Andrew Hines

3.7 Class Implementation

09.10.2022

**Application Description:**

I would like to create an application that acts as a simple DMS (Dealer Management Software). It would allow the user to add vehicles to the database or sell them. The user may also update the vehicle properties if mistakes were made, or changes are required, such as the milage or color. The user may also keep track of their profit in total and on an individual level per vehicle.

**Purpose:**

DMSs are a handy tool for dealers to keep track of their inventory, finances, customers, etc. The program will be the same, acting as a simple, organizational tool that will also keep track of inventory and finances. Handwriting these records can be tedious and time-consuming. The program will add value by speeding up this process and keeping the user organized.

**Output:**

The application would start with a menu using a case statement. The user may choose “Add to Inventory, Sell Vehicle, Browse Inventory, Banking, and Exit. Selecting Add to Inventory would allow the user to add a vehicle into the database after inputting properties about it such as make, model, year, color, milage, purchase price, and selling price. Sell Vehicle would remove a selected vehicle and calculate the profit made from the deal. The Banking option would allow the user to see how much profit they have made in total from selling vehicles and individually. Browse Inventory would allow the user to see the properties of each Vehicle object, as well as update them. Exit will terminate the program.

**Data Storage:**

The program will store the vehicle objects in an array. In turn, it also stores all the properties of the vehicle such as make, model, year, color, selling price, and purchase price. The program will also store the total profit made off each sold vehicle.

**Class Menu:**

This is the base class that will allow users to navigate to different menus depending on user input.

**Properties:**

-mainInput int

-vehicleInput int

-bankInput int

These inputs will be checked with a series of IF statements to determine which menu to display in the console.

If (mainInput == 1) {

DisplayVehicleMenu()

}

Else If(mainInput == 2) {

DisplayBankMenu()

}

Etc.

**Example:** (Main Menu)

-Dealer Management Software-

Please select an option:

1: Inventory (Inputting “1” will call method DisplayVehicleMenu() )

2: Banking (Inputting “2” will call method DisplayBankMenu() )

3: Exit (Terminates Program)

=>

(Bank Menu)

-Finances-

1: Total Profit (Will display total profit made from all sales in soldInventory array using ToString() )

2: View Profit from Sold Inventory (iterates through soldInventory array showing profit made from individual sales using ToString)

3. Back (Navigates to Main Menu using DisplayMainMenu() method)

**Methods:**

+DisplayMainMenu()

+ DisplayVehicleMenu()

+DisplayBankMenu()

**Class Vehicle : Menu [Inheritance]**

Child class of Menu.

**Properties:**

* (inherits) vehicleUserInput int (Used to navigate sub-menu for Vehicles)

**+** vehicleInventory[]

+ soldVehicleInventory[]

* id int
* make string
* model string
* year string
* color string
* milage string
* purchasePrice int
* sellingPrice int
* profit int

**Methods:**

+ AddVehicle() (Prompts user to input properties and pushes new vehicle object to vehicleInventory array() )

+SellVehicle() (Prompts user for vehicle ID, calculates profit, adds that value to variable totalProfit, and adds vehicle object to soldVehicleInventory array)

+UpdateVehicle() (checks user input for variable id and allows them to change property values of that vehicle

+ViewInventory() (Iterates through vehicleInventory array)

+ (inherits)DisplayVehicleMenu()

+ (inherits)DisplayMainMenu()

**Class Bank : Vehicle [Inheritance]**

Child class of Vehicle

**Properties:**

* (inherits) bankUserInput int (Used to navigate through sub-menu for Bank)
* totalProfit int
* (inherits) soldInventory[]

**Methods:**

+ ViewTotalProfit() (displays total profit of all sales using ToString() then displays Bank Menu)

+ ViewIndProfit() (Iterates though soldVehicle array displaying all properties including profit made from it’s sale using ToString() then Bank Menu)

+ (inherits)DisplayBankMenu()

+ (inherits)DipsplayMainMenu()

**Data Storage:**

**-vehicleInventory-**

Vehicle:

* id int (001)
* make string (“Ford”)
* model string (“Mustang”)
* year string (“2002”)
* color string (“Yellow”)
* milage string (“95000”)
* purchasePrice int (5000)
* sellingPrice int (6500)
* profit int (1500)

Vehicle2:

* id int (002)
* make string (“Jeep”)
* model string (“Wrangler”)
* year string (“2007”)
* color string (“Yellow”)
* milage string (“185000”)
* purchasePrice int (8000)
* sellingPrice int (11500)
* profit int (3500)

Vehicle3:

* id int (003)
* make string (“Dodge”)
* model string (“Charger”)
* year string (“2008”)
* color string (“White”)
* milage string (“143000”)
* purchasePrice int (5000)
* sellingPrice int (7500)
* profit int (2500)

**-soldVehicleInventory-**

Vehicle4:

* id int (004)
* make string (“Jeep”)
* model string (“Wrangler”)
* year string (“1992”)
* color string (“Black”)
* milage string (“182000”)
* purchasePrice int (3000)
* sellingPrice int (6000)
* profit int (3000)