Corrigé des exercices sur les références

Exercice 3.1.1 *dessin*

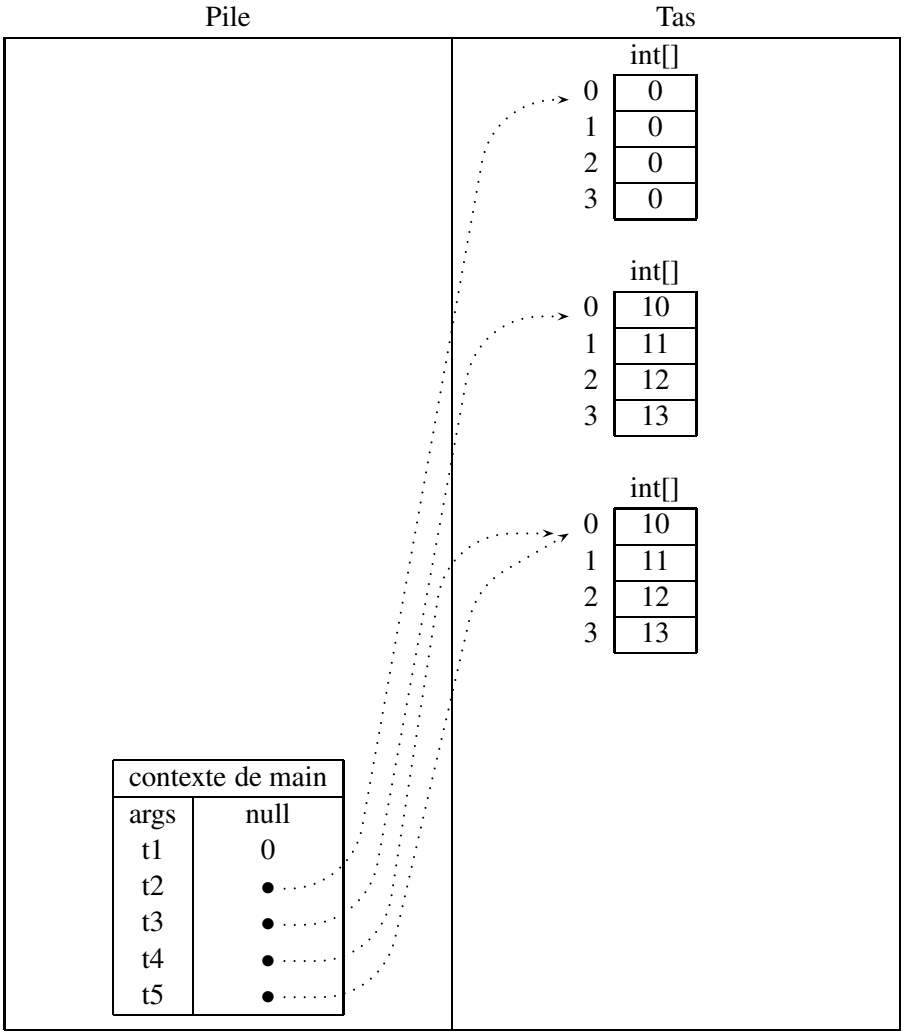
Pour cet exercice, vous allez dessiner des structures de données au moyen de petits schémas analogues à ceux du cours, comportant la pile et le tas. Les références pourront être représentées par des flèches ou des adresses.

Dans ce corrigé, on représentera les références par des flèches.

Question 1 *tableaux*

Dessinez l'état de la mémoire à la fin de l'exécution de ce programme.

```
class Exo_16_1_1{
    public static void main(String[] args){
        int[] t1, t2, t3, t4, t5;
        t2 = new int[4];
        t3 = new int[4];
        t4 = new int[4];
        for (int i=0; i<t3.length; i++){
            t3[i] = 10+i;
            t4[i] = 10+i;
        }
        t5 = t4;
    }
}
```

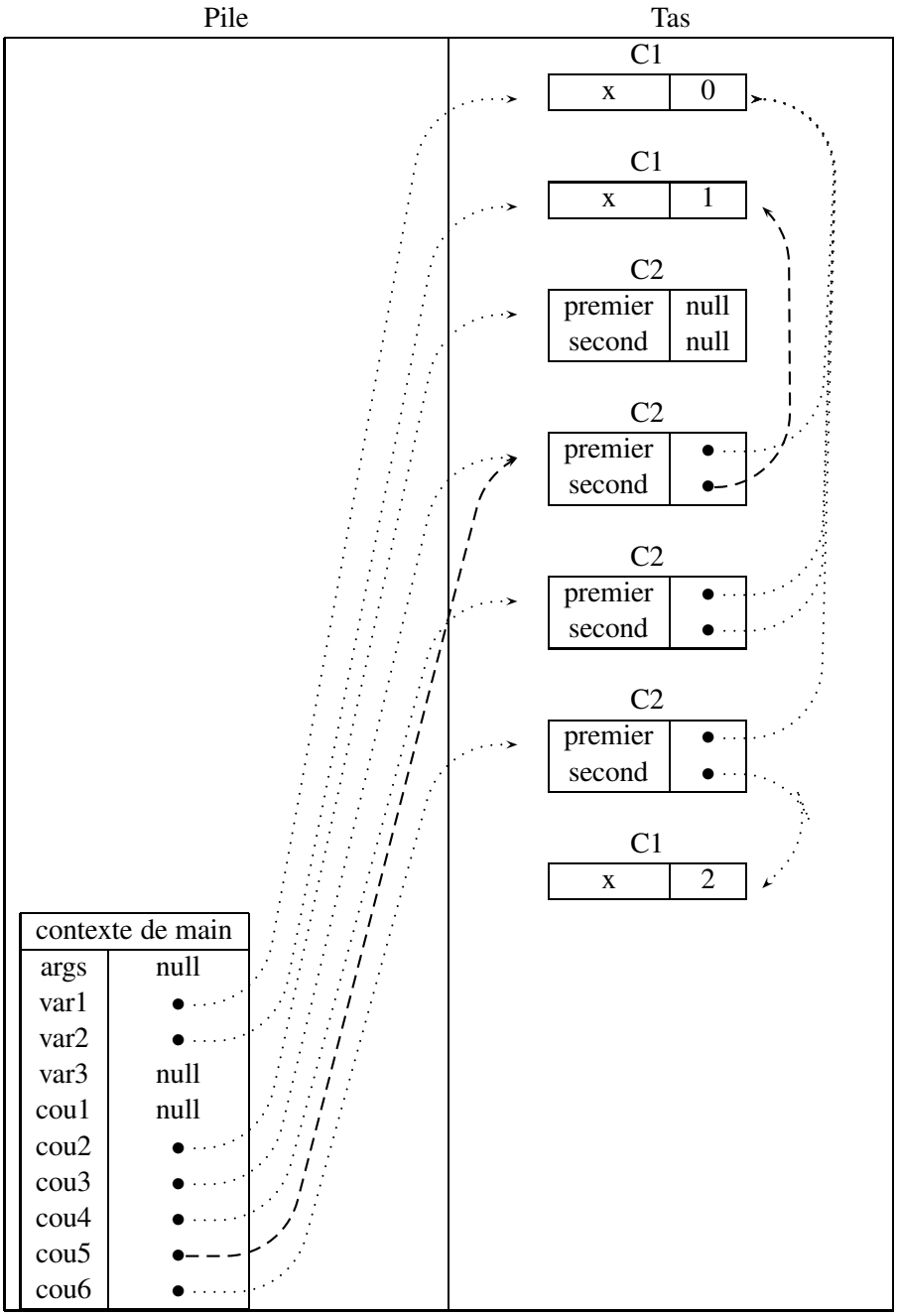


Question 2 *objets*

Même question pour les variables du programme suivant (var1, var2, var3, cou1, cou2, cou3, cou4, cou5, cou6.

```
class Exo_16_1_2{
    public static void main(String[] args){
        C1 var1, var2, var3;
        C2 cou1, cou2, cou3, cou4, cou5, cou6;
        var1 = new C1(0);
        var2 = new C1(1);
        cou2 = new C2();
        cou3 = new C2();
        cou4 = new C2();
        cou6 = new C2();
        cou3.premier = var1;
        cou3.second = var2;
        cou4.premier = var1;
        cou4.second = var1;
```

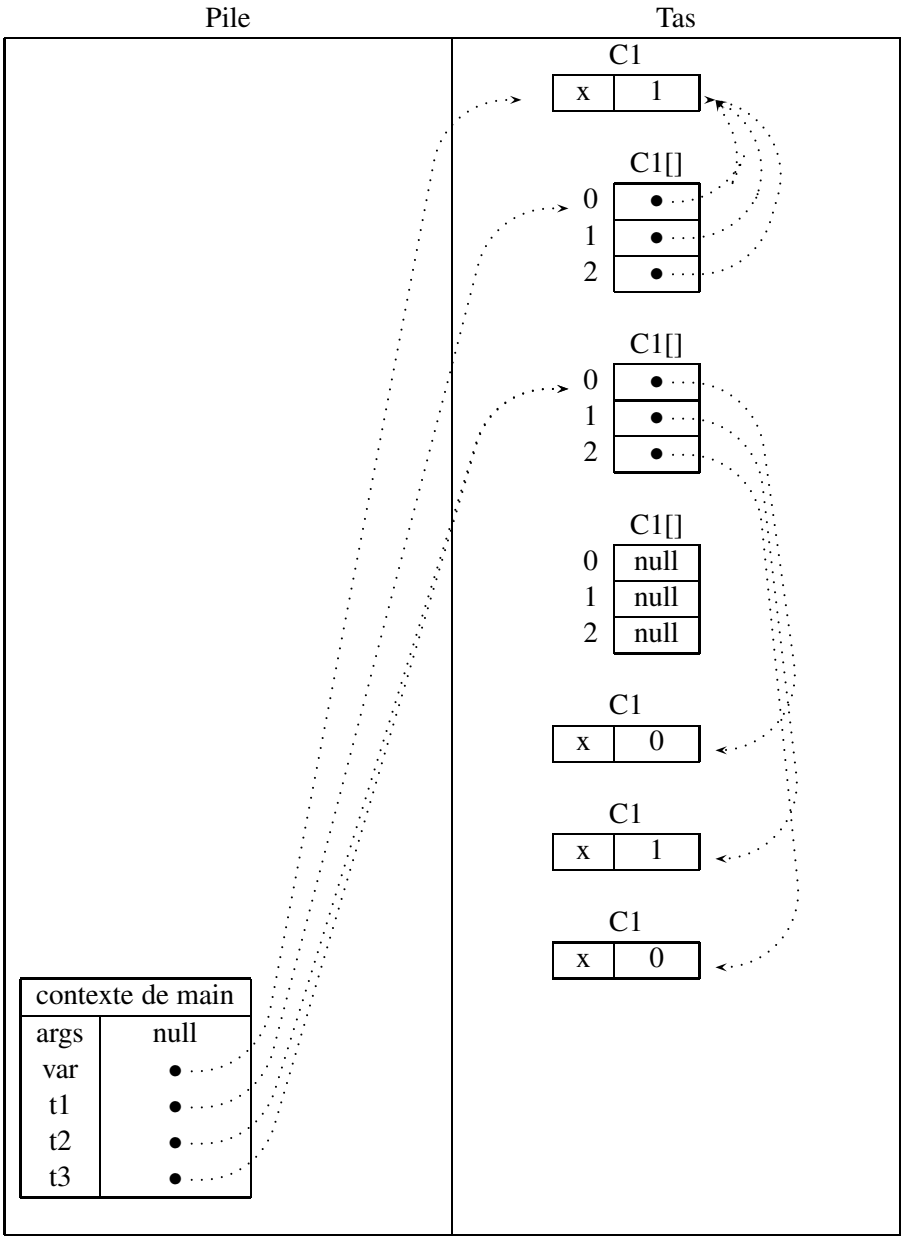
```
        cou5 = cou3;
        cou6.premier = var1;
        cou6.second = new C1(2);
    }
}
class C1{
    int x;
    C1(int i){
        x=i;
    }
}
class C2{
    C1 premier, second;
}
```



Question 3 ***tableaux d'objets***

```
class Exo_16_1_3{
```

```
    public static void main(String[] args){
        C1[] t1, t2, t3;
        C1 var = new C1(0);
        t1=new C1[3];
        t2=new C1[3];
        t3=new C1[3];
        for (int i=0; i<t1.length; i++){
            t1[i] = var;
            t2[i] = new C1(0);
        }
        t1[1].x = 1;
        t2[1].x = 1;
        t3 = t2;
    }
}
```

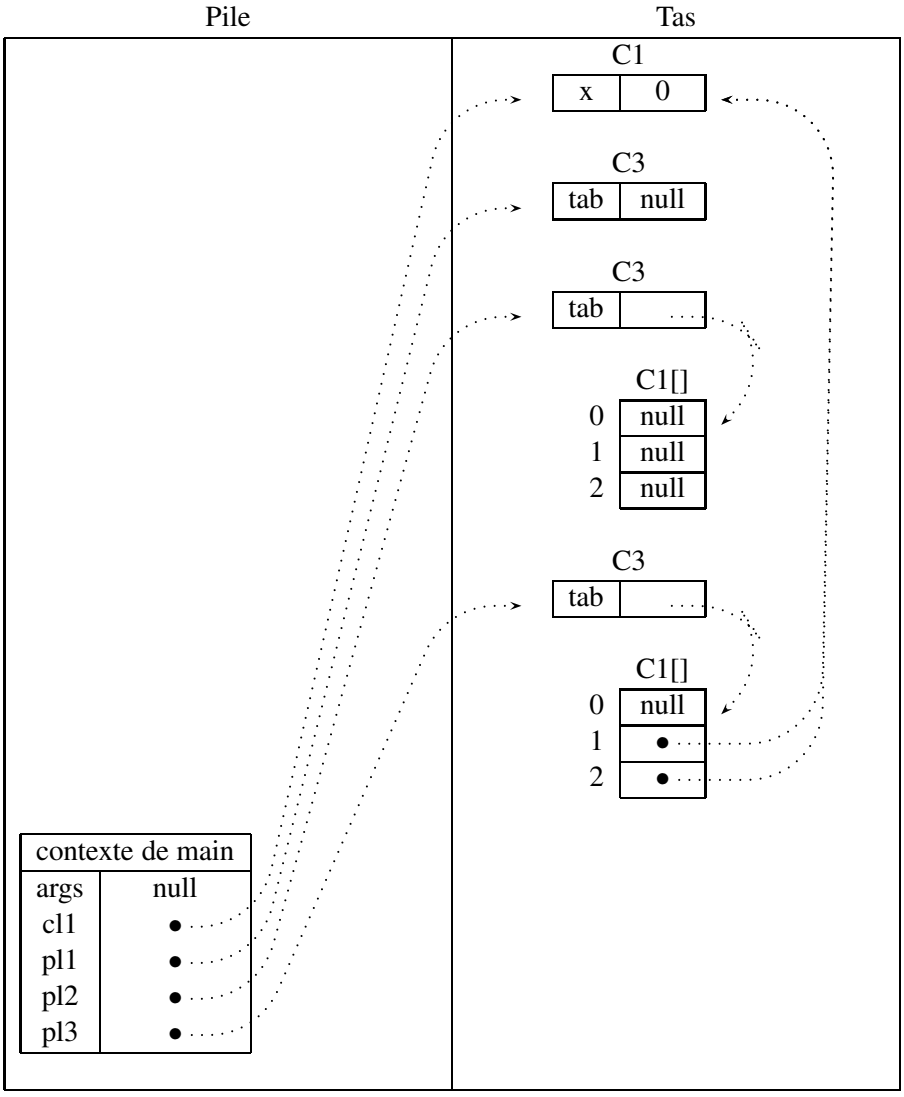


Note : le tableau contenant des valeurs `null` est celui qui est créé par la ligne `t3=new C1[3];` Ce tableau n'est plus accessible depuis la pile (car il y a eu une nouvelle affactation à `t3`) et ne sert plus à rien. Il peut être supprimé à tout moment par le gestionnaire de mémoire de l'interpréteur Java (Runtime).

Question 4 *objets contenant des tableaux*

```
class Exo_16_1_4{
    public static void main(String[] args){
        C3 pl1, pl2, pl3;
```

```
        C1 cl1;
        cl1 = new C1(0);
        pl1 = new C3();
        pl2 = new C3();
        pl2.tab = new C1[3];
        pl3 = new C3();
        pl3.tab = new C1[3];
        pl3.tab[1] = cl1;
        pl3.tab[2] = cl1;
    }
}
class C3{
    C1[] tab;
}
```



Exercice 3.1.2 *comptes bancaires*

On va représenter des comptes bancaires et des titulaires de comptes. Un même compte peut avoir plusieurs titulaires (c'est le cas par exemple des comptes joints pour un couple). Dans ce cas, le même objet sera partagé entre les titulaires. Un titutlaire peut avoir plusieurs comptes.

On vous propose le squelette de classe suivant pour implémenter les comptes et les titulaires.

```
class TableauCompte{
    Compte[] tab;
    int longueur;
    TableauCompte(int n){
        tab = new Compte[n];
    }
    void ajouter(Compte c) throws NonInitialise{
        if (c == null){
            throw new NonInitialise();
        }
        if (longueur < tab.length){
            tab[longueur]=c;
            longueur++;
        }
    }
}
class Banque{
    String nom;
    TableauCompte tous = new TableauCompte(50);
    Banque(String n){
        nom = n;
    }
}
class Titulaire{
    String nom;
    Titulaire(String n){
        nom = n;
    }
    TableauCompte mesComptes = new TableauCompte(10);
}
class Compte{
    int numero;
    int solde;
    void depot(int n){
        solde = solde + n;
    }
    void retrait(int n){
        solde = solde -n;
    }
    void afficher(){
        Terminal.ecrireString("solde␣du␣compte␣numero␣" + numero + ":␣");
        Terminal.ecrireInt(solde);
    }
}
class NonInitialise extends Exception{ }
```

Question 1 *création de comptes*

Ajouter à ce squelette de classes des méthodes. Seule la banque peut créer des nouveaux comptes. Quand elle crée un nouveau compte, elle l'ajoute à la liste des comptes de chacun des titulaires de ce compte et elle l'ajoute aussi à sa propre liste de comptes. Par exemple, si une banque `b` crée un compte `c` pour les titulaires `t1` et `t2`, le même objet c sera ajouté à trois tableaux : `b.tous`, `t1.mesComptes` et `t2.mesComptes`. Dans chacun de ces trois tableaux, il y aura la même adresse, celle de l'objet `c`.

Pour réaliser cette création de comptes, il faut une méthode dans Banque qui crée un compte avec en paramètre la liste des titulaires (par exemple sous forme de tableau). Il peut également être utile d'écrire une méthode permettant d'ajouter un nouveau compte chez un titulaire.

Question 2 *variables*

Ecrivez une méthode `main` qui réalise une situation où trois titulaires, Paul, Pierre et Fatima ont des comptes dans une banque *BNP*. Paul et Fatima ont un compte joint. Fatima a en plus un compte personnel, de même que Pierre.

Dessinez la situation en faisant apparaître sur un schéma tous les objets et tableaux.

Question 3 *virements*

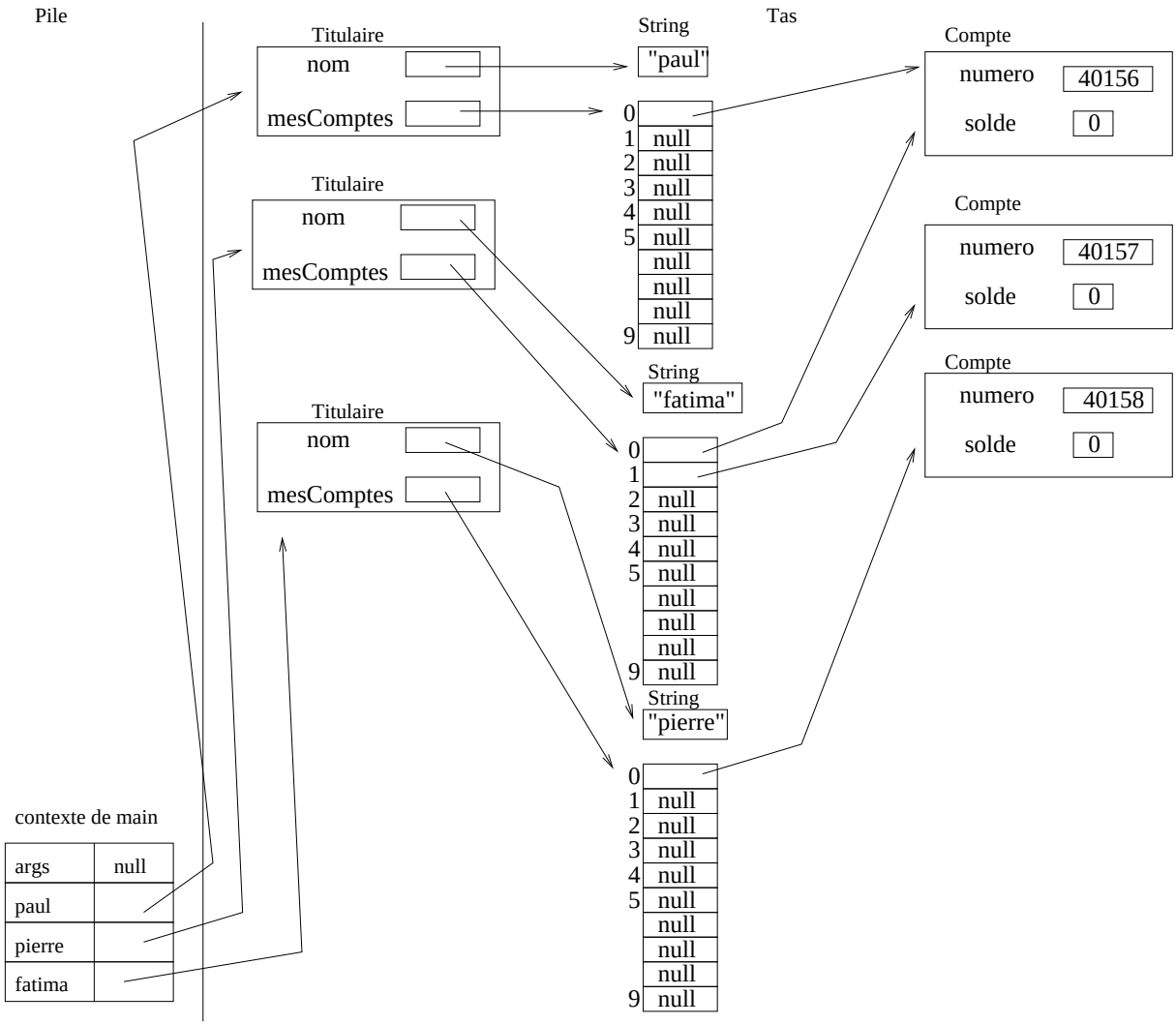
On désire écrire une méthode qui permette de faire un virement depuis un compte dont on est titulaire vers un compte dont on ne connaît que le numéro (et dont on n'est pas forcément titulaire) et la banque (autrement dit, on ne connaît pas son adresse). Dans quelle classe faut-il ajouter cette méthode ? Ecrivez cette méthode.

Cette méthode de virement ne peut être qu'une méthode de la classe banque puisque c'est elle seulement qui peut retrouver un compte à partir de son numéro. Plus précisément, comme c'est dans la variable `tous` de la classe `Banque` que l'on doit chercher le compte dont le numéro est donné, il est logique de mettre cette méthode de virement dans la même classe.

```
class TableauCompte{
    Compte[] tab;
    int longueur;
    TableauCompte(int n){
        tab = new Compte[n];
    }
    void ajouter(Compte c) throws NonInitialise{
        if (c == null){
            throw new NonInitialise();
        }
        if (longueur < tab.length){
            tab[longueur]=c;
            longueur++;
        }
    }
    void affiche(){
        for (int i = 0; i<longueur; i++){
            tab[i].affiche();
        }
    }
}
```

```
class Banque{
    String nom;
    int prochainNumero = 40156;
    TableauCompte tous = new TableauCompte(50);
    Banque(String n){
        nom = n;
    }
    void creerCompte(Titulaire[] tab){
        Compte c = new Compte(prochainNumero);
        prochainNumero++;
        tous.ajouter(c);
        for (int i=0; i<tab.length; i++){
            tab[i].ajouterCompte(c);
        }
    }
    void affiche(){
        Terminal.ecrireStringln("Comptes de la banque " + nom);
        tous.affiche();
    }
    void virement(Compte from, int numeroTo, int montant){
        int i = 0;
        while(true){
            if (i == tous.longueur){
                break;
            }
            if (tous.tab[i].numero == numeroTo){
                break;
            }
            i++;
        };
        if (i<tous.longueur){
            from.retrait(montant);
            tous.tab[i].depot(montant);
        }
    }
}
class Titulaire{
    String nom;
    Titulaire(String n){
        nom = n;
    }
    TableauCompte mesComptes = new TableauCompte(10);
    void ajouterCompte(Compte c){
        mesComptes.ajouter(c);
    }
    void affiche(){
        Terminal.ecrireStringln("Comptes de " + nom);
        mesComptes.affiche();
    }
}
class Compte{
    int numero;
    int solde;
```

```
    Compte(int n){
        numero = n;
        solde = 0;
    }
    void depot(int n){
        solde = solde + n;
    }
    void retrait(int n){
        solde = solde -n;
    }
    void affiche(){
        Terminal.ecrireString("solde␣du␣compte␣numero␣" + numero + ":␣");
        Terminal.ecrireIntln(solde);
    }
}
class NonInitialise extends Error{}
class Exo16_2{
    public static void main(String[] args){
        Banque bnp = new Banque("BNP");
        Titulaire paul = new Titulaire("Paul");
        Titulaire pierre = new Titulaire("Pierre");
        Titulaire fatima = new Titulaire("Fatima");
        Titulaire[] tab1 = {paul, fatima};
        Titulaire[] tab2 = {fatima};
        Titulaire[] tab3 = {pierre};
        bnp.creerCompte(tab1);
        bnp.creerCompte(tab2);
        bnp.creerCompte(tab3);
        bnp.affiche();
        paul.affiche();
        pierre.affiche();
        fatima.affiche();
    }
}
```



Exercice 3.1.3 *Procurations*

```
class TableauCompte{
    Compte[] tab;
    int longueur;
    TableauCompte(int n){
        tab = new Compte[n];
    }
    void ajouter(Compte c) throws NonInitialise{
        if (c == null){
            throw new NonInitialise();
        }
        if (longueur < tab.length){
            tab[longueur]=c;
            longueur++;
        }
    }
```

```
    void affiche(){
        for (int i = 0; i<longueur; i++){
            tab[i].affiche();
        }
    }
    boolean appartient(Compte c){
        for (int i =0; i< longueur; i++){
            if (c == tab[i]){
                return true;
            }
        }
        return false;
    }
    Compte acceder(int i) throws NExistePas{
        if (i<longueur){
            return tab[i];
        }else{
            throw new NExistePas();
        }
    }
}
class Banque{
    String nom;
    int prochainNumero = 40156;
    TableauCompte tous = new TableauCompte(50);
    Banque(String n){
        nom = n;
    }
    void creerCompte(Titulaire[] tab){
        Compte c = new Compte(prochainNumero);
        prochainNumero++;
        tous.ajouter(c);
        for (int i=0; i<tab.length; i++){
            tab[i].ajouterCompte(c);
        }
    }
    void affiche(){
        Terminal.ecrireStringln("Comptes de la banque " + nom);
        tous.affiche();
    }
    void virement(Compte from, int numeroTo, int montant){
        int i = 0;
        while(true){
            if (i == tous.longueur){
                break;
            }
            if (tous.tab[i].numero == numeroTo){
                break;
            }
            i++;
        };
        if (i<tous.longueur){
            from.retrait(montant);
```

```
                tous.tab[i].depot(montant);
            }
        }
}
class Titulaire{
    String nom;
    Titulaire(String n){
        nom = n;
    }
    TableauCompte mesComptes = new TableauCompte(10);
    TableauCompte procurations = new TableauCompte(10);
    void ajouterCompte(Compte c){
        mesComptes.ajouter(c);
    }
    void affiche(){
        Terminal.ecrireStringln("Comptes de " + nom);
        mesComptes.affiche();
        Terminal.ecrireStringln("Procurations de " + nom);
        procurations.affiche();
    }
    void recevoirProcurationDe(Titulaire t, Compte c)
        throws PasTitulaire{
        if (t.mesComptes.appartient(c)){
            procurations.ajouter(c);
        }else{
            throw new PasTitulaire();
        }
    }
}
class Compte{
    int numero;
    int solde;
    Compte(int n){
        numero = n;
        solde = 0;
    }
    void depot(int n){
        solde = solde + n;
    }
    void retrait(int n){
        solde = solde -n;
    }
    void affiche(){
        Terminal.ecrireString("solde du compte numero " + numero + ": ");
        Terminal.ecrireIntln(solde);
    }
}
class NonInitialise extends Error{}
class PasTitulaire extends Exception{}
class NExistePas extends Exception{}
public class Exo_17_1{
    public static void main(String[] args){
        Banque bnp = new Banque("BNP");
```

```
        Titulaire paul = new Titulaire("Paul");
        Titulaire pierre = new Titulaire("Pierre");
        Titulaire fatima = new Titulaire("Fatima");
        Titulaire[] tab1 = {paul, fatima};
        Titulaire[] tab2 = {fatima};
        Titulaire[] tab3 = {pierre};
        bnp.creerCompte(tab1);
        bnp.creerCompte(tab2);
        bnp.creerCompte(tab3);
        bnp.affiche();
        paul.affiche();
        pierre.affiche();
        fatima.affiche();
        try{
            pierre.recevoirProcurationDe(fatima,fatima.mesComptes.acceder(0));
            pierre.affiche();
            paul.recevoirProcurationDe(pierre,fatima.mesComptes.acceder(0));
        }catch(PasTitulaire e){
            Terminal.ecrireStringln("Exception PasTitulaire");
        }catch(NExistePas e){
            Terminal.ecrireStringln("Exception NExistePas");
        }
    }
}
```

Commentaire : le sujet proposé commence à devenir complexe. Le programme donné ici n'est pas très satisfaisant en terme de protection des comptes bancaires. L'idée de base est que seul le titulaire connaît les comptes dont il est titulaire. Mais en fait, on a facilement accès aux comptes d'une personne grâce à la variable `mesComptes` et à la méthode `acceder`. Pour aller plus loin, il faut protéger la variable et/ou la méthode avec un mode d'accès `private` ou `protected`. Ce sujet sera abordé avec la notion de paquetage.