THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Thursday October 2, 2014

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

Previous Class

- What did we discuss?

Announcements

- **Assignment3 – Due today**
- **Lab4 – Due Mon, Oct 6**

Local Variables

```
public class test123 {
    public static void main(String[] args)
    {
        int num1 = 5;
        int count;

        for (count = 0; count <= num1; count++) {
            int num2 = 10;
            System.out.println(count);
             System.out.println(num2);
        }
        System.out.println(num2);  // num2 does not exist outside the loop
        }
}
```

`String` Methods

`replace(`*OldChar*, *NewChar*`)` Returns a new string having the same characters as this string, but with each occurrence of *OldChar* replaced by *NewChar*.
`substring(`*Start*`)` Returns a new string having the same characters as the substring that begins at index *Start* of this string through to the end of the string. Index numbers begin at 0.
`substring(`*Start*, *End*`)` Returns a new string having the same characters as the substring that begins at index *Start* of this string through, but not including, index *End* of the string. Index numbers begin at 0.
`trim()` Returns a new string having the same characters as this string, but with leading and trailing whitespace removed.

Exercise 3

- Ask user to input a string
- Find the length of the string
- If the string length is less than 10, find the 5th character in the string, else find the 11th character
- Compare this string with another string
- Replace a character within a string
- Find a substring within the string

Arrays

- To think about arrays, let's think about loops first
- Why do we need loops?
  - Because we want to repeat things without writing them again and again

Average Score without Loops

- Assuming that we only need 5 scores

```
int score1 = keyboard.nextInt();
int score2 = keyboard.nextInt();
int score3 = keyboard.nextInt();
int score4 = keyboard.nextInt();
int score5 = keyboard.nextInt();

double average = (double) (score1 + score2 +
             score3 + score4 + score5) / 5.0;
```

Average Score with Loops

- Assuming that we only need 5 scores

```
for (int i = 0; i < 5; i++)
      scoreSum += keyboard.nextInt();

double average = (double) scoreSum / 5.0;
```


What if we really need to save them

- If we really need to save these scores, loop won't help you
- Think about this problem
  - Print out if a score is above/below average
  - We have to calculate average first, then decide if a score is above/below average
  - Therefore we must save all these scores, and compare them to the average in the end

Comparing All Scores and the Average

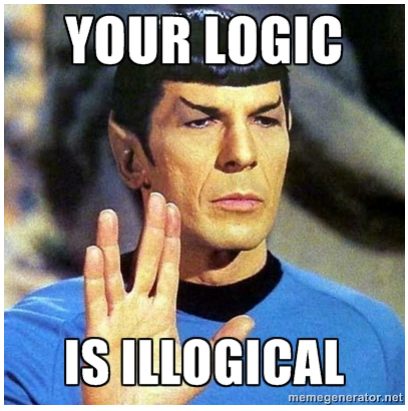
```
System.out.println("Enter 5 basketball scores:");
Scanner keyboard = new Scanner(System.in);
int score1 = keyboard.nextInt();
int score2 = keyboard.nextInt();
int score3 = keyboard.nextInt();
int score4 = keyboard.nextInt();
int score5 = keyboard.nextInt();
double average = (double) (score1 + score2 + score3 + score4 + score5) / 5.0;
System.out.println("Average score: " + average);

// repeat this for each of the 5 scores
if (score1 > average)
        System.out.println(score1 + ": above average");
else if (score1 < average)
        System.out.println(score1 + ": below average");
else
        System.out.println(score1 + ": equal to the average");

// if score2...score3...score4...
```


If we have more Scores…

- Think about 80 scores…
  - Declare 80 variables
  - Check them 80 times
- This is illogical!
- There must be an easier way!
  - What about things like: $Score_1$, $Score_2$, …, $Score_n$



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**

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");
}
```


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

- Where have we seen the word index before?
  - String's indexOf() method

H	o	w		a	r	e		y	o	u	?
0	1	2	3	4	5	6	7	8	9	10	11

  - str.indexOf('e') == 6;
  - str.charAt(6) == 'e';

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

- 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**];

Debugger in Eclipse

- *Debugging* allows you to run a program interactively while watching the source code and the variables during the execution
- Using *breakpoints* in the source code, you specify where the execution of the program should stop
- Add line numbers
- Toggle breakpoint where needed

https://www.youtube.com/watch?v=dHYM3b3ZEjU

Debugger in Eclipse – exercise1

```
int count = 0;
int number = 5;
while ( count <= number) {
    System.out.println(count);
    count++;
}
```

Debugger in Eclipse – exercise2

```
int num = 5;
for ( int i = 0; i <= num; i++) {
    System.out.println(i);
}
```


Next class (Tue, Oct 7)

- More on Arrays