Compte rendu partie 5

1. Quelle conclusion par rapport `a nos deux pr´eoccupations?

On souhaite ne pas avoir à modifer le code existant juste le remanier, or la on doit modifier nos methodes et donc devoir revoir toutes nos implémentation. De plus on souhaite avoir une factorisation du code pour s’adapter à tous les sous type de PassagerStandard, mettre des switch case ou des condition à l’interrieur des méthodes n’est pas du tout une bonne solution.

Deuxieme solution :

Cette solution est loin d’être une solution car la où on voulait de la factorisation, nous copion et collons plusieurs fois le même contenu. Cette solution n’est pas viable rien que à cause de la pérennité du code.

Troisième solution (heritage):

C’est la solution la plus adaptée car elle repond à notre problematique : ne redefinir que les méthodes monterDans() et nouvelArret() ; De plus on aura l’information que qu’elle type est le Passager via son instance.

Il n’y a pas de code dupliqué pour les méthodes monterDans et nouvelArret car on utilise déjà des Passager et donc le cas le plus général. Les methodes sont déjà factorisé.

1. Pour qu’une variable priver devienne accécible aux classes filles on modifie la visibilité en protected sous langage java.

Il est préférable de faire le guetter de cette variable dans la classe mere (ici PassagerAbstrait) car le corps de cette fonction sera le meme . On mettra la visibilité en public car un guetteur peut etre accecible partout sans probleme.

Pourquoi l’appel au constructeur de la classe de base est-il n´ecessaire? Si le corps du constructeur de PassagerStandard est vide, vous obtenez un message d’erreur. Expliquez ce message d’erreur.

On est obliger de faire appel à la classe mere car pour construire une fille il faut obligatoirement qu’une mere exite. On doit faire appelle (via le super() ) au constructeur de la classe mere au minima.

Partie 3

1. D´eclarer deux m´ethodes abstraites choixChangerPlace() et choixPlaceMontee() dans la classe PassagerAbstrait. Pourquoi les d´eclarer abstraites?

Il faut les declarers abstraites car leur comportement est différents selon les classes dérivée de PassagerAbstrait

1. Ces deux méthodes sont protected car elles ne sont utliser que par les classes filles lors de l’appelle des methodes MonterDans() ou allerArretSuivant()
2. On définit les autres méthodes de PassagerAbstrait en final pour eviter toute redefinition par les classes filles

On definit les classes PassagerStandard, PassagerLunatique et PassagerStresse comme final pour bloqué l’héritage et ne pas créer des sous PassagerStandard par exemple.

1. Pourquoi doit-il y avoir toujours le mˆeme nombre de test? Nous devons toujours avoir le même nombre de test qu’avant le déplacement des tests car les test dans la classes AbtraiteTest sont exécuter avant l’exécution de la classe de test courante. Or les tests manquant sont lancer dans la classe courrante juste après.

Pourquoi prendre comme type de retour PassagerAbstrait et non pas Passager ou Usager

PassagerAbstrait implément déjà Passager et Usager donc pas besoin.