جـامعة بيروت العربية
BEIRUT ARAB UNIVERSITY

CMPS 241
Introduction to Programming

Do – While Loop

Arrays – Part I

Text Processing

Remember

Type `char`

- **`char`** : A primitive type representing single characters.
  - A `String` is stored internally as an array of `char`

```
String s = "Ali G.";
```

index	*0*	*1*	*2*	*3*	*4*	*5*
value	`'A'`	`'l'`	`'i'`	`' '`	`'G'`	`'.'`

- The `char`s in a `String` can be accessed using the `charAt` method.
  - accepts an `int` index parameter and returns the `char` at that index

```
String food = "cookie";
char firstLetter = food.charAt(0);    // 'c'
```

The `for loop` and `charAt` method

- You can use a `for` loop to print or examine each character.

```
String major = "CMPS";
for (int i = 0; i < major.length(); i++){ // output:
    char c = major.charAt(i);             // C
    System.out.println(c);                // M
}                                         // P
                                          // S
```

- **Another example**

```
// prints the alphabet
  for (char c = 'a'; c <= 'z'; c++) {
      System.out.print(c);
  }
```

Comparing `char` values

- You can compare `char`s with `==`, `!=`, and other operators:

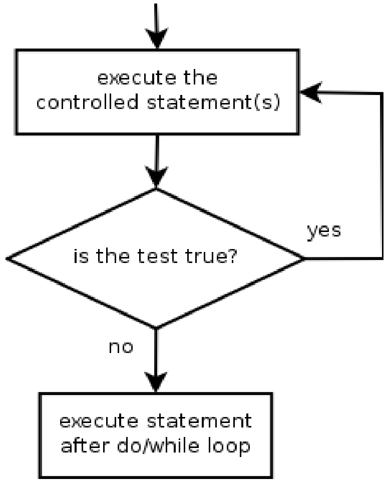
```
   /* count the number of occurrences of letter ‘i’ in
the string “Beirut Arab University” */

     String Univ = “Beirut Arab Univeristy";
     int count = 0;
     for (int j = 0; j < Univ.length(); j++){
         if(Univ.charAt(j)==‘i’) count++;
      }
    System.out.println(count);

   // Output: 3
```

The `do/while` loop

The `do/while` loop

- **`do/while` loop**: Performs its test at the *end* of each repetition.
  - Guarantees that the loop's `{ }` body will run at least once.

```
do {
    statement(s);
} while (test);
```


execute the controlled statement(s)
is the test true?
yes
no
execute statement after do/while loop

```
// Example: prompt until correct password is typed
String phrase;
do {
    System.out.print("Type your password: ");
    phrase = console.next();
} while (!phrase.equals("abracadabra"));
```

`do/while` question

- Rolls two dice until a sum of 7 is reached

```
2 + 4 = 6
3 + 5 = 8
5 + 6 = 11
1 + 1 = 2
4 + 3 = 7
You won after 5 tries!
```

do/while answer

```
// Rolls two dice until a sum of 7 is reached.
public class Dice {
    public static void main(String[] args) {
        int tries = 0;
        int sum;

        do {
            int roll1 = 1 + (int) (Math.random() * 6); // one roll
            int roll2 = 1 + (int) (Math.random() * 6);
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        } while (sum != 7);

        System.out.println("You won after " + tries + " tries!");
    }
```

Do-While Loop vs. While Loop

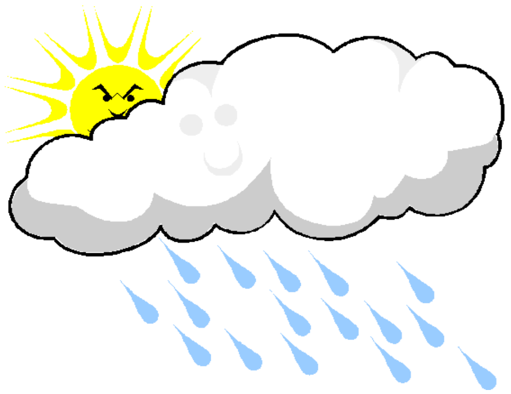
- POST-TEST loop
- The looping condition is tested after executing the loop body.
- Loop body is always executed at least once.

- PRE-TEST loop
- The looping condition is tested before executing the loop body.
- Loop body may not be executed at all.

Can we solve this problem?

- Consider the following program (input underlined):

```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.
```



Why the problem is hard

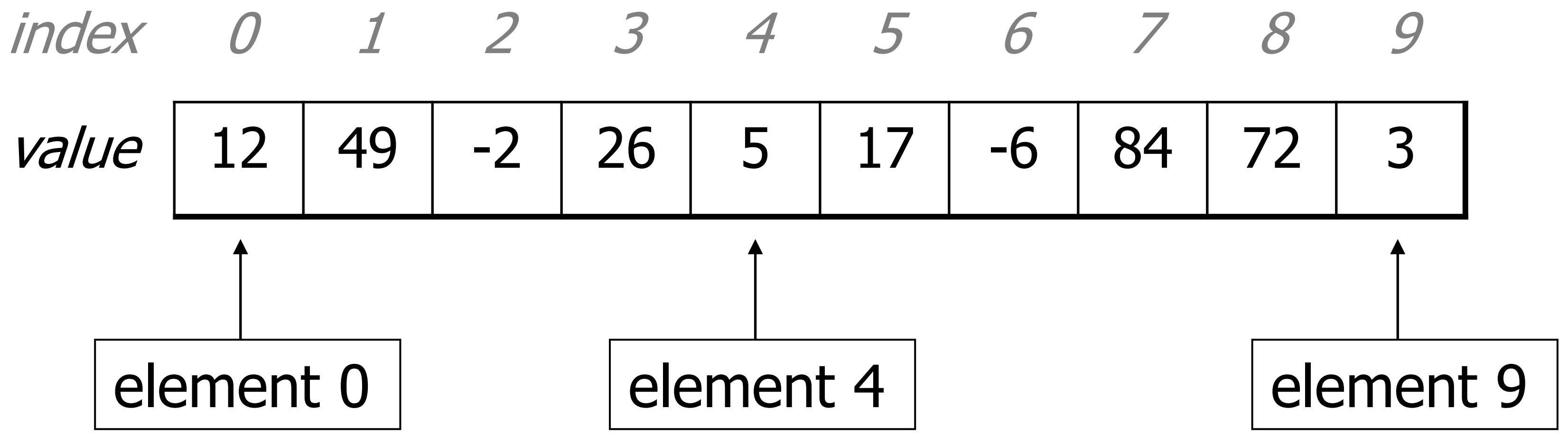
- We need each input value twice:
  - to compute the average (a cumulative sum)
  - to count how many were above average

- We could read each value into a variable... but we:
  - don't know how many days are needed until the program runs
  - don't know how many variables to declare

- We need a way to declare many variables in one step.

Arrays

- **array**: object that stores many values of the same type.
  - **element**: One value in an array.
  - **index**: A 0-based integer to access an element from an array.



Array declaration

type`[]` **name** `= new` **type**`[`**length**`];`

– Example:

```
int[] numbers = new int[10];
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	0	0	0	0	0	0	0	0	0	0

Array declaration, cont.

- The length can be entered by the user

```
Scanner console = new Scanner(System.in)
n = console.nextInt()
int[] numbers = new int[n];
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	0	0	0	0	0	0	0	0	0	0

Array declaration, cont.

- The length can be any integer expression.

```
int x = 2 * 3 + 1;
int[] data = new int[x % 5 + 2];
```

- Each element initially gets a "zero-equivalent" value.

Type	**Default value**
`int`	`0`
`double`	`0.0`
`boolean`	`false`
`String` or other object	`null` (means, "no object")

Accessing elements

```
name[index]                  // access
name[index] = value;         // modify
```

– Example:

```
numbers[0] = 27;
numbers[3] = -6;
```

```
System.out.println(numbers[0]);
if (numbers[3] < 0) {
    System.out.println("Element 3 is negative.");
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	**27**	0	0	**-6**	0	0	0	0	0	0

Arrays of other types

```
double[] results = new double[5];
results[2] = 3.4;
results[4] = -0.5;
```

index	*0*	*1*	*2*	*3*	*4*
value	0.0	0.0	**3.4**	0.0	**-0.5**

```
boolean[] tests = new boolean[6];
tests[3] = true;
```

index	*0*	*1*	*2*	*3*	*4*	*5*
value	false	false	false	**true**	false	false

Out-of-bounds

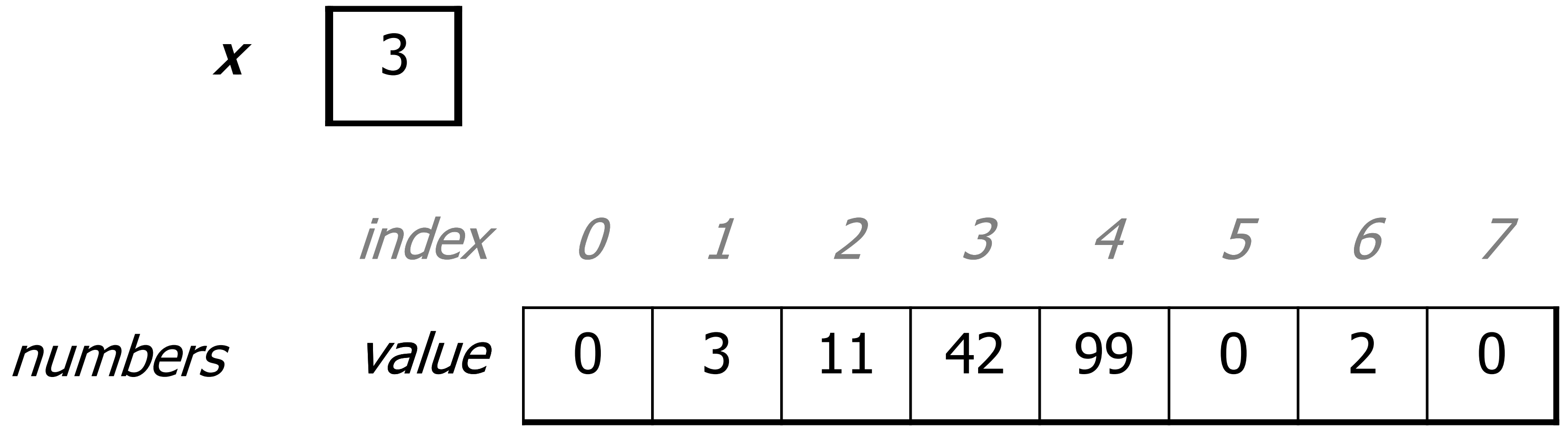
- Legal indexes: between **0** and the **array's length - 1**.
  - Reading or writing any index outside this range will throw an `ArrayIndexOutOfBoundsException`.
- Example:

```
int[] data = new int[10];
System.out.println(data[0]);        //
okay
System.out.println(data[9]);        //
okay
System.out.println(data[-1]);       //
exception
System.out.println(data[10]);       //
exception
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
value	0	0	0	0	0	0	0	0	0	0

Accessing array elements

```
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 99;
numbers[6] = 2;

int x = numbers[1];
numbers[x] = 42;
numbers[numbers[6]] = 11; // use numbers[6] as
index
```



Arrays and `for` loops

- It is common to use `for` loops to access array elements.

```
for (int i = 0; i < 8; i++) {
    System.out.print(numbers[i] + " ");
}
System.out.println();  // output: 0 3 11 42 99 0 2 0
```

- Sometimes we assign each element a value in a loop.

```
for (int i = 0; i < 8; i++) {
    numbers[i] = 2 * i;
}
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
value	0	2	4	6	8	10	12	14

Arrays and `for` loops, cont.

- It is common to use `for` loops to read array elements from the user.

```
for (int i = 0; i <= 9; i++) {
    System.out.print(“Enter element ” + i);
   number[i] = console.nextInt()
}
System.out.println();
```

The `length` field

- An array's `length` field stores its number of elements.

  name`.length`

  ```
  for (int i = 0; i < numbers.length; i++) {
      System.out.print(numbers[i] + " ");
  }
  // output: 0 2 4 6 8 10 12 14
  ```

  – It does not use parentheses like a String's `.length()`.

- What expressions refer to:
  – The last element of any array? **numbers[numbers.length - 1]**
  – The middle element? **numbers[(numbers.length - 1) / 2]**

Weather question

- Use an array to solve the weather problem:

```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average.
```

Weather answer

```java
// Reads temperatures from the user, computes average and # days above average.
import java.util.*;

public class Weather {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("How many days' temperatures? ");
        int days = console.nextInt();

        int[] temps = new int[days];          // array to store days' temperatures
        int sum = 0;

        for (int i = 0; i < days; i++) {      // read/store each day's temperature
            System.out.print("Day " + (i + 1) + "'s high temp: ");
            temps[i] = console.nextInt();
            sum += temps[i];
        }
        double average = (double) sum / days;

        int count = 0;                        // see if each day is above average
        for (int i = 0; i < days; i++) {
            if (temps[i] > average) {
                count++;
            }
        }

        // report results
        System.out.printf("Average temp = %.1f\n", average);
        System.out.println(count + " days above average");
    }
}
```

Quick array initialization

type[] **name** = {**value**, **value**, … **value**};

– Example:

```
int[] numbers = {12, 49, -2, 26, 5, 17, -6};
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	12	49	-2	26	5	17	-6

– Useful when you know what the array's elements will be

– The compiler figures out the size by counting the values