Test 2 Informatique : Corrigé

Exercice

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

```
class Task:
    def __init__(self, nom, duree, priorite=0):
        self.nom = nom
        self.duree = duree
        self.priorite = priorite

    def __str__(self):
        return "Tâche : {}, Durée : {} minutes, Priorité : {}" \
               .format(self.nom, self.duree, self.priorite)
```

2. Implémentez la classe `TaskStack` avec un constructeur et les méthodes `push`, `pop` et `__str__` pour ajouter, retirer et afficher une tâche de la pile.

```
class TaskStack:
    def __init__(self):
        self.taches = []

    def push(self, tache):
        self.taches.append(tache)

    def pop(self):
        if self.is_empty():
            return None
        return self.taches.pop()

    def is_empty(self):
        return len(self.taches) == 0

    def __str__(self):
        return "\n".join(str(tache) for tache in reversed(self.taches))
```

3. Implémentez la classe `TaskQueue` avec un constructeur et les méthodes `enqueue`, `dequeue` et `__str__` pour ajouter, retirer et afficher une tâche de la file.

```
class TaskQueue:
    def __init__(self):
        self.taches = []

    def enqueue(self, tache):
        self.taches.append(tache)

    def dequeue(self):
        if self.is_empty():
            return None
        return self.taches.pop(0)

    def is_empty(self):
        return len(self.taches) == 0

    def __str__(self):
        return "\n".join(str(tache) for tache in self.taches)
```

4. *Simulation du traitement de tâches*

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

   a) Initialisez une instance de `TaskStack` et `TaskQueue`.

   ```
   pile_taches = TaskStack()
   file_taches = TaskQueue()
   ```

   b) Effectuez les opérations suivantes dans l'ordre :

   1. Ajoutez au moins trois tâches à la pile (`TaskStack`) en utilisant la méthode `push`. Affichez l'état de la pile.

      ```
      pile_taches.push(Task("Tâche1", 5))
      pile_taches.push(Task("Tâche2", 8))
      pile_taches.push(Task("Tâche3", 3))

      print("Pile de tâches :")
      print(pile_taches)
      ```

   2. Transférez une tâche de la pile à la file (`TaskQueue`). Affichez l'état de la pile et de la file.

      ```
      tache_transferee = pile_taches.pop()
      file_taches.enqueue(tache_transferee)

      print("\nPile après le transfert d'une tâche :")
      print(pile_taches)

      print("\nFile après l'enfilement d'une tâche transférée :")
      print(file_taches)
      ```

   3. Retirez une tâche de la file et ajoutez-la à la pile. Affichez l'état de la pile et de la file.

      ```
      tache_defilee = file_taches.dequeue()
      pile_taches.push(tache_defilee)

      print("\nFile après le défilement d'une tâche :")
      print(file_taches)

      print("\nPile après l'ajout d'une tâche défilée :")
      print(pile_taches)
      ```

   4. Affichez le temps total nécessaire pour traiter toutes les tâches dans la pile et dans la file.

      ```
      temps_total_pile = sum(tache.duree for tache in pile_taches.taches)
      temps_total_file = sum(tache.duree for tache in file_taches.taches)

      print("Temps total pour les tâches dans la pile : {} minutes".format(temps_total_pile))
      print("Temps total pour les tâches dans la file : {} minutes".format(temps_total_file))
      ```

5. *Ajoutez une nouvelle classe* ***`PriorityTaskQueue`***

   ```
   class FileTachesPrioritaires(TaskQueue):
       def enqueue(self, tache):
           # Insérer la tâche à la position correcte en fonction de la priorité
           index = 0
           while index < len(self.taches) and tache.priorite <= self.taches[index].priorite:
               index += 1
           self.taches.insert(index, tache)
   ```

6. Modifiez le programme principal en Python que vous avez écrit précédemment pour utiliser la nouvelle classe `FileTachesPrioritaires` au lieu de `TaskQueue`.

   ```
   # Remplacez l'instanciation de TaskQueue par FileTachesPrioritaires
   file_taches = FileTachesPrioritaires()
   ```