TD2 : Héritage

Exercice 1. Exécution de code

Que fournit l'exécution du code suivant :

```
class A {
    public A (int n) {
        System.out.println ("Entree Constr A - n=" + this.n + " p=" + this.p);
        this.n = n;
        System.out.println ("Sortie Constr A - n=" + this.n + " p=" + this.p);
    }
    public int n;
    public int p=10;
}

class B extends A {
    public B (int n, int p) {
        super (n);
        System.out.println ("Entree Constr B - n=" + this.n + " p=" + this.p + " q=" + this.q);
        this.p = p;
        this.q = 2*n;
        System.out.println ("Sortie Constr B - n=" + this.n + " p=" + this.p + " q=" + this.q);
    }
    public int q=25;
}

public class Ex1 {
    public static void main (String args[]){
        A a = new A(5);
        B b = new B(5, 3);
    }
}
```

Exercice 2. Héritage et redéfinition

Que fournit l'exécution du code suivant :

```
class A {
    public void affiche() {
        System.out.println ("Je suis un A") ;
    }
}
class B extends A {}
class C extends A {
    public void affiche() {
        System.out.println ("Je suis un C");
    }
}
class D extends C {
    public void affiche() {
        System.out.println ("Je suis un D");
    }
}
class E extends B {}
class F extends C {}

public class Ex2 {
    public static void main (String arg[]) {
        A a = new A() ; a.affiche();
        B b = new B() ; b.affiche();
        C c = new C() ; c.affiche();
        D d = new D() ; d.affiche();
        E e = new E() ; e.affiche();
        F f = new F() ; f.affiche();
    }
}
```

Exercice 3. Polymorphisme

```
class A { public void affiche() {System.out.println ("Je suis un A");}}
class B extends A {}
class C extends A { public void affiche() { System.out.println ("Je suis un C");}}
class D extends C { public void affiche() { System.out.println ("Je suis un D");}}
class E extends B {}
class F extends C {}

public class Ex3 {
    public static void main (String arg[]) {
        A a = new A(); a.affiche();
        B b = new B(); b.affiche();
        a = b;
        a.affiche();
        C c = new C(); c.affiche();
        a = c; a.affiche();
        D d = new D(); d.affiche();
        a = d; a.affiche();
        c = d; c.affiche();
        E e = new E(); e.affiche();
        a = e; a.affiche();
        b = e; b.affiche();
        F f = new F(); f.affiche();
        a = f; a.affiche();
        c = f; c.affiche();
    }
}
```

a. Que fournit l'exécution du code ci-dessus ?

b. Certaines possibilités d'affectation entre objets des types classes A, B, C, D, E et F ne figurent pas dans le programme ci-dessus. Pourquoi ? Donnez un exemple d'affectation entraînant une erreur de compilation.

Exercice 4. Limites du polymorphisme

Soit les classes Point et PointNom ainsi définies :

```
class Point {
    public Point(int x, int y) {
        this.x = x; this.y = y;
    }
    public static boolean identiques(Point a, Point b) {
        return ( (a.x==b.x) && (a.y==b.y) );
    }
    public boolean identique(Point a) {
        return ( (a.x==x) && (a.y==y) );
    }
    private int x, y;
}

class PointNom extends Point {
    PointNom (int x, int y, char nom) {
        super (x, y);
        this.nom = nom;
    }

    private char nom;
}

public class Ex4 {
    public static void main (String args[]) {
        Point p = new Point (2, 4);
        PointNom pn1 = new PointNom (2, 4, 'A');
        PointNom pn2 = new PointNom (2, 4, 'B');
        System.out.println (pn1.identique(pn2));
        System.out.println (p.identique(pn1));
        System.out.println (pn1.identique(p));
        System.out.println (Point.identiques(pn1, pn2));
        System.out.println (pn1.identiques(pn1, pn2));

        Point pn1p = pn1; Point pn2p = pn2;
        System.out.println (pn1p.identiques(pn1p, pn2p)); }}
```

a. Quels résultats fournit ce programme ? Expliciter les conversions mises en jeu et les règles utilisées pour traiter les différents appels de méthodes.

b. Doter la classe PointNom d'une méthode statique identiques et d'une méthode identique fournissant toutes les deux la valeur true lorsque les deux points concernés ont à la fois mêmes coordonnées et même nom. Quels résultats fournira alors le programme précédent ? Quelles seront les conversions mises en jeu et les règles utilisées ?

Exercice 5. Classe Cercle

On suppose donnée la classe Point suivante :

```
class Point {
    public Point (double x, double y) {this.x = x; this.y = y;}
    public void deplace (double dx, double dy) {x += dx; y += dy;}
    public void affiche() {System.out.println("Point de coordonnees " + x + " " + y);}
    public double getX() {return x;}
    public double getY() {return y;}
    private double x,y;
}
```

On souhaite réaliser une classe Cercle disposant des méthodes suivantes :
- un constructeur avec les coordonnées du centre et un rayon
- deplace qui déplace le centre du cercle
- changeRayon
- getCentre
- affiche

a. Définir la classe Cercle comme ayant un objet membre Point

b. Définir la classe Cercle comme dérivée de Point

Exercice 6. Structures algébriques

Hiérarchie des structures algébriques de l'ensemble $\mathbb{Z}$

On souhaite représenter les différentes structures algébriques (ensemble, groupe, anneau) de l'ensemble $\mathbb{Z}$, grâce à la programmation par objets.

a. Proposer une hiérarchie de classes pour représenter les différentes structures de l'ensemble $\mathbb{Z}$, dont les éléments seront représentés par des `int` : ensemble, groupe, anneau.

b. Écrire en Java les implémentations de ces classes.

Généralisation

On souhaite maintenant aussi représenter les différentes structures algébriques de l'ensemble $\mathbb{Z}/n\mathbb{Z} = \{0, 1, \ldots, n-1\}$: ensemble, groupe additif, anneau, corps.

c. Proposer une hiérarchie de classes basée sur la hiérarchie précédente, en détaillant les nouvelles classes et leurs nouveaux membres.

d. Y-a-t-il plusieurs possibilités ? Si oui, laquelle préférez-vous ?

e. Comment isoler les structures communes de groupe, d'anneau, de corps, propres à ces ensembles ?

f. Proposer une nouvelle implémentation basée sur les classes abstraites ou sur les interfaces.

Espace vectoriel et anneau des polynômes $\mathbb{Z}/p\mathbb{Z}[X]$

On souhaite maintenant représenter les polynômes de degré inférieur à un entier d, à coefficients dans un corps $\mathbb{Z}/p\mathbb{Z}$, avec p premier : $(\mathbb{Z}/p\mathbb{Z}[X])_d$. On rappelle que cet ensemble a une structure d'espace vectoriel sur $\mathbb{Z}/p\mathbb{Z}$.

g. Quelle structuration de classe allez-vous employer (interface, héritage, classe interne, objet membre...) pour cette nouvelle classe.

h. Écrire cette classe polynôme.

i. Peut-on utiliser votre classe pour représenter indifféremment les polynômes à coefficients dans $\mathbb{Z}$ et dans $\mathbb{Z}/p\mathbb{Z}$. Faire les modifications nécessaires le cas échéant.

Pour aller plus loin : structures génériques et polymorphisme

On rappelle que l'ensemble des polynômes de degré inférieur à d, à coefficients dans $\mathbb{Z}/p\mathbb{Z}$ possède également la structure d'anneau, de part sa loi de multiplication $\times$. On souhaiterait donc maintenant illustrer le fait que les deux ensembles $Z/p\mathbb{Z}$ et $(Z/p\mathbb{Z}[X])_d$ sont des anneaux.

j. Peut-on utiliser le même procédé que pour la question f ?

k. Proposer une solution. (Indication : on pourra essayer de créer une classe abstraite `Element` de laquelle dériveront des classes représentant les ensembles $\mathbb{Z}/n\mathbb{Z}$ et les polynômes $(\mathbb{Z}/p\mathbb{Z}[X])_d$ pour utiliser le polymorphisme.

Remarque : on verra bientôt en cours que la programmation générique permet de résoudre plus élégamment cette difficulté.