DEVOIR POO en JAVA

-prépare par : **BENHAMMOU MOHAMED (IDAI)**

-encadré par: **PR. El Mokhtar EN-NAIMI**

GitHub

https://github.com/mohamedben12369/devoir_JAVA

Introduction:

Ce rapport aborde les concepts clés de la Programmation Orientée Objet (POO) en Java, en particulier les différences entre une classe et une interface, le rôle des classes abstraites, et leur implémentation dans un exemple concret.

Exercice1:

I)

```
public class TestPoly2 {
    public static void main (String [] args) {
        Graphique [] tab = new Graphique [6];
       //tab[0] = new Graphique (3,2);//Erreur, classe abstraite est non instanciable
      //Même si la classe Graphique est abstraite, il est tout de même possible
     //de déclarer des variables de ce type qui pourront recevoir des objets
//créés à partir des sous-classes concrètes :
        tab[0] = new Cercle (3,2,7);
        tab[1] = new Cercle (10,7,3) ;
        tab[2] = new Rectangle (4,7,8,6) ;
        tab[3] = new Rectangle (8,10,12,10);
        tab[4] = new Cercle (8,5,3) ;
        tab[5] = new Rectangle (10,17,3,8) ;
        for (int i=0 ; i <=5 ; i++) {   tab[i].affiche();   } }
        }
```

Résultat ????

- **Classe Graphique** : Cette classe est définie comme abstraite, ce qui signifie qu'elle ne peut pas être instanciée directement. Elle sert de base pour les autres formes géométriques, comme le Cercle et le Rectangle.

- **Tabulation d'objets Graphique** : Un tableau tab de type Graphique[] est créé pour contenir des objets qui sont des instances concrètes de sous-classes de Graphique. Bien que Graphique soit abstraite, les objets créés dans ce tableau sont des instances de classes concrètes comme Cercle et Rectangle.
- **Instanciation des objets** : Les objets de type Cercle et Rectangle sont instanciés avec des valeurs spécifiques pour leurs coordonnées et dimensions. Par exemple :
- tab[0] = new Cercle(3, 2, 7); crée un cercle avec un centre aux coordonnées (3, 2) et un rayon de 7.
- tab[2] = new Rectangle(4, 7, 8, 6); crée un rectangle avec un coin supérieur gauche aux coordonnées (4, 7) et des dimensions 8x6.
- **Affichage des objets** : La boucle for parcourt le tableau tab et appelle la méthode affiche() sur chaque élément. Cela implique que chaque objet dans le tableau doit implémenter la méthode affiche(), probablement définie dans les sous-classes Cercle et Rectangle. Cette méthode affichera les propriétés spécifiques de chaque objet, telles que les coordonnées et les dimensions pour chaque forme géométrique

```
PS C:\Users\pc\Desktop\devoi JAVA>  & 'C:\Program Files\Java\jdk-23\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,
server=n,suspend=y,address=localhost:60817' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Use
rs\pc\AppData\Roaming\Code\User\workspaceStorage\10610c1bc3ab8f1b259d65c298a62a7c\redhat.java\jdt_ws\devoi JAVA_b1fa29
e3\bin' 'TestPoly2'
Cercle-Centre:(3,2),Rayon:7.0
Cercle-Centre:(10,7),Rayon:3.0
Rectangle-Coin sup gauche: (4,7),Largeur:8.0,Hauteur:6.0
Rectangle-Coin sup gauche: (8,10),Largeur:12.0,Hauteur:10.0
Cercle-Centre:(8,5),Rayon:3.0
Rectangle-Coin sup gauche: (10,17),Largeur:3.0,Hauteur:8.0
PS C:\Users\pc\Desktop\devoi JAVA>
```

II)

Exemple (Devoir no. 1: suite):

```
public interface Affichage {  public void affiche(); }
public abstract class Figure implements Affichage {
        public abstract double perimetre(); // Méthode abstraite
        public abstract double surface(); // Méthode abstraite
}     //Pourquoi cette classe est abstraite??.
```

- **La classe Figure est abstraite** car elle contient des méthodes abstraites (perimetre() et surface()) qui n'ont pas d'implémentation, et ces méthodes doivent être définies dans ses sous-classes concrètes. Elle implémente également l'interface Affichage, qui impose la présence de la méthode affiche(), mais laisse à chaque sous-classe le soin de définir comment afficher ses objets. Ainsi, Figure sert de modèle pour des formes géométriques spécifiques, comme les cercles ou les rectangles, sans pouvoir être instanciée directement. Les sous-classes concrètes hériteront de Figure et fourniront des implémentations spécifiques pour les méthodes abstraites et l'affichage

III)

```
public class Cercle extends Figure {
        public static final double PI = 3.14;
        protected double rayon;
        public Cercle(double rayon) { this.rayon= rayon; }
        public double getRayon() { return rayon; }
        public double perimetre() { return 2*PI*rayon; }
        public double surface() { return PI*rayon*rayon; }
        public void affiche() { ... }
}   //Pourquoi cette classe n'est pas abstraite??.
```

- **La classe Cercle n'est pas abstraite** car elle fournit des implémentations concrètes pour toutes les méthodes abstraites qu'elle hérite de **la classe Figure**. Cela permet à Cercle d'être instanciée directement, ce qui n'est pas possible pour une classe abstraite

Exercice 2:

1.Quelle est la différence entre une interface et une classe ?

- Interface : Une interface définit un contrat, c'est-à-dire un ensemble de méthodes que les classes doivent implémenter. Elle ne contient que des signatures de méthodes (avant Java 8), et des variables qui sont implicitement public, static, et final. Elle ne peut pas être instanciée.
- Classe : Une classe définit à la fois des attributs (état) et des comportements (méthodes). Elle peut être instanciée et peut contenir des méthodes avec ou sans implémentation, des variables d'instance, et des constructeurs.

2. Quel est l'objectif d'une interface ?

L'objectif d'une interface est de définir un ensemble de comportements communs qui peuvent être partagés par différentes classes, même si elles ne sont pas liées par héritage. Elle permet de garantir que certaines méthodes seront présentes dans les classes qui l'implémentent.

3.Qu'est-ce qu'un type en Java ?

En Java, un **type** désigne une catégorie ou une définition qui détermine quel genre de valeur une variable peut contenir, ou quel genre de méthodes un objet peut appeler. Les types peuvent être primitifs (comme int, char, boolean) ou des types de référence (comme les classes, interfaces et tableaux).

4.Quel est le rôle d'une interface Comparable ?

L'interface Comparable permet de définir un ordre naturel pour les objets d'une classe. En implémentant cette interface, une classe indique comment ses objets doivent être comparés entre eux. Cela est utile pour trier des collections d'objets.

5.Comment fonctionne la méthode compareTo() ?

La méthode compareTo() compare deux objets et retourne un entier :

- Un **entier négatif** si l'objet actuel est inférieur à l'objet comparé.
- **Zéro** si les deux objets sont égaux.
- Un **entier positif** si l'objet actuel est supérieur à l'objet comparé.

6.Qu’est-ce qu’un sous-type et qu’est-ce qu’un supertype ?

- **Sous-type** : Un sous-type est une classe ou une interface qui hérite ou implémente un autre type. Par exemple, Chien est un sous-type de Animal si Chien hérite de Animal.
- **Supertype** : Un supertype est une classe ou une interface qui est héritée ou implémentée par un autre type. Par exemple, Animal est un supertype de Chien.

7.Quelle est la différence entre une classe abstraite et une classe concrète ?

- **Classe abstraite** : Une classe abstraite ne peut pas être instanciée directement. Elle peut contenir des méthodes abstraites (sans implémentation) et des méthodes concrètes (avec implémentation). Elle est utilisée pour partager un comportement commun tout en laissant des sous-classes définir certains comportements.
- **Classe concrète** : Une classe concrète peut être instanciée et fournit des implémentations complètes pour toutes ses méthodes.

8.Quelle est la différence entre une classe abstraite et une interface ?

- **Classe abstraite** : Une classe abstraite peut contenir des méthodes concrètes (avec implémentation) et des méthodes abstraites. Elle peut avoir des variables d'instance et des constructeurs.
- **Interface** : Une interface ne contient que des signatures de méthodes (avant Java 8), et des constantes. Elle ne peut pas avoir d'implémentation, sauf si la méthode est marquée default

9.Quel est le rôle d’une classe abstraite ?

Le rôle d’une classe abstraite est de fournir une structure de base avec des méthodes communes et des comportements partagés à ses sous-classes. Elle permet de définir un modèle, tout en laissant la flexibilité d'implémenter des méthodes spécifiques dans les sous-classes.

10.Donner l’Utilité et l’Intérêt de l’interface ?

- **Utilité** : Une interface permet de définir un contrat commun que plusieurs classes non liées peuvent implémenter. Elle est utile pour définir des comportements partagés sans imposer une hiérarchie d'héritage.
- **Intérêt** : Elle permet de faciliter le polymorphisme en permettant à des classes différentes d’être traitées de manière uniforme, ce qui rend le code plus flexible et modulaire.

Conclusion:

Ces concepts fondamentaux de Java permettent de comprendre comment structurer et organiser le code de manière modulaire, réutilisable et extensible. Les interfaces sont utiles pour définir des comportements partagés, tandis que les classes abstraites et concrètes permettent de définir des objets avec des comportements spécifiques.