M3105

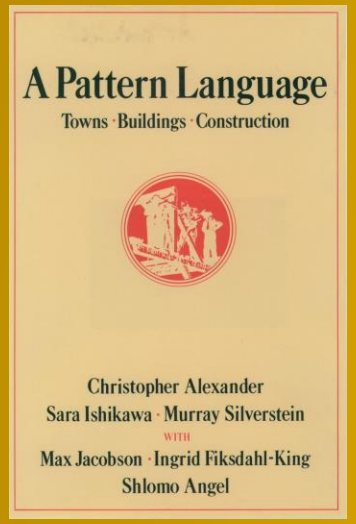
Le pattern Composite

COULAUD Guillaume
LERDUNG Kylian
VIVET Florian
COSTODE Yan

DUT 2
2020/2021

Sommaire

I - Présentation et définition des patterns



1977

Christopher Alexander

1994

Le GoF

Erich Gamma

John Vlissides

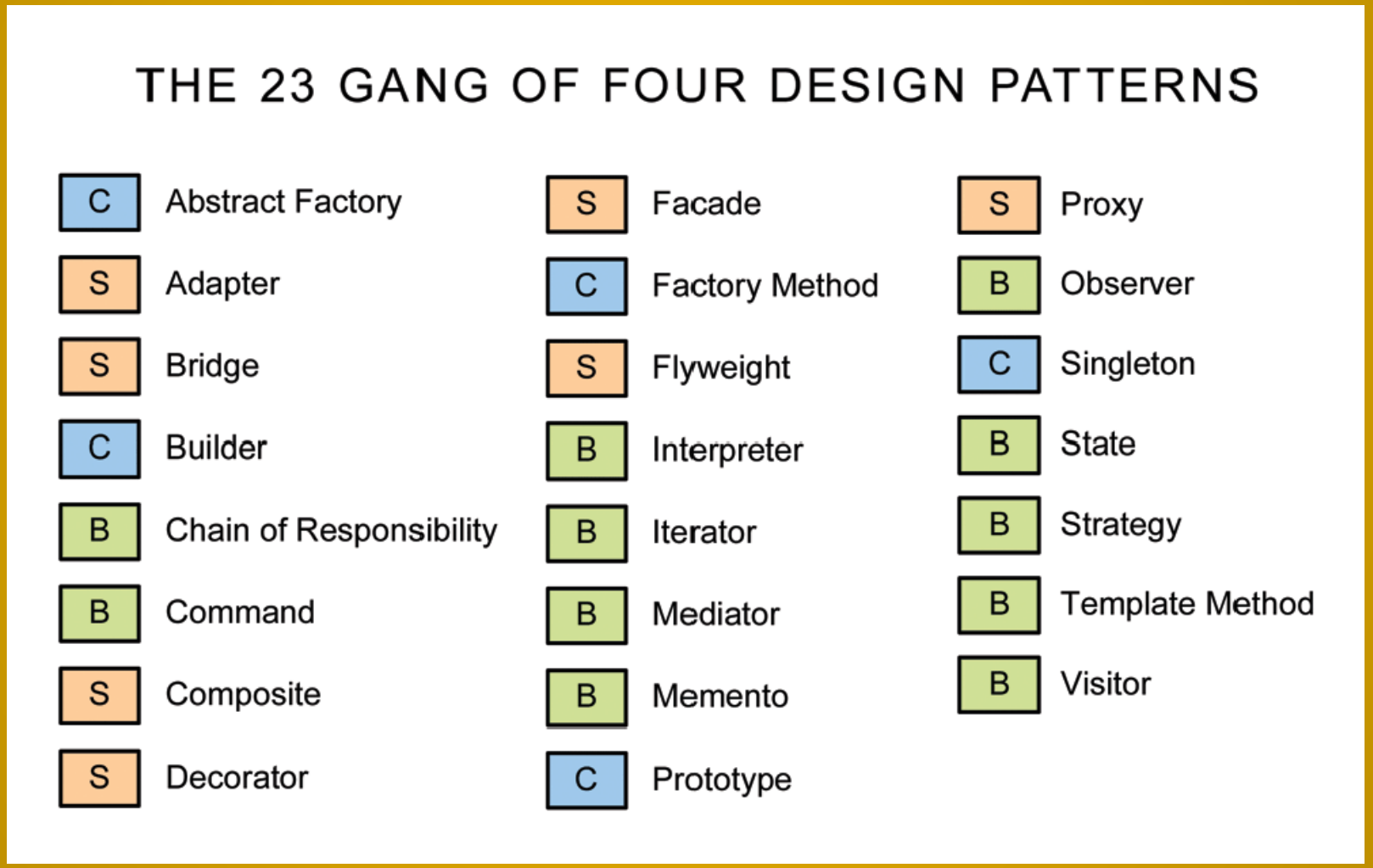
Ralph Johnson

Richard Helm

Les types de patterns

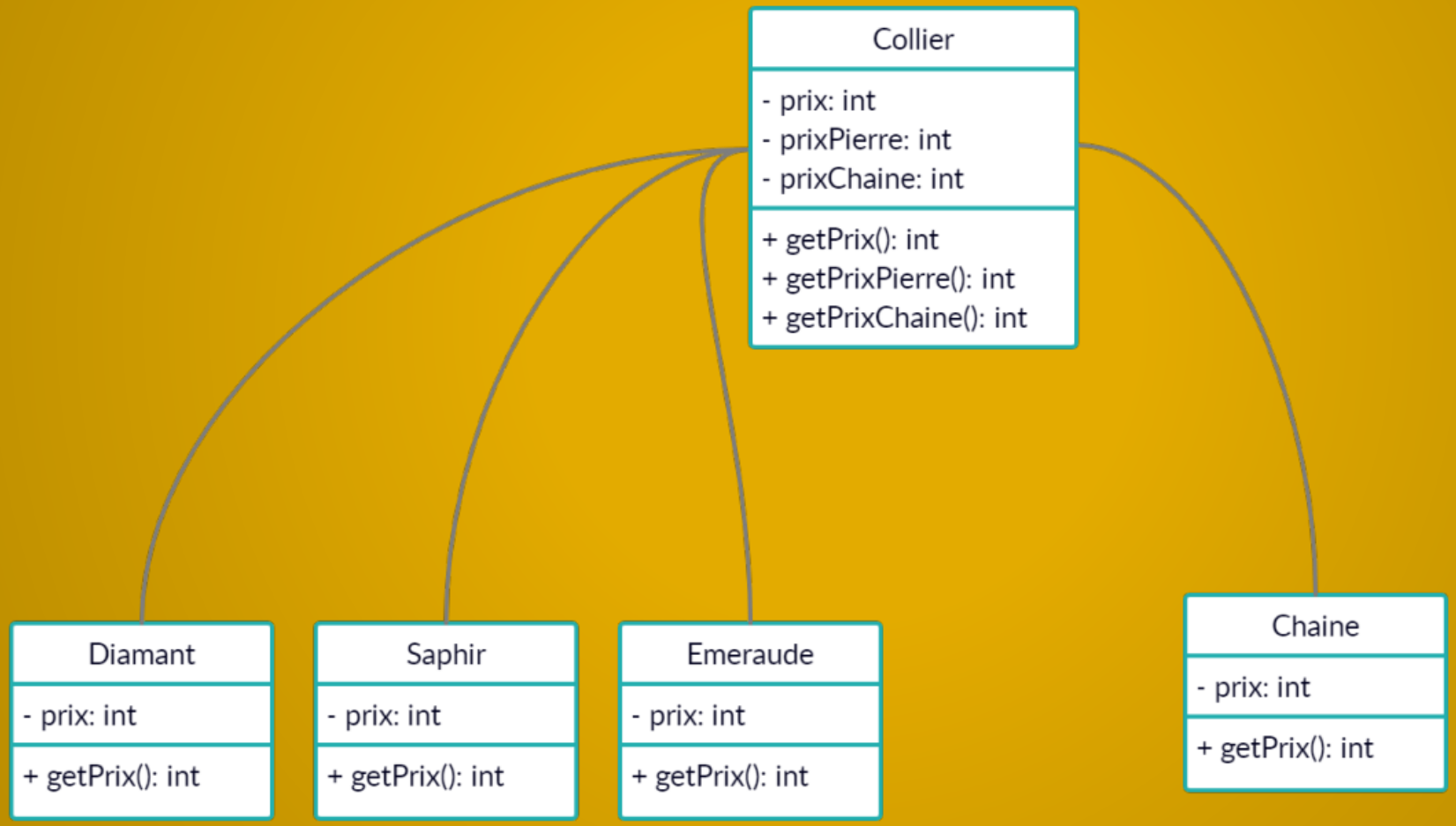
- Les créateurs
- Les structuraux
- Les comportementaux

Les patterns

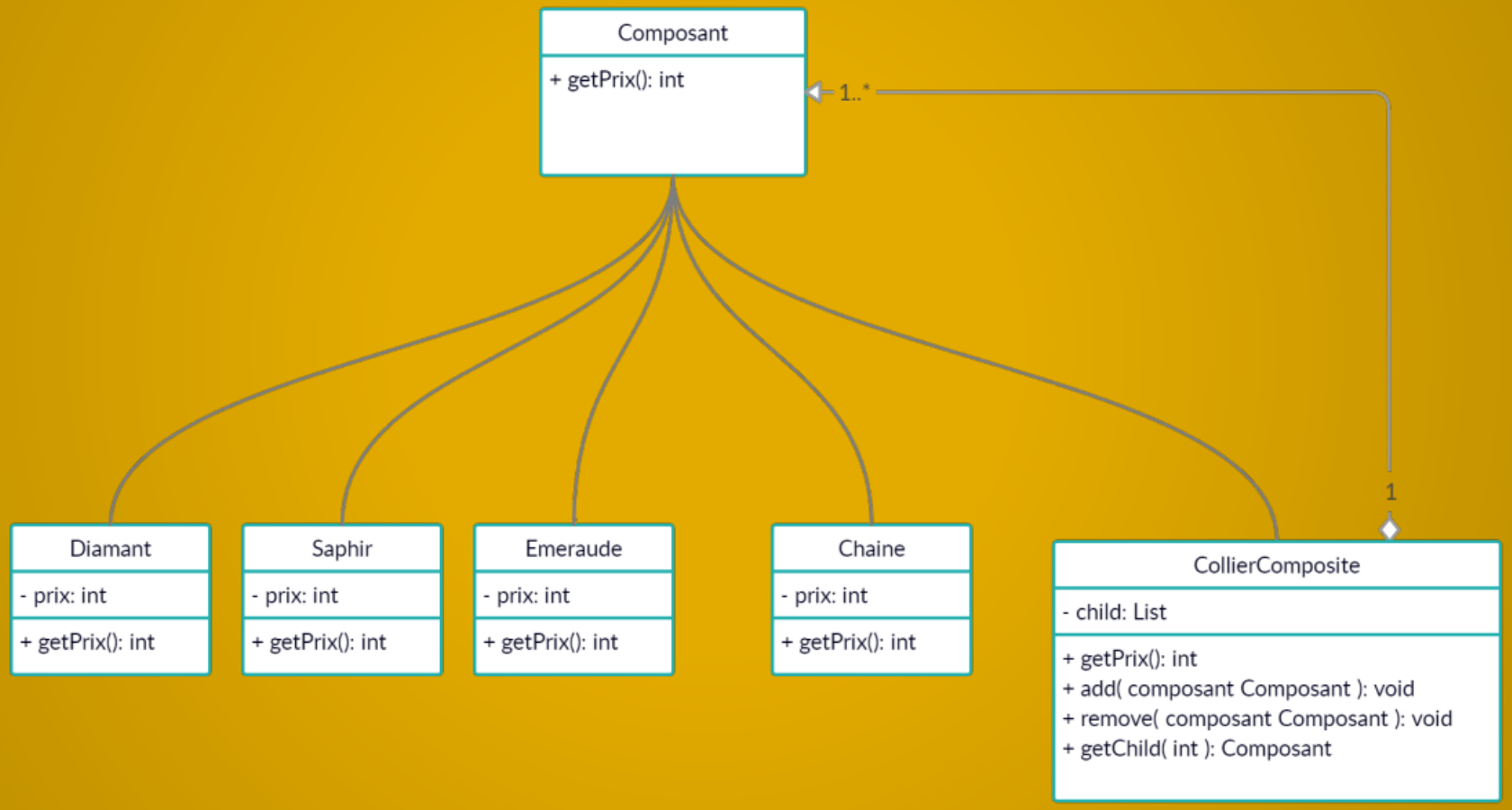


https://www.script-tutorials.com/design-patterns-in-php/

II – Mise en situation

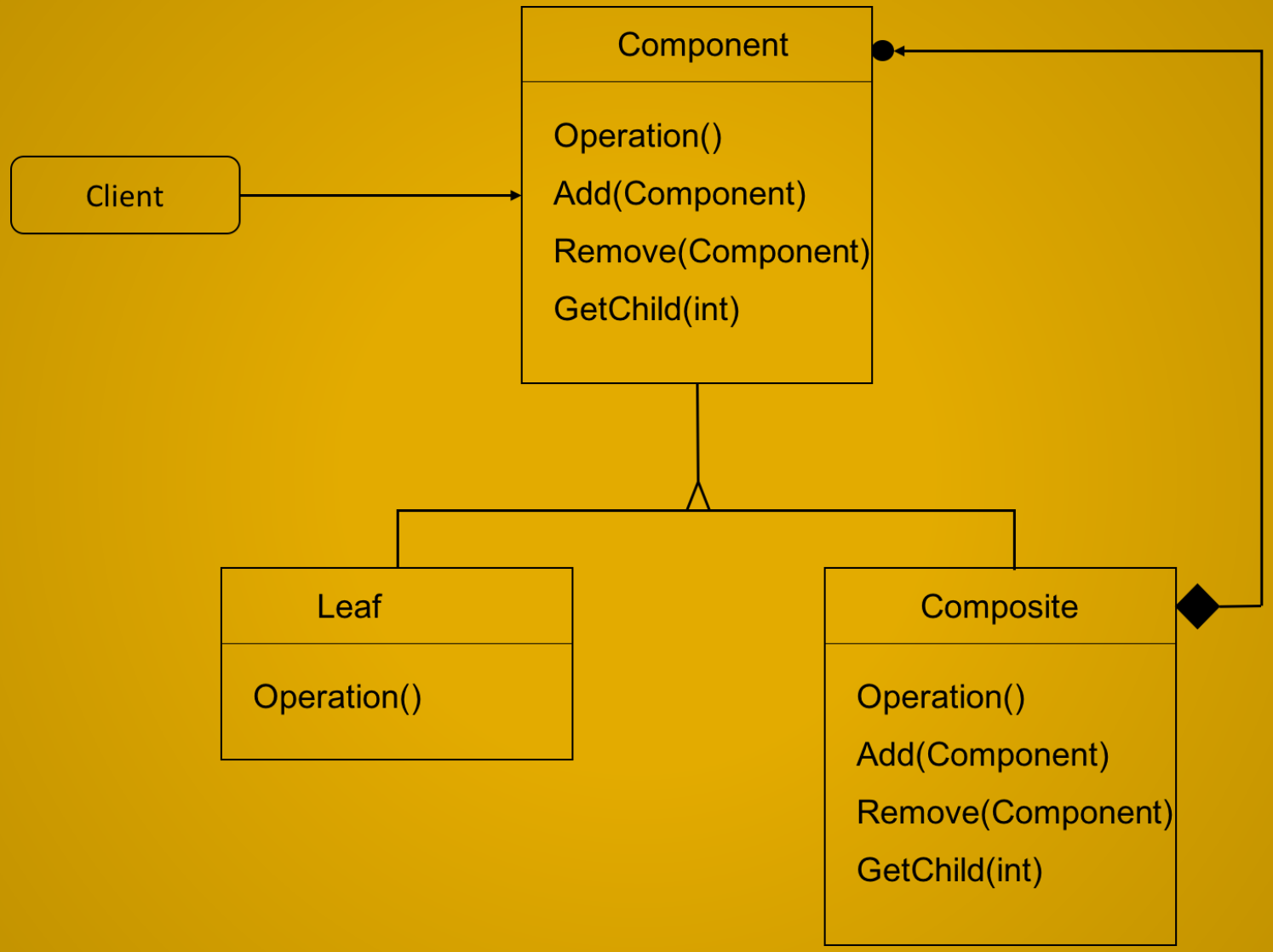


Une autre modélisation...



III - Présentation du Pattern Composite

https://www.geo.fr/voyage/les-arbres-aussi-sensibles-que-les-humains-163875



https://medium.com/elp-2018/composite-design-pattern-7d079a2bc86d

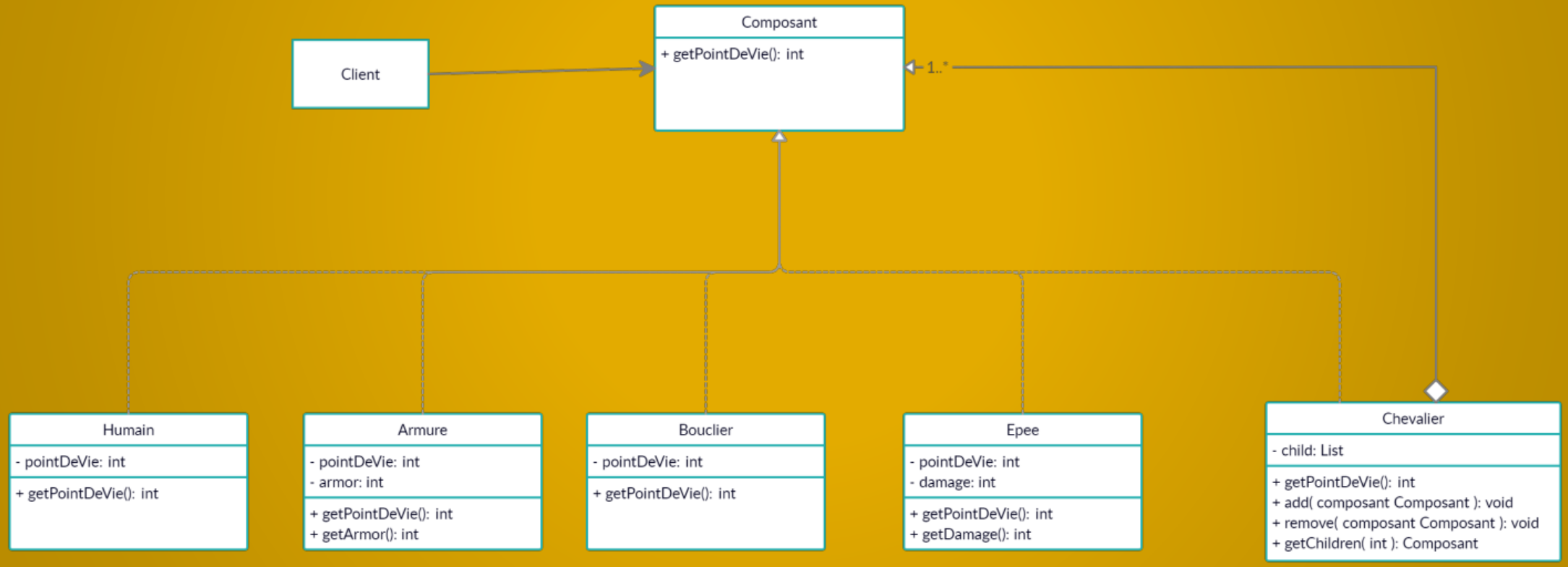
Les principes SOLID

- Single responsability
- Open/closed
- Liskov substitution
- Interface segregation
- Dependency inversion

Liens avec d’autres patterns

- Decorator
- Visitor
- Prototype

Un exemple pour conclure



```java
package composite;

public interface Composant {
    public int getPointdeVie();
}
```

```java
package composite;

import java.util.ArrayList;

public class ChevalierComposite implements Composant {
    private List<Composant> child;

    public ChevalierComposite () {
        this.child = new ArrayList<>();
    }

    public void add(Composant composant) {
        this.child.add(composant);
    }

    public void remove(Composant composant) {
        this.child.remove(composant);
    }

    public List<Composant> getChild() {
        return this.child;
    }

    @Override
    public int getPointdeVie() {
        int result = 0;
        for (int i = 0; i < child.size(); i++) {
            result += child.get(i).getPointdeVie();
        }
        return result;
    }
}
```

```java
package composite;

public class Armure implements Composant{
    private int pointDeVie;
    private int armor;

    public Armure (int pointDeVie, int armor) {
        this.pointDeVie = pointDeVie;
        this.armor = armor;
    }

    @Override
    public int getPointdeVie() {
        return this.pointDeVie;
    }
    public int getArmor() {
        return this.armor;
    }

    public void setPointDeVie(int valeur) {
        this.pointDeVie = valeur;
    }
    public void setArmor(int valeur) {
        this.armor = valeur;
    }
}
```

```java
package composite;

public class Epee implements Composant{
    private int pointDeVie;
    private int damage;

    public Epee (int pointDeVie, int damage) {
        this.pointDeVie = pointDeVie;
        this.damage = damage;
    }

    @Override
    public int getPointdeVie() {
        return this.pointDeVie;
    }
    public int getDamage() {
        return this.damage;
    }

    public void setPointDeVie(int valeur) {
        this.pointDeVie = valeur;
    }
    public void setDamage(int valeur) {
        this.damage = valeur;
    }
}
```

```java
package composite;

public class Bouclier implements Composant{
    private int pointDeVie;

    public Bouclier (int pointDeVie) {
        this.pointDeVie = pointDeVie;
    }

    @Override
    public int getPointdeVie() {
        return this.pointDeVie;
    }

    public void setPointDeVie(int valeur) {
        this.pointDeVie = valeur;
    }
}
```

```java
package composite;

public class Humain implements Composant{
    private int pointDeVie;

    public Humain (int pointDeVie) {
        this.pointDeVie = pointDeVie;
    }

    @Override
    public int getPointdeVie() {
        return this.pointDeVie;
    }

    public void setPointDeVie(int valeur) {
        this.pointDeVie = valeur;
    }
}
```

```java
package composite;

public class Client {
    public static void main(String[] args) {
        String pdv = " PDV.";
        Epee epee = new Epee(2031, 128);
        System.out.println("La vie de l'epee est de : " + epee.getPointdeVie() + pdv);
        Bouclier bouclier = new Bouclier(336);
        System.out.println("La vie du bouclier est de : " + bouclier.getPointdeVie() + pdv);
        Armure armure = new Armure(2035, 200);
        System.out.println("La vie de l'armure est de : " + armure.getPointdeVie() + pdv);
        Humain humain = new Humain(200);
        System.out.println("La vie de l'humain est de : " + humain.getPointdeVie() + pdv);

        ChevalierComposite chevalier = new ChevalierComposite();
        chevalier.add(epee);
        chevalier.add(bouclier);
        chevalier.add(armure);
        chevalier.add(humain);
        System.out.println("La vie du chevalier est de : " + chevalier.getPointdeVie() + pdv);
    }
}
```

```
La vie de l'epee est de : 2031 PDV.
La vie du bouclier est de : 336 PDV.
La vie de l'armure est de : 2035 PDV.
La vie de l'humain est de : 200 PDV.
La vie du chevalier est de : 4602 PDV.
```

IV - QCM

A TOI DE JOUER!

QCM

1- Quel élément n’est pas une caractéristique d’un design pattern?

a) Un langage de programmation
b) Des avantages et des inconvénients
c) Une problématique
d) Un nom

QCM

1- Quel élément n’est pas une caractéristique d’un design pattern?

a) Un langage de programmation ✔
b) Des avantages et des inconvénients
c) Une problématique
d) Un nom

QCM

2- Qui a écrit *A pattern langage* en 1977?

a) Erich Gamma
b) Christopher Alexander
c) Richard Helm
d) Ralph Johnson
e) John Vlissides

QCM

2- Qui a écrit *A pattern langage* en 1977?

a) Erich Gamma
b) Christopher Alexander ✔
c) Richard Helm
d) Ralph Johnson
e) John Vlissides

QCM

3- A quelle famille appartient le pattern Composite?

a) Les patterns structuraux
b) Les patterns comportementaux
c) Les patterns créateurs

QCM

3- A quelle famille appartient le pattern Composite?

a) Les patterns structuraux ✔
b) Les patterns comportementaux
c) Les patterns créateurs

QCM

4- De combien de parties est composé le pattern Composite?

a) Il est composé de 2 parties
b) Il est composé de 3 parties
c) Il est composé de 4 parties
d) Il est composé de 5 parties

QCM

4- De combien de parties est composé le pattern Composite?

a) Il est composé de 2 parties
b) Il est composé de 3 parties
c) Il est composé de 4 parties ✔
d) Il est composé de 5 parties

QCM

5- Quelles sont ces parties?

a) Tronc , branches , feuilles , herbe
b) Interface , classe , feuilles, composant
c) Pierre , feuille , ciseaux , composant
d) Client, l'interface du composant , les feuilles , container

QCM

5- Quelles sont ces parties?

a) Tronc , branches , feuilles , herbe
b) Interface , classe , feuilles, composant
c) Pierre , feuille , ciseaux , composant
d) Client, l'interface du composant , les feuilles , container ✔

QCM

6- Quels design patterns sont liés au design pattern Composite?

a) Abstract factory, Builder et Flyweight
b) Proxy, Facade et Iterator
c) Decorator, Visitor et Prototype
d) Prototype, Mediator et Decorator

QCM

6- Quels design patterns sont liés au design pattern Composite?

a) Abstract factory, Builder et Flyweight
b) Proxy, Facade et Iterator
c) Decorator, Visitor et Prototype ✔
d) Prototype, Mediator et Decorator

QCM

7- Quel est le composite dans notre exemple du chevalier?

a) La classe épée
b) La classe chevalier
c) L’interface composant
d) La classe humain

QCM

7- Quel est le composite dans notre exemple du chevalier?

a) La classe épée
b) La classe chevalier ✔
c) L’interface composant
d) La classe humain

QCM

8- A quoi sert l’interface Composant?

a) A regrouper les opérations communes aux éléments de l’arborescence
b) A implémenter la classe Composite
c) A faire joli dans le diagramme de classe
d) A instancier les composants

QCM

8- A quoi sert l’interface Composant?

a) A regrouper les opérations communes aux éléments de l’arborescence ✔
b) A implémenter la classe Composite
c) A faire joli dans le diagramme de classe
d) A instancier les composants

Des questions ?

Sources

- https://design-patterns.fr/introduction-aux-design-patterns
- https://fr.wikipedia.org/wiki/Patron_de_conception#:~:text=En%20informatique%2C%20et%20plus%20particuli%C3%A8rement,de%20conception%20d'un%20logiciel