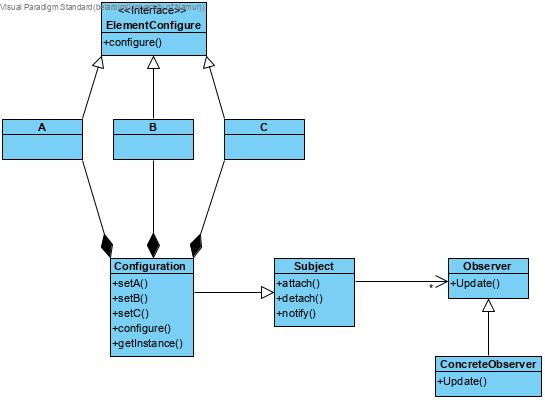
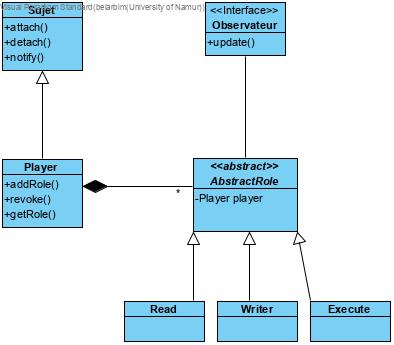
Design pattern

# Exercice 2.15



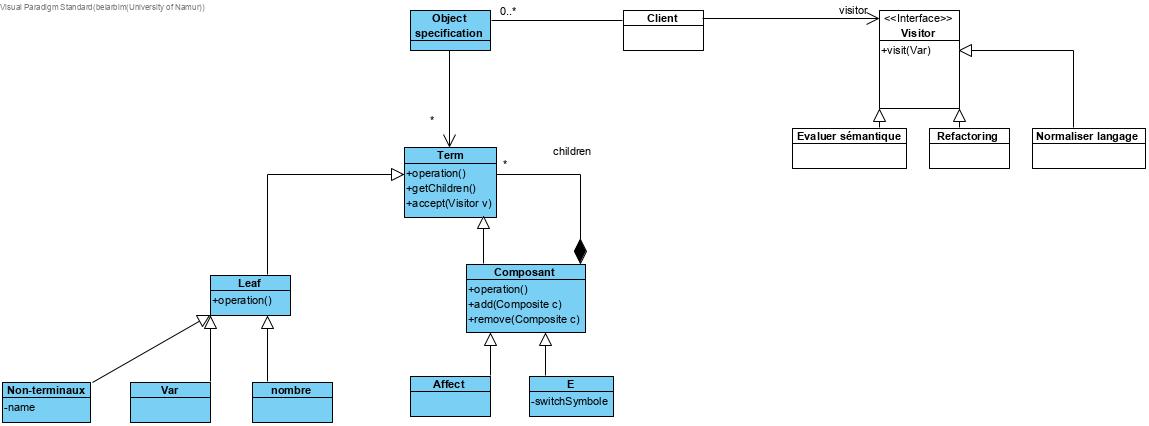
# Exercice 2.18

1. Oui si on n’en aura plus besoin
2. Dans la classe Player une liste<Role> et dans la classe Role un attribut de type Player player.
3. Un Player aura accès à toutes les rôles et si on veut rajouter un nouveau rôle il faut modifier la classe. Tant on a de rôle l’ajout et la modification seront compliqués.
4. Oui, à partir du Player on ne fournit pas des méthodes Business mais il contient uniquement addRole, removeRole…
5. Solution est de mettre des clauses assert qui vérifieront que rôle a une référence vers un Player. Si cette dernière est nulle on interdit l’usage.

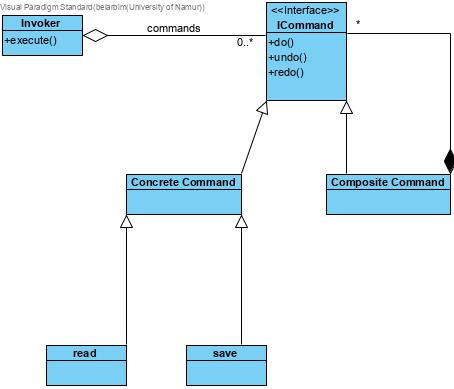


# Exercice 2.21

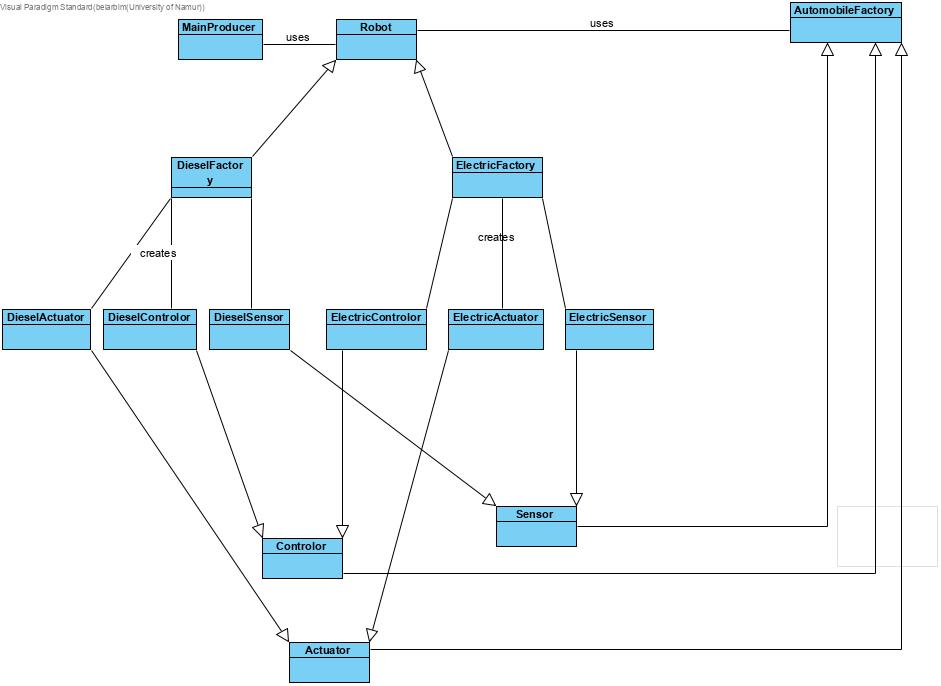
Il faut rajouter toutes les méthodes visit(Var), visit(nombre), visit(E)



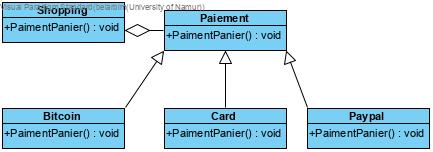
# Exercice 2.31



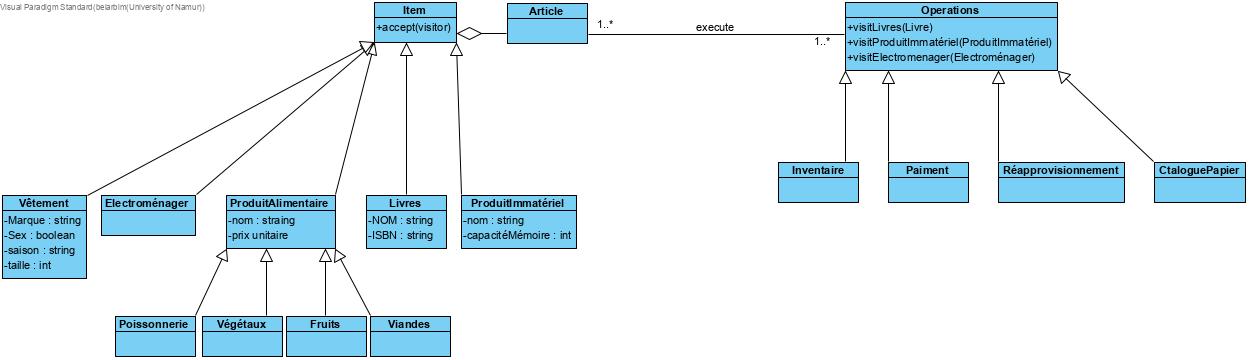
# Exercice 2.38.1

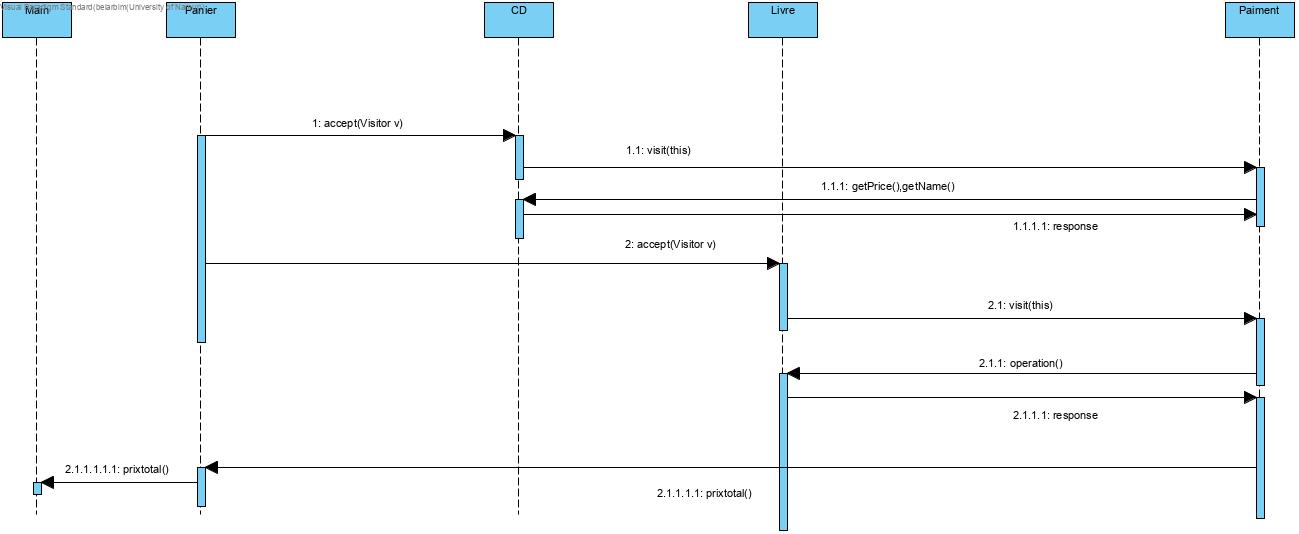


# Exercice 2.38.2

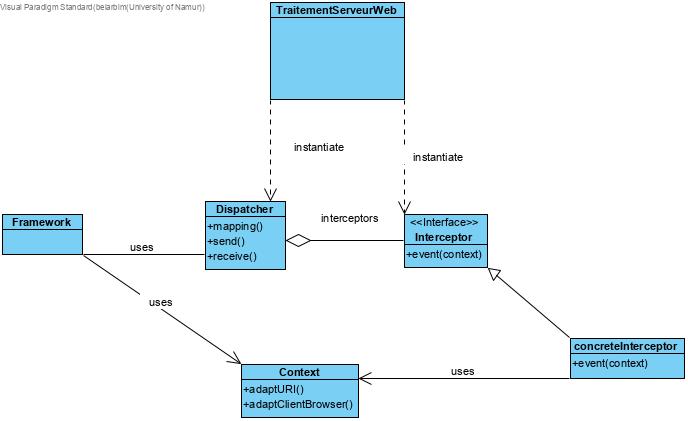


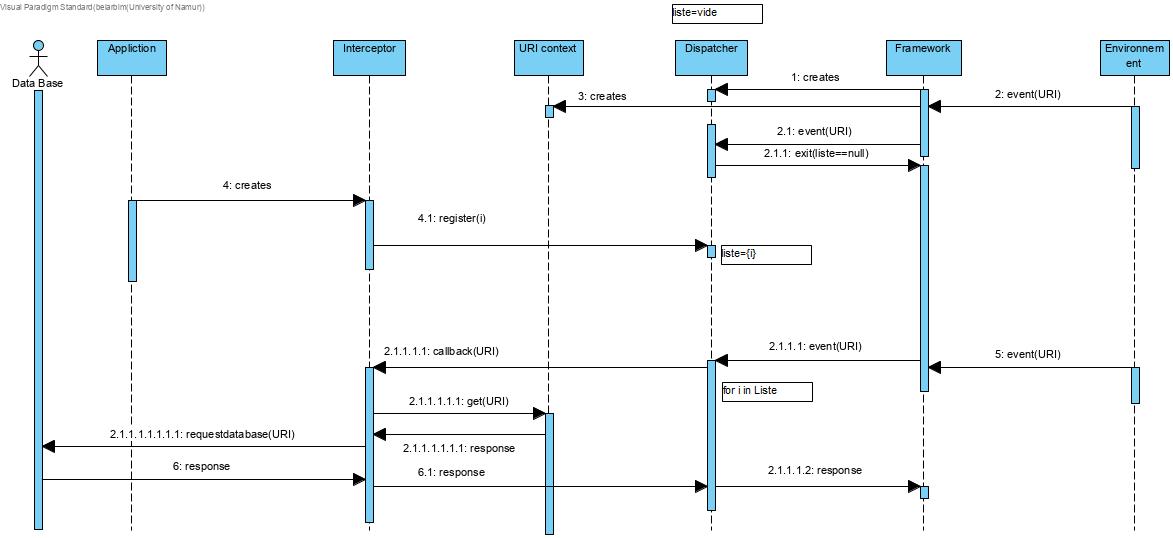
# Exercice 2.38.3



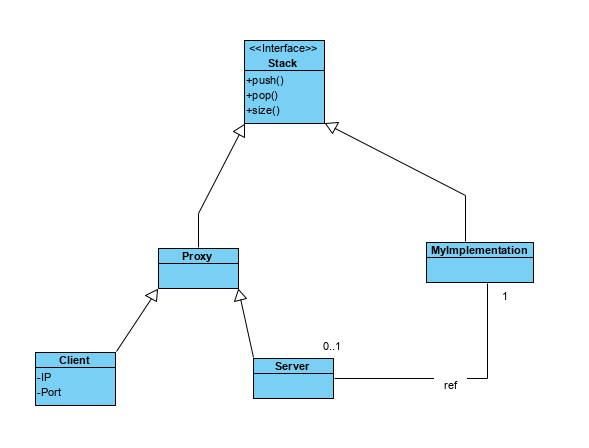


# Exercice 2.38.4





# Exercice 2.39



**Exemple :**

p=new Proxy(1.1.1.1, 9999, "coucou")

p.pop()

{ ouvrir connexion vers 1.1.1.1 port 999

send( "call pop with arguments ... on object named 'coucou')

answer=receive();

}

# Exercice 2.39.1

