Licence Informatique **UE INF203**
Université Bordeaux 1 **Programmation par objets**
2006-2007 **Travaux Dirigés**
feuille 7

Partie 1 : Héritage, classe abstraite, méthode abstraite

Exercice 1. On considère les fichiers fournis en annexe. Se placer dans le répertoire partie1. Compiler puis exécuter le fichier TestFigures1. Étudier les codes sources fournis et répondre plus particulièrement aux questions suivantes :

- Quelle est la signification du mot clé `abstract` dans la déclaration de la classe `FigureCentree` ?
- Déterminer pour chacune des classes `FigureCentree`, `Rectangle`, `Cercle`, `CercleTroue` les méthodes héritées, les méthodes surchargées, les méthodes redéfinies, les nouvelles méthodes ?
- Justifier les lignes suivantes :

```
public FigureCentree(Point u){
  centre = new Point(u);
}
```

- Quel est le rôle du mot clé `super` dans chacune des lignes suivantes

```
CercleTroue(Cercle c, double ray) {
  super(c.monCentre(), c.monRayon());
  trou = new Cercle(super.monCentre(), ray);
}

public double aire() {
  return super.aire() - trou.aire();
}
```

Exercice 2. On veut placer les figures crées dans un tableau de `FigureCentree`. Pour ce faire on utilise TestFigures2.java. Compiler et exécuter cette classe.
On veut maintenant afficher les aires. Décommenter les dernières lignes du programme et apporter les corrections nécessaires aux classes précédemment définies.

Exercice 3. Ajouter à l'interface `FigureAvecCentre` les méthodes suivantes :

```
/** Applique à la figure une rotation  de centre donné */
public void rotationPiSur2(Point centreRot);

/** Applique à la figure une rotation autour de son centre */
public void rotationPiSur2();
```

Modifier les diverses classes en conséquence.

Partie 2 : Notion de paquetage

On désire permettre aux sous-classes (classes dérivées) l'accès aux champs de la super classe.
Si on utilise le modificateur de portée `private`, les champs ne sont connus que pour les instances de la classe. Il ne sont donc pas accessibles pour les classes dérivées.
Si on utilise le modificateur de portée `public`, les champs sont toujours accessibles et on rompt l'encapsulation.
Il existe en Java la notion de paquetage (`package`). Un paquetage est un regroupement de classes défini par le programmeur. Par exemple le paquetage `java.io` regroupe les classes concernant les entrées-sorties.
Le modificateur de portée `protected` donne l'accès au champ pour toutes les classes du paquetage et pour les classes dérivées de la classe considérée. Dans tous les autres cas, le champ n'est pas accessible.

Pour définir un paquetage, on doit appliquer les règles suivantes :
- toutes les classes d'un paquetage sont situées dans un même répertoire `<nompaquetage>` où `<nompaquetage>` est le nom du paquetage.
- chaque fichier du paquetage a pour première ligne l'instruction `package <nompaquetage>;`
- Pour pouvoir utiliser un paquetage lors de la compilation ou de l'exécution d'une classe, on doit préciser où se trouve le paquetage grace à la variable d'environnement `CLASSPATH`
- le fichier de définition de la classe utilisant le paquetage `<nompaquetage>` doit comporter l'instruction `import <nompaquetage>;`

Exercice 4. Création et Utilisation d'un paquetage
Se placer dans le répertoire partie2 et compiler les sources fournis. Analyser les sources et exécuter TestFigures3.
On remarque que le centre du Cercle c2 est accessible alors qu'il est déclaré `protected`. En effet Java crée un paquetage par défaut dans lequel il place toutes les classes dont le paquetage n'est pas précisé. L'encapsulation est donc violée. Nous allons créer un paquetage `figures`.

1. Dans votre répertoire `programmation-par-objets` créer un sous-répertoire `mespaquetages`.
2. Dans un shell créer une variable d'environnement CLASSPATH :
   `export CLASSPATH=.:~/programmation-par-objets/mespaquetages/`
   (On pourra aussi créer cette variable d'environnement dans le fichier `.bashrc` en vue d'utilisations ultérieures)
3. Créer dans le répertoire `mespaquetages` un sous-répertoire `figures` et y déplacer tous les fichiers sources de `partie2` sauf le ficher TestFigures3.java. Détruire tous les fichiers d'extension `class`.
4. Insérer dans chaque fichier du répertoire `figures` la ligne :
   `package figures;`
   puis compiler ces fichiers.
5. Insérer au debut du fichier TestFigures3.java `import figures.*;` puis compiler celui-ci. Corriger les erreurs, recompilez et exécuter le fichier.

Partie 3 : Utilisation de conteneur

Exercice 5. Etudier la documentation de la classe Vector. Cette classe gère un vecteur extensible d'objets. L'étude de la documentation nous montre qu'il faut définir une méthode `equals` pour chacune des classes du paquetage figures. Modifier ces classes en conséquence.

Exercice 6. Utiliser la classe `Vector` pour créer un vecteur extensible de `FigureCentrees` et manipuler ces figures.
Aide : Etudier la signification de l'opérateur `instanceof` (Specification du langage java 15.20.2 `http://java.sun.com/docs/books/jls/third_edition/html/expressions.html#15.20.2`)