CS 1083

Module 8 Assignment

By Ngoc Phuong Anh Nguyen - 3712361

July 23rd 2021

Source Code:

Sorter.java:

```
/**
 * This class represents a Lottery Draw.
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */

public class Sorter<SomeType extends Comparable<SomeType>>
{
    /**
     *This method sorts elements in an array from smallest to largest.
     * @param array The unsorted array.
     */
    public void selectionSort(SomeType[] array)
    {
        int n = array.length;
        SomeType temp;

        for(int i = 0; i < n - 1; i++)
        {
            int max = i;

            for(int j = i + 1; j < n; j++)
            {
                if(array[max] != null && array[max].compareTo(array[j]) > 0)
            {
                max = j;
            }
        }

        temp = array[max];
        array[max] = array[i];
        array[i] = temp;
    }
}

/**
 * Merges two adjacent subrange of an array
 * @param array The array with entries to be merged
 * @param from Index of first element of the first range
 * @param mid Index of last element of the first range
 * @param to Index of first element of the second range
 */
@SuppressWarnings("unchecked")
public void merge(SomeType[] array, int from, int mid, int to)
{
    int n = to - from + 1;

    Object[] b = new Object[n];
    int i1 = from;
    int i2 = mid + 1;
int j = 0;

if(array[i1] != null && array[i2] != null)
{
    while(i1 <= mid && i2 <= to)
    {
        if(array[i1].compareTo(array[i2]) <= 0)
        {
            b[j] = array[i1];
            i1++;
        }
        else
        {
            b[j] = array[i2];
            i2++;
        }
        j++;
    }

    while(i1 <= mid)
    {
        b[j] = array[i1];
        i1++;
        j++;
    }
```

```
        while(i2 <= to)
        {
            b[j] = array[i2];
            i2++;
            j++;
        }

        for (j = 0; j < n; j++)
            array[from + j] = (SomeType) b[j];
    }
}

/**
 * Sorts a range within an array with merge sort
 * @param array The array to be sorted
 * @param from The first index of the range
 * @param to The last index of the range
 */
public void mergeSort(SomeType[] array, int from, int to)
{
    if (from == to) return;

    int mid = (from + to) / 2;

    mergeSort(array, from, mid);
    mergeSort(array, mid + 1, to);
```

```
        merge(array, from, mid, to);
    }

    /**
     * Convenience method for the recursive mergeSort
     */
    public void mergeSort(SomeType[] array,int qty)
    {
        if (qty < 2 || qty > array.length)
            return;
        mergeSort(array, 0, array.length - 1);
    }
}
```

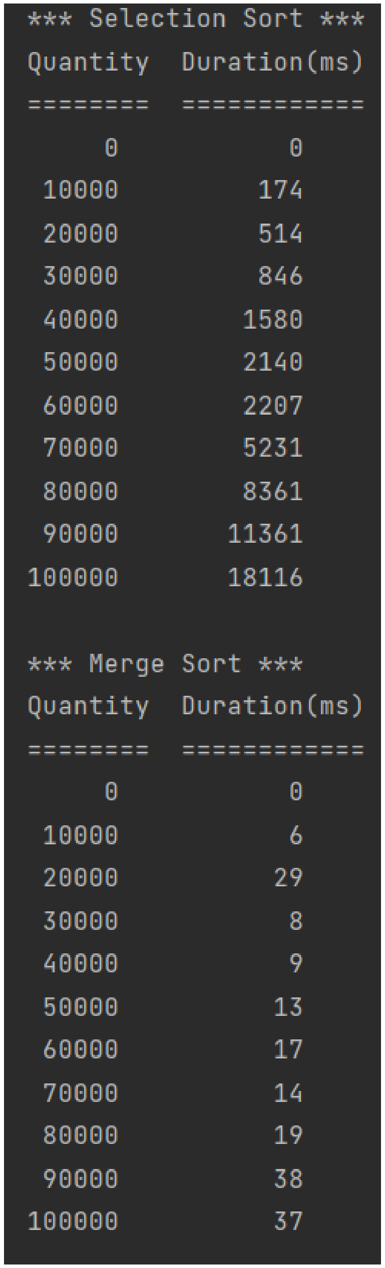
TimeTest.java:

```
/**
 * This is a driver program.
 * @author Ngoc Phuong Anh Nguyen - 3712361
 */

public class TimeTest
{
    public static void main(String[] args)
    {
        int i = 0;
```

```
long[] selectionSort = new long[11];
long[] mergeSort = new long[11];
int[] quantity = new int[11];
int count = 0;

while (i <= 100000)
{
    ComparableDraw comparableDraw = new ComparableDraw(i);
    for (int j = 0; j <= i; j++)
    {
        ComparableTicket comparableTicket = new ComparableTicket((int)
                (Math.random() * (9999 - 1000 + 1) + 1000));
        comparableDraw.addTicket(comparableTicket);
    }

    Sorter<ComparableTicket> ticketSorter = new Sorter<ComparableTicket>();

    long beforeMerge = System.currentTimeMillis();
    ticketSorter.mergeSort(comparableDraw.getTickets(),
            comparableDraw.getTicketQuantity());
    long afterMerge = System.currentTimeMillis();
    mergeSort[count] = afterMerge - beforeMerge;

    long beforeSelection = System.currentTimeMillis();
    ticketSorter.selectionSort(comparableDraw.getTickets());
    long afterSelection = System.currentTimeMillis();
```

```
            selectionSort[count] = afterSelection - beforeSelection;

            quantity[count] = comparableDraw.getTicketQuantity();
            count++;
            i += 10000;
        }
        System.out.println("*** Selection Sort ***\n" +
                "Quantity  Duration(ms)\n========  ============");
        for(i = 0; i < selectionSort.length; i++)
        {
            System.out.printf("%6d%12d\n", quantity[i],selectionSort[i]);
        }

        System.out.println("\n*** Merge Sort ***\n" +
                "Quantity  Duration(ms)\n========  ============");
        for(i = 0; i < mergeSort.length; i++)
        {
            System.out.printf("%6d%12d\n", quantity[i],mergeSort[i]);
        }
    }
}
```

Output And Chat:



```
*** Selection Sort ***
Quantity  Duration(ms)
========  ============
     0            0
 10000          174
 20000          514
 30000          846
 40000         1580
 50000         2140
 60000         2207
 70000         5231
 80000         8361
 90000        11361
100000        18116

*** Merge Sort ***
Quantity  Duration(ms)
========  ============
     0            0
 10000            6
 20000           29
 30000            8
 40000            9
 50000           13
 60000           17
 70000           14
 80000           19
 90000           38
100000           37
```

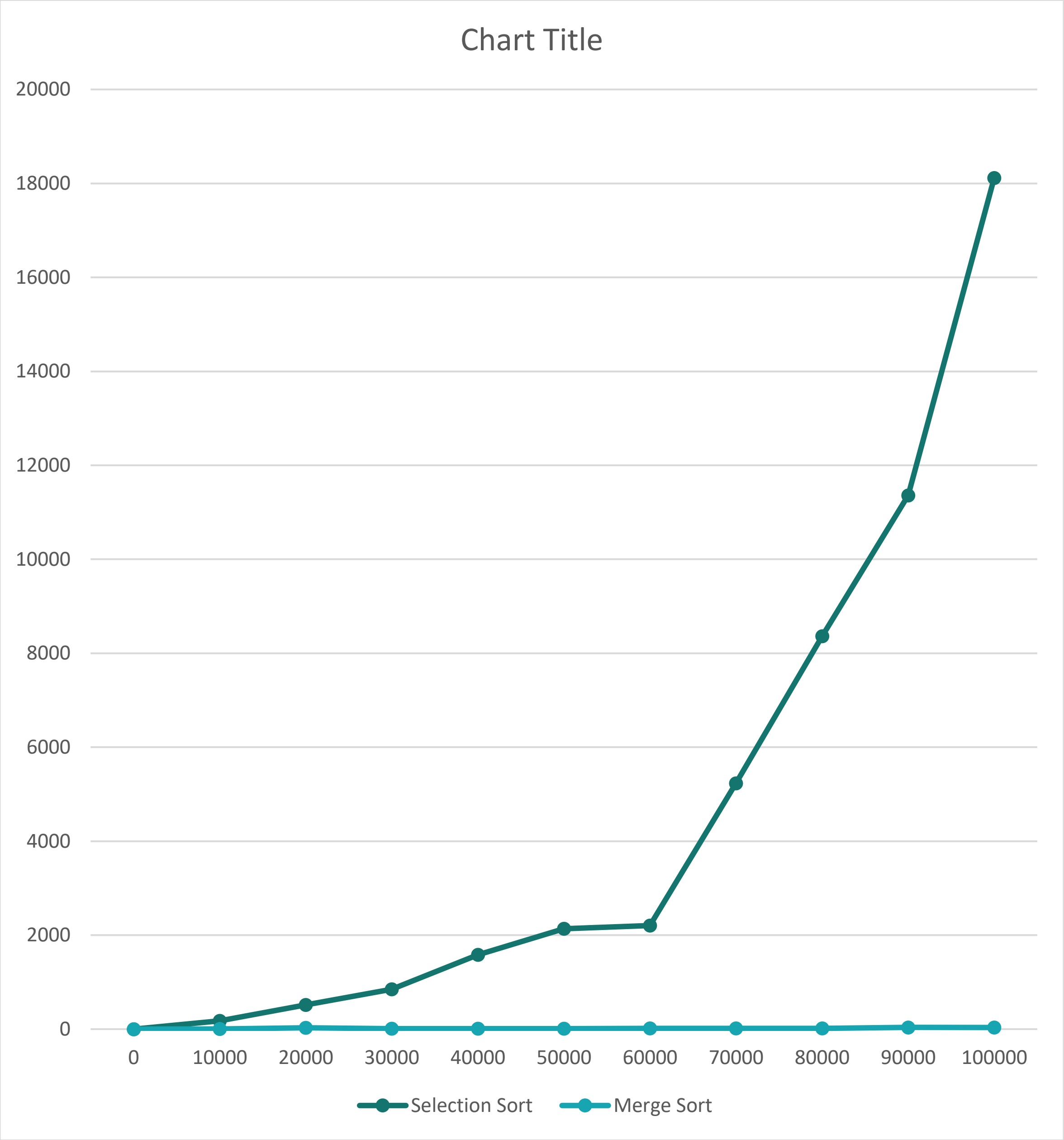


Chart Title
Selection Sort
Merge Sort