Test 2 Informatique : POO, Piles et Files

Exercice

Considérez l'implémentation d'une application de simulation de traitement de tâches à l'aide de piles et de files en Python. L'objectif de cet exercice est de créer ces classes et de les utiliser pour répondre aux questions suivantes. Assurez-vous d'écrire un code propre et lisible.

a) `Task` : Représente une tâche avec un nom et une durée en minutes.

b) `TaskStack` : Une pile pour gérer les tâches à effectuer.

c) `TaskQueue` : Une file pour gérer les tâches en attente.

1. **(2 points)** Implémentez la classe `Task` avec un constructeur et une méthode `__str__` pour afficher les détails de la tâche.

2. **(3 points)** Implémentez la classe `TaskStack` avec un constructeur et les méthodes `push`, `pop` et `__str__` pour ajouter, retirer et afficher l'état de la pile.

3. **(3 points)** Implémentez la classe `TaskQueue` avec un constructeur et les méthodes `enqueue`, `dequeue` et `__str__` pour ajouter, retirer et afficher l'état de la file.

4. **(3 points)** *Simulation du traitement de tâches*

   Écrivez un programme principal en Python qui utilise les classes `Task`, `TaskStack` et `TaskQueue` pour simuler le traitement de n (entier aléatoire) tâches. Affichez l'état des piles et des files à chaque étape significative de la simulation.

   a) **(1 point)** Initialisez une instance de `TaskStack` et `TaskQueue`.

   b) **(2 points)** Effectuez les opérations suivantes dans l'ordre :

      1. Ajoutez n (où n est un entier supérieur ou égal à trois) tâches à la pile (`TaskStack`) en utilisant la méthode `push`. Affichez l'état de la pile.
      2. Transférez une tâche de la pile à la file (`TaskQueue`). Affichez l'état de la pile et de la file.
      3. Retirez une tâche de la file et ajoutez-la à la pile. Affichez l'état de la pile et de la file.
      4. Affichez le temps total nécessaire pour traiter toutes les tâches dans la pile et dans la file.

5. **(9 points)** *Gestion de tâches en fonction de leur priorité*

   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'joutez 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)** Modifiez 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)** Implémentez la classe `PriorityTaskQueue`.
   Créez 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)** Modifiez le programme principal (Question 3) que vous avez écrit précédemment pour utiliser la nouvelle classe `PriorityTaskQueue` au lieu de `TaskQueue`.