Corrigés des exercices sur l'héritage

Exercice 6.1.1 *comptes bancaires*

On reprend les classes de comptes bancaires du cours (chapitre 16) : `CompteBancaire`, `CompteAvecDecouve` `CompteRemunere`, `CompteRemunereAvecDecouvert`.

Voici un morceau de code avec des inconnues :

```
class Exo14_1{
  XXX c1 = new YYY;
  ZZZ c2 = new TTT;
  c1.depot(50.0);
  c2.depot(50.0);
  c2.calculInteret();
}
```

Donnez toutes les combinaisons de classes possibles pour XXX, YYY, ZZZ et TTT (vous ne vous préoccuperez pas des paramètres des constructeurs pour YYY et TTT).

Considérons séparément les cas de `c1` et `c2`. Pour `c1`, il faut une classe ayant la méthode `depot`. Les quatre classes considérées ont toutes les quatre cette méthode, soit directement, soit par héritage. N'importe laquelle des quatre classes convient pour XXX. YYY peut être la même classe que XXX ou une de ses sous-classes, car on peut toujours affecter un objet instance d'une sous-classe de C à une variable déclaré avec un certain type C.

Les combinaisons possibles sont :

XXX	YYY
CompteBancaire	CompteBancaire
CompteBancaire	CompteAvecDecouvert
CompteBancaire	CompteRemunere
CompteBancaire	CompteRemunereAvecDecouvert
CompteAvecDecouvert	CompteAvecDecouvert
CompteRemunere	CompteRemunere
CompteRemunere	CompteRemunereAvecDecouvert
CompteRemunereAvecDecouvert	CompteRemunereAvecDecouvert

Pour ce qui est de `c2`, il faut que l'objet possède les deux méthodes `depot` et `calculInteret`. Deux classes ont ces deux méthodes : `CompteRemunere` et `CompteRemunereAvecDecouvert`.

Les trois combinaisons possibles sont :

ZZZ	TTT
CompteRemunere	CompteRemunere
CompteRemunere	CompteRemunereAvecDecouvert
CompteRemunereAvecDecouvert	CompteRemunereAvecDecouvert

Exercice 6.1.2

On définit comme suit une classe de personnes :

```
class LaDate {
    int jour;
    int mois;
    int annee;
    // ================================================
    // l'année en cours ...
    static int CETTE_ANNE = 2005;

    public LaDate (int j, int m, int a){
        this.jour=j;
        this.mois = m;
        this.annee = a;
    }
    public void ecrire() {
         Terminal.ecrireString(jour+"/"+mois+"/"+annee);
    }
}
class LaPersonne {
    LaDate naissance;
    String nom;

    public LaPersonne(String name, LaDate naiss) {
         this. nom = name;
         this.naissance = naiss;
    }

    public void ecrire() {
         Terminal.ecrireString(nom+" (");
         naissance.ecrire();
         Terminal.ecrireString(")");
    }
}
```

Pour certaines personnes, on connaît en plus le lieu de naissance, mais par pour toutes.

Ecrire une sous-classe des personnes dont on connaît le lieu de naissance. Redéfinir la méthode `ecrire` pour que le lieu de naissance s'affiche.

Ecrivez un petit programme qui utilise la nouvelle sous-classe.

```
class LaDate {
    int jour;
    int mois;
    int annee;
    // ================================================
    // l'année en cours ...
    static int CETTE_ANNE = 2005;

    public LaDate (int j, int m, int a){
        this.jour=j;
        this.mois = m;
```

```
        this.annee = a;
    }
    public void ecrire() {
        Terminal.ecrireString(jour+"/"+mois+"/"+annee);
    }
}
class LaPersonne {
    LaDate naissance;
    String nom;

    public LaPersonne(String name, LaDate naiss) {
        this. nom = name;
        this.naissance = naiss;
    }

    public void ecrire() {
        Terminal.ecrireString(nom+" (");
        naissance.ecrire();
        Terminal.ecrireString(")");
    }
}
class PersonneAvecLieu extends LaPersonne{
    String lieuNaissance;
    PersonneAvecLieu(String name, LaDate naiss, String lieu){
        super(name,naiss);
        lieuNaissance = lieu;
    }
    public void ecrire() {
        Terminal.ecrireString(nom+" (");
        naissance.ecrire();
        Terminal.ecrireString("," + lieuNaissance + ")");
        Terminal.sautDeLigne();
    }
}
class Exo18_2{
    public static void main(String[] args){
        PersonneAvecLieu pal =
            new PersonneAvecLieu("Thierry",
                                 new LaDate(15,6,2004),
                                 "Toulon");
        pal.ecrire();
    }
}
```

Exercice 6.1.3 *des employés*

Une entreprise a un certain nombre d'employés. Un employé est connu par son nom, son matricule (qui l'identifie de façon unique) et son indice salairial. Le salaire est calculé en multipliant cet indice par une certaine valeur qui peut changer en cas d'augmentation générale des salaires, mais qui est la même pour tous les employés.

Question 1

Ecrivez la classe des employés avec les informations utiles et des méthodes pour afficher les caractéristiques d'un employé et pour calculer son salaire.

Question 2

Certains employés ont des responsabilités hiérarchiques. Ils ont sous leurs ordres d'autres employés. Ecrivez une sous-classe des employés qui représente ces responsables en enregistrant leurs inférieurs hiérarchiques directs dans un tableau.

Ecrivez une méthode qui affiche les inférieurs directs (placés directement sous leurs ordres) et une autre qui affiche les employés inférieurs directs ou indirects (c'est à dire les subordonnés des subordonnés). On suppose que la hiérarchie de l'entreprise en pyramidale.

Question 3

Les commerciaux ont un salaire composé d'un fixe et d'un intéressement proportionnel à leurs ventes. Ecrivez une sous-classe des commerciaux qui contient l'information sur leurs ventes du dernier mois, une méthode pour mettre à jour cette information et redéfinissez la méthode de calcul de leurs salaires.

Question 4

Ecrivez une classe représentant tout le personnel de l'entreprise, avec une méthode calculant la somme des salaires à verser.

```
class Employe2{
    String nom;
    int matricule;
    int indiceSalarial;
    static int valeur = 12;
    Employe2(String n, int m, int i){
        nom = n;
        matricule = m;
        indiceSalarial = i;
    }
    void ecrire(){
        Terminal.ecrireString(nom + "␣" + matricule + "␣" + indiceSalarial);
    }
    int salaire(){
        return indiceSalarial *valeur;
    }
}
class Responsable extends Employe2{
    Employe2[] subordonnes;
    String titre;
    Responsable(String n, int m, int i, Employe2[] t, String ti){
        super(n,m,i);
        subordonnes = t;
        titre = ti;
    }
```

```
    void afficheSubordonnesDirects(){
        this.ecrire();
        Terminal.ecrireStringln("␣subordonnés:␣");
        for (int i = 0; i<subordonnes.length; i++){
            Terminal.ecrireString("␣␣␣");
            subordonnes[i].ecrire();
            Terminal.sautDeLigne();
        }
    }
    void ecrire(){
        Terminal.ecrireString(nom + "␣" + matricule + "␣" + indiceSalarial);
        Terminal.ecrireString("␣" + titre);
    }
}
class Commercial extends Employe2{
    int venteDuMois;
    Commercial(String n, int m, int i){
        super(n,m,i);
    }
    void enregistreVentes(int i){
        venteDuMois = i;
    }
    int salaire(){
        return (indiceSalarial *valeur)+(venteDuMois/10);
    }
}
class Personnel{
    Employe2[] tabEmp = new Employe2[100];
    int nbEmp = 0;
    Responsable[] tabResp = new Responsable[10];
    int nbResp = 0;
    Employe2 chercherEmploye(int matr){
        for (int i=0; i<nbEmp; i++){
            if (tabEmp[i].matricule == matr){
                return tabEmp[i];
            }
        }
        return null;
    }
    void ajouterEmploye(Employe2 e){
        if (chercherEmploye(e.matricule) == null){
            tabEmp[nbEmp] = e;
            nbEmp++;
        }
    }
    void ajouterResponsable(Responsable r){
        if (chercherEmploye(r.matricule) == null){
            ajouterEmploye(r);
            tabResp[nbResp] = r;
            nbResp++;
        }
    }
    void affichePersonnel(){
```

```
        for (int i=0; i<nbEmp; i++){
            tabEmp[i].ecrire();
            Terminal.sautDeLigne();
        }
    }
    void afficheHierarchie(){
        for (int i=0; i<nbResp; i++){
            tabResp[i].afficheSubordonnesDirects();
            Terminal.ecrireStringln("====================================");
        }
    }
    int totalSalaires(){
        int somme = 0;
        for (int i=0; i<nbEmp; i++){
            somme = somme + tabEmp[i].salaire();
        }
        return somme;
    }
}
class Exo18_3{
    public static void main(String[] args){
        Commercial c1, c2, c3;
        c1 = new Commercial("Jean",120,12);
        c1.enregistreVentes(1200);
        c2 = new Commercial("Alberto",121,12);
        c2.enregistreVentes(1100);
        c3 = new Commercial("John",122,17);
        c3.enregistreVentes(700);
        Employe2[] t1 = {c1, c2, c3};
        Responsable r1, r2, r3;
        r1 = new Responsable("Luis",125,20,t1,"directeur commercial");
        Employe2 e1, e2, e3;
        e1 = new Employe2("Jerzy",17,15);
        e2 = new Employe2("Ivan",19,14);
        e3 = new Employe2("Joao",21,14);
        Employe2[] t2 = {e1, e2, e3};
        r2 = new Responsable("Helmut",77,21,t2,"directeur technique");
        Employe2[] t3 = {r1, r2};
        r3 = new Responsable("Jan",301,30,t3,"directeur");
        Personnel pers = new Personnel();
        pers.ajouterEmploye(c1);
        pers.ajouterEmploye(c2);
        pers.ajouterEmploye(c3);
        pers.ajouterEmploye(e1);
        pers.ajouterEmploye(e2);
        pers.ajouterEmploye(e3);
        pers.ajouterResponsable(r1);
        pers.ajouterResponsable(r2);
        pers.ajouterResponsable(r3);
        pers.affichePersonnel();
        Terminal.sautDeLigne();
        Terminal.sautDeLigne();
        pers.afficheHierarchie();
```

```
        Terminal.sautDeLigne();
        Terminal.sautDeLigne();
        Terminal.ecrireIntln(pers.totalSalaires());
    }
}
```