TD et TP n° 4 : Mécanismes et stratégies d'héritage (révision), Patron décorateur, Transtypage

Si vous ne les avez pas faits faites ou finissez les exercices 1, 3 et 4 du TP3.

I) Révision : Mécanismes et stratégies d'héritage

Exercice 1 : Personnes

```
1  class Personne {
2      private String nom;
3      Personne(String nom) { this.nom=nom; }
4      void presenteToi() { System.out.println("Je suis " + nom ); }
5      void chante() { System.out.println("la-la-la"); }
6  }
7
8  class Enseignant extends Personne {
9      private String matiere;
10     Enseignant(String nom, String matiere) { super(nom); this.matiere=matiere; }
11     void presenteToi() { System.out.println("Je suis " + nom + ",enseignant de "+ matiere); }
12     void enseigne() { System.out.println(matiere + "is beautiful"); }
13 }
14
15 public class Test {
16     public static void main (String[] args){
17         Personne yan = new Personne ("Jurski");
18         Enseignant isabelle = new Enseignant("Fagnot","CPOO5");
19         Personne aldric = new Enseignant("Degorre","CPOO5");
20         isabelle.chante(); yan.enseigne(); aldric.chante(); aldric.enseigne();
21         Personne[] jury = { yan, isabelle, aldric, new Enseignant("Shen", "CPOO5"), (Personne) new
               Enseignant("Fantome", "CPOO5") };
22         for(Personne p:jury) p.presenteToi();
23     }
24 }
```

1. Lisez, comprenez et corrigez ce code, le cas échéant.
2. Dans la classe `Enseignant`, quels sont les attributs hérités, ajoutés ? Quelles sont les méthodes héritées, ajoutées, redéfinies ?
3. Comment fonctionne le constructeur de `Enseignant` ?
4. Qu'affiche le code ?

Exercice 2 : Liaison dynamique

Qu'affiche le programme suivant :

```
1  class X {} class Y extends X {}
2
3  class A {
4      void f(X y) { System.out.println("A et X"); }
5      void f(Y y) { System.out.println("A et Y"); }
6  }
7
8  class B extends A {
9      void f(X y) { System.out.println("B et X"); }
10     void f(Y y) { System.out.println("B et Y"); }
11 }
12
13 class C extends B {
14     void f(Y y) { System.out.println("C et Y"); }
15 }
16
17 public class Test {
18     public static void main(String args[]) {
19         A a = new C();
20         Y x = new Y();
21         a.f((X) x); // affichage n°1
```

```
        a.f(x); // affichage n°2
    }
}
```

II) Patron décorateur

On reprend la hiérarchie de classes de l'exercice 4 du TP3.
Dans le but de pouvoir dessiner ces formes, on veut pouvoir donner à l'utilisateur la possibilité de préciser certains paramètres du graphismes : La couleur du fond, la largeur du bord, sa couleur, ...

Exercice 3 : Solution simple

Ne pas la programmer
Une solution consisterait à faire une classe `FigureDecoree` contenant une référence vers une instance de `Shape2D` et contenant divers attributs représentant, la couleur du fond, la couleur du bord, la largeur du bord, ...
Et on fait une méthode `setXXX(...)` pour chaque option. Quels inconvénients à cette solution ?

Exercice 4 : Patron Décorateur (decorator pattern)

À programmer !
L'idée est de faire une classe abstraite de `Shape2DDécorée` implémentant l'interface `Shape2D` et contenant un attribut de type `Shape2D`. Chaque classe Concrète héritant de cette classe représentera une option et aura les attributs permettant de connaître les paramètres correspondant : Par exemple, on aura les classes `Shape2DFondColoré`, `Shape2DLargeurBord` ....

- Avant de programmer, dessiner les cases mémoires représentant l'objet `figure` obtenu avec le code

```
Parallelogramme par = new Parallelogramme(new Point2D.Double(10,15),
            new Point2D.Double(5,6), new Point2D.Double(8,20));
Shape2D figure = new Shape2DFondColoré(new Shape2DLargeurBord(par);
```

- Programmez-le pour quelques options. Comment faites-vous pour coder `surface` et `perimetre` ?
- Quels sont les inconvénients de la méthode précédente que l'on peut régler facilement ?
- Y a-t'il des inconvénients qui n'existaient pas dans la précédente solution ?

III) Partie facultative

Exercice 5 : Transtypage

```

class A { }

class B extends A { }

class C extends A { }

public class Tests {
   public static void main(String[] args) {
      System.out.println((int)true);
      System.out.println((int) 'a');
      System.out.println((byte) 'a');
      System.out.println((byte) 257);
      System.out.println((char) 98);
      System.out.println((double) 98);
      System.out.println((char) 98.12);
      System.out.println((long) 98.12);
      System.out.println((boolean) 98.);
      System.out.println((B) new A());
      System.out.println((C) new B());
      System.out.println((A) new C());
   }
}
```

Dans la méthode `main()` ci-dessus,

1. Quelles lignes provoquent une erreur de compilation ?
2. Après avoir supprimé ces-dernières, quelles lignes provoquent une exception à l'exécution ?
3. Après les avoir enlevées, elles aussi, quels affichages provoquent les lignes restantes ?

Exercice 6 : Transtypages d'objets (sur machine)

Même exercice que l'exercice 4 et 5 du TP3 mais sur le programme suivant :

```
1 public class TranstypagesObjets {
2    public static void main(String[] args) {
3       Object vObject = new Object();
4       Integer vInteger = 42;
5       String vString = "coucou";
6       System.out.println("vObject = " + vObject + ", vInteger = "
7          + vInteger + ", vString = " + vString);
8    }
9 }
```

Différence, vous ne verrez plus l'ajout de l'instruction **u2t** mais parfois celle de **checkcast**. Dans quels cas ?
Dans certains cas vous aurez eu besoin, pour compiler, d'un *cast* explicite. Lesquels ? Est-ce les-mêmes que dans la question précédente ?
Dans certains cas, le programme quittera sur `ClassCastException`, lesquels ?