PATRON DE CONCEPTION OBSERVATEUR (OU OBSERVER)

PLAN

1. Présentation des patrons de conception
2. Description & Explication de l'Observateur
3. Exemples d'utilisation de l'Observateur

DÉFINITION

En Informatique :

- Solution **générique** répondant à un problème **spécifique**
- S'applique au développement en **programmation orientée objet**
- Est décrit :
  - Nom : Observateur
  - Problème : *Voir suite*
  - Solution : *Voir suite*

TYPES

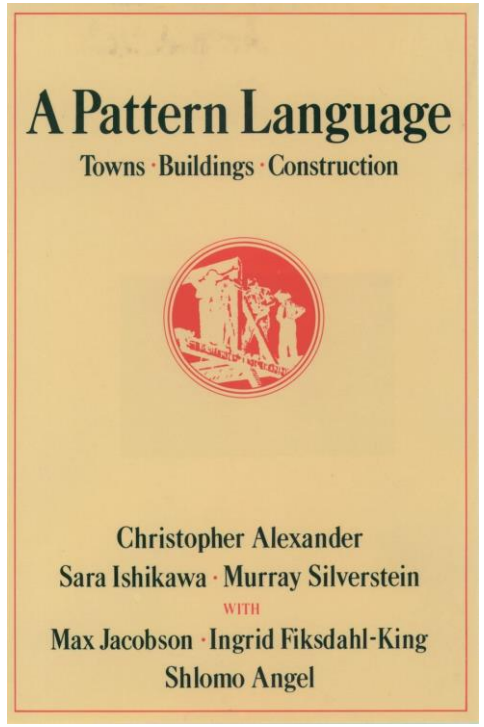
3 Types de Patron de Conception :

- **Création**
- **Structure**
- **Comportement**

Attention !
Patron de Conception =/= Idiotisme de Programmation

ORIGINE

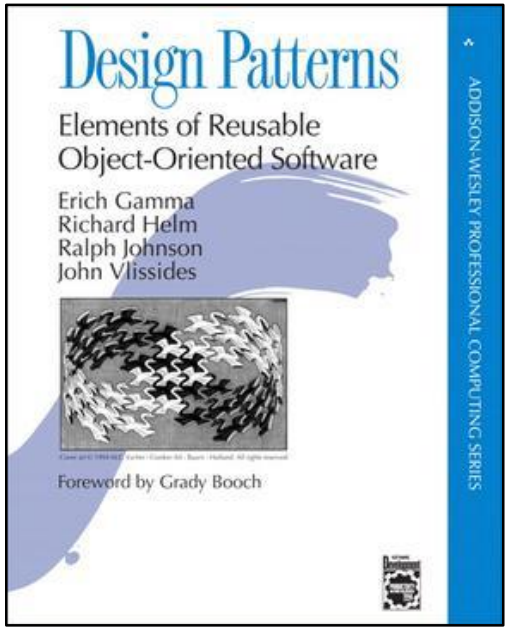
- Concept de **patron/pattern d'architecture** inventé par **Christopher Alexander** (1977)
- Repris par **Kent Beck** et **Ward Cunningham**



ORIGINE

Auteurs de "*Design Patterns – Elements of Reusable Object-Oriented Software*" (1994) :

- Erich Gamma
- Ralph Johnson
- John Vlissides
- Richard Helms



DESCRIPTION OBSERVATEUR

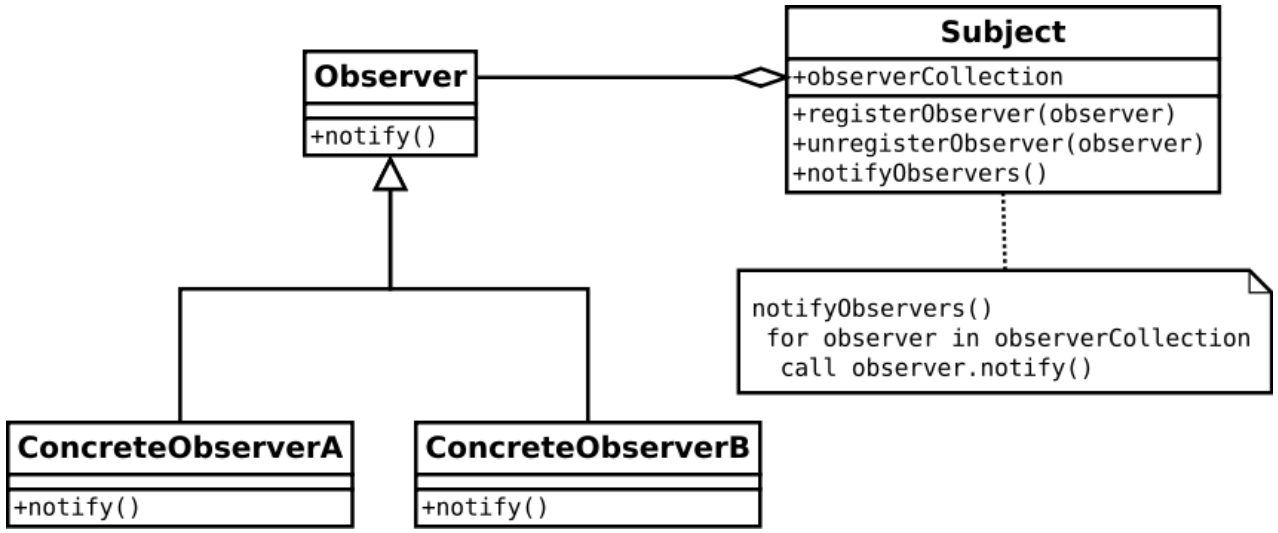
- Pattern de **comportement**
- Utilisé quand une classe **déclenche des actions** dans d'autres
- Permet aussi de **limiter** le couplage entre les modules

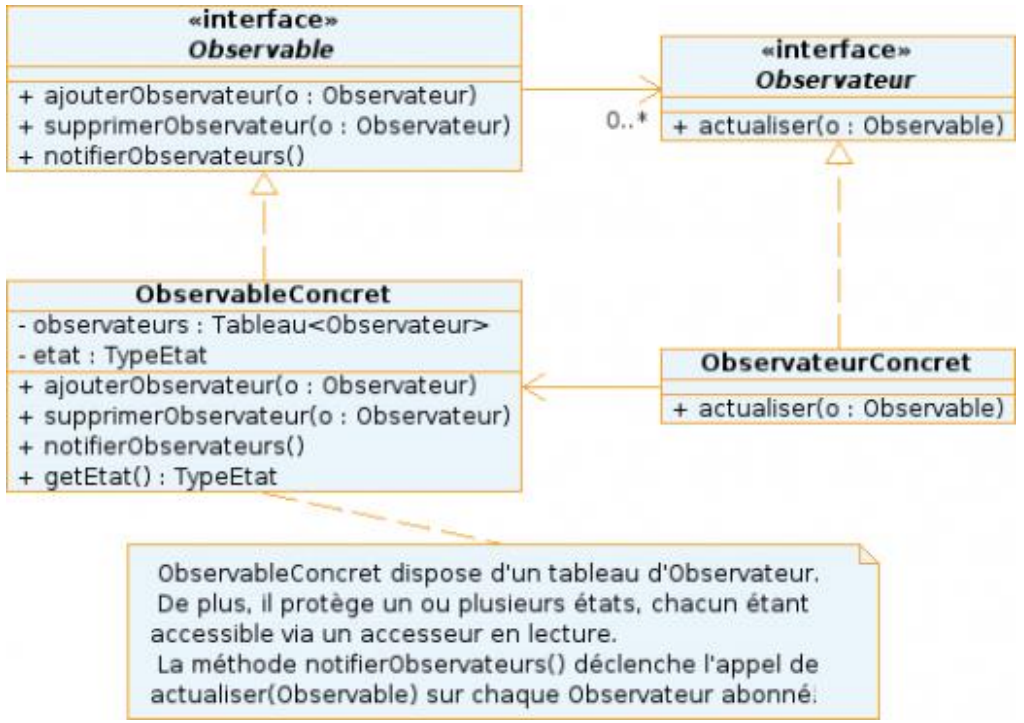
DESCRIPTION OBSERVATEUR

Que faire lorsqu'on souhaite qu'une classe A fasse une action après une **modification** d'une classe B ?

- Soit A **demande** à B si ses attributs ont changé,
- Soit B **informe** A que ses attributs ont changé.

DESCRIPTION OBSERVATEUR





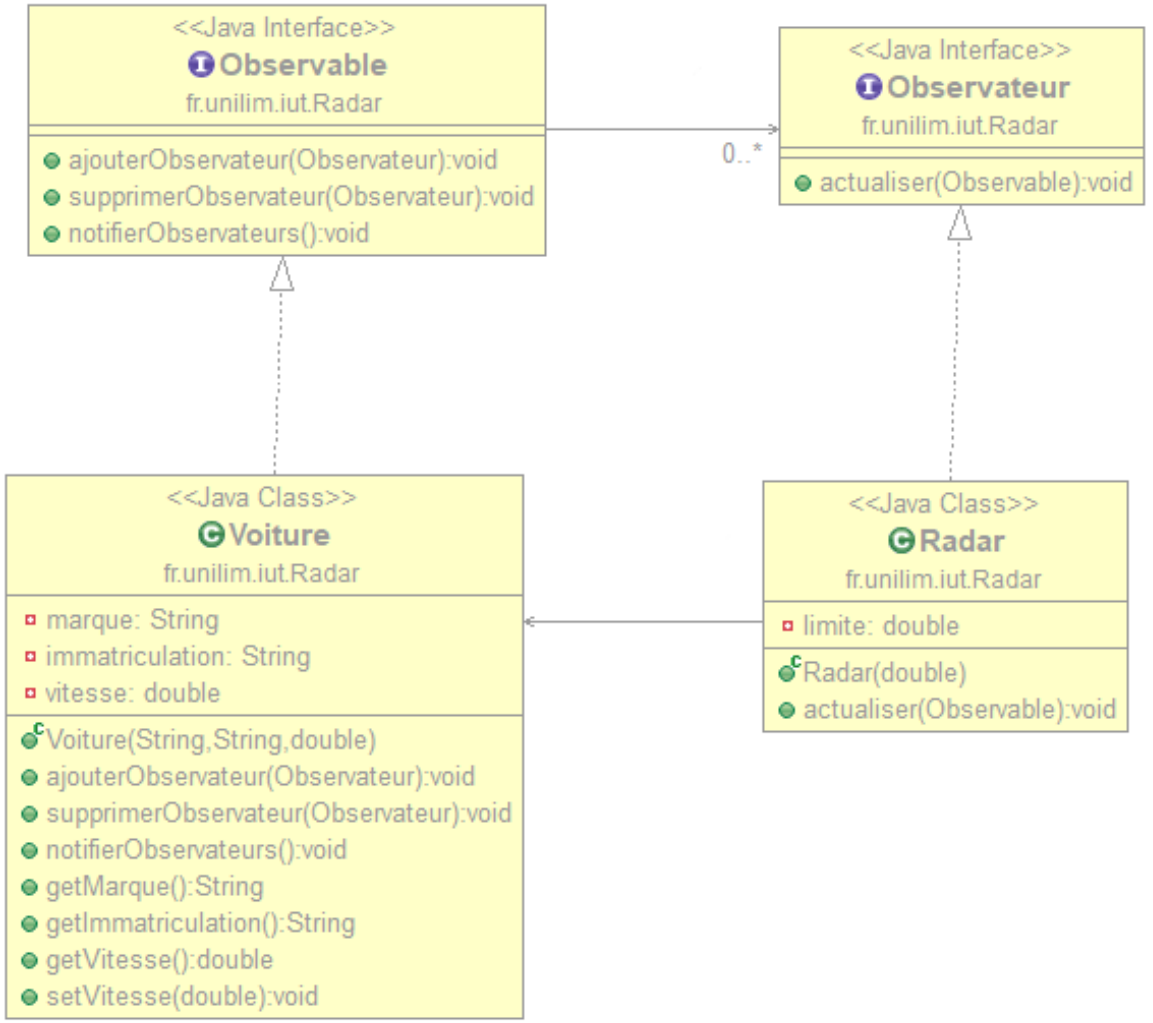
DESCRIPTION OBSERVATEUR

SOLID ?

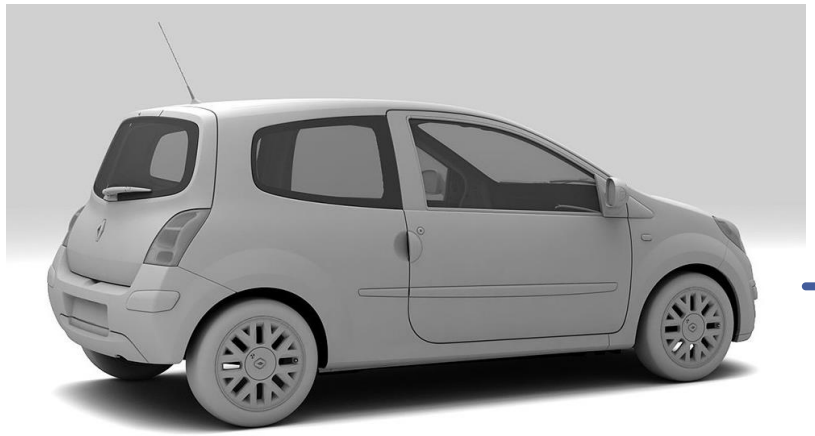
- Respect de **OCP** et **DIP**
- Violation de **SRP** (et **ISP** selon l'implémentation)

EXEMPLE 1 : LE RADAR

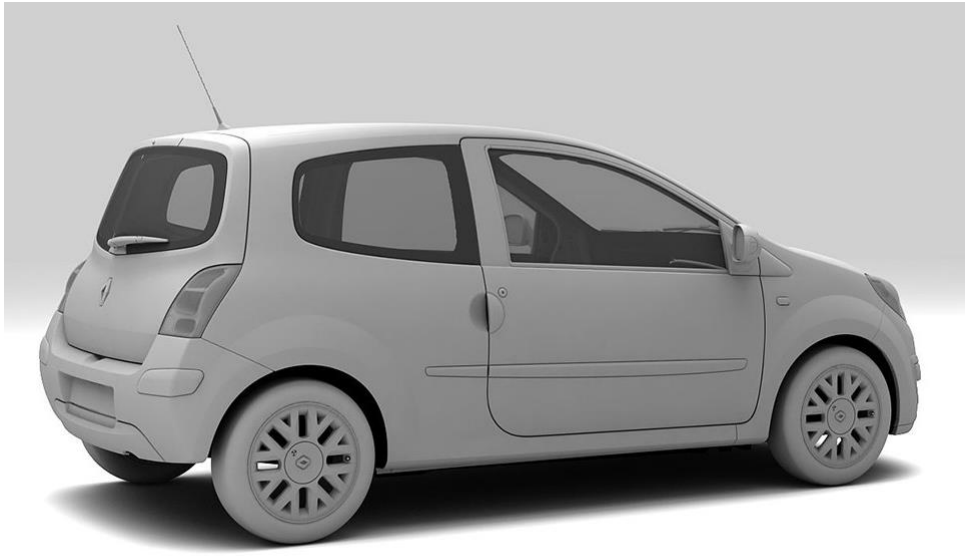
EXEMPLE 1 : LE RADAR



EXEMPLE 1 : LE RADAR



EXEMPLE : LE RADAR



Marque : Renault
Immatriculation : BQ-615-EE
Vitesse actuel : 45 km/h

Observateur = Radar

```java
public Voiture(String marque, String immatriculation, double vitesse) {
    this.marque = marque;
    this.immatriculation = immatriculation;
    this.vitesse = vitesse;
    listeObservateur = new ArrayList<Observateur>();
}
```

```java
public void ajouterObservateur(Observateur observeur) {
    listeObservateur.add(observeur);
    notifierObservateurs();
}

public void supprimerObservateur(Observateur observeur) {
    listeObservateur.remove(observeur);
}
```

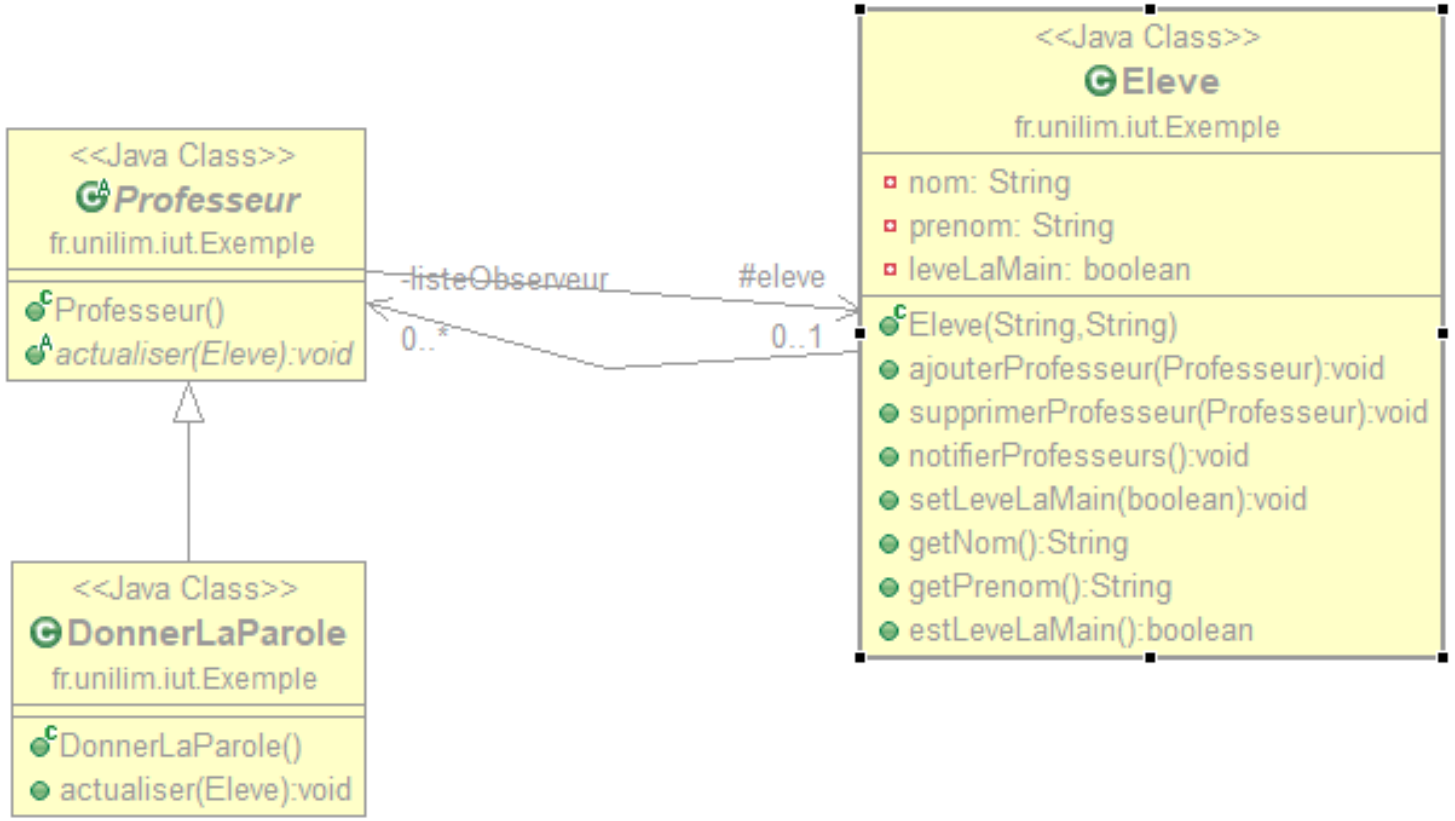
EXEMPLE 1 : LE RADAR

```java
public void notifierObservateurs() {
    if(listeObservateur.size() > 0) {
        for(int i=0; i < listeObservateur.size(); i++)
        {
            listeObservateur.get(i).actualiser(this);
        }
    }
    else {
        System.out.println("Voiture hors de porté !");
    }
}
```

```java
public void setVitesse(double vitesse) {
    this.vitesse = vitesse;
    notifierObservateurs();
}
```

EXEMPLE 2 : LE PROFESSEUR

EXEMPLE 2 : LE PROFESSEUR



EXEMPLE 2 : LE PROFESSEUR

```java
public Eleve(String prenom, String nom) {
    this.prenom = prenom;
    this.nom = nom;
    this.leveLaMain = false;
    listeObserveur = new ArrayList<Professeur>();
}
```

```java
public void ajouterProfesseur(Professeur observeur) {
    listeObserveur.add(observeur);
}

public void supprimerProfesseur(Professeur observeur) {
    listeObserveur.remove(observeur);
}

public void notifierProfesseurs() {
    for(int i=0; i < listeObserveur.size(); i++)
    {
        listeObserveur.get(i).actualiser(this);
    }
}
```

Nom : Dupuy
Prenom : Rémi
Lève la main : Non

EXEMPLE 2 : LE PROFESSEUR

```java
public void setLeveLaMain(boolean leveLaMain) {
    this.leveLaMain = leveLaMain;
    notifierProfesseurs();
}
```

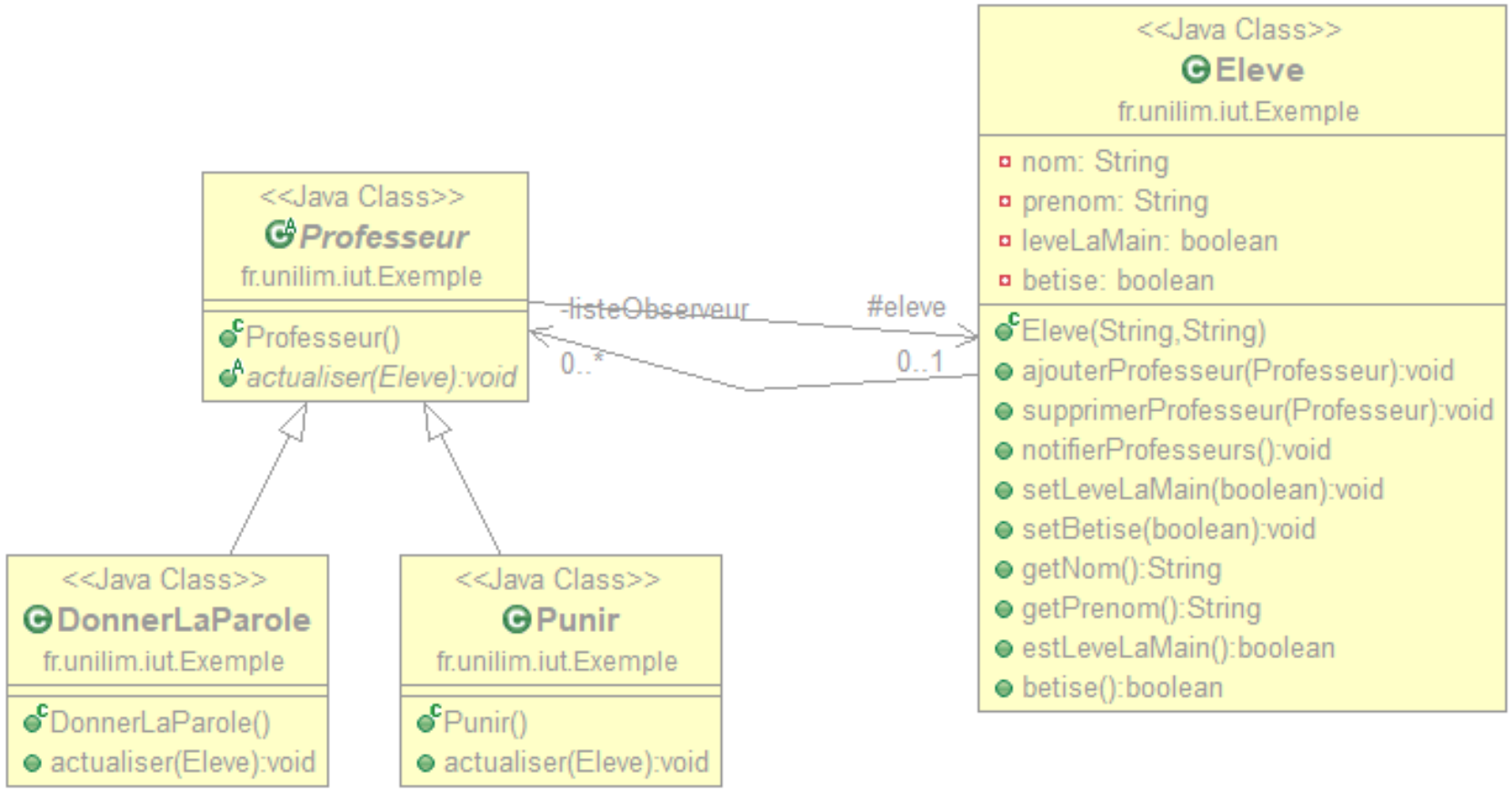
```java
public class DonnerLaParole extends Professeur {
    public void actualiser(Eleve eleve)
    {
        if(eleve.estLeveLaMain()) {
            System.out.println(eleve.getPrenom() + " peut répondre à la question !");
        }
        else {
            System.out.println(eleve.getPrenom() + " a répondu à la question !");
        }
    }
}
```

EXEMPLE 2 : LE PROFESSEUR

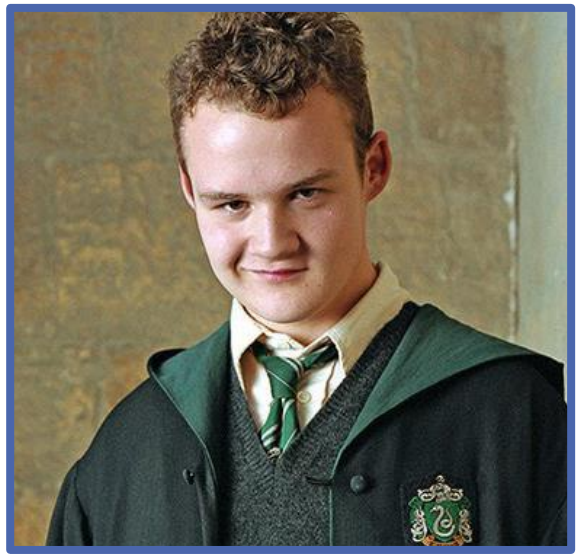
```
Eleve Christian = new Eleve("Christian", "Dupont");
Eleve Remi = new Eleve("Rémi", "Dupuy");
DonnerLaParole donnerLaParole = new DonnerLaParole();

Remi.ajouterProfesseur(donnerLaParole);

System.out.println("Question par le professeur");
System.out.println();

Remi.setLeveLaMain(true);
System.out.println();

Remi.setLeveLaMain(false);
System.out.println();
```

```
Question par le professeur

Rémi peut répondre à la question !

Rémi a répondu à la question !
```

EXEMPLE 2 : LE PROFESSEUR



EXEMPLE 2 : LE PROFESSEUR



Nom : Dupont
Prénom : Christian
A fait une bêtise : Non

```
public class Punir extends Professeur {
    public void actualiser(Eleve eleve)
    {
        if(eleve.betise()) {
            System.out.println("La professeur punit " + eleve.getPrenom() + " " + eleve.getNom());
        }
        else {
            System.out.println("La professeur lève la punition de  " + eleve.getPrenom() + " " + eleve.getNom());
        }
    }
}
```

EXEMPLE 2 : LE PROFESSEUR

```java
Eleve Christian = new Eleve("Christian", "Dupont");
Eleve Remi = new Eleve("Rémi", "Dupuy");

Punir punir = new Punir();

Christian.ajouterProfesseur(punir);

System.out.println(Christian.getPrenom() + " jette une gomme !");
System.out.println();

Christian.setBetise(true);
System.out.println();

System.out.println(Christian.getPrenom() + " s'excuse auprès du professeur");
System.out.println();

Christian.setBetise(false);
System.out.println();

System.out.println(Remi.getPrenom() + " jette une gomme !");

Remi.setBetise(true);
System.out.println("La professeur n'a rien vue");
```

```
Christian jette une gomme !

La professeur punit Christian Dupont

Christian s'excuse auprès du professeur

La professeur lève la punition de  Christian Dupont

Rémi jette une gomme !
La professeur n'a rien vue
```

CONCLUSION