TP Heritage – Polymorphisme

Sommaire

Déroulement :
Vous allez dans ce TP reprendre l'ensemble des notions abordées dans le support : «06_Heritage_Polymorphisme_Interface ».
Pour cela, vous construirez une hiérarchie de classes et d'interfaces, implémenterez les concepts d'**héritage**, d'**implémentation** et de **composition**.

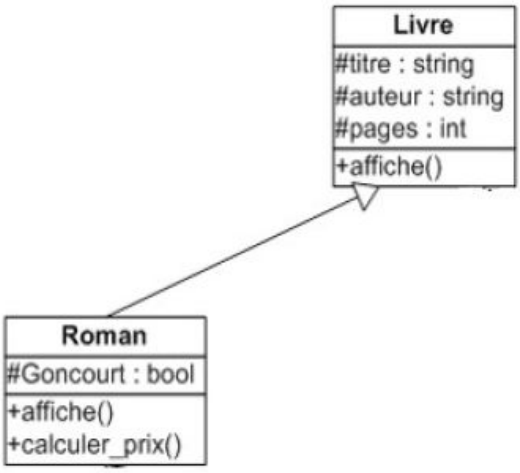
Vocabulaire utilisé : encapsulation, classe, héritage, classe abstraite, interface, polymorphisme, agrégation, …

Environnement technique : J2SE, JDK, un EDI (type *Eclipse/NetBeans*).
La documentation **Java Oracle**:
https://docs.oracle.com/javase/tutorial/java/index.html

Au cours de ce TP, vous mettrez en évidence :

- La notion de **classe usuelle**.
- La notion de **méthode**.
- La notion **d'héritage**.
- La notion de **classe abstraite**.

1.HÉRITAGE UN PETIT PEU DE LITTÉRATURE…



Implémenter la relation d'héritage entre les classes Livre et Roman

Uml: langage de modélisation unifié

– Représentation de la visibilité des membres d'une classe :
  Public : + Privé : - Protégé : #
– Représentation de la relation d'héritage entre classe : ⟶

Le prix d'un Roman est de 0.01 € par pages

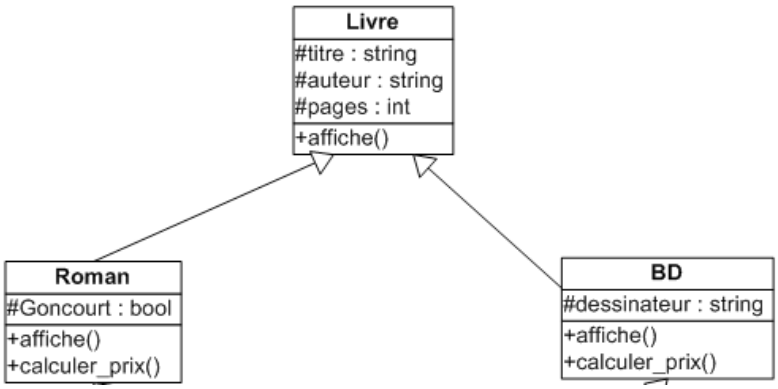
Dans la classe cliente:

```
System.out.println( "*****test classe Roman avec appel explicite constructeur par defaut*******");
roman romanSansParam= new roman();
romanSansParam.affiche();
System.out.println("prix du roman: "+ romanSansParam.calculprix() );
System.out.println("******test classe Roman avec appel explicite constructeur spécialisé********");
roman romanAvecParam = new roman("le meilleur des mondes", " aldous
huxley", 214, true);
romanAvecParam.affiche();
System.out.println("prix du roman: " + romanAvecParam.calculprix());
```

Console

```
*****test classe Roman avec appel explicite constructeur par defaut*******
je passe par le constructeur par defaut de la classe livre initialisation titre auteur nbpage dans le constructeur
je passe par le  constructeur par defaut classe roman ajout de BGoncourt false
affichage des attributs de l'objet instancié:
Titre: Le petit prince  Auteur: Antoine de Saint Exupery  Nombre de pages: 134
 Pas prime au prix goncourt
prix du roman: 1.34
******test classe Roman avec appel explicite constructeur spécialisé********
je passe par le constructeur explicite de la classe livre initialisation titre auteur nbpage
 avec les params transmis par l'utilisateur de la classe
je passe par le  constructeur explicite classe roman ajout de BGoncourt
affichage des attributs de l'objet instancié:
Titre: le meilleur des mondes Auteur:  aldous huxley Nombre de pages: 214
prime prix goncourt
prix du roman: 2.14
```

Suite exercice héritage Implémentez la classe BD selon le diagramme de classe suivant :



Le prix d'une bande dessinée est de 0.3 € par pages.

```
System.out.println("**************test de la classe bd **********");
bede maBede = new bede("Le domaine des dieux", "Goscinny", 48, "Uderzo" ) ;
maBede.affiche();
java.text.DecimalFormat df = new java.text.DecimalFormat("0.##");
 System.out.println("prix bd:   " + df.format(maBede.calculprix()));
```

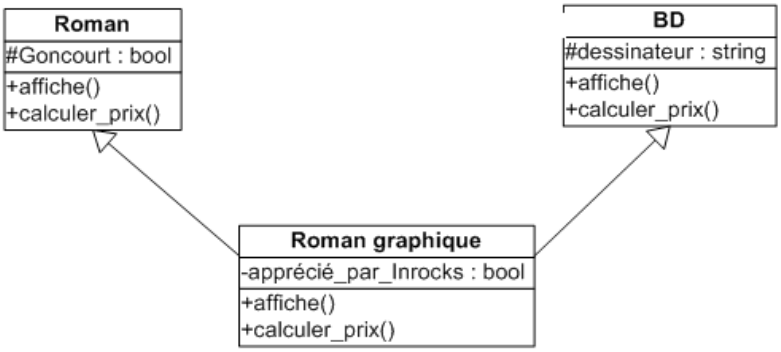
Console

```
**************test de la classe bd **********
je passe par le constructeur explicite de la classe livre initialisation titre auteur nbpage avec les params transmis par l'utilisateur de la classe
je passe par le constructeur explicite classe bd pour ajouter le dessinateur
affichage des attributs de l'objet instancié:
Titre: Le domaine des dieux Auteur: Goscinny Nombre de pages: 48
 nom du dessinateur: Uderzo
prix bd:  14,4
```

Nous ne sommes pas dans le cas d'un héritage multiple mais toujours dans le cadre d'un héritage simple puisque chaque classe fille hérite d'une seule classe mère.

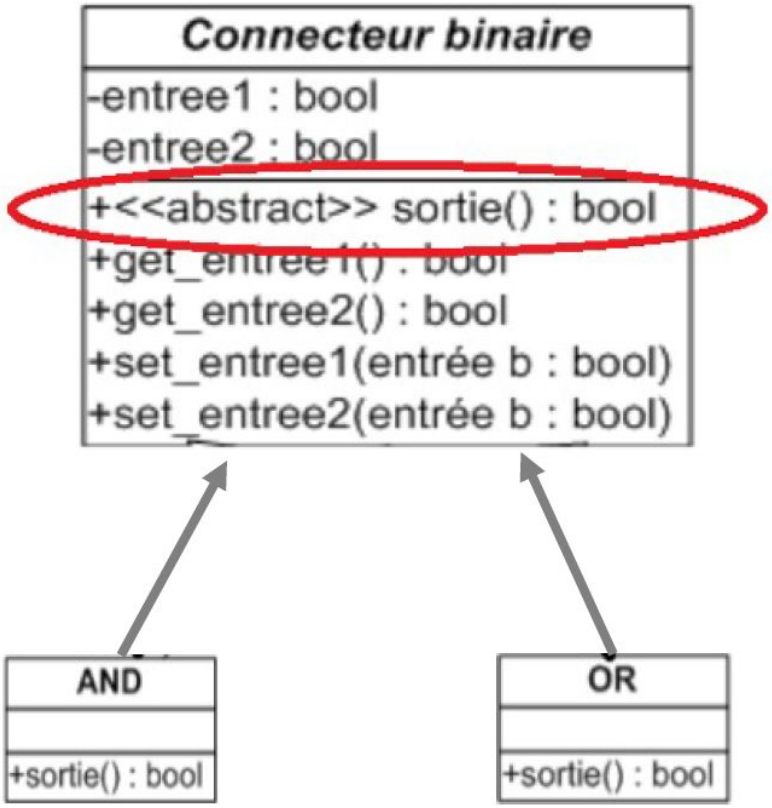
Roman et BD héritent de Livre.

Ci dessous la notation U.M.L traduit un cas d'héritage multiple pour la classe Roman graphique (pour le langage C++) :



classes abstraite sans polymorphisme

2. Portes logiques



Implémenter le diagramme de classe ci-dessus.
Les classe And et Or simulent le comportement respectivement des connecteurs logiques binaires : And, Or
Les connecteurs binaires ont 2 entrées et 1 sortie.
Chaque entrée (input)/ sortie (output) a 2 états possibles → (1/vrai, 0/faux).
Comme les états ne sont pas accessibles directement par l'utilisateur de la classe (encapsulation), ils s'initialisent avec des setters et se récupérent avec des getter. Il est donc pertinent de factoriser ces attributs et méthodes au niveau de la classe mère: la classe connecteur
Cependant l'état de sortie diffère selon la nature de l'opérateur logique (voire table de vérité) donc selon la classe AND, OR.
Conséquence:
le traitement de la méthode sortie ne peut pas se factoriser au niveau de la classe connecteur. On peut seulement imposer la signature de cette méthode (qualifiée d'abstract) dans la classe mère.

De part la relation avec la classe mère, chacune des classes filles se doit de spécialiser sa méthode sortie.

A noter: la classe devient abstract de par la présence de cette méthode abstract.

2. Portes logiques (suite):

Dans le main :
test des classes décrites ci-dessus en affichant l'état des sorties:

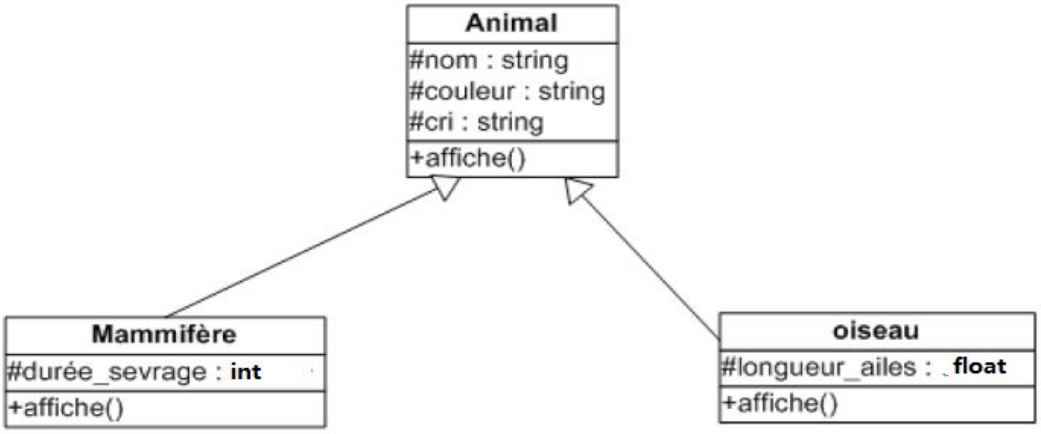
```
System.out.println("Connecteur And");
And MonConnecteurAnd= new And(true, true) ;
System.out.println( MonConnecteurAnd.sortie() );
MonConnecteurAnd= new And(true, false) ;
System.out.println( MonConnecteurAnd.sortie() );

System.out.println("Connecteur Or");
Or MonConnecteurOr = new Or(false, false);
System.out.println(MonConnecteurOr.sortie());
MonConnecteurOr.set_entree1(true);
System.out.println(MonConnecteurOr.sortie());
MonConnecteurOr.set_entree2(true);
System.out.println(MonConnecteurOr.sortie());
```


```
Connecteur And
true
false
Connecteur Or
false
true
false
```

POLYMORPHISME ET CLASSES ABSTRAITES

3.ARCHE DE NOÉ

Implémenter cette relation d'héritage entre les classes Animal (abstraite), Mammifère et Oiseau:
la classe mère est abstraite car pas de sens d'implémenter des animaux mais la méthode affiche de cette classe animal n'est pas abstraite.



1)Dans le programme principal, l'utilisateur saisit un nombre n d'animaux à ranger dans l'arche.
Définir un tableau dynamique d'objet hétérogène nommé « Arche » de n objets de type Animal.
2) On demande ensuite à l'utilisateur la famille de l'animal saisi.
En fonction du choix, on affecte dans une case du tableau Arche un nouveau Mammifère ou bien un nouvel Oiseau.
Les attributs spécifiques à chaque animal sont saisis par l'utilisateur.
Exemple d'utilisation les saisies utilisateurs sont écrites en vert

Console

```
entrer un nombre d'animaux : 2
nom ? Flipper
couleur ? gris-bleu
cri ? bang
type de l'animal --> 1 pour mammifere, 2 pour oiseau  : 1
duree sevrage en mois  ?2
nom ? Toucan
couleur ? rouge
cri ? Tooukaan
type de l'animal --> 1 pour mammifere, 2 pour oiseau  : 2
 longueur des ailes ?0,35
```

3)puis afficher tous les habitants de l'arche avec leurs caractéristiques.

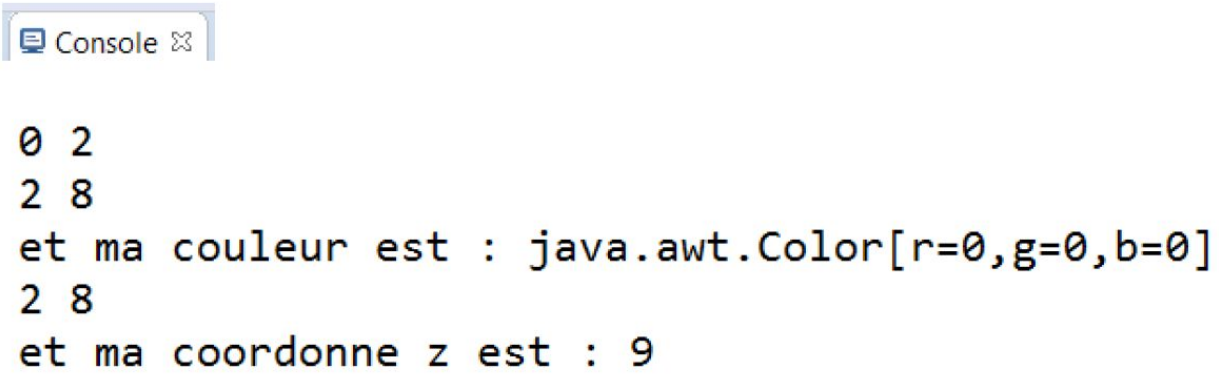
```
les animaux dans l'arche:
Mammifere---->nom : Flipper, couleur : gris-bleu, cri :bang, dure sevrage en mois: 2.0
Oiseau------->nom : Toucan, couleur : rouge, cri :Tooukaan, longueur des ailes : 0,35
```

4 Tableau hétérogène d'objet (polymorphisme particularisation/généralisation)

Ecrire les classes nécessaires au fonctionnement du programme suivant (en ne fournissant que les méthodes nécessaires à ce fonctionnement).

```
public static void main(String[] args){
          Point [] tabPts = new Point [3];
          tabPts [0] = new Point (0, 2) ;
          tabPts [1] = new Pixel (2, 8, Color.black) ;
          tabPts [2] = new Point3D (2, 8, 9)  ;
          for (int i=0 ; i< tabPts.length ; i++)
             tabPts[i].affiche();
}
```

Pour la couleur vous pouvez utiliser la classe java.awt.Color;



utilisez **instanceof** pour déterminer la/les classes d'appartenance des objets