Tests Partie 5

Louis Gentillet & Sebastian Simon

Préoccupations:

1. éviter la duplication de code ;

2. ajouter des caractères sans modifier le code existant dans notre paquetage

# 1 Pourquoi l’héritage

Scenario 1: IF ELSE dans la même classe pour réaliser les passagers lunatiques et passagerStressé

Quelle conclusion par rapport à nos deux préoccupations ?

- Par rapport à notre première préoccupation nous trouvons que nous ne sommes pas très préoccupé: il n'y a pas de code dupliqué...

- par contre, on est obligé de remanier du code existant à chaque fois qu'un nouveau attribut

doit être modifié

Scenario 2: Nouvelle classe pour chaqu'un de ces spécimen.

Quelle conclusion par rapport à nos deux préoccupations ?

- La notion de copier/coller ne colle pas vraiment avec notre première préoccupation

- Par contre on ne va certainement plus toucher au PassagerStandard après.

Scenario 3: L'héritage est un concept humain ancien. On va créer des classes filles de Passager Standard.

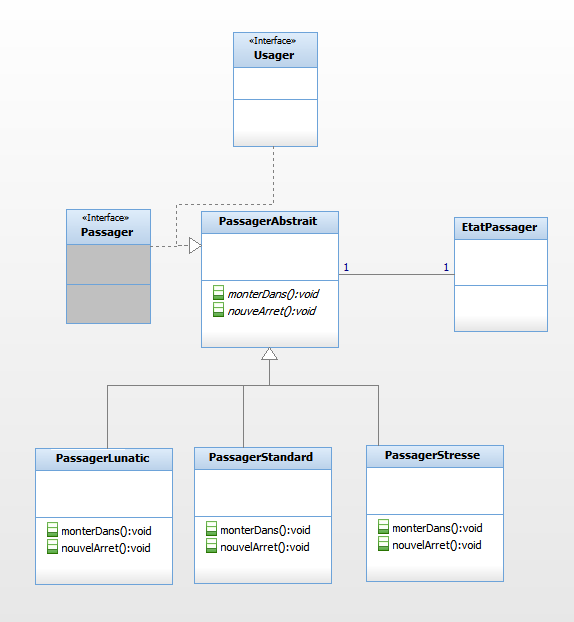
Quelle conclusion par rapport à nos deux préoccupations ?

- On va laisser faire le compilateur le boulot sale de copier/coller (car un héritage se réalise par rien d'autre que de copier-coller au niveau du compilateur (Cours Combemale). Mais, on pourra faire semblant de ne pas copier du tout. Très bien!

- Et en plus on ne va rien changer dans le code existant de PassagerStandard.

Quel code dupliqué?

- Le code pour tester les passagers de classe fille



# 2 Une classe de base abstraite

Pour pouvoir redéfinir la méthode nouvelArret() dans les sous-classes, il faut avoir accès à la valeur de la variable qui stocke la destination.

Comment la rendre accessible aux classes dérivées?

- par un getter ordinaire, ou changer le modificateur de la variable en public ou protected

Dans quel cas est-il préférable de définir une méthode qui renvoie sa valeur ? Quel est le modificateur d’accès de cette méthode ?

- puisque mettre public une variable est une mauvaise pratique, nous allons traiter la question de protected - il est préférable d'avoir une méthode getter dans les cas où on ne veut pas permettre l'accès écriture à la variable en question.

Le modificateur de la méthode est protected, il le faut juste dans les sous-classes.

Pourquoi l’appel au constructeur de la classe de base est-il nécessaire ?

- le constructeur de la classe de base de base ne prend pas d'argument. Pour notre usage il est nécessaire qu'il soit appelé avec des paramètres. D'où la nécessité.

Si le corps du constructeur de PassagerStandard est vide, vous obtenez un message d’erreur. Expliquez ce message d’erreur.

- Si la classe de base n'a pas de constructeur, elle en a un par défaut (sans arguments) qui est appelé. Si un constructeur est implémenté dans la classe de base, le constructeur "défaut" n'est plus dispo et du coup toutes les classes filles doivent appelé le constructeur implémenté. Si le constructeur sans paramètres n'est pas implémenté dans la classe de base, une tentative appel à partir d'une classe de fille échoue (et donc aussi l'appel automatique du constructeur "défaut").

# 3 Définir ce qu’il faut paramEtrer

Vous remarquez que tout le code n’est pas à changer, une partie est à dupliquer. Cette partie doit d’ailleurs être présente dans toutes les classes d'arrivée. Comment factoriser ce code commun obligatoire dans la classe de base ?

- Le code commun à toute sous-classe de passager abstraite et celui qui permet au passager de descendre.

On pourrait le définir dans la classe de base et faire en sorte que les sous-classes s'en servent avant d'appliquer leur comportement spécifique. Pour cela, la methode

Déclarer deux méthodes abstraites choixChangerPlace() et choixPlaceMontee() dans la classe PassagerAbstrait. Pourquoi les déclarer abstraites?

- Pour faire en sorte que les restent toujours à implementer lors de l'heritage des classes dérivées

Ces deux méthodes sont-elles privées, publiques ou protégées ?

- Elles sont protected, seule les sous-classes en ont besoin.

Pourquoi faut-il mieux déclarer les autres méthodes de la classes PassagerAbstrait comme finales ?

- Pour faire en sorte que les objets dérivées gardent un état consistant

# 4 Factoriser les Tests

# 5 Nouvelle Architecture