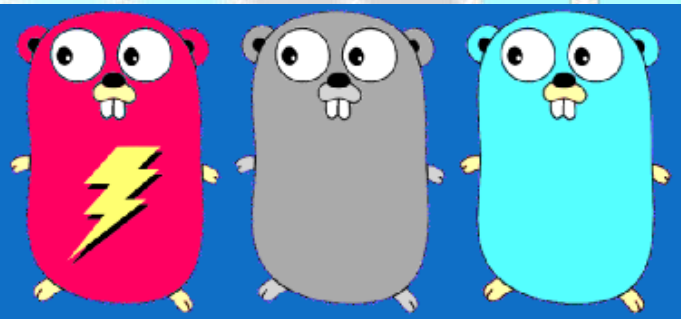
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

# Génie Logiciel pour le Calcul Scientifique



#5

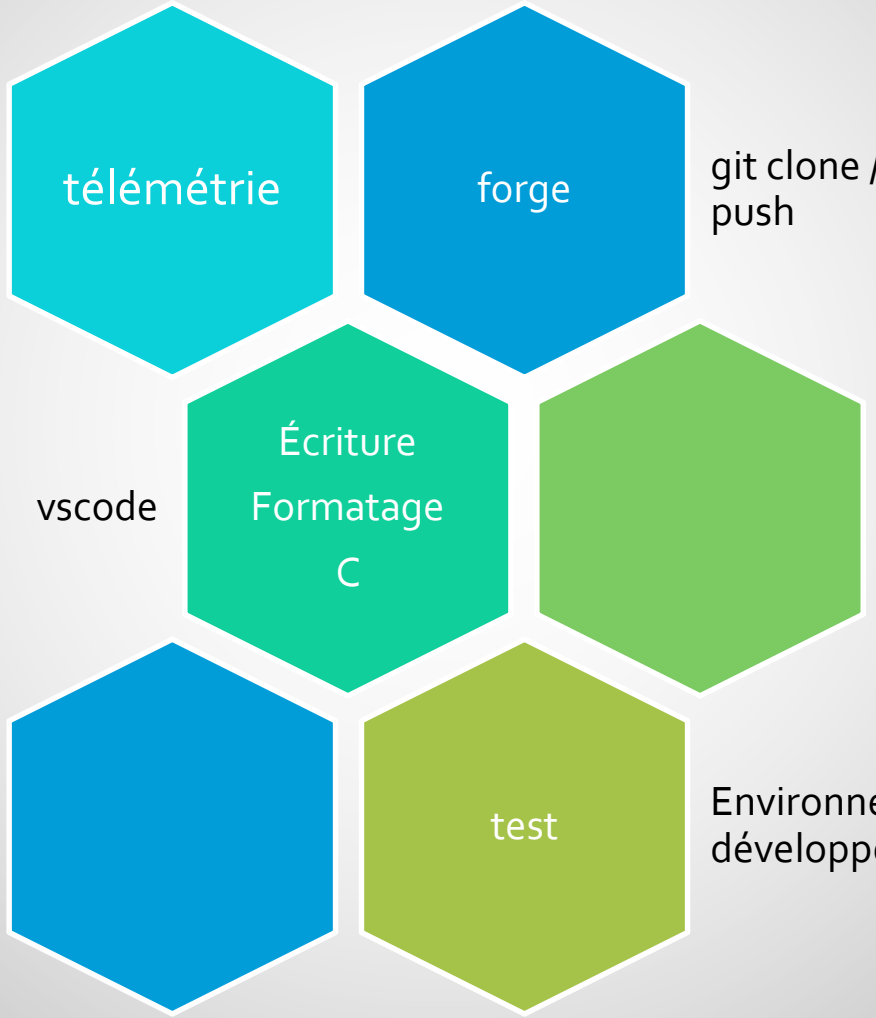
10/01/2025

jean-michel.batto@cea.fr

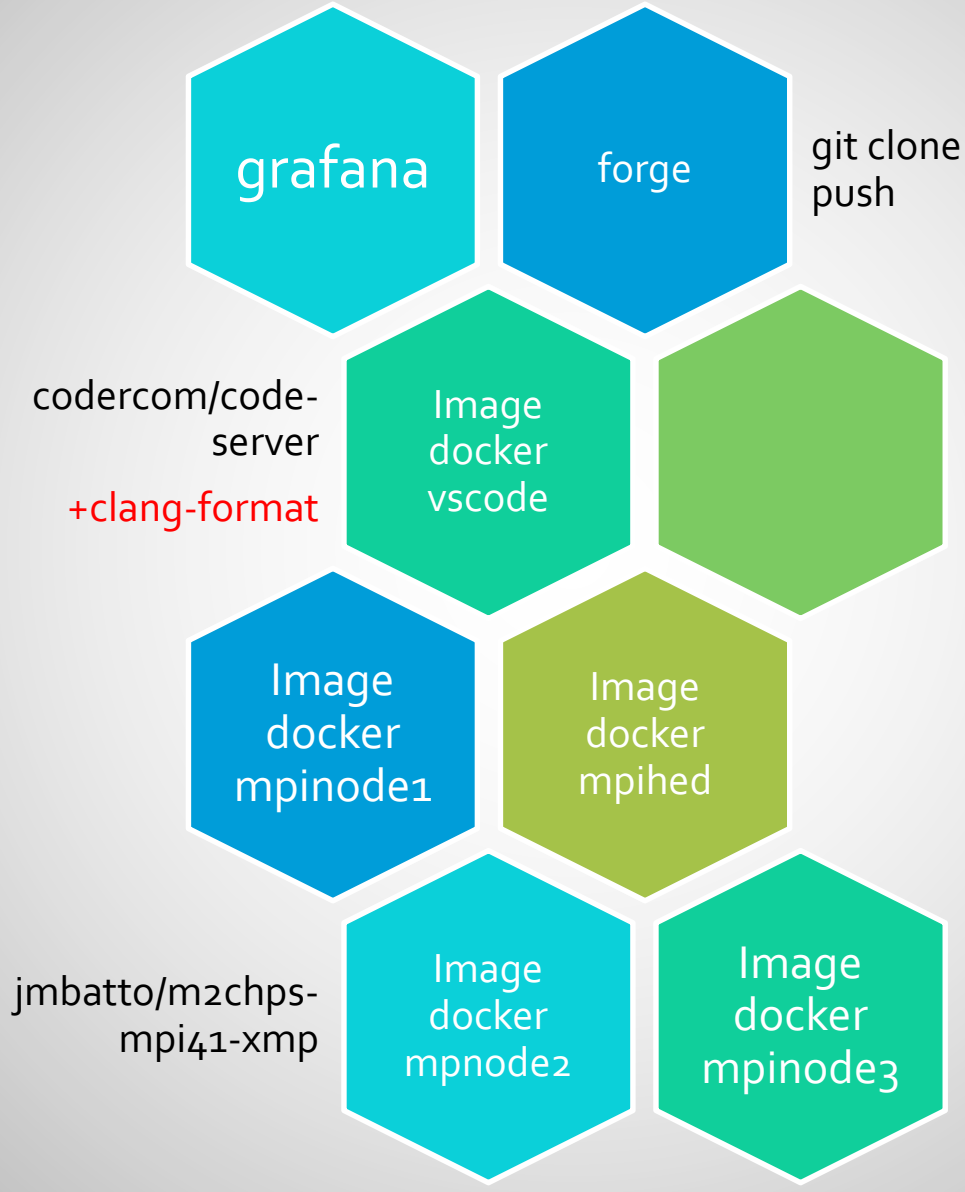
cea

https://gogs.eldarsoft.com/M2_IHPS

# Cycle de vie du développement



# Cycle de vie du développement



# Activité TD

- Séquence de mise en place d'une configuration « remote » ➔ accès au développement via un navigateur web
- Intérêt :
  - Ne pas dépendre d'une connexion ssh
  - Aspect normatif (tous les étudiants ont la même installation)
  - Expérimentation de la conteneurisation
- Ne pas faire de mise en production de la solution : il manque le proxy https (Traefik)

# Activité TD

INTERACTIF

- Nous allons avoir 6 conteneurs:
  - Grafana : la télémétrie
    - philhawthorne/docker-influxdb-grafana:latest
  - Vscode : une version http de visual code studio
    - codercom/code-server:latest
  - 4 nœuds MPI nommés : 1 nœud maitre et 3 nœuds esclaves
    - jmbatto/m2chps-mpi41-xmp:latest
- Remarques:
  - Partage de scripts (certif.sh et install_clang_format_go.sh) via les secrets !
  - Le filesystem /usr/local/var/mpishare est partagé avec vscode (attention aux UID)
  - Les nœuds sont nommés pour avoir un hostname ! (et pas de scale possible)
  - Refresh des images au démarrage – sauf si c'est dans l'espace partagé docker-compose

INTERACTIF

- Vous devez adapter le script docker-compose.yml

```
  environment:
    - PASSWORD=XXXXXXXXXXX
    - DOCKER_USER=votrelogin
    - CODER_ACCESS_URL="http://localhost:8081"
user: "1001:1001"
```

➔pour vous permettre de vous connecter à l'image

# Installation du plugin SSH FS

INTERACTIF

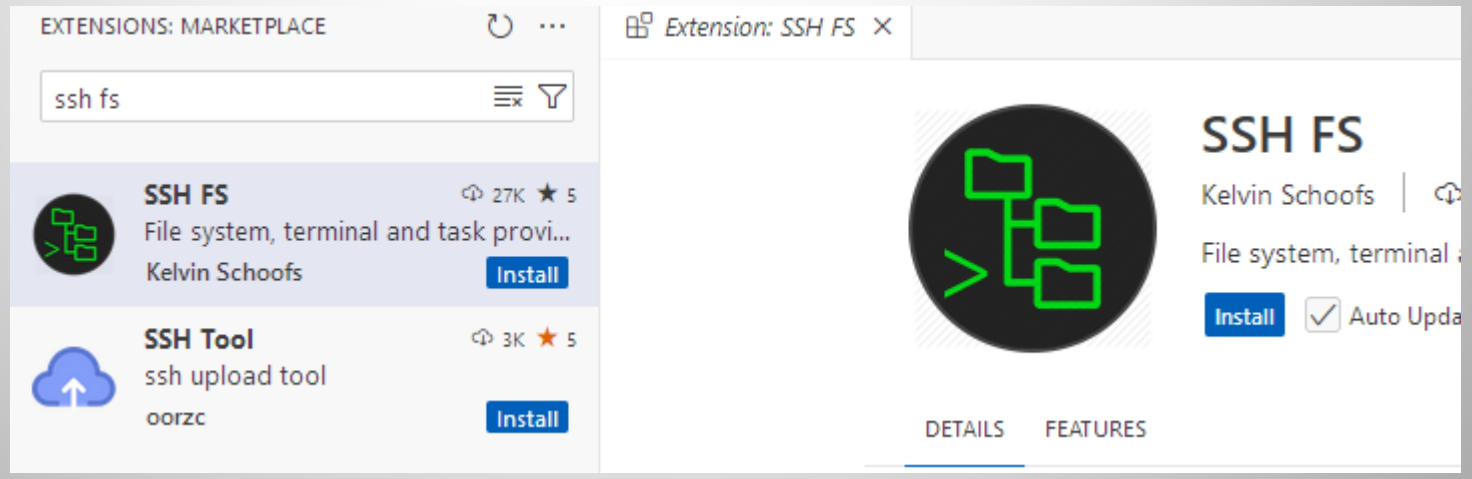
- Dans code-server on veut pouvoir se connecter à un nœud
  - Avant : mettre à jour les certificats dans l'image vscode ➔
    - Ouvrir un terminal, aller dans le dossier .ssh-source, faire un

```
>sh certif.sh
```

Cela va déployer les certificats dans le dossier «utilisateur docker» .ssh

On vérifie en faisant un `ssh mpiuser@mpihead`

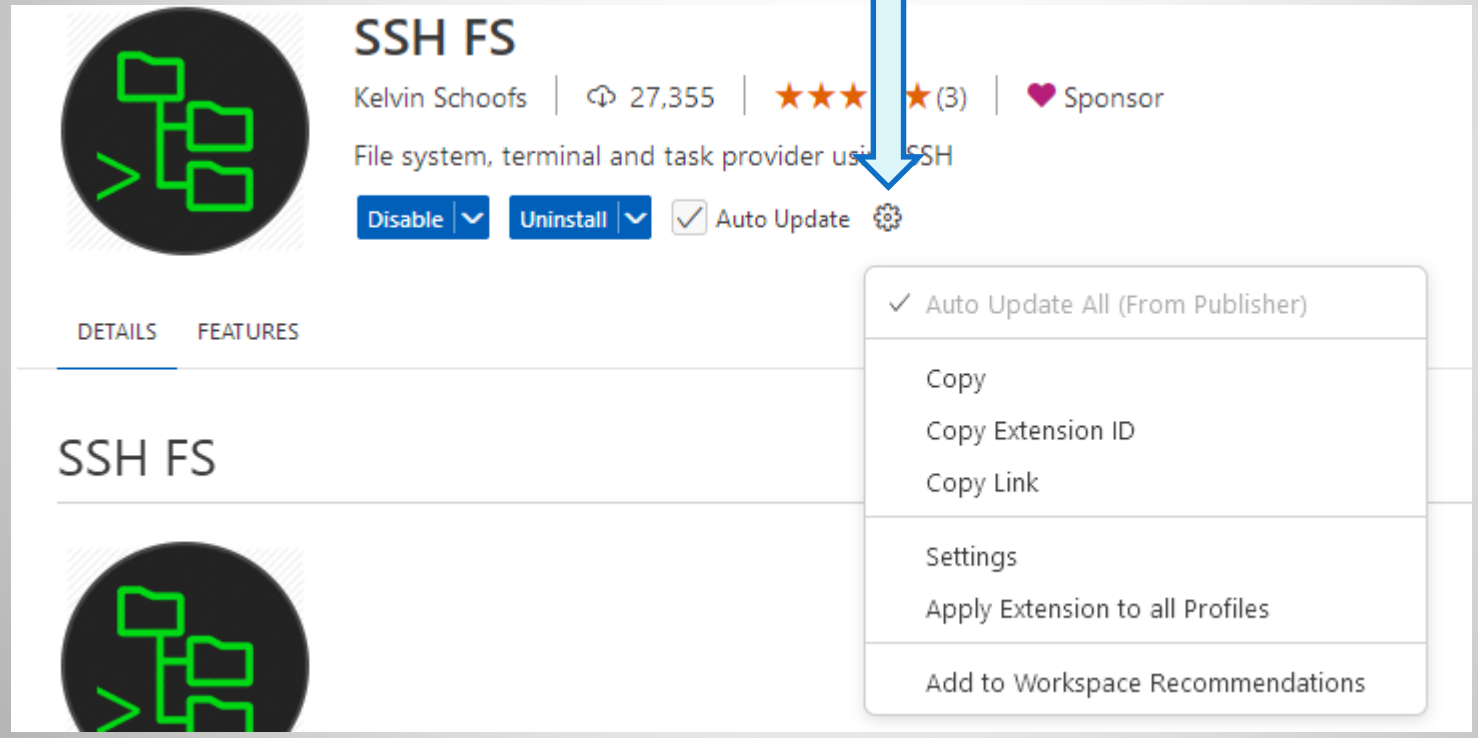
```
//mpiuser : compte non-root associé au login des nœuds et head
```



# Installation du plugin SSH FS

INTERACTIF

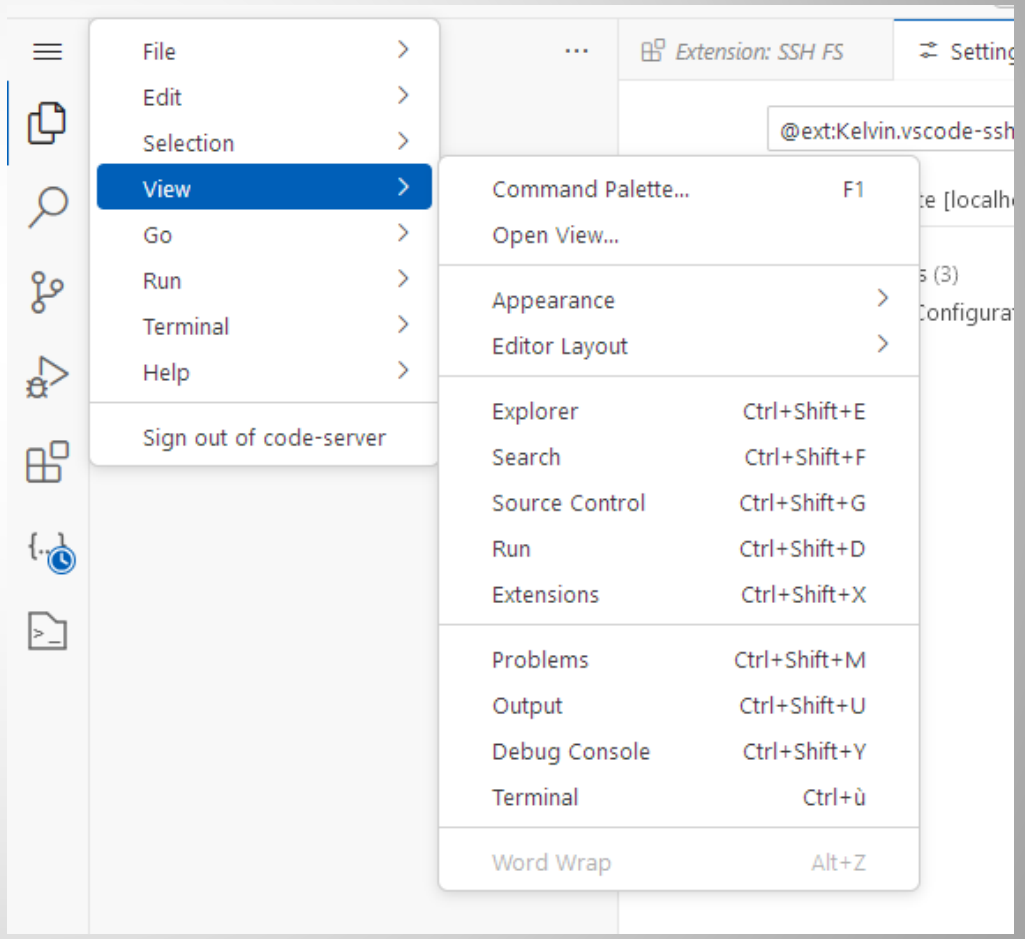
- L'extension a son paramétrage



# Installation du plugin SSH FS

INTERACTIF

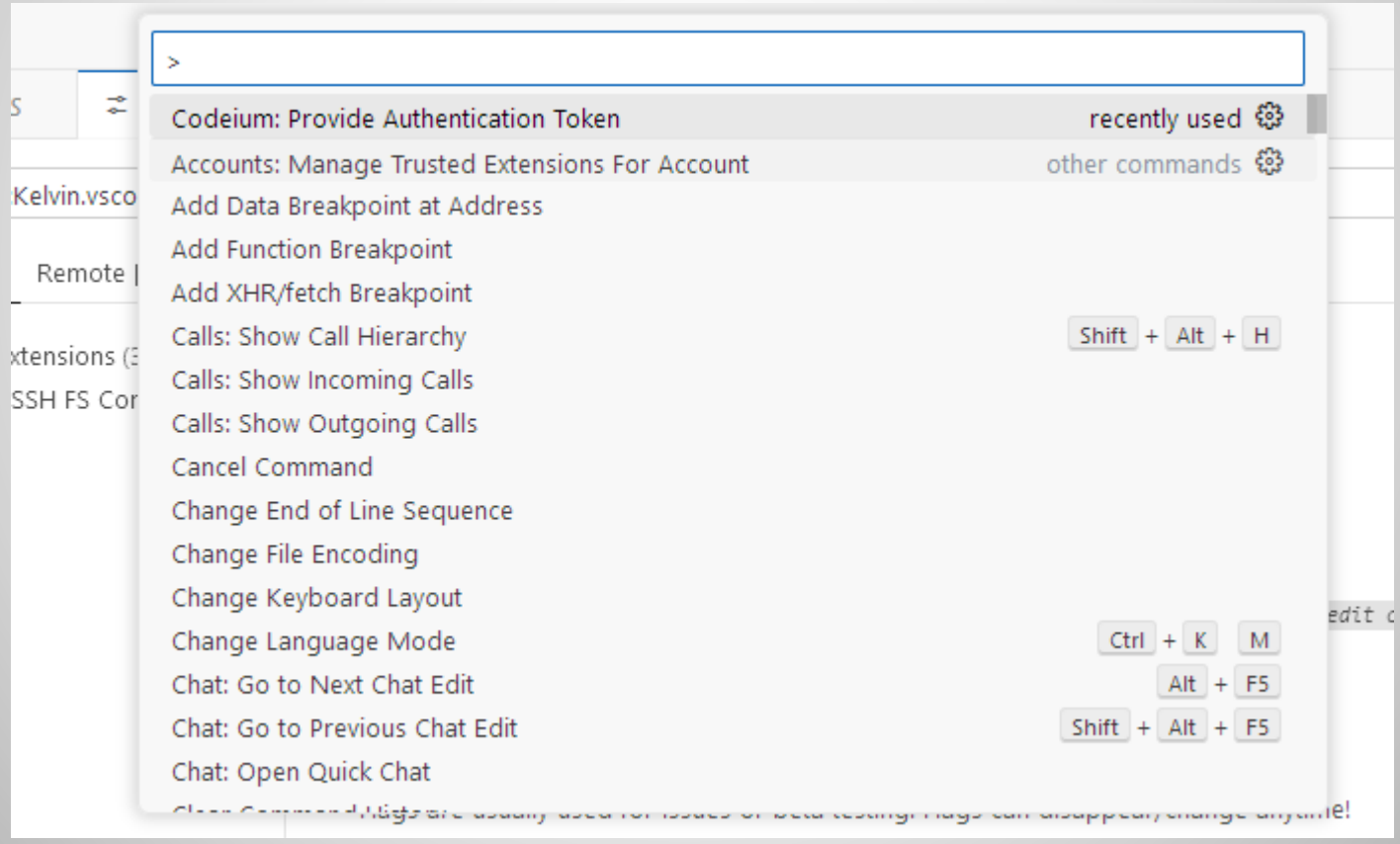
- On peut atteindre le paramétrage via la « Command Palette »



# Installation du plugin SSH FS

INTERACTIF

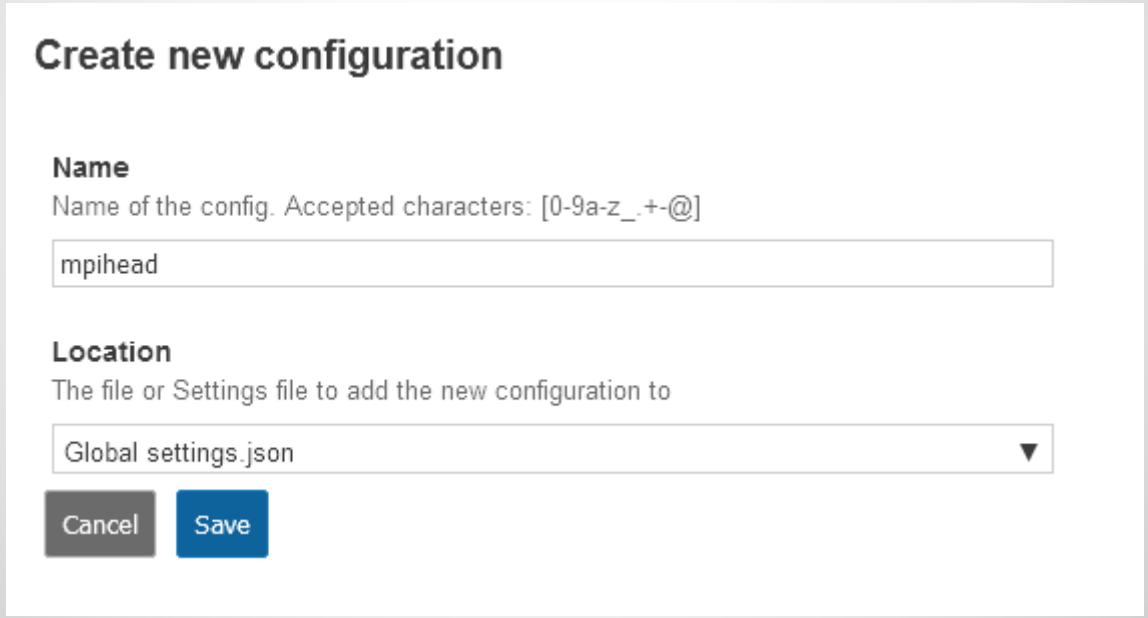
- Command palette



# Installation du plugin SSH FS

INTERACTIF

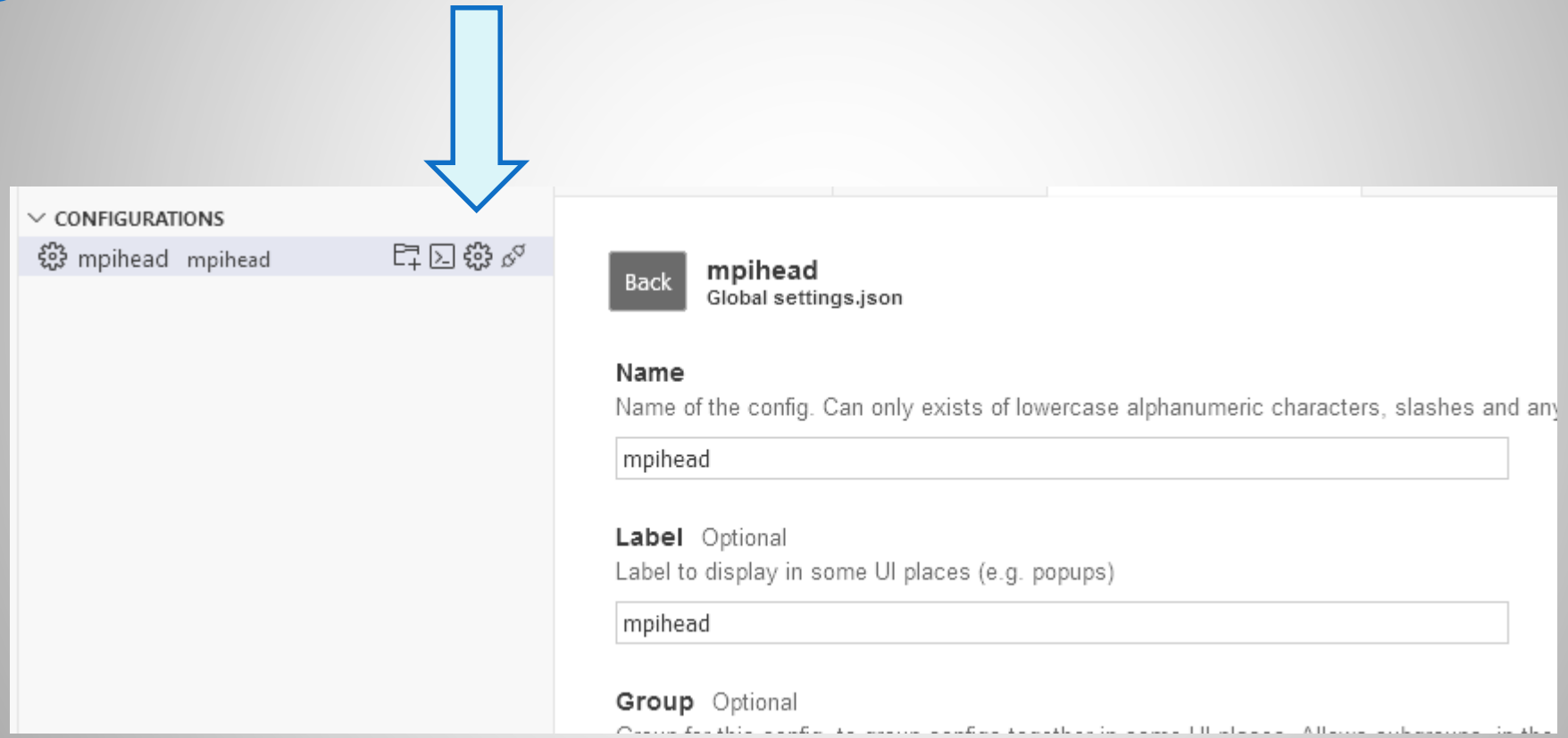
- La configuration
  - Name : mpihead



# Installation du plugin SSH FS

INTERACTIF

- Configuration de ssh fs



# Installation du plugin SSH FS

- Host : mpihead
- Root : /usr/local/var/mpishare
- Username : mpiuser
- Private key : /home/coder/.ssh/id_rsa

**Host** Optional
Hostname or IP address of the server. Supports environment variables, e.g. $HOST
mpihead

**Port** Optional
Port number of the server. Supports environment variables, e.g. $PORT
22

**Root** Optional
Path on the remote server that should be opened by default when creating a terminal or using th
command/button. Defaults to `/`
/usr/local/var/mpishare

**Agent** Optional
Path to ssh-agent's UNIX socket for ssh-agent-based user authentication. Supports 'pageant' for
variables, e.g. $SSH_AUTH_SOCK

**Username** Optional
Username for authentication. Supports environment variables, e.g. $USERNAME
mpiuser

**Username** Optional
Username for authentication. Supports environment variables, e.g. $USERNAME
mpiuser

**Password** Optional
Password for password-based user authentication. Supports env variables. This gets saved in plaintext! Using pr
is recommended!

**Private key** Optional
A path to a private key. Supports environment variables, e.g. `$USERPROFILE/.ssh/myKey.ppk` or `$HOME/.ssh/myKey`
/home/coder/.ssh/id_rsa Prompt

# Installation du plugin clang-format

INTERACTIF

- Installation de clang-format

  Dans le dossier .ssh-source

  sh install_clang_format_go.sh

  ➔ Installation du package et en plus il y a le compilateur go

- Installation de l'extension depuis vscode

# Activité « découverte de XMP »

- Idée : manipuler un bout de code dans un environnement de test multi-nœuds MPI
- Vous allez créer un repo sur gogs
- Vous allez cloner le repo https://gogs.eldarsoft.com/jmbatto/GLCS-CM5-TDXMP et copier le contenu dans votre repo local

- Et nous allons ensemble explorer quelques idées.

# XMP : XcalableMP

- https://xcalablemp.org/handbook
- Auteur : Mitsuhisa Sato
- https://icl.utk.edu/newsletter/presentations/2012/Sato-Updates-on-XcalableMP-PGAS-Language-2012-08-29.pdf
- Modèle de programmation « type SIMD », pour C et Fortran

- Extension du langage basée sur des pragmas
- Programmation parallèle évolutive et tenant compte des performances
- Compilateur source (C+XMP) vers source (C+MPI)
- Cartographie des données et des tâches selon patron de programmation

# Mode opératoire

INTERACTIF

~~1/installation code-server et noeuds~~

2/paramétrage de git dans le terminal vscode

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

3/ dans gogs, vous allez créer un projet, puis faire un `git clone` https://login:password@votreprojet

L'idée est de vous permettre de travailler efficacement !

4/ vous récupérer l'exemple `GLCS-CM5-TDXMP`

# Lancement du code

INTERACTIF

- On peut faire un make run
- Si on se positionne comme root avec la commande ifconfig –a, on voit 2 réseaux (eth0 et eth1)
- mpirun --mca orte_base_help_aggregate 0 --mca btl_tcp_if_include 10.0.1.0/24  -host mpihead,mpinode1,mpinode2,mpinode3 -n 4 worker_program

# Exploration du code init.c

```
#pragma xmp nodes p(2, 2) ➔ on décrit 4 noeuds
#pragma xmp template t(0 : 3, 0 : 3) ➔ la tâche est une matrice 4 * 4
#pragma xmp distribute t(block, block) onto p ➔ la tâche t va être distribuée sur les 4 nœuds (on va donc avoir t(0,0), t(0,1), t(1,0), t(1,1) ➔ p(0) = le head (rank = 0) )
XMP_Matrix A[4][4]; ➔ une matrice 4x4
#pragma xmp align A[i][j] with t(j, i) ➔ inversion des indices
#pragma xmp shadow A[4][4] ➔ on veut un shadow pour la consolidation
```

- Plus loin dans le code
  `#pragma xmp reflect(A)` ➔ synchronisation de l'espace mémoire

https://xcalablemp.org/handbook/distribute.html

# Questions

INTERACTIF

- Que fait ce code ?
- A quoi sert le `usleep(100)` ?
- A quoi servent les barrières MPI
- Est-ce que ce code est explicable par une IA ?

# Les patrons / patterns

Les 23 patterns décrits :

| | Creational | Structural | Behavioral |
|---|---|---|---|
| Class | Factory Method | Adapter (class) | Interpreter |
| | | | Template Method |
| Object | Abstract Factory | Adapter (object) | 13 Chain of Responsibility |
| | 14 Builder | 9 Bridge | Command |
| | Prototype | Composite | Iterator |
| | 8 Singleton | 10 Decorator | Mediator |
| | | 11 Facade | 12 Memento |
| | | Flyweight | Observer |
| | | Proxy | State |
| | | | Strategy |
| | | | 15 Visitor |

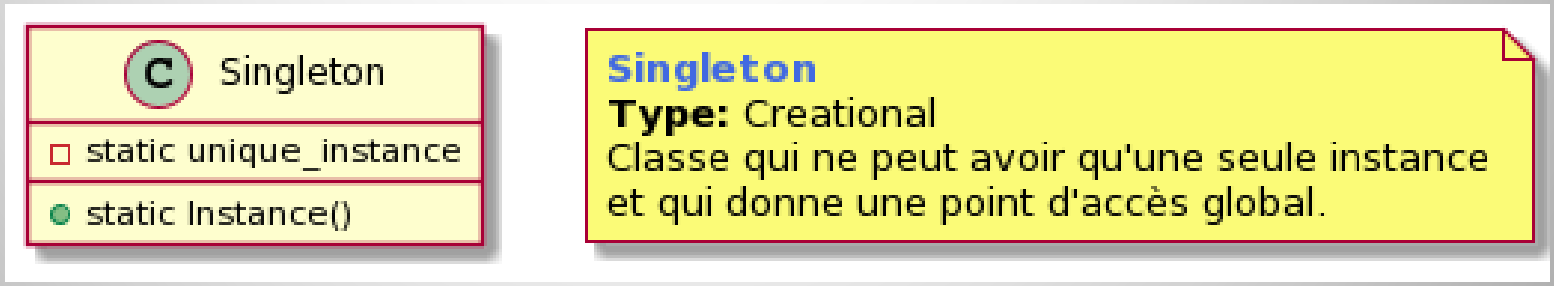
# Les 23 Patterns

| Typologie | Nom du Design Pattern | Ce qui doit être ajuster | verbe | composition sous jacente | mélange de classes ? |
|---|---|---|---|---|---|
| Creational | Abstract Factory | famille d'objets dépendants | | structure | non |
| | Builder | Comment créer un objet composite dont la structure du composite est indépendante | | liste | non |
| | Factory Method | Sous classe d'un objet qui est instanciée sans connaitre la classe ancêtre (connaissance retardée) | | | filtrage vtab |
| | Prototype | Classe d'objet qui est instanciée grâce à un constructeur de copie | copie=verbe | liste | non |
| | Singleton | La seule instance d'une classe | copie=o=verbe | | non |
| Structural | Adapter | accède à un objet en modifiant l'interface | | | non |
| | Bridge | Fait l'implémentation d'un objet par découplage | découplage | | non |
| | Composite | structure et composition d'un objet vue de manière uniforme | | arbre | non |
| | Decorator | Responsabilité d'un objet sans héritage - ajout dynamique | | | filtrage vtab |
| | Facade | Exposer une interface à un sous-système | | | filtrage vtab |
| | Flyweight | cout de stockage d'un objet, partage de l'état | état=verbe | liste | non |
| | Proxy | Comment un objet est accédé, son emplacement (mémoire, disque) - effet miroir | | queue | non |
| Behavioral | Chain of Responsibility | Un objet qui peut répondre à une demande avec découpplage | découplage | queue | filtrage vtab |
| | Command | quand et comment une commande peut être faite - la commande devient un objet | | structure | non |
| | Interpreter | grammaire et interprétation d'un objet | | | non |
| | Iterator | se déplacer dans une structure d'objet sans en connaitre le détail | | liste | filtrage vtab |
| | Mediator | Comment et avec quels objets sont décrites les interactions | | | non |
| | Memento | Quelles sont les informations privées qui sont stockée à part et quand? | état=verbe | état (queue=infinie) | non |
| | Observer | l'effectif des objets observés et quand s'effectue la mise à jour | | queue | non |
| | State | les états d'un objet sont des variables, avec un handler() | état=verbe | structure | filtrage vtab |
| | Strategy | un algorithme | extension=verbe | structure | filtrage vtab |
| | Template Method | les étapes/squelette d'un algorithme | | structure | non |
| | Visitor | les opérations élémentaires sont appliquées à un objet sans modifier sa classe | | liste | non |

# Creational / Singleton

- Classe qui ne donne qu'une seule instance
- https://godbolt.org/z/5916Mc6Ez

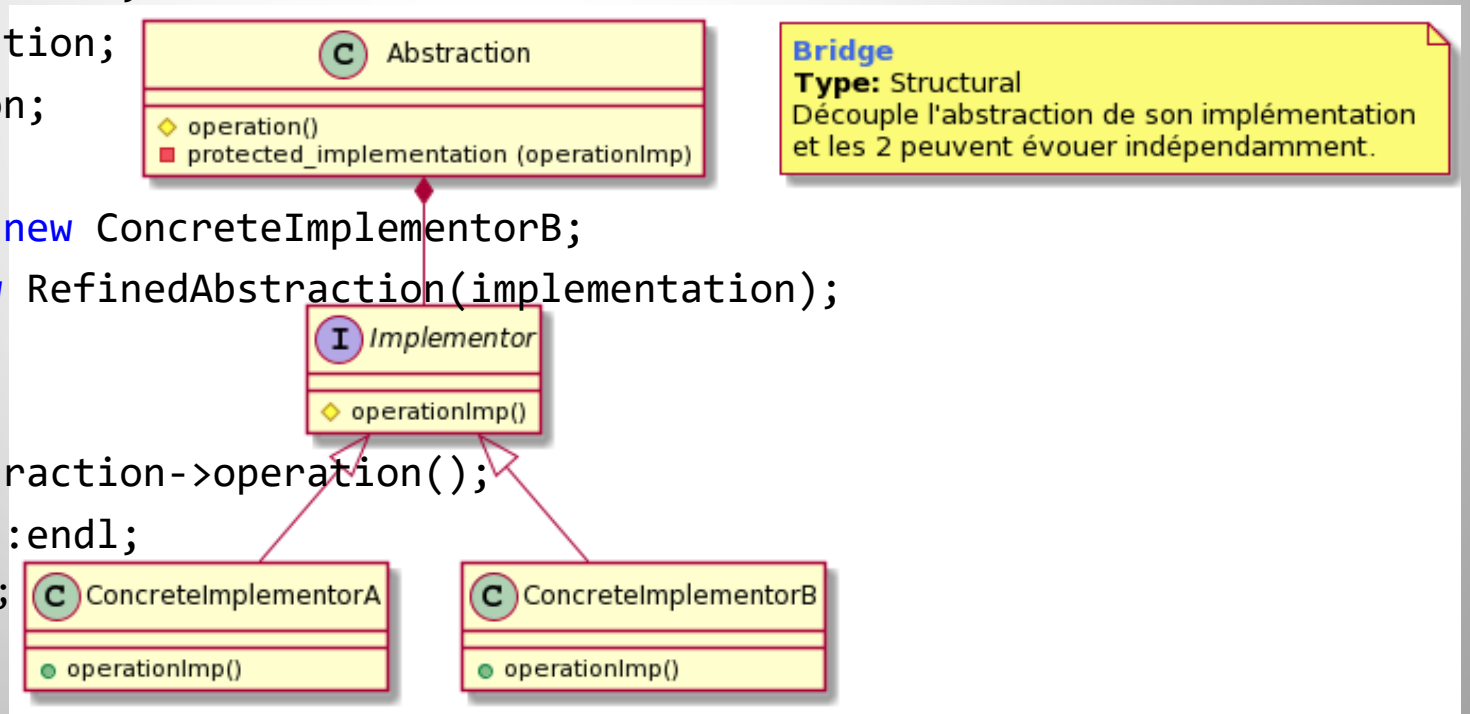
```
int main(int argc, char* argv[]) {
  Singleton *singleton = Singleton::Instance();
  singleton->checkSingleton();
  //we create a new singleton2 but...
  Singleton *singleton2 = Singleton::Instance();
  singleton2->checkSingleton();
}
```



# Structural / Bridge

- Découpler le contrat et l'implémentation
- https://godbolt.org/z/EfafGzMWG

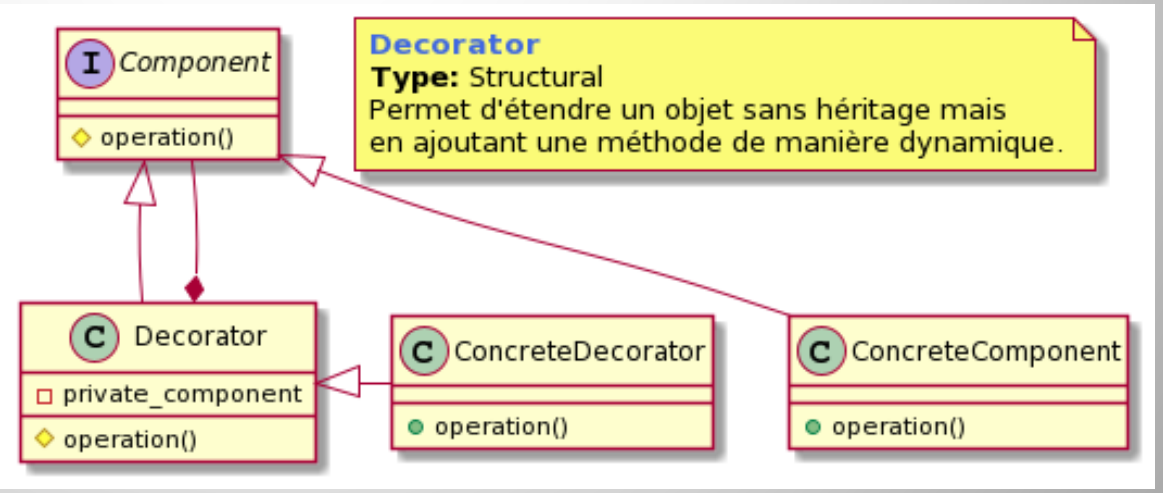
```
int main() {
  Implementor* implementation = new ConcreteImplementorA;
  Abstraction* abstraction = new Abstraction(implementation);
  std::cout << abstraction->operation();
  std::cout << std::endl;
  delete implementation;
  delete abstraction;

  implementation = new ConcreteImplementorB;
  abstraction = new RefinedAbstraction(implementation);



  std::cout << abstraction->operation();
  std::cout << std::endl;
delete implementation;
delete abstraction;
}
```



# Structural / Decorator

- Ajouter des responsabilités dynamiquement
- https://godbolt.org/z/9f8j3xM6r

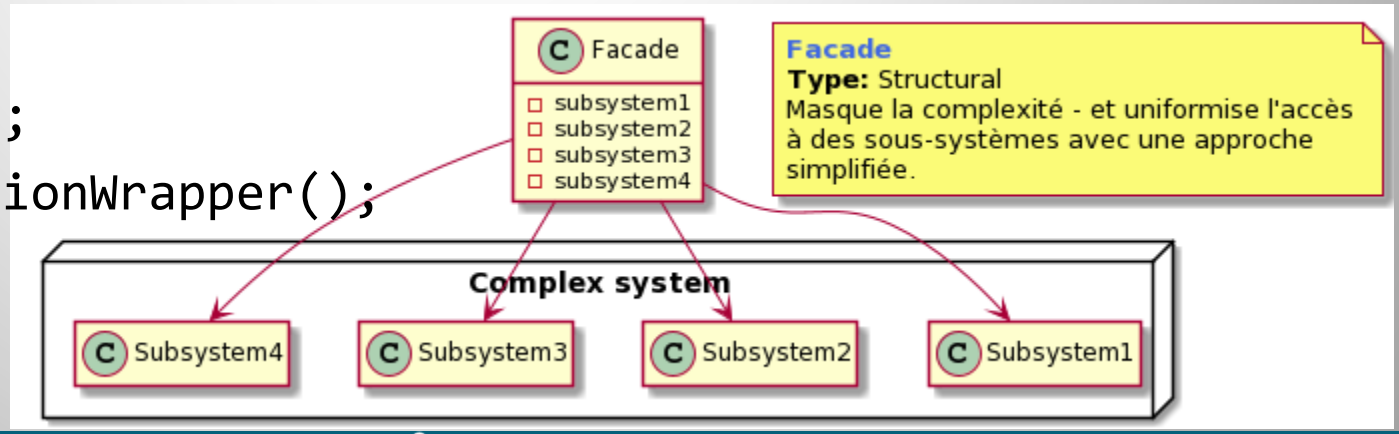
```
int main() {
  Component *cc = new ConcreteComponent();
  ConcreteDecoratorA *cda = new ConcreteDecoratorA(cc);
  ConcreteDecoratorB *cdb = new ConcreteDecoratorB(cc);

  cda->operation();
  cdb->operation();

}
```



# Structural / Facade

- Interface de sous-système simplifiée
- https://godbolt.org/z/oWTWTaPc3
- `private:`

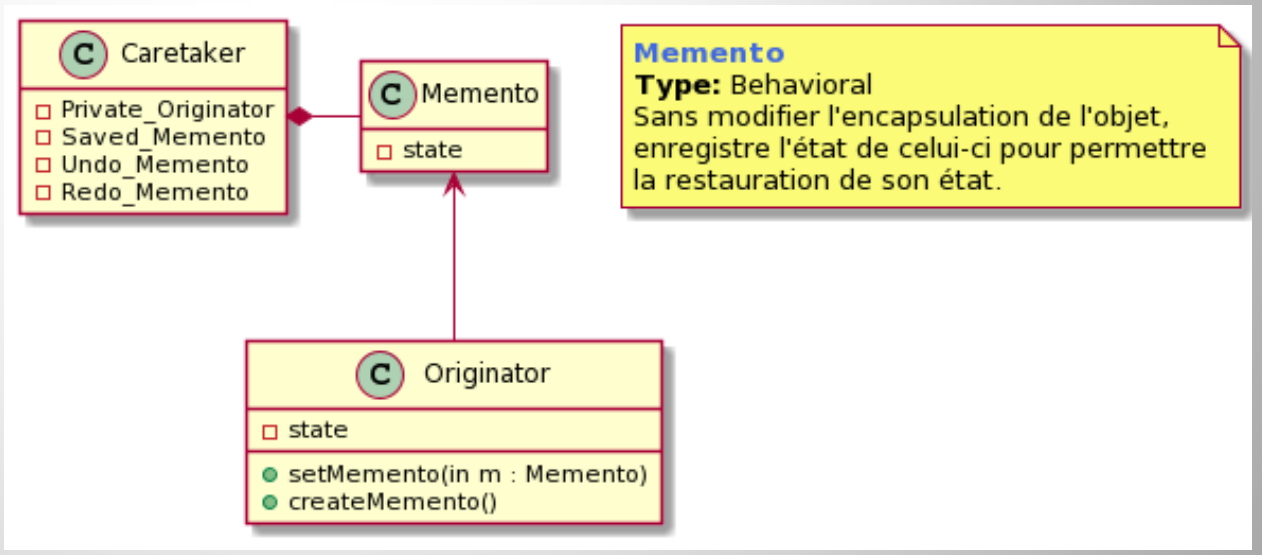
```
  SubSystem1 *subsystem1;
  SubSystem2 *subsystem2;
  SubSystem3 *subsystem3;
  SubSystem4 *subsystem4;
};
int main() {
  Facade facade;
  facade.operationWrapper();
}
```



# Behavioral / Memento

- Enregistrer l'état d'un objet sans briser l'encapsulation
- https://godbolt.org/z/v7qW3TWqb

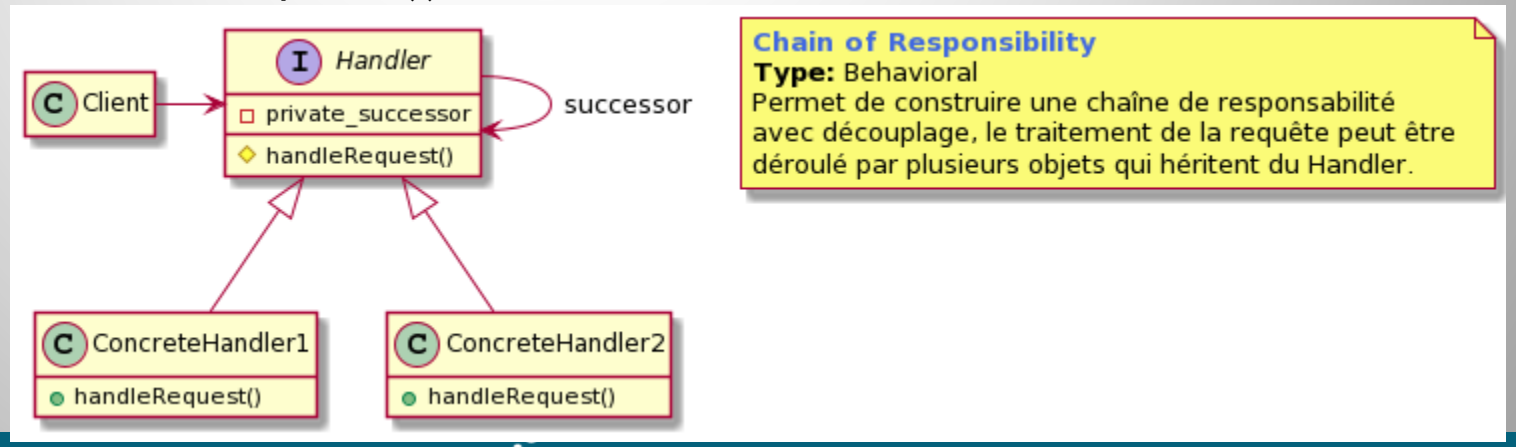
```
int main() {
    Originator *originator = new Originator();
    CareTaker *careTaker = new CareTaker( originator);

    originator->setState(0);
    careTaker->save();
    originator->setState(1);
    careTaker->save();
    originator->setState(2);
    careTaker->save();
    careTaker->printSavedStates();
    careTaker->undo();
    careTaker->printSavedStates();
    careTaker->redo();
    careTaker->printSavedStates();
    careTaker->undo();
    careTaker->undo();
    careTaker->undo();
    careTaker->printSavedStates();
}
```



# Behavioral / Chain of Responsability

- Découpler le client et le fournisseur
- https://godbolt.org/z/KMzn87ThM

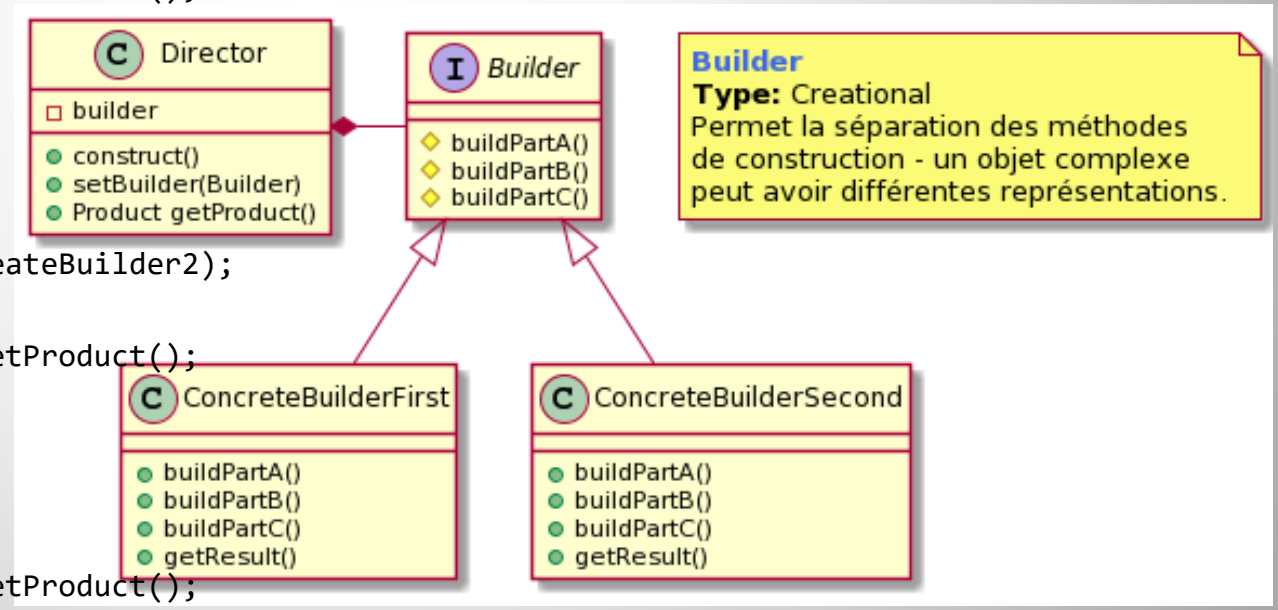
```
int main() {
    Client* client = new Client();
    client->h = new ConcreteHandler1();
    Handler* h2 = new ConcreteHandler2();
    client->h->handleRequest();
    client->h->setSuccessor(h2);
    client->h->handleRequest();
    //h2->setSuccessor(client->h);
    client->h->handleRequest();
}
```



# Creational / Builder

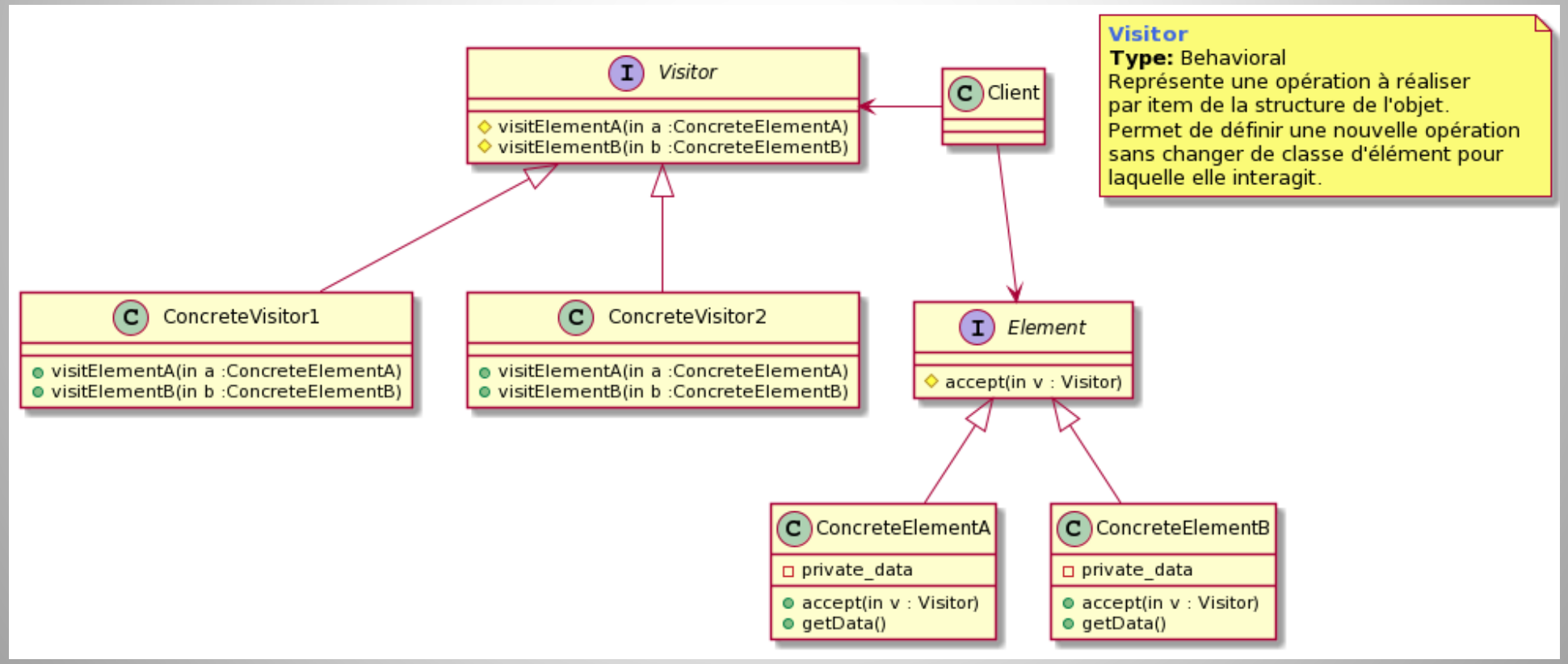
- L'algorithme de création est indépendant de la structure
- https://godbolt.org/z/zh8vjWj44

```
int main(int argc, char* argv[]) {
  Director director;
  director.setBuilder(new ConcreateBuilder1);
  director.construct();
  Product product1 = director.getProduct();
  product1.checkProduct();

  director.setBuilder(new ConcreateBuilder2);
  director.construct();
  Product product2 = director.getProduct();
  product2.checkProduct();

  director.setBuilder(0);
  director.construct();
  Product product3 = director.getProduct();
  product3.checkProduct();
}
```



# Behavioral / Visitor

- les opérations élémentaires sont appliquées à un objet sans modifier sa classe
- https://godbolt.org/z/6zsxsaPoq



# Behavioral / Visitor

```
int main() {
    ConcreteVisitor1 visitor1;
    ConcreteVisitor2 visitor2;
    ConcreteElementA elementA("String ElementA ConcreteElementA");
    elementA.accept(visitor1);
    elementA.accept(visitor2);
    ConcreteElementB elementB("String ElementB ConcreteElementB");
    elementB.accept(visitor1);
    elementB.accept(visitor2);
}
```

# Les 23 Patterns

- Il nous reste à voir 9 patterns
- Existe-t-il le pattern absolu?

- ➔La POO oriente naturellement à concentrer les propriétés.
- ➔la vision modulaire est un choix.

# Les patrons / patterns

Les 23 patterns :

| | Creational | Structural | Behavioral |
|---|---|---|---|
| Class | Factory Method | Adapter (class) | Interpreter |
| | | | Template Method |
| Object | Abstract Factory | Adapter (object) | 13 Chain of Responsibility |
| | 14 Builder | 9 Bridge | Command |
| | Prototype | Composite | Iterator |
| | 8 Singleton | 10 Decorator | Mediator |
| | | 11 Facade | 12 Memento |
| | | Flyweight | Observer |
| | | Proxy | State |
| | | | Strategy |
| | | | 15 Visitor |

# Les patrons / patterns

Les 23 patterns :

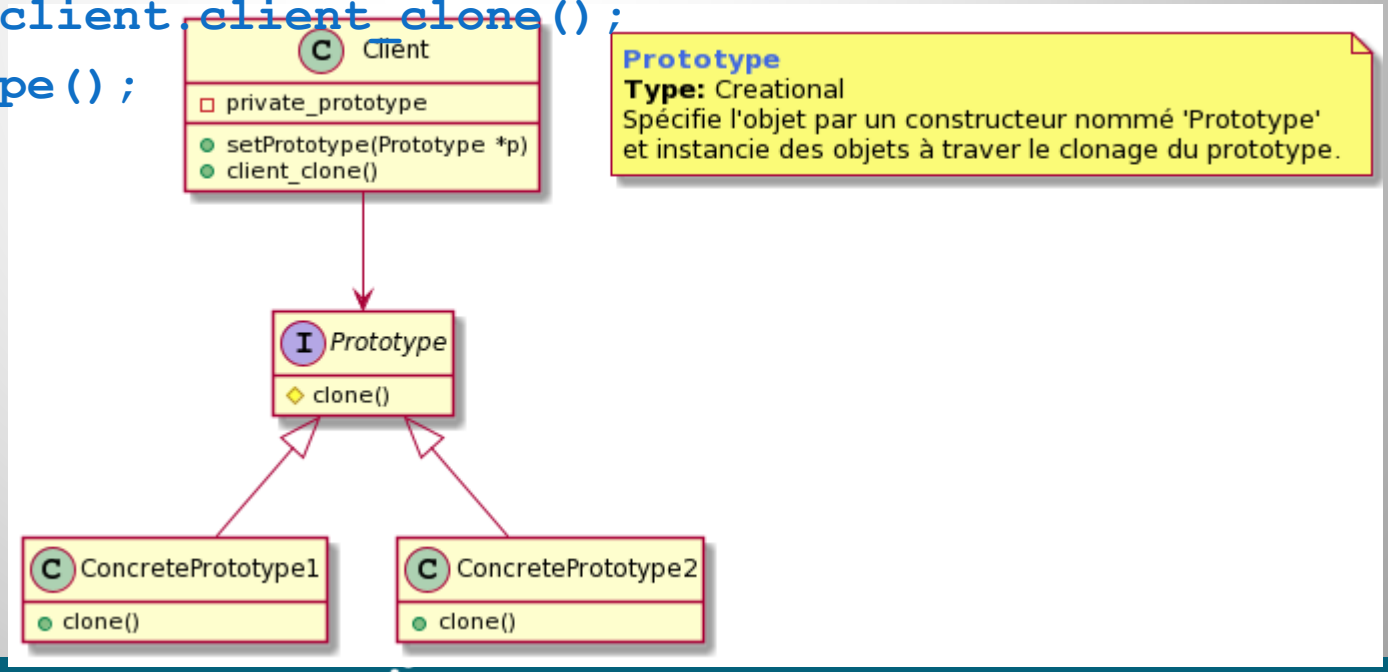
| | Creational | Structural | Behavioral |
|---|---|---|---|
| Class | Factory Method | 17 Adapter (class) | 21 Interpreter |
| | | | Template Method |
| Object | Abstract Factory | 17 Adapter (object) | 13 Chain of Responsibility |
| | 14 Builder | Bridge | 22 Command |
| | 16 Prototype | 18 Composite | Iterator |
| | Singleton | 10 Decorator | 23 Mediator |
| | | 11 Facade | 12 Memento |
| | | 19 Flyweight | Observer |
| | | 20 Proxy | State |
| | | | Strategy |
| | | | 15 Visitor |

| Typologie | Nom du Design Pattern | Ce qui doit être ajuster | verbe | composition sous jacente | mélange de classes ? |
|---|---|---|---|---|---|
| Creational | Abstract Factory | famille d'objets dépendants | | structure | non |
| | Builder | Comment créer un objet composite dont la structure du composite est indépendante | | liste | non |
| | Factory Method | Sous classe d'un objet qui est instanciée sans connaitre la classe ancêtre (connaissance retardée) | | | filtrage vtab |
| | Prototype | Classe d'objet qui est instanciée grâce à un constructeur de copie | copie=verbe | liste | non |
| | Singleton | La seule instance d'une classe | copie=o=verbe | | non |
| Structural | Adapter | accède à un objet en modifiant l'interface | | | non |
| | Bridge | Fait l'implémentation d'un objet par découplage | découplage | | non |
| | Composite | structure et composition d'un objet vue de manière uniforme | | arbre | non |
| | Decorator | Responsabilité d'un objet sans héritage - ajout dynamique | | | filtrage vtab |
| | Facade | Exposer une interface à un sous-système | | | filtrage vtab |
| | Flyweight | cout de stockage d'un objet, partage de l'état | état=verbe | liste | non |
| | Proxy | Comment un objet est accédé, son emplacement (mémoire, disque) - effet miroir | | queue | non |
| Behavioral | Chain of Responsibility | Un objet qui peut répondre à une demande avec découpplage | découplage | queue | filtrage vtab |
| | Command | quand et comment une commande peut être faite - la commande devient un objet | | structure | non |
| | Interpreter | grammaire et interprétation d'un objet | | | non |
| | Iterator | se déplacer dans une structure d'objet sans en connaitre le détail | | liste | filtrage vtab |
| | Mediator | Comment et avec quels objets sont décrites les interactions | | | non |
| | Memento | Quelles sont les informations privées qui sont stockée à part et quand? | état=verbe | état (queue=infinie) | non |
| | Observer | l'effectif des objets observés et quand s'effectue la mise à jour | | queue | non |
| | State | les états d'un objet sont des variables, avec un handler() | état=verbe | structure | filtrage vtab |
| | Strategy | un algorithme | extension=verbe | structure | filtrage vtab |
| | Template Method | les étapes/squelette d'un algorithme | | structure | non |
| | Visitor | les opérations élémentaires sont appliquées à un objet sans modifier sa classe | | liste | non |

# Creational / Prototype

- Création de nouveaux objets par copie d'un modèle
- https://godbolt.org/z/M87PrKxrh

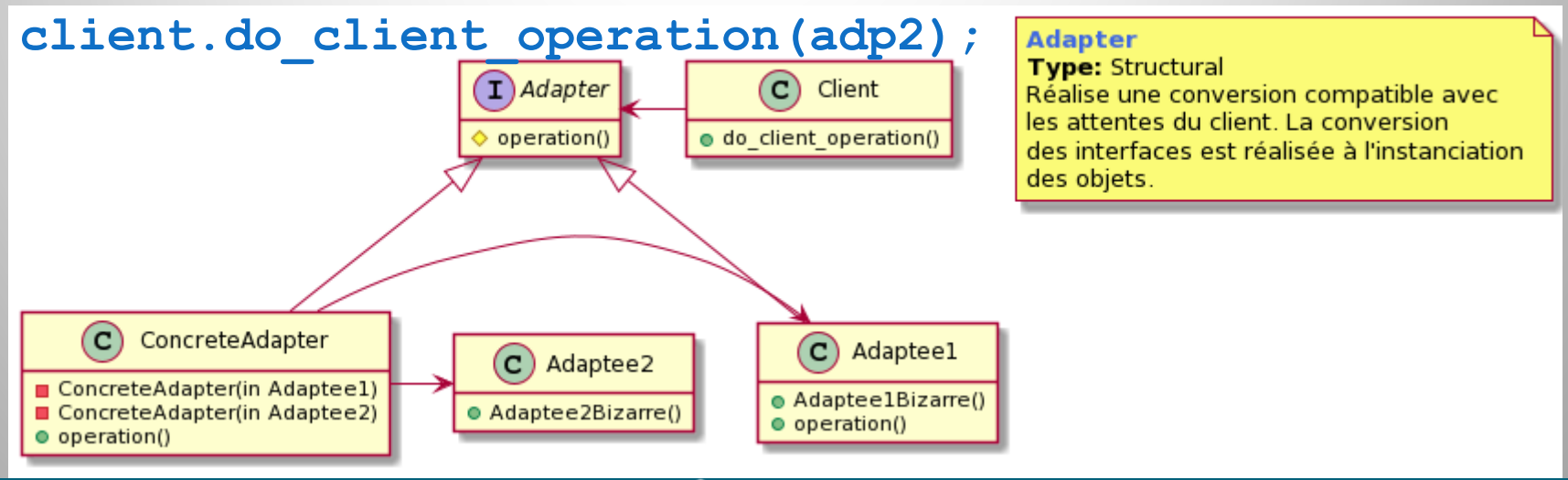
```
int main(int argc, char* argv[]) {
  Client client;
  client.setPrototype(new ConcretePrototype1);
  Prototype *p1 = client.client_clone();
  p1->checkPrototype();
  client.setPrototype(new ConcretePrototype2);
  Prototype *p2 = client.client_clone();
  p2->checkPrototype();
}
```



# Structural / Adapter

- Convertir l'interface d'une classe
- https://godbolt.org/z/xj6reoaGd

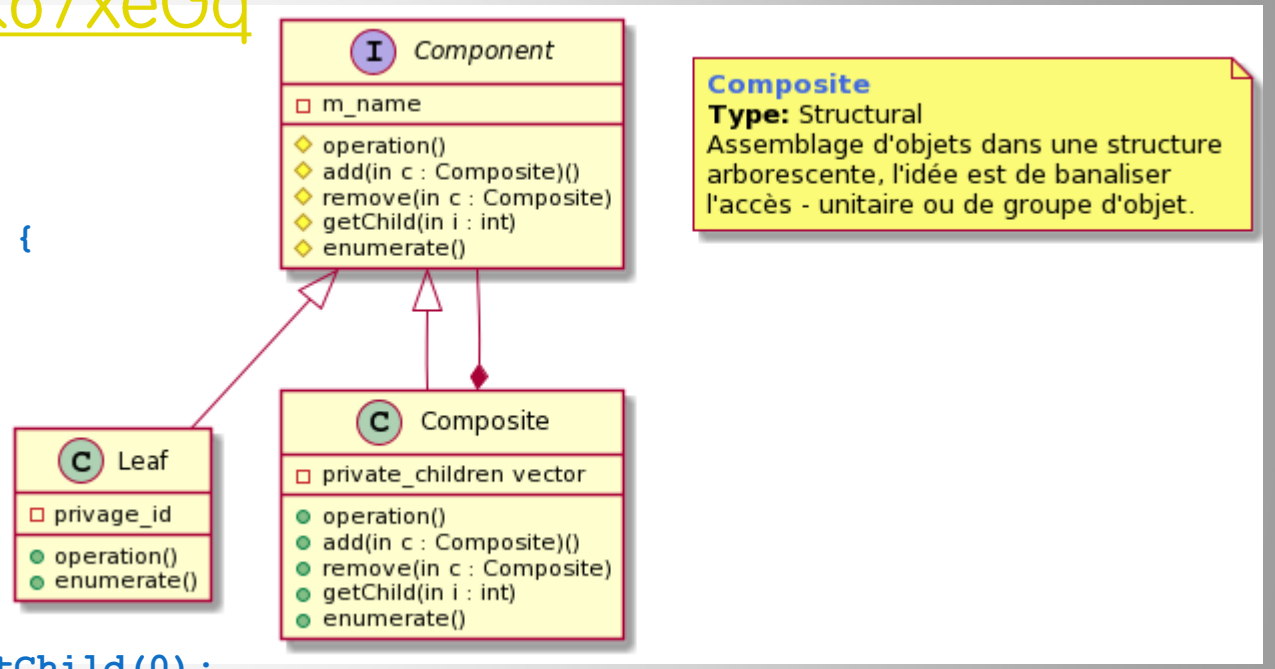
```
int main() {
    Client client;
    std::cout << "new Adaptee1" << std::endl;
    ConcreteAdapter adp1(new Adaptee1());
    client.do_client_operation(adp1);
    std::cout << "new Adaptee2" << std::endl;
    ConcreteAdapter adp2(new Adaptee2());
    client.do_client_operation(adp2);
}
```



# Structural / Composite

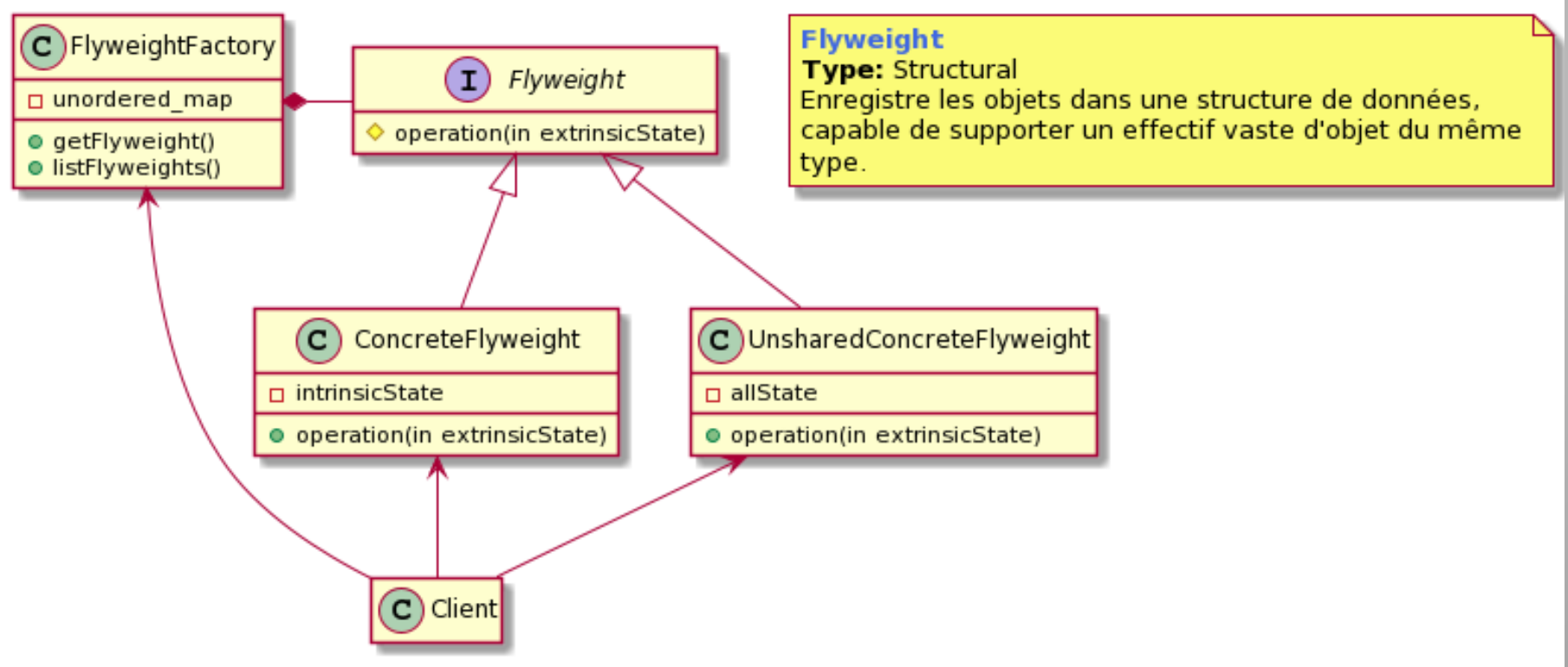
- Représentation de hiérarchies d'objets vus de manière uniforme par le client
- https://godbolt.org/z/ooK67xeGq

```
int main() {
  Composite composite;
  composite.group("principal");
  for (unsigned int i = 0; i < 3; ++i) {
    composite.add(new Leaf(i));
  }
  Composite composite2;
  composite2.group("secondaire");
  composite.add(&composite2);
  composite.remove(0);
  composite.operation();
  Component *component1 = composite.getChild(0);
  component1->operation();
  Component *component2 = composite.getChild(3);
  component2->operation();
  composite.enumerate();
}
```



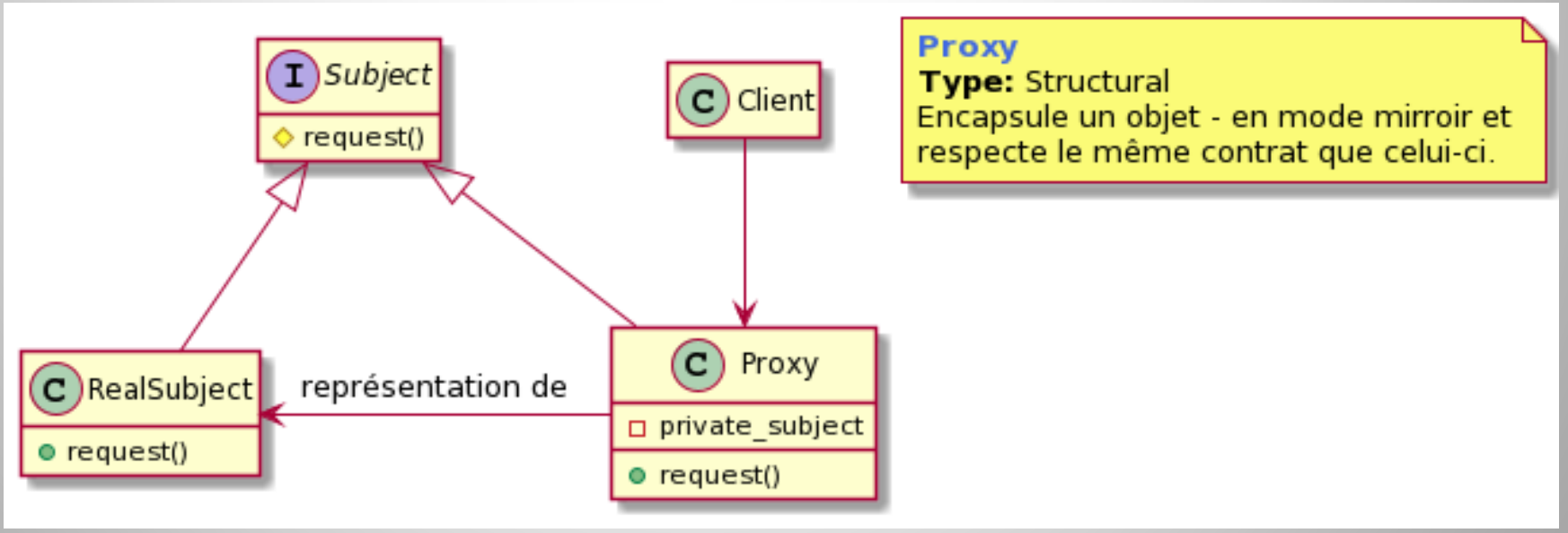
# Structural / Flyweight

- Partager l'état extrinsèque - factorisation
- https://godbolt.org/z/xEfWhT7zc



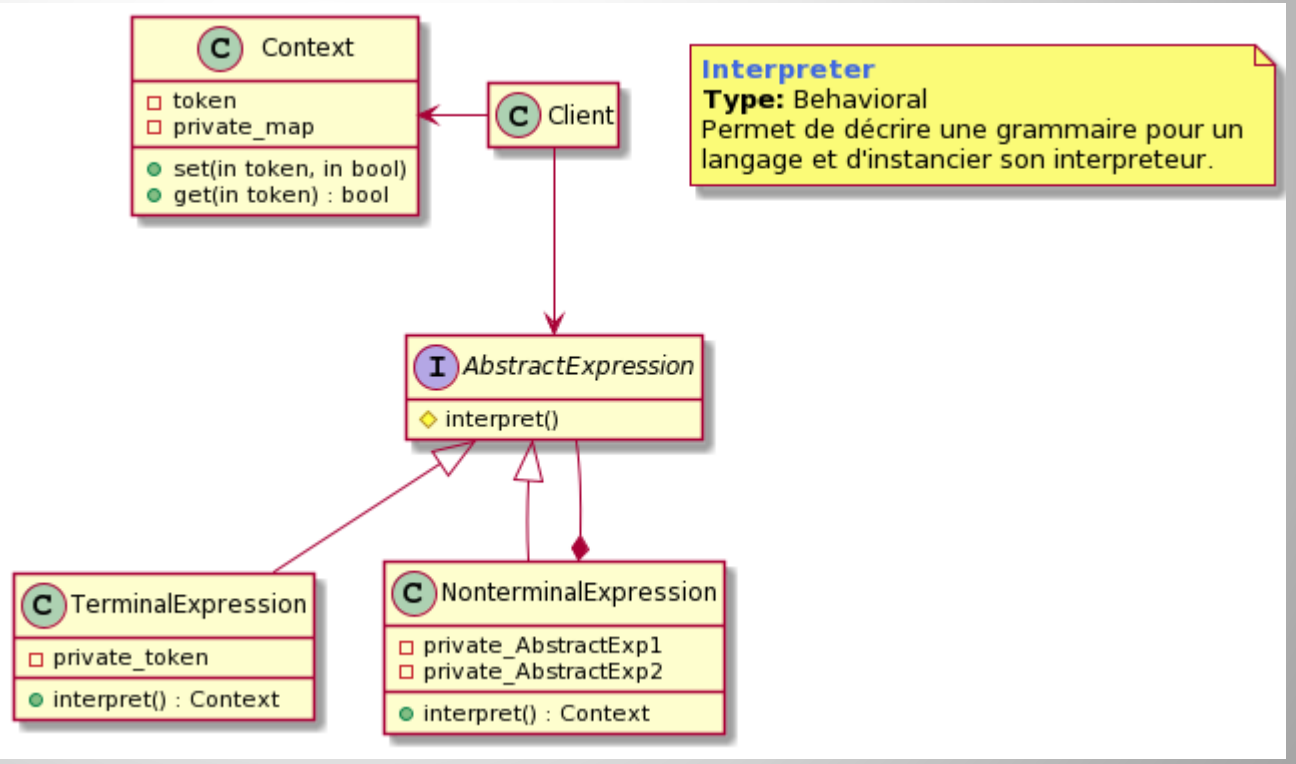
# Structural / Proxy

- Objet miroir d'un autre objet plus lointain
- https://godbolt.org/z/cTbP35sd4



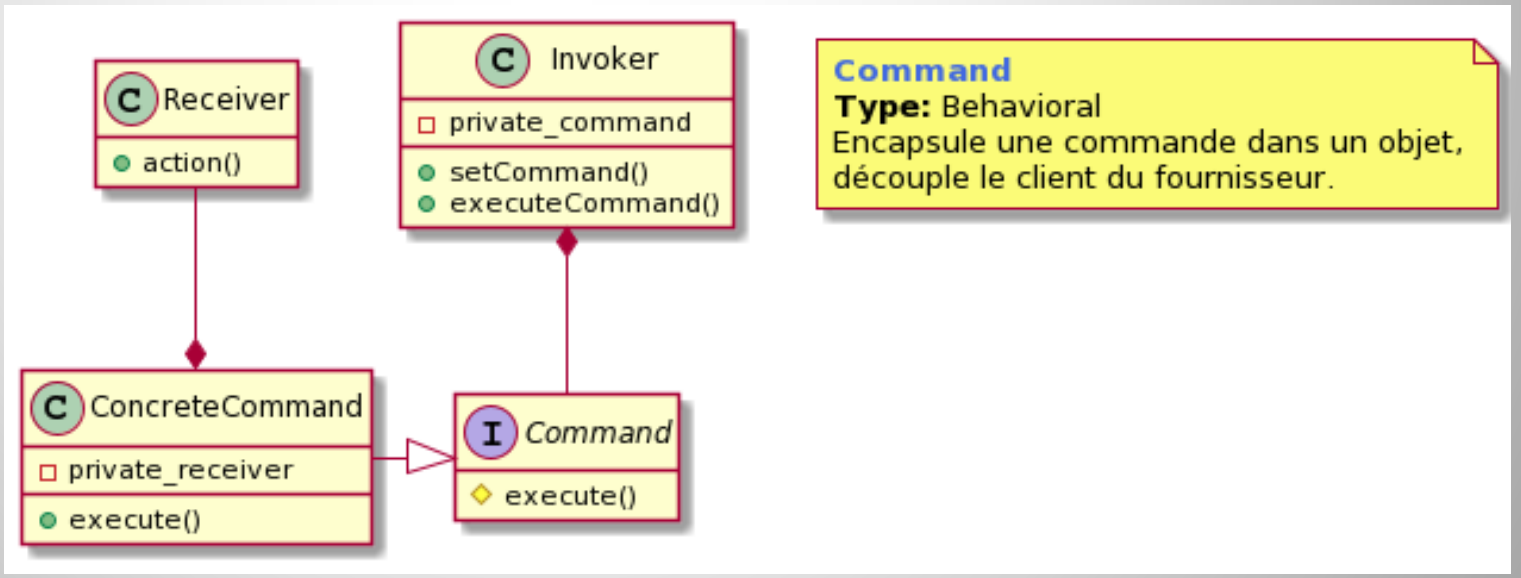
# Behavorial / Interpreter

- Décrit un interpreteur – utile pour un moteur d'état
- https://godbolt.org/z/oes9M9bbT



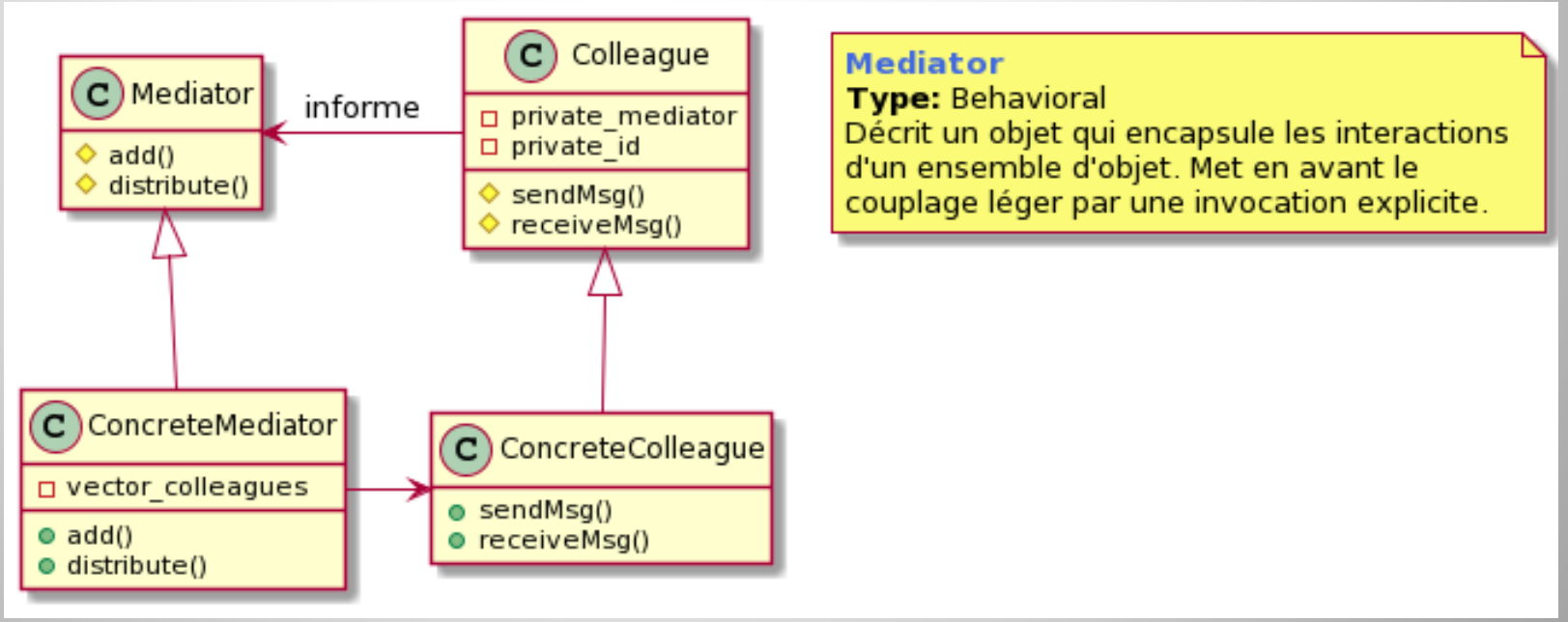
# Behavorial / Command

- Range une commande dans un objet, découple le client du fournisseur
- https://godbolt.org/z/GMcKodP9G



# Behavorial / Mediator

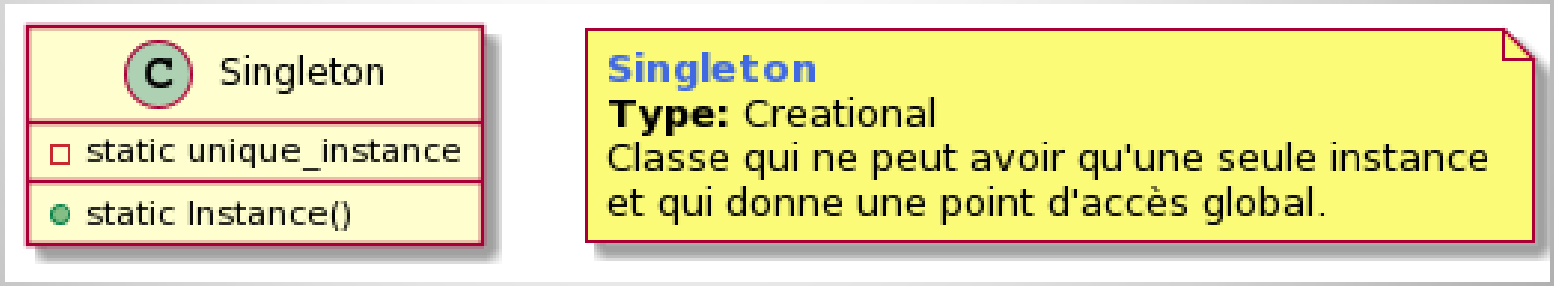
- Permet le couplage léger
- https://godbolt.org/z/o5Eneo1sG



# Creational / Singleton

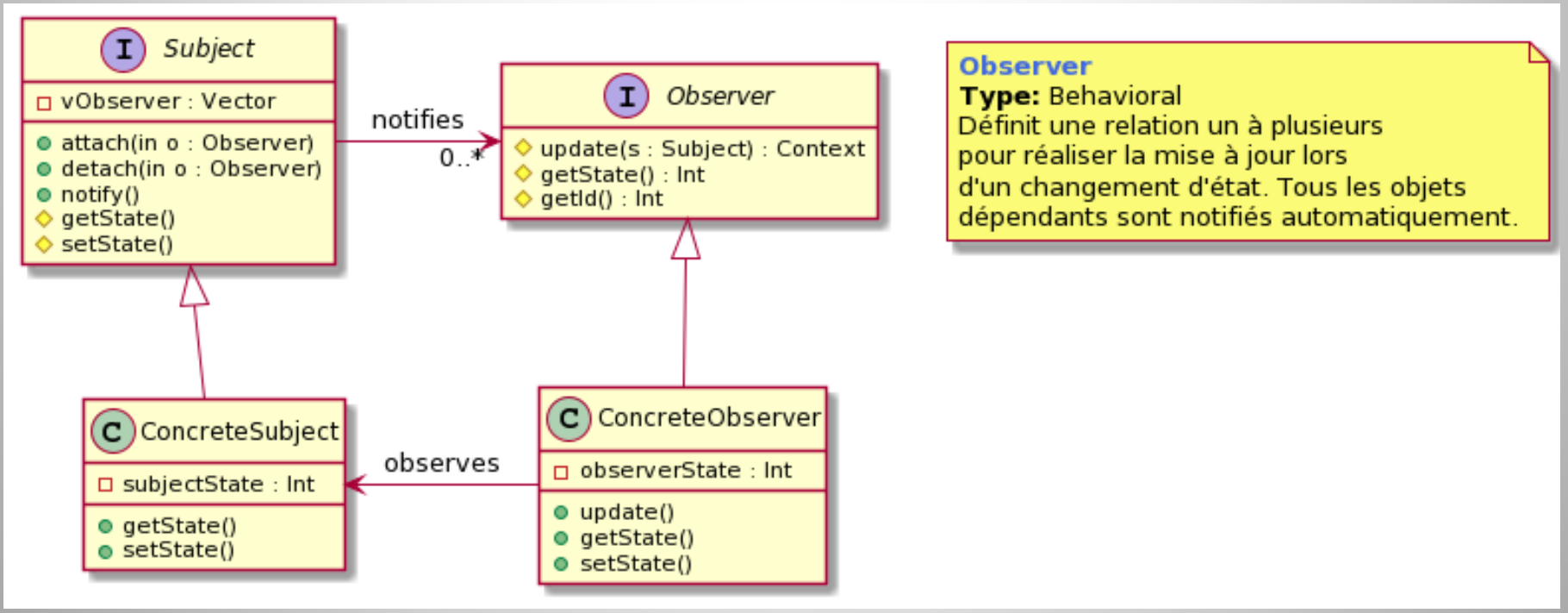
- Classe qui ne donne qu'une seule instance
- https://godbolt.org/z/5916Mc6Ez

```
int main(int argc, char* argv[]) {
  Singleton *singleton = Singleton::Instance();
  singleton->checkSingleton();
  //we create a new singleton2 but...
  Singleton *singleton2 = Singleton::Instance();
  singleton2->checkSingleton();
}
```



# Behavioral / Observer

- https://godbolt.org/z/qMKrsoY7M
- Dépendance Publish-Subscribe, idée de queue

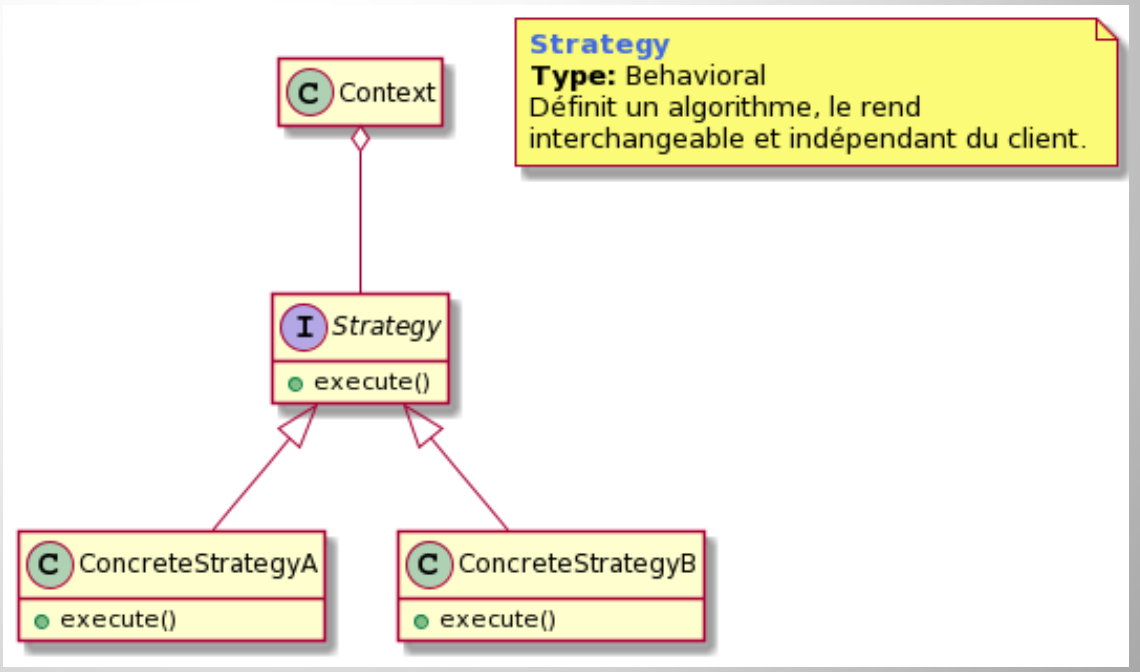


# Behavioral / Strategy

- https://godbolt.org/z/zbaK3Wr5T

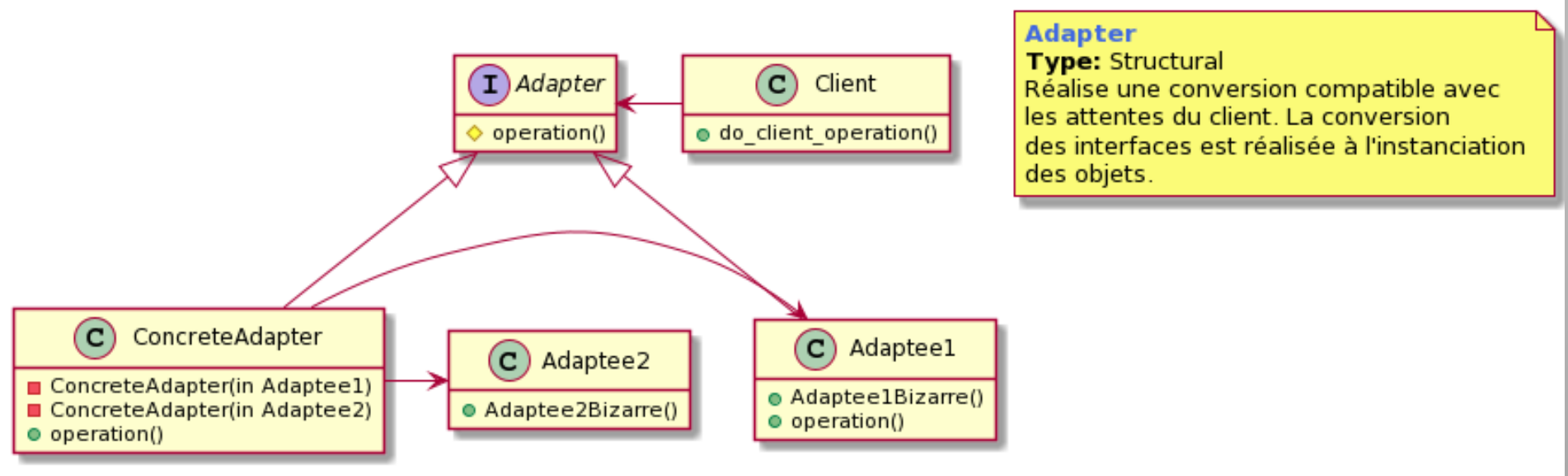
```
int main() {
    Context context(new ConcreteStrategyB());
    context.execute();
}
```

- ➔algorithme



# Structural / Adapter

- Convertir l'interface d'une classe
- https://godbolt.org/z/xj6reoaGd



# Structural / Facade

- Interface de sous-système simplifiée
- https://godbolt.org/z/oWTWTaPc3
- `private:`

```
  SubSystem1 *subsystem1;
  SubSystem2 *subsystem2;
  SubSystem3 *subsystem3;
  SubSystem4 *subsystem4;
};
int main() {
  Facade facade;
  facade.operationWrapper();
}
```