**RAPPORT DU PROJET JAVA : ARBRE généalogique**

Licence 3 MIAGE-GI

PROFESSEUR :MR. GORE BI

MEMBRE DU GROUPE :

-YAO Josué

-BAKAYOKO Myriam ana

-LAGO aya Josiane

# Représentation

# CONCEPTION

L’application est constituée de 4 fichiers. Le but étant de pouvoir implémenter un arbre généalogique et naviguer entre les membres de l’arbre. C’est fichier sont :

* Personne : Qui représente un membre de l’arbre généalogique. Il est lié à ses frères, ses enfants et son parent (tous des objets de type personne) afin de permettre de recherches récurssives ;
* ArbreGenealogique : Qui représente l’arbre Généalogique d’une famille. Cet arbre est caractérisé par l’élément root (Racine) étant le plus haut. Il implémente des algorithmes de recherche afin de trouvé un membre dans l’arbre, représenté l’arbre de manière à comprendre les relations entre les différents membres, naviguer entre les différents éléments de l’arbre, sauvegardé l’arbre ;
* Process : Qui réalise tous les traitements, permet de naviguer dans l’application ;
* Main : Qui permet de démarrer l’application.

# CODE COMPLET DE L’APPLICATION

### Main

*package* com.projet.java;  
  
*public class* Main {  
  
 *public static void* main(String[] args) {  
 Process process = *new* Process();  
 process.mainMenu();  
 }  
}

### Personne

*package* com.projet.java;  
  
  
*import* java.io.*Serializable*;  
*import* java.util.LinkedList;  
*import* java.util.Scanner;  
*import* java.util.UUID;  
  
*public class* Personne *implements Serializable* {  
 String nom;  
 String prenoms;  
 String sexe;  
 String statut;  
 Personne parent;  
 Integer age;  
 LinkedList<Personne> enfants;  
 LinkedList<Personne> freres;  
 *private* UUID id;  
  
  
  
 */\* <<<<< DEBUT CONSTRUCTEURS >>>>> \*/  
 public* Personne(String nom, String prenoms, String sexe, Integer age){  
  
 *this*.nom = nom;  
 *this*.prenoms = prenoms;  
 *this*.sexe = sexe;  
 *this*.age = age;  
 *this*.parent = *null*;  
 *this*.freres = *new* LinkedList<>();  
 *this*.enfants = *new* LinkedList<>();  
 *this*.id = UUID.*randomUUID*();  
 }  
  
 *public* Personne(String nom, String prenoms, String sexe, Personne parent, Integer age){  
  
 *this*.nom = nom;  
 *this*.prenoms = prenoms;  
 *this*.sexe = sexe;  
 *this*.age = age;  
 *this*.parent = parent;  
 *this*.freres = *new* LinkedList<>();  
 *this*.enfants = *new* LinkedList<>();  
 *this*.id = UUID.*randomUUID*();  
 }  
  
 *public* Personne(String option){  
 *if* (option.equals("New")){  
 Scanner sc = *new* Scanner(System.*in*);  
 System.*out*.print("Entrer le nom: ");  
 *this*.nom = sc.nextLine();  
 System.*out*.print("Entrer le prenom: ");  
 *this*.prenoms = sc.nextLine();  
 System.*out*.print("Entrer l'age: ");  
 *this*.age = sc.nextInt();  
 *while*(*true*)  
 {  
 Process.*affichage*("\nSexe:\n 1. Masculin\n 2. Feminin\nChoix: ");  
  
 *int* opt=Integer.*parseInt*(sc.next());  
 *if*(opt==1)  
 {  
 *this*.sexe = "Masculin";  
 *break*;  
 }  
 *else if*(opt==2)  
 {  
 *this*.sexe = "Féminin";  
 *break*;  
 }  
 *else* {  
 Process.*affichage*("\nMauvais choix\nVeuiller recommencer... ");  
 }  
 }  
 }  
 *this*.parent = *null*;  
 *this*.freres = *new* LinkedList<>();  
 *this*.enfants = *new* LinkedList<>();  
 *this*.id = UUID.*randomUUID*();  
  
 }  
  
 */\* <<<<< FIN CONSTRUCTEURS >>>>> \*/  
  
 /\* <<<<< DEBUT GETTERS ET SETTERS >>>>> \*/  
 public* String getNom() {  
 *return* nom;  
 }  
  
 *public void* setNom(String nom) {  
 *this*.nom = nom;  
 }  
  
 *public* String getPrenoms() {  
 *return* prenoms;  
 }  
  
 *public void* setPrenoms(String prenoms) {  
 *this*.prenoms = prenoms;  
 }  
  
 *public* String getSexe() {  
 *return* sexe;  
 }  
  
 *public void* setSexe(){  
 Scanner sc = *new* Scanner(System.*in*);  
 *while*(*true*)  
 {  
 Process.*affichage*("\nSexe:\n 1. Masculin\n 2. Feminin\nChoix: ");  
  
 *int* opt=Integer.*parseInt*(sc.next());  
 *if*(opt==1)  
 {  
 *this*.sexe = "Masculin";  
 *break*;  
 }  
 *else if*(opt==2)  
 {  
 *this*.sexe = "Féminin";  
 *break*;  
 }  
 *else* {  
 Process.*affichage*("\nMauvais choix\nVeuiller recommencer... ");  
 }  
 }  
 }  
  
 *public void* setSexe(String sexe) {  
 *this*.sexe = sexe;  
 }  
  
 *public void* setFreres(LinkedList<Personne> freres) {  
 *this*.freres = freres;  
 }  
  
 *public* UUID getId(){  
 *return this*.id;  
 }  
  
 *public void* setFreres(Personne frere){  
 *if* (*this*.freres.size() == 0){  
 *this*.freres.add(frere);  
 } *else if* (*this*.freres.get(0).age < frere.age) {  
 *this*.freres.add(0, frere);  
 } *else if* (*this*.freres.get(*this*.freres.size() -1).age > frere.age){  
 *this*.freres.add(frere);  
 } *else* {  
 *int* i =0;  
 *while* (*this*.freres.get(i).age > frere.age){  
 i++;  
 }  
 freres.add(i, frere);  
 }  
 }  
  
 *public void* setEnfants(LinkedList<Personne> enfants) {  
 *this*.enfants = enfants;  
 }  
  
 *public void* setEnfants(Personne enfant){  
 *if* (*this*.enfants.size() == 0){  
 *this*.enfants.add(enfant);  
 } *else if* (*this*.enfants.get(0).age < enfant.age) {  
 *this*.enfants.add(0, enfant);  
 } *else if* (*this*.enfants.get(*this*.enfants.size() -1).age > enfant.age){  
 *this*.enfants.add(enfant);  
 } *else* {  
 *int* i =0;  
 *while* (*this*.enfants.get(i).age > enfant.age){  
 i++;  
 }  
 enfants.add(i, enfant);  
 }  
 }  
  
 */\* <<<<< FIN GETTERS ET SETTERS >>>>> \*/  
  
 public* LinkedList<Personne> getAscendant(){  
 *if* (*this*.parent!=*null*){  
 LinkedList<Personne> aux = parent.getAscendant();  
 aux.add(*this*.parent);  
 *return* aux;  
 } *else* {  
 *return new* LinkedList<Personne>();  
 }  
 }  
  
 *public void* addEnfant(Personne enfant){  
 *if* (enfant.equals(*this*))  
 Process.*affichage*("Impossible de s'ajouter soit même comme enfant \n");  
 *else if* (*this*.enfants.contains(enfant)){  
 Process.*affichage*("Impossible d'ajouter 2 fois le même enfant \n");  
 }  
 *else* {  
 *for* (Personne frere : *this*.enfants) {  
 frere.setFreres(enfant);  
 }  
 *this*.setEnfants(enfant);  
 enfant.parent = *this*;  
 }  
 }  
  
 *public* Integer getNbSpace() {  
  
 Personne actu = *this*;  
 *if* (actu.parent!=*null*){  
 *return* actu.parent.getNbSpace() + 2;  
 } *else* {  
 *return* 0;  
 }  
 }  
  
 *public void* afficheEnfant(){  
 *this*.enfants.forEach(enfant-> {  
 Process.*affichage*(enfant + "\n");  
 });  
 }  
  
 *public void* modifierInfo(){  
 *int* repeter = 1;  
 *while*(repeter!=0){  
 Process.*affichage*("Modification de "+*this*.getNom()+"\n");  
 Process.*affichage*("1-Nom\t2-Prenom\t3-Sexe\t4-Nom, prénoms et sexe\t0-retour\n\n");  
 Process.*affichage*("Votre choix: ");  
 String choix = *new* Scanner(System.*in*).next();  
 *switch* (choix){  
 *case* "1":  
 Process.*affichage*("Entrer le nouveau nom: ");  
 *this*.setNom(*new* Scanner(System.*in*).nextLine());  
 Process.*affichage*("Modification effectuée\n");  
 repeter = 0;  
 *break*;  
 *case* "2":  
 Process.*affichage*("Entrer le nouveau prenom: ");  
 *this*.setPrenoms(*new* Scanner(System.*in*).nextLine());  
 Process.*affichage*("Modification effectuée\n");  
 repeter = 0;  
 *break*;  
 *case* "3":  
 *this*.setSexe();  
 repeter = 0;  
 *break*;  
 *case* "4":  
 Process.*affichage*("Entrer le nouveau nom: ");  
 *this*.setNom(*new* Scanner(System.*in*).nextLine());  
 Process.*affichage*("Entrer le nouveau prenom: ");  
 *this*.setPrenoms(*new* Scanner(System.*in*).nextLine());  
 Process.*affichage*("\n");  
 *this*.setSexe();  
 repeter = 0;  
 *break*;  
 *case* "0":  
 repeter=0;  
 *break*;  
 *default*:  
 Process.*affichage*("Mauvaise entrée veuillez recommencer..\n");  
 *break*;  
 }  
 }  
  
 }  
  
 *//REDEFINITION DE LA METHODE toString* @Override  
 *public* String toString() {  
 *return* nom + " " + prenoms + '(' + sexe + ')' ;  
 }  
}

### Arbre généalogique

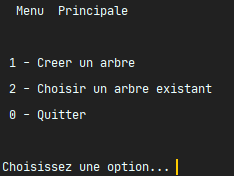
*package* com.projet.java;  
  
*import* java.io.\*;  
*import* java.util.*Iterator*;  
*import* java.util.LinkedList;  
*import* java.util.Scanner;  
  
*public class* ArbreGenealogique *implements Serializable* {  
 *private static* LinkedList<LinkedList<Personne>>*arbres* = *new* LinkedList<>();  
 Personne root;  
  
 *public* ArbreGenealogique(LinkedList<Personne> list\_members){  
 *this*.root = list\_members.get(0);  
 }  
  
 *private* LinkedList<Personne> members = *new* LinkedList<Personne>();  
  
 *public* LinkedList<Personne> getMembers() {  
 *return* getListemembers(root, *new* LinkedList<Personne>());  
 }  
  
 *public* LinkedList<Personne> getListemembers(Personne start, LinkedList<Personne> liste){  
 liste.add(start);  
 start.enfants.forEach(enfant->{  
 getListemembers(enfant, liste);  
 });  
 *return* liste;  
 }  
  
 *public* ArbreGenealogique(Personne first){  
 *this*.root = first;  
 }  
  
 *// METHODE POUR AFFICHER L'ARBRE GENEALOGIQUE  
 public void* afficheArbre(Personne membre, Integer space){  
 Process.*affichage*(Process.*setSpace*(space) + membre + "\n" );  
 membre.enfants.forEach(enfant ->{  
 afficheArbre(enfant, space+2);  
 });  
 }  
  
 *// METHODE POUR RECUPERER UN MEMBRE DE L'ARBRE GENEALOGIQUE  
 public* Personne getMember(){  
 Process.*affichage*("Choisissez le membre :\n");  
 *for* (*int* i = 0; i<*this*.getMembers().size(); i++){  
 Process.*affichage*(i + "- " + *this*.getMembers().get(i) + "\n");  
 }  
 Process.*affichage*("Veuillez choisir une option... ");  
 Integer option = *new* Scanner(System.*in*).nextInt();  
 *return this*.getMembers().get(option);  
 }  
  
 *// METHODE POUR AJOUTER UN MEMBRE A L'ARBRE GENEALOGIQUE  
 public void* addMember(){  
 Process.*affichage*("Choisissez son parent\n");  
 *for* (*int* i = 0; i<*this*.getMembers().size(); i++){  
 Process.*affichage*(i + "- " + *this*.getMembers().get(i) + "\n");  
 }  
 Process.*affichage*(*this*.getMembers().size() + "- Nouveau root \n");  
 Process.*affichage*("Veuillez choisir une option... ");  
 Integer option = *new* Scanner(System.*in*).nextInt();  
 *if* (option < *this*.getMembers().size()){  
 *this*.getMembers().get(option).addEnfant(*new* Personne("New"));  
 } *else* {  
 Personne newRoot = *new* Personne("New");  
 newRoot.addEnfant(*this*.root);  
 *this*.root = newRoot;  
 }  
  
 }  
 *public void* saveTree(){  
 *//On ajoute ici la racine à un grand arbre qui contient tous les arbres crées  
 /\*La classe LinkedListe est une classe sérializable elle va nous permettre de stocker plusieurs objets au lieu  
 d'un seul.  
 \*/  
  
 //On recupère les données pré-sauvegardées  
 getTrees*();  
 *//Pour changer l'affichage  
 boolean* exist= *false*;  
 *//On verifie si l'arbre se trouve dans le fichier data.txt  
 if*(existTree()){  
 exist = *true*;  
 *Iterator*<LinkedList<Personne>> it = *arbres*.iterator();  
 *while*( it.hasNext() ) {  
  
 LinkedList<Personne> pers = it.next();  
 *//On ecrase l'ancienne valeur de l'arbre du root  
 if*( pers.get(0).getId().toString().equals(*this*.root.getId().toString())) {  
 it.remove();  
 }  
  
 }  
 *//On ajoute la nouvelle à la liste des arbres  
 arbres*.add(*this*.getMembers());  
  
 *//On efface le contenu du fichier* PrintWriter pw = *null*; *// >>>> on ajoutera après suppression de ce qui existait éventuellement  
 try* {  
 pw = *new* PrintWriter(*new* BufferedWriter  
 (*new* FileWriter("data.txt", *false*)));  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 pw.print("");  
 pw.close();  
 }*else* {  
 *//On ajoute le root à la liste des arbres  
 arbres*.add(*this*.getMembers());  
 }  
  
 File fichier = *new* File("data.txt");  
 ObjectOutputStream oos = *null*;  
 *try* {  
 *// ouverture d'un flux de sortie sur data.txt* oos = *new* ObjectOutputStream(*new* FileOutputStream(fichier));  
 } *catch* (IOException e) {  
 e.printStackTrace();  
 }  
 *//sérialization de l'objet et sauvegarde dans le fichier data.txt  
 try* {  
 oos.writeObject(*arbres*);  
 oos.close();  
 *if*(exist)  
 Process.*affichage*("Sauvegarde de l'Arbre....ok\n");  
 *else* {  
 Process.*affichage*("Création de l'arbre....ok\n");  
 }  
  
 } *catch* (IOException e) {  
 Process.*affichage*("Erreur dans la sauvegarde de l'arbre\n");  
 e.printStackTrace();  
 }  
  
 }  
  
 *boolean* existTree(){  
 File fichier = *new* File("data.txt") ;  
  
 *try* {  
 *// ouverture d'un flux d'entrée sur data.txt* ObjectInputStream ois = *new* ObjectInputStream(*new* FileInputStream(fichier)) ;  
 *try* {  
 *//On recupère les arbres* LinkedList<LinkedList<Personne>> arbres\_data = (LinkedList<LinkedList<Personne>>) ois.readObject();  
 *for* (LinkedList<Personne> tree:arbres\_data  
 ) {  
 *if* (tree.get(0).getId().equals(*this*.root.getId()))  
 *return true*;  
 }  
  
 } *catch* (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 ois.close();  
  
 } *catch* (IOException e) {  
  
 Process.*affichage*("Aucun arbre enregistré\n");  
 }  
 *return false*;  
 }  
  
 *public static* LinkedList<LinkedList<Personne>> getTrees(){  
 File fichier = *new* File("data.txt");  
 LinkedList<LinkedList<Personne>> arbre = *new* LinkedList<LinkedList<Personne>>();  
 *try* {  
 *// ouverture d'un flux d'entrée sur data.txt* ObjectInputStream ois = *new* ObjectInputStream(*new* FileInputStream(fichier));  
 *try* {  
 *//On recupère les arbres* arbre = (LinkedList<LinkedList<Personne>>) ois.readObject();  
 *arbres* = arbre;  
 }*catch* (Exception e){  
 System.*out*.print("Aucun objet");  
 }  
 }*catch* (Exception e){  
 System.*out*.print("Aucun objet");  
 }  
 *return* arbre;  
 }  
}

### Process

*package* com.projet.java;  
  
*import* java.util.LinkedList;  
*import* java.util.Scanner;  
  
*public class* Process {  
 *private static* Scanner *clavier*;  
  
 ArbreGenealogique arbreGenealogique = *null*;  
 Personne focus = *null*;  
  
 */\* <<<<<<<< DEBUT FONCTIONNALITÉS D'AFFICHAGE ET D'ERGONOMIE >>>>>>>> \*/  
 //METHODE STATIQUE POUR AFFICHER UNE LES DONNÉES D'UNE LISTE  
 private static void* afficherListe(LinkedList<Personne> liste, *int* num, String msg\_error, String mode){  
 *if* ("Suivi".equals(mode)) {  
 *if* (!liste.isEmpty()) {  
 *affichage*(num + " - Resultat:\n\n");  
 liste.forEach(element -> {  
 *if* (element != liste.getLast())  
 *affichage*(element + " -> ");  
 *else* {  
 *affichage*(element + "\n\n");  
 }  
 });  
 *affichage*("Appuyez sur une touche pour quitter.... ");  
 *new* Scanner(System.*in*).nextLine();  
 } *else* {  
 *//Sinon  
 affichage*(num + " - Resultat:\n\n");  
 *affichage*(msg\_error + "\n\n");  
 *affichage*("Appuyez sur une entrer pour quitter.... ");  
 *new* Scanner(System.*in*).nextLine();  
 }  
 } *else* {  
 *if* (!liste.isEmpty()) {  
 *affichage*(num + " - Resultat:\n\n");  
 liste.forEach(element -> {  
 *affichage*("\t" + element + "\n\n");  
 });  
 *affichage*("Appuyez sur une touche pour quitter.... ");  
 *new* Scanner(System.*in*).nextLine();  
 } *else* {  
 *//Sinon  
 affichage*(num + " - Resultat:\n\n");  
 *affichage*(msg\_error + "\n\n");  
 *affichage*("Appuyez sur une entrer pour quitter.... ");  
 *new* Scanner(System.*in*).nextLine();  
 }  
 }  
 *//On vérifie que la liste n'est pas vide* }  
  
 *// METHODE STATIQUE POUR AFFICHER LES MESSAGES  
 public static void* affichage(Object obj){  
 System.*out*.print(obj);  
 }  
  
 *// METHODE STATIQUE POUR AFFICHER DES ESPACES  
 public static* String setSpace(Integer nb){  
 *return new* String(*new char*[nb]).replace("\0", " ");  
 }  
  
 */\* <<<<<<<< FIN FONCTIONNALITÉS D'AFFICHAGE ET D'ERGONOMIE >>>>>>>> \*/  
  
  
 public void* setTree(ArbreGenealogique arb){  
 *this*.arbreGenealogique = arb;  
 *this*.menuGestionFamille();  
 }  
  
 *// METHODE PERMETTANT DE GERE LES ACTIONS SUR LE MENU PRINCIPALE  
 public void* mainMenu(){  
 *int* repeter = 1;  
 *do* {  
 *affichage*(" Menu Principale\n\n");  
 *affichage*(" 1 - Creer un arbre \n");  
 *affichage*(" 2 - Choisir un arbre existant \n");  
 *affichage*(" 0 - Quitter \n\n");  
 *affichage*("Choisissez une option... ");  
 String option = *new* Scanner(System.*in*).next();  
 *switch* (option) {  
 *case* "1":  
 *this*.setTree(*new* ArbreGenealogique(*new* Personne("New")));  
 *break*;  
 *case* "2":  
 *this*.selectTree();  
 *break*;  
 *case* "0":  
 repeter = 0;  
 *break*;  
 *default*:  
 *break*;  
 }  
 }*while* (repeter!=0);  
 }  
  
 *// METHODE PERMETTANT DE GERE LES ACTIONS SUR UNE FAMILLE  
 public void* menuGestionFamille(){  
 *int* repeat = 1;  
 *do* {  
 *affichage*("\*\*\*\* - Arbre Généalogique de la famille "+*this*.arbreGenealogique.root.getNom()+" \*\*\*\*");  
 *affichage*("\n\n 1 - ajouter un membre à la famille \n");  
 *affichage*(" 2 - Lister les membres de la famille\n" );  
 *affichage*(" 3 - Selectionner un membre de la famille\n" );  
 *affichage*(" 0 - Retour \n\n" );  
 *affichage*("Veuillez choisir une option... ");  
  
 String option = *new* Scanner(System.*in*).next();  
 *affichage*("\n");  
 *switch* (option) {  
 *case* "1":  
 *this*.arbreGenealogique.addMember();  
 *break*;  
 *case* "2":  
 *this*.arbreGenealogique.afficheArbre(*this*.arbreGenealogique.root, 0);  
 *break*;  
 *case* "3":  
 *this*.focus = *this*.arbreGenealogique.getMember();  
 *this*.menuGestionPersonne();  
 *break*;  
 *case* "0":  
 *this*.arbreGenealogique.saveTree();  
 repeat = 0;  
 *break*;  
  
 *default*:  
 *break*;  
 }  
 }*while* (repeat!=0);  
 }  
  
 *// METHODE PERMETTANT DE GERE LES ACTIONS SUR UNE PERSONNE  
 public void* menuGestionPersonne(){  
 *int* repeat = 1;  
 *do* {  
 *affichage*("\*\*\*\* - Arbre Généalogique de " + focus.getNom() + " " + focus.getPrenoms() + " \*\*\*\*");  
 *affichage*("\n\n 1 - Modifier les informations" + "\n");  
 *affichage*(" 2 - Lister les enfants" + "\n");  
 *affichage*(" 3 - Lister les freres (ou soeurs)" + "\n");  
 *affichage*(" 4 - Lister les ascendants" + "\n\n");  
 *affichage*(" 5 - Ajouter un enfant \n" );  
 *if*(focus.parent !=*null*)  
 *affichage*(" 6 - Ajouter un ou des freres(ou soeurs) de la personne \n" );  
  
 *if*(focus.parent ==*null*) {  
 *affichage*(" 8 - Ajouter un parent \n");  
 } *else* {  
 *affichage*(" 7 - Afficher le parent \n");  
 }  
 *if*(!*this*.focus.enfants.isEmpty())  
 *affichage*("\n 10 - Afficher l'ainé des enfants\n" );  
  
 *if*(*this*.focus.parent.enfants.size()>1)  
 *affichage*("\n 11 - Afficher l'ainé des frères\n" );  
  
 *affichage*(" 0 - Retour \n\n");  
 *affichage*("Veuillez choisir une option... ");  
 String option = *new* Scanner(System.*in*).next();  
 *affichage*("\n");  
  
 *switch* (option){  
 *case*"1":  
 *this*.focus.modifierInfo();  
 *break*;  
 *case*"2":  
 *//Afficher la liste des enfants  
 afficherListe*(focus.enfants,2,"Aucun enfant enregistré","Suivi");  
 *case*"3":  
 *afficherListe*(focus.freres,3,"Aucun frère enregistré","Normal");  
 *break*;  
 *case*"4":  
 *afficherListe*(focus.getAscendant(),4,"Aucun acsendant enregistré","Normal");  
 *break*;  
 *case*"5":  
 *this*.focus.setEnfants(*new* Personne("New"));  
 *break*;  
 *case*"6":  
 *this*.focus.parent.setEnfants(*new* Personne("New"));  
 *break*;  
 *case*"7":  
 *affichage*("\n " + *this*.focus.parent+ "\n");  
 *break*;  
 *case*"8":  
 Personne parent = *new* Personne("New");  
 parent.setEnfants(*this*.arbreGenealogique.root);  
 *this*.arbreGenealogique.root = parent;  
 *break*;  
 *case*"10":  
 *affichage*("\n " + *this*.focus.enfants.get(0)+ "\n");  
 *break*;  
 *case*"11":  
 *affichage*("\n " + *this*.focus.parent.enfants.get(0)+ "\n");  
 *break*;  
 *case* "0":  
 repeat = 0;  
 *break*;  
 *default*:  
 *break*;  
 }  
 }*while* (repeat!=0);  
 }  
  
 *// METHODE STATIQUE PERMETTANT DE SELECTIONNER UN ARBRE PREALABLEMENT ENREGISTRÉ  
 public void* selectTree(){  
 *int* repeter = 1;  
 *while*(repeter!=0) {  
 *affichage*("Liste des arbres \n");  
 *int* i = 1;  
 *if* (!(ArbreGenealogique.*getTrees*().isEmpty())){  
 LinkedList<LinkedList<Personne>> arbres = ArbreGenealogique.*getTrees*();  
 *for* (LinkedList<Personne> arbre: arbres  
 ) {  
 *affichage*("\t"+i+" -"+arbre.get(0)+"\n");  
 i++;  
 }  
 *affichage*("\nVeuillez selectionner un Arbre (ou 0 pour annuler) ... ");  
 String choix = *new* Scanner(System.*in*).nextLine();  
 *affichage*("\n");  
 *try*{  
 *int* choix1 = Integer.*parseInt*(choix);  
 *if*(choix1 < i && choix1!=0){  
 *this*.arbreGenealogique = *new* ArbreGenealogique(arbres.get(choix1-1));  
 *this*.menuGestionFamille();  
 }*else if*(choix1 == 0)  
 repeter = 0;  
 *else* {  
 *affichage*("Mauvais choix veuillez recommencer !\n");  
 }  
  
 }*catch* (Exception e){  
*// e.printStackTrace();  
 affichage*("Mauvaise saisie veuillez recommencer !\n");  
 }  
  
 }*else* {  
 repeter = 0;  
 *affichage*("Vous n'avez pas encore renseigné d'arbres!\n");  
 }  
 }  
 }  
}

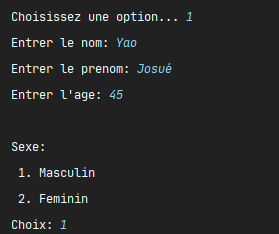
# MANUEL D’UTILISATION DE L’APPLICATION

Après le démarrage, l’application affiche le menu principal :



## Créer un arbre

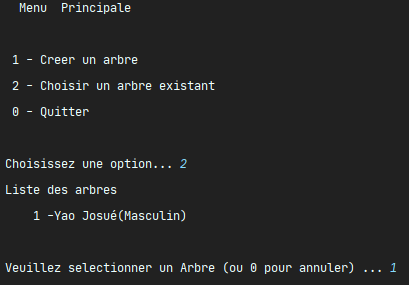
Permet de créer un nouvel arbre en commençant par la création du premier membre de l’arbre :



Ensuite l’application affiche le menu de gestion d’un arbre généalogique.

## Choisir un arbre existant

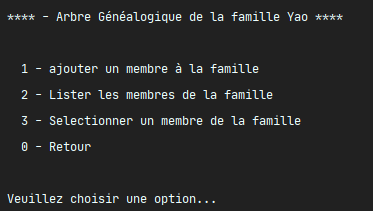
Permet de sélectionner un arbre parmi ceux enregistrés :



Ensuite l’application affiche le menu de gestion d’un arbre généalogique.

## Menu de gestion d’un arbre généalogique

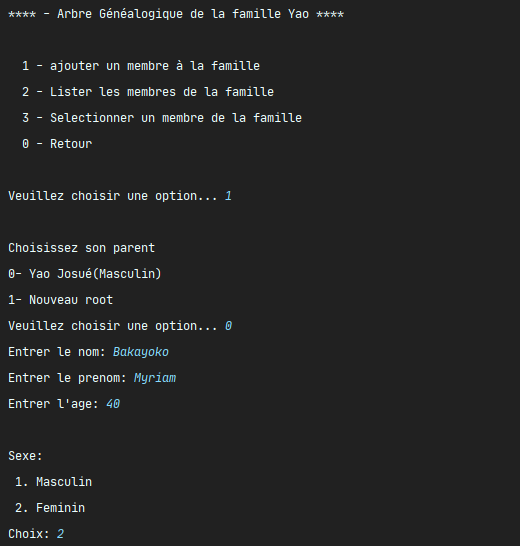
Le menu de gestion d’un arbre généalogique se présente sous la forme suivante :



Il a principalement 3 options :

### Ajouter un membre à la famille

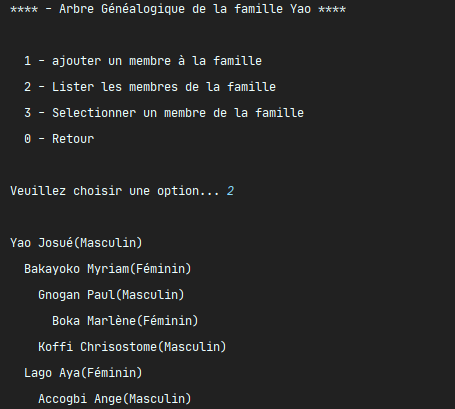
Cette option permet d’ajouter un membre à la famille en sélectionnant son parent :



Si l’utilisateur choisi l’option nouveau root, l’application créer un nouveau utilisateur et le défini comme étant le parent de l’ancien root. Ensuite l’application revient sur le menu de gestion d’un arbre généalogique.

### Lister les membres de l’arbre :

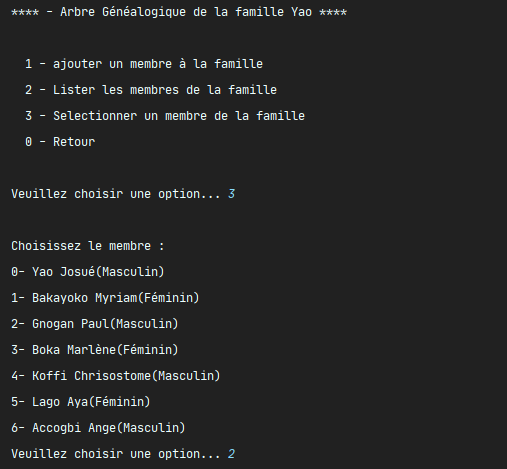
Permet d’afficher tous les membres de la famille :



Puis reviens sur le menu de gestion d’un arbre généalogique.

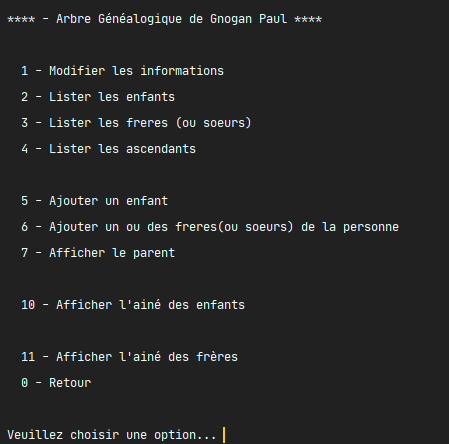
### Sélectionner un membre de la famille :

Permet de choisir un membre de la famille et afficher le menu de gestion d’une personne



## Menu gestion d’une personne

Le menu de gestion d’une personne se présente comme suit :

  
  
Il permet d’interroger l’application sur les frères ascendants ou enfants de la personne sélectionner. Il permet aussi ajouter directement un frère ou un enfant ou de modifier les informations relatives à la personne.