Contrôle Continu : POO (C++)

Projet : Gestion d'une flotte de véhicules

Modalités du Projet

1. Le projet est reçu par les étudiants le Mardi 26-11-2024 avant 18h.
2. Le projet est à renvoyer par mail au plus tard le Jeudi 05-12-2024 avant minuit.
3. La solution doit être envoyée par mail à : **mariame.amin@gmail.com**
4. L'objet de l'Email doit être « **Contrôle-POO-2024** » suivi par le « **nom »** de/des étudiant(s) et la « **classe »**.
5. Le projet est à travailler en **binôme** ou **monôme**.
6. L'évaluation du travail sera faite sur la base du **code source** et **rapport** envoyés par les étudiants tout en respectant le délai d'envoi ainsi que sur la **soutenance du projet**.
   a. Tout projet envoyé en retard ne sera pas pris en considération.
   b. Chaque code source qui ne se compile pas résultera à une note maximale de 8/20.
   c. Dans le cas où le code est fonctionnel, la note sera basée sur l'évaluation du professeur.

Cahier des charges

Projet : Gestion d'une flotte de véhicules

Objectif

Modéliser une flotte de véhicules pour une société, où chaque véhicule peut être suivi en termes d'informations générales (marque, modèle, kilométrage, disponibilité). Les employés peuvent réserver des véhicules pour usage professionnel.

Classes identifiées

1. Classe Véhicule

Représente un véhicule dans la flotte.

Attributs privés :

- std::string marque : la marque du véhicule (ex. : "Toyota").
- std::string modele : le modèle du véhicule (ex. : "Corolla").
- int kilometrage : le kilométrage actuel.
- bool disponible : indique si le véhicule est disponible ou non.

Méthodes :

- Constructeur : Initialise les attributs du véhicule.
- Getters pour les attributs :
    - std::string getMarque() const;
    - std::string getModele() const;
    - int getKilometrage() const;
    - bool getDisponible() const;
- Setters pour modifier les attributs si nécessaire :
    - void setKilometrage(int nouveauKilometrage);
    - void setDisponibilite(bool etat);
- Surcharge d'opérateurs :
    - std::ostream& operator<<(std::ostream& os, const Vehicule& v);
    Permet d'afficher les détails d'un véhicule.
- Autres méthodes :
    - void afficherDetails() const; : Affiche les informations du véhicule.

2. Classe Employé

Représente les employés de l'entreprise qui peuvent réserver des véhicules.

Attributs privés :

- std::string nom : le nom de l'employé.
- std::vector<std::string> vehiculesReserves : liste des modèles des véhicules réservés.

Méthodes :

- Constructeur : Initialise le nom de l'employé.
- Getters :
    - std::string getNom() const;
    - std::vector<std::string> getVehiculesReserves() const;
- Autres méthodes :
    - void reserverVehicule(const std::string& modele); : Ajoute un véhicule à la liste des réservations.
    - void afficherReservations() const; : Affiche tous les véhicules réservés.

3. Classe Flotte

Représente la flotte complète de véhicules disponibles dans l'entreprise.

Attributs privés :

- std::vector<Vehicule> vehicules : la liste des véhicules.
- std::vector<Employe> employes : la liste des employés.

Méthodes :

- Constructeur : Initialise une flotte vide.
- Ajout des véhicules :
    - void ajouterVehicule(const Vehicule& vehicule); : Ajoute un véhicule à la flotte.
- Ajout des employés :
    - void ajouterEmploye(const Employe& employe); : Ajoute un employé.
- Réservation de véhicule :
    - void reserverVehicule(const std::string& modele, const std::string& nomEmploye);
    Permet à un employé de réserver un véhicule s'il est disponible.
- Gestion du kilométrage :
    - void miseAJourKilometrage(const std::string& modele, int nouveauKilometrage);
    Met à jour le kilométrage du véhicule après usage.
- Affichage des véhicules :
    - void afficherVehiculesDisponibles() const; : Affiche tous les véhicules disponibles.
    - void afficherFlotte() const; : Affiche tous les véhicules de la flotte.
- Affichage des employés et leurs réservations :
    - void afficherReservationsEmployes() const; : Affiche les employés avec leurs réservations.

Travail demandé dans la fonction `main()` :

1. Création de véhicules : Créez plusieurs objets de type `Vehicule` avec des caractéristiques variées (marque, modèle, kilométrage, disponibilité).
2. Ajout des véhicules à la flotte : Ajoutez les véhicules créés à la liste des véhicules de la flotte.
3. Création des employés : Créez plusieurs objets `Employe` avec des noms différents.
4. Ajout des employés à la flotte : Ajoutez les employés à la liste des employés de la flotte.
5. Affichage initial : Affichez les véhicules disponibles dans la flotte et les employés.
6. Simuler des réservations : Faites en sorte que certains employés réservent des véhicules disponibles.
7. Mise à jour des véhicules après usage : Simulez l'utilisation de certains véhicules en mettant à jour leur kilométrage.
8. Affichage final : Affichez l'état de la flotte après les réservations et les mises à jour.
9. Ajoutez une classe dérivée VehiculeElectrique qui hérite de la classe Vehicule. Cette classe doit inclure :

    - Un attribut privé supplémentaire :

        - int autonomie : l'autonomie du véhicule en kilomètres.

- Un constructeur qui initialise les attributs de la classe de base ainsi que l'autonomie.

- Une méthode supplémentaire :

  - void afficherAutonomie() const : affiche l'autonomie actuelle du véhicule.

a) Créez plusieurs objets VehiculeElectrique avec des marques, modèles, kilométrages, disponibilités, et autonomies variées.
b) Ajoutez ces objets à la liste des véhicules de la flotte.
c) Affichez les détails des véhicules électriques, y compris leur autonomie, en utilisant la méthode afficherAutonomie.