Data Structures and Algorithms

Lab Exam 1

Time: 1 Hour

You may upload to D2L only Once.

<u>This exam is NOT open book</u>

<u>No internet resources may be used.</u>

Enter your full name and student number in the areas below:

Name	Student Number
Edwin Chung	101422064

Lab Day and Time:	Lab Professor’s name
2024-01-30 8:18AM	Andrew Rudder

Instructions:

1. Write your solution and test your program for the specification given
2. Copy and Paste the code for your solution in the places provided
3. Upload to blackboard (you may submit only ONCE (1 time).

Part 1:

Create a class **EventItem** that holds the location of an event called ***location***(a String), the name of the event called ***eventName*** (a String), a start hour (in 24hour format) called ***startTime*** (an integer) and an end hour (in 24 hour format) called ***endTime*** (an integer). All state values are public and a constructor is needed that takes all four parameters.

[1 marks]

Paste your code for Part 1 here:

```java
public class EventItem {

   public static String location;

   public static String eventName;

   public static int startTime;

   public static int endTime;


   public EventItem(String location, String eventName, int startTime, int endTime) {

       EventItem.location = location;

       EventItem.eventName = eventName;

       EventItem.startTime = startTime;

       EventItem.endTime = endTime;

   }

}
```

Part 2:

Create a class **DailyScheduler** that holds **EventItem** objects in an array called **dailySchedule** (The maximum size of the array is 100). It must also have a variable that holds the maximum number of EventItems(100) and another for the actual number of EventItems.

All state variables must be initialized in a single constructor that takes no parameters.

[1 marks for all necessary class state declarations]

The class **DailyScheduler** has the following operations:

void sortDescendingByEventName() - This method sorts the EventItems in descending order **using insertion sort by their eventName**. **[1 mark]**

void sortAScendingByStartTime() - This method sorts the EventItems in ascending order **using insertion sort by their startTime**. **[0.5 marks]**

void sortDescendingByLength() - This method sorts the EventItems in descending order **using selection sort by their length of time** (endTime – startTime). **[1.5 marks]**

String printDayScheduleByStartTime() - This method returns a string of all EventItems with their location, eventName, startTime and endTime in ascending order by their starting time. **[0.5 marks]**

String printDayScheduleByLongestTime() - This method returns a string of all EventItems with their location, eventName, startTime and endTime in descending order by their Longest time (endTime – startTime). **[0.5 marks]**

int binarySearch(String eventName) – This method uses binary search to return the location of an EventItem with the **eventName** matching the parameters, or -1 if not. YOU MUST ASSSUME THE ARRAY IS SORTED IN DESCENDING ORDER BY EVENTNAME. **[2 marks]**

boolean addEventItem(String location,String eventName,int startTime,int endTime) - This method must in order:

1. Check if there is space to add a new EventItem (returns false if not)
2. Sort the list in descending order by eventName. **[0.5 marks for 1 and 2]**
3. Uses **binarySearch** to check if there is an item with the same **eventName** in the array. **[0.5 mark]**
4. Creates an EventItem and **adds it to the end of the array if there is no item with the same eventName and location**. (returns false if there is a clash and true if not)**[1 marks]**

YOU MAY ASSUME EVENT NAMES ARE UNIQUE (two EventItems will not have the same eventNames).

Paste your code for Part2 here:

```
public class DailyScheduler {

   private static final int maxItems = 100;

   private EventItem[] dailySchedule;
   private int numItems;


   public DailyScheduler() {

       dailySchedule = new EventItem[maxItems];

       numItems = 0;

   }


   public void sortDescendingByEventName() {

       for (int i = 1; i < numItems; i++) {

           EventItem key = dailySchedule[i];

           int j = i - 1;

           while (j >= 0 && dailySchedule[j].eventName.compareTo(key.eventName) < 0) {

               dailySchedule[j + 1] = dailySchedule[j];

               j = j - 1;

           }

           dailySchedule[j + 1] = key;

       }

   }


   public void sortAscendingByStartTime()

   {for (int i = 1; i < numItems; i++) {

           EventItem key = dailySchedule[i];

           int j = i - 1;

           while (j >= 0 && dailySchedule[j].startTime > key.startTime) {

               dailySchedule[j + 1] = dailySchedule[j];

               j = j - 1;

           }

           dailySchedule[j + 1] = key;

       }

   }
```

```
    public void sortDescendingByLength() {
        // Selection sort for descending order by length (endTime - startTime)
        for (int i = 0; i < numItems - 1; i++) {
            int maxIdx = i;
            for (int j = i + 1; j < numItems; j++) {
                int lengthJ = dailySchedule[j].endTime - dailySchedule[j].startTime;
                int lengthMaxIdx = dailySchedule[maxIdx].endTime -
dailySchedule[maxIdx].startTime;
                if (lengthJ > lengthMaxIdx) {
                    maxIdx = j;
                }
            }
            EventItem temp = dailySchedule[maxIdx];
            dailySchedule[maxIdx] = dailySchedule[i];
            dailySchedule[i] = temp;
        }
    }


    public String printDayScheduleByStartTime() {
        sortAscendingByStartTime();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < numItems; i++) {
            EventItem item = dailySchedule[i];
            sb.append("Location: ").append(item.location)
                    .append(", Event: ").append(item.eventName)
                    .append(", Start Time: ").append(item.startTime)
                    .append(", End Time: ").append(item.endTime).append("\n");
        }
        return sb.toString();
    }


    public String printDayScheduleByLongestTime() {
        sortDescendingByLength();

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < numItems; i++) {

            EventItem item = dailySchedule[i];

            sb.append("Location: ").append(item.location)

                    .append(", Event: ").append(item.eventName)

                    .append(", Start Time: ").append(item.startTime)

                    .append(", End Time: ").append(item.endTime).append("\n");

        }

        return sb.toString();

    }


    public int binarySearch(String eventName) {

        int low = 0;

        int high = numItems - 1;


        while (low <= high) {

            int mid = low + (high - low) / 2;

            int compareResult = eventName.compareTo(dailySchedule[mid].eventName);


            if (compareResult < 0) {

                low = mid + 1;

            } else if (compareResult > 0) {

                high = mid - 1;

            } else {

                return mid; // eventName found

            }

        }


        return -1; // eventName not found

    }
```

```
   public boolean addEventItem(EventItem item) {
       if (numItems < maxItems) {
           if (binarySearch(item.eventName) == -1){
               dailySchedule[numItems] = item;
               numItems++;
               sortDescendingByEventName();
           }
           return true;
       }
       return false;
   }
}
```