THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Tuesday October 7, 2014

Jay Aikat
Fall 2014
TR 9:30 – 10:45, GS-G100

Previous Class

- What did we discuss?

Announcements

- **Midterm – Tue, Oct 14**
- **Midterm Study guides on Sakai**
  - **See Resources**

Arrays

- `int[] scores = new int[5];`
- This is like declaring 5 strangely named variables of type int:
  - scores[0]
  - scores[1]
  - scores[2]
  - scores[3]
  - scores[4]
- Especially, you can use **score[i]** to locate a single one

Arrays

- An ***array*** is a collection of items of the same type
- Like a list of different variables, but with a nice, compact way to name them
- A special kind of object in Java
- **Loops repeat things temporally; arrays repeat things spatially**

Index

- Variables such as scores[0] and scores[1] that have an integer expression in square brackets are known as:
  - ***indexed variables***, ***subscripted variables***, ***array elements***, or simply ***elements***
- An ***index*** or ***subscript*** is an integer expression inside the square brackets that indicates an array element
  - ArrayName[index]

Index

- Index numbers start with **0**. They do NOT start with 1 or any other number.
  - Not like counters in loops, you can't change the range of indices
- The reason is that the array name represents a memory address, and the i^{th} element can be accessed by the address plus i

Array and Index

var name	score[0]	score[1]	score[2]	score[3]	score[4]
data	62	51	88	70	74
m address	25131	25132	25133	25134	25135

score *score+1* *score+2*

- In history, computer scientists argued a lot on this
  - *"Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration."* – Stan Kelly-Bootle

Access Elements with Indices

- The number inside square brackets can be any integer expression
  - An integer: *scores[3]*
  - Variable of type int: *scores[index]*
  - Expression that evaluates to int: *scores[index*3]*
- Can use elements just like any other variables:
  - *scores[3] = 68;*
  - *scores[4] = scores[4] + 3;* // just made a 3-pointer!
  - *System.out.println(scores[1]);*

Indices and For-Loops

- In programming, a for-loop usually starts with counter i = 0. There is a reason

```
for (int i = 0; i < 5; i++) {
        scores[i] = keyboard.nextInt();
        scoreSum += scores[i];
}
```

Creating an Array

- Array is a special class and we create its objects
  - Syntax for creating an array:
    - **Base_Type[] Array_Name = new Base_Type[Length];**
  - Example:
    - **int[] pressure = new int[100];**
  - Alternatively:
    - **int[] pressure;**
    - **pressure = new int[100];**

Creating an Array

- The base type can be any type
  - **double**[] temperature = new **double**[7];
  - **Student**[] students = new **Student**[350];
- The number of elements in an array is called its **length** or **size**
  - temperature has 7 elements, temperature[0] through temperature[6]
  - students has 350 elements, students[0] through students[349]

Creating an Array

- Create an array with given length saved in constants
  - **public static final int NUMBER_OF_READINGS** = 100;
  - **int**[] pressure = **new int**[**NUMBER_OF_READINGS**];
- Create an array with user input length
  - System.out.println("How many scores?");
  - **int numScores** = keyboard.nextInt();
  - **int**[] scores = **new int**[**numScores**];

Finding Length of an Existing Array

- An array is a special kind of object
  - It has one public instance variable: *length*
  - *length* is equal to the length of the array

```
Pet[] pets = new Pet[20];
pets.length has the value 20
```

  - You cannot change the value of *length* because it is **final**

Do not be OUT OF BOUNDS!

- Indices MUST be in bounds
  - double[] entries = new double[**5**]; // from [0] to [4]
  - entries[**5**] = 3.7; // ERROR! Index out of bounds
- Your code will compile if you are using an index that is out of bounds, but it will give you a run-time error!

Initializing Arrays

- You can initialize arrays when you declare them
  - int[] scores = { 68, 97, 102 };
- Equivalent to
  - int[] scores = new int[3];
  - scores[0] = 68;
  - scores[1] = 97;
  - scores[2] = 102;
- Or, you can use for-loop
  - When in doubt, for-loop!

Joke

- Q: Why did the programmer quit his job?
- A: Because he didn't get arrays.

  Hint: A raise ;-)

2D Arrays

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

	0: Open	1: High	2: Low	3: Close
0: Apple Inc.	99.24	99.85	95.72	98.24
1: Walt Disney Co.	21.55	24.20	21.41	23.36
2: Google Inc.	333.12	341.15	325.33	331.14
3: Microsoft Corp.	21.32	21.54	21.00	21.50

Declaring and Creating 2D Arrays

- Two pairs of square brackets means 2D
  - **int[][] table = new int[3][4];**
- or
  - **int[][] table;**
  - **table = new int[3][4];**

Declaring and Creating 2D Arrays

- Array (or 1D array) gives you a list of variables
  - int[] score = new int[5] gives you score[0], score[1], … , score[5]
- 2D array gives you a table of variables
  - int[][] table = new int[3][4];

table[0][0]	table[0][1]	table[0][2]	table[0][3]
table[1][0]	table[1][1]	table[1][2]	table[1][3]
table[2][0]	table[2][1]	table[2][2]	table[2][3]

Using a 2D Array

- We use a loop to access 1D arrays

```
for (int i = 0; i < 5; i++) {
        scores[i] = keyboard.nextInt();
        scoreSum += scores[i];
}
```


Using a 2D Array

- We use nested loops for 2D arrays

```
int[][] table = new int[4][3];
for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 3; j++) {
                table[i][j] = i * 3 + j;
                System.out.println(table[i][j]);
        }
}
```

Multidimensional Arrays

- You can have more than two dimensions
  - `int[][][] cube = new int[4][3][4];`
- Use more nested loops to access all elements
  - for (int i…)
    - for (int j…)
      - for (int k…)

Arrays Example

```
import java.util.*;
public class SampleArray
{
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int[] myArray = new int[5];

        for ( int i = 0; i < myArray.length; i++) {
                System.out.println("Please input a number");
                myArray[i] = keyboard.nextInt();
                System.out.println(myArray[i]);
        }
    }
}
```

Comparing Scores/Averages w/ Arrays

```
System.out.println("Enter 5 basketball scores:");
Scanner keyboard = new Scanner(System.in);
int[] scores = new int[5];
int scoreSum = 0;
for (int i = 0; i < 5; i++) {
        scores[i] = keyboard.nextInt();
        scoreSum += scores[i];
}
double average = (double) scoreSum / 5;
System.out.println("Average score: " + average);
```

```
for (int i = 0; i < 5; i++) {
        if (scores[i] > average)
                System.out.println(scores[i] + ": above average");
        else if (scores[i] < average)
                System.out.println(scores[i] + ": below average");
        else
                System.out.println(scores[i] + ": equal to the average");
}
```


Next class (Thu, Oct 9)

- Midterm Review