|  |  |
| --- | --- |
| **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 REMITE 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 application and test it via hard-coding 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: Aliveni thodupunuri Student ID Number: s8061705**

## 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*

|  |
| --- |
| *Enter brief notes or dot points of the discussion*  *It is understood that there is no entry menu as advised.*  *Only 1 car display and driver details display working is sufficient for concept* |

|  |
| --- |
| 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. |
| 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*  [:52](https://softwareengin-fvi3735.slack.com/archives/D03H3CQAEAV/p1658235174112499)  display car 1 specific details               Console.WriteLine bodyType, colour, upholstery, numOfDoors  display car1 general details               Console.WriteLine registrationNumber, make, model, kmDriven  display car 1 general and specific and driver details  Console.WriteLine registrationNumber, make, model, kmDriven, bodyType, colour,   upholstery, numOfDoors, licenseNo, firstName, lastName, mobilePhoneNo,   maxDemerits,                Iterate (loop) through the address array                             Console.WriteLine address[index]                Foreach item in address                              Console.Write address[item] + “ “                Iterate (loop) through the stateLicense array                              Console.WriteLine stateLicense[index]  Add 2 DM to driver 1 if maxDemerits + dm < 0                 maxDemerits = 0  else if maxDemerits + dm > 9 and maxDemerits + dm <= 12                 Console.WriteLine “License suspension imminent”   else if maxDemerits + dm > 12                 maxDemerits = 12                Console.WriteLine “Demerits cannot go over 12, demerits set to 12”  else                maxDemerits = maxDemerits + DM  add 100 to KM for car1, if kmDriven + KM < 0               Console.WriteLine error message  else               update kmDriven = kmDriven + KM  Change colour to red for car1  display car 1 general, specific and driver details Console.WriteLine registrationNumber, make, model, kmDriven, bodyType, colour, upholstery, numOfDoors, licenseNo, firstName, lastName, mobilePhoneNo, maxDemerits,               Iterate (loop) through the address & stateLicense array                             Console.WriteLine address[index]               Foreach item in address                              Console.Write address[item] + “ “               Foreach item in stateLicense array                              Console.WriteLine stateLicense[index]  repeat above for other objects car2, truck1, truck2, driver 2 |

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

Translate your pseudo code and UML class diagram into a C# console program that adheres to the naming, organisation and white space rules of C# at Google Style guide. 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://google.github.io/styleguide/csharp-style.html>   
  
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 |
| Console based application |  |
| The general data to be entered for all vehicles will be registration number, make, model and kilometres driven. |  |
| Additional specific data to be entered for cars only will be body type, colour, upholstery, and number of doors. |  |
| Additional specific data to be entered for trucks only will be maximum load capacity, number of axles and number of wheels. |  |
| 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. |  |
| There will be a method to display a given drivers details |  |
| There will be a method to increase and decrease remaining demerit points for a driver. Should the demerit points rise to 9 or above, 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. |  |
| There will be a method to update the kilometres for any given vehicle |  |
| There will be a method to change the color of any given car |  |
| There will be a way to display a cars general data only and, a way to display a cars specific data only and, a way to display both data sets together. |  |
| There will be a way to display a trucks general data only and, a way to display a ~~cars~~ Trucks specific data only and, a way to display both data sets together. |  |
| There will be a way to display a trucks specific and general data along with an associated given drivers' details |  |
| There will be a way to display a cars specific and general data along with an associated given drivers' details |  |
| Each address detail for a driver will be displayed on a new line as follows  Street : 24 Lincoln Road City : Essendon State : Victoria  Postcode : 3040 |  |
| 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 |  |
| As stated in the scenario there will be no menu system for data entry. Simply hard-coding object data and subsequently displaying it via method calls, is all that is required. *See appendices for examples* |  |
| For proof of concept you are to write and read the drivers details only, to and from an electronic file.This is a simple task as no formatting is required. Write the details and read them back again |  |
| Handle any input without errors |  |

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. If you have an existing repository to which your instructor is a collaborator then you can create a relevant directory structure there rather than creating a new repository.

*Do not produce code with 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 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/Viewer.aspx?id=e7acb955-a779-44e6-852e-aedb004c663c> |

**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** |
| *e.g. Code to view the text file is not being ran* | *The menu option was capitalised but user entry was in lower case* |
| *Assignment of Driver Address & State license causing null exception* | In the constructor for Driver class, the address and state license array were not assigned correctly hence it was throwing runtime exception  Corrected the sequence of value assignment to address and state license and assigned the arrays to the driver constructor |
| *displayGeneralSpecific method was initially set in Vehicle class* | There was error with the logic in calling the display method and was throwing object reference error.  Corrected the logic by providing display general method in Vehicle class and access them from car/truck object |
| *KM were not updated correctly where the update KM were sent in negative* | Added condition to check for KM update value and if the value was setting the KM into negative, then setting them to 0. |

## 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)*** |
| e.g Car1 general and specific data is displayed | Driver1 demerits updated where demerits will go between 9 & 12, demerits updated with given value and warning message given | Driver1 details to be written to text file |
| e.g Car1 kilometres are updated | Driver 1 demerits updated where total demerits go over 12, it should set to 0 and message given | Driver1 details to be read from text file |
| e.g Car1 color updated | Driver1 demerits updated where total demerits are between 0 and 9, it will be updated to given value |  |
| Car1 kilometres cannot go into negative, if going into negative, set to 0 | Driver 1 demerits updated where total demerits go into negative, it should set to 0 |  |
|  |  |  |
| *Add rows and/or columns as needed* | | |

**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: myCar1 Colour change**  **Preconditions: myCar1 object populated with all car details** | | |
| **Steps** | **Expected response or output** | **Actual response or output** |
| 1. Display myCar1 details (Specific)  myCar1.displaySpecific() | Car Details  Body Type: SUV Colour: Grey Upholstery: Leather No of Doors: 4 should be displayed | Car Details  Body Type: SUV Colour: Grey Upholstery: Leather No of Doors: 4 should is displayed |
| 2. Update car1 colour  myCar1.changeColour(“Red”) | N/A | N/A |
| 3. Display myCar1 details (Specific)  myCar1.displaySpecific() | Car Details  Body Type: SUV Colour: Red Upholstery: Leather No of Doors: 4 should be displayed | Car Details  Body Type: SUV Colour: Red Upholstery: Leather No of Doors: 4 should is displayed |
| **Code modification** | N/A | |

|  |  |  |
| --- | --- | --- |
| **Title: Driver 1 update demerits**  **Preconditions: driver1 object populated with all the details** | | |
| **Steps** | **Expected response or output** | **Actual response or output** |
| 1. Display driver1 details  Driver1.display() | Driver details – License No: DL123456 Driver First Name: Aliveni Driver Last Name: Thodupunuri Mobile Phone No: 401998778 Max Demerits: 0  The driver is licensed to drive in States : VIC : NSW  Address is : Street: 16 Tyalla St City: Wyndham Vale State: VIC Post Code: 3024 Should be displayed | Driver details – License No: DL123456 Driver First Name: Aliveni Driver Last Name: Thodupunuri Mobile Phone No: 401998778 Max Demerits: 0  The driver is licensed to drive in States : VIC : NSW  Address is : Street: 16 Tyalla St City: Wyndham Vale State: VIC Post Code: 3024  is displayed |
| 2. Update driver1 demerits Driver1.addDeleteDemerits(2) | Max demerits updated | Max demerits updated |
| 3. Display driver1 details  Driver1.display() | Driver details – License No: DL123456 Driver First Name: Aliveni Driver Last Name: Thodupunuri Mobile Phone No: 401998778 Max Demerits: 2  The driver is licensed to drive in States : VIC : NSW  Address is : Street: 16 Tyalla St City: Wyndham Vale State: VIC Post Code: 3024  Should be displayed | Driver details – License No: DL123456 Driver First Name: Aliveni Driver Last Name: Thodupunuri Mobile Phone No: 401998778 Max Demerits: 2  The driver is licensed to drive in States : VIC : NSW  Address is : Street: 16 Tyalla St City: Wyndham Vale State: VIC Post Code: 3024  is displayed |
| **Code modification** | n/a | |

## 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.   Task 1.8 supervisor review.   1. There should be no need to assign kms to another variable. Rather, use kms as is.   For example public void updateKM(int kms)    {            //update kilometers of the object calling this method if (kms + kmDriven < 0)  //checking if KM update will take total km to negative            {Also, if kms is less than zero, kms should remain unchanged because that is an error the user has made   * Corrected   public void updateKM(int kms)  {  //update kilometers of the object calling this method      if (kms + kmDriven < 0) //checking if KM update will take total km to negative  {  Console.WriteLine("Kilometers driven cannot go into negative, no changes made");  }  else  {  kmDriven = kmDriven + kms; // adding provided km to class variable  }  }   1. DMs is the same in that there is no need to assign the passed   DM to a temporary variable   * Corrected   public void addDeleteDemerits(int DM)  {  if (DM + maxDemerits < 0)  {  maxDemerits = 0;  Console.WriteLine("\nDemerits cannot go negative, Your new demerits are 0");  }  else if( maxDemerits + DM > 9 & maxDemerits + DM <= 12)  {  maxDemerits = maxDemerits + DM;  Console.WriteLine("\nYour new demerits are: " + maxDemerits + ", License suspension imminent");  }  else if ( maxDemerits + DM > 12 )  //else  {  maxDemerits = 12;  Console.WriteLine( "\nDemerits cannot go over 12, demerits set to 12 ");    }  else  {  maxDemerits = maxDemerits + DM;  Console.WriteLine("\nMax demirts updated");  }  }   * Make fields private and/or protected where possible   + Vehicle class is public   + Car/Truck/Driver are private   public class Vehicle  {  //Declare attributes of the Vehicle class  public string registrationNumber;  public string make;  public string model;  public int kmDriven;  public Driver driver;  public class Car : Vehicle  {  //Car class attributes  private string bodyType;  private string colour;  private string upholstery;  private int noOfDoors;  public class Truck : Vehicle  {  //Declare attributes of the truck class  private int maximumLoad;  private int noOfAxles;  private int noOfWheels;  public class Driver  {  //instance variables  private string licenseNo;  private string firstName;  private string lastName;  private int mobilePhoneNo;  private string[] address;  private string[] stateLicense;  private int maxDemerits;   1. The following has confusing variable names.   Use the same names for fields and passed parameters, then use this.  <fieldName> to differentiate between them  public Driver(string lno, string fn, string ln, int mno, string[] addr, string[] sts, int dms)         {               licenseNo = lno;             firstName = fn  ;e.g  public Driver(string licenseNo, string fn, string ln, int mno, string[] addr, string[] sts, int dms)         {               this.licenseNo = licenseNo   * Corrected   //Constructor of Driver class with arguments  public Driver(string licenseNo, string firstName, string lastName, int mobilePhoneNo, string[] address, string[] stateLicense, int maxDemerits)  {  this.licenseNo = licenseNo;  this.firstName = firstName;  this.lastName = lastName;  this.mobilePhoneNo = mobilePhoneNo;  this.address = address;  this.stateLicense = stateLicense;  this.maxDemerits = maxDemerits;    }   1. There is a large issue.   Polymorphism needs to be incorporated. But I cannot see where that is done. Multiple constructors in a class would be ok for that, however although multiple constructors have been created, only one constructor is used.   * Corrected   Created car2 with default constructor.  Car Class – unused constructor    //Constructor of Car class with default assignment  public Car()  {  bodyType = "Sedan";  colour = "Blue";  upholstery = "Cloth trim";  noOfDoors = 4;  }  class Program  {  static void Main(string[] args)  {  //Demonstrating another constructor of car class  Car myCar2 = new Car();  myCar2.displaySpecific();  output showing default values from Car() constructor being displayed    The more obvious may to incorporate polymorphism is to declare displayGeneral() in the Vehicle class as 'virtual' and then override it in the sub-classes.  -Corrected  Since the vehicle general details were same for both car & truck, I called the vehicle class displayGeneral() for each object of Car/Truck. However, as per your suggestion, I have now used Virtual and Override  public virtual void displayGeneral()  {  //display general details of the object calling this method  Console.WriteLine("\nVehicle Details");  Console.WriteLine("Registration is: " + this.registrationNumber + " Make is: " + this.make + " Model is: " + this.model + " Kilometers Driven: " + this.kmDriven);  }    public override void displayGeneral()  {  //display general details of the object calling this method  Console.WriteLine("\nCar Vehicle Details");  Console.WriteLine("Registration is: " + this.registrationNumber + " Make is: " + this.make + " Model is: " + this.model + " Kilometers Driven: " + this.kmDriven);  }  public override void displayGeneral()  {  //display general details of the object calling this method  Console.WriteLine("\nTruck Vehicle Details");  Console.WriteLine("Registration is: " + this.registrationNumber + " Make is: " + this.make + " Model is: " + this.model + " Kilometers Driven: " + this.kmDriven);  }  5.. Suggest to use AppendAllLines() to write an array directly rather than writing strings via indexing e.g File.AppendAllLInes("C:\\csharpfiles\\test.txt", address); (edited)   * Corrected |

## 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 |

Ensure you have pulled all commits from the remote Github and compress your REMOTE 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.
  + 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 an array
* States in which a driver is licensed to drive will be passed as an array
* A ‘for’ loop will be used to extract the drivers address from the array for display

Example display of driver details

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

* A 'for' loop will be used to extract the the states where licenses for a driver are held, for display

Example display of state licensing list display

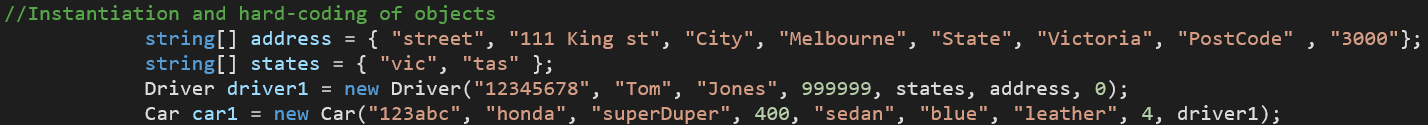
The driver is licenced to drive in the following states : Victoria : New South Wales : Queensland : Western Australia : Tasmania :

* The maximum number of allowed demerit points is to be implemented as a class variable
* 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
* Coding to the Google style guide is required. <https://google.github.io/styleguide/csharp-style.html>

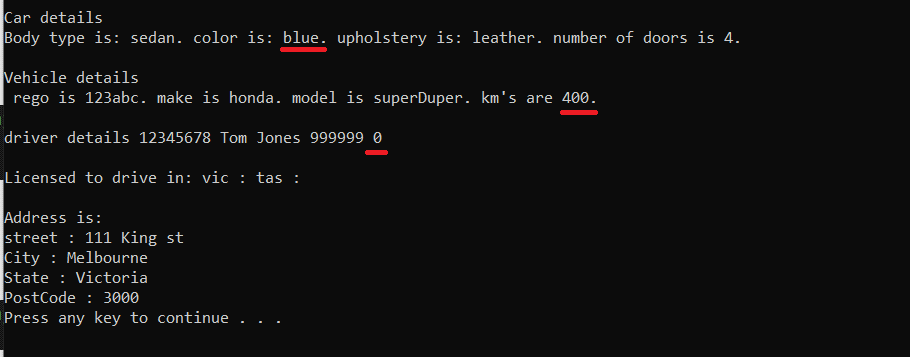
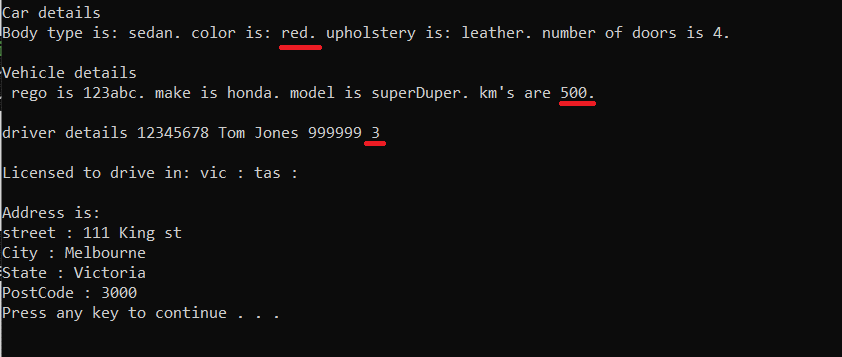
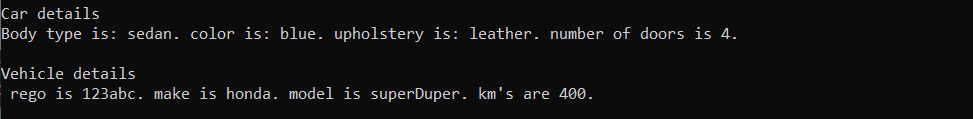
**Appendix B – Coding Checklist**

|  |  |
| --- | --- |
| **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 at least four instance variables |  |
| one class that uses object construction |  |
| one class that uses object aggregation | *Driver is aggregated with vehicle* |
| one class that uses arrays of primitive data types twice | *Addresses and 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 | *e.g string, int, array* |
| three operators |  |
| three expressions |  |
| Syntax for one sequence construct | *Hardcoded method calls* |
| One selection construct | *‘if’ selection based on demerit point value* |
| Two iteration constructs | *'for' loops to print out address lines and state licensing* |
| Read and write to a file once |  |

**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.*