Arrays
CSCI 201L

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

- Arrays
- Program
Arrays

- An array is a group of memory locations related by the fact that they all have the same name and the same type.
- Arrays can be addressed using brackets with the index (or position number) inside the brackets.
  - The first element of an array is in the 0th position.
  - So arrays are addressed from 0 to one less than the length of the array.
- Each element in an array can be used just as the other scalar variables we have discussed.
- To define an array, you have to use the “new” operator.

```java
int nums[] = new int[10];
char names[] = new char[20];
```
public class ArrayClass {
    public static void main(String[] args) {
        double darr[] = new double[4];
        darr[0] = 10.1;
        darr[1] = 11.2;
        darr[2] = 12.3;
        darr[3] = 13.4;
        System.out.println(darr[0]);
        Scanner scan = new Scanner(System.in);
        darr[2] = scan.nextDouble();
        System.out.println(darr[2]);
    }
}

Arrays
Symbolic Constants

- Often array sizes are specified in constants that can be used throughout the program.
- To define a constant, make the variable “public static final”.

```java
public class Constants {
    public static final int SIZE = 10;
    public static void main(String[] args) {
        int numArray[] = new int[SIZE];
        for (int i=0; i < SIZE; i++) {
            numArray[i] = 1;
        }
    }
}
```
Passing Arrays to Methods

- When defining the method, the size of the array is omitted because we can get the size from the array itself
- When calling the method, the brackets are left off completely
- Arrays are not primitive variables, so they are passed by reference

```java
public class ArrayTest {
    public static void myMethod(int numArr[]) {
        int sizeOfArray = numArr.length;
    }
    public static void main(String [] args) {
        int numArray[] = new int[10];
        myMethod(numArray);
    }
}
```
public class ArrayTest {
  public void addToIndex(float myArray[], int index, float amount) {
    myArray[index] += amount;
  }

  public void printArray(float myArr[]) {
    for (int count=0; count < myArr.length; count++) {
      System.out.println("arr[" + count + "] = " + myArr[count]);
    }
  }

  public static void main(String [] args)
  {
    float arr[] = new float[10];
    for (int i=0; i < arr.length; i++) {
      arr[i] = Math.random() * 20.0f;
    }
    ArrayTest at = new ArrayTest();
    at.printArray(arr);
    at.addToIndex(arr, 3, 20.4f);
    at.printArray(arr);
  }
}
Multi-Dimensional Arrays

- You are able to have multiple dimensions on an array instead of just one index

```java
int my2DArray[][] = new int[10][20];
for (int i=0; i < my2DArray.length; i++) {
    for (int j=0; j < my2DArray[i].length; j++) {
        my2DArray[i][j] = 10;
    }
}
```
Outline

- Arrays
- Program
Program

- Write a program that randomly generates dice rolls. The number of rolls will be provided by the user. Output the number of times each number occurred followed by the percentage. Here is a sample execution with user input bolded.

```
c:\> java csci201.Dice
How many rolls? 5000
The number 1 occurred 800 times (16%).
The number 2 occurred 750 times (15%).
The number 3 occurred 850 times (17%).
The number 4 occurred 825 times (16.5%).
The number 5 occurred 775 times (15.5%).
The number 6 occurred 800 times (16%).
c:\>
```