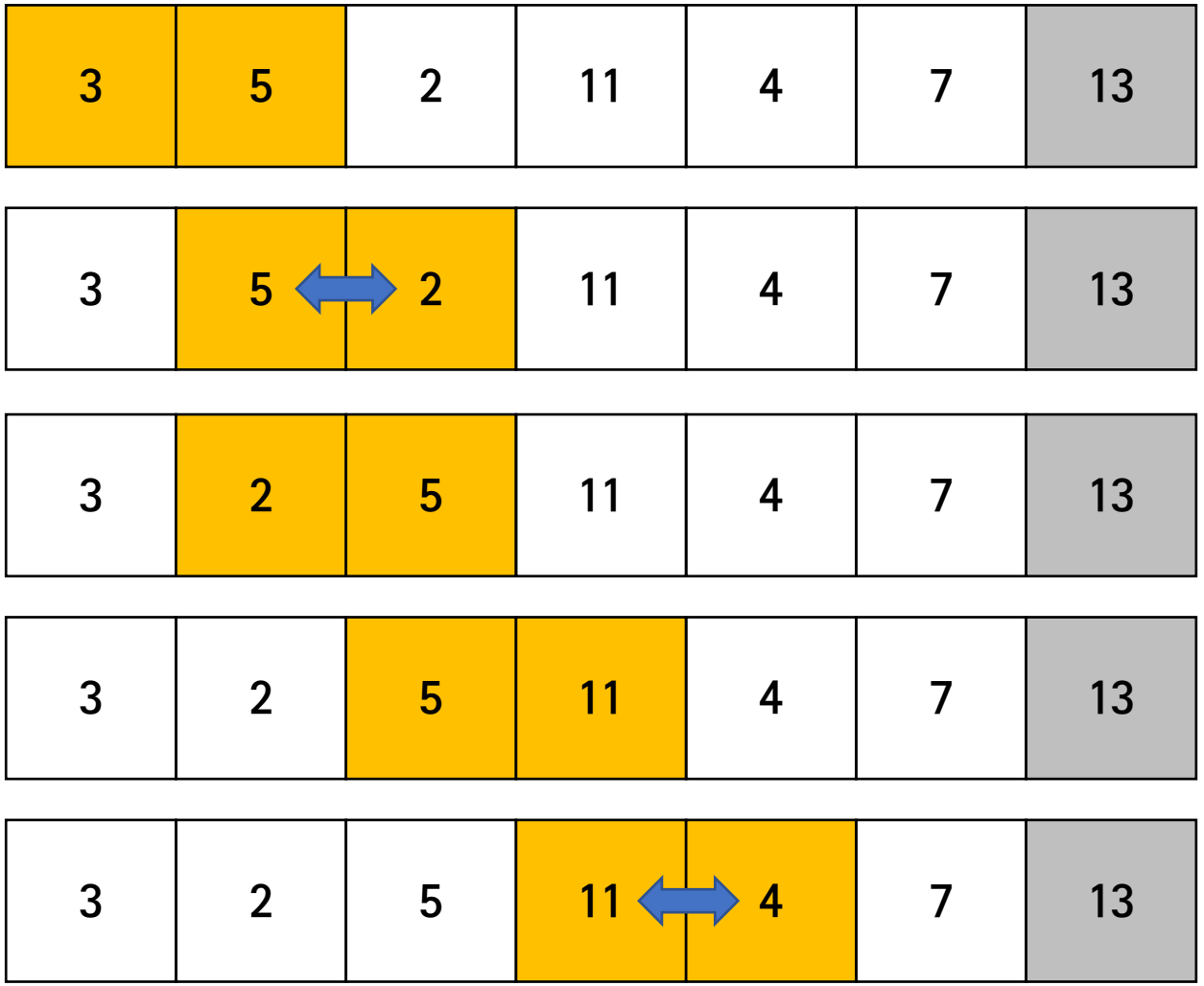
Bubble sort

- Algorithm (오름차순 정렬 기준)



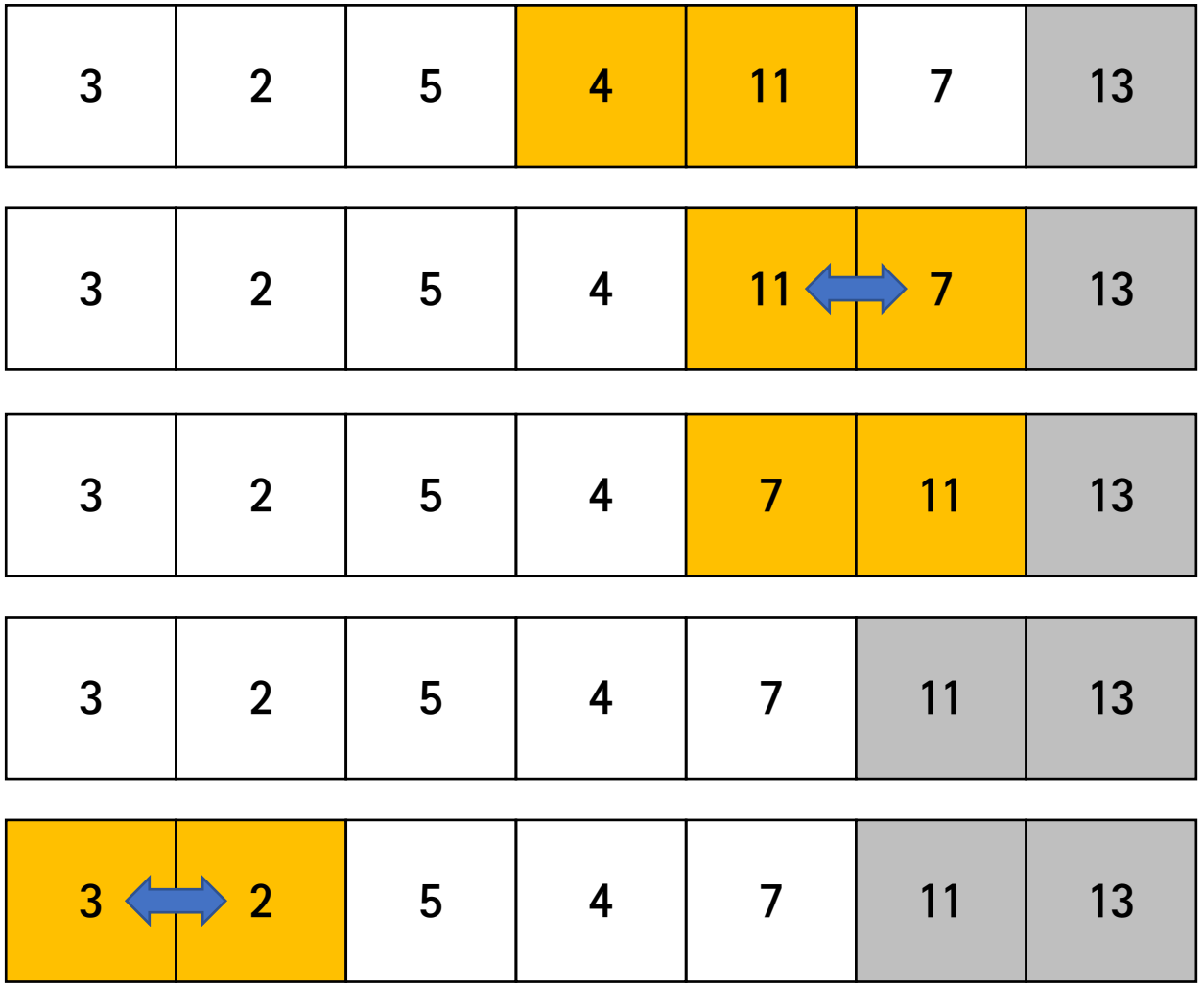
Bubble sort

- Algorithm (오름차순 정렬 기준)



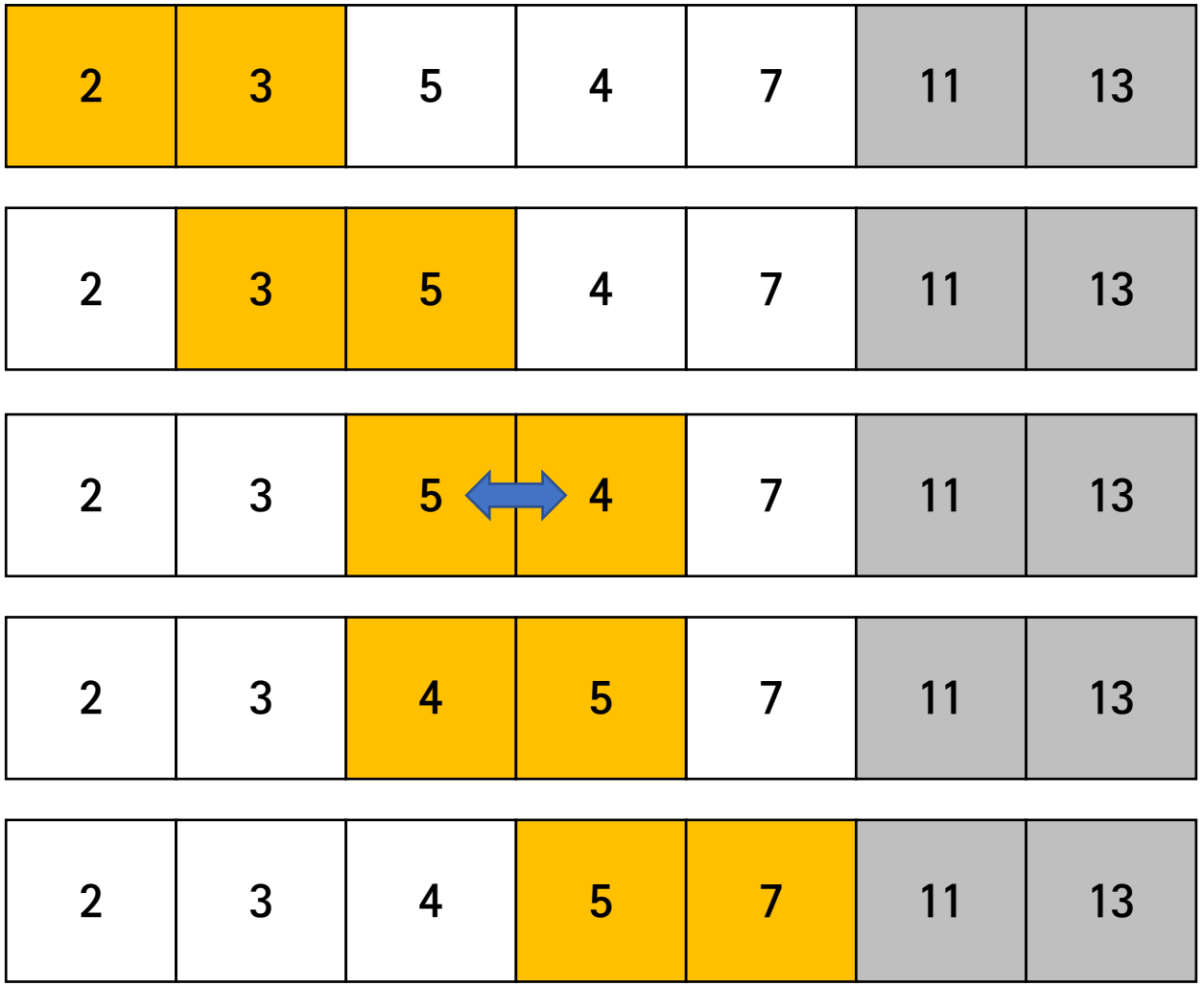
Bubble sort

- Algorithm (오름차순 정렬 기준)



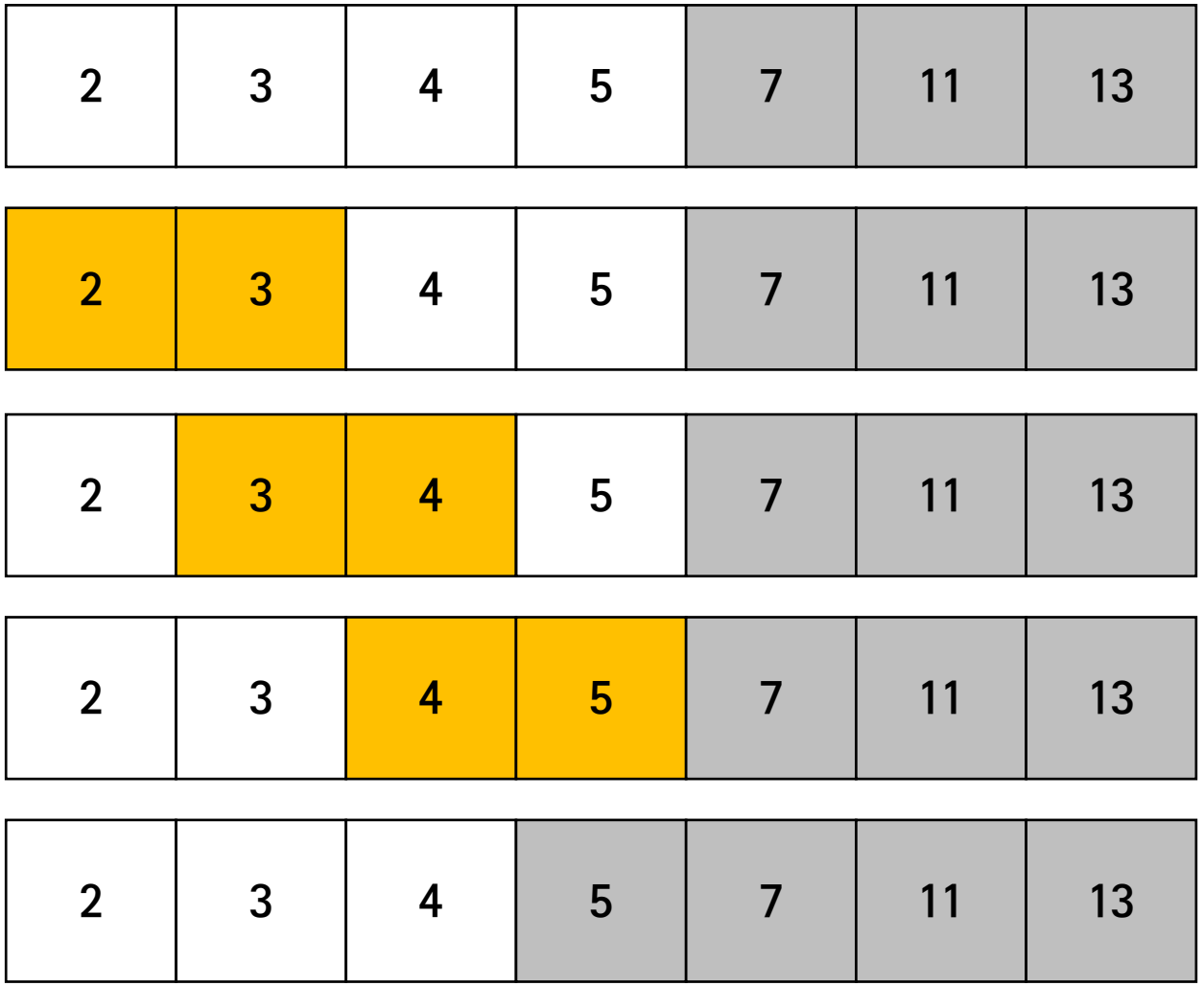
Bubble sort

- Algorithm (오름차순 정렬 기준)



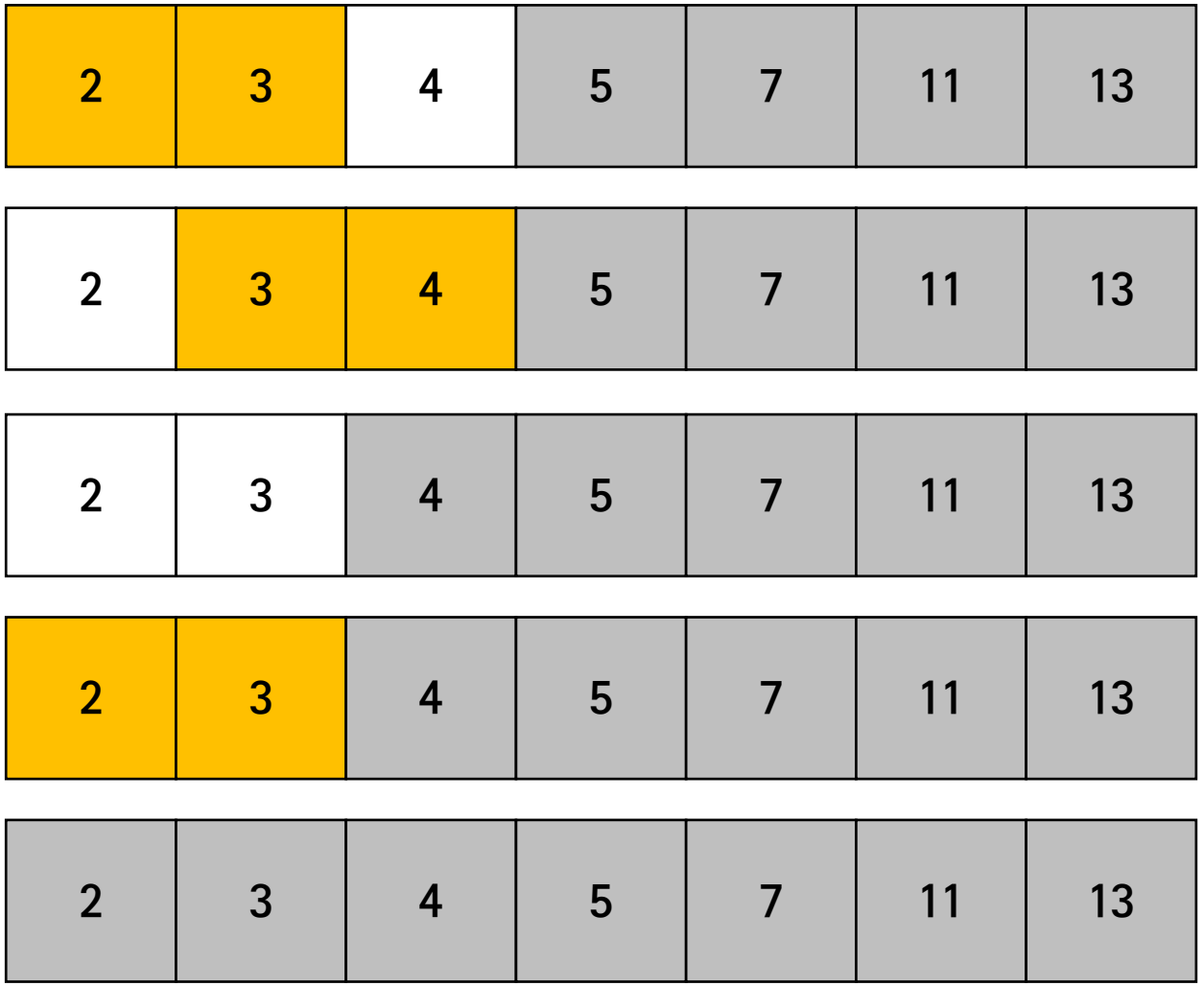
Bubble sort

- Algorithm (오름차순 정렬 기준)



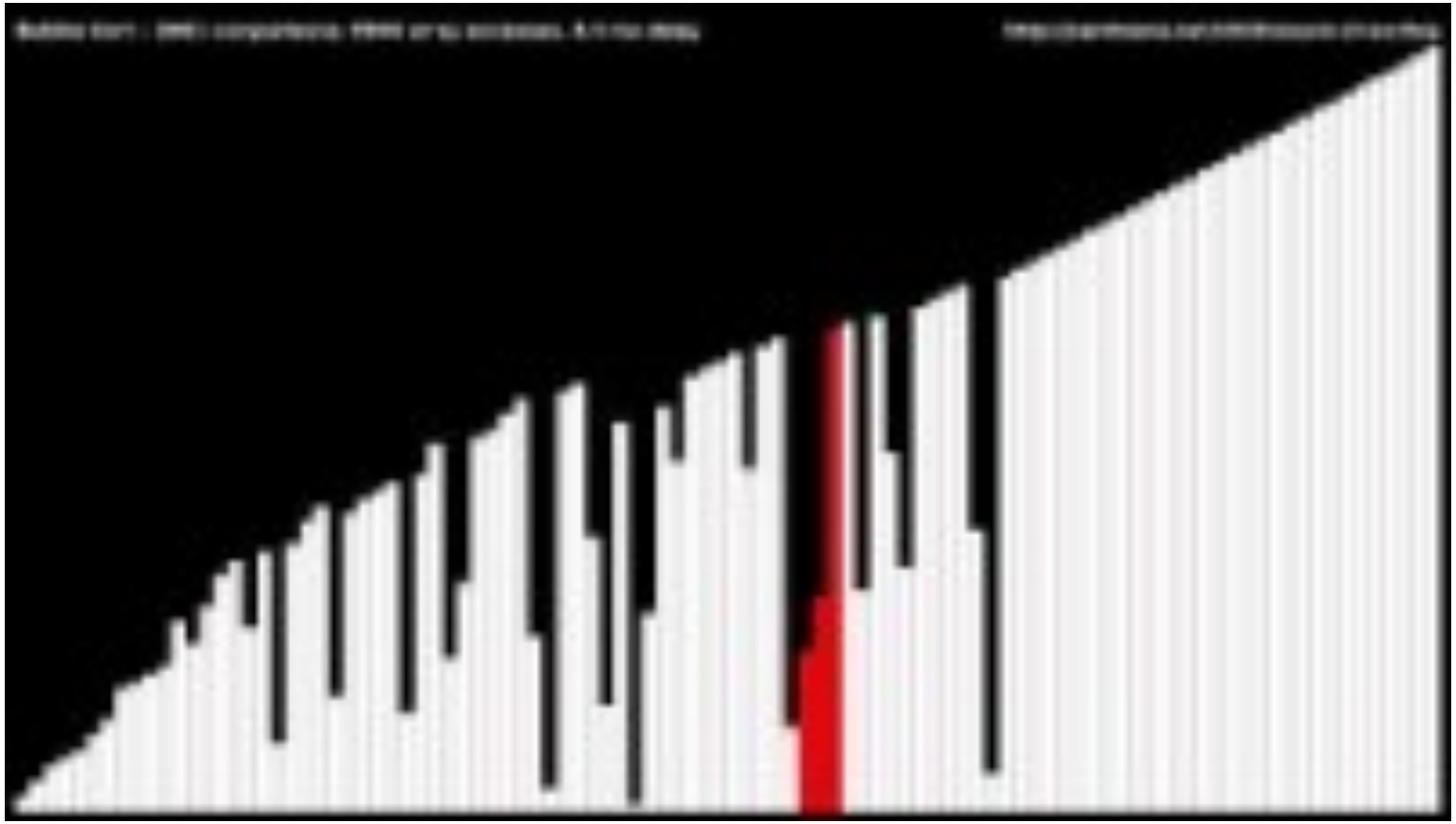
Bubble sort

- Algorithm (오름차순 정렬 기준)



Bubble sort

- Simulation



Bubble sort

- Implementation

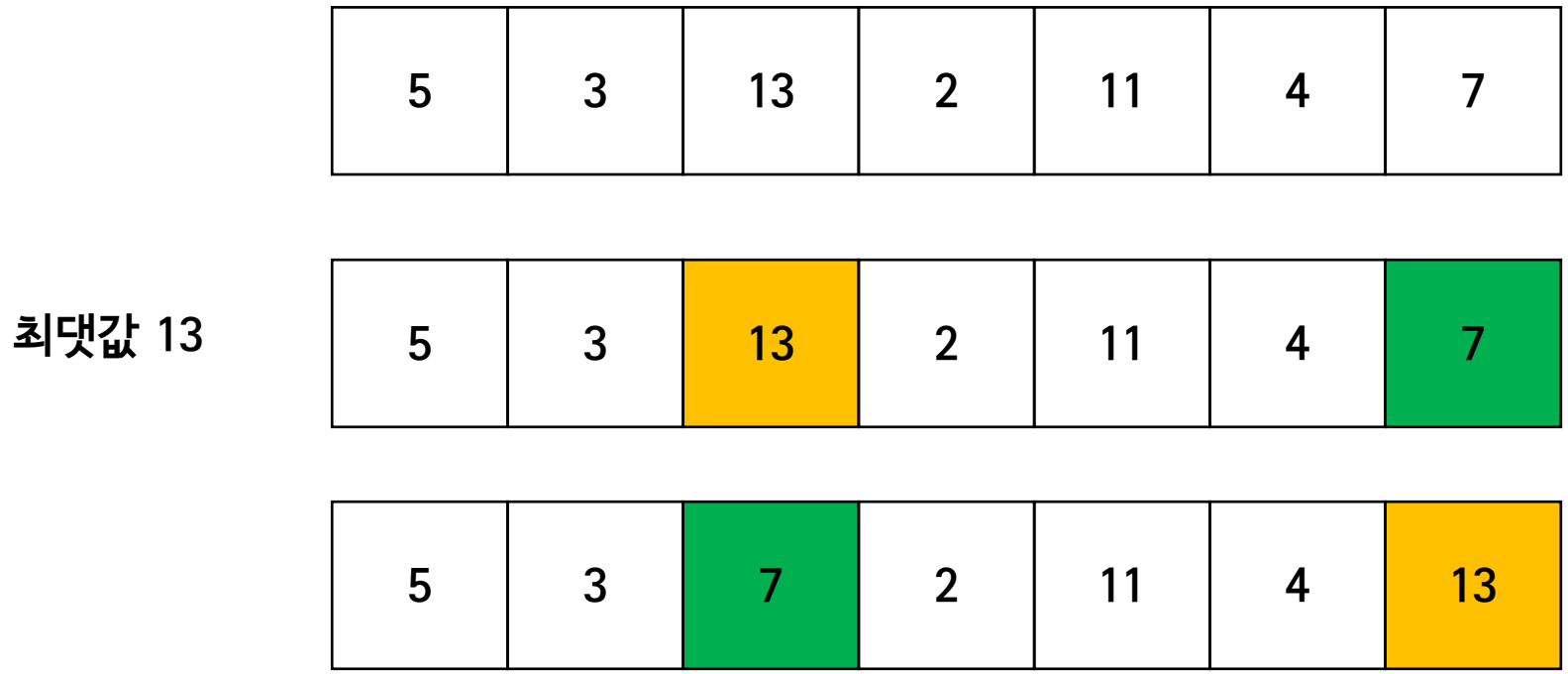
```java
// BubbleSort
public static void main(String[] args) {
  int[] arr = {8, 54, 99, 3, 2, 1, 0};
  final int length = arr.length;

  for (int i = 0; i < length - 1; i++) {
    for (int j = 0; j < length - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}
```

Selection sort

- Concept
  - 현재 위치에 들어갈 데이터를 찾아서 선택하는 알고리즘
    - 1. 주어진 array에서 최솟값 (or 최댓값)을 찾는다.
    - 2. 최솟값을 맨 앞자리 (or 맨 뒷자리)와 교환한다.
    - 3. 맨 앞자리 (or 맨 뒷자리) 원소를 뺀 나머지 1, 2 과정을 정렬이 끝날 때까지 반복한다.

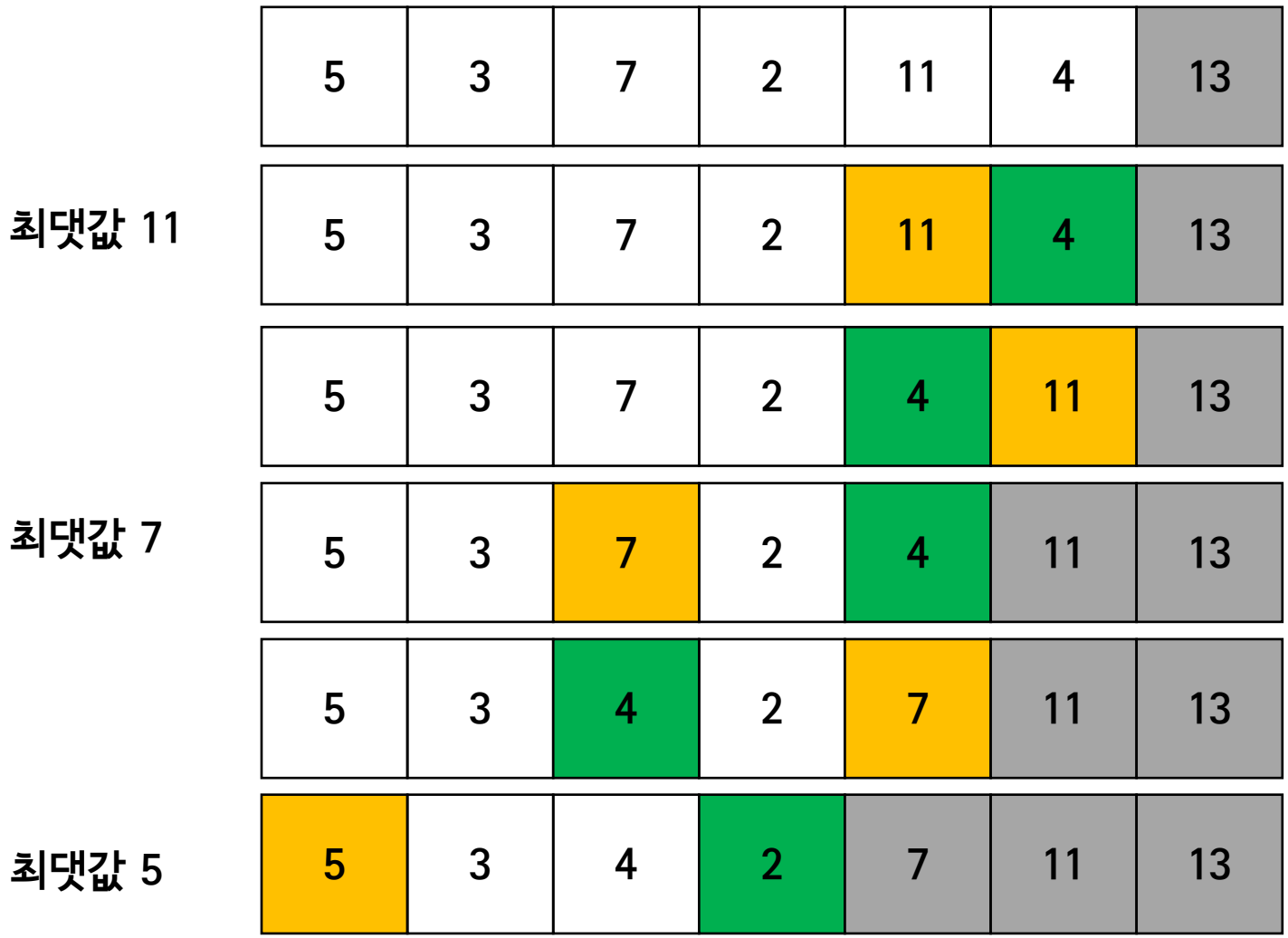
Selection sort

- Algorithm (오름차순 정렬 기준)
  - Input array: `int[] array = {5, 3, 13, 2, 11, 4, 7};`



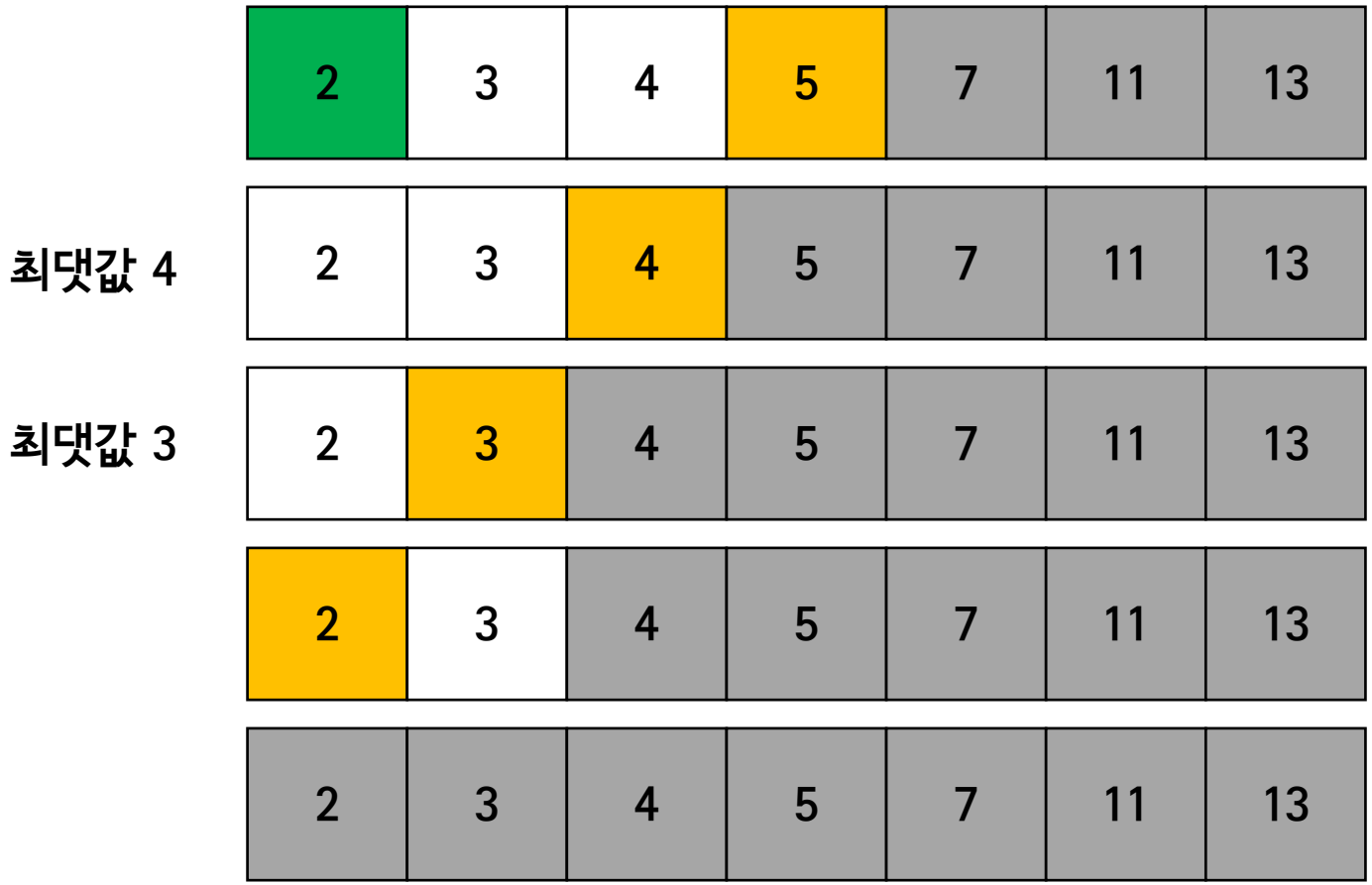
Selection sort

- Algorithm (오름차순 정렬 기준)



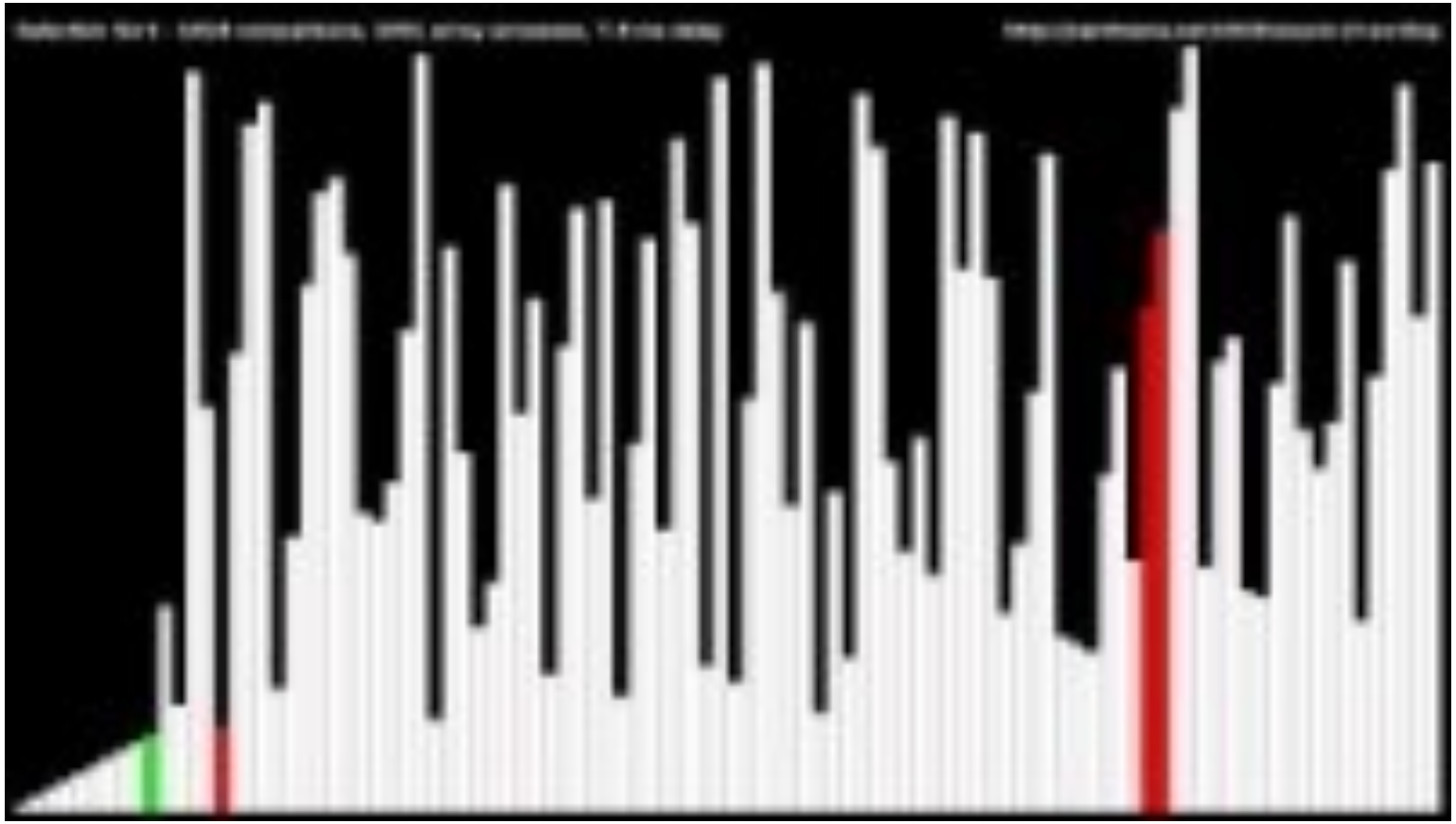
Selection sort

- Algorithm (오름차순 정렬 기준)



Selection sort

- Simulation



Selection sort

- Implementation

```
// Selection sort
int[] arr = {8, 54, 99, 3, 2, 1, 0};
final int length = arr.length;

for (int i = 0; i < n - 1; i++) {
  // Find the index of the minimum element in the unsorted part of the array
  int minIdx = i;
  for (int j = i + 1; j < n; j++) {
    if (arr[j] < arr[minIdx]) {
      minIdx = j;
    }
  }

  int temp = arr[i];
  arr[i] = arr[minIdx];
  arr[minIdx] = temp;
}
```

Summary

- Bubble sort
  - 서로 이웃한 원소의 “자리 바꾸기”

- Selection sort
  - 비교 대상 (array) 중 “최솟값 (또는 최댓값) 찾아서 바꾸기”

Usage of sorting algorithm

- Grade sorting program using bubble sort algorithm
  - ascending order

```
double[] grades = {92.5, 88.3, 99.0, 78.2, 85.6, 97.1, 74.5};

for (int i = 0; i < grades.length - 1; i++) {
  for (int j = 0; j < grades.length - i - 1; j++) {
    if (grades[j] > grades[j + 1]) {
      double temp = grades[j];
      grades[j] = grades[j + 1];
      grades[j + 1] = temp;
    }
  }
}

System.out.println("Sorted grades:");
for (double grade : grades) {
  System.out.println(grade);
}
```

```
Sorted grades:
74.5
78.2
85.6
88.3
92.5
97.1
99.0
```

Usage of sorting algorithm

- Grade sorting program using bubble sort algorithm
  - ascending order

```
public static void swap(double[] num, int index1, int index2) {
  double temp = num[index1];
  num[index1] = num[index2];
  num[index2] = temp;
}
```

```
for (int i = 0; i < grades.length - 1; i++) {
  for (int j = 0; j < grades.length - i - 1; j++) {
    if (grades[j] > grades[j + 1]) {
      swap(grades, j, j+1);
    }
  }
}

System.out.println("Sorted grades:");
for (double grade : grades) {
  System.out.println(grade);
}
```

```
Sorted grades:
74.5
78.2
85.6
88.3
92.5
97.1
99.0
```

Usage of sorting algorithm

- Grade sorting program using bubble sort algorithm
  - focusing on names and scores at the same time with the descending order

```
String[] studentNames = {"John", "Jane", "Alan", "Ada", "Grace"};
double[] studentScores = {82.5, 91.0, 99.5, 88.5, 95.0};
for (int i = 0; i < studentScores.length - 1; i++) {
  for (int j = 0; j < studentScores.length - i - 1; j++) {
    if (studentScores[j] < studentScores[j + 1]) {
      double tempScore = studentScores[j];
      studentScores[j] = studentScores[j + 1];
      studentScores[j + 1] = tempScore;

      String tempString = studentNames[j];
      studentNames[j] = studentNames[j + 1];
      studentNames[j + 1] = tempString;
    }
  }
}
System.out.println("Sorted student scores:");
for (int i=0; i < studentScores.length; i++) {
  System.out.println(studentNames[i] + " - " + studentScores[i]);
}
```

```
Sorted student scores:
Alan - 99.5
Grace - 95.0
Jane - 91.0
Ada - 88.5
John - 82.5
```

Usage of sorting algorithm

- Grade sorting program using bubble sort algorithm
  - make methods for swap; with method overloading

```
public static void swap(String[] str, int index1, int index2) {
  String temp = str[index1];
  str[index1] = str[index2];
  str[index2] = temp;
}
public static void swap(double[] num, int index1, int index2) {
  double temp = num[index1];
  num[index1] = num[index2];
  num[index2] = temp;
}
```

```
Sorted student scores:
Alan - 99.5
Grace - 95.0
Jane - 91.0
Ada - 88.5
John - 82.5
```

```
...
if (studentScores[j] < studentScores[j + 1]) {
  swap(studentScores, j, j+1);
  swap(studentNames, j, j+1);
...
```

3. String

What is a String?

- String ← Class
  - a sequence of character
    - ex) “Hello, World” ➔ ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘ ’, ‘W’, ‘o’, ‘r’, ‘l’, ‘d’ (공백도 문자로 취급)
    - note: String = “ ”, character = ‘ ’
  - technically, ‘String’ is a class as a fundamental part of Java
    - to store and manipulate sequence of characters
  - defined in ‘java.lang’ package, automatically imported

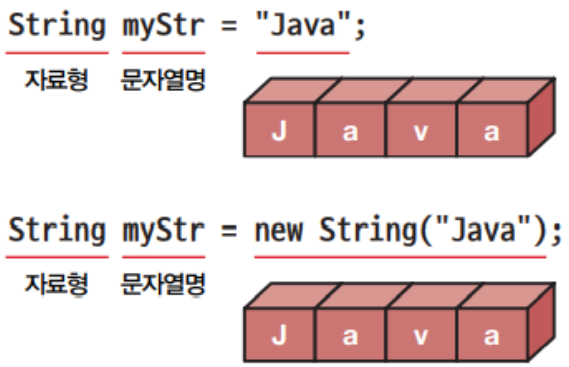
String declaration and initialization

- Creating Strings
  - 1) string literal: assigning a string literal directly to a variable (the most common method) by JVM

```
String myStr = "Java";
```

  - 2) ‘new’ keyword: using the ‘new’ keyword to create a new string object in the memory

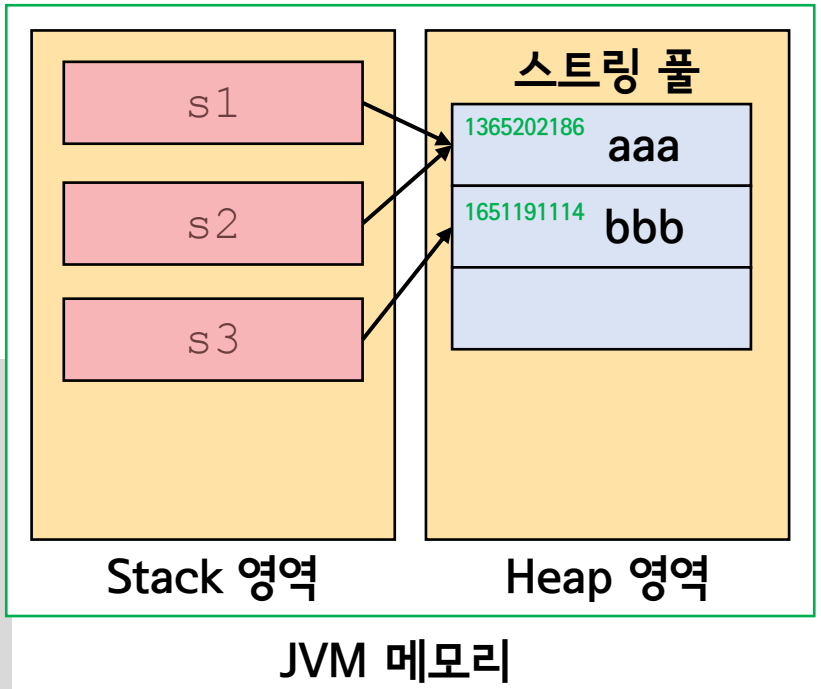
```
String myStr = new String("Java");
```



String declaration and initialization

- Memory positions in creating string literal
  - JVM에 의한 string 객체 생성
  - 같은 문자열이면 같은 장소(주소)에 저장됨

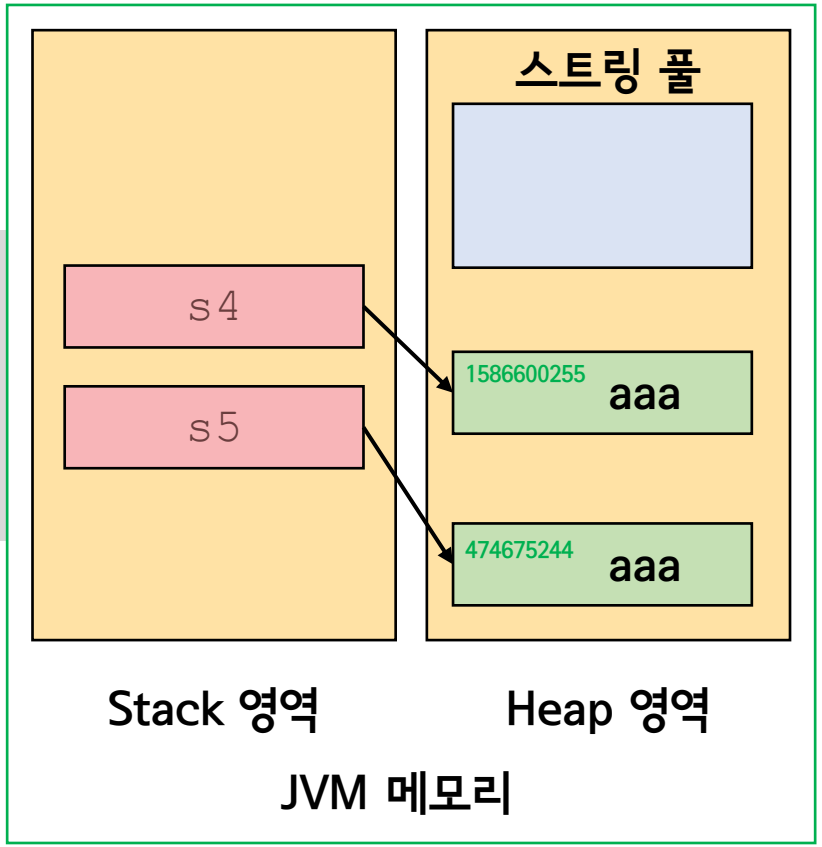
```
String s1 = "aaa";
String s2 = "aaa";
String s3 = "bbb";
System.out.println(s1); // Output: aaa
System.out.println(s2); // Output: aaa
System.out.println(s3); // Output: bbb
System.out.println(System.identityHashCode(s1)); // Output: 1365202186
System.out.println(System.identityHashCode(s2)); // Output: 1365202186
System.out.println(System.identityHashCode(s3)); // Output: 1651191114
```



String declaration and initialization

- Memory positions in creating string with ‘new’ keyword
  - 같은 문자열이어도 다른 주소에 저장됨

```
String s4 = new String("aaa");
String s5 = new String("aaa");
System.out.println(s4); // Output: aaa
System.out.println(s5); // Output: aaa
System.out.println(System.identityHashCode(s4)); // Output: 1586600255
System.out.println(System.identityHashCode(s5)); // Output: 474675244
```



Properties of String class

- Indexing
  - accessing individual characters in a string by their position

```
String s = "test";
System.out.println(s[0]); // Output: t
```

- Immutability
  - once created, the content of a ‘string’ cannot be changed

```
String s = "test";
s[0] = 'e'; // Error!
```

  - index를 통해 직접 접근하는 경우 string의 값을 바꿀 수 없음

Properties of String class

- String comparison
  - DO NOT USE '==' operation when comparing two strings
    - 문자열의 == 연산은 두 문자열의 주소가 같은지 묻는 연산임

```
String s1 = "aaa";
String s2 = "aaa";
String s3 = "bbb";

String s4 = new String("aaa");
String s5 = new String("aaa");

System.out.println("s1 == s2?: " + (s1 == s2)); // Output: true
System.out.println("s1 == s3?: " + (s1 == s3)); // Output: false
System.out.println("s4 == s5?: " + (s4 == s5)); // Output: false
```

Methods in String class

- Various methods in String class

메서드	설명
`int length()`	문자열 길이를 반환한다.
`boolean isEmpty()`	문자열이 비어 있는지 확인한다.
`char charAt(int index)`	특정 인덱스에 대한 char 값을 반환한다.
`String substring(int startIndex)`	주어진 시작 인덱스에 대한 부분 문자열을 반환한다.
`String substring(int startIndex, int endIndex)`	주어진 시작 인덱스와 끝 인덱스에 대한 부분 문자열을 반환한다.
`String concat(String str)`	두 문자열을 결합한다.
`int indexOf(char ch)`	지정된 문자의 인덱스를 반환한다.
`boolean equals(Object anotherObject)`	문자열과 객체가 같은지 확인한다.
`int compareTo(Object obj)`	문자열을 객체와 비교한다.
`String toLowerCase()`	문자열을 소문자로 반환한다.
`String toUpperCase()`	문자열을 대문자로 반환한다.
`String trim()`	선행 및 후행 공백을 생략한다.
`String replace(char oldChar, char newChar)`	문자열의 이전 문자를 새 문자 값으로 바꾼다.

Methods in String class

- Length method
  - *String*.length(): no parameters
  - return the number of characters in the strings, including spaces

```
String greeting = "Hello";
System.out.println(greeting.length());// Output: 5
```

```
String greeting = "Hello, world!!";
System.out.println(greeting.length()); // Output: 14
```

Methods in String class

- Finding a position of specific character
  - *String*.charAt(int index)
  - return the char value at the specified index

```
String greeting = "Hello";
char letter = greeting.charAt(1); // Output: 'e'
System.out.println(letter);
```

```
String greeting = "Hello";
char letter = greeting.charAt(5); // Error
System.out.println(letter);
```

Methods in String class

- Extracting substrings
  - *String*.substring(int beginIndex)
    - return a string that is a substring of this string from *beginIndex* to the end of string
  - *String*.substring(int beginIndex, int endIndex)
    - return a string that is a substring of this string from *beginIndex* to *endIndex*

```
String example = "Hello, World!";
String sub = example.substring(7, 12); // Output: "World"
System.out.println(sub);
```

```
String example = "Hello, World!";
String sub = example.substring(5); // Output: ", World!"
System.out.println(sub);
```

Methods in String class

- Finding the index for a specific character in string
  - *String*.indexOf(char c)
    - find the character in the string, and return the index for first occurrence
    - if not exists, return -1
  - *String*.lastIndexOf(char c)
    - find the character in the string, and return the index for last occurrence
    - if not exists, return -1

```
String myStrt = "abcdeabcde";
System.out.println(myStrt.indexOf('a')); // Output: 0
System.out.println(myStrt.lastIndexOf('a')); // Output: 5
System.out.println(myStrt.indexOf('x')); // Output: -1
```

Methods in String class

- Comparing strings
  - *String*.equals(String str)
    - compares two string for content equality
    - return true or false
  - *String*.equalIgnoreCase(String str)
    - compare two string with ignoring case differences
    - also return true or false

```
String str1 = "Java";
String str2 = "Java";
String str3 = "JAVA";
System.out.println(str1.equals(str2)); // Output: true
System.out.println(str1.equals(str3)); // Output: false
System.out.println(str1.equalsIgnoreCase(str3)); // Output: true
```

Methods in String class

- Converting case
  - *String*.toLowerCase()
  - *String*.toUpperCase()
    - converts all characters to lower/upper case
    - no parameters and return String class

```
String original = "JAVA Programming";
System.out.println(original.toLowerCase()); // Output: "java programming"
System.out.println(original.toUpperCase()); // Output: "Java Programming"
```

Methods in String class

- Modifying and combining strings
  - *String*.trim(): remove whitespace from both ends of a string

```
String padded = "     Java Programming    ";
String trimmed = padded.trim(); // Output: "Java Programming"
System.out.println(trimmed);
```

  - *String*.rplace(char oldchar, char newChar): replaces all occurrences of a specified char

```
String sentence = "Java is fun";
String replaced = sentence.replace('a', 'A'); // Output: "JAvA is fun"
System.out.println(replaced);
```

  - *String*.concat(String str): concatenates the specified string to the end of this string

```
String first = "Java ";
String second = "Programming";
String combined = first.concat(second); // Output: "Java Programming"
System.out.println(combined);
```

Methods in String class

- Splitting strings
  - *String*.split(String str)
    - divides a string into its constituent parts based on a given delimiter
    - return an array of substrings

```
String fruits = "apple,banana,cherry";
String[] splitFruits = fruits.split(","); // ["apple", "banana", "cherry"]
for (String fruit : splitFruits) {
  System.out.println(fruit);
}
```

```
String fruits = "apple, banana, cherry";
String[] splitFruits = fruits.split(", "); // ["apple", "banana", "cherry"]
for (String fruit : splitFruits) {
  System.out.println(fruit);
}
```

Methods in String class

- Splitting strings - advanced
  - *String*.split(String str)
    - using regular expressions (regex) as delimiters for more complex splitting scenarios

```
String sentence = "one1two2three3";
String[] words = sentence.split("\\d"); // ["one", "two", "three“]
```

    - ₩d is a regex that matches any digit
    - double backslash '₩₩' is used in Java string for escapting

Methods in String class

- Comparing strings lexicographically
  - String.compareTo(String str)
    - compares two strings lexicographically based on the Unicode value of each character in the strings
    - return 0 if the strings are equal
    - return a negative number if the first string is lexicographically less than the second
    - return a positive number if the first string is greater

```
String str1 = "apple";
String str2 = "banana";
String str3 = "apple";
System.out.println(str1.compareTo(str2)); // Output: negative number
System.out.println(str1.compareTo(str3)); // Output: 0
System.out.println(str2.compareTo(str1)); // Output: positive number
```

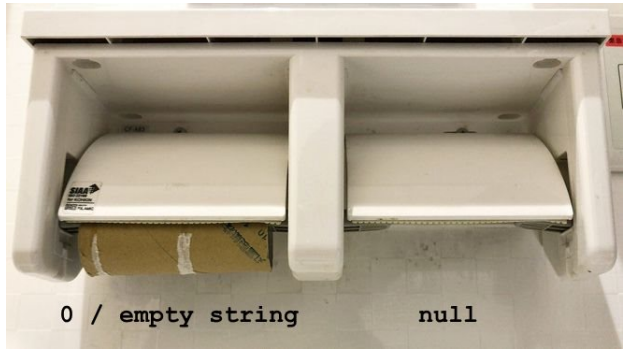
Methods in String class

- Checking for empty strings
  - *String*.isEmpty()
    - check if a string is empty or not
    - return if empty, otherwise false

```
String empty = "";
System.out.println(empty.isEmpty()); // true
```

Note: Null string

- Null
  - a reference that does not point to any object in memory
- Null string
  - no value at all




```
String str = null;
```

  - not the same as empty string
    - empty string (“”) is a string instance with zero length; no characters

```
String strNull1 = new String();
String strNull2 = "";
String strNull3 = null;

System.out.println(strNull1.isEmpty()); // Output: true
System.out.println(strNull2.isEmpty()); // Output: true
System.out.println(strNull3.isEmpty()); // Error! - NullPointerException
```

Note: Null string

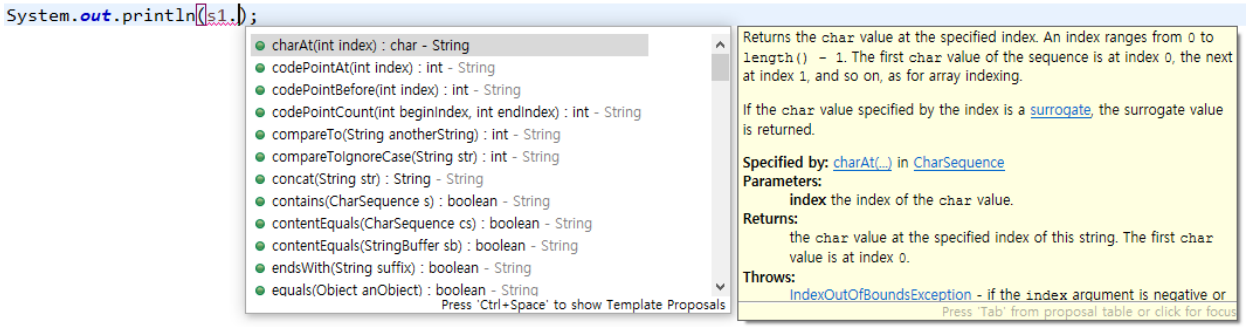
- Common usage of null check
  - before uses a string object

```
String str = null;
if (str == null) {
  System.out.println("The string is null.");
} else {
  System.out.println("The string is not null.");
}
```

```
String anotherStr = "Java";
if (str != null && str.equals(anotherStr)) {
  System.out.println("The strings are equal.");
} else {
  System.out.println("The strings are not equal or str is null.");
}
```

Tips

- You can check the how to use the method in String class in Eclipse



- You can find the more methods and descriptions on

  https://www.w3schools.com/java/java_ref_string.asp

All String Methods

The String class has a set of built-in methods that you can use on strings.

Method	Description	Return Type
charAt()	Returns the character at the specified index (position)	char
codePointAt()	Returns the Unicode of the character at the specified index	int
codePointBefore()	Returns the Unicode of the character before the specified index	int
codePointCount()	Returns the number of Unicode values found in a string.	int
compareTo()	Compares two strings lexicographically	int
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences	int
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean
contentEquals()	Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer	boolean
copyValueOf()	Returns a String that represents the characters of the character array	String

Usage of methods in String class

- Remove all space in the string

```
String text = "What are you doing?";
String newText = text.replace(" ", ""); // Output: Whatareyoudoing?
System.out.println(newText);
```

- Count the spaces in the string

```
String text = "Count the spaces";
int spaces = text.length() - text.replace(" ", "").length(); // Output: 2
```

- Extract file name

```
String filename = "document.pdf";
int dotPosition = filename.indexOf('.'); // Output: 8
filename = filename.substring(0, dotPosition);
System.out.println("File name: " + filename); // Output: File name: document
```

Usage of methods in String class

- Extract file name from full path

```
String fullPath = "C:\\user\\user\\document\\subfolder\\textfile.txt";
int filePosition = fullPath.lastIndexOf('\\');
String filename = fullPath.substring(filePosition+1, fullPath.length());
System.out.println("File name: " + filename); // Output: File name: textfile.txt
```

- Split into array of strings from a comma-separated values (CSV)

```
String csvLine = "John,Doe,30,New York";
String[] values = csvLine.split(",");
System.out.print("Values: ");
for (int i = 0; i < values.length; i++) {
  System.out.print(values[i]);
  if (i < values.length - 1) {
    System.out.print(", ");
  }
}
```

```
Values: John, Doe, 30, New York
```

Usage of methods in String class

- Count the words

```
String document = "The apple is sweet. I like apple.";
String searchFor = "apple";
int count = 0;
int fromIndex = 0;
  while ((fromIndex = document.indexOf(searchFor, fromIndex)) != -1 ) {
    count++;
    fromIndex++;
  }
System.out.println("The word '" + searchFor + "' appears " + count + " times.");
```

```
The word 'apple' appears 2 times.
```

Examples and practices for String class

- 사용자로부터 문자열을 입력받고, 입력받은 문자열을 거꾸로 출력하는 프로그램을 작성해보세요.
  - file path and name: Chap05Example/StringPractice01.java
  - inputs and outputs

```
Enter the string: Hello, World!!
Original string: Hello, World!!
Reversed string: !!dlroW ,olleH
```

```
Enter the string: 오 필승 코리아
Original string: 오 필승 코리아
Reversed string: 아리코 승필 오
```

Examples and practices for String class

- 사용자로부터 문자열을 입력받고, 입력받은 문자열의 공백을 제외한 문자의 개수를 세는 프로그램을 작성해보세요.
  - file path and name: Chap05Example/StringPractice02.java
  - requirement
    - 원래의 문자열을 변형 해서는 안됨
  - inputs and outputs

```
Enter the string: Hi, this is Java class.
Origin string: Hi, this is Java class.
The number of character without sapces: 19
```

4. Exception handling

Concept of exception handling

- Exceptions
  - an event that disrupts the normal flow of a program's instructions
  - 프로그램 실행 중 오동작이나 결과에 악영향을 미치는 예상치 못한 상황이 발생
  - example
    - 분모가 0인 경우
    - null string에 값을 대입하는 경우
    - 값을 3개 받아야하는데 2개만 받은 경우
    - array의 크기보다 큰 index 혹은 음수 index로 array에 접근하는 경우
    - 주민번호 입력란에 문자를 입력하는 경우
    - 영어 이름 입력란에 한글을 입력하는 경우
    - etc.

Concept of exception handling

- Exception example
  - 분모가 0인 경우

```
Scanner scanner = new Scanner(System.in);
int dividend; // 나뉨수
int divisor; // 나눔수
System.out.print("Enter the dividend: ");
dividend = scanner.nextInt();
System.out.print("Enter the divisor: ");
divisor = scanner.nextInt();
System.out.println(dividend + " / " + divisor + " = " + dividend/divisor);
```

```
Enter the dividend: 5
Enter the divisor: 4
5 / 4 = 1
```

```
Enter the dividend: 5
Enter the divisor: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Chap05Example.ExceptionExample01.main(ExceptionExample01.java:18)
```

Concept of exception handling

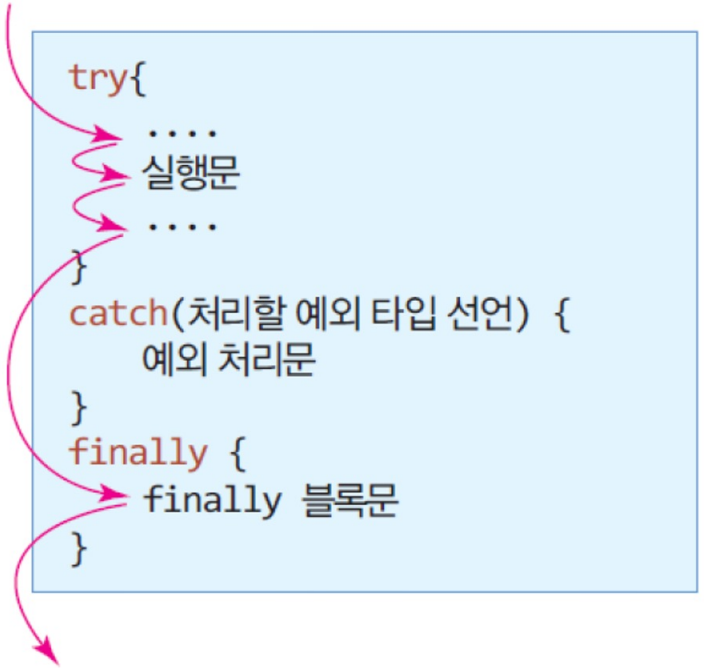
- Exception handling
  - allows you to “catch” exceptions thrown by a program and take corrective actions
    - rather than letting the program terminate unexpectedly
  - ➔ 실행 중 발생하는 error를 예외로 처리하여 프로그램 가동을 멈추지 않도록 함
  - USE ‘try-catch’ or ‘try-catch-finally’ block

```
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)
}
```

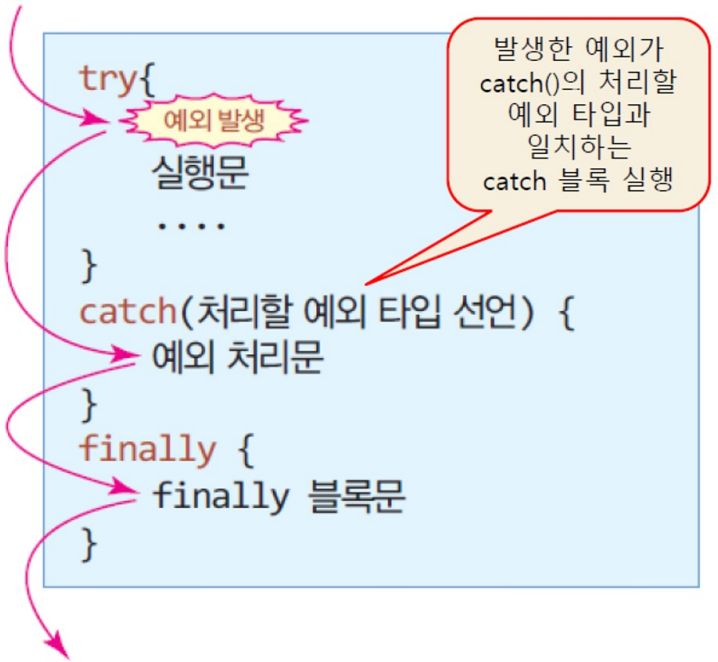
생략 가능

Concept of exception handling

- Flow of exception handling
  - flow when an error does not occur



  - flow when an error occurs



Exception types in Java

- Java provides the primitive exception types

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

‘try-catch’ block

- Basic try-catch example

```
try {
  int division = 10 / 0; // This will cause a divide-by-zero error
} catch (ArithmeticException e) {
  System.out.println("ArithmeticException caught: Cannot divide by zero.");
}
```

```
ArithmeticException caught: Cannot divide by zero.
```

```
int intArray[] = new int[5];
try {
  intArray[3] = 10;
  intArray[7] = 5; // Exception!
} catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("배열의 범위를 초과하여 원소에 접근하였습니다.");
}
```

```
배열의 범위를 초과하여 원소에 접근하였습니다.
```

‘try-catch’ block

- try-catch-finally example

```
try {
  String text = null;
  System.out.println(text.length());
} catch (NullPointerException e) {
  System.out.println("NullPointerException caught: String is null.");
} finally {
  System.out.println("This block is executed regardless of exceptions.");
}
```

```
NullPointerException caught: String is null.
This block is executed regardless of exceptions.
```

Usage of exception handling

- Sum of three integers by user input
  - retrying on non-integer input
  - `import java.util.InputMismatchException;`

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the three integers");
int sum = 0, n = 0;
for (int i = 0; i < 3; i++) {
  System.out.print(i+ ">> ");
  try {
    n = scanner.nextInt(); // only integer;
  } catch (InputMismatchException e) {
    System.out.println("Not integer type. Please enter again");
    scanner.next(); // 입력 스트림에 있는 정수가 아닌 token을 버림
    i--;
    continue;
  }
  sum += n;
}
System.out.println("Sum = " + sum);
scanner.close();
```

```
Enter the three integers
0>> 5
1>> 4
2>> 1
Sum = 10
```

```
Enter the three integers
0>> 4
1>> R
Not integer type. Please enter again
1>> 2
2>> Q
Not integer type. Please enter again
2>> c
Not integer type. Please enter again
2>> 7
Sum = 13
```

End of slide