

A large blue geometric shape, resembling a parallelogram or a tilted rectangle, occupies the right half of the slide. It has a slight gradient and a dark blue border.

Design Pattern **Façade**

Réalisé par Yannick Berckmans

Introduction

- ▶ Le Design Pattern
 - ▷Présentation
 - ▷Explications
- ▶ Application réalisée
 - ▷Présentation
 - ▷Introduction du Pattern
 - ▷Mode d'emploi
- ▶ Conclusion

A large yellow geometric shape, consisting of a parallelogram and a triangle, occupies the right half of the slide. It is composed of two shades of yellow, with a darker shade forming the main body and a lighter shade forming a triangular section on the right.

1.

**Le Design
Pattern**

Le Design Pattern

Présentation

- ▶ Design pattern **façade**
- ▶ **Problématiques**
 - ▷ Systèmes de plus en plus **complexe**
Difficulté de prendre des systèmes en main
 - ▷ Systèmes de plus en plus **couplés**
Un unique changement dans un système entraîne un changement de tout les systèmes liés

Le Design Pattern

Présentation

► Solutions

- La façade est un « **interface** » simple

Abstraction du fonctionnement interne difficile

Utilisation simple pour un système complexe

- Point d'accès **unique** au système

► Conséquences

- Implémentation du système **modifiable** sans changer l'interface

Couplage diminue, modularité augmente

- **Tri** des actions réalisable par l'utilisateur

- **Grand simplification** du système pour l'utilisateur

Le Design Pattern

Explications

- ▶ Classe « façade » **intermédiaire**

Elle se trouve entre le système et l'extérieur

- ▶ Séparation « **Client – Serveur** »

- ▶ Point d'accès unique au système

Système simplifié et protégé

- ▶ Joue le rôle d'une « **API** »

- ▶ **Abstraction** du fonctionnement interne

On définit ce que peut faire l'utilisateur

Le design pattern façade consiste à créer une classe intermédiaire qui proposera l'ensemble des actions disponibles au client/utilisateur. Ce dernier n'a accès qu'à cette classe, et n'a aucune idée du fonctionnement du système sous-jacent. L'utilisateur doit uniquement comprendre comment utiliser la classe « façade » et savoir ce que cette dernière renvoie. C'est dans cette classe que peuvent se trouver l'ensemble des tests et vérifications précédant l'exécution du système.

La classe « façade » réalise en elle-même une simplification du système, mais également un « début de mode d'emploi ».

Le Design Pattern

Explications

► Trois grands acteurs

► Utilisateurs/client

Utilise le programme et les méthodes de la façade

► Façade

Connait l'implémentation interne du système

Utilise les différentes classes pour réaliser les requêtes utilisateurs

► Classes du système

Fonctionnent comme en temps normal

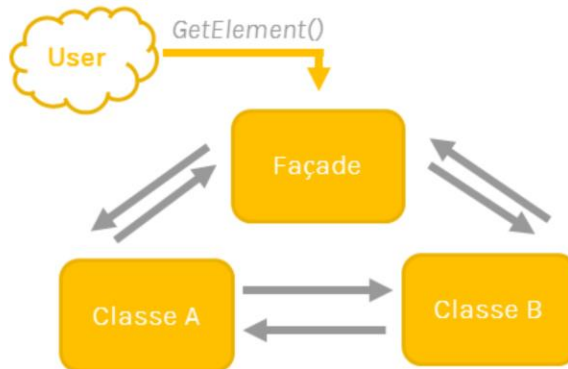
Reçoivent les requêtes de la façade et les traitent

Elles sont le cœur même du programme

Le Design Pattern

Explications

- ▶ Exemple système avec un design pattern « façade »
- ▶ Accès **similaire** malgré l'implémentation

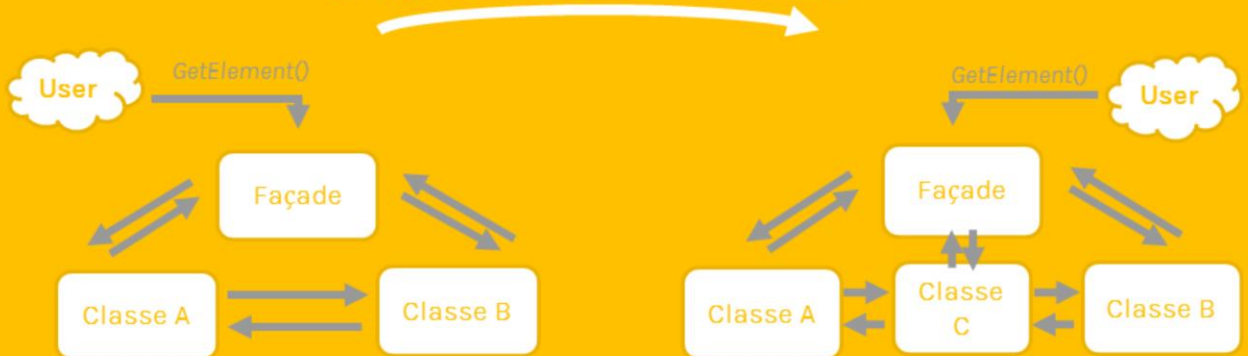


Le Design Pattern

Explications

- Accès similaire malgré un changement d'implémentation

Changement de fonctionnement interne



Le but de ce design pattern est de pouvoir changer l'implémentation interne du système sans devoir notifier tout les clients du « changement de fonctionnement » du système. Ce dernier peut changer du tout au tout, il n'y aura aucun impacte pour l'utilisateur tant que la façade utilise les mêmes entrée et renvoi les même éléments en sortie.

A large red diagonal graphic element that starts from the top right and extends towards the bottom left, creating a sharp triangular shape on the right side of the slide.

2.

**Application
réalisée**

Application réalisée

Présentation

- ▶ Application à design **simpliste**
- ▶ Application de gestion d'objets
Classes élèves, professeurs et « groupes »

Introduction du pattern

- ▶ Fonctionnement interne **indépendant**
Pas d'adaptation nécessaire en cas de changement interne (utilisation matricule, ...)
- ▶ Fonctionnement à partir de l'extérieur **simplifié**
Pas de manipulation de variable, classes, objets

Application réalisée

Mode d'emploi

- ▶ Quelques objets déjà **pré-générés**
- ▶ **Shell** avec commandes pour utilisation simple
- ▶ **Toutes les commandes passent par la façade**

```
waiting for input (type cmd to get all the commands available)...
>>>cmd
-----
List of all commands available
-Create a student: newS firstname lastname age avg
-Create a teacher: newT firstname lastname age subject
-Create a group (note the - between students): newG groupName teacherName student1-student2-student3
-printS to get all the students
-printT to get all the teachers
-printG to get all the groups
-----
-blond to get all the blonds
-brunet to get all the brunets
```

Il n'y a pas de manipulation directe des objets: on passe par des méthodes et des fonctions simplifiées au maximum.

Le traitement interne est totalement indépendant de l'exécution de l'utilisateur. Il n'a aucune idée de la manière dont les objets sont créés ou stockés.

Il connaît et n'a besoin que des méthodes qu'on lui propose directement, rien de plus.

Application réalisée

Mode d'emploi

- ▶ Classe façade utilise les méthodes des objets
- ▶ Uniquement quelques actions possibles

```
def newGroup(self, name, teacher, students):
    StudentGroup(name, teacher, students)

def getGroup(self, groupname):
    for group in StudentGroup.getGroups():
        if (group.getName() == groupname):
            return group
    print("no group with this name found")
    return null
```

```
classroom unit1
classroom unit2
classroom unit3
```

```
def printTeachers(self):
    string = ""
    for teacher in Teacher.getTeachers():
        string += teacher.toString()

    return string

def printGroups(self):
    string = ""
    for group in StudentGroup.getGroups():
        string += group.toString()

    return string
```

```
classroom unit1
```

Application réalisée

Mode d'emploi

► **Création** d'élèves et de professeurs

► **Création d'un groupe**

*Mentionner le prénom des professeurs et
étudiants à ajouter dans le groupe*

► **Exemple**

newS Yannick Berckmans 23 16

newS Charles Vandevoorde 22 18

newT Sebastien Combefis 30 Informatique

newG BestGroup Sebastien Yannick-Charles

PrintG

3.

Conclusion

Conclusion

- ▶ Design pattern simple à prendre en main
- ▶ Facilite grandement les modifications
- ▶ Couche supplémentaire simplificatrice
- ▶ Force à réfléchir « utilisation concrète »