Rapport de Projet Final

Introduction

Dans ce projet, il est question pour nous de concevoir une application Java pour la gestion d'évènements. Les technologies utilisées dans ce projet sont les suivantes :

Concepts de Programmation Orientée Objet (POO)

Les concepts de POO implémentés dans ce projet sont :

- **Abstraction** : `Evenement` par exemple est une classe abstraite commune aux types d'évènements.
- **Héritage** : Les classes `Concert` et `Conference` héritent de la classe `Evenement`.
- **Polymorphisme** : Chaque type d'évènement implémente différemment la méthode `annuler()`.
- **Encapsulation** : Les attributs sont protégés. Nous avons défini les attributs d'évènement comme étant des attributs protégés pour pouvoir les utiliser dans les sous-classes (`Concert` et `Conference`) sans utiliser de getters ni de setters.

Design Pattern Observer

L'implémentation de ce pattern s'est faite à l'aide de deux interfaces :

- L'interface `EvenementObservable` permet de notifier les observateurs.
- L'interface `ParticipantObserver` permet aux participants de recevoir une notification par rapport à l'évènement concerné.

Ainsi, lorsqu'un évènement est annulé ou modifié, les participants inscrits reçoivent directement une notification.

Sérialisation JSON avec Jackson

Nous utilisons la bibliothèque **Jackson** pour :

- Sauvegarder les évènements dans un fichier `evenements.json`.
- Recharger les évènements tout en indiquant leur type (`concert`, `conference`).

Les annotations utilisées sont :

- `@JsonTypeInfo` : pour inclure le champ `"type"` dans le JSON.
- `@JsonSubTypes` : pour spécifier les sous-classes connues.
- `JavaTimeModule` : pour permettre la gestion des types `LocalDateTime`.

Pourquoi JSON plutôt que JAXB ?

La préférence de la sérialisation JSON par rapport à XML avec JAXB repose principalement sur :

- La lisibilité : JSON est plus simple à lire.
- La légèreté : les fichiers JSON sont moins verbeux.
- L'usage moderne : JSON est le format standard dans les architectures modernes (REST, Web, etc.).

Conclusion

Nous voyons donc clairement que ce projet intègre des principes solides tels que la ségrégation d'interface et bien d'autres. Il inclut également des concepts fortement liés à la programmation orientée objet (POO) comme présenté plus haut, ainsi que des bibliothèques modernes telles que Jackson.