UNIVERSITY OF BOLTON
BSc COMPUTING
COURSEWORK SUBMISSION FORM

Student/Centre to complete:

SURNAME/FAMILY NAME: ..TOWNLEY.................. FORENAMES: ...AARON.......................

BOLTON STUDENT ID: ....2201525............. EMAIL: at4crt@bolton.ac.uk.....................................

DATE OF SUBMISSION: .....17/11/2023..

MODULE NO./TITLE:…SWE5202 Data Structures and Algorithms…………………………

TUTOR'S NAME:………Abdul Razak…………………..... ……………………

COURSEWORK TITLE: Portfolio item 2: Queues, Priority Queues and Linked lists.

Please state if this is your FIRST submission OR REFERRED/DEFERRED submission
OR a REPEAT submission? FIRST

……………………..…………………………………………………………………………………

Declaration

I hereby declare that this work is my own work. I understand that if I am suspected of plagiarism or another form of cheating, my work be referred to Academic Registrar and/or the Board of Examiners, which may result in me being expelled from the programme. I understand once I submit this work, it will automatically belong to the University of Bolton.

Academic staff to complete:

Feedback: ………………………………………………………………………………………

……………………………………………………………………………………....................

………………………………………………………………………………………………….

Date Issued: W/C 30 October 2023 Hand-In Date: (**17 November 2023 @ 16:00**)

Other Relevant Date e.g. Demonstration **In class demonstration W/C 13 November 2023.**

Received: On Time ☐ Late ☐

Mark awarded: ………..% Do not apply mark penalty unless the work was submitted late.

Assessors Name: …A. Razak………………. Signature:..

Date:…………………………

Degree Conversions A: 70-100%	B: 60-69%	C: 50-59%	D: 40-49%	F: 0-39%
HND Conversions	Pass: 40-49%	Merit: 50-66%	Distinction: 67-100%	

Late submission

For late submission, see Assessment Regulations for Undergraduate Programmes:
https://www.bolton.ac.uk/assets/Assessment-Regulations-for-Undergraduate-Programmes-2023-24-V10-v2.pdf

Creative Technologies	
Course / Programme:	**BEng (Hons) in Software Engineering**
Module name and code:	**Data Structures and Algorithms** **SWE5202**
Tutor:	**Abdul Razak**
Assessment Number:	**2**
Assessment Title:	**Queues, Priority Queues and Linked Lists**
Weighting	**25%**
Issue Date:	**W/C 30 October 2023**
Submission Deadline:	**17 November 2023 @ 16.00**

Learning Outcomes:

LO3: Design and implement Queues and Priority Queues using Linked lists.

Assessment:

Using Linked lists and Queues data structures.

HE5 – Assessment is set appropriate to level HE5.

Grading

A percentage mark will be provided as feedback. Grading is as follows:

A:	70-100%
B:	60-69%
C:	50-59%
D:	40-49%
F:	Below 40%

Marks below 40% will be classed as fail.

Assessment 2 (Queues Priority Queues and Linked lists)

Program Development Part 1 (30 marks)

Create a new Java project called **Queue part1**. In the project create the following 2 classes called FlightQueue and Flight which could be part of a flight controller type application

Class name: FlightQueue (Attributes)	
private LinkedList flights	
Class name: FlightQueue (Public methods)	
void joinQueue(Flight f)	Inserts a flight into the queue with no priority order.
void landFlight()	lands flight (i.e. removes flight from the queue)
int size()	returns number of aircraft in the queue
void clear()	lands all flights leaving the queue empty. You must use an Iterator in a loop.
void display()	displays a list of flights in the queue. You must use a for-each loop.

Class name: Flight (Attributes)	
private String flightID	e.g. BA378
private int priority	1 – 9 incl. (1 = lowest & 9 highest priority)

Both classes must have

1. a default constructor that initialises its attributes to sensible values.

The Flight class must also have

2. a parameter constructor that sets all the attribute values based on the parameter values.
3. a setter method for each attribute.
4. a getter method for each attribute.
5. a toString method to return a suitably formatted string of the attribute values.

Create a class called **FlightTest001** that simply creates a number of flights and joins them to the queue *without* maintaining priority order.

Provide suitable code to fully test your classes. You must use type-safe code with parameterised types.

Q1. What would be the advantages and disadvantages if FlightQueue were to implement the Queue interface?

Program Development Part 2 - Implementing a Priority Queue. (50 marks)

Create a new Java project called **Queue Part2**

We now require a priority queue (as well as the 'normal' queue in part 1) that will always land flights in priority order.

Create an abstract class called AbstractFlightQueue. Then create three concrete subclasses – PriorityFlightQueue1, PriorityFlightQueue2, and NormalFlightQueue.

Q2. Which method(s) should be abstract in AbstractFlightQueue and which should remain concrete? Explain your choice.

In PriorityFlightQueue1 you should simply add the new flight to the back of the queue, then sort the whole queue with the Collections sort method.

In PriorityFlightQueue2 you should add by iterating through the sorted loop to find the correct point at which to insert the flight into the linked list.

Create a class called **FlightTest002** that re-runs the previous test, but this time with all three types of FlightQueue.

Q3. Explain the algorithm that you implemented for PriorityFlightQueue2, in particular evaluate the performance and the order of the algorithm using Big O notation.

Program Development Part 3 - Implementing the java.util.PriorityQueue. (20 marks)

Create a new class in the project called **JavaPriorityFlightQueue**. Instead of using a linked list as the container use the concrete java.util.PriorityQueue.

Create a class called **FlightTest003** that re-runs the previous test but utilises the JavaPriorityFlightQueue class instead.

Q4. Does the flight queue appear sorted when you print it? If not why not, see the JavaDocs for a hint? Is it evident that queue was sorted when the flights are landed?

IMPORTANT (For all parts)

At all stages in the development, you should add JavaDoc comments to explain the purpose of the new methods added to each class. In addition to the JavaDoc comments you should add additional normal style comments where appropriate.

Specific Assessment Criteria

(Please note that the General Assessment Criteria will also apply.)
First class (70-100%):

- Programme development part 1,2, and 3 will be implemented to an excellent standard.
- Detailed and Comprehensive answers to all questions will be provided based on extensive research and testing where appropriate.
- Very well-structured code with excellent use of identifier names.
- Excellent use of comments – JavaDoc and normal comments.
- Extensive research demonstrating the use of a wide range of current secondary research sources will be evident.

- Academic style and referencing will be excellent.

Second class (50-69%):

- Programme development part 1,2, 1nd 3 will be implemented to a very good standard.
- Answers to all questions will be provided based on research and testing where appropriate.
- Well-structured code with good use of identifier names.
- Very good use of comments – JavaDoc and normal comments
- Research demonstrating the use of a wide range of current secondary research sources will be evident.
- Academic style and referencing will be good.

Third class (40-49%):

- Programme development part 1,2,and 3 will be implemented to a good standard.
- Answers to most of the questions will be provided based on research and testing where appropriate.
- Reasonably structured code with good use of identifier names.
- Good use of comments – JavaDoc and normal comments
- Research demonstrating the use of a wide range of current secondary research sources will be evident.
- Academic style and referencing will be fair.

Fail (39% and below)

Students who do not meet the requirements of a third-class grade will not successfully complete the assessment activity.

The mark you get is based on:

- the stage of development you have achieved.
- the quality of the code provided:
  - well structured
  - good use of identifier names
  - good use of comments (both JavaDoc and normal comments)
- clear concise algorithms that use appropriate Java programming constructs
- Answers to the questions

Referencing

All written work should be referenced using the standard University of Bolton referencing style– see:

https://www.bolton.ac.uk/leaponline/My-Academic-Development/My-Writing-Techniques/Referencing/Level-2/Harvard-Referencing.aspx

Minimum Secondary Research Source Requirements:

Level HE5 - It is expected that the Reference List will contain between **ten and fifteen sources**. As a MINIMUM the Reference List should include **two refereed academic journals and four academic books**

Academic misconduct

Academic misconduct may be defined as any attempt by a student to gain an unfair advantage in any assessment. This includes plagiarism, collusion, commissioning (contract cheating) amongst other offences. In order to avoid these types of academic misconduct, you should ensure that all your work is your own and that sources are attributed using the correct referencing techniques. You can also check originality through Turnitin.
Please note that penalties apply if academic misconduct is proven. See the following link for further details: https://www.bolton.ac.uk/student-policy-zone/student-policies-2023-24/academic-misconduct-regulations-and-procedures-23-24

What you must submit

You must submit the work to the Moodle link by the time and the date shown on page 1 of this assessment brief.

Your submission must include:

2) This assessment brief completed and signed on the front page, also complete the table below.
3) Your answers (and discussion if appropriate) to the questions highlighted in yellow.

4) complete UML class diagram(s)

5) The entire Eclipse project should be submitted on the Moodle submission link clearly identified with your name and student ID number (copy the entire project folder from the Eclipse workspace but **do not** change the folder name afterwards)

In this assignment I have achieved the following objectives			
Tick appropriate box NA – not attempted : Part – part completed : Full – fully completed	NA	Part	Full
Part 1			
Part 2			
Part 3			

NOTE: You need to demonstrate your programs and explain how they work during the practical session.

No demonstration means zero marks.