Selection Sort

- There are multiple algorithms for sorting

Selection Sort Algorithm:

Find the lowest item in a list
Swap that item with the first item
Now ignore the first item and repeat (look at items 2 through n)
etc. Until the last item is reached

- Can be implemented with nested for loops as shown:

Selection Sort Pseudocode:

```
for (arrayIndex = 0 to numItems -1)//outer loop
   for (subarrayIndex = arrayIndex to numItems -1) //inner loop
      if (items[subarrayIndex] < items[arrayIndex]) {
         swap items[subarrayIndex] and items[arrayIndex]
      }
   }
 }
```

Outer loop: controls which element to compare

Inner loop: iterates through the array after the element (the subarray)

Here is a Sorts class that implements a selectionSort() method

```
/*
 * Sorts.java
 * A class that implements sorting algorithms.
 */

 public class Sorts {

      /**
       * Sorts an array of data from low to high
       * pre: none
       * post: items has been sorted from low to high
       */
      public static void selectionSort(int[] items) {

            for (int index = 0; index < items.length; index++) {
                  for (int subIndex = index; subIndex < items.length;
subIndex++) {
                        if (items[subIndex] < items[index]) {
                              int temp = items[index];
                              items[index] = items[subIndex];
                              items[subIndex] = temp;
                        }
                  }
            }
      }
}
```

The client code TestSorts below generates an array of integers and then calls selectionSort() to sort them

```
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 */
            test = new int[numItems];
            for (int i = 0; i < test.length; i++) {
                  test[i] = (int)(101 * Math.random());
            }
            System.out.println("Unsorted:");
            displayArray(test);

            Sorts.selectionSort(test);

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

      }
```

Output:

```
Enter number of elements: 10
Unsorted:
79  50  10  18  77  31  64  18  14  96

Sorted:
10  14  18  18  31  50  64  77  79  96
```

Sorting Objects

- Recall that relational operators (<, >) cannot be used to compare **objects**
- Objects must use methods of their class to determine if one object is greater, less or equal to another
    - equals() method in a class is used to determine equality
    - compareTo() is used to determine order
- String, Double, Integer and even the Circle class from before implement the Comparable interface (needed to be sorted)

The Sorts class has been modified to include an overloaded SelectionSort() method, which has a Comparable array parameter

```
      /**
       * Sorts an array of objects from low to high
       * pre: none
       * post: objects have been sorted from low to high
       */
      public static void selectionSort(Comparable[] items) {

            for (int index = 0; index < items.length; index++) {
                  for (int subIndex = index; subIndex < items.length;
subIndex++) {
                        if (items[subIndex].compareTo(items[index]) < 0) {
                              Comparable temp = items[index];
                              items[index] = items[subIndex];
                              items[subIndex] = temp;
                        }
                  }
            }
      }
```

And here is the client code TestSorts modified to sort an array of Circle objects

```
import java.util.Scanner;

 public class TestSorts {

      public static void displayArray(Circle[] array) {
            for (int i = 0; i < array.length; i++) {
                  System.out.println(array[i] + "   ");
            }
            System.out.println("\n");
      }
 public static void main(String[] args) {

     Scanner input = new Sanner(System.in);
     int numObjects;
     Circle[] test;

     System.out.print("Enter number of objects: ");
     numObjects = input.nextInt();
     input.close();

     /* populate array with Circle objects of varying radii */
     test = new Circle[numObjects];
     for (int i = 0; i < test.length; i++) {
          test[i] = new Circle((int)(10 * Math.random() + 1));
     }
     System.out.println("Unsorted:");
     displayArray(test);

     Sorts.selectionSort(test);

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

Output:

```
Enter number of objects: 3
Unsorted:
Circle has radius 1.0
Circle has radius 9.0
Circle has radius 3.0

Sorted:
Circle has radius 1.0
Circle has radius 3.0
Circle has radius 9.0
```

Programming Exercise:

Create a AlphaOrder application that implements a selection sort on an list of user-entered Strings. The application should have output that looks similar to:

```
Enter number of words
3
Enter a word:
STP
Enter a word:
McDougall
Enter a word:
ICS4U
Unsorted
STP
McDougall
ICS4U


Sorted
ICS4U
McDougall
STP
```

Submit your source code to the Google Doc "ICS4U – Activity Submission Form" **AND** submit your project via the dropbox. Recall proper naming conventions "Last name, First name – Module 6, Ex 1a - AlphaOrder"