Travail Pratique 2

Cours : 420-D13-MA Programmation Objet et patrons de conception

Groupe : 0001

Date de remise : 1 décembre 2020(vous avez 5% de moins pour chaque jour de retard)

Pondération : 15 %

Professeur : Yousra Tagmouti

Bonne Chance ☺

Nom : Bellahsen Hamza

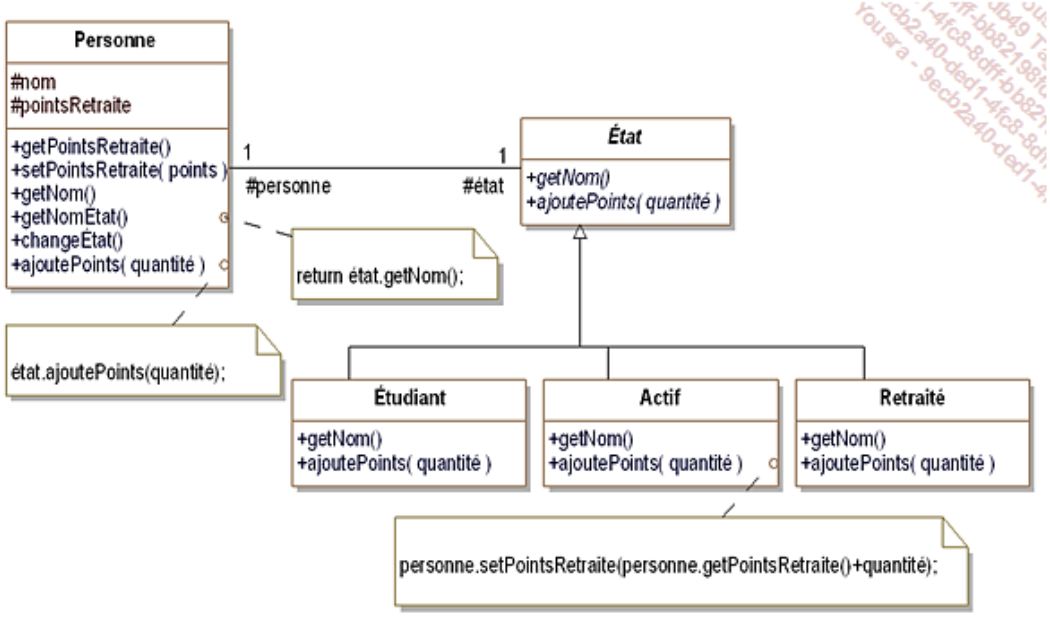
Modèle 1 :

États de la vie professionnelle d'une personne

Une personne possède un cycle au long de sa vie professionnelle. Elle est d'abord étudiante, puis intègre la vie active puis prend sa retraite.

Son comportement varie en fonction de l'état où elle se trouve. Notamment, elle ne cotise pour sa retraite que lors de sa vie active. Les cotisations pour la retraite donnent lieu à l'ajout de points de retraite.

- La figure ci-dessous permet de décrire une personne au long de sa vie professionnelle avec ses différents états.



- Le comportement des méthodes **`getNom`** et **`ajoutePoints`** dépend de l'état de la personne.
- La méthode **`getNom`** renvoie le nom de l'état de la personne.
- La méthode **`ajoutePoints`** ajoute des points de retraite. La méthode **`ajoutePoints`** ne possède un comportement que dans la vie active. Elle utilise la méthode **`setPointsRetraite`** de la classe **`Personne`** dont l'usage est réservé à la classe **`État`** et à ses sous-classes.

Remarque : le modèle tel quel est basé sur le patron **State**.

Il s'agit maintenant de créer une **interface graphique** qui affiche la personne ainsi que son état professionnel. Chaque fois que des données de la personne ou de son état sont changées, les données affichées dans l'interface graphique sont mises à jour automatiquement. Il s'agit ici du nom de l'état et des points de retraite.

1. Quel patron est le mieux adapté pour concevoir cette interface graphique ? Justifiez brièvement votre réponse.

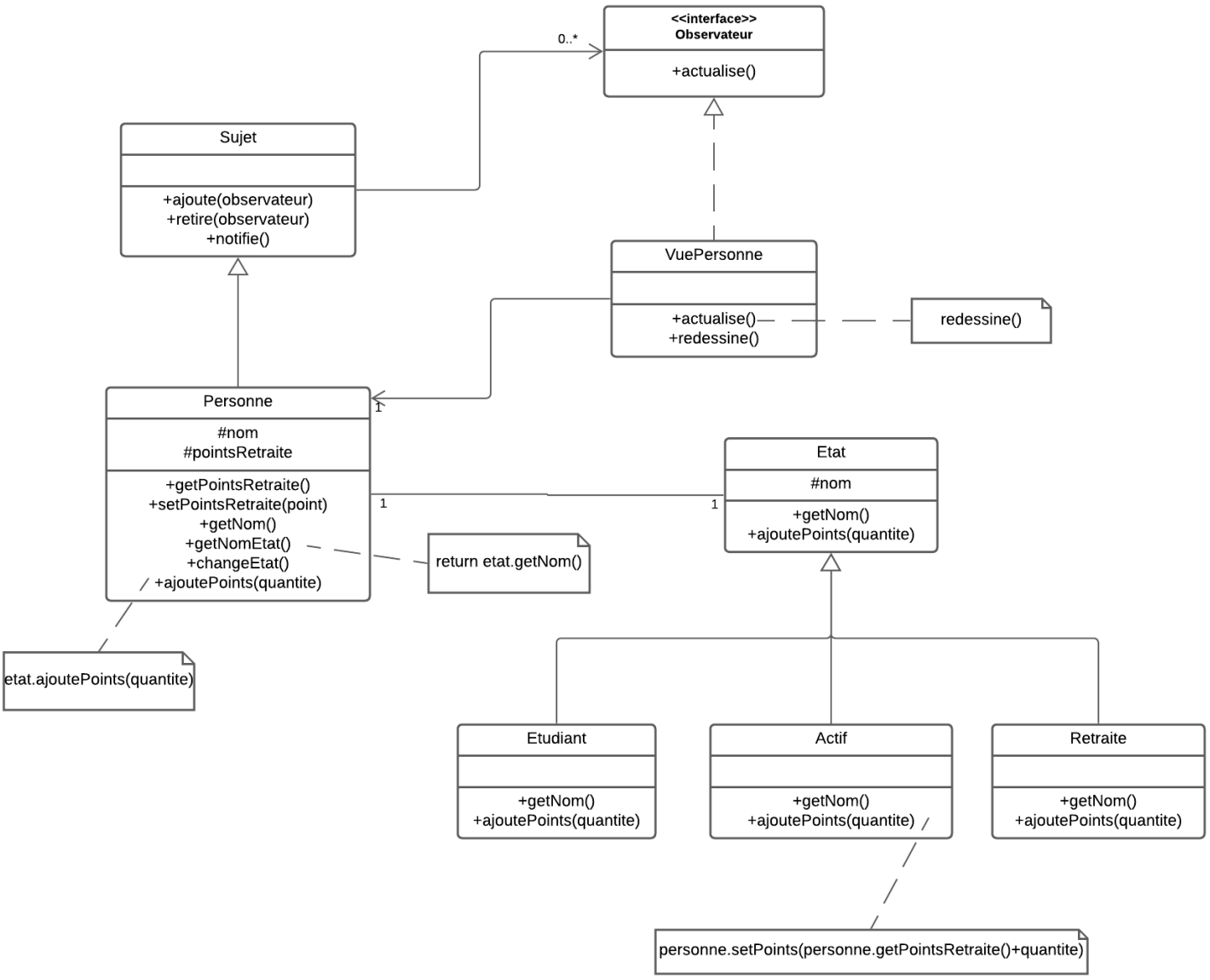
   Le patron observateur est le mieux adapté pour cette situation, puisqu'il permet de créer une **dépendance entre un sujet et des observateurs** afin que chaque modification de ce sujet soit **notifiée aux observateurs** afin qu'ils mettent à jour leur état. Ainsi, lors du changement d'état ou du nombre de points de retraite d'une personne, les données affichées seront mises à jour automatiquement.

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

   - **État** est d'une classe **abstraite** surclasse des classes d'états. Elle contient deux méthodes abstraites : **getNom** qui retourne le nom de l'état et **ajoutePoints** qui permet l'ajout de points de retraite **(seulement si la personne est à l'état actif).**
   - **Actif, Etudiant et Retraite** sont des classes **concrètes** qui héritent de la classe État. Elles implémentent la méthode getNom qui permet de retourner le nom de l'état et ajoutPoints qui permet l'ajout de points de retraite, **seulement si une personne est à l'état actif**. Dans les autres cas, la méthode ajoutPoints n'a aucun comportement.
   - **Sujet** est une classe abstraite qui introduit tout objet qui **notifie** d'autres objets des modifications de son état interne.
   - **Personne** contient une **méthode changeEtat** qui lui permet de changer d'état et **une méthode ajoutePoints** qui permet l'ajout de points de retraite C'est aussi la sous-classe de la classe Sujet qui décrit les personnes.

- **Observateur** est l'interface des objets qui a besoin d'être notifiée lors des changements d'états provenant des objets auprès desquels il s'est préalablement inscrit.
- **VuePersonne** est la sous-classe concrète implémentant l'interface **Observateur** dont les instances affichent les informations d'une personne.

3. Concevez le diagramme de classes correspondant à cette solution à partir du modèle ci-dessous.



4. Concevez un programme Java de votre application.

   Package « numero1 » dans le projet Java

Modèle 2 :

La banque

Partie 1

Les clients d'une banque sont classés en deux catégories :

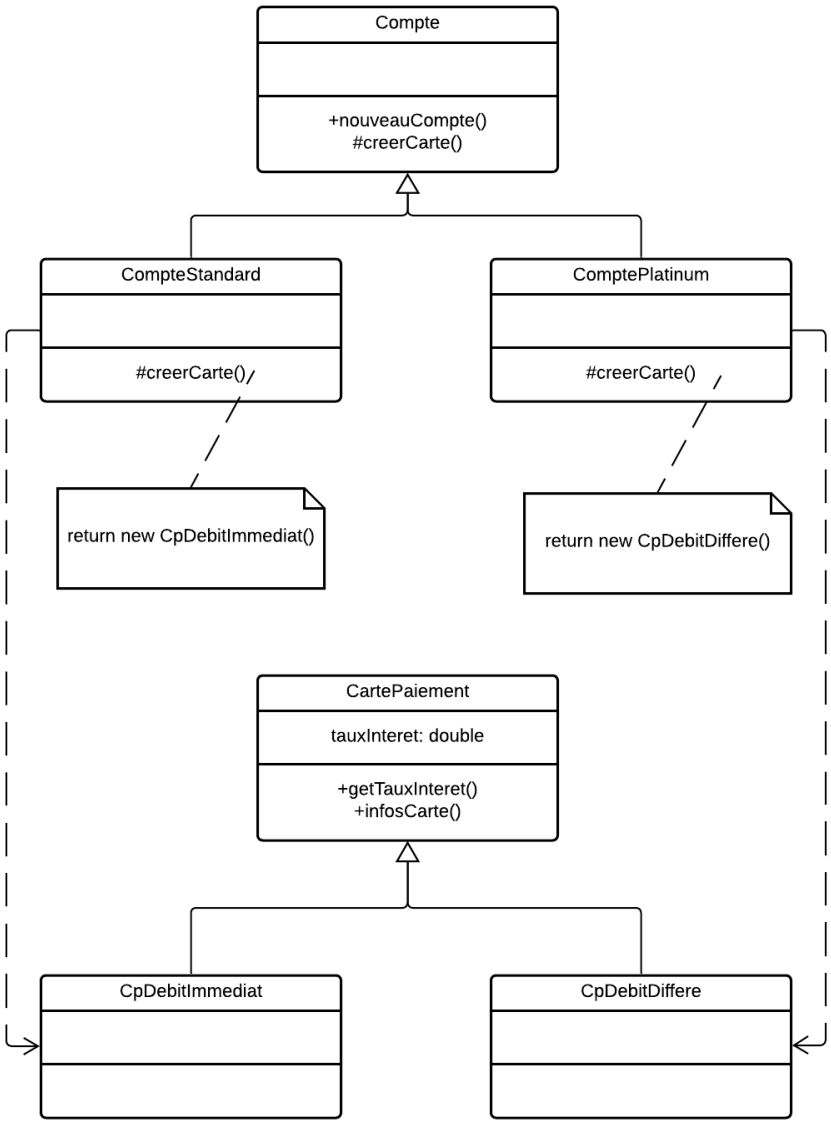
1. Ceux qui ont le droit à un compte standard avec un taux d'intérêt de 2%
2. Ceux qui ont le droit à un compte Platinum avec un taux d'intérêt de 0.9%

Lors de la demande de la création d'un compte, les premiers reçoivent une carte de paiement (à débit immédiat sur leur compte) alors que les seconds peuvent avoir une carte de paiement à (débit différé sur leur compte).

1. Quel patron de conception permet-il de modéliser la création de la carte de paiement en fonction du client ?
   Le patron **Factory Method** permet de modéliser cette situation, car il permet il définit une interface permettent la création d'objets dérivant d'une classe mère (**les cartes de paiements**) dans une Fabrique (**Compte créé**)
2. Donnez une brève explication du rôle des classes du patron et leur équivalent dans la forme générique du patron.
   - **Compte** représente le **créateur abstrait**. Il s'agit d'une classe abstraite qui introduit **la méthode de fabrique creerCarte** permettant de créer une une carte de paiement.
   - **CompteStandard** représente **un créateur concret**. Il s'agit d'une classe concrète qui implante la méthode creerCarte afin de **créer une carte de paiement à débit immédiat**.
   - **ComptePlatinum** représente un **créateur concret**. Il s'agit d'une classe concrète qui implante la méthode creerCarte afin de **créer une carte de paiement à débit différé**.

- **Carte** représente un produit. Il s'agit d'une **classe abstraite** qui décrit les propriétés communes des produits.
- **CpDébitImmédiat** représente un produit concret. Il s'agit d'une classe concrète qui **dérive de la classe Carte** et elle décrit le comportement de la **création d'une carte de paiement à débit immédiat**.
- **CpDébitDifféré** représente un produit concret. Il s'agit d'une classe concrète qui **dérive de la classe Carte** et elle décrit le comportement de la **création d'une carte de paiement à débit différé**.

3. Modélisez son utilisation par un diagramme de classes.



4. Concevez un programme Java de votre application, afin de tester le fonctionnement de votre patron.

Package numero 2 dans le projet

Partie2 :

Notre banque veut ouvrir une succursale en suisse et offre à ses clients en suisse uniquement une nouvelle catégorie compte Prestige, avec des avantages très différents des comptes standards et Platinum (découvert, taux d'intérêt, conseillers de voyage, des personal shopper, des invitations pour des événements exclusifs, etc.)

1. Quel patron de conception permet l'intégration de ce nouveau type de compte dans notre modèle

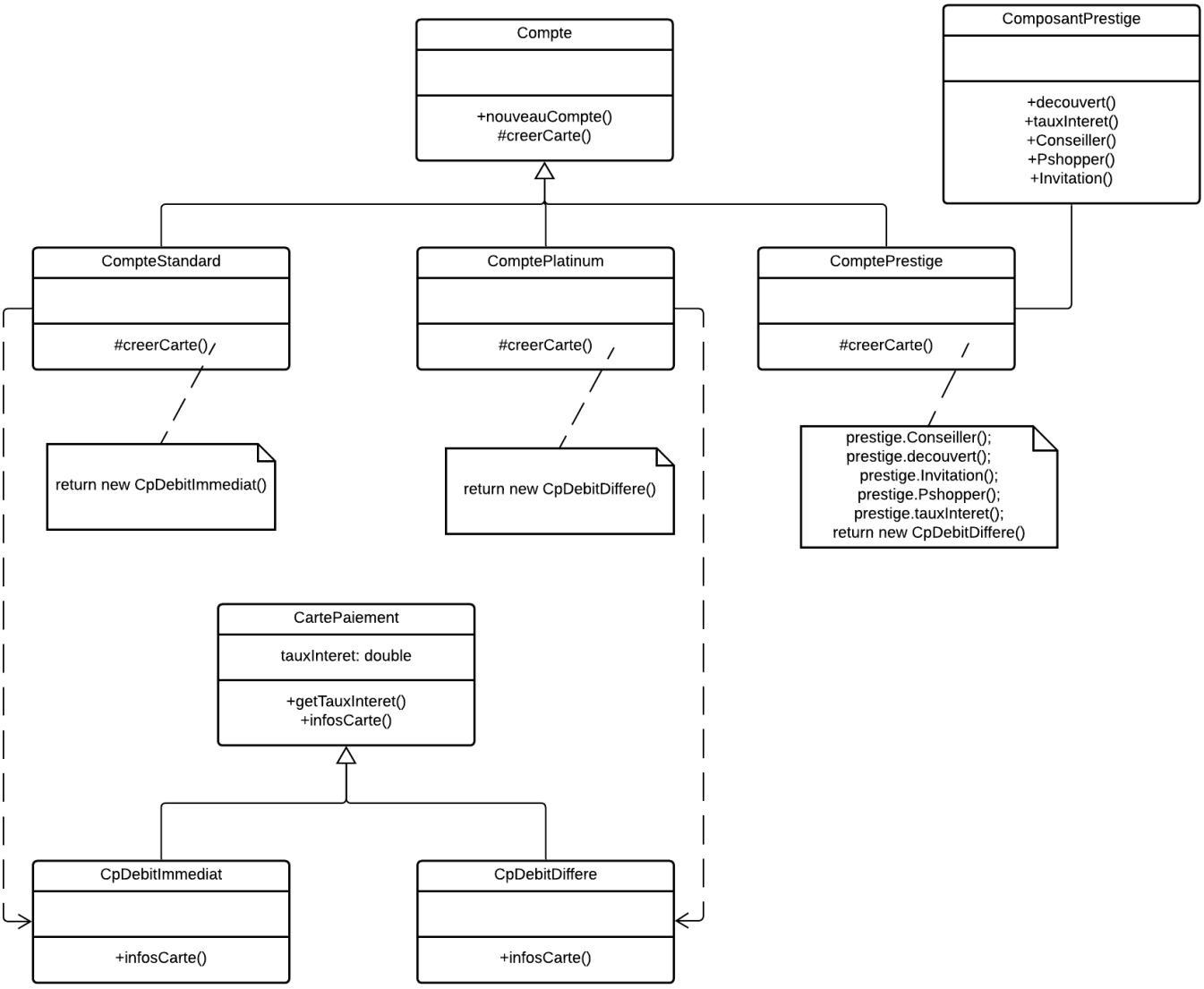
   Le patron **Adaptateur** permet d'intégrer ce nouveau type de compte dans notre modèle initial en **convertissant l'interface déjà existante** en l'interface attendue par un autre type de clients, soit **ceux en Suisse qui ont un compte Prestige**. Ainsi, ce patron de conception nous permet d'adapter notre modèle initial à leurs besoins sans le changer.
2. Donnez une brève explication du rôle des classes du patron et leur équivalent dans la forme générique du patron.

   Classes ajoutées :

   ComptePrestige représente **l'adaptateur**. Il s'agit d'une classe concrète qui représente **les comptes bancaires prestige** que peuvent avoir les clients en Suisse et elle **implante la classe Compte**. Elle est aussi associée l'objet adapté, soit **ComposantPrestige.**

   ComposantPrestige représente **l'adapté.** C'est une classe associée à **ComptePrestige** qui contient les différentes méthodes qui concernent les comptes prestiges. Donc, elle introduit la catégorie prestige dont l'interface doit être adaptée pour correspondre à Compte

3. Modélisez son utilisation par un diagramme de classes.



4. Concevez un programme Java de votre application, afin de tester le fonctionnement de votre patron.

Package numero2 dans le projet Java (ajout de deux classes)