WRAV102/MSEV102: Practical 10

No completion marks will be allocated to this practical

Objectives

- Classes and Objects
- Text files
- List class
- Arraylist
- Sorting using bubble sort, selection sort and insertion sort algorithms
- Using the sorted state of the list to determine the appropriate searching algorithm – linear search or binary search.

How to save your program for later use?

1. Use the **File/Save All** option. This saves it in the location you identified in the beginning.

How do I submit my work?

This prac does not have to be submitted to be marked to an assistant in the practical session. Your attendance at the session will be recorded.

Important note:

When working with bigger projects, it is good practice to clean any temporary files from your project folder. To do this in Visual Studio, click on Build, then select Clean Solution. Save your projects on your Submissions folder (S drive folder).

Task

For this task, you are returning to the scenario of the Car Dealer *(prac 5 and prac 6*). For this version, the functional required have been adjusted. Some functionality removed, some new functionality added.

For this version, a starter project has been created and is available on Moodle. The starter project is compressed to a zipped format. Once you have downloaded the project folder, you need to unzip the folder first.

- Right-click on the file `CarDealer.zip`
- Select Extract All
- Browse to the folder where you would like the CarDealer folder to be (I would suggest putting it into a Prac 10 folder)
- Click on Extract

You should follow one of the following methods to continue with this task:

- Open the application *(by opening the .sln file)*, complete the task by checking the requirements below. The text file is already in the correct folder.
- Create a new console application *(select the option to create a new Console App (.NET Framework)*. Create the required classes in your application. Copy the provided code from the starter application to the relevant classes of your newly created application. You will also need to copy the provided text file to the bin\Debug folder of your project folder.

The skeleton project contains:

- An object class named **Car**
- A list class named **CarList**
- The regular application class (program.cs)

Car class

The `Car` class already contains some aspects, do not alter the existing code, but add any methods required for the functionality of the program.

A car object has the following properties: `Registration number`, `ModelName`, `Year`, `Colour` and `Price`.

CarList class

You can decide to implement the list class using an array or an arraylist – then ensure that you create methods that will work for your choice of data structure.

The CarList class already contains some aspects, do not alter the existing code, but add any aspects and methods to do the following:

- Initialise the array/ arraylist from the provided text file (CarData.txt).
- Add a new car to the end of the list.
- Delete a car from the list
- Search for a car object, based on a Registration number provided – depending on the sorted state of the list, you should determine whether to make use of a linear search or binary search algorithm (both search algorithms should be provided for). You should use binary search if the list is sorted on Registration number at the time of the request. *For the sake of this task, include a feedback message indicating whether the process is making use of the linear or binary search algorithm.*
- Insertion sort to sort the car list in ascending order of registration number.
- Selection sort to sort the car list in descending order of year.
- Bubble sort to sort the car list in descending order of price.
- Display all the cars in the list – your method must make use of a method from the object class.

For the duration of this task, keep track of sorted state of the list as follows:

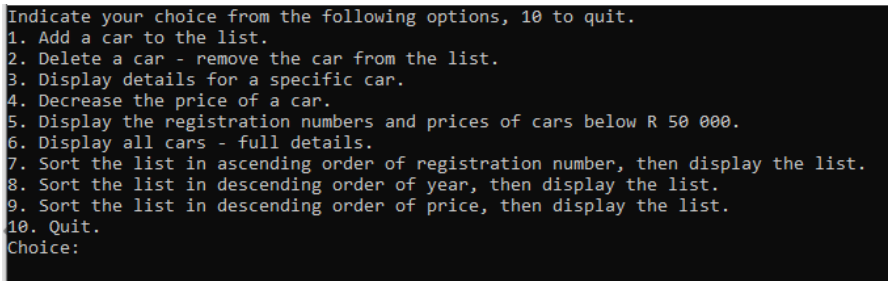
- List not sorted: sortedState = 0
- List sorted in ascending order of registration number: sortedState = 1
- List sorted in descending order of year: sortedState = 2
- List sorted in descending order of price: sortedState = 3

Make use of private methods to provide the implementation of the sorting and searching algorithms, with public methods that will call to these private methods, and set the sorted state as required – the public methods is what you should call from the application class.

Application class

The main method allows the user to interact with the car list. The application class already contains some aspects, do not alter the existing code, but add methods as needed to provide the required functionality:

Functionality required:



- Allow the user to add a new car to the list. Prompt the user for the all the required values. Display a confirmation message when the car was added to the list.
- Delete a car (delete the car from the list). Create a method to do this, your method must prompt the user for the registration number of the car (convert the input to uppercase). Your method must use methods from the list and object class. Display relevant messages.
- Prompts the user for a registration number and display all the data for the specific car. If the car is not found, display an appropriate message.
- Decrease the price of a car. Create a method to do this, your method must prompt the user for the registration number of the car (convert the input to upper case) and for a percentage to decrease the

price with, calculate the new price, then use one of the new methods from the object class to decrease the price.

- Displays the registration numbers and prices for all cheap cars (call a method from the list class to do this).
- Displays all the fields for all the cars in the list (you should have created this method for prac 5).
- Sort the list in ascending order of registration numbers, then display the list – display full details. (You should use the insertion sort algorithm for this task.)
- Sort the list in descending order of year then display the list – display full details. (You should use the selection sort algorithm for this task.)
- Sort the list in descending order of price then display the list- – display full details. (You should use the bubble sort algorithm for this task.)

As a last function of your application, you should write the data from the array/arraylist to a comma delimited file (one line of data for each car). **For the sake of this exercise, write your data to a different file.**

The user should be in control of invoking the sort functionality. You should not call any sorting methods unless that is what the user requested.

Important Note:
Since you are making use of a list class there should be NO array/arraylist operations in your Application class; while **all** user input belongs in the application class.