|  |  |
| --- | --- |
| **Course** | **TNABT Software Engineering** |
| **Unit Code** | **ICTPRG430** |
| **Unit Title** | **Apply Introductory Object Orientated Language Skills** |
|  | |
| **Assessment Task Title** | **Assessment 2 – Skill Tasks** |
| **Assessment Type** | **Practical** |

## Overview

The assessment provides the opportunity for you to demonstrate the following skills and knowledge:

* Develop design specifications
* Build an object orientated application to specifications
* Apply programming language syntax, sequence, selection, iteration and, OOP constructs
* Test and debug an application
* Apply documentation conventions

## Instructions:

This assessment is to be completed in your own time. Time may also be made available for completing the assessment during class sessions.

You are required to enter your responses in the spaces provided in this assessment document, completed assigned activities and, follow instructions to upload resources

To achieve a ‘satisfactory’ result for this assessment you must complete all tasks and be deemed competent in all tasks by your assessor. In the event that you receive an unsatisfactory result, you will be required to review feedback from your assessor and then resubmit the assessment after making required corrections.

You will have one opportunity for resubmission. If your second assessment attempt is ‘not yet satisfactory’ you must contact your teacher or assessor to discuss how to proceed.

All responses must be your own work.

## To be submitted:

* This completed assessment document titled - Assessment 2 – Skills
* Panopto video of IDE debugger use
* Zipped up LOCAL GitHub repository folder (.zip)

Upload all documents by the due date to the drop box for ICTPRG430 Assessment 2 on VU Collaborate.

## Assessment scenario – Trucking company

Your client, ‘Trucking Australia’, has requested a program that will enable them to enter data for their trucking company and record the details to an electronic file. Their inventory consists of trucks and cars as well as employed drivers from every state and territory of Australia. Such a tracking system will enable ‘Trucking Australia’ to maintain accurate records that can be used to improve logistics and better structure business operations.

The program will record data such as vehicle makes and models, driver's personal details, kilometers driven and other data.

Although ultimately the company will want to be able to enter data via a menu sytem, that is not your task. Your task is to create the underlying structure of the program and hard-code the entry of data in preparation for use within a menu system.

*\*Creation of a menu system is not required at anytime during this assessment.*

## **Student Name: Sama Priyangani jayarathne siriwardhanage Student ID Number:** **S8057089**

## Task 1.1 Clarify task with user

Discuss the user requirements given in task 1.3 with your instructor who is acting as your client. This is your opportunity to ask any questions to ensure you fully understand what the client needs the application to do. Provide brief notes or dot points of the discussion that show your attempts to clarify and understand the users’ requirements. *Note that your instructor may work with you individually, in small groups or as one large group for this task*

|  |
| --- |
| * *How would be the user interface looks like?* * *What format do you prefer for displaying cars and trucks general data?* * *How do you want to store data?* * *What type of proof do you need for written drivers’ details?* * *Do you need to implement any security mechanism to the file?* |

|  |
| --- |
| Before proceeding you should review with your instructor the supervisor consultation notes, coding checklist, hard-coding examples and, sequence and display examples, in the appendices to ensure you fully understand the assessment requirements |

## Task 1.2 Plan and determine the applications design specifications to meet the clients requirements

Referring to the user requirements in task 1.3 as well as supervisor consultation in Appendix A, complete the following design specification

|  |
| --- |
| **Design specification** |
| Create a UML class diagram of the proposed application.  *Enter your UML class diagram here* |
| Produce a design of the logic of your intended program using pseudo-code or a flowchart with explanatory comments. *This is an adjunct to your UML class diagram and will include details of any sequence, selection and iteration that might occur within a class and/or within the main body of code (main())*  *Enter your pseudo code here*  Function displayDriverDetails()  display driver1 details- first name, last name, license no, mobile  use for loop to display address line by line  use for loop to display states in which driver is licensed to drive  Function displalyDemeritPoints()  Display value of demeritPoints  Function addDemeritPoints()  Calculate demerit points  Check if new demerit points > max demerit points  Print “Invalid entry”  else  Update demeritPoints new value  Function deleteDemeritPoints()  Check if entered demerit points value <= demerit points  Decrement demerit points  If decremented demerit points < = 3  Print “License Suspension is imminent”  End if  Else  Print “Invalid entry”  Function writeDriverFile()  Write driver details to the file called driverDetails.txt  Function readDriveFile()  Read data from the file called driverDetails.txt  function displayVehicleData()  display values of specified vehicle  function displayGeneralData()  display vehicle1 data -registration no, make, model, driven km  function updateKM()  write new value  Function updateColour()  write new value  main()  Hardcoding  Populate driver 1 & 2 states list  Populate driver 1 & 2 address  Populate drive 1 & 2 other details  Populate car 1 & 2 details  Populate truck 1 & 2 details  Display message “Increase demerit points - Driver 1 “  Call Function displayDemeritPoints() for driver 1  Call Function addDemeritPoints() for driver 1  Call Function displayDemeritPoints() for driver 1  Display message “Removing demerit points from Driver 1  Call function displayDemeritsPoints() for driver1  Call function deleteDemeritPoints() for driver1  Display message “Driver 1 and Driver 2 details”  Call function displayDriverDetails for driver1  Call function displayDriverDetails for driver2  Display message “update vehicle colour & KMs driven  Call function displayVehicleData for car 1  Call function updateColour() for car1  Call function updateKM() for car1  Call function displayVehicleData for car1  Display message “Writing Driver Details to the driverDetails.txt file”  Call function writeDriverFile() for driver1  Display message “Reading driverDetails.txt file content”  Call function readDriverFile() |

## Task 1.3 Develop the application according to the design and organisational code conventions

Translate your pseudo code and UML class diagram into a Python3 script that adheres to the code layout, white space and comments recommendations of the PEP 8 Style guide for Python code. Include Docstrings compliant with the conventions of PEP 257 to all functions, methods and classes. Also include at the top of the program the authors name, the date created, and an overview of the codes functionality. Include other comments on the code only where explanation is needed.

<https://www.python.org/dev/peps/pep-0257/>  
<https://www.python.org/dev/peps/pep-0008/>  
  
Adhere to the user requirements checklist below and the Coding checklist in appendix B to ensure that the application contains all required elements.

|  |  |
| --- | --- |
| **User Requirements checklist** Your code will be written to the meet the following user requirements | |
|  | Done |
| The general data to be entered for all vehicles will be registration number, make, model and kilometres driven. | **done** |
| Additional specific data to be entered for cars only will be body type, colour, upholstery, and number of doors. | **done** |
| Additional specific data to be entered for trucks only will be maximum load capacity, number of axles and number of wheels. | **done** |
| The data to be entered for drivers are licence number, first name, last name, mobile phone number, address, and states/territories they are licensed to drive in and, the number of demerit points remaining on their license. | **done** |
| There will be a method to display a given drivers' details and, a method to reduce remaining demerit points and, a method to increase remaining demerit points for a driver. Should the remaining demerit points fall to 3 or below, a warning message should be printed stating that “License suspension is imminent”. Demerit points should not be allowed to fall below zero or increase beyond 12. | **done** |
| There will be a method to update the kilometres for any given vehicle | **done** |
| There will be a method to change the color of any given car | **done** |
| There will be a way to display a trucks specific and general data along with an associated given drivers' details (name, license number, demerit points and, the states/territories in which a driver is licensed to drive) | **done** |
| There will be a way to display a cars specific and general data along with an associated given drivers' details (name, license number, demerit points and, the states/territories in which a driver is licensed to drive) | **done** |
| There will be a way to display a cars general data only. *(registration number, make, model and kilometres driven)* | **done** |
| There will be a way to display a trucks general data only.  *(registration number, make, model and kilometres driven)* | **done** |
| There will be a way to display all the data for a given driver | **done** |
| Each address detail for a driver will be printed on a new line as follows  Street : 24 Lincoln Road City : Essendon State : Victoria  Postcode : 3040 | **done** |
| The states/territories in which a driver is licensed to drive are to be displayed and formatted as follows.  The driver is licenced to drive in the following states : Victoria : New South : Tasmania | **done** |
| As stated in the scenario there will be no menu system for data entry. Simply hard-coding object data and subsequently displaying it is all that is required. *See appendices for examples* | **done** |
| For proof of concept you are to write and read the drivers details only, to and from an electronic file. | **Done** |
| Handle any input without errors | **done** |

Record the development of your script using the GitHub development platform. Push and pull changes at the end of a day to keep the contents of the local and remote repositories synchronised. Add your instructor as a collaborator to your repository. You instructor, acting as your supervisor, will check the repository and make comments during code development.

*Do not produce code with Window elements or complexity beyond what is required.*

## Task 1.4 Debug

Use an IDE and its inbuilt debugger to debug your script

* Provide a short 3 minute Panopto video with commentary that shows the IDE debugger in action, including stopping at a breakpoint, stepping into an object instantiation process, viewing a class variable and changing its value, viewing an instance variable and changing its value and, identification of the cause of a logic error. *Be brief by not including any more content than is required for the task*

|  |
| --- |
| Upload your video to the assessment drop box for this unit and enter the hyperlink to the Panopto video here. **Ensure** that your instructor has read access to the video  <https://vu.ap.panopto.com/Panopto/Pages/Capture.aspx?folderId=c577838c-54af-41df-9255-ae2f01776f6d> |

**Task 1.5 Deciding a course of action when debugging**

* Provide a list of three semantic errors you have encountered and how you rectified them.   
  *Note that syntax errors are not acceptable. The errors must be caused through incorrect logic*

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| *Updating demerit points higher than 12 points.* | User entered incorrect values for demerit points. Mechanism should be implemented. |
| *Calculating negative values to the demerit points value when decrementing demerit points* | User entered invalid inputs. Should implement mechanism to avoid the error. |
| *Resulting negative values when calculating driven KMs* | User entered negative values as inputs. Should check inputs before calculation. |

## Task 1.6 Develop tests

Develop test cases to confirm the code meets the program specifications and user requirements. List all the test cases that you will run.  
Be sure to include all tests required to check all class methods. See examples at Appendix C - Hardcoding and Display examples.

|  |  |  |
| --- | --- | --- |
| **Brief Description:  *(what is being tested)*** | **Brief Description:  *(what is being tested)*** | **Brief Description:  *(what is being tested)*** |
| Car1 general and specific data is displayed | Driver 1 details are displayed | Display warning message “License suspension is imminent” when the demerit points are decrementing below 3. |
| Car1 kilometres are updated | Driver 2 details are displayed | Testing to verify new demerit points value is greater than maximum demerit points value (12).  Display error message “Invalid entry” after adding invalid new demerit points. |
| Car1 colour is updated | Driver 1 demerit points are displayed | Testing invalid inputs when decrementing demerit points. Error message “Invalid entry” is displayed. |
| Car2 general and specific data is displayed | Driver 1 demerit points are increased | Validate drivenKM as a positive value. Display error message for negative values. |
| Car 2 kilometres are updated | Driver 1 demerit points are decremented | Display contents of driverDetails.txt |
| Care 2 colour is updated | Driver 2 demerit points are displayed | Driver details are written to a text file called driverDetails.txt |
| Truck 1 general and specific data is displayed | Driver 2 demerit points are increased |  |
| Truck 1 kilometres are updated | Driver 2 demerit points are decremented |  |

**Task 1.7 Document tests**

Record two of your test cases below as well as any subsequent code modification that occurred if any.

|  |  |  |
| --- | --- | --- |
| **EXAMPLE**  **Title: Car 1 Kilometers updated**  **Preconditions: car1 object populated with all car data** | | |
| **Steps** | **Expected response or output** | **Actual response or output** |
| 1. Display car1 details  car1.display() | Vehicle registration number is 3WS4HY. Make is Suzuki. Model is Swift. Odometer 16300 km  The Car details are: body type Sedan, colour silver, leather interior, 5 doors | Vehicle registration number is 3WS4HY. Make is Suzuki. Model is Swift. Odometer 16300 km  The Car details are: body type Sedan, colour silver, leather interior, 5 doors |
| 2. Update car1 Kilometers  car1.updatekm() | NA |  |
| 3. Display car1 details  car1.display() | Vehicle registration number is 3WS4HY. Make is Suzuki. Model is Swift. Odometer 18150 km  The Car details are: body type Sedan, colour silver, leather interior, 5 doors | Vehicle registration number is 3WS4HY. Make is Suzuki. Model is Swift. Odometer 18150 km  The Car details are: body type Sedan, colour silver, leather interior, 5 doors |
| **Code modification** | None required | |

|  |  |  |
| --- | --- | --- |
| **Title: Updating truck2 driven Kms.**  **Preconditions: truck 2 object populated with all truck details** | | |
| **Steps** | **Expected response or output** | **Actual response or output** |
| 1. Display truck 2 details  truck2.displayVehicleData() | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 76520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 76520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels |
| 2. Update truck2 driven KMs  truck2.updateKM(70000)  Driven KMs = 70000 | Display massage “successfully updated driven KMs”. | Display message “successfully updated driven KMs”. |
| 3. Display truck 2 details  truck2.displayVehicleData()  Driven KMs = 70000 | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 146520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 146520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels |
| 3. Update truck2 driven KMs  truck2.updateKM(-70000)  Driven KMs = -70000 | Display message “Invalid Entry” | Display message “Invalid Entry” |
| 4. Display truck 2 details  truck2.displayVehicleData()  Driven KMs = -70000 | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 76520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels | Vehicle registration number is ARC542 is a Hyundai. Model is iLoad. Odometer 76520 km  The driver of the vehicle is Boris Johnson  Additional details: maximum load 2 tonnes, no of Axles 2 , 4 wheels |
| **Code modification** | None required | |

|  |  |  |
| --- | --- | --- |
| **Title: Removing demerit points – driver 2**  **Preconditions: drive 2 object populated with driver’s remaining demerit points** | | |
| **Steps** | **Expected response or output** | **Actual response or output** |
| 1. display driver 2 remaining demerit points  Driver2.displayDemeritPoints() | The driver Boris Johnson has 5 x Demerit Points remaining | The driver Boris Johnson has 5 x Demerit Points remaining |
| 2. remove demerit points from driver 2  Driver2.deleteDemeritPoints(3) | Display following messages:  WARNING MESSAGE:  Licence suspension is imminent  2 x Demerit points remaining | Display following messages:  WARNING MESSAGE:  Licence suspension is imminent  2 x Demerit points remaining |
| 3. . display driver 2 remaining demerit points  Driver2.displayDemeritPoints() | The driver Boris Johnson has 2 x Demerit Points remaining | The driver Boris Johnson has 2 x Demerit Points remaining |
| **Code modification** | None required | |

## Task 1.8 Review application against users requirements

* Contact your instructor, who is acting as your supervisor, when your application is complete to confirm that your application meets the design specification and user requirements. Document this discussion via bullet points or brief notes, make any required adjustments to the code and make a further GitHub commit stating the words “Acted on review with supervisor” in the summary field.

|  |
| --- |
| *Enter notes of*   * The discussion with your supervisor confirming that your application meets the design specification and user requirements. * Asked to validate kilometres and demerit points values. They shouldn’t below zero. * Asked to implement functions to write and read drivers’ details. * Asked to implement function to display vehicle general data. |

## Task 1.9 Obtain client sign-off

* Contact your instructor, who is acting as your client, when your application is complete to confirm that your application meets the user requirements and to obtain sign-off. Document this discussion via bullet points or brief notes, make adjustments to the code if required and make a further GitHub commit stating the words “Sign-off from client” in the summary field. Schedule another meeting to confirm requirements if code changes are made.

|  |
| --- |
| *Enter notes of*   * The discussion with your client confirming that your application meets their requirements * **Asked to display demerit points when displaying driver details.** |

Ensure you have pulled all commits from the remote Github and compress your LOCAL GitHub repository folder in .zip format and, submit as part of your assessment along with this assessment document and the Panopto video.

**Appendix A: Supervisor consultation**

Consultation with your supervisor has identified the following.

* The sequence of the code in which the objects are instantiated, and methods called, is to be hard coded in a main() function.
  + Using your own made-up data, you are to populate two instantiated objects each for the driver, truck, and car classes.
  + You should call every method within the classes to show that all methods are working as expected for all instantiated objects.
* Drivers address details will be passed as a dictionary
* States in which a driver is licensed to drive will be passed as a list
* The maximum number of allowed demerit points is to be implemented as a class variable
* A ‘for’ loop will be used to extract the keys and values from a drivers address for display

Example display of driver details dictionary

The driver Clarice Starling, has a driver licence number : 038121243

Contact phone number is : 0404323232.

Driver address is :

Street : 40 Beaconsfield Road

City : Mackay

State : VIC

PostCode : 4740

* When displaying the states where licenses for a driver are held, a ‘for’ loop will be used to extract the elements without brackets and quotes

Example display of state licensing list display

The driver is licenced to drive in the following states : Victoria : New South Wales : Queensland : Western Australia : Tasmania :

* Objects of the driver class are to be aggregated with objects of the vehicle class.
* Values for kilometres or demerit points must be valid and not less than zero
* Embedded descriptive comments (#) to clarify the meaning of the code are required
* Embedded Docstrings documentation for all classes, methods and functions are required
* Style to the PEP 8 style guide is required.

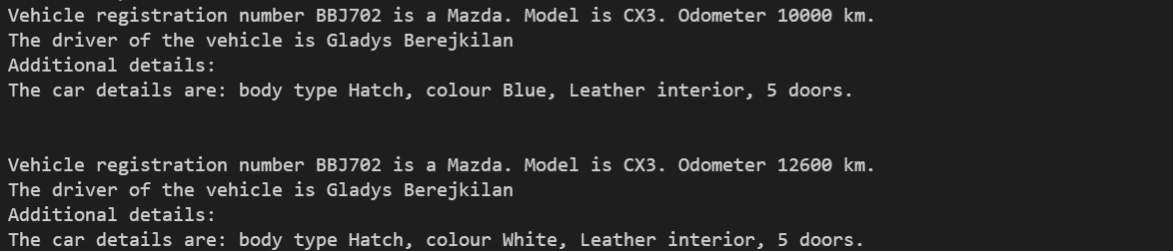
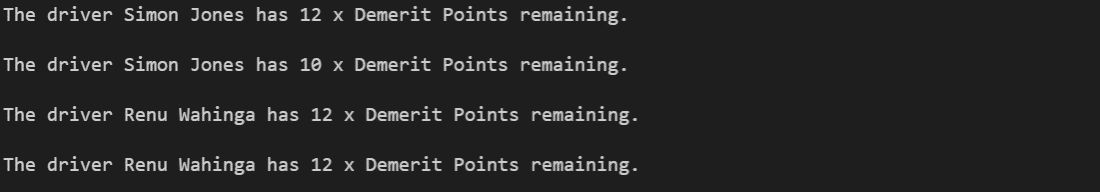
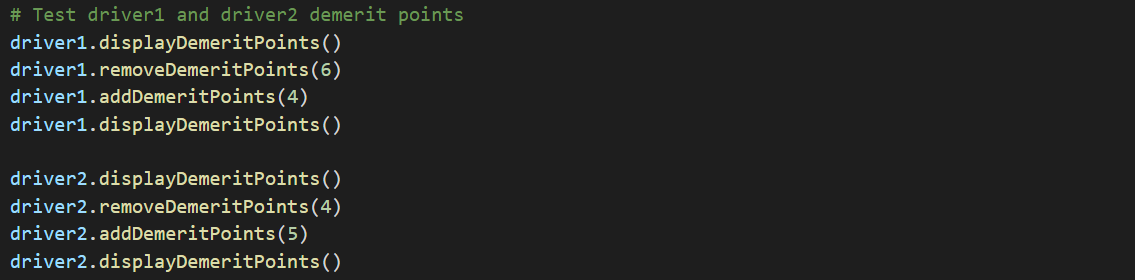
**Appendix B – Coding Checklist**

|  |  |
| --- | --- |
| **Coding Checklist**  Your code must include the following criteria at a minimum. *The checklist is given to* ***ensure*** *all aspects of the assessment are covered. Your assessor will refer to this checklist when marking your submission* | |
| Criteria | Note where the implementation occurs in your code |
| two classes that each contain four instance variables |  |
| one class that uses object construction |  |
| one class that uses object aggregation |  |
| one class that uses arrays of primitive data types twice | *dictionary to store addresses;  List to store state licensing* |
| one instance of polymorphism for code extensibiltiy |  |
| one baseclass and one subclass for inheritance |  |
| one class variable | *maximum demerit points* |
| three language data types |  |
| three operators |  |
| three expressions |  |
| Syntax for one sequence construct | *main()* |
| One selection construct | *‘if’ based selection based on demerit point value* |
| Two iteration constructs | *'for' loops to print out address lines and state licensing* |

**Appendix C – Hardcoding examples**



**Appendix D –Sequence and display examples**



*\*Note that these are examples only and do not indicate every detail required for hardcoding, nor every detail required for update and display, within the assessment task. You should refer to the user requirements check list and supervisor consultation notes to confirm assessment task requirements.*