université PARIS DIDEROT PARIS 7

INITIATION À LA PROGRAMMATION 2 (IP2)

TP3 bis : liste de tâches

Ce TP n'est pas à rendre sauf demande contraire de votre enseignant qui pourrait vous demander d'en rendre une partie à la place d'une autre partie du TP3.

Il s'agit de construire un petit système de gestion de tâches. Tout au long du sujet, vous pouvez utiliser des tableaux ou des `ArrayList` suivant ce que vous préférez.

Tâches On veut définir une classe `Tache` qui possède comme attributs un `nom` (une chaîne de caractères non modifiable) et un tableau `necessairePour` de `Tache`. L'idée est qu'il faut nécessairement réaliser la tâche avant d'avoir le droit d'exécuter les tâches du tableau `necessairePour`. Par exemple, il faut avoir fait les courses avant de faire la cuisine. Il faut avoir fait la cuisine avant de pouvoir manger. Il s'agit de dépendances directes (il faut avoir fait les courses pour manger, mais c'est une dépendance indirecte qui n'est pas considérée dans le tableau `necessairePour`).

1. Écrivez une classe `Test` contenant une méthode `main`, et veillez à tester votre code à chaque question.
2. Écrivez la classe `Tache`. Le tableau `necessairePour` ne doit pas pouvoir être utilisé par une autre classe.
3. Écrivez un constructeur avec comme seul argument le nom. `necessairePour` est vide par défaut.
4. Écrivez une méthode `declareNecessairePour` qui prend un objet `Tache` et l'ajoute au tableau `necessairePour`.
5. Essayez dans votre `main` de la classe `Test` le code :

```
Tache courses = new Tache("faire les courses");
Tache mains = new Tache("se laver les mains");
Tache cuisine = new Tache("faire la cuisine");
Tache manger = new Tache("manger");
courses.declareNecessairePour(cuisine);
mains.declareNecessairePour(cuisine);
cuisine.declareNecessairePour(manger);
```

Numéroter les tâches et afficher

6. Ajoutez un attribut entier `numero` à la classe `Tache`, et ajoutez au constructeur un mécanisme permettant d'associer un unique numéro a chaque tâche (la première a le numéro 1, la seconde le numéro 2, etc). Tenez compte du fait que le numéro d'une tâche ne changera jamais pour décrire l'attribut.

7. Ecrivez une méthode `afficher` qui écrit sur la sortie standard la tâche sous une forme semblable à «`Tâche 3: faire les courses. Est nécessaire avant les tâches 4,5.`». On veillera à n'afficher qu'une phrase si aucune autre tâche ne dépend de la tâche affichée.

Dépendances

8. Ajoutez un attribut entier `nbDependances` qui compte le nombre d'instances de `Tache` nécessaires pour la réaliser. Par exemple, dans l'exemple de la question 5, `nbDependances` vaut 0 pour l'objet `courses`, et 2 pour l'objet `cuisine`. Modifiez le code de la méthode `necessairePour` de manière adéquate.
9. On dira qu'une tâche est *exécutable* si elle ne dépend d'aucune autre tâche. Écrivez une méthode `estExecutable` renvoyant un booléen permettant de savoir si la tâche est exécutable. Le code ne doit contenir ni «`if`», ni «`?`».

Agent Un agent possède un `nom` et un tableau des tâches qu'il peut exécuter immédiatement.

10. Créez une nouvelle classe `Agent` possédant comme attribut un `nom` (une chaîne de caractères) et un tableau de tâches appelé `tachesExecutables`. Écrivez un constructeur pour un `Agent` n'ayant aucune tâche exécutable par défaut.
11. Ajoutez une méthode permettant de tester si `tachesExecutables` est vide.
12. Ajoutez un attribut `agent` à `Tache` de type `Agent` qui doit contenir l'agent qui devra réaliser cette tâche. Modifiez le constructeur de `Tache` pour qu'il prenne comme argument supplémentaire cet agent. La tâche nouvellement créée est alors ajoutée à `tachesExecutables`. Testez dans `main`.
13. Modifiez la méthode `necessairePour` afin que seules les tâches exécutables apparaissent dans `tachesExecutables` (on pourra se servir de la méthode `estExecutable`). Testez dans `main`.

Faire travailler l'agent

14. Écrivez une méthode `faire` dans la classe `Tache`, qui exécute une tache (qui doit être exécutable). Cela consiste à : (1) afficher un texte similaire à «`Agent Thomas: faire les courses.`», (2) enlever la tâche du tableau `tachesExecutables` de l'agent, et (3) ajouter à `tachesExecutables` les tâches qui sont devenues exécutables.
15. Écrivez une méthode `donneTacheExecutable` dans la classe `Agent` qui renvoie une tâche exécutable pour cet agent s'il en existe une, et `null` sinon.
16. Écrivez une méthode `executerTout` dans la classe `Agent`, qui fait exécuter à l'agent toutes les tâches jusqu'à exhaustion. Testez.