Array traversals, text processing

Array traversals

- **traversal**: An examination of each element of an array.

```
for (int i = 0; i < array.length; i++) {
    do something with array[i];
}
```

- Examples:
  - printing the elements
  - searching for a specific value
  - rearranging the elements
  - computing the sum, product, etc.

Quick array initialization

type`[]` **name** `=` `{`**value**`,` **value**`,` … **value**`};`

- Example:

```
int[] numbers = {12, 49, -2, 26, 5, 17, -6};
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	12	49	-2	26	5	17	-6

- Useful when you know what the array's elements will be
- The compiler figures out the size by counting the values

"Array mystery" problem

- What element values are stored in the following array?

```
int[] a = {1, 7, 5, 6, 4, 14, 11};
for (int i = 0; i < a.length - 1; i++) {
    if (a[i] > a[i + 1]) {
        a[i + 1] = a[i + 1] * 2;
    }
}
```

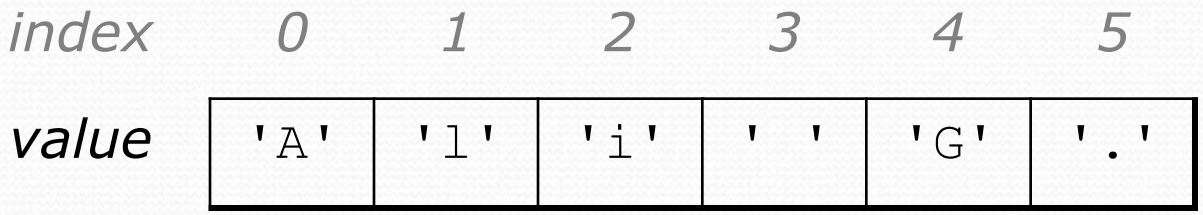
index	*0*	*1*	*2*	*3*	*4*	*5*	*6*
value	1	7	10	12	8	14	22

Text processing

- **text processing**: Examining, editing, formatting text.
  - Often involves `for` loops to examine each letter of a `String`.
    - Count the number of times the letter 's' occurs in a file.
    - Find which letter is most common in a file.
    - Count A, C, T and Gs in `String`s representing DNA strands.

- `String`s are represented internally as arrays of `char`.

```
String str = "Ali G.";
```



Recall: type `char`

- **`char`**: A primitive type representing a single character.
  - Values are surrounded with apostrophes: `'a'` or `'4'` or `'\n'`

- Access a string's characters with its `charAt` method.

```
String word = console.next();
char firstLetter = word.charAt(0);
if (firstLetter == 'c') {
    System.out.println("That's good enough for me!");
}
```

- Use `for` loops to examine each character.

```
String coolMajor = "CSE";
for (int i = 0; i < coolMajor.length(); i++) {
    System.out.println(coolMajor.charAt(i));
}
```

Text processing question

- Write a method `tallyVotes` that accepts a `String` parameter and prints the number of McCain, Obama and independent voters.

```
// (M)cCain, (O)bama, (I)ndependent
String voteText = "MOOOOOOMMMMMOOOOOOMOMMIMOMMIMOMMIO";
tallyVotes(voteText);
```

  - Output:

```
Votes: [16, 14, 3]
```

Arrays.toString

- `Arrays.toString` accepts an array as a parameter and returns a `String` representation of its elements.

```
int[] e = {0, 2, 4, 6, 8};
e[1] = e[3] + e[4];
System.out.println("e is " + Arrays.toString(e));
```

Output:

```
e is [0, 14, 4, 6, 8]
```

- Must `import java.util.*;`

The `Arrays` class

- Class `Arrays` in package `java.util` has useful static methods for manipulating arrays:

Method name	Description
`binarySearch(`**array**`, `**value**`)`	returns the index of the given value in a sorted array (< 0 if not found)
`equals(`**array1**`, `**array2**`)`	returns `true` if the two arrays contain the same elements in the same order
`fill(`**array**`, `**value**`)`	sets every element in the array to have the given value
`sort(`**array**`)`	arranges the elements in the array into ascending order
`toString(`**array**`)`	returns a string representing the array, such as `"[10, 30, 17]"`

Text processing answer

```
public static int[] tallyVotes(String votes) {
    int[] tallies = new int[3];   // M -> 0, O -> 1, I -> 2

    for(int i = 0; i < votes.length(); i++) {
        if(votes.charAt(i) == 'M') {
            tallies[0]++;
        } else if(votes.charAt(i) == 'O') {
            tallies[1]++;
        } else {                        // votes.charAt(i) == 'I'
            tallies[2]++;
        }
    }

    System.out.println("Votes: " + Arrays.toString(tally));;
}
```

Arrays as parameters and returns; values vs. references

reading: 7.1, 3.3, 4.3

self-checks: Ch. 7 #5, 8, 9

exercises: Ch. 7 #1-10

Swapping values

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

    // swap a with b (incorrectly)
    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;
```

A `swap` method?

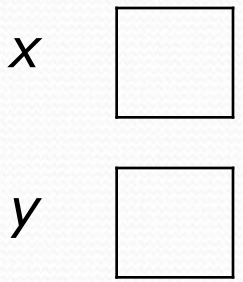
- Does the following `swap` method work? Why or why not?

```
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 (primitives)

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

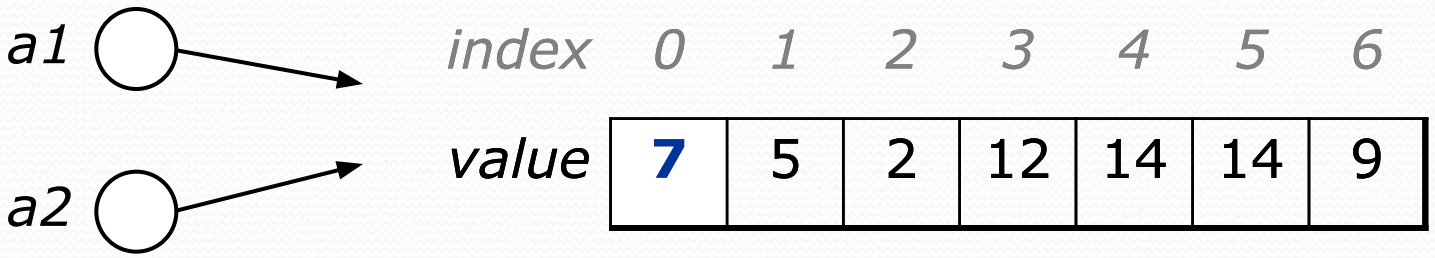
```
int x = 5;
int y = x;        // x = 5, y = 5
y = 17;           // x = 5, y = 17
x = 8;            // x = 8, y = 17
```



Reference semantics (objects)

- **reference semantics**: Behavior where variables actually store the address of an object in memory.
  - When one reference 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.

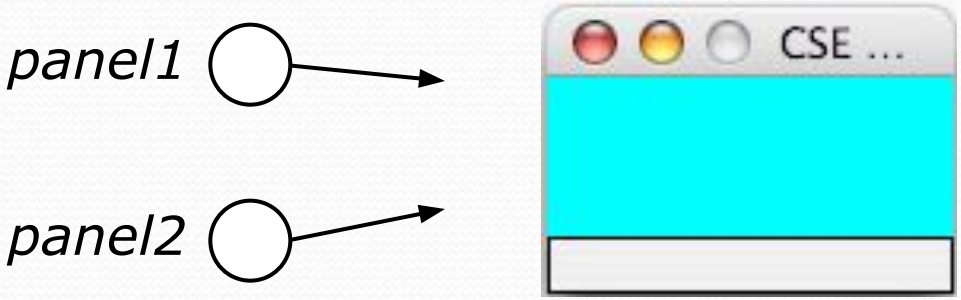
```
int[] a1 = {4, 5, 2, 12, 14, 14, 9};
int[] a2 = a1;      // refer to same array as a1
a2[0] = 7;
System.out.println(a1[0]);   // 7
```



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.

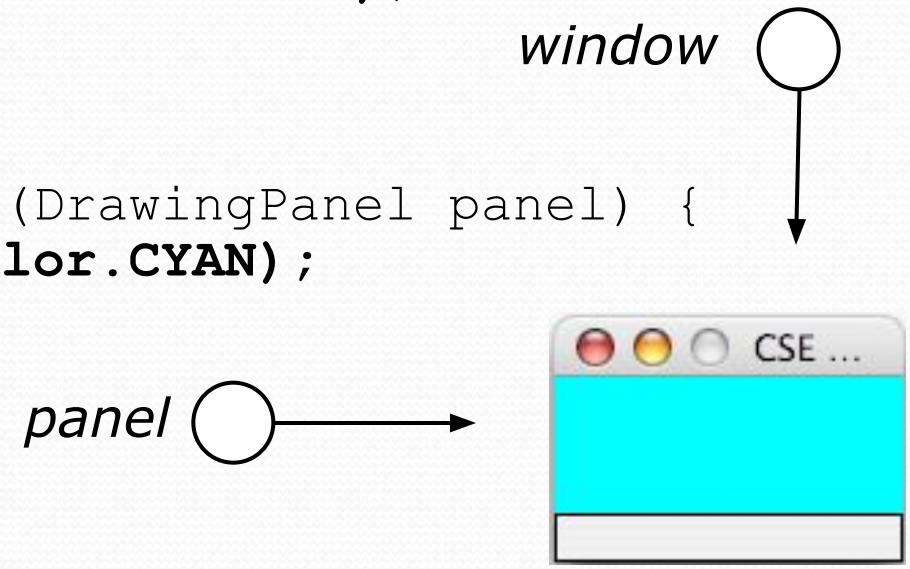
```
DrawingPanel panel1 = new DrawingPanel(80, 50);
DrawingPanel panel2 = panel1;   // same window
panel2.setBackground(Color.CYAN);
```



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);
}
```



Arrays as parameters

- Declaration:

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

  - Example:

```
public static double average(int[] numbers) {
```

- Call:

```
methodName(arrayName);
```

  - Example:

```
int[] scores = {13, 17, 12, 15, 11};
double avg = average(scores);
```