CST2550 Coursework 1

Dr Barry D. Nichols

November 23, 2022

1 Brief Task Description

A vehicle hire company needs a system to track the details of their available vehicles. They have three types of vehicles available: motorcycle; car and van, each with different information.

A file containing details of vehicles will be made available to you for implementation and testing, but the system should allow any file to be used, a different file will be used for marking (**do NOT hardcode the filename**).

You should use UML diagrams to design the software and the Git version control system with **regular** commits which are pushed to a repository on BitBucket.

You should **NOT** use any third party libraries or code as a part of your solution and all code must be written by you, e.g. not automatically generated by an IDE. Do not use any non-standard code or libraries or **system** calls in your program.

2 Submission

The submission will consist of a single **zip** file of all required source code; a single **MP4** video file of a screencast presentation and a single **PDF** file of the presentation slides. The deadline is **5pm on Friday, 13th January 2023**.

The source code zip file should include:

- C++ source code files of your program
- C++ source code of *catch2* tests which test your code
- a *makefile* to compile your program

In the presentation you should show your slides while you describe each section. Full details of what should be included in the presentation are given below. The video should be a maximum of 6 minutes long. If the video is longer, only the first 6 minutes will be marked. The video must also include a short demonstration of your software where you demonstrate understanding of the **implementation** *(note: just running the program will not demonstrate any knowledge of the implementation).*

Note:

- **As anonymous marking will be applied, you should *not* include your name in your source code or presentation**
- **To allow the presentation and code to be marked together whilst maintaining anonymous marking, your presentation slides; code zip file and screencast must be named using your student ID, e.g. M00123456.pdf; M00123456.zip and M00123456.mp4**
- **Only source code (header and cpp files) and a Makefile should be included in the zip file, no other files (any other files will not be used when marking your work)**

- **If code does not compile and run it will severely limit your marks for the code**
- **Do not use any non-standard or OS specific libraries or system calls, which will prevent your program from working correctly when being marked**

3 Scenario

A vehicle hire company has three types of vehicles:

- motorcycle
- car
- van

All vehicles have a make, model, registration number, colour, price (per day), whether it is currently available and return date if it is not available. Motorcycles also have engine size (in cubic centimetres), whether it has a passenger seat and if it has luggage space. Cars have engine size (in litres), number of seats, number of doors. Vans have engine size (in litres), number of seats, luggage space.

The program will only be used by staff at the vehicle hire company (not by customers), and will not take payment details.

The system should include the following functionality:

- load vehicles from file
- save vehicles to file
- add vehicle
- remove vehicle
- list all vehicles (only vehicle make, model, registration number, availability)
- display all details of a vehicle

Your program must have input validation to ensure it performs correctly and helpful error messages informing the user of any correct input. It should be intuitive, i.e. the user shouldn't need to read the source code or attend training to know how to use the system.

4 Detailed Description

It is recommended that you complete the tasks in the following order as the later sub-tasks will require the earlier ones.

4.1 Plan Software

Begin by planning which classes you need and how the software will work using UML diagrams, you should include:

- use case diagram(s)
- class diagram(s)
- activity diagram(s)

Make sure you include classes for each vehicle type and a *Vehicle* base class.

4.2 Create a Git Repository

Create a Git repository for your coursework on BitBucket. As you create new files, you should add them to the Git repository. Commit regularly with clear, useful commit messages.

4.3 Implement Classes

Once you are happy with your design, you should implement the classes, you should have a base class for *Vehicle* with any shared member variables and member functions, and derived classes for each specific type of vehicle with specific member variables and functions.

You should write test cases and Catch2 test code to ensure these classes are working correctly.

4.4 Write Makefile

Create a Makefile to compile the classes. As you implement other source code files add them to the Makefile. Your Makefile should be able to compile you project using the command **make** and should be able to clean your project (remove any compiled files) using **make clean**.

4.5 Implement the Program

Implement the system using objects of the various classes mentioned above.

Write additional tests for any functions to ensure they work correctly and implement a menu for the user to interact with the program. The user input should be validated.

4.6 Plan Presentation

Plan a presentation and create presentation slides detailing the design, implementation, testing and evaluation of your work. This should be about your work, not generic comments about programming in general.

The presentation should include:

- introduction, including:
  - your student number
  - a brief description of your project (**do not read the coursework task**)
  - brief overview of the presentation
- design:
  - clearly show each of your UML diagrams
  - you should briefly describe what the diagrams are showing (**do not just read the diagrams**)
- implementation, including:
  - your approach, i.e. how you translated the design into working software
  - explain what the Makefile was used for
  - explain how and why version control was used
  - include a screenshot of the BitBucket repository which clearly shows all commits and commit messages
- testing approach, including:
  - a statement of the approach used
  - how you applied this approach

  - details of test cases, i.e. what was being tested (**don't show the code**)
- software demonstration
  - Show your understanding of the implementation, not just that you can run the program.
- conclusion, including:
  - a brief summary of the work
  - limitations of your work (**don't list features which aren't required**)
  - how you would approach a similar project in the future to avoid the limitations (**don't list features which aren't required**)

5 Academic Misconduct

This is individual work and you should complete it yourself. You should not work as a group and each submit the same work (even with minor changes) as your own. Any material or ideas found online, in textbooks, etc should be properly referenced.

You should familiarise yourself with the university's academic integrity and misconduct policy: https://www.mdx.ac.uk/about-us/policies/university-regulations

6 Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet an assessment deadline or affect your performance in an assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse and outside of your control. Please see link for further information and guidelines:

https://unihub.mdx.ac.uk/your-study/assessment-and-regulations/extenuating-circumstances

7 Marking

The presentation video and code will be marked according the to attached marking scheme. Marking will be **anonymous**, i.e. the marker will not see the name of the student while marking.

As the presentation also includes your demonstrated understanding of your program, **if there is no presentation video you will not be awarded any marks for this coursework task.**

8 Feedback

Provisional marks and written feedback will be available on Moodle within 15 working days of your submission. If you would like clarification or more detailed feedback on your coursework contact your module tutor.

9 Marking Scheme

9.1 Presentation

Item	**Marks**
Correctly submitted files (correct naming, format, contents, etc.)	5
Introduction (description of the project, not the coursework task)	6
Software design (description and UML diagrams)	9
Software testing (description of testing approach used and evidence of testing)	6
Implementation (description of approach and how the Makefile and version control were used).	9
Conclusion, including: - summary of work done - limitations and critical reflection - how would change approach on similar task in future	5
Layout and clarity of presentation and slides	5
Timing (6 minutes long)	5

9.2 Code

Item	**Marks**
Code quality (follows code guidelines and user input validation)	10
Use of Git (regular commits, clear commit messages)	5
Makefile	5
Implementation matches design	10
Implementation meets requirements	10
Quality of test code (and matches testing approach described in presentation)	10