Lecture 4

Arrays

- Up to this point, the only variables we have used were for individual values such as numbers or strings.

- In this part, we'll learn how to store multiple values of the same type using a single variable. This language feature will enable you to write programs that manipulate larger amounts of data.

What is array?

- Data structure which allows a collective name to be given to a group of elements which **all have the same type**. An individual element of an array is identified by its own unique **index**.

- An array is a **contiguous** series of elements of the **same type** where each element can be accessed by a **name** and an **index**.

Purpose of Array

- Why do we need it?
    - When we are programming, often we have to process a large amount of information. We can do so by creating a lot of variables to keep track of them. However this approach is not the optimal approach.
- Arrays are great for keeping track of similar group of data, where the **number of elements** required is known.

Declaring and Creating an Array

- Declare an array name and create an array in almost the same way that you create and name objects of classes. There is only a slight difference in the syntax.

- SYNTAX: Base_Type[] Array_Name = new Base_Type[Length];

- The *Length* may be given as any expression that evaluates to a nonnegative integer. In particular, *Length* can be an int variable.

Java Arrays

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. To declare an array, define the variable type with square brackets

```
String[] cars;
```

- We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
String[] cars = {"Volvo", "BMW", "Ford", “Mazda"};
```

- To find out how many elements an array has, use the length property:

```
String[] cars = {"Volvo", "BMW", "Ford", “Mazda"};
System.out.println(cars.length); // Outputs 4
```

Accessing elements

- When you create an array of **ints**, the elements are initialized to zero.

- The arrow indicates that the value of **counts** is a **reference** to the array. You should think of *the array* and *the variable* that refers to it as two different things. As we'll soon see, we can assign a different variable to refer to the same array, and we can change the value of **counts** to refer to a different array.

- The large numbers inside the boxes are the elements of the array. The small numbers outside the boxes are the **indexes** (or indices) used to identify each location in the array. Notice that the index of the first element is 0, not 1, as you might have expected.

Assign and Access Array

- Assign Example:

```
num[0]=5, num[1]=6, num[2]=8, num[3]=7, num[4]=3;
```

- Access Example:

```
System.out.print(num[0]);// output 5
System.out.print(num[1]);// output 6
System.out.print(num[2]);// output 8
System.out.print(num[3]);// output 7
System.out.print(num[4]);// output 3
```

- NOTE: System.out.print(num); //Check what it prints out!

Loop Through an Array

- You can loop through the array elements with the for loop, and use the length property to specify how many times the loop should run.

```
String[] cars = {"Volvo", "BMW", "Ford", “Mazda”};
for (int i = 0; i < cars.length; i++) {
   System.out.println(cars[I]);
}
```

Loop Through an Array with For-Each

- There is also a "**for-each**" loop, which is used exclusively to loop through elements in arrays:

- The following example outputs all elements in the **cars** array, using a "**for-each**" loop

```
String[] cars = {"Volvo", "BMW", "Ford", “Mazda”};
for (String i : cars){
   System.out.println(cars[I]);
}
```

Ex1: Print the array

Write a program that initializes array of eight integer with 5,15,25,35,45,55,65, and 75.

1. Prints each of these out.

2. Prints sum.

Ex2: Print the array

Write a program that prompts the user for each of eight values stores these in an array.

1. Prints each of these out.

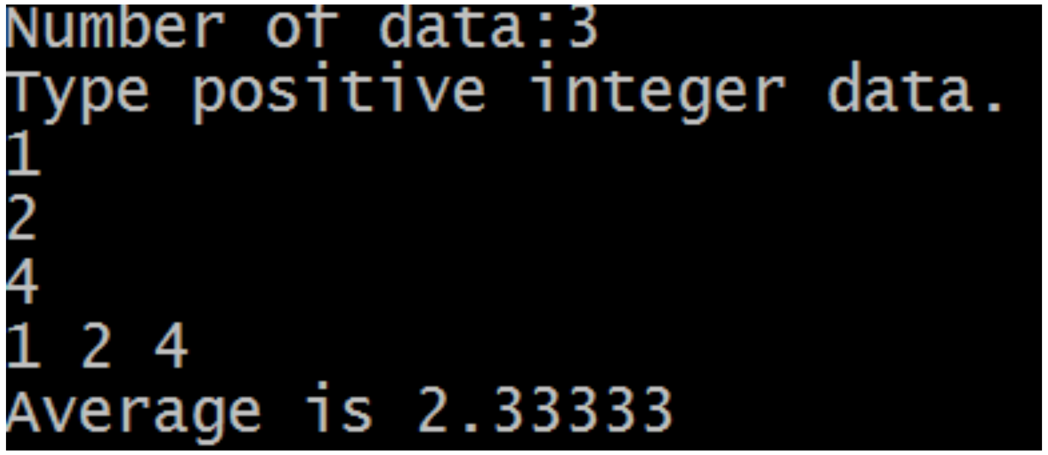
2. Prints their total.

Ex3: Print the array

Write a program that get the number of data and get values of data. Store these in an array.

1. Prints each of these out.

2. Prints their average.
(To show average, change to double.)



```
Number of data:3
Type positive integer data.
1
2
4
1 2 4
Average is 2.33333
```

Exercise

- Exercise 4: Write a program that initializes array with 4, 5, 3, 10, 9, and string array with "Max", "Arthur", "Freddy", "Kelly", “Jack". Print the name and score in the same line to the console.

- Exercise 5: Create a short program that uses array.
  - Ask the user for number of students.
  - Declare an array of string to store the names.
  - Using a loop, read in each name into the array.
  - Using a loop, display the names to the user.

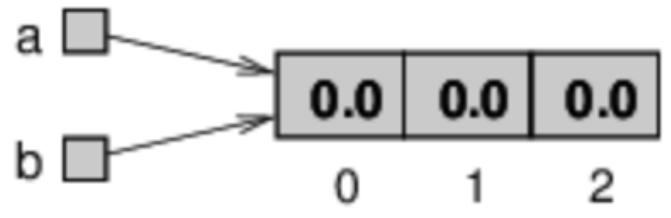
- Exercise 6: Once you have the above tasks working do the following:
  - Declare an array of grades. (Type int)
  - Using random function to assign the grade of each student, the student grade should be in range of 60-100.
  - Using a loop, display the name and the grade of each students.

Copying arrays

- Array variables contain *references* to arrays. When you make an assignment to an array variable, it simply copies the reference. But it doesn't copy the array itself!

```
double[] a = new double[3];
double[] b = a;
```

- These statements create an array of three doubles and make two different variables refer to it, as shown in



- Any changes made through either variable will be seen by the other. Because **a** and **b** are different names for the same thing, they are sometimes called **aliases**.

Copying arrays

- If you actually want to copy the array, not just a reference, you have to create a new array and copy the elements from the old to the new.

```
double[] a = new double[3];
for (int i = 0; i < 3; i++) {
    b[i] = a[i];
}
```

- Another option is to use **java.util.Arrays**, which provides a method named **copyOf** that copies an array. You can invoke it like this:

```
double[] b = Arrays.copyOf(a, 3);
```

- The second parameter is the number of elements you want to copy, so you can also use **copyOf** to copy just part of an array.

Array length

- All arrays have a built-in constant, **length**, that stores the number of elements. The expression **a.length** may look like a method invocation, but there are no parentheses and no arguments.

- The last time this loop gets executed, **i** is **a.length - 1**, which is the index of the last element. When **i** is equal to **a.length**, the condition fails and the body is not executed – which is a good thing, because trying to access **a[a.length]** would throw an exception.

Multi-Dimensional Arrays

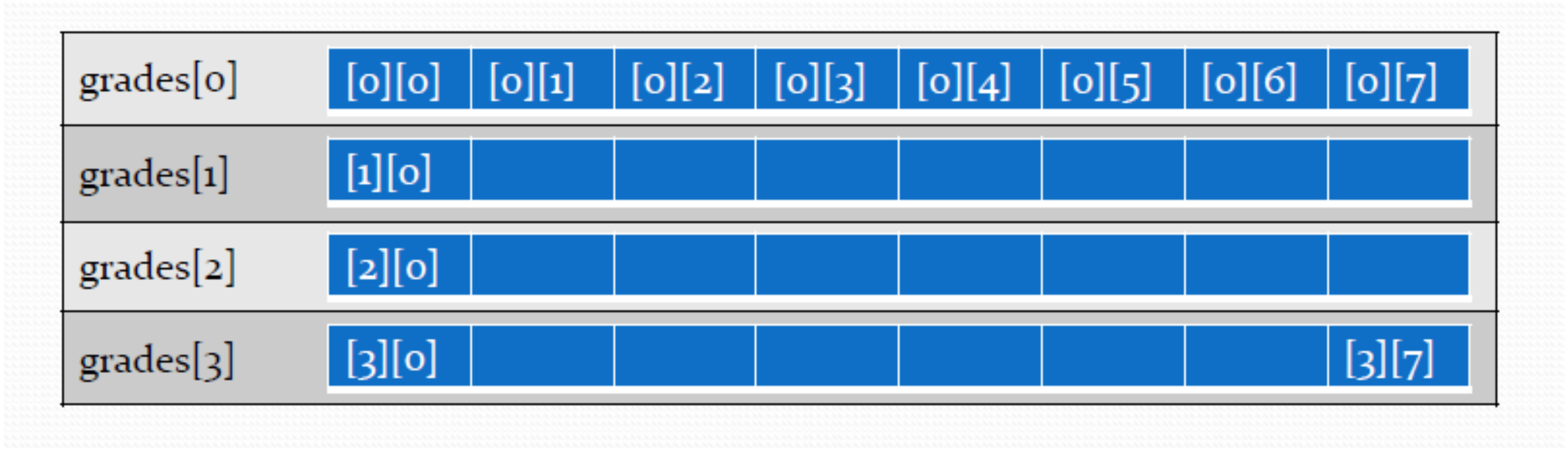
- A multidimensional array is an array of arrays. To create a two-dimensional array, add each array within its own set of **curly braces**

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

- **myNumbers** is now an array with two arrays as its elements.

2D Array data structure

- Say we have the following array:
- int grades[][] = new int[4][8];
- Here is the graphical representation:



Multidimensional Arrays

- To access the elements of the **myNumbers** array, specify two indexes: one for the array, and one for the element inside that array. This example accesses the third element (2) in the second array (1) of myNumbers:

  ```
  int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
  int x = myNumbers[1][2];
  System.out.println(x); // Outputs 7

  // int[][] myNumbers = new int[2][3];
  ```

Multidimensional Arrays

- We can also use a **for loop** inside another **for loop** to get the elements of a two-dimensional array (we still have to point to the two indexes):

```
public class Main {
    public static void main(String[] args) {
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
        for (int i = 0; i < myNumbers.length; ++i) {
            for(int j = 0; j < myNumbers[i].length; ++j) {
                System.out.println(myNumbers[I][j]);
            }
        }
    }
}
```

Exercise

- Exercise 7: Write a program that declare an integer array with size 4x3, it saves quiz grade for 4 students where each column stands for three quiz grade for each student. Assign random number between 60-100 to quiz grade, and calculate the average grade for each student then print them out.

Exercise

- Exercise 8: Create a 2-dimensional array with 4 rows and 30 columns. Row represents sections of a course and column represents the students, value inside each position of the array is the Final exam grade for each students. Fill the array with random numbers between 40 and 100. Calculate the total, average, maximum, minimum for each section.

Exercise

- Exercise 9: Write a program that sets up an array of integers with capacity 5. Then user enter the 5 entries in turn. Each entry must be an integer between 1 and 20, if user entries are out of range, system should repeatedly make new request until a legal entry value is entered. The program should finish by printing the list of 5 array values and the average.

Exercise

- Exercise 10: Write a program that initializes array with 3,4,5,2,3. Write a function call minArray which accepts array and size of array, and return the minimum of the array.

- Exercise 11: Write a program that initializes array with 3,4,5,2,3. Write a function call avArray which accepts array and size of array, and return the average of the array.

Exercise

- Exercise 12: Write a program that initializes 2D array with {{3,4,5,2,3}, {2,3,4,5,1}, {6,5,4,3,2}}. Write a function call minArray which accepts array and size of row and column of array, and return the minimum of the 2D array.

- Exercise 13: Write a program that initializes array with {{3,4,5,2,3}, {2,3,4,5,1}, {6,5,4,3,2}}. Write a function call avArray which accepts array and size of row and column of array, and return the average of the 2D array.

Exercise

- Exercise 14: Write a complete program that does the following.
  a. It asks the user to enter a positive integer value, r that is at most 100.
  b. The program reads a value entered by the user. If the value is not in the right range, the program should terminate.
  c. The program reads and stores r integers from the user and then prints a pattern of r rows of stars, the lengths of which are the other integers entered by the user.
  For example, the following represents one run of the program.

```
How many rows? 4
Enter 4 row lengths: 2 7 1 5
**
*******
*
*****
```

Exercise

- Exercise 15: Write a complete program that does the following.
  a. It asks the user to enter a 5-digit integer value, n.
  b. The program reads a value entered by the user. If the value is not in the right range, the program should terminate.
  c. The program calculates and stores the 5 individual digits of n.
  d. The program outputs a "bar code" made of 5 lines of stars that represent the digits of the number n.
  For example, the following represents one run of the program. (The user chooses the number 16384.)

```
Enter a 5 digit integer: 16384
*
******
***
********
****
```

Exercise

- Exercise 16: Write a complete program that does the following.
  1. It asks the user to enter 9 integers as the entries of a 3 x 3 table.
  2. The program reads the 9 entries, row by row and prints the table.
  3. If every row and column of the table have the same sum then the program adds the message: MAGIC.

  Here is an example of how the program should work:

```
Enter 9 entries of a 3 x 3 table: 10 14 18 15 16 11 17 12 13
output:
10 14 18
15 16 11
17 12 13
MAGIC
```

Exercise

- Exercise 17: Eight queens are to be placed on an 8 x 8 chessboard in such a way that one queen is to be in each column. A program will store an array x[] with capacity 8 to represent such a configuration. If x[c] has value r then in row r there is a queen in column c. Write a program that asks a user to enter the rows that contain queens in the 8 columns. The program then prints the board.

  For example, if the user enters: 2,3,4,0,1,7,6,5 the program should print:

```
. . . Q . . . .
. . . . Q . . .
Q . . . . . . .
. Q . . . . . .
. . Q . . . . .
. . . . . . . Q
. . . . . . Q .
. . . . . Q . .
```

Exercise

- Exercise 18: Eight queens are to be placed on an 8 x 8 chessboard in such a way that one queen is to be in each row and one queen is to be in each column. A program will store an array x[] with capacity 8 to represent such a configuration. If x[c] has value r then in row r there is a queen in column c. Write a program that asks a user to enter the rows that contain queens in the 8 columns. The program then checks whether there is just one queen per row. For example, if the user enters: 2,3,4,0,1,7,6,5 the program should print: OK (because the user has entered the configuration that was entered in problem 3). But if the user enters 0,0,1,2,3,4,5,6 the program should print: No good. (Why?) 0 6 4 7 1 3 5 2 -- good solution, 2,3,4,0,1,7,6,5 -- bad solution

- Exercise 19: Write a program to read an array of 11 integers from a user and compute the median entry of the array. (Hint: sort the array and get the middle position, bubble sort, merge sort, selection sort)