Travaux Dirigés #2:

Exercice 1 :

1. Peut-on instancier une interface ?une classe abstraite ?
2. Peut-on y mettre un constructeur ?un constructeur avec un corps ?
3. Peut-on écrire le code suivant : A a = new B();
   - si A est une classe abstraite, dérivée par la classe B?
   - si A est une interface, implémentée par une classe B?
4. Une interface/classe abstraite peut-elle contenir des méthodes abstraites? Non-abstraites ? statique et abstraite ?
5. Une interface/classe abstraite peut-elle contenir des attributs?avec quels modificateurs doivent-il être instanciés ?
6. Une interface peut-elle hériter d'une autre interface ?d'une classe abstraite?
7. Une classe abstraite peut-elle hériter d'une autre classe abstraite?d'une interface?

Exercice 2 :

Certains animaux peuvent crier, d'autres sont muets. On représentera le fait de crier au moyen d'une méthode affichant à l'écran le cri de l'animal.

– écrire une interface contenant la méthode permettant de crier.

– écrire les classes des chats, des chiens et des lapins (qui sont muets)

– écrire un programme avec un tableau pour les animaux qui savent crier, le remplir avec des chiens et des chats, puis faire crier tous ces animaux. Décrire ce qui s'affiche à l'écran à l'exécution de ce programme.

Exercice 3 :

On va définir une classe abstraite **Figure** dont voici le début :

```
public abstract class Figure{
    // coordonnées du centre approximatif de la figure
    private int posX;
    private int posY;

    public Figure(int x, int y){
        posX = x;
        posY = y;
    }
    ............
}
```

On définira aussi dans Figure les méthodes concrètes :

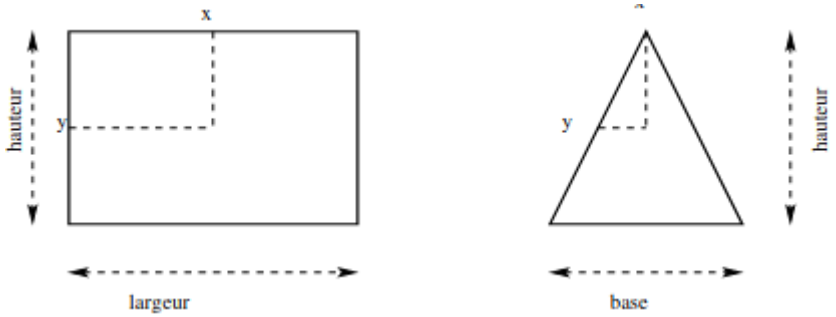
- *public int getPosX()* qui donnera la position horizontale du centre de la figure (abscisse) ;
- *public int getPosY()* qui donnera la position verticale du centre de la figure (ordonnée) ;

- *public abstract void affiche();* qui affichera un résumé des propriétés de la figure.

Écrivez le code de Figure.
On définira aussi les classes concrètes suivantes : Rectangle, Carre et Triangle.
Donnez la hiérarchie de classes que vous choisiriez. Pour l'instant, n' écrivez pas de code pour ces classes.

Exercice 4 :

Écrivez le code de la classe Rectangle. On doit passer en paramètres du constructeur la position du centre, la largeur et la hauteur. Dans la figure ci-dessous,x et y représente la position du centre.



Exercice 5 :

Écrivez le code de la classe Carre. On doit passer en paramètres du constructeur la position du centre et la longueur d'un côté.

Exercice 6 :

Écrivez le code de la classe Triangle. On considère qu'un triangle est toujours isocèle et positionné comme sur le dessin ci-dessus. On doit passer en paramètres du constructeur la position du centre, la base et la hauteur (cf. dessin). Dans la figure ci-dessus, x et y représente la position du centre, y est à la moitié de la hauteur.

Exercice 7 :

Soit les classes suivantes:

une classe Personne qui comporte trois champs privés, nom, prénom et date de naissance.
Cette classe comporte un constructeur pour permettre d'initialiser les données. Elle comporte également une méthode polymorphe Afficher pour afficher les données de chaque personne.

une classe Employé qui dérive de la classe Personne, avec en plus un champ Salaire accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.

une classe Chef qui dérive de la classe Employé, avec en plus un champ Service accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.

une classe Directeur qui dérive de la classe Chef, avec en plus un champ Société accompagné de sa propriété, un constructeur et la redéfinition de la méthode Afficher.

Écrire les classe Personne, Employé, Chef et Directeur.
Créez un programme de test qui comporte tableau de huit personnes avec cinq employés, deux chefs et un directeur (8 références de la classe Personne dans lesquelles ranger 5 instances de la classe Employé, 2 de la classe Chef et 1 de la classe Directeur).

Affichez l'ensemble des éléments du tableau.

Exercice 8 :

1. Faire deux interfaces *A* et *B* et une classe *C* qui implémente ces deux interfaces.
2. Que se passe t-il si les deux interfaces *A* et *B* déclarent une même méthode *f()*? Une même constante *x*?

Indication: pour le savoir, créer une classe Test qui utilise *f()* et *x*.

Exercice 9 :

Dans cet exercice, on manipule des formes géométriques que l'on définit par l'interface suivante.

```
Interface FormeGeometrique {
    double perimetre();
    double surface();
}
```

1. Écrire par exemple les classes *Cercle*, *Triangle*, *Rectangle*, *Carre*.
2. Un polygone est la forme définie par une ligne brisée
   - qui ne se coupe pas elle-même ;
   - qui commence et termine au même point.

Quel sera le type de *Polygone*? Quelle sera sa relation avec les classes précédentes ? (On ne demande pas de vérifier si la ligne se coupe.) On rappelle que si(x1, y1) et (x2, y2) sont deux points du plan, alors la distance les séparant est donnée par : $sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)$

3. Un polygone est convexe si, lorsque l'ont race un segment entre deux points quelconques du polygone, alors tous les points du segment appartiennent au polygone. Si les sommets d'un polygone convexe, pris dans le sens trigonométrique (au contraire du sens des aiguilles d'une montre), sont$(x_1, y_1),(x_2, y_2), \ldots,(x_n, y_n)$ sa surface vaut

$$1/2[(x_1y_2+x_2y_3+x_3y_4+\cdots+x_ny_1)-(y_1x_2+y_2x_3+y_3x_4+\cdots+y_nx_1)]$$

Quel sera le type de *PolygoneConvexe*? Quelle est sa relation avec les types précédents ? (On ne demande pas de vérifier si un polygone est convexe et on supposera que ses sommets sont donnés dans le sens trigonométrique.)