MOOC Intro POO Java
Corriges semaine 2

Les corrigés proposés correspondent à l'ordre des apprentissages : chaque corrigé correspond à la solution à laquelle vous pourriez aboutir au moyen des connaissances acquises jusqu'à la semaine correspondante.

Exercice 4 : poeme (Constructeurs)

Voici une solution possible (il en existe d'autres !) :

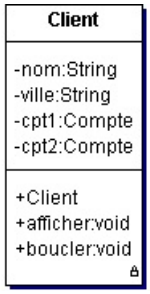
```
class Fleur {
    private String espece;
    private String couleur;

    public  Fleur(String espece, String couleur) {
            this.espece = espece;
            this.couleur = couleur;
            System.out.println(espece + " fraichement cueillie");
    }

    public Fleur(Fleur autre) {
            couleur = autre.couleur;
            System.out.print("Fragile corolle taillée ");
    }

    public void eclore() { System.out.println("veiné de " + couleur); }

    public String toString() {
            return "qu'un simple souffle" ;
    }

}

class Poeme
{
    public static void main(String[] args) {
            Fleur f1 = new Fleur("Violette", "bleu");
            Fleur f2 = new Fleur(f1);
            System.out.print("dans un cristal ");
            f2.eclore();
            System.out.print("ne laissant plus ");
            System.out.println(f1);
            System.out.println(f2);
    }
}
```

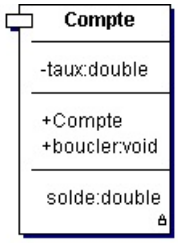
Exercice 5 : Banque (Transformation d'un programme non OO en un programme OO, POO de base)

Le code complet est donné ci-dessous:

```
/* Version orientée objets de Banque1. */

class Banque2 {

    public static void main(String[] args) {
        // Variables locales pour les taux d'intérets (afin d'éviter de
        // répéter les mêmes chiffres pour chaque client):
        double taux1 = 0.01;
        double taux2 = 0.02;

        // Construction des deux clients:
        Client c1 = new Client("Pedro", "Genève", taux1, 1000.0, taux2, 2000.0);
        Client c2 = new Client("Alexandra", "Lausanne", taux1, 3000.0, taux2, 4000.0);

        System.out.println("Donnees avant le bouclement des comptes:");
        c1.afficher();
        c2.afficher();

        // Bouclement des comptes des deux clients:
        c1.boucler();
        c2.boucler();

        System.out.println("Donnees apres le bouclement des comptes:");
        c1.afficher();
        c2.afficher();
    }
}

class Client {

    private String nom;
    private String ville;
    private Compte cpt1;
    private Compte cpt2;

    public Client(String nom, String ville, double taux1, double solde1,
                double taux2, double solde2) {
        this.nom = nom;
        this.ville = ville;
        // Construction d'un compte privé:
        cpt1 = new Compte(taux1, solde1);
        // Construction d'un compte d'épargne:
        cpt2 = new Compte(taux2, solde2);
    }

    public void afficher() {
        // Cette méthode affiche les données du client
        System.out.println("   Client " + nom + " de " + ville);
        System.out.println("      Compte prive:     "
                    + cpt1.getSolde() + " francs");
        System.out.println("      Compte d'epargne: "
                    + cpt2.getSolde() + " francs");
    }

    public void boucler() {
        // Cette méthode boucle les deux comptes du client
        cpt1.boucler();
        cpt2.boucler();
    }
}

class Compte {

    private double taux;
    private double solde;

    public Compte(double taux, double solde) {
        this.taux = taux;
        this.solde = solde;
    }
```

```
    public double getSolde() {
        return solde;
    }

    public void boucler() {
        // Cette méthode ajoute les intérets au solde
        double interets = taux * solde;
        solde = solde + interets;
    }
}
```





Banque avec des clientes

```
class Banque3 {

    public static void main(String[] args) {
        // ... comme avant

        // Construction des deux clients, notez l'argument booléen:
        // Nouveau (parmètre booléen)
        Client c1 = new Client("Pedro", "Geneve", taux1, 1000.0, taux2, 2000.0, true);
        Client c2 = new Client("Alexandra", "Lausanne", taux1, 3000.0, taux2, 4000.0, false);

        // ... comme avant
    }
}

class Client {

    private String nom;
    private String ville;
    private Compte cpt1, cpt2;
    // Nouvelle variable d'instance
    private boolean masculin;

    // Méthode constructeur, notez le paramètre booléen:
    public Client(String nom, String ville, double taux1, double solde1,
                  double taux2, double solde2, boolean masculin) {  // Nouveau
        this.nom = nom;
        this.ville = ville;
        cpt1 = new Compte(taux1, solde1);
        cpt2 = new Compte(taux2, solde2);
        // Nouveau
        this.masculin = masculin;
    }

    public void afficher() {
        // Nouvelle instruction if..else
        if (masculin) {
            System.out.print("   Client ");
        } else {
            System.out.print("   Cliente ");
        }
        System.out.println(nom + " de " + ville);
        System.out.println("      Compte prive:     " +
                cpt1.getSolde() + " francs");
        System.out.println("      Compte d'epargne: " +
                cpt2.getSolde() + " francs");
    }
    // ... comme avant
}

class Compte {
```

```
    // ... comme avant
}
```

Exercice 6 : Supermarché (poo de base)

```
import java.util.ArrayList;
import java.util.Date;
import java.text.SimpleDateFormat;

class Article {

    private String nom;
    private double prix;
    private boolean action;

    public Article(String nom, double prix, boolean action) {
        this.nom = nom;
        this.prix = prix;
        this.action = action;
    }

    public double getPrix() {
        return prix;
    }

    public String getNom() {
        return nom;
    }

    public boolean estEnAction() {
        return action;
    }

}

class Achat {

    private Article article;
    private int quantite;

    public Achat(Article article, int quantite) {
        this.article = article;
        this.quantite = quantite;
    }

    public double getPrix() {
        double rabais = 1.0;
        if (article.estEnAction()) {
            rabais = 0.5;
        }
        return quantite * article.getPrix() * rabais;
    }

    public void affiche() {
        String nom = article.getNom();
        double prixUnite = article.getPrix();
        double prixTotal = getPrix();
        String rabais = "";

        if (article.estEnAction()) {
            rabais = " (1/2 prix)";
        }
        System.out.println(nom + " : " + prixUnite + " x " + quantite + " = " + prixTotal + " Frs" + rabais);
    }

}

class Caisse {

    private int numero;
    private double total;

    public Caisse(int numero, double total) {
        this.numero = numero;
        this.total = total;
    }

    public void totalCaisse() {
```

```
        // printf permet de formatter l'affichage (par exemple pour avoir
        // 2 décimale seulement après la virgule)
        // %c signifie que l'argument 'numero' a pour format d'affichage
        // celui d'une chaine de caracteres
        // %.2f signifie que l'argument 'total' a pour format d'affichage
        // une valeur réelle avec 2 décimales après la virgule
        // %n signfie: saut de ligne
        System.out.printf("La caisse numero %c a encaisse %.2f Frs aujourd'hui%n" , numero, total);
    }

    public void scanner(Caddie caddie) {
        System.out.println("========================================");

        // Affichage de la date courante
        Date dateCourante = new Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("dd/MM/yy");
        System.out.println(formatDate.format(dateCourante));

        System.out.println("Caisse numéro " + numero);
        System.out.println();

        int nbAchat = caddie.getNbAchats();
        double montantTotal = 0;

        for (int i = 0; i < nbAchat; i++) {
            Achat achat = caddie.getAchat(i);
            double prix = achat.getPrix();
            achat.affiche();
            montantTotal += prix;
            total += prix;
        }

        System.out.println();
        System.out.println("Montant a payer : " + montantTotal + " Frs");
        System.out.println("========================================");
    }

}

class Caddie {
    // Ce corrigé vous permet de voir un exemple d'utilisation
    // des tableaux dynamiques (juste entrevus en cours)
    // Les tableaux de ce type peuvent avoir une taille qui
    // augmente ou diminue en cours d'exécution (il n'est pas nécessaire
    // de fixer une taille à la création du tableau)

    // listeAchats est un tableau dynamique d'Achats
    private ArrayList<Achat> listeAchats;

    public Caddie() {
        // le constructeur du tableau dynamique est invoqué
        // pour créer le tableau (qui est vide de contenu au départ)
        this.listeAchats = new ArrayList<Achat>();
    }

    public void remplir(Article article, int quantite) {
        Achat achat = new Achat(article, quantite);

        // la méthode add des ArrayList permet de
        // leur ajouter dynamiquement des données en cours
        // d'exécution du programme.
        // la contrepartie de suppression (remove) existe aussi.
        listeAchats.add(achat);
    }

    public Achat getAchat(int index) {
        // l'accès au ieme élément d'un tableau
        // dynamique se fait par la méthode get
        return listeAchats.get(index);
    }

    public int getNbAchats() {
        // les tableaux dynamiques disposent d'une méthode size
        // retournant la taille du tableau
        return listeAchats.size();
    }
```

```
}


public class Supermarche {

    public static void main(String[] args) {
        // Les articles vendus dans le supermarché
        Article choufleur = new Article("Chou-fleur extra", 3.50, false);
        Article roman = new Article("Les malheurs de Sophie", 16.50, true);
        Article camembert = new Article("Cremeux 100%MG", 5.80, false);
        Article cdrom = new Article("C++ en trois jours", 48.50, false);
        Article boisson = new Article("Petit-lait", 2.50, true);
        Article petitspois = new Article("Pois surgeles", 4.35, false);
        Article poisson = new Article("Sardines", 6.50, false);
        Article biscuits = new Article("Cookies de grand-mere", 3.20, false);
        Article poires = new Article("Poires Williams", 4.80, false);
        Article cafe = new Article("100% Arabica", 6.90, true);
        Article pain = new Article("Pain d'epautre", 6.90, false);

        // Les caddies du supermarché
        Caddie caddie1 = new Caddie();
        Caddie caddie2 = new Caddie();
        Caddie caddie3 = new Caddie();

        // Les caisses du supermarché
        // le premier argument est le numero de la caisse
        // le second argument est le montant initial de la caisse.
        Caisse caisse1 = new Caisse(1, 0.0);
        Caisse caisse2 = new Caisse(2, 0.0);

        // les clients font leurs achats
        // le second argument de la méthode remplir
        // correspond à une quantité

        // remplissage du 1er caddie
        caddie1.remplir(choufleur, 2);
        caddie1.remplir(cdrom, 1);
        caddie1.remplir(biscuits, 4);
        caddie1.remplir(boisson, 6);
        caddie1.remplir(poisson, 2);

        // remplissage du 2eme caddie
        caddie2.remplir(roman, 1);
        caddie2.remplir(camembert, 1);
        caddie2.remplir(petitspois, 2);
        caddie2.remplir(poires, 2);

        // remplissage du 3eme caddie
        caddie3.remplir(cafe, 2);
        caddie3.remplir(pain, 1);
        caddie3.remplir(camembert, 2);

        // Les clients passent à la caisse
        caisse1.scanner(caddie1);
        caisse1.scanner(caddie2);
        caisse2.scanner(caddie3);

        caisse1.totalCaisse();
        caisse2.totalCaisse();
    }
}
```

Exercice 7 : Segmentation de mots (poo de base)

```java
import java.util.Scanner;

public class TokenizableString {

    private static Scanner scanner = new Scanner(System.in);
    private String content;

    // Index de début et longueur d'une séquence
    private int len;
    private int from;


    public TokenizableString(String content) {
        this.content = content;
    }

    /* La fonction suivante teste si le caractère est un séparateur
     *
     * Ecrire une fonction présente l'avantage de pouvoir redéfinir facilement
     * la notion de séparateur (et éventuellement d'en définir plusieurs)
     */
    public boolean issep (char c) {
        return (c == ' ');
    }

    public void tokenize() {
        System.out.println("Les mots de \"" + content + "\" sont :");
        from = 0;
        len = 0;
        while (nextToken()) {
            System.out.println("'" + content.substring(from, from+len) + "'");
            from += len;
        }
    }

    /* Il y a de nombreuses autres façons d'écrire cette fonction.
     *
     */
    public boolean nextToken(){
        int taille = content.length();

        // saute tous les separateurs à partir de from
        while ((from < taille) && issep(content.charAt(from))) ++from;

        // avance jusqu'au prochain séparateur ou la fin de str
        len = 0;
        for (int i = from; ((i < taille) && !issep(content.charAt(i))); ++len, ++i);

        return (len != 0);
    }

    // ----------------------------------------------------------------------
    public static void main(String[] args){
        String phrase;

        System.out.println("Entrez une chaine : ");
        phrase = scanner.nextLine();

        TokenizableString toToken = new TokenizableString(phrase);
        toToken.tokenize();
    }

}
```
