UBS-ENSIBS
Cours : Modélisation Objet Avancée
Intervenant : Mohammed Lamine Kerdoudi

Mini-Projet : Gestionnaire de Tâches (Task Manager)

Ce projet consiste à développer un gestionnaire de tâches. Bien qu'une Interface Graphique (GUI) soit requise pour interagir avec le système, **l'évaluation** portera principalement sur la **Modélisation Objet Avancée**, ainsi que sur la **traduction correcte** et **cohérente** du modèle conceptuel vers le code (respect des classes et relations définis).

Le projet est divisé en deux parties :

1. **Partie 1 :** la conception orientée objet, les opérations CRUD de base et la validation des entrées.
2. **Partie 2 :** des fonctionnalités avancées telles que l'authentification, la collaboration, les notifications

Partie 1 :

1. Description : Développez la logique métier pour un gestionnaire de tâches. Le système doit permettre aux utilisateurs de créer, voir, éditer et supprimer des tâches. La solution doit imposer une séparation stricte entre le **Model** (Logique Métier) et la **View** (GUI).

2. Exigences du Modèle de Domaine : Vous devez implémenter la hiérarchie de classes suivante pour démontrer votre compréhension des Interfaces et de l'Abstraction :

- **Task (Interface) :** Définit le contrat pour les comportements d'une tâche. Les méthodes doivent inclure des opérations pour l'édition des détails, et le marquage comme « Completed », « In Progress », « Abandoned »,…
- **TaskImpl (Classe Concrète) :** L'entité concrète implémentant l'interface Task.
  - **Attributs :** name, description, dueDate, status.
- **TaskList (Classe Abstraite) :** Définit le modèle pour gérer une collection de tâches. Inclut des définitions abstraites pour add, remove, update, et display les tâches.
- **TaskListImpl (Classe Concrète) :** Étend TaskList. Cette classe gère la structure de données réelle (ex: ArrayList, LinkedList) et agit comme le **Controller** pour les données.

3. Exigences Fonctionnelles

- **Opérations CRUD :** Create, Read, Update, Delete sur les tâches.
- **Priorités (Priorities) :** Assigner des niveaux : *Low, Medium, High*.
- **Filtrage (Filtering) :** Filtrer par status (Completed, In Progress, Abandoned) ou par priorité
- **Tri (Sorting) :** Trier les tâches par dueDate ou par priorité
- **Recherche (Search) :** Trouver des tâches par mots-clés dans **name** ou **description**.
- **Commentaires :** Ajouter du contexte ou des notes à des tâches spécifiques.
- **Persistance :** Import/Export vers un fichier (XML, JSON) ou une base de données.

Vous devez valider toutes les entrées utilisateur *avant* qu'elles n'atteignent l'objet TaskImpl.

- **Contrainte :** Les champs **name** et **description** doivent être nettoyés pour garantir:
  - Les champs ne doivent être ni vides, ni nuls (gestion des chaînes vides).
  - La longueur du texte doit rester raisonnable (ex: limite de caractères).

5. Interface Utilisateur (The View)

- **Classe TaskManagerGUI :** Créez une interface conviviale en utilisant par exemple **Swing** ou **SWT**.
- L'interface graphique doit afficher la liste des tâches dans une zone dédiée et déléguer toute la logique à la classe TaskListImpl.

6. Travail à rendre (Partie 1)

- Créez un diagramme de cas d'utilisation pour illustrer les interactions entre les utilisateurs et le système.
- Créez un diagramme de classes pour ce système.
- Élaborez un diagramme de séquence pour montrer la création et l'ajout d'une de tâche à la liste,
- Élaborez un diagramme d'état-transition pour illustrer le cycle de vie de la tâche et les déclencheurs de changement d'état.
- Implémentation ce système en Java.

<u>Partie 2</u> : Gestionnaire de Tâches Avancé

1. Description

Étendez l'application pour supporter des environnements multi-utilisateurs et une organisation avancée.

2. Fonctionnalités Avancées

- **Authentification & Collaboration :**
  - Mettre en place une manière d'authentification des utilisateurs permettant à plusieurs utilisateurs de créer et de gérer leurs propres listes de tâches.
  - Permettre aux utilisateurs de partager des listes de tâches avec d'autres.
  - Empêcher l'Utilisateur A d'accéder aux tâches privées de l'Utilisateur B.
- **Statistiques :**
  - Visualisation de la productivité des utilisateurs (ex: tâches completed vs. in progress).

3. Système de Notification

Intégrez un moteur de notification. Lorsqu'un événement lié à une tâche survient, le système doit alerter l'utilisateur.

- **Déclencheurs (Triggers) :**
  - **Rappels d'échéance :** 15 mins, 1 heure, ou 1 jour avant la deadline.
  - **Alertes de Retard (Overdue) :** Quand CurrentTime > DueDate.
  - **Tâches Récurrentes :** Rappels pour les tâches quotidiennes/hebdomadaires.
  - **Mises à jour de Collaboration :** Notifier l'Utilisateur A quand l'Utilisateur B modifie une tâche partagée.

4. Travail à rendre (Partie 2)

1. Mettez à jour votre diagramme de cas d'utilisation pour prendre en compte ces nouvelles fonctionnalités.
2. Raffinez **votre Diagramme de Classes**
3. **Diagrammes de Séquence (Mis à jour) :** Montrez l'Utilisateur A éditant une tâche (partagée) et l'Utilisateur B recevant une mise à jour.
4. **Implémentation :** Code source Java complet.
5. Préparer une vidéo présentant le déroulement de l'utilisation de votre système