LAB 211 Assignment

Type:	**Long Assignment**
Code:	**J1.L.P0029**
LOC:	**500**
Slot(s):	**N/A**

Title

Car insurance Management

Background

- An insurance company in Ho Chi Minh City needs to build a software to manage insurance information of vehicles. Key features include registering car information, search and update or delete car information, manage insurance contracts related to registered vehicles such as: Create insurance contracts, Display list of insurance contracts, related reports, ...
- Students are required to analyze and design the program using an object-oriented programming (OOP) approach. Features such as abstraction, polymorphism, encapsulation, and inheritance must be applied during the development process.

Program Specifications

Build the CarInsurance project with the menu as follows:

1. Add new car
2. Find a car by license plate
3. Update car information
4. Delete car information
5. Add an insurance statement
6. List of insurance statements
7. Report uninsured cars
8. Save data
9. Quit

Each menu selection calls an appropriate function to execute the selected menu item to solve its meaning that serve the needs of the user. Your program must display a menu after each action and wait for the user to choose another option until the user chooses to quit the program.

Features:

Function 1. Add a car information - 50 LOC

Description:
This function allows the creation of a new vehicle information by collecting necessary details, validating inputs, …

Input Requirements:
The function requires the following customer details:

- **License plate**: A unique 9-character string.
- **Car owner**: Name of the person who owns the vehicle.
- **Phone Number**: Owner's phone number.
- **Car brand**: Brand of the vehicle.

- **The value of the vehicle**: The value of the vehicle that the owner has paid for
- **Registration Date:** The date on which the vehicle was registered for traffic
- **Place of registration**: Is the name of a district in Ho Chi Minh City.
- **Number of seats**: The number of seats

Validation Rules:

1. **License plate:**
   - The first two characters are numbers in the range from 50 to 59, the next two characters include a character describing the district code of Ho Chi Minh City (*for example, P-Tan Binh, S-Binh Thanh, X-Thu Duc, ...*) combined with a digit from 1 to 9, followed by 5 digits
   - Must be unique.
   - Cannot be empty
2. **Car owner:**
   - Cannot be empty.
   - Length must be between 2 and 25 characters.
3. **Phone Number:**
   - Must contain exactly 10 digits.
   - Must belong to a valid Vietnamese network operator.
4. **Car brand:**
   - Cannot be empty.
   - Length must be between 5 and 12 characters.
5. **The value of the vehicle:**
   - Is an integer greater than 999.
6. **Registration Date:**
   - Cannot be null.
   - Valid date value
   - Before the current date
7. **Place of registration:**
   - Determined based on the 3rd character in the license plate
   - Is the name of a district in Ho Chi Minh City. For example: X - Thu Duc; S - Binh Thanh; T - District 1; V - Go Vap,... *You can google to know about this issue*.
8. **Number of seats:**
   - Is an integer with a value from 4 to 36.

Operation Workflow:

1. Require user to enter vehicle details.
2. Validate each input based on the rules above.
3. Save the vehicle information if all inputs are valid.

4. Prompt the user to either continue entering new vehicle information or return to the main menu

Function 2. Find a car by license plate – 50 LOC

Description:

This function allows users to look up information about a vehicle based on the license plate number registered in the system.

Operation Workflow:

1. **Input the "license plate"**:
   - Requires users to enter the license plate number of the car they want to search for.
2. **Search Logic:**
   - Perform a search across the vehicle records to find license plate number that match the input.
3. **Handle Cases:**
   - **If the license plate number matches**:
     - Display vehicle information, including: License plate, Car owner, Phone Number, Car brand, Place of registration, Number of seats, Registration Date, The value of the vehicle.
   - **If no matches are found**:
     - Display the message: **"No one matches the search criteria!"**.
4. **Return to Main Menu:**
   - After displaying the message, prompt the user to return to the main menu or continue searching for others.

Function 3. Update car information – 50 LOC

Description:
This function allows users to update information about registered vehicle in the system. Information that can be updated includes.

- Car owner,
- Phone Number
- Car brand
- Number of seats

Input Requirements:

1. **License plate**: The unique identifier of the vehicle.
2. Fields for update:
   - **Car owner**: Name of the person who owns the vehicle.
   - **Phone Number**: A 10-digit number belonging to a network operator in Vietnam
   - **Number of seats**: Is an integer between 4 and 36.
   - **Car brand**: between 5 and 12 characters

Validation Rules:

- Ensure that the License plate exists in the profile database before allowing updates.
- Apply the respective validation rules for each field being updated.
- Keep old information, if not enter new data.

Operation Workflow:

1. Prompt the user to enter the **License plate**.
2. Check if the vehicle exists in the profile:
   - If the vehicle exists:
     a. Prompt the user to update the desired fields (Car owner, Phone, Number of seats, brand)
     b. Validate the inputs based on the specified rules.
     c. Save the updated information.
     d. Display a success message.
   - If the vehicle does not exist:
     - Display the message: **"This vehicle does not exist ."**
3. Ask the user whether to continue with another update or return to the main menu

Function 4. Delete car information – 50 LOC

Description:
This function allows users to delete a registered vehicle's information based on their **License plate**. If the **License plate** exists, the program will display the relevant details and ask for confirmation before proceeding with the deletion. If the vehicle is not registered, an appropriate notification will be displayed.

Operation Workflow:

1. **Input the License plate:**
   - Prompt the user to enter the **License plate**.
2. **Check the License plate:**
   - Search the registration records for the entered **License plate**.
3. **Handle Cases:**
   - **If the vehicle exists**:
     - Check the vehicle information in the list of insurance statements;
       - if the vehicle has been registered for insurance, display message and return to the main menu
       - if the vehicle is not registered for insurance, display the vehicle's details (e.g., Owner, Phone, Brand, Value of vehicle, Number of seats, Registration date).
     - Ask the user for confirmation to delete the record:
       - If the user confirms, delete the record and display the message: **"The vehicle information has been successfully deleted."**
       - If the user cancels, return to the main menu without making changes.
   - **If the vehicle does not exist**:

- Display the message: **"This vehicle has not registered yet."**

4. **Return to Main Menu:**
   - After handling the deletion (or lack thereof), return to the main menu.

Validation Rules:

- Ensure that the **License plate** follows the correct format
- Only delete the record if the vehicle exists, is not registered for insurance and the user confirms the action.

Sample Messages:

Case 1: Vehicle Exists and Confirmation is Given

Vehicle Details:

```
--------------------------------------------------
License plate    : 51F056789
Owner            : Alan Nguyen
Phone            : 0987654321
Car brand        : TOYOTA
Value of vehicle : 1,300,000,000
Number of seats  : 7
Registration date: 28/10/2019
--------------------------------------------------
```

Are you sure you want to delete this registration? (Y/N): Y

The registration has been successfully deleted.

Case 2: Vehicle Does Not Exist

This vehicle has not registered yet.

Function 5. Add an insurance statement – 50 LOC

Description:
This function allows the creation of a new new vehicle information by collecting necessary details, validating inputs, …

Input Requirements:
The function requires the following customer details:

- **Insurance id**: Insurance contract number.
- **Lisence plate**: License plate of the vehicle is registered for insurance
- **Established date**: Effective date of insurance.
- **Insurance period**: integer number.
- **Insurance fees**: Amount payable for the respective insurance statement.
- **Insurance owner**: Name of the beneficiary

Validation Rules:

1. **Insurance id:**
    - A unique 4-character string.
    - Cannot be empty
2. **Lisence plate**
    - Cannot be empty
    - Must be unique.
    - According to the license plate convention, introduced in the vehicle information management section
3. **Established date:**
    - Cannot be empty.
    - Invalid date.
4. **Insurance period:**
    - Must be one of the following: 12, 24 or 36.
5. **Insurance fees:**
    - Must be a positive integer value and calculate by formular
        - 25% of "the value of the vehicle" field if insurance period is 12
        - 20% of "the value of the vehicle" field multiplied by 2 if insurance period is 24
        - 15% of "the value of the vehicle" field multiplied by 3 if insurance period is 36
6. **Insurance owner:**
    - Cannot be empty.
    - Length must be between 2 and 25 characters

Operation Workflow:

1. Require user to enter insurance statement details.
2. Validate each input based on the rules above.
3. Save the vehicle information if all inputs are valid.

Prompt the user to either continue entering new insurance statement or return to the main menu

Function 6. List of insurance statements – 75 LOC

Description:

This function allows users to display information about contracts created in a specific year in the system.

Operation Workflow:

1. **Input the "Year"**:
    - Requires users to enter the year they want to list.
2. **Search Logic:**
    - Perform a search on the insurance records to find the insurance statement that it has "***Established date***" of year entered.

3. **Handle Cases:**
    - **If data is available,** the corresponding report will be displayed on the screen in a table format, sorted by insurance fees ascending.
    - **If no matches are found**:
        - Display the message: **"There are no statements in this year "**.
4. **Return to Main Menu:**
    - After displaying the message, prompt the user to return to the main menu or continue with another report.

Sample Messages:

Case 1: Identify the list of insurance contracts in 2023

Report : INSURANCE STATEMENTS
From : 01/01/2023 To: 12/31/2023

Sorted by: Established Date
Sort type : ASC

No.	Insurance Id	Established Date	License plate	Customer	Insurance period	Insurance fees
1	0006	01/17/2023	50H3-123.45	John Adams	12	$ 6000
2	0007	01/28/2023	51H1-888.88	Jamie Oliver	12	$ 9000
3	0008	02/14/2023	54H1-567.89	Will Smith	24	$ 14,400
4	0009	03/11/2023	50H1-222.22	Alan Nguyen	36	$ 12,600

Case 2: No data matches the request

There are no statements in this year.

Function 7. Report uninsured cars – 75 LOC

Description:

This function allows users to display a list of vehicles without insurance contracts in the system.

Operation Workflow:

1. **Create a list of uninsured vehicles**:
    - Retrieve current vehicle list information from the system, check license plate number in insurance statement to create a list of uninsured vehicles.
2. **Display Data:**
    - If the list contains entries:
        - Display the following details for each vehicle in a formatted table or list, , sorted by "***Value of vehicle***" descending:
            - License plate

        - Registration Date
        - Vehicle Owner
        - Brand
        - Number of seats
        - Value of vehicle
    - If the registration list is empty:
        - Display the message: **" No information available "**
3. **Return to Main Menu:**
    - After handling, return to the main menu.

Sample Output:

Case 1: List Contains Data

Report: UNINSURED CARS

Sorted by : Vehicle type
Sort type : DESC

No.	License plate	Registration Date	Vehicle Owner	Brand	Number of seats	Value of vehicle
1	51H2-370.11	11/15/2022	Tuan Nguyen	Ford	7	$ 41,000
2	50H3-612.00	02/26/2023	John Smith	Mitsubishi	7	$ 38,500
3	51H5-011.12	01/30/2023	Martin Vu	Honda	5	$ 21,800
4	50H1-412.33	10/01/2022	Hoa Tran	Toyota	5	$ 16,500

Function 8. Save data – 50 LOC

Description:
This function saves all the registration data to a file in object format. This enables persistent storage and easy retrieval of vehicle registration records and insurance statements. The data should be stored in a structured format that preserves the relationships between vehicle details and insurance statements.

Operation Workflow:

1. **Data Collection**:
    - Gather all current registration data from the program, including vehicle details (*License plate, Car owner, Phone Number, Car brand, Registration Date, ...*), insurance statement (*Insurance id, The established date, License plate, Customer name, Insurance period, and Insurance fees*)
2. **Serialization**:
    - Convert the registration data into an object format suitable for file storage as a binary object file.
3. **Save to File**:
    - Write the serialized data to a file. The file should be named appropriately (e.g., ***carInfo.dat, insurances.dat***).

4. **Confirmation Message**:
    - Display a confirmation message once the data is successfully saved.
5. **Return to Main Menu**:
    - After saving the data, prompt the user to return to the main menu.

Function 9. Quit – 50 LOC

Description:
This function exits the program. The program will only exit when the user presses the designated exit key (9). If any other function key is pressed, a message will be displayed indicating that the function is not available. If the user chooses to exit without saving the data, a reminder will prompt the user to save the information before exiting.

Operation Workflow:

1. **User Input**:
    - Display a menu or prompt where the user can select a function key.
    - If the user selects function keys other than those listed on the menu:
        - Display the message: **"This function is not available."**
2. **Exit Confirmation:**
    - If the user selects the exit key (9):
        - If there are unsaved changes:
            - Prompt the user with the message:
              "Do you want to save the changes before exiting? (Y/N)"
                - If the user responds with **'Y'**:
                    - Call the function to save the data to the file.
                - If the user responds with **'N'**:
                    - Exit the program directly.
        - If there are no changes to save, exit the program immediately.
3. **Return to Main Menu:**
    - If the user selects a function key other than the exit key:
        - Display the message: **"This function is not available."** and remain in the program.
4. **Reminder Message:**
    - If the user exits without saving data, display a reminder message:

      You have unsaved changes. Are you sure you want to exit without saving? (Y/N)

The above specifications **provide basic information**. You are required to ***conduct a detailed requirements analysis and build the application based on the real-world requirements***.

The lecturer will explain the **full set of requirements only once during the initial slot of the assignment**.