Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

Workshop 2

Due Date: Feb 09, 2024

INSTRUCTIONS

- *This workshop must be completed individually without any outside collaboration. All work must be your own. Copying or reproducing the work done by others (in part or in full) or letting others to copy or reproduce your own work is subject to significant grade reduction or getting no grade at all and/or being treated as academic dishonesty under the College's Academic Dishonesty Policy.*

- *Your goal is to finish the design part during the lab time, unless otherwise specified by your instructor.*

- *The backend coding for your design can be submitted as a part of DIY.*

- *Your application must compile and run upon download to receive any mark.*

- *To submit the workshop, please follow the Submission Guideline provided at the end of this document.*

- *You must submit your workshop by the due date. Late submissions policy is specified in the Academic Procedures for Evaluations document available through the class plan on Blackboard.*

Description:

The third workshop allows students to design and develop a Vehicle Maintenance and Usage Management System (VMUMS). This desktop application helps the client manage vehicle information, maintenance schedules, and usage logs efficiently.

Note: For designing the front end, you can see example in Week 5 lectures like grouping and container.

Task:

You are being hired by a company specializing in providing software solutions to vehicle rental and fleet management businesses. Your job is to design and deliver to one of their clients a desktop application that will:

1. Record vehicle details (e.g., model, make, year, type).
2. Log maintenance records (e.g., date, description, cost).
3. Track vehicle usage (e.g., start date, end date, kilometers driven).
4. Provide a summary of saved data based on user choice (e.g., Vehicle Details, Maintenance Records, Usage Logs).

(Consider I am your client, and you must show me your designs (Front end) by Wednesday lab February 5th.)

Requirements:

1. **Follow the MVC Design Pattern**
    - Create model classes for Vehicle, MaintenanceRecord, and UsageLog.
    - Use appropriate encapsulation and abstraction techniques.
2. **Vehicle Details**
    - Fields: Model, Make, Year, Type (e.g., Sedan, SUV, Truck).
    - Type selection can be a ChoiceBox or ComboBox.
3. **Maintenance Records**
    - Fields: Date, Description, Cost.
    - Input via a dedicated section or popup form.
4. **Usage Logs**
    - Fields: Start Date, End Date, Kilometers Driven.
    - Input via a dedicated section or popup form.
5. **User Interface Elements**
    - TextFields for vehicle details.
    - DatePicker for date fields.
    - Buttons for actions: Clear, Save.

- A TableView to display records.

6. **Event Handlers:**
   - **Clear** button should clear all the fields without saving the data.
   - **Save** button should store the entered data in appropriate data structures.
   - **Show/View Summary** button should pop up a window displaying all saved records based on the user's choice (e.g., show only vehicles, maintenance, or usage data).
7. **Summary Display:**
   - Use a pop-up window (new Stage) or a new Scene to display the summary.
   - Include a ChoiceBox or ComboBox for the user to select the type of data to view.
   - Display data in a formatted TextArea or a TableView.
   - Add a **Summary Button** to the main application interface labeled "View Summary."
   - When clicked, a new window should open, allowing the user to select the type of data to view (Vehicle Details, Maintenance Records, Usage Logs) via a dropdown.
   - Display the selected data in a TextArea or a TableView in the new window.
   - Example of the Summary Window:
     1. **Dropdown Options:** "Vehicles," "Maintenance Records," "Usage Logs."
     2. **Summary Display Area:** TextArea/TableView.
8. **Data Storage:**
   - Use a data structure of your choice (e.g., ArrayList, Map, LinkedList) to store vehicle, maintenance, and usage data.
   - Be prepared to discuss your choice in terms of performance and usage scenarios.

- Discuss during the lab days about the data structure you want to choose.
- You are required to follow the java naming conventions for classes and member variables. Your solution should be designed to follow the OO-design concepts using encapsulation, abstraction etc.

Workshop Header

/***
Workshop #
Course:*<subject type> - Semester*
Last Name:*<student last name>*
First Name:*<student first name>*
ID:*<student ID>*
Section:*<section name>*
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:*<submission date>*
***/

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Do Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- Design should be discussed/ created during the lab and show. 20%.
- Complete the code behind as the part of your DIY. 50%
- Submit a `reflect.txt` file with the following questions answered: 15%
  - How did you ensure scalability in your application design? Provide examples.
  - What challenges did you face during data handling, and how did you overcome them?
  - How does your solution balance usability and technical efficiency? Reflect on specific choices.
- Video submission explaining core code pointers and showing the full working application (3 – 8 minutes max). 15%
- All submission goes to Black Board.
- Your submission should include
  - Video file with audio
  - Reflect.txt file
  - Complete zipped project.

- Late submissions would result in additional 10% penalties for each day or part of it.

Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the workshops, but the final solution may not be copied from any source.