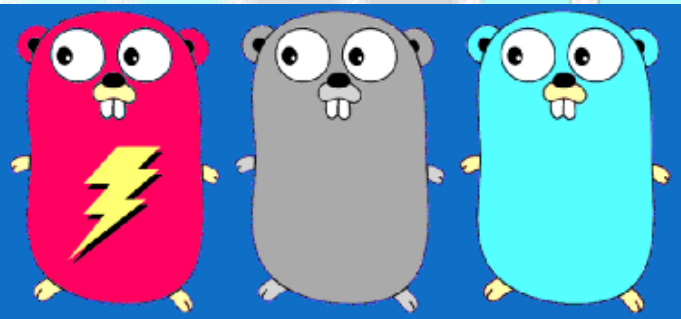
Génie Logiciel pour le Calcul Scientifique

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY



#6

16/01/2025

jean-michel.batto@cea.fr

cea

https://gogs.eldarsoft.com/M2_IHPS

Activité TD langage XMP (suite du cours5)

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

TD XMP

INTERACTIF

- Vous clonez https://gogs.eldarsoft.com/jmbatto/GLCS-CM6-TDXMP dans votre image docker **mpihead**
- **Dans ce TD seuls les images mpihead et mpinode1 sont utilisées**
- Il s'agit d'un dépôt « starter » pour travailler sur une implémentation de calcul d'histogramme en utilisant MPI et XMP.
  - Ce programme est très simple : en entrée un fichier texte avec des valeurs float comprises entre 0.0 et 20.0
  - En sortie un fichier histogramme, au format texte

TD XMP

INTERACTIF

- Position du problème :
  - Construire un histogramme à partir d'un fichier de valeurs float,
  - L'histogramme décrit l'effectif de 20 entiers associés aux valeurs float
  - Le fichier au format texte est :

```
100
1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0
2.0,0.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0
3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0
4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0
5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0
6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0
7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0,16.0
8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0
9.0,10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0
10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0
```

Effectif des valeurs

Les valeurs

Principe de l'histogramme

```
//Le résultat est stocké dans une variable globale
//Pour calculer un histogramme il nous faut un tableau de
//valeur (*data) et un effectif (rows)
void calcule_histo(float *data, int rows) {
  for (int i = 0; i < rows; i++) {
    int j = (int)data[i];
    if ((j >= 0) && (j < 20)) {
      data_out[j]++;
    }
  }
}
```

Principe XMP

- Quelques instructions pour XMP

```
#pragma xmp nodes p(1)
#pragma xmp template t[:]
#pragma xmp distribute t(block) onto p
```

Permet de tester sur 1 noeud

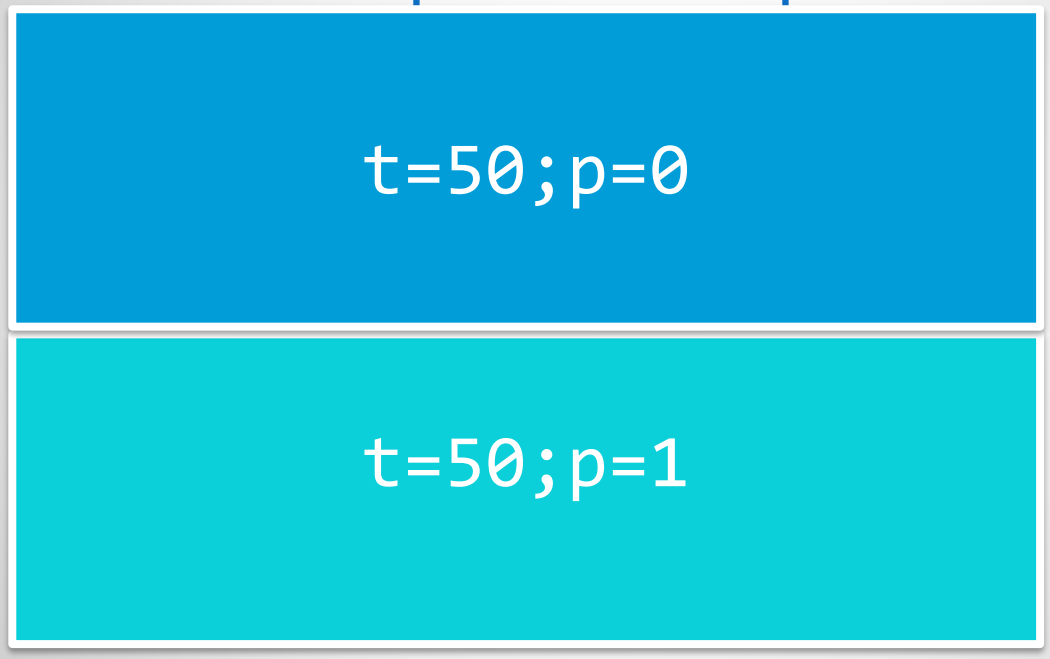
Retarde l'allocation des templates

Distribue les templates sur les noeuds

```
float *data_in;
#pragma xmp align data_in[i] with t(i)
#pragma xmp shadow data_in[*]
```

Positionne la variable data_in sur les templates
➔objet de l'optimisation XMP

- #pragma xmp nodes p(2) ➔ notation pour dire qu'il y a 2 noeuds ➔ ils découpent l'espace des 100 templates



- **mpirun -n 2**

```
xmp_init_mpi(&argc, &argv);
```

- ➔initialise le contexte MPI

```
xmp_finalize_mpi();
```

- ➔finalise le contexte MPI

```
//aspect dynamique à fixer dans le code C
```

- #pragma xmp template_fix t[rows]
- Il s’agit de définir la taille des templates et d’allouer la mémoire
- data_in = xmp_malloc(xmp_desc_of(data_in), rows);

Optimisation XMP

```
void calcule_histo(float *data, int rows) {
#pragma xmp loop on t(i)
  for (int i = 0; i < rows; i++) {
    int j = (int)data[i];
    if ((j >= 0) && (j < 20)) {
      data_out[j]++;
    }
  }
}
```

Si on enlève ce pragma, plus d'intérêt de XMP

INTERACTIF

- Vous allez tester sur 1 nœud le mécanisme
- Vous allez l'étendre à 2 nœuds
- ➔ Quel est le constat sur les résultats ? (par rapport à 1 nœud)

- Vous allez ajouter une réduction pour obtenir un résultat conforme au cas sur 1 noeud

INTERACTIF

- Problème il faut faire une réduction

- `#pragma xmp reduction(+ : tmp)` ➔ effectue la réduction par rapport aux noeuds

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 :

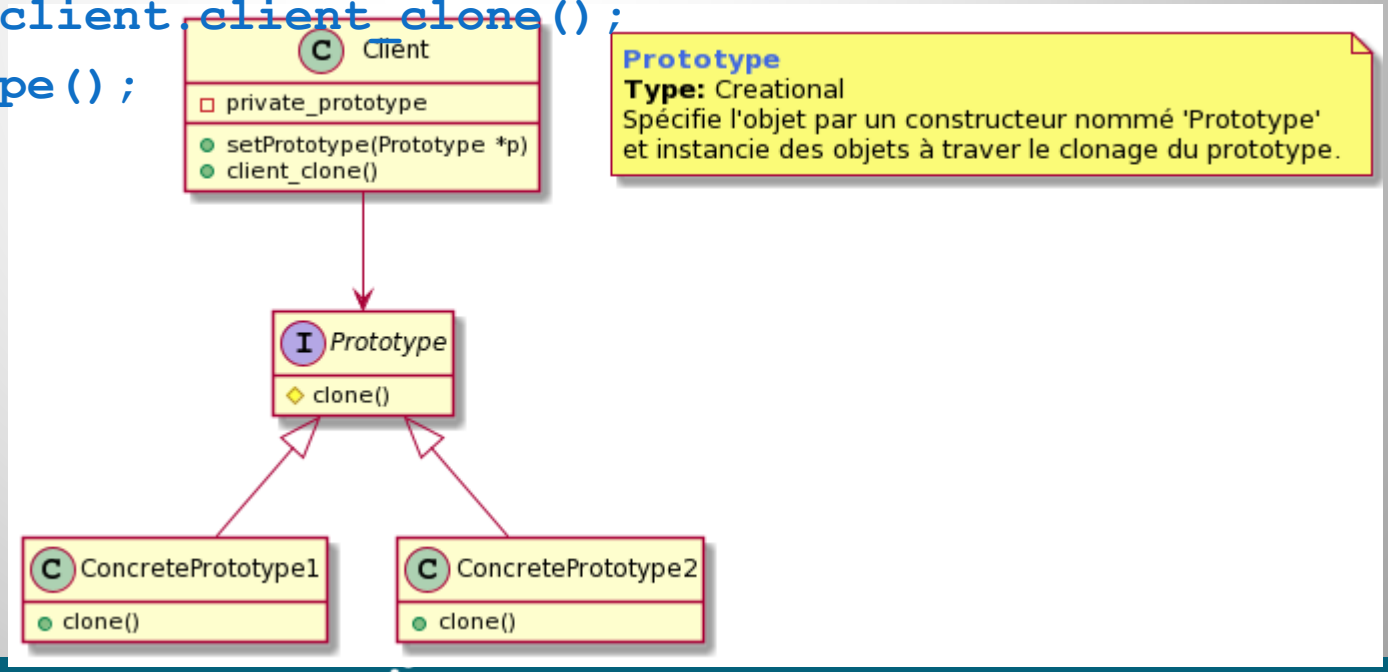
	Creational	Structural	Behavioral
Class	Factory Method	17 Adapter (class)	21 Interpreter
			Template Method
Object	Abstract Factory	17 Adapter (object)	13 Chain of Responsibility
	14 Builder	Bridge	22 Command
	16 Prototype	18 Composite	Iterator
	Singleton	10 Decorator	23 Mediator
		11 Facade	12 Memento
		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

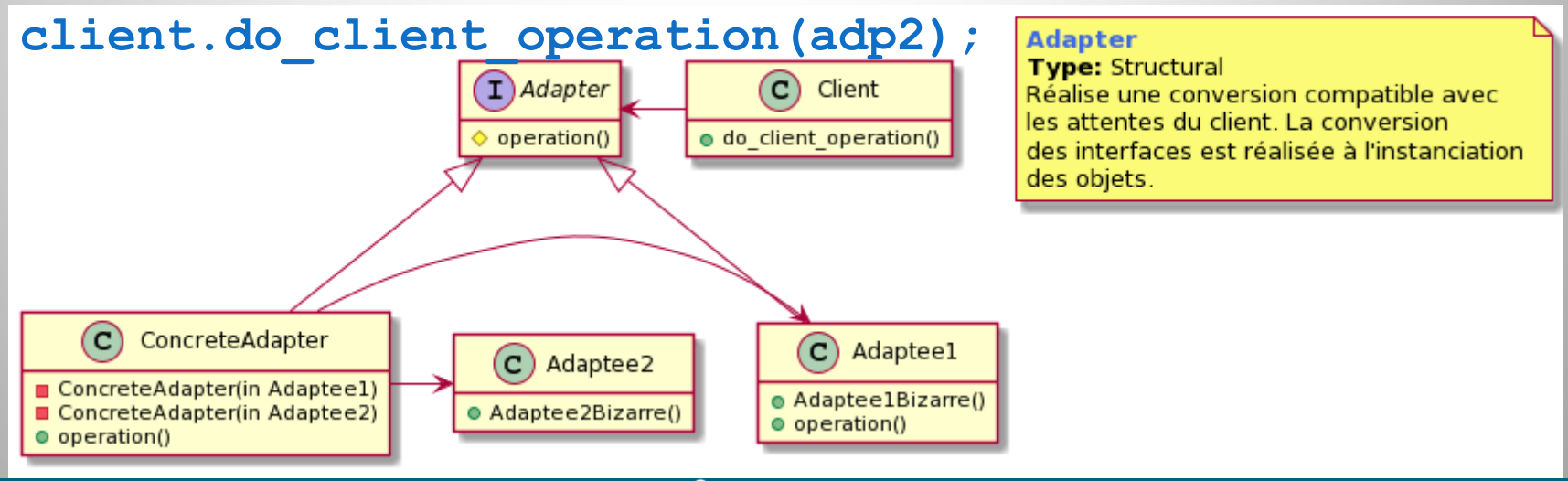
```
int main(int argc, char* argv[]) {
  Client client;
  client.setPrototype(new ConcretePrototype1);
  Prototype *p1 = client.client_clone();
  p1->checkPrototype();
  client.setPrototype(new ConcretePrototype2);
  Prototype *p2 = client.client_clone();
  p2->checkPrototype();
}
```



Structural / Adapter

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

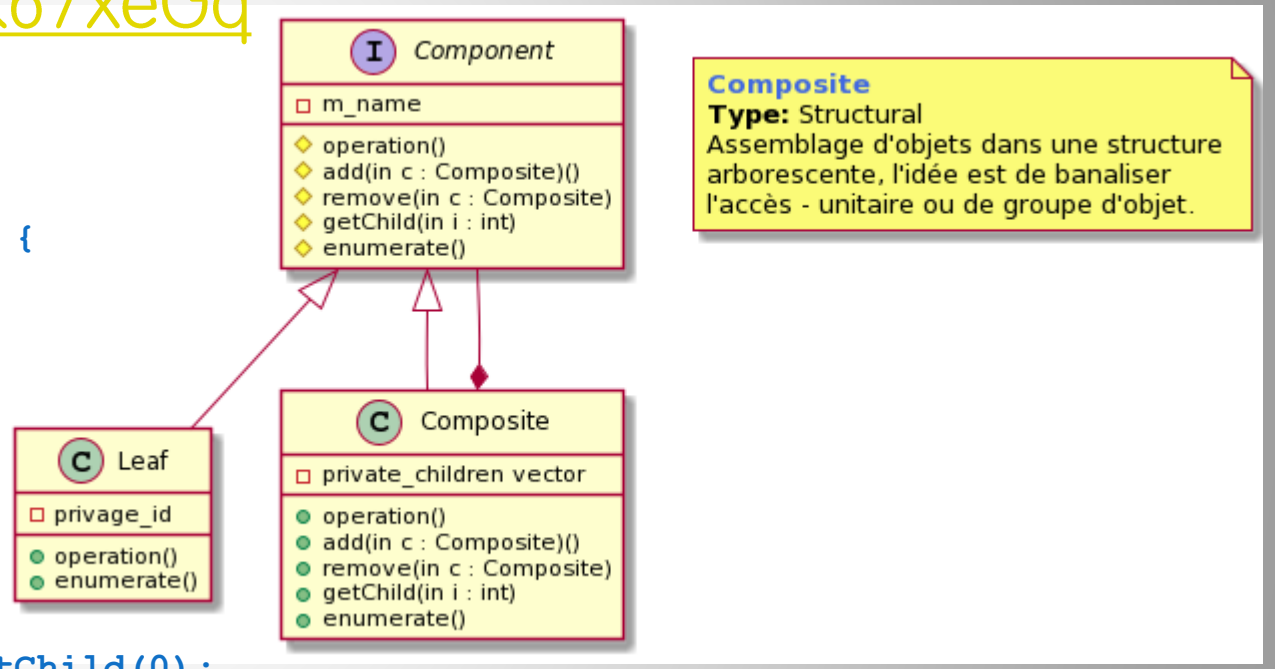
```
int main() {
    Client client;
    std::cout << "new Adaptee1" << std::endl;
    ConcreteAdapter adp1(new Adaptee1());
    client.do_client_operation(adp1);
    std::cout << "new Adaptee2" << std::endl;
    ConcreteAdapter adp2(new Adaptee2());
    client.do_client_operation(adp2);
}
```



Structural / Composite

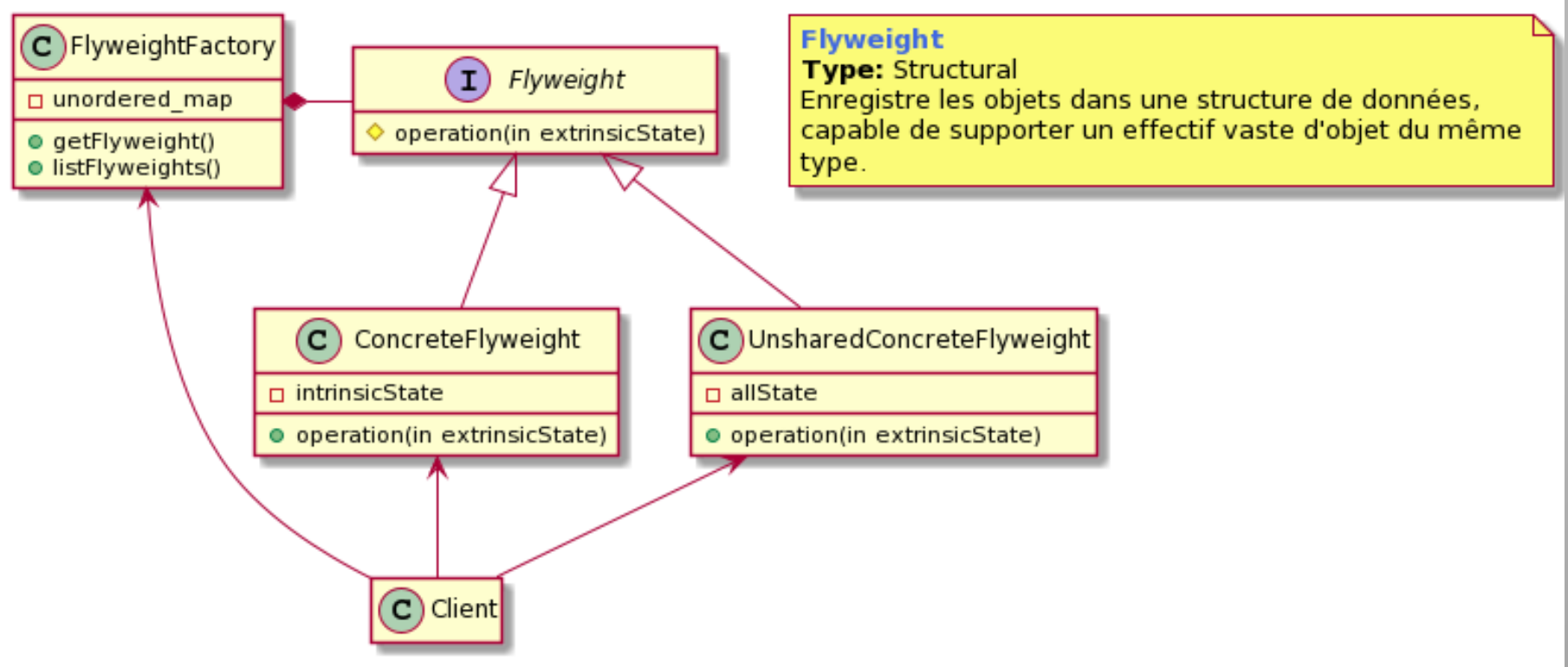
- Représentation de hiérarchies d'objets vus de manière uniforme par le client
- https://godbolt.org/z/ooK67xeGq

```
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();
}
```



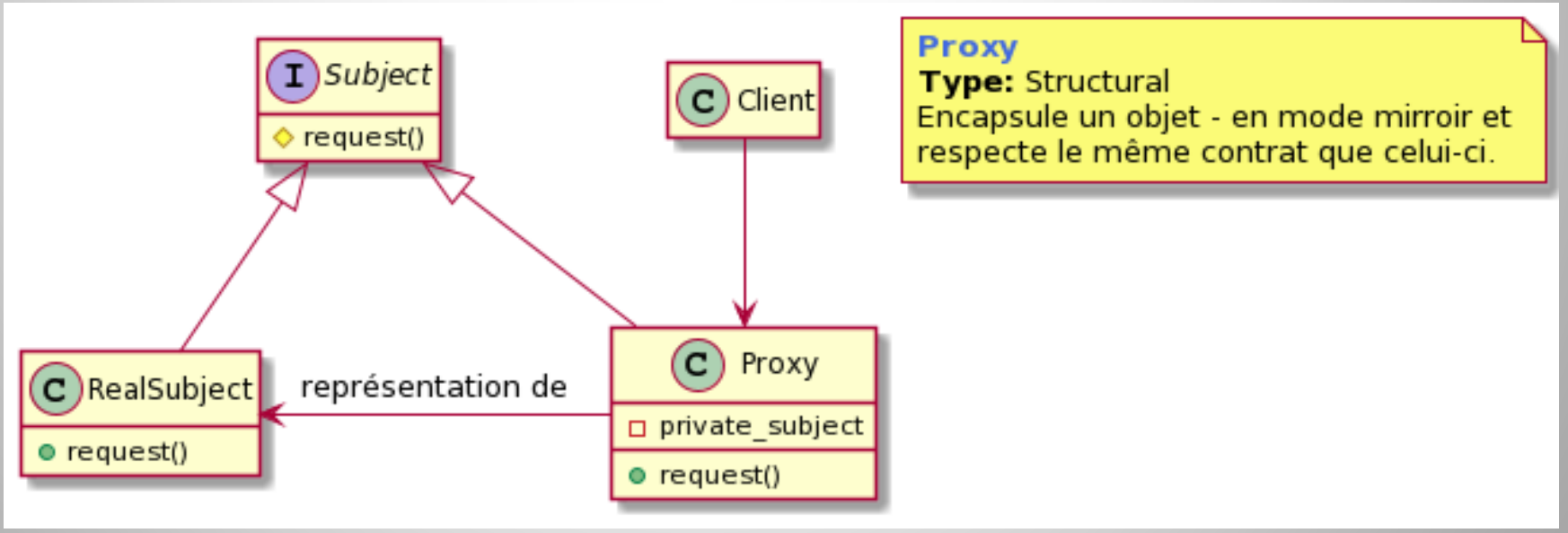
Structural / Flyweight

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



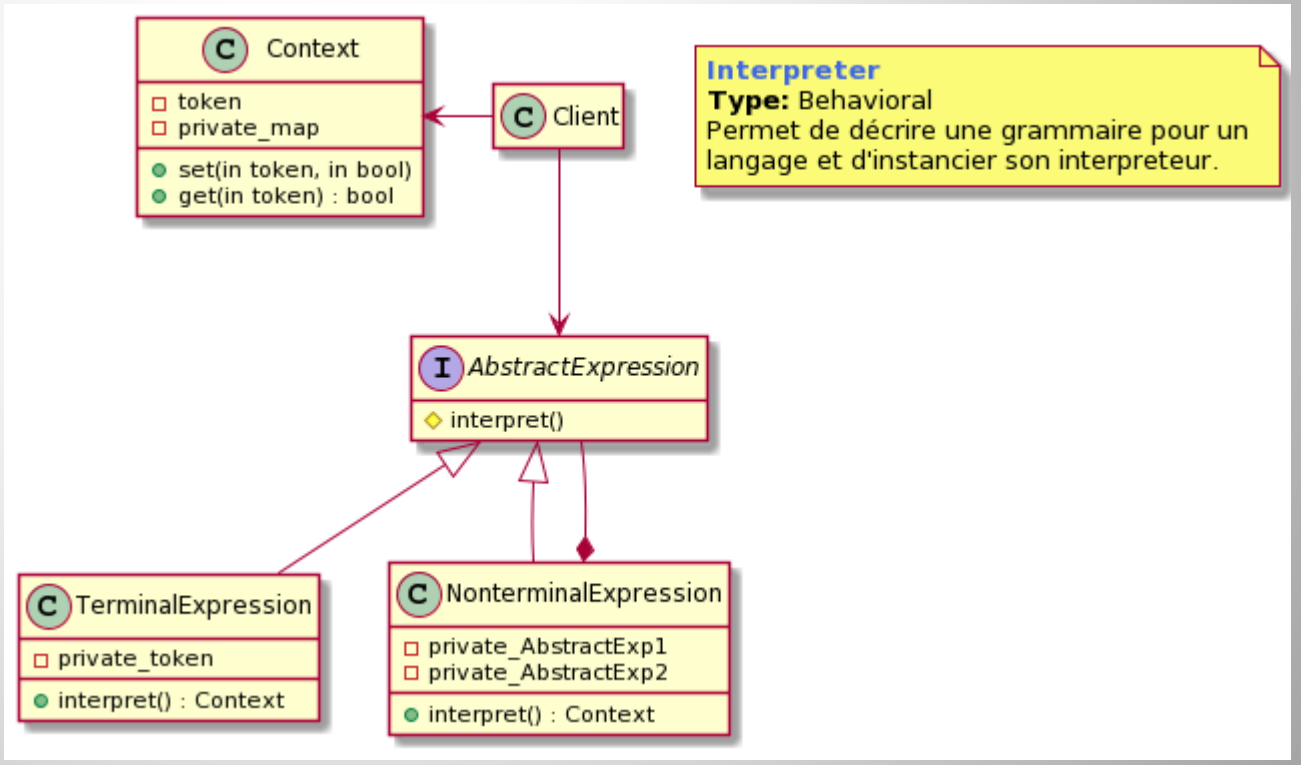
Structural / Proxy

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



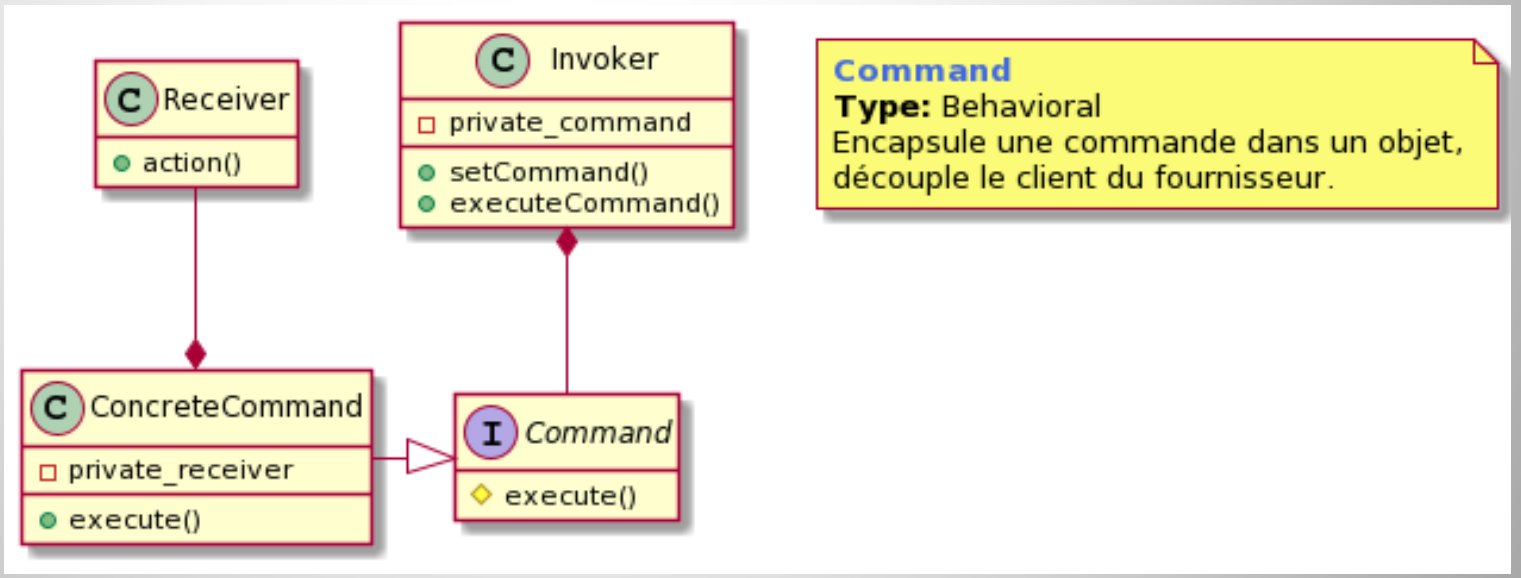
Behavorial / Interpreter

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



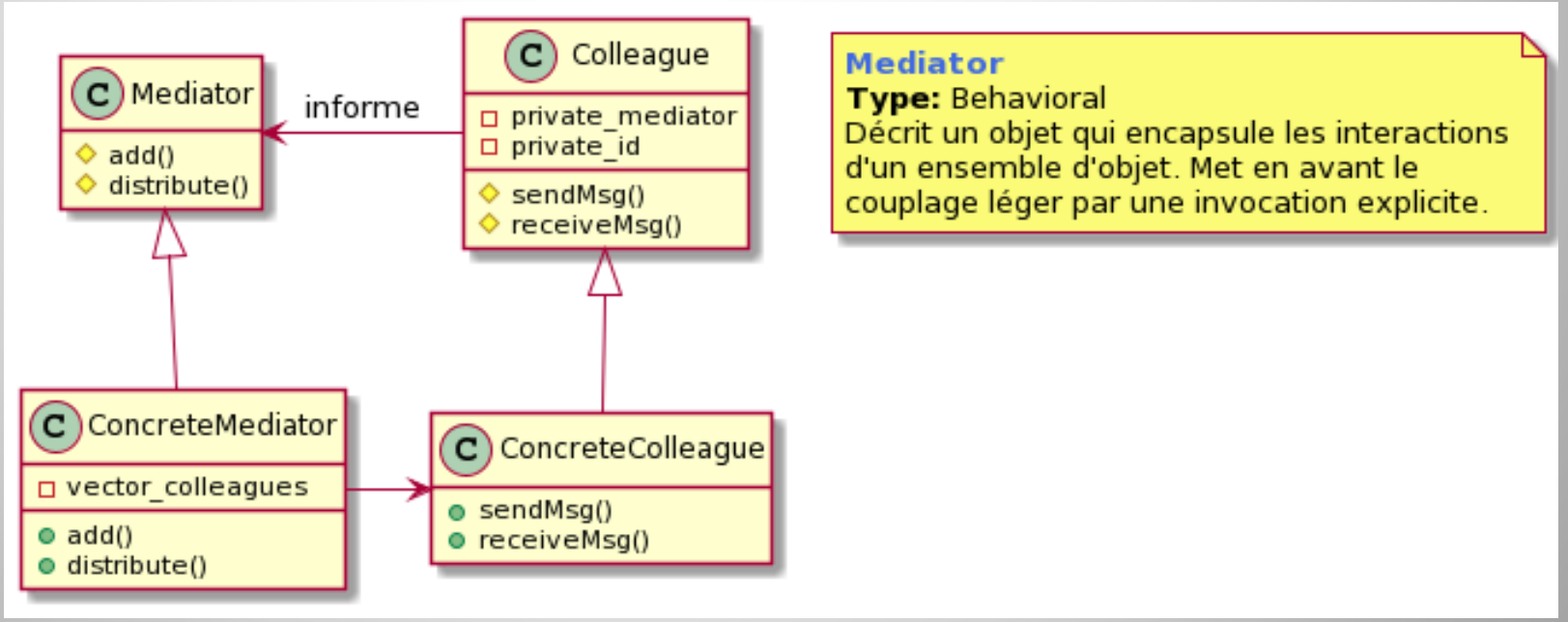
Behavorial / Command

- 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



Comment appliquer les 23 patterns?

Biblio➔Exploring Game Architecture Best-Practices 2011.pdf

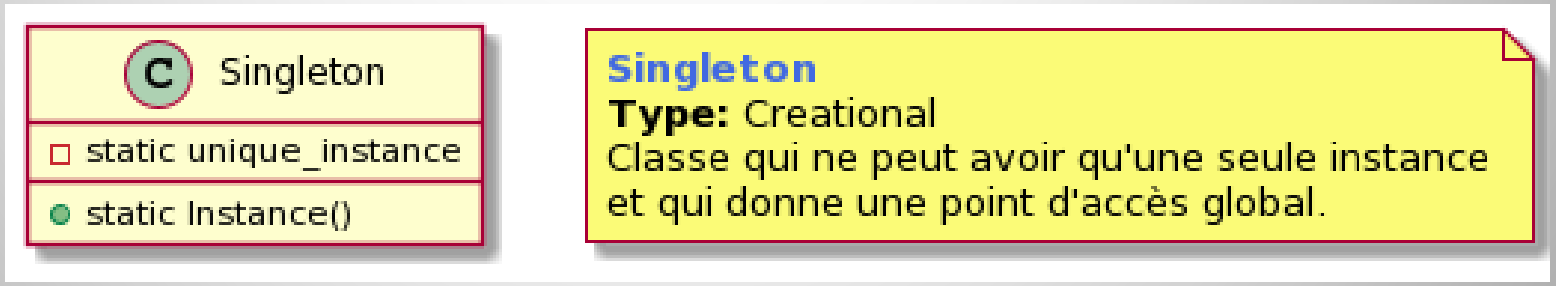
- https://doi.org/10.1145/1984674.1984682
- Exploring Game Architecture Best-Practices with Classic Space Invaders – 2011
- ➔Quels sont les patterns utiles d'après l'article?

- Quelques patterns à connaitre
  - Singleton
  - Observer
  - Strategy
  - Adapter
  - Facade

Creational / Singleton

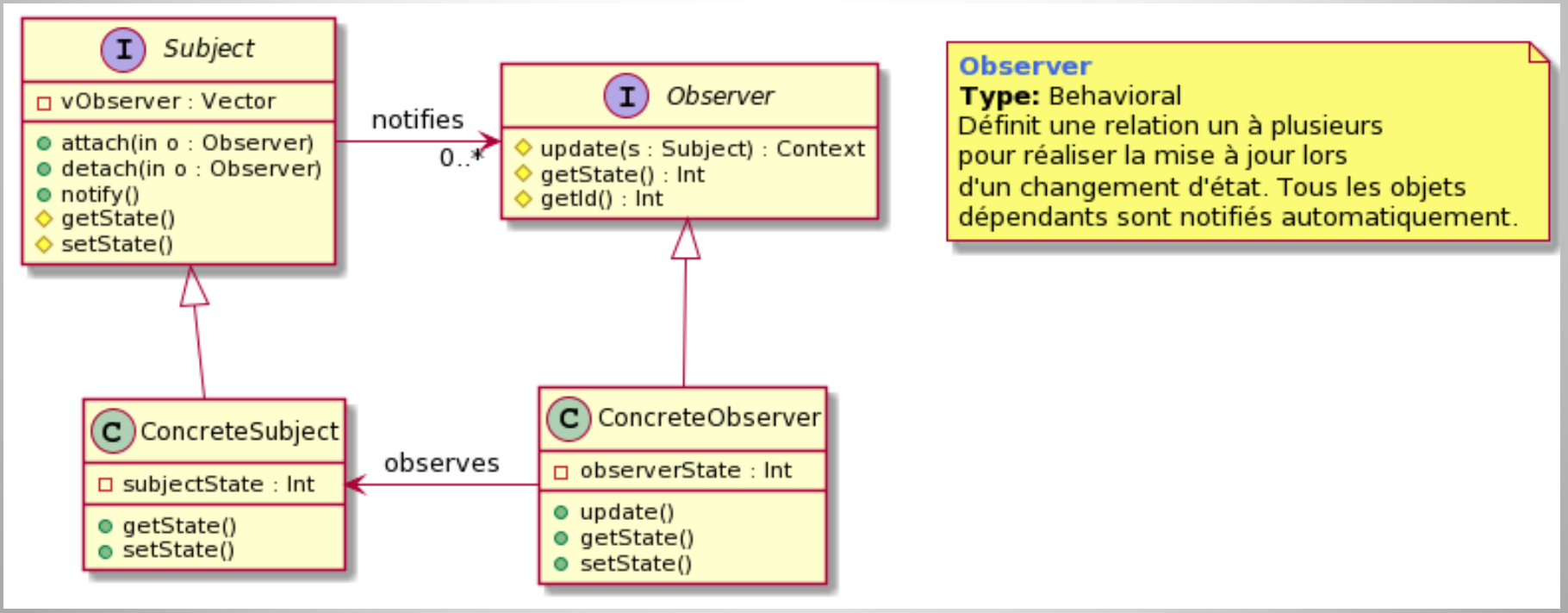
- Classe qui ne donne qu'une seule instance
- https://godbolt.org/z/5916Mc6Ez

```
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

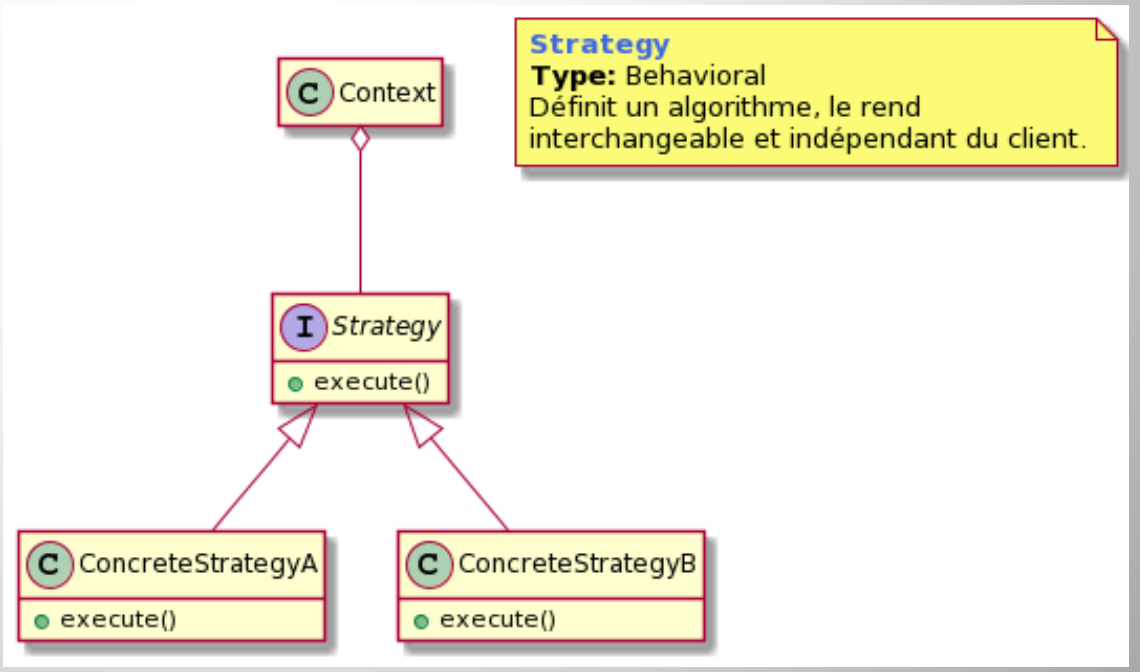


Behavioral / Strategy

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

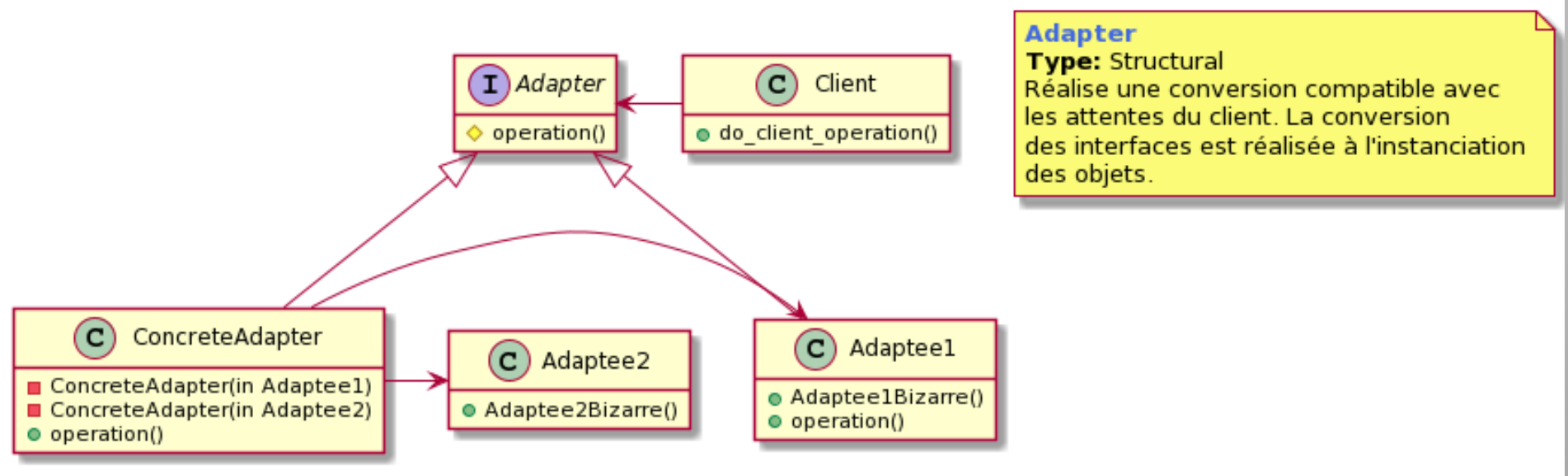
```
int main() {
    Context context(new ConcreteStrategyB());
    context.execute();
}
```

- ➔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();
}
```