

## TD n°3 : Héritage et Polymorphisme

### Informatique

### M2TTT

#### Exercice 1

Quels résultats fournit le programme suivant :

```

class A
{
    public A(int nn)
    {
        System.out.println("In - Construteur A : n = " + n + " p = " + p);
        this.n = nn;
        System.out.println("Out - Construteur A : n = " + n + " p = " + p);
    }
    protected int n;
    protected int p = 10;
}

class B extends A
{
    public B (int n, int p)
    {
        super(n);
        System.out.println("In - Construteur B : n = " + n + " p = " + p + " q = " + q);
        this.p = p;
        q = 2*n;
        System.out.println("Out - Construteur A : n = " + n + " p = " + p + " q = " + q);
    }
    private int q = 25;
}

public class TestConst
{
    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
        A a = new A(5);
        B b = new B(5,3);
    }
}

```

#### Exercice 2

Quels résultats fournit le programme suivant

```

class A
{
    public void f(double x)
    {
        System.out.println("A.f(double) = " + x);
    }
}

class B extends A {}

class C extends A
{
    public void f(long q)
    {
        System.out.println("C.f(long) = " + q);
    }
}

class D extends C
{
    public void f(int n)
    {
        System.out.println("D.f(int) = " + n);
    }
}

class E extends B {}

class F extends C
{
    public void f(float x)
    {
        System.out.println("F.f(float) = " + x);
    }

    public void f(int n)
    {
        System.out.println("F.f(int) = " + n);
    }
}

public class TestSurdef
{
    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
        byte bb = 1; short p = 2; int n = 3; long q = 4;
        float x = 5.f; double y = 6.;

        A a = new A(); a.f(bb); a.f(x);
        B b = new B(); b.f(bb); b.f(x);
        C c = new C(); c.f(bb); c.f(q); c.f(x);
        D d = new D(); d.f(bb); c.f(q); c.f(y);
        E e = new E(); e.f(bb); e.f(q); e.f(y);
        F f = new F(); f.f(bb); f.f(n); f.f(x); f.f(y); f.f(p);
    }
}

```

}

### Exercice 3

1. L'objectif est de définir une classe triangle connaissant la classe Point telle que dans le cours
  - a. Ecrire la classe Point
    - i. en faisant attention à la nature de l'accès aux données
    - ii. en implémentant un constructeur par défaut et un constructeur permettant de d'initialiser les coordonnées d'un point
    - iii. en créant des accesseurs aux données
    - iv. en implémentant une méthode affiche()
    - v. en implémentant une méthode void deplace(int dx, int dy)
  - b. Sachant qu'un triangle est défini par 3 points, définir les données de la classe Triangle et l'écrire. Pour chaque objet, quel est le droit d'accès ?
  - c. Implémenter une méthode
    - i. void affiche() qui affiche les 3 points d'un triangle. On fera attention a ne pas répéter du code
    - ii. void deplacer(int dx, int dy) qui permettent de déplacer un triangle.
2. En vous inspirant de la classe Triangle, implémenter une classe Rectangle possédant une méthode
  - a. void affiche()
  - b. void deplacer(int dx, int dy)
  - c. Une méthode verifier(), qui vérifie la validité du rectangle
3. De même que pour les classes Triangle et Rectangle, implémenter la classe Cercle, un cercle étant défini par son rayon et son diamètre.
4. Créer une classe Forme. Les classes Triangle, Rectangle et Cercle héritent de la classe Forme.
5. Ajouter à la classe Forme et à ses classes dérivées
  - a. Une méthode périmètre qui calcule le périmètre des formes géométriques appartenant à la classe Forme
  - b. Même chose pour la surface
6. Dans une application DoForme, définir un tableau de dimension 3 contenant un triangle passant par les points A(1,0), B(3,5) et C(6,2), le cercle de centre C et de rayon 5 et le rectangle de sommet D(1,2), E(1, 5) et F(5,4).
7. A partir de ce tableau, calculer
  - a. le périmètre
  - b. l'airedes formes contenues dans ce tableau
8. Déplacer les figures de 1 en abscisse et de 2 en ordonnée