More on Array & ArrayList

Cheng, Wei  COMP110-001  June 5, 2014
Today in COMP 110

• More about arrays

• 2D arrays

• Examples of using ArrayList
Arrays as instance variables

```
public class Weather
{
    private double[] temperature;
    private double[] pressure;

    public void initializeTemperature(int len)
    {
        temperature = new double[len];
    }
}
```
Arrays of objects

• When you create an array of objects like this:
  Student[] students = new Student[35];
• Each of the elements of students is not yet an object
• You have to instantiate each individual one
  students[0] = new Student();
  students[1] = new Student();
• ...or do this in a loop
Arrays of objects

Smiley[] smilies = new Smiley[3];
for (int i = 0; i < smilies.length; i++)
{
    smilies[i] = new Smiley();
}

smilies[0].color
    = Color.GREEN;
...

```
?   ?   ?
true GREEN
   3
false BLUE
   1
false CYAN
   4
```
Arrays of objects

Student[] students = new Student[5];
for (int i = 0; i < students.length; i++)
{
    students[i] = new Student(keyboard.nextInt());
    students[i].printAge();
}
Arrays as parameters

```java
public void changeArray(int[] arr)
{
    int len = arr.length;
    arr[len - 1] = 25;
}
```

| 23 | 47 | 52 | 14 | 25 |
Arrays as return types

```java
public double[] buildArray(int len) {
    double[] retArray = new double[len];
    for (int i = 0; i < retArray.length; i++) {
        retArray[i] = i * 1.5;
    }

    return retArray;
}
```
Indexed variables as method arguments

- No different from using a regular variable

```java
public void printNum(int num) {
    System.out.println(num);
}

public void doStuff() {
    int[] scores = { 15, 37, 95 };

    for (int index = 0; index < scores.length; index++) {
        printNum(scores[index]);
    }
}
```
2D arrays

- Arrays having more than one index are often useful
  - Tables
  - Grids
  - Bingo games

<table>
<thead>
<tr>
<th></th>
<th>0: Open</th>
<th>1: High</th>
<th>2: Low</th>
<th>3: Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Apple Inc.</td>
<td>99.24</td>
<td>99.85</td>
<td>95.72</td>
<td>98.24</td>
</tr>
<tr>
<td>1: Walt Disney Co.</td>
<td>21.55</td>
<td>24.20</td>
<td>21.41</td>
<td>23.36</td>
</tr>
<tr>
<td>2: Google Inc.</td>
<td>333.12</td>
<td>341.15</td>
<td>325.33</td>
<td>331.14</td>
</tr>
<tr>
<td>3: Microsoft Corp.</td>
<td>21.32</td>
<td>21.54</td>
<td>21.00</td>
<td>21.50</td>
</tr>
</tbody>
</table>
Declaring and creating 2D arrays

```java
int[][] table = new int[4][3];
```

or

```java
int[][] table;
table = new int[4][3];
```
 Declaring and creating 2D arrays

```java
int[][] table = new int[4][3];

gives you the ability to use

table[0][0]
table[0][1]
table[0][2]
table[1][0]
table[1][1]
table[1][2]
table[2][0]
table[2][1]
table[2][2]
table[3][0]
table[3][1]
table[3][2]
```
How do you use a 2D array?

• We used a loop to iterate over a 1D array

```java
int[] scores = { 13, 57, 93, 60, 102 };
for (int i = 0; i < scores.length; i++)
{
    System.out.println(scores[i]);
}
```
How do you use a 2D array?

- How about a 2D array?

```java
int[][] table = new int[4][3];
```

- Use a nested loop

```java
for (int row = 0; row < 4; row++)
{
    for (int column = 0; column < 3; column++)
    {
        table[row][column] = 37;
    }
}
```
Multidimensional arrays

• You can have more than two dimensions

```java
int[][][] table = new int[4][3][5];
```

• Use more nested loops to access all elements
public void print2DArray(int[][][] arr)
{
    for (int row = 0; row < arr.length; row++)
    {
        for (int column = 0; column < arr[row].length; column++)
        {
            System.out.print(arr[row][column] + " ");
        }
        System.out.println();
    }
}
Multidimensional arrays as return types

```java
public int[][][] giveMeAnArray()
{
    int[][][] table = new int[4][3];
    // put values in the table
    return table;
}
```
length for a 2D array

int[][] table = new int[4][3];

• table.length is the number of rows, or the integer in the first pair of brackets (4)
• table[i].length is the number of columns, or the integer in the second pair of brackets (3)
Why? Arrays of arrays

```java
int[] scores = new int[5];

• scores is a one-dimensional array
  – base type is int

int[][] table = new int[4][3];

• table is also in fact a one-dimensional array
  – base type is int[
• We still refer to table as a two-dimensional array
```
We can have 2-D array of irregular shape

```java
int[][] x = new int[3][];
x[0] = new int[6];
x[1] = new int[3];
System.out.println(x[0].length);
System.out.println(x[1].length);
System.out.println(x[2].length);
```

What’s wrong with this line?
ArrayList

- Array – fixed size once declared
  - 1D, 2D, ... n-D
  - 2D array does not have to be rectangle

- ArrayList – dynamic size
  - A list that can be manipulated in many ways
  - Very useful in practice
  - In case some students do not read ArrayList documentation, I have several boring slides here.
Creating an ArrayList

- In this example we will create an ArrayList of String in Java. This Java ArrayList will only allow String and will throw compilation error if we try to any other object than String.

```java
ArrayList<String> stringList = new ArrayList<String>();
//ArrayList to Store only String objects
```
Putting an Item into ArrayList

- Second line will result in compilation error because this Java ArrayList will only allow String elements.

```java
stringList.add("Item");
    //no error because we are storing String

stringList.add(new Integer(2));
    //compilation error
```
Size & IndexOf

• Size of an ArrayList in Java is total number of elements currently stored in ArrayList.

```java
int size = stringList.size();
```

• Checking Index of an Item in Java Arraylist

```java
int index = stringList.indexOf("Item");
// location of Item object in List
```
Retrieving Item from ArrayList in a loop

```java
for (int i = 0; i < stringList.size(); i++)
    String item = stringList.get(i);
    System.out.println("#" + i + " : " + item);
}
```

• Optional:
  From Java 5 onwards you can use foreach loop as well

```java
for (String item: stringList){
    System.out.println("retrieved element: " + item);
}
```
Checking if ArrayList is Empty

- We can use `isEmpty()` method of Java ArrayList to check whether ArrayList is empty. `isEmpty()` method returns true if this ArrayList contains no elements. You can also use `size()` method of List to check if List is empty.

```java
boolean result = stringList.isEmpty(); // isEmpty() will return true if List is empty

if (stringList.size() == 0){
    System.out.println("ArrayList is empty");
}
```
Removing an Item from ArrayList

- There are two ways to remove any elements from ArrayList in Java. You can either remove an element based on its index or by providing object itself. Remove remove (int index) and remove (Object o) method is used to remove any element from ArrayList in Java. Since ArrayList allows duplicate its worth noting that remove (Object o) removes the first occurrence of the specified element from this list, if it is present. In below code first call will remove first element from ArrayList while second call will remove first occurrence of item from ArrayList in Java.

```java
stringList.remove(0);
stringList.remove(item);
```
A lot more with ArrayList

We can easily

• Copy a list
• Sort a list
• Convert from array to ArrayList or in the reverse direction
One more Example

- A stupid vacation resort advisor
- It asks for user’s name and then tells user a resort based on
- ..... How long the name is
A stupid vacation resort advisor

First, let’s create a resort list:

```java
ArrayList<String> myArr = new ArrayList<String>();
myArr.add("Italian Riviera");
myArr.add("Jersey Shore");
myArr.add("Puerto Rico");
myArr.add("Los Cabos Corridor"); myArr.add("Lubmin");
myArr.add("Coney Island");
myArr.add("Karlovy Vary");
myArr.add("Bourbon-l'Archambault");
myArr.add("Walt Disney World Resort"); myArr.add("Barbados");
```
A stupid vacation resort advisor

System.out.println("Stupid Vacation Resort Adviser");
System.out.println("Enter your name:");
String name = keyboard.readLine();
int nameLength = name.length();
int vacationIndex = nameLength % myArr.size();
System.out.println("Your name is " + name + ", its length is " + nameLength + " characters,\n + "that's why we suggest you to go to " + myArr.get(vacationIndex));