ICS4U Module 6: Note + Exercise 1b

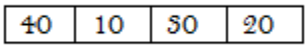
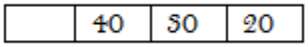
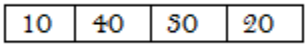
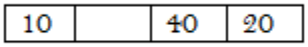
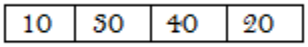
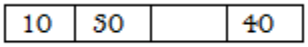
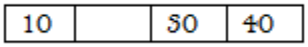
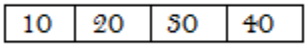
Insertion Sort

- Insertion sort is more efficient than the selection sort

Insertion Sort Algorithm:

Sort the first two items in a list
Insert the third item into the first two
Repeat for remaining elements

Insertion Sort Illustration

Step 1: 	The original list
Step 2:  	40 is shifted to make room for the second item, 10
Step 3:  	30 is compared to the value in the previous position (40). 40 is shifted to position 3. 30 is then compared to the value in the previous position (10). 30 is placed in position 2.
Step 4:   	Repeat the same process.

Insertion Sort Pseudocode:

```
for (index = 1 to array.length – 1) {
      temp = array[index];
      previousIndex = index -1
      while (array[previousIndex] > temp && previousIndex > 0) {
            shift array[previousIndex] up one element
            previousIndex = previousIndex - 1
      }
      if (array[previousIndex] > temp) {
            swap the two elements
      } else {
            insert element at appropriate location
      }
}
```

Insertion Sort Implementation:

The Sorts class has been modified to include an insertionSort() method:

```
/**
 * Sorts an array of integer from low to high
 * pre: none
 * post: Integers have been sorted from low to high
 */
public static void insertionSort(int[] items) {
      int temp, previousIndex;

      for (int index = 1; index < items.length; index++) {  //start with second item
            temp = items[index];
            previousIndex = index - 1;
            while ((items[previousIndex] > temp) && (previousIndex > 0)) {
                  items[previousIndex + 1] = items[previousIndex];
                  /* decrease index to compare current item to next previous item */
                  previousIndex -= 1;
            }
            if (items[previousIndex] > temp) {
                  /* shift item in first element up into next element position */
                  items[previousIndex + 1] = items[previousIndex];
                  /* place current item at index 0 (first element) */
                  items[previousIndex] = temp;
            } else {
                  /* place current item at index ahead of previous item */
                  items[previousIndex + 1] = temp;
            }
      }
}
```

The TestSorts application has been modified to use the insertionSort() method to sort an array of integers:

```
import java.util.Scanner;

public class TestSorts {

	public static void displayArray(int[] array){
		for (int i = 0; i< array.length; i++){
			System.out.print(array[i] + " " );

		}
		System.out.println("\n");
	}

	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		int numItems;
		int[] test;

		System.out.print("Enter number of elements: ");
		numItems = input.nextInt();

		/* populate array with random integers */
		test = new int[numItems];
		for (int i = 0; i < test.length; i++){
			test[i] = (int)(100 * Math.random());
		}
		System.out.println("Unsorted:");
		displayArray(test);

		Sorts.insertionSort(test);

		System.out.println("Sorted:");
		displayArray(test);

	}

}
```

Output:

```
Enter number of elements: 10
Unsorted:
76  6  95  34  61  35  52  1  60  76

Sorted:
1  6  34  35  52  60  61  76  78  95
```

Programming Exercise:

Create an ObjectsInsertionSort application that implements an insertion sort on an array of objects. Test the sort on an array of String objects. Refer to the differences between the SelectionSort for integers and the SelectionSort for objects provided in lesson 1A.

Submit your source code to the Google Doc "ICS4U – Activity Submission Form"