

Sorting and String

Byeongjoon Noh

Dept. of AI/Bigdata, SCH Univ.

powernoh@sch.ac.kr

Contents

1. Sorting
2. String handling

1. Sorting

Sorting이란?

Sorting (정렬)

- 데이터의 집합을 어떤 기준의 대소관계를 따져 일정한 순서로 줄지어 세우는 것
- 데이터의 집합: 1, 5, 6, 7, 2, 4, 9, 8, 3
- 오름차순: 1, 2, 3, 4, 5, 6, 7, 8, 9
- 내림차순: 9, 8, 7, 6, 5, 4, 3, 2, 1

정렬 알고리즘

- 다양한 정렬 알고리즘
 - Bubble sort, Selection, Insertion, Merge, Quick, Heap, etc...
 - 상황별로 유리한 정렬 알고리즘이 존재함

Bubble sort

Concept

- 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘
 - ➔ 인접한 원소를 비교하여 크기가 순서대로 되어 있지 않으면 서로 위치 **교환**

Swap

- 두 변수(혹은 원소)의 값을 바꾸는 것
- 임시 저장소 (temp 변수)를 활용!!

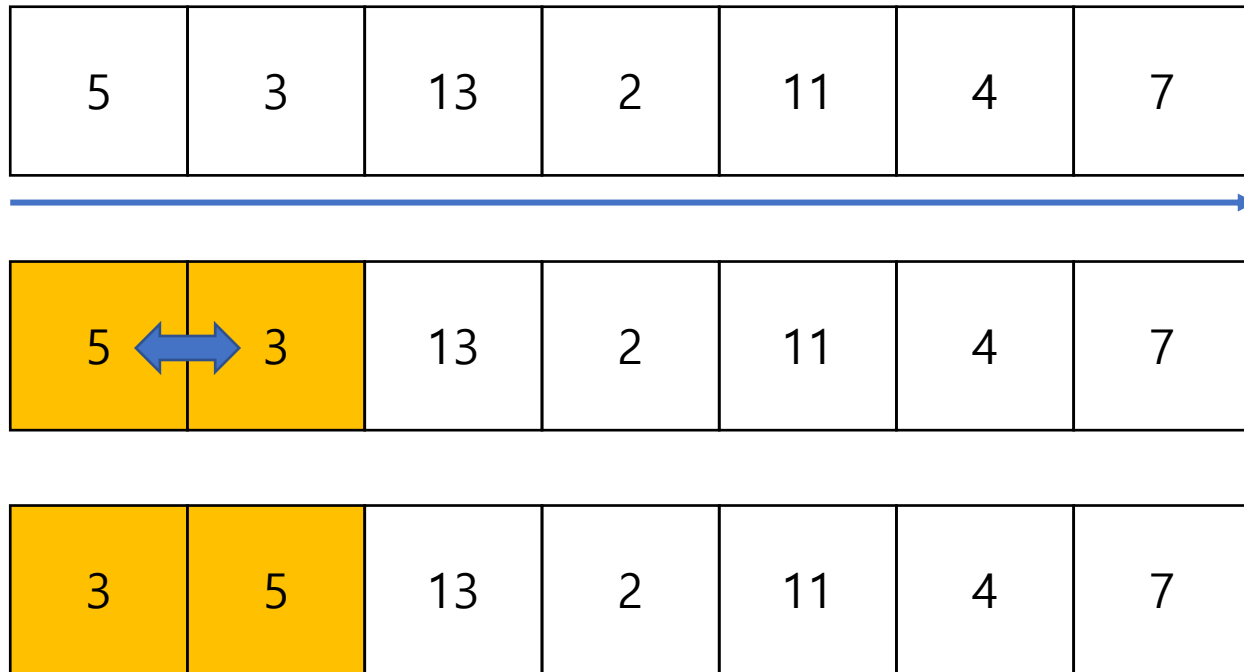
```
int a = 10;  
int b = 5;  
int temp;
```

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

Bubble sort

Processing

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



Bubble sort

3	5	13	2	11	4	7
---	---	----	---	----	---	---

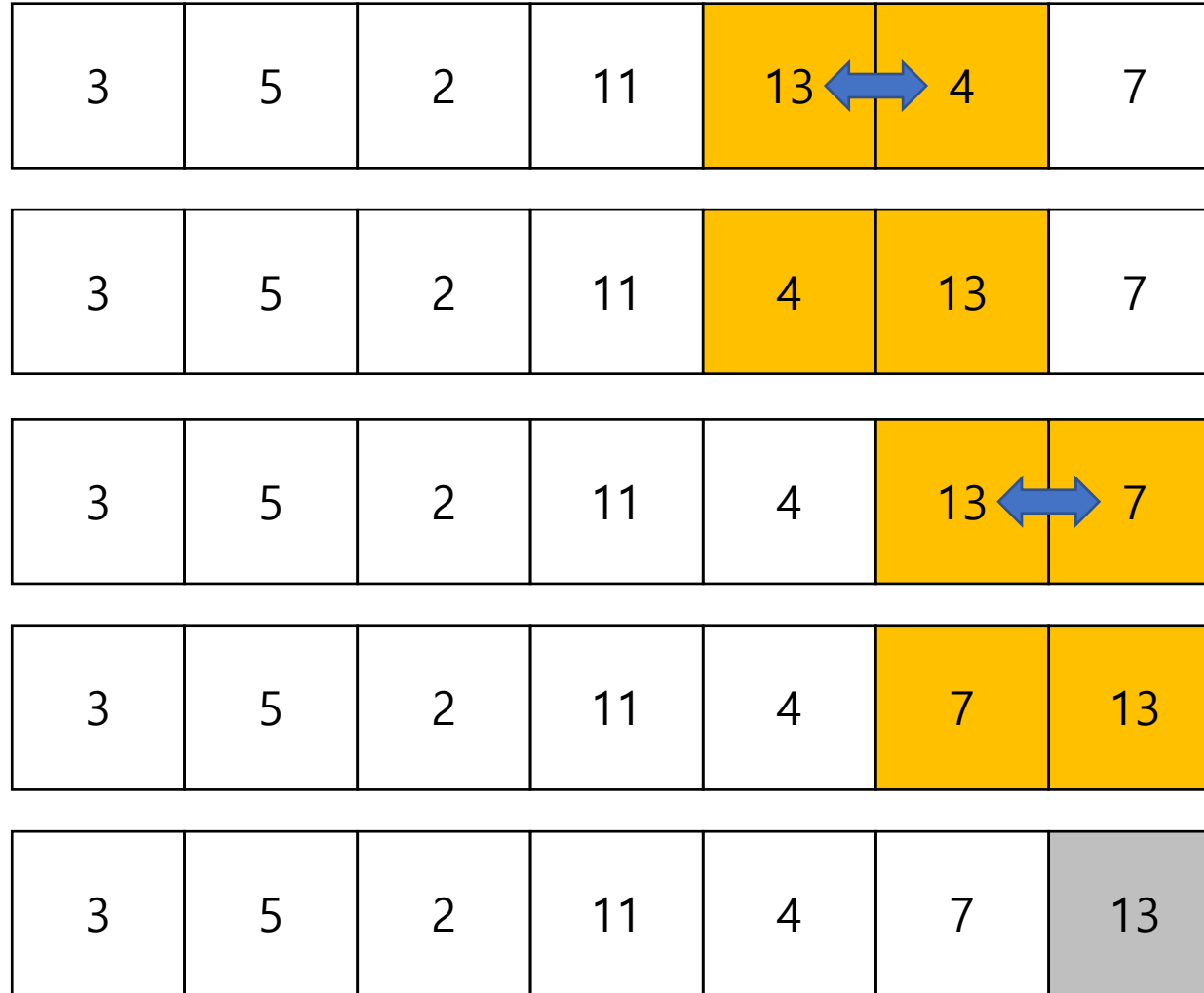
3	5	13	↔	2	11	4	7
---	---	----	---	---	----	---	---

3	5	2	13	11	4	7
---	---	---	----	----	---	---

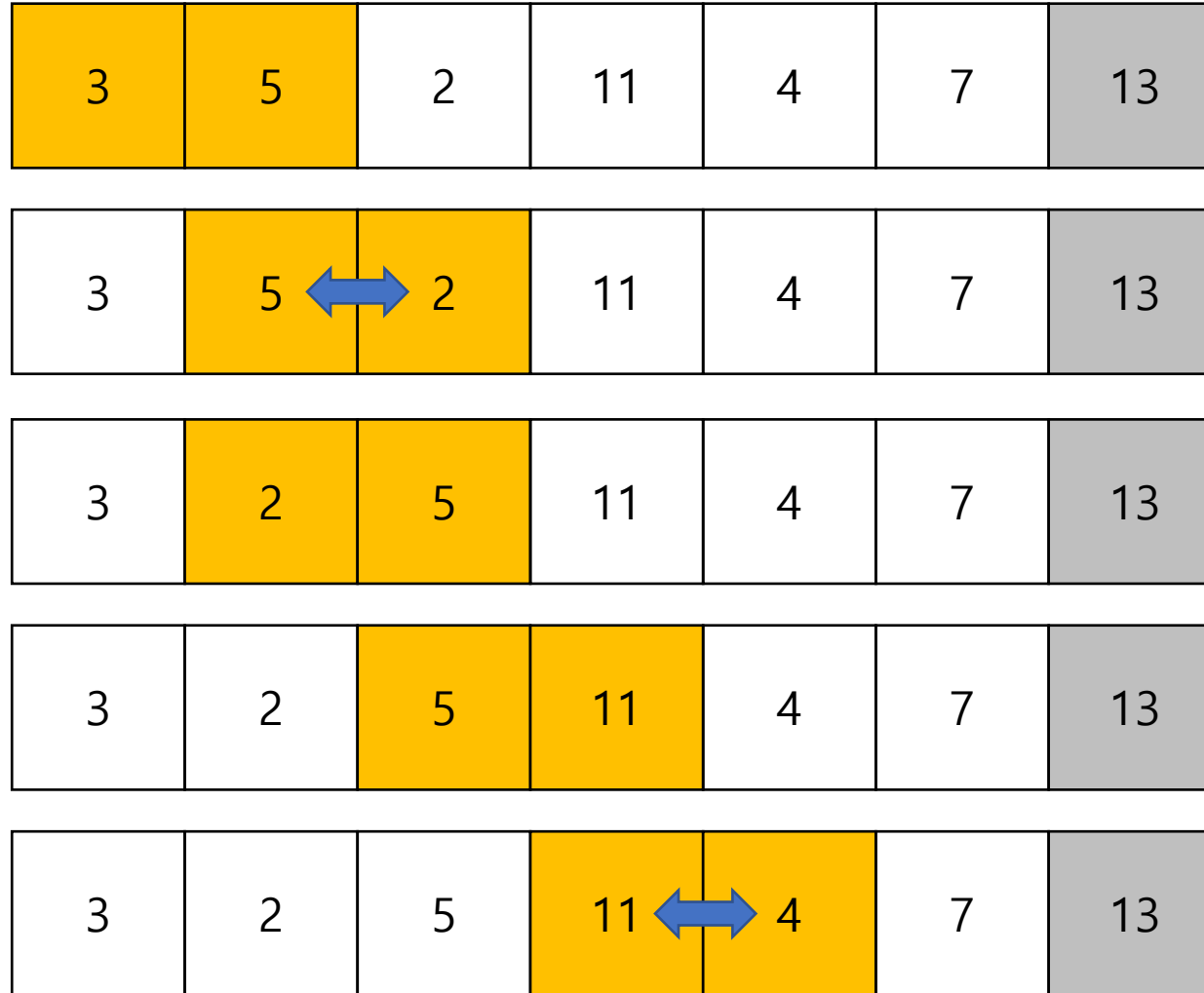
3	5	2	13	↔	11	4	7
---	---	---	----	---	----	---	---

3	5	2	11	13	4	7
---	---	---	----	----	---	---

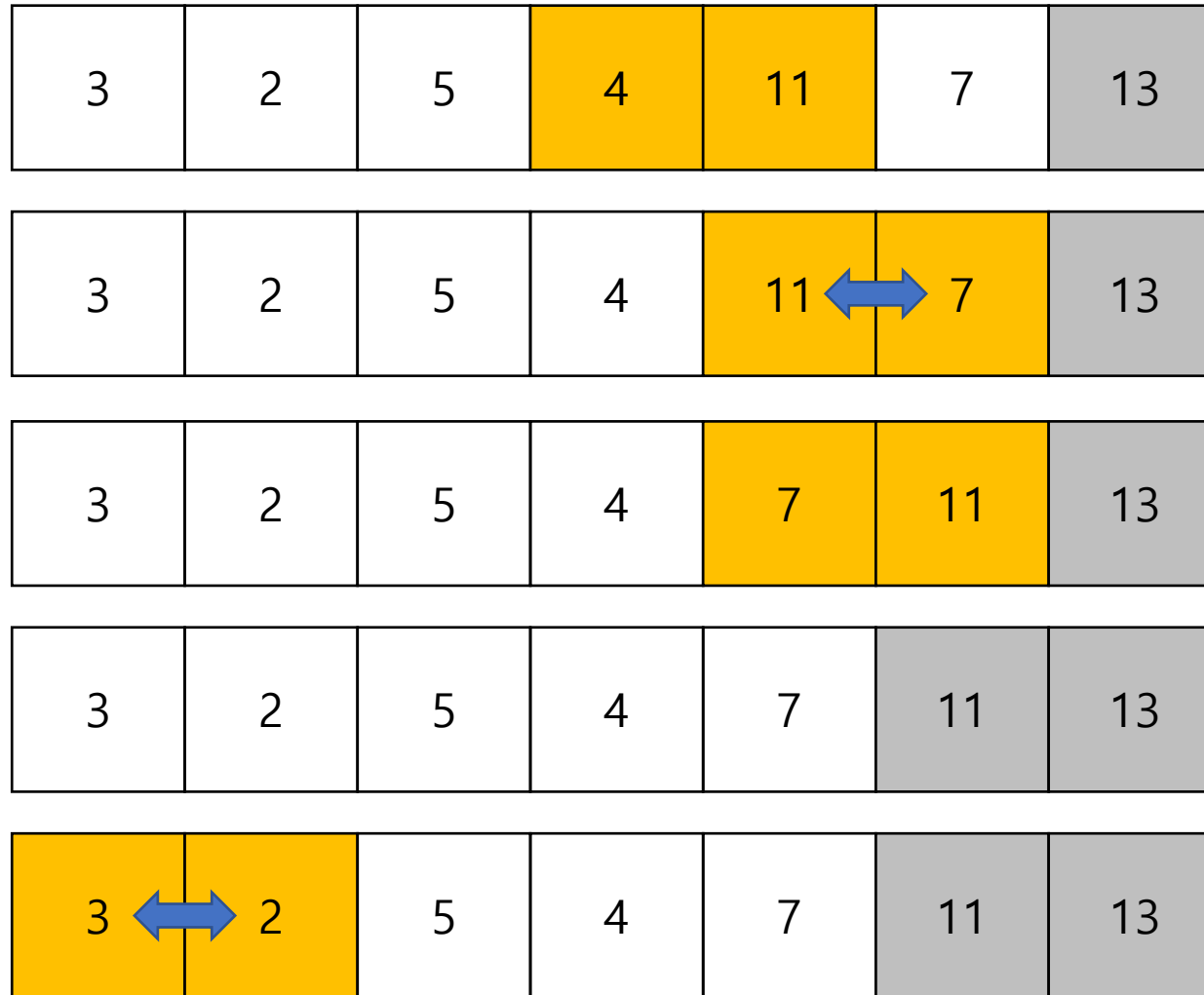
Bubble sort



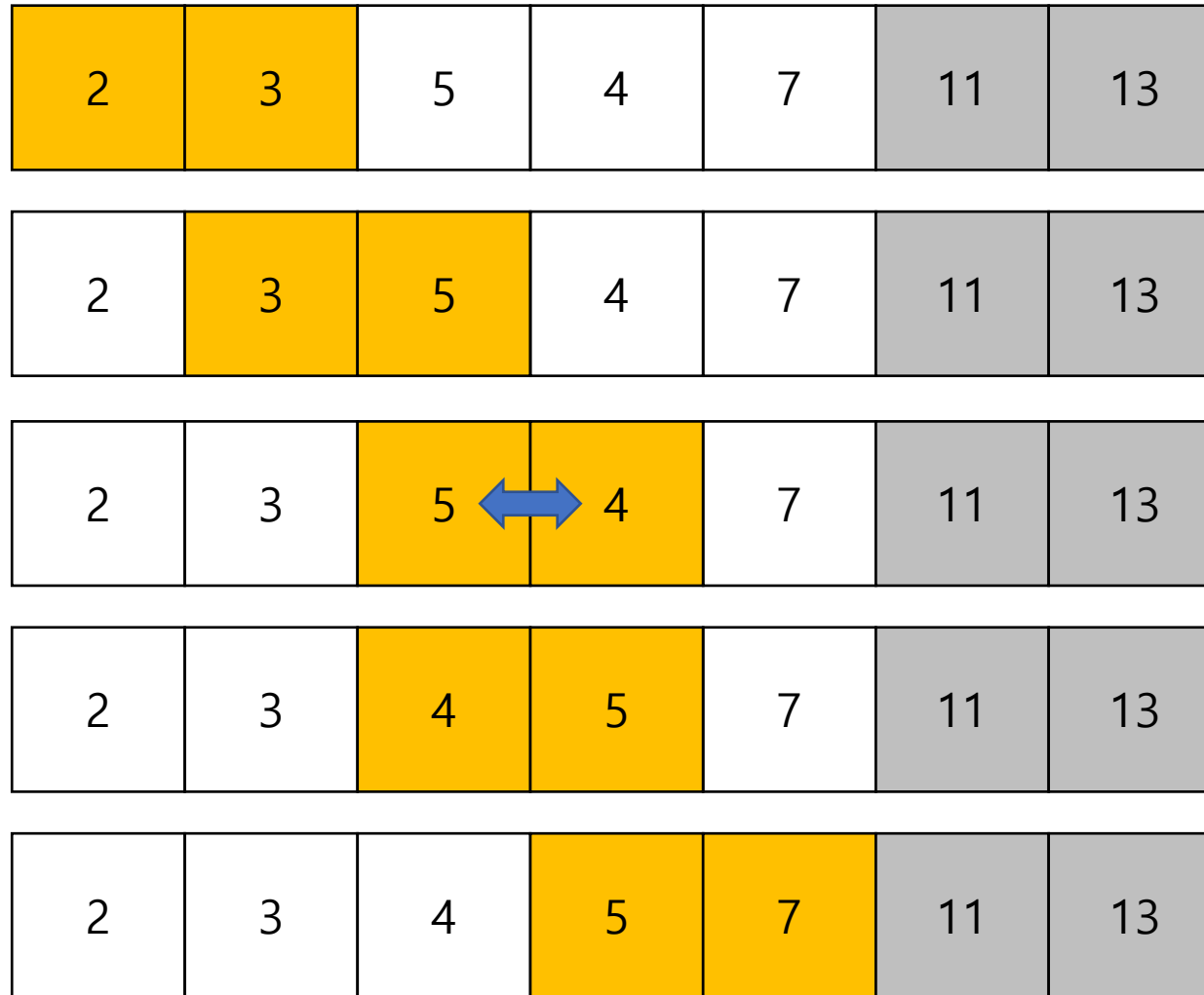
Bubble sort



Bubble sort



Bubble sort



Bubble sort

2	3	4	5	7	11	13
2	3	4	5	7	11	13
2	3	4	5	7	11	13
2	3	4	5	7	11	13
2	3	4	5	7	11	13

Bubble sort

2	3	4	5	7	11	13
---	---	---	---	---	----	----

2	3	4	5	7	11	13
---	---	---	---	---	----	----

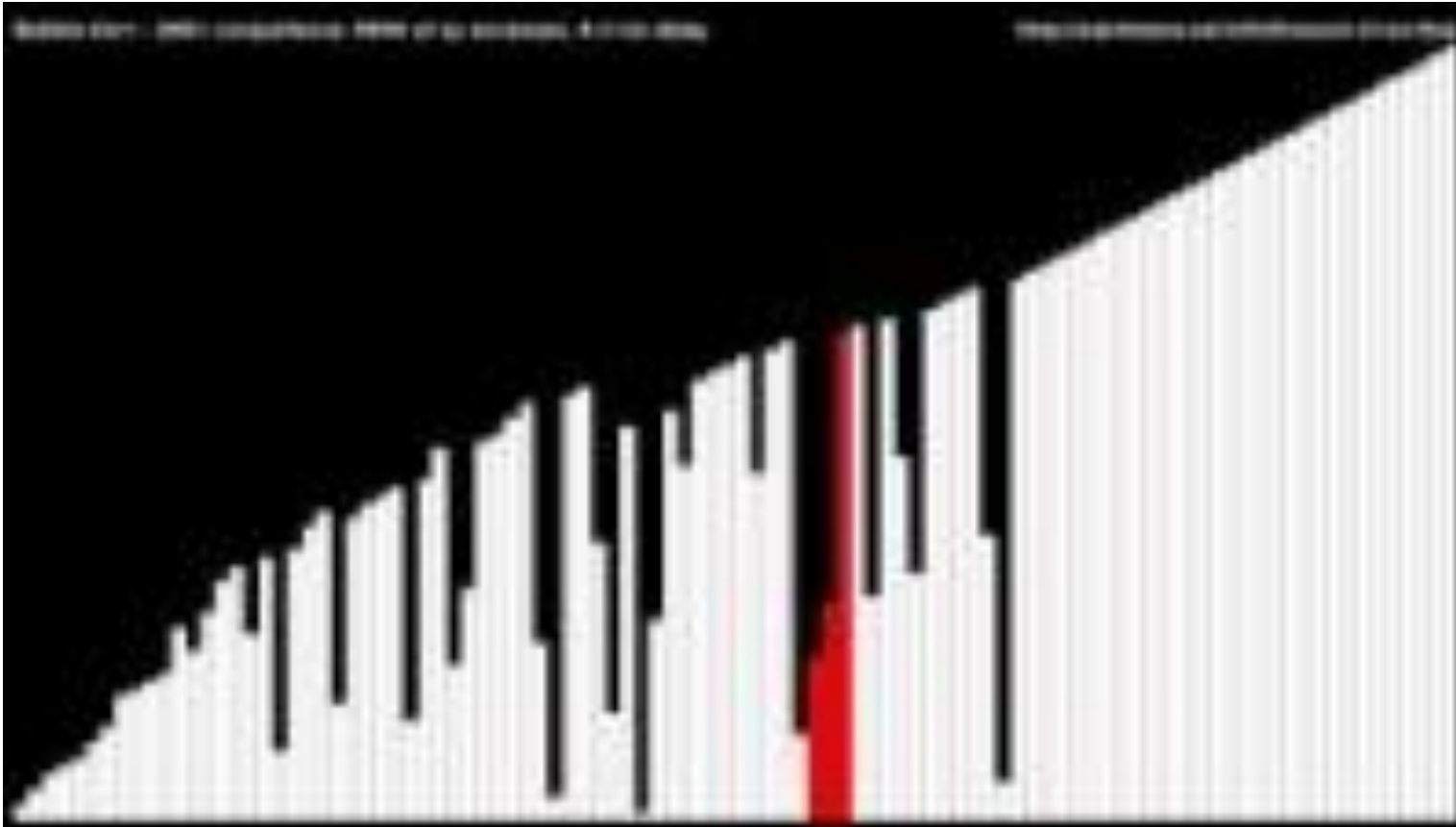
2	3	4	5	7	11	13
---	---	---	---	---	----	----

2	3	4	5	7	11	13
---	---	---	---	---	----	----

2	3	4	5	7	11	13
---	---	---	---	---	----	----

Bubble sort

Simulation



Bubble sort

Implementation

```
int[] arr = {8, 54, 99, 3, 2, 1, 0};  
final int length = arr.length;  
  
for (int i = 0; i < length - 1; i++) { // 배열의 길이만큼 Loop  
    for (int j = 0; j < length - i - 1; j++) { // 0 ~ size-1까지  
        if (arr[j] > arr[j + 1]) { // 인접원소간 비교 → 자리교체  
            int temp = arr[j];  
            arr[j] = arr[j + 1];  
            arr[j + 1] = temp;  
        }  
    }  
}
```

Selection sort

Concept

- 현재 위치에 들어갈 데이터를 찾아 선택하는 알고리즘
- 1. 주어진 array에서 최솟값(혹은 최댓값)을 찾는다.
- 2. 최솟값을 맨 앞자리 (혹은 맨 뒷자리)와 교환한다.
- 3. 맨 앞자리(혹은 맨 뒷자리) 원소 빼고 나머지를 1, 2 과정을 반복한다.

Selection sort

Processing

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

5	3	13	2	11	4	7
---	---	----	---	----	---	---

최댓값 13

5	3	13	2	11	4	7
---	---	----	---	----	---	---

3	5	7	2	11	4	13
---	---	---	---	----	---	----

Selection sort

최댓값 11

3	5	7	2	11	4	13
---	---	---	---	----	---	----

3	5	7	2	11	4	13
---	---	---	---	----	---	----

3	5	7	2	4	11	13
---	---	---	---	---	----	----

최댓값 7

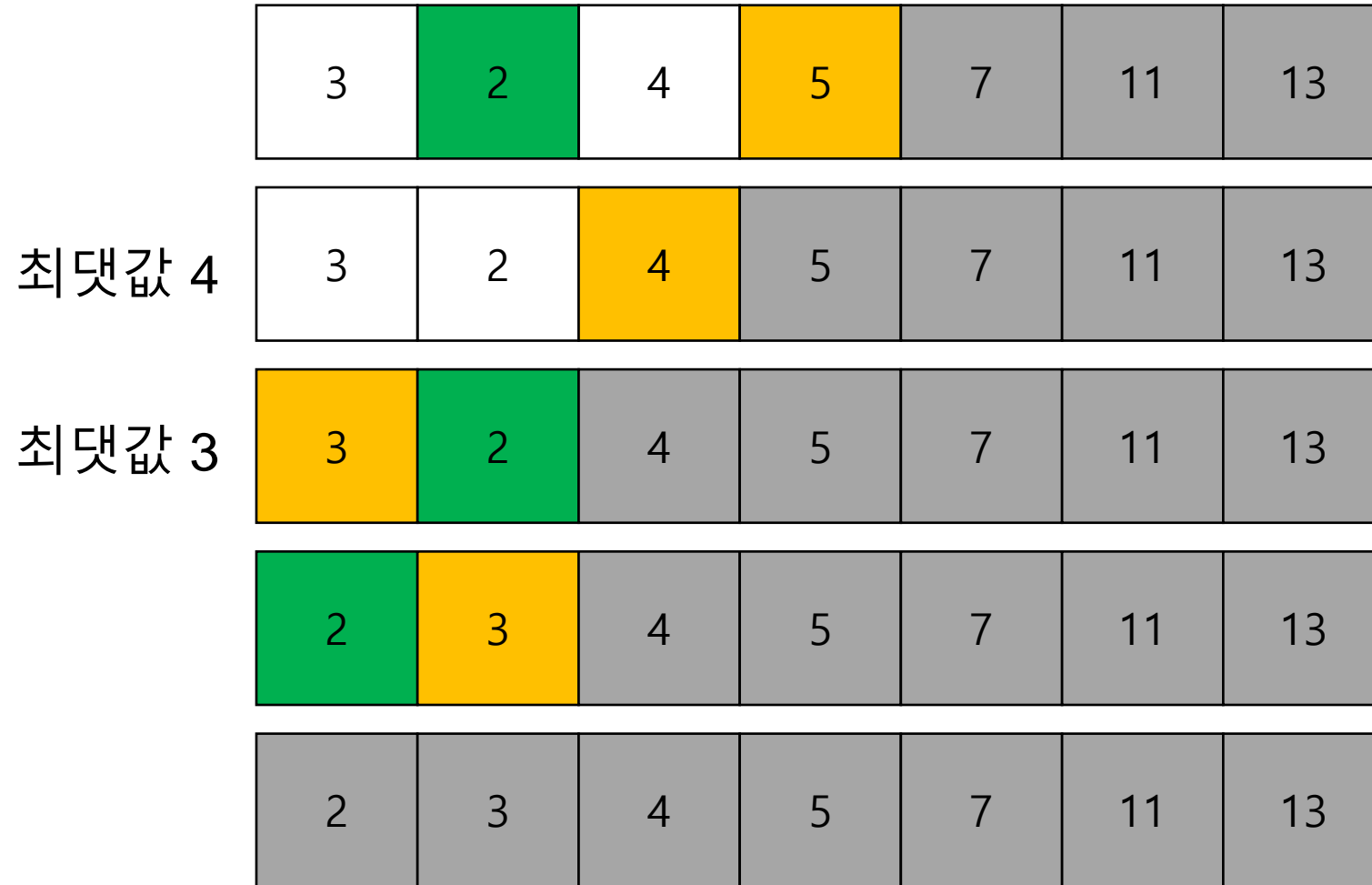
3	5	7	2	4	11	13
---	---	---	---	---	----	----

3	5	4	2	7	11	13
---	---	---	---	---	----	----

최댓값 5

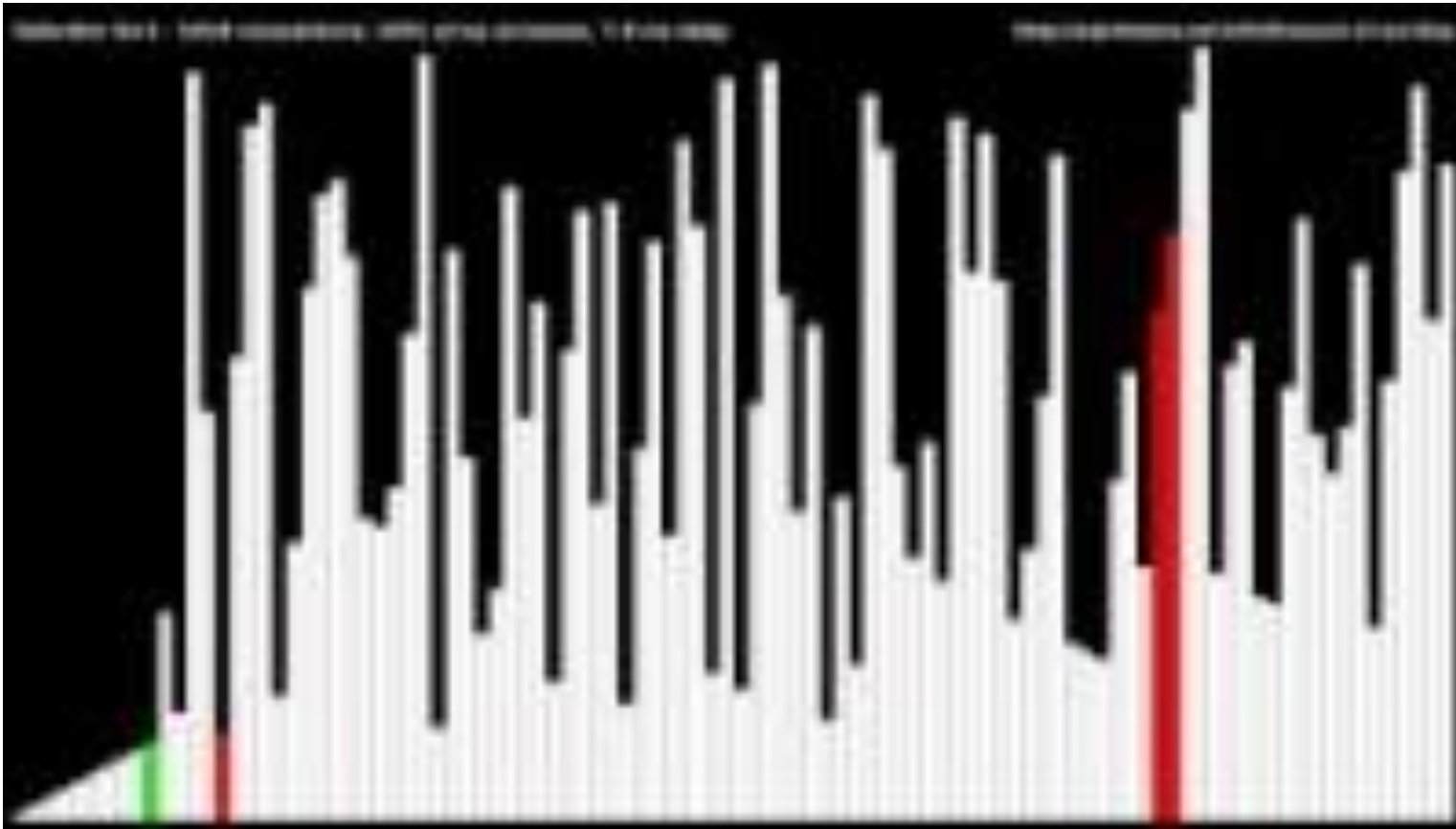
3	5	4	2	7	11	13
---	---	---	---	---	----	----

Selection sort



Selection sort

Simulation



Selection sort

Implementation

```
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;  
}
```

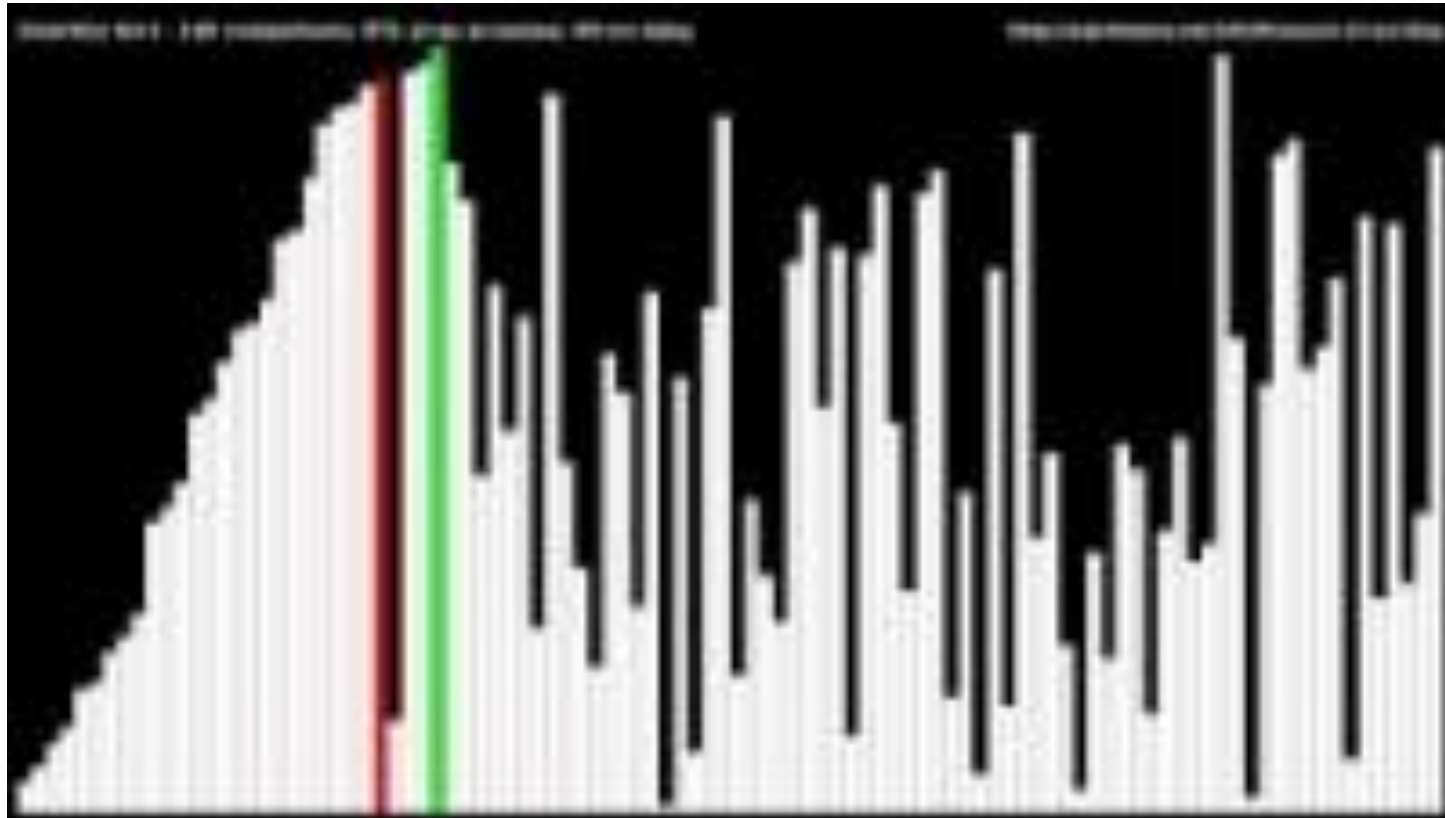
Insertion sort

Concept

- 앞에서부터 해당 원소가 위치할 곳을 탐색 ➔ 해당 위치에 삽입
- 1. 현재 타겟이 되는 숫자와 이전 위치에 있는 원소들을 비교한다.
 - 첫번째 타겟은 두번째 원소부터 시작
- 2. 타겟이 되는 숫자가 이전 위치에 있던 원소보다 작다면 위치를 교환한다.
- 3. 그 다음 타겟을 찾아 1, 2 과정을 반복한다.

Insertion sort

Simulation



Summary

Bubble sort

- 서로 붙은 원소의 “자리 바꾸기”

Selection sort

- 비교하는 원소 중 “최솟값 찾기”

Sorting in method

Grade sorting in method

```
public static void gradeSorting(int[] arr) {  
    int n = arr.length;  
    int temp;  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            if (arr[j] > arr[j+1]) {  
                temp = arr[j];  
                arr[j] = arr[j+1];  
                arr[j+1] = temp;  
            }  
        }  
    }  
}
```

```
public static void main(String[] args) {  
    Scanner input = new Scanner(System.in);  
    // int n = input.nextInt();  
    // int[] grades = new int[n];  
    int[] grades = {10, 50, 71, 23, 13, 72, 99, 100, 14, 67, 42};  
    int len = grades.length;  
  
    gradeSorting(grades);  
  
    System.out.println("Sorted grades:");  
    for (int i = 0; i < len; i++) {  
        System.out.println(grades[i]);  
    }  
}
```

2. String handling

String 클래스

문자열 개요

- Java에서는 문자열을 객체(object)로 취급
- java.lang 패키지에 포함
- java.lang.String 클래스와 java.lang.StringBuffer 클래스 제공 (기본 포함)

```
String s = "I like Java and Python and C.";
System.out.println(s);
```

String 클래스

문자열 생성 예

- JVM에 의한 객체 생성
 - 같은 문자열 → 같은 주소에 저장

```
String s1 = "aaa";  
String s2 = "aaa";  
String s3 = "bbb";  
System.out.println(s1);  
System.out.println(s2);  
System.out.println(s3);  
System.out.println(System.identityHashCode(s1));  
System.out.println(System.identityHashCode(s2));  
System.out.println(System.identityHashCode(s3));
```

Memory (Java heap)

String constant pool

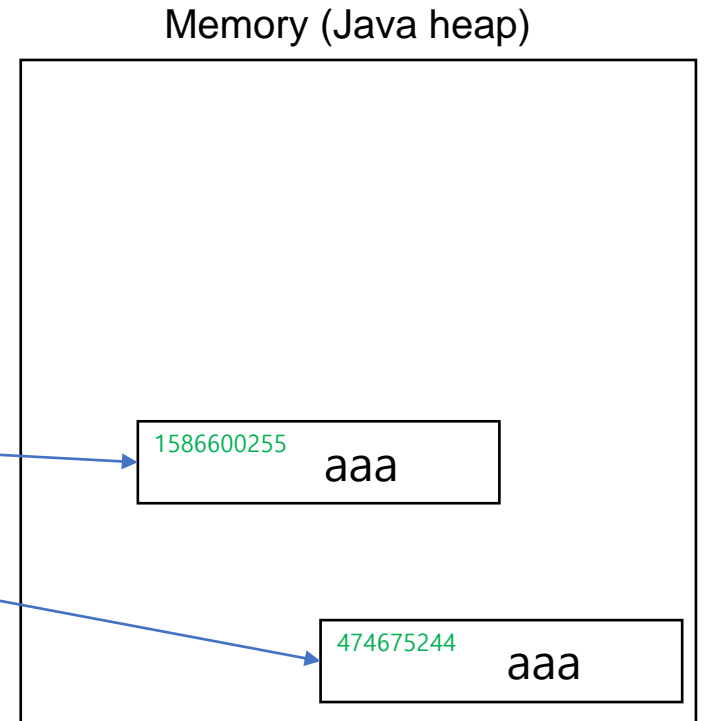
1365202186	aaa
1651191114	bbb

String 클래스

문자열 생성 예

- 사용자에 의한 객체 생성
 - 같은 문자열 → 다른 주소에 저장

```
String s4 = new String("aaa");  
String s5 = new String("aaa");  
System.out.println(s4);  
System.out.println(s5);  
System.out.println(System.identityHashCode(s4));  
System.out.println(System.identityHashCode(s5));
```



String 클래스

문자열 생성 예

- null String
 - String 객체의 내용을 null로 초기화
 - But, 객체에 값을 초기화한 경우는 null String이 아님.

```
String strNull1 = new String();  
String strNull2 = "";  
String strNull3 = null;  
System.out.println(strNull1.isEmpty());  
System.out.println(strNull2.isEmpty());  
System.out.println(strNull3.isEmpty());
```

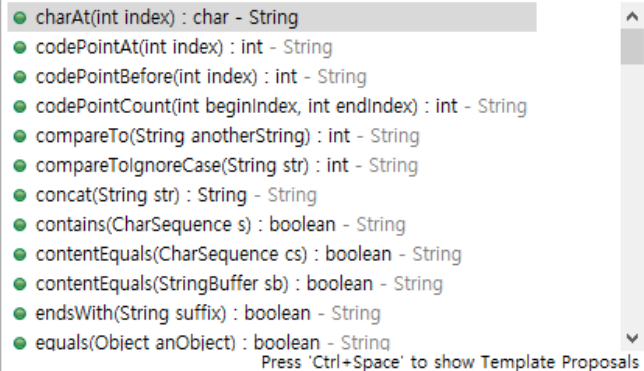
주요 메소드

* Method

- 클래스 내부에 선언된 함수
- 접근 방법 “.”

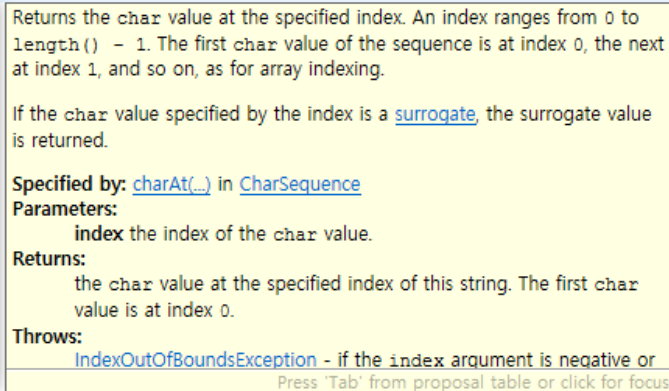
```
String s1 = "Java";  
System.out.println(s1.equals("Java"));
```

```
System.out.println(s1.);
```



● charAt(int index) : char - String
● codePointAt(int index) : int - String
● codePointBefore(int index) : int - String
● codePointCount(int beginIndex, int endIndex) : int - String
● compareTo(String anotherString) : int - String
● compareToIgnoreCase(String str) : int - String
● concat(String str) : String - String
● contains(CharSequence s) : boolean - String
● contentEquals(CharSequence cs) : boolean - String
● contentEquals(StringBuffer sb) : boolean - String
● endsWith(String suffix) : boolean - String
● equals(Object anObject) : boolean - String

Press 'Ctrl+Space' to show Template Proposals



Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

If the char value specified by the index is a surrogate, the surrogate value is returned.

Specified by: charAt(...) in CharSequence

Parameters:
index the index of the char value.

Returns:
the char value at the specified index of this string. The first char value is at index 0.

Throws:
IndexOutOfBoundsException - if the index argument is negative or

Press 'Tab' from proposal table or click for focus

All String Methods

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

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

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

주요 메소드

String 클래스 주요 메소드

- String 클래스는 내부에 편의를 위한 다양한 메소드가 미리 정의되어 있음
- String(), String(byte[]), String(char[]), String(String) 등
 - 생성자의 역할
- int length(): 문자열의 길이를 반환
 - Array에서는 array.length ← method가 아님!
- int charAt(int): 문자열에서 i번째 문자를 반환
- String concat(String): 문자열과 문자열을 결합함
- String substring(int): 문자열을 자름
- int indexOf(char): 문자가 위치한 인덱스를 반환함

주요 메소드

String 클래스 주요 메소드

- `String replace(String, String)`: 문자열의 일부를 다른 문자열로 바꿈
- `String toUpperCase()`, `String toLowerCase()`: 문자열을 대문자/소문자로 변환
- `String trim(String)`: 문자열의 앞뒤 공백을 제거
- `Boolean equals(String)`, `Boolean equalsIgnoreCase(String)`: 문자열 비교
- `int compareTo(String)`: 문자열 비교

char charAt(int)

해당 index에 위치한 문자를 반환하는 메소드

```
String str = "Hello Java";  
System.out.println(str.charAt(1));  
// 출력 결과 : e
```

int length()

문자열의 길이를 반환함

- 문자열 앞, 중간, 뒤의 공백을 포함하여 count (“” 기준)

```
String str = "Hello Java";  
System.out.println(str.length());  
// 출력결과 : 10
```

String concat(String)

문자열과 문자열을 결합함

```
String str = "Hello ";  
String lang = "Java";  
System.out.println(str.concat(lang));  
// 출력결과 : Hello Java
```

String substring(int)

문자열을 자르는 메소드

- Parameter에 자르는 처음~끝(바로 앞) index
- Parameter가 1개 일 경우 해당 index부터 끝까지 자름

```
String str = "Hello Java";  
String lang = str.substring(6, 7);  
System.out.println(lang);  
// 출력 결과 : J
```

```
String str = "Hello Java";  
String lang = str.substring(3, 8);  
System.out.println(lang);  
// 출력 결과 : lo Ja
```

```
String str = "Hello Java";  
String lang = str.substring(6);
```

```
System.out.println(lang);  
// 출력 결과 : Java
```

int indexOf(char)

문자가 위치한 index를 반환하는 메소드

- Parameter로 char 데이터 타입의 문자를 입력 받음
- 문자가 2개 이상인 경우 가장 먼저 오는 문자의 index를 반환함
 - lastIndexOf(): 가장 뒤에 오는 문자의 index를 반환함

```
String str = "Hello Java";  
System.out.println(str.indexOf('a'));  
// 출력 결과 : 7
```

```
String str = "Hello Java";  
System.out.println(str.lastIndexOf('a'));  
// 출력 결과 : 9
```

String replace(String, String)

문자열의 일부를 다른 문자열로 바꾸는 메소드

- 첫번째 parameter: 바뀔 문자열
- 두번째 parameter: 새로운 (바꿀) 문자열
- 바뀔 문자열이 해당 문자열에 없으면 원래 문자열을 반환함

```
String str = "Hello Java";  
String newStr = str.replace("Java", "Python");
```

```
System.out.println(newStr);  
// 출력 결과 : Hello Python
```

```
String str = "Hello Java";  
String lang = str.replace("test", "test");  
System.out.println(lang);  
// 출력 결과 : Hello Java
```

String toUpperCase() String toLowerCase()

문자열의 대/소문자를 변환하는 메소드

```
String str1 = "hello java";  
String strUp = str1.toUpperCase();
```

```
System.out.println(strUp);  
// 출력 결과 : HELLO JAVA
```

```
String str2 = "HELLO JAVA";  
String strLo = str2.toLowerCase();
```

```
System.out.println(strLo);  
// 출력 결과 : hello java
```

```
String str = "hello java";  
String newStr = str.substring(0, 1).toUpperCase();  
// subString으로 첫글자만 가져와서 toUpperCase()를 적용한 코드이다.
```


String trim(String)

문자열의 앞/뒤 공백을 제거하는 메소드

- 문자열中间的 공백은 제거하지 않음

```
String str = " Hellooo Java World ";  
String trimStr = str.trim();
```

```
System.out.println(trimStr);  
// 출력 결과 : Hellooo Java World
```

Boolean equals(String)

Boolean equalsIgnoreCase(String)

문자열이 같은지 비교하는 메소드

- 같으면 true 반환, 다르면 false 반환
- equalsIgnoreCase() → 대/소문자를 상관없이 문자열 비교

```
String str1 = "Hello Java";  
String str2 = "Hello Java";
```

```
System.out.println(str1.equals(str2));  
System.out.println(str1.equals("Hello Java"));  
System.out.println(str1.equalsIgnoreCase("hello java"));
```

int compareTo(String)

두 문자열을 비교함

- 비교대상에 문자열이 포함되어있을 경우
➔ 두 문자열의 **길이의 차이**를 반환함
- 비교대상과 전혀 다른 문자열인 경우
➔ 비교가 **불가능한 지점의 문자열 ASCII값**을 반환함

compareToIgnoreCase() : 대소문자 상관없이 비교

int compareTo(String)

```
String str = "abcd";
```

```
// 1) 비교대상에 문자열이 포함되어있을 경우
```

```
System.out.println(str.compareTo("abcd")); // 0 (같은 경우는 0을 리턴)
```

```
System.out.println(str.compareTo("ab")); // 2
```

```
System.out.println(str.compareTo("a")); // 3
```

```
System.out.println(str.compareTo("c")); // -2
```

```
System.out.println("").compareTo(str); // -4
```

```
// 2) 비교대상과 전혀 다른 문자열인 경우
```

```
System.out.println(str.compareTo("zefd")); // -25
```

```
System.out.println(str.compareTo("zEFd")); // -25
```

```
System.out.println(str.compareTo("ABCD")); // 32
```

Other methods

Boolean isEmpty()

- 빈 문자열인지 확인함 (빈 문자열일 때 true 반환)

```
String str = "Hello Java";  
System.out.println(str.isEmpty());  
// 출력 결과 : false
```

- String[] split(String)
 - 주어진 문자열을 parameter 내 문자열을 기준으로 분리함
 - String array의 형태로 반환됨

```
String str = "A,B,C,D,E,F";  
String[] strArray = str.split(",");  
  
for(String substr : strArray) {  
    System.out.println(substr);  
}
```

End of slide