

TD1

Exercice 1:

Voici la source de la classe Livre :

```
public class Livre {  
    // Variables  
    private String titre, auteur;  
    private int nbPages  
  
    // Constructeur  
    public void Livre(String unAuteur, String unTitre) {  
        auteur = unAuteur;  
        titre = unTitre;  
    }  
  
    // Accesseur  
    public String getAuteur() {  
        return auteur;  
    }  
  
    // Modificateur  
    int setNbPages(int n) {  
        nbPages = nb;  
    }  
}
```

- a- Corrigez quelques petites erreurs
- b- ajoutez une méthode main() pour Créer 2 livres, Faire afficher les auteurs de ces 2 livres.

Exercice 2:

Accesseurs et modificateurs

1) Modifiez la classe Livre :

- Ajoutez un accesseur pour la variable titre et la variable nbPages.
- Ajoutez un modificateur pour les variables auteur et titre.
- Changez le modificateur de nbPages : il ne devra changer le nombre de pages que si on lui passe en paramètre un nombre positif, et ne rien faire sinon, en affichant un message d'erreur.

2) Dans la méthode main(),

- a- indiquez le nombre de pages de chacun des 2 livres,
- b- faites afficher ces nombres de pages,
- c- calculez le nombre de pages total de ces 2 livres et affichez-le.

Exercice 3:

- 1) Dans La classe Livre, ajoutez une méthode afficheToi() qui affiche une description du livre (auteur, titre et nombre de pages).
- 2) Ajoutez une méthode toString() qui renvoie une chaîne de caractères qui décrit le livre. Modifiez la méthode afficheToi() pour utiliser toString().
Voyez ce qui est affiché maintenant par l'instruction System.out.println(livre).
- 3) Ajoutez 2 constructeurs pour avoir 3 constructeurs dans la classe :
 - Un constructeur qui n'a pas de paramètre
 - Un qui prend en paramètre l'auteur et le titre du livre,
 - et l'autre qui prend en plus le nombre de pages.

Utilisez les 3 constructeurs (et éventuellement d'autres méthodes) pour créer 3 livres de 300 pages dans la méthode main() de la classe TestLivre.

Exercice 4

1) Contrôle des variables private par les modificateurs

- a- Ajoutez un prix aux livres (nombre de type Java float ou double) avec 2 méthodes getPrix et setPrix pour obtenir le prix et le modifier.
- b- Ajoutez au moins un constructeur qui prend le prix en paramètre.
- c- Testez. Si le prix d'un livre n'a pas été fixé, la description du livre (toString()) devra indiquer "Prix pas encore fixé".
- d- On bloque complètement les prix : un prix ne peut être saisi qu'une seule fois et ne peut être modifié ensuite (une tentative pour changer le prix ne fait qu'afficher un message d'erreur).
Réécrivez la méthode setPrix (et autre chose si besoin est). Vous ajouterez une variable booléenne prixFixe (pour "prix fixé") pour savoir si le prix a déjà été fixé.
- e- Réécrire la méthode main () et prévoir le deux cas (prix non fixé ou bien prix fixé plusieurs fois) afficher le résultat de l'exécution.

Exercice 5

Compléter la classe Cercle suivante :

```
Public class point
{private int abs;
private int ord;
public point ()
{abs =0; ord = 0;}
Public point (int a, int o) {abs = a; ord =o;}
int getabs() {return (abs);}
int getord () {return (ord);}
} // fin de la classe point
```

```
Public class Cercle
{
    private point centre ;
    private double rayon ;
    private double epaisseur ;
    public Cercle ( )
    {.....}

    public Cercle (int a, int o, double rayon, double epaisseur)
    {.....}

    //Méthode surface pour calculer la surface d'un cercle
    {.....}

    //Méthode getrayon pour retourner le rayon
    {.....}

    //Méthode getepaisseur
    {.....}

    //Méthode getabscentre qui retourne l'abscisse de centre
    {.....}

    //Méthode getabsord qui retourne l'ordonnée de centre
    {.....}

    //Méthode égale qui permet de vérifier l'égalité de deux cercles
    {.....}
}
```

Exercice 6

Soit la classe « Livre » écrite en Java, on propose de tester cette classe en utilisant 2 classes de test (Test1 et Test2).

Questions

- 1- Dégager les erreurs qui existent dans les 2 classes de Test en justifiant la réponse.
- 2- Proposer une correction pour chaque erreur dégagée.

//Classe Livre

```
package bibliographie ; //La classe livre appartient au package bibliographie
public class Livre {
    public String titre;
    public String auteur;
    public String edition;
    private int nbre_page;
    public Livre (String t, String a, String e, int n)
    {titre = t ; auteur = a; edition = e; nbre_page = n ;}
    public void Affiche ()
    {
        System.out.println (titre) ;
        System.out.println (« Auteur : » + auteur) ;
        System.out.println (« Edition: » + edition);}
    boolean compare (Livre l)
    {
        return ((l.titre==titre) && (l.edition ==edition) && (l.auteur==auteur) &&
            (l.nbre_page == nbre_page)) ;}
} //fin de la classe livre
```

//Classe Test1

```
package bibliographie;
public class Test1
{
    public static void main (String args[ ])
    {Livre l1;
    Livre l2 = new Livre ("Java2", "Antoine Mirecourt", "Eyrolles", 957);
    Livre l3 = new Livre();
    Livre l4 = new Livre ("Java2", "Pierre Saumont", "Eyrolles", 957);
    if (l2.compare(l4)==true)
        System.out.println("Le même livre");
    else
        System.out.println("Les 2 livres sont différents");

    l2.titre = "Programmer en Java2";
    l2.edition = "Eyrolles 1999";
    l2.nbre_page = 1000;
    l2.Affiche();
```

```
l1.affiche();
}}
```

// Classe Test2

```
package utilitaires;
public class Test2
{
    public static void main (String [] args)
    {
        Livre l1 = new Livre ("Java2", "Antoine Mirecourt", "Eyrolles", 957);
        Livre l2 = new Livre ("Java2", "Pierre Saumont", "Eyrolles", 957);
        if (l1.compare(l2)==true)
            System.out.println("Le même livre");
        else
            System.out.println("Les 2 livres sont différents");

        l1.titre = "Programmer en Java2";
        l1.edition = "Eyrolles 1999";
        l1.nbre_page = 1000;
        l1.Affiche();
    }
}
```

Exercice 7

Soit la classe suivante

```
Public class X
{
    public static int n ;
    public int v;
    public X ( int v)
    {this.v = v; n++;}
}
```

Question : Donner le résultat de l'exécution de cette classe de test

```
public class Test
{public static void main ( String []args)
{X.n = 0;
X monobjet1 = new X (5);
System.out.println ("n= " + monobjet1.n + " v = " + monobjet1.v);
X monobjet2 = new X (14);
System.out.println ("n= " + X.n + " v = " + monobjet2.v );
X monobjet3 = new X (20);
System.out.println ("n= " + X.n + " v = " + monobjet3.v);
}}
```

Exercice 8

Soit l'arborescence suivante constituée des classes et des packages : les packages sont marqués par des astérisques (*)

Compléter les entêtes des classes suivantes :

// Classe cercle

public class Cercle

{.....}

// classe Circulaires *

public

{.....}

public class Cercle

{.....}

public class Ellipse

{.....}

// Pour utiliser les classes dans une classe de test on fait :

import..... ;

..... ;

Public class Test

{.....}

Exercice 9

Etape1 :

Ecrire en Java une classe nommée 'Mot_dict' représentée par les attributs suivants :

- **Mot** : de type String
- **Définition** : de type String qui représente la définition du mot.

Et les méthodes suivantes :

- **String getMot ()** : pour retourner le mot.
- **String getDéfinition ()** : pour retourner la définition.
- **Void setDéfinition (String)** : Pour donner ou changer une définition d'un mot.
- **Void setMot (String)** : pour donner ou changer le mot.

- **Boolean synonyme (String)** : vérifie si un mot est synonyme d'un autre.

Etape 2 :

Ecrire une classe nommée Dictionnaire composée des attributs suivants :

- **nb_mots** : retourne le nombre des mots d'un dictionnaire.
- **Dict** : c'est un tableau de Mot_dict.
- **Nom** : c'est le nom de dictionnaire.

Et les méthodes suivantes :

- 1ou plusieurs **constructeurs** qui permettent d'initialiser la taille et le nom d'un dictionnaire.
- **Void Ajouter_Mot (Mot_dict)** : Ajoute un mot au dictionnaire de façon que le dictionnaire reste toujours trié.
- **Void Trier ()** : trie le dictionnaire selon l'ordre alphabétique.
- **Void Supprimer_Mot (Mot_dict)** : c'est une méthode qui supprime un mot de dictionnaire.
- **String Recherche_dicho (String)** : C'est une méthode qui permet de retourner la définition d'un mot dans le dictionnaire.
- **Void Lister_dictionnaire ()** : permet de lister tout le contenu de dictionnaire.
- **Int Nombre_synonyme (Mot_dict)** : retourne le nombre de synonymes d'un mot.