TP 4

Interface

Buts poursuivis par ce TP

- Assimiler la notion de classe abstraite et d'interface.
- Mettre en œuvre les interfaces.

Exercice 1

Question 1 / classe FormeGeometrique

Vous avez à votre disposition la classe FormeGeometrique (ci-dessous et sur moodle) :

```
public abstract class FormeGeometrique {
	protected double posX, posY;

	public FormeGeometrique(double x, double y){
		posX=x;
		posY=y;
	}

	public void deplacer(double x,double y) {
		posX=x;
		posY=y;
	}

	public void afficherPosition() {
		System.out.println("position : ("+posX+","+posY+")");
	}

	public String toString(){
	return posX+" "+posY ;
	}

	public void afficher(){
		System.out.println("("+posX+", "+posY+")");
	}

	public abstract double surface() ;
	public abstract double perimetre() ;
}
```

Une forme géométrique est caractérisée par sa position (x, y), son périmètre et sa surface. Prenez le temps de vous familiariser avec son code, et plus particulièrement ses méthodes, puis répondez aux questions suivantes :

1. Que peut-on dire de la classe FormeGeometrique ? Est-elle instanciable ?
2. Que peut-on dire des méthodes *surface()* et *perimetre()* ? Quelle en est la conséquence pour les classes filles de FormeGeometrique ?

Question 2 / classe Rectangle et classe Cercle

Ecrivez deux classes Rectangle et Cercle qui héritent de la classe FormeGeometrique.

1. Elles définissent ses méthodes abstraites héritées.
2. Ecrivez pour chaque classe un constructeur qui reçoit en paramètres toutes les données nécessaires :
   Un rectangle est une *FormeGeometrique* caractérisé par une largeur et une hauteur.
   Un cercle est *FormeGeometrique* caractérisé par un rayon.

Question 3 / ***interface*** Dessinable

Supposons maintenant que nous voulions développer une classe *Dessinable* affichant le type de forme géométrique en couleur. Nous pourrions donc créer une classe Dessinable et en dériver les classes Cercle et Rectangle. Mais ces deux classes héritent déjà de la classe FormeGeometrique et l'héritage multiple est interdit en Java.
C'est pour débloquer ce genre de situation qu'ont été créées les interfaces de Java.

1. Ecrivez une interface *Dessinable* avec la méthode *dessiner (String couleur)* qui affiche le type (Cercle/Rectangle) et la couleur d'une instance de *FormeGeometrique*.
2. Modifiez les classes *Cercle* et *Rectangle* qui implémentent l'interface *Dessinable*.

Question 4 / interface `Comparable`

Supposons que nous voulions utiliser la méthode *sort* de la classe `Arrays` pour trier un tableau d'objets *Rectangle*.
La classe Rectangle doit donc implémenter l'interface `Comparable` ; la seule méthode à définir est *compareTo(Object autre)* qui renvoie un entier positif si la surface du Rectangle this est supérieure au rectangle autre, 0 s'ils sont égaux, et négatif sinon.

Iut Lannion **R2.01 Dev. Orienté Objets**
Département informatique

TP 4

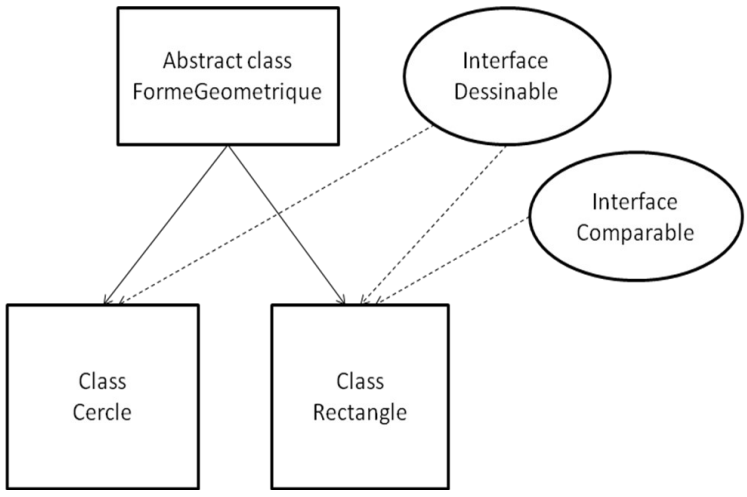
Interface

Question 5 /

Ecrivez un programme qui crée 4 rectangles dans un tableau et « dessine » (méthode *dessiner()*) en jaune le plus grand Rectangle.

Pour cela, définissez une classe ListeRectangles ayant en variable d'instance une instance *listeR* d'`ArrayList` de Rectangle et les méthodes :

1. void ajouter(Rectangle r) qui insère *r* dans *listeR*,
2. void afficher() qui affiche les *nbR* Rectangle de *listeR*,
3. void trier() qui trie les *nbR* Rectangle présents dans liste*R*,
4. Rectangle obtenirIemeRectangle(int i), qui délivre le *i* ème Rectangle de *listeR* (le 1er rectangle est le rectangle d'indice 0 dans *listeR*),
5. int obtenirNombreRectangles(), qui délivre le nombre de Rectangle présents dans *tR*,
6. le constructeur ListeRectangles(int taille) qui réserve la place pour *taille* Rectangle.
7. La méthode d'instance *trier()* copie les rectangles de la liste *listeR* dans un tableau de rectangles, trie le tableau par appel à la méthode de classe *sort* () de la classe `Arrays`, puis recopie le tableau trié dans la liste *listeR*.



Conclusion

Une interface est telle que :

- *Toutes ses méthodes sont abstraites (c-à-d non définies)*
- *Elle ne peut avoir que des variables static final (des constantes !)*
- *Une telle interface n'est pas instanciable, et n'est pas une classe.*

Exercice complémentaire

Question 1 /

Une entreprise peut être une entreprise unipersonnelle ou bien une société comportant plusieurs employés.
Une entreprise unipersonnelle est caractérisée par un nom, une adresse, un nombre de prestations et un bénéfice effectués dans l'année.
Une société a un nom, une adresse, un nombre d'employés, et une masse salariale.
Définissez en Java les classes correspondantes.

Question 2 /

Un territoire peut être une ville ou un pays. Une ville possède un nom et un nombre d'habitants, un pays possède un nom, un nombre d'habitants et une superficie.
Définissez en Java les classes correspondantes.

Question 3 /

Proposez une interface *Statistique* qui propose une méthode calculant et délivrant une moyenne, et une méthode affichant une moyenne.
La définition de la moyenne varie suivant la classe qui l'implémente : nombre moyen d'habitants au km^2 pour un pays, bénéfice moyen par prestation pour un entreprise unipersonnelle, salaire moyen par salarié pour une société.

Question 4 /

Définissez une classe *LesStats* comportant en variable d'instance une instance d'*ArrayList* de *Statistique* .
Prévoir les méthodes permettant d'insérer une *Statistique*, d'afficher toutes les statistiques, d'obtenir le nombre de statistiques présentes, sans oublier le constructeur.

Question 5 /

Définissez, dans une classe *Principale*, un *main*() qui :
crée quelques instances de *Ville, Pays, Unipersonnelle, Société*, et les affiche,
crée une instance de *LesStats*,
y insère les statistiques correspondant aux instances de *Pays, Unipersonnelle, Société,* précédemment créées,
affiche les moyennes présentes dans l'instance de *LesStats*, et le nombre de moyennes affichées.