

Programmation Orientée Objets avec Java

Projet POO

Stéphane Malandain / Yassin Rekik
Semestre d'automne 2024

Introduction

Dans le cadre du cours de Poo / Java, vous allez réaliser un projet entier. Ce travail se divise en 4 étapes. Chaque étape fait l'objet d'un rendu à une date fixe, avec une note. La moyenne des 4 notes vous donne une note de projet.

Le projet est individuel. Plagiat ou copie seront sévèrement puni ! Les modalités de dépôt vous seront communiquées sur le Git ultérieurement.

1 Première étape : les bases d'un gestionnaire de tâches

Ce projet consiste en la création d'un gestionnaire de tâches. Par définition, une tâche est un travail qui doit être effectué dans un temps donné. Les tâches peuvent être donc très différentes et très variées, réalisées par une, 2 ou plusieurs personnes.

Pour commencer, vous devez réaliser la classe `Task` (avec des variables de classes et d'instance). Pour l'instant, une tâche est représentée par un numéro unique, un titre, un descriptif et un booléen permettant de savoir si elle a été effectuée ou non.

1. Les classes `Task`, `Tasks` et la classe principale

1.1 Créer la classe `Task` avec les variables d'instances nécessaires. Créer la donnée adéquate afin de gérer automatiquement l'identificateur de la tâche de telle sorte que chaque tâche aille un numéro unique.

1.2 Écrire des constructeurs de la classe `Task`. Le premier constructeur prend en argument le titre, le libellé qui est descriptif de la tâche. Le second constructeur prend en argument uniquement le titre de la tâche. Les constructeurs permettent de construire une tâche valide.

```
public Task (String title, String description)
public Task (String title)
```

1.3 Écrire les méthodes (getter) retournant l'id, le titre, le descriptif et l'option `etat` qui décrit si la tâche est effectuée ou non.

```
public int getId()
public String getTitle()
public String getDescriptif()
public boolean getState()
```

1.4 Réaliser les setters, toString() et affichage :

- Modification du titre
- Modification du descriptif
- Modification de l'état effectué ou non
- Méthode toString() permettant de renvoyer la chaîne de caractère de l'objet
- Méthode permettant d'afficher une tâche

1.5 Tableau de tâches : La classe Tasks

- Écrire une classe Tasks permettant de stocker n tâches (Task) dans un tableau.
- Écrire les constructeurs, getters/setters adéquats pour utiliser convenablement la classe. Le constructeur prendra la taille du tableau en paramètre.
- Écrire les méthodes permettant d'ajouter une tâche, de la supprimer et de la modifier
- Écrire une méthode permettant de supprimer les tâches effectuées du tableau
- Écrire une méthode permettant de changer l'état d'une tâche
- Créer les méthodes qui permettent de rechercher une tâche par id ou par titre.

1.6 tests

A ce stade, tester vos classes et méthodes avec l'exécution suivante :

```
malandai@MacBook16 tp1 % java tp1_2024
idTache 2      t3
description :  tache 3
Etat :         false

idTache 1      t2
description :  tache 2
Etat :         false

idTache 0      t1
description :  tache 1
Etat :         false

idTache 4      t5
description :  tache 5
Etat :         false

idTache 3      t4
description :  tache 4
Etat :         false

***** après suppression taches 3 et 4 *****

idTache 1      t2
description :  tache 2
Etat :         false

idTache 0      t1
description :  tache 1
Etat :         false

idTache 4      t5
description :  tache 5
Etat :         false

***** après modif taches 1 et 2 effectuées *****

idTache 1      t2
description :  tache 2
Etat :         true

idTache 0      t1
description :  tache 1
Etat :         true

idTache 4      t5
description :  tache 5
Etat :         false
```

Correspondant à la création de 5 tâches, dans un tableau de taille 10. Les tâches sont ajoutées dans le désordre dans le tableau (t3,t2,t1,t5,t4). Ensuite, on supprime les tâches 3 et 4, puis on modifie (effectué à true) les tâches 1 et 2.

1.7 La classe `DynTasks` : une nouvelle version avec des tableaux dynamiques

Vous devez implémenter une amélioration de la classe `Tasks` en implémentant des tableaux dynamiques (objets). En effet, plus besoin avec cette implémentation, de définir la taille du tableau contenant les tâches, puisque celle-ci est dynamique. `DynTasks` est une classe instanciable. Utilisez un tableau statique comme champ privé. La fonctionnalité `append` par exemple, ajoute une nouvelle tâche à la fin du tableau de tâches et renvoie un nouveau tableau. N'exposez que les méthodes nécessaires. La classe est immuable. Par exemple, la méthode `append` ne modifie pas le tableau d'origine, mais retourne un nouveau tableau.

Adaptez les fonctionnalités définies précédemment.

Ajoutez de nouvelles fonctionnalités :

`Size`, `Of`, `Head`, `TaskDone`, `TaskInProgress`.

Tester cette implémentation selon l'exécution précédente et en ajoutant 2 tâches supplémentaires.

1.8. Réaliser un petit menu permettant à l'utilisateur d'avoir les fonctionnalités suivantes :

- Création d'un tableau de tâches
- Ajouter une tâche
- Affichage des tâches.
- Affichage des tâches effectuées / non effectuées
- Suppression d'une tâche.
- Modification d'une tâche

2 Seconde étape : héritage, interface et collections.

A Préciser ultérieurement

Avant tout, vous devez remplacer toutes les structures de données présentes dans la partie 1 sous forme de tableau par les collections pertinentes de votre choix.

Ensuite, L'interface `comparable` modélise les objets qui possède un ordre total : Elle déclare une seule méthode `compareTo` :

```
1 interface Comparable {  
2     int compareTo(Object o);  
3 }
```

Cette méthode retourne un entier selon que l'objet auquel elle est appliquée est plus grand, est égal ou est plus petit que o.

Vous devez réaliser une méthode type `compareTo()` permettant de comparer 2 tâches en fonction de leur id et de leur

3 Troisième étape : A Préciser ultérieurement

4 Quatrième étape : A Préciser ultérieurement