4. Sorting Algorithms :: Selection Sort

You learned the selection sort algorithm in CSE110 so we will briefly review it. Here is a Java implementation that sorts an ArrayList<T> list (where T can be any class which implements the *Comparable*<T> interface).

```
//*************************************************************************************
// CLASS: SelectionSorter (SelectionSorter.java)
//*************************************************************************************
import java.util.ArrayList;

public class SelectionSorter {

    public static final int SORT_ASCENDING  = 0;
    public static final int SORT_DESCENDING = 1;

    /**
     * Searches pList starting at index pStartIndex up to the last element for the maximum
     * element in that range. Returns the index of the maximum element.
     */
    private static int findMaxIndex(ArrayList<T> pList, int pStartIndex) {
        int maxIndex = pStartIndex;
        for (int i = pStartIndex + 1; i < pList.size(); ++i) {
            if (pList.get(i).compareTo(pList.get(maxIndex)) > 0) {
                maxIndex = i;
            }
        }
        return maxIndex;
    }
```

4. Sorting Algorithms :: Selection Sort (continued)

```
/**
 * Searches pList starting at index pStartIndex up to the last element for the minimum
 * element in that range. Returns the index of the minimum element.
 */
private static int findMinIndex(ArrayList<T> pList, int pStartIndex) {
   int minIndex = pStartIndex;
    for (int i = pStartIndex + 1; i < pList.size(); ++i) {
        if (pList.get(i).compareTo(pList.get(minIndex)) < 0) {
            minIndex = i;
        }
    }
    return minIndex;
}
```

4. Sorting Algorithms :: Selection Sort (continued)

```
    /**
     * Sorts pList into ascending (pOrder = SORT_ASCENDING) or descending (pOrder =
     * SORT_DESCENDING) order using the selection sort algorithm.
     */
    public static void selectionSort(ArrayList<T> pList, int pOrder) {
        for (int startIndex = 0; startIndex < pList.size() - 1; ++startIndex) {
            if (pOrder == SORT_ASCENDING) {
                int minIndex = findMinIndex(pList, startIndex);
                swap(pList, startIndex, minIndex);
            } else {
                int maxIndex = findMaxIndex(pList, startIndex);
                swap(pList, startIndex, maxIndex);
            }
        }
    }

    /**
     *  Swaps the elements in pList at pIndex1 and pIndex2.
     */
    private static void swap(ArrayList<T> pList, int pIndex1, int pIndex2) {
        T temp = pList.get(pIndex1);
        pList.set(pIndex1, pList.get(pIndex2));
        pList.set(pIndex2, temp);
    }
}
```