ICS 111
Introduction to Computer Science I

Nikki Manuel
Leeward Community College

Arrays (cont.)

Week 9
Fall 2019

Processing Arrays

Processing Arrays

- When processing arrays, we will probably be using:
  - Counters
    - To represent indexes in the array
  - Accumulators
    - To build Strings and add numbers
  - Sentinel Values
    - To exit out of a loop early

Processing Arrays

- When processing arrays, we will often use a `for` loop
  - Why?
    - All of the elements in the array are of the same type
    - We know the size of the array, so it’s natural to use `for`

Initializing Arrays with Input Values

Initialize the array **`myArray`** with user input values:

```
Scanner reader = new Scanner(System.in);
System.out.print("Enter " + myArray.length +
   " values: ");
for (int i = 0; i < myArray.length; i++) {
   myArray[i] = input.nextDouble();
}
```

Initializing Arrays with Random Values

Initialize the array **`myArray`** with random values between 0.0 and 100.0, but less than 100.0:

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

Displaying Arrays

To print an array, you have to print each element in the array using a loop like the following:

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

Displaying `char` Arrays

Note: For an array of the **`char`**[] type, you can print using one print statement. For example, the following code displays **`apple`**

```
char[] fruit = {'a', 'p', 'p', 'l', 'e'};
System.out.println(fruit);
```

Summing All Elements in an Array

Use a variable named `total` to store the sum of all elements in an array. Initially, `total` is `0`. Add each element in the array to `total` using a loop like this:

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

Let’s Try This : Fill Up an Array

– Read Strings to add to an array, but store only those that have length of 3 or more

– STOP when 10 elements have been read
OR
– STOP if the user enters an empty String
  - An empty String is a String of length zero

Searching Arrays

- As humans we can see everything at once
- Computers must do things sequentially

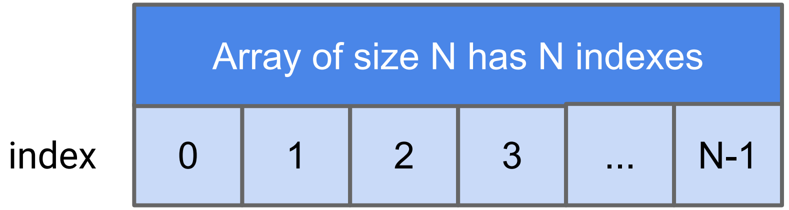
21	4	89	54	787	1	36

Let’s Try This : Searching an Array

- Use a **`while`** loop to search for a number
  - While: the number is not found AND there are still elements we need to search
- If the number is found, print the index and a message
- If the search ends and the number was not found, print a message
- BUT WAIT! Are we looking for one instance or as many as we can find?

ArrayIndexOutOfBoundsException

– Remember that an array of size N has indexes from `0` to **`N-1`**



– If you try to access an index that is not between 0 and N-1, you will get an exception: ArrayIndexOutOfBoundsException

Test Yourself : Arrays #1

Declare an array reference variable for an array of boolean type.

Create the array so it is able to hold 5 elements.

Fill all of the array elements with values.

Test Yourself : Arrays #2

What is the output of the following code?

```
int x = 5;
int[] myArray = new int[x];
x = 100;
System.out.println("x is " + x);
System.out.println("The size of myArray is: " +
    myArray.length);
```

Test Yourself : Arrays #3

True or False?

- Every element in an array has the same type.
- The array size is fixed after an array reference variable is declared.
- The array size is fixed after it is created.
- The elements in an array must be a primitive data type.

Test Yourself : Arrays #4

How do you access elements in an array?

What is the lowest index of an array?

How do we represent the third element in an array named myArray?

What happens if your program tries to access an array element with an invalid index?

Copying, Searching, and Sorting Arrays

Copying Arrays

To copy the contents of an array, use a loop to copy elements one by one into a different array:

```
int[] array1 = {1, 11, 6, 1, 6, 22};
int[] array2 = new int[array1.length];
for (int i = 0; i < array1.length; i++) {
    array2[i] = array1[i];
}
```

Test Yourself : Copying Arrays

You have created the following array:

```
    String[] groceries = {"milk", "candy", "kale"};
```

However, you realize that you need to add a few more things to your grocery list!

Create an array of size 10 called `groceryList` and copy the elements from your first array into your new array.

Searching Arrays

When we **search** an array, we are looking for a specific element in an array. This is a common task in computer programming!

Two common ways to search arrays:

- Linear search
- Binary search

Linear Search

With linear search, we look through the array sequentially until we find the element **`x`** that we want.

```
for (int i = 0; i < myArray.length; i++) {
    if (x == myArray[i]) {
        System.out.println(i);
        break;
}
```

Use `break;` to exit out of the loop!

Binary Search

Binary search assumes that elements in a list are already ordered. Assume that the array is in ascending order.

For the element **`x`** that we want, we first compare it to the element in the middle of the array.

- If **`x`** is less than the middle element, search for **`x`** in the first half of the array.
- If **`x`** is greater than the middle element, search for **`x`** in the second half of the array.

Binary Search – How To

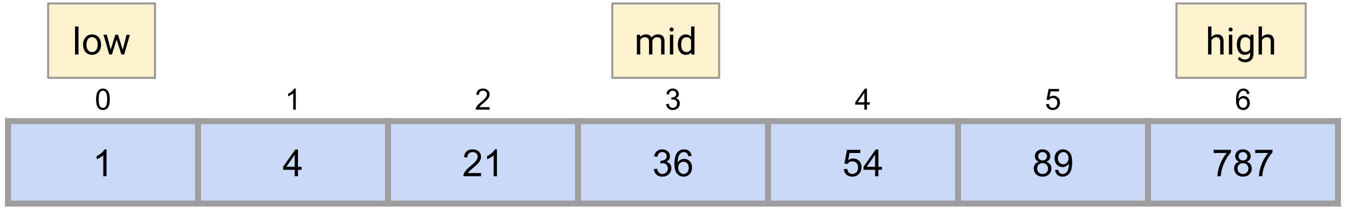
Binary search eliminates at least half the array after each comparison. This can make it more efficient than linear search!

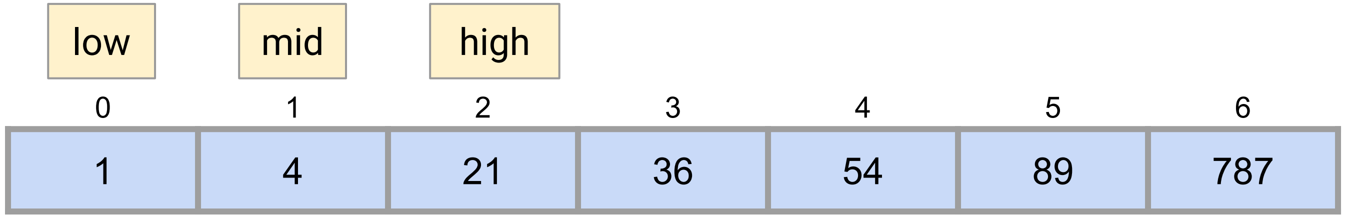
We will use the following values:

- **`low`** : first index of the array currently being searched
- **`high`** : last index of the array currently being searched
- **`mid`** : the index of the middle element : **`(low + high) / 2`**

Binary Search – Example 1

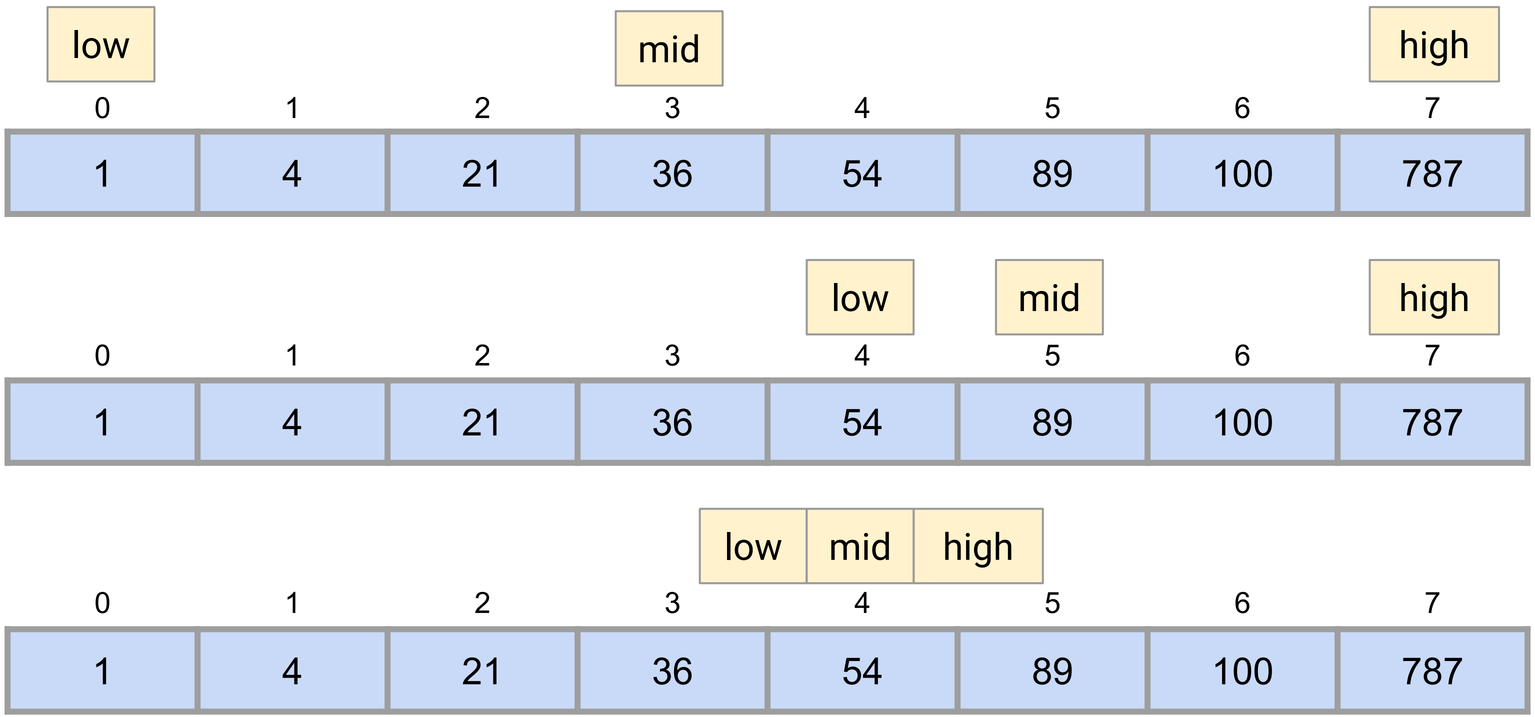
x = 4





Binary Search – Example 2

x = 54



```
int low = 0
int high = myArray.length - 1;

while (high >= low) {
    int mid = (low + high) / 2;
    if (x < myArray[mid])
        high = mid - 1;
    else if (x == myArray[mid]) {
        System.out.println(mid);
        break;
    }
    else
        low = mid + 1;
}
```

Test Yourself – Binary Search

Similar to the Binary Search - Example slides, draw out the steps to show how binary search is applied to search for the element `10` in the following array:

```
{2, 4, 7, 10, 11, 45, 53, 59, 60, 66, 69, 77}
```

Sorting Arrays

When we **sort** an array, we are rearranging the elements in an array. This is also a common task in computer programming!

Selection sort sorts an array in ascending order

- Finds the smallest number
- Swaps the smallest number with the 1st element
- Finds the next smallest number
- Swaps it with the 2nd element, and so on…
- … until only a single number remains

Selection Sort – Example

21	4	787	54	89	Swap 4 (the smallest) with 21 (the first)
4	21	787	54	89	No swap necessary
4	21	54	787	89	Swap 54 with 787
4	21	54	89	787	Swap 89 with 787

```java
for (int i = 0; i < myArray.length - 1; i++) {
    // Finds the minimum in the array
    double currentMin = myArray[i];
    int currentMinIndex = i;
    for (int j = i + 1; j < myArray.length; j++) {
        if (currentMin > myArray[j]) {
            currentMin = myArray[j];
            currentMinIndex = j;
        }
    }
    // Swaps myArray[i] with myArray[currentMinIndex]
    if (currentMinIndex != 1) {
        myArray[currentMinIndex] = myArray[i];
        myArray[i] = currentMin;
    }
}
```

Test Yourself – Selection Sort

Use the code in the previous slide to trace the steps of Selection Sort in order to sort the following array:

```
{3.4, 5, 3, 3.5, 2.2, 1.9, 2}
```