3.4: Arrays

IT1406 - Introduction to Programming

Level I - Semester 1

3.3.1 Arrays

- Arrays provide a way of storing a list of variables of the same data type, one after the other

- Arrays must be declared. For example, an array of 10 integers might be declared as:

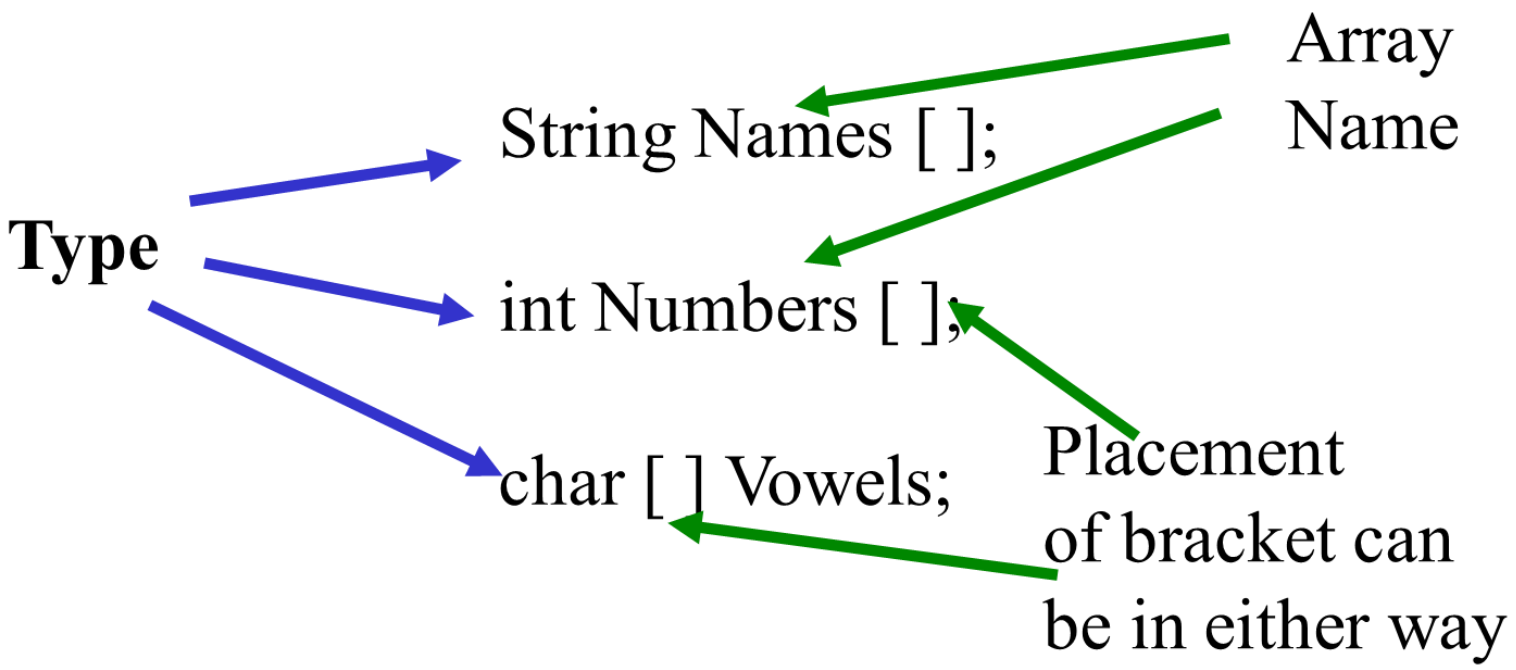
```
int[] a = new int[10];
```

- Elements of an array can be accessed by indicating the index (position) of the element in the array inside square brackets. Indexing (Position numbering) begins with 0.

- For example, to access the first element in an array, **a[0]**, the second element **a[1]**, and so on.

Declaring Array Variables

- Arrays are declared using enclosing square brackets.



Declaring Array Variables cont...

- By placing the bracket before the Array Name
  - we can declare multiple arrays of same type in the same line.

```
int [ ] firstArray, secondArray ;
```

Both are arrays of Type int

Declaring Array Variables cont …

- By placing the bracket after the Array Name
  - we can declare variables and arrays of same Type in the same line.

```
int  firstArray[ ], justAVariable ;
```

Array of Type int

Variable of Type int

Declaring Array Variables cont …

- If the return type is an array object,
  - the square brackets can go after the return type or after the parameter list

int [ ] **SortedList** (int List [ ]);

OR

int **SortedList** (int List [ ]) [];

Creating Array Objects

- Array Objects can be created using two methods.
  - **Using the new operator**

Number of elements

```
String firstName [ ] = new String [5];
```

  - **Directly Initializing the contents**

```
String firstName [ ] = { "Thisara", "Nimali" , "Hiran",
                         "Achala", "Kaushika" } ;
```

Accessing Array elements

- After initializing, Array elements can be accessed using subscript expression ([ ]) .

Subscript starts with 0
ends with 4

firstName [subscript];

Array with 5 elements

Example:
firstName[0] will give "Kamal"
firstName[1] will give "Amal"
firstName[2] will give "Nimal"

Array of Arrays (Multi Dimensional Arrays)

- In Java Multi Dimensional Arrays are just Arrays of Arrays.

[0][0]

Example:
twoDMatrix
2 Dimensional
Array of int

	0	1	2	3	4	5
0	5	1	2	8	6	9
1	8	4	1	6	2	1
2	3	9	2	5	8	9
3	1	7	3	6	4	3

Indvidual
Arrays of
int
[2][3]

Multi Dimensional Arrays (continued)

- Can be created in either way as follows

```
int twoDMatrix [ ][ ] = new int [4][6];

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

Note:
twoDMatrix.length=4
(Number of rows)
twoDMatrix[0].length=6
(lenth of first row Array)
twoDMatrix[1].length=6

Multi Dimensional Arrays Cont..

- Since Java Multi Dimensional Arrays are Arrays of Arrays,
  - It is possible to create *Ragged Arrays*

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

Variable, row array lenghths

twoDMatrix[1].length=4

(lenth of second row Array)

twoDMatrix[2].length=5

(lenth of third row Array)

Multi Dimensional Arrays Cont …

- **Alternatively**
  - It is possible to declare the same *Ragged Array* as follows:

```
int twoDMatrix [ ][ ] = new  int [4][ ];
twoDMatrix [0] = new  int [6];
twoDMatrix [1] = new  int [4];
twoDMatrix [2] = new  int [5];
twoDMatrix [3] = new  int [6];
```

Size of second Dimension is not Specified

Multi Dimensional Arrays Cont…

- Each element can be accessed using two subscripts

```
twoDMatrix [0][0];
twoDMatrix [0][1];
```

- Each element can be changed as follows

```
twoDMatrix [0][0] = 6;
twoDMatrix [0][1] = 8;
```

Accessing Array Elements (continued)

- Java Run Time will check to verify that the Array bounds are not exceeded.
- Each array object has a property called length which will yield the size of Array.

Example

firstName[5] will throw an **Exception**

firstName.length will yield 5

maximum subscript is always **firstName.length -1**

Changing Array Elements

- **To change an Array Element,**

  just use an assignment statement after the Array Access Expression

Example

firstName[3] = “Kamala”;

now the element 3 will contain value “Kamala”

Command-Line Arguments

- A Java application can accept any number of arguments from the command line. This allows the user to specify configuration information when the application is launched.

- The user enters command-line arguments when invoking the application and specifies them after the name of the class to be run. For example, suppose a Java application called Sort sorts lines in a file. To sort the data in a file named friends.txt, a user would enter:

```
java Sort friends.txt
```

- When an application is launched, the runtime system passes the command-line arguments to the application's main method via an array of Strings. In the previous example, the command-line arguments passed to the Sort application in an array that contains a single String: "friends.txt".

Command-Line Arguments

- The Echo example displays each of its command-line arguments on a line by itself:

```
public class Echo {
    public static void main (String[] args) {
        for (String s: args) {
            System.out.println(s);
        }
    }
}
```

Command-Line Arguments

Parsing Numeric Command-Line Arguments

- If an application needs to support a numeric command-line argument, it must convert a String argument that represents a number, such as "34", to a numeric value. Here is a code snippet that converts a command-line argument to an int:

```
int firstArg;
if (args.length > 0) {
try {
        firstArg = Integer.parseInt(args[0]);
    } catch (NumberFormatException e) {
        System.err.println("Argument" + args[0] + " must be an integer.");
        System.exit(1);
    }
}
```