Contrôle Continu : POO 2 (C++)

Modalité du Projet

1. Le projet est reçu par les étudiants le Lundi 08-04-2024 avant 18h.
2. Le projet est à renvoyer par mail au plus tard le Lundi 22-04-2024 avant minuit.
3. La solution doit être envoyée par mail à : **m.amine@emsi.ma**
4. L'objet de l'Email doit être « **Contrôle-POO2-2024** » 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

Sujet du Projet : Gestion de trafic routier en C++

Introduction :

Le projet de gestion de trafic routier en C++ vise à développer une plateforme de gestion du trafic routier en utilisant les principes de la programmation orientée objet, notamment l'héritage et le polymorphisme. Cette plateforme permettra de simuler et de gérer efficacement la circulation des véhicules sur les routes, en fournissant des fonctionnalités avancées de gestion et de surveillance du trafic.

Objectif :

L'objectif principal de ce projet est de créer un système de gestion du trafic routier en C++ pour surveiller et gérer la circulation des véhicules sur les routes. En utilisant des classes de base et des classes dérivées pour différents types de véhicules, le système permettra d'ajouter, de visualiser et de contrôler les véhicules en circulation, tout en évitant les collisions potentielles.

Fonctionnalités attendues :

Le système comprendra une classe de base pour les véhicules, avec des classes dérivées pour les différents types tels que les voitures et les camions. Une classe de gestion du trafic sera responsable de la gestion des véhicules en circulation, avec des fonctionnalités telles que l'ajout de véhicules, l'affichage des détails de tous les véhicules et la vérification des collisions. Le système offrira également la possibilité d'ajouter d'autres types de véhicules avec des attributs spécifiques.

1. **Classe de base Vehicule :**
    - **Attributs :**
        - **marque :** une chaîne de caractères représentant la marque du véhicule.
        - **modele :** une chaîne de caractères représentant le modèle du véhicule.
        - **anneeFabrication :** un entier représentant l'année de fabrication du véhicule.
    - **Méthodes :**
        - **afficher() :** une méthode virtuelle permettant d'afficher les détails du véhicule.
2. **Classes dérivées :**
    - **Voiture :**
        - **Attribut supplémentaire :**
            - **couleur :** une chaîne de caractères représentant la couleur de la voiture.
        - **Méthode :**
            - **afficher() :** redéfinition de la méthode afficher() pour inclure la couleur de la voiture.
    - **Camion :**
        - **Attribut supplémentaire :**
            - **capaciteCharge** : un entier représentant la capacité de charge du camion en tonnes.
        - **Méthode :**
            - **afficher() :** redéfinition de la méthode afficher() pour inclure la capacité de charge du camion.
3. **Classe GestionTrafic :**
    - **Attributs :**
        - **vehicules :** un tableau de pointeurs vers des objets de type Vehicule.
        - **capacite :** un entier représentant la capacité maximale du tableau.
        - **taille :** un entier représentant le nombre de véhicules actuellement dans le tableau.
    - **Méthodes :**
        - **ajouterVehicule(Vehicule* vehicule) :** permet d'ajouter un véhicule à la plateforme.
        - **afficherVehicules() :** permet d'afficher les détails de tous les véhicules dans le trafic.
        - **verifierCollisions() :** permet de vérifier les collisions entre les véhicules présents dans le trafic.

Travail à faire :

1. Implémentez les classes **Vehicule**, **Voiture** et **Camion** en utilisant l'héritage et le polymorphisme.
2. Implémentez la classe **GestionTrafic** pour gérer les véhicules en circulation en utilisant un tableau de pointeurs.
3. Ajoutez d'autres types de véhicules (2 à 3) avec des attributs spécifiques et redéfinissez la méthode **afficher()** en conséquence.

4. Implémentez la méthode **verifierCollisions()** dans la classe **GestionTrafic** pour éviter les collisions entre les véhicules.
5. Testez votre application en ajoutant quelques véhicules à la plateforme et en affichant les détails de tous les véhicules.

Remarque :

Assurer une gestion appropriée de la mémoire allouée pour les véhicules à la fin du programme afin d'éviter les fuites de mémoire.

Note :

Il faut respecter les noms des classes, attributs, et méthodes pour que les tests fonctionnent.