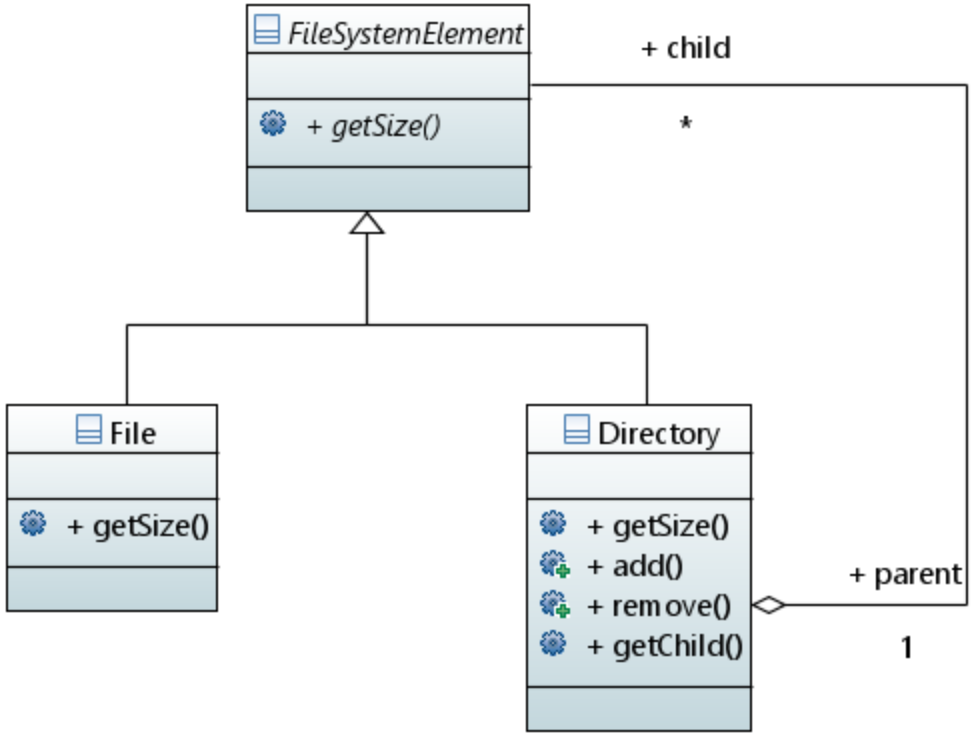
Patrons de Conception

Patrons de Conception pour $200

- Vous travaillez pour un bureau de comptabilité et vous développez un logiciel pour calculer la taille totale du système de fichiers. Le taille (`size`) d'un dossier (`Directory`) est le somme des tailles des fichiers (`File`) dans le dossier.

Solution #1 (composite)

Directory contient File (donc est composé de File)

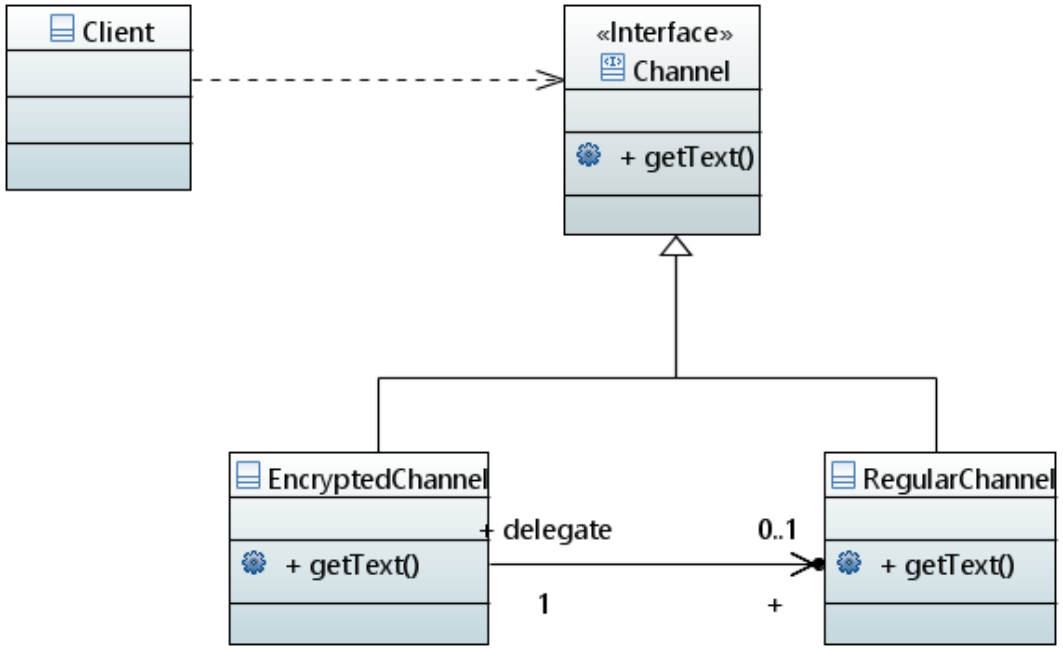


- Vous avez des objets qui doivent être traités de la même manière (c'est-à-dire ajouter de la taille).
- Mais certains d'entre eux sont des *composites* contenant d'autres plus simples. (Les dossiers contiennent des fichiers).
- State/Strategy définissent également des hiérarchies, mais vous n'avez pas la relation composite.

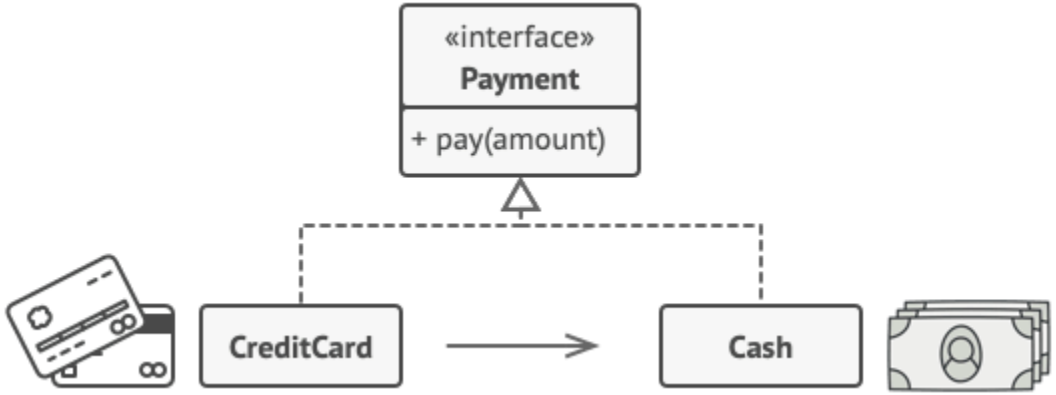
Patrons de Conception pour $400

- Nous avons une configuration où nous envoyons et recevons des données sur le réseau. À un moment donné, le flux de données en entrée commence à être encrypté. Comment doit-on changer le code client?

Solution #2 (proxy)



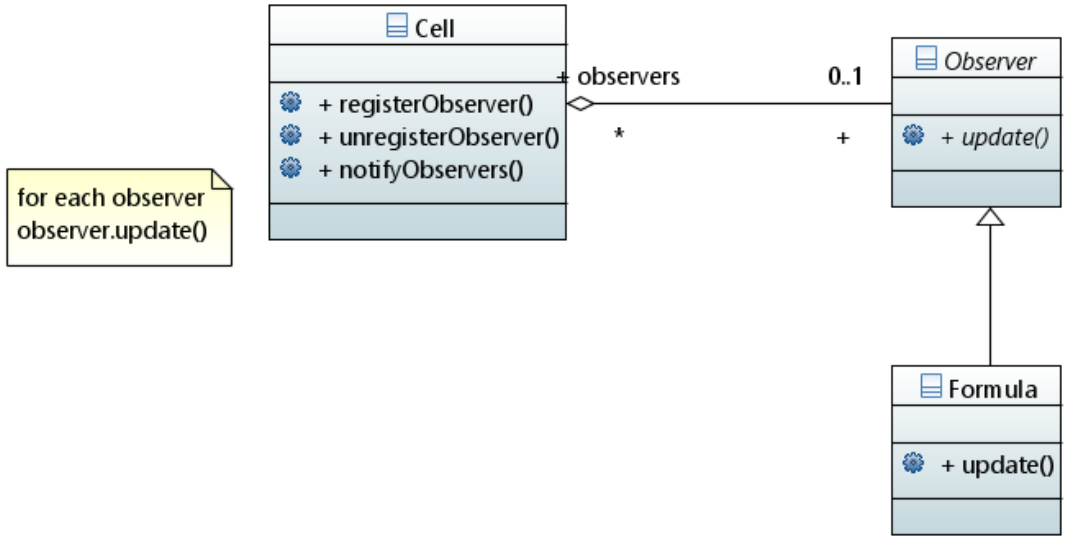
- Vous devez contrôler l'accès à un objet donné (RegularChannel).
- Vous fournissez un espace réservé (Channel) qui détermine dynamiquement comment l'objet sera accédé.



Patrons de Conception pour $600

- Vous développez un tableur qui permet de calculer automatiquement les cellules en fonction des formules, qui dépendent d'autres cellules.

Solution #3 (observer)

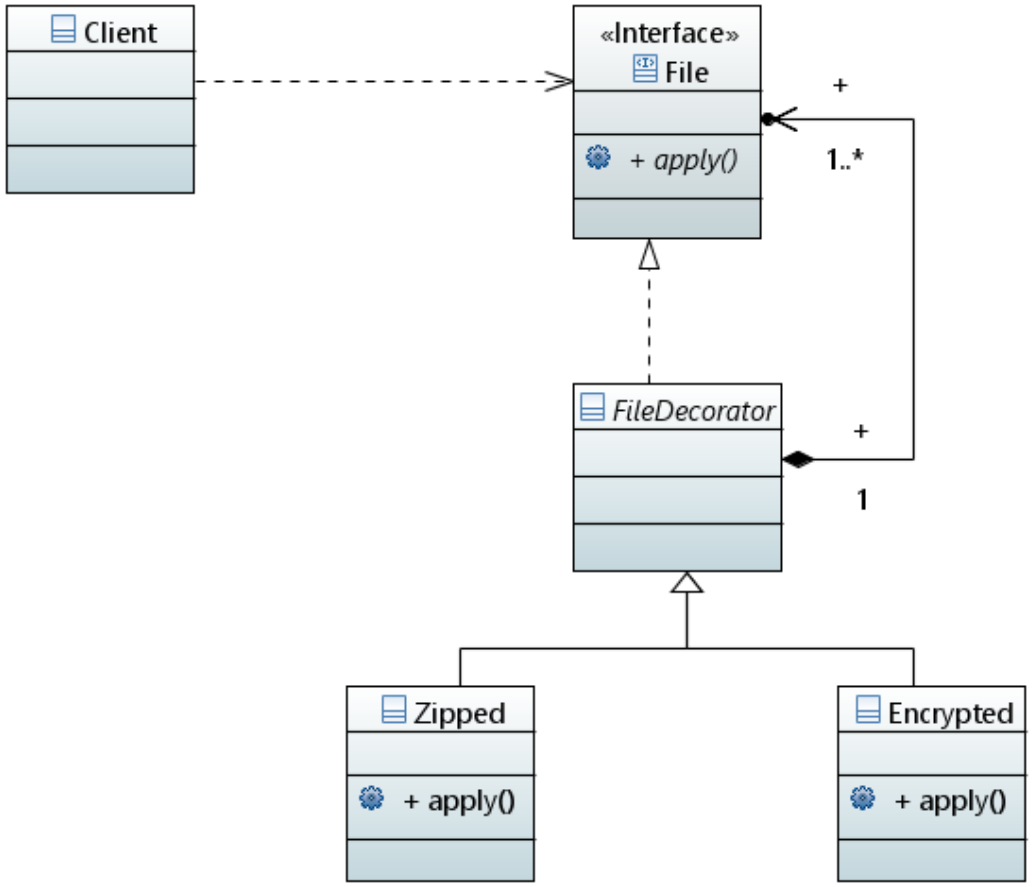


- Un objet (Cellule) doit changer d'état en fonction de ce qui arrive à un autre objet (Formule).
- L'objet qui change est le sujet et l'objet qui doit changer est un observateur qui souscrit au sujet.

Patrons de Conception pour $800

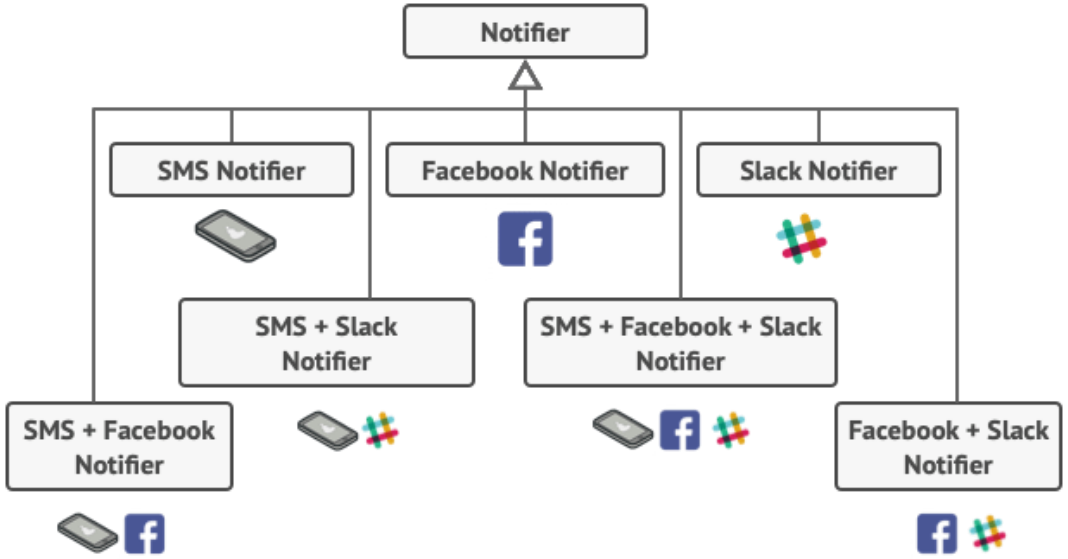
- Vous souhaitez développer un lecteur de fichier capable de lire un fichier, qui peut être (a) compressé, (b) encrypté (c) compressé et encrypté ou (d) encrypté puis compressé et encrypté à nouveau.

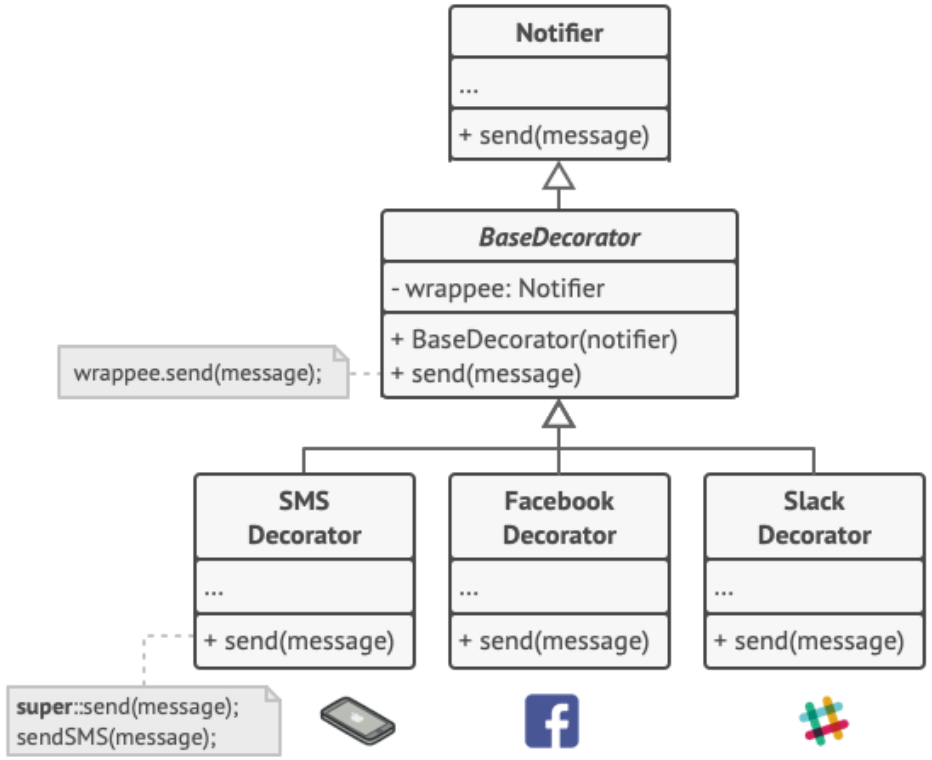
Solution #4 (decorator)



- Vous avez des objets qui peuvent avoir des comportements ou des états différents (tels que compressés ou encryptés).
- En plus de cela, vous voulez que ces états soient combinés de toutes les manières possibles.
- State/Strategy vous obligerait à créer des sous-classes pour toutes les combinaisons possibles et à ne pas tenir compte des extensions futures.

Solution #4 (decorator)

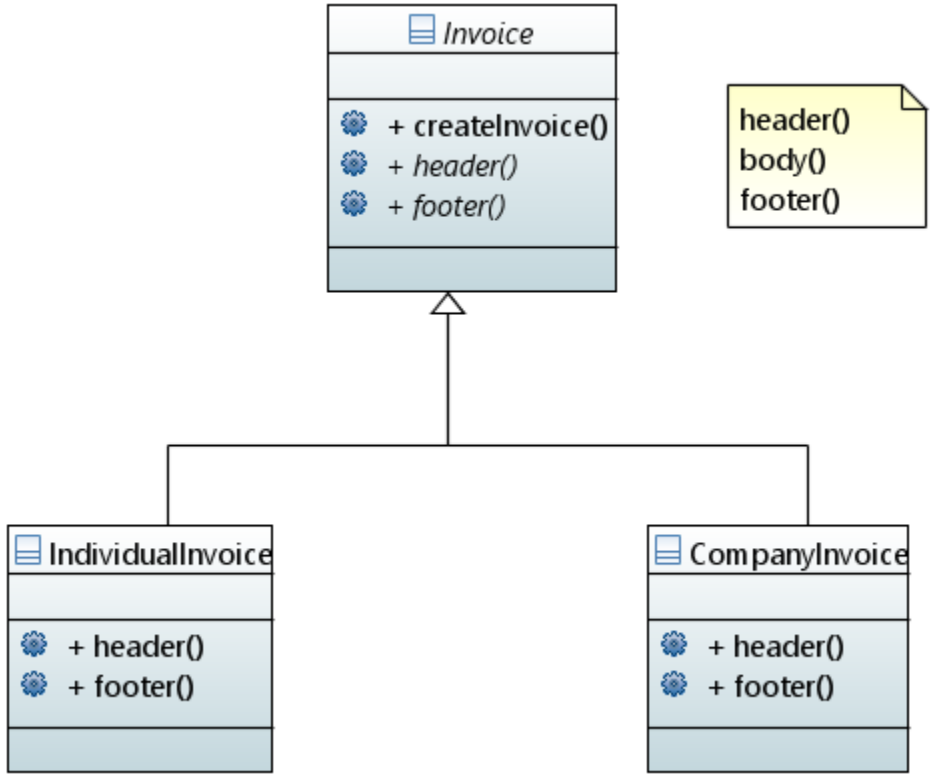




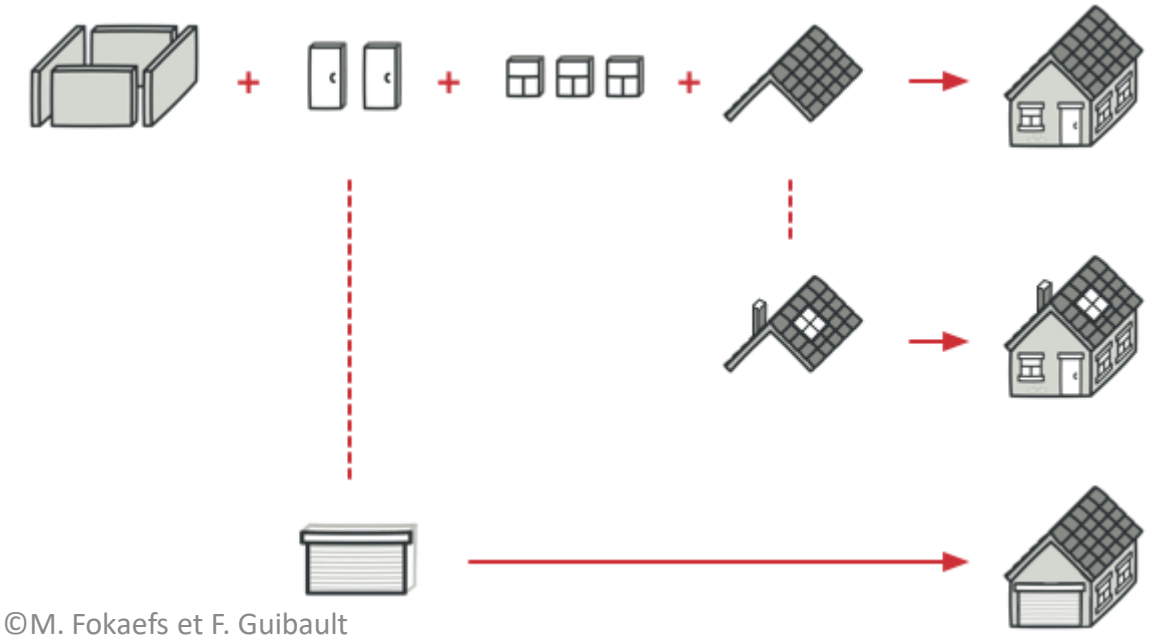
Patrons de Conception pour $1000

- Vous avez un système qui imprime deux types de factures, un pour les individus et un autre pour les compagnies, qui diffèrent entre eux par le graphisme d'en-tête et de pied de page. Le contenu au milieu est une liste de tous les éléments de la facture, leurs prix et le total.

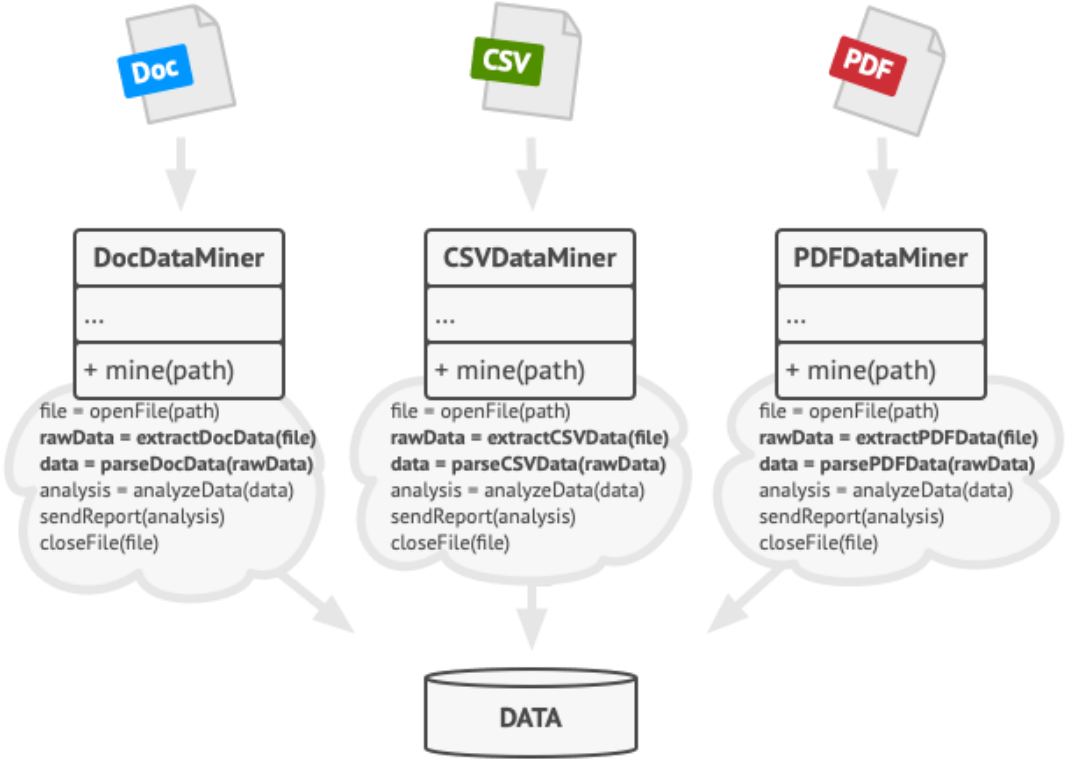
Solution #5 (template method)

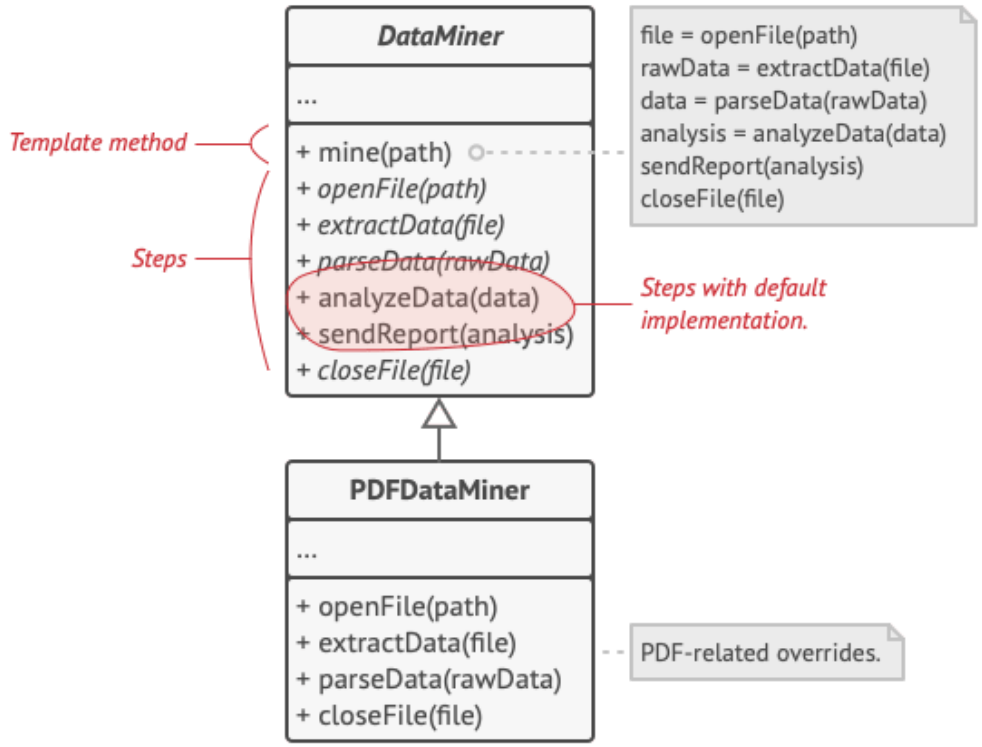


- Vous avez un comportement complexe (createInvoice()) qui change selon les différents scénarios (IndividualInvoice, CompanyInvoice).
- Cependant, certaines parties du comportement restent les mêmes quel que soit le scénario (header(), footer()).



Solution #5 (template method)



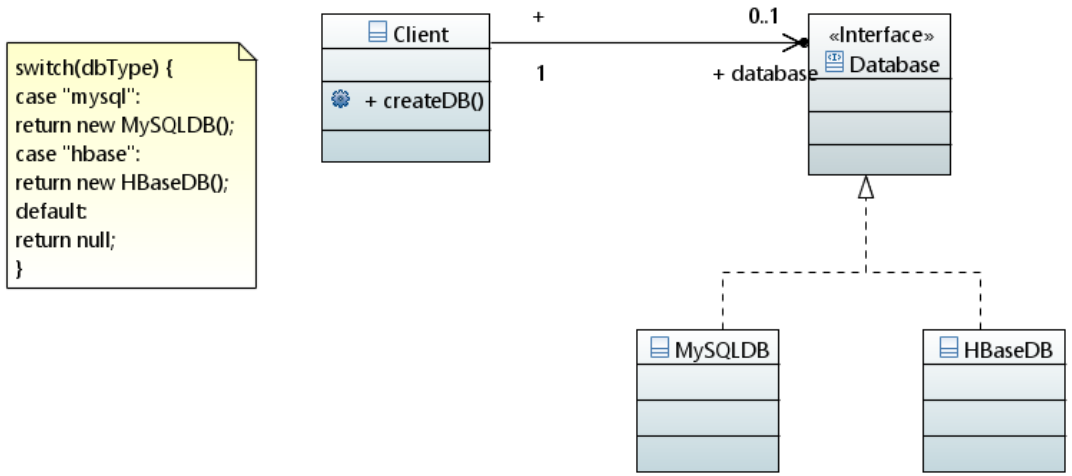


Patrons de Conception pour $1200

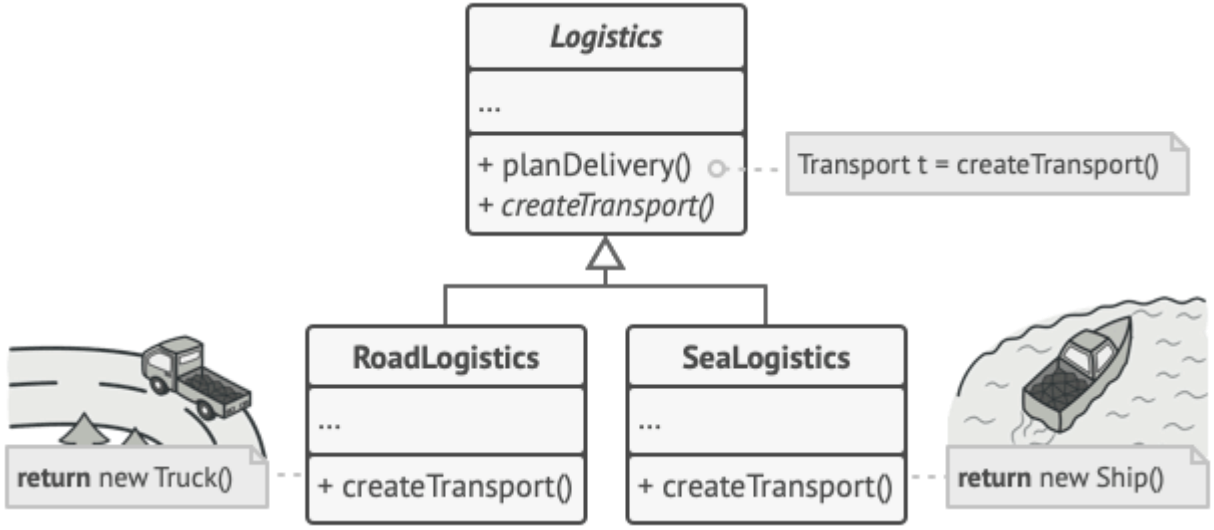
- Vous avez plusieurs bases de données qui ont chacune leur façon de s’initialiser et vous souhaitez que votre application puisse construire et utiliser chacune d’entre elles de façon interchangeable (db1, db2, ...) au choix de l'utilisateur.

Solution #6 (factory method)

Choix du client



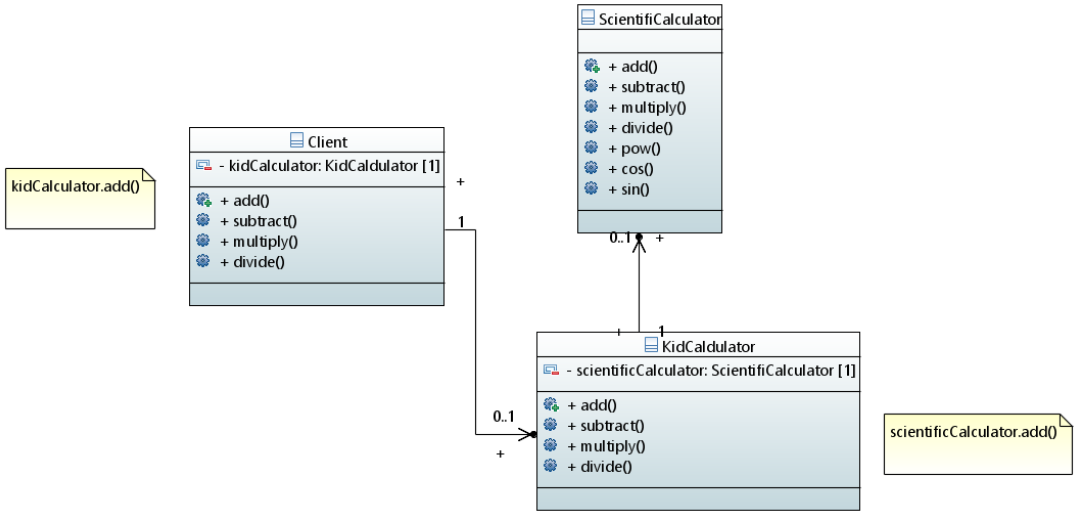
- Vous disposez d'un objet (Database) avec différentes implémentations (MySQLDB, HBaseDB).
- Une implémentation n'est créée que sur le choix d'un client.



Patrons de Conception pour $1400

- Vous êtes embauchés par MathWorks (les développeurs de Matlab). La compagnie voudrait fournir des logiciels aux écoles primaires, mais ils veulent réutiliser le code existant. On vous fournit le code pour une calculatrice scientifique et votre tâche est de développer une calculatrice pour les enfants avec juste les opérations de base.

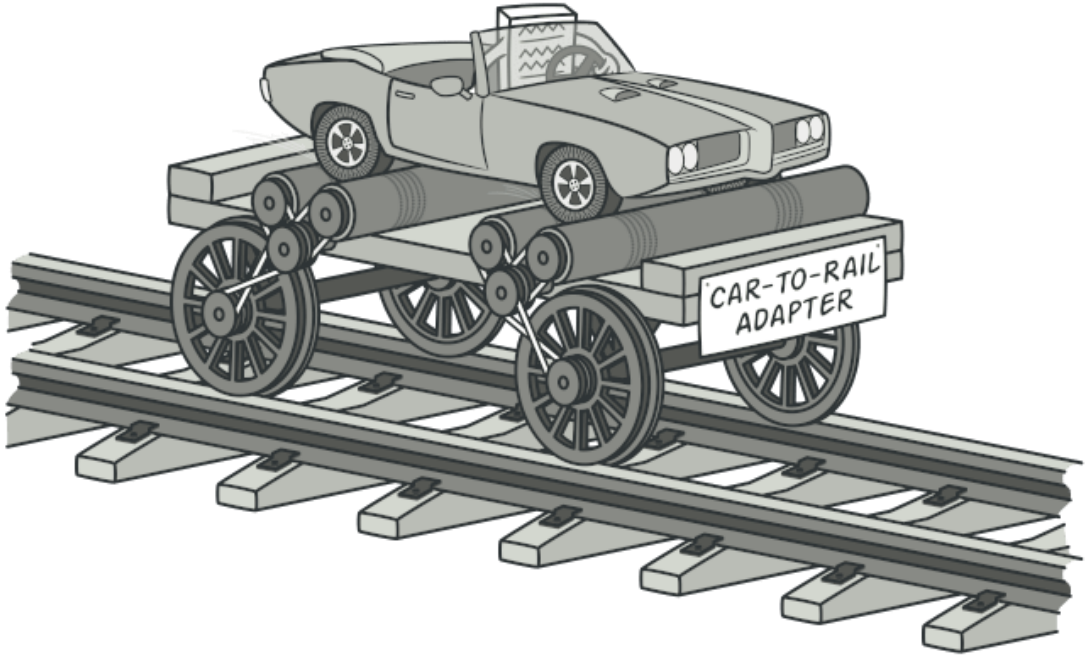
Solution #7 (adapter or façade) Adapter le code, le réutiliser



- Vous disposez d'une interface complexe (ScientificCalculator) et d'un objet (Client) qui ne peut pas la comprendre en grande partie.
- L'adaptateur traduit une interface en une autre comprise par le client.
- Façade simplifie une interface complexe ou en fusionne plusieurs pour ne publier que certaines opérations.

Solution #7 (adapter or façade)

Adapter



Façade