

Military Vehicle Database

CSI 2300 - Object Oriented Programming

Brandon Praet

April, 21, 2025

Agenda

1. Guiding Principles
2. UI
3. UML
4. Implementation
5. Reflections
6. Demo / Questions

My Background:

- Post-bachelor's student with a B.S. in Mechanical Engineering.
- Experience in automotive regulatory compliance
- Interested in Computer Science, advanced mobility, and intelligent systems.

Guiding Principles

Main Course Objectives:

- Build a Java program using object-oriented principles
- Apply UML class diagrams
- Design a GUI with JavaFX



Personal Goals:

- Explore career opportunities in a new industry
- Build and showcase software development skills



Problem Statement:

Develop a Military Vehicle Database to organize data on local defense manufacturers and their products.

Vehicle Table:

Name	Manufacturer	Cost	Year Introduced	Description	
M1Abrams	General Dynamics Land Systems	10.66	1980	Main battle tank	<div></div>
Bradley Fighting Vehicle	BAE Systems	4.35	1981	Infantry fighting vehicle	
ACV	BAE Systems	5.2	2020	Amphibious combat vehicle	
MaxxPro	Navistar Defense	0.73	2007	Mine-resistant vehicle	
Saratoga	Navistar Defense	0.27	2011	Light armored truck	
Humvee NXT 360	AM General	0.29	2022	Modern tactical Humvee	
Joint Light Tactical Vehicle	AM General	0.4	2020	Next-gen utility vehicle	

Name: Manufacturer: Cost: Year: Description

Add Vehicle

Save

Remove Vehicle

Company Table:

Company	Location	Product/Service	Year Founded	
General Dynamics Land Systems	Sterling Heights	Armored Combat Vehicles	1952	
BAE Systems	Sterling Heights	Combat Vehicles	1899	
Navistar Defense	Madison Heights	Military Trucks	1902	
Detroit Arsenal	Warren	Research & Development	1940	
AM General	Auburn Hills	Light Tactical Vehicles	1903	
Magna International	Troy	Defense Electronics	1957	

Company: Location: Product/Service: Founded:

Add Company

Save

Remove Company

Data Management UI

Military Vehicle Database

Vehicle Table:

Name	Manufacturer	Cost	Year Introduced	Description
M1Abrams	General Dynamics Land Systems	10.66	1980	Main battle tank
Bradley Fighting Vehicle	BAE Systems	4.35	1981	Infantry fighting vehicle
ACV	BAE Systems	5.2	2020	Amphibious combat vehicle
MaxxPro	Navistar Defense	0.73	2007	Mine-resistant vehicle
Saratoga	Navistar Defense	0.27	2011	Light armored truck
Humvee NXT 360	AM General	0.29	2022	Modern tactical Humvee
Joint Light Tactical Vehicle	AM General	0.4	2020	Next-gen utility vehicle

Name:

Manufacturer:

Cost:

Year:

Description

Add Vehicle

Save

Remove Vehicle

Company Table:

Company	Location	Product/Service	Year Founded
General Dynamics Land Systems	Sterling Heights	Armored Combat Vehicles	1952
BAE Systems	Sterling Heights	Combat Vehicles	1899
Navistar Defense	Madison Heights	Military Trucks	1902
Detroit Arsenal	Warren	Research & Development	1940
AM General	Auburn Hills	Light Tactical Vehicles	1903
Magna International	Troy	Defense Electronics	1957

Company:

Location:

Product/Service:

Founded:

Add Company

Save

Remove Company

The UI is divided into two sections for structured data management:

Vehicle Table:

Field	Description:
Name	Common name of the vehicle
Manufacturer	Company that produces the vehicle
Cost	Estimated unit cost (in million USD)
Year Introduced	Year the vehicle entered service
Description	Vehicle type or operational role

Company Table:

Field	Description:
Company	Full name of the manufacturer or contractor
Location	City of primary operations or major facility
Product/Service	Main product or service provided
Year Founded	Year the company was established

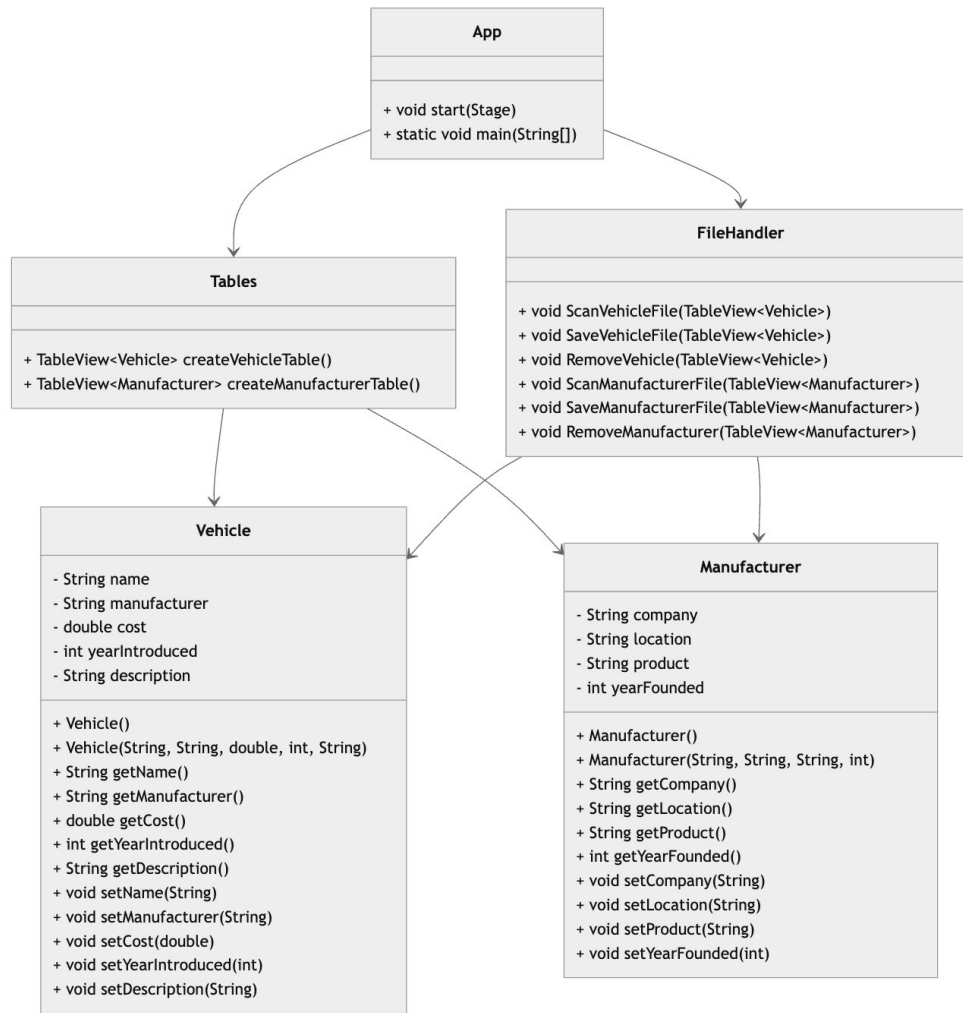
Button Functions

Name:	<input type="text" value="Humvee NXT 360"/>	Manufacturer:	<input type="text" value="AM General"/>	Cost:	<input type="text" value=".29"/>	Year:	<input type="text" value="2022"/>	Description	<input type="text" value="Modern tactical Humvee"/>
<div><input type="button" value="Add Vehicle"/> <input type="button" value="Save"/> <input type="button" value="Remove Vehicle"/></div>									

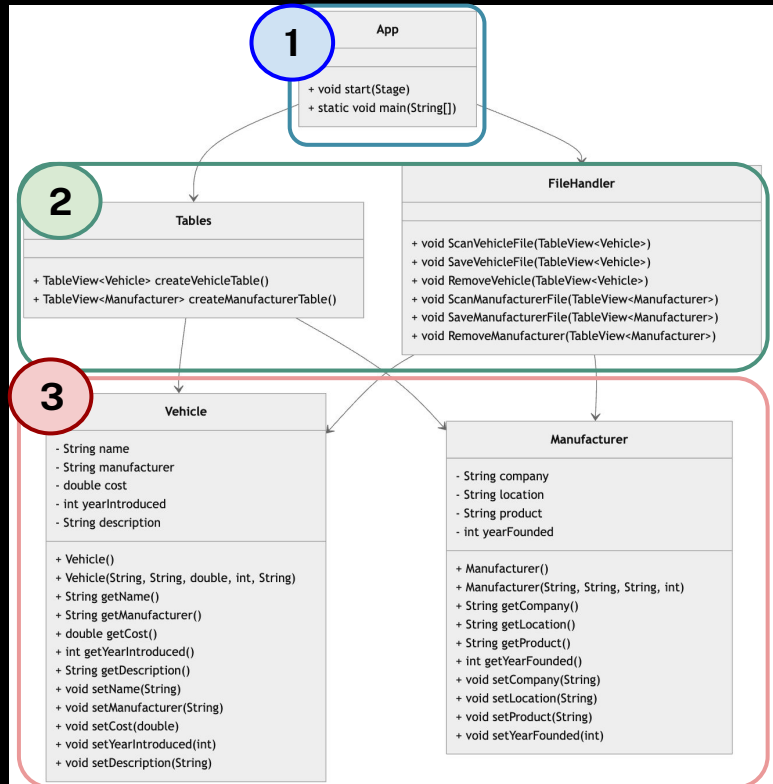
Company:	<input type="text" value="AM General"/>	Location:	<input type="text" value="Auburn Hills"/>	Product/Service:	<input type="text" value="Light Tactical Vehicles"/>	Founded:	<input type="text" value="1903"/>
<div><input type="button" value="Add Company"/> <input type="button" value="Save"/> <input type="button" value="Remove Company"/></div>							

1. **Add:** Enter details in the text fields, then click “Add Vehicle” or “Add Company” to submit the entry.
2. **Save:** Click “Save” to write all current table entries to the text file.
3. **Remove:** Select a row, then click “Remove Vehicle” or “Remove Company” to delete the entry.

UML



Class Breakdown



Class Categories:

1. GUI

- `App.java` : Manages layout, event handling, and application launch.

2. Utility

- `Tables.java` : builds and returns tables using `TableView` class.
- `FileHandling.java` : Handles loading and saving data to text files.

3. Data

- `Vehicle.java` : Represents a military vehicle entry.
- `Manufacturer.java` : Represents a company or contractor.

Implementation

Class: Tables.java

```
public static TableView<Vehicle> createVehicleTable(){  
    // Create a TableView  
    TableView<Vehicle> table = new TableView<>();  
  
    // Create columns and cell factory for each  
    TableColumn nameColumn = new TableColumn<Vehicle,String>("Name");  
    nameColumn.setCellValueFactory(new PropertyValueFactory<Vehicle,String>("name"));  
  
    TableColumn manufacturerColumn = new TableColumn<Vehicle,String>("Manufacturer");  
    manufacturerColumn.setCellValueFactory(new PropertyValueFactory<Vehicle,String>("manufacturer"));  
  
    TableColumn costColumn = new TableColumn<Vehicle,String>("Cost");  
    costColumn.setCellValueFactory(new PropertyValueFactory<Vehicle,String>("cost"));  
  
    TableColumn<Vehicle, Integer> yearColumn = new TableColumn<>("Year Introduced");  
    yearColumn.setCellValueFactory(new PropertyValueFactory<>("yearIntroduced"));  
  
    TableColumn<Vehicle, String> descriptionColumn = new TableColumn<>("Description");  
    descriptionColumn.setCellValueFactory(new PropertyValueFactory<>("description"));  
  
    // Add columns to table  
    table.getColumns().addAll(nameColumn, manufacturerColumn, costColumn, yearColumn, descriptionColumn);  
  
    return table;  
}
```

Constructs `TableView` instances for vehicle and company data.

- Defines table columns for object fields like name, cost, and year.
- **Connects columns to class fields** using 'setCellValueFactory'.
- Returns the fully configured 'TableView' for use in the UI.

Implementation

Class: FileHandler.java

```
// Scanner method to populate the table with info from the VehicleInfo.txt file
public static void ScanVehicleFile( TableView<Vehicle> table) {
    File infoFile = new File("VehicleInfo.txt");    // Create a file object for the text file
    try {
        Scanner scanner = new Scanner(infoFile);    // Scan from that file
        while (scanner.hasNextLine()) {              // While there is info in the file
            String line = scanner.nextLine();        // Scan line by line
            String[] vehicleEntry = line.split(","); // Create vehicleEntry array by splitting the line
            String name = vehicleEntry[0];           // Takes the 0th index of array
            String manufacturer = vehicleEntry[1];   // Takes the 1st index of array
            double cost = Double.parseDouble(vehicleEntry[2]); // Takes the 2nd index of array
            int year = Integer.parseInt(vehicleEntry[3]); // Takes the 3rd index of array
            String description = vehicleEntry[4];    // Takes the 4th index of array
            //***** Can add more entries here later *****

            Vehicle vehicle = new Vehicle(name, manufacturer, cost, year, description);
            table.getItems().add(vehicle);           // and adds it to the table
        }
        scanner.close();                            // Close the scanner to prevent resource leak
    } catch (FileNotFoundException ex) {             // Catch portion of our try-catch block
        ex.printStackTrace();
    }
}
```

(Not pictured: SaveVehicleFile() and RemoveVehicle())

Handles file input/output using simple `.txt` files with CSV formatting.

- Loads data into tables from VehicleInfo.txt and ManufacturerInfo.txt.
- Saves table data back to the same files.
- Removes entries based on selected row in the table.

Enables saving and loading data across sessions!

Reflections

What Went Well

- First time designing and building a complete java program.
- Gained hands-on experience with GUI design and file I/O.

What I Learned

- A clear initial UML diagram helps prevent design issues.
- Simple, consistent commenting makes code easier to revisit.

What I'd Improve

- Implement in-table editing.
- Use inheritance for vehicle subtypes (e.g. GroundVehicle -> LightTransport)
- Upgrade from text files to a more advanced database.

Demo + Questions