1. TP 3 PIZZERIA

2. INTERFACE ET FACTORY

Objectifs du TP

L'objectif de ce TP va être de mettre en œuvre plusieurs concepts :

- mettre en place une couche dite « de persistence »,
- mettre en place une couche dite « de service »,
- mettre en place le design pattern Factory.

Mise en place de la couche de persistence

En programmation, on désigne par le mot « persistence » ce qui concerne le stockage des informations. Généralement le stockage se fait dans une base de données pour les applications d'entreprises mais il peut arriver, rarement, que cela prenne d'autres formes : cache mémoire distribué ou fichier XML par exemple.

Une classe qui réalise des opérations de persistence est souvent suffixé par **Dao** (Data Access Object).

Exemple : Pizza**Dao**

Consignes :

- Créer une classe **PizzaMemDao** qui implémente l'interface ci-dessous :

```
public interface IPizzaDao {

	Pizza[] findAllPizzas();
	void saveNewPizza(Pizza pizza);
	void updatePizza(String codePizza, Pizza pizza);
	void deletePizza(String codePizza);
	Pizza findPizzaByCode(String codePizza);
	boolean pizzaExists(String codePizza);
}
```

- La classe **PizzaMemDao** est **la seule** à posséder une référence sur le tableau de pizzas.

- **Attention** : c'est désormais cette classe qui initialise le tableau de pizzas.

- Réorganiser le code de manière à utiliser cette classe pour toutes les opérations de CRUD sur le tableau.

- **Attention** : le code concernant les questions posées à l'utilisateur ne doit pas se retrouver dans cette classe. Il faut bien séparer les responsabilités.

Mise en place de la couche de services

- La classe principale et la méthode main constitue ce qu'on appelle le « Contrôleur ». En POO un contrôleur n'a pas la responsabilité d'exécuter la totalité du code de tous les cas d'utilisation.
    - Un contrôleur va jouer les aiguillages pour choisir la couche de services à exécuter en fonction d'un choix effectué par l'utilisateur.
    - Dans cette application il y a 5 cas d'utilisation
- Créer une classe abstraite MenuService avec une méthode abstraite :

  void executeUC(…)

  La liste des paramètres de la méthode est à définir selon vos besoins.

  - Créer 4 classes qui héritent de MenuService:
    - ListerPizzasService
    - AjouterPizzaService
    - ModifierPizzaService
    - SupprimerPizzaService
  - Pour chacune de ces classes
    - implémenter la méthode executeUC(…) avec la totalité du cas d'utilisation correspondant
  - Réorganiser le code du contrôleur afin d'utiliser ces classes de service.

Mise en place d'une « Factory »

En POO, la **factory** est un design pattern.

Ce design pattern est utilisé dans le cas où on a une classe mère avec de nombreuses classes filles. La factory va posséder une méthode qui va retourner une instance d'une des classes filles en fonction d'un paramètre. Le paramètre peut être un entier par exemple.

Dans le cas présent on va réaliser une classe MenuServiceFactory qui va instancier une classe fille de MenuService en fonction d'un paramètre.

Dans le cas présent, ce paramètre va être le choix réalisé par l'utilisateur dans le menu :

- Créer la classe **MenuServiceFactory**
- Créer cette méthode qui retourne une instance d'une des classes de services ci-dessus en fonction du paramètre de menu.
- Réorganiser le code afin d'utiliser cette Factory et de simplifier au maximum votre **contrôleur**.

Commitez vos développements sur GitHub