Exercice 1

Dans cet exercice, nous allons concevoir un programme C++ pour modéliser des véhicules, en mettant l'accent sur les classes et l'héritage. Nous allons créer une classe de base appelée Vehicule, qui représentera les caractéristiques communes à tous les véhicules, telles que la vitesse maximale et le kilométrage. Ensuite, nous étendrons cette classe pour créer une classe spécifique appelée Bus, qui héritera des propriétés de base de Vehicule et ajoutera des attributs propres aux bus, tels que la capacité de sièges.

Nous mettrons en œuvre des constructeurs, des méthodes d'accès et de modification pour les attributs, et nous veillerons à ce que les valeurs soient toujours valides selon les spécifications données. Enfin, nous créerons une méthode pour afficher toutes les informations sur un bus spécifique.

Voyons maintenant en détail les étapes à suivre pour réaliser cet exercice.

a) Créez une classe Vehicule avec les attributs d'instance max_speed et mileage (propriétés privées). Notez que les deux propriétés sont de type int. Cette classe n'a qu'un seul constructeur, un constructeur sans argument qui initialise la valeur de max_speed par 240 et le kilométrage par 0.
b) Dans la classe Vehicule, ajoutez les méthodes publiques setmax_speed et getmax_speed, setmileage et getmileage pour obtenir et définir chacune des propriétés privées. Notez que lors de la définition de max_speed, elle ne peut pas être < 200 ; sinon, il est réglé sur 240. De plus, lors du réglage du kilométrage, il ne peut pas être <0 ; sinon, il est mis à 0.
c) Créez une classe Bus qui hérite de la classe Vehicule et possède, en plus des propriétés héritées, la propriété privée seat_capacity de type int. Cette classe n'a qu'un seul constructeur, un constructeur sans argument qui initialise la valeur de seat_capacity par 10.
d) Dans la classe Bus, ajoutez les méthodes getseating_capacity et setseating_capacity pour obtenir et définir le nombre de places assises d'un bus. Notez que lors de la définition de la valeur du nombre de places assises, elle doit être comprise entre 10 (inclus) et 50 (inclus), sinon sa valeur est définie sur 50.

e) Dans la classe Bus, ajoutez la méthode get_info qui affiche à l'écran toutes les informations sur un Bus, comme suit : Par exemple, si max_speed est de 240, le kilométrage est de 200 000 et la seat_capacity est de 40, alors la méthode get_info() affichera :

   Il s'agit d'un bus d'une capacité de 40 places, avec une vitesse maximale de 240 kmh et son kilométrage est de 200 000 km.

f) on considère la classe suivante :

```
#include <string>

class SchoolBus : public Bus {
private:
   std::string school_name;

public:
   // Constructeur
   SchoolBus(std::string sn) {
      school_name = sn;
   }

   // Getter pour school_name

   std::string getSchoolName() {
      return school_name;
   }

   // Setter pour school_name
   void setSchoolName(std::string sn) {
      school_name = sn;
   }
};
static int countbigbus(SchoolBus t[]){
//type your code here
}
```

Complétez la méthode countbigbus définie ci-dessus afin de renvoyer le nombre de bus scolaires ayant une capacité >= 40 places et leur kilométrage < 200 000 km.

Exercice 2

L'objectif de cet exercice est de mettre en œuvre la maquette d'une maison. On considère qu'une maison est un ensemble de pièces.
Une pièce possède un nombre de fenêtres (nb_windows) qui est de type entier et a une couleur (String). Une salle de bain est une pièce qui possède également une propriété has_Shower pour savoir si une salle de bain dispose d'une douche ou non.
Une chambre est une pièce qui possède également une propriété nb_beds contenant le nombre de lits dans la pièce.
a) Pour chaque classe dans le texte ci-dessus, définissez la classe et un seul constructeur qui définit toutes les propriétés de la classe.
N'oubliez pas l'héritage si nécessaire.

Astuce : pour créer une maison, le nombre de pièces est donné en argument, donc un tableau avec le nombre de pièces correspondant est créé dans le constructeur.

b) Pour chaque classe, ajoutez les méthodes get et set pour chaque propriété privée.

c) Dans la classe House, ajoutez deux méthodes : getNbOfBathroom qui imprime le nombre de salles de bain de la maison, et getNbOfBedroom qui imprime le nombre de chambres de la maison.

Questions:

Une classe abstraite peut-elle définir à la fois des méthodes abstraites et des méthodes non abstraites ?

Une classe parent abstraite peut-elle avoir des enfants non abstraits ?

Une méthode abstraite peut-elle être définie dans une classe non abstraite ?