E. Grislin
UPHF

Sujet pour la série de TPs

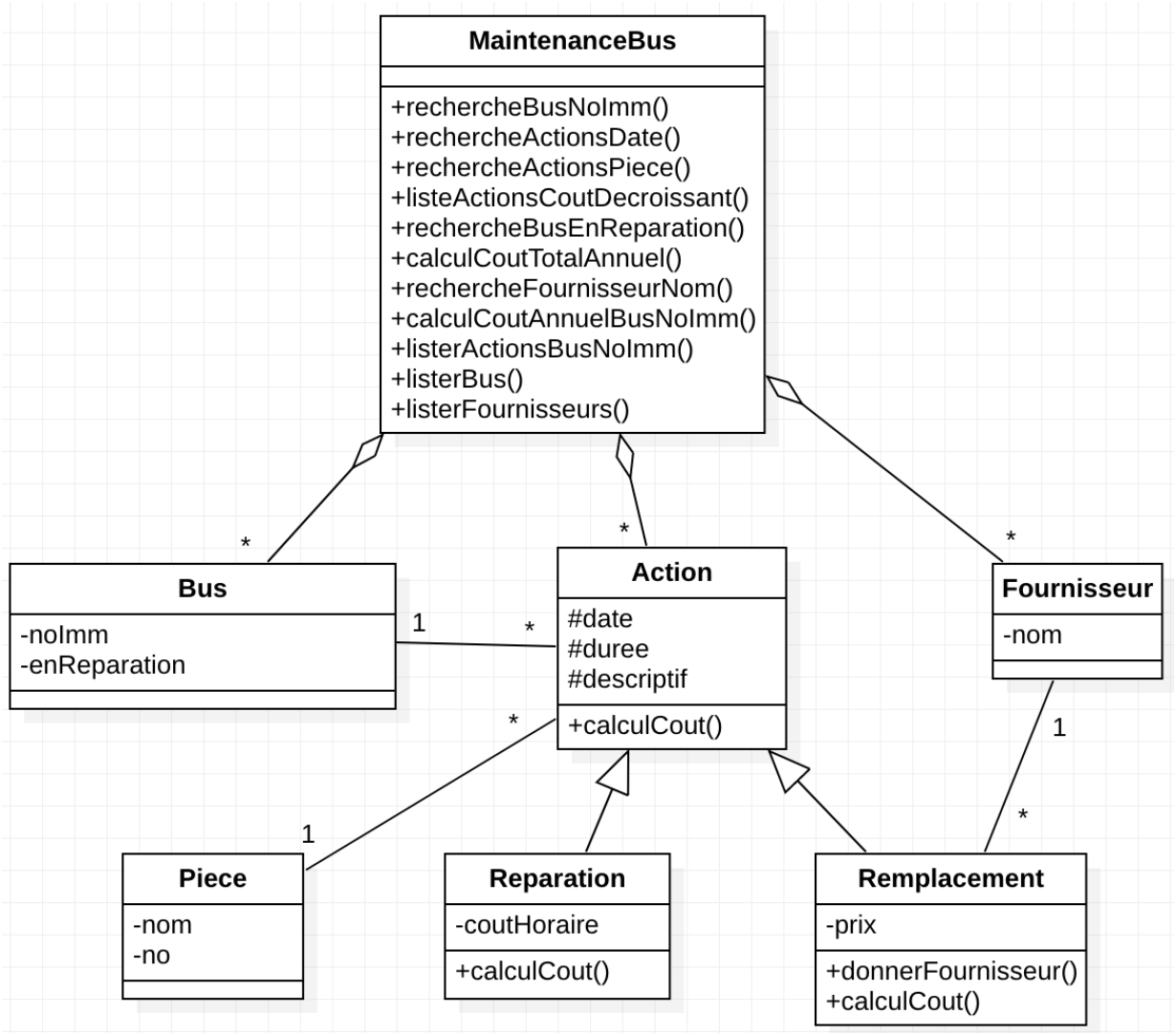
Avant de démarrer : si ce n'est pas encore fait, installer un éditeur et un compilateur pour le C++ :

- *Sous Windows : CodeBlocks ou équivalent ;*
- *Sous Unix, Linux : votre éditeur habituel et g++. Sur les machines ISTV, les install. éventuelles seront faites sur votre espace réservé (généralement W :) ;*
- *Sous MacOS : XCode ou équivalent.*

***A la fin de chaque TP**, vous déposerez sur Moodle, dans le cours "Introduction à l'Orienté Objet et C++", vos programmes sources uniquement (.h ou .hpp, et .cpp) dans un fichier compressé votre-Nom.zip. Si vous êtes en binôme, vos2Noms.zip.*

Attention les sujets des séances ne sont pas indépendants. Il est recommandé de finir ce qui est demandé dans un TP avant d'aborder le suivant.
Séparer les déclarations dans des fichiers "headers" (.h ou .hpp selon votre environnement de développement) et les corps des méthodes dans des fichiers "bodies" (.cc ou .cpp).

Dans ces séances de TP, vous allez créer quelques classes et fonctions pour un service de maintenance de bus. Les classes à développer sont présentées dans le diagramme ci-dessous. Ce diagramme n'est pas exhaustif, il vous donne des indications générales, à vous de compléter.



TP 1 : Menu, les bus, les fournisseurs

1. Dans un fichier de test, créer un ou plusieurs menus utilisateur qui réponde aux fonctionnalités définies par les opérations de la classe MaintenanceBus. Ce(s) menu(s) lancés dans le *main* doivent permettre à l'utilisateur d'interagir avec votre programme. Vous remplirez les fonctionnalités demandées au fur-et-à-mesure des TPs. En attendant, les fonctionnalités non encore développées provoquent un affichage "Fonctionnalité à développer".
2. Un Bus est défini par un numéro d'immatriculation unique (string) et un état actuel "enReparation" (1= en réparation, 0 = pas en réparation). Un Fournisseur est défini par son nom. Développer ces classes selon les indications du diagramme en ajoutant aussi les méthodes usuelles (constructeurs, affichage, get et set).
3. Tester en donnant directement les valeurs des attributs aux Bus et Fournisseurs.

TP 2 : Fichier de bus, fichier de fournisseurs et début de MaintenanceBus

1. La classe MaintenanceBus contiendra au final 3 ensembles d'objets, sous forme de `vector`s, pour les Bus, les Actions et les Fournisseurs. Commencez par définir cette classe avec uniquement les Bus et les Fournisseurs.
2. Ecrire une méthode de MaintenanceBus qui crée le `vector` de pointeurs de Bus à partir d'un fichier "dataBus.txt" contenant des données permettant de construire plusieurs objets Bus. Tester avec un affichage de la liste de Bus.
3. Faites de même pour les Fournisseurs à partir d'un autre fichier ("dataFournisseurs.txt") contenant les données des fournisseurs.

TPs 3 : Les Pieces et les Actions

1. Une action de maintenance est définie par une date (string ou autre), une durée (nombre d'heures), un descriptif (string) et un objet Piece. Développer les classes Piece et Action selon les indications du diagramme en ajoutant aussi les méthodes usuelles (constructeurs, affichage, get et set).
2. Ecrire les méthodes de MaintenanceBus pour créer le `vector` de pointeurs d'Actions à partir de données contenues dans un fichier "dataActions.txt", et l'afficher.

TP4 : les Actions de réparation et de remplacement

1. Il existe 2 types d'actions, les actions qui consistent à réaliser une réparation et celles qui nécessitent le remplacement de la pièce en question. Une réparation comporte un coût horaire (float). Un remplacement comporte un prix (float) et un pointeur vers un objet Fournisseur. Les 2 classes redéfinissent la méthode `calculCout` présente dans la classe Action. Ecrire les classes dérivées Reparation et Remplacement.
2. Il faut maintenant prendre en compte les 2 types d'actions pour l'entrée des données. 2 solutions au choix : à partir de 2 fichiers différents ou à partir d'une version modifiée du fichier précédent (du TP3). De même, l'affichage doit être différent selon que l'action concerne une réparation ou un remplacement.

TP5 : les méthodes de MaintenanceBus

1. La méthode `listeActionsCoutDecroissant` de MaintenanceBus liste les actions selon leur coût décroissant. Il existe pour cela des méthodes de tri sur les vectors. Mais ces méthodes ont besoin de pouvoir comparer 2 Actions : redéfinir l'opérateur "$<$" pour la classe Action en faisant une comparaison selon leurs coûts.
2. Finir d'écrire les méthodes de MaintenanceBus visibles sur le diagramme UML, en ajouter si vous pensez que certaines fonctionnalités ne sont pas représentées.

TP 6 : Finalisation

Ce dernier TP doit vous permettre de terminer si besoin votre code, de le tester et de le documenter (commentaires).