Héritage multiple
- Parent travailleur

Presque tous les parents québécois de nos jours sont des travailleurs. Pour modéliser cette relation, vous devez dessiner le diagramme UML qui représente la classe `Pere` (ou `Mere`) qui hérite de `Parent` et de `Travailleur` à la fois. L'une des deux classes de base doit être une simple interface. À la fin, nous voulons que la classe `Pere` puisse être contrôlé à travers les méthodes suivantes :

A) Constructeur par défaut (pas d'enfants ,0 en salaire, Chômeur pour le travail)
B) Constructeur par paramètre (`vector<Enfant*>`, salaire (`double`), travail (`string`))
C) Méthode pour ajouter un enfant (Enfant*)
D) Méthode d'accès pour le nombre d'enfants (`unsigned int`)
E) Méthodes d'accès et de modification pour le salaire (`double`)
F) Méthodes d'accès et de modification pour la nature du travail (`string`)
G) Méthode `void` travailler()
H) Méthodes d'accès et de modification pour l'expérience de travail (`unsigned int`)
I) Méthodes d'accès et de modification pour l'âge (`unsigned int`)
J) Méthodes d'accès et de modification pour s'il a une conjointe (`bool`)
K) Méthodes d'accès et de modification pour son passetemps favori (`string`)

C'est à vous de décider quelle(s) méthode(s) va (vont) dans quelle(s) classe(s). Mettez aussi les attributs aux bons endroits. N'oubliez pas qu'une des classes de bases doit être une interface.

Solution sur la page suivante

Solution

Cette solution utilise `Travailleur` comme interface, car l'implémentation des méthodes de la classe `Travailleur` sont bien plus sujettes à des modifications que celles de la classe `Parent`. Les méthodes de la classe `Travailleur` sont dupliquées, car nous devons les implémenter dans la classe `Pere` puisqu'elles sont déclarées virtuelles pures dans la classe de base.