**AKOKA Théo 33**

**EXERCICE 1**

**1.2 a :** Oui car l’ajout de nouveaux types d’instruments inclus l’ajout d’une nouvelle classe qui correspond au type d’instrument mais inclut aussi la modification de la classe Orchestra en créant une nouvelle liste et les deux méthodes pour accorder et ajouter un instrument à la liste.

Nouvelle classe Stringed qui sera utilisée dans la classe Orchestra qui a une variable liste de Stringed et on ajoute une méthode tuneStringed(Stringed) et addStringed(Stringed)

**1.2 b :** Oui il faut implémenter chacun des instruments et accorder chacun d’eux un à un.

**1.3 :** Il ne respect pas le principe ouvert/fermé parce que la modification des classes des types d’instruments peut impliquer de modifications partout, il n’y a pas d’abstraction et les instruments sont tous implémenter avec des classes différentes, il n’y a rien de fixé. Ainsi si les classes percussion et wind sont modifiés, on doit modifier la classe Orchestra qui utilise ces deux types d’instruments.

**1.4 :** Pour mettre l’instruments de type Brass en pause on peut créer une méthode pause dans la classe Brass. Mais cela empêchera le respect du principe de Substitution de Liskov car la classe Brass qui hérite de la classe Wind, ne pourrait plus remplacer Wind.

**1.5 :** Il n’y a aucune abstraction donc ce design ne respecte pas le principe des Inversion des dépendances.

**EXERCICE 2**

**2.3 a :** L’ajout d’instrument n’est pas compliqué il suffit de créer une classe qui hérite de la classe abstraite Instrument et d’implémenter toutes les méthodes. Aucune autre modification n’est nécessaire.

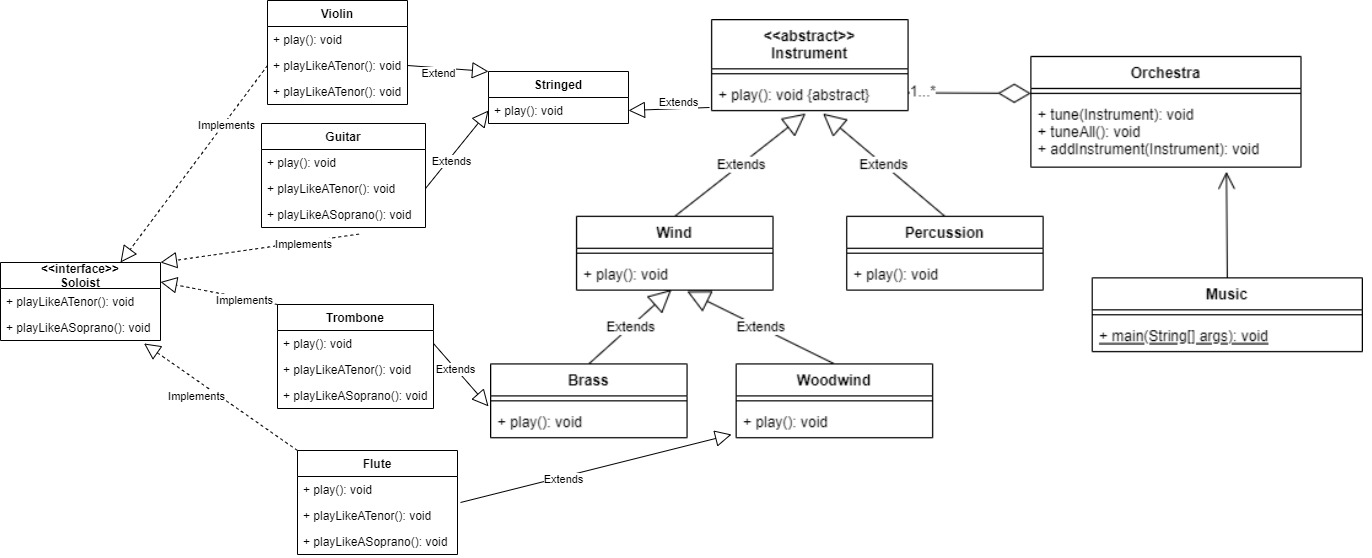
**2.3 b :** La réutilisation de la classe Orchestra est plus facile car les instruments qui sont utilisé dans les méthodes de Ochestra sont de la classe instruments ainsi il suffit de créer les variables instruments qui peuvent être n’importe quel instrument ensuite les inclure dans l’orchestre.

**2.4 :** Les instruments sont issus de la même classe abstraite ainsi la classe Orchestra qui utilise les instruments est ouverte aux extensions et ainsi l’implémentation des instruments est inviolable.

**2.5 :** Pour mettre l’instruments de type Brass en pause on peut créer une méthode pause dans la classe Brass. Mais cela empêche le respect du principe de Substitution de Liskov car la classe Brass qui hérite de la classe Wind, ne pourrait plus remplacer Wind.

**2.6 :** Le principe d’Inversion des dépendances n’est pas respecté car les classes Brass et Woodwind hérite de la classe concrète Wind.

**EXERCICE 3**

**3.8 : **

**3.9** Il est inutile d'avoir d'implémenter cette méthode playLikeASoprano pour la guitare comme il ne peut pas jouer de cette manière.

**3.10 a :** L’ajout d’instrument n’est pas compliqué il suffit de créer une classe qui hérite de la classe abstraite Instrument et d’implémenter toutes les méthodes. Aucune autre modification n’est nécessaire.

**3.10 b :** La réutilisation de la classe Orchestra est plus facile car les instruments qui sont utilisé dans les méthodes de Ochestra sont de la classe instruments ainsi il suffit de créer les variables instruments qui peuvent être n’importe quel instrument ensuite les inclure dans l’orchestre.

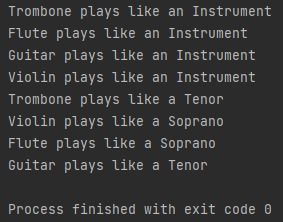
**3.11 :** Les instruments sont issus de la même classe abstraite ainsi la classe Orchestra qui utilise les instruments est ouverte aux extensions et ainsi l’implémentation des instruments est inviolable.

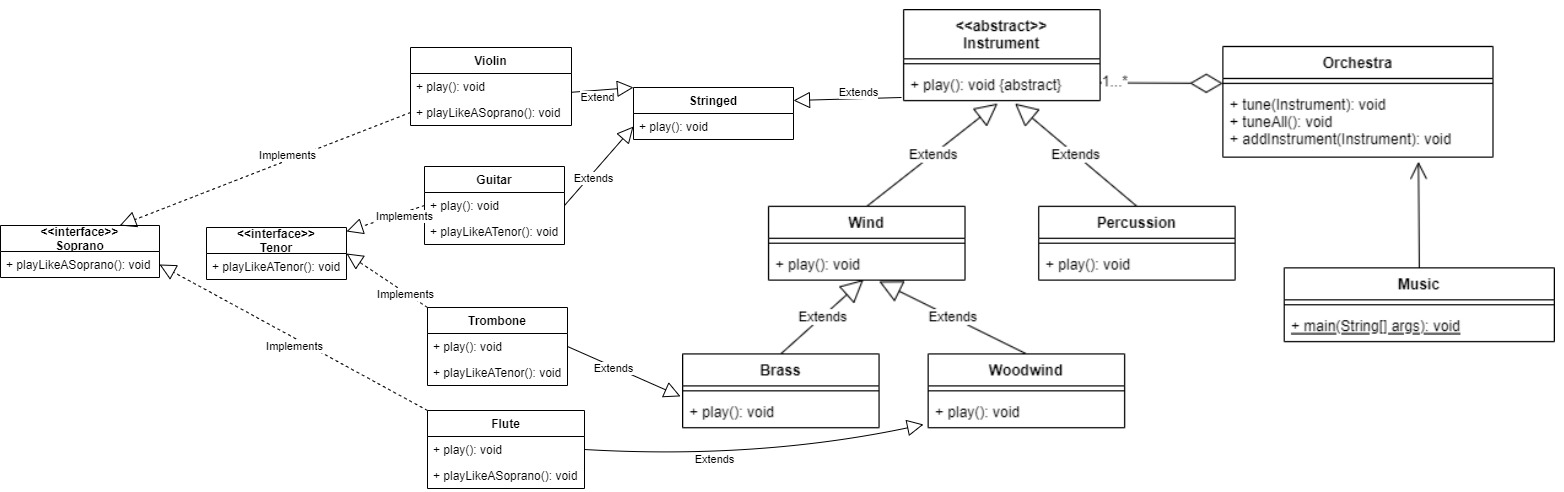
**3.12 :** Pour mettre l’instruments de type Brass en pause on peut créer une méthode pause dans la classe Brass. Mais cela empêche le respect du principe de Substitution de Liskov car la classe Brass qui hérite de la classe Wind, ne pourrait plus remplacer Wind.

**3.13 :** Le principe d’Inversion des dépendances n’est pas respecté car les classes Brass et Woodwind hérite de la classe concrète Wind.

**3.14 :** Comme l’indique la question 3.9 la Guitar utilise une méthode inutile alors la Ségrégation des Interfaces n’est pas respectée.

**EXERCICE 4**

**4.7 :** 

**4.8 :**

**4.9 a :** Pour rajouter la Trompette, on crée une classe Trompette qui hérite de Brass et Soprano.

**4.9 b :** Non il suffit de modifier la méthode play dans Guitar.

**4.9 c :** La réutilisation de la classe Orchestra est plus facile car les instruments qui sont utilisé dans les méthodes de Ochestra sont de la classe instruments ainsi il suffit de créer les variables instruments qui peuvent être n’importe quel instrument ensuite les inclure dans l’orchestre.

**4.10 : Les principes SOLID sont-ils respectés ?  
Responsabilité unique** ; chaque module, classe, ou méthode doit être responsable d’une seule partie de la fonctionnalité que le logiciel fournit, et cette responsabilité devrait être entièrement encapsulée par la classe, le module ou la méthode : **OUI**

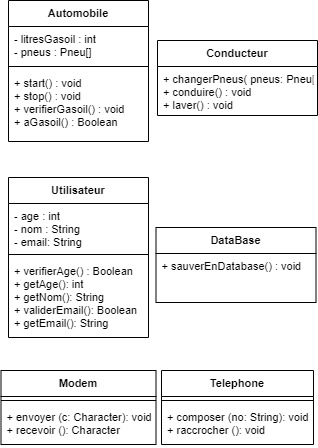
**Ouvert/Fermé** ; toutes les entités logicielles doivent être ouvertes à l’extension, mais fermées à la modification : **OUI**

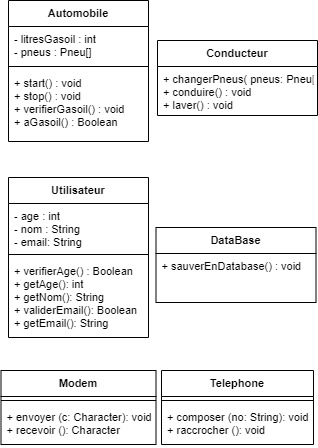
**Substitution de Liskov** ; pour mettre l’instruments de type Brass en pause on peut créer une méthode pause dans la classe Brass. Mais cela empêche le respect du principe de Substitution de Liskov car la classe Brass qui hérite de la classe Wind, ne pourrait plus remplacer Wind : **NON**

**Ségrégation des interfaces** ; une classe ne doit jamais être forcée à implémenter une interface qu’elle n’utilise pas ou une méthode qui n’a pas de sens pour elle : **OUI**

**Inversion des dépendances** ; le principe d’Inversion des dépendances n’est pas respecté car les classes Brass et Woodwind hérite de la classe concrète Wind : **NON**

**EXERCICE 5**

**5.1 :**  


**5.2 :**

**5.3 :**