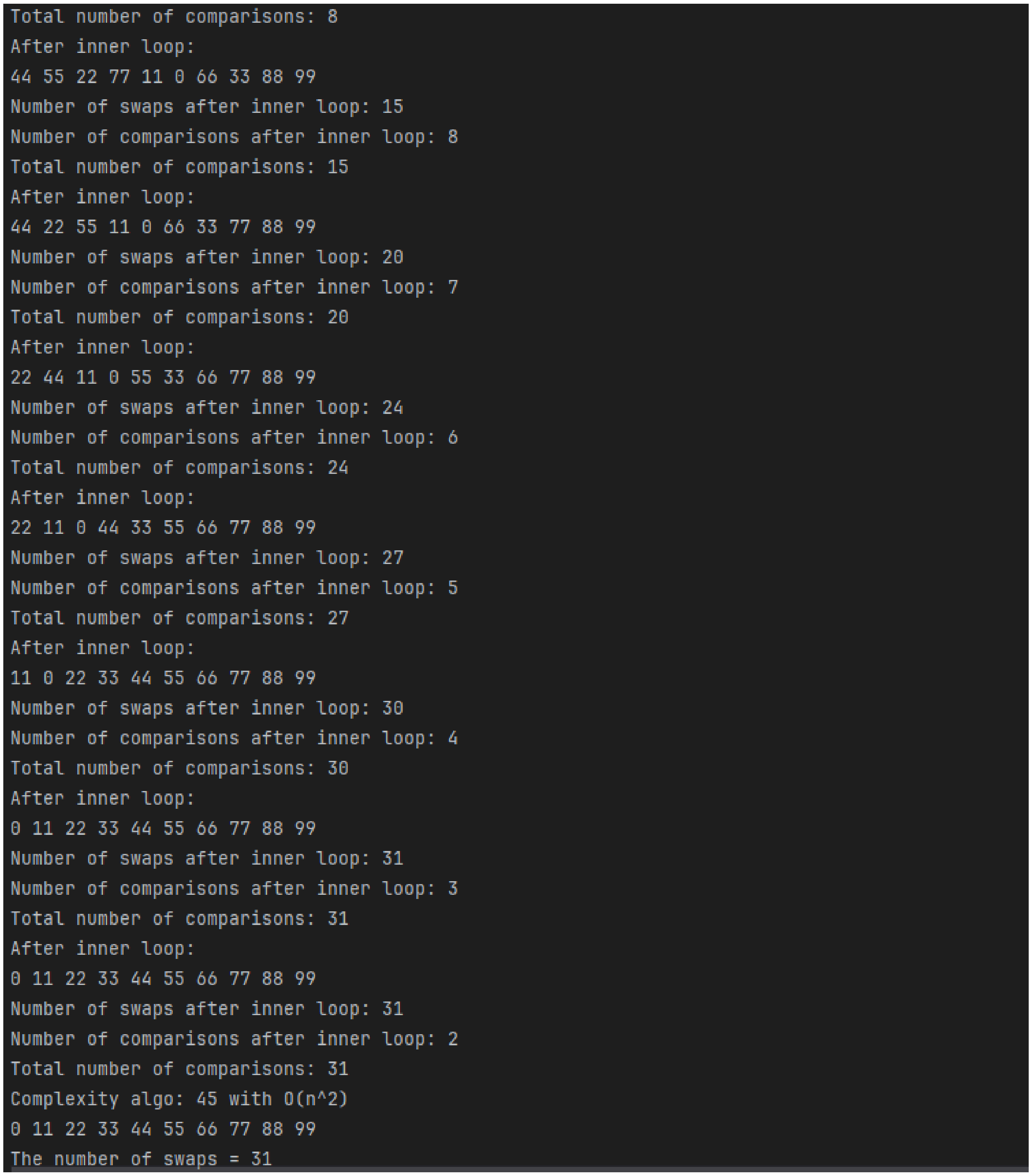
ITITWE23014
Lê Thành Danh
Week 2 - DSA's Lab - Simple Sorting
Note: I switch to IntelliJ IDE so if my code is kinda off or different compared with the previous lab, please ignore.

Problem 1: BubbleSortApp.java
- Trace the algorithm (display the array inside after inner or outer loop)
- Display the number of swaps after the inner loop
- Display the number of comparisons after the inner loop and the total number of comparisons, and
estimate the algorithms' complexity (n*(n-1)/2, O(n^2))

```java
package Problem_1;

// bubbleSort.java
// demonstrates bubble sort
// to run this program: C>java Problem_1.BubbleSortApp
////////////////////////////////////////////////////////////////
class ArrayBub {
  private long[] a; // ref to array a
  private int nElems; // number of data items
  private int nSwaps; // number of swaps
  // --------------------------------------------------------------

  public ArrayBub(int max) // constructor
  {
     a = new long[max]; // create the array
     nElems = 0; // no items yet
     nSwaps = 0; // no swaps yet
  }

  // --------------------------------------------------------------
  public void insert(long value) // put element into array
  {
     a[nElems] = value; // insert it
     nElems++; // increment size
  }

  // --------------------------------------------------------------
  public void display() // displays array contents
  {
     for (int j = 0; j < nElems; j++) // for each element,
        System.out.print(a[j] + " "); // display it
     System.out.println("");
  }

  // --------------------------------------------------------------
  public void bubbleSort() {
     int out, in;
     int comparisons = 0;
     int totalComparisons = 0;

     for (out = nElems - 1; out > 1; out--) {
         for (in = 0; in < out; in++) {
             comparisons++;
             if (a[in] > a[in + 1]) {
                 swap(in, in + 1);
                 totalComparisons++;
             }
         }
         System.out.println("After inner loop:");
         display();
         System.out.println("Number of swaps after inner loop: " + nSwaps);
         System.out.println("Number of comparisons after inner loop: " +
comparisons);
         System.out.println("Total number of comparisons: " +
totalComparisons);

         comparisons = 0;
     }
     System.out.println("Complexity algo: " + ((nElems*(nElems-1)/2)) +" with
O(n^2)");


 }

  private void swap(int one, int two) {
     long temp = a[one];
     a[one] = a[two];
     a[two] = temp;

     nSwaps++; // increase number of swap by 1
  }

  public int getSwapNumber() {
     return nSwaps;
  }
  // --------------------------------------------------------------
} // end class Problem_1.ArrayBub
 ////////////////////////////////////////////////////////////////

class BubbleSortApp {
  public static void main(String[] args) {
     int maxSize = 100; // array size
     ArrayBub arr; // reference to array
     arr = new ArrayBub(maxSize); // create the array
```

```java
     arr.insert(77); // insert 10 items
     arr.insert(99);
     arr.insert(44);
     arr.insert(55);
     arr.insert(22);
     arr.insert(88);
     arr.insert(11);
     arr.insert(00);
     arr.insert(66);
     arr.insert(33);

     arr.display(); // display items

     arr.bubbleSort(); // bubble sort them

     arr.display(); // display them again

     // display the number of swaps
     System.out.println("The number of swaps = " + arr.getSwapNumber());

  } // end main()
} // end class Problem_1.BubbleSortApp
 ////////////////////////////////////////////////////////////////
```
Output:



```
Total number of comparisons: 8
After inner loop:
44 55 22 77 11 0 66 33 88 99
Number of swaps after inner loop: 15
Number of comparisons after inner loop: 8
Total number of comparisons: 15
After inner loop:
44 22 55 11 0 66 33 77 88 99
Number of swaps after inner loop: 20
Number of comparisons after inner loop: 7
Total number of comparisons: 20
After inner loop:
22 44 11 0 55 33 66 77 88 99
Number of swaps after inner loop: 24
Number of comparisons after inner loop: 6
Total number of comparisons: 24
After inner loop:
22 11 0 44 33 55 66 77 88 99
Number of swaps after inner loop: 27
Number of comparisons after inner loop: 5
Total number of comparisons: 27
After inner loop:
11 0 22 33 44 55 66 77 88 99
Number of swaps after inner loop: 30
Number of comparisons after inner loop: 4
Total number of comparisons: 30
After inner loop:
0 11 22 33 44 55 66 77 88 99
Number of swaps after inner loop: 31
Number of comparisons after inner loop: 3
Total number of comparisons: 31
After inner loop:
0 11 22 33 44 55 66 77 88 99
Number of swaps after inner loop: 31
Number of comparisons after inner loop: 2
Total number of comparisons: 31
Complexity algo: 45 with O(n^2)
0 11 22 33 44 55 66 77 88 99
The number of swaps = 31
```

With this

```
        System.out.println("Number of swaps after inner loop: " + nSwaps);
        System.out.println("Number of comparisons after inner loop: " +
comparisons);
        System.out.println("Total number of comparisons: " +
totalComparisons);
```

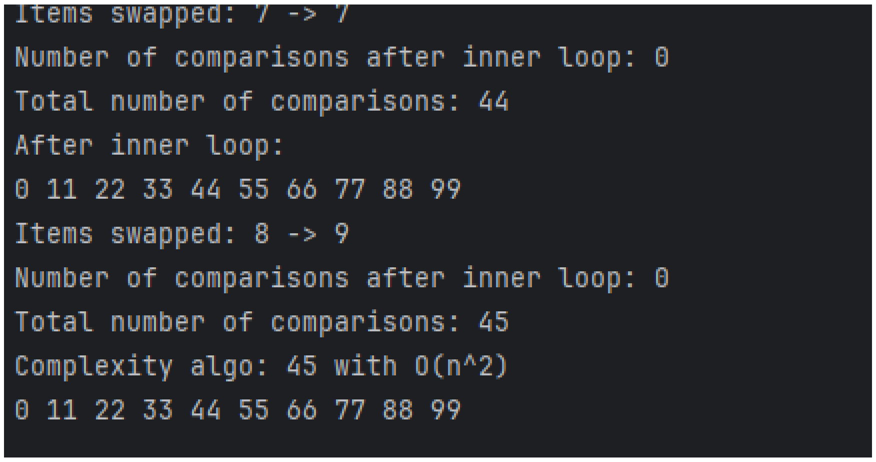
I did what the tasks require, just a simple declaration, nothing fancy.

Problem 2: SelectSortApp.java
- Trace the algorithm (display the array after the inner loop)
- Print the items that are swapped. Are swaps always needed?
Answer: No, there could be no swaps in one iteration.
- Display the number of comparisons after the inner loop and the total number of comparisons, and
estimate the algorithms' complexity (n*(n-1)/2, O(n^2))

```
package Problem_2;

// selectSort.java
// demonstrates selection sort
// to run this program: C>java Problem_2.SelectSortApp
////////////////////////////////////////////////////////////////
class ArraySel
  {
  private long[] a;                    // ref to array a
  private int nElems;                  // number of data items
//--------------------------------------------------------------
  public ArraySel(int max)             // constructor
     {
     a = new long[max];                    // create the array
     nElems = 0;                           // no items yet
     }
//--------------------------------------------------------------
  public void insert(long value)       // put element into array
     {
     a[nElems] = value;                // insert it
     nElems++;                         // increment size
     }
//--------------------------------------------------------------
  public void display()                // displays array contents
     {
     for(int j=0; j<nElems; j++)          // for each element,
        System.out.print(a[j] + " ");     // display it
     System.out.println("");
     }
//--------------------------------------------------------------
public void selectionSort() {
  int out, in, min;
  int comparisons = 0;
  int totalComparisons = 0;

  for(out=0; out<nElems-1; out++) {
      min = out;
      for(in=out+1; in<nElems; in++) {
          comparisons++;
          if(a[in] < a[min]) {
              min = in;
          }
      }
      swap(out, min);
      totalComparisons += comparisons;
      comparisons = 0;
      System.out.println("After inner loop:");
      display();
      System.out.println("Items swapped: " + out + " -> " + min);
      System.out.println("Number of comparisons after inner loop: " +
comparisons);
      System.out.println("Total number of comparisons: " + totalComparisons);
  }
  System.out.println("Complexity algo: " + ((nElems*(nElems-1)/2)) +" with
O(n^2)");


}
//----------------------------------------------------------------
  private void swap(int one, int two)
     {
     long temp = a[one];
     a[one] = a[two];
     a[two] = temp;
     }
//----------------------------------------------------------------
  }  // end class Problem_2.ArraySel
////////////////////////////////////////////////////////////////
class SelectSortApp
  {
  public static void main(String[] args)
     {
     int maxSize = 100;            // array size
     ArraySel arr;                 // reference to array
     arr = new ArraySel(maxSize);  // create the array

     arr.insert(77);               // insert 10 items
     arr.insert(99);
     arr.insert(44);
     arr.insert(55);
     arr.insert(22);
     arr.insert(88);
     arr.insert(11);
     arr.insert(00);
     arr.insert(66);
     arr.insert(33);

     arr.display();                // display items
```

```
    arr.selectionSort();           // selection-sort them

    arr.display();                 // display them again
    }  // end main()
  }  // end class Problem_2.SelectSortApp
////////////////////////////////////////////////////////////////
```

Output:



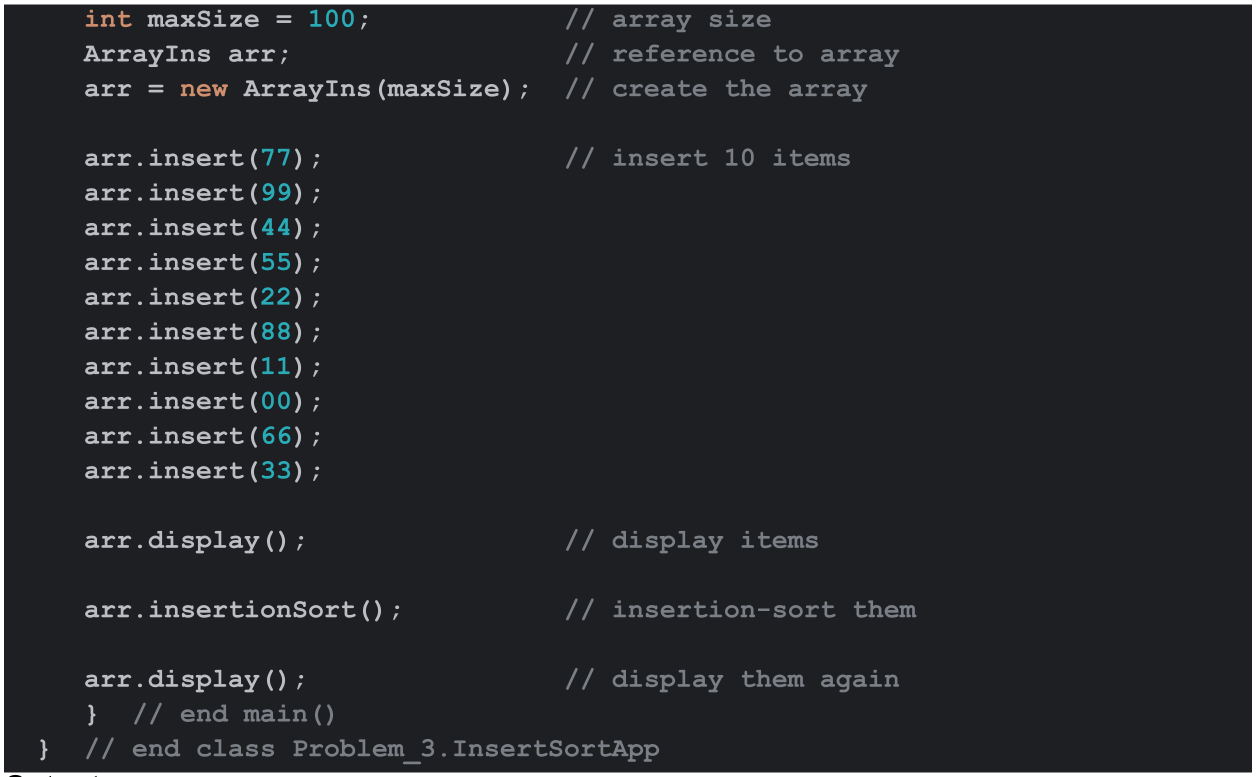
```
Items swapped: 7 -> 7
Number of comparisons after inner loop: 0
Total number of comparisons: 44
After inner loop:
0 11 22 33 44 55 66 77 88 99
Items swapped: 8 -> 9
Number of comparisons after inner loop: 0
Total number of comparisons: 45
Complexity algo: 45 with O(n^2)
0 11 22 33 44 55 66 77 88 99
```

Pretty similar to the first one, this basically just print out the log,swaps made and fixed Complexity.

Problem 3: InsertSortApp.java
- Trace the algorithm (display the array after each pass of the outer loop)
- Display the number of passes of the inner loop and total number of passes, and estimate the
algorithms' complexity (n*(n-1)/4, O(n^2))

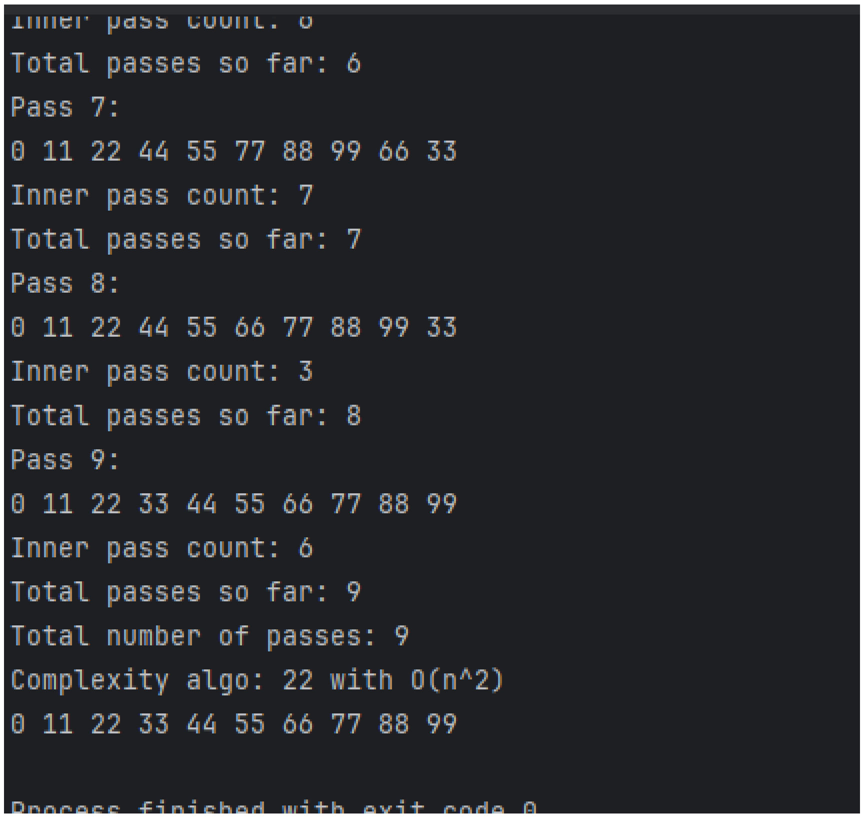
```
package Problem_3;

// insertSort.java
// demonstrates insertion sort
// to run this program: C>java Problem_3.InsertSortApp
//--------------------------------------------------------------
class ArrayIns
  {
  private long[] a;                 // ref to array a
  private int nElems;               // number of data items
//--------------------------------------------------------------
  public ArrayIns(int max)          // constructor
     {
     a = new long[max];                 // create the array
     nElems = 0;                        // no items yet
     }
//--------------------------------------------------------------
  public void insert(long value)    // put element into array
```

```java
     {
     a[nElems] = value;             // insert it
     nElems++;                      // increment size
     }
//----------------------------------------------------------------
  public void display()             // displays array contents
     {
     for(int j=0; j<nElems; j++)       // for each element,
        System.out.print(a[j] + " ");  // display it
     System.out.println("");
     }
//----------------------------------------------------------------
public void insertionSort()
{
   int in, out;
   int innerPassCount = 0;

   for(out=1; out<nElems; out++)
   {
       long temp = a[out];
       in = out;
       innerPassCount = 0;
       while(in>0 && a[in-1] >= temp)
       {
           a[in] = a[in-1];
           --in;
           innerPassCount++;
       }
       a[in] = temp;

       System.out.println("Pass " + (out) + ":");
       display();
       System.out.println("Inner pass count: " + innerPassCount);
       System.out.println("Total passes so far: " + (out));
   }

   System.out.println("Total number of passes: " + (nElems - 1));
   System.out.println("Complexity algo: " + ((nElems*(nElems-1)/4)) +" with
O(n^2)");

}

//----------------------------------------------------------------
  }  // end class Problem_3.ArrayIns
////////////////////////////////////////////////////////////////
class InsertSortApp
  {
  public static void main(String[] args)
     {
```



```
      int maxSize = 100;            // array size
      ArrayIns arr;                 // reference to array
      arr = new ArrayIns(maxSize);  // create the array

      arr.insert(77);               // insert 10 items
      arr.insert(99);
      arr.insert(44);
      arr.insert(55);
      arr.insert(22);
      arr.insert(88);
      arr.insert(11);
      arr.insert(00);
      arr.insert(66);
      arr.insert(33);

      arr.display();                // display items

      arr.insertionSort();          // insertion-sort them

      arr.display();                // display them again
      }  // end main()
   }  // end class Problem_3.InsertSortApp
```

Output:



```
Inner pass count: 6
Total passes so far: 6
Pass 7:
0 11 22 44 55 77 88 99 66 33
Inner pass count: 7
Total passes so far: 7
Pass 8:
0 11 22 44 55 66 77 88 99 33
Inner pass count: 3
Total passes so far: 8
Pass 9:
0 11 22 33 44 55 66 77 88 99
Inner pass count: 6
Total passes so far: 9
Total number of passes: 9
Complexity algo: 22 with O(n^2)
0 11 22 33 44 55 66 77 88 99


Process finished with exit code 0
```

Similar to the other 2 swap algorithm code, I just copied and pasted the code, the only different is the complexity, it's not

```
  System.out.println("Complexity algo: " + ((nElems*(nElems-1)/2)) +" with
O(n^2)");
```
But
```
   System.out.println("Complexity algo: " + ((nElems*(nElems-1)/4)) +" with
O(n^2)");
```

Problem 4
Create an array of integer numbers, fill the array with random data and print the number of comparisons,
copies, and swaps made for sorting 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 50000
items and fill in the table below. Analyze the trend for the three different algorithms.

I coded the analysis of this question, just for you know, my computer crashed twice in the debugging state, so I actually code it.
```
package Problem_4;

import java.util.Random;

public class SortingOperations {

   public static void bubbleSort(int[] arr) {
       int comparisons = 0;
       int swaps = 0;
       int n = arr.length;

       for (int i = 0; i < n - 1; i++) {
           for (int j = 0; j < n - 1 - i; j++) {
               comparisons++;
               if (arr[j] > arr[j + 1]) {
                   int temp = arr[j];
                   arr[j] = arr[j + 1];
                   arr[j + 1] = temp;
                   swaps++;
               }
           }
       }

       System.out.println("Size: " + arr.length + " | Bubble Sort Comparisons:
" + comparisons + " | Swaps: " + swaps + " | Total Operations: " +
(comparisons + swaps));
   }

   public static void insertionSort(int[] arr) {
       int comparisons = 0;
       int swaps = 0;
       int n = arr.length;
```

```java
       for (int i = 1; i < n; i++) {
           int key = arr[i];
           int j = i - 1;

           comparisons++;
           while (j >= 0 && arr[j] > key) {
               arr[j + 1] = arr[j];
               swaps++;
               j--;
           }
           arr[j + 1] = key;
       }

       System.out.println("Size: " + arr.length + " | Insertion Sort
Comparisons: " + comparisons + " | Swaps: " + swaps + " | Total Operations: "
+ (comparisons + swaps));
   }

   public static void selectionSort(int[] arr) {
       int comparisons = 0;
       int swaps = 0;
       int n = arr.length;

       for (int i = 0; i < n - 1; i++) {
           int minIndex = i;
           for (int j = i + 1; j < n; j++) {
               comparisons++;
               if (arr[j] < arr[minIndex]) {
                   minIndex = j;
               }
           }
           if (minIndex != i) {
               int temp = arr[minIndex];
               arr[minIndex] = arr[i];
               arr[i] = temp;
               swaps++;
           }
       }

       System.out.println("Size: " + arr.length + " | Selection Sort
Comparisons: " + comparisons + " | Swaps: " + swaps + " | Total Operations: "
+ (comparisons + swaps));
   }

   public static int[] generateRandomArray(int size) {
       Random random = new Random();
       int[] arr = new int[size];
       for (int i = 0; i < size; i++) {
           arr[i] = random.nextInt(100000);
```

```
        }
        return arr;
    }

    public static void main(String[] args) {
        int[] sizes = {10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000,
50000};

        System.out.println("Size\t\tBubble Sort\t\tInsertion Sort\t\tSelection
Sort");

System.out.println("-------------------------------------------------------
---");

        for (int size : sizes) {
            System.out.println("\nSorting array of size: " + size);

            int[] bubbleArr = generateRandomArray(size);
            int[] insertionArr = generateRandomArray(size);
            int[] selectionArr = generateRandomArray(size);

            System.out.print("Bubble Sort: ");
            bubbleSort(bubbleArr);

            System.out.print("Insertion Sort: ");
            insertionSort(insertionArr);

            System.out.print("Selection Sort: ");
            selectionSort(selectionArr);
        }
    }
}
```

It may look simple, but it work precisely and just maybe makes the question easier, I tried to create a table, but it look uglier than I thought it would be. So what do we have to explain ? Kinda nothing, just fit the following size of random element array to any sorting method, and we got the result we wanted. (My Google Docs crashed twice, upon importing the value, I use GPT for table visualization)

Array Size	Sort Type	Comparisons	Swaps	Total Operations
10000	Bubble Sort	49995000	25134907	75129907
	Insertion Sort	9999	25169756	25179755

	Selection Sort	**49995000**	**9987**	**50004987**
15000	**Bubble Sort**	**112492500**	**56633656**	**169126156**
	Insertion Sort	**14999**	**56812400**	**56827399**
	Selection Sort	**112492500**	**14993**	**112507493**
20000	**Bubble Sort**	**199990000**	**99834246**	**299824246**
	Insertion Sort	**19999**	**99804223**	**99824222**
	Selection Sort	**199990000**	**19988**	**200009988**
25000	**Bubble Sort**	**312487500**	**155693288**	**468180788**
	Insertion Sort	**24999**	**155186705**	**155211704**
	Selection Sort	**312487500**	**24990**	**312512490**
30000	**Bubble Sort**	**449985000**	**225437576**	**675422576**
	Insertion Sort	**29999**	**224634056**	**224664055**
	Selection Sort	**449985000**	**29989**	**450014989**
35000	**Bubble Sort**	**612482500**	**306240706**	**918723206**
	Insertion Sort	**34999**	**305048296**	**305083295**
	Selection Sort	**612482500**	**34984**	**612517484**
40000	**Bubble Sort**	**799980000**	**398911623**	**1198891623**
	Insertion Sort	**39999**	**401468023**	**401508022**
	Selection Sort	**799980000**	**39987**	**800019987**
45000	**Bubble Sort**	**1012477500**	**507447687**	**1519925187**
	Insertion Sort	**44999**	**508023143**	**508068142**
	Selection Sort	**1012477500**	**44984**	**1012522484**
50000	**Bubble Sort**	**1249975000**	**622518635**	**1872493635**
	Insertion Sort	**49999**	**624382314**	**624432313**

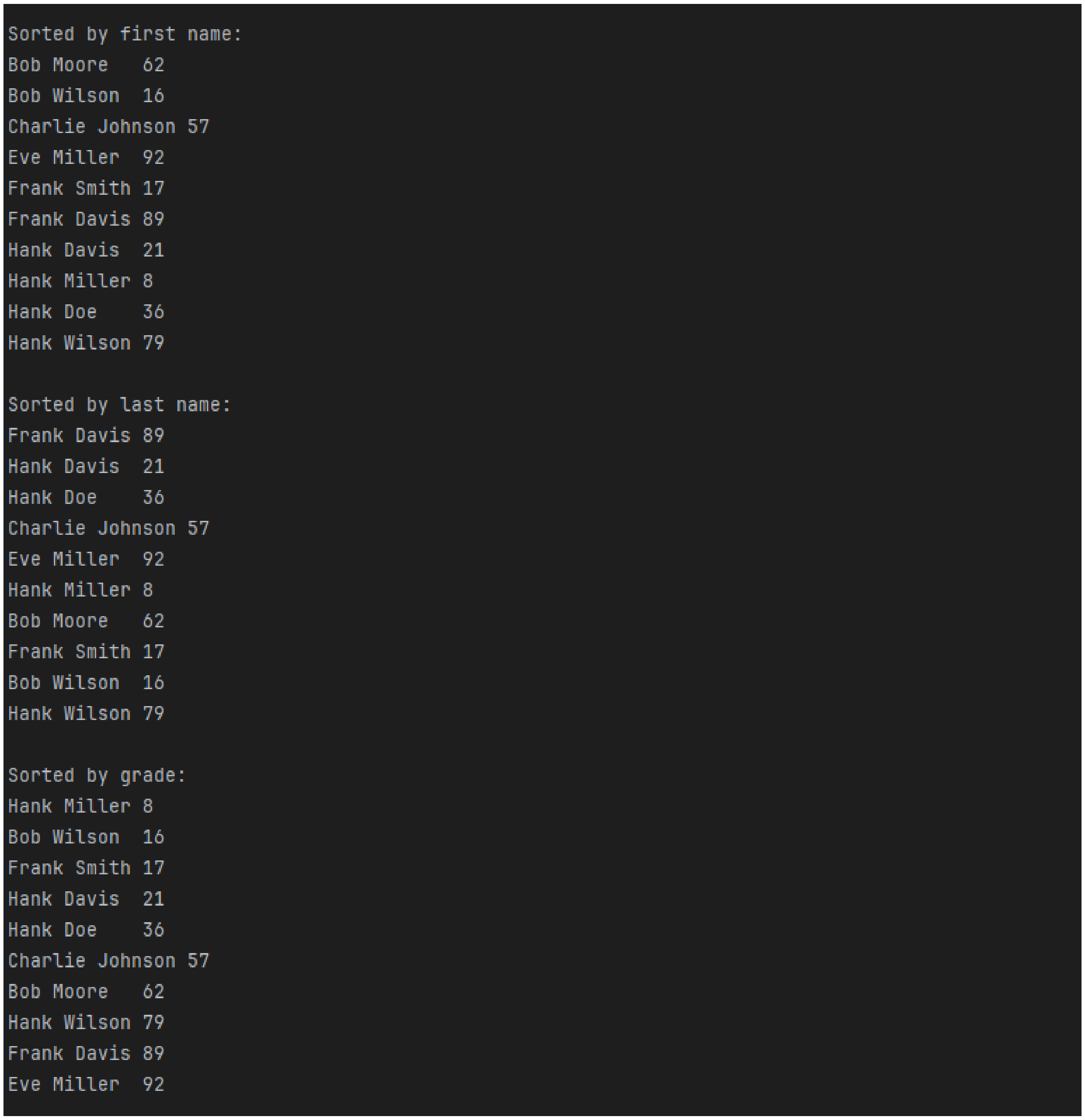
	Selection Sort	**1249975000**	**49990**	**1250024990**

Problem 5: ObjectSortApp.java (sort the array by first name or by age)
(Option 1) Given the class Student.java that has variables of first name, last name, grade
Add a main() method and add create an array of 10 students
Add methods to sort the array by first name, last name, and by grade.

```
package Problem_5;

import java.util.Random;

public class Student {
   private String fname, lname;
   private int grade;

   public Student(String fname, String lname, int grade) {
       this.fname = fname;
       this.lname = lname;
       this.grade = grade;
   }

   @Override
   public String toString() {
       return fname + " " + lname + "\t" + grade;
   }

   public String getFname() {
       return fname;
   }

   public String getLname() {
       return lname;
   }

   public int getGrade() {
       return grade;
   }

   public static void bubbleSortByFirstName(Student[] students) {
       int n = students.length;
       for (int i = 0; i < n - 1; i++) {
           for (int j = 0; j < n - i - 1; j++) {
               if (students[j].getFname().compareTo(students[j +
1].getFname()) > 0) {

                   Student temp = students[j];
                   students[j] = students[j + 1];
                   students[j + 1] = temp;
               }
```

```java
            }
        }
    }

    public static void bubbleSortByLastName(Student[] students) {
        int n = students.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (students[j].getLname().compareTo(students[j +
1].getLname()) > 0) {

                    Student temp = students[j];
                    students[j] = students[j + 1];
                    students[j + 1] = temp;
                }
            }
        }
    }

    public static void bubbleSortByGrade(Student[] students) {
        int n = students.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (students[j].getGrade() > students[j + 1].getGrade()) {

                    Student temp = students[j];
                    students[j] = students[j + 1];
                    students[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        Random rand = new Random();
        String[] firstNames = {"John", "Jane", "Bob", "Alice", "Charlie",
"Diana", "Eve", "Frank", "Grace", "Hank"};
        String[] lastNames = {"Smith", "Doe", "Johnson", "Williams", "Brown",
"Jones", "Miller", "Davis", "Wilson", "Moore"};
        Student[] students = new Student[10];

        for (int i = 0; i < students.length; i++) {
            String fname = firstNames[rand.nextInt(firstNames.length)];
            String lname = lastNames[rand.nextInt(lastNames.length)];
            int grade = rand.nextInt(101);
            students[i] = new Student(fname, lname, grade);
        }

        System.out.println("Before sorting:");
        for (Student student : students) {
            System.out.println(student);
        }

        bubbleSortByFirstName(students);
        System.out.println("\nSorted by first name:");
        for (Student student : students) {
            System.out.println(student);
        }

        bubbleSortByLastName(students);
        System.out.println("\nSorted by last name:");
        for (Student student : students) {
            System.out.println(student);
        }

        bubbleSortByGrade(students);
        System.out.println("\nSorted by grade:");
        for (Student student : students) {
            System.out.println(student);
        }
    }
}
```

Output:


```
Sorted by first name:
Bob Moore   62
Bob Wilson  16
Charlie Johnson 57
Eve Miller  92
Frank Smith 17
Frank Davis 89
Hank Davis  21
Hank Miller 8
Hank Doe    36
Hank Wilson 79

Sorted by last name:
Frank Davis 89
Hank Davis  21
Hank Doe    36
Charlie Johnson 57
Eve Miller  92
Hank Miller 8
Bob Moore   62
Frank Smith 17
Bob Wilson  16
Hank Wilson 79

Sorted by grade:
Hank Miller 8
Bob Wilson  16
Frank Smith 17
Hank Davis  21
Hank Doe    36
Charlie Johnson 57
Bob Moore   62
Hank Wilson 79
Frank Davis 89
Eve Miller  92
```

I created 10 random student from the array of first and last name.
Instead of using the builtin method in java for sorting, I have to make three Bubble sort method for sorting the properties of the Student object. But I do use the compareTo method or the code will be much longer.

Problem 6:

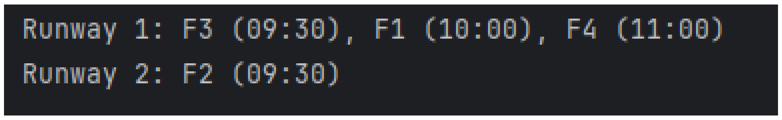
```
package Problem_6;

import java.util.*;

class Flight {
```

```java
    String flightId;
    String time;
    int priority;

    public Flight(String flightId, String time, int priority) {
        this.flightId = flightId;
        this.time = time;
        this.priority = priority;
    }

    public int getTimeInMinutes() {
        String[] parts = time.split(":");
        return Integer.parseInt(parts[0]) * 60 + Integer.parseInt(parts[1]);
    }

    @Override
    public String toString() {
        return "(" + flightId + ", " + time + ", Priority: " + priority + ")";
    }
}

public class FlightScheduler {

    public static void mergeSort(List<Flight> flights, int left, int right) {
        if (left < right) {
            int mid = (left + right) / 2;
            mergeSort(flights, left, mid);
            mergeSort(flights, mid + 1, right);
            merge(flights, left, mid, right);
        }
    }

    public static void merge(List<Flight> flights, int left, int mid, int
right) {
        int n1 = mid - left + 1;
        int n2 = right - mid;

        List<Flight> leftTemp = new ArrayList<>(n1);
        List<Flight> rightTemp = new ArrayList<>(n2);

        for (int i = 0; i < n1; i++) {
            leftTemp.add(flights.get(left + i));
        }
        for (int j = 0; j < n2; j++) {
            rightTemp.add(flights.get(mid + 1 + j));
        }

        int i = 0, j = 0;
        int k = left;
```

```
       while (i < n1 && j < n2) {
           Flight leftFlight = leftTemp.get(i);
           Flight rightFlight = rightTemp.get(j);

           if (leftFlight.priority > rightFlight.priority ||
                   (leftFlight.priority == rightFlight.priority &&
leftFlight.getTimeInMinutes() < rightFlight.getTimeInMinutes())) {
               flights.set(k, leftFlight);
               i++;
           } else {
               flights.set(k, rightFlight);
               j++;
           }
           k++;
       }

       while (i < n1) {
           flights.set(k, leftTemp.get(i));
           i++;
           k++;
       }

       while (j < n2) {
           flights.set(k, rightTemp.get(j));
           j++;
           k++;
       }
   }// di ngu

   public static List<String> scheduleFlights(List<Flight> flights, int R) {

       mergeSort(flights, 0, flights.size() - 1);

       List<List<Flight>> runways = new ArrayList<>(R);
       for (int i = 0; i < R; i++) {
           runways.add(new ArrayList<>());
       }

       List<Flight> unscheduledFlights = new ArrayList<>();
       for (Flight flight : flights) {
           boolean assigned = false;
           for (int i = 0; i < R; i++) {
               List<Flight> runwayFlights = runways.get(i);

               if (runwayFlights.isEmpty() ||
runwayFlights.get(runwayFlights.size() - 1).getTimeInMinutes() <=
flight.getTimeInMinutes()) {
                   runwayFlights.add(flight);
```

```java
                    assigned = true;
                    break;
                }
            }
            if (!assigned) {
                unscheduledFlights.add(flight);
            }
        }

        List<String> output = new ArrayList<>();

        for (int i = 0; i < R; i++) {
            List<Flight> runwayFlights = runways.get(i);
            StringBuilder sb = new StringBuilder("Runway " + (i + 1) + ": ");
            if (!runwayFlights.isEmpty()) {
                for (Flight f : runwayFlights) {
                    sb.append(f.flightId).append("
(").append(f.time).append("), ");
                }
                sb.setLength(sb.length() - 2);
            } else {
                sb.append("No flights scheduled.");
            }
            output.add(sb.toString());
        }

        if (!unscheduledFlights.isEmpty()) {
            StringBuilder unscheduled = new StringBuilder("Unscheduled Flights:
");
            for (Flight f : unscheduledFlights) {
                unscheduled.append(f.flightId).append("
(").append(f.time).append("), ");
            }
            unscheduled.setLength(unscheduled.length() - 2);
            output.add(unscheduled.toString());
        }

        return output;
    }

    public static void main(String[] args) {
        List<Flight> flights = Arrays.asList(
                new Flight("F1", "10:00", 2),
                new Flight("F2", "09:30", 1),
                new Flight("F3", "09:30", 2),
                new Flight("F4", "11:00", 1)
        );

        int runways = 2;
        List<String> scheduledFlights = scheduleFlights(flights, runways);

        scheduledFlights.forEach(System.out::println);
    }
}
```

Output:



```
Runway 1: F3 (09:30), F1 (10:00), F4 (11:00)
Runway 2: F2 (09:30)
```

It took me way longer than i imagine it would took, aside from that, I didn’t know if I done right.
implemented the mergeSort() and merge() methods to sort the list of flights based on priority and time. The merge sort algorithm splits the list into two halves, recursively sorts each half, and then merges the sorted halves together
The merge step is where we compare the flights based on priority (higher priority flights come first). If two flights have the same priority, the one scheduled earlier (based on the time) will be considered "less" and come first
Merge Sort has a time complexity of O(NlogN), where N is the number of flights
The complexity of merging two lists is O(N), where N is the size of the list being merged
I really don't know what else to explain…