AP Computer Science A

Java Programming Essentials   [Ver. 3.0]

Unit 3: Basic Data Structures
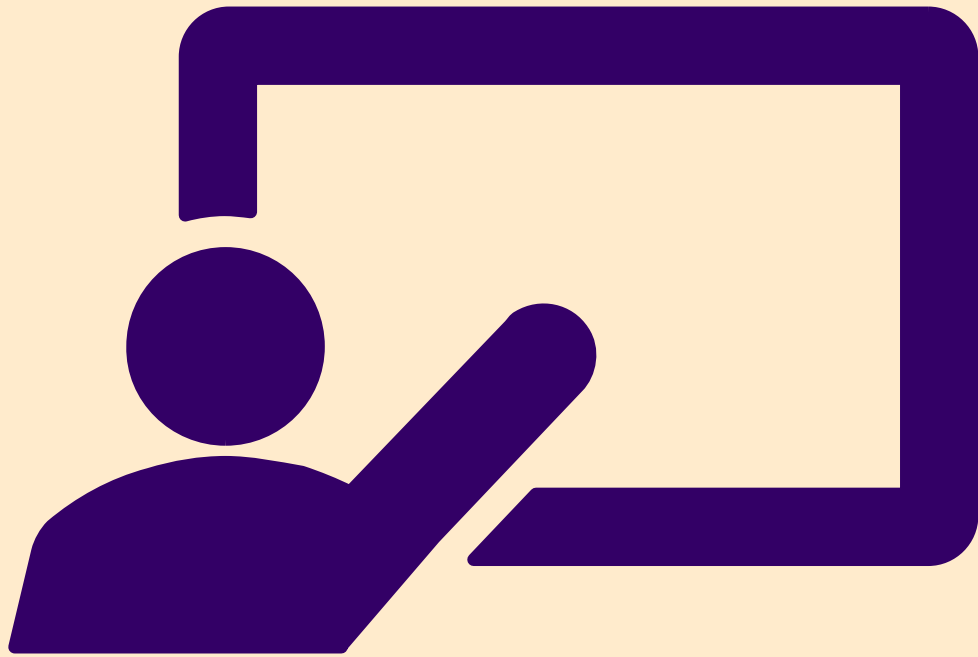
AI

CHAPTER 7A: ARRAYS

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Declaration – Instantiation – Initialization
- Basic Array Data Type (Reference Data Type)
- Array Processing I: traversal, assignments, finding max, min, sum, avg, difference, shift, rotation, Shuffling
- Array Processing II: reversal, transcopy, stepping, stack (tail-indexing), running indexing
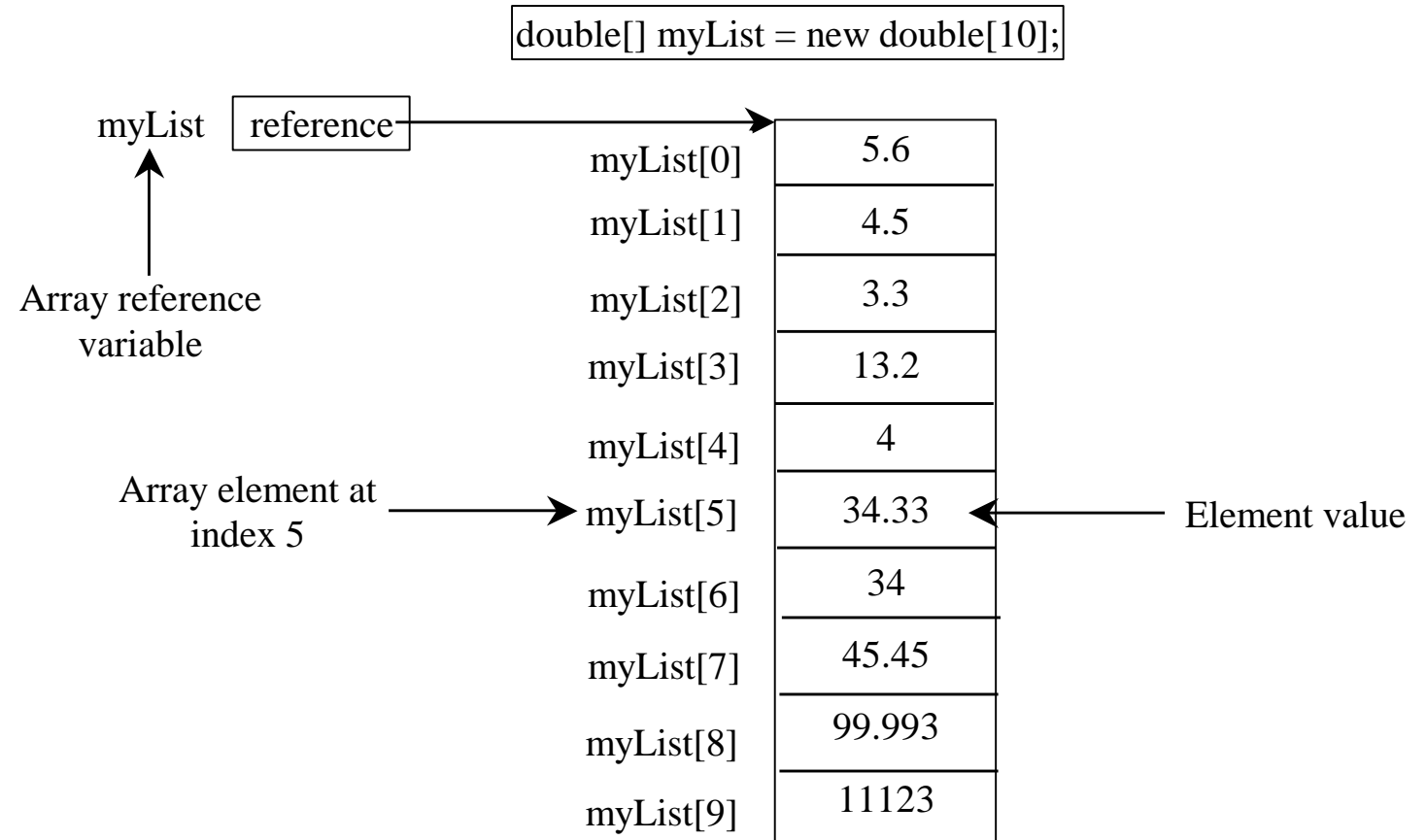
Array Basics

LECTURE 1

Introducing Arrays

- Array is a data structure that represents a collection of the same types of data.

double[] myList = new double[10];

myList | reference

Array reference variable

Array element at index 5

myList[0] 5.6
myList[1] 4.5
myList[2] 3.3
myList[3] 13.2
myList[4] 4
myList[5] 34.33 ← Element value
myList[6] 34
myList[7] 45.45
myList[8] 99.993
myList[9] 11123

Declaring Array Variables

- datatype[] arrayRefVar;

  Example:

  double[] myList;

- datatype arrayRefVar[]; <u>// This style is allowed, but not preferred</u>

  Example:

  double myList[];

Creating Arrays

arrayRefVar = new datatype[arraySize];

Example:

myList = new double[10];

myList[0] references the first element in the array.

myList[9] references the last element in the array.

Declaring and Creating in One Step

- datatype[] arrayRefVar = new datatype[arraySize];

    double[] myList = new double[10];

- datatype arrayRefVar[] = new datatype[arraySize];

    double myList[] = new double[10];

Declaration → Instantiation → Initialization
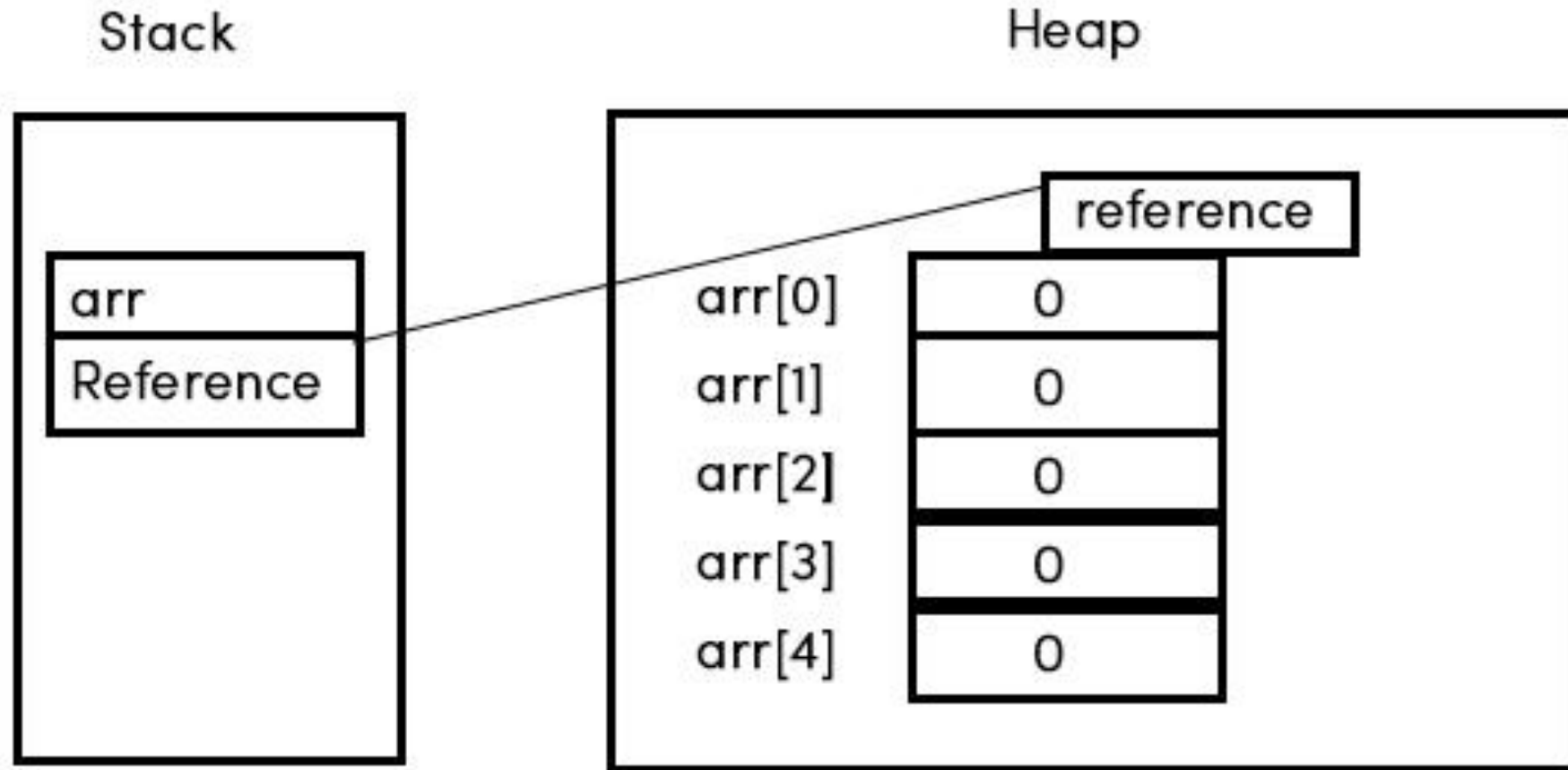
Declaration: Variable name with a data type

Instantiation: "New" keyword creates an object

Initialization: Call to a constructor, initializes new object

The Length of an Array

- Once an array is created, its size is fixed. It cannot be changed. You can find its size using

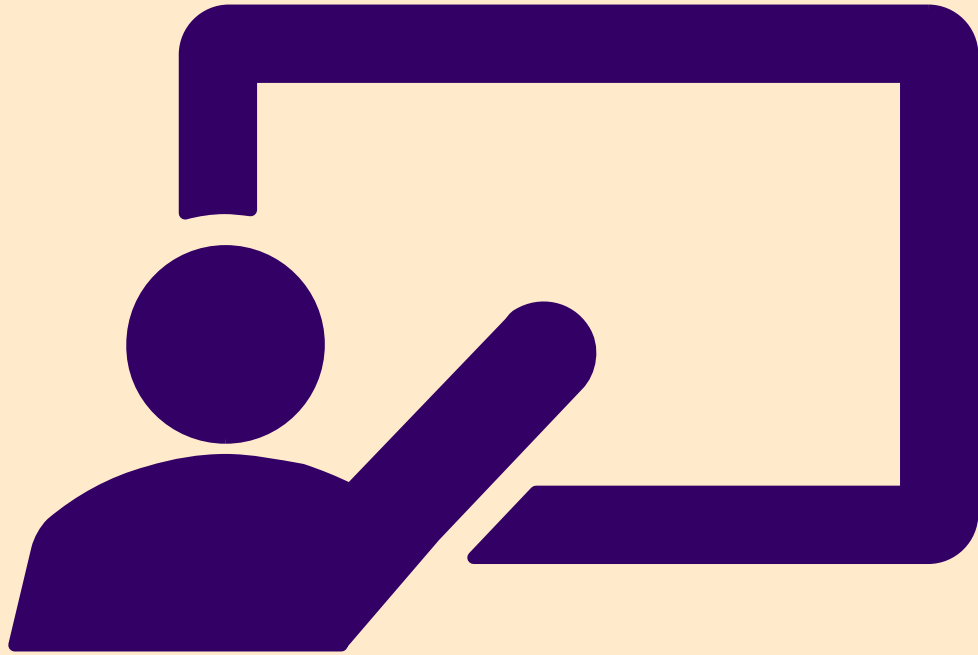 arrayRefVar.**length**

- For example,

  myList.length returns 10

Default Values

- When an array is created, its elements are assigned the default value of

  0 for the numeric primitive data types, '\u0000' for char types, and false for boolean types.

Index Variables

LECTURE 2

Indexed Variables

- The array elements are accessed through the index. The array indices are *0-based*, i.e., it starts from 0 to arrayRefVar.length-1. In the example in Figure 6.1, myList holds ten double values and the indices are from 0 to 9.

- Each element in the array is represented using the following syntax, known as an *indexed variable*:

    arrayRefVar[index];

Using Indexed Variables

- After an array is created, an indexed variable can be used in the same way as a regular variable. For example, the following code adds the value in <u>myList[0]</u> and <u>myList[1]</u> to <u>myList[2]</u>.

```
myList[2] = myList[0] + myList[1];
```

Array Initializers

- Declaring, creating, initializing in one step:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

- This shorthand syntax must be in one statement.

```
myList = {2.9, 3.9, 4.4, 4.5}; // Compilation Error.
```

Declaring, creating, initializing Using the Shorthand Notation

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

- This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];
myList[0] = 1.9;
myList[1] = 2.9;
myList[2] = 3.4;
myList[3] = 3.5;
```

CAUTION

- Using the shorthand notation, you have to declare, create, and initialize the array all in one statement. Splitting it would cause a syntax error. For example, the following is wrong:

```
double[] myList;

myList = {1.9, 2.9, 3.4, 3.5};
```

Trace Program with Arrays

Declare array variable values, create an array, and assign its reference to values

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

i becomes 1

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
     values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

i (=1) is less than 5

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

After this line is executed, value[1] is 1

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

After i++, i becomes 2

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

```java
public class Test {
  public static void main(String[]
      args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] +
      values[4];
  }
}
```

i (= 2) is less than 5

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

After this line is executed,
values[2] is 3 (2 + 1)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this, i becomes 3.

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

i (=3) is still less than 5.

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this line, values[3] becomes 6 (3 + 3)

```
public class Test {
  public static void main(String[] args)
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

After this, i becomes 4

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

i (=4) is still less than 5

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

After this, values[4] becomes 10 (4 + 6)

```java
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

After i++, i becomes 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5, i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

i (=5) < 5 is false. Exit the loop

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

After this line, values[0] is 11 (1 + 10)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

0	11
1	1
2	3
3	6
4	10

Demo Program

LECTURE 3

Demo Program: Analyzing Numbers

- The problem is to read 100 numbers, get the average of these numbers, and find the number of the items greater than the average. To be flexible for handling any number of input, we will let the user enter the number of input, rather than fixing it to 100.

java.util.Scanner; // **project**, **package**, **class (module)**

Demo Program: Analyzing Numbers

(1) Build an array of **specific** length:

```
System.out.print("Enter the number of items: ");
// class.stream.method
int n = input.nextInt();
double[] numbers = new double[n];
```

Note:

   Array size assignments: from console input,
                                from random number,
                                form an int literal,
                                from a constant, and from a variable.

Demo Program: Analyzing Numbers

(2) Find the average:

```
double sum = 0;
System.out.print("Enter the numbers: ");
for (int i = 0; i < n; i++) {
  numbers[i] = input.nextDouble();
  sum += numbers[i];
}
double average = sum / n;
```

Demo Program: Analyzing Numbers

(3) Find the count of numbers greater than average:

int count = 0; // The numbers of elements above average

for (int i = 0; i < n; i++)

if (numbers[i] > average)

count++;

(4) Output:

System.*out*.println("Average is " + average);

System.*out*.println("Number of elements above the average is "+ count);

Demonstration Program

ANALYZINGNUMBERS.JAVA

Swap of Two Integers

Rotational Swap

int tmp = a;

a = b;

b = tmp;

Swap by buffering

int c = a;

Int d = b;
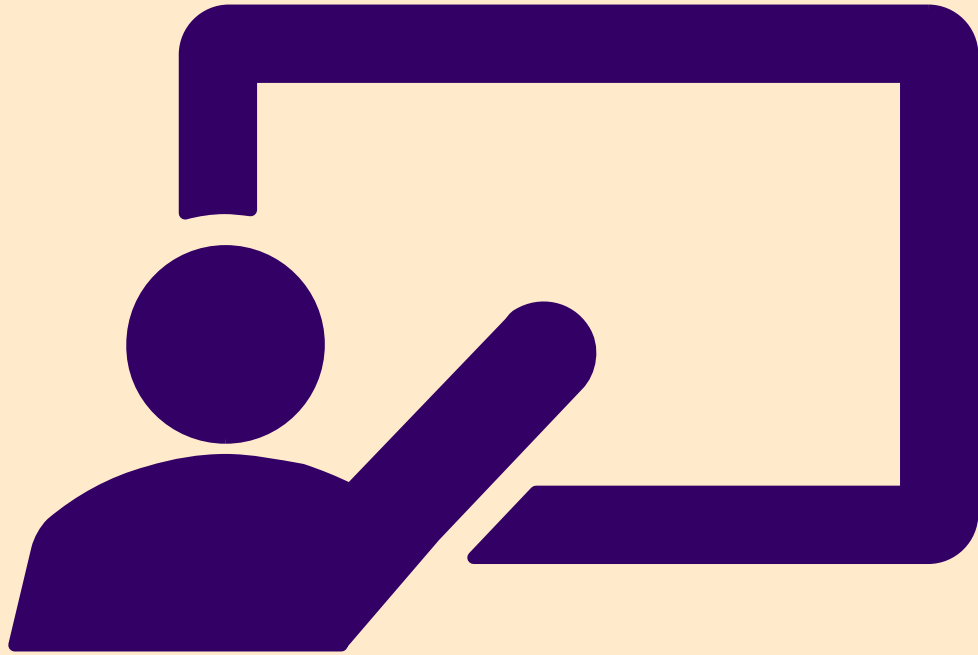
a = d;

b = c;

In-Class Demonstration Program

SWAPPING TWO INTEGERS

Array Processing I

LECTURE 5

Processing Arrays

1. (Initializing arrays with input values)
2. (Initializing arrays with random values)
3. (Printing arrays)
4. (Summing all elements)
5. (Finding the largest element)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)
8. (*Shifting elements*)

Initializing arrays with input values

```java
java.util.Scanner input = new java.util.Scanner(System.in);

System.out.print("Enter " + myList.length + " values: ");

for (int i = 0; i < myList.length; i++)
  myList[i] = input.nextDouble();
```

Initializing arrays with random values

```
for (int i = 0; i < myList.length; i++) {
  myList[i] = Math.random() * 100;
}
```

Printing arrays

```java
for (int i = 0; i < myList.length; i++) {
  System.out.print(myList[i] + " ");
}
```

Summing all elements

```
double total = 0;
for (int i = 0; i < myList.length; i++) {
  total += myList[i];
}
```

Finding the largest element

```
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
  if (myList[i] > max) max = myList[i];
}
```

Finding the largest element (2nd Version)

```
double max = Double.MIN_VALUE;
for (int i = 0; i < myList.length; i++) {
  if (myList[i] > max) max = myList[i];
}
```

Random shuffling

```java
for (int i = 0; i < myList.length; i++) {
  // Generate an index j randomly
  int index = (int)(Math.random()
    * myList.length);

  // Swap myList[i] with myList[index]
  double temp = myList[i];
  myList[i] = myList[index];
  myList[index] = temp;
}
```
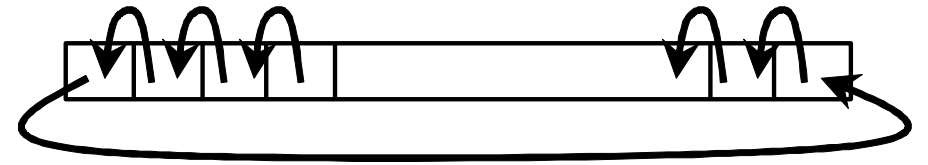
myList

i ⟶ [0]

[1]

.
.
.

swap

[index]

A random index
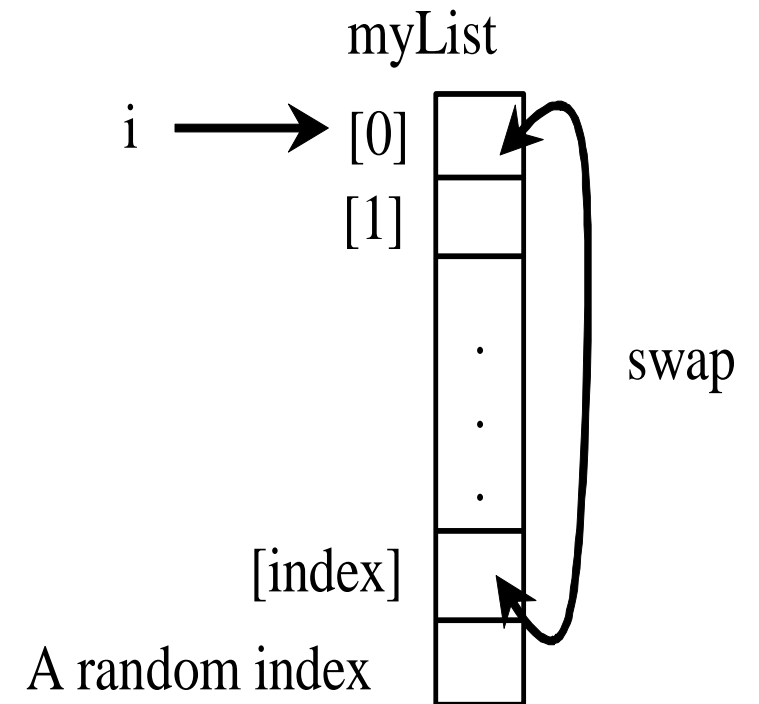
Shifting Elements (Left Shifting)

```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```
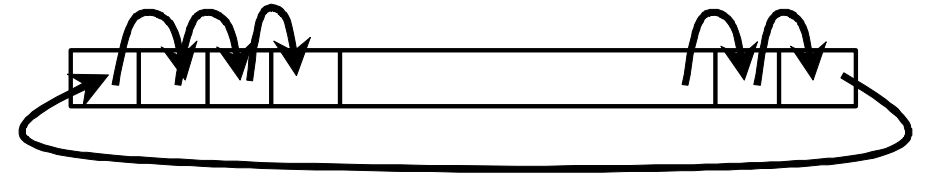
myList

Shifting Elements (Right Shifting)

```
double temp = myList[myList.length-1]; // Retain the last element

// Shift elements left
for (int i = myList.length-2; i >=0; i--) {
  myList[i + 1] = myList[i];
}

// Move the last element to fill in the first position
myList[0] = temp;
```
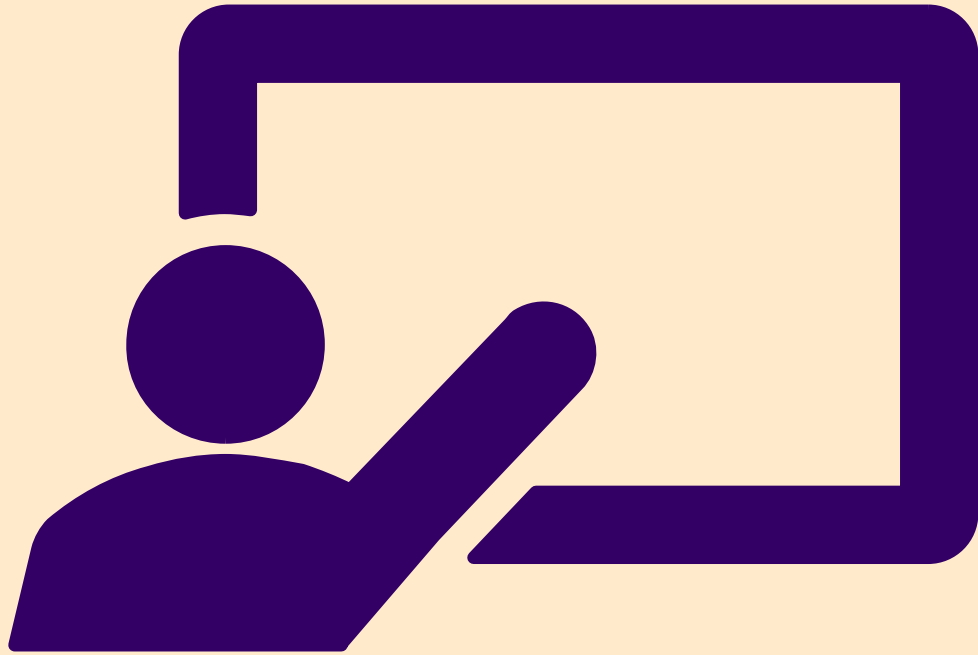
myList

Demonstration Program

ARRAYPROCESSINGI.JAVA

Array Processing II

LECTURE 6

Array Processing II

1. Digits Reverse
2. String Reverse
3. Array Reverse
4. Running Index
5. Transcopy
6. Reverse of Array by Running Index
7. Basic Statistics Methods

Digits Reversal

```java
public static void reverseOfInteger(){
    int x = 34567;
    int y = x;
    int reverseX = 0;
    while (x != 0){
        int d = x % 10;
        reverseX = reverseX * 10 + d;
        x = x /10;
    }
    System.out.println("Reverse Digits of " + y + " is " + reverseX);
}
```

Iterations	X (end of loop)	reverseX
1	3456	7
2	345	76
3	34	765
4	3	7654
5	0	76543

String Reverse

```java
public static void reverseOfString(){
    String x = "ABCDE";
    String y = x;
    String reverseX = "";
    while (x.length() != 0){
      reverseX += x.charAt(x.length()-1);
      x = x.substring(0, x.length()-1);
    }
    System.out.println("Reverse String of " + y + " is " + reverseX);
  }
```
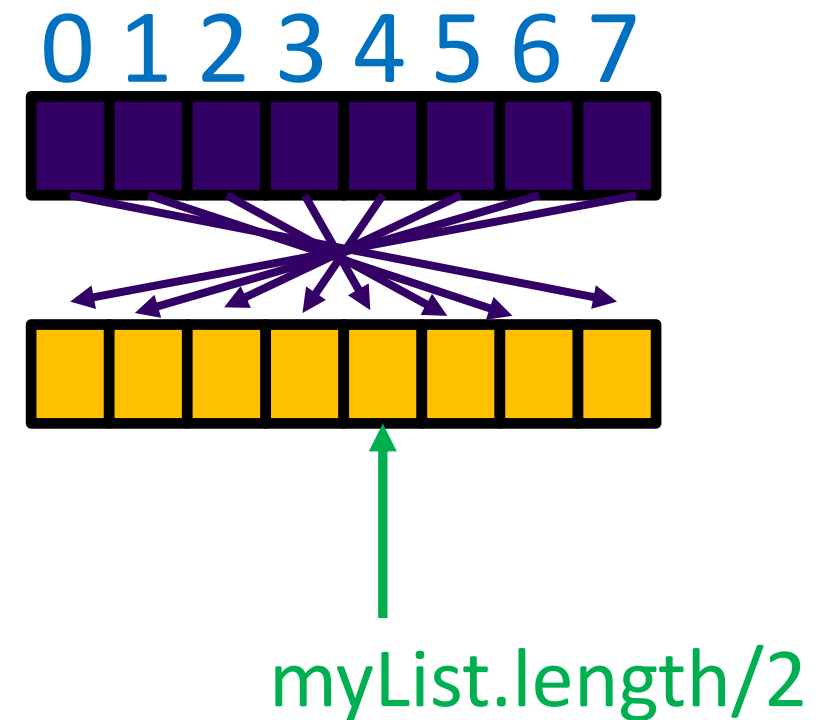
Reverse of an Array

```java
public static void reverse(double[] myList){
    double tmp = 0.0;
    for (int i=0; i<(myList.length/2); i++){
        tmp = myList[i];
        myList[i] = myList[myList.length-1-i];
        myList[myList.length-1-i] = tmp;
    }
}
```

0 1 2 3 4 5 6 7

myList.length/2

Demonstration Program

ARRAYPROCESSINGII.JAVA

Running Index

- An array can be traversed by a running index.

- A running index is an integer which update itself by increment (p++) or decrement (--p) after access an array element.

- A running index can be used for traversal which always start from 0.

- A running index can also be used to keep track of the next available empty spot.

Simple Traversal

```java
public static double[] list = {1.0, 2.0, 3.0, 4.0, 5.0,
                                6.0, 7.0, 8.0, 9.0, 10.0};
```

```java
public static void main(String[] args){
    System.out.print("\f");
    int p=0;
    while (p<list.length){
        System.out.println(list[p++]);
    }
}
```

Demonstration Program

AP2_RUNNINGINDEX.JAVA

Transcopy

- Transcopy is to copy a part of an array to another array at different index location.

- Running index is a good technique to make such copy operations.

Demonstration Program

AP2_TRANSCOPY.JAVA

Reverse of Array by Running Index

- Running Index is very useful in Stack operations. For the time being, it is hard to understand what stack is. Yet, we can look at the reverse of an array by using running index.

Demonstration Program

AP2_ARRAYREVERSAL.JAVA

Demo Program: Statistics
Stats02.java

LECTURE 7

Basic Statistics Methods

Sum: sum of a numerical data set.

Avg: average of a numerical data set.

Max: maximum of a numerical data set.

Min: minimum of a numerical data set.

Range: the difference between the maximum and minimum of a numerical data set. It is the range that data are located.

Median: The middle value of a sorted data set. If the number of items is even, then take average of the two number around the middle.

Mode: The item which occurs the most times in a data set.

Finding mode of a data set

myList: data set to be processed.

myMode: number of occurrence for the value myList[i] (element stored in index i location of myList.)

myModexIndex: the last occurrence index of a certain value.

Pseudo Code for mode

mode (myList: the data set to find mode with){
      create occurrence array myMode of same length as myList;
      create last occurrence index array myModeIndex.
      (for-loop i) calculate the **frequency count** for each element in myList.
            (for-loop j) **count** for the frequency of the value of elements.
      (call max) find the **maximum** of the frequency counts.
      (for-loop k) locate the **index** of the last occurrence of the mode and
            the mode value.
      return **mode value**.
}

Demonstration Program

STATS02.JAVA

Command Line Argument List

LECTURE 8

Command Line Arguments

It is a String argument array.

```java
public class A {
    public static void main(String[] args){
        for (int i=0; i<args.length; i++)
            System.out.println(args[i]);
    }
}
```

A main method is just a regular method. Furthermore, you can pass arguments from the command line..

Command Line Arguments

It is a String argument array.

Usually, it is named argv, or args.

```java
public class TestMain {
    public static void main(String[] args){
        String[] strings=
                {"New York","Boston","Atlanta"};
        A.main(strings);
    }
}
```

A main method is just a regular method. Furthermore, you can pass arguments from the command line..

Passing Strings to the Main Method

In the Windows Command Line:

C:> java A arg0 arg1 arg2

Arg0, arg1, arg2 are strings, but they don't have to appear in double quotes on the command line. The strings are separated by space. A string that contains a space must be enclosed in double quotes. Consider the following command line:

Java A "First num" alpha 53

Windows will pass "First num" as one String argument. Therefore, this command line returns three strings: First num, alpha and 53

Demonstration Program

TESTMAIN.JAVA+A.JAVA

Demonstration Program

ARGSARRAYTESTER.JAVA