Introduction to java

COMP1030
Lecture #7

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes during class.
- Weekly optional tutorials take place every week on Wednesdays, from 3-4pm in room N111 (note new location)

Housekeeping

- Assignment #1 Answer key is posted under the assignments link
- Misterm-Exam question paper and answer key are posted under course information link
- Mid-term exams can be photographed during todays lab
- Labs will no longer be checked as homework
- Students will be expected to work in a lab group – be prepared to change seats for lab

Review

- java packages
- static methods and fields
- concatenation
- Method invocation
- Method call stack
- Scope
- Method overloading

Arrays - Introduction

- A data structure is a collection of related data items.
- Arrays are objects that represent data structures consisting of related data items of the same type.
- Arrays are a convenient way to process related groups of values.

Arrays - Introduction

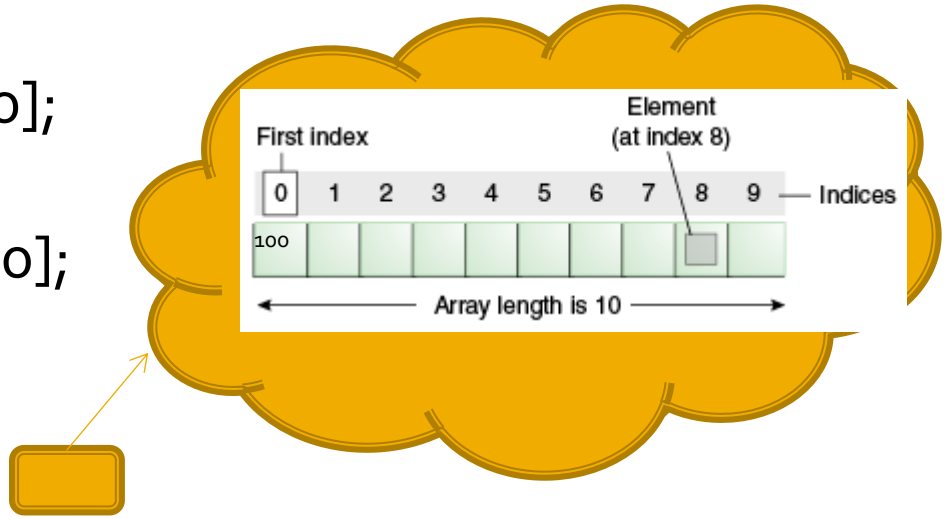
- An array is a group of variables (elements) containing values of the **same type**.
- Arrays are objects which can hold either primitive or reference types.
- To refer to a specific element of an array we must use the reference to the array and the position number of the element in the array.

Arrays - Introduction

- The position number of the element is called the elements index.
- Arrays remain the same length once they are created.

Array Instantiation

```
int [] myArray = new int [10];
or
int  myArray [] = new int [10];
```




```
myArray [0] = 100;
myArray [1] = 200;
.
.
.
myArray[9] = 1000;
System.out.println("Element at index 0:"  + myArray[0]);
```

Array Rules

- An index cannot be negative.
- An array element can be an expression
  myArray[a +b] – assuming a and b are valid integers

- An index must be an int or type that can be promoted to an int: byte, short, char

- myArray.length gives the arrays length as it is an instance variable of the array object.

- We can use array elements in a mathematical expression:

```
sum = myArray[0] + myArray[1];
```

Declaring and Populating Arrays

- When an array is created each element is set to the default value for that type.

```
int [ ] c = new int [12];
```

or

```
int [ ] c;
c = new int [12];
```

or

```
int [ ] c = {100,200,300,400};*
```

* the length of the array is determined by the # of elements in the list.

Declaring and Populating Arrays

String b [ ] = new String [100];

This creates an array of references to 100 string objects. Therefore, b[0] is a reference to a String object.

We can use this reference b[0] to invoke methods within the object.

Sample Code

```
class ArrayDemo
{
        public static void main(String[] args)
        {
                int[ ] anArray;
                anArray = new int[10];
                anArray[0] = 100;
                anArray[1] = 200;
                anArray[2] = 300;
                anArray[3] = 400;
                anArray[4] = 500;
                anArray[5] = 600;
                anArray[6] = 700;
                anArray[7] = 800;
                anArray[8] = 900;
                anArray[9] = 1000;
                System.out.println("Element at index 0: " + anArray[0]);
                System.out.println("Element at index 1: " + anArray[1]);
                System.out.println("Element at index 2: " + anArray[2]);
                System.out.println("Element at index 3: " + anArray[3]);
                System.out.println("Element at index 4: " + anArray[4]);
                System.out.println("Element at index 5: " + anArray[5]);
                System.out.println("Element at index 6: " + anArray[6]);
                System.out.println("Element at index 7: " + anArray[7]);
                System.out.println("Element at index 8: " + anArray[8]);
                System.out.println("Element at index 9: " + anArray[9]);
        }
}
```

Sample Code

Output looks like this

```
Element at index 0: 100
Element at index 1: 200
Element at index 2: 300
Element at index 3: 400
Element at index 4: 500
Element at index 5: 600
Element at index 6: 700
Element at index 7: 800
Element at index 8: 900
Element at index 9: 1000
```

Sample Code

```java
import java.util.Scanner;
public class Array_Sum
{
    public static void main(String[] args)
    {
        int n, sum = 0;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter no. of elements you want in array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter all the elements:");
        for(int i = 0; i < n; i++)
        {
            a[i] = s.nextInt();
            sum = sum + a[i];
        }
        System.out.println("Sum:"+sum);
    }
}
```