COMP 110-001
More About Arrays

Yi Hong

June 05, 2015

Today

- 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

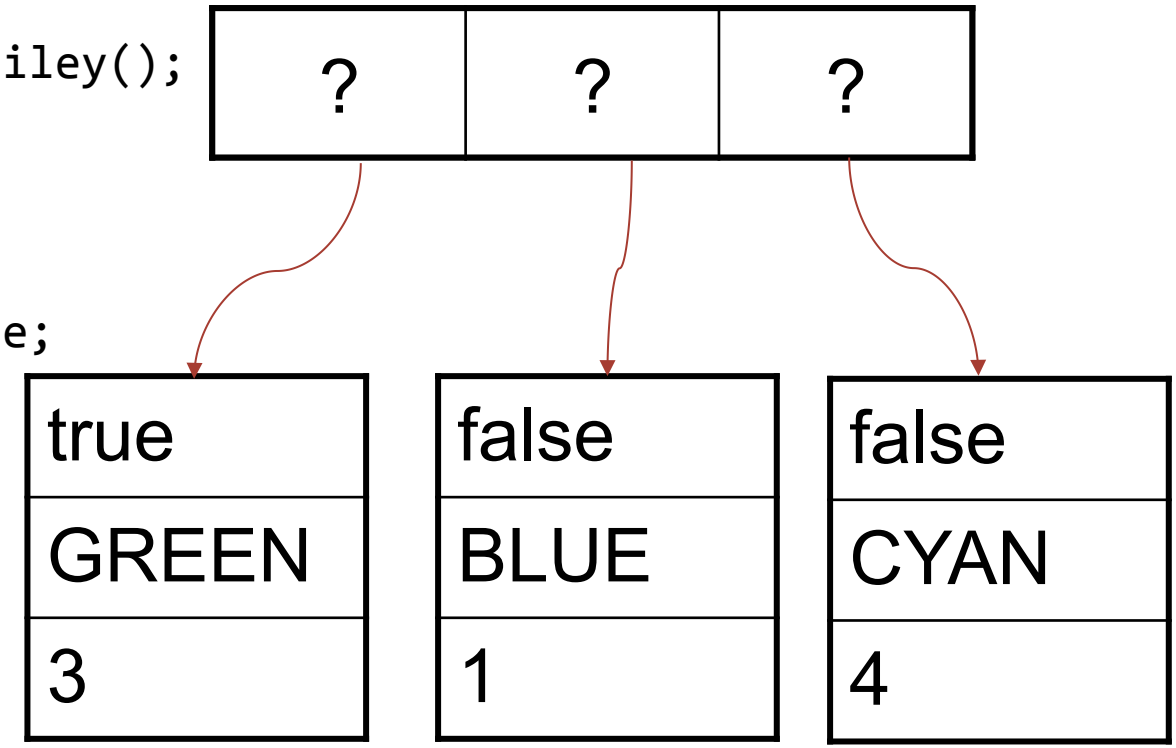
```
students[0] = new Student();
students[1] = new Student();
```

- …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].bSmile = true;
……
```

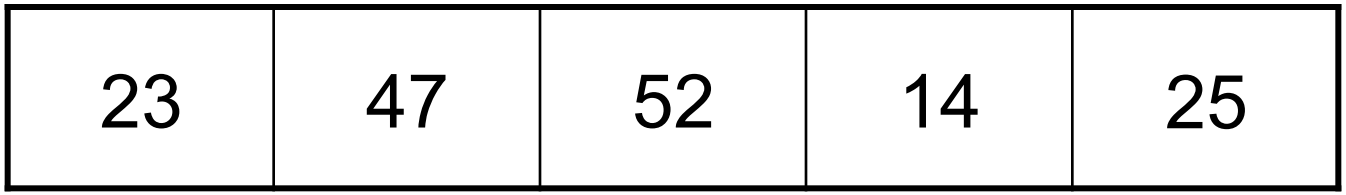


Arrays of Objects

```
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

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



Arrays as Return Types

```
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

```
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

	0: Open	1: High	2: Low	3: Close
0: Apple Inc.	129.50	130.58	128.91	129.36
1: Facebook, Inc.	82.14	82.95	81.51	82.05
2: Google Inc.	537.07	540.59	534.32	536.70
3: Microsoft Corp.	46.77	47.16	46.20	46.36

Declaring and Creating 2D Arrays

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

or

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

Declaring and Creating 2D Arrays

```
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

```
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?

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

- Use a nested loop

```
for(int row = 0; row < table.length; row++)
{
    for(int column=0; column < table[row].length; column++)
    {
        table[row][column] = 30;
    }
}
```

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)

Multidimensional Arrays

- You can have more than two dimensions

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

- Use more nested loops to access all elements

Multidimensional Arrays as Parameters

```
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

```
public int[][] giveMeAnArray()
{
    int[][] table = new int[4][3];
    // put values in the table
    return table;
}
```

Why? Arrays of Arrays

```
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

2D Array of Irregular Shape

```
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 the last 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.

```
//ArrayList to Store only String objects
ArrayList<String> stringList
        = new ArrayList<String>();
```

Putting an Item into ArrayList

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

```
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.

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

- Checking Index of an “Item” in Java Arraylist

```
int index = stringList.indexOf("Item");
        //location of Item object in List
```

Retrieving Item from ArrayList in a Loop

```
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

```
for(String item: stringList){
  System.out.println("retrieved element: " + item);
}
```

Checking if ArrayList is Empty

- Use isEmpty() method of Java ArrayList to check whether ArrayList is empty
  - isEmpty() method returns true if this ArrayList contains no elements.

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

- Or size() method of List to check if List is empty

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

Removing an Item from ArrayList

- Two ways: remove an element based on its index or by providing object itself.
  - remove (Object o) removes the first occurrence of the specified element from this list, if it is present.
- Remove first element from ArrayList

```
stringList.remove(0);
```

- Remove first occurrence of item from ArrayList in Java

```
stringList.remove(item);
```

More About ArrayList

- Check Java API

Next Class

- Lab 7