Previous Class

• What did we discuss?
Today

- Assignment 4: Extension!
  - ALL of it (parts A and B) will be due on Wed, 12/2
  - Here are the instructions again: [http://comp110.com/topics/getting-started/assignment-4-submission-instructions](http://comp110.com/topics/getting-started/assignment-4-submission-instructions)
  - Here is the site you submit to: [http://comp110.com/grader](http://comp110.com/grader)

- Today – Sorting

Sorting

- Put elements of an array in some order
  - alphabetize names
  - order grades lowest to highest

- Two simple sorting algorithms
  - selection sort
  - insertion sort
Selection Sort

- Sorts by putting values directly into their final, sorted position

- For each value in the list, the selection sort finds the value that belongs in that position and puts it there

Selection Sort

- Scan the list to find the smallest value
- Exchange (swap) that value with the value in the first position in the list
- Scan rest of list for the next smallest value
- Exchange that value with the value in the second position in the list
- And so on, until you get to the end of the list
Selection Sort at work

98 68 83 74 93

68 98 83 74 93

68 74 83 98 93

68 74 83 93 98 SORTED!

Selection Sort

• Sorts in ascending order

• Can be changed to sort in descending order
  – look for max instead of min
Selection Sort – another example

and so on...

private static void swap(int i, int j, int[] a) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}

• This method will swap the value of a[i] and a[j]
http://www.sorting-algorithms.com/

Selection Sort Pseudocode

```plaintext
for (index = 0; index < length; index++){
    Find index of smallest value of array between current index and end of array;
    Swap values of current index and the index with the smallest value;
}
```
Selection Sort - example

• Open up Eclipse

• Create a new Java Project – call it Sorting

• Create a new class – call it SelectionSortExample

Selection Sort - example

• Go to:
  http://cs.unc.edu/~aikat/courses/comp110/docs/SelectionSort.pdf

• Don’t copy and paste this into Eclipse!

• Your console should show the unsorted and sorted arrays:
  [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Selection Sort - discussion

• There is one class

• How many methods?
  – main
  – selectionSort
  – getIndexOfSmallest
  – interchange

Selection Sort – part1 (main method)

```java
public static void main(String[] args) {

    int[] myArray = {10,9,8,7,6,5,4,3,2,1};

    // using an Array method to convert the input array to a string...
    // ... because println takes a string as argument
    // print the input (unsorted) array
    System.out.println( Arrays.toString( myArray ) );

    // calling your own method "selectionSort" (defined below); array is input
    selectionSort(myArray);

    System.out.println( Arrays.toString( myArray ) ); // print the sorted array
}
```
Selection Sort – part2 (selectionSort method)

// Method selectionSort takes the array as input, and sorts it; in turn, it calls two more methods

public static void selectionSort(int[] myArray) {
    for (int index = 0; index < myArray.length-1; index++) {
        // calling method "getIndexOfSmallest" with two inputs;
        // then, store return integer value
        int indexOfNextSmallest = getIndexOfSmallest(index, myArray);

        // calling method "interchange" with three arguments
        interchange(index, indexOfNextSmallest, myArray);
    }
}

Selection Sort – part3 (getIndexOfSmallest)

private static int getIndexOfSmallest(int startIndex, int[] a) {
    int min = a[startIndex];
    int indexOfMin = startIndex;

    for (int index = startIndex + 1; index < a.length; index++) {
        if (a[index] < min) {
            min = a[index];
            indexOfMin = index;
        }
    }
    return indexOfMin;
}
Selection Sort – part4 (interchange)

// Method interchange used to swap the two array elements

private static void interchange(int i, int j, int[] a) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp; // original value of a[i]
}

Insertion Sort

• Take an unsorted list and build a final sorted list by adding in one item at a time (we humans sort like this too)

• Insert each new item into an already sorted list

• Each unsorted element is inserted at the appropriate spot in the sorted subset until the list is sorted
Insertion Sort: General Algorithm

- Sort the first two values (swap, if necessary)
- Repeat:
  - insert list's next value into the appropriate position relative to the first ones (which are already sorted)
- Each time insertion made, number of values in the sorted subset increases by one
- Other values in array shift to make room for inserted elements

Insertion Sort at work

98 -→ 68 83 74 93

68 -→ 98 -→ 83 74 93

68 83 -→ 98 -→ 74 93

68 74 83 98 -→ 93

68 74 83 93 98 SORTED!
Insertion Sort

- Outer loop controls the index in the array of the next value to be inserted

- Inner loop compares the current insert value with values stored at lower indexes

- Each iteration of the outer loop adds one more value to the sorted subset of the list, until the entire list is sorted

Sorting Things other than numbers

- characters
  - same as integers (compare with < and >)

- Strings
  - use the built-in compareTo method

- Other Objects
  - we write a compareTo method
  - use the compareTo method
Next class

• More Searching and Sorting!