



Le génie pour l'industrie

LOG121

Conception orientée objet

Groupe 3
Hiver 2020

Patron Memento et retour sur MVC

Chargé de cours: Cédric St-Onge

- Patron Memento

- Retour sur l'architecture MVC

Exemple de problème de conception

3

- Un gestionnaire simplifié de transaction permet d'exécuter des transactions sur les tables d'une base de données (BD)
- Une transaction peut contenir plusieurs opérations (update, delete, insert) agissant sur les tables de la BD
 - ▣ Exemple: un client qui transfère de l'argent de son compte courant à un compte d'épargne
 - Cette transaction doit mettre à jour les deux comptes en conséquence et enregistrer la transaction dans un journal

Exemple de problème de conception

4

□ Notre exemple de transaction aura la forme:

UPDATE compte-courant

SET solde = solde - 50

WHERE num-compte = 81566979

UPDATE compte-hecke

SET solde = solde + 50

WHERE num-compte = 81566988

INSERT INTO journal-transactions VALUES

(num_transaction, type_transaction, 81566979,
81566988, 50)

COMMIT WORK;

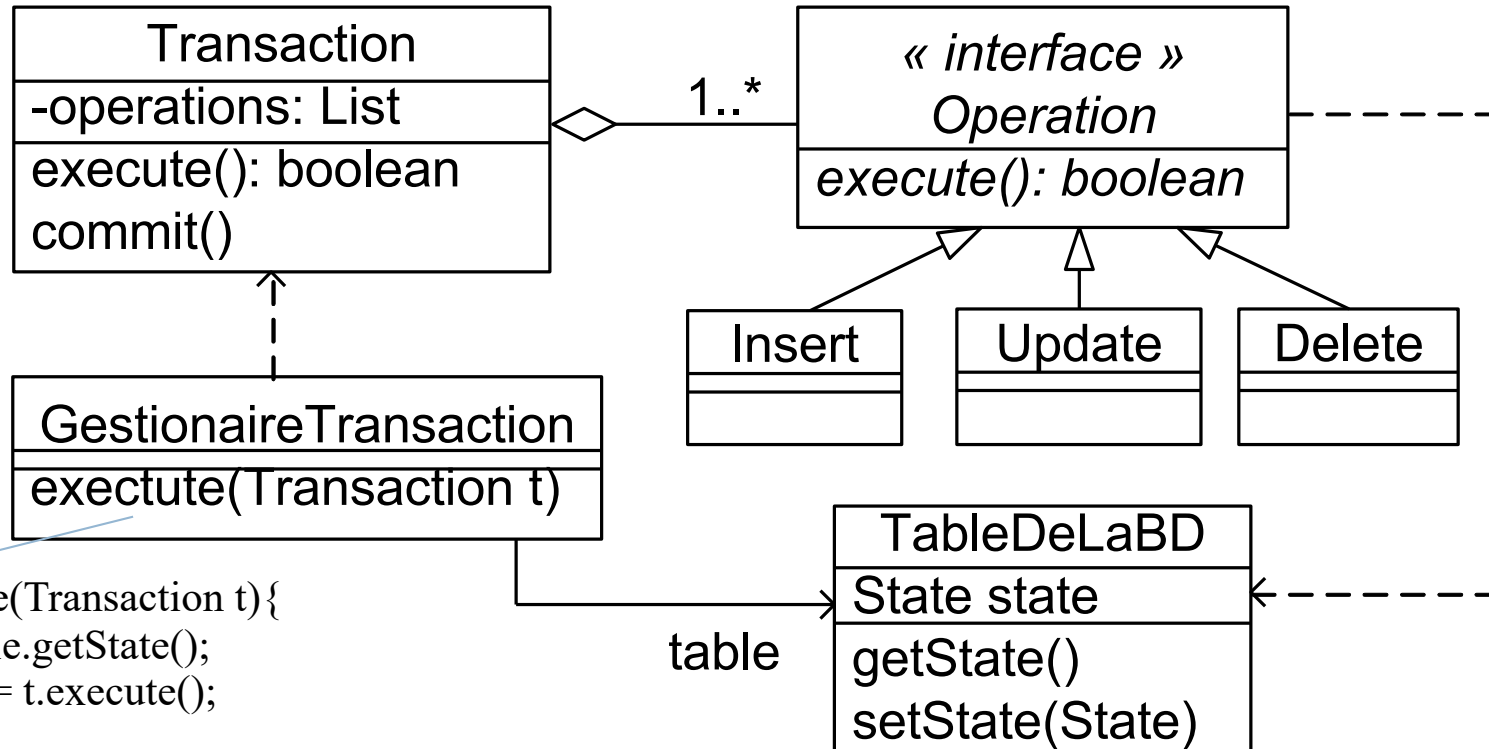
Exemple de problème de conception

5

- Une transaction a la propriété d'être ACID
 - ▣ A: Atomique, C: Cohérence, I: Isolée et D: Durable
 - ▣ Si toutes les opérations d'une transaction sont exécutées avec succès, le gestionnaire appelle la méthode « commit » de la transaction
 - ▣ Si une opération échoue, toute la transaction doit être annulée. Dans ce cas, les tables concernées par la transaction doivent retourner à leur état avant le début de la transaction

Exemple de problème de conception

6



```
public void execute(Transaction t){
    State old = table.getState();
    boolean succes = t.execute();
    if (success) {
        t.commit();
    } else {
        table.setState(old);
    }
}
```

□ Quel est le problème avec cette conception?

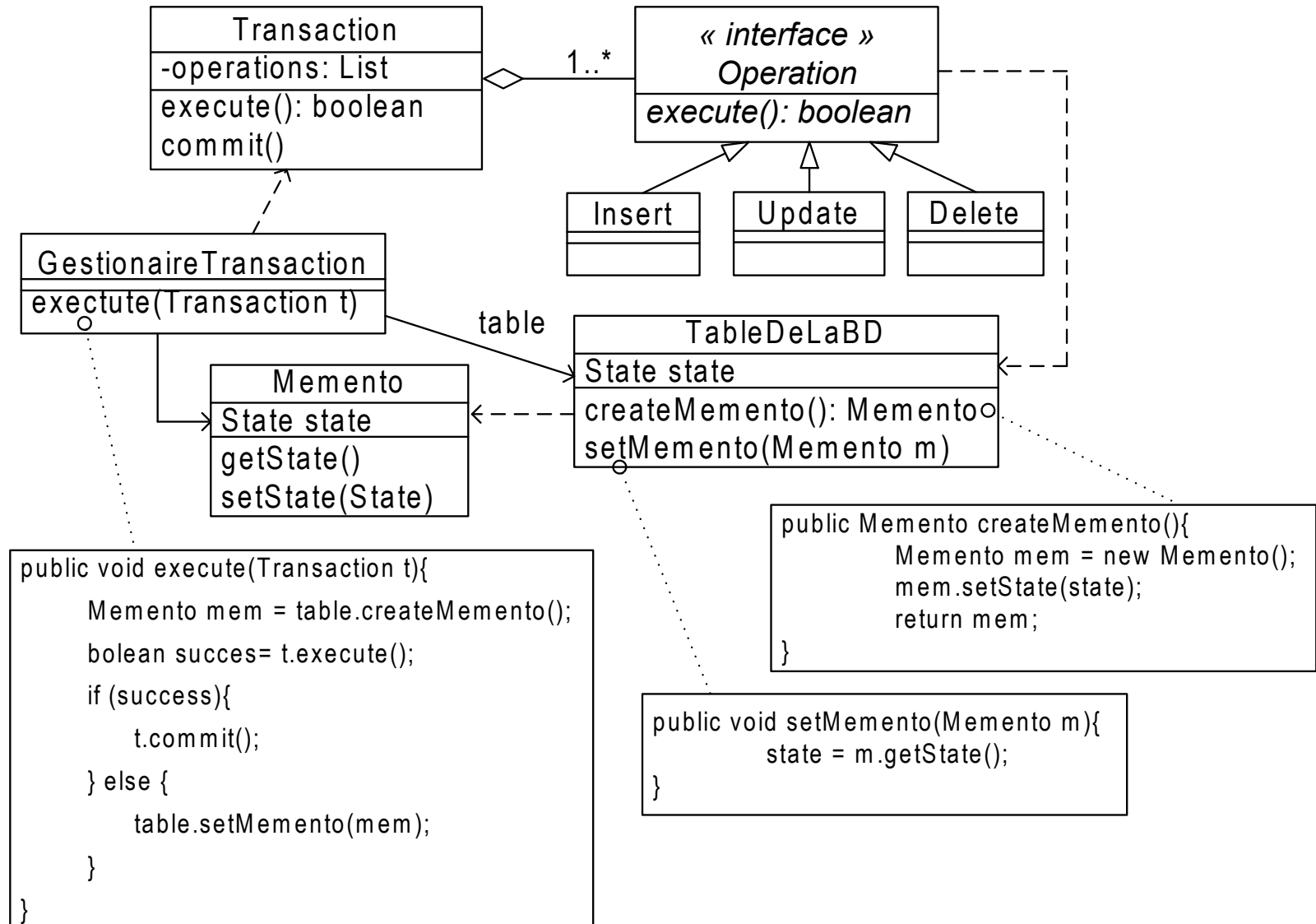
Solution au problème

7

- On crée un objet (Memento) qui garde trace de l'état d'un objet de type TableDeLaBD.
- Avant d'exécuter une transaction, le gestionnaire de transaction demande à un objet TableDeLaBD son objet Memento.
- Un objet TableDeLaBD est le seul responsable de créer son Memento, de l'initialiser et de le mettre à jour pour qu'il garde une copie de l'état courant de la table.
- Lorsqu'une transaction échoue, le gestionnaire de transaction renvoie le Memento à l'objet TableDeLaBD
- L'objet TableDeLaBD utilise le Memento reçu pour récupérer l'information sur son état précédent.

Solution au problème

8



□ Contexte

- ▣ L'état (ou une partie de l'état) d'un objet doit être sauvegardé pour que l'objet puisse retourner à cet état en cas de besoin
- ▣ Un accès direct à l'état de l'objet exposerait des détails d'implémentation et violerait l'encapsulation de l'objet

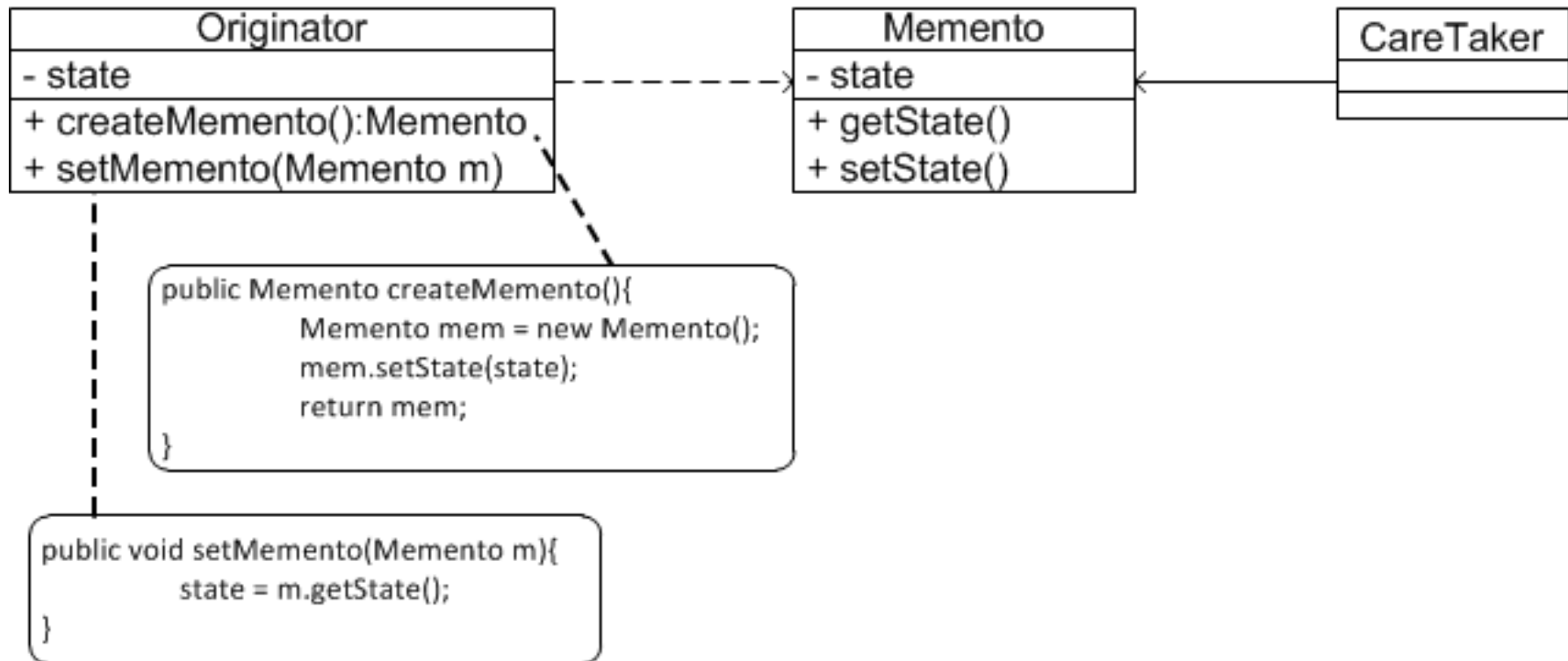
□ Solution

- On crée un objet Memento (aussi appelé Snapshot) qui garde trace de l'état de l'objet initial (Originator). Selon le contexte, le Memento peut garder juste une partie de l'état de l'objet initial.
- L'objet Originator est le seul responsable de créer son Memento, de l'initialiser et de le mettre à jour.
- Un objet (Caretaker) qui utilise l'objet original est responsable de garder une référence sur le Memento et renvoie le Memento à l'objet original en cas de besoin.
- L'objet Caretaker ne change pas l'objet Memento.

Patron Memento

11

□ La structure du patron dans GoF



Patrons Memento et Commande

12

□ Que peuvent-ils accomplir ensemble?

☐ Patron Memento

☐ Retour sur l'architecture MVC

Retour sur l'architecture Model-View-Controller

14

- Exigences d'une application interactive*
 - ▣ La même information peut être présentée différemment aux utilisateurs
 - ▣ L'affichage et le comportement de l'application doit immédiatement refléter les manipulations faites sur les données
 - ▣ Les changements des interfaces doivent être faciles
 - Changer le « look and feel » ne devrait pas affecter le noyau de l'application

* «pattern-oriented software architecture», Buschman et al., 1996

Retour sur l'architecture Model-View-Controller

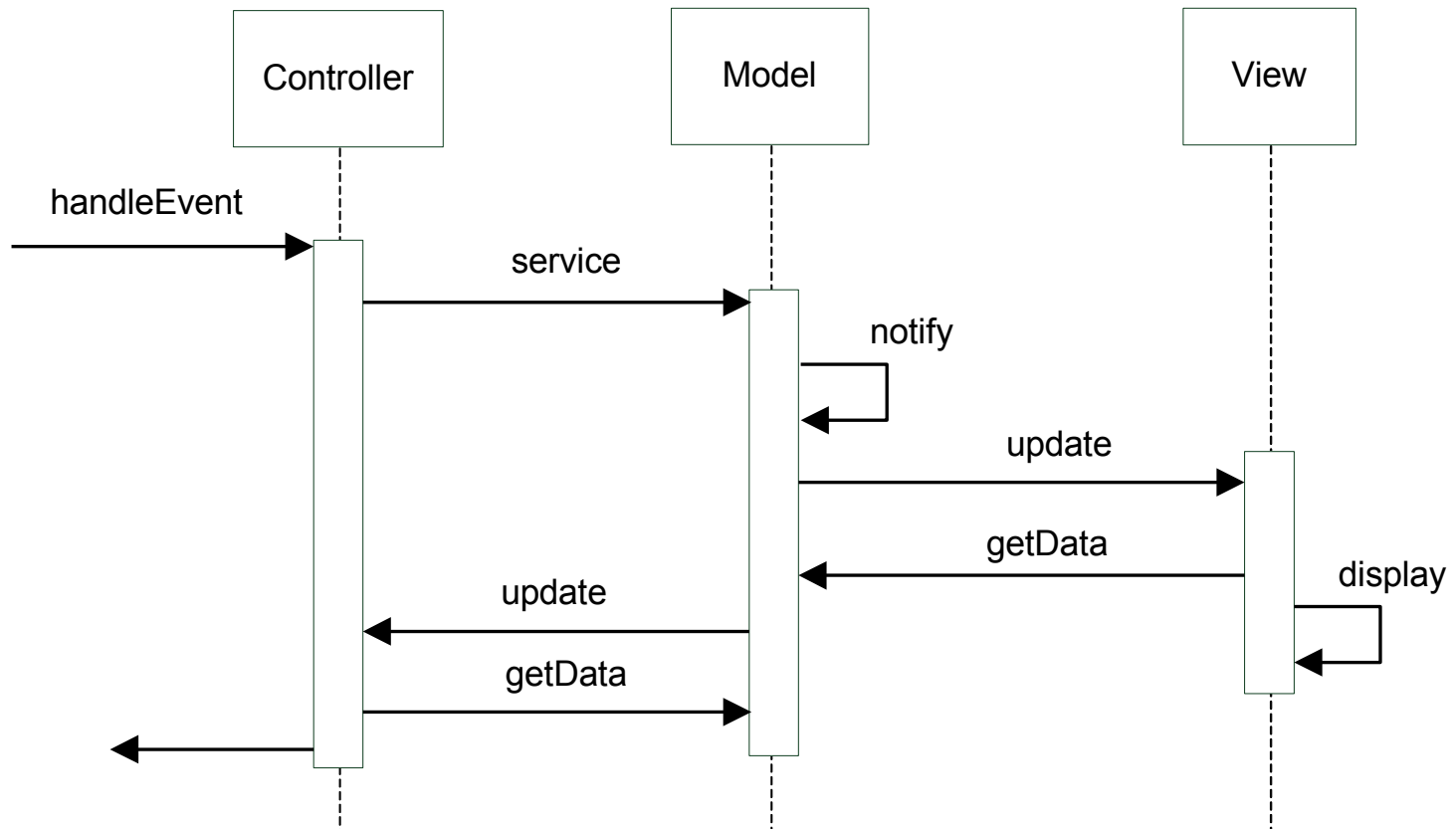
15

- MVC divise une application en trois parties
 - ▣ Le Modèle: il encapsule les données et les fonctions noyau de l'application. Il est indépendant des représentations visuelles.
 - ▣ Les vues: une vue affiche l'information à l'Utilisateur. Elle lit les données à partir du modèle. Une vue correspond aussi à une interface à travers laquelle l'utilisateur déclenche des actions.
 - ▣ Contrôleurs: un contrôleur est associée à chaque vue. Un contrôleur reçoit les entrées sous forme d'événements. Ces événements sont traduits en requêtes sur le modèle ou la vue.

Retour sur l'architecture Model-View-Controller

16

- ❑ Scénario où l'entrée faite par l'utilisateur change l'état du modèle*



* Diagramme adapté de «pattern-oriented software architecture», Buschman et al., 1996

- Éléments minimaux à retenir sur le fonctionnement de MVC
 - ▣ Le modèle doit offrir des méthodes qui permettent aux vues et aux contrôleurs de lire les données.
 - ▣ Durant l'initialisation, chaque vue crée un contrôleur. La vue offre, au besoin, les fonctionnalités qui permettent au contrôleur de manipuler l'affichage.
 - ▣ Le contrôleur reçoit les entrées de l'utilisateur sous forme d'événements qu'il va traduire en demandes de service au modèle.
 - ▣ Lorsque ces demandes de service changent l'état du modèle, toutes les vues reliées à ce modèle doivent se mettre à jour.

- **MVC minimise le couplage** entre le modèle, les vues et les contrôleurs
 - ▣ Le modèle ne connaît rien des vues à part qu'elles doivent être notifiées des changements
 - ▣ Les vues ne connaissent pas les contrôleurs
- ➔ On peut facilement ajouter des vues

- Quels patrons de conception seraient utiles pour implémenter une application organisée selon l'architecture MVC?

□ L'architecture MVC et les patrons de conception

- Les vues doivent s'enregistrer auprès du modèle pour qu'il les notifie en cas de changement.
- Certains contrôleurs ont aussi besoin d'être informés du changement.

□ Patron Observateur

□ L'architecture MVC et les patrons de conception

- Une vue peut être composée d'autres vues

- Patron Composite

- Certains composants graphiques de la vue peuvent nécessiter des comportements additionnels selon leur taille ou forme (ex: scrolling)

- Patron Décorateur

□ L'architecture MVC et les patrons de conception

- Une vue contient des composants graphiques (ex. Barre d'outils) pouvant correspondre à différentes actions ou des actions similaires qu'on veut manipuler, défaire ou refaire.

- Patron Commande

- Patron Memento

- Implémentation de MVC dans Java
 - ▣ Le Modèle: un ensemble de classes.
 - ▣ Les vues: des interfaces avec des composants graphiques (Swing): boutons, zones de textes, etc.
 - ▣ Contrôleurs: Ce sont les listeners (ex: ActionListener)