PROIECT PIU

ADMINISTRARE PARC AUTO

**Tecleanu Ionut-Dumitru**

**3124b**

# **1)Prezentare generala**

Aplicatia respective are ca si scop managerarea unui parc auto, in care avem 2 tipuri de utilizatori. Administartori, care pot adauga, sterge sau schimba annunturi, iar Clientul poate cumpara listari.

Aceasta aplicatie are scop, eliminarea necesitaii de a vinde autovehicule in parcuri auto. Drept rezultat, se vor reduce semnificativ coturile pentru perosnaele dornice de a-si vinde autoturismele si pentru clientii dornici de un nou mod de transport.

# **2)Rolurile principale**

* Administartor

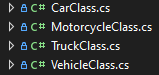
1. Acesta poate adauga listari in aplicatie, in scopul de a fi cumparate de catre client.
2. Utilizatorul se poate inregistra in aplicatie cu rolul de Administrator, doar daca cunoaste codul respective.
3. Se pot face modificari asupra oricarei listari, si anume stergerea sau modificarea acesteia.

* Client

1. Clientul poate cumpara diferite listari din baza de date si le poate vizualiza in aplicatie.
2. Orice utilizator se poate inregistra ca si Client.

# **3)Modele**

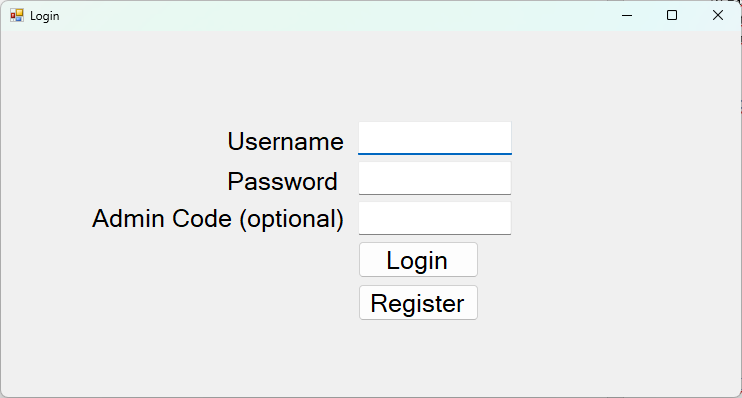
Aplicatia are 3 modele derivate dintr-o clasa de baza. Acesta sunt CarClass, MotorcycleClass si TruckClass care deriva din VehicleClass.

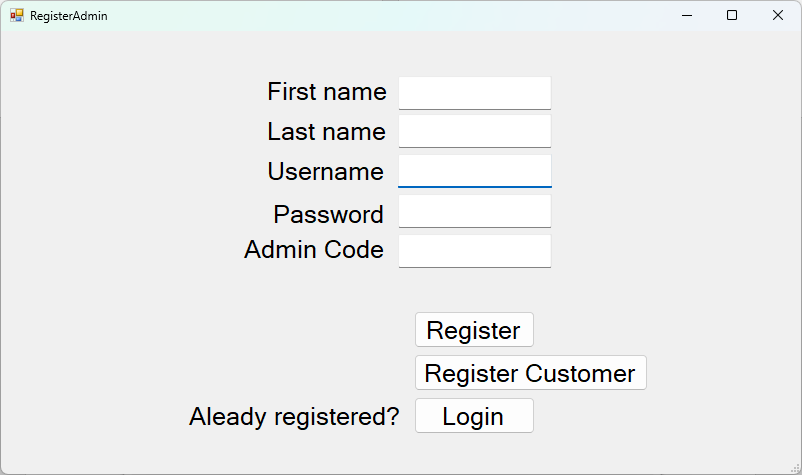


# **4)Prezentare UI**

1. ***Interfata LogIn***

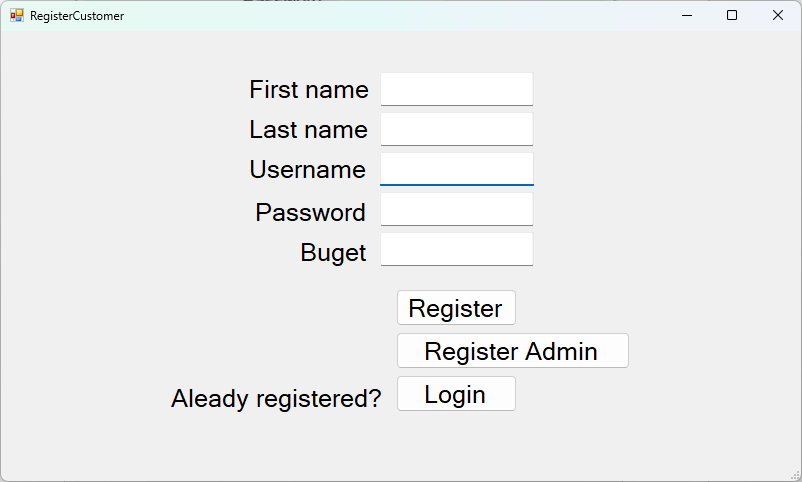
Aceasta interfata face posibili autentificarea in program a utilizatorului

1. ***Interfata RegisterAdmin***

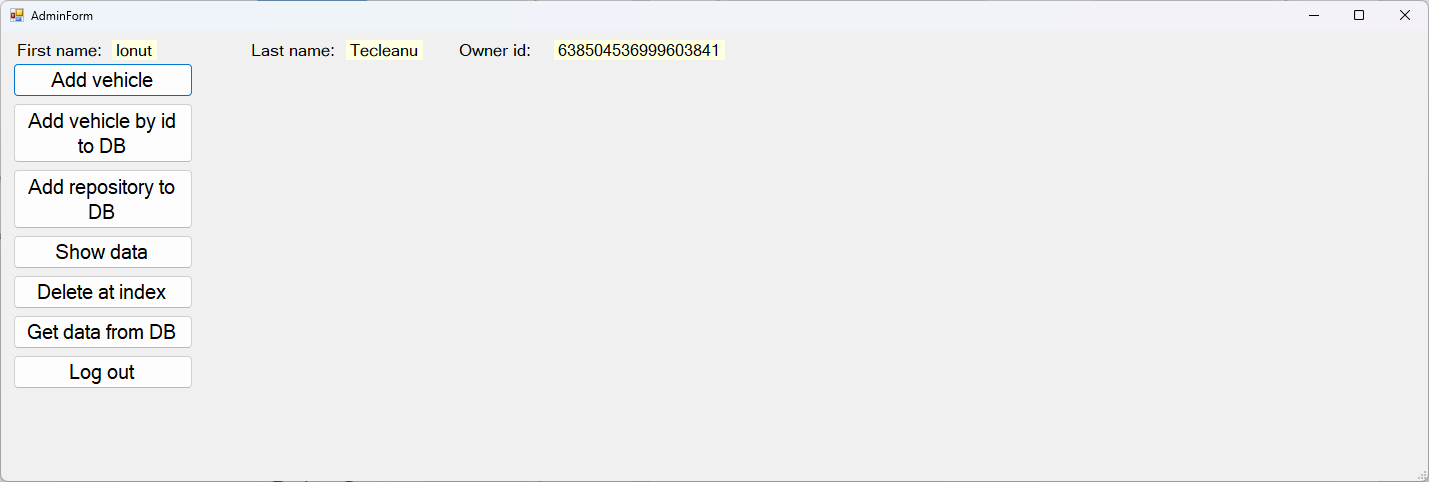
Interfata face posibili inregistrarea unui Administrator in aplicatie

1. **Interfata RegisterCutomer**

Interfata pentru inregistrarea Clientului

1. **Interfata AdminForm**

Interfata pentru operatiile destinate Administratorului

1. **Interfata CutomerForm**

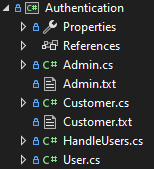
Interfata pentru operatiile destinate Clientului



# **5)Prezentare detaliata a claselor**

1. **Authentification**

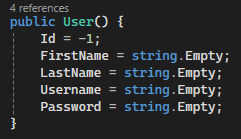
Clasele din aceasta sectiune au ca rol diferentierea dintre Administrator si Client. Acestea deriva dintr-o clasa de baza si anume User.

1. **Clasa User**

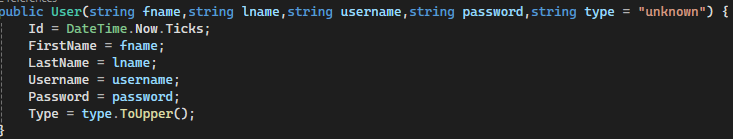
* **Constructori**

**User()**

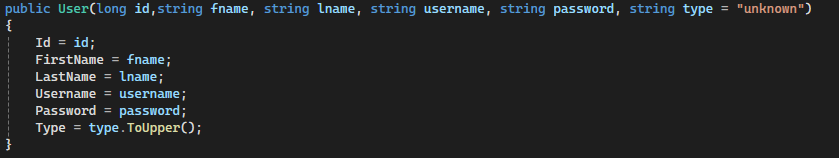
* Constructor implicit.
* Inițializează o nouă instanță a clasei User cu valori implicite.
* Id este setat la -1, iar FirstName, LastName, Username și Password sunt setate la șiruri de caractere goale.

****

**User(string fname, string lname, string username, string password, string type = "unknown")**

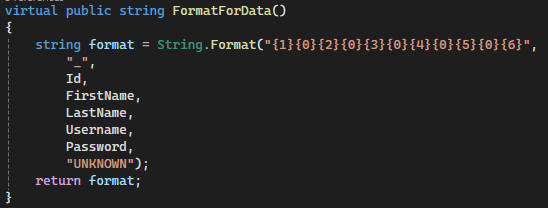
* Constructor cu parametri.
* nițializează o nouă instanță a clasei User cu prenumele, numele de familie, numele de utilizator, parola și tipul specificate.
* Id este setat la timestamp-ul curent în ticks.
* Type este convertit la majuscule pentru comparare mai usoara.

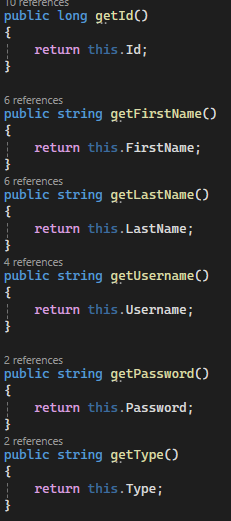
**User(long id, string fname, string lname, string username, string password, string type = "unknown")**

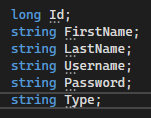
* Constructor cu parametri incluzând ID-ul.
* ****Inițializează o nouă instanță a clasei User cu ID-ul, prenumele, numele de familie, numele de utilizator, parola și tipul specificat.
* **Metode**

**FormatForData()**

* Returnează o reprezentare în format șir de caractere a datelor utilizatorului, potrivită pentru stocare.

****

* **Getteri**
* Metode pentru obținerea proprietăților utilizatorului.
* **Proprietati**

****

1. **Clasa Admin**

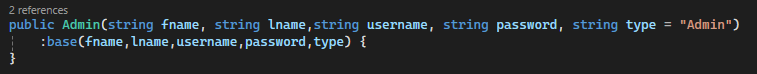
* **Constructori**

**Admin()**

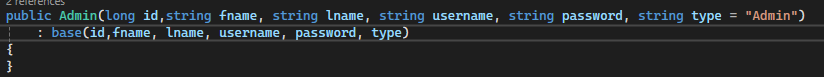
* Constructor implicit.
* Inițializează o nouă instanță a clasei Admin cu valori implicite.

****

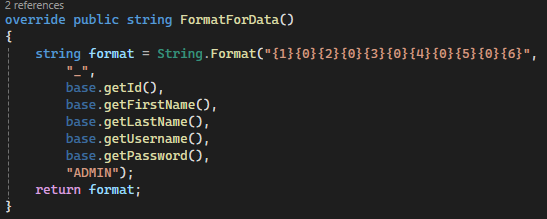
**Admin(string fname, string lname, string username, string password, string type = "Admin")**

* Constructor cu parametri.
* ****Inițializează o nouă instanță a clasei Admin cu prenumele, numele de familie, numele de utilizator, parola și tipul “Admin”.

**Admin(long id, string fname, string lname, string username, string password, string type = "Admin")**

* Constructor cu parametri incluzând ID-ul.
* ****nițializează o nouă instanță a clasei Admin cu ID-ul, prenumele, numele de familie, numele de utilizator, parola și tipul “Admin”.
* **Metode**

**FormatForData()**

* Suprascrie metoda FormatForData din clasa User.
* ****Returnează o reprezentare în format șir de caractere a datelor administratorului, potrivită pentru stocare.

**getCode()**

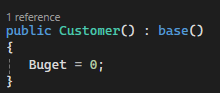
* Returnează un întreg care reprezintă codul administratorului.
* **Proprietati**

****

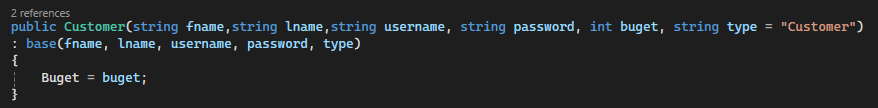
1. **Clasa Customer**

* **Constructori**

**Customer()**

* Constructor implicit.
* ****Inițializează o nouă instanță a clasei Customer cu valori implicite.

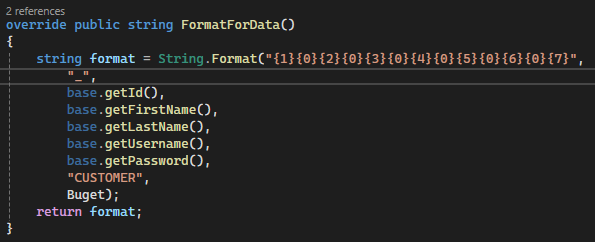
**Customer(string fname, string lname, string username, string password, int buget, string type = "Customer")**

* Constructor cu parametri.
* ****Inițializează o nouă instanță a clasei Customer cu prenumele, numele de familie, numele de utilizator, parola, bugetul și tipul “Customer”.

**Customer(long id, string fname, string lname, string username, string password, int buget, string type = "Customer")**

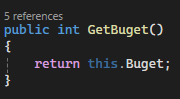
* Constructor cu parametri incluzând ID-ul.
* Inițializează o nouă instanță a clasei Customer cu ID-ul, prenumele, numele de familie, numele de utilizator, parola, bugetul și tipul “Customer”.
* **Metode**

**FormatForData()**

* Suprascrie metoda FormatForData din clasa User.
* ****Returnează o reprezentare în format șir de caractere a datelor clientului, potrivită pentru stocare.

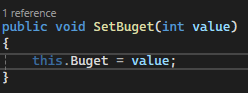
**GetBuget()**

* Obține bugetul clientului.



**SetBuget(int value)**

* Setează bugetul clientului.

****

* **Proprietati**

****

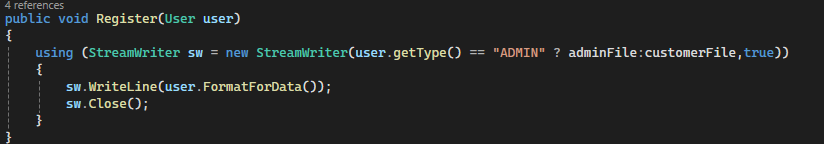
1. **Clasa HandleUsers**

* **Campuri**

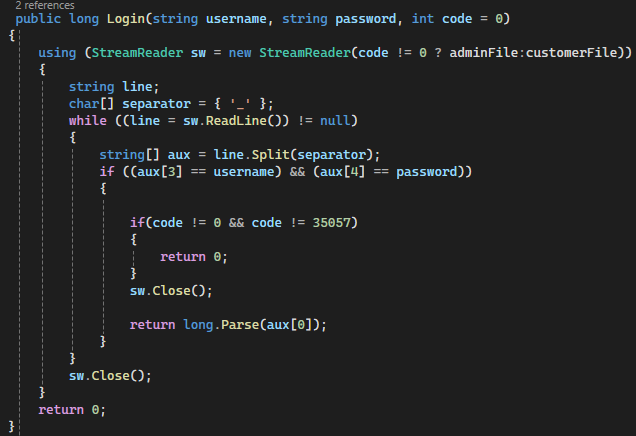
****

* Calea către fișierul unde sunt stocate datele administratorilor.
* Calea către fișierul unde sunt stocate datele clienților.
* **Metode**

**Register(User user)**

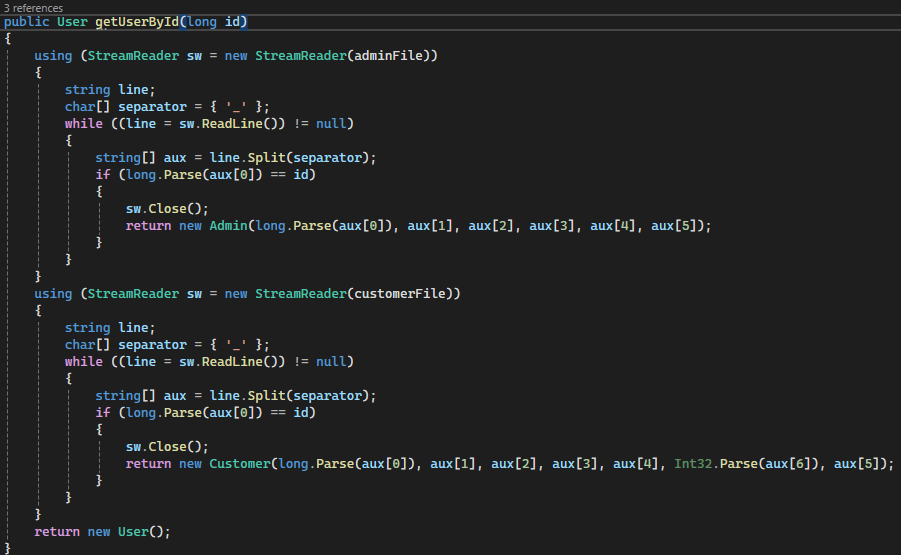
* Înregistrează un utilizator scriind datele acestuia în fișierul corespunzător
* ****Primește un obiect User ca parametru.

**Login(string username, string password, int code = 0)**

* Autentifică un utilizator verificând datele de autentificare în raport cu datele stocate.
* Primește username, password și un code opțional pentru administratori ca parametri.
* ****Returnează ID-ul utilizatorului dacă autentificarea este reușită; în caz contrar, returnează 0.

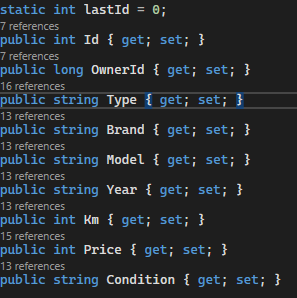
**getUserById(long id)**

* Obține un utilizator după ID.
* Primește id ca parametru.
* Returnează un obiect User.

****

1. **Clasa VehicleClass**

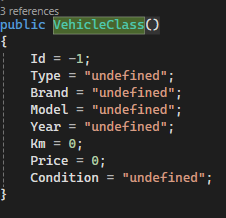
* **Proprietati**

****

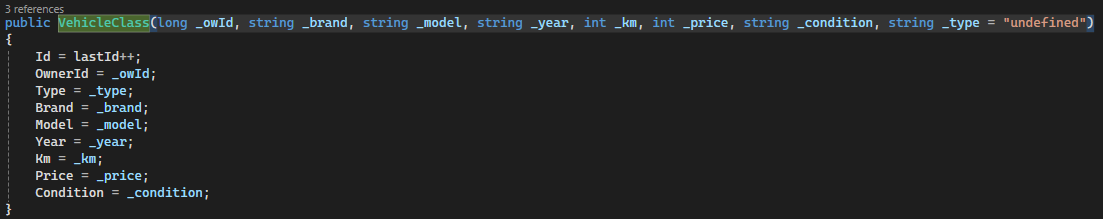
* **Constructori**

**VehicleClass()**

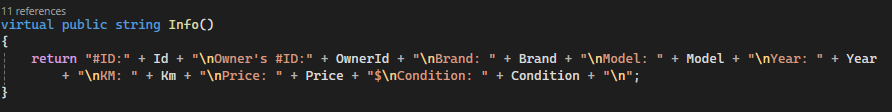
* Inițializează un obiect VehicleClass cu valori implicite.



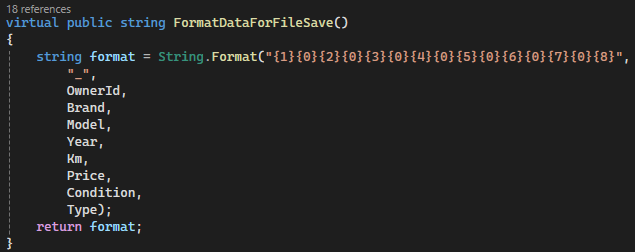
**VehicleClass(long \_owId, string \_brand, string \_model, string \_year, int \_km, int \_price, string \_condition, string \_type = "undefined")**

* Inițializează un obiect VehicleClass cu valori specificate.
* Metode

**Info()**

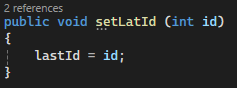
* Returnează informații despre vehicul sub formă de string.
* Returnează string

**FormatDataForFileSave()**

* Formatează datele vehiculului pentru a fi salvate într-un fișier.
* ****Returnează string

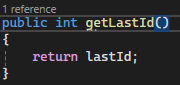
**setLatId(int id)**

* Setează ultima valoare a ID-ului.



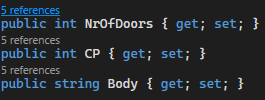
**getLastId()**

* Returnează ultima valoare a ID-ului.



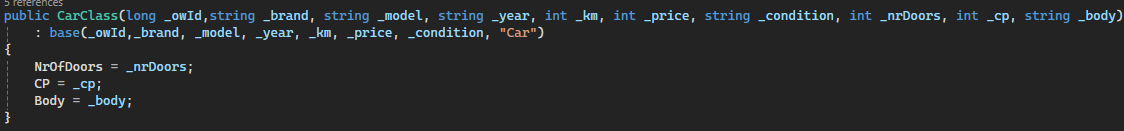
1. **Clasa CarClass**

* **Proprietati**

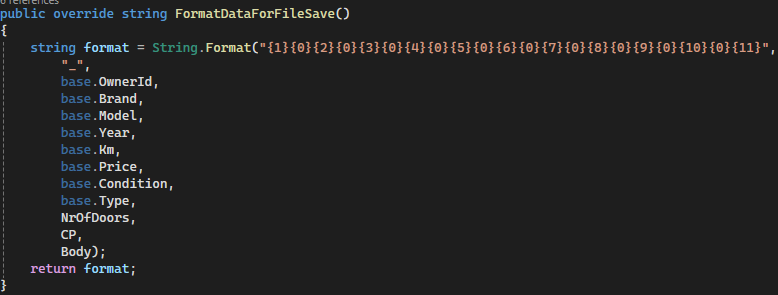
****

* **Constructori**

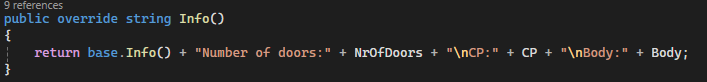
**CarClass(long \_owId, string \_brand, string \_model, string \_year ,int \_km, int \_price, string \_condition, int \_nrDoors, int \_cp, string \_body) : base(\_owId, \_brand, \_model, \_year, \_km, \_price, \_condition, "Car")**

* ****Inițializează un obiect CarClass cu valori specificate.
* **Metode**

**FormatDataForFileSave()**

* ****Suprascrie metoda FormatDataForFileSave din VehicleClass pentru a include informațiile specifice mașinii.

**Info()**

* ****Suprascrie metoda Info din VehicleClass pentru a include informațiile specifice mașinii.

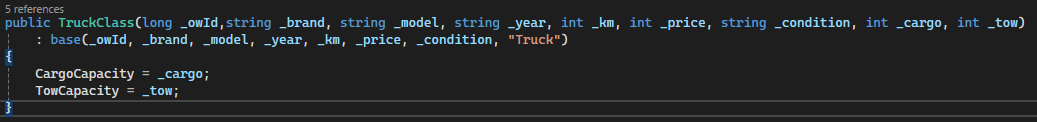
1. **Clasa TruckClass**

* **Proprietati**

****

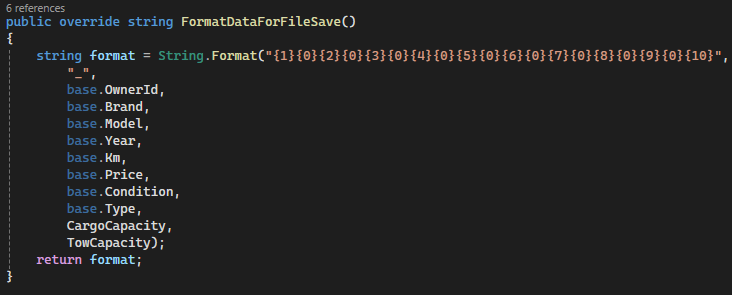
* **Constructori**

**TruckClass(long \_owId, string \_brand, string \_model, string\_year, int \_km, int \_price, string \_condition, int \_cargo, int \_tow) : base(\_owId, \_brand, \_model, \_year, \_km, \_price, \_condition, "Truck")**

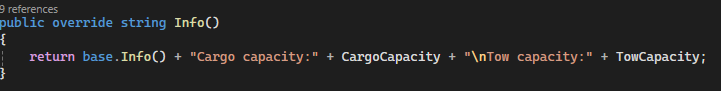
* ****Inițializează un obiect TruckClass cu valori specificate.
* **Metode**

**FormatDataForFileSave()**

* Suprascrie metoda FormatDataForFileSave din VehicleClass pentru a include informațiile specifice camionului.

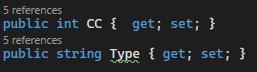
****

**Info()**

* ****Suprascrie metoda Info din VehicleClass pentru a include informațiile specifice camionului.

1. **Clasa MotorcycleClass**

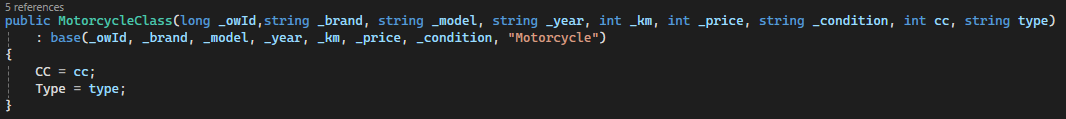
* **Proprietati**

****

* **Constructori**

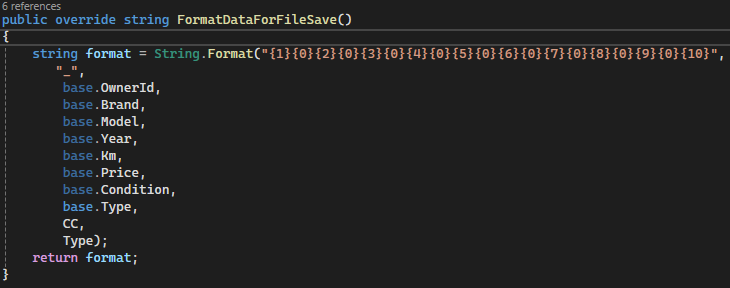
**MotorcycleClass(long \_owId, string \_brand, string \_model, string \_year, int \_km, int \_price, string \_condition, int cc, string type): base(\_owId, \_brand, \_model, \_year, \_km, \_price, \_condition, "Motorcycle")**

* Inițializează un obiect MotorcycleClass cu valori specificate.

****

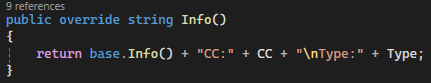
* **Metode**

**FormatDataForFileSave()**

* ****Suprascrie metoda FormatDataForFileSave din VehicleClass pentru a include informațiile specifice motocicletei.

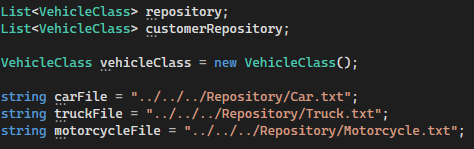
**Info()**

* Suprascrie metoda Info din VehicleClass pentru a include informațiile specifice motocicletei.

****

1. **Clasa RepositoryClass**

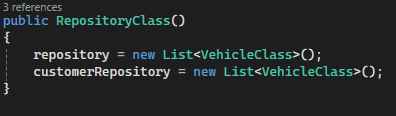
* **Proprietati**

****

* **Constructori**

**RepositoryClass()**

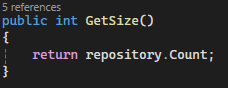
* Inițializează un obiect RepositoryClass, instantiind listele repository și customerRepository.

****

* **Metode**

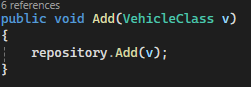
**GetSize()**

* Returnează numărul de vehicule din repository.

****

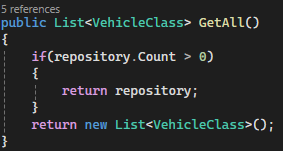
**Add(VehicleClass v)**

* Adaugă un vehicul în repository.

****

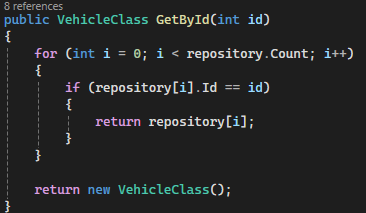
**GetAll()**

* Returnează toate vehiculele din repository.

****

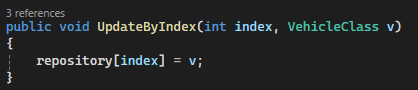
**GetById(int id)**

* Returnează un vehicul din repository pe baza unui ID.

****

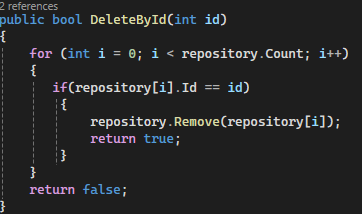
**UpdateByIndex(int index, VehicleClass v)**

* Actualizează un vehicul din repository după index.

****

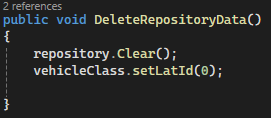
**DeleteById(int id)**

* Șterge un vehicul din repository pe baza unui ID dat.

****

**DeleteRepositoryData()**

* Șterge toate vehiculele din repository.

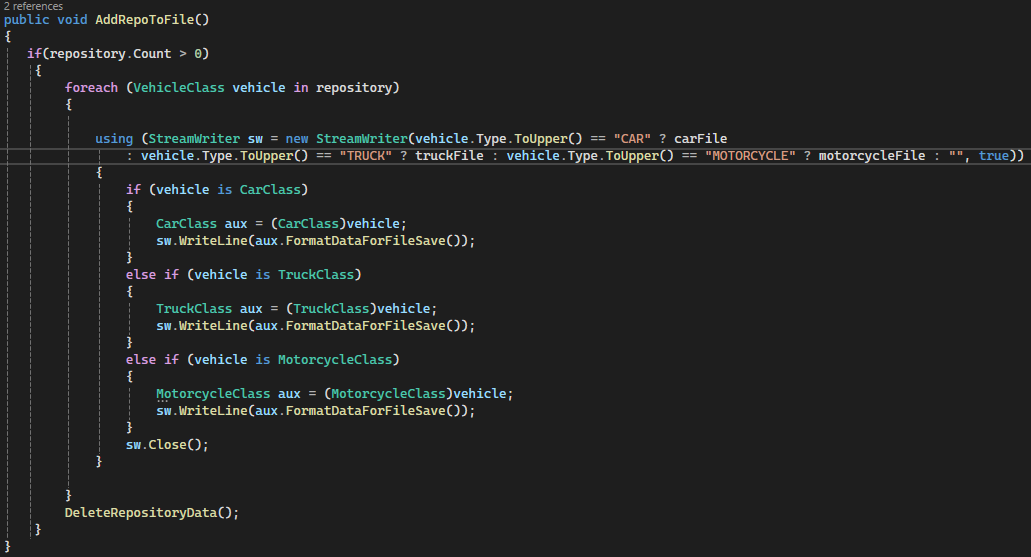
****

**AddByIdToFile(int id)**

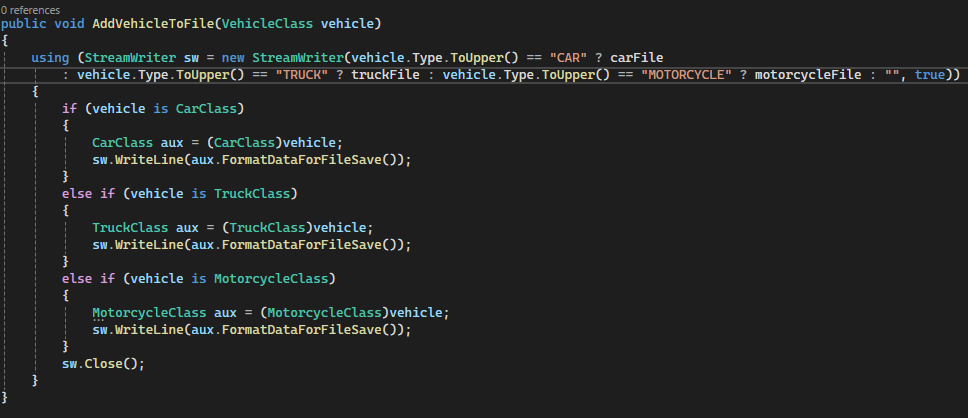
* ****Adaugă un vehicul într-un fișier pe baza unui ID dat.

**AddRepoToFile()**

* Adaugă toate vehiculele din repository în fișiere corespunzătoare și șterge apoi toate vehiculele din repository.

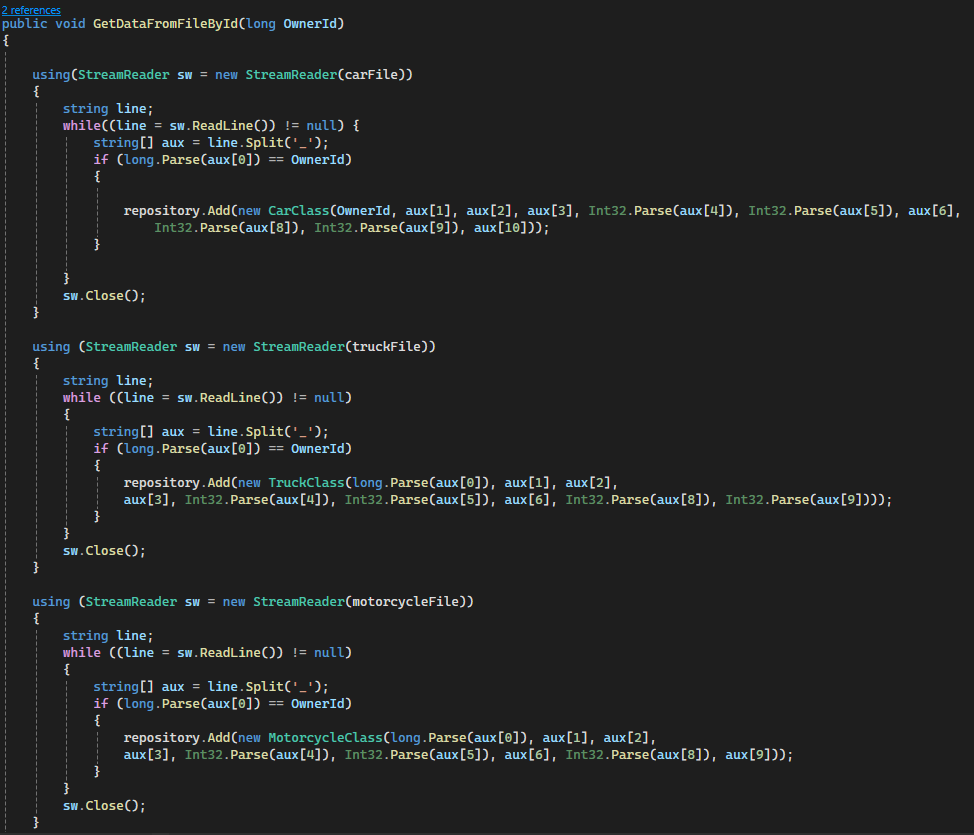
****

**AddVehicleToFile(VehicleClass vehicle)**

* ****Adaugă un vehicul in fisierul corespunzator.

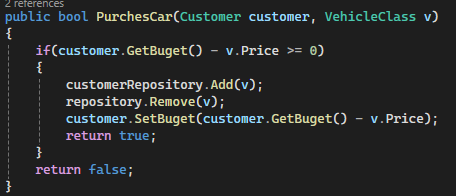
**GetDataFromFileById(long OwnerId)**

* Obține datele din fișiere pe baza unui Admin Id și le adaugă în repository.



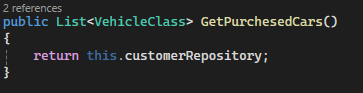
**PurchesCar(Customer customer, VehicleClass v)**

* Procesează achiziționarea unui vehicul de către un client.

****

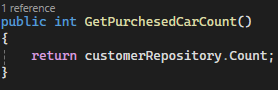
**GetPurchesedCars()**

* Returnează vehiculele achiziționate de clienți.

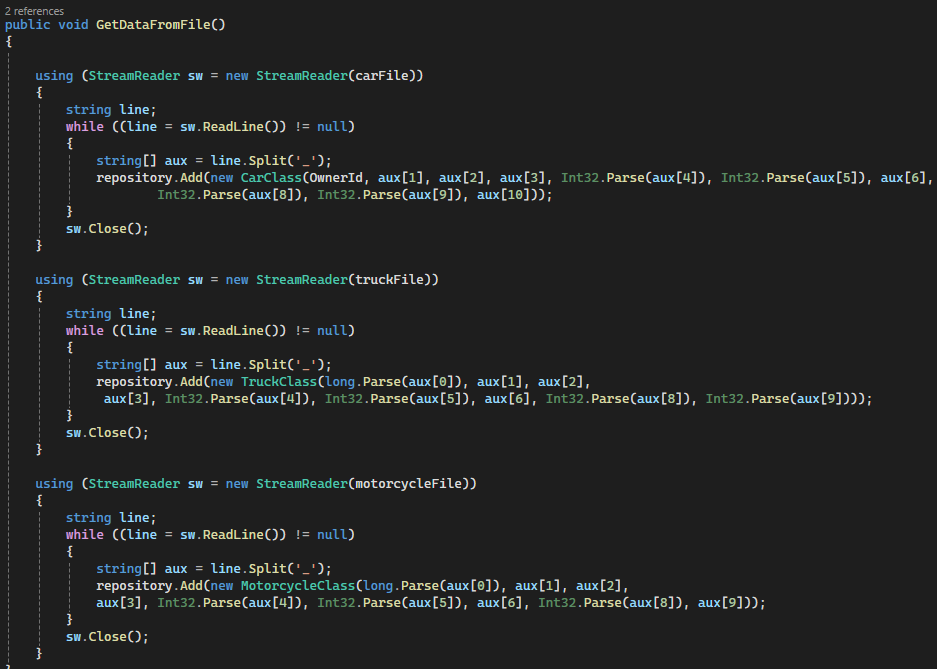
****

**GetPurchesedCarCount()**

* Returnează numărul de vehicule achiziționate de clienți.

****

**GetDataFromFile()**

* ****Obține masinile din fișiere și le adaugă în repository.