  
  
 MODULAR PROGRAMME

# COURSEWORK ASSESSMENT SPECIFICATION

## Module Details

|  |  |  |
| --- | --- | --- |
| **Module Code** UFCFB6-30-2 | **Run** 03/Oct /2022 | **Module Title** OBJECT-ORIENTED SYSTEMS DEVELOPMENT |
| **Module Leader** Udhuma | **Module Coordinator** | **Module Tutors** Ibrahim Shahid |
| **Component and Element Number** B: CW1 | | **Weighting: (% of the Module's assessment)** 50% |
| **Element Description** DEMONSTRATION AND PRESENTATION (Demonstration and Presentation of Design Work) | | **Total Assignment time** |

## Dates

|  |  |
| --- | --- |
| **Date Issued to Students**  10/07/2019 | **Date to be Returned to Students** See inside the assignment |
| **Submission Place**  **Blackboard** | **Submission Date** 04/04/2023 (Also see inside the assignment for the different stages) |
| Submission Time 18:00 pm |

## Deliverables

|  |
| --- |
| See inside the assignment. |

## Module Leader Signature

|  |
| --- |
| Udhuma |

UFCFB6-30-2 Object-oriented Software Development Assignment (2022/2023)

Contents

[1 Your tasks and deliverables 3](#_Toc456090690)

[2 Submission and presentation schedules 3](#_Toc456090691)

[3 Marking criteria 4](#_Toc456090692)

[4 The case study scenario 5](#_Toc456090693)

[4.1 Overview 5](#_Toc456090694)

[4.2 User interface design 6](#_Toc456090695)

[4.3 Detailed requirements on functionalities and the user interface 7](#_Toc456090696)

[4.4 Data representations 8](#_Toc456090697)

[4.4.1 Text files for populating test data 8](#_Toc456090698)

[4.4.2 Unique IDs 8](#_Toc456090699)

[4.5 Information for the implementation 9](#_Toc456090700)

[4.5.1 GUI front end 9](#_Toc456090701)

[4.5.2 Using Twitter APIs 9](#_Toc456090702)

[5 Test plan for the final system demonstration 10](#_Toc456090703)

[Appendix A How does this assignment relate to the module specification 11](#_Toc456090704)

# Your tasks and deliverables

The assignment is an individual assignment. You are asked to produce the following deliverables for the scenario given below.

1. Produce a UML use case diagram to capture the functionality for the system to be built
2. Produce a UML class diagram to meet all the requirements captured in the use case diagram.
3. Produce UML sequence diagrams for some use cases. The specific use cases will be given to you in due course.
4. Implement the system in Java and write test cases.

***All UML diagrams must be done in Astah. All Java code must be done in Netbeans.***

You are required to submit and to present each of the above. Submission will be done on Blackboard. Presentations will take place in your lab sessions. After each stage, summative and formative feedback will be provided. A possible solution will be given for students to use for the subsequent tasks.

# Submission and presentation schedules

There will be four submissions and presentations:

* *30th November 2022: Submit your use case diagram on Moodle*
* *21st December 2022: Submit your class diagram on Moodle*
* *14th February 2023: Submit your sequence diagram on Moodle*
* *4th April 2023: Submit the final implementation and documentation on Moodle*

There will be individual presentations of your work in your scheduled lab sessions following each submission.

* ***Work submitted or presented after the scheduled deadlines, but before the final deadline (04/04/2023), gets capped at 30% of the available marks. Work submitted after the final deadline is subject to the standard university rules.***
* ***Students can arrange with their tutors to do the presentation earlier than the scheduled time if they are unable to do it in the scheduled time.***
* ***For students with extenuating circumstances, please contact student advisors and the tutors to make alternative arrangements.***

# Marking criteria

|  |  |  |
| --- | --- | --- |
| **­Elements:** | **Key Point:** | **Criterion:** |
| **Use case diagram (10%)** | **Correct use of notation and sensible naming** | *Notations for actors, use cases and relationships are correct*  *Follows the naming convention* |
| **Correct identification of actor(s) and use cases.** | *All actors are identified*  *All functional requirements are captured by some use cases.*  *All use cases map back to the problem domain*  *Use cases are at suitable level of abstraction.* |
| **Correct use of the relationships** | *Correct use of communication link, include, extend, generalization. Note that it is not necessarily the case that you will need to use all of these relationships.* |
| **Class diagram (20%)** | **Correct use of notations** |  |
| **Appropriate identification of classes** | *Maps back to the problem domain and the use cases* |
| **Appropriate identification of attributions and operations** | *Demonstration of encapsulations and separation of concerns*  *Promotion of high cohesion and loose coupling* |
| **Correct use of class relationships** | *Correct use of these class relationships: Inheritance, Composition, Aggregation, Association, Dependency*  *Note that it is not necessarily the case that you will need to use all of these relationships.* |
| **Sequence diagram (20%)** | **Correct use of notations** |  |
| **Consistent with the class diagram** | *Matches the class names and the method names* |
| **Appropriate logical sequence** | *Matches the description in the problem domain and the use case behaviours* |
| **Implementation (15%)** | **Consistent with the class diagram design** | *Matching to the class diagram: Class names, attributes, operations and class relationships* |
| **Consistent with the use cases and the sequence diagrams** | *Message passing follows the sequence diagrams and able to satisfy the use cases.* |
| **Test cases (35%)** | **Should be generated from use cases.** | |

# The case study scenario

## Overview

A vehicle service centre (All Service Centre) has to manage the vehicles that are brought in by customers for servicing. Information about the vehicle that must be recorded includes the model, registration number, chassis number and registration date. Information about the owner must be recorded, whether they are private owners or companies. The service centre must generate information about inspections on a commercial vehicle, including the inspection date, status and remarks.

Information about any servicing done on a vehicle is recorded in a Job Sheet. When the customer brings the vehicle in, the service advisor will generate a job sheet for the service by recording any notes for the service. The service advisor will also record the required job items to be done. The service advisor will print a copy of the job sheet for the customer.

The car is then moved to the service area where it will be attended to by the mechanics. The mechanics will retrieve the job sheet for the car, and view the required job items. A job item will be composed of related component items such as labour or parts replacement.

If a spare part is used, the quantity used and the part number is recorded. If the labour should be charged, the description of the labour item is recorded. The total price of each labour item depends on the Mechanics’ charge rate. The mechanic who worked on the job will be recorded together with his or her charge rate and the number of hours they worked. Some jobs may be so simple that the labour items are not charged, or recorded.

On the other hand, some jobs may be complex and have multiple labour items recorded, or have labour only and no parts consumed. When the service is completed, the service advisor prints out the bill for the service. The job items would be updated with the labour done and parts used. A discount may be given on some parts.

Finally the prototype includes a system administrator who will load vehicle data or demo data to the system. The system administrator also will be creating and managing users and in addition to this, will conduct a market review via twitter API for recently posted tweets about the vehicles and new service methods.

You may design and build a system as simple or as complicated as you wish. However a more complete system will attract better marks. You must also design and write a main class – which will test your system (e.g. motorcycle management system, etc.).

## User interface design

The students are required to show their creativity in designing user friendly screens for the forms to use in system.

Four views must be there: System admin view, Customer view, Service Advisor view and Mechanic view.

## Data representations and initial data population

### Text files for populating test data

The “Load specified test data” function for the system admin should read information from the file “services.txt” to load the list of services and prices. Here is an example of the kind of content in this file.

*“Changing brakes,150.00*

*Washing 200.00,*

*Replacing head lights, 300.00l”*

After loading the services, it should allow to create new service order for each of the category created above. Each service order will need to have a date. You should use the information (dates) provided in the file “testServices.txt”. Here is an example of the kind of content in the file:

*“1/9/2020 cleaning Motorcyle,*

*6/9/2020 Changing brakes Car,*

*11/9/2018 cleaning Car,*

*…”*

Both of these files will be submitted by the students.

### Unique IDs

Each service order should have a unique ID.

Each user will also have a staff ID.

## Information for the implementation

### GUI front end

Students are required to create GUI’s according to the user requirements. The more creative and interactive the screens are then, the better the marks will be awarded.

### Using Twitter APIs

In order to be able to use some of the Twitter APIs, you need to do the following:

1. download and unzip the “twitter4j-4.0.4” folder. It will be provided on Blackboard.
2. add related jar files to your assignment project in Netbeans
   1. by right clicking on the Library node for your assignment project and then select “add jar/folder” option
   2. Find the above “twitter4-4.0.4” folder, go to the “lib” sub-folder and select all the jar files and add to the Library node for your project.

# Test plan for the final system demonstration

(To be created according to the use case diagram. JUNIT must be used)

# Appendix A: How does this assignment relate to the module specification

This assignment assesses the following module learning outcomes as specified in the module specification: [Extract from the module specification]

3. Apply object-oriented analysis and design techniques for a number of problem domains scoped at level 2 complexity (A,B)

4. Understand and use a Unified Modelling Language (UML) modelling tool and a Java-based Interactive Development Environment (IDE) to develop object-oriented software implementations appropriate to level 2 complexity (B)

6. Use an appropriate development environment to design and implement persistence within a distributed architecture at an introductory level. (A, B)

7. Design and implement concurrent systems and distributed systems (A, B)

8. Design and implement Graphical User Interfaces (GUIs) (B)

The assignment is based on the following case study. You are required to design and develop a computer system to support the scenario described in the case study. This involves producing some UML artefacts and a Java implementation.

Working on this assignment will help you to

* Learn how to model complex problems using UML.
* Gain experience in evaluating object-oriented designs to get an insight into what is a good design
* Improve your Java skills
* Boost your confidence in implementing fairly complex systems
* To Develop your research skills to explore new programming techniques and skills from the vast amount of resources available outside what is being taught in the module