



1

Inquiétudes pour le projet de session ...



Exigences pour le Produit final

- Le code doit compiler et tester sans échec avec la commande `mvn clean package` lancée à la racine de votre dépôt;
 - La version livrée doit être étiquetée (`git tag`) avec l'étiquette `version_finale`;
 - L'exécution dans l'arène de championnat doit impérativement respecter les exigences techniques sur les îles déjà rencontrées
- On doit trouver à la racine de votre dépôt un fichier `rapport_X_final.pdf` (où `X` est votre lettre d'équipe), de 10 pages maximum (police 11 pt, interligne simple), contenant :
 - Le nom des participants de l'équipe
 - Les modèles de conceptions pertinents pour illustrer votre projet
 - La justification de leur pertinence, *typiquement les patrons utilisés*
 - Une analyse critique (force / faiblesse) du projet
 - Les évolutions prioritaires qui doivent être mise en place par la suite

Toute remise ne respectant pas ces consignes ne sera pas évaluée et obtiendra la note de zéro (0).

2

		Objectif		
		Création	Structure	Comportement
Portée	classe	Factory	Adapter	Interpreter Template Method
		Abstract Factory Builder Prototype Singleton	Chain of Responsibility Command Bridge Composite Decorator Facade Flyweight Proxy	Iterator Mediator Memento Observer State Strategy Visitor

3

4

State

1

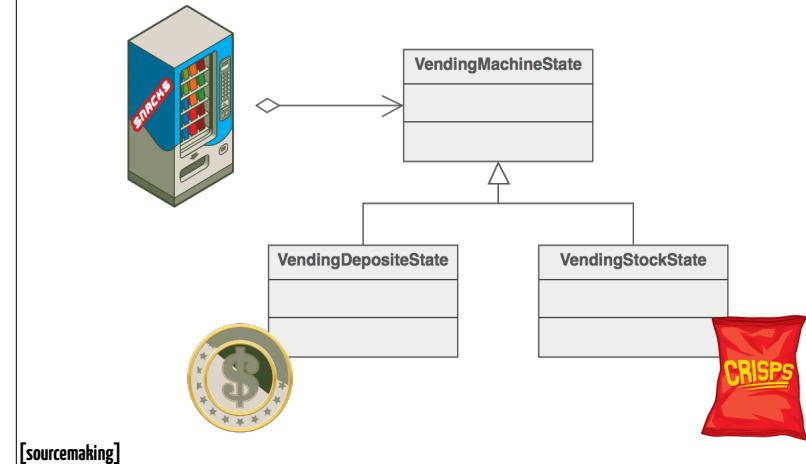
5

Problème

Comment modifier le comportement d'un objet quand son état interne change et obtenir des traitements différents ?

7

Exemple

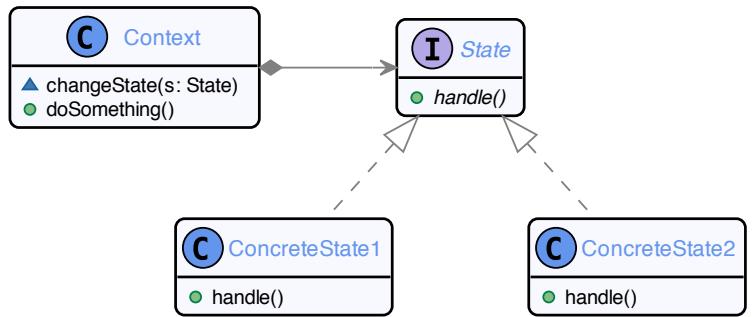


6

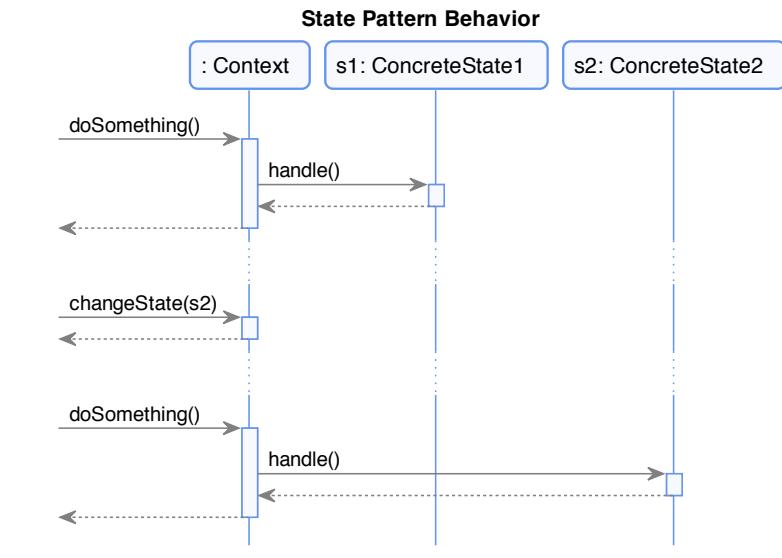
Intention

- Eviter les instructions conditionnelles en masse
- Simplifier l'ajout suppression de nouveaux états
- Donner l'impression que l'objet a été modifié alors que c'est uniquement son état qui a varié

8



9



10

Conséquences

- Séparation des comportements relatifs à chaque état
- Transitions explicite par action sur l'objet
- Transparence pour l'appelant

11

Où le trouver ?

- Gestion des connexion réseaux
- Mise en oeuvre d'une machine à état
- Javax Lifecycle

12

Builder

9

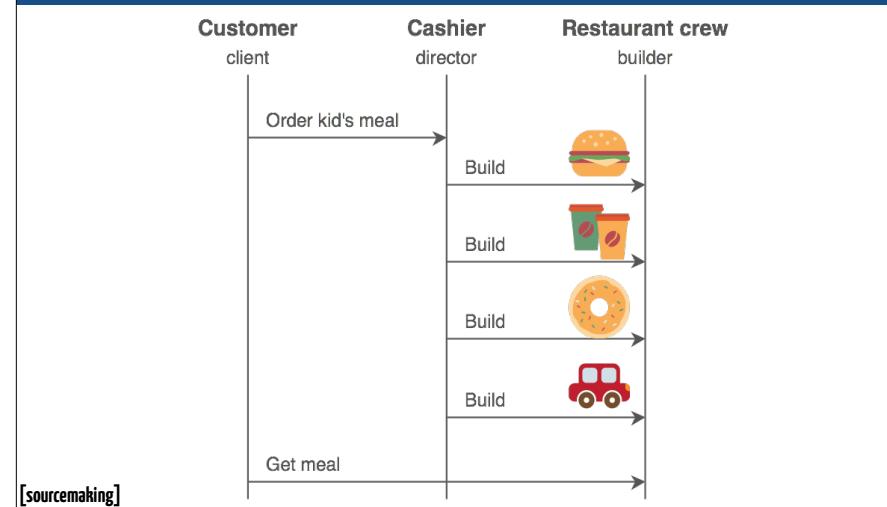
13

Problème

Comment créer par assemblage un objet complexe, indépendamment des parties qui le compose ?

15

Exemple

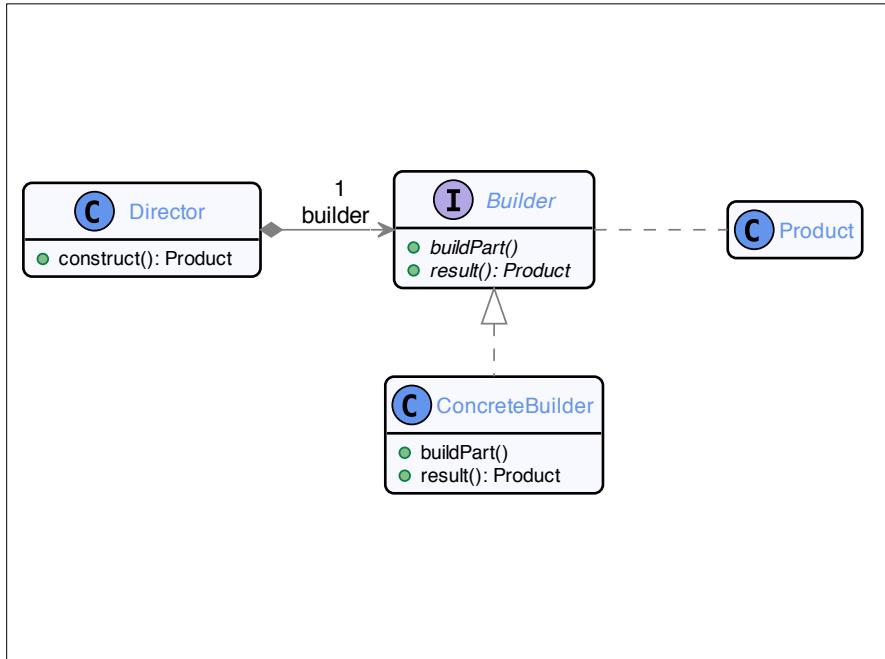


14

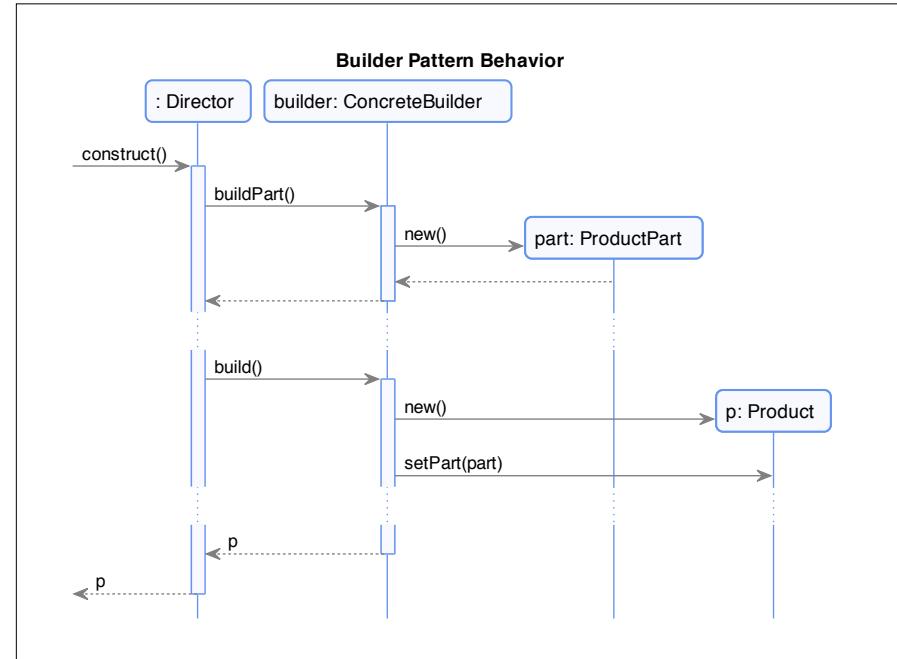
Intention

- Isoler la construction de l'objet de sa représentation
- Contrôler le procédé de construction
- Changer au besoin la représentation interne du produit

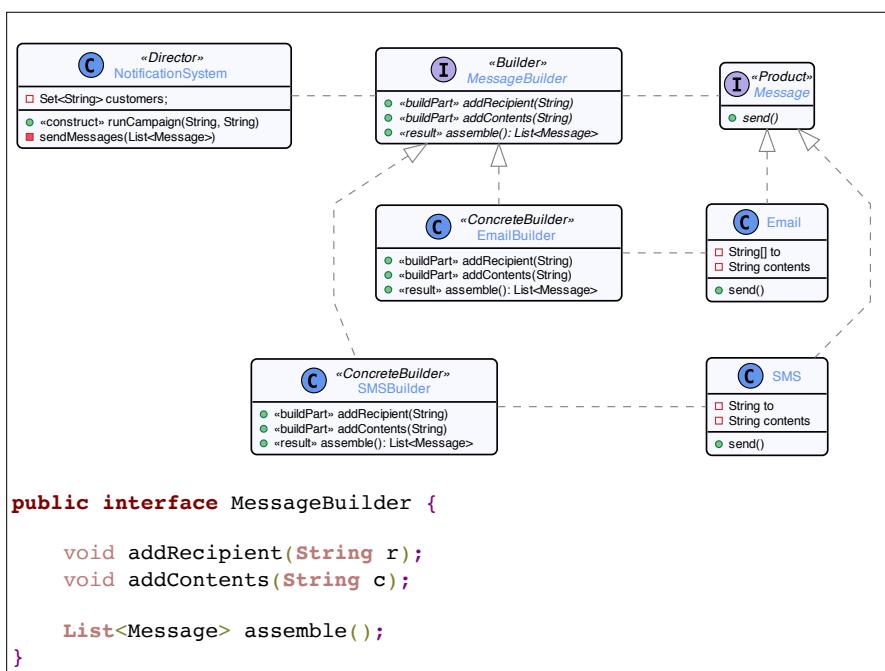
16



17



18



19

```

public interface MessageBuilder {
    void addRecipient(String r);
    void addContents(String c);
    List<Message> assemble();
}

public class SMSBuilder implements MessageBuilder {
    private Set<String> recipients = new HashSet<>();
    private String contents = null;

    @Override public void addRecipient(String r) { this.recipients.add(r); }

    @Override public void addContents(String c) { this.contents = c; }

    @Override
    public List<Message> assemble() {
        if (recipients.isEmpty() || contents == null)
            throw new IllegalStateException();
        return recipients.stream()
            .map((r) -> new SMS(r,contents))
            .collect(Collectors.toList());
    }
}

```

This code snippet provides the implementation for the SMSBuilder class, which implements the MessageBuilder interface. The SMSBuilder maintains a set of recipients and a single content string. It overrides the `addRecipient` and `addContents` methods to add them to the internal state. The `assemble` method returns a list of SMS messages, each created by mapping a recipient and content pair using a lambda expression.

20

Conséquences

- Définition abstraite du processus de construction
 - Réutilisation du process, mais produits différents
- On peut utiliser du polymorphisme pour construire des produits différents
- Le client n'a pas besoin de connaître le processus de fabrication

21

Prototype

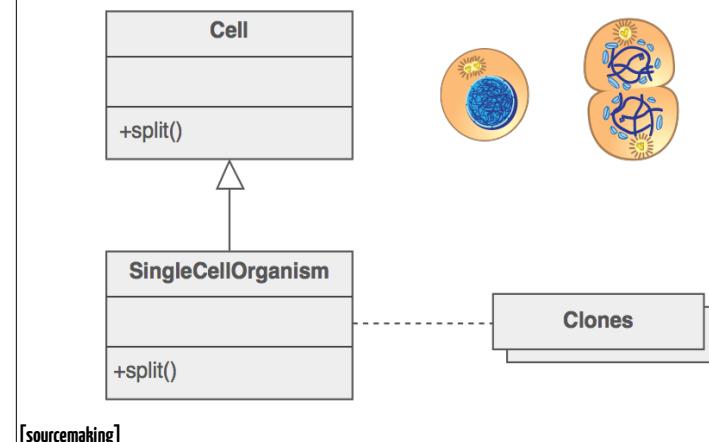
3

Où le trouver ?

- Dès qu'on parle de "Fluent API"
 - Mockito
 - EntityBuilder (spring, .Net, ...)
- Dans l'API Java (StringBuilder, StringBuffer, ...)
- Interfaces graphiques (composition de composants)

22

Exemple

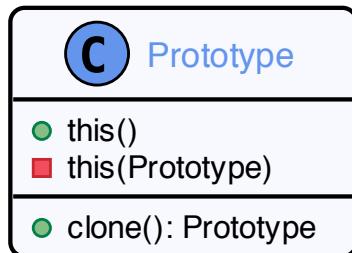


23

Problème

Comment utiliser un objet comme exemple d'instantiation pour les autres, par exemple quand sa création est couteuse mais sa copie aisée ?

25

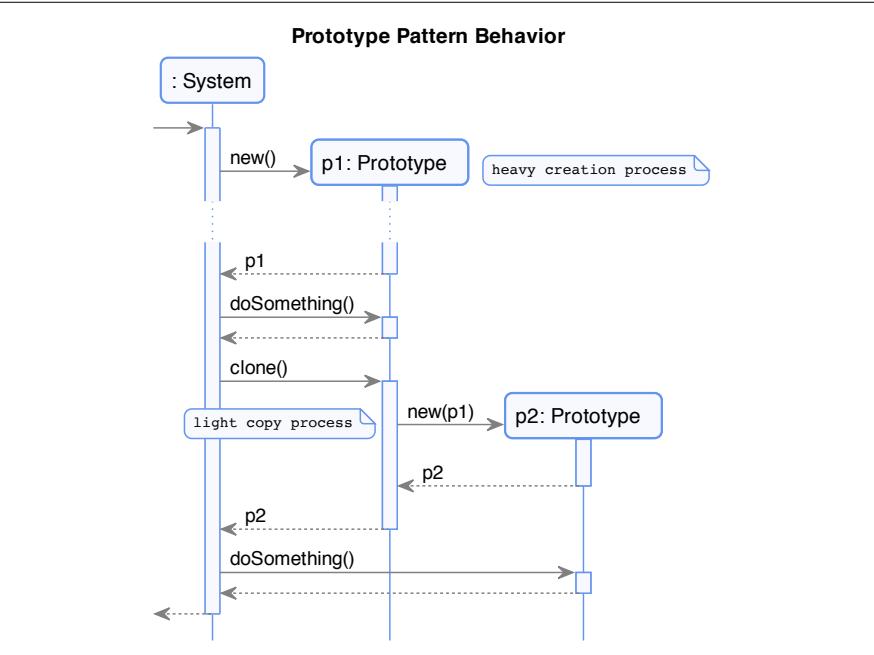


27

Intention

- Spécifier une classe d'objet constructible par clonage
- Permettre de se passer du new
- Co-opter une instance comme représentative des suivantes

26



28

```
private static Set<Student> usingConstructor() {
    Set<Student> result = new HashSet<>();

    Student bob = new Student("Bob");
    bob.registerTo("INF5151");
    bob.registerTo("INF5153");
    bob.registerTo("OPT6000");
    result.add(bob);

    Student alice = new Student("Alice");
    alice.registerTo("INF5151");
    alice.registerTo("INF5153");
    alice.registerTo("OPT3000");
    result.add(alice);

    Student eve = new Student("Eve");
    eve.registerTo("INF5151");
    eve.registerTo("INF5153");
    eve.registerTo("OPT8000");
    result.add(eve);

    return result;
}

private static Set<Student> usingClones() {
    Set<Student> result = new HashSet<>();

    Student bob = new Student("Bob");
    bob.registerTo("INF5151");
    bob.registerTo("INF5153");

    Student alice = bob.duplicate();
    alice.setName("Alice");

    Student eve = bob.duplicate();
    eve.setName("Eve");

    bob.registerTo("OPT6000");
    result.add(bob);

    alice.registerTo("OPT3000");
    result.add(alice);

    eve.registerTo("OPT8000");
    result.add(eve);

    return result;
}
```

29

Conséquences

- On peut factoriser des opérations coûteuses dans le prototype
 - Le clonage est une opération difficile :
 - Clonage superficiel ?
 - Clonage en profondeur ?
 - Modèle de programmation peu habituel

} Collections / refs

{ Collections / refs

```
lucifer:prototype mosser$ mvn -q exec:java

# Classical instantiation
Registering [Bob] to [INF515]
=> disconnect ... validate ... register ... disconnect ... <-
Registering [Bob] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Bob] to [DPT6000]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Alice] to [INF515]
=> disconnect ... validate ... register ... disconnect ... <-
Registering [Alice] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Alice] to [DPT3000]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Eve] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Eve] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Eve] to [DPT8000]
=> dbconnect ... validate ... register ... disconnect ... <-
->>> Time consumed: 5512ms

## Resulting student set
Student{name='Alice', courses=[DPT2000, INF515, INF515]}
Student{name='Bob', courses=[DPT6000, INF515, INF515]}
Student{name='Eve', courses=[DPT8000, INF515, INF515]}

# Using clones
Registering [Bob] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Bob] to [INF515]
=> dbconnect ... validate ... register ... disconnect ... <-
Cloning Student{name='Bob', courses=[INF5151, INF5153]}
Cloning Student{name='Bob', courses=[INF5151, INF5153]}
Registering [Bob] to [DPT6000]
=> disconnect ... validate ... register ... disconnect ... <-
Registering [Alice] to [DPT3000]
=> dbconnect ... validate ... register ... disconnect ... <-
Registering [Eve] to [DPT8000]
=> dbconnect ... validate ... register ... disconnect ... <-
->>> Time consumed: 3049ms

## Resulting student set
Student{name='Alice', courses=[DPT2000, INF515, INF515]}
Student{name='Bob', courses=[DPT6000, INF515, INF515]}
Student{name='Eve', courses=[DPT8000, INF515, INF515]}

# Are resulting sets equivalents ?
Equivalence
lucifer:prototype mosser$
```

30

Où le trouver ?

- Interface Cloneable de Java
 - Modèle objet de Javascript
 - Et plus généralement des langages objets à prototype
 - La POO n'inclue pas que les langages à Classe !

31

Composite

4

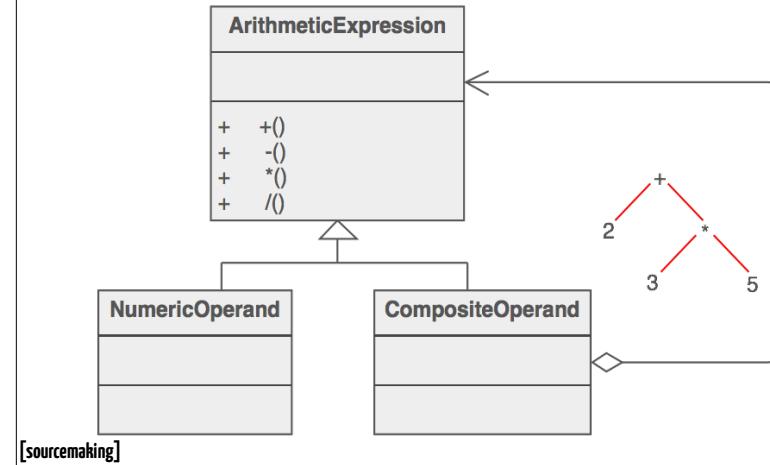
33

Problème

Comment représenter de manière uniforme une hiérarchie d'objets ?

35

Exemple

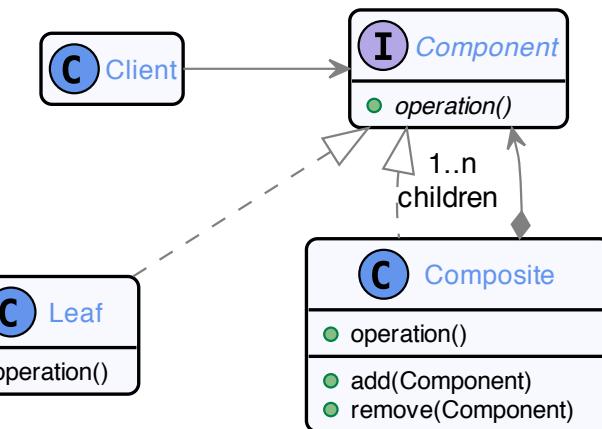


34

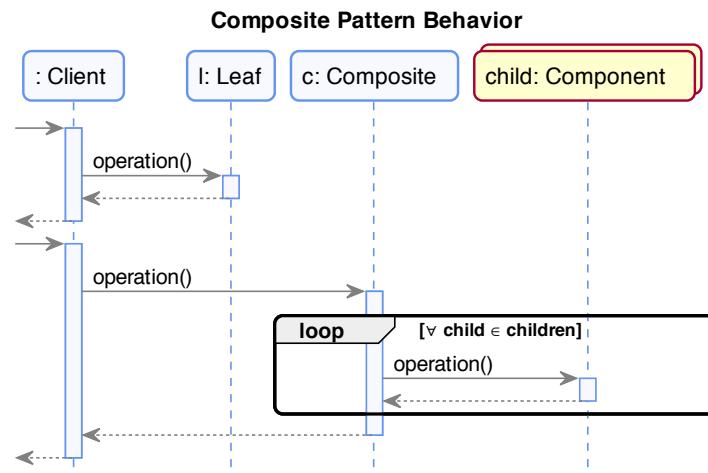
Intention

- Formaliser la structuration arborescente des objets
- Traiter feuilles et noeuds de manière uniforme
 - Propagation récursive des comportements

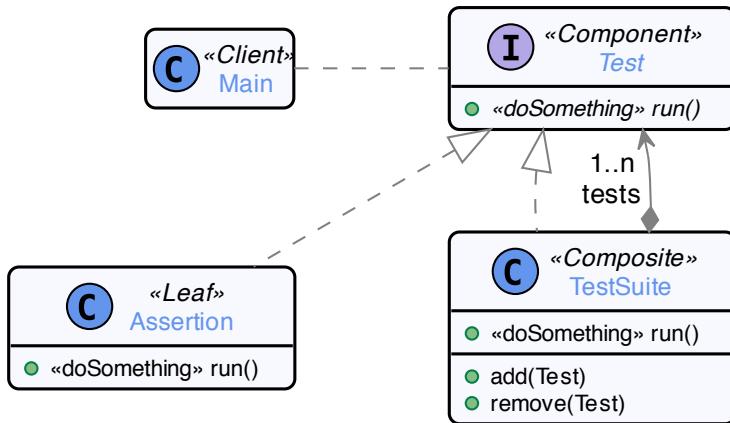
36



37



38



39

```

public static void main(String[] args) {
    TestSuite unit = new TestSuite("Unit tests");
    unit.add(new Assertion("1st unit test"));
    unit.add(new Assertion("2nd unit test"));
    unit.add(new Assertion("3rd unit test"));

    TestSuite accept = new TestSuite("Acceptance tests");
    accept.add(new Assertion("1st scenario"));
    accept.add(new Assertion("2nd scenario"));

    TestSuite ext1 = new TestSuite("Integration with EXT-1");
    ext1.add(new Assertion("Test with external partner #1"));
    TestSuite ext2 = new TestSuite("Integration with EXT-2");
    ext2.add(new Assertion("Test with external partner #2"));
    TestSuite integration = new TestSuite("Integration tests");
    integration.add(ext2);
    integration.add(ext1);

    TestSuite complete = new TestSuite("Complete test suite");
    complete.add(unit);
    complete.add(accept);
    complete.add(integration);

    System.out.println("\n# Running a single assertion");
    (new Assertion("single unit test")).run();

    System.out.println("\n# Running unit tests only");
    unit.run();

    System.out.println("\n# Running the all product suite");
    complete.run();
}

```

40

Conséquences

- Structure hiérarchique simple et uniforme
- Facile à étendre pour de nouveaux objets
- Comportement propagable de manière automatique
- Mais ...
 - potentiellement trop rigide
 - Les composants fils sont ils ordonnés ou non ?

41

Visitor



(le visiteur c'est un peu le boss de fin des patrons)

43

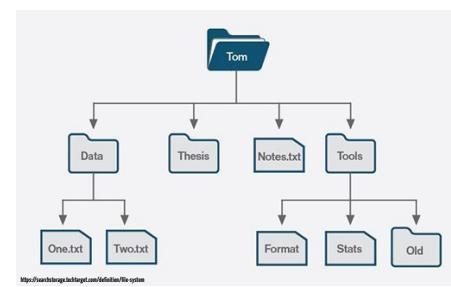
Où le trouver ?

- Dès qu'on fait face à une structure arborescente
 - Système de fichiers, GUI, compilation, documents
 - **Il se compose particulièrement bien avec la Commande :**
 - Une macro-commande est un composite de commandes
- **Et on va le faire en exercice à la fin du cours !**

42

Exemple

Rechercher
un fichier ?



Supprimer
un répertoire ?

Copier un répertoire ?

44

Problème

Comment rendre externe à une hiérarchie d'objets la mise en oeuvre d'un comportement, pour pouvoir changer celui-ci sans avoir à changer les objets ?

45

“Double Dispatch”

- 1.La hiérarchie “Accepte” l’opération
- 2.Elle délègue immédiatement à l’opération

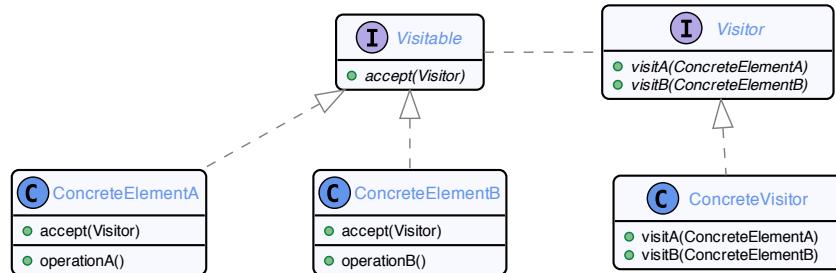
Un élément accepte, et le visiteur visite.

47

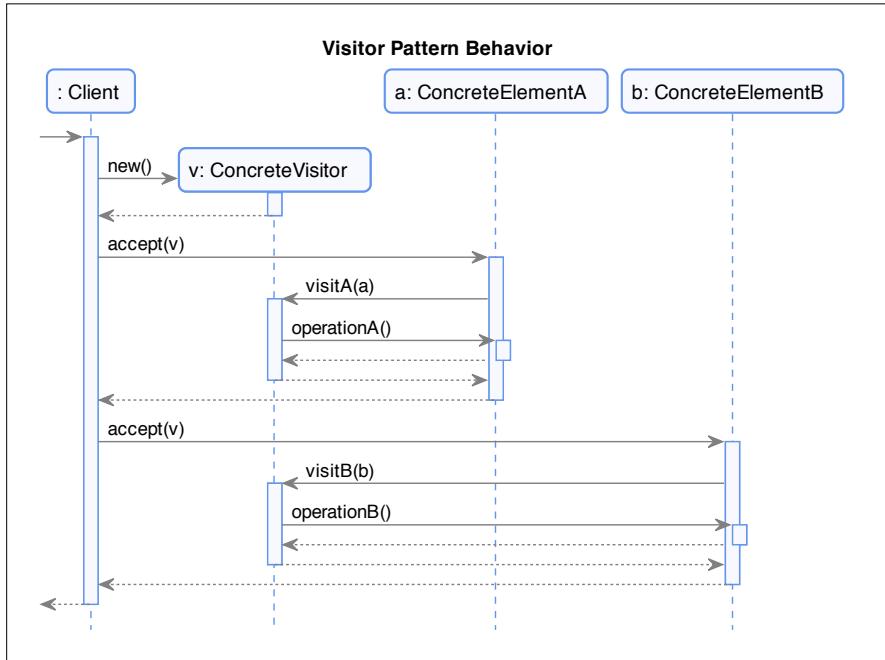
Intention

- Définir une abstraction pour les opérations à effectuer
- Définir UNE FOIS POUR TOUTES le parcours de la hiérarchie
- “Lancer” une opération sur une hiérarchie,
- Sans avoir à modifier la hiérarchie pour créer de nouvelles opérations

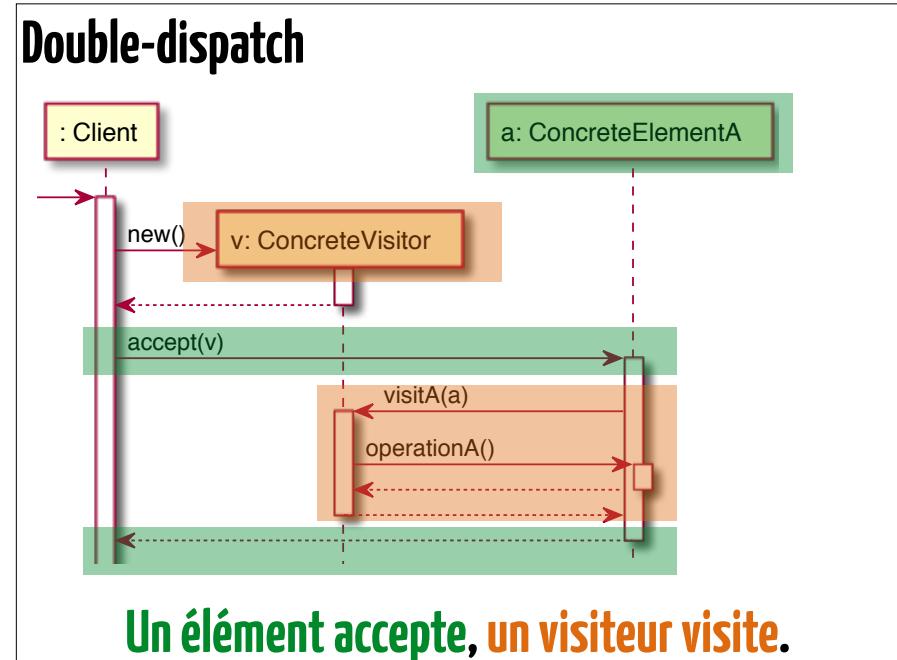
46



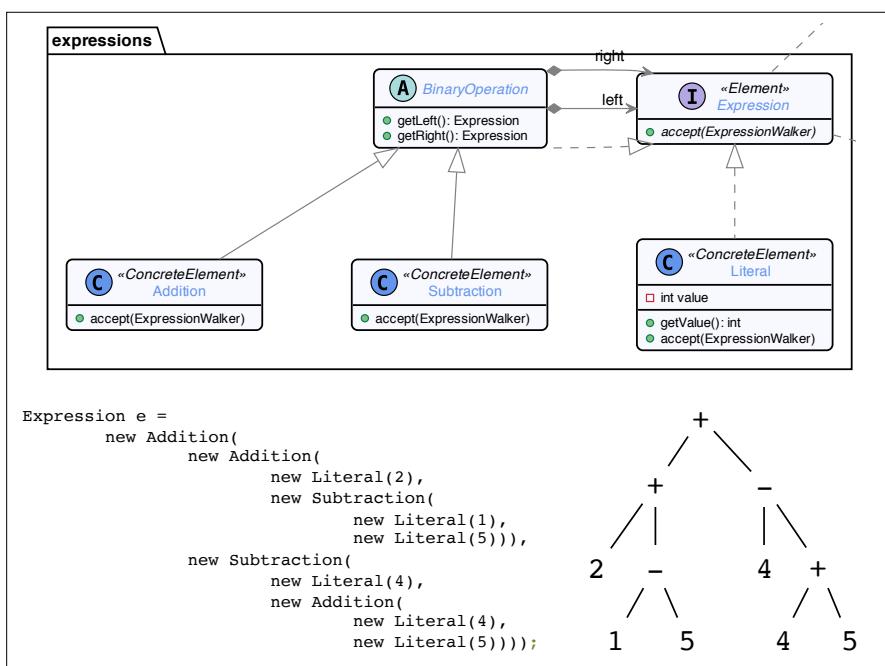
48



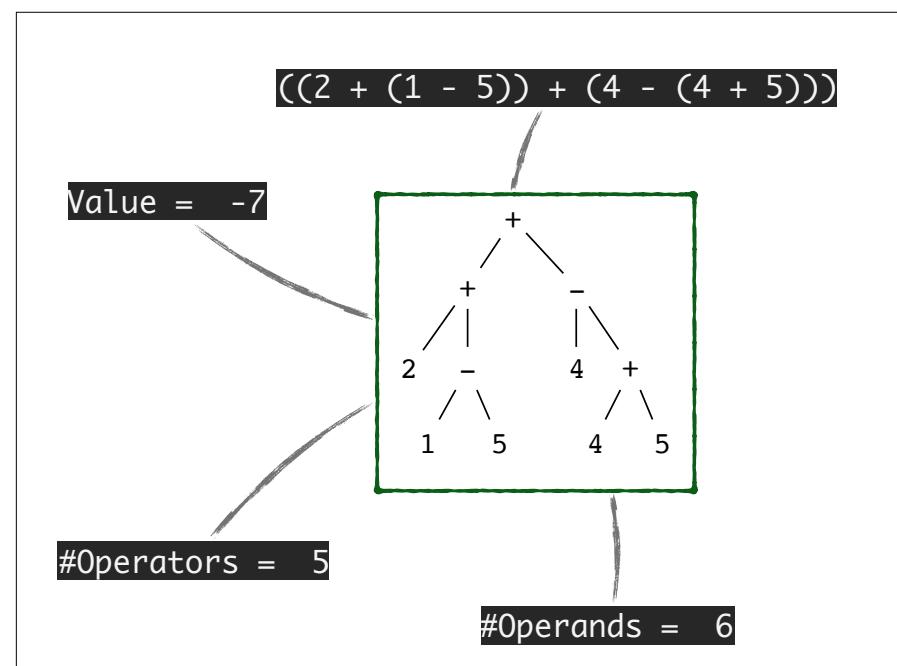
49



50



51



52

```

PrettyPrinter printer = new PrettyPrinter();
e.accept(printer);
System.out.println("Expression: " + printer.getResult());

OperandCounter opc = new OperandCounter();
e.accept(opc);
System.out.println(" #Operands = " + opc.getResult());

OperatorCounter ops = new OperatorCounter();
e.accept(ops);
System.out.println(" #Operators = " + ops.getResult());

Evaluator eval = new Evaluator();
e.accept(eval);
System.out.println("Value = " + eval.getResult());

```

53

```

public class PrettyPrinter implements ExpressionWalker<String> {
    private StringBuffer buffer = new StringBuffer();

    @Override
    public void visitLiteral(Literal literal) {
        buffer.append(literal.getValue());
    }

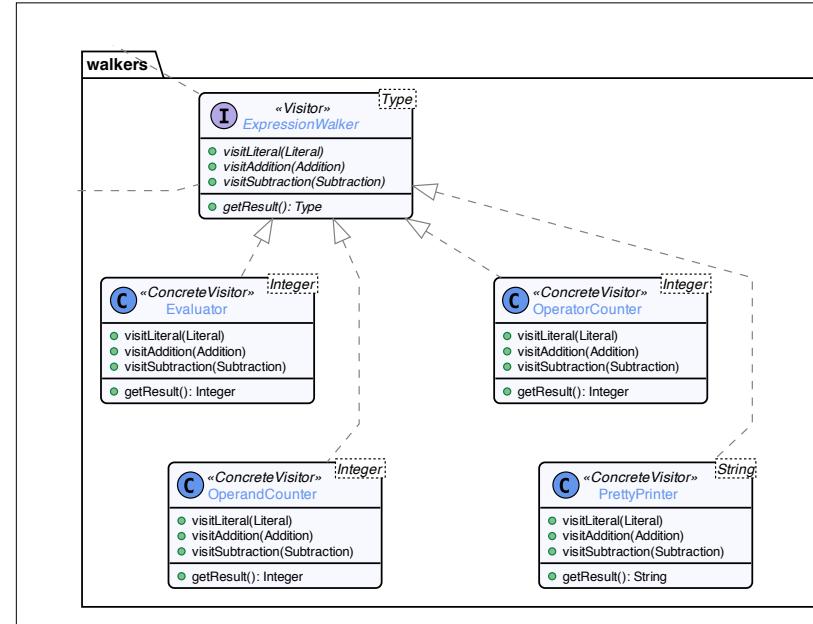
    @Override
    public void visitAddition(Addition addition) {
        buffer.append("(");
        addition.getLeft().accept(this);
        buffer.append(" + ");
        addition.getRight().accept(this);
        buffer.append(")");
    }

    @Override
    public void visitSubtraction(Subtraction subtraction) {
        buffer.append("(");
        subtraction.getLeft().accept(this);
        buffer.append(" - ");
        subtraction.getRight().accept(this);
        buffer.append(")");
    }

    @Override
    public String getResult() { return buffer.toString(); }
}

```

55



54

Conséquences

- On peut définir autant de visiteurs ≠ qu'on le souhaite
 - Et on peut le faire avec une approche fonctionnelle
 - (Donc un peu moins 🐷 que cet exemple, sans effets de bord)
- L'introduction du visiteur est invasive dans la hiérarchie
- Le double-dispatch à un coût non négligeable à l'exécution
 - Mais on peut faire plus sioux

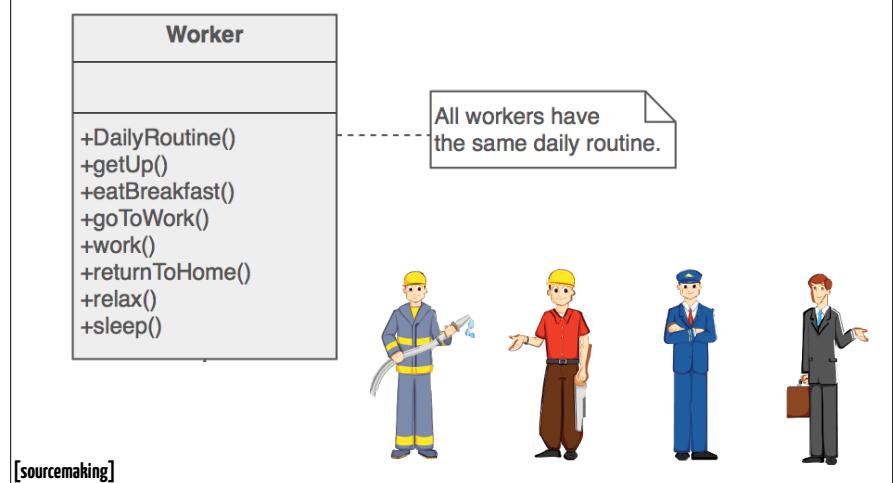
56

Où le trouver ?

- Compilation
 - On visite des arbres de syntaxes abstraits
- Transformation de modèles
 - Norme QVT, langage ATL, ...
- Langage intégrant le “pattern-matching”

57

Exemple



59

Template Method



58

Problème

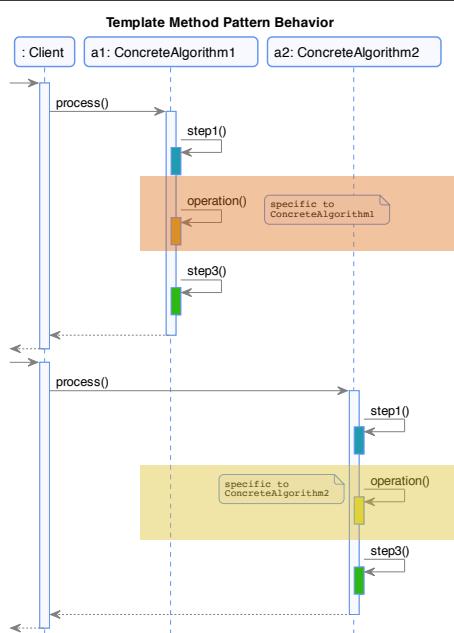
**Comment gérer un algorithme général,
et dont on pourra spécialiser
certaines étapes ?**

60

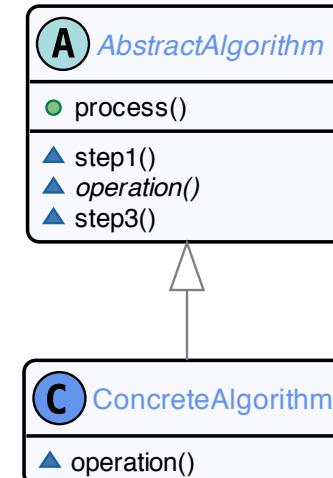
Intention

- Définir un squelette d'algorithme dans une opération
 - L'algorithme référence des étapes bien définies
- Permettre par héritage de redéfinir certaines étapes
 - Modifier l'algorithme sans modifier sa structure générale

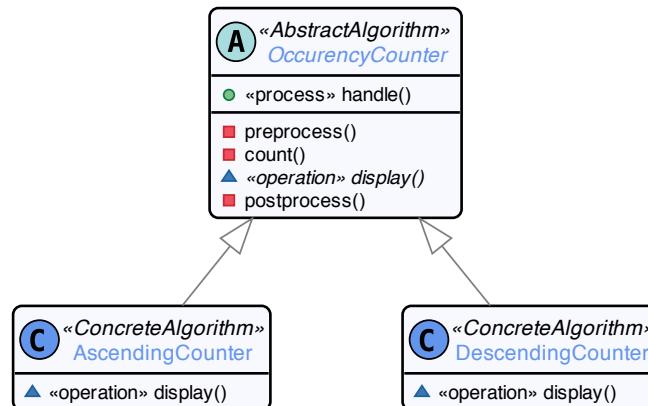
61



63



62



64

```

public abstract class OccurrencyCounter {

    private String data;

    protected Map<String, Integer> counter;

    public OccurrencyCounter(String input) {
        this.data = input;
        this.counter = new HashMap<>();
    }

    public void handle() {
        preprocess();
        count();
        display();
        postprocess();
    }

    private void preprocess() {
        System.out.println("Preprocessing data");
        this.data = this.data.replaceAll("[^a-zA-Z ]", " ").toLowerCase();
    }

    private void count() {
        System.out.println("Counting word occurrences");
        for(String w: data.split("\\s+")) {
            counter.put(w, counter.getOrDefault(w, 0)+1);
        }
    }

    protected abstract void display();

    private void postprocess() {
        System.out.println("Found ["+counter.size()+"] distinct words");
    }
}

```

65

Conséquences

- Réutilisation du code
- Structure de contrôle inversé
- La visibilité permet de jouer sur les capacités d'extension
- Mais ...
 - Repose sur des sous-classes (cf stratégie)

67

```

public class AscendingCounter extends OccurrencyCounter {
    public AscendingCounter(String input) { super(input); }

    @Override protected void display() {
        this.counter.entrySet().stream()
            .sorted(comparingByValue())
            .forEach(e -> System.out.println(" " + e));
    }
}

```

```

System.out.println("\n# Counting word occurrences using an ASC counter");
OccurrencyCounter asc = new AscendingCounter(text);
asc.handle();

System.out.println("\n# Counting word occurrences using a DSC counter");
OccurrencyCounter dsc = new DescendingCounter(text);
dsc.handle();

```

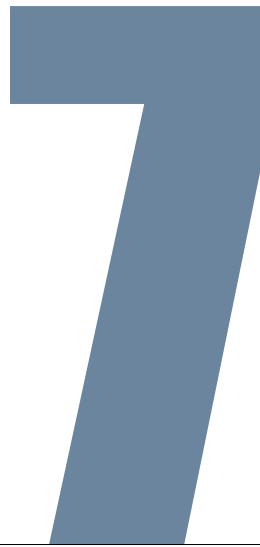
66

Où le trouver ?

- Canevas de tests unitaire
- Dans l'API Java :
 - Entrées / Sorties (InputStream, Writer, ...)
 - Structure de données abstraites (AbstractList, ...)
 - Gestion du protocole HTTP (HTTPServlet.doXXX())

68

Proxy



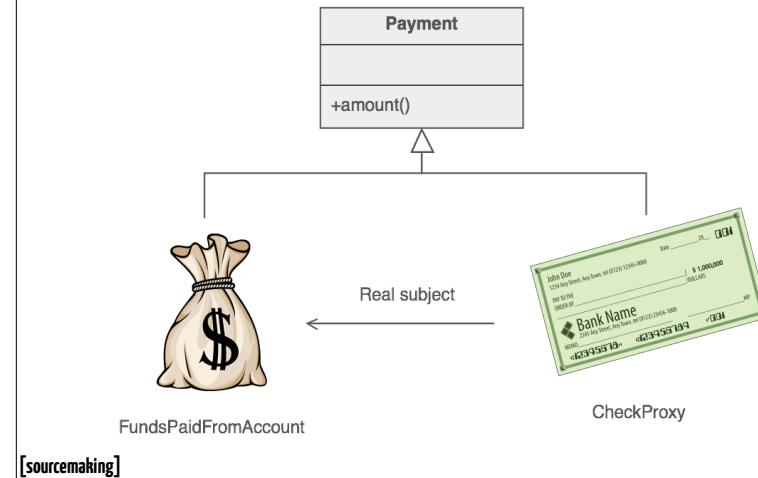
69

Problème

**Comment fournir un substitut
à un objet qui n'est pas
accessible facilement ?**

71

Exemple

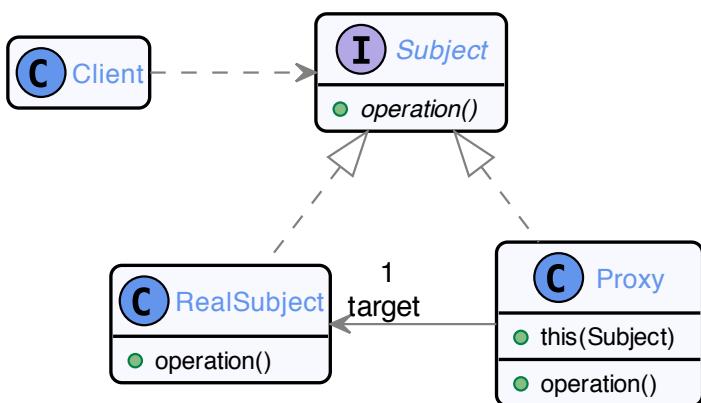


70

Intention

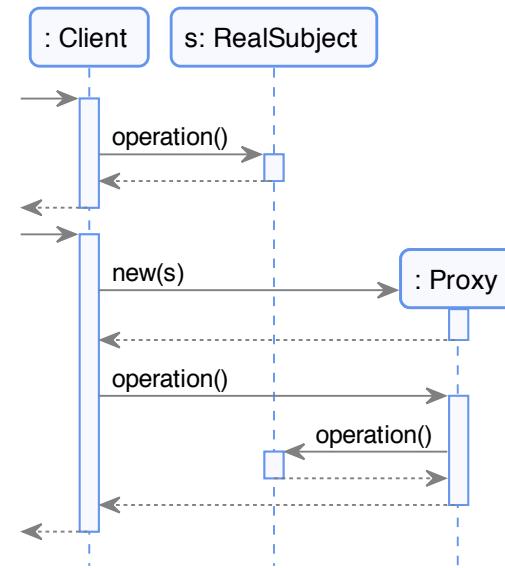
- Séparer l'interface de l'implémentation
- Réifier le concept de “procuration” pour substituer un objet inaccessible par un autre, qui lui est accessible.
- Travailler avec le substitut et le vrai objet de manière transparente

72



73

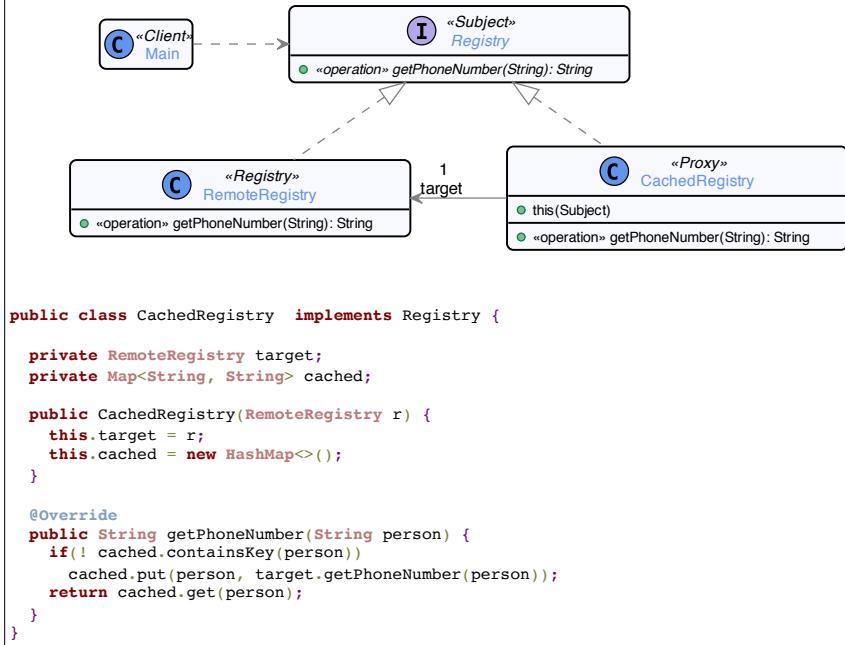
Proxy Pattern Behavior



74

Conséquences

- L'inaccessibilité du sujet est transparente
- Proxy et Sujet peuvent être aisément échangés
- Le proxy n'est pas une extension (sous-classe) du sujet, mais simplement une réplique exacte au même niveau dans la hiérarchie



75

76

Où le trouver ?

- Principe d'échantillonnage de données
- Image volumineuse à charger
- Gestion des droits d'accès
 - Le proxy gère les droits, l'objet le métier
- Objets distants
 - Java RMI, talons (stubs) d'accès à des WebServices
- Bouchons / Espions pour le test logiciel

77

Ceci est une question typique d'examen

79

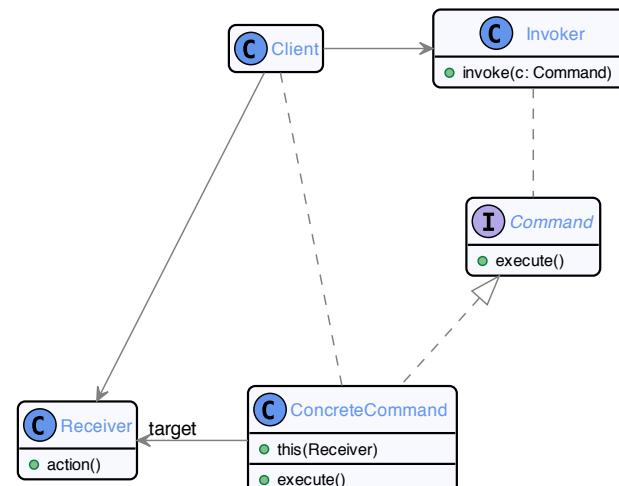


Composition de patrons

"Quand y'en a un ça va, c'est quand il y en a plusieurs que ça pose des problèmes".

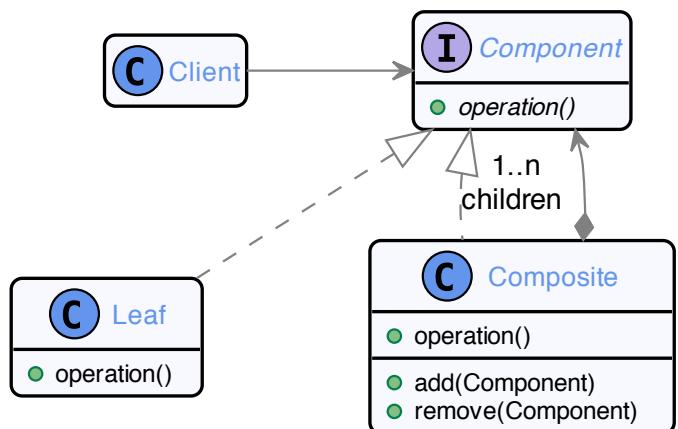
78

Principe de Commandes ...

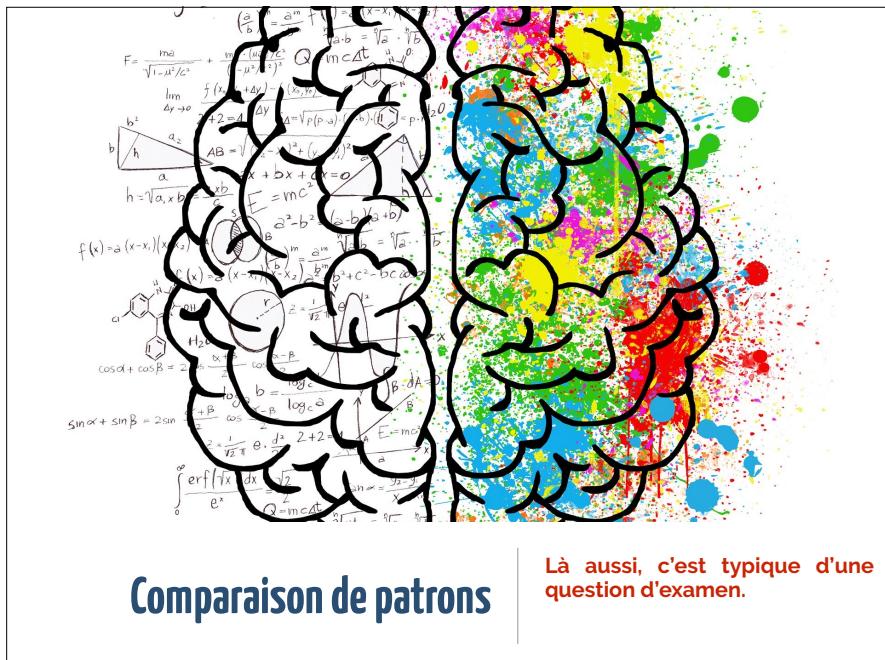


80

Principe de Composite ...

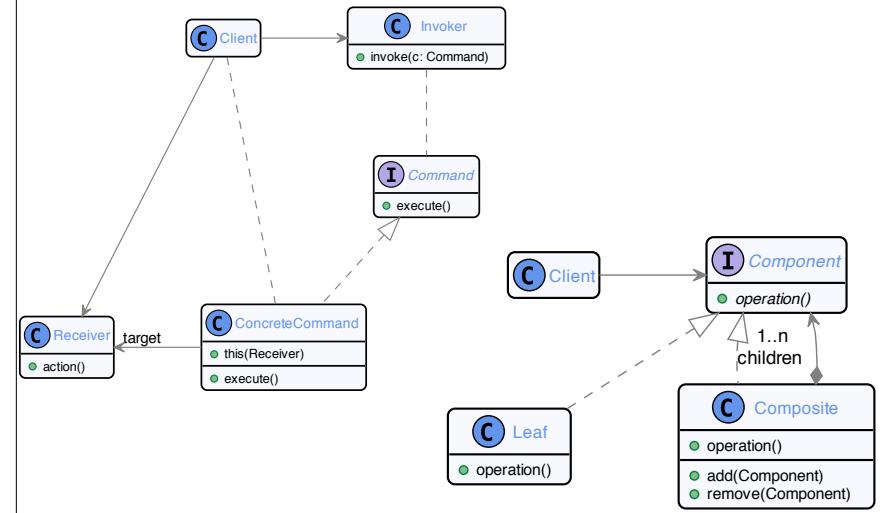


81



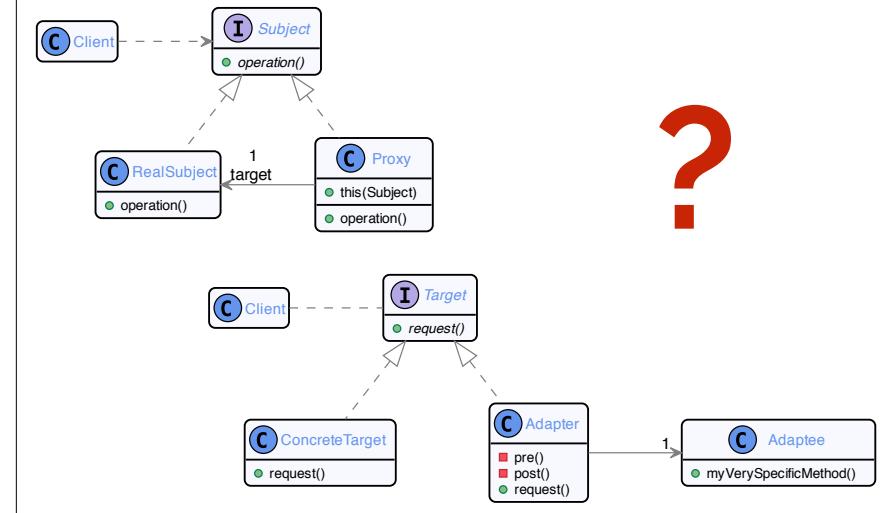
83

Macros = Commande \oplus Composite



82

Proxy versus Adapter



84



85