University of Westminster

School of Computer Science and Engineering

4COSC010C Programming Principles 2 – Coursework/Test - Queues – Train simulation	
Module leader	Guhanathan P
Unit	Coding Assignment with in-class test (Coursework 02)
Weighting:	50% of the module
Qualifying mark	You must get an average of 40% over assessments 1 and 2.
Description	Denuwara Express – implementation of Queue and simulation (Classes, Objects, Queues)
Learning Outcomes Covered in this Assignment:	LO1,LO2,LO3,LO4,LO5.
Handed Out:	19th March 2020
Due Date	Code due on Blackboard coursework upload Tuesday 16th April 2020 1pm.
Expected deliverables	Java program code upload, plus in-class test answers.
Method of Submission:	Blackboard + In-class test
Type of Feedback and Due Date:	Your in-class test mark (worth 50%) and java code mark (worth 50%) should appear on Blackboard Gradecentre within 3 weeks of the test. Individual written feedback will be available via the Blackboard Gradecentre. If you would like extra feedback please speak to the tutor where you did your in-class test. All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

Penalty for Late Submission: If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following

website:http://www.westminster.ac.uk/study/current-students/resources/academic-regulations
All coursework code on this module is submitted via Blackboard. It may be automatically scanned through a text matching system

(designed to check for possible plagiarism).

• You DO NOT need to attach a copy of the CA1 form;
To submit your assignment:
• Log on to Blackboard at http://learning.westminster.ac.uk; and follow the instructions below.
If you are unable to submit your work on Blackboard due to a finance hold you must email your work to com_submission@iit.ac.lk by the same deadline, putting on the subject line the module code, assessment number, and your name. This shows that you have

completed your work by the deadline. After the finance hold is lifted you must then submit the same work as normal on Blackboard, otherwise it will not be marked and you will get a fail for that part of the assessment.

Denuwara Express - Coursework 02

You are to be assessed on how well you know Classes, objects, and data structures. Concentrate on understanding the main code, and then getting additional program extras to work.

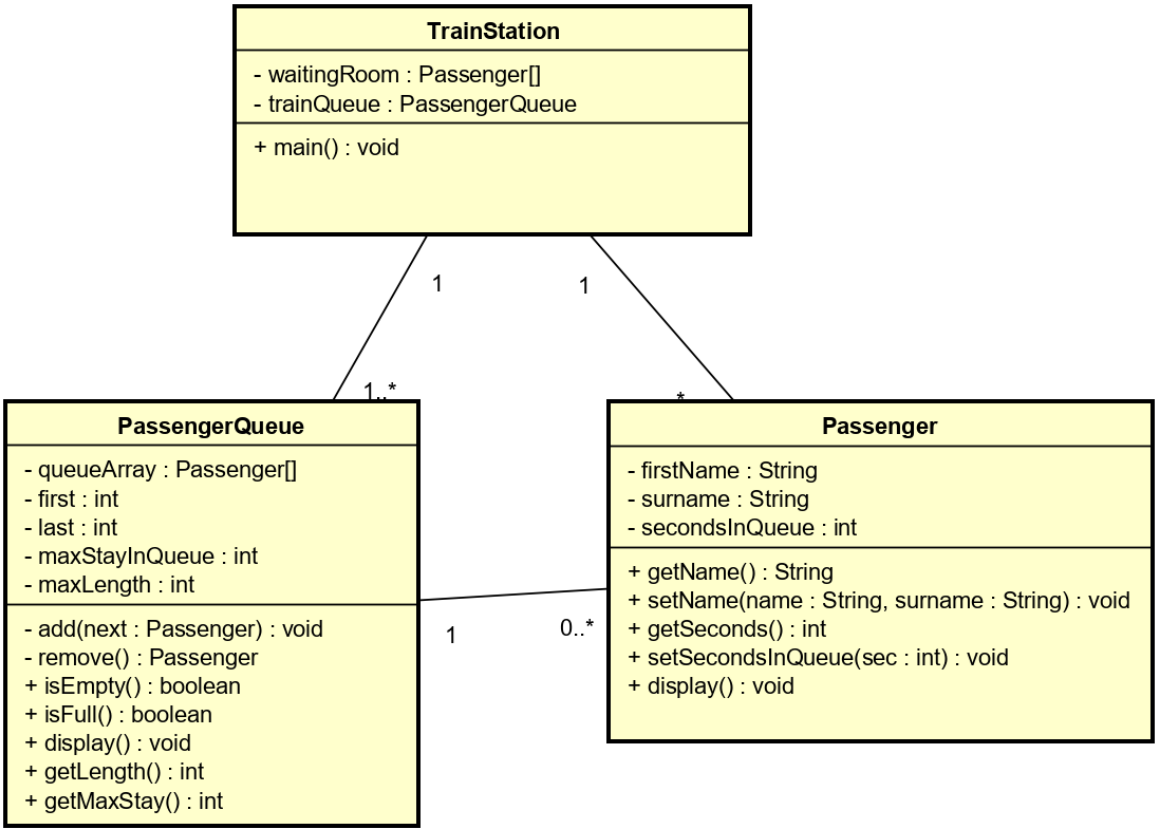
Denuwara Express Program.

1. You are to implement three classes for this coursework that will be used to simulate passengers queuing at a boarding gate and boarding the denuwara express train. The three classes and the relevant fields and methods are described in the Class diagram below. You may include additional fields and methods as required to complete the solution.

Passenger. This class will contain the properties of a passenger that are relevant to our simulation.
Fields - firstName:String, surname:String, secondsInQueue:int
Methods - getters and setters (see class diagram).

PassengerQueue. This class will represent the queue at the boarding gate. Fields queueArray: Array of Passenger, first:int, last:int, maxStayInQueue:int Methods: add(), remove(), display(), getMaxStay(), isEmpty(), isFull().

TrainStation: This is the main class that will drive the program.
Fields: waitingRoom: Array of Passenger, trainQueue: PassengerQueue Methods: main() and other methods you may implement.



In a similar way to coursework 1, you should create a menu system in the main() method (menu must be displayed in the console) of your TrainStation class which allows the user to choose which operation they want the program to perform. You should also create a Passenger class and note that

the waitingRoom and passengerQueue will hold Passenger objects. Each operation should be implemented as a separate method and the menu should allow the following operations:

'A' to add a passenger to the trainQueue – when 'A' is pressed in the console the GUI must open and the train queue must be visually shown and let passenger waiting in the waiting room to be added to trainQueue without the need of reentering the data but adding from waiting room passenger list to queue

'V' to view the trainQueue – again when 'V' is pressed train queue must visualized with 42 slots and passenger name should be displayed against seat if s/he has arrived to board the train otherwise display empty

'D': Delete passenger from the trainQueue – when 'D' is pressed through console application the passenger detail should be taken as input and passenger should be deleted – NO GUI to be used

'S': Store trainQueue data into a plain text file

'L': Load data back from the file into the trainQueue

'R' : Run the simulation and produce report (see details below).

PassengerQueue Details

Create the PassengerQueue class and implement the methods that are listed in the class diagram. The queue should be based on an array and hold 42 Passengers. If the queue becomes full then an error message should be displayed. Note: Do not use any Queue classes from the Java libraries. Implement your own queue using circular array or linked list

'R' – running the simulation details.

Your objective is to simulate passengers going from the waiting room and joining the queue to board the train at the Denuwara terminal, and gather time and queue length statistics. You have been given a passenger list with passenger names which represents the passengers currently waiting in the waiting room. The pseudocode is as follows:

Read in passenger booked for specific date in coursework 01 should be added one by one to passenger list in waiting room based on as and when they check into waiting room. User of application shouldn't be reentering the data manually but should be retrieved from where the data is stored in coursework 01 into the waitingRoom passenger list.

Repeat until the passenger trainQueue is empty

Randomly generate a number using 1 six sided die. This will represent how many passengers will join the trainQueue.

Move that number of passengers from the waitingRoom into the trainQueue.

Randomly generate a processing delay using 3 six sided dice. This will represent how many seconds it takes to check the boarding ticket and let the next passenger through to board the train.

Add the processing delay to all the passengers that are already in the trainQueue. Remove the next passenger from the queue.

Gather waiting times and other statistics for the passengers and the queue

(length, waiting time, and average time spent in the queue)
Produce a report and output to the GUI screen, and also output to a text file.

The report should display the details of the passengers from passengerArray and then print a summary of the maximum length of the queue attained, the maximum waiting time, the minimum waiting time, and the average waiting time of all the passengers.

For an additional challenge you can create two queues and have each passenger join the shortest queue. Compare the different summary statistics with the single queue implementation. (You can receive an extra 10% for this that may only be used to offset lost marks).

End repeat.

Submission Instructions:

Java Program code: zip your project folder and upload it to the Blackboard 4COSC010C 'Submit Coursework - java' link. Do not change your code after it has been submitted.

Marking Criteria

50% of the marks will come from the in-class test result. If you score 4/15 or less in the in-class test then your code solution mark will be limited to a maximum of 50% as the test will have proved you did not really understand the code.

50% of the marks will come from your 'Train' code solution.

Code marking:

40% A basic solution with menu options A, V, D implemented as methods. +20% Options S and L working.
+30% Simulation and report. Option R.
+10% Demonstration and completed self-assessment form.

Warning: All code must be done by yourself to ensure you can answer questions during the demonstration. You may be capped at 30% if you cannot explain the code.
Pass grade: Average of 40% over the two assessments.
Distinction grade (>=70%).