Test 1 Informatique : POO, Piles et Files

Exercice

Considérez l'implémentation d'une application de simulation d'un gestionnaire de tâches par un processeur en utilisant une file (queue) en Python. Chaque tâche est caractérisée par des attributs tels que le nom et la durée nécessaire pour son exécution. Dans cette simulation, les tâches seront défilées une par une pendant une durée constante régulièrement. Si le temps d'exécution d'une tâche n'est pas totalement consommé pendant ce défilement, elle sera remise dans la file pour être exécutée à nouveau.
L'objectif de cet exercice est de créer des classes Python et de les utiliser pour répondre aux questions suivantes. Assurez-vous d'écrire un code propre et lisible.
Classes principales à définir :

a) `Task (Tâche)` : Représente une tâche avec des attributs tels que le nom et la durée d'exécution. La durée indique le temps nécessaire pour compléter la tâche.

b) `TaskQueue (File de Tâches)` : Une file utilisée pour stocker les tâches en attente de traitement. Les tâches sont ajoutées à la file et retirées une par une pour simuler le traitement.

1. **(2 points)** Implémentez la classe `Task` avec :
   - Un constructeur `__init__`
   - Une méthode spéciale `__str__` pour afficher les détails de la tâche.
   - Un destructeur `__del__` qui affiche "Fin d'exécution de la tâche 'nom de la tâche' ".
2. **(3 points)** Implémentez la classe `TaskQueue` avec :
   - Un constructeur `__init__`
   - Une méthode `enqueue` pour ajouter une tâche à la file de tâches.
   - Une méthode `dequeue` pour retirer et renvoyer une tâche à partir de la file de tâches.
   - Une méthode spéciale `__str__` pour afficher l'état de la file.
3. **(6 points)** *Simulation du traitement de tâches*
   La simulation doit réduire le temps d'exécution restant pour chaque tâche pendant une durée constante (à définir). Si le temps d'exécution restant d'une tâche devient nul, la tâche doit être supprimée en appelant le destructeur de la classe *Task*
   L'objectif ici est d'écrire un programme principal en Python qui utilise les classes `Task` et `TaskQueue` pour simuler le traitement de n (entier aléatoire) tâches.

   Effectuer les opérations suivantes dans l'ordre :
   a. Définir un temps d'exécution constant *DURATION* (choisi aléatoirement entre 10^{-6} et 10^{-9} secondes) qui sera utilisé comme une variable globale.
   b. Initialiser une instance de `TaskQueue`.
   c. Ajouter n (où n est un entier supérieur ou égal à trois) tâches à la file (`TaskQueue`) en utilisant la méthode `enqueue`. Affichez l'état initial de la file.
   d. Créer une fonction `execute(task)` qui simule l'exécution d'une tâche. Si le temps d'exécution restant pour une tâche devient inféreur à la durée constante prévue pour l'exécution de chaque tache (*DURATION*), la tâche sera reexécutée une autre fois (sans l'enfiler) et elle sera marquée par un attribut d'instance supplémentaire nommé *finished* de type *bool* .
   e. Afficher le temps total nécessaire pour traiter toutes les tâches dans la file.

f. Exécuter toutes les tâches de la file (`TaskQueue`), en suppriment chaque tache marquée comme *finished*.

4. **(9 points)** *Simulation de l'exécution des tâches en fonction de leurs priorités*

   Vous êtes chargé d'améliorer l'application de simulation de traitement de tâches en introduisant la notion de priorité pour chaque tâche. L'objectif est d'ajouter une nouvelle classe appelée `PriorityTaskQueue`, qui étend la classe `TaskQueue` et gère les tâches en fonction de leur priorité (plus la priorité est élevée, plus la tâche est prioritaire) et d'implémenter les méthodes nécessaires pour respecter ce comportement.

   a) **(2 points)** Modifier la classe `Task` pour inclure un attribut de priorité (entier positif).
   La classe `Task` doit maintenant inclure un nouvel attribut, `priority`, qui représente la priorité de la tâche. Assurez-vous de mettre à jour le constructeur et la méthode `__str__` en conséquence.

   b) **(5 points)** Implémenter la classe `PriorityTaskQueue`.
   Créer la classe `PriorityTaskQueue` qui étend la classe `TaskQueue`. Cette nouvelle classe doit être capable d'ajouter des tâches en fonction de leur priorité et de les retirer de manière à respecter cette priorité. Implémentez les méthodes nécessaires telles que `enqueue`, `dequeue`, et `__str__` pour refléter le comportement de gestion de priorité.

   c) **(2 points)** Modifier le programme principal (Question 3) que vous avez écrit précédemment pour utiliser la nouvelle classe `PriorityTaskQueue` au lieu de `TaskQueue`.