iUT ORSAY
Département Informatique

DUT Informatique
Module JAVA Apprentis
2008 / 2009

Travaux Pratiques n° 3 : Interfaces

Nom(s) :
Groupe :
Date :

Objectifs : Maîtriser le mécanisme d'héritage. Savoir manipuler les classes abstraites et les interfaces.

1 Introduction

On désire réaliser une application graphique dans laquelle on manipule différents types de figures. Toutes les figures se caractérisent par une couleur de fond, une couleur de bordure et des coordonnées dans un espace à deux dimensions. Une figure doit pouvoir afficher toutes les informations la concernant et se cloner pour créer une nouvelle figure identique. Il doit être possible de consulter et modifier tous les paramètres d'une figure.
Une figure est en réalité un type abstrait (on ne peut en créer des instances). Les figures réelles sont par exemple des cercles, des carrées, des rectangles qui vont implémenter de façon un peu différente certaines méthodes, ou avoir des structures de données plus adaptées.
Le but de ce TP est de réaliser une application manipulant ces figures réelles.

2 Classe principale

Créez une classe `Principale` dans un paquetage `TP3` qui ne contiendra qu'une méthode `main()` (vide pour l'instant) et sera destinée à tester les autres classes.

3 Héritage et classes abstraites

Créez la classe abstraite `Figure`, implantez dans `Figure` les méthodes communes à toutes les figures, par exemple les accesseurs en consultation et modification pour les couleurs.

Certaines méthodes devront être implantées différement en fonction de la figure réelle finale, on souhaite par exemple disposer des méthodes suivantes :
– La méthode de signature `Figure clone()` qui retourne un nouvel objet, copie de l'objet appelant (mêmes valeurs des attributs).
– La méthode de signature `String toString()` qui retourne une chaîne de caractères et servira à afficher textuellement l'objet, par exemple lors d'un appel à `System.out.println()` avec en paramètre une référence à un objet de type `Figure`.

Déclarez dans `Figure` les méthodes abstraites `clone()` et `toString()`.

Créez les classes concrètes `Cercle` et `Rectangle`, qui héritent de `Figure`. Pensez à créer des constructeurs adaptés et à implanter les méthodes abstraites héritées.

Créez dans la méthode `main()` de la classe `Principale` un jeu d'instructions permettant de tester convenablement votre travail dans cette partie (on ne demande cependant pas un test JUnit !).

4 Interfaces

Les figures peuvent être dotées de propriétés spéciales comme être dessinables, déplaçables ou transformables. Ces propriétés sont des « contrats » que l'on passe. On souhaite donc les formaliser au sein d'interfaces qui indiquent les signatures des méthodes à utiliser et implanter.

Définissez une interface `Deplacable` ainsi que la signature d'une ou plusieurs méthodes destinées à modifier les coordonnées.

Définissez une interface `Transformable` ainsi que les signatures des méthodes pour effectuer une symétrie verticale et une symétrie horizontale d'une figure.

Modifiez la classe `Figure` afin qu'elle implante les nouvelles interfaces.

Modifiez les classes `Cercle` et `Rectangle` en implantant les différentes méthodes.

Créez dans la méthode `main()` de la classe `Principale` un jeu d'instructions permettant de tester convenablement votre travail dans cette partie non plus, sauf si vous le souhaitez).

5 Collections et variables de classe

On veut créer une classe `Chevalet` contenant une structure de données de type `Vector` (ou `ArrayList`, au choix) pour stocker toutes les figures que l'on souhaite dessiner (voir la Javadoc pour des informations concernant ces classes).

Écrivez une classe `Chevalet`.

Créez une variable de classe permettant de conserver le nombre de figures crées au sein de tous les chevalets.

Créez une méthode pour déplacer toutes les figures d'un chevalet.

Créez dans la méthode `main()` de la classe `Principale` un jeu d'instructions permettant de tester convenablement votre travail dans cette partie (pas de test JUnit).

Vous rendrez avec votre compte rendu tous les codes correctement commentés, on devra en particulier pouvoir générer une javadoc appropriée à partir de vos codes sources.