Assignment Two – LaTeX Sorts

Daniel Bilotto
danielbilotto1@marist.edu

October 9, 2021

1 Main Method

.

```
//this main method prints out the number of comparisons from all the sorts
//it reads in the array from magic items after every sort as well
public static void main(String[] args)
  {
          String[] list = new String[size];
          file(list);
          selectSort(list);
          file(list);
          insertSort(list);
          file(list);
          System.out.println("Merge sort comparisons:" + MergeSort.sort(list));
          file(list);
          System.out.println("Quick sort comparisons:"
          + QuickSort.sort(list, 0, list.length -1));

  }//main
```

2 File Method

.

```
  //this is basically the same file as lab 1
  //all it does is read in the items from magic items.
```

```
public static void file(String[] items)
 {
         String fileName = null;
         String line = null;

           try
           {
             fileName = "magicitems.txt";

             File theFile = new File(fileName);

             Scanner input = new Scanner(theFile);

             for (int i = 0; i<size; i++)
             {
                 line = input.nextLine().toLowerCase();
                 items[i] = line;
             }

             input.close();
             keyboard.close();
           }//try

           catch(Exception ex)
           {
             System.out.println("Oops, something went wrong!");
           }//catch

           //this sends back a message if something goes wrong in
           importing the text into the array from magic items
 }//file
```

3 Selection Sort Method

. This method is the selection sort method. All it does

```
public static void selectSort(String[] items)
  {
         int n = items.length;
         int compare = 0;
         for (int i = 0; i < n - 1; i++)
         {
                 int smallPos = i;
                 for (int j = smallPos + 1; j < n; j++)
                 {
```

```
                        compare++;
                        if (items[j].compareTo(items[smallPos]) < 1)
                        {
                                smallPos = j;
                        }
                }
                String swap = items[i];
                items[i] = items[smallPos];
                items[smallPos] = swap;
        }
        System.out.println("Selection sort comparisons:" + compare);
}//SS
```

4 Insert Sort Method

.

```
public static void insertSort(String[] items)
  {
        int n = items.length;
        int compare = 0;

        for (int i = 1; i < n ; i++)
        {
                String key = items[i];
                int j = i - 1;
                while ((j > -1) && (items[j].compareTo(key) > 1))
                {
                        compare++;
                        items[j + 1] = items [j];
                        j--;
                }
                items[j+1] = key;
        }

        System.out.println("Insert sort comparisons:" + compare);
}//IS
```

5 Merge Sort Class

.

```
public class MergeSort
{
        static int compare = 0;
```

```
public static int sort(String[] items)
{
        if (items.length > 1)
        {
                String[] left = new String[items.length / 2];
                String[] right =
                new String[items.length - items.length / 2];

                for (int i = 0; i < left.length; i++)
                {
        left[i] = items[i];
    }

    for (int i = 0; i < right.length; i++)
    {
        right[i] = items[i + items.length / 2];
        compare++;
    }

    sort(left);
    sort(right);
    merge(items, left, right);
        }
        return compare;
}//sort

public static void merge(String[] items, String[] left, String[] right)
{
        int i = 0;
        int j = 0;

        for (int k = 0; k < items.length; k++)
        {
                if (j >= right.length || (i < left.length &&
                left[i].compareToIgnoreCase(right[j]) < 1))
                {
                        items[k] = left[i];
                        i++;
                        compare++;
                }
                else
                {
                        items[k] = right[j];
                        j++;
                        compare++;
                }
```

```
                }
        }//merge
}
```

6 Quick Sort Method

.

```
public class QuickSort
{
        static int compare = 0;

        public static int sort(String[] items, int begin, int end)
        {
                if (begin < end)
                {
                        int partitionInd = partition(items, begin, end);

                        sort(items, begin, partitionInd - 1);
                        sort(items, partitionInd + 1, end);
                }

                return compare;
        }

        public static int partition(String[] array, int begin, int end)
        {
                String pivot = array[end];
                int i = (begin - 1);

                for (int j = begin; j < end; j++)
                {
                        if (array[j].compareTo(pivot) < 1)
                        {
                                i++;
                                String swap = array[i];
                                array[i] = array[j];
                                array[j] = swap;
                        }
                        compare++;
                }

                String swapTwo = array[i + 1];
                array[i+1] = array[end];
                array[end] = swapTwo;
```

```
                return i + 1;
        }//partition
}
```