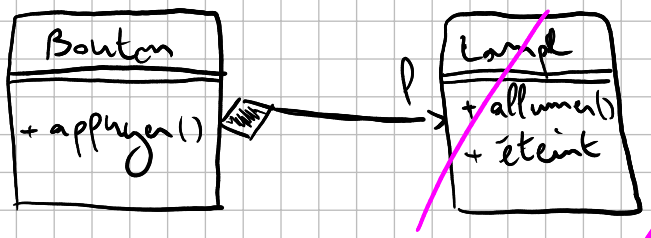
TD - Exercice 1



analyse

S : oui (single)

O : non (ouvert aux extensions & fermé aux modifs)

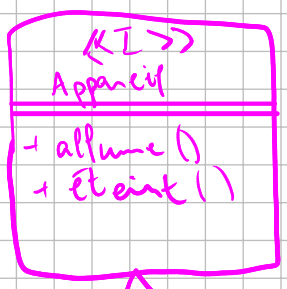
L : pas d'objet - pas d'héritage

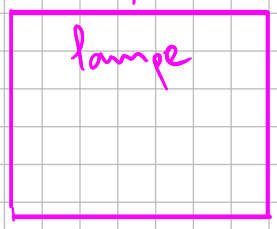
I : oui (interface minimale)

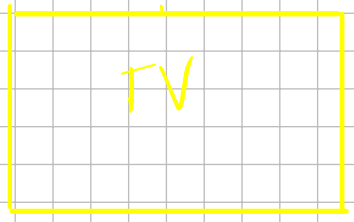
D : non (version de la dépendance)

↓

pas de dépendance du générique vers le particulier

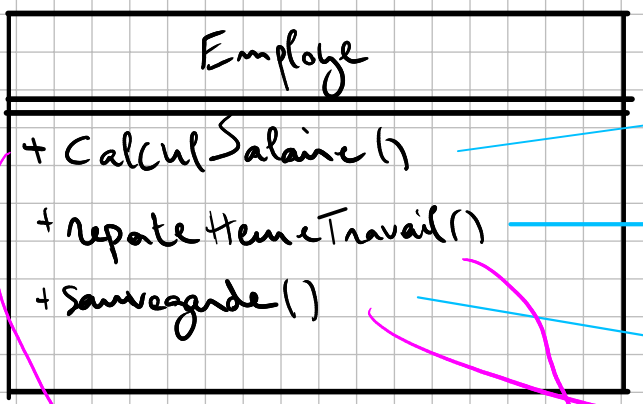







```
Button b = new Button();
b.associe(new Lampe);
b.associe(new TV);
```

Exercice 2 :



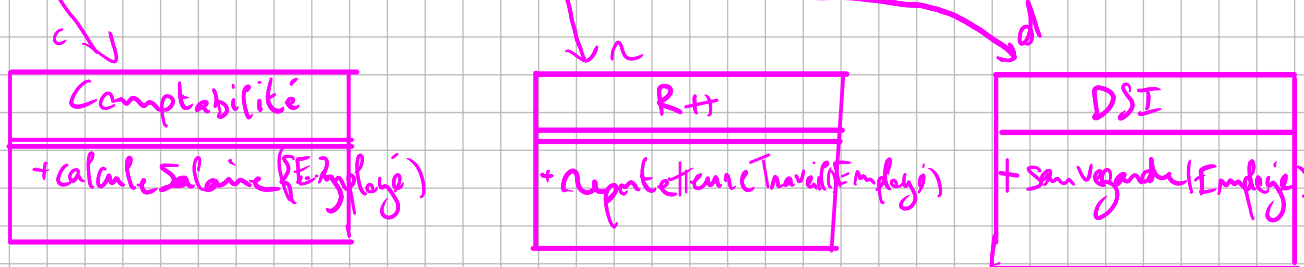
Analyse :

S = non

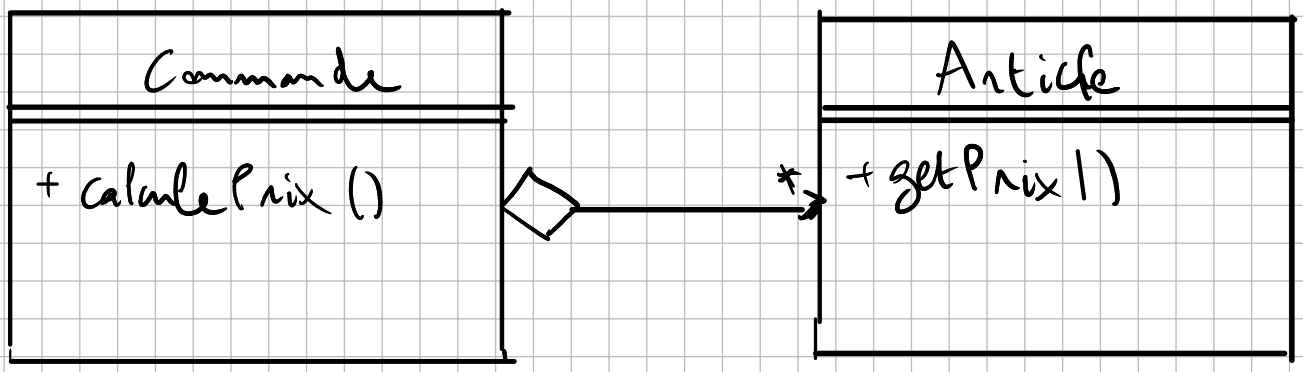
O = oui

L : pas d'objet

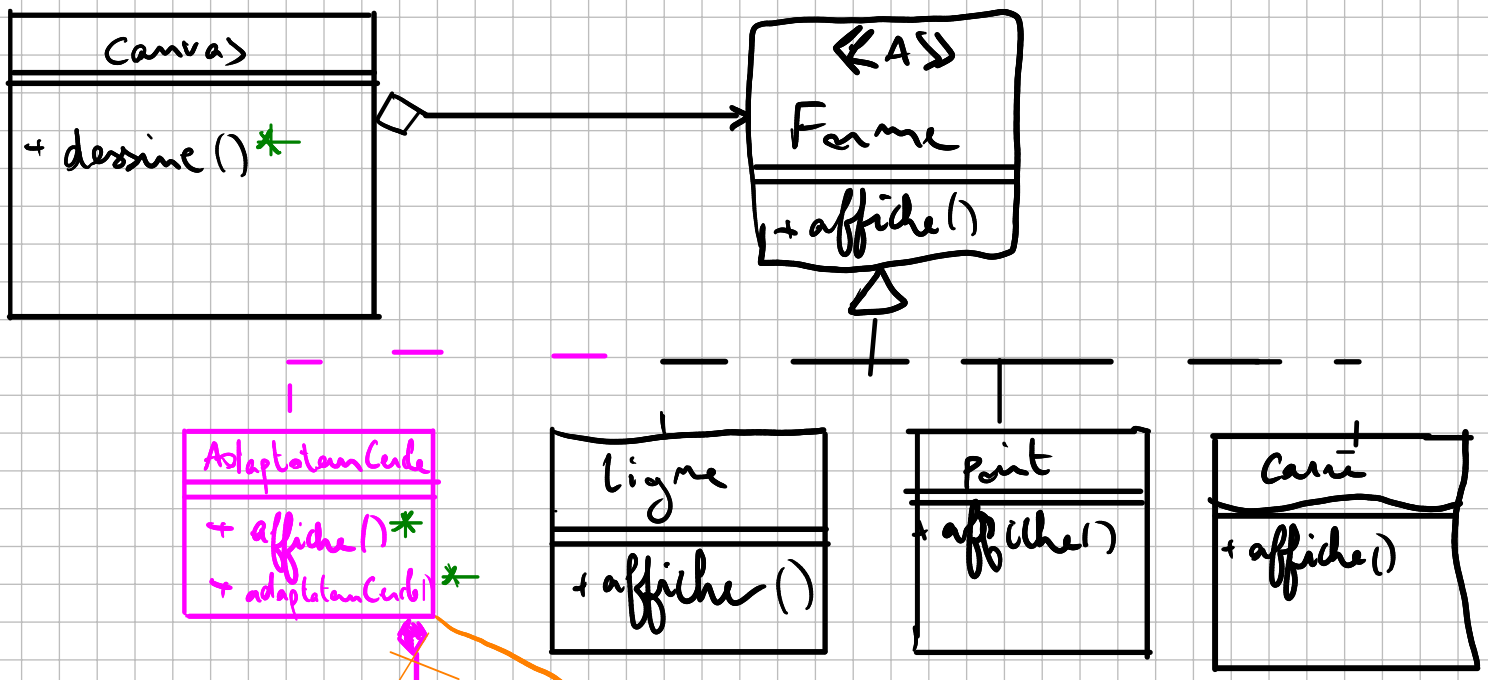
D



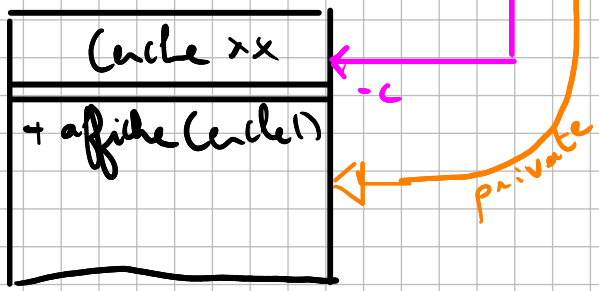
Exercice 3 :



Exercice 4 : Cas d'étude logiciel de dessin vectoriel



3) Intégrer un cercle

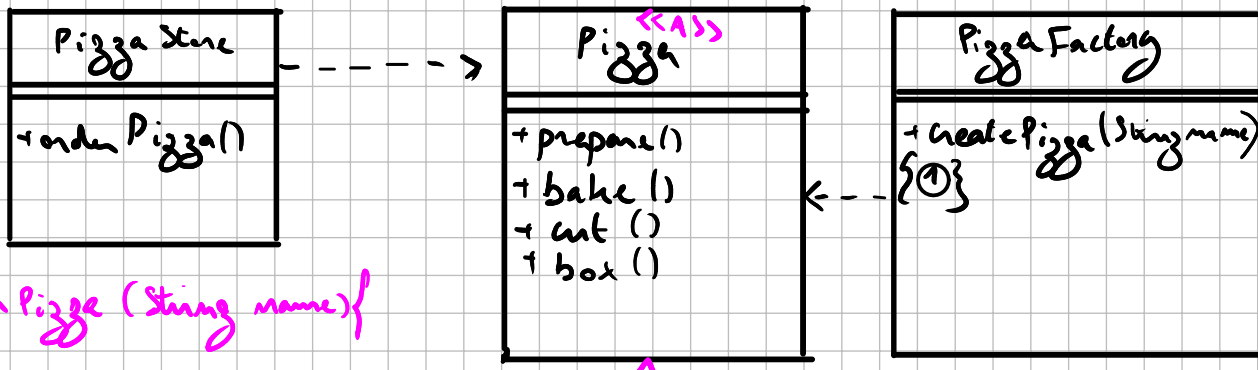



```
public void dessine(){
    for( Forme f:fs){
        f.affiche(),
    }
}
```

// fs.forEach(Forme::affi

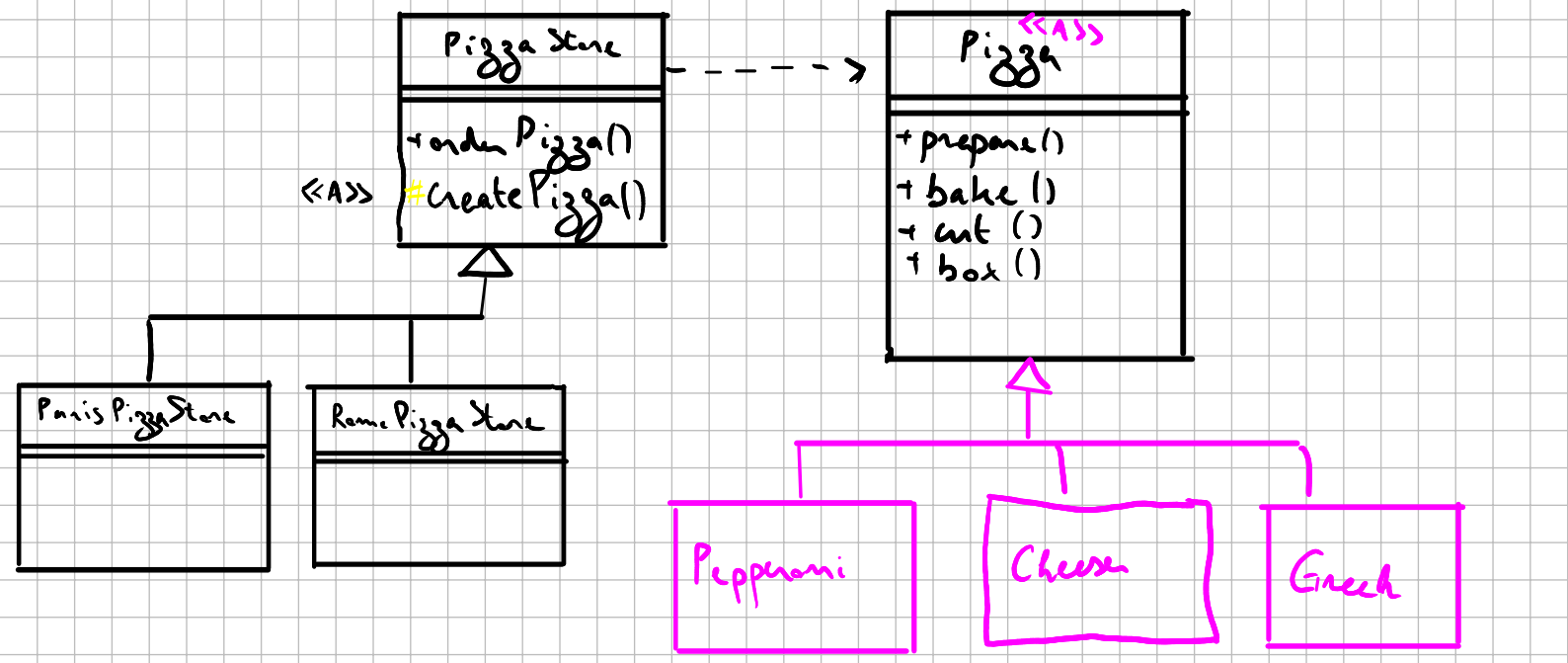
```
public final class AdaptateurCercle implements Forme{
    private CercleXX _c;
    public AdaptateurCercle(){_c= new CercleXX();}
    public void affiche(){_c.afficheCercle();  afficheCercle();
    }
}
```

Exo 5 : Pizzeria



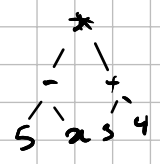

```
public Pizza orderPizza (String name){
    Pizza p;
    if("Pepperoni".equals(name)){
        p = new Pepperoni();
① } else if ("Cheese".equals(name)){
        p = new Cheese();
    }
    else { throw new Exception();
    p.prepare();
    p.cook();
② p.bake();
    p.box();
    return p;
}
```

SOLID



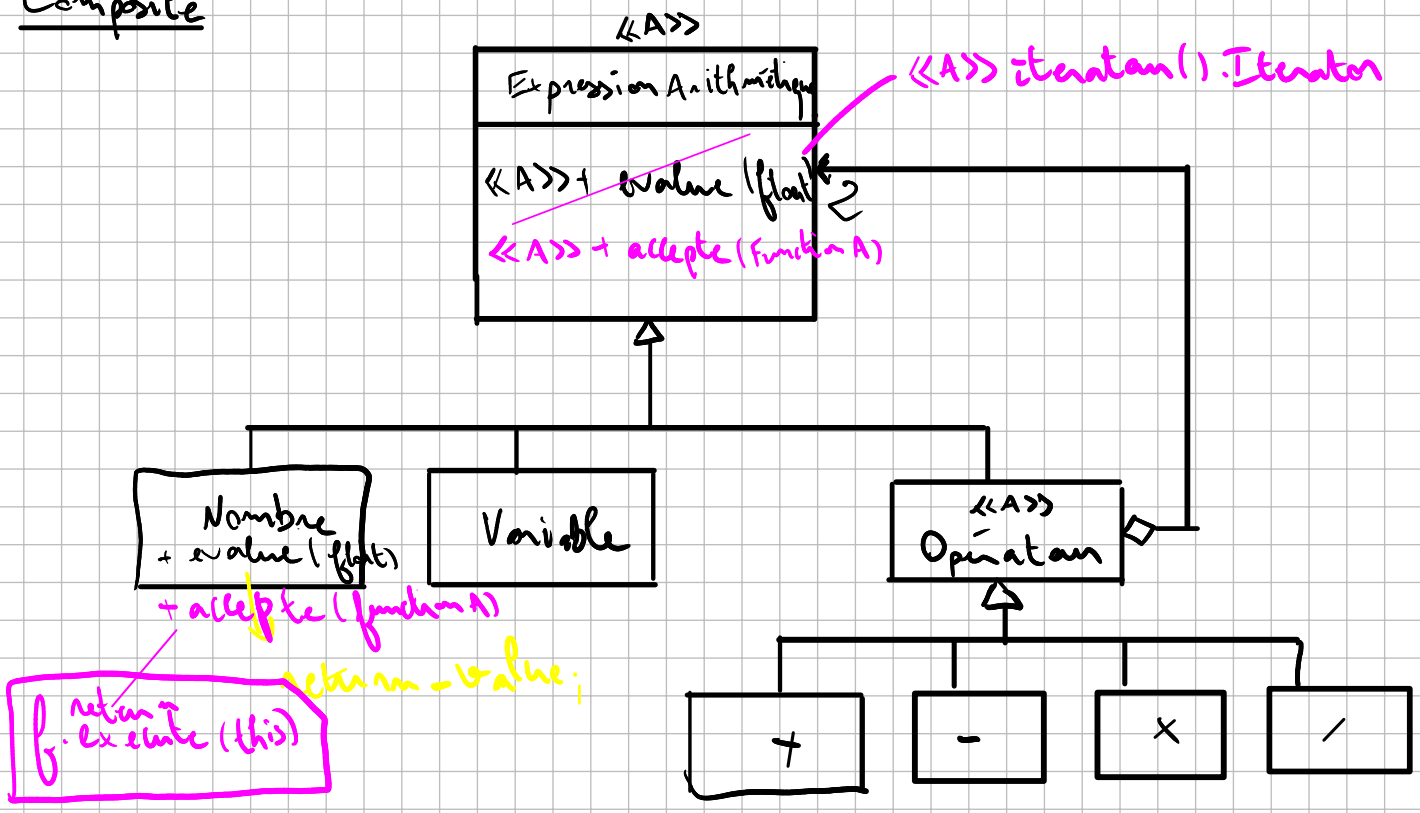
Cas d'étude : logiciel de calcul scientifique

expression littérale arithmétique $(5 - 2x) \times (3 + 4)$



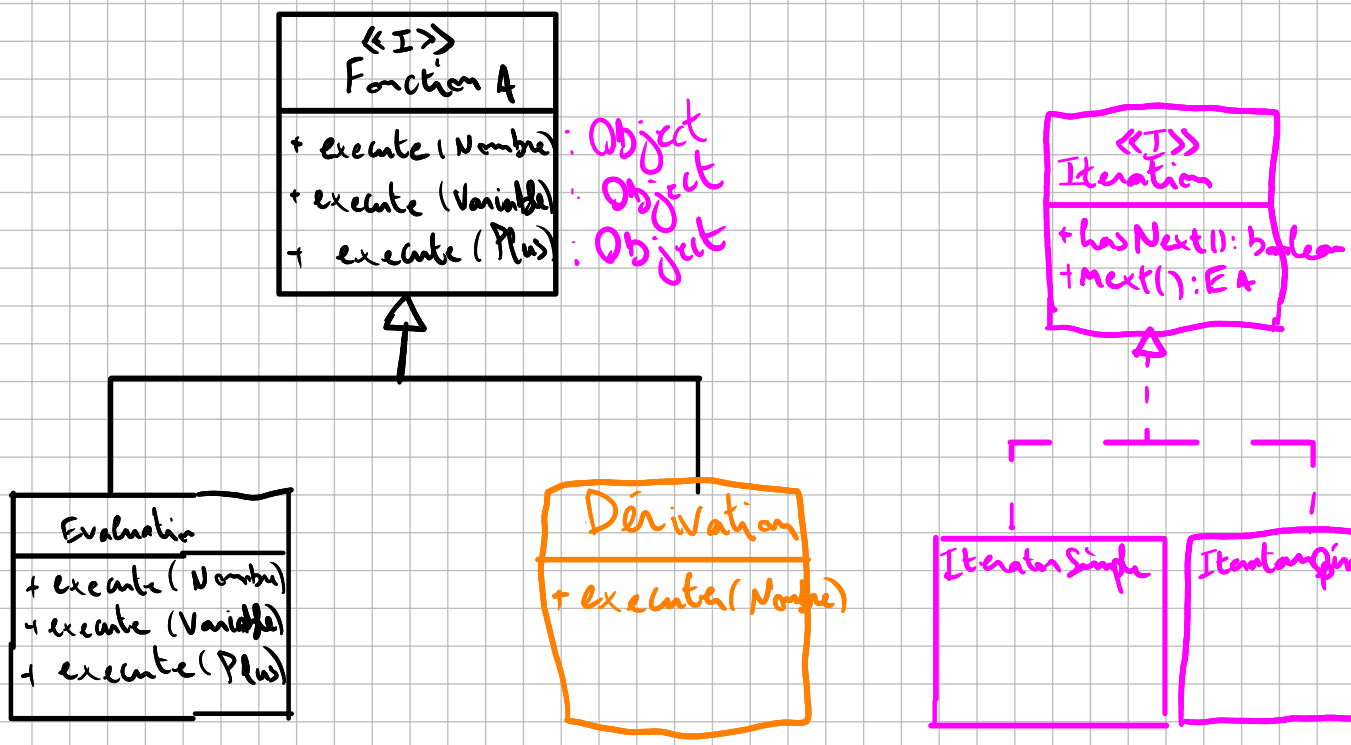
Modélisation ? limité aux 4 opérateurs binaires +, -, *, /

Composite



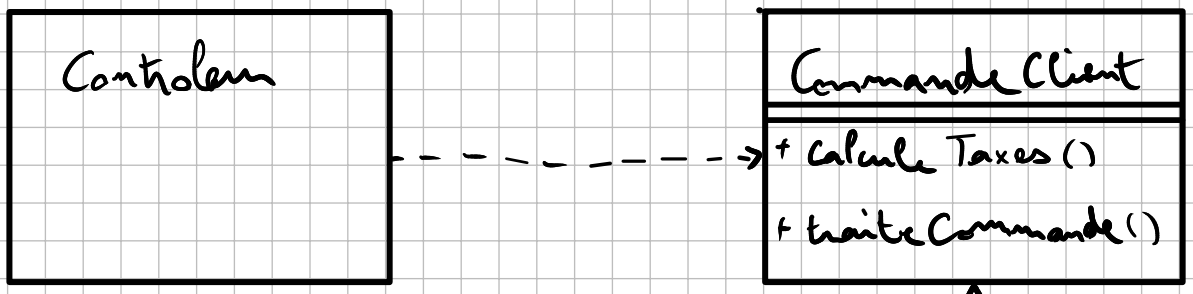

```
EA ea1 = new Nombre(5)
EA ea2 = new Variable("2x");
EA ea3 = new * ( new - ( new Nombre(5)
                         new Variable("2x"))
                 new + ( new Nombre(3)
                         new Nombre(4)));
```

7.

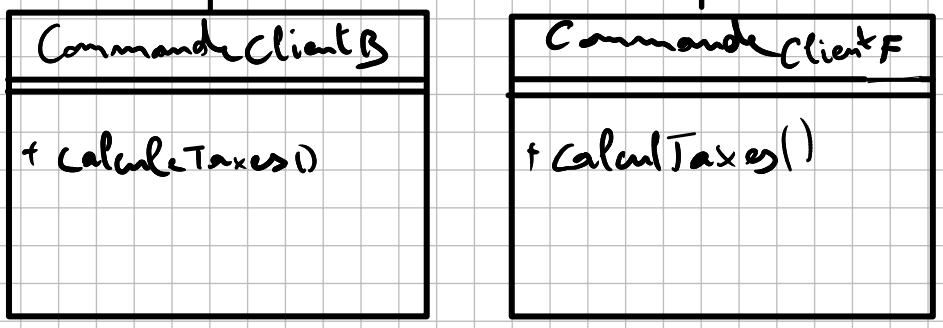


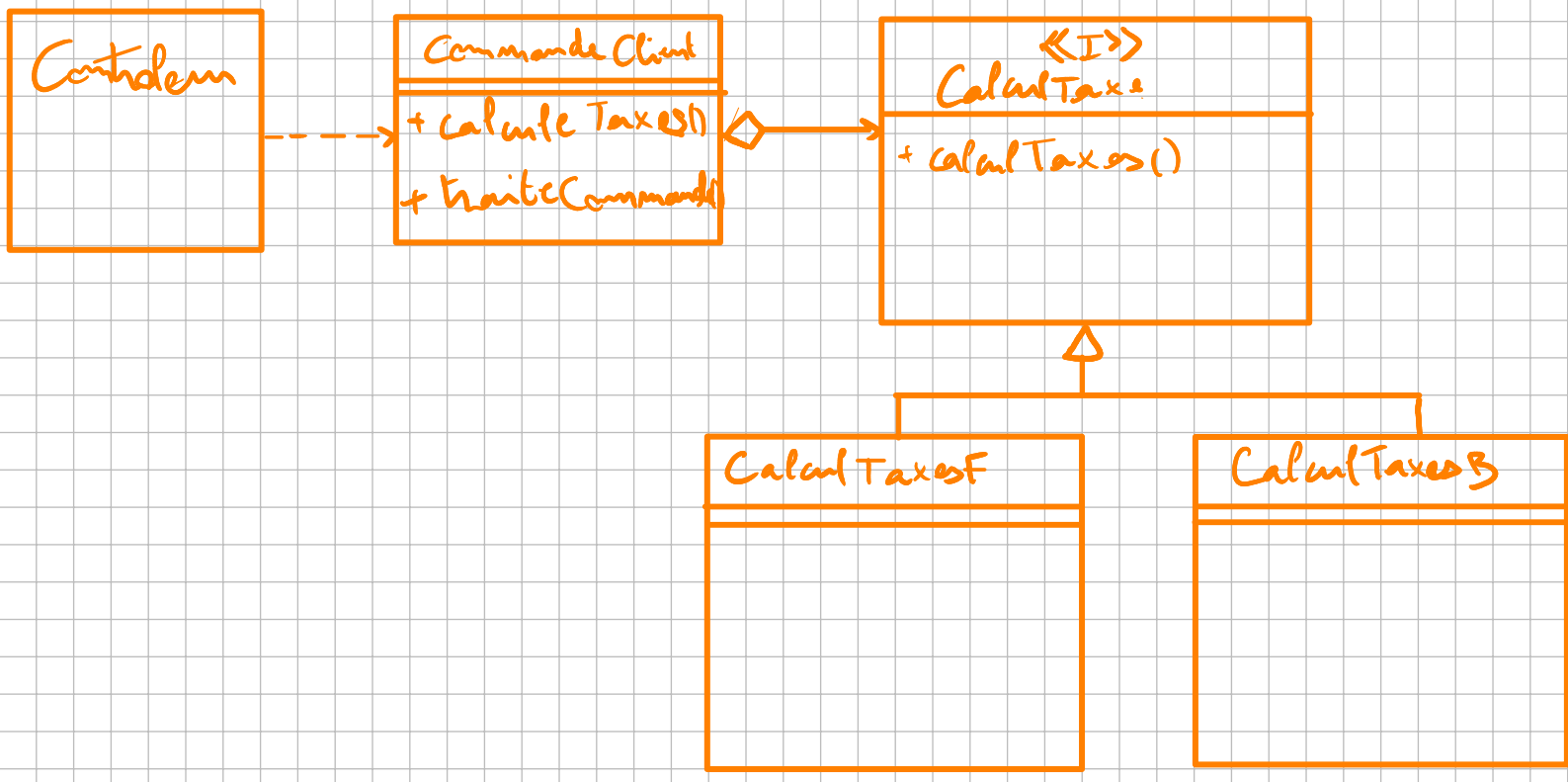
Etude de cas : logiciel de commerce électronique

Stratégie



SoLiD





Controleur

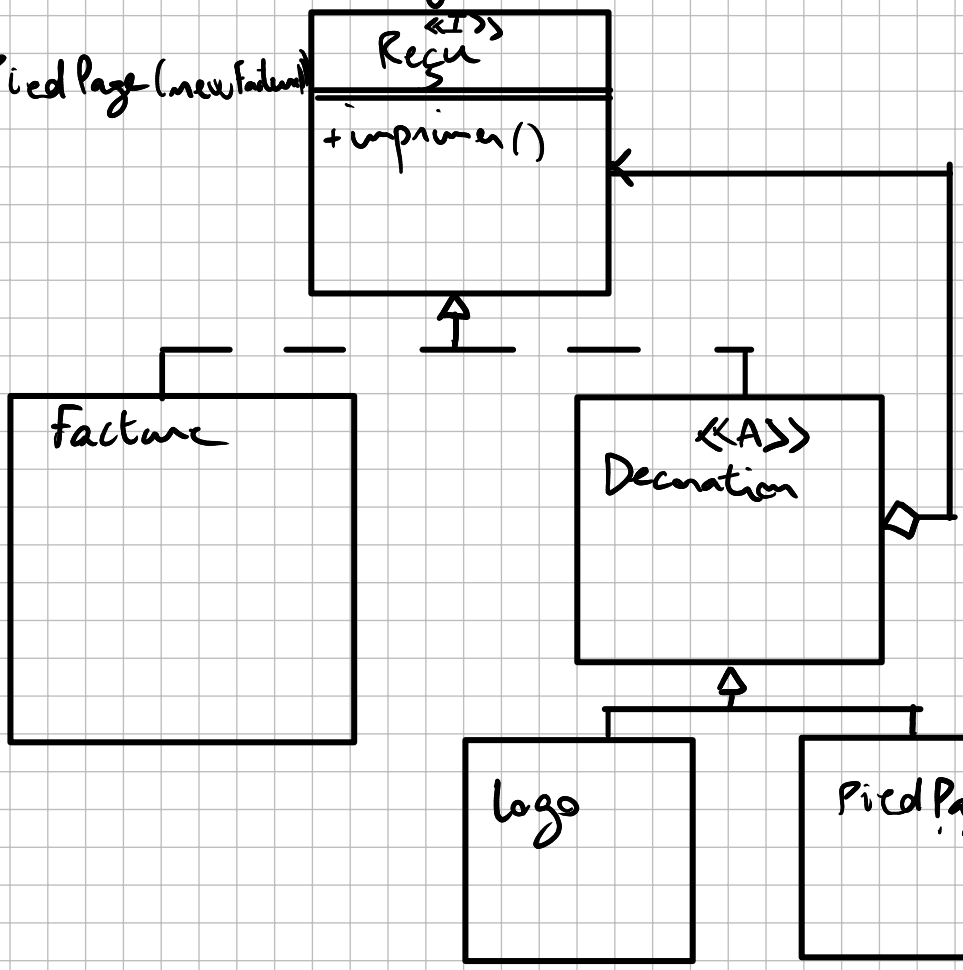
CommandeClient

+ calculeTaxes()

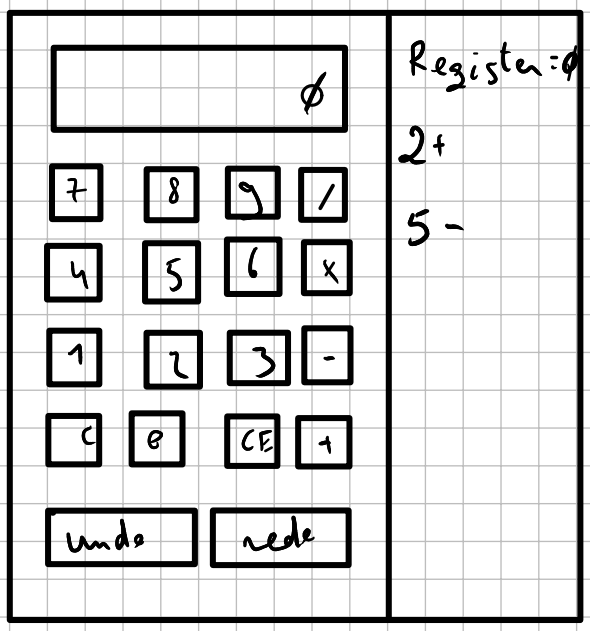
+ traiteCommande()

Décorateur

```
Recu r = new PiedPage(new Facture());
r.imprime();
Recu r = new Logo(new PiedPage(new Facture()));
```

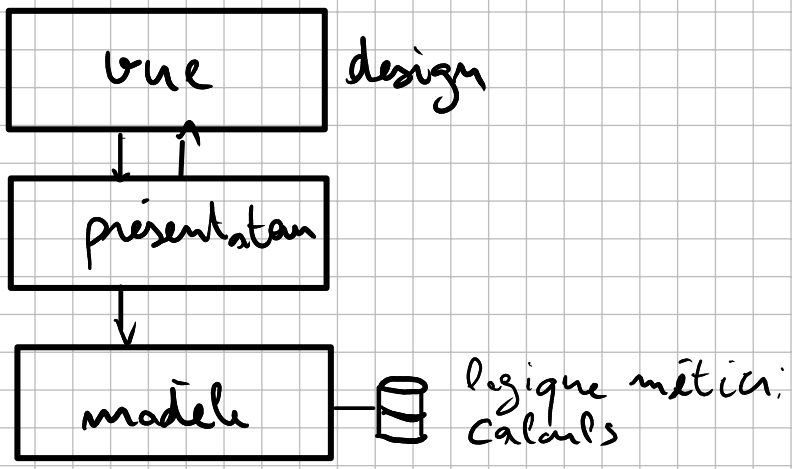


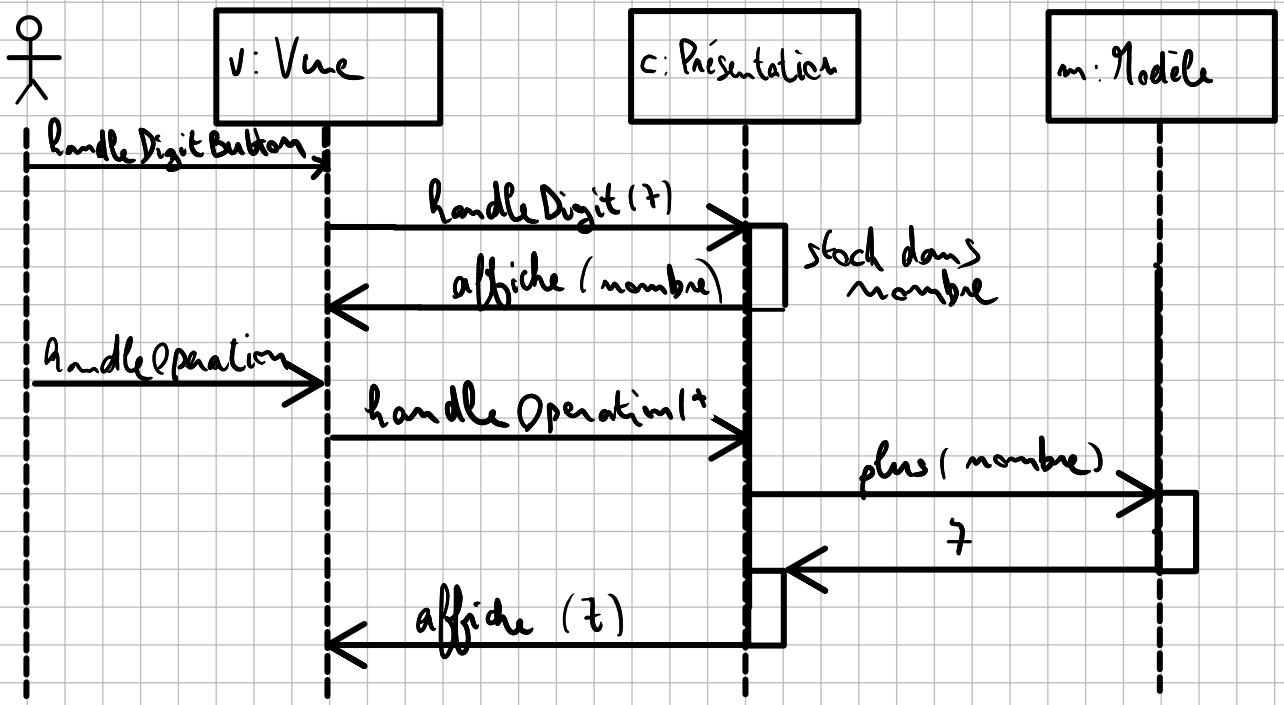
Calculatrice avec interface graphique (JavaFX)

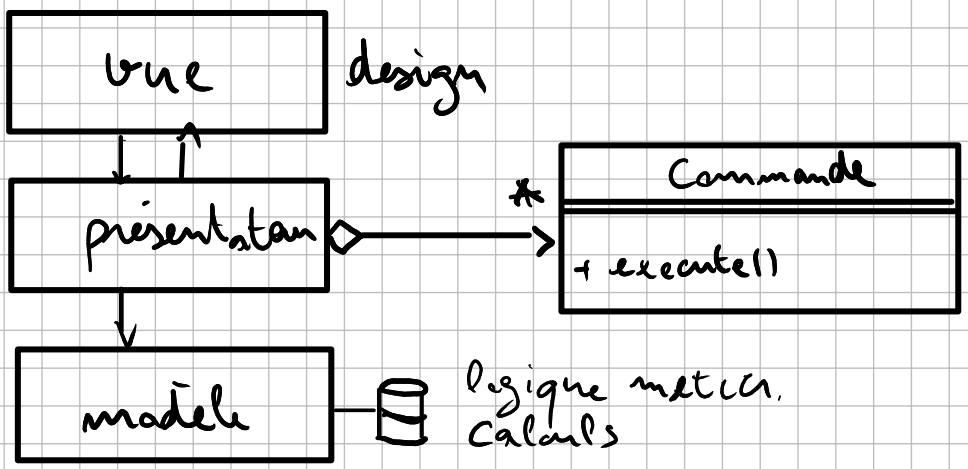


1° Architecture

Je partirai sur un modèle MVP





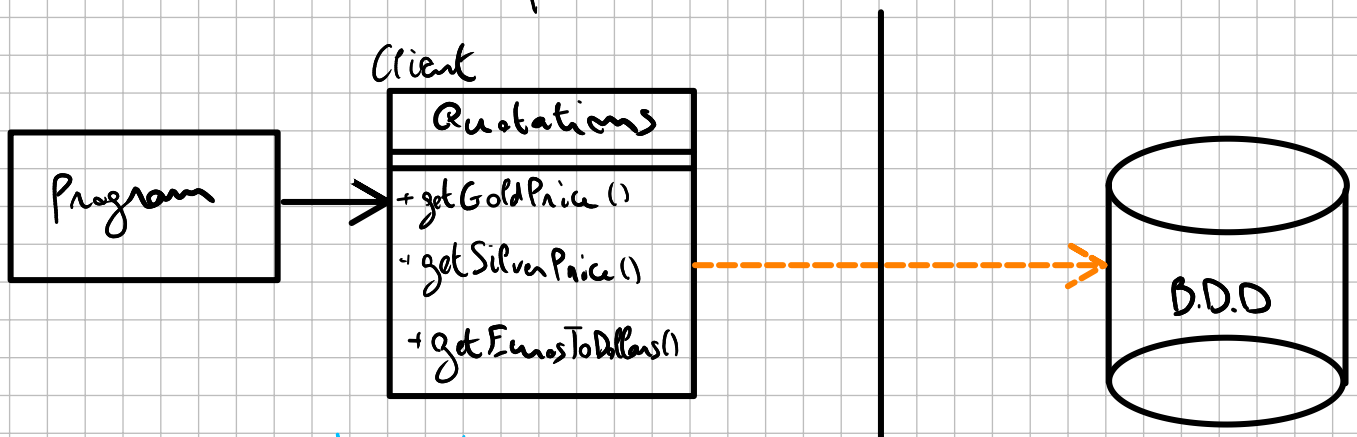


Exercice 15 : client - serveur

Système d'exploitation : or
: argent
: conversion euro/dollars

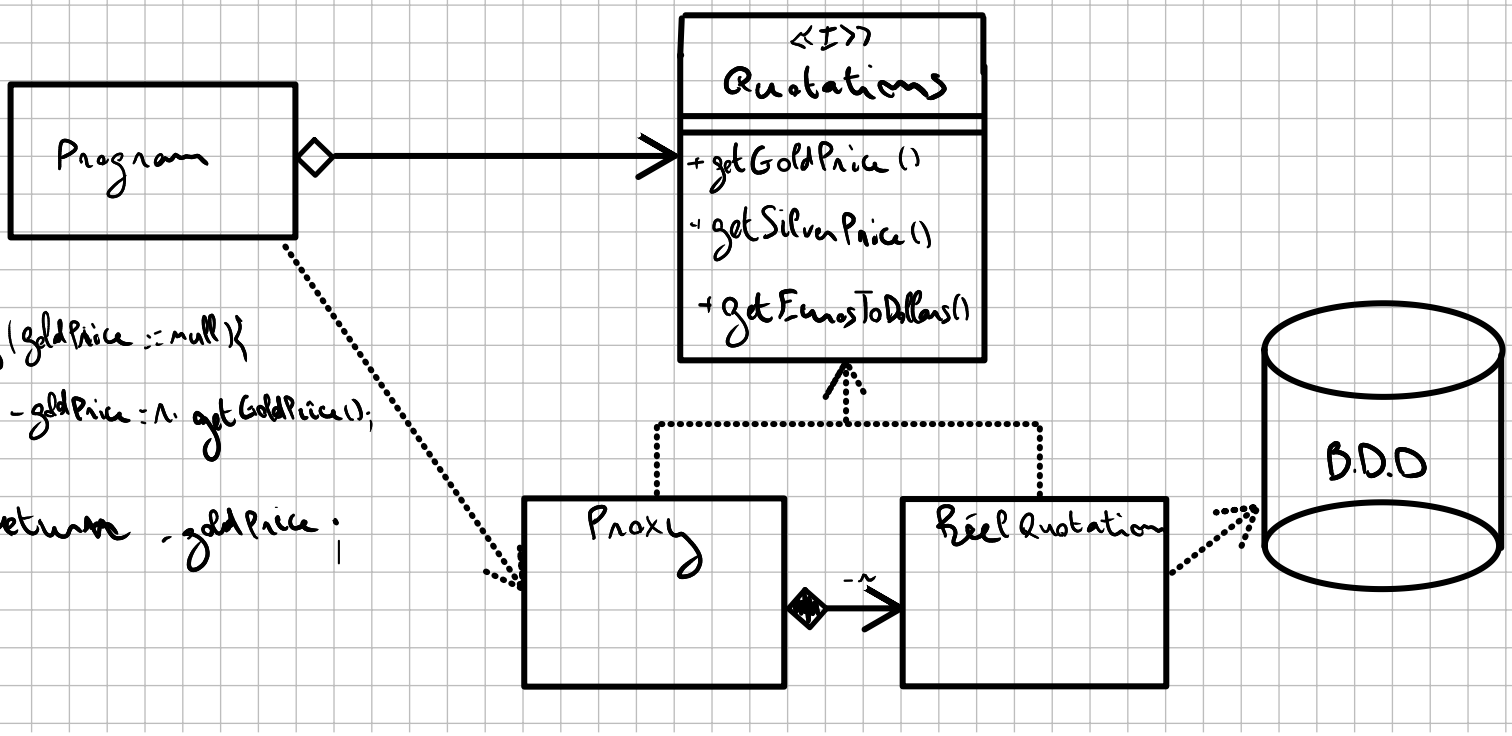
Interface | serveur

la cotation n'évolue pas

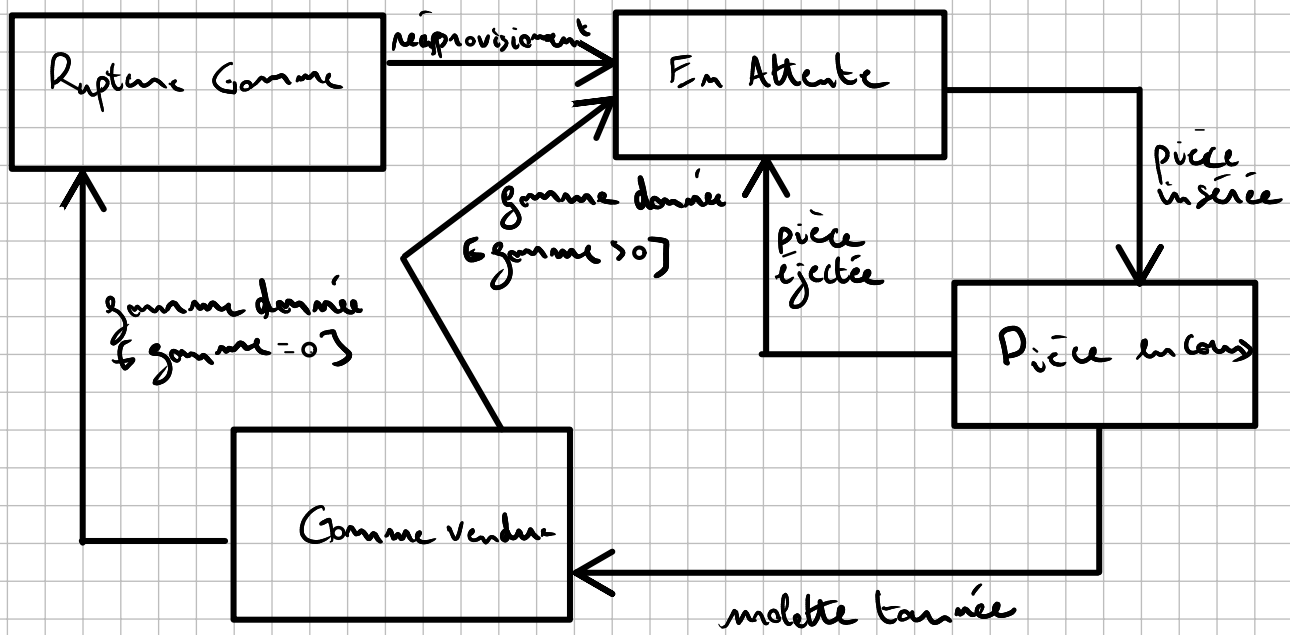


S x ↔ communication réseau
stocker les données
O √
L √
I √
D √

Conception : Proxy



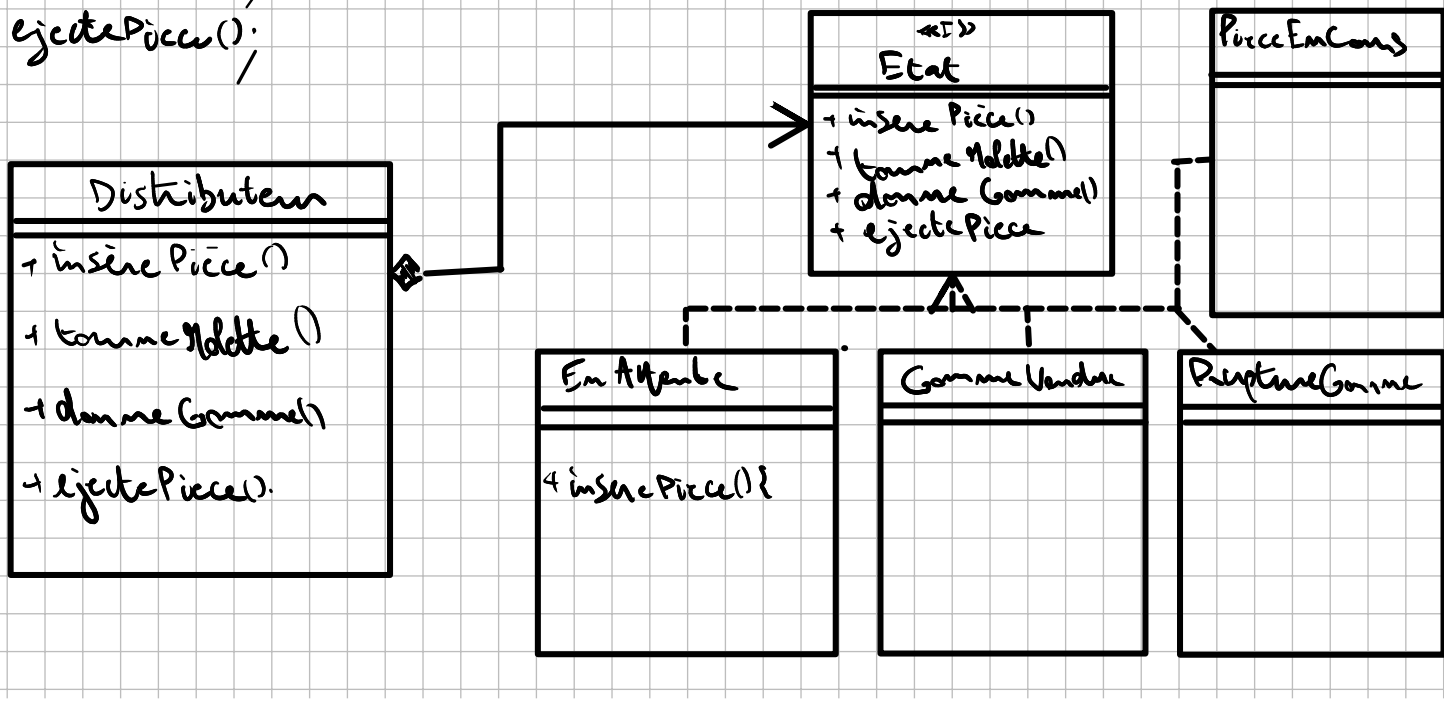
Exo 12 :



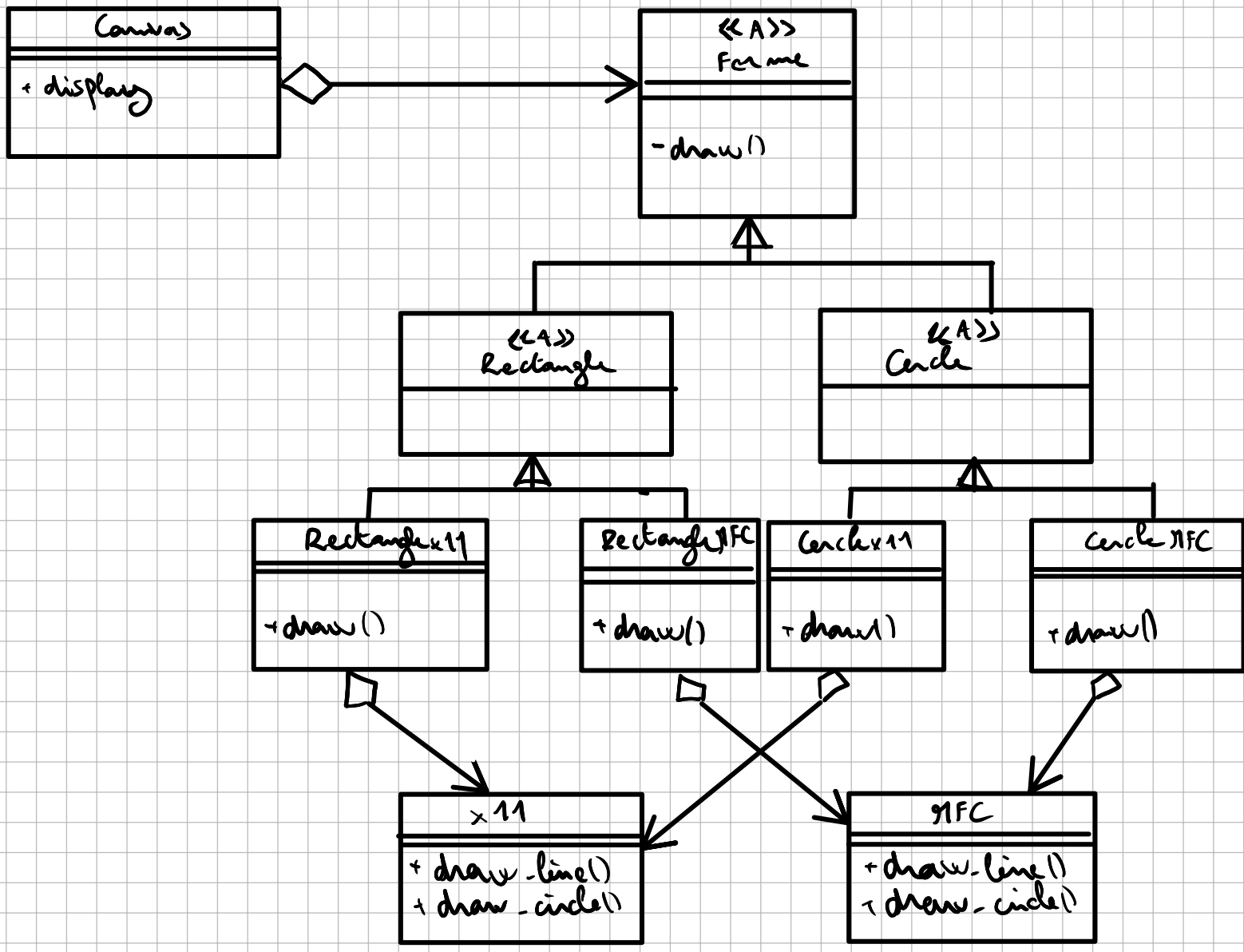

```
enum Etat { EN_ATTENTE, PIECE_EN_COURS, GOMME_VENDUE, RUPTURE_GOMME }

public class Distributeur {

    Etat etat = EN_ATTENTE;
    public insertPiece(){
        switch (_etat){
            case EN_ATTENTE -> { sout("credit : 1 piece"); _etat = PIECE_EN_COURS}
            case GOMME_VENDUE -> {Sout("Produit déjà distribué"),}
            case RUPTURE_GOMME -> {Sout("Machine hors service"), ejectePiece()}
            Case PIECE_EN_COURS -> {sout("Comfet")}.
        }
    }
    tourneMolette().
    donneGomme().
    ejectePiece().
```



TD6 : Etude de cas : dessin vectoriel



pb avec Lishov basique : classes concretes héritent l'une de l'autre. Quand c'est une classe abstraite c'est fait pour à priori.

S ✓

O ✓

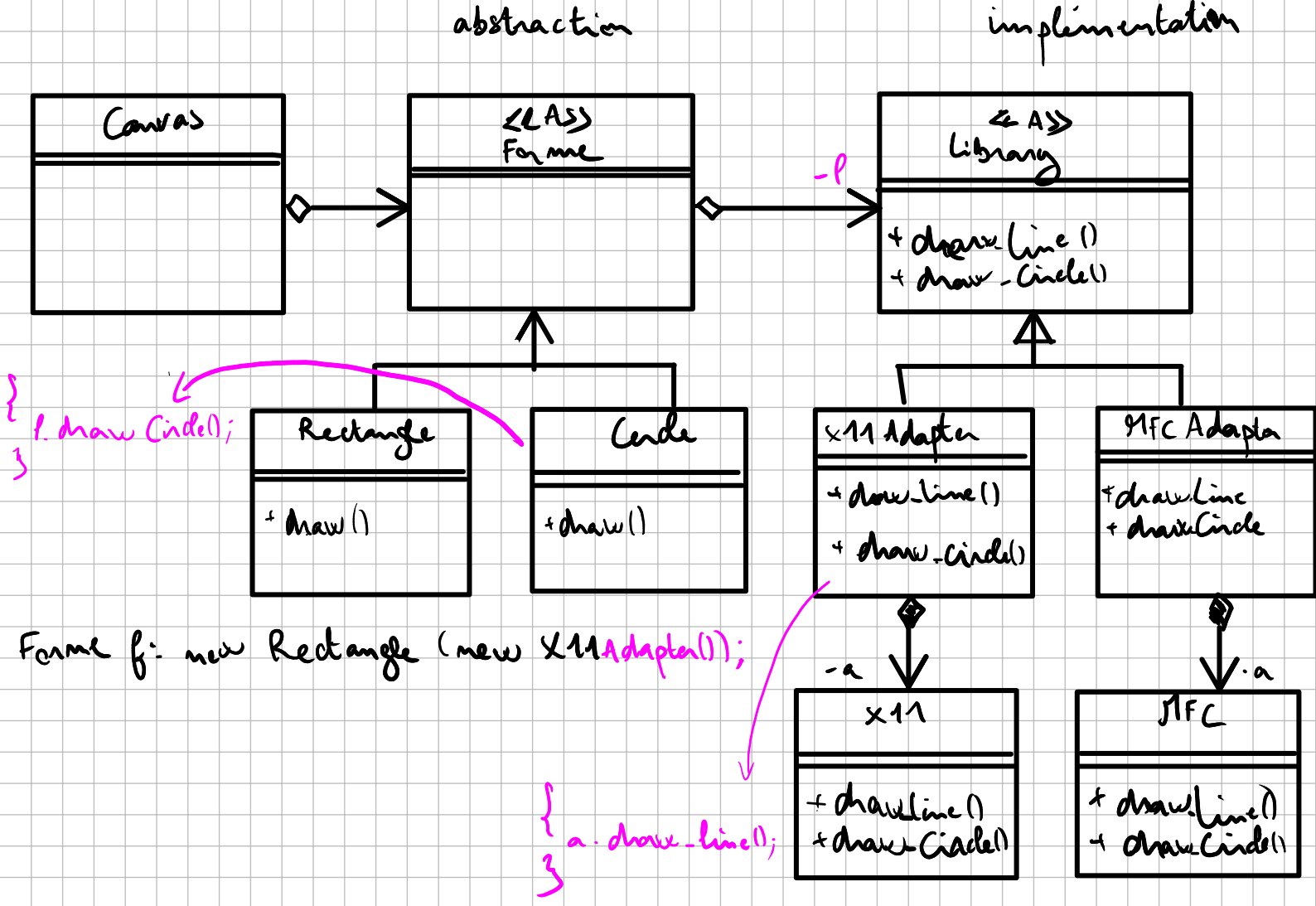
L ✓

I ✓

D ✓

pb : Si on ajoute une bibliothèque ou une forme on a bcp de classes.

-> il faudrait mettre en place un ~~décorateur~~ Pont


abstraction
implementation
Canvas
<<AS>>
Forme
<<A>>
Library
+ draw_line()
+ draw_Circle()
-p
{
f.drawCircle();
}
Rectangle
+ draw()
Cercle
+ draw()
x11Adapter
+ draw_line()
+ draw_circle()
MfcAdapta
+ drawLine
+ drawCircle
Forme f: new Rectangle (new X11Adapter());
-a
.a
x11
+ drawLine()
+ drawCircle()
MFC
+ drawLine()
+ drawCircle()
{
a.draw_line();
}