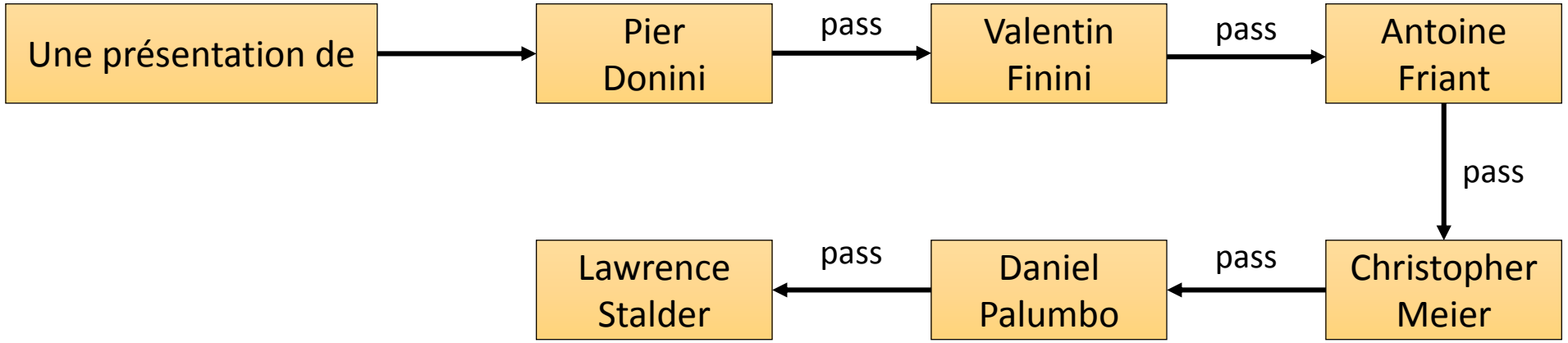
Chaine de responsabilité



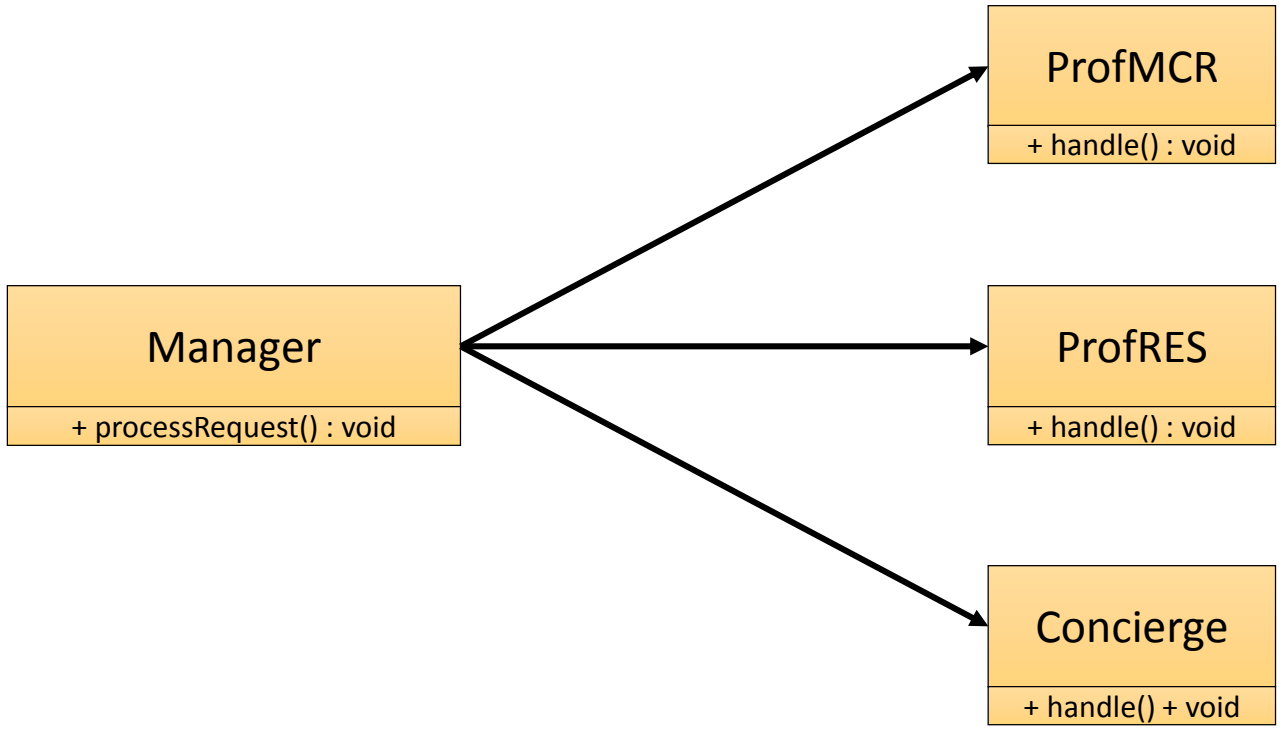
Rôle, Domaine et Intention

- GoF -> [Objet, Comportemental]
- Éviter d'associer l'expéditeur d'une requête à son récepteur en donnant plus d'un objet pour traiter la demande.
- Chaîner les objets récepteurs et passer la requête le long de la chaîne jusqu'à ce qu'un objet la traite.

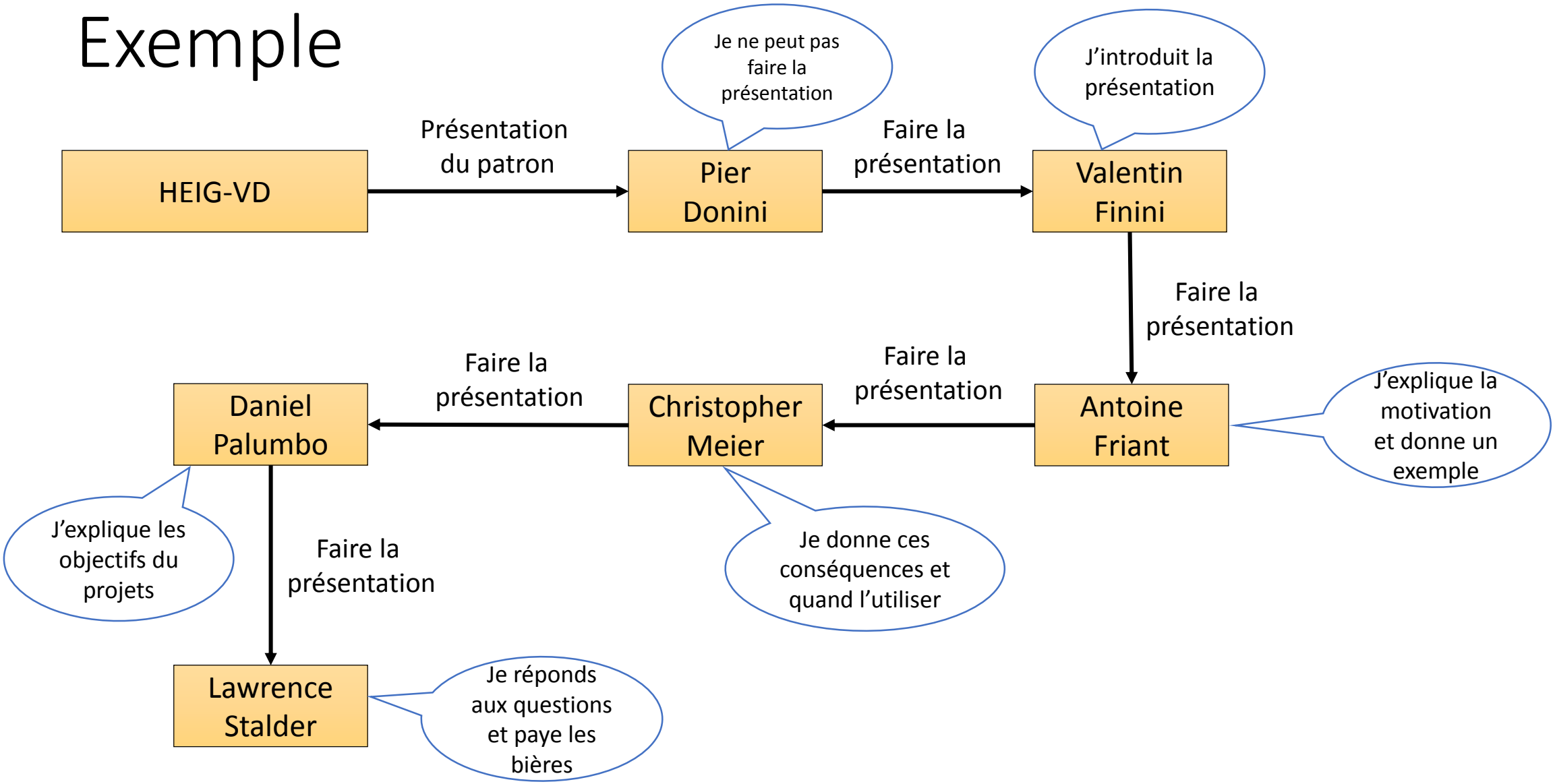
Motivation

- L’objet «L’HEIG-VD» fait la requête qu’une présentation sur le modèle «chaîne de responsabilité» doit être créée et présentée à une classe.
- La requête est traitée par l’un des nombreux objets «professeurs», mais qui dépend du contexte et de la spécificité de la requête.
- Le problème est que «HEIG-VD» ne sait pas quel «professeur» va traiter la requête.

Approche intuitive



Exemple


HEIG-VD
Présentation du patron
Pier Donini
Je ne peut pas faire la présentation
Faire la présentation
Valentin Finini
J'introduit la présentation
Faire la présentation
Antoine Friant
J'explique la motivation et donne un exemple
Faire la présentation
Christopher Meier
Je donne ces conséquences et quand l'utiliser
Faire la présentation
Daniel Palumbo
J'explique les objectifs du projets
Faire la présentation
Lawrence Stalder
Je réponds aux questions et paye les bières

Conséquences

- Réduction du couplage
  - L'expéditeur ne sait pas quel objet va traiter la requête
  - Un objet de la chaîne ne connaît pas la structure de la chaîne
- Simplification de l'interconnexion entre objets
  - Nul besoin de maintenir des références dans tous les sens.
  - 1 seul référenceur au successeur.
- Flexibilité dans l'assignation de responsabilités
  - La structure de la chaîne et la responsabilité donnée au membre de la chaîne peuvent être modifiées durant l'exécution.
- Le traitement n'est pas garantie
  - Aucun objet peut traiter la requête
  - La chaîne est mal configurée

A utiliser

- Lorsque plus d’un objet pourrait traiter la requête et que le récepteur n’est pas connu.
- Si l’on veut envoyer une requête à plusieurs objets sans spécifier le récepteur.
- Lorsque l’ensemble d’objets qui pourrait traiter la requête peut être spécifier dynamiquement.
- Ne pas confondre délégation et chaine de responsabilité

Exemple d'implémentation (0)

```
public class Currency {
	private int amount;
	public Currency(int amt){
		this.amount=amt;
	}
	public int getAmount(){
		return this.amount;
	}
}

public interface DispenseChain {
	void setNextChain(DispenseChain nextChain);
	void dispense(Currency cur);
}
```

Exemple d'implémentation (2)

```
public class Dollar50Dispenser implements DispenseChain {
    private DispenseChain chain;
    @Override
    public void setNextChain(DispenseChain nextChain) {
        this.chain=nextChain;
    }
    @Override
    public void dispense(Currency cur) {
        if(cur.getAmount() >= 50){
            int num = cur.getAmount()/50;
            int remainder = cur.getAmount() % 50;
            System.out.println("Dispensing "+num+" 50$ note");
            if(remainder !=0) this.chain.dispense(new Currency(remainder));
        }else{
            this.chain.dispense(cur);
        }
    }
}
```

Exemple d'implémentation (3)

```
public class Dollar20Dispenser implements DispenseChain{
    private DispenseChain chain;
    @Override
    public void setNextChain(DispenseChain nextChain) {
        this.chain = nextChain;
    }
    @Override
    public void dispense(Currency cur) {
        if(cur.getAmount() >= 20){
            int num = cur.getAmount()/20;
            int remainder = cur.getAmount() % 20;
            System.out.println("Dispensing "+num+" 20$ note");
            if(remainder !=0) { this.chain.dispense(new Currency(remainder)); }
        }else{
            this.chain.dispense(cur);
        }
    }
}
```

Exemple d'implémentation (4)

```
public class ATMDispenseChain {

    private DispenseChain c1;

    public ATMDispenseChain() {
        // initialize the chain
        this.c1 = new Dollar50Dispenser();
        DispenseChain c2 = new Dollar20Dispenser();
        DispenseChain c3 = new Dollar10Dispenser();

        // set the chain of responsibility
        c1.setNextChain(c2);
        c2.setNextChain(c3);
    }

    public static void main(String[] args) {
        /* some code to make it work */
    }
}
```