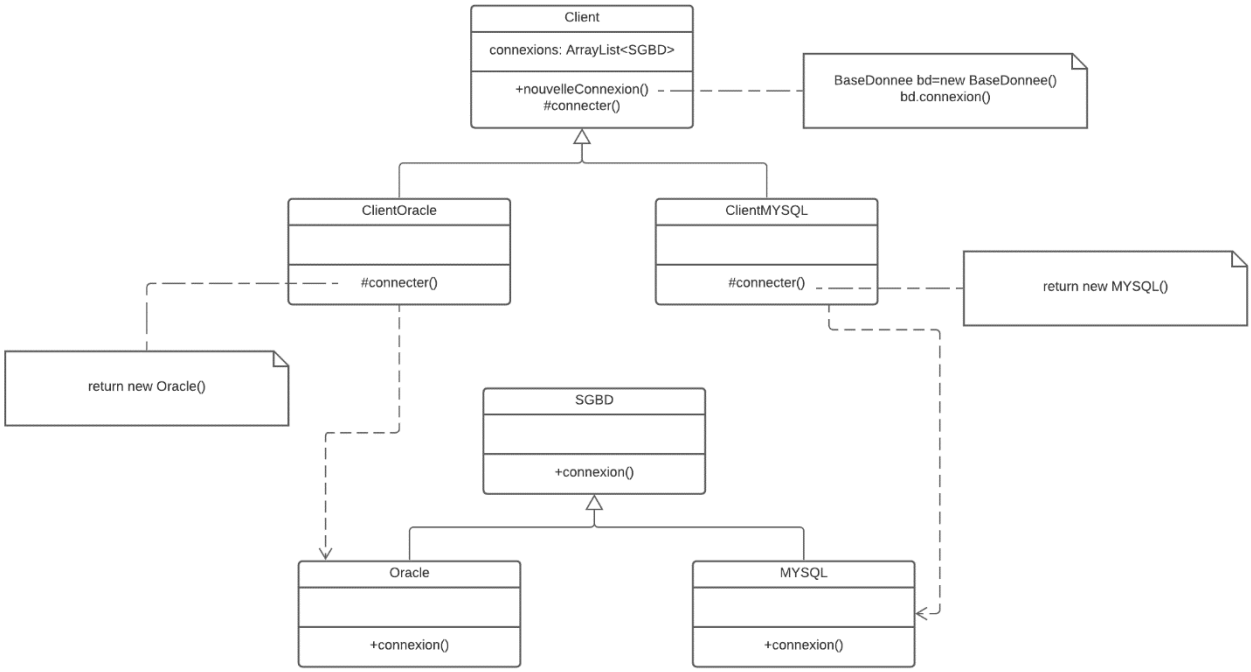
Bellahsen Hamza

Exercice Pratique 6 :

Modèle : Connexion

Un serveur dispose de deux types de bases de données : des bases de données oracle et des bases de données MYSQL. Lorsqu'un client se connecte, le SGBD crée un objet de connexion pour gérer la communication entre celui-ci et la base de données. Deux types de connexions sont donc disponible sur le serveur : les connexions **Oracle** et les connexions **MYSQL**.

1. Quel patron de conception permet de modéliser la création d'une connexion selon le type de la base de données à laquelle on se connecte ?
   Le patron Factory Method permet de modéliser cette situation. En effet, lorsqu'un client se connecte, deux types de connexions peuvent être effectués (connexion Oracle ou connexion MYSQL), sachant que chacune des classes Oracle et MYSQL possèdent une méthode connexion, ce qui cause la duplication de code. Il se peut même que l'on ait un nouveau type de connexion. Ainsi, grâce au patron Factory Method, il y aura une seule classe qui se charge de la connexion, peu importe le type, ce qui permettra d'éviter la duplication de code. De plus, le client interagira seulement avec la fabrique de méthode au lieu d'interagir avec toutes les classes qui représentent les types de connexion.
2. Donnez une brève explication du rôle des classes du patron et leur équivalent dans la forme générique du patron.



Client: elle contient une méthode **connecter()** qui permet de créer une connexion SGBD. Un client peut faire une connexion SGBD de type Oracle ou MYSQL. Ainsi, cette méthode crée une nouvelle instance de la classe Oracle ou MYSQL. Elle est donc abstraite. La classe Client est donc un créateur abstrait.

ClientOracle : il s'agit d'un créateur concret qui représente le client qui se connecte à une base de données Oracle. Elle implémente la méthode **connecter()** afin de créer une instance de la classe Oracle.

ClientMYSQL : il s'agit d'un créateur concret qui représente le client qui se connecte à une base de données MYSQL. Elle implémente la méthode **connecter()** afin de créer une instance de la classe MYSQL.

SGBD : il s'agit d'une classe abstraite qui décrit les propriétés communes des classes MYSQL et Oracle. C'est la classe produit.

Oracle : il s'agit d'une classe produit concret. Elle décrit le comportement d'une connexion à la base de données Oracle

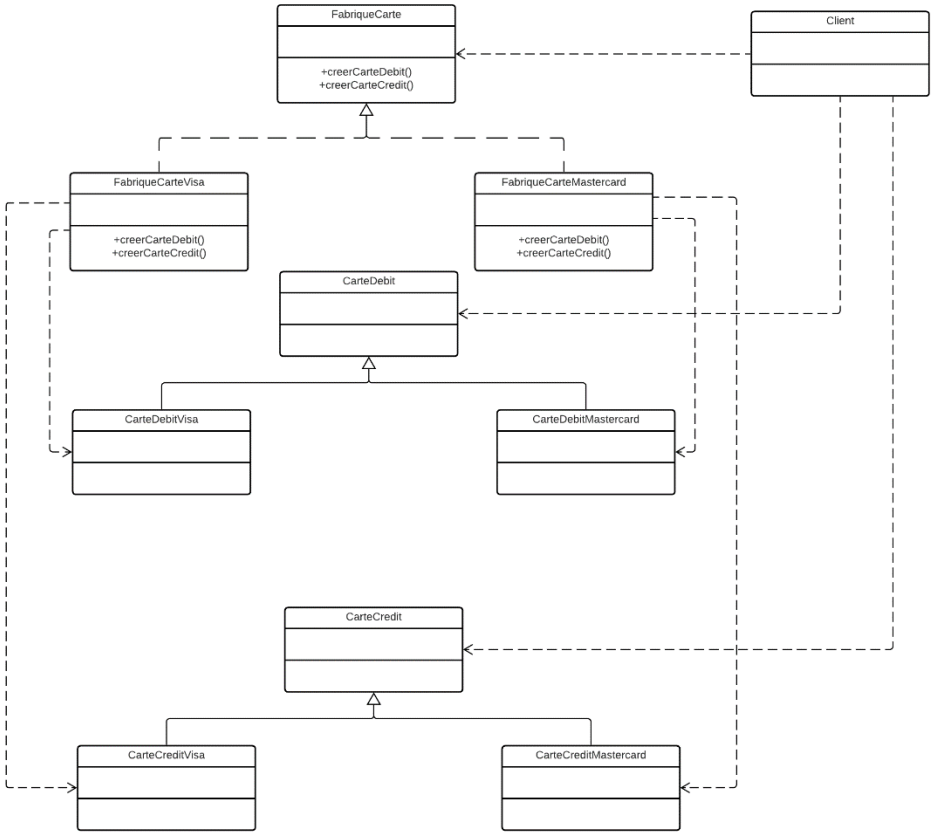
MYSQL : il s'agit d'une classe produit concret. Elle décrit le comportement d'une connexion à la base de données MYSQL

3. Définissez en Java les différentes classes utiles à la création des connexions puis testez leur fonctionnement. Le SGBD stocke les connexions dans un **ArrayList**

Module : Création à l'aide d'une fabrique

Il existe deux modèles de cartes de débit et de crédit, à savoir les cartes Visa et les cartes MasterCard.

1. Modélisez, à l'aide d'un diagramme de classes, la création d'une carte de paiement en fonction de sa famille (de crédit ou de débit) en utilisant le pattern Abstract Factory.



2. Donnez une brève explication du rôle des classes du patron et leur équivalent dans la forme générique du patron.

FabriqueCarte : il s'agit d'une interface qui contient les signatures des méthodes qui vont créer les différentes cartes (Fabrique Abstraite)

FabriqueCarteVisa: il s'agit d'une classe concrète qui implémente les méthodes de l'interface FabriqueCarte afin de créer des cartes de débit ou de crédit **Visa**. (Fabrique Concrète)

FabriqueCarteMastercard: il s'agit d'une classe concrète qui implémente les méthodes de l'interface FabriqueCarte afin de créer des cartes de débit ou de crédit **Mastercard**. (Fabrique Concrète)

CarteDébit : classe abstraite dont héritent les sous-classes concrètes **CarteDébitVisa** et **CarteDébitMastercard.** (Produit)

CarteCrédit : classe abstraite dont héritent les sous-classes concrètes **CarteCréditVisa** et **CarteCréditMastercard.** (Produit)

Client : classe utilisant l'interface FabriqueCarte