Cazillac Cyril
Monié Louis
Méry Éva
Limousin Rodolphe

FAÇADE

SOMMAIRE

PRÉSENTATION DES DESIGN PATTERNS

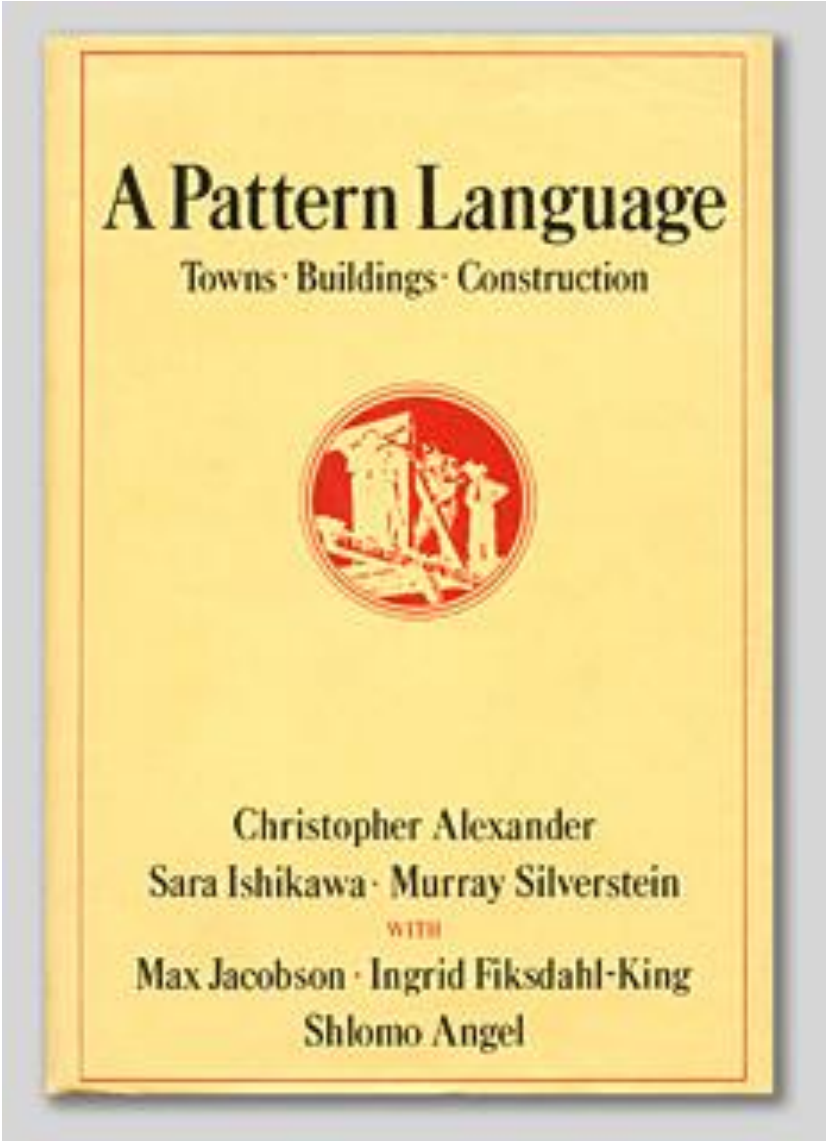
DÉFINITION

- Arrangement caractéristique de modules
- Solution standard
- Bonne pratique
- ~~Algorithme~~
- ~~Code~~

CARACTÉRISTIQUES

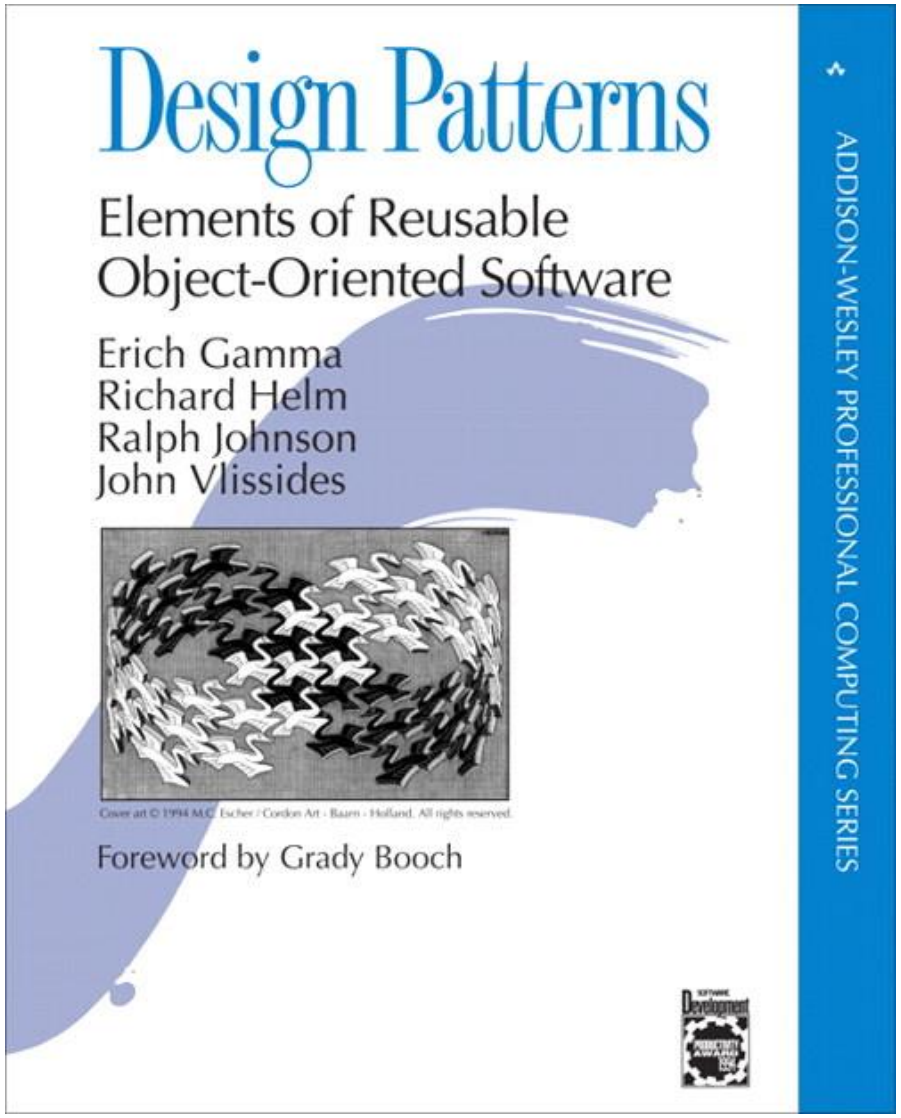
- Nom
- Description du problème
- Description de la solution
  - Éléments
  - Relations
- Conséquences

ORIGINES


A Pattern Language
Towns · Buildings · Construction
Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

ORIGINES

Ralph Johnson, Erich Gamma, Richard Helm et John Vlissides

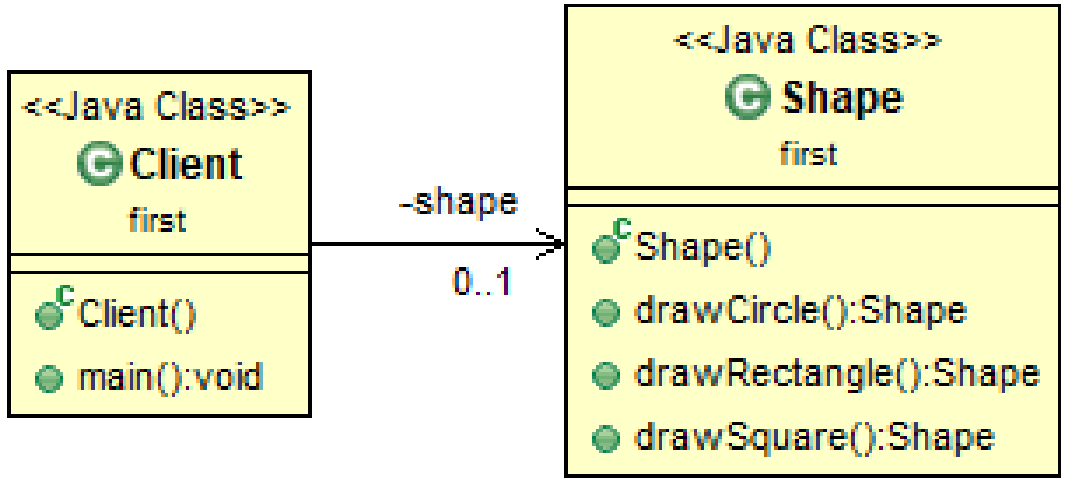


TYPES DE DESIGN PATTERNS

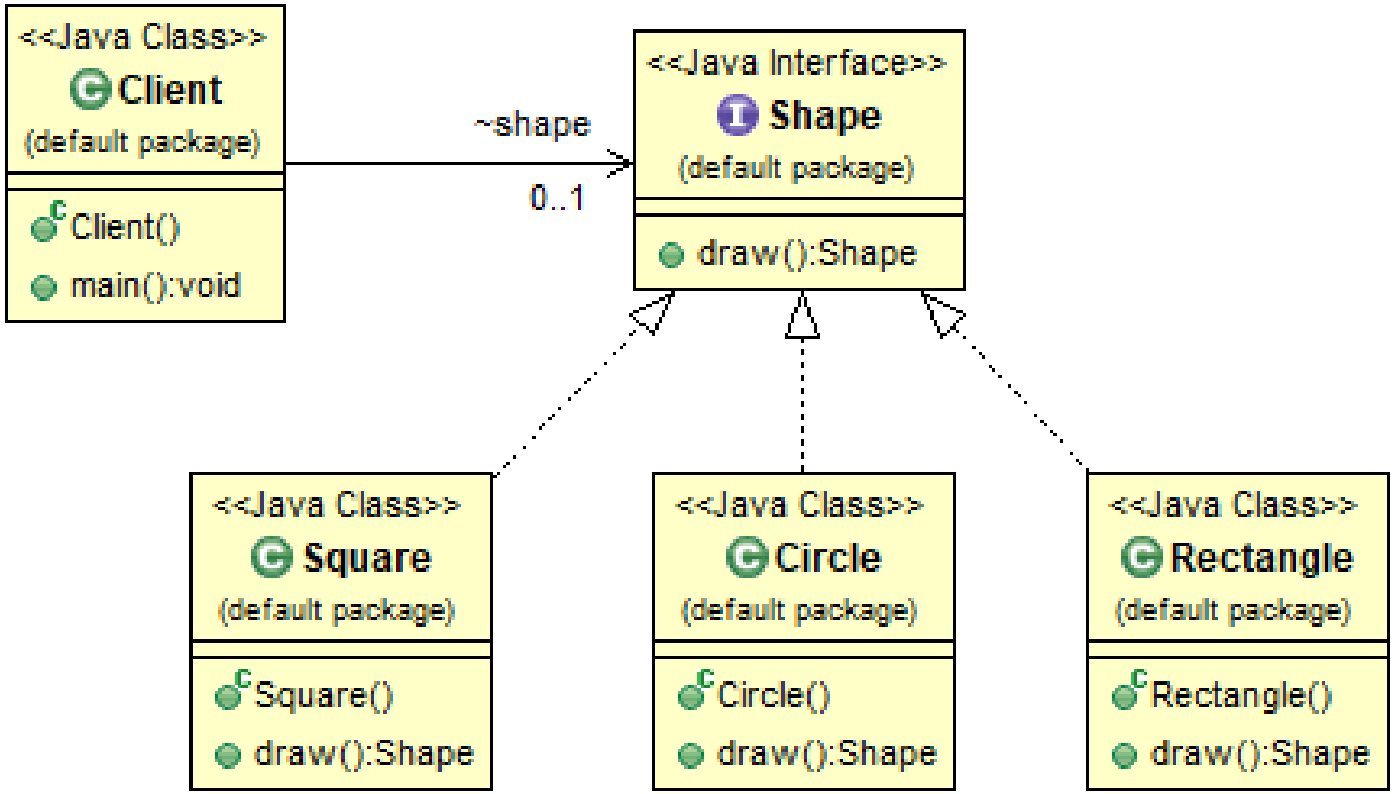
Créationnels	Structurels	Comportementaux
➢ Singleton	➢ Adaptateur	➢ Template Methode
➢ Fabrique	➢ Composite	➢ Médiateur
➢ Fabrique abstraite	➢ Proxy	➢ Chaine de Responsabilité
➢ Monteur	➢ Fly Weight	➢ Observateur
➢ Prototype	➢ Façade	➢ Stratégie
	➢ Pont	➢ Commande
	➢ Décorateur	➢ State
		➢ Visiteur
		➢ Itérateur
		➢ Interpréteur
		➢ Memento

INTRODUCTION AU PATTERN FAÇADE

Exemple simple



Amélioration
Exemple simple



LE DESIGN PATTERN FAÇADE

Le design pattern Façade

Définition du livre Gang of Four :

« Fournir une interface unifiée à un ensemble d'interfaces dans un sous-système. Façade définit une interface de niveau supérieur qui facilite l'utilisation du sous-système »

Type de conception structurelle :

- Traite généralement des relations entre les entités, facilite leur collaboration

- Etudie la façon de composer des classes et des objets pour réaliser des structures plus importantes.

Le design pattern Façade

Caractéristiques :

- Permet de simplifier l'interface lors de l'utilisation d'un sous-système
- Ne fait appel qu'aux fonctions/méthodes qui nous intéressent
- Peut même en créer de nouvelles à partir de celles existantes
- Se présente sous la forme d'une petite quantité de code
- Situé entre le kit d'outils et une application complète
- Est configurable, prévue pour être en production et être réutilisable

Le design pattern Façade

Intérêt et intention :

- Objectif principal : simplifier une interface.

- Fournir une interface unifiée à un ensemble d'interfaces dans un sous-système pour faciliter l'utilisation de celui-ci.

- Envelopper un sous-système compliqué avec une interface plus simple.

Le design pattern Façade

Répond à plusieurs besoins :

- Présenter volontairement un objet plus adapté
- Permet d’agréger les appels dans un seul objet
- Cacher la complexité de l’implémentation interne
- Présenter une interface simple à utiliser
- Simplifier l’appel à de nombreux objets internes
- Limiter les dépendances des classes clientes en exposant le moins possible d’objets internes
- Simplifier l'utilisation, la compréhension et rendre le code source de la bibliothèque plus lisible

Le design pattern Façade

Quand l'utiliser ?

- Trop de complexité dans un sous-système
- Mise en place un patron de conception implique de créer de nouvelles classes
- Quantité de travail pour configurer et écrire le code de base ne fait que croitre

Alors la façade :

- Fourni un raccourci aux fonctionnalités du sous-système qui sont adaptées au besoin du client
- Structurer un sous-système
- Définir des points d'entrée à chaque niveau d'un sous-système
- Permet de réduire le couplage

Le design pattern Façade

Comment ?

1) Vérifie-s'il est possible de fournir une interface plus simple

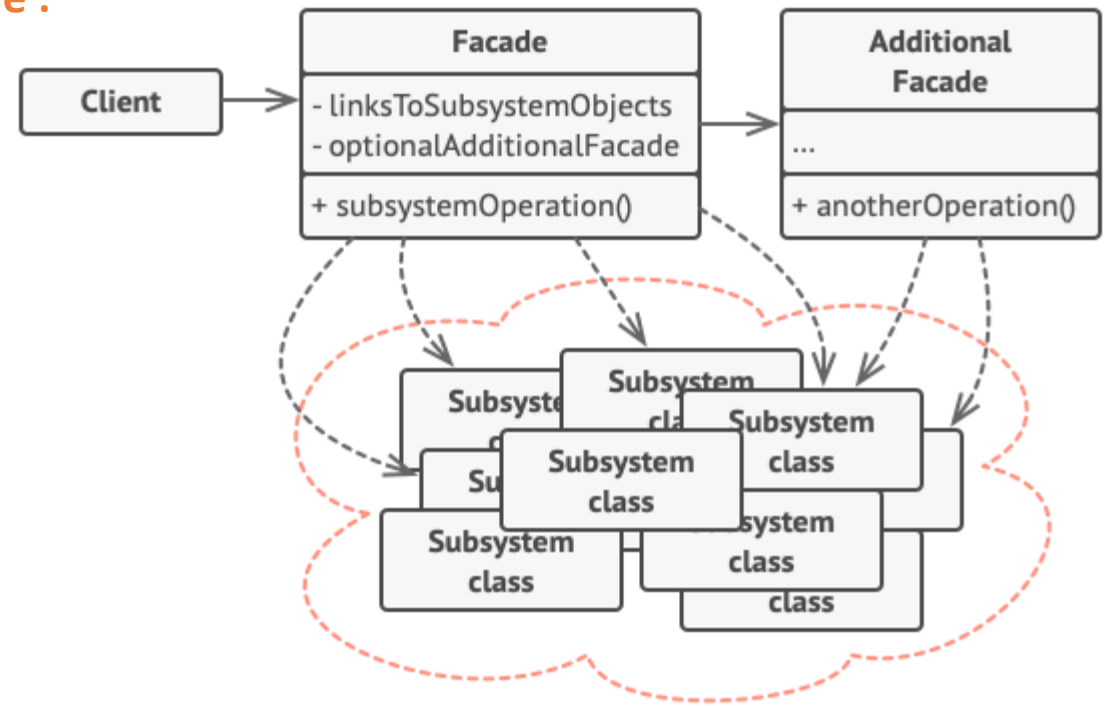
2) Déclarez et implémentez cette interface en tant que nouvelle façade

3) Toute communication avec le sous-système doit passer par elle

4) Transférer une partie de ses comportements vers une nouvelle façade plus spécialisée, si la façade devient trop grande

Le design pattern Façade

Structure :



1) Façade procure un accès pratique aux différentes parties des fonctionnalités

2) Façade Supplémentaire évite de polluer une autre façade avec des fonctionnalités qui pourraient la rendre trop complexe

3) Sous-système Complexe est composé de dizaines d'objets variés

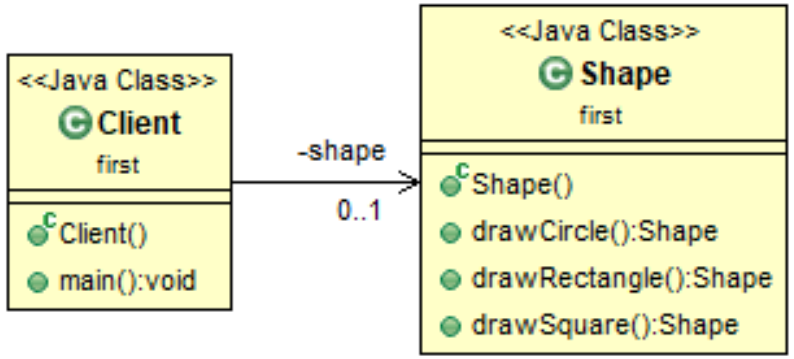
4) Client passe par la façade plutôt que d'appeler les objets directement

Le design pattern Façade

Problématique :

Reprenons l'implémentation classique de l'exemple d'introduction :

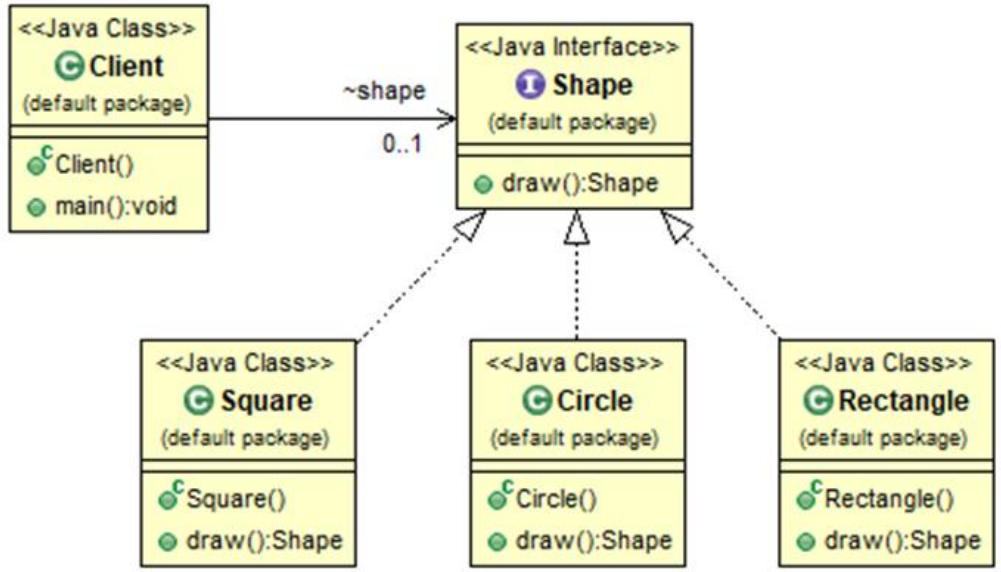
- But : créer un programme permettant de dessiner des formes géométriques (cercle, rectangle, carré...) souhaitées par un client



- On se rend compte que ce dernier ne respecte pas les principes SOLID

Le design pattern Façade

Problématique :



- On manipule ici un ensemble d'objets de type Shape
- Logique métier de nos classes devient fortement couplée
- Donc rend le code difficile à comprendre et à maintenir

Le design pattern Façade

Solution :

- Ajouter une classe qui procure une interface simple vers un sous-système complexe

- Les fonctionnalités plus limitées que si on interagit directement avec le sous-système

- N’inclure que les fonctionnalités qui intéressent notre client

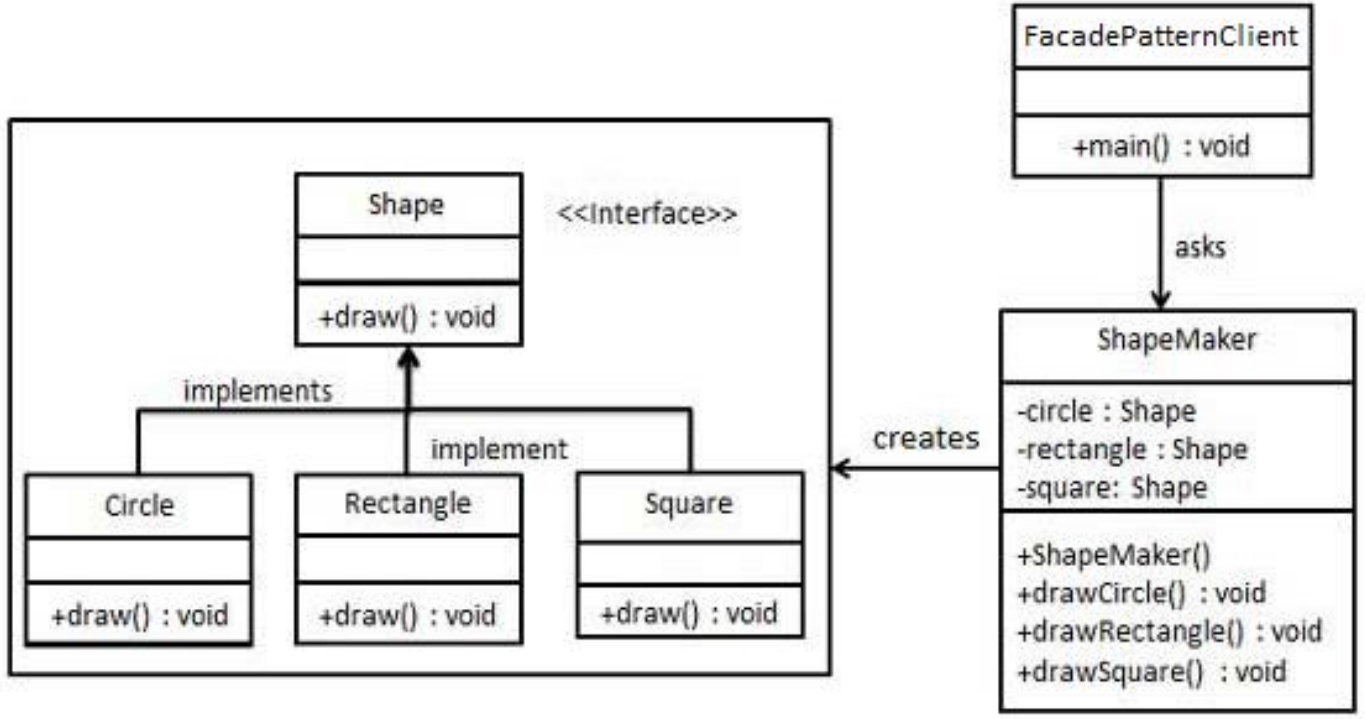
Application du Design Pattern Façade:

- Diagramme de classe permet de visualiser un modèle de conception type au moment de la compilation mais surtout les classes et les relations qu’elles ont entre elles

- Diagramme de séquences au moment de l’exécution et montre les objets et les interactions entre eux

Le design pattern Façade

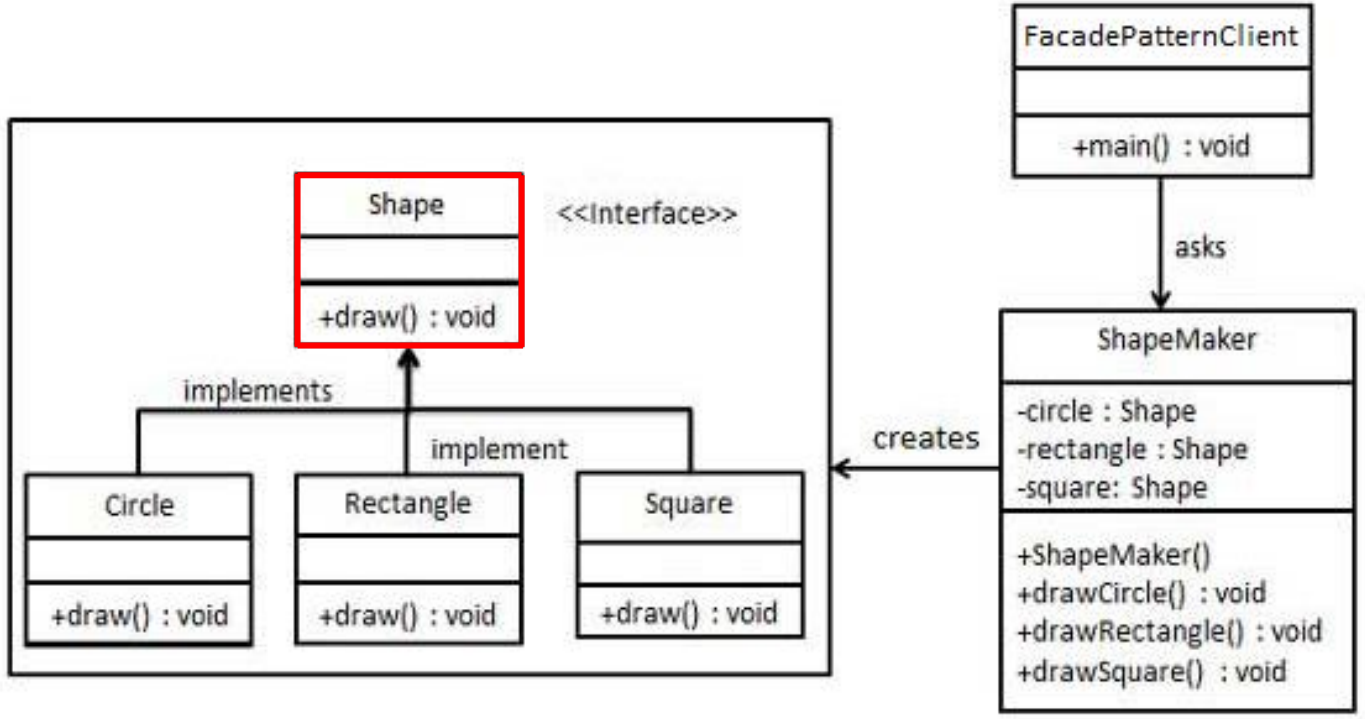
Diagramme de Classe :



Shape :

- Interface
- Méthode dessiner
- Pas de paramètres.
- But : dessiner la forme géométrique demandée !

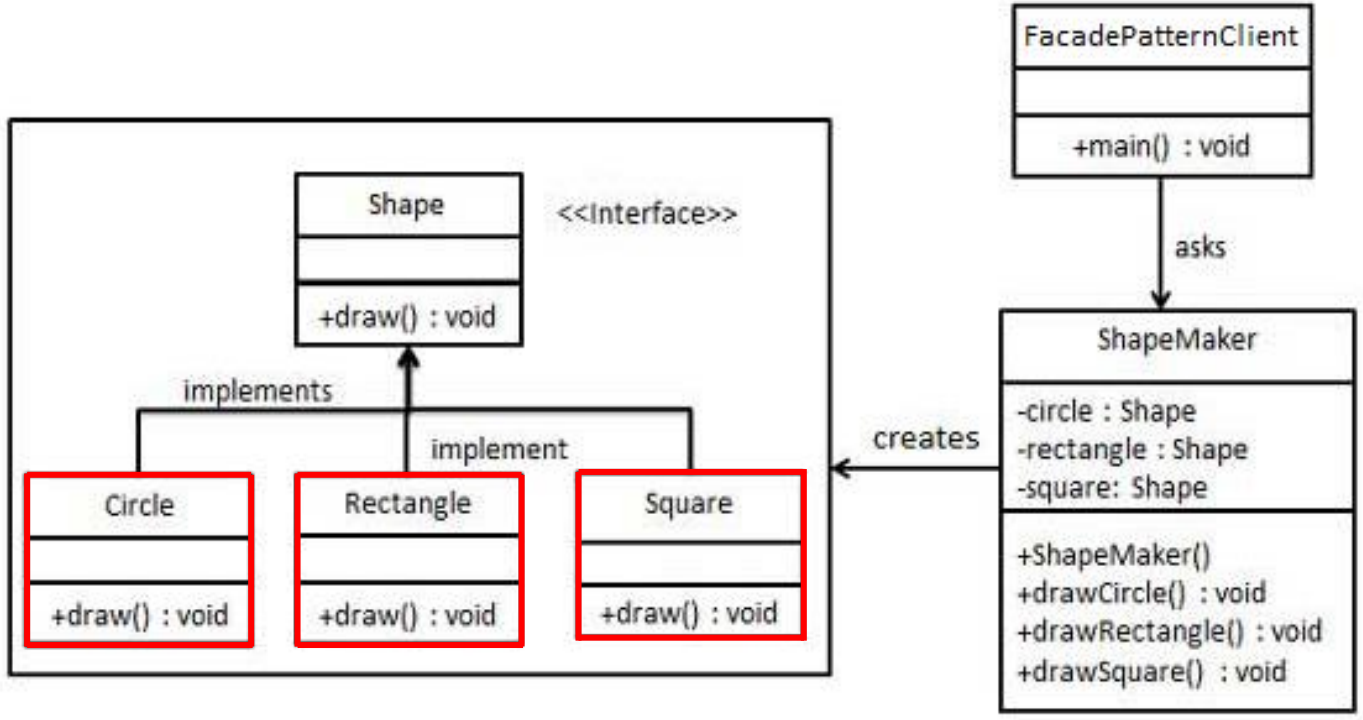
Diagramme de Classe :



Circle, Rectangle et Square :

- Classes concrètes
- Implémentent Shape.
- Méthode dessinée.
- But : rediriger l'appel vers la méthode appropriée

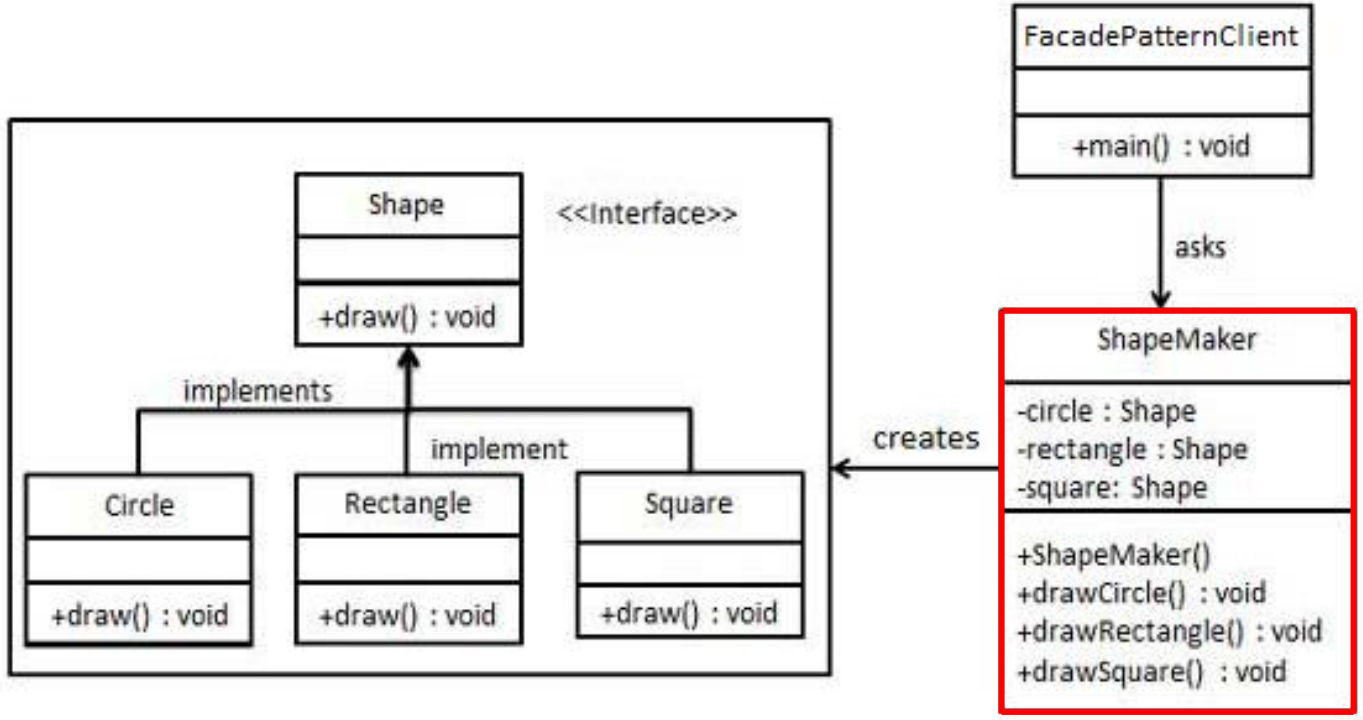
Diagramme de Classe :



ShapeMaker :

- Classe de façade (concrète)
- Prend trois attributs
- Utilise l'interface Shape
- Appelle chaque méthode dessiner !

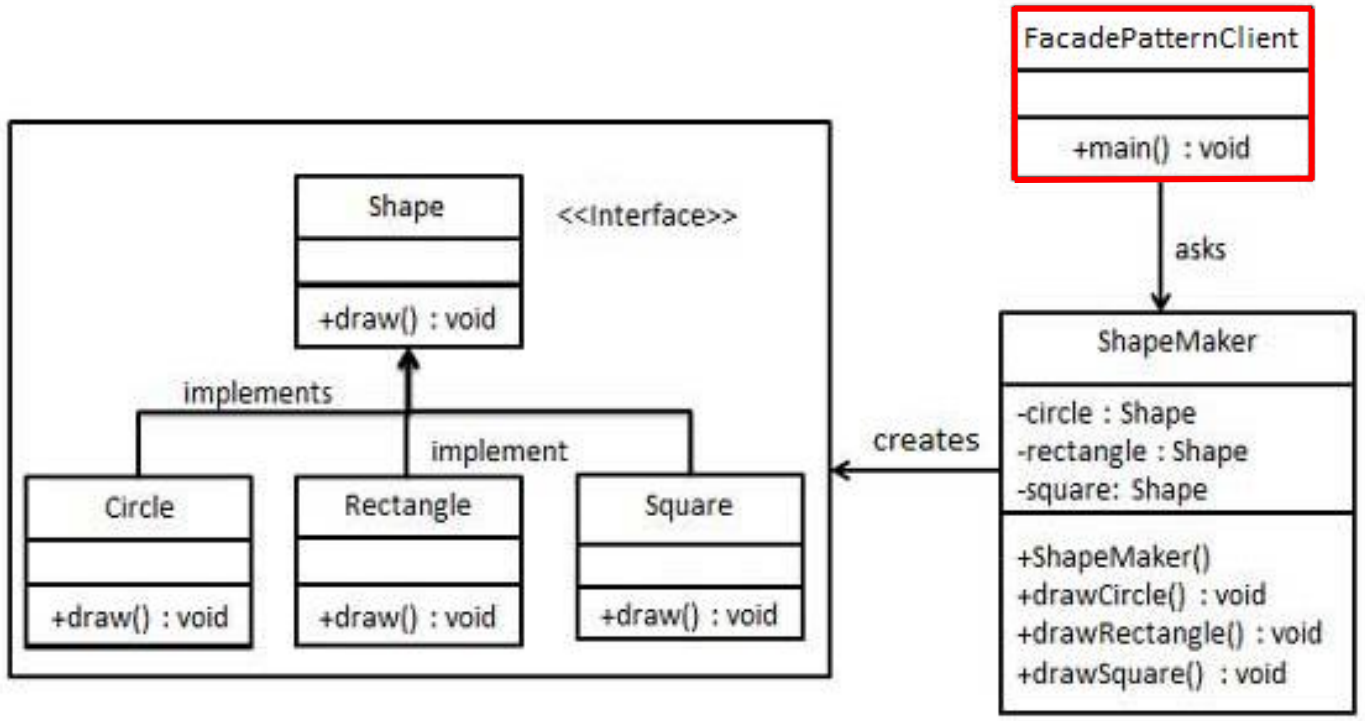
Diagramme de Classe :



FacadePatternClient :

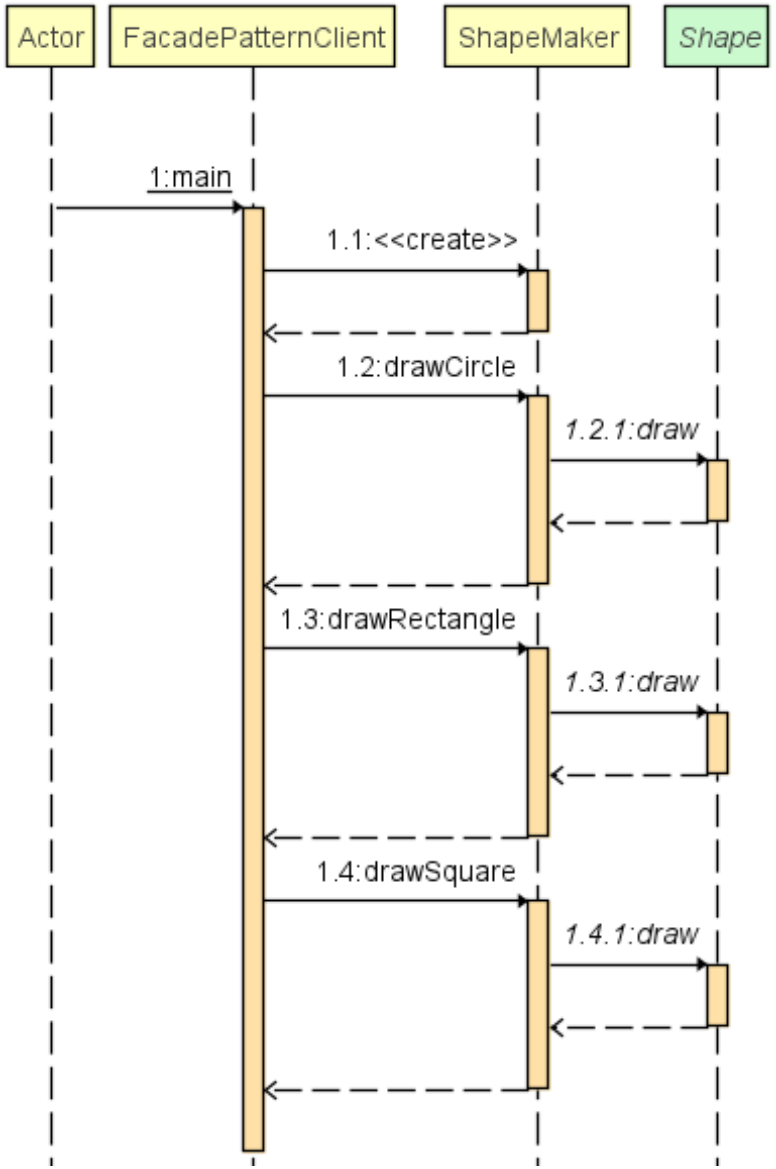
- Classe concrète
- Utilise la façade
- Joue le rôle de client
- But : dessiner des formes via la classe façade

Diagramme de Classe :



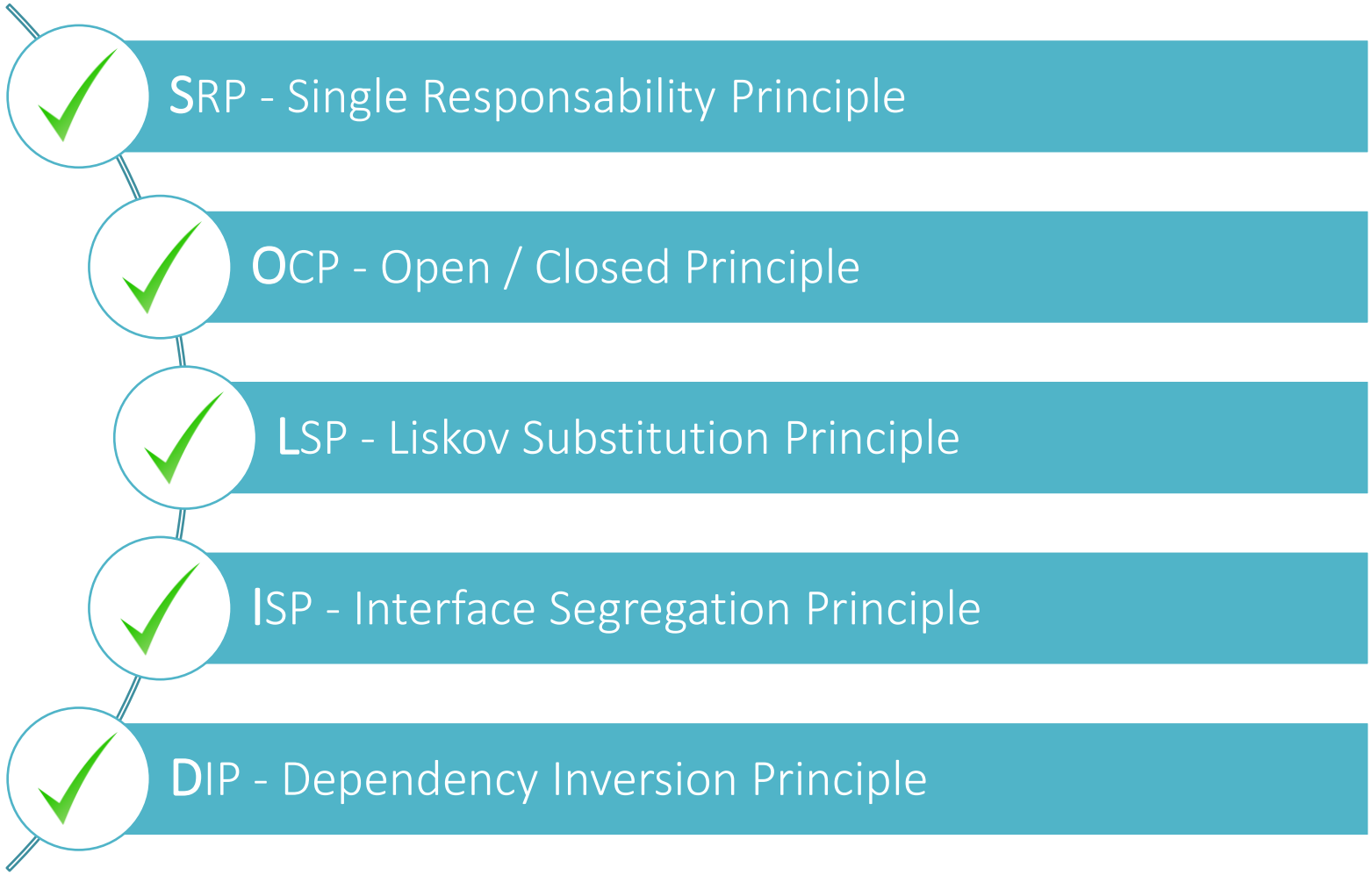
Le design pattern Façade

Diagramme de Séquences :



PRINCIPE SOLID

SOLID - PRÉSENTATION



LIVE CODING

https://unilim-my.sharepoint.com/:v:/g/personal/louis_monie_etu_unilim_fr/EZ3hZ0INqIhEoPYWIkmTdAIBloewUGWZK63vkr_CSJSHgg?e=lGH2N2

Diagramme de Classe – Live Coding

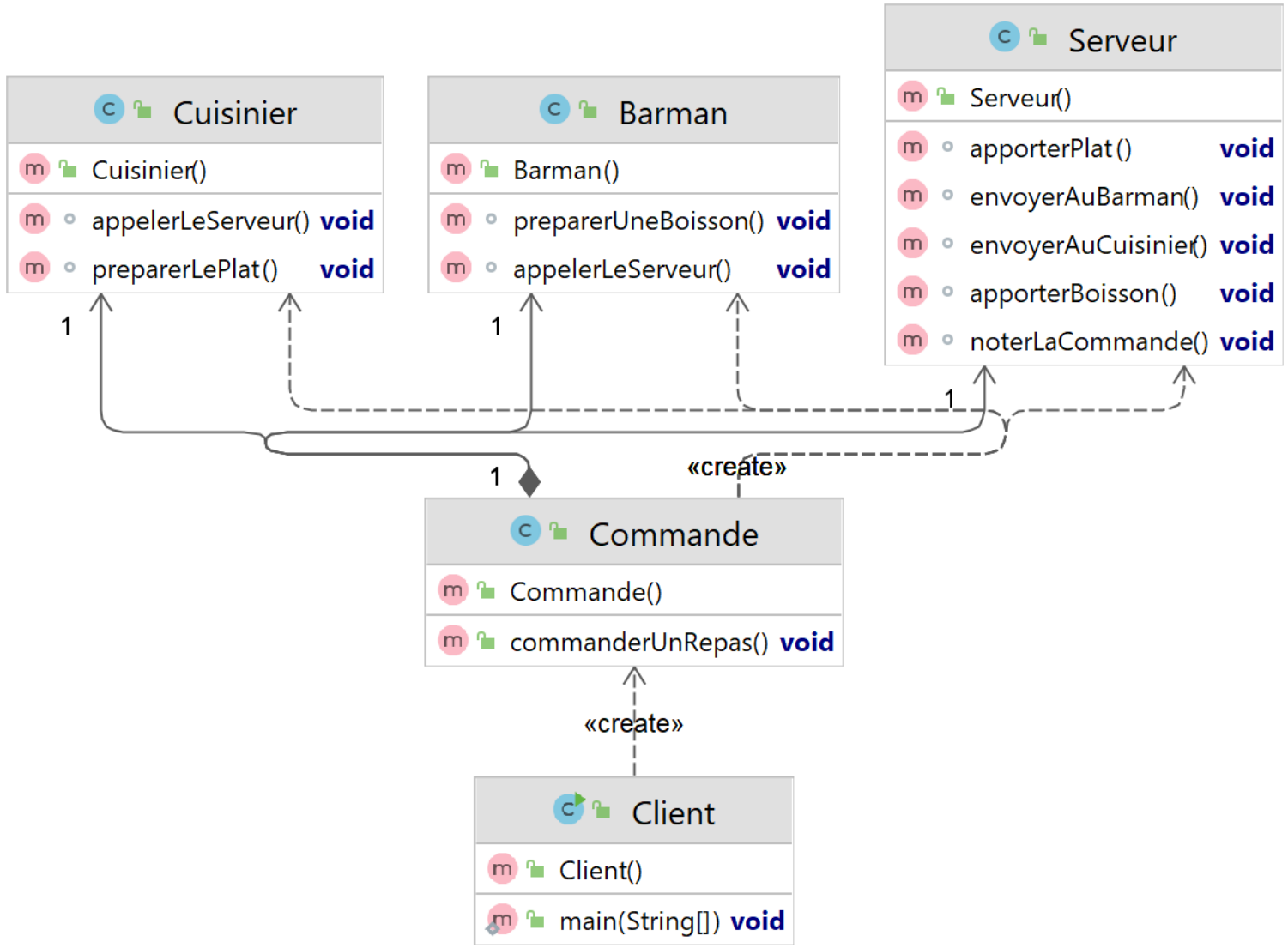
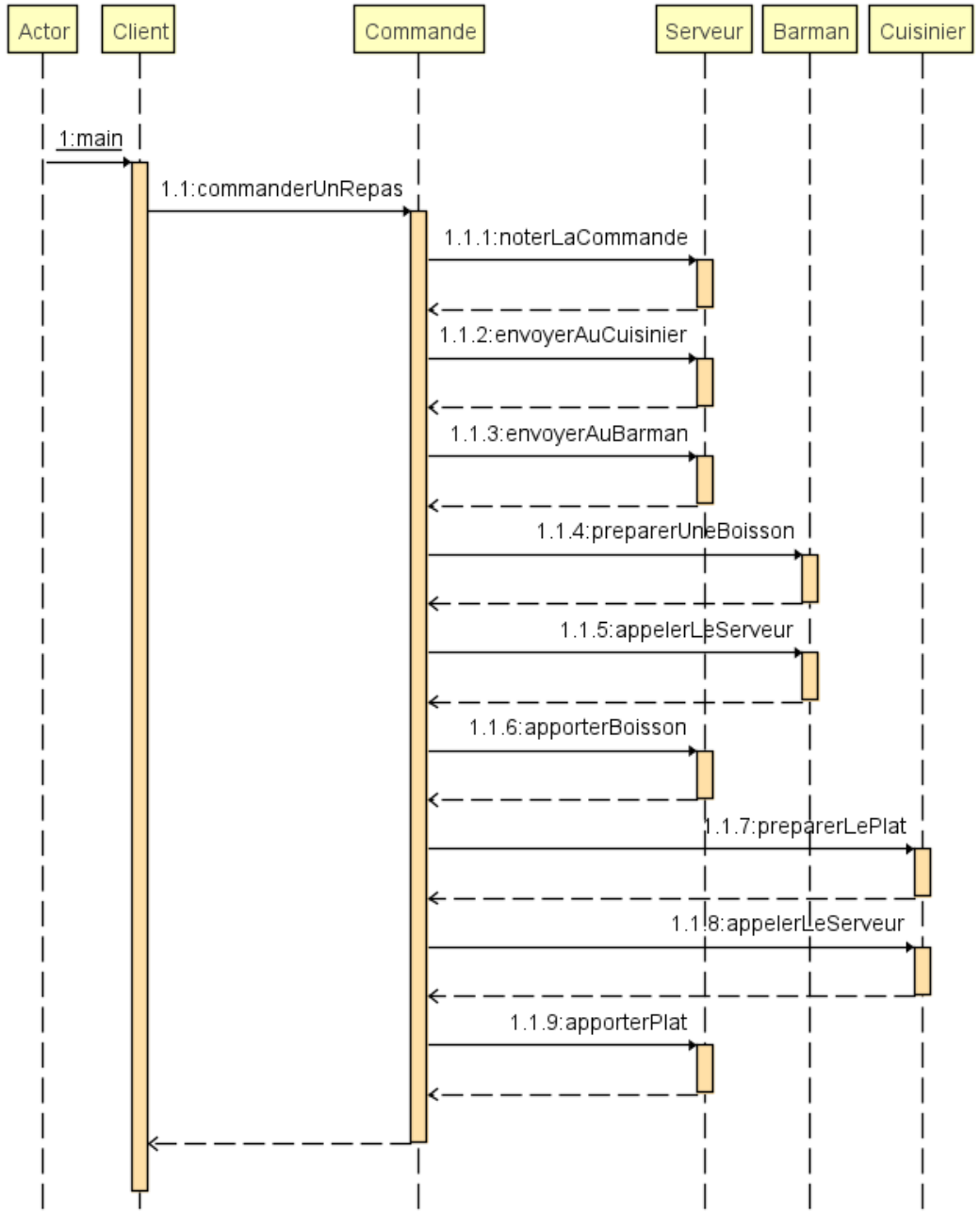
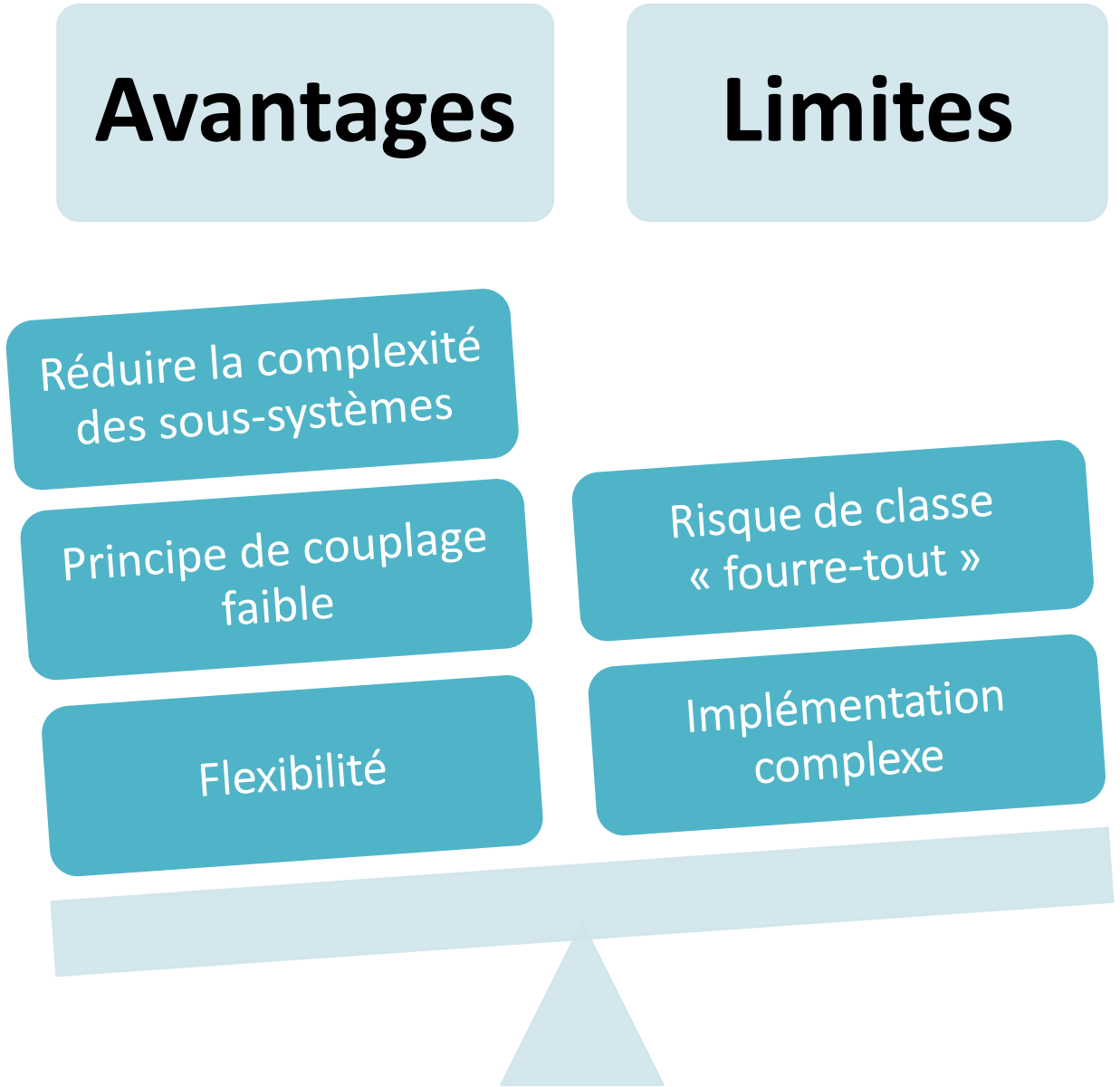


Diagramme de Séquence – Live Coding

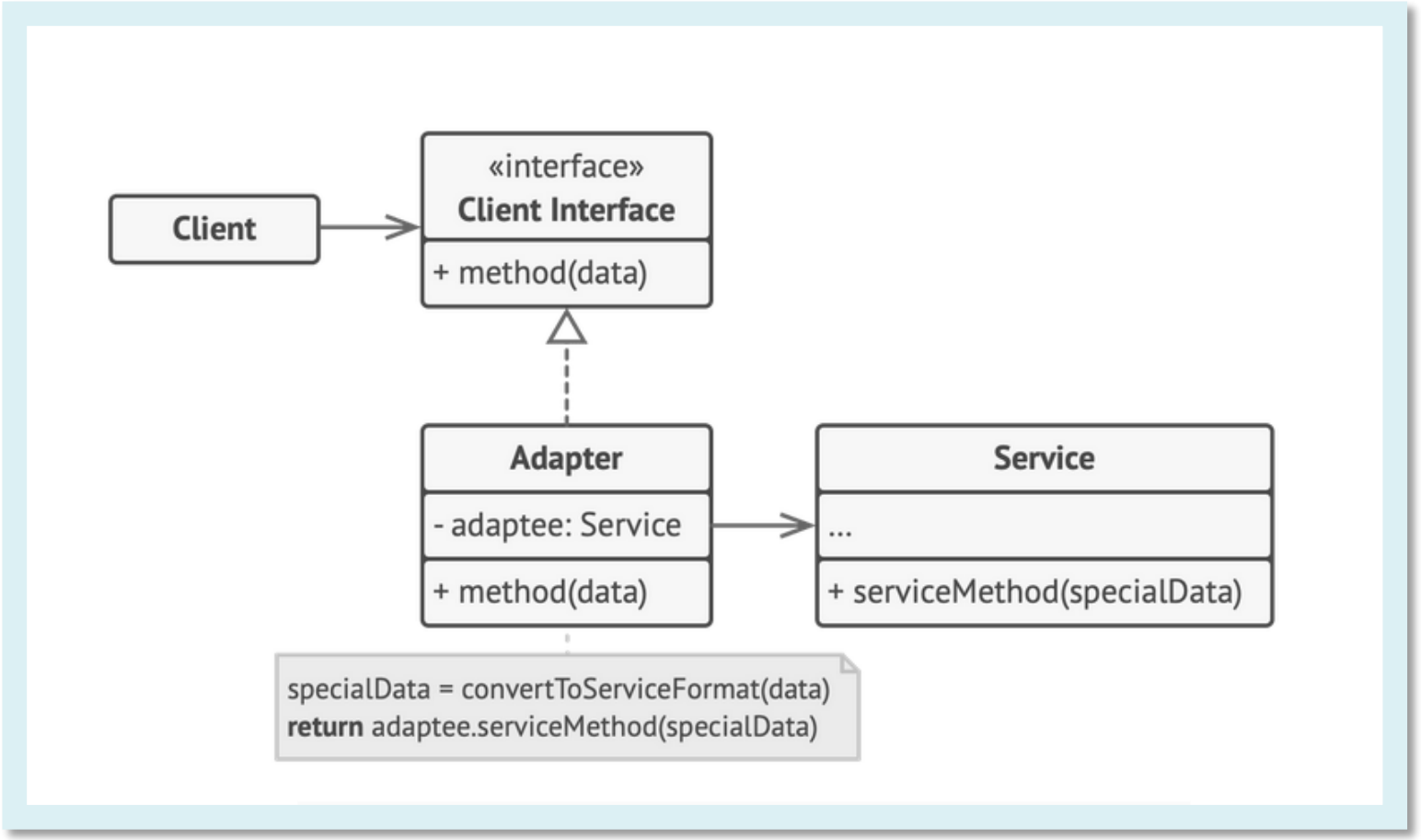


AVANTAGES ET LIMITES


Avantages
Limites
Réduire la complexité des sous-systèmes
Principe de couplage faible
Flexibilité
Risque de classe « fourre-tout »
Implémentation complexe

RAPPROCHEMENTS

Rapprochement avec Adaptateur



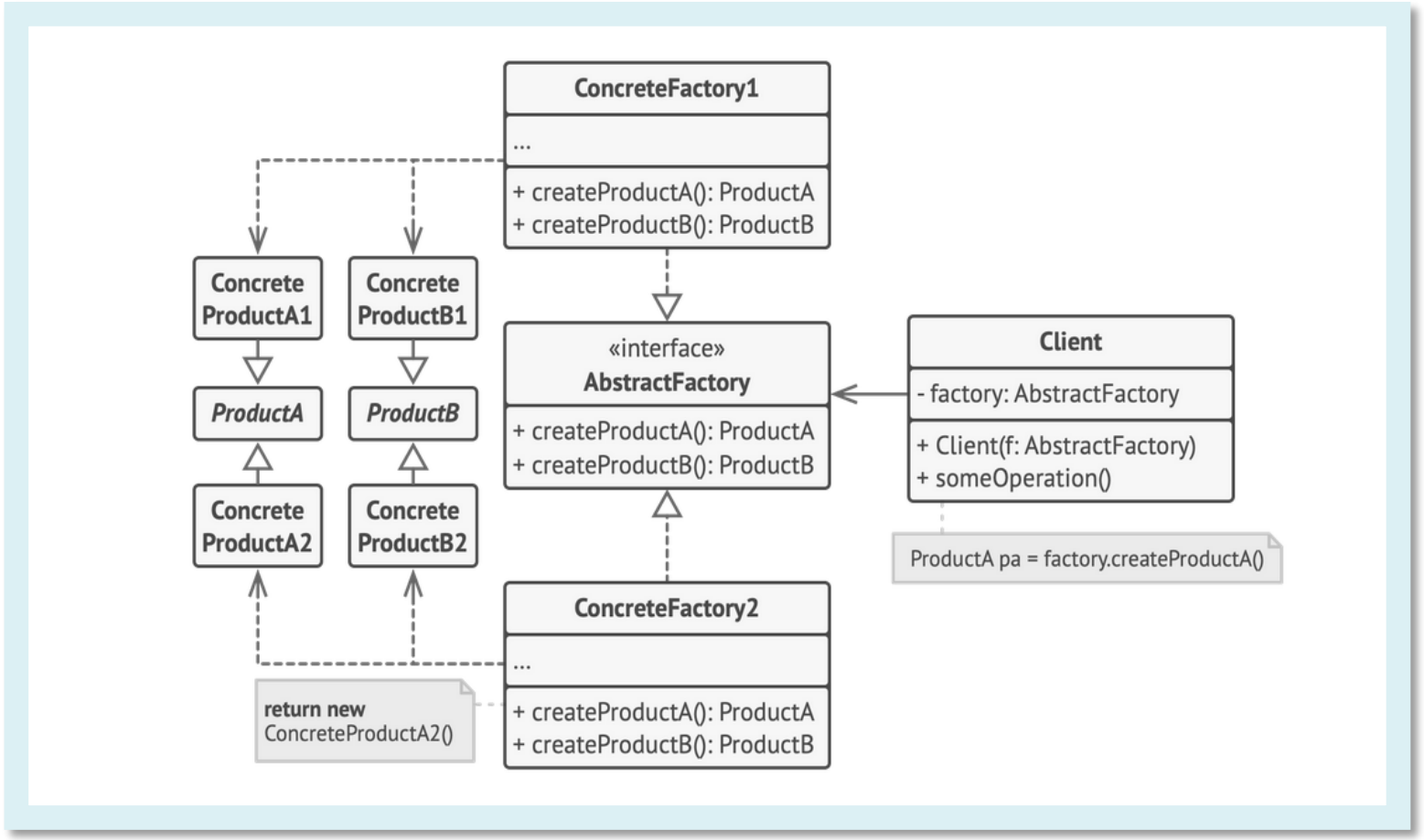
Façade

Simplifier les interactions entre un Client et une Interface

Adaptateur

Permettre la conversion entre une interface A et une interface B

Rapprochement avec Fabrique Abstraite



Façade

Cacher l'implémentation de n'importe quelle opération

Fabrique Abstraite

Cacher seulement la création des objets

QCM

Metsker, Steven John, et William C. Wake. *Les design patterns en Java [Texte imprimé] : les 23 modèles de conception fondamentaux*. Pearson, 2009. Catalogue des BU, IUT Limoges 005.133(JAV)
MET, *EBSCOhost*, http://ezproxy.unilim.fr/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat06519a&AN=cbu.250404&lang=fr&site=eds-live.

Design patterns Texte imprimé les 23 modèles de conception descriptions et solutions illustrées en UML 2 et Java [Laurent Debrauwer] https://eds.p.ebscohost.com/eds/detail/detail?vid=1&sid=e24bb888-d545-4c62-88e3-84c0e5df7530%40redis&bdata=Jmxhbmc9ZnImc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=cbu.232225&db=cat06519a

https://anceret-matthieu.fr/2018/08/les-design-patterns-et-les-principes-solid-en-d%C3%A9veloppement-logiciel-1/4/

https://anceret-matthieu.fr/2019/03/les-design-patterns-structural-3/4/

https://sourcemaking.com/design_patterns

https://refactoring.guru/fr/design-patterns/facade

https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-quun-facade-pattern/

https://cdiese.fr/design-pattern-facade/

https://fr.wikibooks.org/wiki/Patrons_de_conception/Fa%C3%A7ade

https://sourcemaking.com/design_patterns/facade

https://riptutorial.com/design-patterns/example/15810/facade-example-in-java

Références