COMPTE-RENDU PROGRAMMATION ORIENTÉE OBJET

2023 - 2024

Les Polymorphistes

DUTHIL Thomas

FORME Julian

MISTOU Paul

Table des matières

TD 3

2. Utiliser des objets

2.1. Tester la classe Jauge

2.2. Tester la classe Position

2.3. Boutez vos neurones

Question 1

Une méthode d'instance qui n'est pas applicable à la méthode *main()* nécessite une instance de classe pour être appelée. Tandis qu'une méthode *static* n'a pas besoin d'instance de classe pour être appelée. Une méthode main n'a pas besoin d'instance.

Question 2

Le fait d'instancier une classe de test permet d'avoir des cas de tests indépendants les uns des autres. Cela permet donc de garantir que les cas de tests s'exécutent tous dans les mêmes conditions.

Question 3

Les entiers négatifs posent aussi un problème car la valeur de la jauge peut aller de [0, max[. De plus, si la valeur de départ est strictement supérieure au maximum défini, il devrait y avoir un problème.

L'instanciation devrait faire un test pour vérifier que la valeur de départ est inférieure ou égale à la valeur maximale et que cette valeur maximale soit un entier positif.

```
if (max < 0 || depart > max || depart < 0)
    // KO
} else {
    // OK
}
```

Question 4

Dans le cas de test 3, on change la valeur de l'objet Jauge, tandis que dans le cas de Position, on ne modifie pas l'objet, c'est un objet constant. On se contente seulement de renvoyer un nouvel objet Position.

3. Un coup d'œil à le source ExempleAvecChaine.java

Question 1

La classe String représente une chaîne de caractères immuables, ce qui signifie que la chaîne ne peut pas être modifiée une fois créée. Toutes les opérations de modification d'une instance de String créent une nouvelle instance de String. StringBuffer et StringBuilder représentent une chaîne de caractères modifiable.

En ce qui concerne le comportement des instances des classes Jauge et Position, on peut faire un lien avec les classes String, StringBuffer et StringBuilder en termes de mutabilité. Les instances de Jauge et Position peuvent être modifiées en appelant des méthodes spécifiques pour changer leur état interne. Cela est similaire à la modification des instances de StringBuffer et StringBuilder en ajoutant ou en supprimant des caractères. Cependant, contrairement aux classes String, les instances de Jauge et Position sont mutables et peuvent être modifiées directement sans créer de nouvelles instances.

Question 2

Le partage d'un objet signifie que si on crée un deuxième objet "Un Truc", il pointera vers la même adresse mémoire que notre premier objet "Un Truc". Par conséquent, toutes les chaînes de caractères "Un Truc" sont représentées par une seule instance de la classe String.

Question 3

a. Si le mot-clef *final* s'applique à l'objet

L'état de l'objet ne pourra pas être modifié.

b. Si le mot-clef *final* s'applique à la variable

La référence d'objet contenu dans la variable en question ne pourra pas être redéfinie.

Le langage Java a été choisi car on peut justement appliquer le mot-clef *final* sur l'objet et sur la variable. C'est une spécificité de Java.

Question 4

Le premier objet String est converti en une instance de StringBuilder (ou StringBuffer si la concaténation est effectuée dans un environnement multithread) interne. Cela se fait pour permettre des opérations de concaténation plus efficaces

La méthode append() de l'objet StringBuilder (ou StringBuffer) est appelée pour ajouter la deuxième chaîne de caractères à la fin de la première.

Une nouvelle instance de la classe String est créée à partir du contenu de l'objet StringBuilder (ou StringBuffer) résultant de la concaténation.

TD 4

3. Le test de recette

3.1. Huston! Nous avons un problème

L'erreur "*java.util.ConcurrentModificationException*" est levée lorsque la structure d'une collection est modifiée pendant que l'itérateur est en cours d'utilisation. Dans notre cas, la méthode *allerArretSuivant()*, une modification a été apportée à la collection pendant que l'itérateur parcourait les éléments, ce qui a provoqué une exception.

Effectuer une copie de la structure de données peut être très coûteux en termes de mémoire et de performance.

Une méthode alternative serait de stocker les passagers qui doivent sortir dans une liste et de les supprimer à la fin de l'itérateur.

TD 5

1. Partage des instances de la classe Position

Actuellement, il nous faudrait 30 instances de *Position* pour 10 classes de *PassagerStandard*.

Après avoir ajouté trois variables d'instances, la compilation du code provoque une erreur *StackOverflow*. Cela est dû au fait que lorsqu'on instancie la classe *Position*, les 3 variables précédemment ajoutées créent elles aussi une instance de *Position*, et donc cela provoque une boucle infinie. Les variables d'instances sont créées à l'instanciation.

On définit la visibilité des constructeurs en *private* pour qu'on ne puisse pas instancier la classe *Position* avec le mot clé *new*. On passera plutôt par une méthode.

La classe *Position* est instanciée lors du chargement de la classe (référence à *Position* dans *PassagerStandard)*. La classe est instanciée plus tôt, et ne le sera qu'une seule fois.

On peut supprimer le constructeur *private Position()* ainsi que l'attribut *COURANT*.

L'option *-cp* signifie que le compilateur Java recherchera les dépendances de la classe Simple dans le répertoire "build". Sans l'option *-cp*, le compilateur ne serait pas en mesure de trouver les dépendances nécessaires et la compilation échouerait.

Les deux classes *Autobus* et *PassagerStandard* font référence l'une à l'autre, par conséquent il faut les compiler en même temps avec la commande : *javac -d build -cp build src/Autobus.java src/PassagerStandard.java*.

Le contenu du répertoire *build* possède donc un package *tec*.

L'ambiguïté dans cette commande provient du fait que le caractère "." est utilisé à la fois pour séparer les répertoires dans le chemin de classe et pour indiquer l'extension du fichier.

Lorsque qu’on spécifie "*Jauge.class*" dans la commande, l'interpréteur Java peut interpréter le "." comme faisant partie du nom de fichier et non comme séparateur de répertoire. Cela peut entraîner une erreur car il ne peut pas trouver le fichier "*Jauge/class*" dans le répertoire spécifié.