Langage C++

- CIR2 -

TP8 : Héritage simple

Ayoub KARINE (ayoub.karine@isen-ouest.yncrea.fr)

Ce TP consiste à démarrer une hiérarchie de classes à partir de la classe Vehicle implémentée dans le TP n°6.
Il est conseillé de démarrer un nouveau projet, dans lequel vous récupérerez les classes Point et Vehicle comme base de travail.

1 – Modélisation

Modéliser les spécifications suivantes sous la forme d'un diagramme UML :

- Un véhicule possède à présent :
    - une quantité de carburant exprimée en L ⟶ float
    - une consommation exprimée en L / 100 km ⟶ float
- un véhicule peut avoir une motorisation diesel (DieselVehicle) ou essence (GasVehicle)
- un véhicule diesel peut être équipé d'un filtre à particules ou non (filtreAParticules)
- un véhicule essence possède un indice d'octane exprimé en % (octaneRate) --> entier

2 – Implémentation

L'objectif va être l'implémentation du diagramme UML de la partie 1 en C++.

1. Changer le type de l'attribut de char* à string
2. Programmer les getters et les setters des trois classes
3. Programmer les destructeurs des trois classes qui affichent juste le prototype et l'adresse de l'instance manipulée
4. Pour les constructeurs, merci de respecter les contraintes suivantes :
    a. chaque constructeur personnalisé doit afficher le prototype et l'adresse de l'instance manipulée
    b. un véhicule doit pouvoir être instancié avec une position, avec une immatriculation, une quantité de carburant et une consommation
    c. un véhicule diesel doit pouvoir être instancié avec tous les paramètres précédents + le fait d'être équipé ou pas d'un filtre à particules

d. un véhicule à essence doit pouvoir être instancié avec tous les paramètres précédents + son indice d'octane
e. chaque constructeur doit initialiser **uniquement** les membres présents dans sa classe

3 – Tests

1. Soit la fonction de test suivante :
    a. Vérifier l'appel des constructeurs et destructeurs
    b. Mettre des points d'arrêt et lancer le débogage

```
void displayTests1(){

   Vehicule vehicle(Point(5,6), "CC-987-JU", 50, 5.3);

   GasVehicle gasVehicle(Point(25,3), "XY-358-PQ", 60, 6.8, 95);

   DieselVehicle dieselVehicle(Point(2,10), "HD-888-ZY", 40, 5.5, true);

}
```

2. Dans une fonction de test séparée, instanciez un véhicule essence et accédez en lecture et en écriture :

    - aux attributs hérités

    - aux attributs spécifiques (indice d'octane ici)

Dans une autre fonction, faites les mêmes manipulations sur un véhicule diesel.

3. Affichage :
    a. Implémenter dans la classe Vehicle une méthode affichage() qui utilise la méthode print() déjà implémentée en TP6 et permet l'affichage suivant :

        ```
        [identifier] position: (X,Y), consumption: C, fuel left: F
        ```

    b. Proposer deux manières d'implémenter la méthode affichage() dans les classes DieselVehicle et GasVehicle (ci-dessous un exemple d'affichage):
        i. Une première qui redéfinit la méthode affichage() de la classe Vehicle
        ii. Une deuxième qui récupère les différents valeurs d'attributs de la classe mère et de la classe fille. **On interdit d'utiliser les getters pour cette deuxième méthode (sauf getX() et getY())**

    - affichage pour un véhicule essence :

```
[identifier] position: (X,Y), consumption: C, fuel left: F, octane rating: R
```

- affichage pour un véhicule diesel (pour cela, utiliser les conditions ternaires):

```
[identifier] position: (X,Y), consumption: C, fuel left: F, filter: yes/no
```

c. Tester la fonction affichage via le test suivant :

```
void displayTests2(){

   Vehicule vehicle(Point(5,6), "CC-987-JU", 50, 5.3);
   vehicle.affichage();

   GasVehicle gasVehicle(Point(25,3), "XY-358-PQ", 60, 6.8, 95);
   gasVehicle.affichage();

   DieselVehicle dieselVehicle(Point(2,10), "HD-888-ZY", 40, 5.5, true);
   dieselVehicle.affichage();
}
```

3 – Déplacement avec consommation

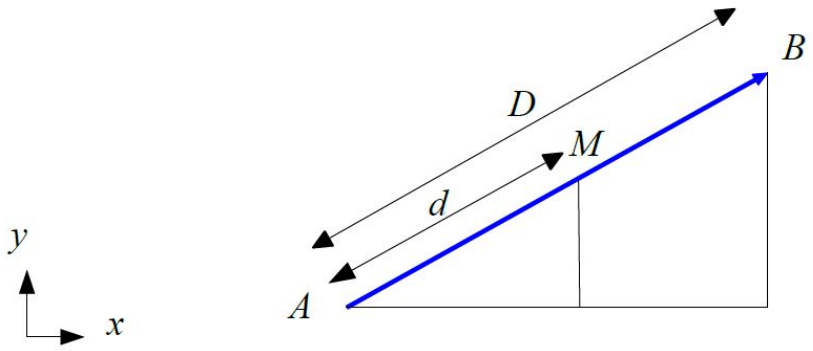
3-1 Implémentation

La fonction moveTo() doit à présent avoir le comportement suivant :

- elle déplace une voiture de sa position actuelle à une autre position
- elle doit calculer la quantité de carburant nécessaire pour effectuer le déplacement
- si la quantité de carburant restante est suffisante, le véhicule est déplacé au point visé
- sinon, le véhicule effectue le plus grand déplacement possible en direction du point visé
- dans tous les cas, la quantité de carburant restant doit être mise à jour

Nous supposerons que les coordonnées des points (et donc les distances) sont exprimées en km.

Pour vous faire gagner du temps, nous vous donnons les formules suivantes :



Supposons qu'un véhicule occupe actuellement la position *A* et doit être déplacé en *B*.

On note *D* la distance entre *A* et *B*, et *Q* la quantité de carburant nécessaire pour parcourir la distance *D*.

En consommant une quantité de carburant $q \leq Q$, le véhicule parcourt une distance :

$d = 100q / c$

où *c* est la consommation du véhicule, exprimée en L / 100km.

Il atteint alors le point *M* défini par :

$M = (1-r)A + rB$

où $r = d / D$.

On remarque de *M* se situe entre *A* et *B*, proportionnellement au ratio *r*.

Deux cas particuliers à noter :

- si on a assez de carburant, alors $q = Q \Rightarrow d = D \Rightarrow r = 1 \Rightarrow M = B$
- si on a plus de carburant, alors $q = 0 \Rightarrow d = 0 \Rightarrow r = 0 \Rightarrow M = A$

Concrètement, pour que cette égalité soit utilisable dans votre code, il faut la projeter sur les axes x et y.

3-1 Tests

1. Essayez d'anticiper ce qui est affiché par la fonction de test suivante :

```
void consumptionTests(){
   DieselVehicle dieselVehicle(Point(2,10), "XY-358-PQ", 8, 5, false);
   cout << "*******" << endl;
   float traveletDist1 = dieselVehicle.moveTo(0, 100);
   std::cout << "Traveled distance: " << traveletDist1 << std::endl;
   //dieselVehicle.affichage();
   cout << "*******" << endl;
```

```
    float traveletDist2 = dieselVehicle.moveTo(0, 200);
    std::cout << "Traveled distance : " << traveletDist2 << std::endl;
    //dieselVehicle.affichage();
}
```

2. Vérifiez en exécutant cette fonction dans votre IDE
3. Dans une fonction séparée, faites un test similaire avec un véhicule essence.