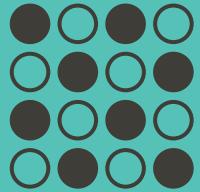


# LOG210 SÉANCE #09



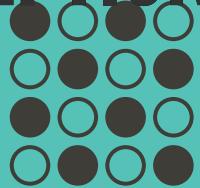
Created by Jonathan Li  
from the Noun Project

## ANALYSE ET CONCEPTION DE LOGICIELS

1. Découverte des patrons GOF ← 10.23m
2. GRASP sont une généralisation d'autres patterns
3. Adaptateur, Fabrique concrète, Singleton
4. Logique de tarification élaborée avec patron Stratégie et Composite
5. Actualisation interface usager
6. Recouvrement avec Proxy
7. Faute, erreur, échec et exception
8. Patron faire soi-même
9. Attention à la patternite



# DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION

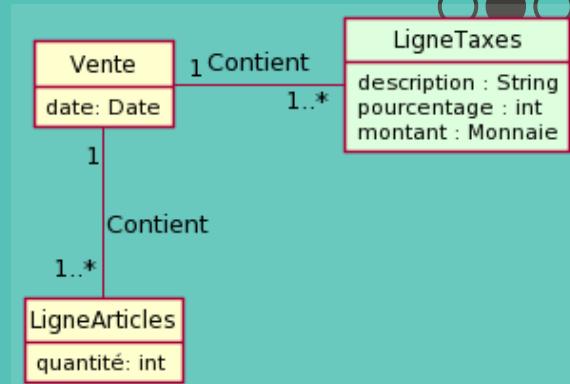
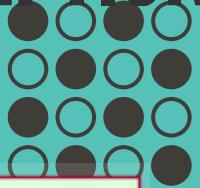


Created by Jonathan Li  
from the Noun Project





# DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION





# DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION

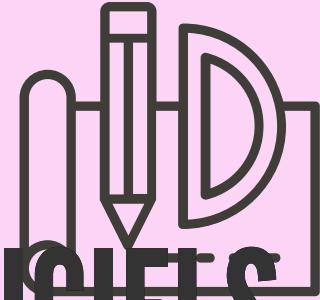


- Calculateurs de taxes retournent LigneTaxes
  - pas envisagé lors de l'analyse initiale
- Concept dans le modèle du domaine
  - mettre à jour le modèle du domaine
  - mettre à jour le glossaire
- Avec l'expérience ceci devient un réflexe
  - travail d'un ingénieur (vs technicien ou codeur)
  - nécessaire dans un processus itératif



# LOG210 SÉANCE #09

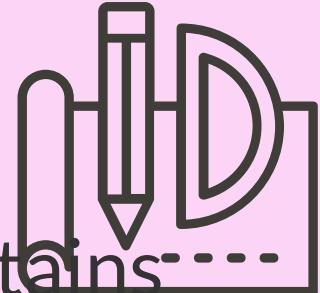
## ANALYSE ET CONCEPTION DE LOGICIELS



- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns! ← 16.43m
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite



# GRASP DANS LES GOF



- Adaptateur est une spécialisation de certains principes GRASP

Faible couplage est une façon d'obtenir une protection à un point de variation.

Polymorphisme est une façon d'obtenir une protection à un point de variation et un faible couplage.

Une indirection est une façon d'obtenir un faible couplage.

Le pattern Adaptateur est une sorte d'indirection et une Fabrication pure qui applique le principe de Polymorphisme.

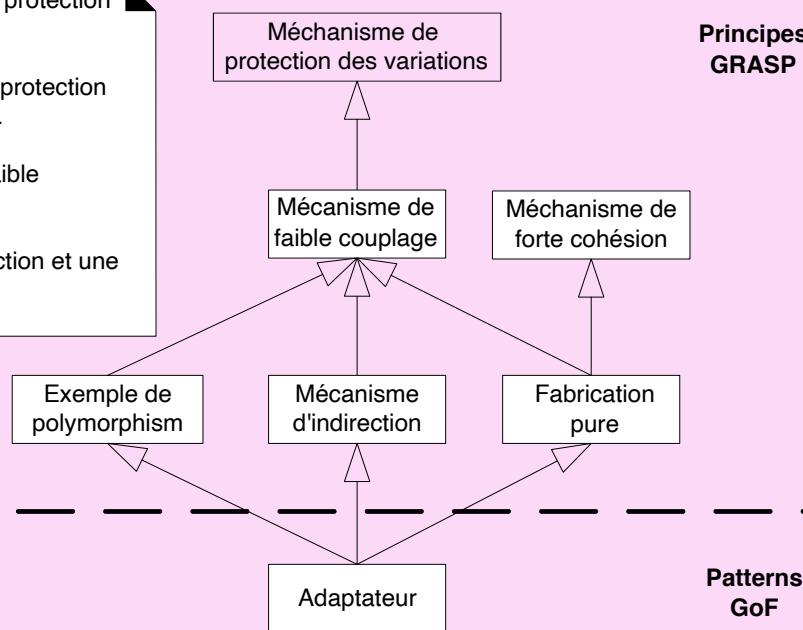
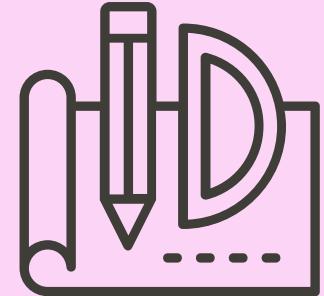


fig. F23.3

# GRASP DANS LES GOF



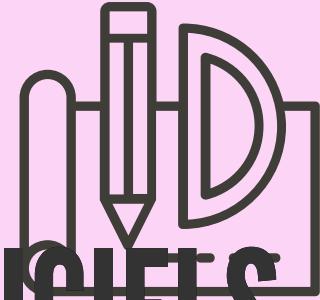
Exercice Google Classrooms  
“Identification des GRASP dans les GoF”

Salles de travail Zoom

20 minutes, suivre le modèle

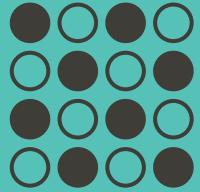
# LOG210 SÉANCE #09

## ANALYSE ET CONCEPTION DE LOGICIELS



- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton  9.21m
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite





Created by Jonathan Li  
from the Noun Project

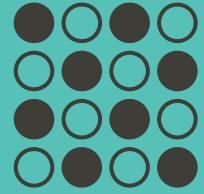
# \*PROBLÈME

- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

## Patron GOF?



# ADAPTATEUR (GOF)



Created by Jonathan Li  
from the Noun Project

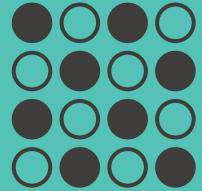
## \*PROBLÈME

- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

## Patron GOF?



# ADAPTATEUR (GOF)



Created by Jonathan Li  
from the Noun Project

- Problème (suite)
  - On veut éviter ce genre de solution:

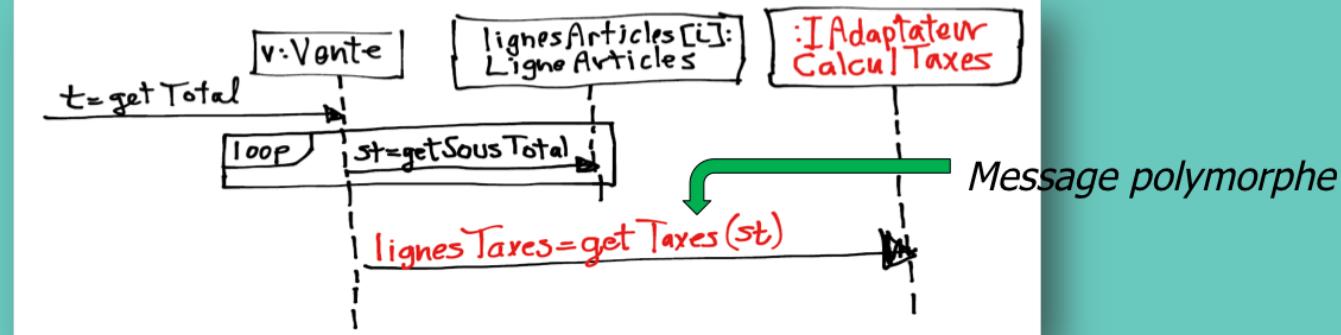
```
if [TaxMaster] {  
    taxes = tm.caclulateTaxes(montant);  
}  
elseif [GoodAsGold] {  
    taxes = gag.computeTaxes(montant);  
}  
else if [...] {  
...  
}
```

GRASP Polymorphisme

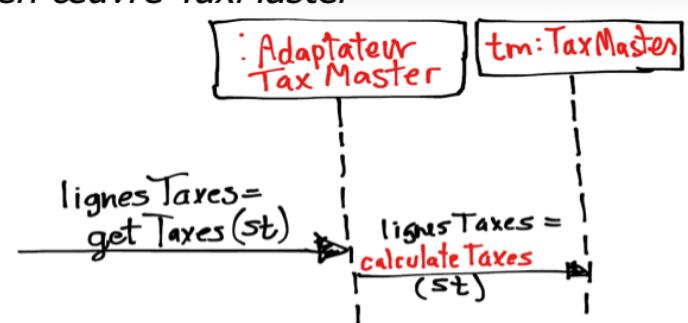
*L'API de chaque service est distincte et immuable.*

*L'API de chaque service est distincte et immuable.*

# ADAPTATEUR POUR LE CALCUL DES TAXES

Created by Jonathan Li  
from the Noun Project

mise en œuvre TaxMaster



mise en œuvre GoodAsGold

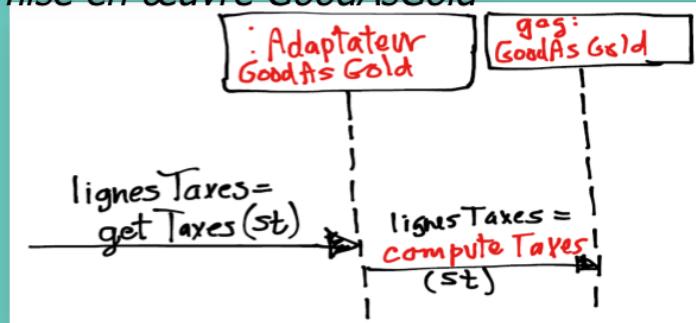
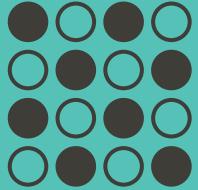


fig. F14.21

# ADAPTATEUR



Created by Jonathan Li  
from the Noun Project

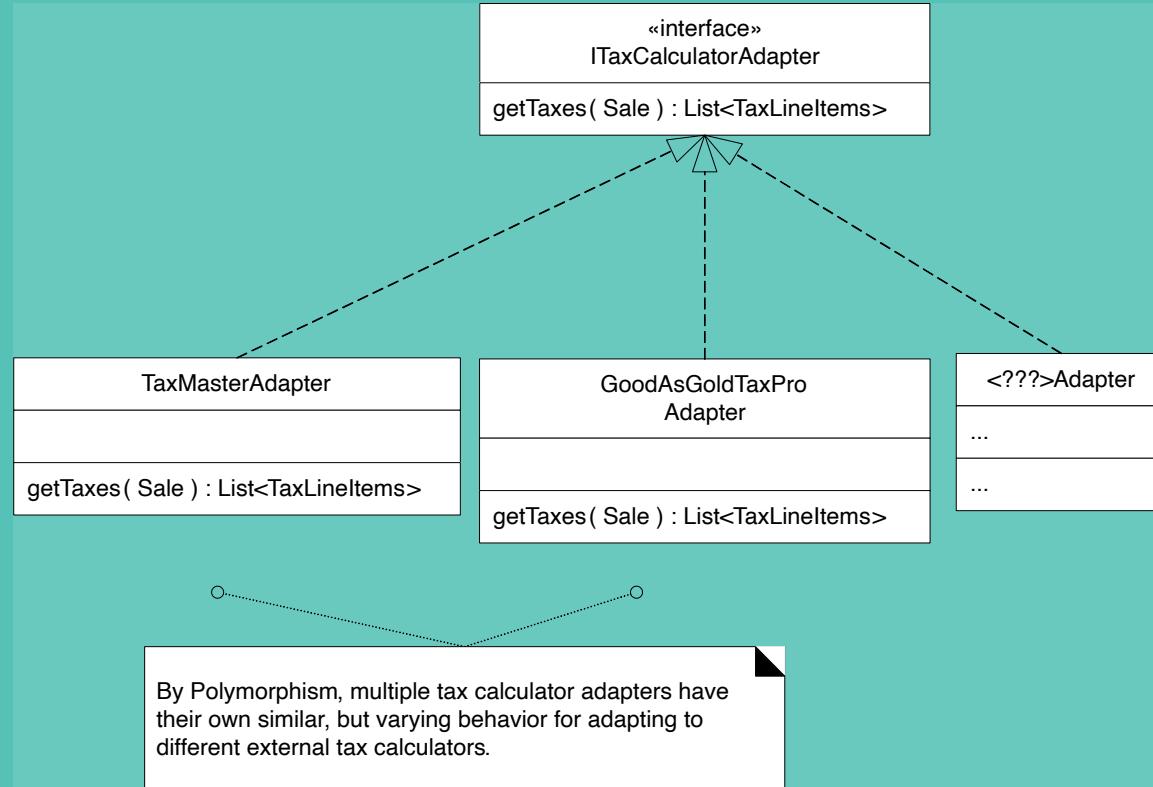
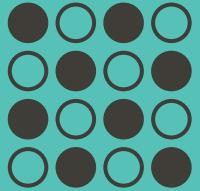


fig. F22.1/A25.1

# ADAPTATEUR (GOF)



Created by Jonathan Li  
from the Noun Project

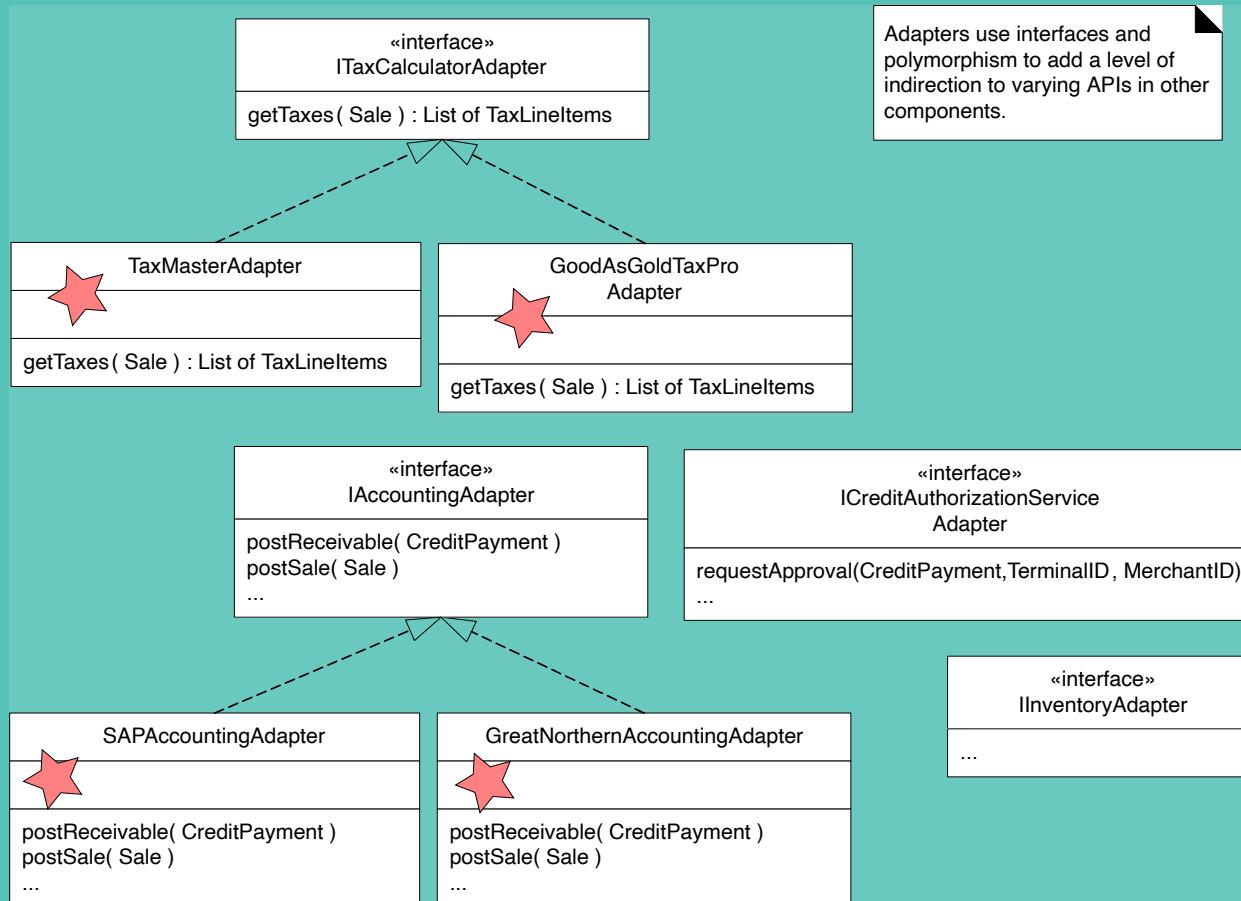
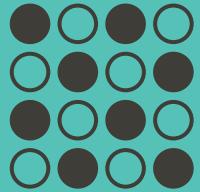


fig. F23.1

# COMPTABILITÉ



Created by Jonathan Li  
from the Noun Project

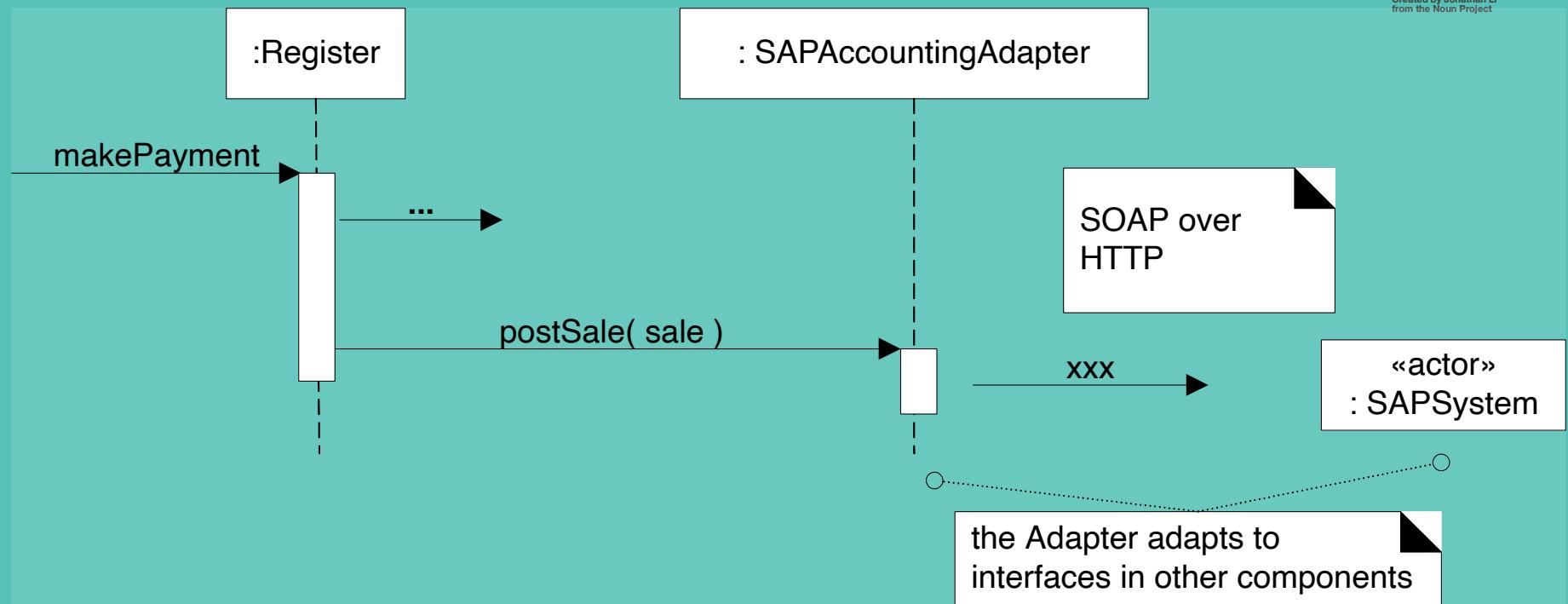
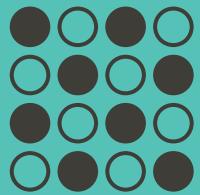


fig. F23.2

# \*GRASP DANS LE LABORATOIRE

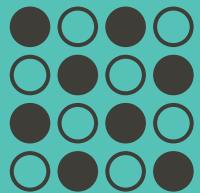


Created by Jonathan Li  
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi



# \*GRASP DANS LE LABORATOIRE

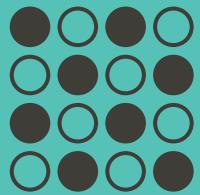


Created by Jonathan Li  
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi
- Adaptateur? -> cache mémoire accès information



# \*GRASP DANS LE LABORATOIRE

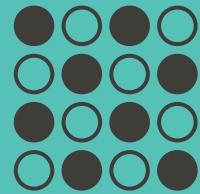


Created by Jonathan Li  
from the Noun Project

- SGB est un système externe alors avez vous utilisé un patron GOF pour que SGA se connecte a SGB.
- Si oui, quel patron et pourquoi
- Si non, pourquoi
- Adaptateur? -> cache mémoire accès information
- Proxy? Persistance pour écrire lorsque SGB n'est pas disponible



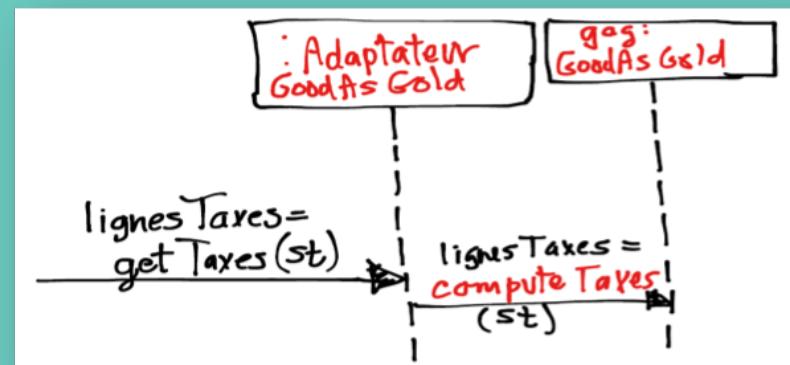
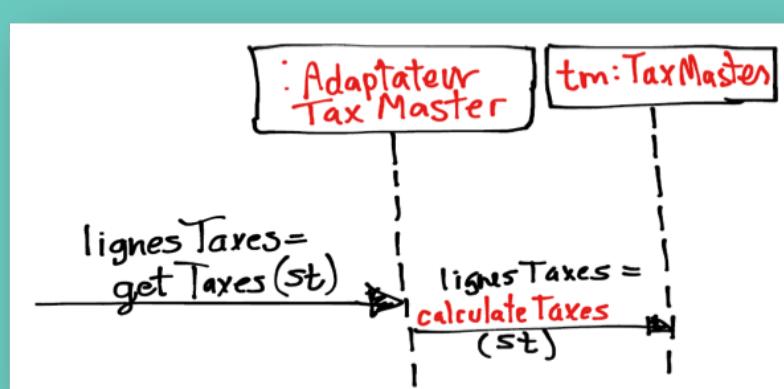
# ADAPTATEUR - PROBLÈMES

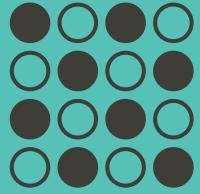


Created by Jonathan Li  
from the Noun Project

- Qui instancie les adaptateurs?
- Comment déterminer la classe d'adaptateur à instancier?

## AdaptateurTaxMaster vs AdaptateurGoodAsGold



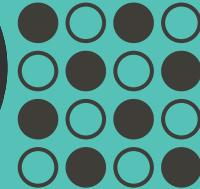


Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - Créer des objets
  - logique de création complexe, ou
  - séparer les responsabilités de créations (+ cohésion)
- Solution
  - Fabrication Pure nommé FabriqueConcrète pour gérer la création



# FABRIQUE CONCRÈTE (PAS GOF)

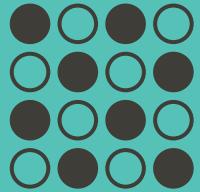


Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - Créer des objets
  - logique de création complexe, ou
  - séparer les responsabilités de créations (+ cohésion)
- Solution
  - Fabrication Pure nommé FabriqueConcrète pour gérer la création



# FABRIQUE CONCRÈTE



Created by Jonathan Li  
from the Noun Project

FabriqueDeServices
adaptateurCompta : IAdaptateurCompta
adaptateurStock : IAdaptateurStock
adaptateurCalculateurTaxes : IAdaptateurCalculateurTaxes
getAdaptateurCompta () : IAdaptateurCompta
getAdaptateurStock () : IAdaptateurStock
getAdaptateurCalculateurTaxes () : IAdaptateurCalculateurTaxes
...

notez que les méthodes d'une Fabrique concrète retournent des objets typés interface (et non typés classe), afin de pouvoir retourner n'importe quelle implémentation de l'interface

```

if ( adaptateurCalculateurTaxes == null )
{
    // une approche réflexive ou pilotée par les données pour trouver la bonne classe :
    // lire dans une propriété externe

    String nomClasse = System.getProperty( "calculateurtaxes .classe.nom" );
    adaptateurCalculateurTaxes = (IAdaptateurCalculateurTaxes ) Class.forName(nomClasse ).newInstance();

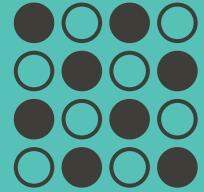
}
return adaptateurCalculateurTaxes ;

```

Documentation `System.getProperty()`



# \*PROBLÈME SUIVANT...

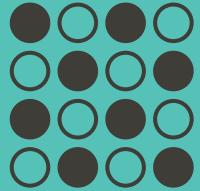


Created by Jonathan Li  
from the Noun Project

- Qui instancie la Fabrique elle-même?
- Comment y accéder?
- Indices
  - besoin de seulement une instance de chaque Fabrique
  - méthodes de la Fabrique appelées à partir d'endroits différents dans le code
  - problème de visibilité



# \*PROBLÈME SUIVANT...



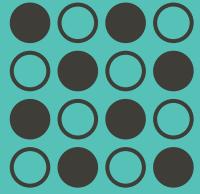
Created by Jonathan Li  
from the Noun Project

## PATRON SINGLETON

- Qui instancie la Fabrique elle-même?
- Comment y accéder?
- Indices
  - besoin de seulement une instance de chaque Fabrique
  - méthodes de la Fabrique appelées à partir d'endroits différents dans le code
  - problème de visibilité



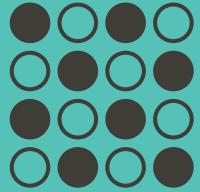
# SINGLETON



Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - Une seule instance est permise: le « singleton »
  - besoin d'un point d'accès global et unique
- Solution
  - méthode statique qui retourne le singleton

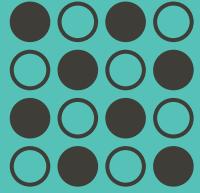
# SINGLETON



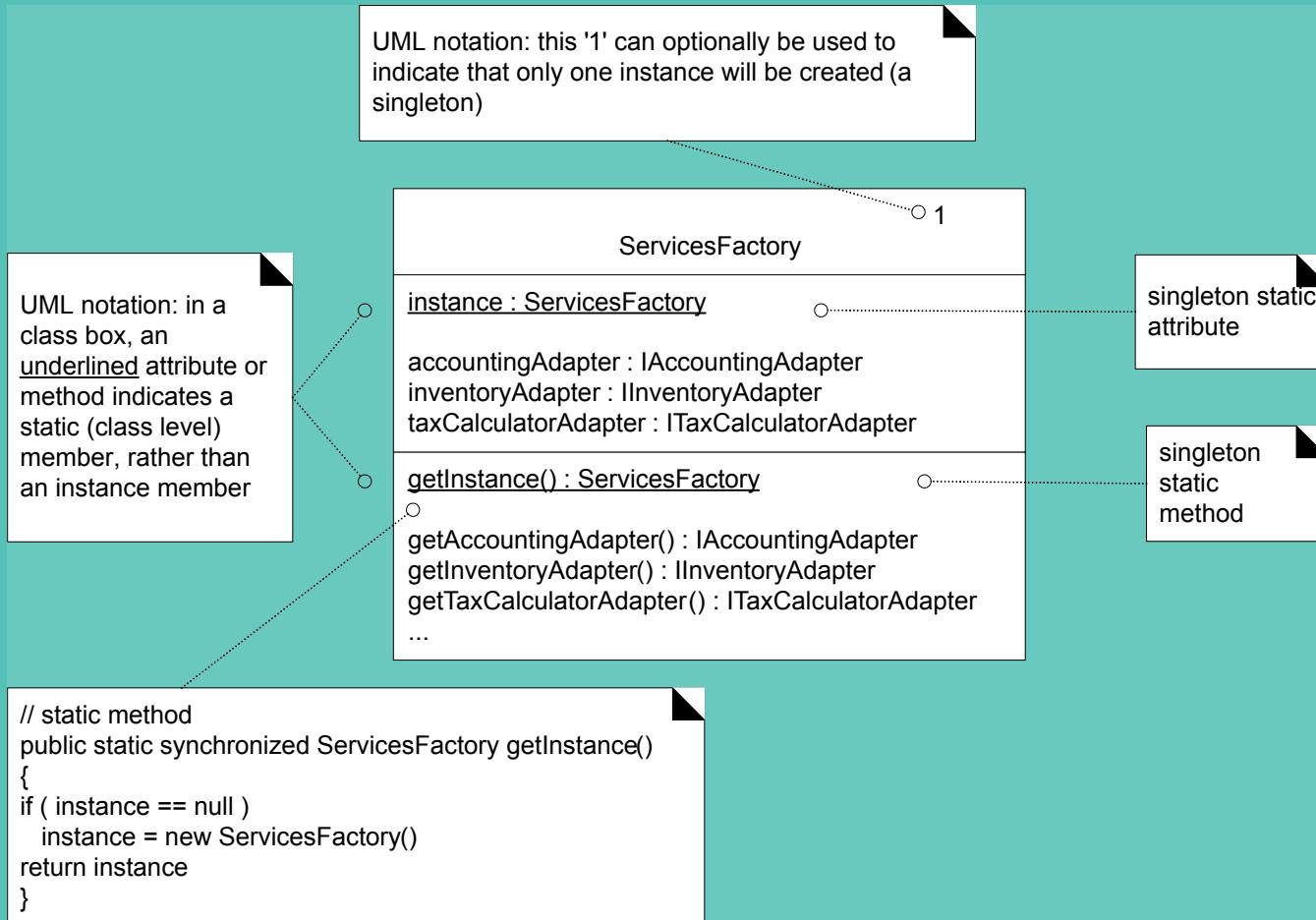
Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
    - Une seule instance est permise: le « singleton »
    - besoin d'un point d'accès global et unique
  - Solution
    - méthode statique qui retourne le singleton
- ! IL EST DIFFICILE DE TESTER UN SINGLETON**

# SINGLETON



Created by Jonathan Li  
from the Noun Project



# SINGLETON – IMPLÉMENTATION JAVA



Created by Jonathan Li  
from the Noun Project

- Initialisation différée
- `getInstance()` est souvent fréquemment appelé
- Avec plusieurs fils d'exécution (threads), l'étape de création d'une logique d'initialisation différée (lazy initialization) est critique
  - nécessite un contrôle de la concurrence des threads (p.ex. une méthode atomique)

```
public static synchronized FabriqueDeServices getInstance () {  
    if (instance == null ) {  
        // section critique pour une application multithread  
        instance = new FabriqueDeServices();  
    }  
    return instance;  
}
```

référence: <http://www.cs.wustl.edu/~schmidt/PDF/monitor.pdf>



# SINGLETON - IMPLÉMENTATION JAVA

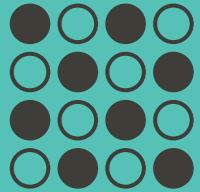


```
public class FabriqueDeServices {  
    // initialisation immédiate  
    private static FabriqueDeServices instance =  
        new FabriqueDeServices();  
    public static FabriqueDeServices getInstance () {  
        return instance;  
    } // autres méthodes...  
}
```

- Initialisation immédiate
- Quelle initialisation est préférée?
- Initialisation différée a plusieurs avantages:
  - si l'objet Singleton n'est jamais utilisé, on évite un travail de création inutile
  - getInstance() comprend parfois une logique de création complexe et conditionnelle (difficile à mettre dans une initialisation immédiate)



# SINGLETON



Created by Jonathan Li  
from the Noun Project

- Message implicite « getInstance »
- Pas besoin de l'écrire dans la dynamique

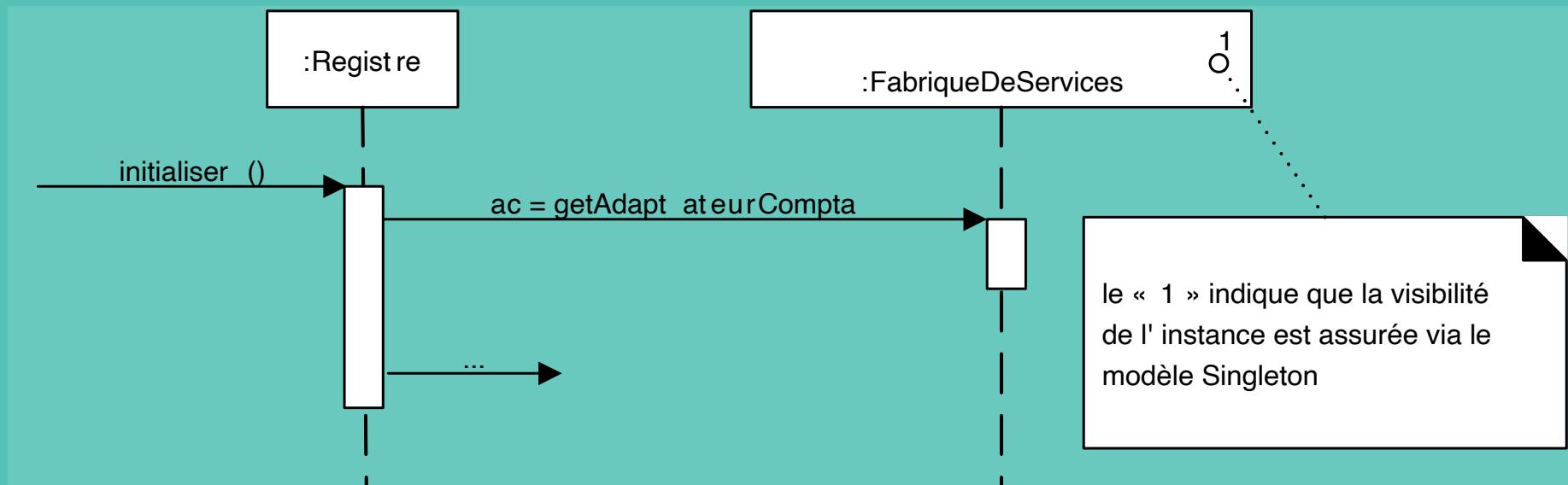
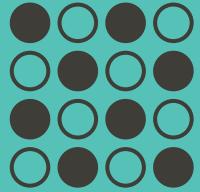
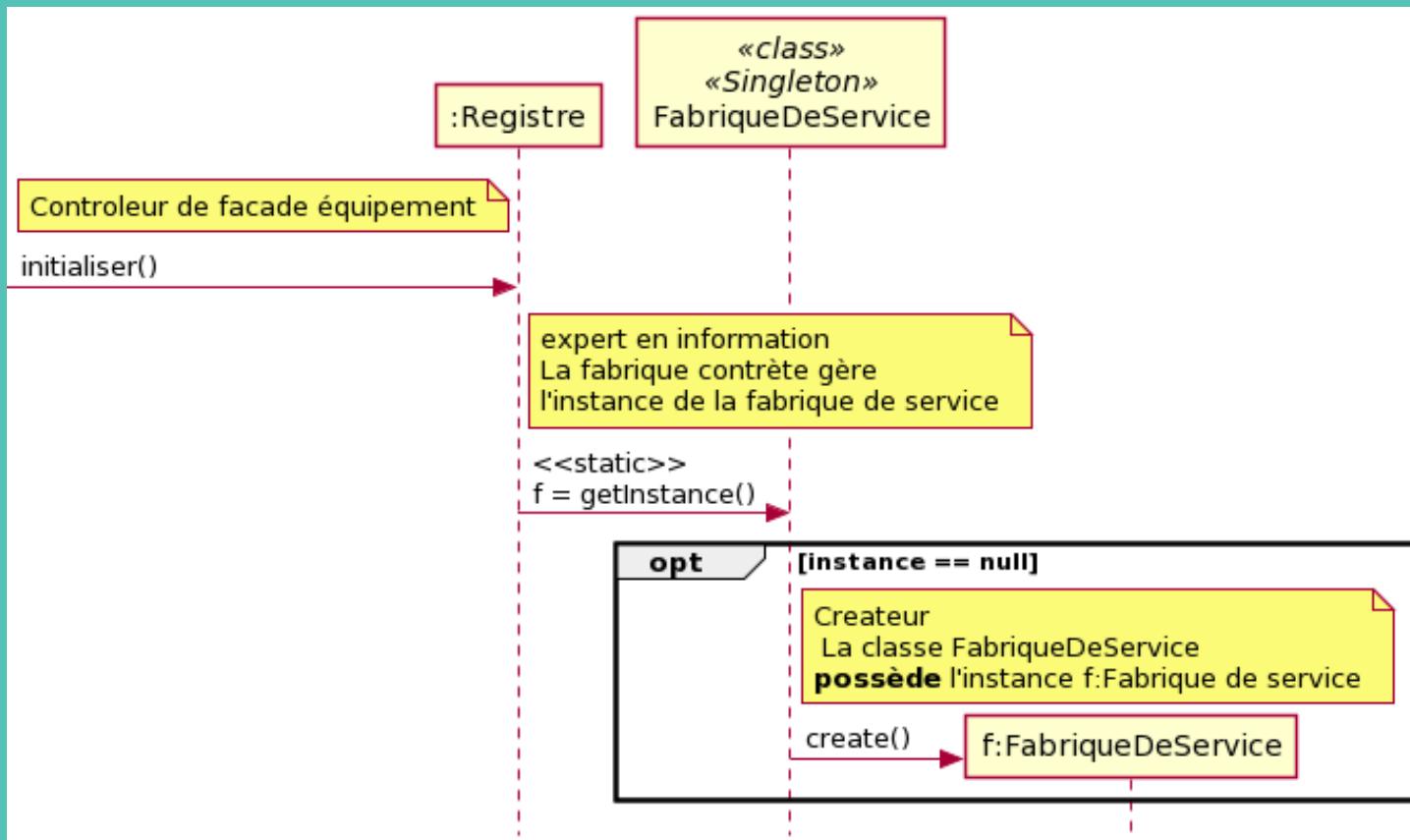


fig. F23.7

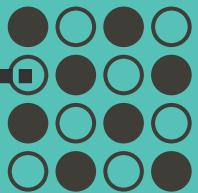
# SINGLETON



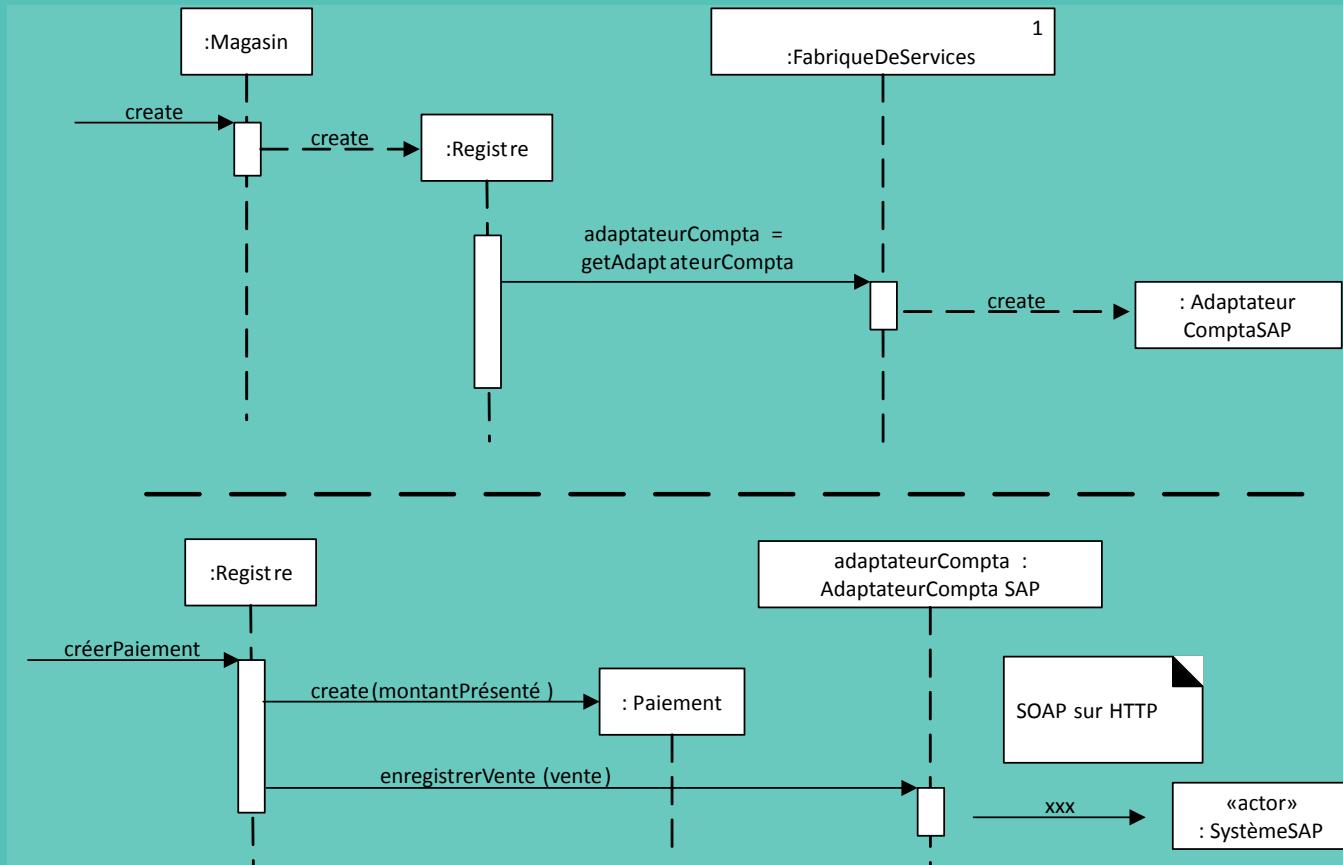
FAIRE LE DS DE LA SOLUTION PRÉCÉDENTE EN INCLUANT LA DYNAMIQUE DE GETINSTANCE

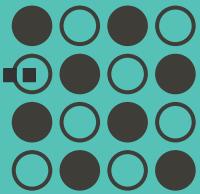


# \*CONCLUSION SUR LES SERVICES EXTERNES..

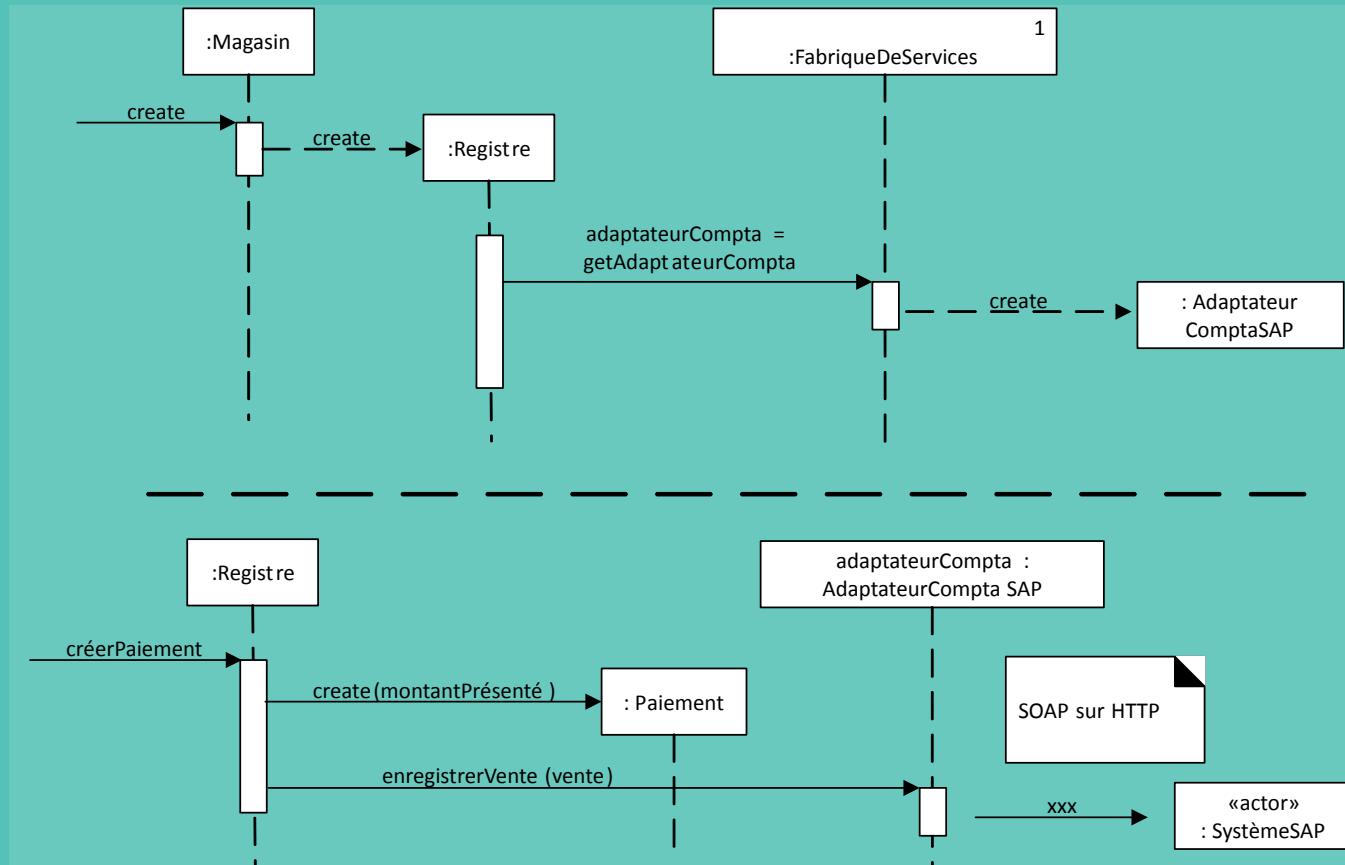


Created by Jonathan Li  
from the Noun Project





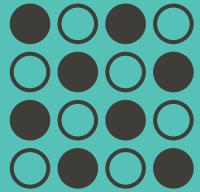
# \*CONCLUSION SUR LES SERVICES EXTERNES...



Quel message est polymorphe?



# LOG210 SÉANCE #09

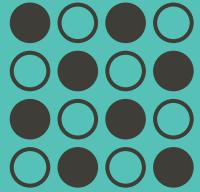


Created by Jonathan Li  
from the Noun Project

## ANALYSE ET CONCEPTION DE LOGICIELS

- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite ← 18.57m
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite





Created by Jonathan Li  
from the Noun Project

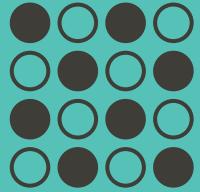
# LOGIQUE DE TARIFICATION ÉLABORÉE

- Support pour la logique des promotions
  - Réductions, soldes, rabais, etc.
- « Stratégie de tarification » peut varier
  - Règle, algorithme, politique, etc.
- Exemples
  - réduction de 10% sur toute les ventes pour une période
  - réduction fixe de 10 dollars pour tout achat d'un montant supérieur à 200\$
  - etc.
- Il ne faut pas reprogrammer le logiciel pour chaque nouvelle promotion
- Comment faire un bon design face à cette exigence?

Patron GOF?



# STRATEGIE



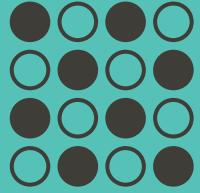
Created by Jonathan Li  
from the Noun Project

## LOGIQUE DE TARIFICATION ÉLABORÉE

- Support pour la logique des promotions
  - Réductions, soldes, rabais, etc.
- « Stratégie de tarification » peut varier
  - Règle, algorithme, politique, etc.
- Exemples
  - réduction de 10% sur toute les ventes pour une période
  - réduction fixe de 10 dollars pour tout achat d'un montant supérieur à 200\$
  - etc.
- Il ne faut pas reprogrammer le logiciel pour chaque nouvelle promotion
- Comment faire un bon design face à cette exigence?

Patron GOF?





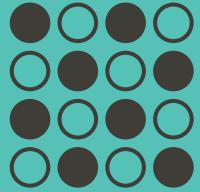
Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - algorithmes ou politiques sont variables mais se ressemblent (politiques de tarification)
  - ils doivent pouvoir évoluer
- Solution
  - définir les algorithmes/politiques/stratégies dans une classe séparée, avec une interface commune.

## Patron GOF?



# STRATÉGIE



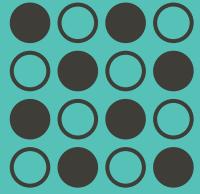
Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - algorithmes ou politiques sont variables mais se ressemblent (politiques de tarification)
  - ils doivent pouvoir évoluer
- Solution
  - définir les algorithmes/politiques/stratégies dans une classe séparée, avec une interface commune.

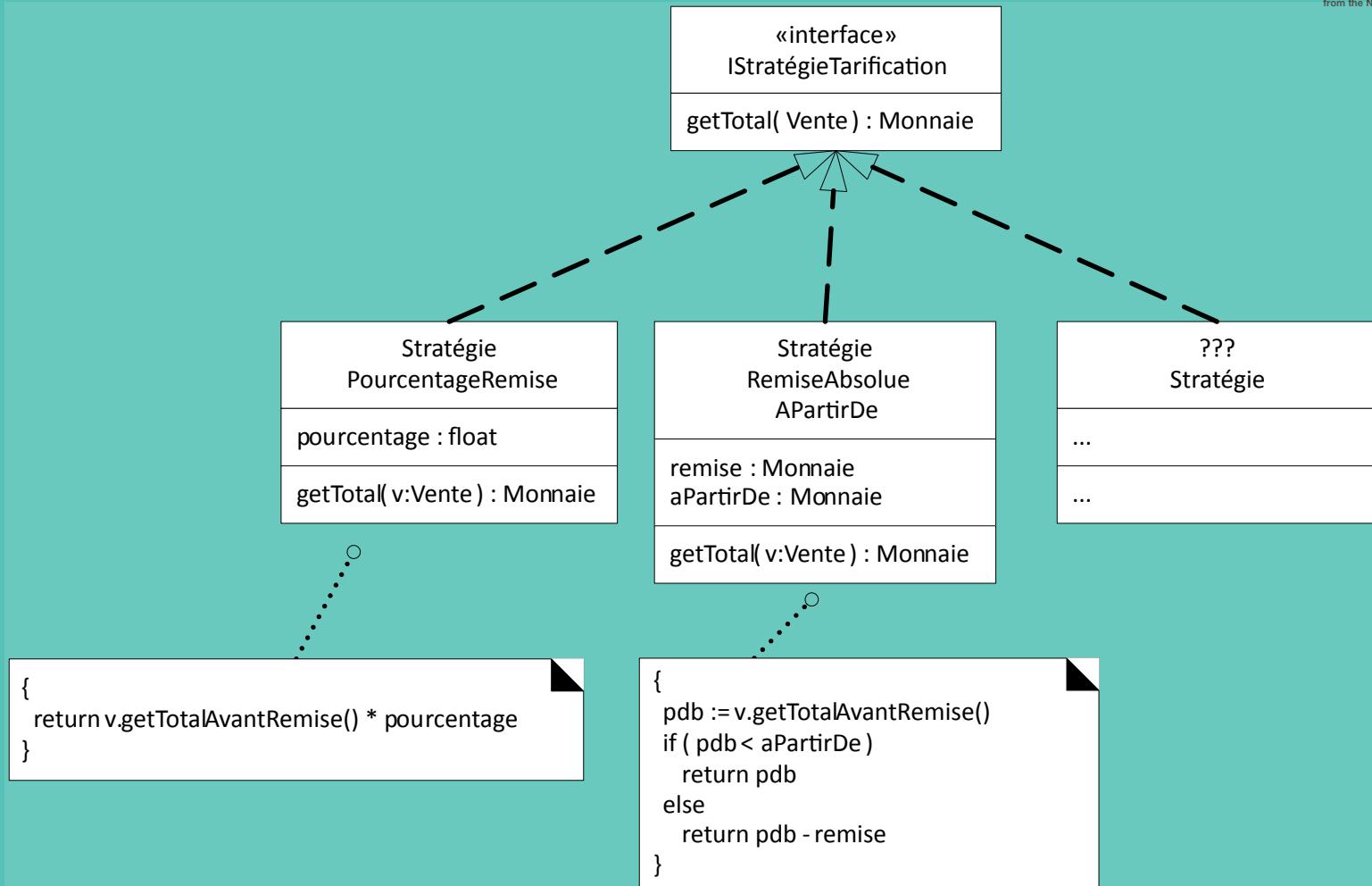
## Patron GOF?



# STRATÉGIES DES PRIX

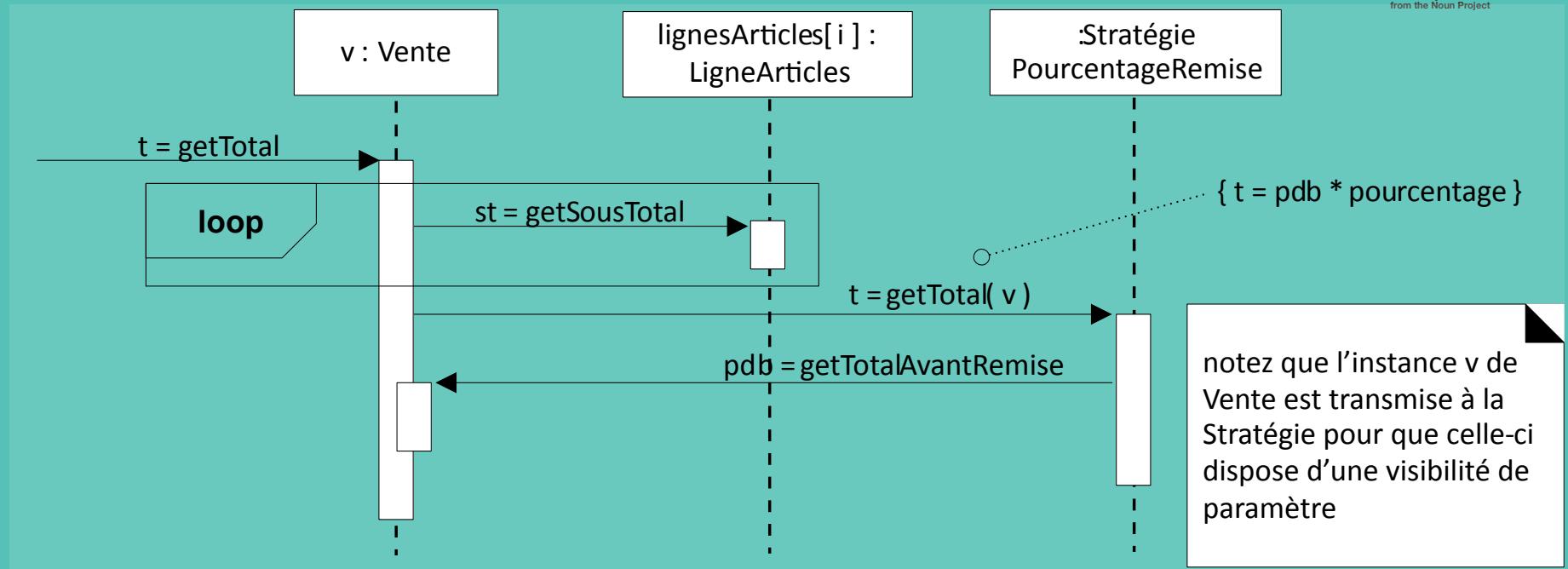


Created by Jonathan Li  
from the Noun Project

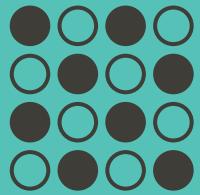


# STRATÉGIE ET OBJET CONTEXT (VENTE)

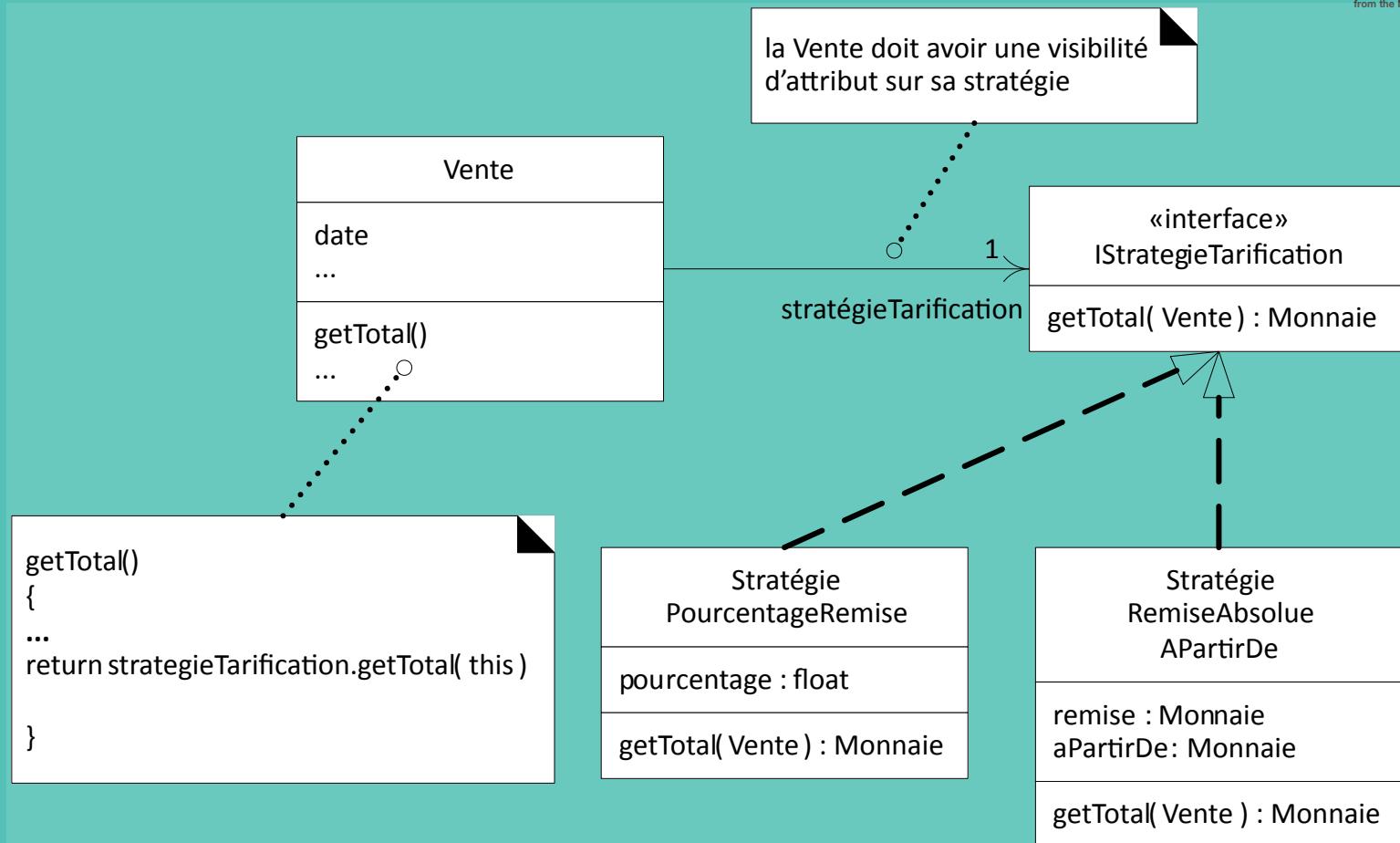
Created by Jonathan Li  
from the Noun Project



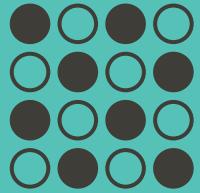
# L'OBJET CONTEXTE (VENTE)



Created by Jonathan Li  
from the Noun Project



# CRÉER UNE STRATÉGIE



Created by Jonathan Li  
from the Noun Project

1

FabriqueDeStratégiesTarification

instance : FabriqueDeStratégiesTarification

getInstance() : FabriqueDeStratégiesTarification

getStratégieTarification() : IStratégieTarification

getStratégieTarificationSenior() : IStratégieTarification

...

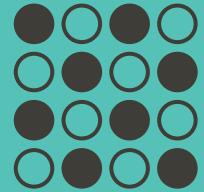
{

```
String nomClasse= System.getProperty( "strategietarification.classe.nom" );
strategie = (IStratégieTarification) Class.forName( nomClasse ).newInstance();
return strategie;
```

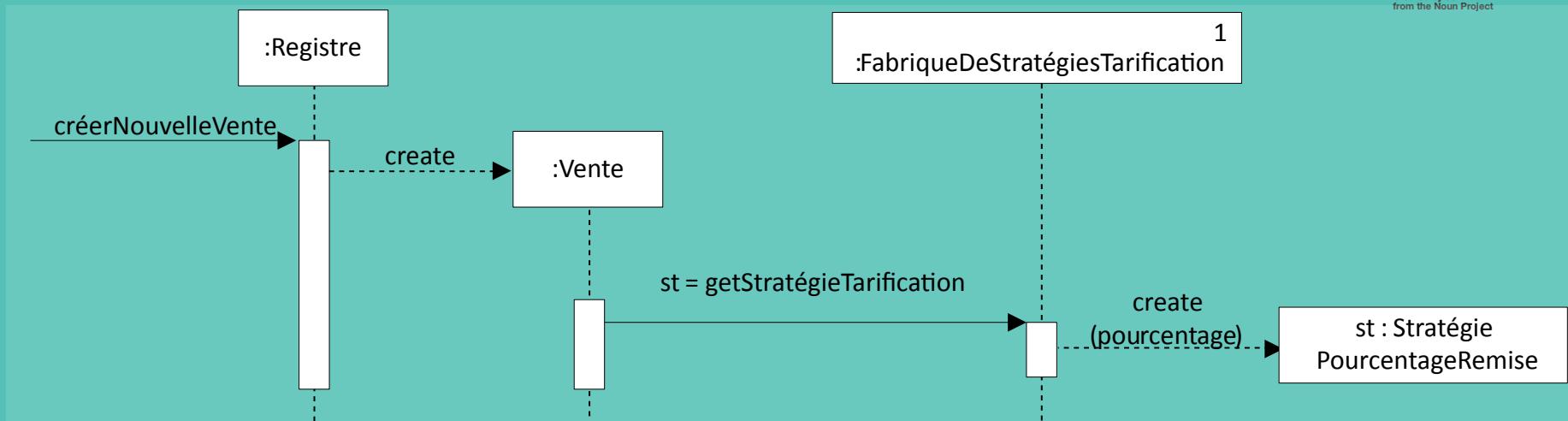
}



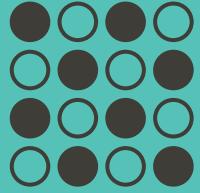
# CRÉER UNE STRATÉGIE



Created by Jonathan Li  
from the Noun Project

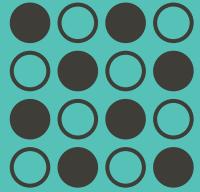


# EN RÉSUMÉ



Created by Jonathan Li  
From the Noun Project

- *Stratégie* et *Fabrique concrète* assurent la *Protection des variations* dues aux changements dynamiques de politique tarifaire
- *Stratégie* → *Polymorphisme* (GRASP)
- Fabriques sont souvent utilisées pour créer les stratégies pour autoriser des algorithmes insérables



Created by Jonathan Li  
from the Noun Project

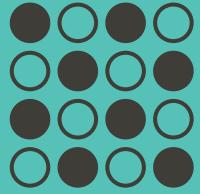
# CONFLITS DUS AUX POLITIQUES TARIFAIRES

- Politiques de prix pour aujourd’hui (lundi)
  - 20% de rabais pour les personnes du troisième âge
  - 15% de rabais sur un achat supérieur à 400\$
  - 50 dollars de rabais pour sur un achat supérieur à 500\$ (ce lundi seulement)
  - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
  - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600\$

Patron GOF?



# COMPOSITE



Created by Jonathan Li  
from the Noun Project

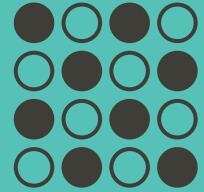
## CONFLITS DUS AUX POLITIQUES TARIFAIRES

- Politiques de prix pour aujourd’hui (lundi)
  - 20% de rabais pour les personnes du troisième âge
  - 15% de rabais sur un achat supérieur à 400\$
  - 50 dollars de rabais pour sur un achat supérieur à 500\$ (ce lundi seulement)
  - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
  - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600\$

Patron GOF?



# FACTEURS DES STRATÉGIES

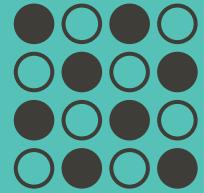


Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)



# FACTEURS DES STRATÉGIES

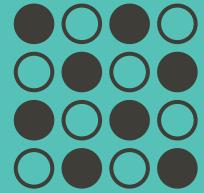


Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)



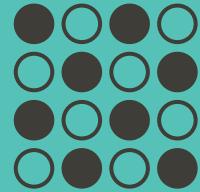
# FACTEURS DES STRATÉGIES



Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)

# FACTEURS DES STRATÉGIES

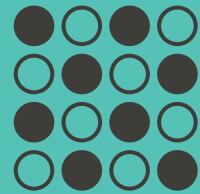


Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits



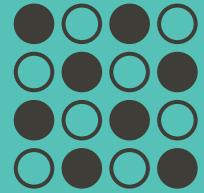
# FACTEURS DES STRATÉGIES



Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas

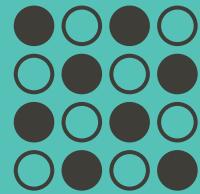
# FACTEURS DES STRATÉGIES



Created by Jonathan Li  
from the Noun Project

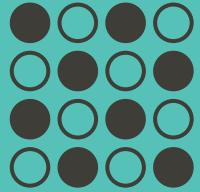
- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas
  - Prix le plus haut

# FACTEURS DES STRATÉGIES



Created by Jonathan Li  
from the Noun Project

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas
  - Prix le plus haut
  - Etc.



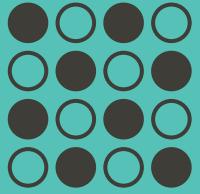
Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - parfois on traite un seul objet
    - atomique
  - parfois on traite un groupe
    - composition d'objets
  - les traiter de la même façon
- Solution
  - définir des classes pour les objets
    - composites
    - atomiques
- implémentent la même interface

## Patron GOF?



# COMPOSITE



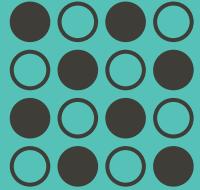
Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - parfois on traite un seul objet
    - atomique
  - parfois on traite un groupe
    - composition d'objets
  - les traiter de la même façon
- Solution
  - définir des classes pour les objets
    - composites
    - atomiques
- implémentent la même interface

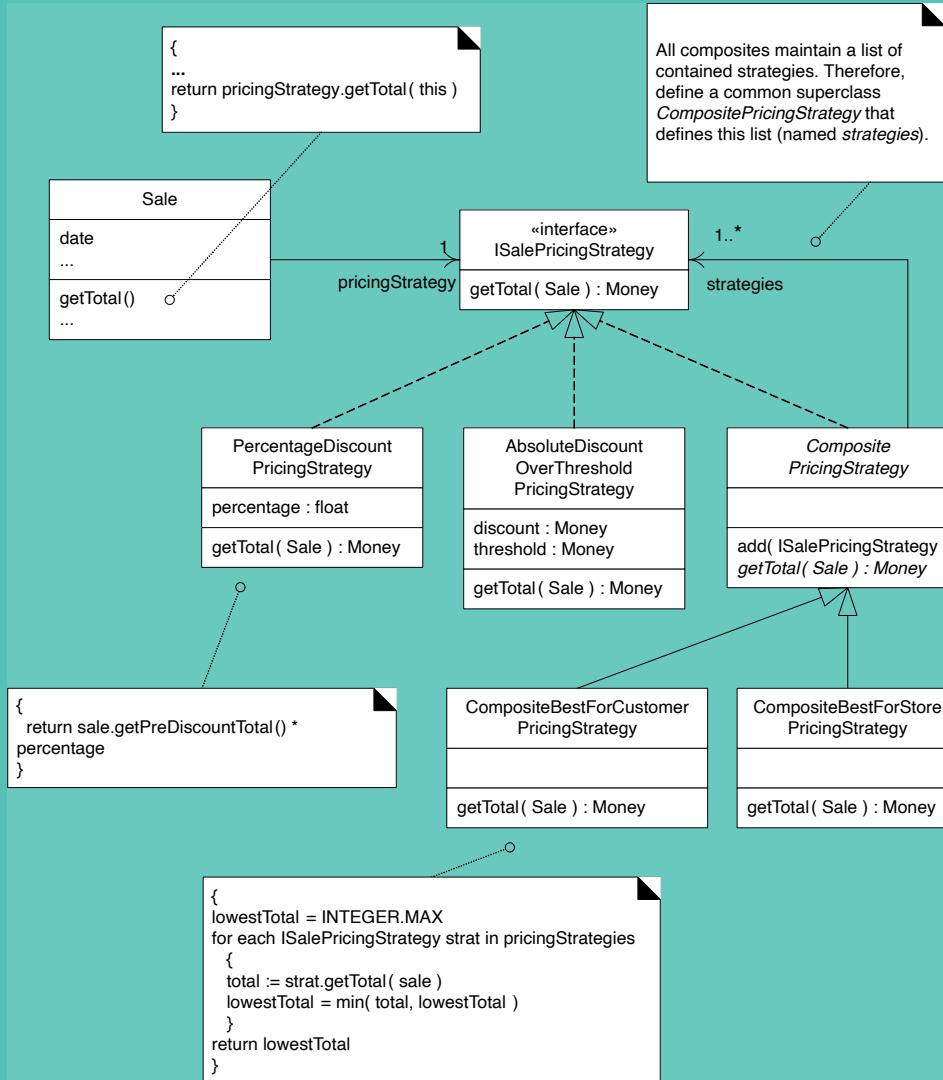
## Patron GOF?



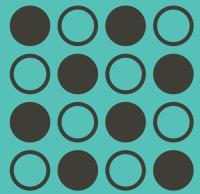
# COMPOSITE POS



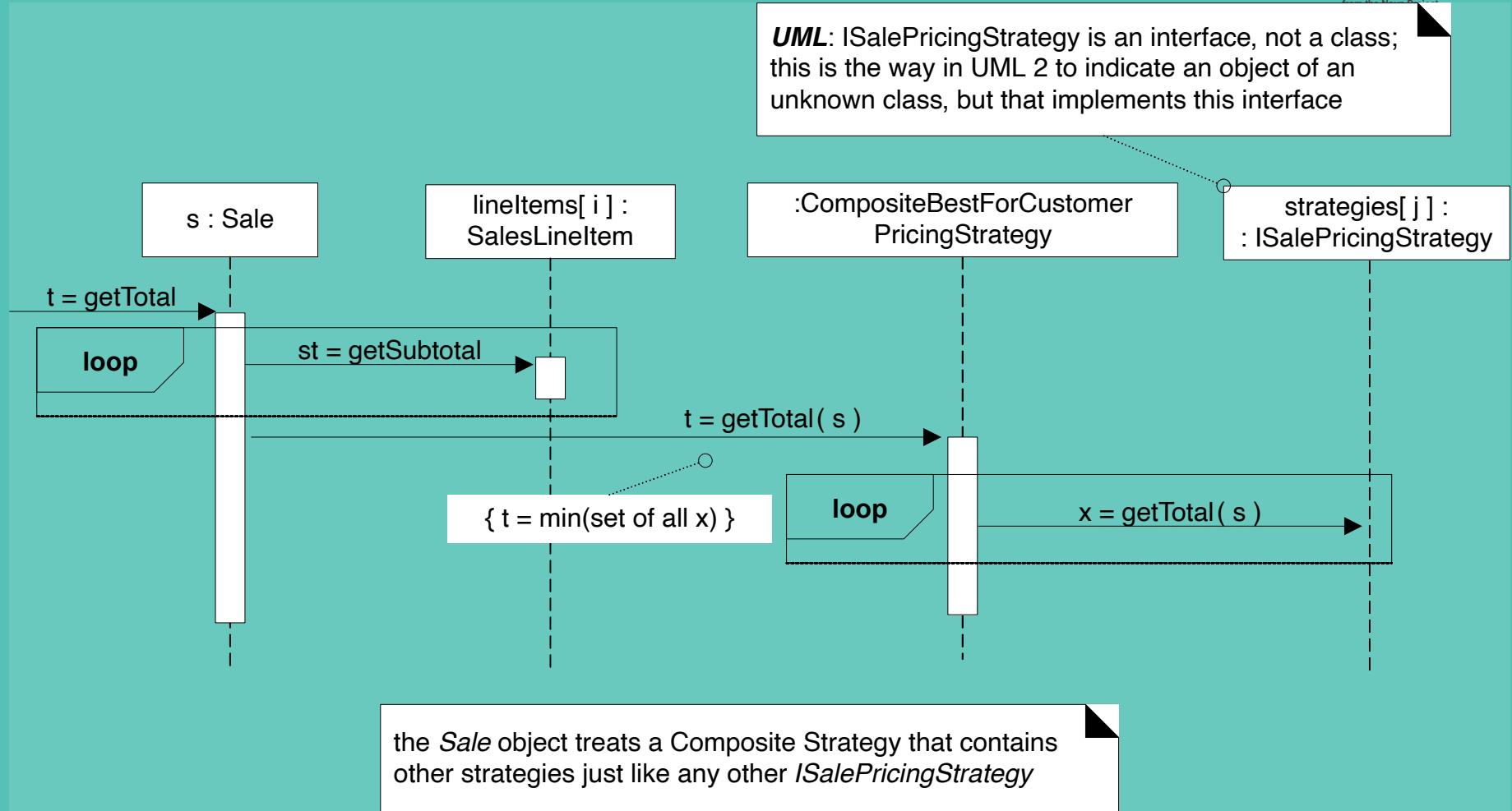
Created by Jonathan Li  
from the Noun Project



# COMPOSITE – COLLABORATION

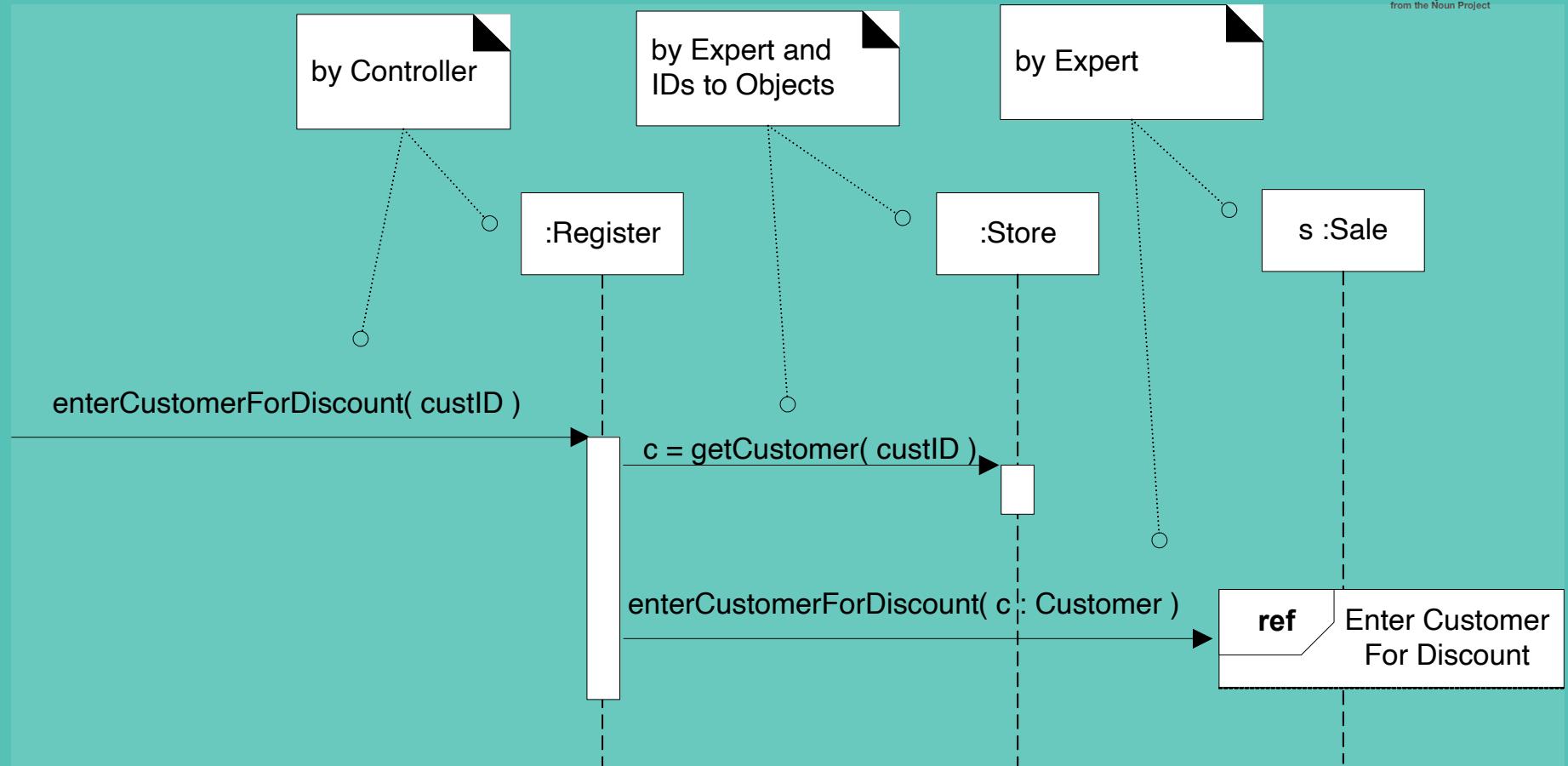


Created by Jonathan Li



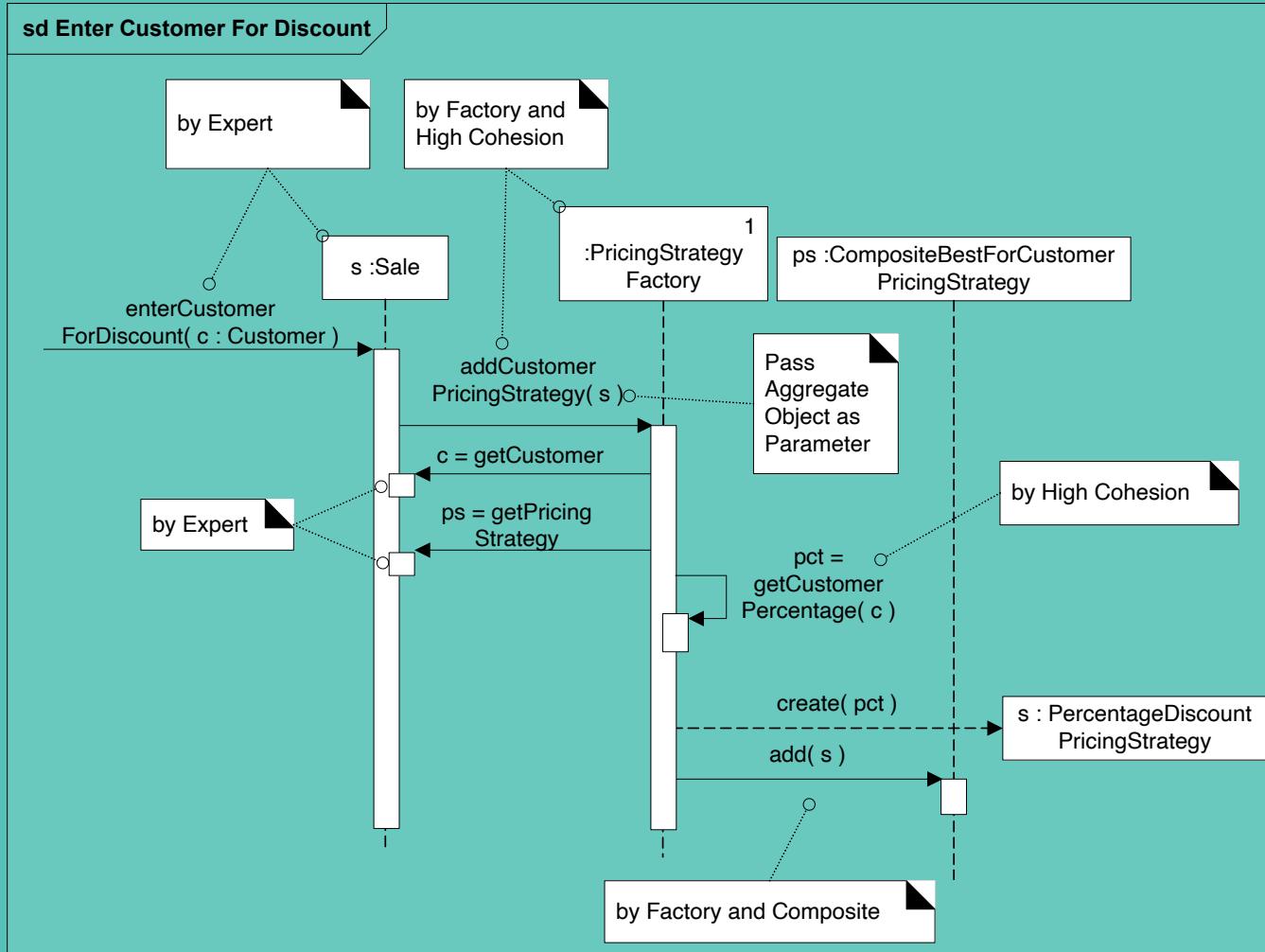
# RÉALISATION DE CAS D'UTILISATION

Created by Jonathan Li  
from the Noun Project

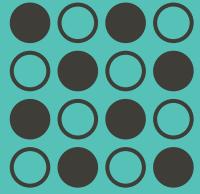


# RÉALISATION DE CAS D'UTILISATION

Created by Jonathan Li  
from the Noun Project



# RÈGLES D'AFFAIRES

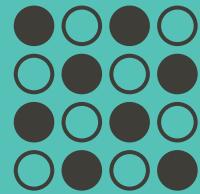


Created by Jonathan Li  
from the Noun Project

- Règles d'affaires personnalisables
- À certains points dans les scénarios
  - e.g., makeNewSale, enterItem
- Exemple : Paiement par bon de cadeau
  - limite l'utilisation d'un bon à un seul item
  - il n'y a pas de monnaie à rendre au client
  - etc.



# IMPLÉMENTATION DES RÈGLES

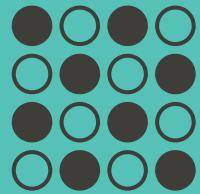


Created by Jonathan Li  
from the Noun Project

- Implémentation des règles est inconnue
  - et peut être extensible
- Voudrait permettre
  - utilisation de plusieurs méthodes
  - patron stratégie/composite
  - interpréteurs de règles « open source »
  - interpréteurs de règles propriétaires (COTS)



# LOG210 SÉANCE #09



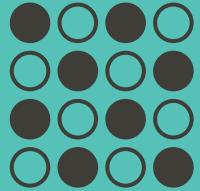
Created by Jonathan Li  
from the Noun Project

## ANALYSE ET CONCEPTION DE LOGICIELS

- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager  6.21m
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite



# ACTUALISATION



Created by Jonathan Li  
from the Noun Project

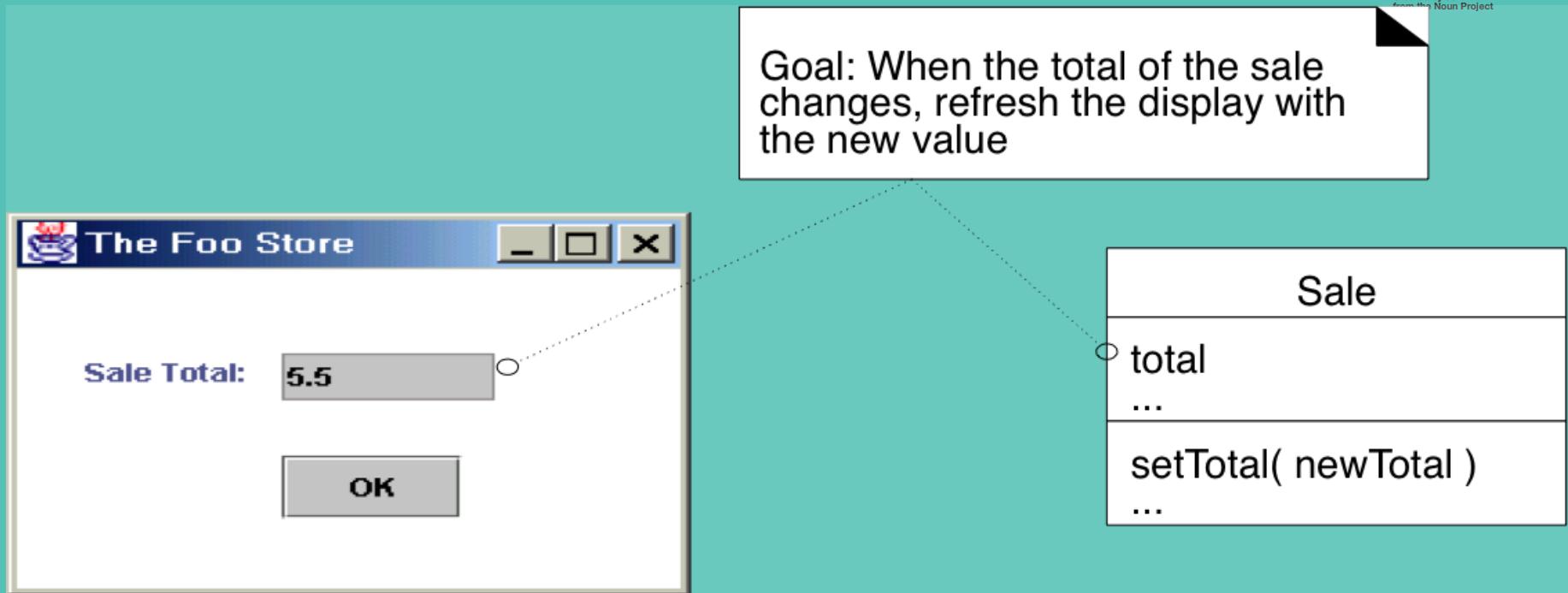
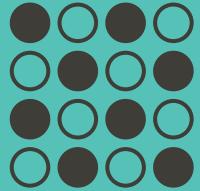


fig. F26.21

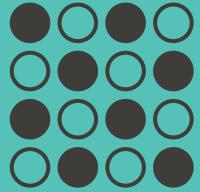
# SOLUTION POSSIBLE...



Created by Jonathan Li  
from the Noun Project

- Total de Sale est changé
  - envoie un message à la fenêtre GUI
  - lui demande de se mettre à jour
- Cette solution n'est pas recommandée
  - Sale ne devrait pas connaître le GUI
    - c'est une dépendance vers une classe dont le code risque de changer (la GUI est peu stable)
- Par exemple
  - si la classe Sale dépend de Swing
  - alors il est difficile de changer l'interface



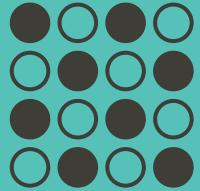


Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - objet observateur
    - réagit selon les changements du sujet
  - objet sujet
    - annonce un changement
- minimiser le couplage du sujet vers l'observateur

## Patron GOF?

# OBSERVATEUR

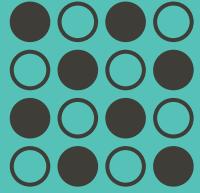


Created by Jonathan Li  
from the Noun Project

- Contexte / Problème
  - objet observateur
    - réagit selon les changements du sujet
  - objet sujet
    - annonce un changement
- minimiser le couplage du sujet vers l'observateur

## Patron GOF?

# OBSERVATEUR

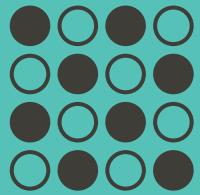


Created by Jonathan Li  
from the Noun Project

- Solution
  - interface observateur
  - les observateurs s'inscrivent au sujet
  - le sujet ne connaît pas les observateurs particuliers
    - seulement des objets implémentant l'interface
  - le sujet envoie un message aux observateurs
    - lorsqu'un certain changement (événement) survient

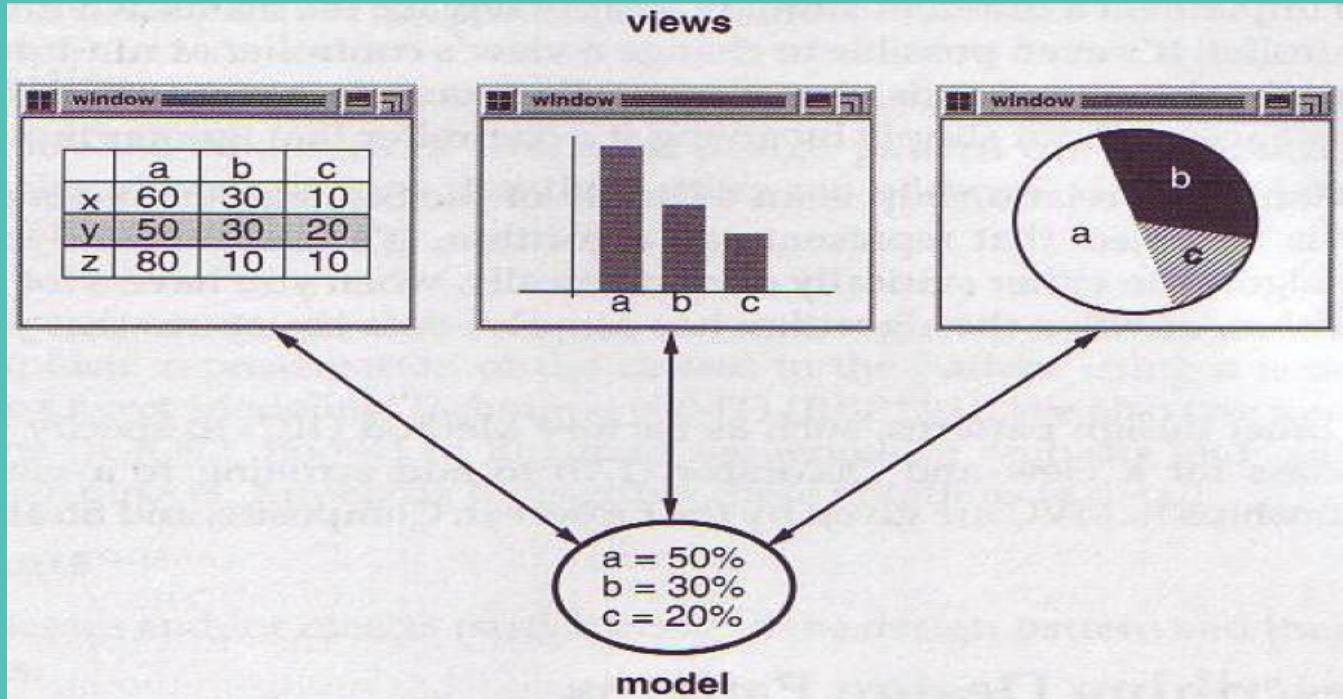


# OBSERVATEUR



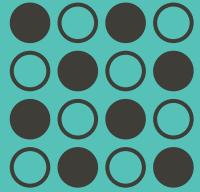
Created by Jonathan Li  
from the Noun Project

## Plusieurs observateurs, un sujet



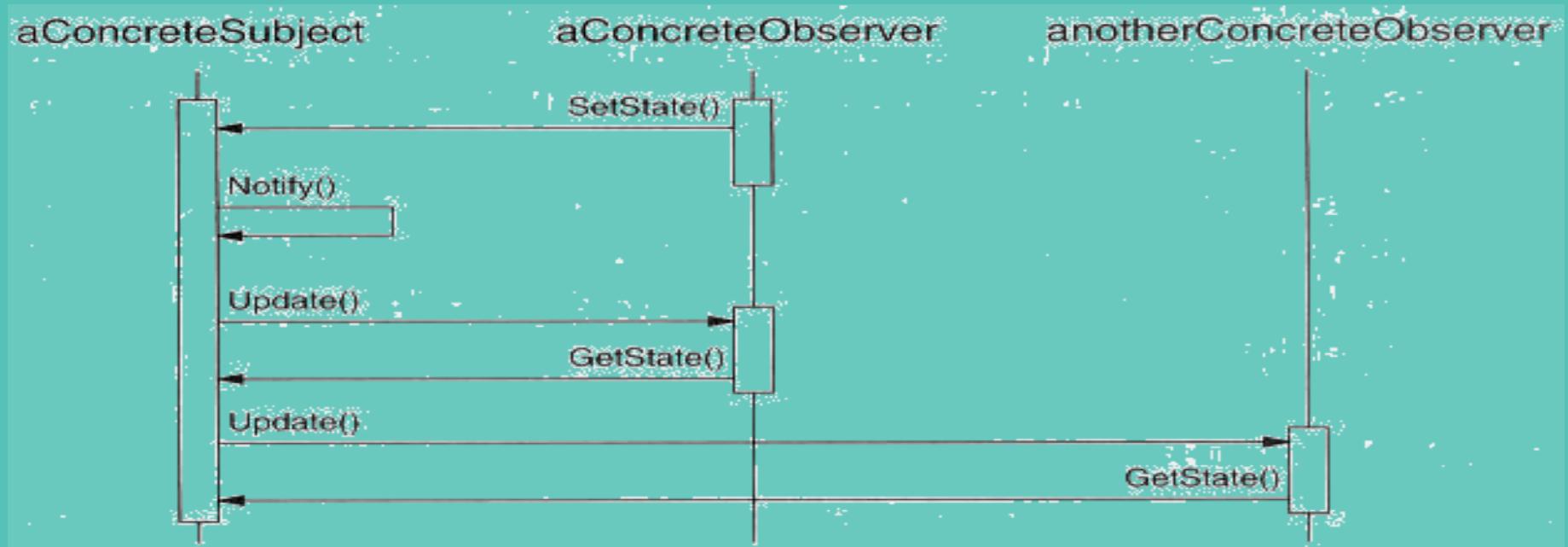
ref: Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995

# OBSERVATEUR

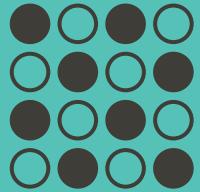


Created by Jonathan Li  
from the Noun Project

## Communication



ref: Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995



Created by Jonathan Li  
from the Noun Project

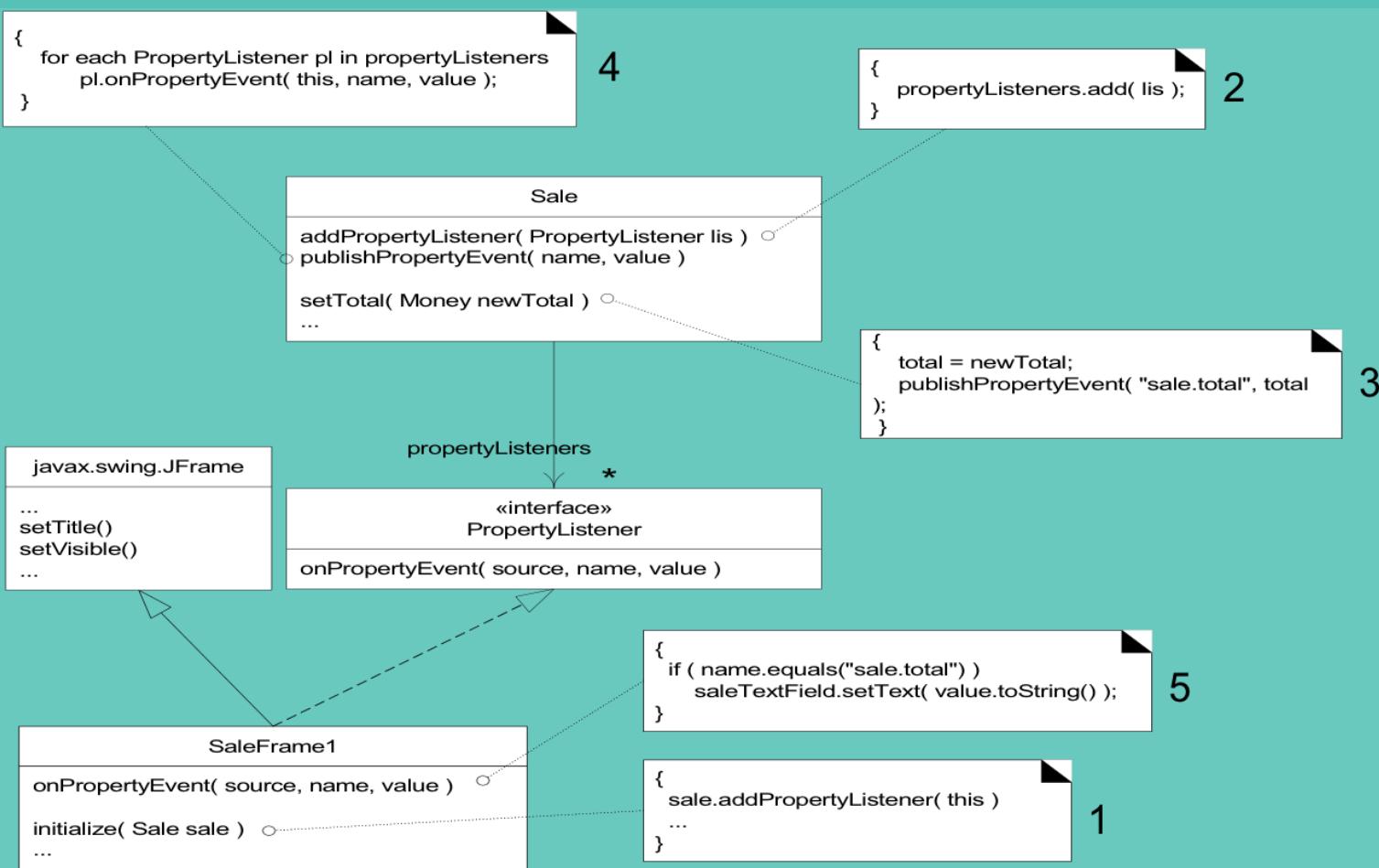


fig. F26.22

# LOG210 SÉANCE #09

## ANALYSE ET CONCEPTION DE LOGICIELS

- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy  16.45m
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite



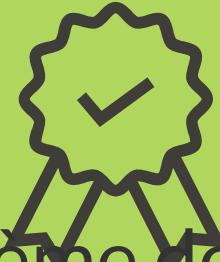
# ATTRIBUTS DE QUALITÉ



- Appliquer les GoF et GRASP dans les RDCU pour traiter les attributs de qualité (exigences URPS)
  - NextGen
    - basculement sur les services locaux,
    - gestion du terminal POS et
    - autorisations de crédit
  - Monopoly
    - différentes cases
    - achat de propriétés
    - paiement de loyers

Created by Bharat  
from the Noun Project

# EXIGENCES NEXTGEN



Created by Bharat  
from the Noun Project

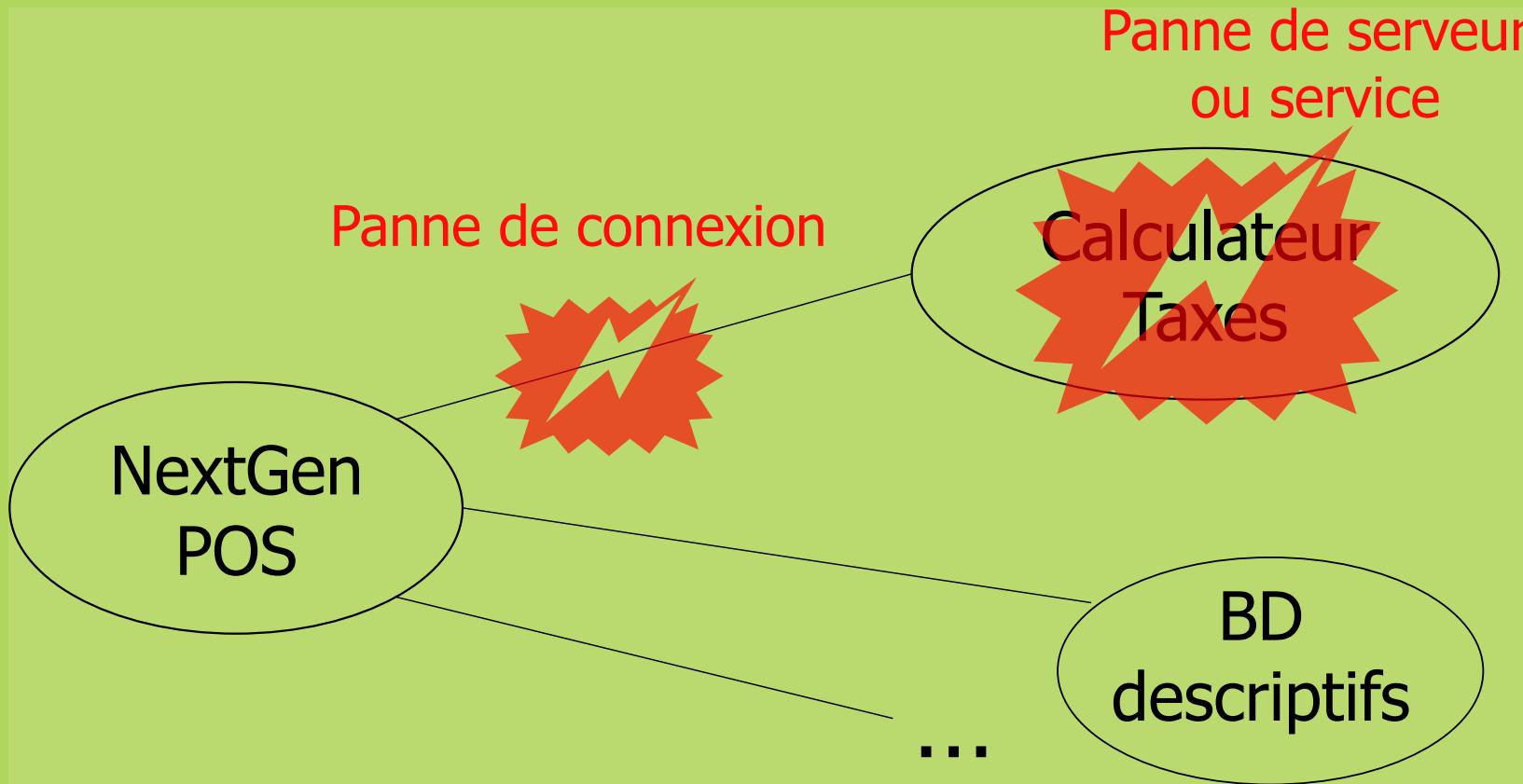
- En cas d'échec d'un service distant, le système doit continuer à fonctionner (tolérance aux pannes, **fiabilité**)
- Mise en cache locale (**performance**)
- Prise en charge d'équipements POS tiers, notamment différents scanners (**adaptabilité**)
- Gestion des paiements par carte de crédit, carte de débit ou chèque (**adaptabilité**)

# PROBLÈME : FIABILITÉ – REPRISE SUR DÉFAILLANCE

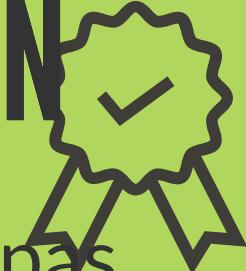


- Tolérance à la panne de services distants
- Plusieurs scénarios de panne

Created by Bharat  
from the Noun Project



# RECOUVREMENT – MOTIVATION



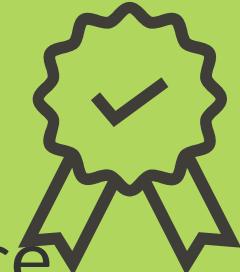
- Les propriétaires des magasin ne veulent pas manquer des ventes en cas de panne
- Le développeur NextGen POS sait que les concurrents n'offrent pas ce degré de fiabilité

Created by Bharat  
from the Noun Project



travailleur

# REPRISE SUR DÉFAILLANCE



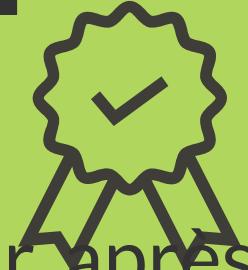
- Récupération robuste en cas de défaillance
  - d'un service distant (calculateur de taxes, gestion des stocks...)
  - de la BD des produits (descriptifs et prix)

Created by Bharat  
from the Noun Project



reprise-defaillance

# PRINCIPE DE RECOUVREMENT



Created by Bharat  
from the Noun Project

Exemple: Pneu de secours permet de continuer, après un délai, perte de performance (pneu plus petit)



# CONSEILS DE L'ARCHITECTE : CACHE LOCAL



- Augmente performance (et permet récupération quand l'accès à BD échoue)
- Utilise un cache local (simple fichier sur disque) des objets DescriptionProduit
- Conséquence : toujours consulter le cache local pour chercher les informations avant d'accéder au service distant
- Principe informatique « Cache »
  - [http://en.wikipedia.org/wiki/Cache\\_\(computing\)](http://en.wikipedia.org/wiki/Cache_(computing))
  - [http://fr.wikipedia.org/wiki/Mémoire\\_cache](http://fr.wikipedia.org/wiki/Mémoire_cache)

Created by Bharat  
from the Noun Project

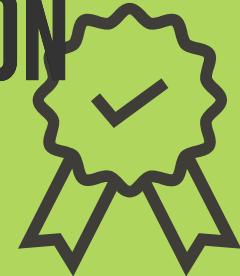
# RECOUVREMENT - PRODUCT SPECIFICATION



- Recherche d'une spécification d'article
  - informations locales
    - performance
    - recouvrement
  - puis requête au système distant

Created by Bharat  
from the Noun Project

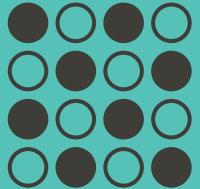
# RECOUVREMENT - PRODUCT SPECIFICATION



Created by Bharat  
from the Noun Project

- Plusieurs niveaux de caches
  - mémoire vive (~1000)
    - performance
  - disque dur (~100 MB)
    - pannes de courant
  - serveur local au commerce
    - indépendance au réseau
  - web
  - etc.

# RAPPEL – ADAPTATEURS



Created by Jonathan Li  
from the Noun Project

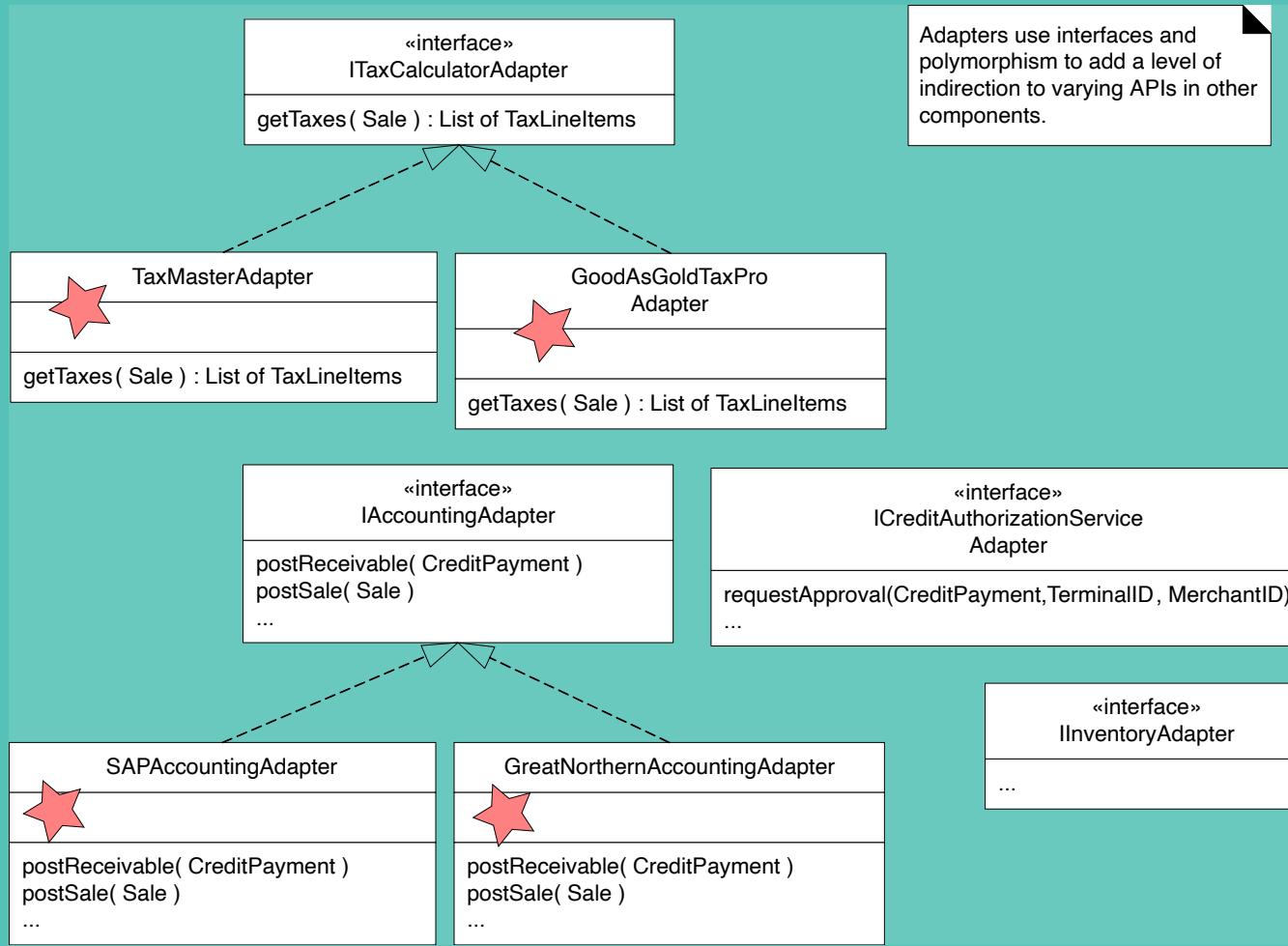
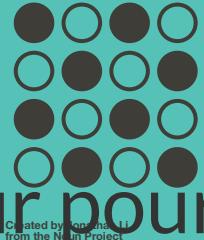


fig. F23.1

# RAPPEL – ADAPTATEURS

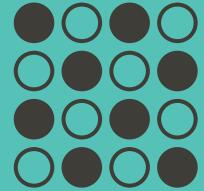


- FabriqueDeServices retourne un adaptateur pour accéder aux variantes de services BD Produits



FabriqueService

# RAPPEL – ADAPTATEURS

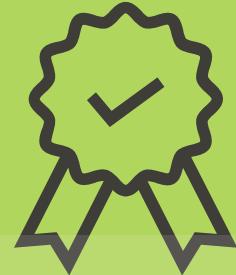


Created by Jonathan Li  
from the Noun Project

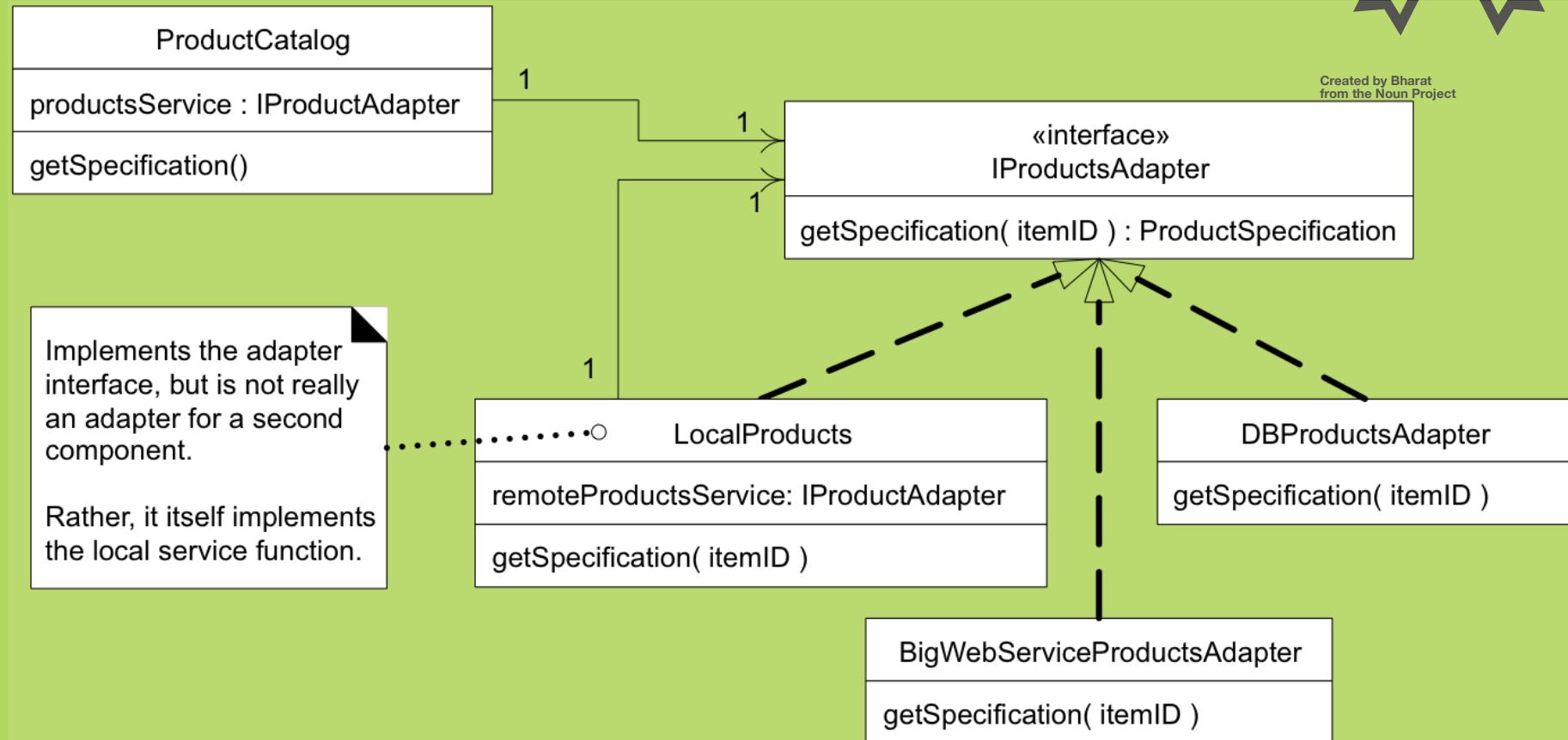
- FabriqueDeService retourne un adaptateur pour accéder à la BD Produits



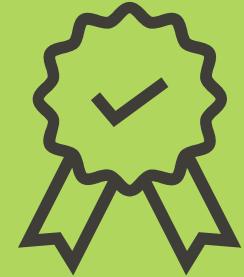
# DESIGN DE RECOUVREMENT



Created by Bharat  
from the Noun Project

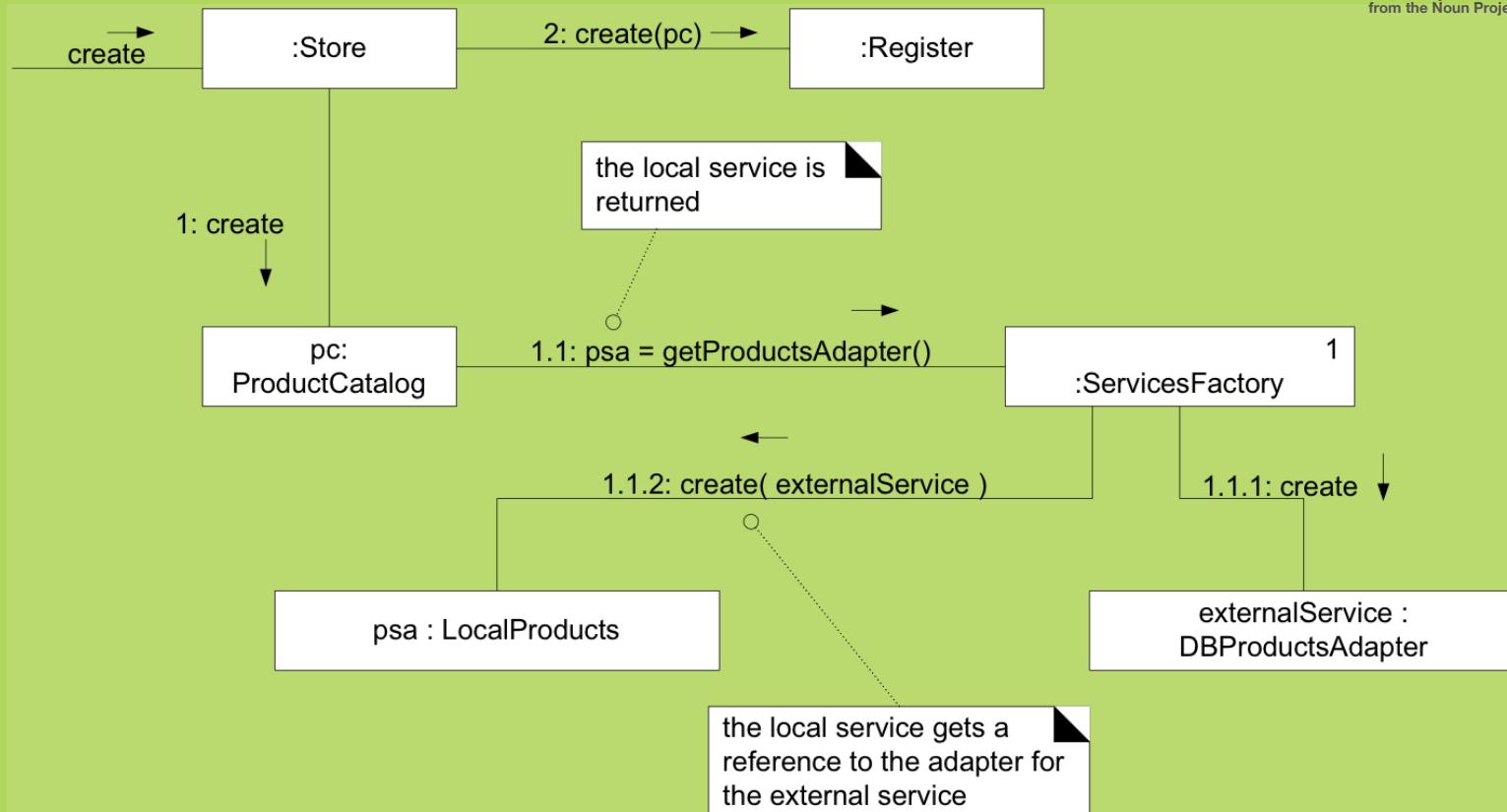


# DESIGN DE RECOUVREMENT

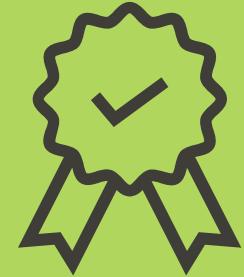


## Initialisation

Created by Bharat  
from the Noun Project



# DESIGN DE RECOUVREMENT



## enterItem

Created by Bharat  
from the Noun Project

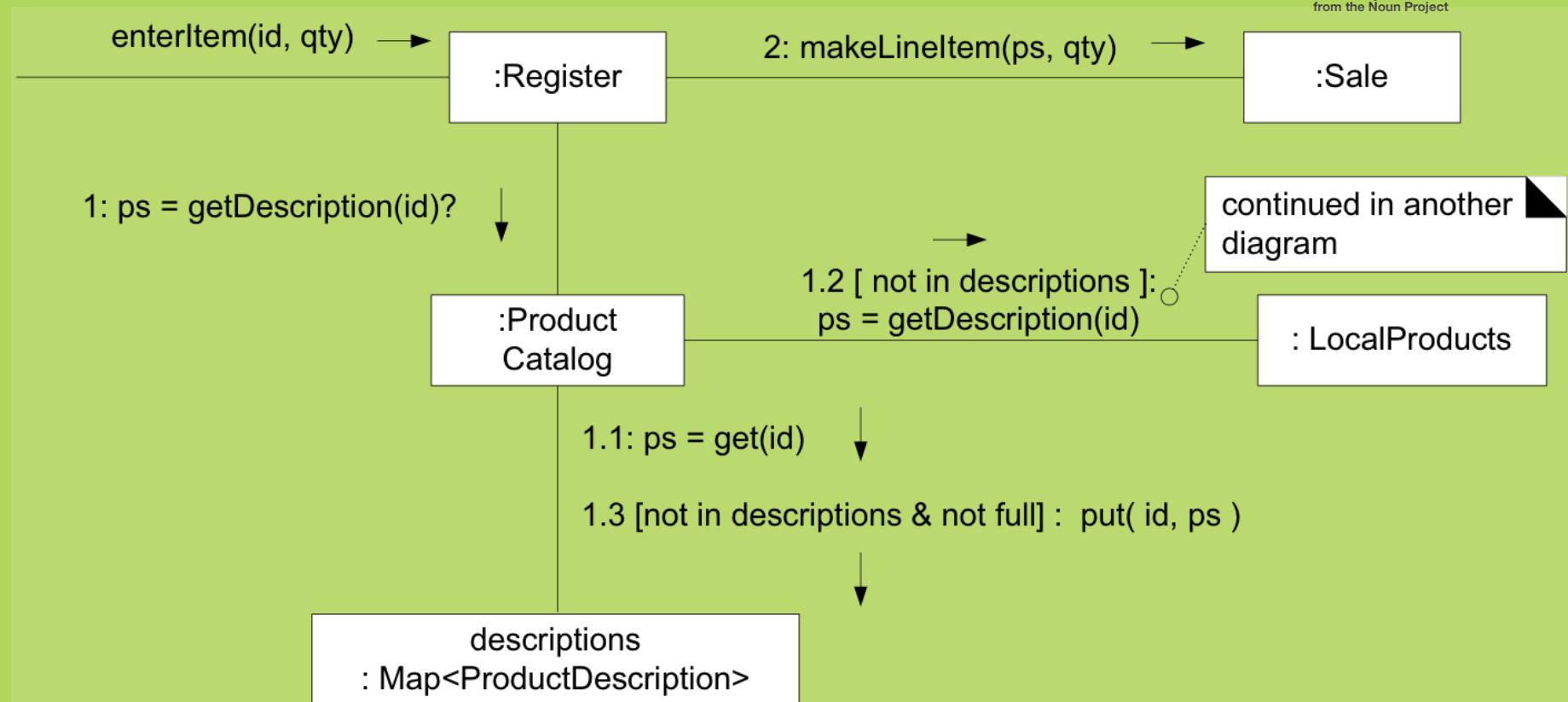
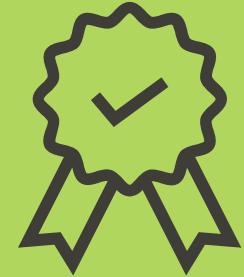


fig. F30.3

# DESIGN DE RECOUVREMENT



enterItem...

Created by Bharat  
from the Noun Project

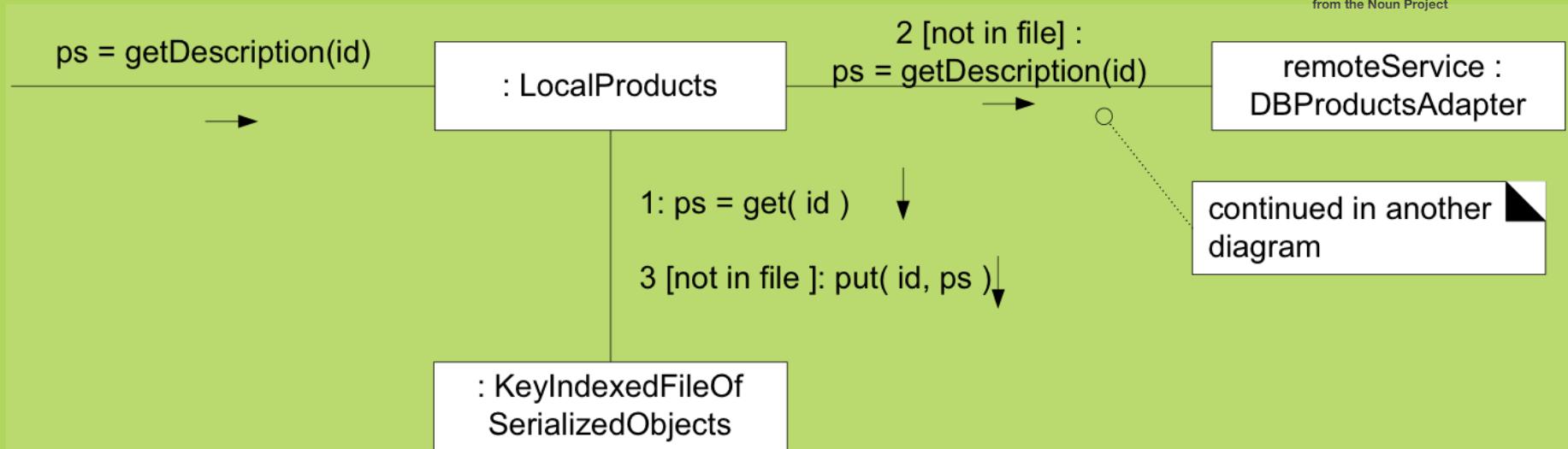
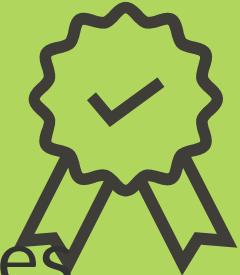


fig. F30.4

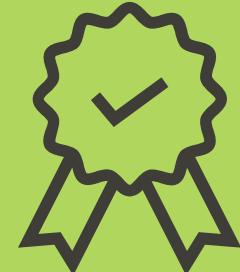
# MISE EN MÉMOIRE CACHE



- Approches pour initialiser les caches
  - Paresseuse
    - au fur et à mesure
  - Stricte
    - à l'initialisation

Created by Bharat  
from the Noun Project

# COHÉRENCE DE CACHE



- Les prix peuvent changer
- Incohérence (niveaux de cache ↔ serveur externe)
- Mise-à-jour des caches
  - initiée par le serveur
  - initiée par le client

Created by Bharat  
from the Noun Project

# LOG210 SÉANCE #09

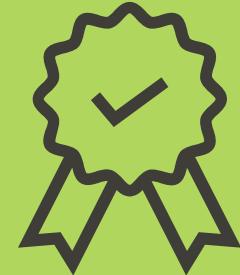


## ANALYSE ET CONCEPTION DE LOGICIELS

- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception ← 14.33m
- Patron faire soi-même
- Attention à la patternite



# TOLÉRANCE AUX FAUTES

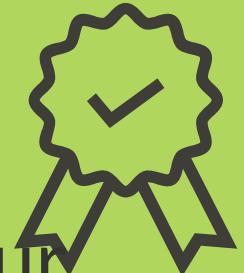


## Définitions

Created by Bharat  
from the Noun Project

- Faute
  - source d'un mauvais comportement
- Erreur
  - manifestation de la faute dans le système en fonctionnement
- Échec (Défaillance)
  - service échoue à cause de l'erreur

# FAUTE VS. ERREUR (1)



Faute ne produit pas toujours une erreur

Created by Bharat  
from the Noun Project

 fauteVSerreur

ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

# FAUTE VS. ERREUR (2)



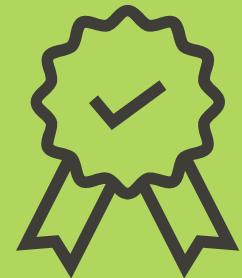
L'erreur est la manifestation de la faute dans le système

Created by Bharat  
from the Noun Project



ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

# SOURCE D'UNE FAUTE



## Problème algorithmique

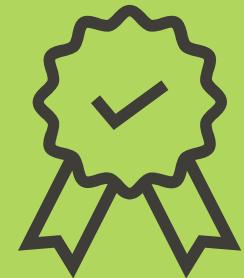
Created by Bharat  
from the Noun Project



ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

# SOURCE D'UNE FAUTE

Problème environnemental



Created by Bharat  
from the Noun Project



source-faute-environnement

ref: « Object-Oriented Software Engineering » BerndBruegge, B.Bruegge, AllenDutoit

# ENCHAÎNEMENT



Created by Bharat  
from the Noun Project

- Une faute peut provoquer une erreur, qui peut provoquer une défaillance
- Vue d'un niveau plus haut, la défaillance peut être aperçue comme une faute

 enchainement

# TRAITEMENT DES DÉFAILLANCES



- Quand rien ne va plus: *article n'est pas dans la cache et service web n'est pas accessible*
- Intervenants résolvent le problème: *entrée à la main du prix et de la description*



# TRAITEMENT DES ERREURS </>

- Exceptions
- Utiles pour des ressources telles que
  - disque dur
  - mémoire
  - réseau
  - base de donnée
  - services externes



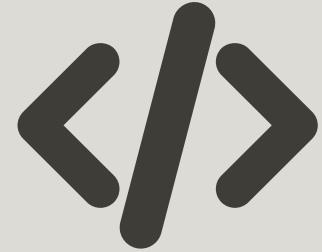
# EXCEPTIONS: GRANDES LIGNES

- Patrons

- « Donner le nom du problème et non celui de l'objet qui lance l'exception »
- « Convertir exceptions »
- « Journal centralisé d'erreurs »
- « Dialogue d'erreur »



# CONVERTIR EXCEPTIONS



- Comment présenter l'exception?
- Niveau d'abstraction
  - équivalent : faute -> erreur -> défaillance (faute)
- Exception de haut niveau
  - encapsule exception de bas niveau



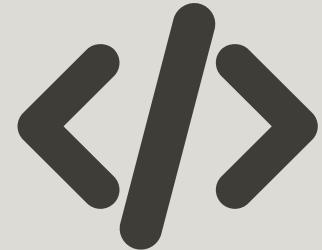
# JOURNAL CENTRALISÉ D'ERREURS



- Objet singleton global
  - lui acheminer toutes les exceptions
  - flexibilité des définitions de flux de sortie
- `java.util.logging` (depuis JDK 1.4)



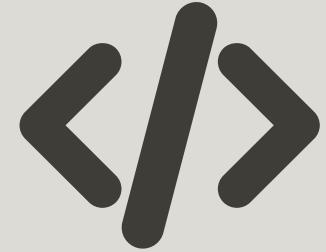
# DIALOGUE D'ERREUR



- Objet singleton
  - mécanisme centralisé de notification des erreurs
  - pour notifier l'utilisateur
  - protection contre les variations dans le mécanisme de sortie
  - pas un objet de l'IU
  - indépendant de l'application



# PATRON: DIALOGUE D'ERREUR



patron-dialog-erreur

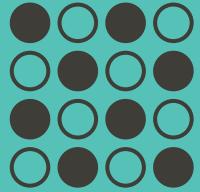
fig A35.11 p 592

# PROBLÈME - SERVICE NON DISPONIBLE



Created by Bharat  
from the Noun Project

- Dans le cas où on doit envoyer les ventes au services de comptabilité, il faut agir le plus rapidement possible, donc directement communiquer avec les services de comptabilité externes.
- Dans l'éventualité où le service comptable est temporairement non disponible, comment peuvent-on éviter que cette faute devienne une défaillance.

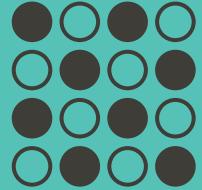


Created by Jonathan Li  
from the Noun Project

- Problème
  - l'accès direct à un objet est impossible (ou est indésirable). Que faire?
- Solution
  - Utiliser un objet proxy qui est un substitut à un objet
  - Le proxy implémente la même interface
  - Cela ajoute un niveau d'indirection



# PATRON PROXY (GOF)

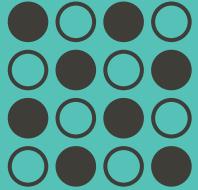


Created by Jonathan Li  
from the Noun Project

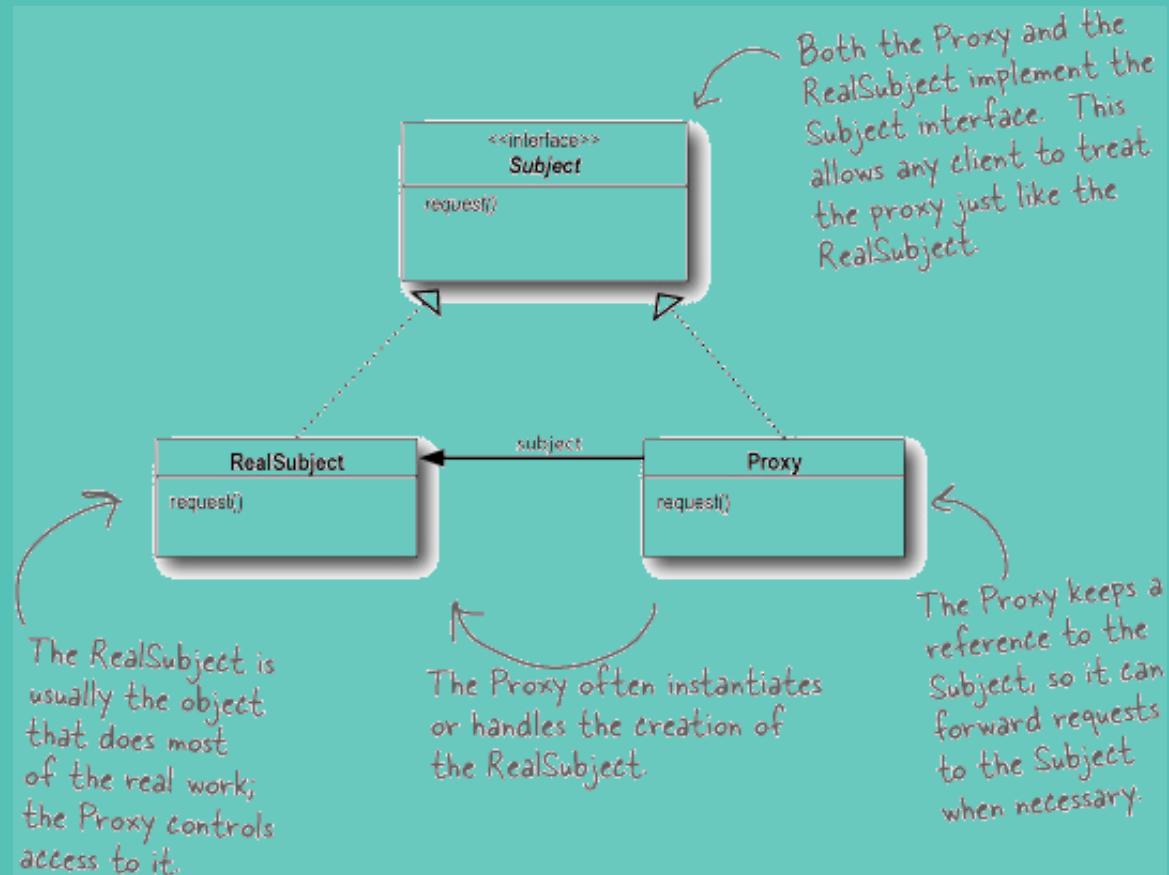
- Problème
  - l'accès direct à un objet est impossible (ou est indésirable). Que faire?
- Solution
  - Utiliser un objet proxy qui est un substitut à un objet
  - Le proxy implémente la même interface
  - Cela ajoute un niveau d'indirection



# PROXY

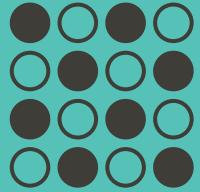


Created by Jonathan Li  
from the Noun Project

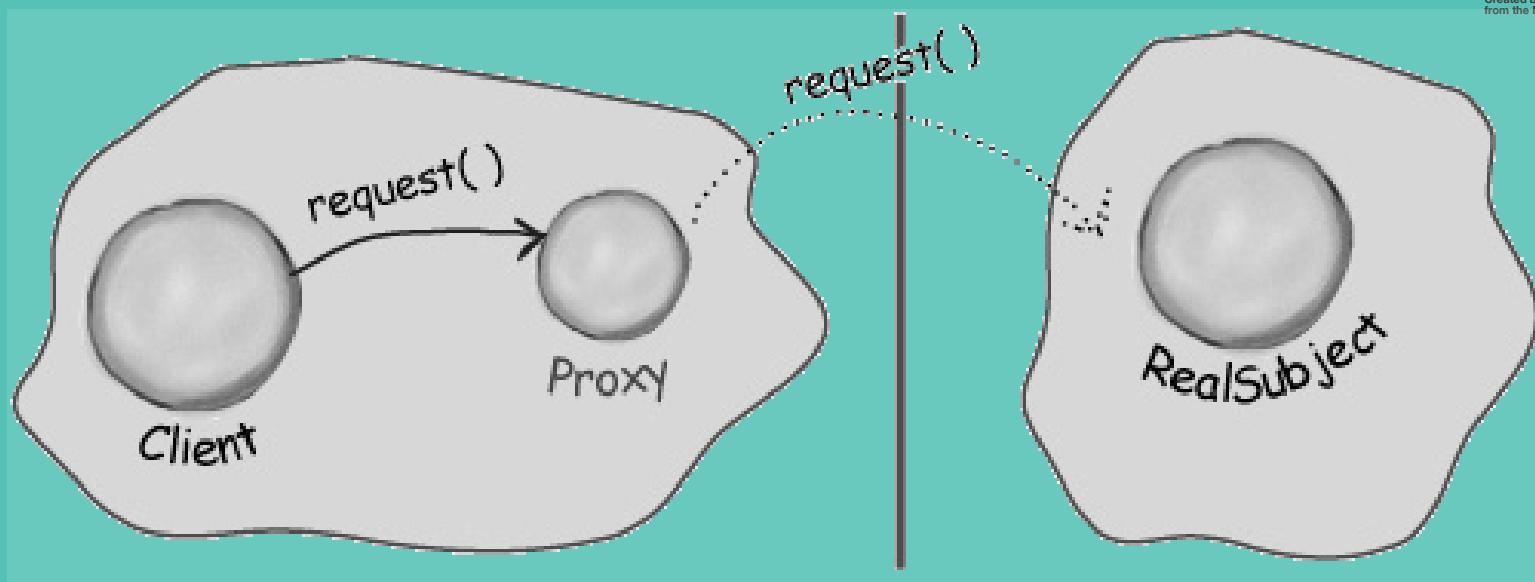


Freeman, Eric; Robson, Elisabeth; Bates, Bert; Sierra, Kathy. Head First Design Patterns.

# PROXY



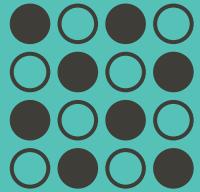
Created by Jonathan Li  
from the Noun Project



We know this diagram  
pretty well by now...

Freeman, Eric; Robson, Elisabeth; Bates, Bert; Sierra, Kathy. Head First Design Patterns.

# PATRON PROXY (GOF)



Created by Jonathan Li  
from the Noun Project

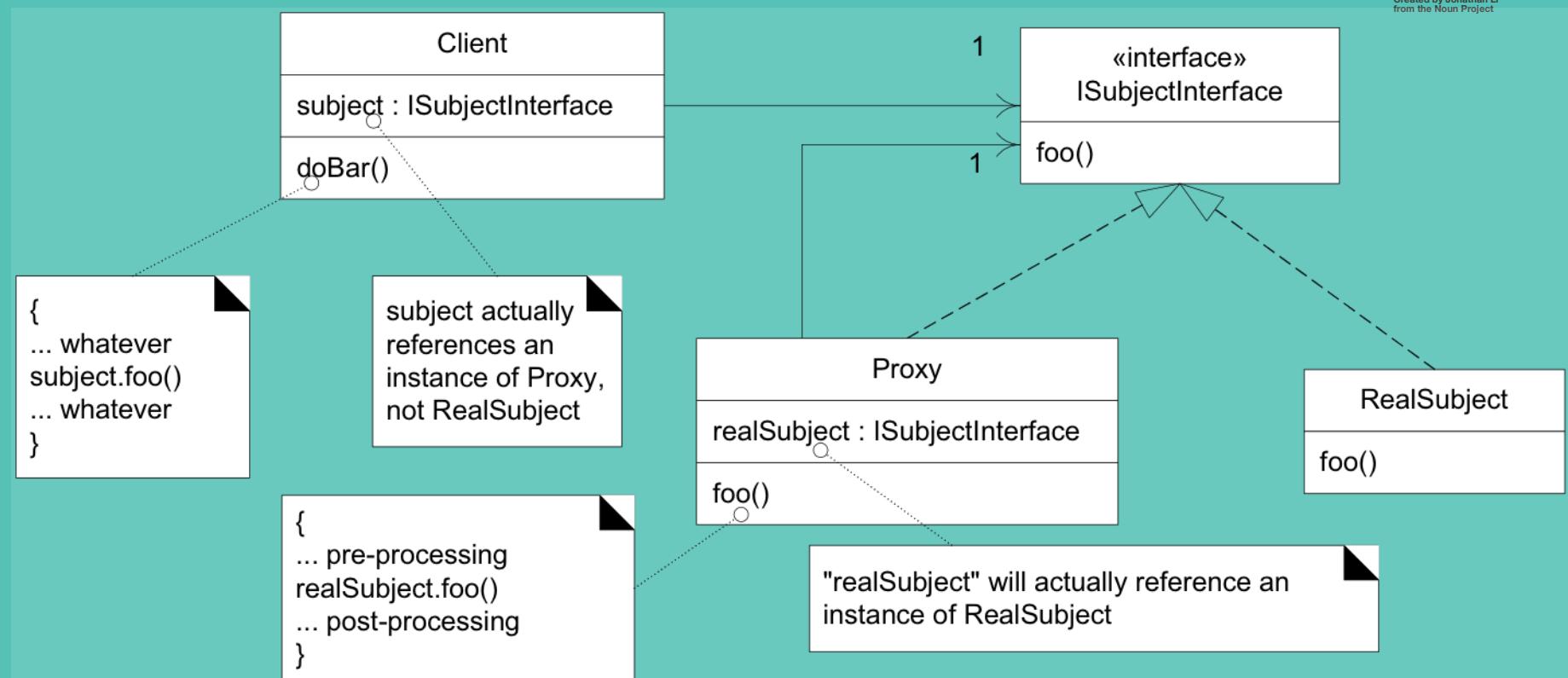
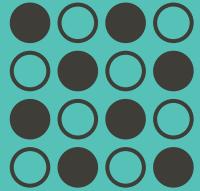
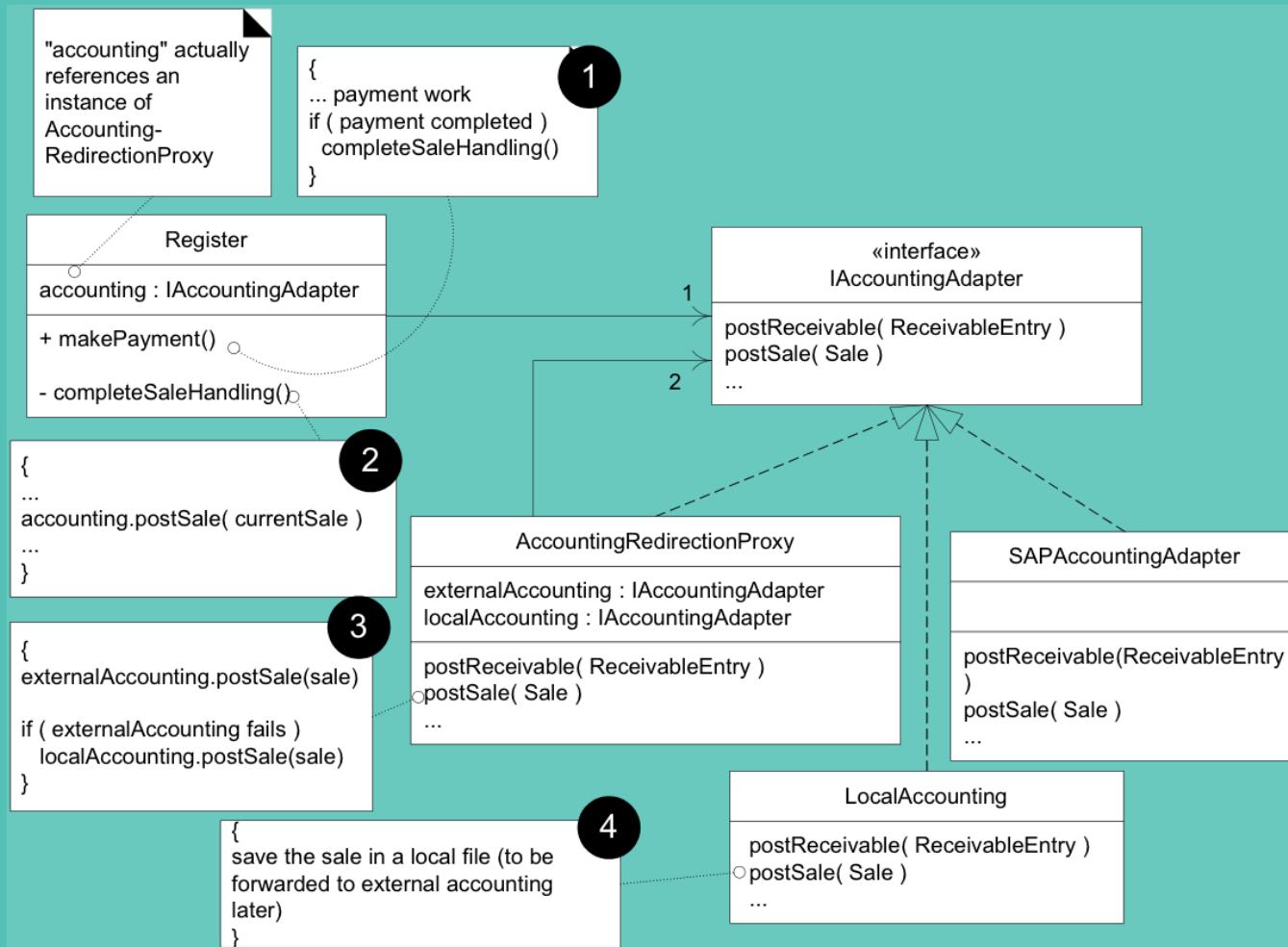


fig. F30.12

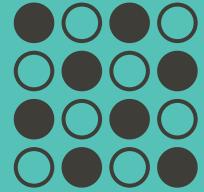
# LES PROXY DANS POS



Created by Jonathan Li  
from the Noun Project

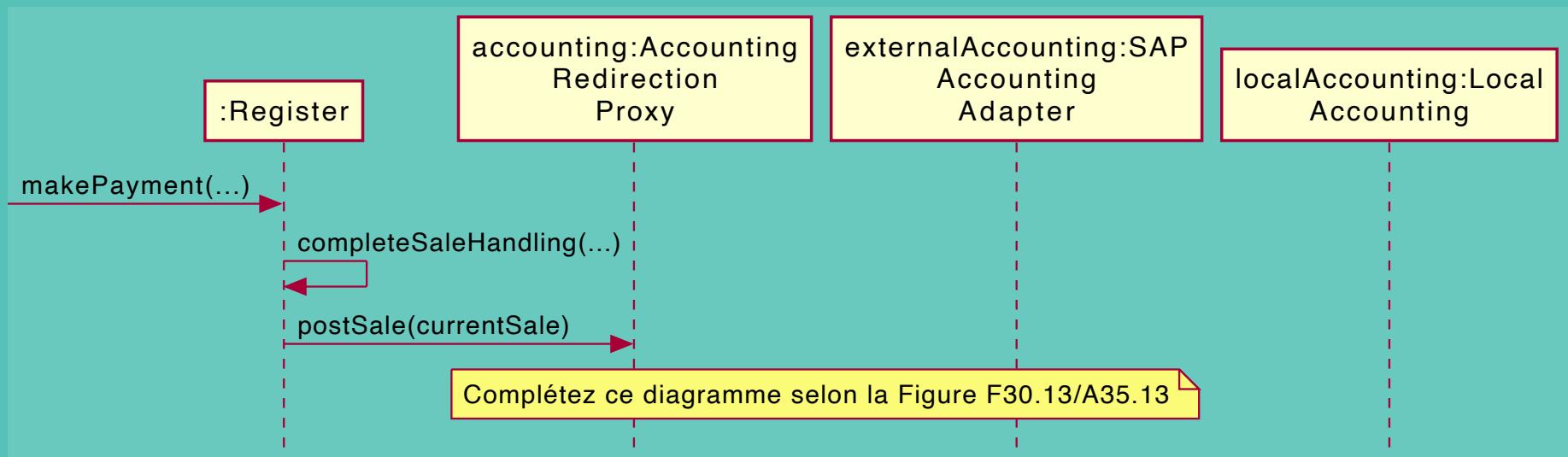


# PROXY DE REDIRECTION



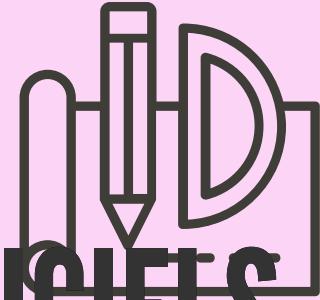
Created by Jonathan Li  
from the Noun Project

## Exercice



# LOG210 SÉANCE #09

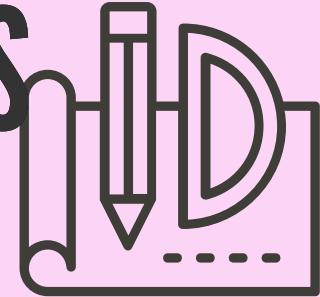
## ANALYSE ET CONCEPTION DE LOGICIELS



- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même ←
- Attention à la patternite

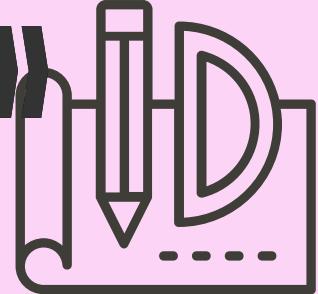


# TRAITEMENT DE PAIEMENTS



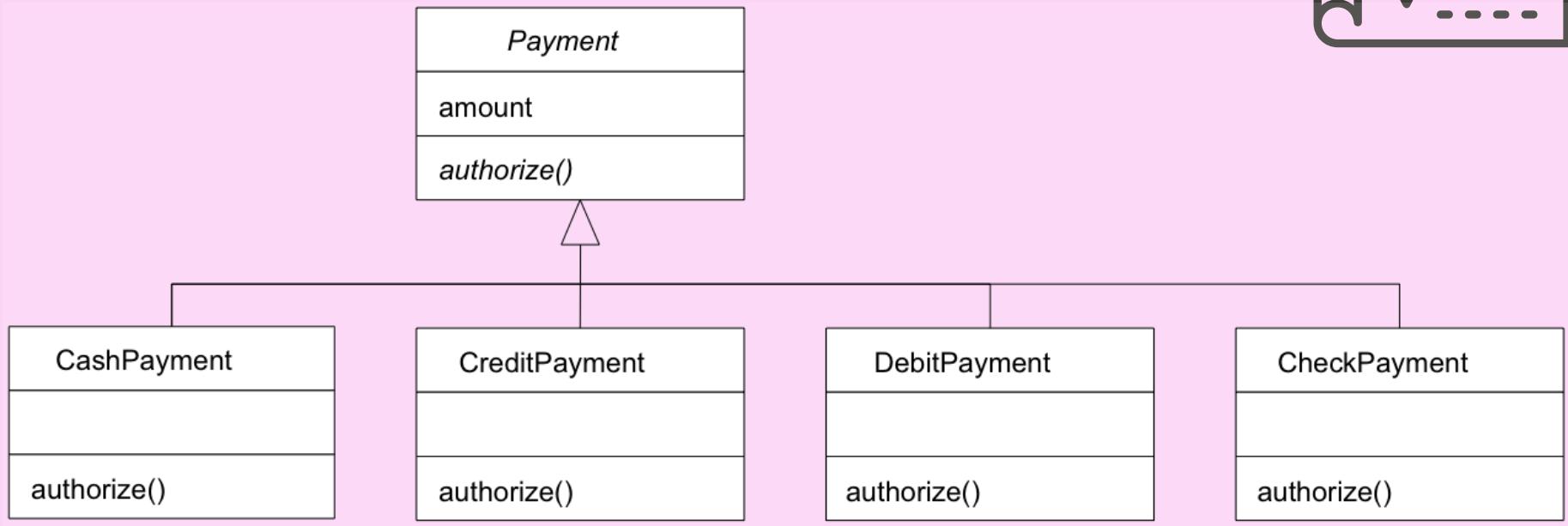
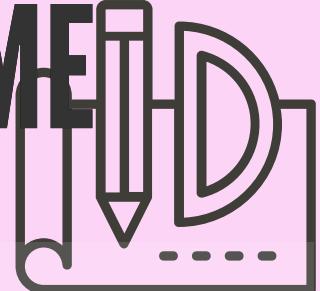
- Multiple paiements
  - chèque
  - carte de crédit
  - carte de débit
  - comptant

# PATRON « FAIRE SOI-MÊME »



- Objet logiciel
  - abstraction d'un objet réel
  - fait ce qui est normalement fait à l'objet réel
  - avantage: réduire le décalage des représentations
- Exemple
  - objet Texte qui vérifie sa propre orthographe

# PAIEMENT ET POLYMORPHISME



By Polymorphism, each payment type should authorize itself.

This is also in the spirit of "Do it Myself" (Coad)

fig. F33.17

# PAIEMENT PAR CHÈQUE

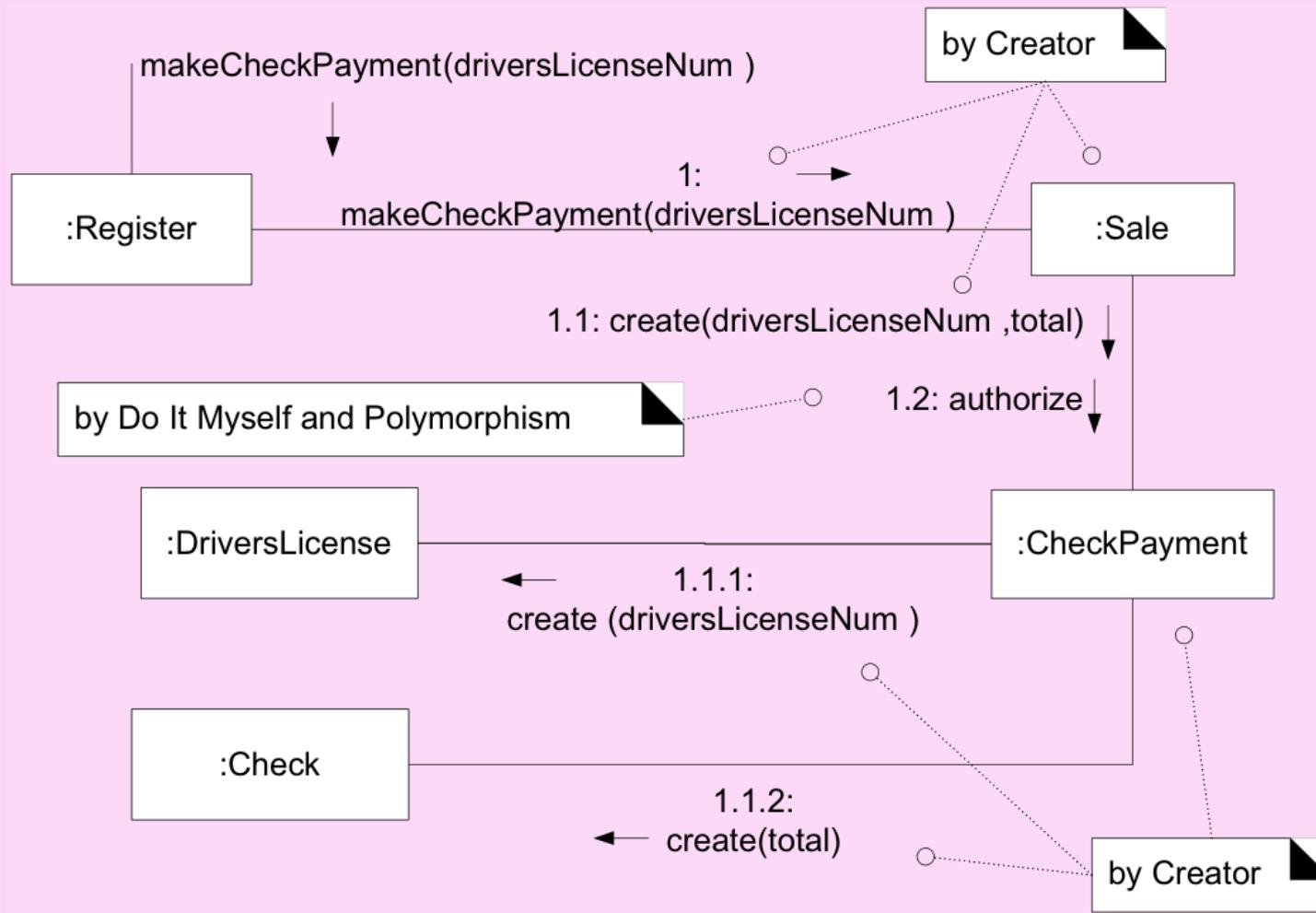
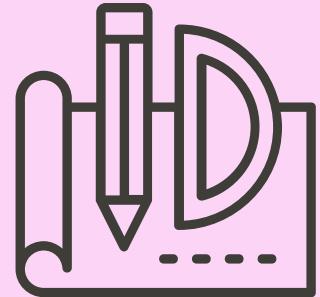


fig. F30.19

# PAIEMENT PAR CARTE DE CRÉDIT

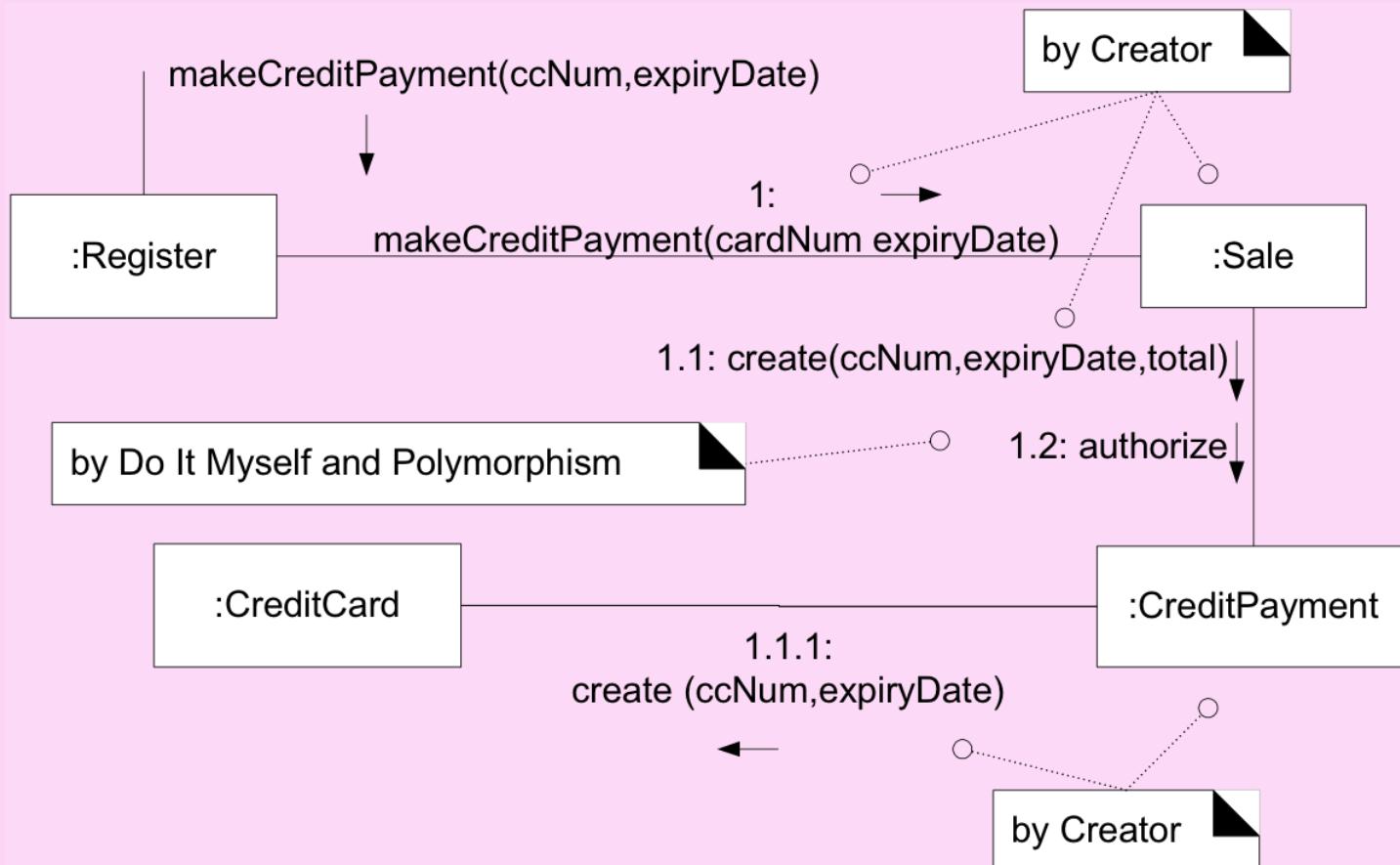
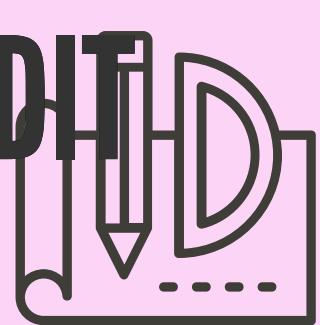


fig. F30.18

# AUTORISATION

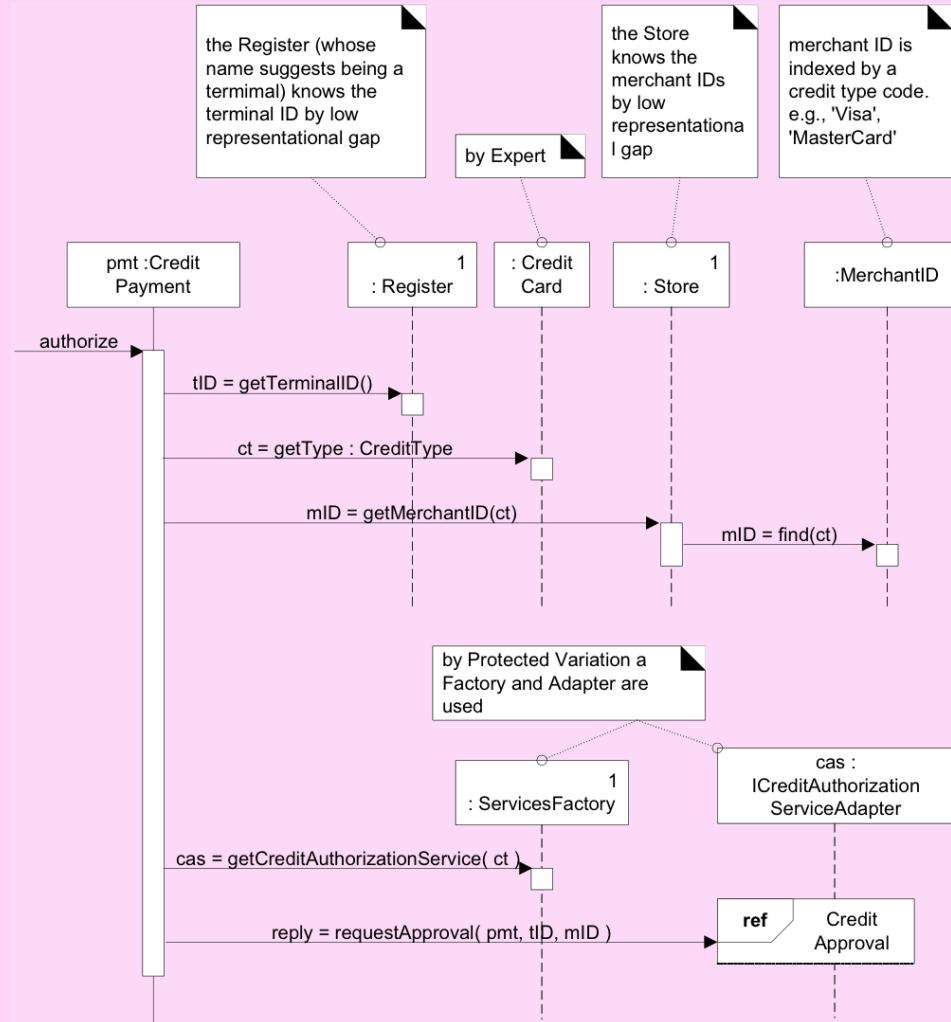
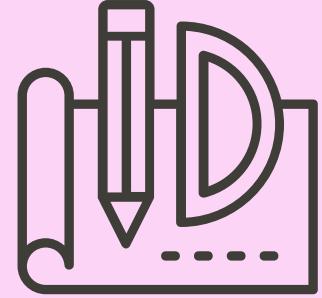


fig. F30.20

# AUTORISATION

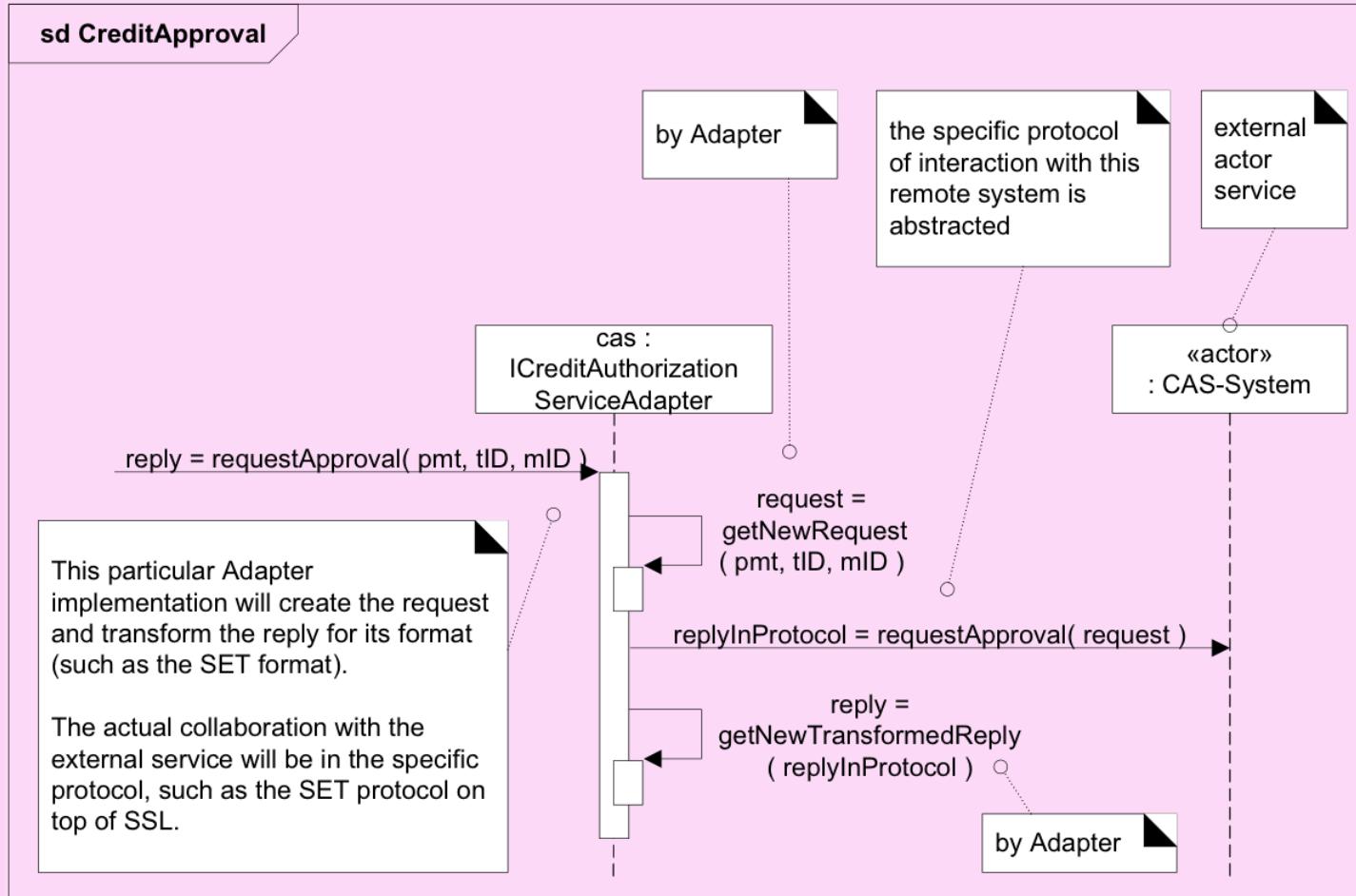
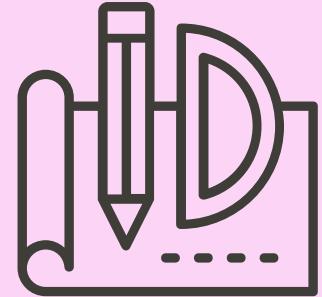


fig. F30.21

# PAIEMENT APPROUVÉ

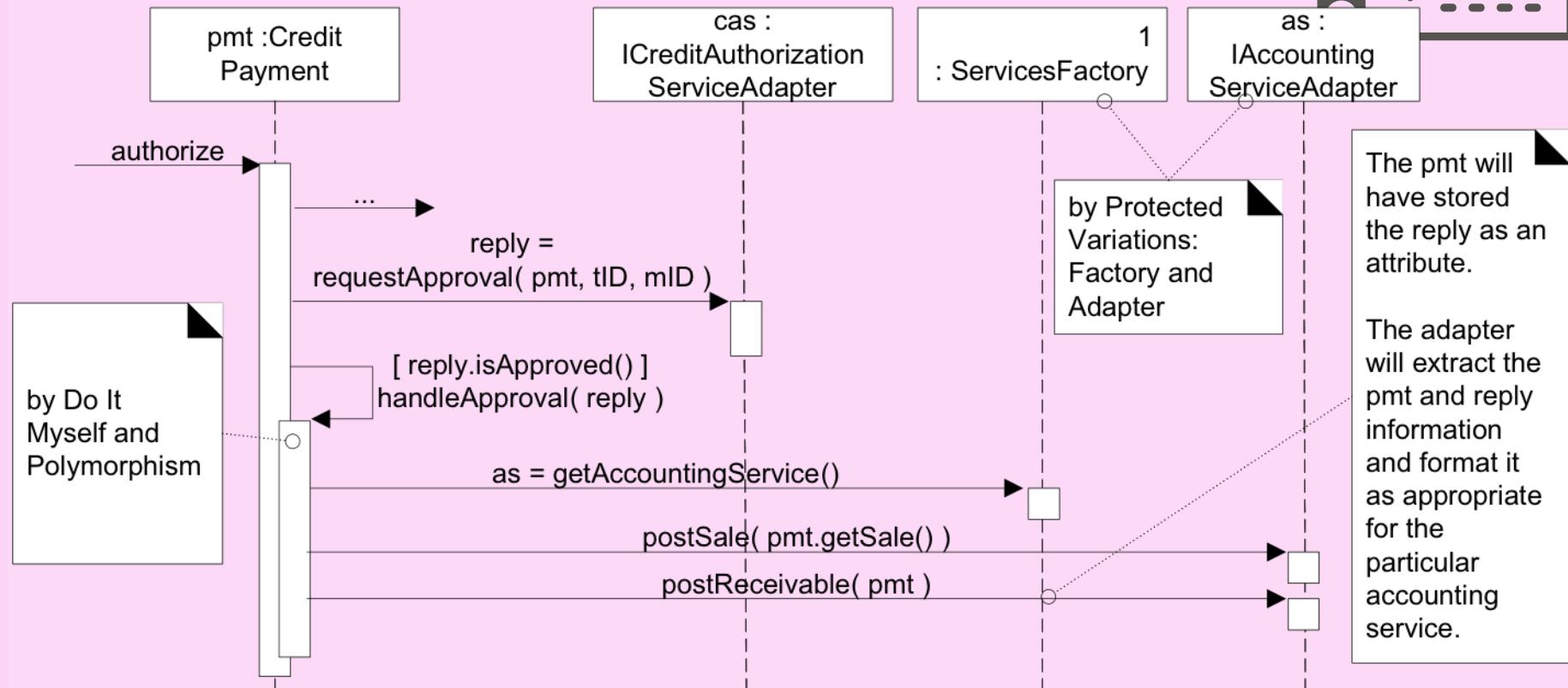
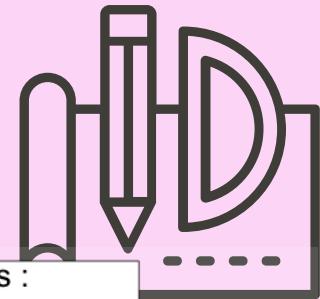
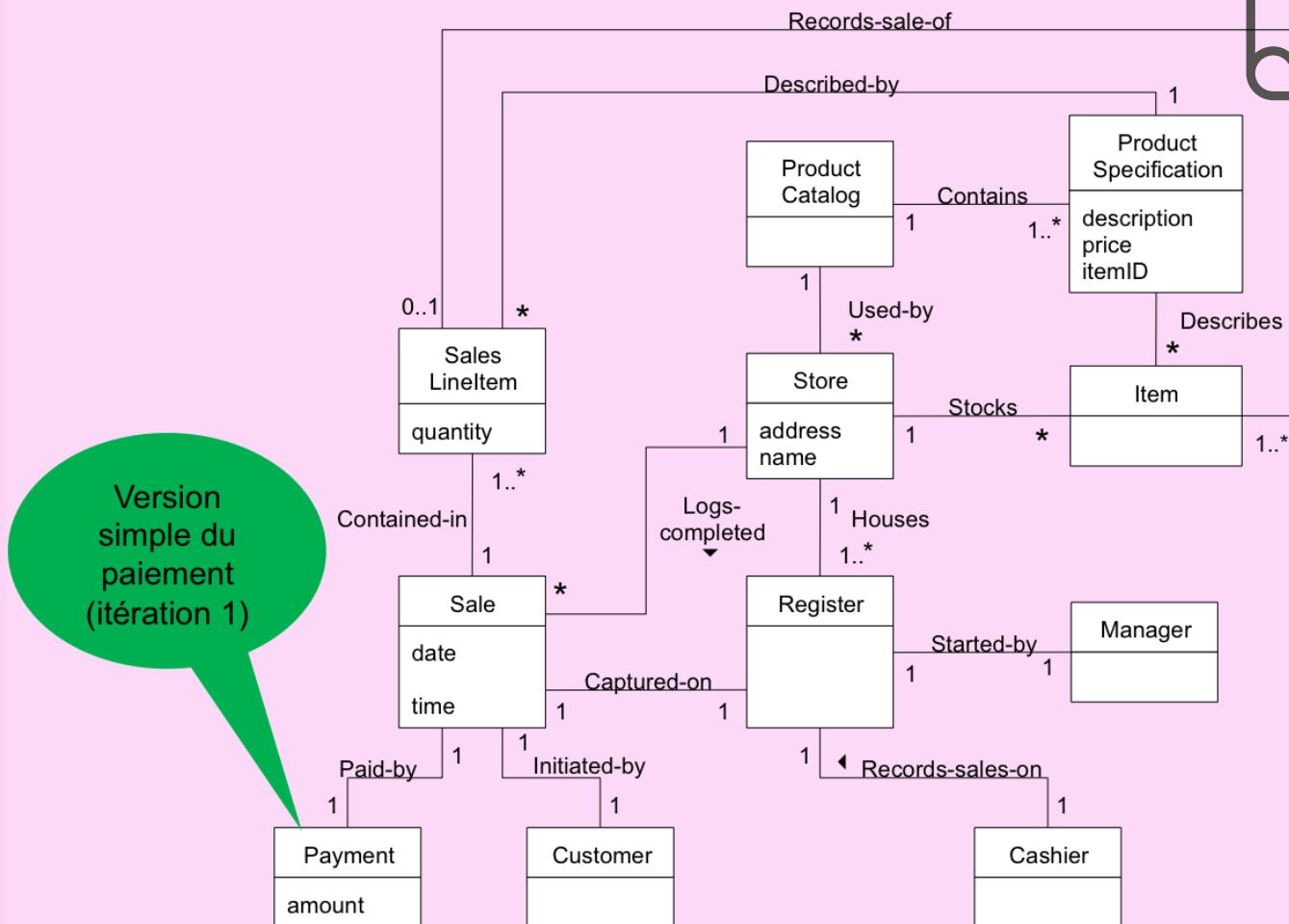
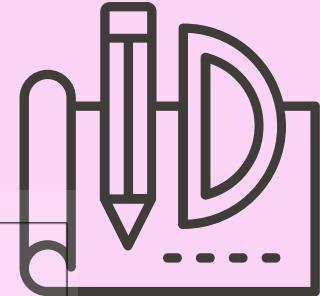
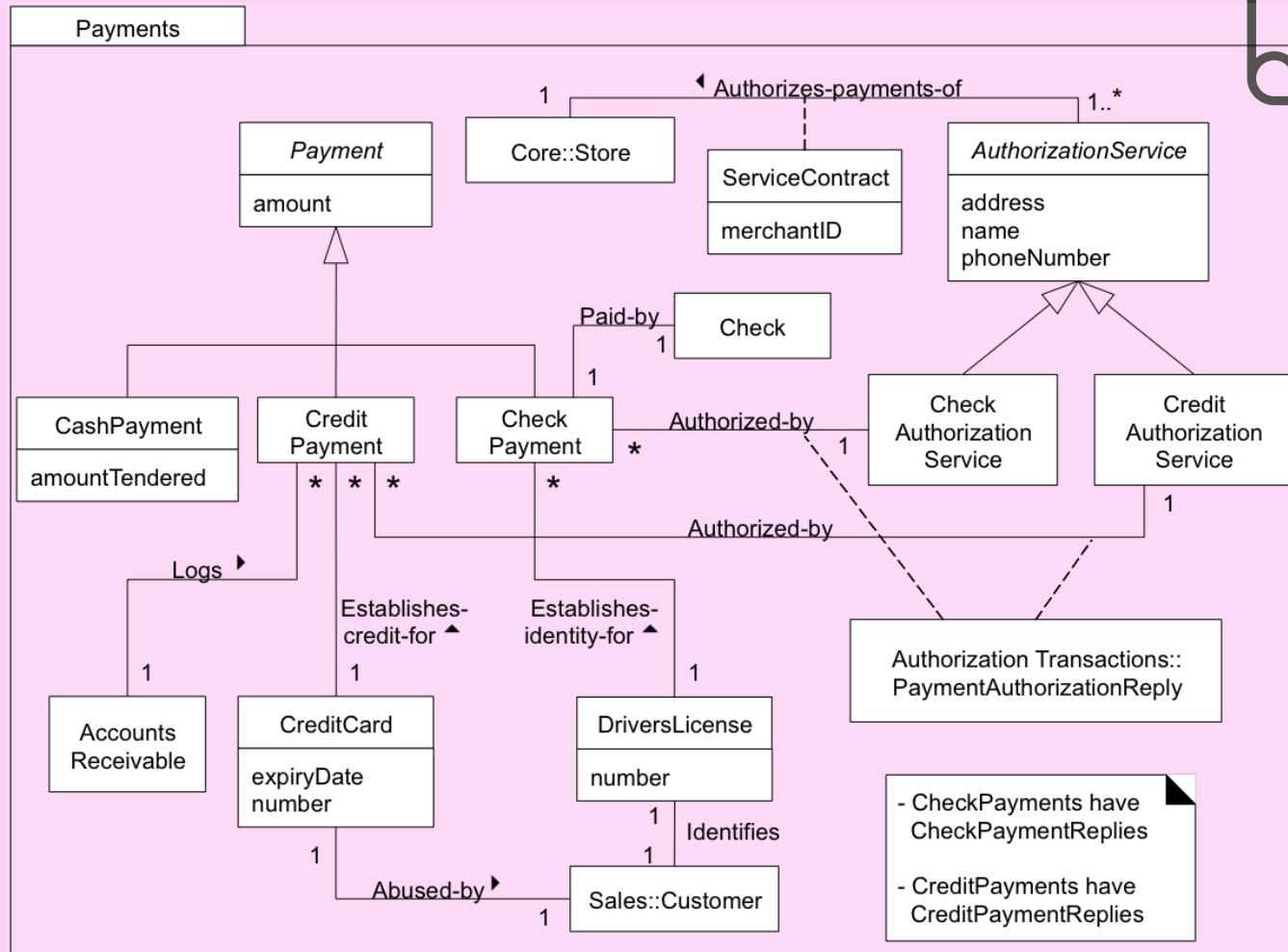
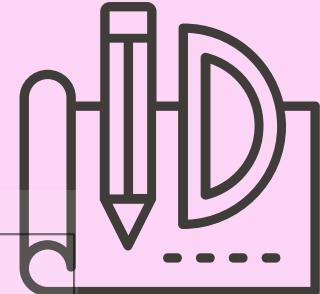


fig. F30.22

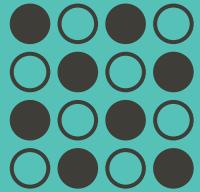
# VERSION SIMPLE DU PAIEMENT



# PACKAGE: PAIEMENT



# LOG210 SÉANCE #09



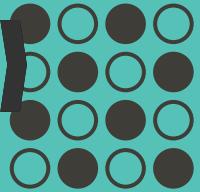
Created by Jonathan Li  
from the Noun Project

## ANALYSE ET CONCEPTION DE LOGICIELS

- Découverte des patrons GOF
- GRASP sont une généralisation d'autres patterns!
- Adaptateur, Fabrique concrète, Singleton
- Logique de tarification élaborée avec patron Stratégie et Composite
- Actualisation interface usager
- Recouvrement avec Proxy
- Faute, erreur, échec et exception
- Patron faire soi-même
- Attention à la patternite



# ATTENTION À LA « PATTERNITE »

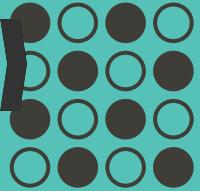


Created by Jonathan Li  
from the Noun Project

« Patternite » : tendance de rentrer de force dans les patterns GoF  paternite



# ATTENTION À LA « PATTERNITE »



Created by Jonathan Li  
from the Noun Project

- Tout pattern GoF amène une complexité
- Est-ce que les bénéfices du pattern justifient la complexité?
- La simplicité (KISS) est aussi une forme d'élégance.

 paternites



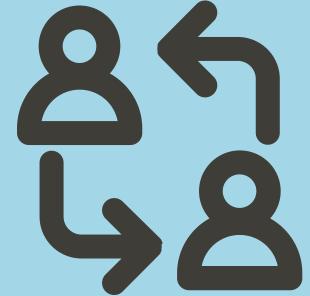
# ATTENTION À LA « PATTERNITE »

## Connaître vos patrons de conception



# SÉANCE #09

## RÉTROACTION: PAGE D'UNE MINUTE



Created by Prithvi  
from the Noun Project

1. Quels sont les deux [trois, quatre, cinq] plus importants [utiles, significatives, surprenantes, dérangeantes] choses que vous avez apprises au cours de cette session?
2. Quelle (s) question (s) reste (s) en tête dans votre esprit?
3. Y a-t-il quelque chose que tu n'as pas compris?

<https://1drv.ms/u/s!An6-F73ulxAOhVyiCB46jTeINVLS>

