Trace écrite java Bibliothèque
TP_4

PS :

-Voici la création des objets des clases que j'utiliserais pendant le tp (démonstration)

```
//Creation d'objet livres
Livre livre1=new Livre("978-3-16-148410-0","004. 178 K20PM","Introduction à Java","J. Leblanc","Didier",2015,50);
Livre livre2=new Livre("999-9-99-999999-9","967. 4987 T248O","Structures de Données","M. Machin","Paul",2017,75);
Livre livre3=new Livre();
//Creation CD
CD cd1 = new CD ();
CD cd2 = new CD ();
//Creation DocURL
DocURL docUrl1 = new DocURL();

//Creation d'objet Membre
MembreEtudiant membreEtudiant1=new MembreEtudiant("Patric","Frita","0687532136","15 rue des champignons");
MembrePersonnel membrePersonnel1=new MembrePersonnel("PartDieu","Jean","0784563210","3 rue des Potiers");
//Creation Catalogue Bibliotheque
CatalogueBibliotheque listeDocuments=new CatalogueBibliotheque();
//Creation catalogue membre
ListeMembres listeMembres=new ListeMembres();
```

La bibliothèque contiendra 3 livres 2 cd et 1 document url.

- True correspondra à = Opération Réussie
- False correspondra à = Opération non Réussie

Trace écrite java Bibliothèque
TP_4

1) Démonstration du catalogue pouvant stocker différents types de DocBibliotheque

La classe Catalogue bibliothèque contient 3 listes de chaque type et permet donc de stocker différents types de documents

```
//Livre
private ArrayList<Livre> listeLivres; //type=1
//CD
private ArrayList<CD> listeCds; //type=2
//DocURL
private ArrayList<DocURL> listeDocUrls; //type=3
```

On test donc l'ajout des 6 documents et j'essaye d'ajouter un même document (pour vérifier qu'on ne peut pas avoir de doublons)

```
//On ajoute les Documents dans le catalogue
System.out.println(listeDocuments.ajouterLivre(livre1));
System.out.println(listeDocuments.ajouterLivre(livre2));
System.out.println(listeDocuments.ajouterLivre(livre3));
System.out.println(listeDocuments.ajouterCd(cd1));
System.out.println(listeDocuments.ajouterCd(cd2));
System.out.println(listeDocuments.ajouterDocUrl(docUrl1));
//On test l'ajout en double d'un doc
System.out.println(listeDocuments.ajouterLivre(livre1));
```

Vérifions les actions avec la console

```
run:
true
true
true
true
true
true
true
false
```

On remarque que toutes les actions ont fonctionner sauf la dernière car dans la fonction ajouter, il regarde s'il n'est pas déjà dans le catalogue :
Exemple code pour le livre :

```
if (listeLivres.contains(livre)) //Si la liste contient le document ou pas
{
    return false;
}
else
{
    listeLivres.add(livre);
    return true;
}
```

Suite à ces opérations, on peut les retrouver dans le catalogue en affichant celui-ci :

```
//Affichage des documents du catalogue
listeDocuments.afficherDocuments(4); //1:Livre 2:CD 3:DocUrl 4:TOUT
```

```
Livre 1 =
CodeArchivage = 004. 178 K20PM
ISBN = 978-3-16-148410-0
Titre = Introduction à Java
Auteur = J. Leblanc
Editeur = Didier
Annee = 2015
Nombre de pages = 50
MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere

Livre 2 =
CodeArchivage = 967. 4987 T248O
ISBN = 999-9-99-999999-9
Titre = Structures de Données
Auteur = M. Machin
Editeur = Paul
Annee = 2017
Nombre de pages = 75
MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere

Livre 3 =
CodeArchivage = 123JKMM3
ISBN = 978-3-16-148410-0
Titre = LeMonde
Auteur = Jean
Editeur = Didier
Annee = 1999
Nombre de pages = 50
MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere

Fin des livres
Cd 1 =
    CodeArchivage = 123JKMM3
Titre = LeMonde
Auteur = Jean
Annee = 1999
Liste Morceaux :

MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere
```

```
Cd 2 =
    CodeArchivage = 123JKMM3
Titre = LeMonde
Auteur = Jean
Annee = 1999
Liste Morceaux :

MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere

Fin des Cd
DocURL 1 =
    CodeArchivage = 123JKMM3
Titre = LeMonde
Auteur = Jean
Date de publication = 1999
Url = http://www.siteweb.com/
Description = Blablabla
Fin des DocUrl
```

On remarque que les « listes des Morceaux » des Cds sont vides : car je n'ai pas ajouté de morceaux à eux. Rajoutons des morceaux au Cd1 puisque la classe CD contient aussi une liste :

```
public class CD extends DocBibliotheque{
    //Atributs
    ArrayList<String> listeMorceaux;
```

```
cdl.ajouterMorceau("La Republique");
cdl.ajouterMorceau("La Cite");
System.out.println(cdl.toString());
```

Et Voila :

```
CodeArchivage = 123JKMM3
Titre = LeMonde
Auteur = Jean
Annee = 1999
Liste Morceaux : La Republique
                 La Cite


MembreEmprunteur = Ce document n'est pas emprunter
MembreReservant = Ce document n'est pas reserver
Emplacement = sur Etagere
```

2) Démonstration qu'un document Url ne peut être emprunté

```
//Testons l'emprunt d'un docUrl
System.out.println(docUrl1.emprunter(membreEtudiant1));


Operation Impossible : Ceci est un objet non physique
```

La console me renvoi « Opération impossible » car un document url n'est pas un objet physique.
Comme on ne peut pas réserver un document QUI n'est pas emprunté, toutes les actions (reserver, retourner…) sont impossible.
Cette action se réalise automatiquement grâce à l'overriding dans la classe docUrl :
Il va donc réaliser cette fonction :

```
//Outrepassement
@Override
public String emprunter(MembreBibliotheque membre)
{
    return "Operation Impossible : Ceci est un objet non physique";
}
```

A la place de l'autre (qui est situé dans la classe mère DocBibliotheque)

```
//Action
public String emprunter(MembreBibliotheque membreEmprunteur)
{
    if (etageres==true)
    {
        emprunt=true;
        etageres=false;
        nbrEmprunt++;
        this.membreEmprunteur=membreEmprunteur;

        membreEmprunteur.incrementerNbrDocEmpruntes();
        return "Operation Reussie";


    }
    else if (sectionReservation==true)
    {
        emprunt=true;
        sectionReservation=false;
        nbrDocSectReservation--;
        nbrEmprunt++;
        this.membreEmprunteur=membreEmprunteur;
        this.membreReservant=null;
        return "Operation Reussie";

    }
    else
        return "Operation Impossible";
}
```

3) Démonstration qu'un membre ne peut emprunter plus de 4 documents

Essayons d'emprunter des documents avec un membre Etudiant (donc limité à 4 emprunts)

On envoit donc 3 paramètre : listeDocuments.empruntеDoc(ID,membre,TYPE)

Id= position de l'objet dans la liste (comme à 0)

Membre = Objet membre (membreEtudiant1 ou membreEtudiant2)

TYPE = Nous demandons à l'utilisateur quel « type » de document veut-il emprunter
(soit Livre, cd, docUrl)
=1 =2 =3

Si le type docUrl est choisi, le programme renvoit une erreur (Operation impossible objet non physique… (conf 2))

Empruntons alors 3 livres et 1 cd et essayons d'emprunter un 5ème document (le cd2)

```
System.out.println(listeDocuments.empruntеDoc(0,membreEtudiant1,1));     //Id = 0 = livre 1/ Type = 1 = Livre
System.out.println(listeDocuments.empruntеDoc(1,membreEtudiant1,1));     //Id = 1 = livre 2
System.out.println(listeDocuments.empruntеDoc(2,membreEtudiant1,1));     //Id = 1 = livre 3
System.out.println(listeDocuments.empruntеDoc(0,membreEtudiant1,2));     //Id = 0 = cd1 / Type = 2 = Livre
System.out.println(listeDocuments.empruntеDoc(1,membreEtudiant1,2));     //Id = 0 = cd2  / Type 2 = Livre
```

Réponse de la console :

```
true
true
true
true
Operation impossible : Quota de document emprunte maximum atteint
```

Le quota est dépassé donc l'étudiant ne peut pas emprunter un 5 ème document tant qu'il n'a pas retourner ses anciens documents.

Cela fonctionne aussi grâce à de l'overriding :

```
//Actions
@Override
public boolean peutEmprunterAutreDocument()
{
    return nbrDocEmpruntes<4;
}
```

Il retourne true si nbrDocEmpruntes<4

Sinon, il retourne false.

Voici la fonction qui regarde le nbrDocEmpruntes avant de faire l'emprunt

```
public boolean empruntеDoc(int indiceDoc, MembreBibliotheque membre, int choix) //choix = livre 1 / cd
{
    if(membre.peutEmprunterAutreDocument())
    {
        switch(choix)
        {
            case 1://Livre
                if (indiceDoc < listeLivres.size())
                {
                    if (listeLivres.get(indiceDoc).emprunter(membre).equals("Operation Reussie"))
                    {
                        return true;
                    }
                    else
                    {
                        System.out.println("Operation impossible");
                        return false;
                    }
                }
                else
                {
                    System.out.println("Erreur d'indice Doc");
                    return false;
                }
```

```
else
{
    System.out.println("Operation impossible : Quota de document emprunte maximum atteint");
    return false;
}
```

Faisons aussi le test avec un membrePersonnel (qui peut emprunter au maximum 8 documents

```
System.out.println(listeDocuments.empruntedoc(0,membrePersonnel1,1));    //Id = 0 = livre 1/ Type = 1 = Livre
System.out.println(listeDocuments.empruntedoc(1,membrePersonnel1,1));    //Id = 1 = livre 2
System.out.println(listeDocuments.empruntedoc(2,membrePersonnel1,1));    //Id = 1 = livre 3
System.out.println(listeDocuments.empruntedoc(0,membrePersonnel1,2));    //Id = 0 = cd1 / Type = 2 = Livre
System.out.println(listeDocuments.empruntedoc(1,membrePersonnel1,2));    //Id = 0 = cd2  / Type 2 = Livre
```

```
true
true
true
true
true
```

On en conclut que l'opération à réussie car la classe membrePersonnel contient un override différent :

```
@Override
public boolean peutEmprunterAutreDocument()
{
    return nbrDocEmpruntes<8;
}
```