FCDS

Programming I

Lecture 10: Arrays II

Arrays as parameters

Array parameter (declare)

```
public static type methodName(type[] name) {
    ...
}
```

- Example:

```
// Returns the average of the given array of numbers.
public static double average(int[] numbers) {
    int sum = 0;
    for (int i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    return (double) sum / numbers.length;
}
```

Array parameter (call)

```
methodName(arrayName);
```

- Example:

```
public class MyProgram {
    public static void main(String[] args) {
        // figure out the average TA IQ
        int[] iq = {126, 84, 149, 167, 95};
        double avg = average(iq);
        System.out.println("Average IQ = " + avg);
    }
    ...
}
```

  - **Notice that you don't write the `[]` when passing the array.**

Read Array

```
public class MyProgram {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter array size:");
        int n = s.nextInt();
        int[] array1 = new int[n];
        int[] array2 = new int[n];
        readArray(array1);
        readArray(array2);
    }
    static void readArray(int[] a) {
        Scanner x = new Scanner(System.in);
        for (int i = 0; i < a.length; i++){
            System.out.print("a[" + i + "]= ");
            a[i] = x.nextInt();
        }
    }
}
```

Print Array

```
public class MyProgram {
    public static void main(String[] args) {
        int[] array1 = {1, 6, 8, 34, 7};
        int[] array2 = {4, 98, 63, 9};
        printArray(array1);
        printArray(array2);
    }
    static void printArray(int[] a) {
        for (int i = 0; i < a.length; i++){
            System.out.println(a[i]);
        }
    }
}
```

Arrays return

Array return (declare)

```
public static type[] methodName(parameters) {
```

- Example:

```
// Returns a new array with two copies of each value.
// Example: [1, 4, 0, 7] -> [1, 1, 4, 4, 0, 0, 7, 7]
public static int[] stutter(int[] numbers) {
    int[] result = new int[2 * numbers.length];

    for (int i = 0; i < numbers.length; i++) {
        result[2 * i]     = numbers[i];
        result[2 * i + 1] = numbers[i];
    }

    return result;
}
```

Array return (call)

type[] **name** = **methodName**(**parameters**);

- Example:

```
public class MyProgram {
        public static void main(String[] args) {
            int[] iq = {126, 84, 149, 167, 95};
            int[] stuttered = stutter(iq);
            System.out.println(Arrays.toString(stuttered));
        }
        ...
}
```

- Output:

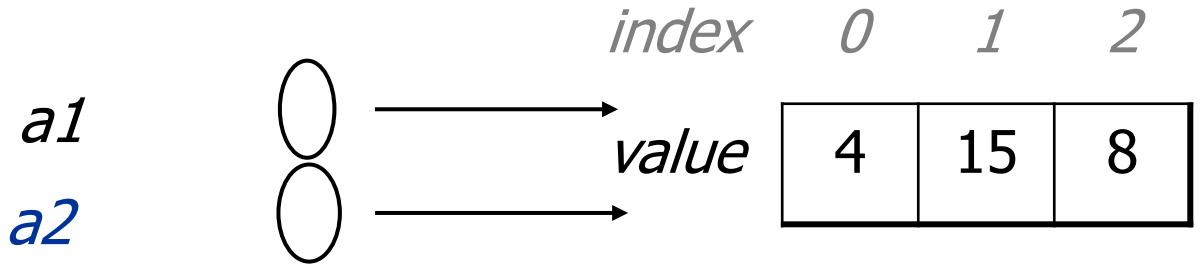
```
[126, 126, 84, 84, 149, 149, 167, 167, 95, 95]
```

Reference semantics

Reference semantics (objects)

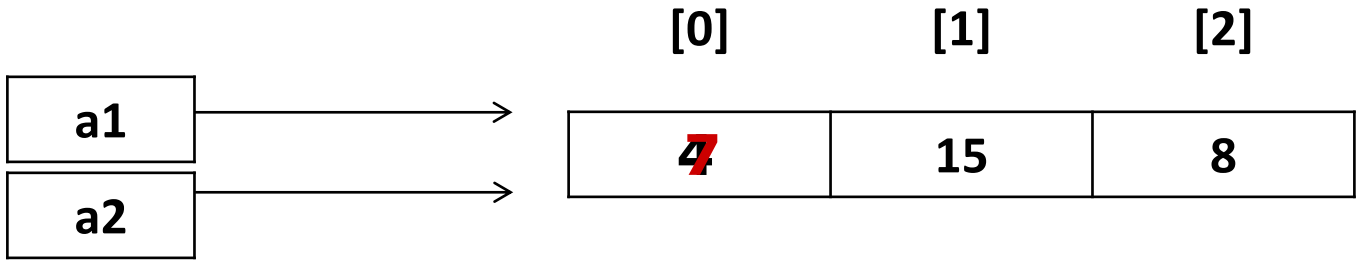
- **reference semantics**: Behavior where variables actually store the address of an object in memory.
  - When one variable is assigned to another, the object is *not* copied; both variables refer to the *same object*.
  - Modifying the value of one variable *will* affect others.

```
int[] a1 = {4, 15, 8};
int[] a2 = a1;          // refer to same array as a1
```



Reference semantics (cont.)

```
public static void main(String[] args) {
        int[] a1 = {4, 15, 8};
        System.out.println(a1);
        int[] a2 = a1;          // refer to same array as a1
        a2[0] = 7;
        System.out.println(Arrays.toString(a1));
}
```



output:

```
[I@4b71bbc9
[7, 15, 8]
```

Arrays pass by reference

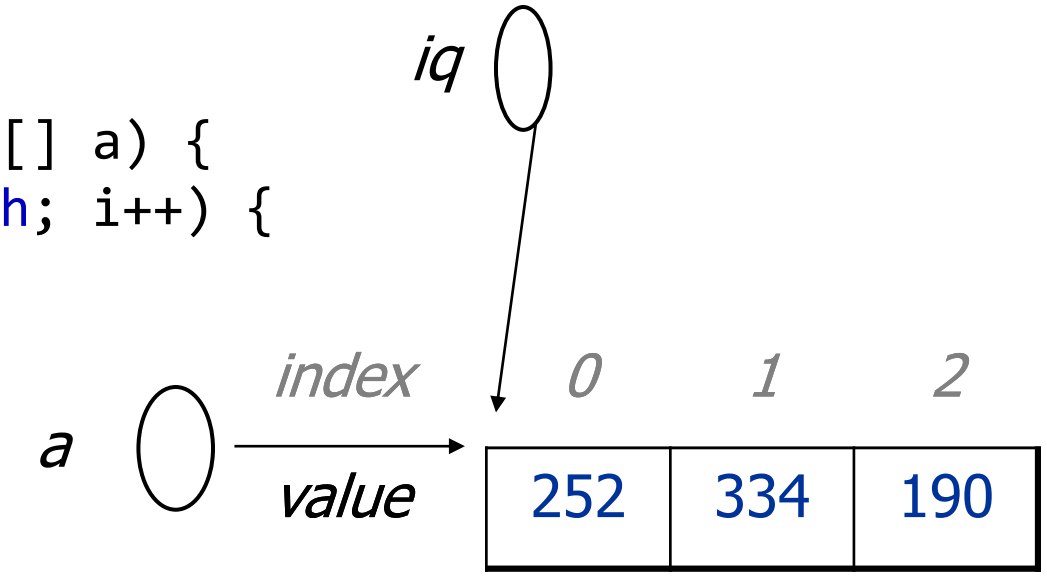
- Arrays are passed as parameters ***by reference***.
  - Changes made in the method are also seen by the caller.

```
public static void main(String[] args) {
    int[] iq = {126, 167, 95};
    increase(iq);
    System.out.println(Arrays.toString(iq));
}

public static void increase(int[] a) {
    for (int i = 0; i < a.length; i++) {
        a[i] = a[i] * 2;
    }
}
```

  - Output:

```
[252, 334, 190]
```



A `swap` method?

- Does the following `swap` method work? Why or why not?

```
public static void main(String[] args) {
    int a = 7;
    int b = 35;

    // swap a with b?
    swap(a, b);

    System.out.println(a + " " + b);
}

public static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

7 35

swap2 (Example)

- Write a method `swap` that accepts an arrays of integers and two indexes and swaps the elements at those indexes.

```
int[] a1 = {12, 34, 56};
swap(a1, 1, 2);
System.out.println(Arrays.toString(a1)); // [12, 56, 34]

// Swaps the values at the given two indexes.
public static void swap(int[] a, int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

swapALL (Example)

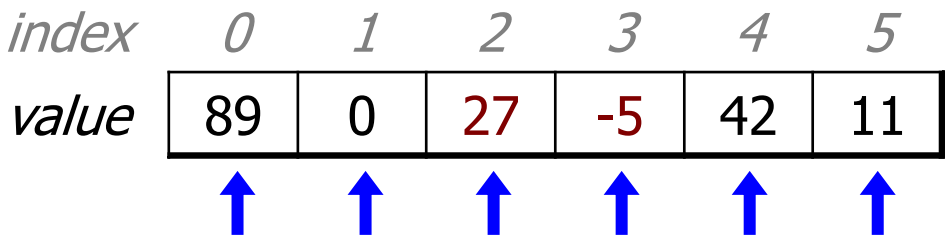
- Write a method `swapAll` that accepts two arrays of integers as parameters and swaps their entire contents. Assume that the two arrays are the same length.

```
int[] a1 = {12, 34, 56};
int[] a2 = {20, 50, 80};
swapAll(a1, a2);
System.out.println(Arrays.toString(a1)); // [20, 50, 80]
System.out.println(Arrays.toString(a2)); // [12, 34, 56]

// Swaps the entire contents of a1 with those of a2.
public static void swapAll(int[] a1, int[] a2) {
    for (int i = 0; i < a1.length; i++) {
        int temp = a1[i];
        a1[i] = a2[i];
        a2[i] = temp;
    }
}
```

Array reverse (Example)

- Write a method `reverse` that accepts an array of integers and reverses the elements of the array.



Array reverse (Example)

```
    public static void main(String[] args) {
    int[] a = {11, 42, -5, 27, 0, 89};
    reverse(a);
     System.out.println(Arrays.toString(a)); // [89, 0, 27, -5, 42, 11]
}


    public static void reverse(int[] numbers) {
        for (int i = 0; i < numbers.length / 2; i++) {
            int temp = numbers[i];
            numbers[i] = numbers[numbers.length - 1 - i];
            numbers[numbers.length - 1 - i] = temp;
        }
    }
```

Array return example (Merge2)

- Write a method `merge` that accepts two arrays of integers and returns a new array containing all elements of the first array followed by all elements of the second.

```
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};
int[] a3 = merge(a1, a2);
System.out.println(Arrays.toString(a3)); // [12, 34, 56, 7, 8, 9, 10]

// Returns a new array containing all elements of a1
// followed by all elements of a2.
public static int[] merge(int[] a1, int[] a2) {
    int[] result = new int[a1.length + a2.length];

    for (int i = 0; i < a1.length; i++) {
        result[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        result[a1.length + i] = a2[i];
    }
  return result;
}
```

Array return example (Merge3)

- Write a method `merge3` that merges 3 arrays similarly.

```
int[] a1 = {12, 34, 56};
int[] a2 = {7, 8, 9, 10};
int[] a3 = {444, 222, -1};
int[] a4 = merge3(a1, a2, a3);
System.out.println(Arrays.toString(a4));
// [12, 34, 56, 7, 8, 9, 10, 444, 222, -1]

// Returns a new array containing all elements of a1
// followed by all elements of a2
// followed by all elements of a3.
```

Array return example (Merge3)

```
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    int[] a4 = new int[a1.length + a2.length + a3.length];

    for (int i = 0; i < a1.length; i++) {
        a4[i] = a1[i];
    }
    for (int i = 0; i < a2.length; i++) {
        a4[a1.length + i] = a2[i];
    }
    for (int i = 0; i < a3.length; i++) {
        a4[a1.length + a2.length + i] = a3[i];
    }
    return a4;
}

// Shorter version that calls merge.
public static int[] merge3(int[] a1, int[] a2, int[] a3) {
    return merge(merge(a1, a2), a3);
}
```