

## Virtual Lecture Notes

Julia and Jorge would like to be able to sort their home listings according to price. They would like us to do this using bubble sort.

First, let us define a class called **HouseListing** for a house listing.

```
public class HouseListing
{
    // instance variables
    private double cost;
    private String address;
    private String city;
    private String state;
    private String zip;

    /**
     * Constructor for objects of class HouseListing
     */
    public HouseListing (String a, String ct, String s,
                        String z, double c)
    {
        // initialise instance variables
        cost = c;
        address = a;
        city = ct;
        state = s;
        zip = z;
    }
    public String toString()
    {
        String listing;
        listing = address + "\n" + city + ", " + state + " " +
            zip;
        listing = listing + "\nCost: " +
            String.format("%,10.2f", cost) + "\n";
        return listing;
    }
}
```

```

    }
    public String getAddress()
    {
        return address;
    }
    public void setAddress(String a)
    {
        address = a
    }
    public String getCity()
    {
        return city;
    }
    public void setCity(String c)
    {
        city = c;
    }
    public String getState()
    {
        return state;
    }
    public void setState(String s)
    {
        state = s;
    }
    public String getZip()
    {
        return zip;
    }
    public void setZip(String z)
    {
        zip = z;
    }
    public double getCost()
    {
        return cost;
    }
    public void setCost(double c)
    {
        cost = c;
    }

```

Note that in the `toString()` method of **HouseListing**, we use the String format to format the **cost** as

currency.

Next, we create a file called **TestListing**, which will be where we create our array of houses, and we make a list of five houses.

```
HouseListing [] houses = HouseListing [5]

Houses[0] = new HouseListing ("123 Any Street", "St. Cloud",
                                "FL", "34769", 79000);
Houses[1] = new HouseListing ("456 Cherry Lane", "St. Cloud",
                                "FL", "34772", 11000);
Houses[2] = new HouseListing ("1892 Ocean Blue", "Kissimmee",
                                "FL", "34647", 212000);
Houses[3] = new HouseListing ("339 Curry Ave", "Kissimmee", "FL",
                                "34647", 88000);
Houses[4] = new HouseListing ("612 Orange Street", "Orlando",
                                "FL", "32196", 451000);
```

Now we would like to do a bubble sort.

The general Java code for a bubble sort on an integer array called a is:

```
Public static void bubbleSort(int[] a)
{
    int out, in, temp;

    for (out=a.length-1; out>0; out--) // outer loop (backward)
        for (in=0; in<out; in++) // inner loop (forward)
            if ( a[in] > a[in+1] ) // out of order?
                { // swap them
                    temp = a[in + 1];
                    a[in + 1] = a[in];
                    a[in] = temp;
                }
}
```

Remember that a bubble sort works by repeatedly stepping through the array, comparing two array elements at a time. If the elements are out of order, then they are swapped. After going through the array, the largest (or smallest, if you are sorting in descending order) element will be in the last position of the array. This process is called **bubbling** and is why the sorting algorithm is called bubble sort. The next pass through the array will not involve this last element, as it is now in its proper place. This process is repeated until going through the array results in no swap.

Therefore, the outer **for** loop goes from the end of the array towards the beginning of the array. This is so the inner loop will always go through smaller and smaller portions of the array, since the inner loop will always stop before **out's** value. Since **out** constantly decreases by 1, each element is bubbled into its correct spot and then we can stop looking at it. Why look at something that is already where it should be?

This method sorts ascending, because the **if** statement compares using **>**. If it were sorting descending, we would just switch the **>** with a **<**. The swap is essential, as we need a temporary variable to help us in swapping two array locations.

You should memorize the process of a bubble sort, as it is commonly used in practically every programming language.

Now, we will apply the process to our **houses** array. Let us sort the houses based upon cost, from lowest to highest.

All that we have to change about the bubble sort is the comparison the **if** statement makes and the swapping procedure.

The **if** statement will become as shown below, as array **a** will now be of type **HouseListing**:

```
if (a[in].getCost() < a[in + 1].getCost())
```

The swap procedure will change in this fashion:

```
HouseListing temp
```

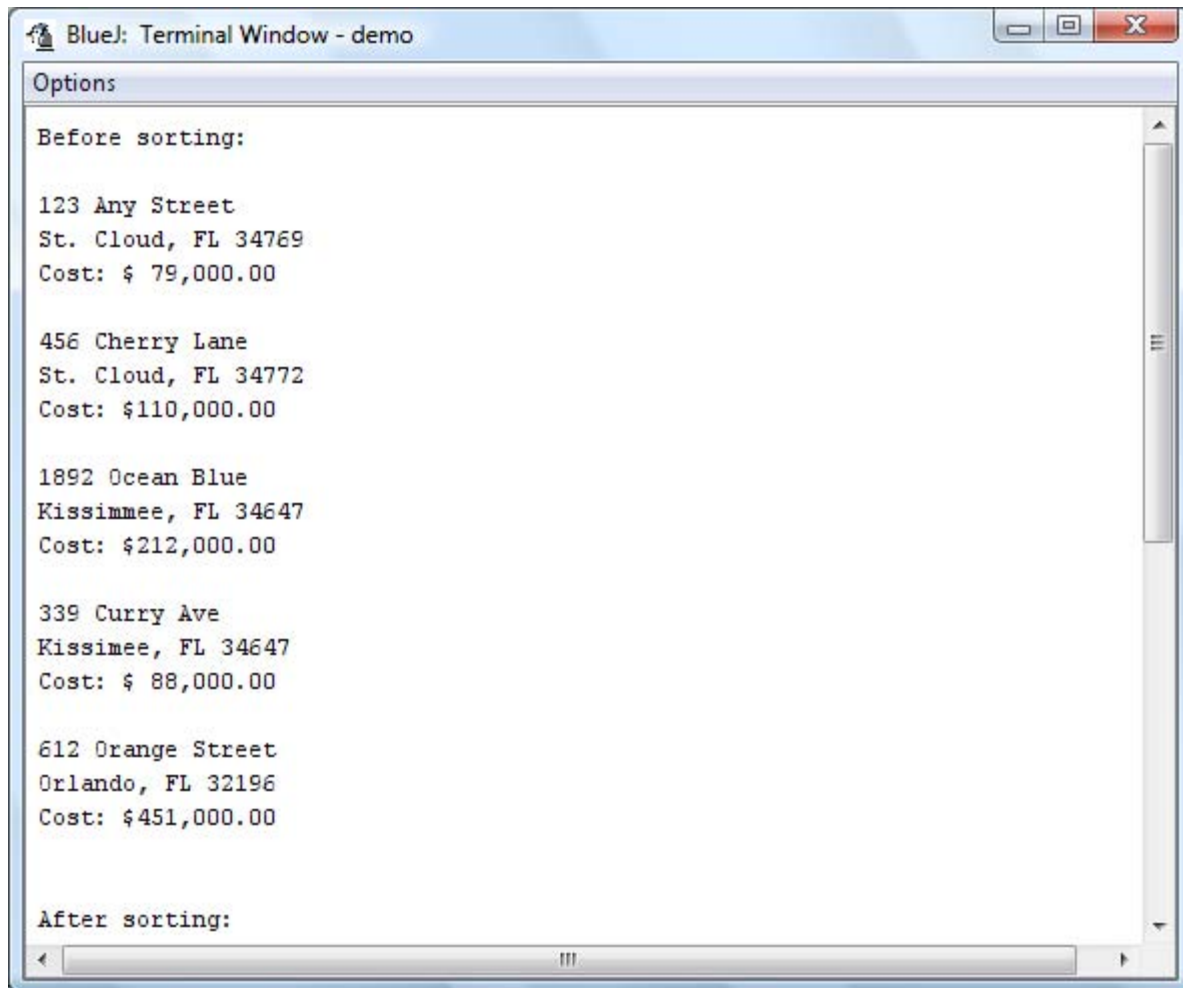
Notice that the only thing we had to change was to make **temp** of type **HouseListing**.

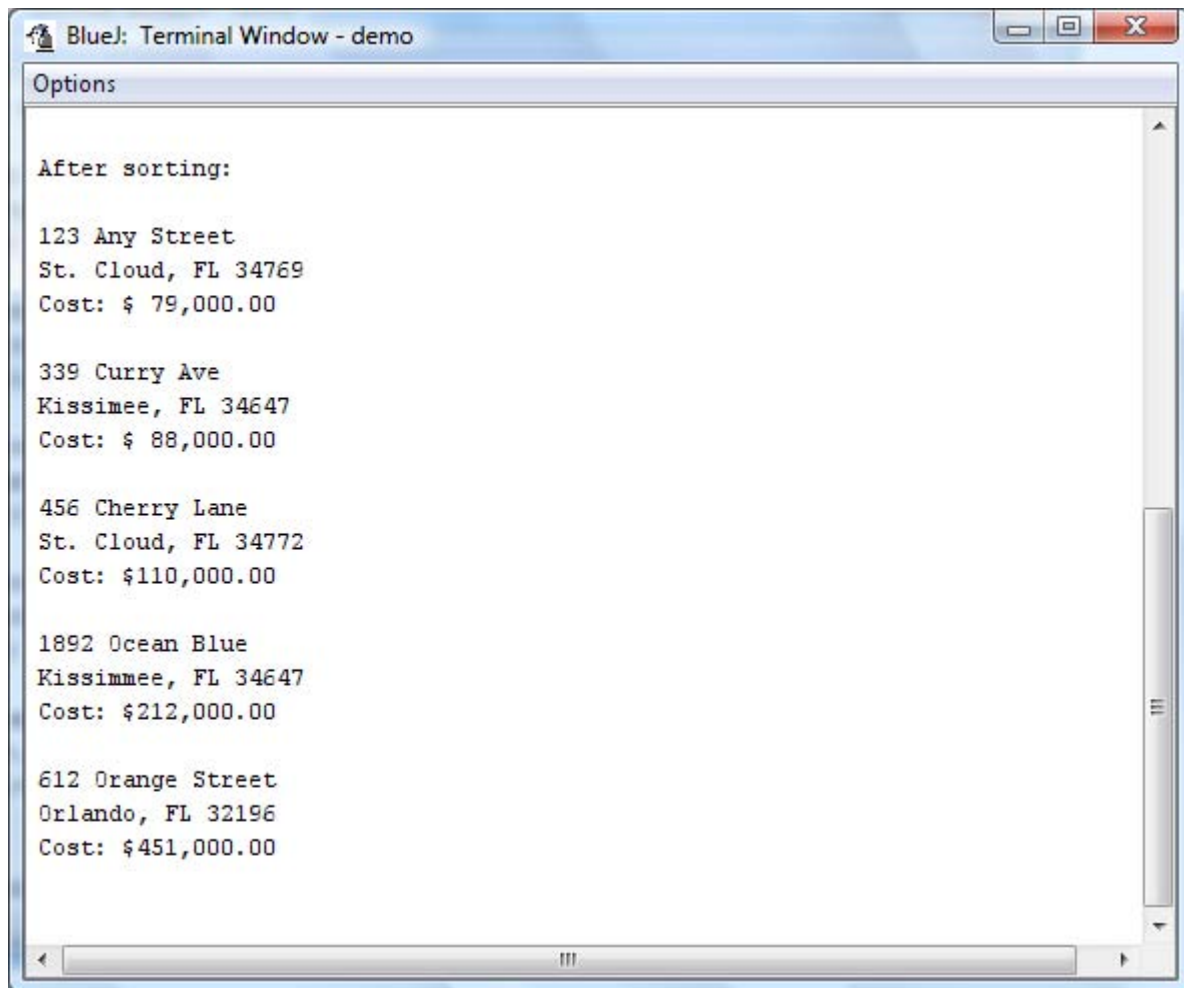
Putting it all together, we get:

```
Public static void bubbleSort (HouseListing [] a)
{
    int out, in;
    HouseListing temp;

    for (out=a.length-1; out>0; out-- )    // outer loop (backward)
        for(in=0; in<out; in++)    // inner loop (forward)
            if( a[in].getCost () > a[in+1].getCost () )
                // out of order?
                {
                    // swap them
                    temp = a[in + 1];
                    a[in +1] = a[in];
                    a[in] = temp;
                }
}
```

In **TestListing**, we first print out the houses array unsorted, and then sorted, using **bubbleSort()**. The output looks like this:





```
BlueJ: Terminal Window - demo
Options

After sorting:

123 Any Street
St. Cloud, FL 34769
Cost: $ 79,000.00

339 Curry Ave
Kissimmee, FL 34647
Cost: $ 88,000.00

456 Cherry Lane
St. Cloud, FL 34772
Cost: $110,000.00

1892 Ocean Blue
Kissimmee, FL 34647
Cost: $212,000.00

612 Orange Street
Orlando, FL 32196
Cost: $451,000.00
```

Notice that the houses are now sorted. Take a look at the demo programs. Make sure you understand them. Try changing the > to a < and then run the program to see the results.

- Download the [HouseListing.java](#) and [TestListing.java](#) files to your unit 7 demo programs directory and open them.
- Run the files and make sure you understand them before you continue.