1. Projet

1.1. Créer projet [NomPrenom]_Prog_Synth_H15

2. Packages

2.1. modele

2.2. vue

2.3. controleur

3. Objets package modele

3.1. Créer objets

3.1.1. Livre

3.1.2. Membre

3.1.3. Bd

3.1.4. Roman

3.1.5. Revue

3.1.6. Dictionnaire

3.1.7. Editeur

3.1.8. Auteur

3.2. Ajouter relations d'héritage

3.3. Tous les objets doivent implémenter l'interface Serializable :

public class BD **extends** Livre **implements** Serializable

3.4. Créer toutes les propriétés de chaque objet

3.4.1. Selon diagramme de classes

3.5. Créer fonctions / méthodes

3.5.1. Générer getters + setters

4. Objets / classes package controleur

4.1. Fichier

4.2. Reservation

4.3. Prêt

4.4. Main

4.4.1. Seule la classe Main doit avoir la fonction main (sauf pour des fins de tests, évidemment). Ce sera votre point d'entrée dans l'application

4.5. Les obvjets Reservation et Prêt doivent implémenter l'interface Serializable :

public class BD **extends** Livre **implements** Serializable

4.6. La classe Fichier est « final », ce qui permet de l'utiliser sans avoir à l'instancier :

public final class Fichier

Fichier.*Ecriture*(listMembre);

5. Objets / classes package vue

5.1. Menu

6. Fichiers

6.1. Il doit y avoir un fichier par objet

6.2. Le fichier doit porter le nom de l'objet

6.2.1. Livre.txt

6.2.2. Revue.txt

6.2.3. Dictionnaire.txt

6.2.4. BD.txt

6.2.5. Roman.txt

6.2.6. Editeur.txt

6.2.7. Auteur.txt

6.2.8. Membre.txt

6.2.9. Pret.txt

6.2.10. Reservation.txt

6.3. Les fichiers doivent être enregistrés sur System.getProperty("user.home")/Bibliotheque

6.3.1. Une variable nommée path doit être crée à cet effet, et contenir la chaine de caractère du chemin des fichier de l'application

7. Classe Menu

7.1. Ajoutez une fonction pour chaque menu

7.1.1. MenuPrincipal

```
******************** BIBLIOTHEQUE ********************
*                                                    *
* 1 - Membres                                        *
* 2 - Livres                                         *
* 3 - Reservation                                    *
* 4 - Pret                                           *
* 5 - Quitter                                        *
*                                                    *
******************************************************
VOTRE CHOIX: 3
```

7.1.2. MenuMembre

```
********************** Membres **********************
*                                                   *
* 1 - Ajouter                                       *
* 2 - Supprimer                                     *
* 3 - Modifier                                      *
* 4 - Retour au menu principal                      *
*                                                   *
*****************************************************
```

7.1.3.MenuMembreSupprimer

```
***************** Membre - Supprimer ****************
*                                                   *
* 1 - Par numéro de membre                          *
* 2 - Afficher liste                                *
* 3 - Retour au menu Membres                        *
*                                                   *
*****************************************************
```

7.1.4.MenuMembreModifier

```
***************** Membre - Modifier *****************
*                                                   *
* 1 - Par numéro de membre                          *
* 2 - Afficher liste                                *
* 3 - Retour au menu Membres                        *
*                                                   *
*****************************************************
```

7.1.5.MenuLivre

```
********************** Livres **********************"
*                                                   *
* 1 - Ajouter                                       *
* 2 - Supprimer                                     *
* 3 - Modifier                                      *
* 4 - Retour au menu principal                      *
*                                                   *
*****************************************************
```

7.1.6.MenuLivreSupprimer

```
***************** Livre - Supprimer *****************
*                                                   *
* 1 - Par ISBN                                      *
* 2 - Afficher liste                                *
* 3 - Retour au menu Livres                         *
*                                                   *
*****************************************************
```

7.1.7.MenuLivreModifier

```
***************** Livre - Modifier ******************
*                                                   *
* 1 - Par ISBN                                      *
* 2 - Afficher liste                                *
* 3 - Retour au menu Livres                         *
*                                                   *
*****************************************************
```

7.1.8.MenuReservation

```
********** Reservation **********
*                                *
* 1 - Ajouter                    *
* 2 - Supprimer                  *
* 3 - Modifier                   *
* 4 - Retour au menu principal   *
*                                *
**********************************
```

7.1.9. MenuReservationSupprimer

```
********** Reservation - Supprimer **********
*                                           *
* 1 - Par numéro de réservation             *
* 2 - Par numéro de membre                  *
* 3 - Afficher liste                        *
* 4 - Retour au menu Reservation            *
*                                           *
*********************************************
```

7.1.10. MenuReservationModifier

```
********** Reservation - Modifier **********
*                                           *
* 1 - Par numéro de réservation             *
* 2 - Par numéro de membre                  *
* 3 - Afficher liste                        *
* 4 - Retour au menu Reservation            *
*                                           *
*********************************************
```

7.1.11. MenuPret

```
'********** Pret **********
'*                        *
'* 1 - Ajouter            *
'* 2 - Supprimer          *
'* 3 - Modifier           *
'* 4 - Retour au menu principal *
'*                        *
'**************************
```

7.1.12. MenuPretSupprimer

```
********** Prêt - Supprimer **********
*                                     *
* 1 - Par numéro de réservation       *
* 2 - Par numéro de membre            *
* 3 - Afficher liste                  *
* 4 - Retour au menu Prêt             *
*                                     *
***************************************
```

7.1.13. MenuPretModifier

```
********** Prêt - Modifier **********
*                                     *
* 1 - Par numéro de réservation       *
* 2 - Par numéro de membre            *
* 3 - Afficher liste                  *
* 4 - Retour au menu Prêt             *
*                                     *
***************************************
```

7.2. Tous les menus doivent renvoyer à la fonction Main qui l'appelle un nombre représentant la sélection de l'utilisateur

7.3. La classe Menu est « final », ce qui permet de l'utiliser sans avoir à l'instancier :

public final class Menu

menuSelect = Menu.*MenuPrincipal*();

7.4. La classe Menu ne sert qu'à afficher les menus, et rien d'autre. Elle n'exécute aucune autre fonction.

8. Classe Fichier - Chargement

8.1. Dans la classe Fichier, ajouter une méthode OnLoad() qui lit tous les fichiers présents dans le répertoire au démarrage de l'application, qui place les objets de chaque fichier dans une ArrayList, et chaque ArrayList d'objets dans une autre ArrayList pour enfin retourner cette ArrayList d'ArrayList.

```
Algorithme Fonction OnLoad : ArrayList<ArrayList<?>>

Création ObjectInpusStream ois
Creation ArrayList d'ArrayList al

Création File folder System.getProperty("user.home") + "/Bibliotheque"
Création tableau File listOfFiles folder.listFiles();

POUR i allant de 0 à listOfFiles.grandeur
      Création File file = listOfFiles[i]
      SI (file est un fichier ET nom du fichier se termine par « .txt »  ALORS
            ESSAYER
                  Création Objet obj
                  obj PREND_LA_VALEUR (LIRE file avec ois)
                  ajouter obj dans al
                  fermer ois
            ATTRAPER EXCEPTIONS

retourner al
```

Afin d'être en mesure de tester votre Fonction OnLoad, vous devriez ajouter la fonction main dans la classe Fichier;

- Créez une variable pour chaque objet :
  Auteur auteur = **new** Auteur();
- Créez une ArrayList pour chaque objet :
  ArrayList<Auteur> maListeAuteur = **new** ArrayList<Auteur>();
- Ajoutez l'objet dans la ArrayList :
  maListeAuteur.add(auteur);
  - Il est suggéré d'ajouter plusieurs fois un objet dans la liste. Vous pouvez ajouter plusieurs fois le même. Il sera plus facile de tester par la suite.

- Écrivez la liste d'objets dans le fichier :

```
ObjectOutputStream oos = new ObjectOutputStream(new
BufferedOutputStream(new FileOutputStream(new
File(System.getProperty("user.home") +
"/Bibliotheque/Auteur.txt"))));

oos.writeObject(maListeAuteur);
oos.close();
```

- Appelez la fonction OnLoad :

  Fichier.*OnLoad*();

- Vous pourrez ensuite démarrer l'application en mode debug, entrer dans la fonction OnLoad et vérifier que vous avez bien une ArrayList d'ArrayList d'objets de chaque type, et vérifier que chaque objet que vous avez écrit dans le fichier est bien présent dans la liste.

9. Classe Main - Chargement

9.1. Objets

9.1.1. Ajouter une arrayList d'ArrayList mesObjets

9.1.1.1. Ajouter une ArrayList pour chaque objet du modele. L'ArrayList doit être du type de l'objet.

9.1.1.2. Appeler la fonction Fichier.OnLoad afin de lire tous les fichiers, et de charger en mémoire tous les objets.

9.1.1.3. Séparer les ArrayList par type d'objets

9.1.1.3.1. Pour chaque ArrayList contenue dans mesObjets, créer une ArrayList maListeObjets, l'affecter et obtenir le nom de sa classe.

9.1.1.3.2. Selon le nom de la classe obtenu, affecter la liste d'objets créée précédemment selon son type.

Ex. :

```
switch (maListeObjets.get(0).getClass().getSimpleName()) {
        case "Auteur":
                listAuteur = (ArrayList<Auteur>) maListeObjets;
```

Toutes les listes devraient contenir les objets de chaque fichier à ce point.

9.2. Boucle principale

9.2.1. Suite à ces opérations, le programme commencera le cours normal de son exécution.

9.2.1.1. Ajoutez la boucle appropriée

9.2.1.2. Ajoutez un switch qui sert à appeler les menus appropriés et contrôler les choix de l'utilisateur (différents menus)

9.2.2. Dans la switch, les fonctions de la classe Menu devraient être appelées :

```
while (true) {

        menuSelect = Menu.MenuPrincipal();
        .
        .
        .
        case 2: // livres

                    menuSelect = Menu.MenuLivre();
```

9.2.3. Le menu quitter devrait inclure l'appel System.exit(0)

9.2.4. Le cas default devrait signaler à l'utilisateur Votre choix est invalide, veuillez sélectionner un item du valide du menu., et ensuite lui afficher à nouveau le menu principal

9.2.5. Assurez-vous que tous les menus et sous-menus sont bien appelés et gérés (préparons-les pour plus tard, quand nous ajouterons du code!)

10. Classe Fichiers - Écriture

10.1. Dans la classe Fichiers, ajoutez une fonction Ecriture, qui servira à écrire dans chaque fichier correspondant les ArrayList.

```
Algorithme Fonction Ecriture (ArrayList<?> listeObjets)

Création ObjectOutputStream oos
Creation String type

type PREND_LA_VALEUR listeObjets.getClass().getSimpleName();

ESSAYER
      oos PREND_LA_VALEUR new ObjectOutputStream(new
BufferedOutputStream(new FileOutputStream(new File(path + "/" + type
+ ".txt"))));
      oos.writeObject(listeObjets);
      fermer oos
ATTRAPER EXCEPTIONS
```

Chaque fois que cette méthode sera appelée, on lui envoie en paramètre la liste d'objets à écrire dans son fichier.

On crée ensuite une chaîne de caractère, qui servira à mémoriser le nom de la classe d'objets. Ce nom est aussi le nom du fichier dans lequel on doit enregistrer les données. Il faut ensuite ajouter le path et l'extension de fichier pour accéder au bon fichier, et écrire la liste d'objets.

Chaque fonction Ajouter aura besoin d'écrire une liste d'objets, aussitôt que l'utilisateur (bibliothécaire) aura ajouté un objet dans sa liste.

11. Classe Menu – Ajout – Suppression – Modification

Ajoutez les menus nécessaires pour l'ajout, la suppression et la modification des objets :

Le menu renverra en valeur de retour une valeur de différent type, selon ce qui est requis

Certains menus (suppression et modification) doivent prendre en paramètre la liste des objets, afin de la faire afficher et demander à l'utilisateur l'id de l'objet à modifier ou supprimer de la liste

11.1. Ajout

11.1.1. Aucune fonction n'est nécessaire, puisqu'on ajoute directement dans la liste, et qu'on fait l'écriture dans le fichier.

11.2. Suppression

11.2.1. MenuLivreSupprimerListe

```
************* Livre - Supprimer - Liste **************
*                                                      *
ID      TITRE
0       null
0       null
0       null
Entrez l'ID du livre à supprimer: 0
```

11.2.2. MenuMembreSupprimerID

```
* Entrez l'ID du membre à supprimer: 33
```

11.2.3. MenuPretSupprimerNoMembre

```
************ Prêt - Supprimer - No Membre ************
*                                                        *
ID Membre       Nom             Prénom          IDPret
0               null            null            0
0               null            null            0
0               null            null            0
Entrez l'ID du membre dont il faut supprimer le prêt:
```

11.2.4. MenuReservationSupprimerNoMembre

```
******** Reservation - Supprimer - No Membre *********
*                                                      *
ID Membre       Nom             Prénom          IDPret
0               null            null            0
0               null            null            0
0               null            null            0
Entrez l'ID du membre dont il faut supprimer la réservation:
```

11.3. Modification

11.3.1. MenuLivreModifierISBN

```
* Entrez l'ISBN à modifier:
```

11.3.2. MenuLivreModifierListe

```
************ Livre - Modifier - Liste **************
*                                                    *
ID      TITRE
0       null
0       null
0       null
Entrez l'ID du livre à modifier:
```

11.3.3. MenuMembreModifierID

```
* Entrez l'ID du membre à modifier:
```

11.3.4. MenuMembreModifierListe

```
************ Membre - Supprimer - Liste **************
*                                                    *
ID      Nom     Prénom
0       null    null
0       null    null
0       null    null
Entrez l'ID du membre à modifier:
```

11.3.5. MenuPretModifierNoLivre

```
************ Prêt - Modifier - No Livre ************
*                                                    *
ID Livre        Titre           IDPret
0               null            0
0               null            0
0               null            0
Entrez l'ID du livre dont il faut modifier le prêt:
```

11.3.6. MenuReservationModifierNoRes

```
Entrez le no de réservation à modifier:
```

Plusieurs fonctions de suppression et modification prendront en paramètre d'entrée une ArrayList;

MenuLivreSupprimerListe(ArrayList<Livre> liste)
MenuPretSupprimerNoMembre(ArrayList<Pret> liste)
MenuReservationSupprimerNoMembre(ArrayList<Reservation> liste)
MenuLivreModifierListe(ArrayList<Livre> liste)
MenuMembreModifierListe(ArrayList<Membre> liste)
MenuPretModifierNoLivre(ArrayList<Pret> liste)

En effet, si aucune liste n'est envoyée, il ne sera pas possible de faire afficher cette liste à l'utilisateur afin qu'il sélectionne l'item à modifier ou supprimer, pour le renvoyer à la classe Main.

12. Classe Main – Ajout Suppression - Modification

Chaque fois qu'un objet est ajouté, supprimé ou modifié dans le programme, on doit l'écrire sur le champ dans le fichier correspondant, afin que le fichier soit synchronisé avec la liste en mémoire dans le programme.

12.1. Ajout

12.1.1. Pour chaque item du menu qui nécessite l'ajout d'un objet dans la liste en mémoire, ajoutez l'objet dans la liste et appelez ensuite la fonction d'Ecriture;

```
menuSelect = Menu.MenuLivre();

switch (menuSelect) {
	case 1: // ajouter

		Livre livre = new Livre();

		// donner ses valeurs au membre ici

		listLivre.add(livre);

		Fichier.Ecriture(listLivre);

.
.
.
```

12.2. Suppression

12.2.1. Pour chaque item du menu qui nécessite la suppression d'un objet dans la liste en mémoire, supprimez l'objet dans la liste et appelez ensuite la fonction d'Ecriture, afin e synchroniser les fichiers avec les listes contenues en mémoire;

```
case 2: // supprimer

 int menu = Menu.MenuLivreSupprimer();

 switch (menu){
	case 1: // supprimer par ISBN

		String isbn = Menu.MenuLivreSupprimerISBN();
		for(Livre liv : listLivre){

			if(liv.getIsbn().equalsIgnoreCase(isbn)){

				listLivre.remove(liv);
			}
		}
```

```
Fichier.Ecriture(listLivre);

break;
```

12.2.2. Notez que pour chaque objet le comportement peut être légèrement différent, dépendamment des possibilités offertes par le menu. Par exemple, le menu des livres nous permet de supprimer un objet livre soit par son ISBN, soit par son id, en affichant une liste afin que l'utilisateur (bibliothécaire) puisse sélectionner un livre parmi la liste affichée. Les valeurs retournées seront donc différentes en fonction des différentes options possibles.

12.2.3. Les comparaisons s'effectuant sur des String sont également différentes de celles effectuées sur des numériques. Vous devrez donc adapter les portions de code en fonction des particularités de chaque menu / sous-menu.

```
case 2: // supprimer par ID (Afficher Liste)
      int id = Menu.MenuLivreSupprimerListe(listLivre);
      for (Livre liv : listLivre) {

            if (liv.getId() == id) {

                  listLivre.remove(liv);
                  Fichier.Ecriture(listLivre);

            }
      }
```

12.3. Modification

12.3.1. Pour chaque item du menu qui nécessite la modification d'un objet dans la liste en mémoire, modifiez l'objet dans la liste et appelez ensuite la fonction d'Ecriture, afin de synchroniser les fichiers avec les listes contenues en mémoire;

```
case 3: // modifier

      int modif = Menu.MenuLivreModifier();

      switch (modif) {

            case 1: // modifier par ISBN

                  String isbn = Menu.MenuLivreModifierISBN();

                  for (Livre liv : listLivre) {

                        if (liv.getIsbn().equalsIgnoreCase(isbn)) {

                              // ici ajouter modification objet;
                              Fichier.Ecriture(listLivre);
```

```
                    }
                }
```

13. Correction à effectuer

Il faut modifier l'item 1 du menu Prêt Modifier :

```
****************** Prêt - Modifier *****************
*                                                  *
* 1 - Par numéro de livre                          *
* 2 - Par numéro de membre                         *
* 3 - Afficher liste                               *
* 4 - Retour au menu Prêt                          *
*                                                  *
****************************************************
```

14. Réservation

14.1. Reservation

14.1.1. Ajouter un constructeur à Reservation.

14.1.1.1. Celui-ci devrait prendre en paramètre d'entrée l'id, la date de réservation. La date de fin de réservation, le livre réservé, et le membre effectuant la réservation.¨

14.1.1.2. Lorsqu'une nouvelle réservation est effectuée, il faut modifier le statut du livre et le mettre à 2 (dans le constructeur)

14.2. Livre

14.2.1. Ajouter un constructeur prenant en paramètre tous les champs de la classe.

14.2.2. Affecter tous les champs de la classe avec les paramètres du constructeur

14.3. Membre

14.3.1. Ajouter un constructeur prenant en paramètre tous les champs de la classe

14.3.2. Affecter tous les champs de la classe avec les paramètres du constructeur

14.4. Main

14.4.1. Lorsqu'une nouvelle réservation est ajoutée, il faut demander à l'utilisateur d'entrer toutes les informations pertinentes dans le case 3 (réservation) :

14.4.1.1. Le membre qui réserve le livre

14.4.1.1.1. Normalement, nous devrions sélectionner le membre parmi la liste des membres existants. Pour des fins de simulation, nous entrerons chaque fois manuellement toutes les informations du membre et en créerons un nouveau. Il faut donc demander à l'utilisateur toutes les informations du membre

14.4.1.2. Le livre réservé

14.4.1.2.1. Normalement, nous devrions sélectionner le membre parmi la liste des membres existants. Pour des fins de simulation, nous entrerons chaque fois manuellement toutes les informations du membre et en créerons un nouveau. Il faut donc demander à l'utilisateur toutes les informations du livre.

14.4.1.3. La date de début réservation (aujourd'hui)

14.4.1.4. La date de fin de réservation (dans 15 jours)

14.4.1.4.1. Afin de calculer automatiquement la date dans 15 jours, il faut utiliser l'objet Calendar;

```
Date dateReservation = new Date(); // new Date() = aujourd'hui
Date dateFinReservation;

Calendar cal = Calendar.getInstance();
cal.setTime(dateReservation);
cal.add(Calendar.DATE, 15); // ajouter 15 jours

dateFinReservation = cal.getTime();
```

14.4.1.5. L'id devrait être généré automatiquement (simulation) :

int id = (int) (Math.random() * 1000);

14.4.2. Une fois toutes les informations obtenues concernant la nouvelle reservation, il faut instancier la nouvelle réservation :

14.4.3. Il faut ensuite l'ajouter à la liste des réservations :

14.4.4. Puis écrire immédiatement la liste de réservations dans son fichier afin que les données en mémoire dans le programme se « synchronisent » avec les fichiers :

```
// ici ajouter println et scanner pour
// demander toutes les infos a l'utilisateur

// instancier membre
membre = new Membre(idMemb, nom, prenom, adresse, codePostal,
telephone, statutMembre);

// instancier Livre
livre = new Livre(id, isbn, titre, auteur, editeur, statut,
nbrPages);
// creer nouvelle reservation avec toutes les infos obtenues
Reservation res = new Reservation(idReservation, dateReservation,
dateFinReservation, livre, membre);

// ajouter reservation dans liste
listReservation.add(res);
Fichier.Ecriture(listReservation);
```

15. Prêt

15.1. Prêt

15.1.1. Ajouter un constructeur à Pret.

15.1.1.1. Celui-ci devrait prendre en paramètre d'entrée l'id, la date de prêt, La date de fin de prêt, le livre emprunté, et le membre effectuant l'emprunt

15.1.1.2. Vérifier si le livre avait été réservé avant d'être emprunté (vérifier dans la liste de réservations s'il y a une réservation pour ce **id livre ET id membre**);

15.1.1.2.1. Si le livre a été réservé par ce membre, annuler la réservation (mettre dateFinReservation = dateEmprunt)

15.1.1.2.2. Si le livre a été réservé par un autre membre, le prêt ne peut s'effectuer : il faut afficher un message à l'utilisateur.

15.1.1.3. Vérifier le statut du livre

Statut	Description	Action
0	Disponible	Peut être emprunté ou réservé
1	Emprunté	Afficher message livre non disponible
2	Réservé	Vérifier pour quel membre il est réservé
3	En réparation	Afficher message livre non disponible
4	Perdu ou incertain	Afficher message livre non disponible
5	Hors service	Afficher message livre inexistant

15.1.1.3.1. Lorsqu'un nouveau prêt est effectué, il faut modifier le statut du livre et le mettre à 1 (dans le constructeur).

15.1.1.4. Ajouter le prêt à la liste de prêt

15.1.1.5. Écrire la liste dans le fichier approprié