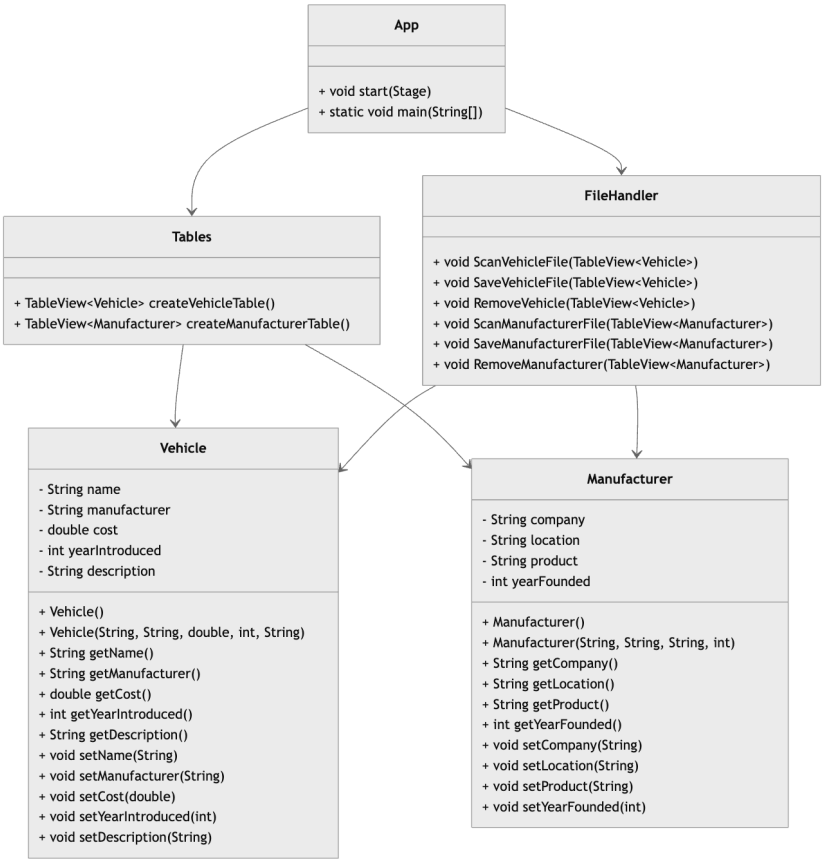
Implementation Manual

I. Project Overview

- This application catalogs military vehicles and their manufacturers into a JavaFX GUI. Users can add and remove information using labeled buttons and textfields. Changes are saved to text files in CSV format.

II. UML Diagram



- Demonstrates a modular, object-oriented design. The classes can be grouped into the following three categories:
    1. **GUI**
        - App.java : handles layout, event handling, and application launch.
    2. **Utility**
        - Tables.java : constructs the tables and their columns.
        - FileHandling.java : Handles saving/loading from text files.
    3. **Data**
        - Vehicle.java : Custom class representing a military vehicle.
        - Manufacturer.java : Custom class representing a manufacturer.

III. Class Descriptions

`App.java`

- Entry point for the application:
    - Sets up the GUI layout - labels, fields, buttons, and tables.
    - Connects button event handlers for adding, removing, saving entries.
    - Launches the GUI initializing Tables and loading data from FileHandler

`FileHandler.java`

- Handles file input/output using .txt CSV files.
    - Loads data into tables from VehicleInfo.txt and ManufacturerInfo.txt.
    - Saves table content, writing data back to .txt files.
    - Provides remove functionality based on selected table row.

`Tables.java`

- Constructs TableView instances for both vehicles and manufacturers.
    - Defines table columns for class attributes like Name, location, cost, etc.
    - Links columns to objects' fields.
    - Returns populated TableView to be used in the GUI.

`Vehicle.java`

- A class representing military vehicle objects.
    - Private attributes: name, manufacturer, cost, year introduced, description.
    - Constructor is called when adding a vehicle to the vehicle table.
    - Getters and setters for each attribute.

`Manufacturer.java`

- Represents a defense manufacturer:
    - Private attributes: company, location, product, and founding year.
    - Getters and setters for each attribute.
    - Constructor is called when adding a manufacturer to the company table.

IV. Data Storage Format

- **VehicleInfo.txt**
    - Format: `name,manufacturer,cost,year`
- **ManufacturerInfo.txt**
    - Format: `company,location,product,yearFounded`

V. Future Improvements

The following areas stand out as opportunities for further improvement in the application.

- Input validation:
    - Verify input fields match with the expected data type.
- Exception handling:
    - Error handling is minimal, additional exception handling could provide more robustness to the application.
- Editing existing entries
    - Currently, the user is only able to add and remove entire rows. Editing typo's or updating a piece of information is a cumbersome process. Editing individual cells would drastically improve user experience.

Appendix

- Java version: 11
- JavaFX version: 20
- User Interface: