CAB201 Programming Principles

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

Semester 1, 2020

Progress Submission Due Date: Friday 29th May 2020 11:59PM

Final Submission Due Date: Friday 5th June 2020 11:59 PM

Weighting: 45%

Available Points: 100

Assessment type: Individual

CRA Specification version: Version 1.0 (28/4/2020)

CHANGE LOG

Version	Date	Description
1.0	28/04/2020	Initial CRA specification

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, guide on Blackboard). 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 are not allowed to reuse code for the entire project, but you must give credit where it is due to protect yourself against plagiarism accusations.

No additional marks are available in this assignment for extended functionality (contrary to Part A).

You must fill this CRA and upload it as part of your submission. Failure to do so will result in a deduction from your final grade 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:

$$\text{Final Score} = \frac{\text{Points Scored}}{\text{Available Points}} \times \text{Weighting}$$

CRITERIA TABLES

Submission Items Each of the items below are required for submission, failure to submit any of these items will result in a point deduction. If any of the items are submitted but left incomplete you will be deducted a portion of the amount listed on the right of the table. The amount deducted will be proportional to the amount of incomplete work.	**Points (Total: 10)**	
Project Documentation (User Manual)	✔	**2**

Statement of Completeness	✔	**2**
Program Transcript	✔	**2**
Self-filled CRA	✔	**2**
Class Diagram	✔	**2**
Total		**10**

Functionality – File I/O To gain points for this section, it must be clear that your program can read from and write to the provided samples files. To demonstrate the write functionality your program must be able to rent vehicles to customers and save the rental data 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: 10)**	
Read	Rental data can be read from a file (`rentals.csv`)	✔	**2**
Write	Rental data can be written to a file (`rentals.csv`)	✔	**3**
Implementation	Files names and path are entered by users via the command line or interaction with the program (rather than hard-coded within source code).	✔	**2**
Exception Handling	Exceptions are used to catch File handling errors and to provide meaningful error messages e.g. file does not exist, file cannot be open etc.	✔	**3**
Total			**10**

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: 15)**	
CRM	Remove customer Validation – Customer is not renting vehicle	✔	**1**
Fleet	Remove vehicle Validation – Vehicle is not being rented vehicle	✔	**1**
Rentals	View list of currently rented vehicles showing the vehicle, customer and daily cost of the rental.	✔	**3**

	Rent a vehicle to a customer (prompt ID, registration and rental time and display total cost)	✔	**3**
	Validation – Customer is not renting vehicle Validation – Vehicle is not rented	✔ ✔	**1** **1**
	Return vehicle from customer (registration) Validation – Vehicle is being rented by customer	✔ ✔	**3** **1**
	Search vehicles for rental (menu only)	✔	**1**
Total			**15**

Functionality – Search To gain points for this section, the following types of queries must be functional in the program's rental search feature. For inaccurate results, you will not get all marks available. *All searches should only return vehicles not currently rented out.*		**Points (Total: 20)**	
Simple	Blank query (shows all vehicles)	✔	**2**
	Single attribute query (shows all vehicles of a given attribute) Examples: - `Red` - `Family` - `8-Seater`	✔	**2**
	Disjunction query (shows all vehicles of one of two given attributes) Examples: - `Family OR Luxury` - `Red OR Blue` - `GPS OR Family` *Attributes here do not need to be the same attribute, so you should be able to search `GPS OR Family` [GPS attribute and grade attribute]*	✔	**2**
	Conjunction query (shows all vehicles of both given attributes) Examples: - `Family AND Luxury` - `Red AND Blue` - `GPS AND Family` *See above. Impossible searches are allowed (`Luxury AND Economy`), but should still return correct results (i.e. no vehicles)*	✔	**3**

Intermediate	Multi-disjunction query (shows all vehicles of one of many given attributes) Examples: - `Family OR Luxury OR Red` - `Red OR Blue OR Green` - `GPS OR Family OR Luxury OR 4-Seater`	✔	**3**
	Multi-conjunction query (shows all vehicles of all given attributes) Examples: - `Luxury AND Red` - `4-Seater AND Family AND GPS`	✔	**3**
Advanced	Combination query - Query uses both `AND` and `OR`, where the operators `AND` and `OR` should have the same priority (precedence) and are evaluated from left to right. Examples: - `Economy OR Family AND 4-Cylinders`	✔	**5**
	- `Red OR Blue OR Green OR Purple AND GPS`	✔	
Total			**20**

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: 20)**	
Naming: Maintained, consistent and clear standard in variable, method and class naming.	✔	**2**
Magic numbers: Magic numbers have been replaced with appropriately named constants.	✔	**1**
Format: Consistent and appropriate white spacing, line length, indentation, and separation into files within the project (i.e. one class per file and use of references to separately compiled components)	✔	**3**
Comments: 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.	✔	**2**
Methods: Methods are single purpose and clear, and code is reasonably efficient and succinct.	✔	**2**

Polymorphism: a method that acts differently in different situations properly implemented (do not use different method names where polymorphism should be followed)	✔	**1**
Classes: Each class has a coherent, simple, easily stated purpose	✔	**2**
Design of classes: Low coupling, high cohesion, proper inheritance. Minimal public and static methods or properties.	✔	**2**
Code reuse: no unnecessarily long or repeated code. No redundant methods.	✔	**1**
Exception handling: Program does not crash. Meaningful error messages provided.	✔	**4**
Total		**20**

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)**	
All user inputs are NOT case-sensitive	✔	**2**
Incorrect input is handled appropriately and clearly.	✔	**2**
The overall design is clean and uncluttered. Data is displayed clearly.	✔	**2**
All necessary functionality of the UI is clear, easy, and quick to access.	✔	**2**
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).	✔	**2**
Total		**10**

Advanced Query Processing - Priority combination query To gain points for this section, you must implement an advanced expression evaluation algorithm in your search code. Priority combination query (shows all vehicles based on a combination of disjunctions and conjunctions, with operator priority). Query uses both `AND` and `OR`, with `AND` having a higher priority than `OR`, supporting parenthesis to resolve priority. **Note:** In the following criteria, example queries are offered. **These are examples only.** The program will be tested with many different queries of these types, not necessarily these exact queries. The solution must be general, not specific to certain queries. Ask the teaching team if you require further advice or guidance.		**Points (Total: 15)**	
Priority Combination Query	Query without parenthesis (AND has higher priority than OR) `Economy OR Family AND 4-Cylinders`	✔	**2**
	Query with simple parenthesis `(GPS OR Sunroof) AND (Red OR Green)`	✔	**3**
	Query with nested parenthesis `((GPS AND Sunroof) OR (Red OR Green)) AND Commercial OR Luxury`	✔	**3**
Implementation	Use proper abstract collection such as `Stack`	✔	**2**
Exception Handling	Provides meaningful error messages when exceptions occur (e.g. divided by error, invalid operator, mis-matched parenthesis, non-existent attributes, malformed expressions etc.) and returns to user input.	✔	**4**
	Graceful degradation: the program does not crash when exception occurs.	✔	**1**
Total			**15**