CS 1803

Module 8 Assignment

Name: Omar Habib

ID: 3742418

- **Complete code for your updated Sorter and TimeTest classes.**
- **Sample output from executing TimeTest.**
- **The chart you produced based on that sample output.**

Sorter Class:

```
/**
 * @author Omar Habib ID:3742418
 * A Sorter class that find the largest element and swap it with the element at the end
 *of the unsorted portion of the array.
 */

public class Sorter <SomeType extends Comparable<SomeType>> {

    public void selectionSort(SomeType[] array, int objNum){

        for(int pass = 0; pass<objNum-1; pass++){
            int max = pass;
            for(int i=pass+1; i<objNum; i++){
                if(array[i].compareTo(array[max])<0){
                    max = i;
                }
            }
            SomeType temp = array[pass];
            array[pass]=array[max];
```

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

public void mergeSort(SomeType[] a, int qty) {
    if (qty < 2 || qty > a.length) {
        return;
    }
    mergeSort(a, 0, qty - 1);
}

private void mergeSort(SomeType[] a, int from, int to) {
    if (from == to) {
        return;
    }
    int mid = (from + to) / 2;
    mergeSort(a, from, mid);
    mergeSort(a, mid + 1, to);
    merge(a, from, mid, to);
}

public void merge(SomeType[] a, 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;
    while (i1 <= mid && i2 <= to) {
        if (a[i1].compareTo(a[i2]) <= 0) {
            b[j] = a[i1];
```

```
                i1++;
            } else {
                b[j] = a[i2];
                i2++;
            }
            j++;
        }
        while (i1 <= mid) {
            b[j] = a[i1];
            i1++;
            j++;
        }
        while (i2 <= to) {
            b[j] = a[i2];
            i2++;
            j++;
        }
        for (j = 0; j < n; j++) {
            a[from + j] = (SomeType) b[j];
        }
    }
}
```

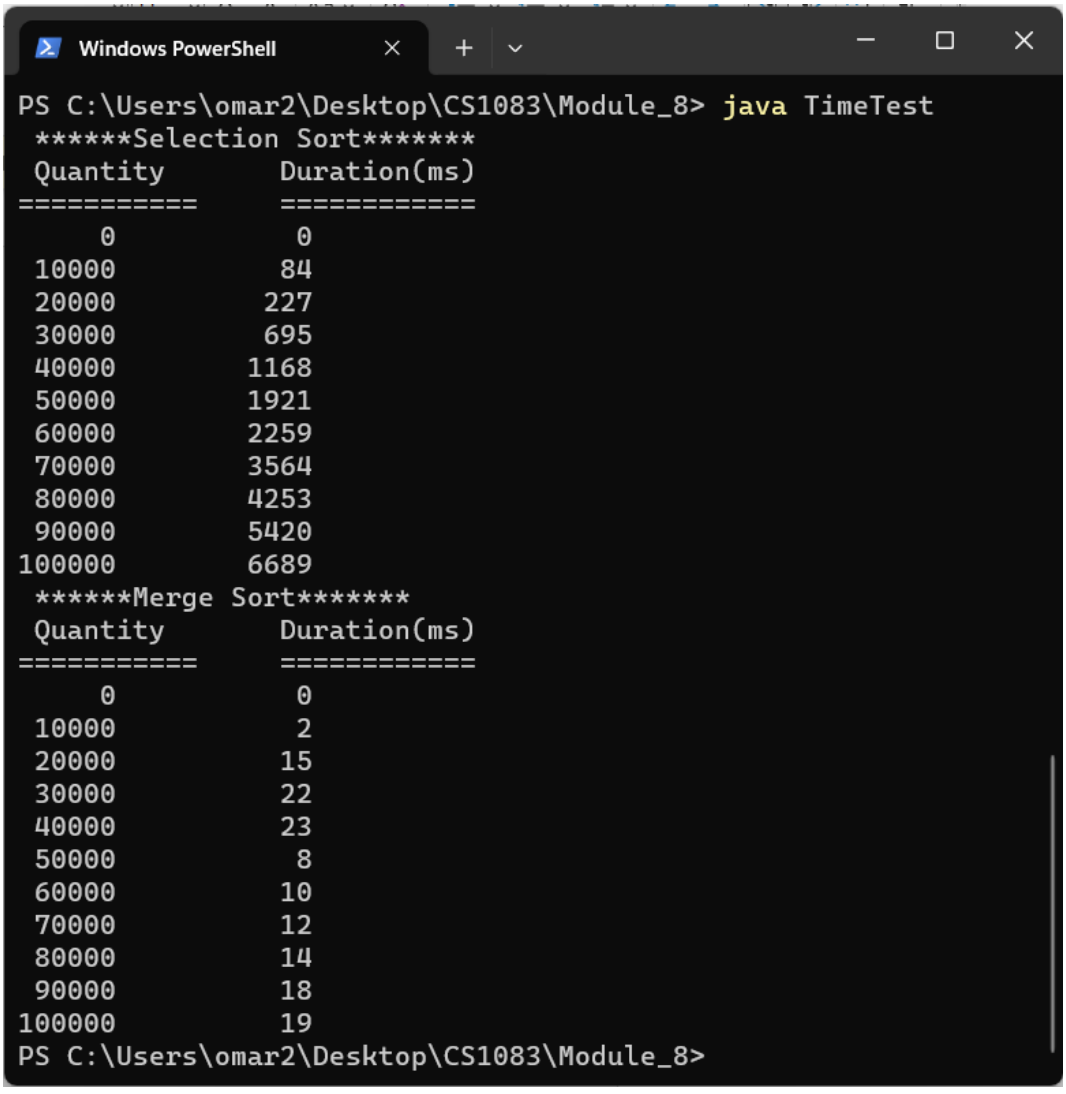
TimeTest classes:

```
import java.util.*;
import java.text.NumberFormat;

public class TimeTest {
    public static void main(String[] args) {
        int maxQuant = 100000;
        int increment = 10000;

        System.out.println(" ******Selection Sort*******");
        System.out.println(" Quantity       Duration(ms)");
        System.out.println("===========     ============");

        for (int quantity = 0; quantity <= maxQuant; quantity += increment) {
            ComparableDraw compDraw = new ComparableDraw(quantity);
            Random ran = new Random();
            long before;
            long after;
            long duration;

            for (int i = 0; i < quantity; i++) {
                int randomNum = ran.nextInt(9000) + 1000;
                ComparableTicket ticket = new ComparableTicket(randomNum);
                compDraw.addTicket(ticket);
            }
            Sorter<ComparableTicket> ticketSorter = new
Sorter<ComparableTicket>();

            before = System.currentTimeMillis();
```

```
            ticketSorter.selectionSort(compDraw.getTickets(),
compDraw.getTicketQuantity());

            after = System.currentTimeMillis();

            duration = after - before;
            System.out.printf("%6d%12d\n", quantity,duration);
        }

        System.out.println(" ******Merge Sort*******");
        System.out.println(" Quantity       Duration(ms)");
        System.out.println("===========     ============");

        for(int quantity = 0; quantity <= maxQuant; quantity += increment){
            ComparableDraw compDraw2 = new ComparableDraw(quantity);

            Random ran = new Random();
            long before;
            long after;
            long duration;

            for (int i = 0; i < quantity; i++) {
                int randomNum = ran.nextInt(9000) + 1000;
                ComparableTicket ticket = new ComparableTicket(randomNum);
                compDraw2.addTicket(ticket);
            }
            Sorter<ComparableTicket> ticketSorter2 = new
Sorter<ComparableTicket>();
            before = System.currentTimeMillis();
```

```
            ticketSorter2.mergeSort(compDraw2.getTickets(),
compDraw2.getTicketQuantity());
            after = System.currentTimeMillis();
            duration = after - before;
            System.out.printf("%6d%12d\n", quantity,duration);
        }
    }
}
```

Sample output from executing TimeTest:



The chart you produced based on that sample output: