CSC212 2023: Practical 2 Term 2
Due date: 5 May 2023, 10 PM
Lecturer: Mr. C. K. Baker
Total: 100 marks

Instructions

This practical will test and apply your knowledge of priority queues. You are required to program in java and use object-oriented programming concepts. Do not use built-in java methods for your tasks: write your own code. Save all your files in a compressed (.zip) folder with the following naming convention **XXYYZZZ_CSC211_2023_Practical_2_Term_2.zip** where **XXYYZZZ** corresponds to your student number. Remember to include the following information at the beginning of each java file:

```
// Surname:        <Your Surname>
// Name:           <Your Name>
// Student no:     <XXYYZZZ>
// Course:         CSC211
// Year:           2023
// Assignment:     Practical 2 Term 2
// File:           <example.java>
```

(leave a line open between this header and your first line of code)

Air Traffic Control (ATC) is a vital unit within any airport. It is a service provided by ground-based air controllers who direct aircraft on the ground and through a given section of airspace.

You have been appointed by Cape Town International Airport to manage aircraft landings and takeoffs. As part of your training, you are required to implement a priority queue using the flight data from 1 January 2023.

Question 1 [10 marks]

The following is an interface for a `class FlightNode`:

```
class FlightNode
{
      /* attributes */
      String registration_no; // flight registration number
      Date arrival_date // date and time at which flight will land
}
```

1. Create a `class FlightNode` and write code for the following

   1.1 Encapsulate/hide each attribute by adding an appropriate modifier to its definition

   1.2 A default constructor

   1.3 A loaded constructor

   1.4 Accessor methods for each attribute

   1.5 Mutator methods for each attribute

   1.6 A `toString()` method returns a formatted `String` representation of each attribute for a given flight:

   e.g. “ Flight registration no.:   ZA1234
   Arrival date: Mon Jan 02 09:02:49 SAST 2023 ”

Question 2 [45 marks]

The following is an interface for a `class PriorityQueue`:

```
class PriorityQueue
{
      /* attributes */
      int currentSize; // stores current number of elements in PQ
      FlightNode[] array; // array to store flight nodes, implements PQ
      int DEFAULT_CAPACITY = 2>>10; // maximum heap size

      /* methods */
      void clear();
      Boolean isEmpty();
      Boolean add(FlightNode x);
      FlightNode remove();
      void percolateDown(int hole);
}
```

Create a `class PriorityQueue` and write code for the following

2.1 Encapsulate/hide each attribute by adding an appropriate modifier to its definition

2.2 A default constructor

2.3 A loaded constructor

2.4 Accessor methods for each attribute

2.5 Mutator methods for each attribute

2.6 A method `void clear()` which empties the heap and resets the `currentSize` attribute

2.7 A method `Boolean isEmpty()` which returns a `Boolean` value indicating whether the heap contains elements

2.8 A method `Boolean add(FlightNode x)` which adds a new element, `x`, of type `FlightNode`, to the heap in its correct position. This method should make use of a “percolate up” strategy. The return value indicates whether the element was added successfully.

2.9 A method `FlightNode remove()` which removes the element at the top of the heap and returns the element that was removed. This method should invoke `percolateDown(int hole)` to be defined in 2.10.

2.10 A method `percolateDown(int hole)` which restores the heap property after deletion by moving the hole to the outermost level of the heap.

Question 3 [45 marks]

To test your code, you will create a driver program that reads in flight data from `planes.csv` and builds the corresponding heap.

The data in `planes.csv`:

```
registration_no,departure_timestamp,flight_duration
ZA7117,1/1/2023 14:01:36,15:33:22
ZA3642,1/1/2023 18:06:20,5:33:59
ZA5453,1/1/2023 23:43:56,4:21:13
ZA0559,1/1/2023 16:09:05,8:58:24
ZA6903,1/1/2023 13:54:48,11:59:58
ZA9604,1/1/2023 12:29:53,9:09:07
ZA5348,1/1/2023 2:59:13,6:23:48
ZA3835,1/1/2023 8:42:41,14:55:44
ZA4949,1/1/2023 0:42:13,3:05:13
ZA2951,1/1/2023 23:40:10,8:05:41
ZA4940,1/1/2023 9:11:30,12:25:47
ZA3121,1/1/2023 4:42:50,7:53:03
ZA6470,1/1/2023 22:59:30,10:03:19
ZA4981,1/1/2023 19:37:53,12:58:38
ZA0647,1/1/2023 9:19:03,14:50:22
ZA5632,1/1/2023 12:23:53,1:46:22
ZA2321,1/1/2023 20:46:09,5:07:25
ZA3408,1/1/2023 22:48:49,1:17:55
ZA2851,1/1/2023 4:28:12,13:45:53
ZA7759,1/1/2023 18:35:41,4:25:57
```

Create a `class Main` in which to write your driver code. The driver must include:

3.1 A method `Date format(int day, int month, int year, int hours, int minutes, int seconds)` which returns a `Date` object using the parameters supplied. This method can be used to obtain a `Date` object representation of a `String` date and will be needed in 3.2

3.2 Inside the executable part of the main program, write code that reads the file `planes.csv` and builds the corresponding heap:

- Open the file `planes.csv`
- Read the contents of `planes.csv`
- For each entry
    - Extract the flight registration number
    - Extract the flight departure timestamp ("dd/mm/yy hh:mm:ss")
    - Extract the flight duration ("hh:mm:ss")
    - Compute the arrival date by summing the departure time and duration
    - Create a new `FlightNode` with the registration number and arrival date
    - Add the `FlightNode` to the priority queue (use method in 2.8)

3.3 Inside the executable part of the main program, write code to display the flights using `toString()`, defined in 1.6, ordered by arrival time (earliest to latest). To do this, print the item at the top of the heap and remove the top item from the heap, until the heap is empty.

Submission and Grading

Only submit the following files:

- `FlightNode.java`
- `PriorityQueue.java`
- `Main.java`

Review the attached rubric for the mark allocation per question.