CAB201 Programming Principles

Part A CRA: Major Project - Mates Rates Rent-a-Car (MRRC)

Semester 1, 2020

Due Date: 04/05/2020 11:59 PM

Weighting: 25%

Available Points: 70

Assessment type: Individual

Specification version: Version 1.1 (26/4/2020)

CHANGE LOG

Version	Date	Description
1.0	26/03/2020	Initial CRA specification
1.1	26/04/2020	Update to point weighting in Presentation (prior mistake)

CRITERIA INFORMATION

You must test your program in one of the CAB201 labs prior to submission (this can be done remotely using a virtual machine, more details closer to the submission date). Markers will attempt to compile and run your code in that environment only. It is your responsibility to ensure that your code compiles and runs in this environment. **Code that does not compile will not receive any marks.**

If you have used or drawn inspiration from others' code, you must include a reference to the original author or source in the method comments. You must give credit where it is due to protect yourself against plagiarism accusations. You are not allowed to reuse code for the entire project.

Some additional marks are available in this assignment for extended functionality. While these additional marks cannot allow your score for this assignment to go above 25% for each part, they can make up for potential weaknesses in other areas of your assignment. Additional marks are indicated with a '**+**' character and have a blue shading.

You must fill this CRA (along with the SoC, transcript and documentation) and upload it as part of your submission. Failure to do so will result in a loss of points from your final marks for each part. The marks indicated in the tables below are the highest mark you can get for a specific section. You may receive partial marks based on your delivery quality.

Your final score (as a percentage of your final grade for this unit) for each part will be calculated using the following formula:

CRITERIA TABLES

Submission Items Each of the items below are required for submission, failure to submit any of these items will result in a loss of points. If any of the items are submitted but left incomplete you will lose a portion of the amount listed on the right of the table. The amount lost will be proportional to the amount of incomplete work.	**Points (Total: 10)**	
Project Documentation	2	**2**
Statement of Completeness	2	**2**
Program Transcript	2	**2**
Self-filled CRA	2	**2**
Class Diagram	2	**2**
Total	**9**	**10**

Functionality – File I/O To gain points for this section, it must be clear that your program can read and write to and from the provided samples files. To demonstrate the write functionality your program must be able to add/remove/modify the vehicles/customers data and save it to the appropriate file. To demonstrate the read functionality your program must be able to correctly interpret data from the files before and after writing.		**Points (Total: 12)**	
Read	Customer data can be read from a file (`customers.csv`)	3	**3**
	Vehicle data can be read from a file (`vehicles.csv`)	3	**3**
Write	Customer data can be written to a file (`customers.csv`)	3	**3**
	Vehicle data can be written to a file (`vehicles.csv`)	3	**3**
Total		**12**	**12**

Functionality – User Interface To gain points for this section, the following features must be easily accessible and testable via the console user interface. All of the following functionalities must be demonstrated in the program transcript.		**Points (Total: 28)**	
CRM	View list of customers	0	**2**
	Add customer Validation – Unique customer ID Validation – Valid input fields	3 1 1	**3** **1** **1**
	Modify customer Validation – Unique customer ID	3 1	**3** **1**

	Validation – Valid input fields	1	**1**
	Remove customer	1	**2**
Fleet	View list of vehicles	0	**2**
	Add vehicle Validation – Unique registration Validation – Valid input fields	3 1 1	**3** **1** **1**
	Modify vehicle Validation – Unique registration Validation – Valid input fields	3 1 1	**3** **1** **1**
	Remove vehicle	2	**2**
Total		**24**	**28**

Code Quality To gain points for this section, you must maintain good code quality throughout your whole project. While following the CAB201 C# Coding Style Guide will help you meet these criteria, you do not have to follow it to the letter (at the least you must be consistent and clear). **Important:** Your target reader is a programmer, not an absolute beginner.	**Points (Total: 10)**	
Maintained, consistent and clear standard in variable, method and class naming.	2	**2**
Magic number have been replaced with appropriately named constants.	1	**1**
Consistent and appropriate white spacing, line length, indentation, and separation into files within the project (i.e. one class per file)	2	**2**
Class header comment at beginning of each class, comment before every method, and in-line comments to explain complex or not easily discernible code. In-line comments are not excessive.	3	**3**
Methods are single purpose and clear, and code is reasonably efficient and succinct.	2	**2**
Total	**10**	**10**

Presentation To gain points for this section, you must ensure your user interface is of a high quality. Your interface does not need to be flashy or overly visually appealing – instead you should focus on making it clear, intuitive and easy to use.	**Points (Total: 10)**	
Incorrect input is handled appropriately and clearly.	2	**2**
The overall design is clean and uncluttered. Data is displayed clearly.	2	**2**

All necessary functionality of the UI is clear, easy, and quick to access.	4	**4**
Special cases are handled nicely (e.g. when the user requests a list of vehicles and there are no vehicles, a message is displayed rather than just an empty list of vehicles).	1	**2**
Total	**9**	**10**

Bonus Criteria (+) To gain points for this section, you must implement advanced OOP concepts in your code. You have freedom in where/how these concepts are implemented in your code. Ask the teaching team if you require some advice or guidance.	**Points (Total: 10)**	
Project code exhibits at least one good use of encapsulation (separate from the classes detailed in the class diagram). *Recommended Approach:* There are many opportunities to implement some level of encapsulation in your code, some good examples are: - A *Table* class which can store and print formatted data for a table. - A *Menu* class which can control program flow.	5	**5**
Project code exhibits at least one good use of inheritance. *Recommended Approach:* The *Vehicle* class can be inherited by four distinct subclasses, each categorised by the vehicle grades (Economy, Family, Luxury, Commercial). Appropriate constructors can be set which make use of the default values of the different classes.	5	**5**
Total	**10**	**10**