Coursework Assignment Brief

Assessment - Undergraduate

Academic Year 2022-23

Module Title:	Object Oriented Programming	
Module Code:	CMP5332	
Assessment Title:	Flight Bookings System (FBS)	
Assessment Type:	CWRK	Weighting: 100 %
School:	School of Computing and Digital Technology	
Module Co-ordinator:	ABDEL-RAHMAN TAWIL	
Submission deadline date:	3pm on Friday the 6th of January 2023	
Return of Feedback date and format	20 working days from date of submission (see Moodle for details).	
Re-assessment hand in deadline date:		
Support available for students required to submit a re-assessment:	Timetabled support sessions will be arranged for the period immediately preceding the hand-in date	
NOTE:	At the first assessment attempt, the full range of marks is available. At the re-assessment attempt the mark is capped and the maximum mark that can be achieved is 40%.	
Assessment Summary	This assessment comprises of two components, • Weekly Submission of Lab Exercises (Total weight 20%). • Program implementation, source code documentation and final project presentation (Total weight 80%) – This **group project** is to implement, test and document an object-oriented software product according to a predefined requirements specification. Your submission consists of program implementation, source code documentation and final project presentation. Your project must use the skeleton code	

	provided and conform to the requirements specified in the Javadoc documentation file provided. Both can be found under the "Assessment" block on Moodle. Further details for each component are given below.

IMPORTANT STATEMENTS

Undergraduate Regulations

Your studies will be governed by the BCU Academic Regulations on Assessment, Progression and Awards. Copies of regulations can be found at https://www.bcu.ac.uk/student-info/student-contract

For courses accredited by professional bodies such as the IET (Institution of Engineering and Technology) there are some derogations from the standard regulations and these are detailed in your Programme Handbook

Cheating and Plagiarism

Both cheating and plagiarism are totally unacceptable and the University maintains a strict policy against them. It is YOUR responsibility to be aware of this policy and to act accordingly. Please refer to the Academic Registry Guidance at https://icity.bcu.ac.uk/Academic-Services/Information-for-Students/Assessment/Avoiding-Allegations-of-Cheating

The basic principles are:

- Don't pass off anyone else's work as your own, including work from "essay banks". This is plagiarism and is viewed extremely seriously by the University.
- Don't submit a piece of work in whole or in part that has already been submitted for assessment elsewhere. This is called duplication and, like plagiarism, is viewed extremely seriously by the University.
- Always acknowledge all of the sources that you have used in your coursework assignment or project.
- If you are using the exact words of another person, always put them in quotation marks.
- Check that you know whether the coursework is to be produced individually or whether you can work with others.
- If you are doing group work, be sure about what you are supposed to do on your own.
- Never make up or falsify data to prove your point.
- Never allow others to copy your work.
- Never lend disks, memory sticks or copies of your coursework to any other student in the University; this may lead you being accused of collusion.

By submitting coursework, either physically or electronically, you are confirming that it is your own work (or, in the case of a group submission, that it is the result of joint work undertaken by

members of the group that you represent) and that you have read and understand the University's guidance on plagiarism and cheating.

You should be aware that coursework may be submitted to an electronic detection system in order to help ascertain if any plagiarised material is present. You may check your own work prior to submission using Turnitin at the Formative Moodle Site. If you have queries about what constitutes plagiarism, please speak to your module tutor or the Centre for Academic Success.

Electronic Submission of Work

It is your responsibility to ensure that work submitted in electronic format can be opened on a faculty computer and to check that any electronic submissions have been successfully uploaded. If it cannot be opened it will not be marked. Any required file formats will be specified in the assignment brief and failure to comply with these submission requirements will result in work not being marked. You must retain a copy of all electronic work you have submitted and re-submit if requested.

Learning Outcomes to be Assessed:

1. Demonstrate knowledge of the fundamental principles of object-oriented programming.
2. Apply object-oriented principles to design and implement programs from high level requirements specifications.
3. Use a unit testing framework in the design, testing and debugging of object-oriented programs.
4. Follow standard software development practices including pair programming and code review.
5. Use and create technical documentation for object-oriented code.

Assessment Details:

Title: Flight Booking System

Style: Implementation/ Presentation

Rationale: This programming assignment is to apply the object-oriented programming principles covered in tutorials and lectures to develop an application for a Flight Booking System. The implementation is based on the topics covered in the lectures and lab sessions for this module.

There are mainly two reasons behind the selection of a flight booking application as the topic for this coursework. Firstly, the students are familiar with the dynamics of flight booking, therefore they do not need to spend time in order to understand the functionalities of the system that they are going to implement. Rather they can directly be involved in the design and development of the command line system by applying the programming knowledge and skills they learnt. Secondly, implementing such programs make it easier to interact with the developed system and demonstrate a direct relationship between the user interaction and the system classes, methods and objects.

Introduction

This programming assignment is to apply object-oriented programming principles covered in tutorials and lectures to develop an application for an interactive flight booking system. The aim of the exercise is to enhance students' experience of programming by applying object-oriented programming principles to a larger problem through developing and testing a complete command line and GUI based application.

Your software system must be developed according to the detailed requirements specification provided on Moodle.

Objectives

Automation of flight book management has been studied over the years by many stakeholders. The main purpose of such system is to mechanise booking operations predominantly by computerisation. In this coursework you will create an interactive 'prototype' booking management application to focus on basic operations in a flight booking system like adding new flights, new passengers, searching flights and passengers, and facility to make and cancel bookings; also, to update information about bookings and passengers. The application should store all the information about bookings and customers in a persistent storage e.g. files.

Sample Prototype Application

A partial implementation of a prototype flight booking system will be made available on Moodle. The prototype is implemented as a standard Java project and contains partial implementation for the set of commands that provides the different functionalities for the booking system:

listflights	print all flights
listcustomers	print all customers
addflight	add a new flight
addcustomer	add a new customer
showflight [flight id]	show flight details
showcustomer [customer id]	show customer details
addbooking [customer id] [flight id]	add a new booking
updatebooking [customer id] [flight id]	update a booking
cancelbooking [customer id] [flight id]	cancel a booking
loadgui	loads the GUI version of app
help	prints this help message
exit	exits the program

The prototype uses text files as its backend storage to store the information of flights, customers and bookings. These files are named as "flights.txt", "customers.txt" and "bookings.txt" respectively, and are placed under the "resources/data" project folder. The different properties of each entity are stored as a single line in the text file, where each property is separated by double colon delimiter (i.e. ::). For example, the following flight information:

Flight Id: 23
Flight number: AF12456
Origin: Birmingham
Destination: Paris
Departure date: 29/10/2023

Can be stored as the line shown below in the file booking.txt:

23::AF12456::Birmingham::Paris::2023-10-29T15:30:00::

See the **marking criteria** below for more information about the functionalities required to be completed and the associated marks.

Additional information:
Your group are required to give a 15 minutes presentation covering the following:

- 10 Minutes - Describe the protocol design and implementation for your library system application
- 5 Minutes - Demonstrate your implementation

Note: The group Viva presentation is **mandatory** for all members of the group. Submitted work will not be marked if not presented.

Workload:
This is the main assessment for the module (weighing 80% of the overall mark), and is expected to take approximately 60/75 hours of work in total. This is roughly equivalent to a 4000-word report.

Transferable skills:

This assignment develops numerous transferable skills that are highly prized by employers. These include:

- Programming and OOP modelling skills
- Software application troubleshooting, testing and debugging
- Oral and written communication skills
- Time management and planning

This assessment should help students to develop various transferable skills which are necessary to obtain a decent job in the future.

Problem Solving
Being a programmer involves the ability to identify real-world issue and coming up with a technological solution to address it. This requires having strong analytical skills to understand the issue and evaluate different solutions in order to find the one that best fits the problem.

Ability to express your thoughts effectively
Using flowcharts provide an effective way of communicating the logic of a system to all people involved in the project regardless of their background and expertise. They can serve as an excellent bridge between software developers and end users.

Project evaluation skills
It is the process that critically examines a program. It involves collecting and analysing information about program activities, characteristics, and outcomes in relation to customer 10 specifications. Its purpose is to make judgements about a programme that you have developed, to improve its effectiveness and inform programming decisions.

Time management and planning
Design documentation serve as the basis for the estimating, scheduling, and validating efforts.

Marking Criteria:

A skeleton code of the application is provided having the following functionalities implemented:

To add new flights to the system. The application stores the following information for each flight including: fligh**N**umber, origin, destination, date and time of flight. In addition, the flight has a property that contains the list of passengers for a particular flight. The FlightBookingSystem object has a systemDate property which emulates the date of the flight system application. This can be used to easily change the system date and time to test functionalities of the system such as hiding flights that have already departed.

The listing of all flights stored in the system is also implemented.

Achieving a mark of 40% to maximum of 49%
The application **must** implement **all** the following:

- Add new customers (passengers) to the system. System should store at least the following information for each customer: ID, Name, Phone Number and List of Bookings made.
- List all customers stored in the system.
- Issue bookings for customers. When a booking is issued for a customer, a Booking object must be created holding a reference to the outbound and return flights booked and to the customer that made the booking. This object should be added to the customer list of flight bookings. In addition, the Customer object should be added to the list of passengers in the Flight object.
- Cancel bookings. A customer can decide to cancel a booking. The status of the booking should be updated to reflect cancelation and also the flight object should be updated to reduce the number of passengers for that particular flight.
- Display details for a particular customer including details for bookings (flight number, origin, destination, date and price) they have made [showcustomer command].
- Display details for a particular flight including details for passengers (name, phone number) [showflight command].

- Save the status of the system to the backend storage (i.e. text file storage) when the system is closed. The flight booking data should be stored in three different files (flights.txt, customers.txt and bookings.txt). A sample format to save the different properties for each object is given in the **Sample Prototype Application** section above. When the system starts it should load the status of the booking system from the text files to the memory.

Achieving a mark of 50% to maximum of 59%

The application **must** implement **all** the above and the following:

- Add a 'number of seats' (capacity) property and a price property to the Flight class and make the appropriate changes to the program to ensure that this information can be captured when a new flight is created. Also ensure that this information will be stored to and correctly loaded from the file storage.
- Add an email property to the Customer class and make the appropriate changes to the program to ensure that this information can be captured when a new customer is created. Also ensure that this information will be stored to and correctly loaded from the file storage.
- Implement Unit Tests to validate and demonstrate that the above changes made to the Flight and Customer classes work as expected.

Achieving a mark of 60% to maximum of 69%

The application **must** implement **all** the above and the following:

- Extend the prototype implementation for the GUI application provided to allow for the following basic functionalities:
    - Display a popup window that will show the list of customers (passengers) for a particular flight.
    - List all customers and their details including the number of bookings they made.
    - Display a popup window that will show the Booking details if a customer is selected from the above created list.
    - Display a popup window when the "Add" submenu of the "Customer" menu item is selected. The popup should display a form that allows the addition of a new Customer to the system.
- Extend the functionality of the flight system to allow for storing data to the file storage after the execution of commands that change the state of the system (e.g. "addFlight", "updateBooking"). If the system fails to store the data on the file storage due to an error (e.g. file is already in use or corrupted), the program must inform the user and rollback any changes made to the system prior to the error. *Hint: You can change the file permission to "read-only" in order to test this functionality.*

Achieving a mark of 70% to maximum of 79%

The application **must** implement **all** the above and the following:

- Remove (hide) existing flights from the system. When a flight is removed, it should not appear in the flight list view. Instead of completely deleting a flight, use a Boolean property in the Flight class to indicate whether a flight is deleted. Change the affected functions appropriately to return only the flights that are not deleted.
- Remove (hide) existing Customers from the system. When a customer is removed, it should not appear in the customers list view. Instead of completely deleting a customer, use a Boolean property in the Customer class to indicate that the customer is deleted. Change the affected functions appropriately to return only the customers that are not deleted.

- Impose a limit on the maximum number of passengers that can be added to a flight using the capacity property of the Flight class. The system should not allow to make a booking for a flight that is in full capacity and a message should be displayed to the user.
- Extend the implementation for the GUI application to add a delete functionality for both flights and Customers using the GUI.
- Add Javadoc documentation only to the new methods created as part of this marking band.

Achieving a mark of 80% and over

The application **must** implement **all** the above and the following:

- List only flights that are in the future and have not departed. To implement this functionality, you need to use the systemDate to indicate whether a flight has departed and a booking is completed. You have the flexibility to decide on how to complete the implementation of this functionality.
- Impose a cancellation/rebook fee when customers cancel/update their booking. The cancellation/rebook fee must be shown in the bookings in addition to the flights price.
- Calculate a different (increased) price for each flight, based on how many days left for the flight to depart on the day of the booking (current systemDate) and the capacity (number of seats left) for the flight. This price should be displayed when listing all flights before a booking is made. In addition, after a booking is made, this price should be stored and displayed when showing the booking details for a customer. This means that two bookings for the same flight, made by different customers on different dates would display different prices.
- Complete the implementation for the GUI application provided to allow for the following functionalities:
    - Implement the issue booking functionality to book a flight using the GUI.
    - Implement the update booking functionality to allow for updating a booking using the GUI.
    - Implement the cancel booking functionality to cancel a booking using the GUI.

Assessment Criteria for Component A (Lab Exercises):

Table of Assessment Criteria and Associated Grading Criteria

Assessment Criteria →	**1. Quality of code implementation (targeting LO 1, 2 and 4)**
Weighting:	**20%**
Grading <u>Criteria</u> **0 – 29%**	Significantly incomplete program. Does not compile and execute.
30 – 39%	Program compiles but gives errors during execution.
40 – 49%	Program compiles and demonstrates only few basic features of the given specification.
50 – 59%	Program demonstrates most of the basic features of the given specification.
60 – 69%	Program demonstrates all of the basic features of the specification

	and some other advanced features of the design.
70 – 79%	Program demonstrates all of the basic features of the specification and some other advanced features of the design.
80 – 100%	Program demonstrates all of the basic features of the specification and all other advanced features of the design.
Checklist	Source Code

Assessment Criteria for Component B (Program Implementation):

Assessment Criteria →	**1. Code Quality (targeting LO 1, 2 and 5)**	**2. Application Implementation (targeting all LOs)**	**3. Project Presentation (targeting LO 1 and 4)**
Weighting:	**20%**	**45%**	**15%**
Grading Criteria **0 – 29%**	Code disorganised, indentation astray, comments absent or scattered.	Significantly incomplete. Does not compile	Incomplete presentation.
30 – 39%	Little evidence of thought applied to which code should be implemented in which module or class.	Implementation is missing features to achieve a pass as described for the application.	Presentation provides only few screen shots of the implemented application. Usage scenarios are not demonstrated.
40 – 49%	Application functionality implemented but at the cost of good code structure. Some comments.	Source code implements specific features for the application as defined in the 40% to 49% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
50 – 59%	Code appropriately commented and structured based on appropriate class and object design.	Source code implements specific features for the application as defined in the 50% to 59% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
60 – 69%	Good class and object design with good judgement evident in code layout and design	Source code implements specific features for the application as defined in the 60% to 69% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band.
70 – 79%	Good class and object design, well-structured code, appropriate method and variable names with enough comments to understand the code.	Source code implements specific features for the application as defined in the 70% to 79% criteria	Presentation describes detailed usage scenarios for all the features related to this mark band. Good presentation skills are demonstrated.

80 – 100%	Excellent class and object design and a high level of attention to detail achieving high quality in code design and coding documentation.	Source code implements extra features as described in the specific criteria for the application.	Presentation describes detailed usage scenarios for all the features related to this mark band. Excellent presentation skills are demonstrated.
Checklist	Source Code	Source Code	Project presentation

Submission Details:

Format: All work should be submitted on Moodle as a single zip file. The zip file should contain your PowerPoint presentation and all of your code with associated project files. Include any additional components that are required for your application to work. The zip file should be structured so that the location of its contents is clear and unambiguous. Please submit only one version of your work.

You must submit a zip file containing:
(Group Submission)
1. The Java source code for your product implementation.
2. A PowerPoint Presentation describing the design and implementation of your ILS. Your presentation should clearly indicate individual contribution for each member of the team.

Also, the zip file should be named using the following format StudetName_studentID.zip
For example: ARTawil_123456789.zip

Note: The group Viva presentation is mandatory for all members of the group. Submitted work will not be marked if not presented.

Regulations:

- The minimum pass mark for a module is 40%
- Re-sit marks are capped at 40%

Full academic regulations are available for download using the link provided above in the IMPORTANT STATEMENTS section

For IET accredited courses ONLY (delete if required)

- For modules with multiple items of assessment, you must achieve a minimum of 30% in each item of assessment in order to pass the module.
  e.g. assessment 1 - coursework 50% and assessment 2 - Exam 50%,
  You must achieve an aggregate mark of 40% **WITH** every single assessment having a minimum mark of 30% or greater. For example if you achieved 90% in example assessment 1 and 20% in example assessment 2, the aggregate would be over 40%

((90+20)/2 = 55%), however you will still fail the module due to the 30% qualifying rule.

Late Penalties
If you submit an assessment late at the first attempt then you will be subject to one of the following penalties:

- if the submission is made **between 1 and 24 hours** after the published deadline the original mark awarded will be reduced by **5%**. For example, a mark of 60% will be reduced by 3% so that the mark that the student will receive is 57%. ;
- if the submission is made between **24 hours** and **one week (5 working days)** after the published deadline the original mark awarded will be reduced by 10%. For example, a mark of 60% will be reduced by 6% so that the mark the student will receive is 54%.

- **if the submission is made after 5 days following the deadline, your work will be deemed as a fail and returned to you unmarked.**

The reduction in the mark will not be applied in the following two cases:

- the mark is below the pass mark for the assessment. In this case the mark achieved by the student will stand
- where a deduction will reduce the mark from a pass to a fail. In this case the mark awarded will be the threshold (i.e.40%)

Please note:

- **If you submit a re-assessment late then it will be deemed as a fail and returned to you unmarked.**

Feedback:

Marks and Feedback on your work will normally be provided within 20 working days of its submission deadline via Moodle.

Where to get help:

Students can get additional support from the library for searching for information and finding academic sources. See their iCity page for more information: http://libanswers.bcu.ac.uk/

The Centre for Academic Success offers 1:1 advice and feedback on academic writing, referencing, study skills and maths/statistics/computing. See their iCity page for more information: https://icity.bcu.ac.uk/celt/centre-for-academic-success

Additional assignment advice can be found here: https://libguides.bcu.ac.uk/MA

Fit to Submit:

Are you ready to submit your assignment – review this assignment brief and consider whether you have met the criteria. Use any checklists provided to ensure that you have done everything needed.

Check Link on Moodle