

# Travaux Dirigés No3

## Conception par Contrats

Frédéric Peschanski

9 février 2017

Dans ce TD nous mettons en pratique la conception par contrat à partir des spécifications semi-formelles élaborées lors du TD précédent.

### Exercice 1 : Compte bancaire

Décrire une interface Java `Compte` pour le service de compte bancaire du TD2 (cf. annexes).

Indiquer également les invariants et les préconditions/postconditions sur les méthodes (contrats).

Décrire une implémentation des contrats séparant le code métier (`CompteImpl`) du code de contractualisation (`CompteContract`). Indiquer le *design pattern* employé.

### Exercice 2 : Agence bancaire

Mêmes questions pour le service Agence

### Annexe A : Spécifications du compte

```
service : Compte
observers :
  const nom : [Compte] → String
  const numero : [Compte] → int
  solde : [Compte] → double
  const limite : [Compte] → double
  montantDecouvert : [Compte] → double
    pre montantDecouvert(C) require estDecouvert(C)
  estDecouvert : [Compte] → boolean
  peutPrelever : [Compte] × double → boolean
    pre peutPrelever(C,s) require s > 0
Constructors :
  init : String × int × double → [Compte]
    pre init(n,num,dec) require n ≠ "" ∧ num > 0 ∧ dec ≥ 0
  init : Compte → [Compte]
Operators :
  depot : [Compte] × double → [Compte]
    pre depot(C,s) require s > 0

  retrait : [Compte] × double → [Compte]
    pre depot(C,s) require peutPrelever(C,s)
Observations :
```

```

[invariants]
  montantDecouvert(C)  $\stackrel{\text{min}}{=}$  -solde(C)
  estDecouvert(C)  $\stackrel{\text{min}}{=}$  solde(C) < 0
  peutPrelever(C,s)  $\stackrel{\text{min}}{=}$  solde(C)-s  $\geq$  limite(C)
  solde(C)  $\geq$  limite(C) // invariant utile
[init]
  nom(init(n,num,dec)) = n
  numero(init(n,num,dec)) = num
  solde(init(n,num,dec)) = 0
  limite(init(n,num,dec)) = - dec
  nom(init(C)) = nom(C)
  numero(init(C)) = numero(C)
  solde(init(C)) = solde(C)
  limite(init(C)) = limite(C)
[depot]
  solde(depot(C,s)) = solde(C) + s
[retrait]
  solde(retrait(C,s)) = solde(C) - s

```

## Annexe B : Spécifications de l'agence

```

service : Agence
observers :
  const nom : [Agence]  $\rightarrow$  String
  numeros : [Agence]  $\rightarrow$  Set<int>
  nbComptes : [Agence]  $\rightarrow$  int
  compteExiste : [Agence]  $\times$  int  $\rightarrow$  bool
  compte : [Agence]  $\times$  int  $\rightarrow$  Compte
  pre compte(A,num) require compteExiste(A,num)
Constructors :
  init : String  $\rightarrow$  [Agence]
  pre init(n) require n  $\neq$  ""
Operators :
  creerCompte : [Agence]  $\times$  String  $\times$  int  $\times$  double  $\rightarrow$  [Agence]
  pre creerCompte(A,n,num,dec) require  $\neg$ compteExiste(A,num)
  fermerCompte : [Agence]  $\times$  int  $\rightarrow$  [Agence]
  pre fermerCompte(A,num) require compteExiste(A,num)
  virement : [Agence]  $\times$  int  $\times$  int  $\times$  double  $\rightarrow$  [Agence]
  pre virement(A,n1,n2,s) require n1  $\neq$  n2  $\wedge$  compteExiste(A,n1)  $\wedge$  compteExiste(A,n2)
   $\wedge$  Compte::peutPrelever(compte(A,n1),s)
Observations :
[invariants]
  nbComptes(A)  $\stackrel{\text{min}}{=}$  card(numeros(A))
  compteExiste(A,num)  $\stackrel{\text{min}}{=}$  num  $\in$  numeros(A)
[init]
  nom(init(n)) = n
  numeros(init(n)) =  $\emptyset$ 
[creerCompte]
  numeros(creerCompte(A,n,num,dec)) = numeros(A)  $\cup$  { num }
  compte(creerCompte(A,n,num,dec),num) = Compte::init(n,num,dec)
  pour tout n  $\in$  numeros(A), compte(creerCompte(A,n,num,dec),n) = compte(A)
[fermerCompte]
  numeros(fermerCompte(A,num)) = numeros(A)  $\setminus$  { num }
  pour tout n  $\in$  numeros(A)  $\setminus$  { num }, compte(fermerCompte(A,num,dec),n) = compte(A)
[virement]
  numeros(virement(A,n1,n2,s)) = numeros(A)
  compte(virement(A,n1,n2,s),n1) = Compte::retrait(compte(A,n1),s)
  compte(virement(A,n1,n2,s),n2) = Compte::depot(compte(A,n2),s)
   $\forall n \in$  numeros(A)  $\setminus$  { n1,n2 }, compte(virement(A,n1,n2,s),n) = compte(A)

```