CS1150
Principles of Computer Science
Arrays

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

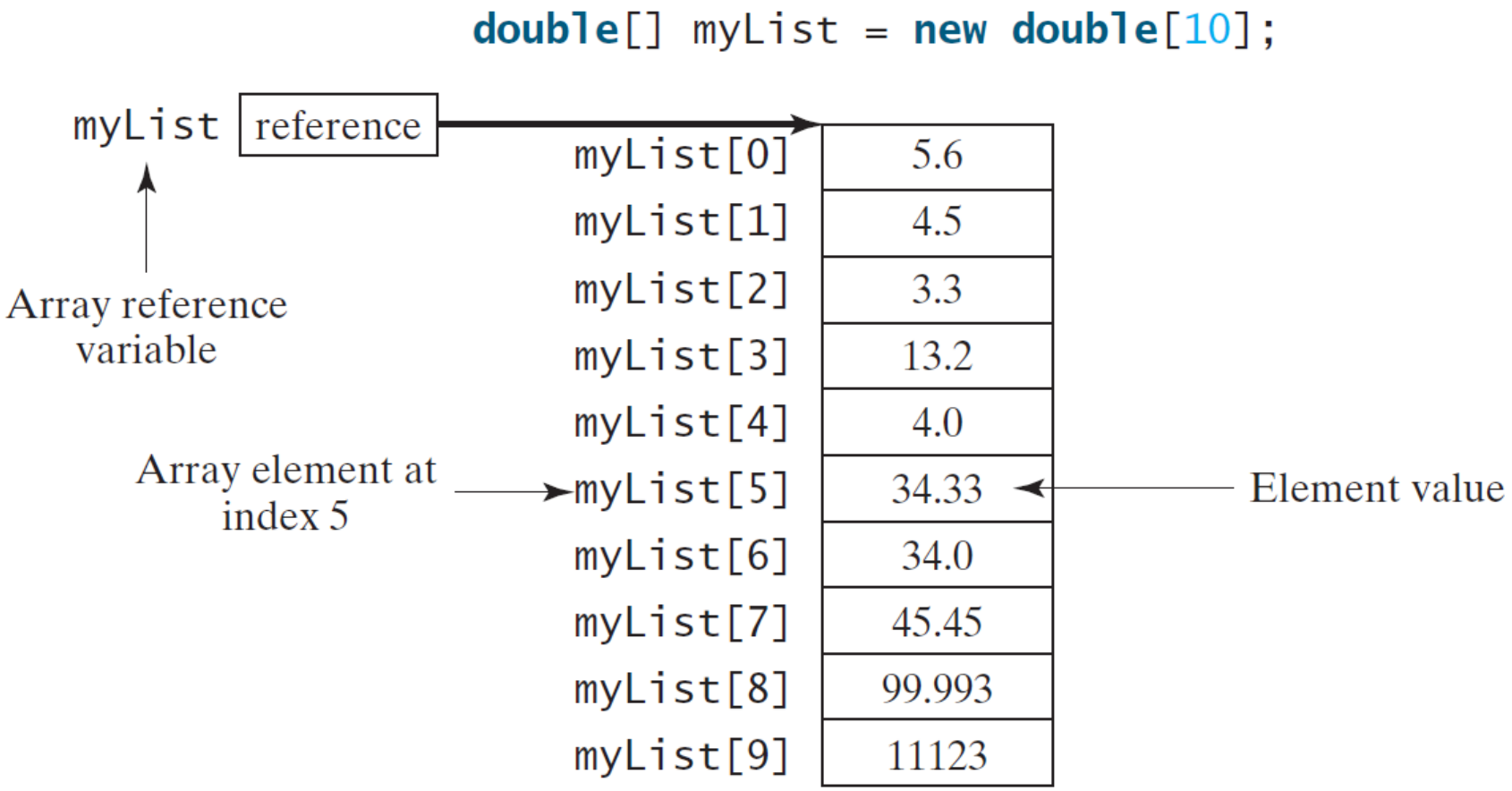
Opening Problem

Read one hundred numbers, compute their average, **and** find out how many numbers are above the average.

Need to find a place to hold all the numbers

Introducing Arrays

Array is a data structure that represents a collection of the **same type** of data.



Declaring Array Variables

- `datatype[] arrayRefVar;`

Any type (int, float, double, etc.)
All elements in array will be of this type

Variable used to access the array elements
It is considered a "reference" variable

Example:

```
double[] myList;
```

- `datatype arrayRefVar[];` // Allowed, but not preferred

Example:

```
double myList[];
```

Beware: when declaring array

- Primitive Types (int, long, float, double etc.)
  - Declare an integer -> int number ;
  - The declaration indicates
    - **Type of data** you are declaring, and
    - **Allocates memory**
- Array Types
  - Declare an array -> int[] arrayRefVar;
  - The declaration indicates
    - **Type of data** the array will store, but
    - **DOES NOT allocate memory**
  - Declaration allocates memory for a pointer/reference only
    - i.e., memory to store an address to the beginning of an array

Creating Arrays

Cannot do anything with an array variable until after the array has been constructed with the **new** operator:

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

Example:

(1) creates an array with 10 double (i.e. it allocates memory)

```
myList = new double[10]; //use new to give a size
                         //allocate memory for array
```

(2) assigns the reference of the array to the variable myList

`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];
```

Example

- Declare an array that holds 5 integers

```
int[] numberList;      // Declares an array numberList - note that
                       // 1) the size of the array is not set here
                       // 2) no memory has been allocated
numberList = new int[5];     // Creates an array with 5 integers
                             // assigns it to reference numberList
```

- Rewritten as one step

```
int[] numberList = new int[5];
```

Example: Arrays1.java

The Length of an Array

Once an array is created, its **size is fixed.** The length of the array is obtained by access the "length" property:

myList.length

For example,

myList.length returns 10

The Length of an Array

- The length of the array is obtained by access the "length" property
  - "length" is a property of an array object
    - Established when the array is created
    - Cannot change after array is created: length is a constant
  - We would expect a method, like length(), but for arrays we directly access the value

```
// Print the length of the array
System.out.println ("The length of the array numberList = " +
numberList.length); // Example: Arrays2.java
```

Default Values

Unlike other types, when an array is created, its elements are assigned the default value of

zero for the numeric primitive data types,

'\u0000' (Unicode for null) for char types, and

false for boolean types.

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.

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 myList[0] and myList[1] to myList[2].

```
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.

Note **new** is *not* used in this approach.

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};
```

Example: Array3

Accessing arrays

- For loops generally used with arrays since we know how many times the loop will occur
  - Example: assign the numbers 0 to 4 to numberList

```
// Assign the numbers 0 to 4 to numberList array
int[] numberList = new int[5];
for (int i = 0; i < numberList.length; i++) {
        numberList[i] = i;
        System.out.println("numberList[" + i + "] = " + numberList[i]);
}
```

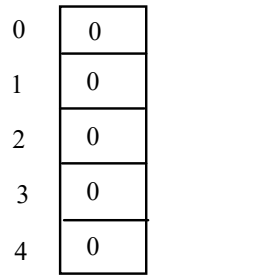
- Trying to access an element outside the range of an array it is an error

Trace Program with Arrays

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

```
public class Test {
  public static void main(Stri[] 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



Trace Program with Arrays

i becomes 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 array is created

0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

i (==1) is 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 array is created

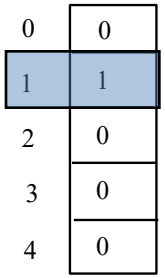
0	0
1	0
2	0
3	0
4	0

Trace Program with Arrays

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

```
public class Test {
  public static void main(Strin      gs) {
    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



Trace Program with Arrays

```
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 i++, i becomes 2

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

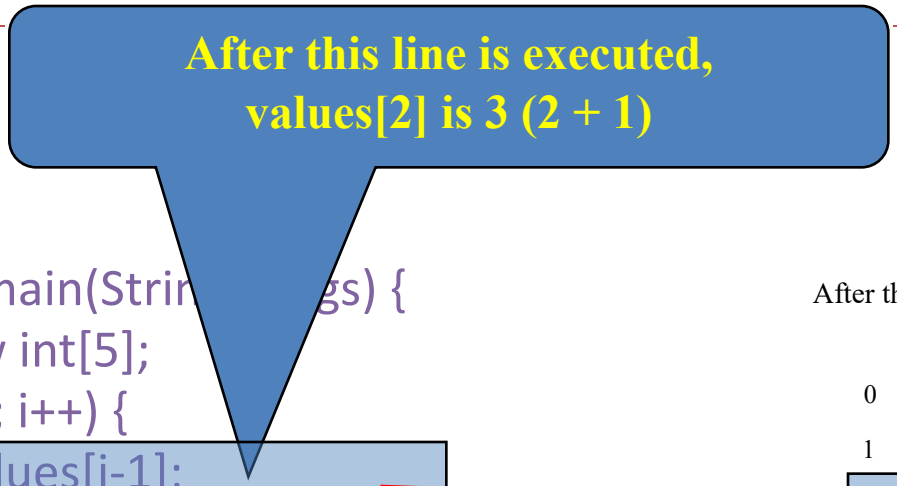
```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

i (== 2) is less than 5

After the first iteration

0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays




```
public class Test {
  public static void main(Strin      gs) {
    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

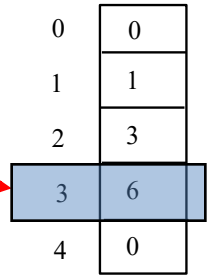
0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

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

After the third iteration



Trace Program with Arrays

After this, i becomes 4

```
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

```
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 (==4) is still less than 5

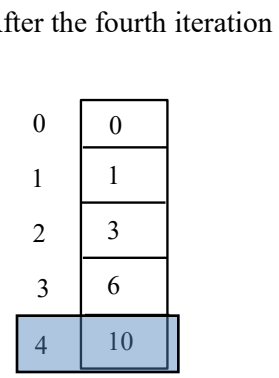
After the third iteration

0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

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

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



Trace Program with Arrays

```
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 i++, i becomes 5

After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

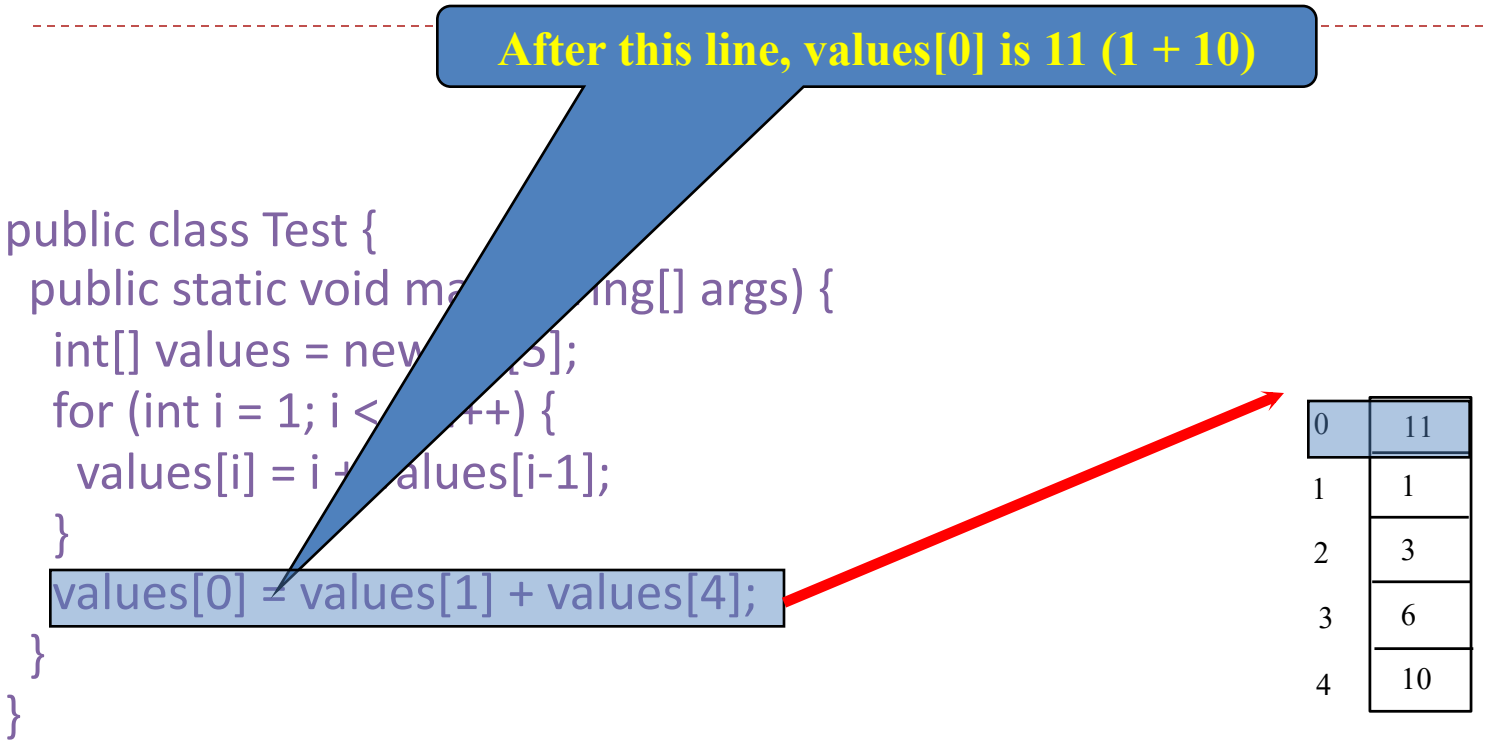
i (==5) < 5 is false. Exit the loop

```
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



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

```
double[] myList = new double[5];

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

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

Example: Array4

Foreach loops

- Shortcut: Sequentially processes loop without stating at an index
  - "For each value in my numbers array, do something with it"
  - DO NOT specify index in the loop
  - Example: Displaying array numbers using a foreach loop

```
for (int i: numbers) {
   System.out.println ("value = " + i);
}
```

In general, the syntax is

```
for (elementType value: arrayRefVar) {
  // Process the value
}
```

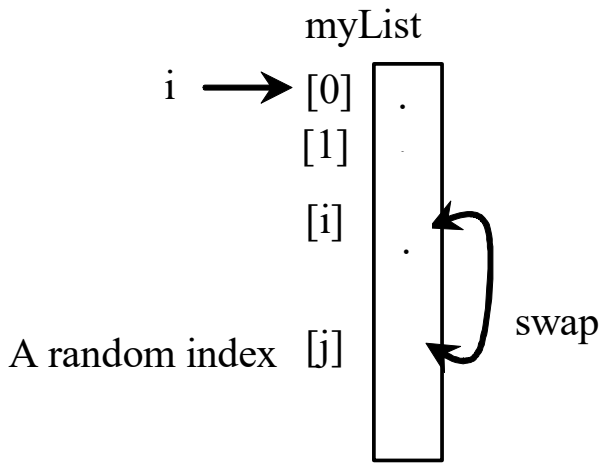
  - Can only move through array sequentially from 1st element to last

Example: Array5

Random shuffling

```
for (int i = 0; i < myList.length - 1; i++) {
  // Generate an index j randomly
  int j = (int)(Math.random()
    * myList.length);

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



Shifting Elements

```
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;
```