

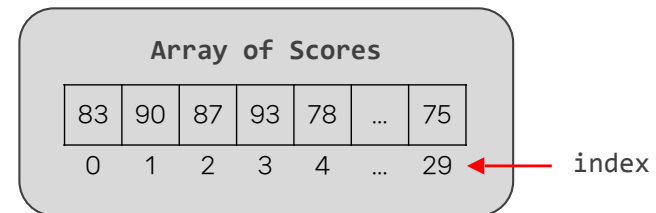
03_2 Arrays

Object-Oriented Programming

이 강의에서는 array에 대해 강의하겠습니다.

Definition of Array

- A data structure that lists **data of the same type** in a contiguous array, with **each element** having an **index**



```
score[0] = 83;  
score[1] = 90;  
...  
score[29] = 75;  
score[5] = score[2] + 3;
```

Array란 같은 type의 데이터 element들이 모인 자료구조로서,
메모리의 연속된 공간을 차지하고 있고,
각 element를 index를 이용하여 access할 수 있습니다.
그림을 보면, scores array에는 30개의 정수들이 모여 있는데
각 element는 0부터 29까지의 index를 이용하여 access 가능합니다.
오른쪽처럼 각 element를 읽어오거나 element에 value를 쓸 때에는
bracket [] 안에 index를 넣어 개별적인 element를 참조할 수 있습니다.

Declaration of Array

```
int[] iarray1;  
double[] darray1;  
String[] stArray1;
```

```
int iarray2[];  
double darray2[];  
String stArray2[];
```

- People generally prefer the former one.
- At this stage, memory has not yet been allocated for each array element.

array를 선언, 즉, declaration 하는 크게 두 가지 방법이 있습니다.
Type과 빈 bracket을 쓰고, 뒤에 array 변수를 쓰는 첫번째 방법이 있고
Type과 변수이름을 쓰고 난 뒤에 빈 bracket을 쓰는 두번째 방법이 있습니다.
어느쪽을 쓰든지 상관이 없으나
이 중에 첫번째 방법이 더 많이 쓰이고 있습니다.
또 한가지 언급할 것은, 이 상태로는 각 element를 위한 memory가
아직 allocate, 즉, 할당되지 않았다는 것입니다.

Creating Array Objects

- Creating an array from a list of values
 - ex) `String[] capitals = {"Seoul", "Tokyo", "Beijing", "London"}`
 - ex) `int[] scores = {1, 2, 5, 13, -39}`
- Creating an array using the new operator
 - ex1) `int[] a1 = new int[5];`
`a1[0] = 35;`
`a1[3] = 70;`
 - ex2) `String[] str = new String[3];`
`str[0] = "Seoul";`
`str[1] = "Tokyo";`
`...`

4

페이지 4

Array를 declare하면서 동시에 데이터를 initialize하는 방법이 있습니다.
Declaration의 variable 뒤에 이퀄을 쓰고 브레이스 { 를 열고
element를 하나씩 나열한 후 브레이스 }를 닫는 것입니다.
이런식으로 해서 String이나 int, double array등을 initialize할 수 있습니다.
한편, explicit하게 new operator를 사용하여
memory를 할당하는 방법을 사용할 수 있습니다.
example 1을 보면 a1 array에는 int type 5개의 memory가 할당되었으며,
a1[0], a1[3] 과 같은 형태로 array의 값을 초기화 할 수 있습니다.
example 2에서는 str에 3개의 String을 위한 reference가 초기화 되었고
str[0]는 "Seoul", str[1]은 "Tokyo" 와 같은 식으로 String에 대한 reference를
가질 수 있게 됩니다.

Default Initialization Values

- Each array element is automatically initialized as the **default initialization value** of that type.
- ex) `int[] a[3];` // initialized as `a[0] = a[1] = a[2] = 0`
- Default initialization values
 - byte, short, int, long, float, double: 0 (or 0.0f, 0.0)
 - char: `'\u0000'` (null character)
 - boolean: false
 - Reference Types (including String): null

5

페이지 5

각 array element들은 따로 initialize하지 않더라도 default initialization value로 초기화 됩니다.
예를 들면, 사이즈가 3인 `int a[3]` array가 메모리에 잡히는 순간에
`a[0]`, `a[1]`, `a[2]`의 value는 모두 0으로 초기화 됩니다.
default initialization value들을 type별로 알아보면,
byte, short, int와 같은 integer와 float, double의 초기값은 모두 0 (또는 0.0) 이 됩니다.
char type은 null character가 됩니다.
boolean은 false가 되고,
String을 비롯한 reference type은 null로 자동 초기화 됩니다.

Length of an Array

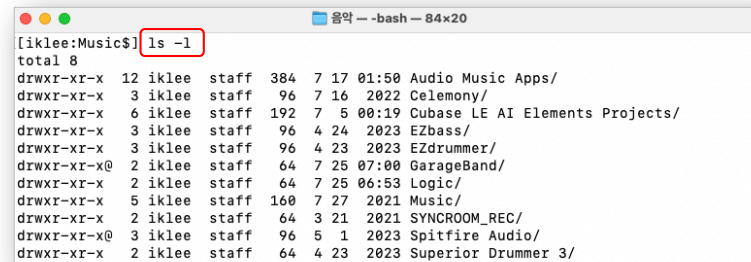
- 'length' field of an array
- 'length' is a read-only field
- ex) `int[] intArray = {10, 20, 30};`
`int size = intArray.length; // 3`

array에는 length 필드가 있는데 array의 size value를 담고 있습니다.
이 value는 read only입니다.
당연하게도 length의 value는 바꿀 수 없습니다.
{ 10, 20, 30} 으로 초기화 된 int array 의 length는 3 입니다.
String에는 그 size를 return하는 length() method가 있었습니다.
그러나 array의 경우에는 length가 method가 아니라 variable 입니다.

Command Line Arguments (1/4)

- An array of strings passed as a parameter to the main method.

```
public class CommandLineArguments {  
    public static void main(String[] args) {  
        System.out.println("args[0] = " + args[0]);  
        System.out.println("args[1] = " + args[1]);  
    }  
}
```



A terminal window titled '음악 -- -bash -- 84x20' showing the execution of the command 'ls -l'. The output lists files and directories with their permissions, owner, group, size, and modification date. The command 'ls -l' is highlighted with a red box.

```
[iklee:Music$] ls -l  
total 8  
drwxr-xr-x 12 iklee staff 384 7 17 01:50 Audio Music Apps/  
drwxr-xr-x 3 iklee staff 96 7 16 2022 Celemony/  
drwxr-xr-x 6 iklee staff 192 7 5 00:19 Cubase LE AI Elements Projects/  
drwxr-xr-x 3 iklee staff 96 4 24 2023 EZbass/  
drwxr-xr-x 3 iklee staff 96 4 23 2023 EZdrummer/  
drwxr-xr-x@ 2 iklee staff 64 7 25 07:00 GarageBand/  
drwxr-xr-x 2 iklee staff 64 7 25 06:53 Logic/  
drwxr-xr-x 5 iklee staff 160 7 27 2021 Music/  
drwxr-xr-x 2 iklee staff 64 3 21 2021 SYNCROOM_REC/  
drwxr-xr-x@ 3 iklee staff 96 5 1 2023 Spitfire Audio/  
drwxr-xr-x 2 iklee staff 64 4 23 2023 Superior Drummer 3/
```

그동안 별 생각없이 사용했던 main method의 header에도 array parameter가 존재합니다.
args는 String의 array로서 프로그램 이름 뒤에 나오는 command line argument들을 가지고 있습니다.
예를들면, args[0]는 프로그램 이름 바로 뒤의 arguments, args[1]은 그 뒤 arguments와 같습니다.
예를들어, 터미널에서 'ls -l'이라는 명령을 실행했을 때
ls는 프로그램 이름, -l은 args[0]가 되는 것입니다.

Command Line Arguments (2/4)

- An array of strings passed as a parameter to the main method.

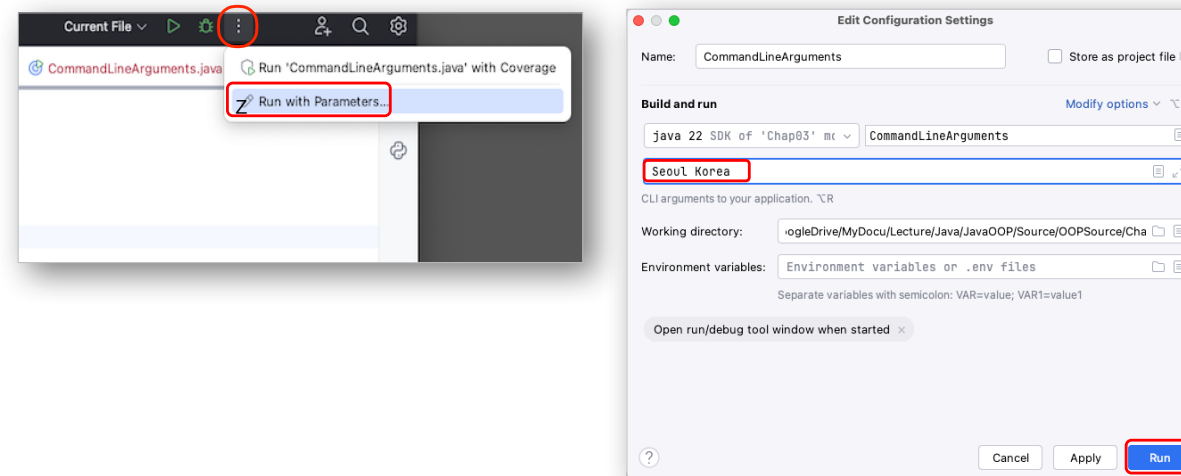
```
public class CommandLineArguments {  
    public static void main(String[] args) {  
        System.out.println("args[0] = " + args[0]);  
        System.out.println("args[1] = " + args[1]);  
    }  
}
```

- If we try to run the program, exception occurs:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
at CommandLineArguments.main(CommandLineArguments.java:3)

8

Command Line Arguments (2/3)

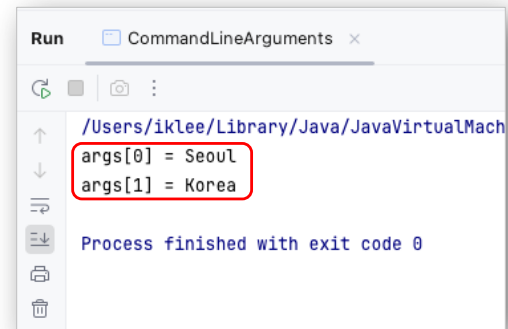


9

페이지 9

IntelliJ IDEA에서 command line arguments를 사용하는 방법은 play button 옆에 있는 점 세개의 메뉴 버튼을 눌러 "run with parameters"를 선택합니다. 여기서 나온 메뉴에서 빈칸에 command line arguments를 넣어 주는 것입니다. 이 예에서는 'Seoul' 과 'Korea' 가 각각 args[0]와 args[1] 으로 주어졌습니다. 그리고 'run' button을 누릅니다.

Command Line Arguments (3/3)

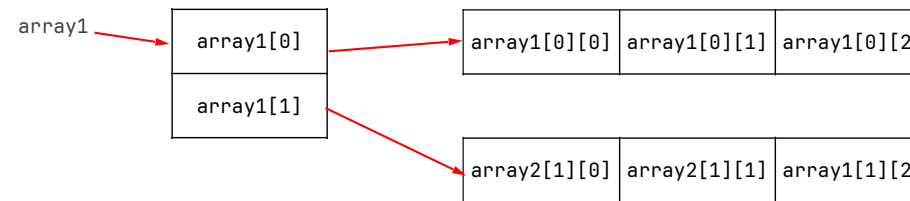


The screenshot shows the 'Run' console of an IDE. The title bar says 'Run' and 'CommandLineArguments'. The console output shows the file path `/Users/ikLee/Library/Java/JavaVirtualMach`, followed by `args[0] = Seoul` and `args[1] = Korea`, which are highlighted with a red box. Below this, it says 'Process finished with exit code 0'.

```
Run CommandLineArguments x
/Users/ikLee/Library/Java/JavaVirtualMach
args[0] = Seoul
args[1] = Korea
Process finished with exit code 0
```

Multidimensional Array (1/2)

```
public class MultiDArray1 {  
    public static void main(String[] argv) {  
        int[][] array1 = new int[2][3];  
  
        for (int i = 0; i < array1.length; i++) // fill the array1  
            for (int j = 0; j < array1[i].length; j++) {  
                array1[i][j] = i + j;  
            }  
    }  
}
```



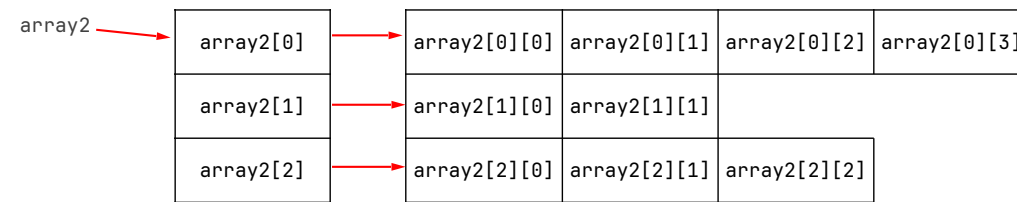
11

페이지 11

이 slide는 다차원 array를 정의하는 예를 보여주고 있습니다.
array1은 2개의 row와 3개의 column을 가진 2 x 3의 다차원 array입니다.
array의 element값들을 채우기 위해 nested for문이 사용되었습니다.
각 element array1[i][j]는 i + j 값으로 채워졌습니다.
그림을 보면 array1 reference variable은 첫번째 row의 reference array1[0]를 가리키고 있습니다.
array1[0]와 array1[1]은 첫번째 dimension을 나타내는 reference들인데 그들이 연속된 메모리 공간에 존재함을 눈여겨보아야 합니다.
array1[0]는 array1[0][0]를 가리키고 있고, 여기서부터 같은 row에 속하는 element 3개가 연속되어 있습니다.
또, array2[1]이 가리키는 array2[1][0], array2[1][1], array2[1][2]의 세 element들도 연속된 메모리 공간에 있습니다.
하지만 array1[0][2]와 array2[1][0]가 반드시 연속된 공간에 자리할 필요는 없습니다.

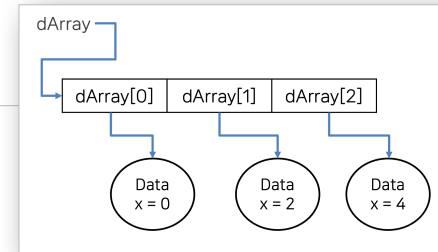
Multidimensional Array (2/2)

```
int[][] array2 = {{1, 2, 3, 4}, {5, 6}, {7, 8, 9}};
```



Array of Objects

```
class Data {  
    int x;  
}  
  
public class ArrayOfObjects {  
    public static void main(String[] args) {  
        Data[] dArray = new Data[3]; // array of class object 'Data'  
        for (int i = 0; i < dArray.length; i++) {  
            dArray[i] = new Data(); // each object dArray[i] should be created  
            dArray[i].x = i * 2;    // 0, 2, 4  
        }  
        for (int i = 0; i < dArray.length; i++) {  
            System.out.print(dArray[i].x + " "); // 0 2 4  
        }  
    }  
}
```



class object들의 array를 고려하기 위해
int x 라는 하나의 instance variable을 가지고 있는 간단한 Data class를 먼저 정의하였습니다.
이 Data class의 array를 만들 경우, 먼저 Data object의 reference를 담을 array를 create합니다.
그 후에 reference array의 element를 하나씩 돌며 Data object를 하나씩 개별적으로 생성합니다.
class object를 element로 가지는 array의 경우, 이 두가지의 step이 필요합니다.

Enumeration Type (1/2)

- A type that stores one of the enumeration constants

```
enum Day {  
    SUNDAY,  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY  
}
```

```
public class DayEnumDemo {  
    public static void main(String[] args) {  
        Day today = Day.WEDNESDAY;  
        System.out.println("Today is: " + today);  
        // Today is: WEDNESDAY  
        switch (today) {  
            case SATURDAY:  
            case SUNDAY:  
                System.out.println(today + " is a weekend.");  
                break;  
            default:  
                System.out.println(today + " is a weekday.");  
                break;  
        } // WEDNESDAY is a weekday.  
    }  
}
```

14

페이지 18

enumeration type은 enumeration constant 중 하나의 값을 가질 수 있는 type을 말합니다.
먼저 Day라는 enumeration type이 정의되었는데
SUNDAY부터 SATURDAY까지 각 요일을 나타내는 constant들이 이 type에 정의되었습니다.
프로그램에서 today variable이 enumeration인 Day type으로 declare되면서
Day.WEDNESDAY로 initialize되었습니다.
today variable을 print하면 WEDNESDAY가 프린트됩니다.
그 아래 switch 문에서는 today가 SATURDAY나 SUNDAY이면
weekend 라고 프린트되고
토, 일요일이 아닌 나머지 요일이면 weekday라고 프린트됩니다.

Enumeration Type (2/2)

```
System.out.println("All days of the week:");
for (Day day : Day.values()) {
    String name = day.name();
    int order = day.ordinal();
    System.out.println(order + ") " + day + " " + name + " ");
}
Day theDay = Day.valueOf("FRIDAY");
}
```

```
All days of the week:
0) SUNDAY SUNDAY
1) MONDAY MONDAY
2) TUESDAY TUESDAY
3) WEDNESDAY WEDNESDAY
4) THURSDAY THURSDAY
5) FRIDAY FRIDAY
6) SATURDAY SATURDAY
```

15

페이지 19

모든 요일들을 프린트하는 코드입니다.
enumeration type인 day의 method인
day.name() 은 enumeration constant를 String으로 바꾸어 return 합니다.
day.ordinal() 은 0번째 부터 시작하여 몇번째 element인지를 return합니다.
day 자체를 프린트 시도하면 역시 day.name() 과 마찬가지로
String으로 바뀌어서 프린트 됩니다.
valueOf method는 주어진 String을 이름으로 가진 enumeration value가 return 됩니다.