TP 3 : programme de dessin, classe `Forme`

Dans ce TP nous allons continuer à travailler sur la petite interface graphique écrite au TP précèdent. Il est donc vivement conseillé d'avoir fini le TP précédent. Nnous allons ajouter utiliser les notions d'héritage et de polymorphisme pour manipuler un tableau de formes.

1 Héritage : introduction

Au TP précédent nous avons crée trois classes : `Cercle`, `Carre` et `Triangle`. Ces trois classes décrivent des formes géométriques qui peuvent être dessinées à l'écran. Au fur et à mesure que nous développons ces classes, nous nous rendons compte qu'elles ont des fonctionnalités en commun : elles pourront avoir des attributs en commun (comme le fait d'avoir une couleur) et elles pourront avoir des comportements en commun (comme le fait de pouvoir être dessinées à l'écran).

La programmation orientée objet propose la notion d'héritage (introduite au TD 2, vous pouvez le consulter si vous avez des difficultés) pour mettre en commun ces fonctionnalités. Nous allons créer une classe `Forme` qui regroupera ces fonctionnalités communes. Les classes `Cercle`, `Carre` et `Triangle` hériteront alors de la classe `Forme`. En d'autres termes, un `Cercle` *est une* `Forme`, de même qu'un `Carre` *est une* `Forme`. La notion d'héritage s'exprime en Java par le mot clé `extends` dans la déclaration de la classe :

```
class Cercle extends Forme
{
    // ...
}
```

Tous les attributs non privés et les méthodes non privées de la classe `Forme` seront accessibles à partir des classes dérivées. C'est un peu comme si les champs de la classe mère (ici `Forme`) étaient recopiés dans les classes filles.

2 Classe `Forme`

Écrivez une classe `Forme` et faites-en hériter les classes `Carre`, `Cercle` et `Triangle`. La classe `Forme` contiendra un attribut nommé `couleur` de type `Color`. Vérifiez que votre programme fonctionne comme avant après cette modification.

3 Polymorphisme

Au lieu d'utiliser trois tableaux différents pour stoker les cercles, les carrés et les triangles, nous allons utiliser un seul tableau de formes.

- Créez un tableau de formes dans votre classe principale (celle qui dérive de `JApplet`).
- Faites en sorte que chaque fois qu'on clique dans la fenêtre, un cercle, un carré ou un triangle (choisi aléatoirement) soit ajouté au tableau de formes.
- À la classe `Forme`, ajoutez une méthode `dessiner` dont le prototype sera identique à celui des classes `Carre`, `Cercle` et `Triangle`. Pour l'instant, cette méthode ne fera rien.
- Dans la méthode `paint` de votre classe principale, appelez la méthode `dessiner` de chaque élément du tableau de formes.

Comme vous pouvez le constater, même s'il s'agit d'un tableau de références vers des objets de type `Forme`, Java appelle automatiquement la méthode `dessiner` de la bonne classe dérivée. Cela s'appelle le polymorphisme.

4 Classe `Vector`

Nous avons jusque là utilisé un tableau classique pour stoker les formes. Les tableaux classiques ne sont pas très pratiques car ils ne peuvent pas se redimensionner. Nous allons utiliser à sa place la classe `Vector` (du package `java.util`) fourni par l'API Java.

- Regardez dans la documentation de la classe `Vector`. En particulier, vous allez avoir besoin des méthodes `add`, `size` et `get`.
- Remplacez le tableau de votre programme par un `Vector`. Attention, la méthode `get` retournant une référence vers un `Object`, il faudra faire un cast :

```
((Forme) mesFormes.get(i))
```

5 Saisie de touches

En vous inspirant de l'exemple suivant, faites en sorte que l'on puisse changer la taille ou la couleur des prochains objets crées en tapant sur une touche :

```
import javax.swing.*;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class KeyTest extends JApplet  implements KeyListener
{
    public void init()
    {
        System.out.println("adding KeyListener");
        addKeyListener(this);
    }

    public void paint(Graphics g)
    {
        System.out.println("paint");
        requestFocus();
    }

    // KeyListener methods
    public void keyPressed(KeyEvent e)  { System.out.println("event : " + e); }
    public void keyReleased(KeyEvent e) { System.out.println("event : " + e); }
    public void keyTyped(KeyEvent e)    { System.out.println("event : " + e); }
}
```