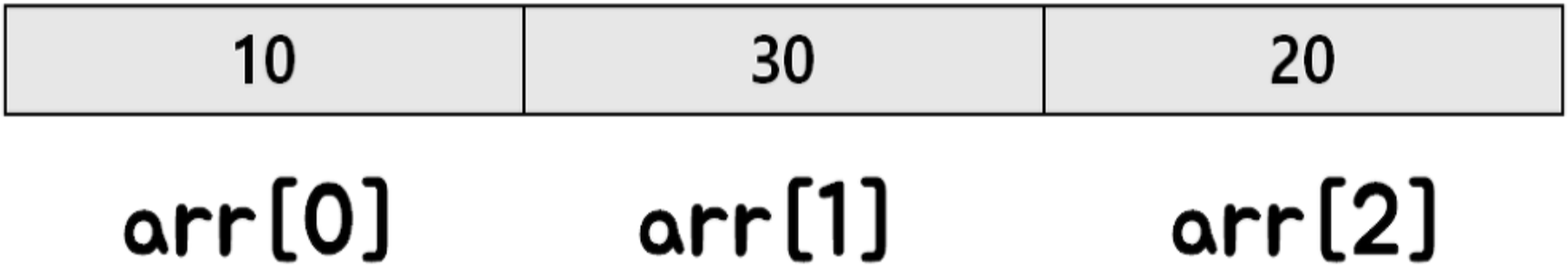
자바 배열과 문자열

목표

➔ 데이터를 묶어서 사용하는 방법

1. 데이터 묶는 방법

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



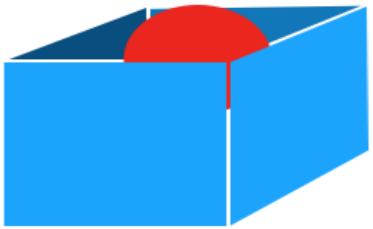
A[6] :

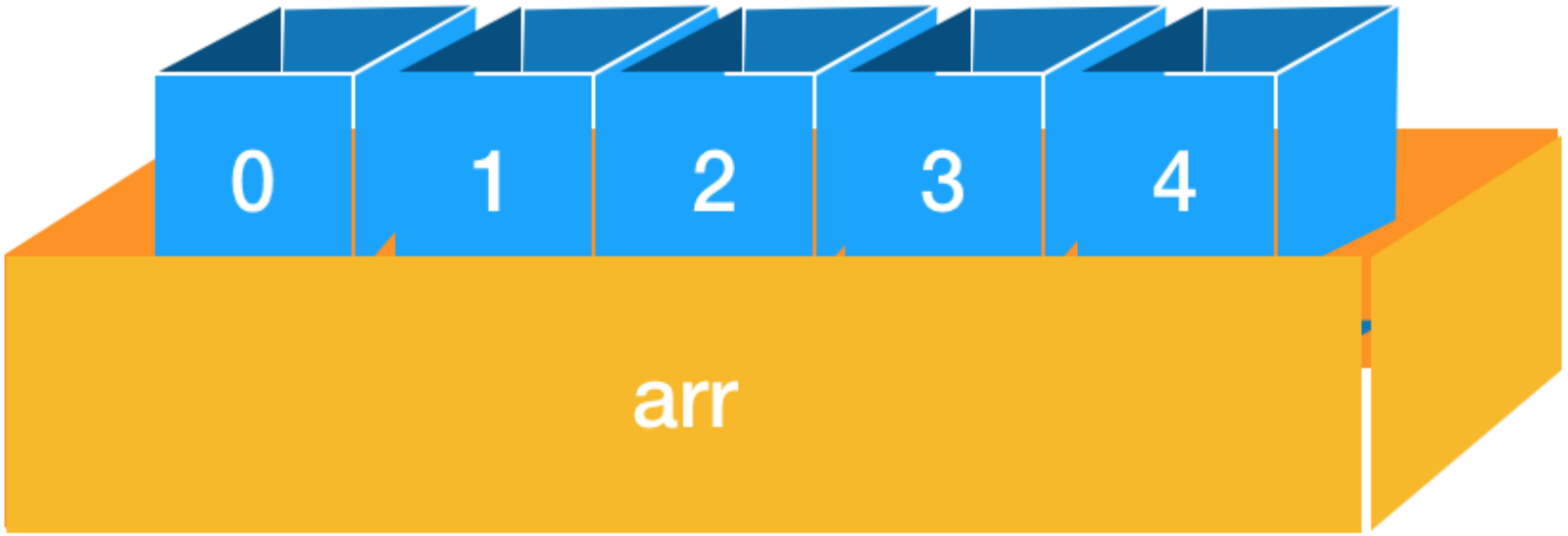
10	20	30	40	50	60

```
int A[6] = {10,20,30,40,50,60};
```

1. 데이터 묶는 방법

변수





1. 데이터 묶는 방법

배열이란?

프로그램의 특성상 많은 수의 변수가 필요하거나 데이터를 순서대로 저장해야 한다면 변수를 사용하는 것이 불편하므로 배열이라는 기억 공간을 이용한다.
배열과 변수의 공통점 : 데이터를 기억시킬 수 있는 공간
차이점 : 배열은 변수와 달리 번호가 붙은 저장 공간

아파트나 건물 입구에 있는 우편함과 같이 층과 호수를 나타내는 번호로 구별되는 동일한 크기의 연속된 공간과 같이
배열 역시 번호로 구별되는 동일한 데이터 형의 연속된 기억 공간

1. 데이터 묶는 방법

배열의 선언, 배열 이름, 배열 크기

```
{
    int korea[4]={15, 17, 27, 32};
    int china[4]={27, 16, 19, 11};
    ...

}
```

korea[0]	korea[1]	korea[2]	korea[3]
15	17	27	32

china[0]	china[1]	china[2]	china[3]
27	16	19	11

1. 배열선언

2. 배열이름

3. 배열크기

1. 데이터 묶는 방법

1차원 배열의 선언과 사용

1차원 배열 : korea[0], korea[1]과 같이 배열 요소를 구별하는데 있어서 한 개의 첨자가 사용되는 배열

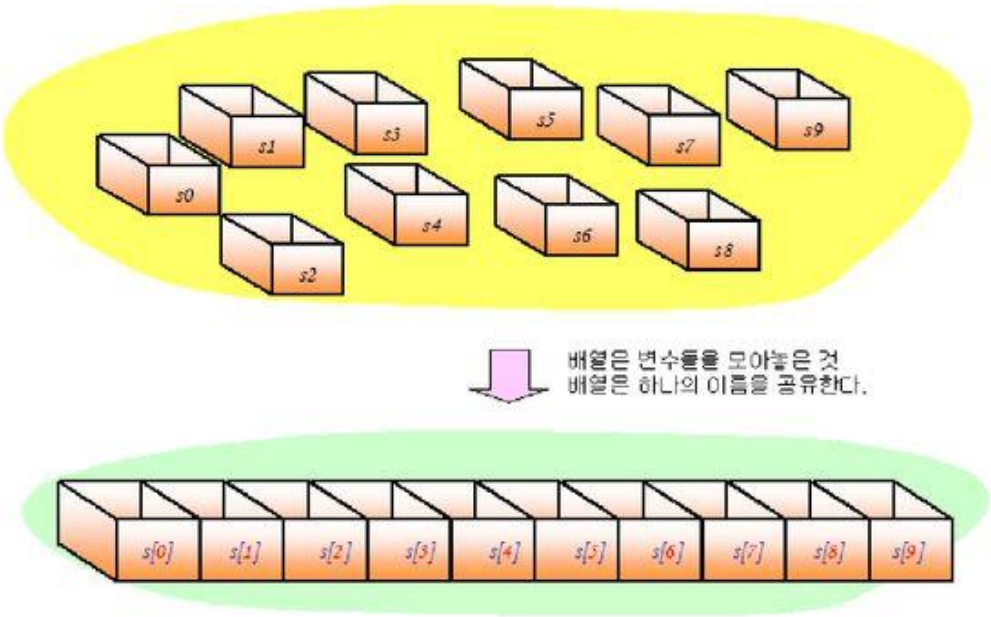
1 차원 배열 선언 방법	프로그램
배열 크기만 선언	`int korea[4];`
배열의 초기화	`int korea[4]={15, 17, 27, 32};`
배열 초기화에서 배열 크기 생략	`int korea[ ]={15, 17, 27, 32};`

배열의 크기와 자료의 개수	저장 상태
배열 크기 < 자료의 개수 `int korea[2]={15, 17, 27};`	컴파일 오류 발생(too many initializes)
배열 크기 > 자료의 개수 `int korea[4]={15, 17};`	나머지 요소들은 모두 0으로 초기화 됨. korea[0]에는 15, korea[1]에는 17이 초기화되지만 나머지 korea[2]와 korea[3]은 모두 0으로 초기화 됨.

1. 데이터 묶는 방법

배열의 개념

- 배열(array): 동일한 타입의 변수들의 모임



배열을 만드는 절차

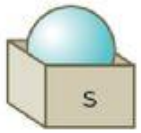
오류

```
int intArray [5];  // 크기 지정 안됨
```

① 먼저 배열 참조 변수부터 **선언**

```
int[]    s;
```

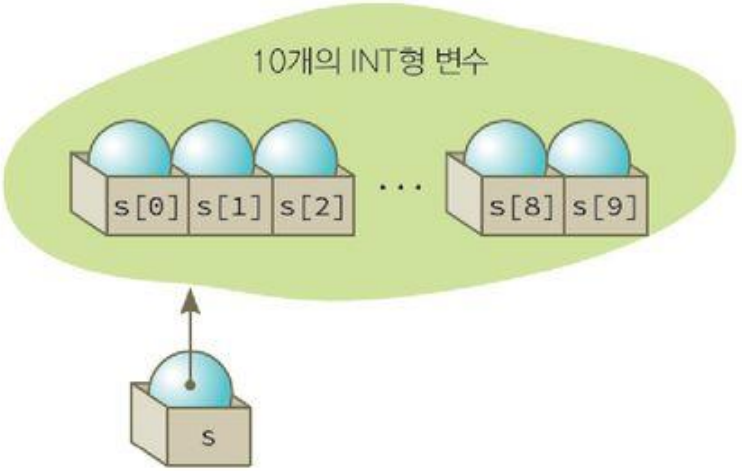
배열 요소의 자료형 — 배열의 이름



② 배열을 new 연산자를 사용하여서 **생성**

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

배열의 크기



배열의 인덱스

- 다음과 같은 배열을 가정하자.

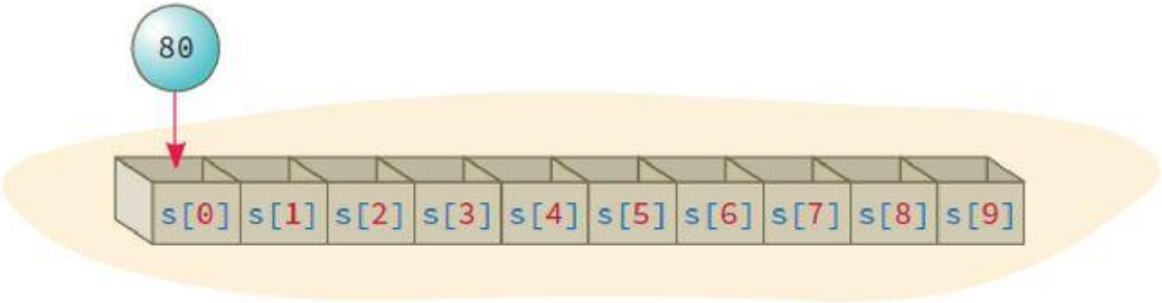
```
int[]   s = new int[10];
```

- 배열 요소에는 번호가 붙어 있는데 이것을 인덱스(index)라고 부른다.
- 첫 번째 요소의 번호는 0이고, 마지막 요소의 번호는 9가 된다.

인덱스를 통한 요소의 접근

- 배열은 변수들이 모인 것이니, 배열을 이루고 있는 배열 요소는 하나의 변수로 생각하면 된다.
- 배열의 첫 번째 요소에 80을 저장하려면 다음과 같이 한다.

```
s[0] = 80;
```



배열의 인덱스 범위

- 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다.

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


```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at ArrayTest4.main(ArrayTest4.java:16)
```

for-each 루프

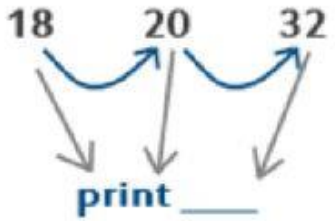
- 향상된 루프 구조

형식

```
for ( 변수 : 배열 ) {
    ...
}
```

변수에 배열의 요소가 차례대로 대입되면서 반복된다.

```
seq = [18, 20, 32]
for each x of seq
    print x
end
```



for-each 루프

for each 루프를 사용할 수 없는 경우
- 배열 원소의 값을 변경하는 경우
- 역순으로 배열을 처리하는 경우
- 일부 원소만을 처리하는 경우
- 하나의 반복문에서 두 개 이상의 배열을 처리하는 경우

→ 전통적인 for 문 사용!!

자바 배열

배열이란 ?

- 배열은 같은 타입의 여러 변수를 하나의 묶음으로 다르는 것

서로 다른 타입의 변수들로 구성된 배열은 만들 수 없다.

배열 선언과 생성

선언방법	선언 예
타입[] 변수이름; (권장)	int[] score; String[] name;
타입 변수이름[];	int score[]; String name[];

배열 선언

배열을 선언한 다음에는 배열을 생성해야 한다.

배열을 선언하는 것은
* 단지 생성된 배열을 다루기 위한 참조변수를 위한 공간이 만들어질 뿐이고,
* 배열을 생성해야만 비로소 값을 저장할 수 있는 공간이 만들어지는 것이다.

- 선언 방법

타입[] 변수이름; 배열을 선언 (배열을 다루기 위한 참조변수 선언)
변수이름 = new 타입[길이]; 배열을 생성 (실제 저장공간을 생성)

참조) 배열을 생성하기 위해서는 연산자 'new'와 함께 배열을 타입과 길이를 지정해 주어야 한다.

배열 선언

배열을 선언한 다음에는 배열을 생성해야 한다.

배열을 선언하는 것은
* 단지 생성된 배열을 다루기 위한 참조변수를 위한 공간이 만들어질 뿐이고,
* 배열을 생성해야만 비로소 값을 저장할 수 있는 공간이 만들어지는 것이다.

- 선언 방법

타입[] 변수이름; 배열을 선언 (배열을 다루기 위한 참조변수 선언)
변수이름 = new 타입[길이]; 배열을 생성 (실제 저장공간을 생성)

참조) 배열을 생성하기 위해서는 연산자 'new'와 함께 배열을 타입과 길이를 지정해 주어야 한다.

예) int[] score; int타입의 배열을 다루기 위한 참조변소 score선언
score - new int [5]; int타입의 값 5개를 저장할 수 있는 배열

[간략하게 한줄로 생성]
타입[] 변수이름 = new 타입 [길이]; 배열의 선언과 생성을 동시에.
int[] score = new int[5]; 길이가 5인 int배열

배열 선언

배열을 선언한 다음에는 배열을 생성해야 한다.

배열을 선언하는 것은
* 단지 생성된 배열을 다루기 위한 참조변수를 위한 공간이 만들어질 뿐이고,
* 배열을 생성해야만 비로소 값을 저장할 수 있는 공간이 만들어지는 것이다.

- 선언 방법

타입[] 변수이름; 배열을 선언 (배열을 다루기 위한 참조변수 선언)
변수이름 = new 타입[길이]; 배열을 생성 (실제 저장공간을 생성)

참조) 배열을 생성하기 위해서는 연산자 'new'와 함께 배열을 타입과 길이를 지정해 주어야 한다.

예) int[] score; int타입의 배열을 다루기 위한 참조변소 score선언
score - new int [5]; int타입의 값 5개를 저장할 수 있는 배열

[간략하게 한줄로 생성]
타입[] 변수이름 = new 타입 [길이]; 배열의 선언과 생성을 동시에.
int[] score = new int[5]; 길이가 5인 int배열

배열 선언

int[] score;
: 배열 참조 변수 score를 선언하였고 공간은 없다.

score = new int[5];
: 연산자 'new'에 의해서 메모리의 빈 공간 5개의 int형 데이터를 저장할 수 있는 공간이 마련되었다.

[배열의 길이와 인덱스]

생성된 배열의 각 저장공간을 '배열의 요소(element) ' 라고 하며, '배열이름[인덱스]'의 형식으로 배열의 요소에 접근한다.

인덱스(index)는 배열의 요소마다 붙여진 일련번호로 구별

인덱스(index)의 범위는 0부터 '배열길이-1까지'

배열 초기화

int[] score = new int[]{50, 60, 70, 80, 90 }; 배열의 생성과 초기화 동시에

- 괄호{}안에 저장할 값들을 쉼표로 구분해서 나열하면 되며 괄호[]안에 배열의 길이는 적지 않는다.

- 괄호{}안에 적힌 값의 개수에 의해 배열의 길이가 자동적으로 결정되기 때문이다.

new타입[] 생성

①

1) int[] score = **new int[]** {50, 60, 70, 80, 90};
2) int[] score = {50, 60, 70, 80, 90}; //new int[]를 생략할 수 있음

②

int[] score;
score = **new int[]** {50, 60, 70, 80, 90}; -- OK
score ={ 50, 60, 70, 80, 90}; -- 에러. new int[]를 생략할 수 없음

배열 출력

배열을 초기화할 때 for문을 사용하듯이,
배열에 저장된 값을 확인할 때도 다음과 같이 for문을 사용하면 된다.

```
int[] iArr = {100, 95, 80, 70, 60};

// 배열의 요소를 순서대로 하나씩 출력
for(int i=0; i< iArr.length; i++){
    System.out.println(iArr[i]);
}
```

배열의 필요성

- 학생이 10명이 있고 이들의 평균 성적을 계산한다고 가정하자.

방법 #1 : 개별 변수 사용

```
int s0;
int s1;
  …
int s9;
```

방법 #1 : 배열 사용

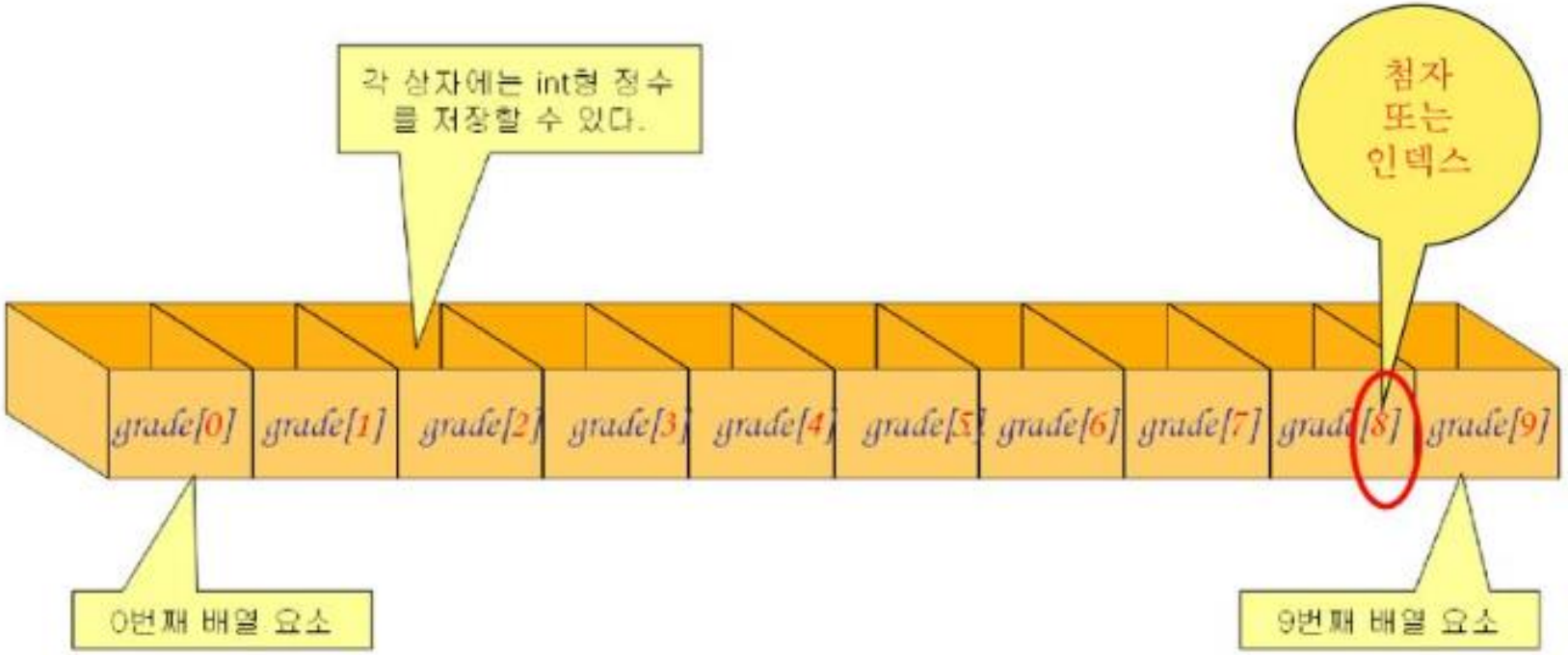
```
int[10];
```

배열이란?

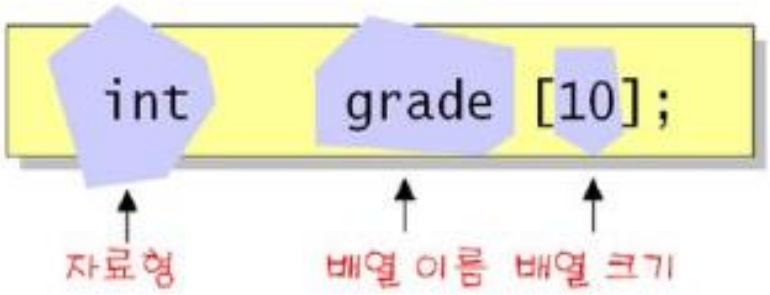
- 배열(array): 동일한 타입의 데이터가 여러 개 저장되어 있는 데이터 저장 장소
- 배열 안에 들어있는 각각의 데이터들은 정수로 되어 있는 번호(첨자)에 의하여 접근
- 배열을 이용하면 여러 개의 값을 하나의 이름으로 처리할 수 있다.

배열 원소와 인덱스

- *인덱스(index):* 배열 원소의 번호



배열의 선언



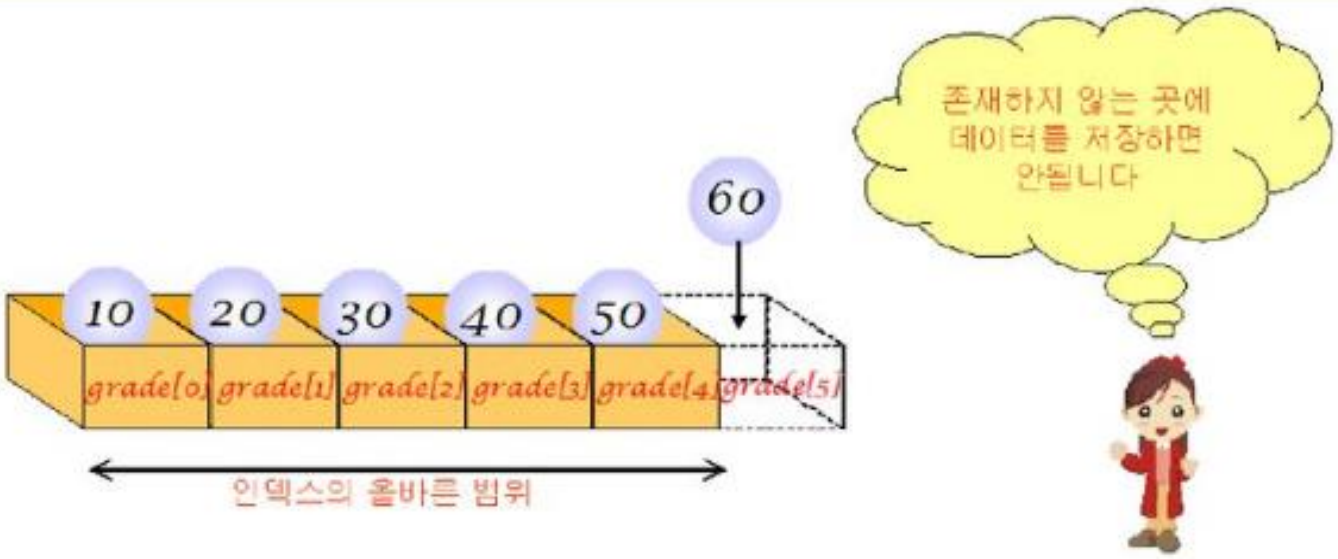
- 자료형: 배열 원소들이 int형라는 것을 의미
- 배열 이름: 배열을 사용할 때 사용하는 이름이 grade
- 배열 크기: 배열 원소의 개수가 10개
- 인덱스(배열 번호)는 항상 0부터 시작한다.

```
int  score[60];          // 60개의 int형 값을 가지는 배열 grade
float  cost[12];         // 12개의 float형 값을 가지는 배열 cost
char  name[50];          // 50개의 char형 값을 가지는 배열 name
char  src[10], dst[10];  // 2개의 문자형 배열을 동시에 선언
int  index, days[7];     // 일반 변수와 배열을 동시에 선언
```

잘못된 인덱스 문제

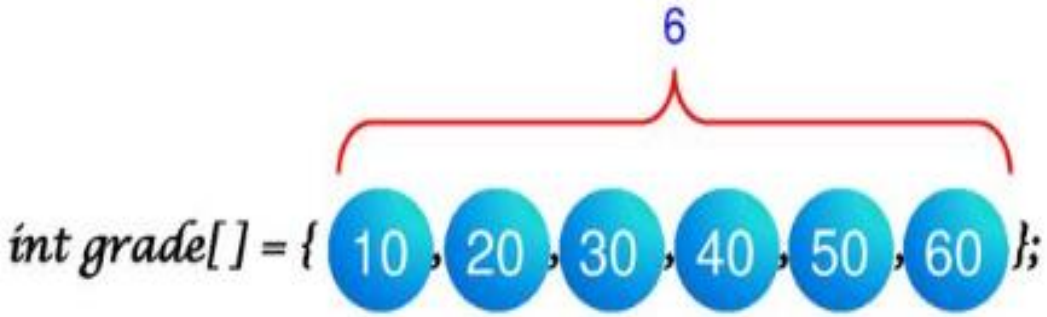
- 인덱스가 배열의 크기를 벗어나게 되면 프로그램에 치명적인 오류를 발생시킨다.
- C에서는 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다.

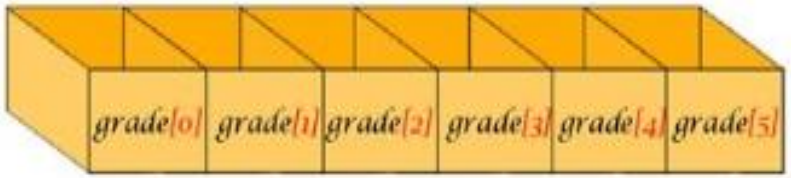
```
int   grade[5];
...
grade[5] = 60;       // 치명적인 오류!
```



배열의 초기화

- 배열의 크기가 주어지지 않으면 자동적으로 초기값의 개수만큼이 배열의 크기로 잡힌다.





자바 String 정리

String 메소드 정리

startWith: 문자열이 지정한 문자로 시작하는지 판단 같으면 true반환 아니면 false를 반환한다.(대소문자구별)

예)

```
String str = "apple";
boolean startsWith = str.startsWith("a");
System.out.println("startsWith: " + startsWith);
```
결과값:true

endWith:문자열 마지막에 지정한 문자가 있는지를 판단후 있으면 true, 없으면 false를 반환한다.(대소문자구별)

예)

```
String str = "test";
boolean endsWith = str.endsWith("t");
System.out.println("endsWith: " + endsWith);
```
결과값:true

equals:두개의 String에 값만을 비교해서 같으면 true, 다르면 false를 반환한다.(대소비교)

예)

```
String str1 = "java";
String str2 = "java";
boolean equals = str1.equals(str2);
System.out.println("equals: " + equals);
```
결과값:true

String 메소드 정리

indexOf:지정한 문자가 문자열에 몇번째에 있는지를 반환한다.

예)

```
String str = "abcdef";
int indexOf = str.indexOf("d");
System.out.println("indexOf: " + indexOf);
```
결과값:3

lastindexOf:문자열에 지정한 문자가 마지막몇번째에 있는 int를 반환한다.

예)

```
String str = "AdnroidApp";
int lastIndexOf = str.lastIndexOf("A");
System.out.println("lastIndexOf:" + lastIndexOf);
```
결과값:7

length:문자열의 길이를 반환한다.

예)

```
String str = "abcdef";
int length = str.length();
System.out.println("length: " + length);
```
결과값:6

String 메소드 정리

replace:문자열에 지정한 문자" "가 있으면 새로 지정한 문자" "로 바꿔서 출력한다.

예)

```
String str = "A*B*C*D";
String replace = str.replace("*", "-");
System.out.println("replace: " + replace);
```
결과값: A-B-C-D

replaceAll:정규표현식을 지정한 문자로 바꿔서 출력한다.

예)

```
String str = "AB CD";
String replaceAll = str.replaceAll("₩₩p{Space}", "*");
System.out.println("replaceAll: " + replaceAll);
```
결과값: AB*CD

split:지정한 문자로 문자열을 나눌수 있다.(배열로 반환)

예)

```
String str = "A:B:C:D:abcd";
String[] split = str.split(":");
System.out.println("split: " + split[1]);
```
결과값:B

String 메소드 정리

substring:문자열에 지정한 범위에 속하는 문자열을 반환한다.(시작범위에 값은 포함하고, 끝나는 범위에 값은 포함하지않는다.)

예)

```
String str = "ABCDEF";
String substring = str.substring(0, 2);
System.out.println("substring: " + substring);
```
결과값:AB

toLowerCase: 문자열에 대문자를 소문자로 변환한다.

예)

```
String str = "abcDEF";
String toLowerCase = str.toLowerCase();
System.out.println("toLowerCase: " + toLowerCase);
```
결과값:abcdef

toUpperCase:문자열에 소문자를 대문자로 변환한다.

예)

```
String str = "abcDEF";
String toUppercase = str.toUpperCase();
System.out.println("toUppercase: " + toUppercase);
```
결과값:ABCDEF

String 메소드 정리

trim:문자열에 공백을 없에준다.

예)

```
String s = "      java java java      ";
String v;
v = s.trim();
System.out.println("trim:" + v);
```
결과값:java java java

valueOf:지정한 개체의 원시 값을 반환

```
int i = 12345;
long l = 1L;
char c = '1';
System.out.println("valueOf: " + String.valueOf (i));
System.out.println("valueOf: " + String.valueOf (l));
System.out.println("valueOf: " + String.valueOf (c));
```
결과값:
valueOf: 12345
valueOf: 1
valueOf: 1

String 메소드 정리

contains:두개의 String을 비교해서 비교대상 String을 포함하고 있으면true, 다르면 false를 반환한다.

예)

```
String str1 = "abcd";
String str2 = "c";
boolean contains = str1.contains(str2);
System.out.println("contains: " + contains);
```
결과값:true

charAt:지정한 index번째에 문자를 반환한다.

예)

```
String str = "charAt";
char charAt = str.charAt(2);
System.out.println("charAt: " + charAt);
```
결과값:a

String 메소드 정리

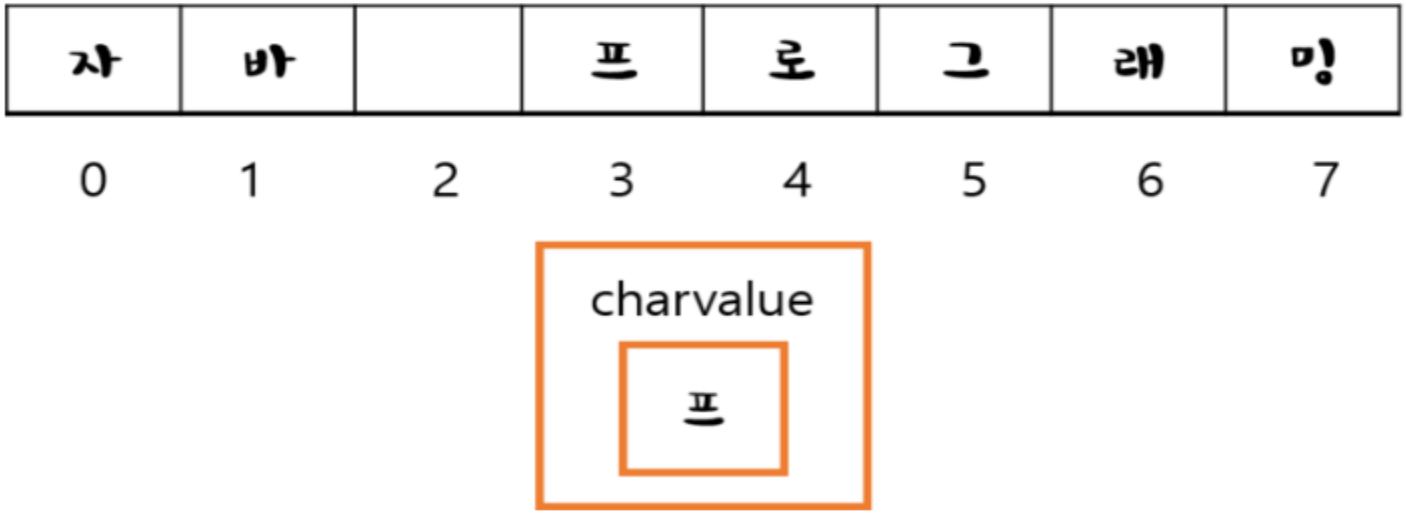
char	charAt(int index) 지정된 인덱스에 있는 문자를 반환한다.
int	compareTo(String anotherStirng) 사전적 순서로 문자열을 비교한다. 문자열 인스턴스가 작으면 음수, 같으면 0, 크면 양수가 반환 된다.
String	concat(String str) 주어진 문자열을 현재의 문자열 뒤에 붙인다.
boolean	equals(Object anObject) 주어진 객체와 현재의 문자열을 비교한다.
boolean	equalsIgnoreCase(String antherString) 대소문자를 무시하고 비교한다.
boolean	isEmpty() length()가 0 이면 true를 반환한다.
int	length() 현재 문자열의 길이를 반환 한다.
String	replace(char oldChar, char newChar) 주어진 문자열에서 oldChar를 newChar로 변경한, 새로운 문자열을 생성하여 반환한다.
String	substring(int beginIndex, int endIndex) 현재 문자열의 일부를 반환한다.
String	toLowerCase() 문자열의 모든 문자열을 소문자로 변경한다.
String	toUpperCase() 문자열의 모든 문자열을 소문자로 변경한다.

String 참조

문자 추출 (charAt())

charAt () 메소드는 매개값으로 주어진 인덱스의 문자를 리턴합니다. 여기서 인덱스란 () 에서 "문자열 길이 -1" 까지의 번호를 말합니다.

```
String subject = "자바 프로그래밍";
char charvalue = subject.charAt(3);
```

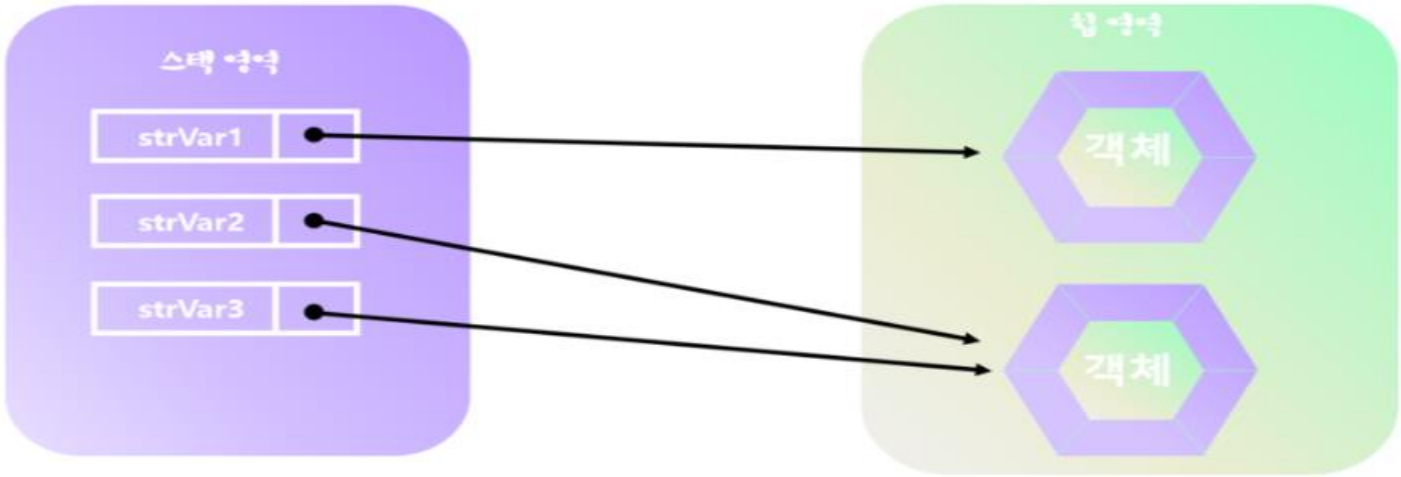


String 참조

문자열 비교 (equals())

기본 타입 변수의 값을 비교할 때에는 == 연산자를 사용합니다. 그러나 문자열을 비교할 때에는 == 연산자를 사용하면 원하지 않는 결과가 나올수도 있습니다.
이전에도 학습 했듯이, == 는 비교 대상의 번지 값을 비교하고, eqauls 메소드는 객체의 값을 비교하기 때문에 String 을 비교할 때는 eqauls 메소드를 사용합니다.

```
String strVar1 = new String("객체");
String strVar2 = "객체";
String strVar3 = "객체";
```

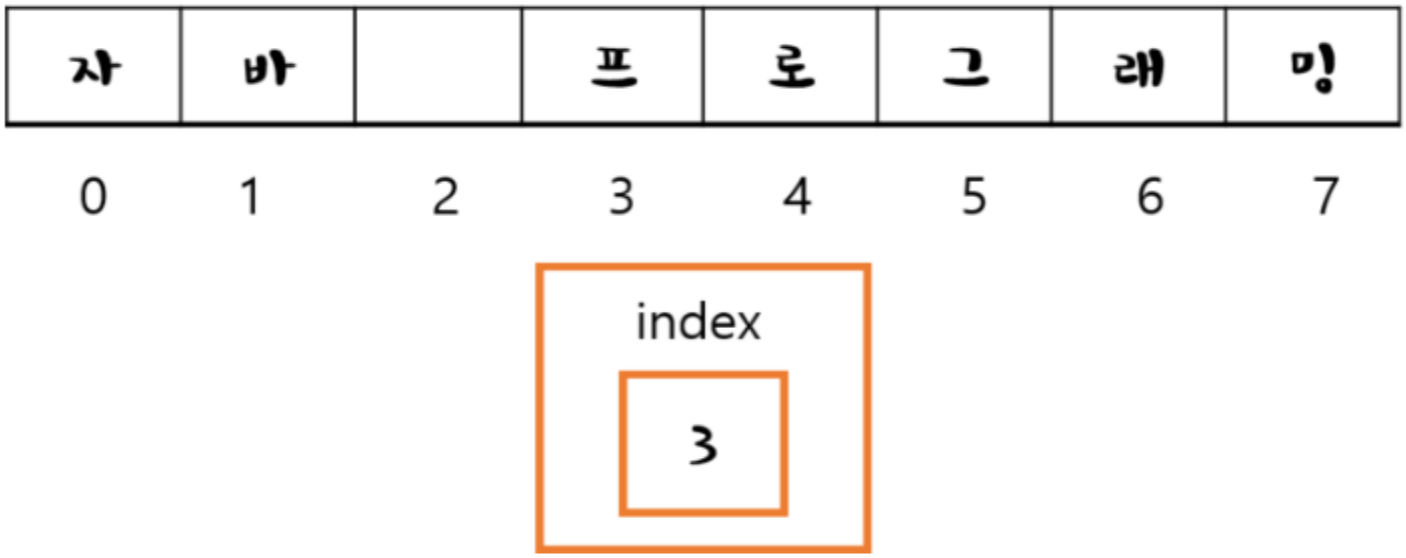


String 참조

문자열 찾기 (indexOf())

indexOf() 메소드는 매개값으로 주어진 문자열이 시작되는 인덱스를 리턴합니다. 만약 주어진 문자열이 포함되어 있지 않으면 -1 을 리턴합니다.

```
String subject = "자바 프로그래밍";
int index = subject.indexOf("프로그래밍")
```

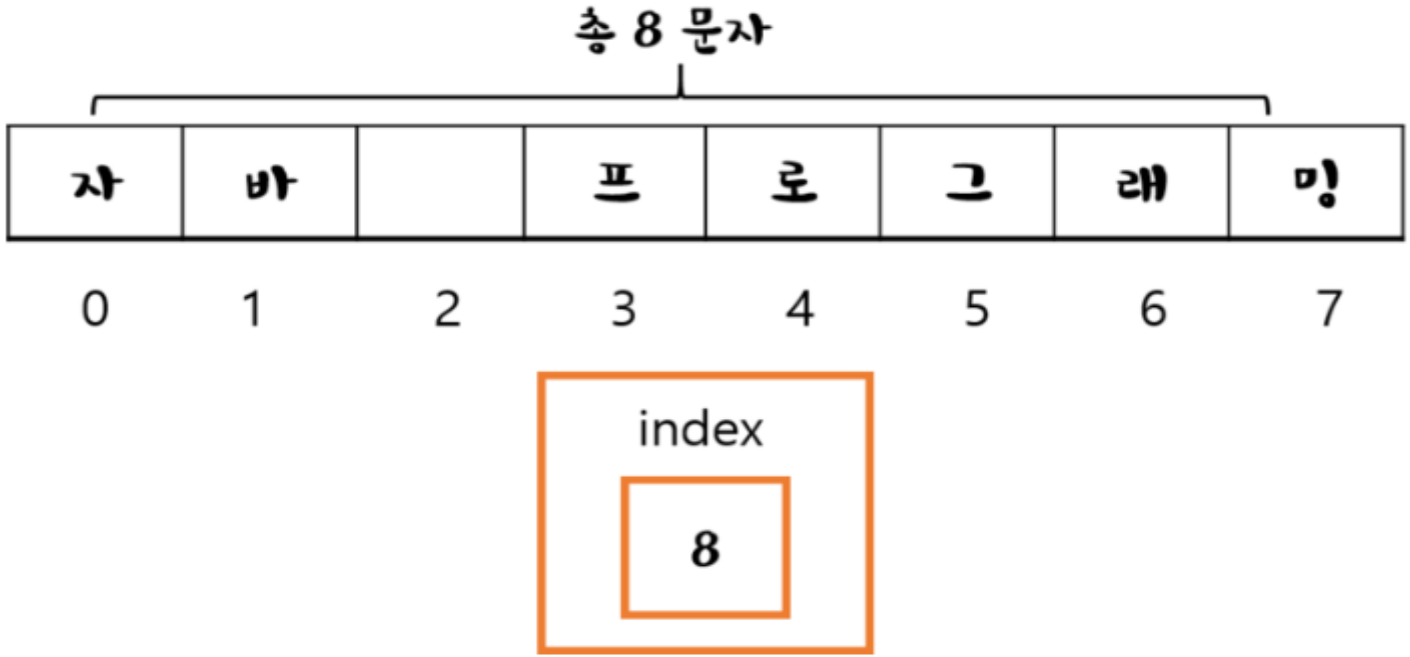


String 참조

문자열 길이 (length())

length() 메소드는 문자열의 길이 (문자의 수) 를 리턴합니다.

```
String subject = "자바 프로그래밍";
int index = subject.length();
```

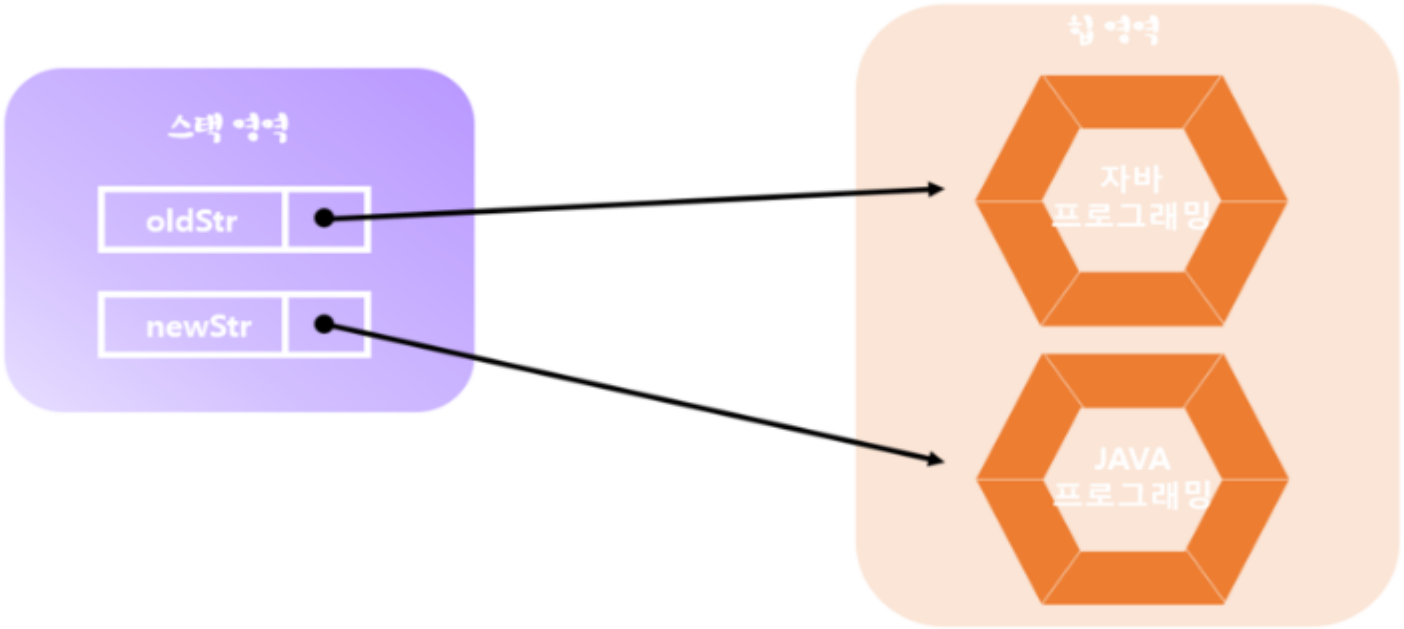


String 참조

문자열 대치 (replace())

replace () 메소드는 첫 번째 매개값인 문자열을 찾아 두 번째 매개값인 문자열로 대치한 새로운 문자열을 생성하고 리턴합니다.

```
String oldStr = "자바 프로그래밍";
String newStr = oldStr.replace("자바","JAVA")
```

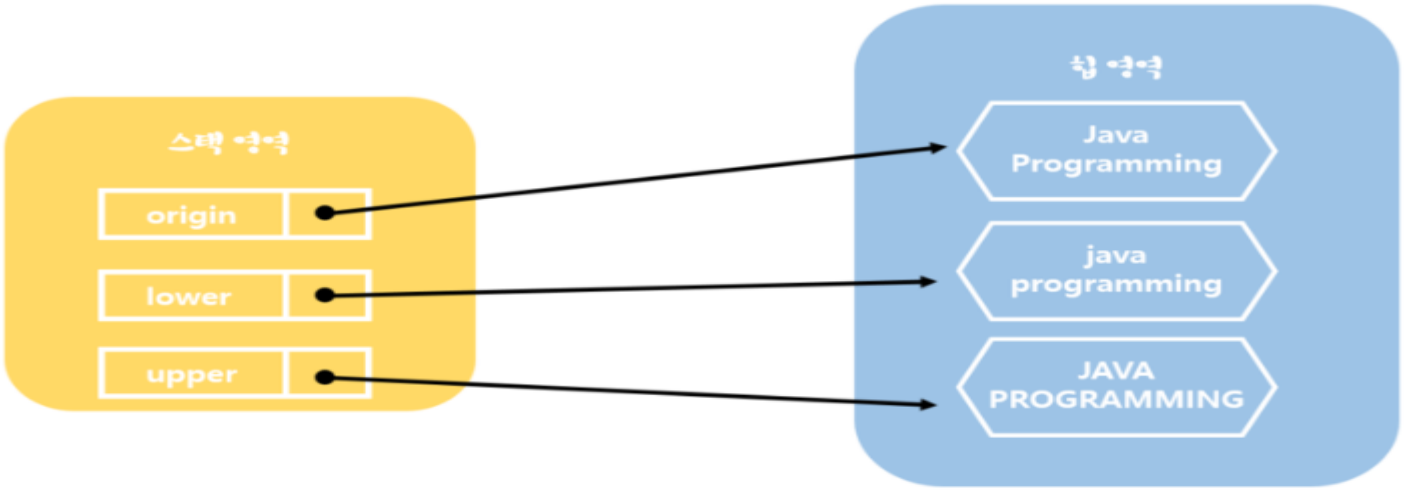


String 참조

알페벳 소 대문자 변경 (toLowerCase(), toUpperCase())

toLowerCase() 메소드는 문자열을 모두 소문자로 바꾼 새로운 문자열을 생생한 후 리턴합니다. 반대로 toUpperCase() 메소드는 문자열을 모두 대문자로 바꾼 새로운 문자열을 생성한 후 리턴합니다.

```
String origin = "Java Programming";
String lower = origin.toLowerCase();
String upper = origin.toUpperCase();
```

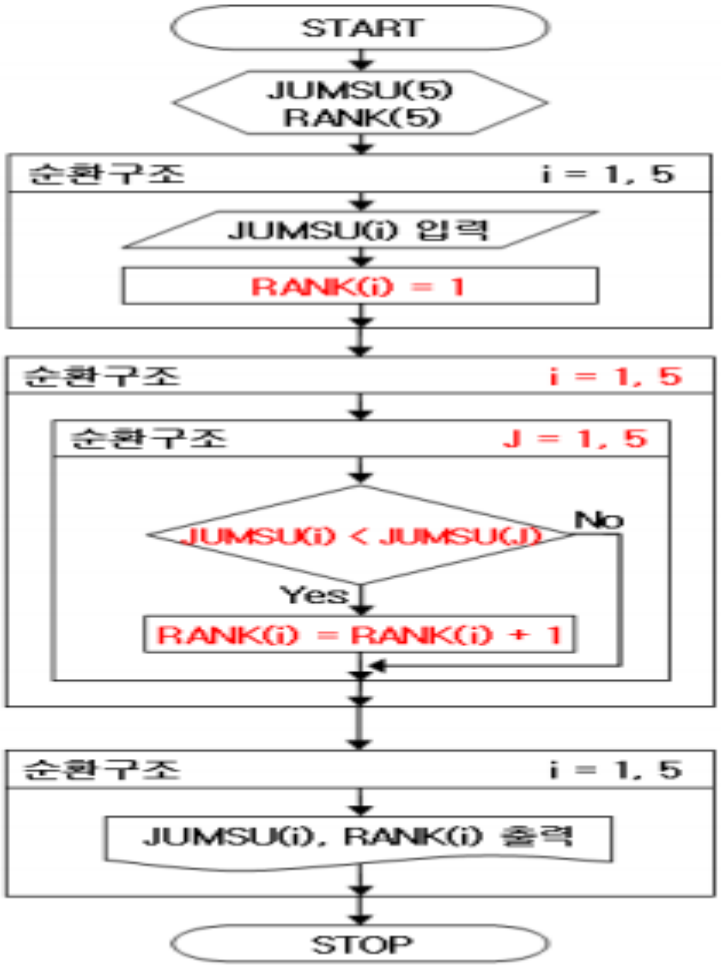


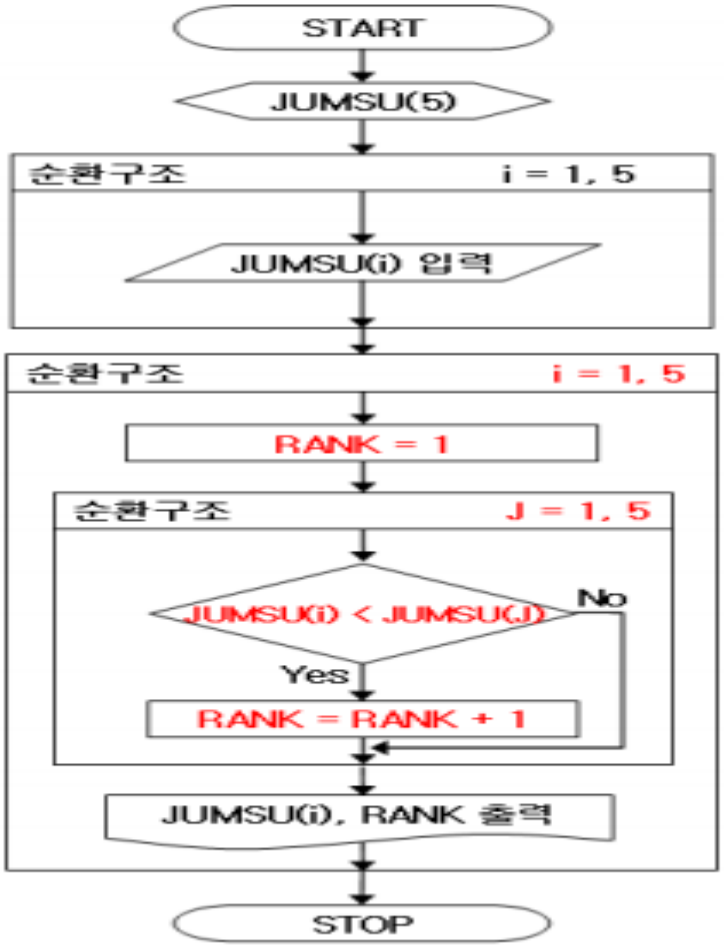
자바 배열 활용

석차 구하기

- 핵심: ① 한 사람의 점수에 대한 석차를 알려면 다른 사람들과 점수를 비교해 보면 된다.
  ② 다른 사람의 점수와 비교하다가 자신보다 점수가 높은 사람이 있으면 자신의 석차를 1씩 증가시키면 된다.
  ③ (Type A) 배열 저장 후 출력, (Type B) 바로 출력
- 사용되는 변수
  - JUMSU(5) : 점수를 저장할 배열
  - RANK(5) : 석차를 저장할 배열
  - RANK : 석차

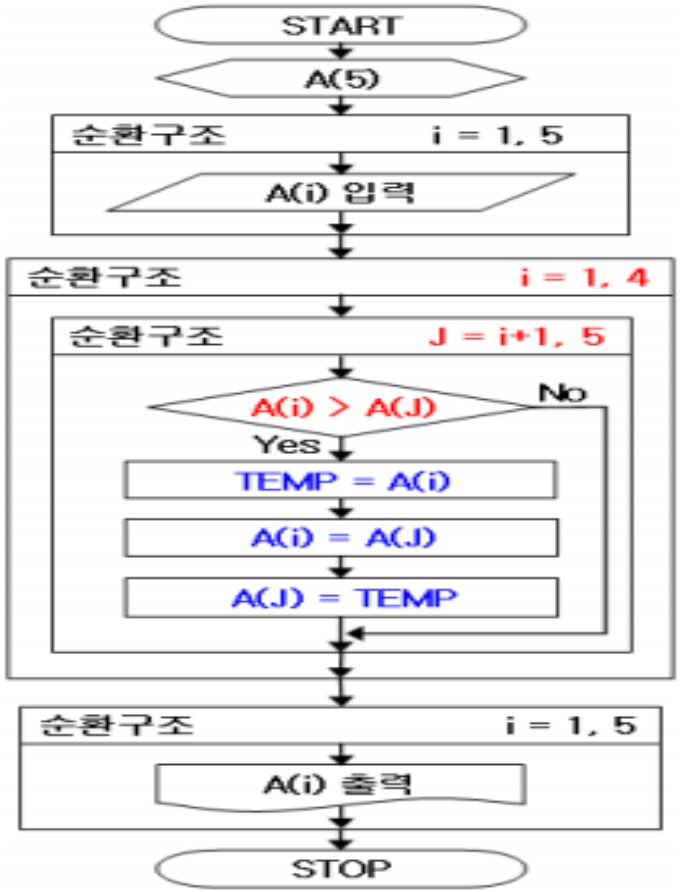
※ 5명의 학생 성적에 따른 석차 구하기





선택정렬 (Selection Sort)

- 핵심: ① n개의 입력 data 중 최소값을 찾아 첫 번째 위치에 놓고, 나머지 (n-1)개의 입력 data 중에서 최소값을 찾아 두 번째 위치에 놓는 방식을 반복하는 정렬 방식 (오름차순)
  ② 앞에서부터 정렬됨



※ 오름차순 정렬 조건 : **A(i)>A(J)**
내림차순 정렬 조건 : **A(i)<A(J)**

※ 변수 A와 변수 B를 맞바꾸는 과정
(추가로 변수가 한 개 더 필요함)

TEMP = A (A값을 임시변수에 넣고)
A = B (B값을 A에 넣고)
B = TEMP (임시변수 값을 B에 넣음)

버블정렬 (Bubble Sort)

- 핵심: ① 서로 이웃한 값들을 차례대로 비교하여 기준값(J)이 비교값(J+1)보다 크면 값을 교환
  ② 뒤에서부터 정렬됨

9	6	7	3	5

버블정렬 (오름차순)

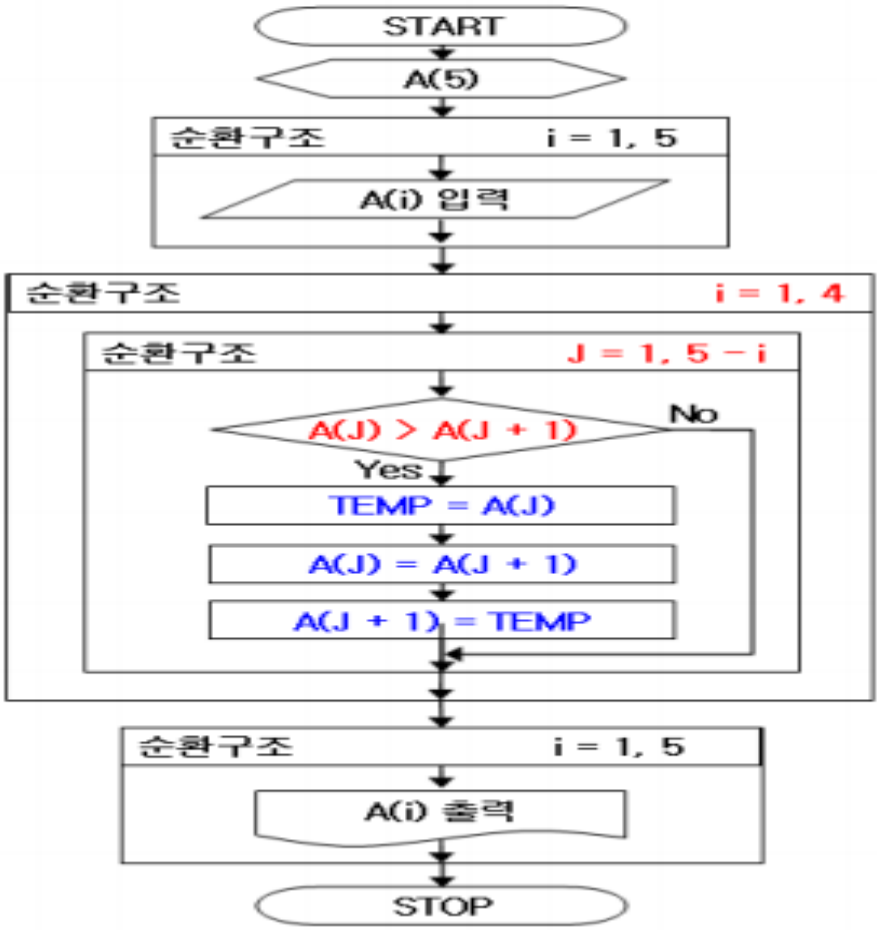
: 기준값(J) : 비교값(J+1) : 정렬완료

Pass 1 : 9 6 7 3 5 → 6 9 7 3 5 → 6 7 9 3 5 → 6 7 3 9 5 → 6 7 3 5 9

Pass 2 : 6 7 3 5 9 → 6 7 3 5 9 → 6 3 7 5 9 → 6 3 5 7 9

Pass 3 : 6 3 5 7 9 → 3 6 5 7 9 → 3 5 6 7 9

Pass 4 : 3 5 6 9 7 → 3 5 6 7 9



※ 오름차순 정렬 조건 : A(J)>A(J+1)
내림차순 정렬 조건 : A(J)<A(J+1)

윤년

윤년은 4년마다 돌아오는데 100년으로 나누어 떨어지는 해는
평년이다
하지만 400으로 나누어 떨어지는 해는 윤년이다

자바 코딩

```
(year%4==0 && year%100!=0)||(year%400==0)
```