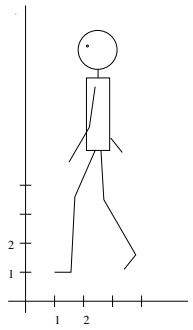
M1 Informatique (Master IFPRU) - FMIN107 - Ingénierie Logicielle
Partie A - Schémas de Réutilisation et de Conception

Christophe Dony - Clémentine Nebut

Année 2010-2011, session 1 de janvier 2011.

Durée : 1h00. Tous documents autorisés. La précision et la concision des réponses sont notées, ainsi que la lisibilité des codes.

Contexte. Plaçons nous dans le contexte de la réalisation de *examdraw*, un éditeur de dessins minimal. Chaque dessin (instance de la classe `Dessin`) y est représenté par un titre et par une collection d'objets instances de sous-classes de la classe `Figure`. Les types de figures, matérialisés par des sous-classes de `Figure` sont Ligne, Cercle, Carré, Triangle, etc. (voir un exemple de dessin *examdraw* en figure ci-contre).



Un dessin avec *examdraw*.

La représentation des *figures* utilise une ou des instances de la classe prédéfinie `java.awt.Point` (Un *point* est un objet doté d'une abscisse et d'une ordonnée). Une ligne peut ainsi être définie par deux points, un cercle par un point (son centre) et un entier (son rayon), un carré par exemple par un point (un de ses coins) et une longueur de côté, etc.

Dans la version "examen" de ce programme, pour ne pas compliquer l'exercice avec du code AWT ou SWING, il n'y a pas de méthodes d'affichage des dessins mais simplement une méthode `(drawDescription())` qui affiche sur un terminal une description textuelle du dessin.

Voici un exemple de création et d'affichage de description d'un dessin : [1] [2] [3].

```
Dessin d1 = new Dessin("mon premier dessin");
d1.add(new Cercle(5,5,1)); //les coordonnées de l'origine et le rayon
d1.add(new Ligne(5,4,5,2)); //les coordonées des deux points extrémité
d1.drawDescription(); // --> affiche le texte ci-dessous
Pour dessiner mon premier dessin, tracez :
- un cercle  de centre: 5.0@5.0 et rayon 1.0
- une ligne de 5.0@4.0 à 5.0@2.0
```

Voici un extrait du code des classes `Dessin` et `Figure` :

```
class Dessin {
    Vector<Figure> contenu;
    String title;

    Dessin(String t){
     contenu = new Vector<Figure>();
     title = t;}

    public void add(Figure s){contenu.add(s);}

    public void drawDescription(){
     System.out.println("Pour dessiner " + title + ", tracez : ");
     for (Figure sh : contenu) sh.drawDescription(); } }

public abstract class Figure{
    public void drawDescription(){System.out.println(this.toString());} }
```

[1] Les coordonnées ne sont pas signifiantes et ne correspondent pas au dessin en figure 1).
[2] la méthode `toString()` possède une définition sur la classe `Object` et est redefinie sur de nombreuses sous-classes, dont `Point`
[3] La méthode `toString()` de la classe `Point` produit la chaine : "abscisse@ordonnée".

1 Bases réutilisation - Environ 4 points

Questions.

1. La méthode `drawDescription` des figures est factorisée sur la classe `Figure`. Selon quel principe est-elle paramétrée (selon la terminologie définie en cours) ? Par quelle méthode est-elle paramétrée ?
2. L'instruction : "`Figure f1 = new Cercle(3,3,1) ;`" est-elle valide ? Qu'en diriez vous si vous deviez l'expliquer à quelqu'un.
3. Donnez en Java la définition de la classe `Cercle` : constructeurs et méthodes utiles à l'énoncé uniquement.
4. Donnez la liste complète des méthodes exécutées par l'envoi du message précédent : `f1.drawDescrition()`. Précisez pour chaque méthode exécutée quel est son receveur courant (objet référencé par `this`).

2 Schéma de conception Décorateur : Environ 4 points

On souhaite que chaque figure d'un dessin puisse être décorée, par exemple, par une couleur ou par une annotation textuelle (qui pourra être affichée dans la description textuelle ou graphique). On souhaite pouvoir ajouter dynamiquement des décorations à une figure.

Voici un exemple de code écrit avec cette nouvelle fonctionnalité, une décoration de type "annotation textuelle" est ajoutée aux deux figures d'une dessin.

```
public class Bonhomme extends Dessin {

    public Bonhomme(){this("un bonhomme");}

    public Bonhomme(String t){
        super(t);
        this.add(
            new FigureAnnotée(new Cercle(new Point(5,6), 1), "La tête"));
        this.add(
            new FigureAnnotée(new Ligne(new Point(5,5), new Point(5,3)), "Le cou"));
        // etc
      }}
```

L'instruction "`new Bonhomme().drawDescription()`" produit alors le texte suivant :

```
Pour dessiner un bonhomme, tracez :
- un cercle  de centre: 5.0@6.0 et rayon 1.0 (La tête)
- une ligne de 5.0@5.0 à 5.0@3.0 (Le cou)
```

Questions :

1. Donnez le schéma UML de la nouvelle hiérarchie des "Figures" correspondant au code précédent.
2. Donnez, de la façon la plus simple possible, le code des nouvelles classes que vous avez dû ajouter.

3 Schéma de conception Adapteur : Environ 2 points

On souhaiterait qu'un point puisse faire partie des figures constituant un dessin (par exemple pour faire l'oeil du bonhomme de la figure 1).

Questions :

1. Pourquoi n'est-ce pas possible d'écrire : `unDessin.add(newPoint(x,y))` ?
2. Définissez en Java une classe d'adaptation `PointAdapter`, la plus simple possible, rendant la chose possible et opérationnelle. Expliquez.
3. Montrer qu'il est possible d'ajouter une annotation à une figure de type `PointAdapter` avec votre solution ?