Topic 22
arrays - part 2

Based on slides by Marty Stepp and Stuart Reges
from http://www.buildingjavaprograms.com/

Swapping values

```
public static void main(String[] args) {
    int a = 7;
    int b = 35;
    // swap a with b?
    a = b;
    b = a;
    System.out.println(a + " " + b);
}
```

– What is wrong with this code? What is its output?

- The red code should be replaced with:

```
int temp = a;
a = b;
b = temp;
```

Array reversal question

- Write code that reverses the elements of an array.
  - For example, if the array initially stores:

    ```
    [11, 42, -5, 27, 0, 89]
    ```

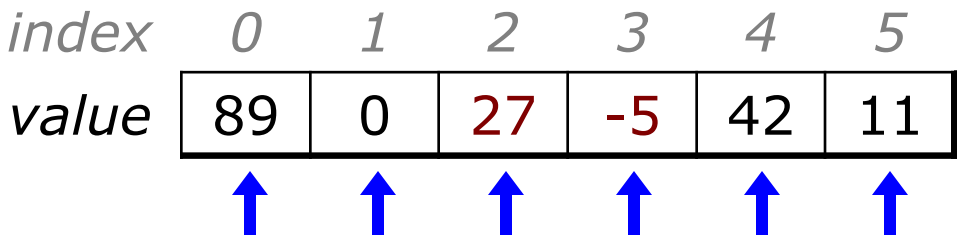
  - Then after your reversal code, it should store:

    ```
    [89, 0, 27, -5, 42, 11]
    ```

    - The code should work for an array of any size.
    - Hint: think about swapping various elements...

Algorithm idea

- Swap pairs of elements from the edges; work inwards:



Flawed algorithm

- What's wrong with this code?

```
int[] numbers = [11, 42, -5, 27, 0, 89];

// reverse the array
for (int i = 0; i < numbers.length; i++) {
    int temp = numbers[i];
    numbers[i] = numbers[numbers.length - 1 - i];
    numbers[numbers.length - 1 - i] = temp;
}
```

- The loop goes too far and un-reverses the array! Fixed version:

```
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 reverse question 2

- Turn your array reversal code into a `reverse` method.
  - Accept the array of integers to reverse as a parameter.

```
int[] numbers = {11, 42, -5, 27, 0, 89};
reverse(numbers);
```

  - How do we write methods that accept arrays as parameters?
  - Will we need to return the new array contents after reversal?

    ...

Array parameter (declare)

```
public static <type> <method>(<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;
}
```

  - You don't specify the array's length (but you can examine it).

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.

Array return (declare)

```
public static <type>[] <method>(<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>* = *<method>*(*<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

reading: 7.3

clicker 1

- What is output by the following code?

```
int[] data = {1, 5, 3};
foo(data);
System.out.print(Arrays.toString(data));

public static void foo(int[] d) {
    int temp = d[0];
    d[0] = d[d.length - 1];
    d[d.length - 1] = temp;
    System.out.print(Arrays.toString(d) + " ");
}
```

A. [3, 5, 1] [1, 5, 3]

B. [3, 5, 1] [3, 5, 1]

C. [1, 5, 3] [1, 5, 3]

D. [5, 3, 1] [1, 5, 3]

E. Something else

clicker 2

- What is output by the following code?

```
int[] data = {1, 5, 3};
bar(data);
System.out.print(Arrays.toString(data));

public static void bar(int[] d) {
    d[0]++;
    d = new int[] {4, 6};
    System.out.print(Arrays.toString(d) + " ");
}
```

A. [4, 6] [2, 5, 3]

B. [4, 6] [4, 6]

C. [1, 5, 3] [1, 5, 3]

D. [2, 5, 3] [2, 5, 3]

E. Something else

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;
}
```

Value semantics

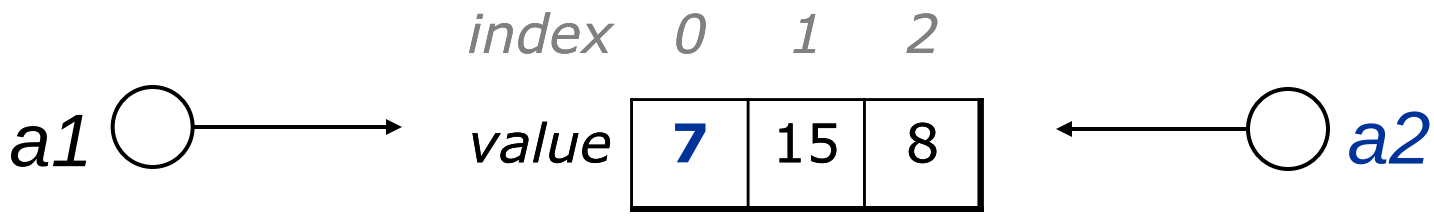
- **value semantics**: Behavior where values are copied when assigned, passed as parameters, or returned.
  - All primitive types in Java use value semantics.
  - When one variable is assigned to another, its value is copied.
  - Modifying the value of one variable does not affect others.

```
int x = 5;
int y = x;          // x = 5, y = 5
y = 17;             // x = 5, y = 17
x = 8;              // x = 8, y = 17
```

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.

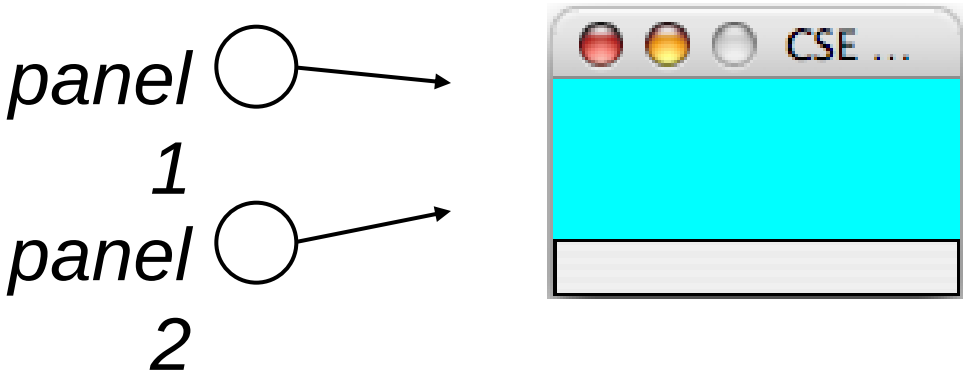
```
int[] a1 = {4, 15, 8};
int[] a2 = a1;              // refer to same array as a1
a2[0] = 7;
System.out.println(Arrays.toString(a1)); // [7, 15, 8]
```



References and objects

- Arrays and objects use reference semantics. Why?
  - *efficiency.* Copying large objects slows down a program.
  - *sharing.* It's useful to share an object's data among methods.

```
DrawingPanel panel1 = new DrawingPanel(80, 50);
DrawingPanel panel2 = panel1;   // same window
panel2.setBackground(Color.CYAN);
```

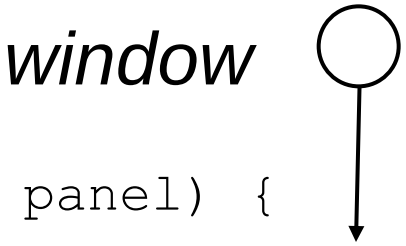


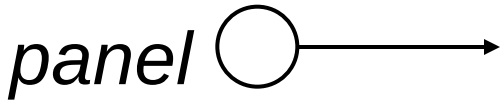
Objects as parameters

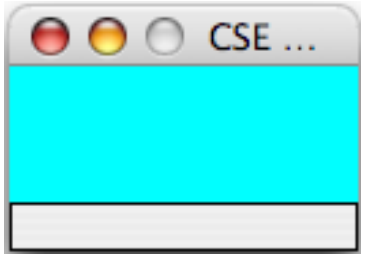
- When an object is passed as a parameter, the object is *not* copied. The parameter refers to the same object.
  - If the parameter is modified, it *will* affect the original object.

```
public static void main(String[] args) {
    DrawingPanel window = new DrawingPanel(80, 50);
    window.setBackground(Color.YELLOW);
    example(window);
}

public static void example(DrawingPanel panel) {
    panel.setBackground(Color.CYAN);
    ...
}
```







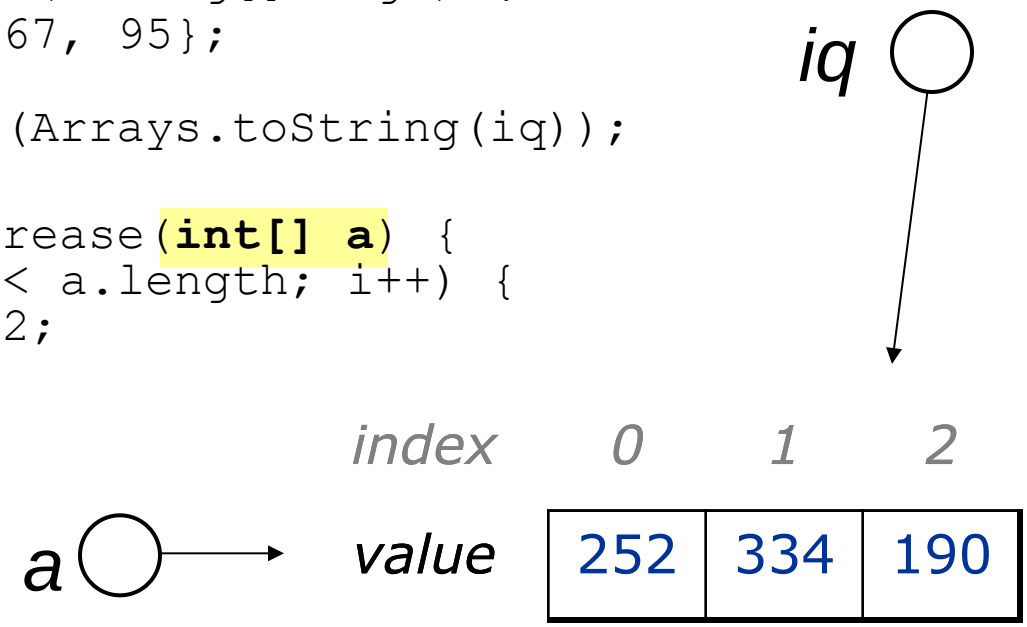
Copy of a reference

- Array variables are references
- A parameter is a copy of the same reference the argument stores.
- Changes made in the method **to the elements** 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]
```



Array reverse question 2

- Turn your array reversal code into a `reverse` method.
  - Accept the array of integers to reverse as a parameter.

```
int[] numbers = {11, 42, -5, 27, 0, 89};
reverse(numbers);
```

- Solution:

```
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 parameter questions

- Write a method `swap` that accepts an array 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]
```

- 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]
```

Array parameter answers

```
// 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;
}

// 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 return question

- 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]
```

- 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]
```

Array return answer 1

```
// 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 answer 2

```
// Returns a new array containing all elements of
   a1,a2,a3.
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);
}
```