Exercice 1. Définition d'une classe dérivée

```
class Point2D{
   public int getX(){ return x ; }
   public int getY(){ return y ; }

   public void setPosition(int newX, int newY){ x = newX ; y = newY ; }

   private int x, y ;
}
```

- Définir une classe dérivée de `Point2D`, `Point2DAffichable`, qui dispose d'une méthode `affiche` afin d'afficher les coordonnées du point. Ecrire un programme de test utilisant les 2 classes avec tous leurs services (au sein d'une classe *`TestPoints2D`*).
- Définir une classe dérivée de `Point2D`, `Point3D`, qui dispose d'une coordonnée entière supplémentaire (z). Ecrire un programme de test pour tester les fonctionnalités de la classe `Point3D`.
- Si *T'* est une classe dérivée d'une classe *T*, *T'* vérifie la relation « est » un *T*, tandis que si une classe *A* utilise un objet de type *B* au sein de sa classe (composition ou agrégation), *A* vérifie la relation « a » un *B*.
    - Lorsqu'on doit écrire une nouvelle classe à partir d'une classe existante, comment choisir entre un héritage et une composition ?
    - Finalement pour la classe `Point3D`, valait-il mieux utiliser un héritage ou une composition ?
    - Pour définir une classe `Point3DAffichable`, faut-il mieux hériter de `Point2DAffichable` ou de `Point3D` ? Justifier votre réponse.

Exercice 2. Héritage et constructeurs

- Le constructeur d'une classe dérivée doit « fournir » les arguments effectifs au constructeur de la classe de base si ce dernier n'est pas un constructeur sans argument.
    - Proposer 2 classes, `Personne` et `Etudiant`, une personne ayant un nom et un prénom, un étudiant ayant en plus un numéro d'étudiant. Munir ces 2 classes des constructeurs et méthodes que vous jugerez nécessaires. Tester ces 2 classes.

Exercice 3. Héritage et redéfinition

- Reprendre les classes `Personne` et `Etudiant` de l'exercice précédent et munir ces 2 classes d'une méthode publique `void affiche()` affichant le nom, le prénom (ainsi que le numéro d'étudiant pour l'étudiant) **en réutilisant le code au maximum**. Tester ces 2 méthodes.
- Si une classe `EtudiantBoursier` est construite par héritage, à partir de la classe `Etudiant` précédente, sans redéfinir la méthode `affiche`, alors quelle méthode `affiche` sera appelée sur un objet de type `EtudiantBoursier`, celle de `Personne` ou celle de `Etudiant`?

Exercice 4. Dérivations successives et surdéfinitions

```
class A
{ public void f(double x) { System.out.print( "A.f(double=" + x + ") " ) ; }
}
class B extends A {}
class C extends A
{ public void f(long q) { System.out.print( "C.f(long=" + q + ") " ) ; }
}
class D extends C
{ public void f(int n) { System.out.print( "D.f(int=" + n + ") " ) ; }
}
class E extends B {}
class F extends C
{ public void f(float x) { System.out.print( "F.f(float=" + x + ") " ) ; }
  public void f(int n) { System.out.print( "F.f(int=" + n + ") " ) ; }
}

public class Test
{ public static void main(String arg[])
  {
     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) ; System.out.println() ;
     B b = new B() ; b.f(bb) ; b.f(x) ; System.out.println() ;
     C c = new C() ; c.f(bb) ; c.f(q) ; c.f(x) ; System.out.println() ;
     D d = new D() ; d.f(bb) ; d.f(q) ; d.f(y) ; System.out.println() ;
     E e = new E() ; e.f(bb) ; e.f(q) ; e.f(y) ; System.out.println() ;
     F f = new F() ; f.f(bb) ; f.f(n) ; f.f(x) ; f.f(y) ; f.f(p) ; f.f(q) ;
  }
}
```

Quels résultats fournit le programme précédent ?