

Coding Week

École Centrale
Casablanca
2026

Casablanca, Maroc

CODING WEEK

Objets et conception



Héritage, composition, contrats à A2, Coding Week 1A

Hermann Agossou

Doctorant, UTER MID@S, ECC / UH2C

¹École Centrale Casablanca

Le critère

01

Rappel : le coût du changement

Deux questions sur le module précédent

Le critère de A1

Comment décide-t-on qu'un découpage est meilleur qu'un autre ? En quelle unité mesure-t-on ?

Ce qu'un module cache

Que cache un module, et pourquoi est-ce la bonne question plutôt que *à* que contient-il ?

RÉPONSES RAPIDES, PUIS ON ENCHAÎNE

Aujourd'hui, on applique ce critère aux objets et on découvre qu'*à* héritage ou composition *à* se tranche, au lieu de se discuter.

Une hiérarchie devenue douloureuse

Elle semblait juste au départ

01 'Compte'

Promet l'opération 'retirer(montant)'.

02 'CompteEpargne' hérite de 'Compte'

Redéfinit le retrait avec un plafond mensuel.

03 'CompteBloque' hérite de 'CompteEpargne'

Redéfinit le retrait pour toujours lever 'OperationInterdite'.

LA QUESTION QU'ON NE SAIT PAS ENCORE TRANCHER

ñ Utilisez la composition ÿ est une réponse que tout le monde connaît. Sur quelle base la défendriez-vous devant un collègue qui n'est pas d'accord ?

Couplage, cohésion, contrat, délégation

Le vocabulaire avant le jugement

Couplage

Ce qui oblige deux morceaux de code à changer ensemble.

Cohésion

Ce qui justifie que des éléments soient réunis au même endroit.

Contrat

Ce que l'appelant doit garantir, et ce que l'appelé promet en retour.

Délégation

Un objet en tient un autre et lui confie une part du travail.

GoF : préférer la composition à l'héritage

Et programmer vers une interface, pas une implémentation

LES DEUX PRINCIPES DE L'INTRODUCTION

- ▶ Programmer vers une interface, non vers une implémentation.
- ▶ Préférer la composition d'objets à l'héritage de classe.

HÉRITAGE : RÉUTILISATION EN BOÎTE BLANCHE

La sous-classe voit l'intérieur du parent.

- ▶ Dépend de l'implémentation, pas seulement de l'interface
- ▶ Rompt donc l'encapsulation

COMPOSITION : RÉUTILISATION EN BOÎTE NOIRE

L'objet composé n'est vu qu'à travers son interface.

- ▶ Ne dépend que de ce qui est promis
- ▶ Se remplace sans rien réécrire

Voir: [1] Gamma et al. (1994)

L'héritage est une réutilisation en boîte blanche

La sous-classe dépend de l'implémentation du parent

AVANT DANS 'COMPTE'

Le retrait modifie directement '`__solde`'.

- Les sous-classes lisent ce détail interne

APRÈS DANS 'COMPTE'

Le retrait ajoute '-montant' à '`__mouvements`'.

- Chaque sous-classe qui lisait '`__solde`' doit changer

C'EST L'AMPLIFICATION DU CHANGEMENT DE A1

Le parent n'a rien promis sur '`__solde`' : les sous-classes s'en servaient quand même. L'héritage leur en avait donné le droit.

Liskov : quand l'héritage est légitime

Le critère de substitution

LE CRITÈRE DE SUBSTITUTION

Un type est un sous-type d'un autre seulement si le remplacer partout où le sur-type est attendu laisse inchangé le comportement de tout programme écrit contre le sur-type.

Voir: [2] Liskov (1987)

APPLIQUÉ AU COMPTE BLOQUÉ

Tout code qui appelle 'retirer' sur un 'Compte' cesse de fonctionner si on lui passe un 'CompteBloque'. L'héritage n'était donc pas légitime. Cela se démontre, cela ne se discute pas.

Contrats : précondition, postcondition, invariant

Rendre le critère vérifiable

01 Précondition

Ce que l'appelant doit garantir avant d'appeler. Elle l'engage, lui.

02 Postcondition

Ce que l'appelé promet à la sortie. Elle engage le fournisseur.

03 Invariant

Ce qui reste vrai avant et après chaque opération publique.

LA RÈGLE DE REDÉFINITION

Une méthode redéfinie peut affaiblir la précondition et renforcer la postcondition, jamais l'inverse. Le compte bloqué renforce la précondition : c'est précisément le sens interdit.

Voir: [3] Meyer (1997)

Décider : composition ou héritage

Exemple entièrement traité

Méthode	Substitution tenue ?	Conclusion
solde()	oui	héritage acceptable
deposer()	oui	héritage acceptable
retirer()	non : lève une exception	héritage illégitime

VÉRIFICATION

Pour chaque méthode héritée, substituez une instance de la sous-classe dans tous les appelants existants et exécutez-les. Un appelant qui doit tester le type concret, ou qui échoue, prouve que l'héritage n'était pas légitime.

Terminer la décision commencée

La moitié de l'analyse est donnée

Deuxième hiérarchie : 'Rectangle' et 'Carre'. Un carré est un rectangle, en français.

Méthode	Substitution tenue ?	Conclusion
aire()	oui	acceptable
fixer_largeur()	?	?
fixer_hauteur()	?	?

À COMPLÉTER, PUIS TRANCHER

Nommez l'appelant qui casse. Si vous n'en trouvez aucun, l'héritage tient, et il faudra le dire aussi.

La transformation

02

Fowler : refactorer sans changer le comportement

Petits pas, tests au vert après chacun

DÉFINITION

Refactorer, c'est modifier la structure interne d'un logiciel pour le rendre moins coûteux à modifier, sans changer son comportement observable.

LA DISCIPLINE

Petits pas, tests après chacun, et retour au vert avant l'étape structurelle suivante. Si une étape échoue, corrigez-la dans cette portée bornée ou annulez-la ; n'empilez pas une nouvelle transformation sur un état rouge.

Voir: [4] Fowler (2018)

Remplacer l'héritage par la délégation

Replace Superclass with Delegate, étape par étape

AVANT : HÉRITER

'CompteBloque' est présenté comme un 'Compte'.

- ▶ Il promet 'retirer'
- ▶ Puis refuse toujours cette opération

APRÈS : DÉLÉGUER

'CompteBloque' contient un compte et expose seulement ce qu'il tient.

- ▶ Délègue 'solde' et 'deposer'
- ▶ Ne promet plus 'retirer'

CE QUE LA DÉLÉGATION ACHÈTE ICI

La classe ne promet plus 'retirer'. Il n'y a donc plus d'appelant à casser : le contrat dit la vérité sur ce que l'objet fait.

Nom au catalogue : Replace Superclass with Delegate (2e éd.) ; Replace Inheritance with Delegation dans la 1re.

Strategy : ce que le patron achète vraiment

Reconnaître un patron déjà présent

INTENTION

Définir une famille d'algorithmes, encapsuler chacun et les rendre interchangeables, afin que l'algorithme varie indépendamment des clients qui l'utilisent.

WORKFLOW 01

'facturer'

Calcule le prix de base, sans choisir de politique.

WORKFLOW 02

'calcul_remise'

Le paramètre est le point de variation.

WORKFLOW 03

Interchanger

Injecter 'remise_etudiant' ou 'remise_black_friday'.

LA BONNE QUESTION

Que coûterait sa suppression ? Ici : chaque nouvelle remise rouvrirait 'facturer'. Ce que le patron achète, c'est de ne plus toucher au code qui marche.

Refactorer vers Strategy, tests au vert

Chaque étape est un commit

01 1. Extraire le calcul dans une fonction

Tests au vert commit.

02 2. Passer la fonction en paramètre, valeur par défaut inchangée

Tests au vert commit.

03 3. Déplacer les appelants vers le nouveau paramètre

Tests au vert commit.

04 4. Supprimer la valeur par défaut devenue inutile

Tests au vert commit.

VÉRIFICATION

Exécutez la suite après chaque étape et ne committez que sur du vert. Annuler une étape rouge et la reprendre plus petite est ce qui rend la transformation conservatrice du comportement, plutôt qu'une réécriture.

À vous : refactorer par petits pas

Sans aide, historique à l'appui

CONSIGNE

Refactorisez la classe distribuée vers la délégation. Un commit par étape, la suite au vert à chaque commit.

- ▶ Le rendu est l'historique, pas le fichier final
- ▶ Une étape rouge s'annule ; elle ne se corrige pas en avançant
- ▶ Si une étape est trop grosse pour rester verte, coupez-la en deux

On rejouera un 'git log' au tableau. A3 expliquera pourquoi cet historique est une preuve.

Ce qui reste quand les patrons passent de mode

Un critère et une discipline

- ▶ La substitution est un test, pas un slogan : elle se vérifie sur les appelants.
- ▶ Un contrat le rend mécanique : affaiblir la précondition, renforcer la postcondition.
- ▶ Refactorer conserve le comportement : petits pas, vert à chaque pas, annuler plutôt qu'avancer.
- ▶ Un patron s'évalue à ce qu'il achète ici, dans ce code, pas dans le catalogue.

CE QUI N'ÉTAIT QUE DE L'OUTILLAGE

Le catalogue GoF, les classes Python, les ABC. Les patrons changeront de nom et de mode ; le critère de substitution et la discipline des petits pas, non.

Références

03