SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Java

SAMSUNG SW ACADEMY FOR YOUTH

Programming Language

Array

- 배열
- 다차원 배열

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

배열(Array)

배열(Array)

- 생각해 봅시다.

배열(Array)

배열

- 같은 종류의 데이터를 저장하기 위한 자료구조
- 크기가 고정되어 있음. (한번 생성된 배열은 크기를 바꿀 수 없음)
- 배열을 객체로 취급(참조형)
- 배열의 요소를 참조하려면 배열이름과 색인(index)이라고 하는 음이 아닌 정수 값을 조합하여 사용.
- index 번호를 가지고 각 요소에 접근
- index 번호는 0부터 시작
- 배열이름.length를 통해 배열의 길이 조회 가능
- 배열의 길이는 임의로 변경 불가함
- 길이 변경 필요시 새로운 배열을 생성 후 내용을 옮김
- 배열의 각 요소는 해당 타입의 기본값으로 초기화(ex, int는 0, boolean은 false)

배열(Array)

배열의 선언

- 데이터타입[ ] 배열이름
- 데이터타입 배열이름[ ]

타입	배열 이름(변수명)	배열 선언문
int	intArray	int[] intArray;
char	charArray	char[] charArray;
boolean	boolArray	boolean[] boolArray;
String	strArray	String[] strArray;
float	floatArray	float[] floatArray;

배열(Array)

배열의 생성과 초기화

- 자료형[] 배열이름 = new 자료형[길이]; //배열 생성(자료형의 초기값으로 초기화)
- 자료형[] 배열이름 = new 자료형[] {값1, 값2, 값3, 값4}; //배열 생성 및 값 초기화
- 자료형[] 배열이름 = { 값1, 값2, 값3, 값4 }; //선언과 동시에 초기화

자료형	기본값	비고
boolean	false	
char	'₩u0000'	공백 문자(널 문자)
byte, short, int	0	
long	0L	
float	0.0f	
double	0.0	
참조형 변수	null	아무것도 참조하지 않음

배열(Array)

배열의 메모리 생성과정

```
int[] nums = new int[3];
nums[0] = 11;
nums[1] = 7;
nums[2] = 23;
```

```
int[] nums = new int[] { 27, 54, 83 };
```

배열(Array)

배열의 메모리 생성과정

```
String[] names = new String[3];
names[0] = "lime";
names[1] = "pear";
names[2] = "grape";
```

```
String[] names = new String[]{"fig", "plum", "date"};
```

배열(Array)

배열의 순회

- 반복문을 이용하여 배열의 요소를 순회할 수 있음.

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

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

배열(Array)

배열의 순회 (for-each)

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

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

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

==

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

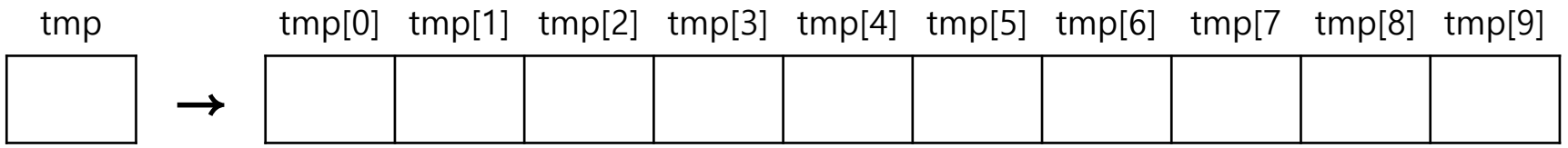
배열(Array)

- 배열의 출력
  - 반복문을 통해서 출력
  - Arrays.toString(배열) : 배열 안의 요소를 [값1, 값2, …] 형태로 출력

배열(Array)

배열의 복사

- 배열은 생성하면 길이를 변경할 수 없기 때문에 더 많은 저장공간이 필요하다면 큰 배열을 생성하고 이전 배열의 값을 복사 해야함.



- 새로운 배열 = Arrays.copyOf(복사하고_싶은_배열, 새로운_배열의 크기)
- System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)

배열(Array)

- 1차원 배열 실습: 최대, 최소, 합, 평균 구하기
- 다음 정수형의 배열이 주어졌을 때, 이 수들의 최대, 최소, 합, 평균을 구하시오.

```
int[] nums = {64, 53, 123, 23, 444, 98, 12};
```

```
최소값: 12
최댓값: 444
합계: 817
평균: 116.7142857143
```

배열(Array)

- 배열 실습 문제: 빈도수 구하기
  - 다음 배열에서 각 정수가 출현하는 횟수를 구하시오

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

int[] used = new int[10];

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

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

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다차원 배열

다차원 배열

다차원 배열이란?

- 2차원 이상의 배열을 의미
- 선언 시 []가 1개: 1차원 배열
- 선언 시 []가 2개 이상: 다차원 배열
- 배열의 배열
- 배열 객체의 참조값을 요소로 갖는 배열
- 2차원 배열은 배열 요소로 1차원 배열의 참조를 가지는 배열
- 3차원 배열은 배열 요소로 2차원 배열의 참조를 가지는 배열
- ...
- n차원 배열은 배열의 요소로 n-1차원 배열의 참조를 가지는 배열
- n차원 배열의 요소에 접근하기 위해서는 n개의 인덱스가 필요

2차원 배열

2차원 배열 선언

- 데이터타입[][] 배열이름
- 데이터타입 배열이름[][]
- 데이터타입[] 배열이름[]

2차원 배열 생성

- 배열이름 = new 데이터타입[1차원 배열갯수(행의 갯수)][1차원 배열의 크기(열의 갯수)];
- 배열이름 = new 데이터타입[1차원 배열갯수(행의 갯수)][];
- 배열이름 = new 데이터타입[][] {
  { 첫번째 1차원 배열의 초기값 },
  { 두번째 1차원 배열의 초기값 },
  ...
  }
- 데이터타입[][] 배열이름 = {
  { 첫번째 1차원 배열의 초기값 },
  { 두번째 1차원 배열의 초기값 },
  ...
  }

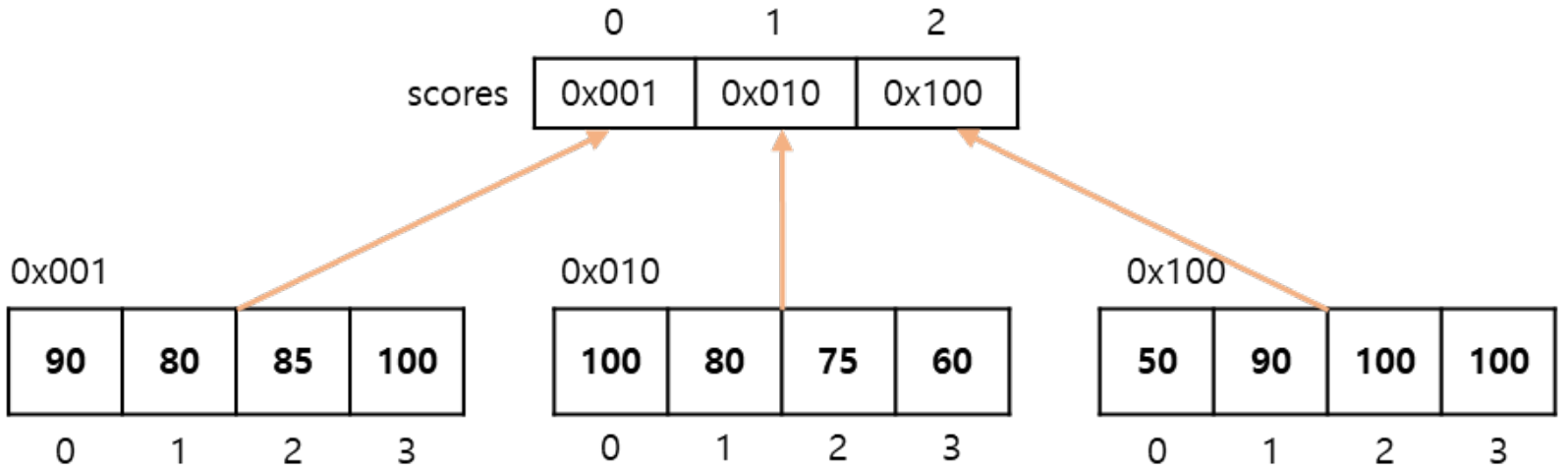
2차원 배열

2차원 배열

```
int[][] scores = {{90,80,85,100},
                  {100,80,75,60},
                  {50,90,100,100}};
```

scores

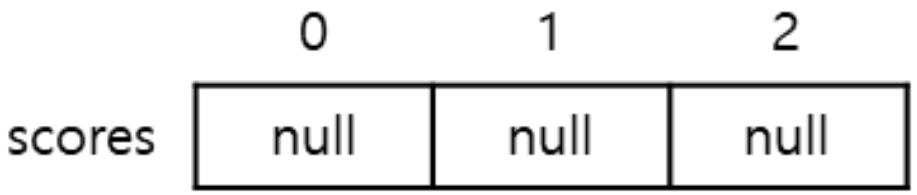
	0	1	2	3
0	90	80	85	100
1	100	80	75	60
2	50	90	100	100



2차원 배열

2차원 배열

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



2차원 배열

2차원 배열 메모리

```
int a = 10;




int [] arr = new int [4];




int [][] arr2 = new int[2][];
arr2[0] = new int [3];
arr2[1] = new int [10];
arr2[1][1] = 100;
```

2차원 배열

- 2차원 배열 실습 문제
- 모든 2차원 배열의 원소 중
  3의 배수의 개수와 그들의 합을 출력

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

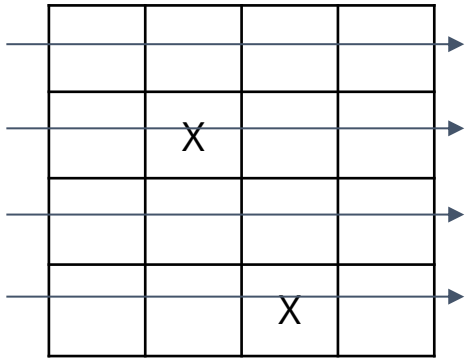
```
public static void main(String[] args) {

    int[][] grid = {
                        {2, 3, 1, 4, 7},
                        {8, 13, 3, 33, 1},
                        {7, 4, 5, 80, 12},
                        {17, 9, 11, 5, 4},
                        {4, 5, 91, 27, 7}
    };
    int count = 0;
    int sum = 0;
    for(int [] row : grid) {
        for(int num : row) {
            if(num % 3 == 0) {
                count++;
                sum+=num;
            }
        }
    }
    System.out.printf("개수: %d, 총합: %d%n", count, sum);
}
```

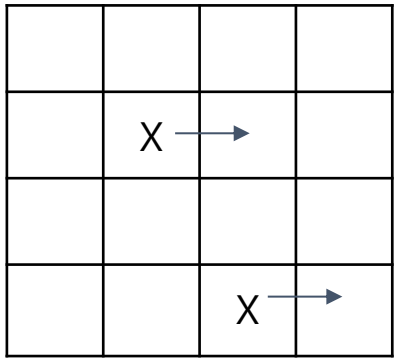
2차원 배열

Confidential

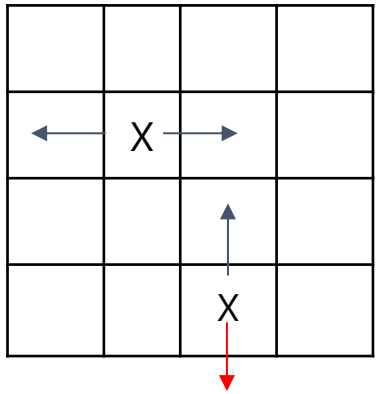
2차원 배열 탐색



전체 중 X를 만나면



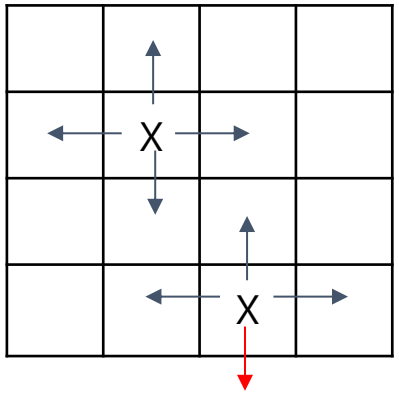
X가 움직이면서



X 주변 탐색

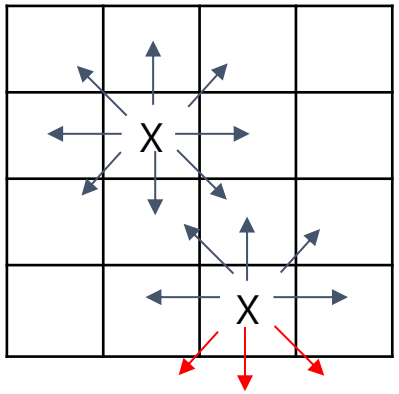
좌/우

상/하



X 주변 탐색

4방



X 주변 탐색

8방

3차원 배열

3차원 배열 선언

- 데이터타입[][][] 배열이름

3차원 배열 생성

- 배열이름 = new 데이터타입[크기a][크기b][크기c];
  크기 a: 3차원 배열의 크기
  크기 b: 2차원 배열의 크기
  크기 c: 1차원 배열의 크기

- 배열이름 = new 데이터타입[크기a][][];
- 배열이름 = new 데이터타입[][][] { 3차원 배열의 초기값 }

3차원 배열

- 다음 2x2 픽셀 이미지를 배열로 저장한다면?


```java
// 2x2 이미지의 RGB 값 저장을 위한 3차원 배열 생성
int[][][] image = new int[2][2][3];

// 각 픽셀의 RGB 값 할당
// 픽셀 (0, 0)
image[0][0][0] = 243; // Red
image[0][0][1] = 104; // Green
image[0][0][2] = 224; // Blue

// 픽셀 (0, 1)
image[0][1][0] = 255; // Red
image[0][1][1] = 159; // Green
image[0][1][2] = 67; // Blue

// 픽셀 (1, 0)
image[1][0][0] = 0; // Red
image[1][0][1] = 210; // Green
image[1][0][2] = 211; // Blue

// 픽셀 (1, 1)
image[1][1][0] = 84; // Red
image[1][1][1] = 160; // Green
image[1][1][2] = 255; // Blue

// 이미지의 RGB 값 출력
for (int i = 0; i < image.length; i++) {
    for (int j = 0; j < image[i].length; j++) {
    System.out.println("Pixel (" + i + ", " + j + ") - " +
        "R: " + image[i][j][0] + ", " +
        "G: " + image[i][j][1] + ", " +
        "B: " + image[i][j][2]);
    }
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미