Projet Programmation Orientée Objet

Explications du code

Fait par : ASSY Eliel Onésime
Licence 2B MIAGE

Enseignant : Dr KONATE N'golo

Notes : texte italique (élément du code)

- **Classe Bibliotheque**

```
import java.util.ArrayList;
import java.time.LocalDate;

public class Bibliotheque{
//Constructeur vide
    public Bibliotheque(){}
//Variables
    private String auteur;
    private String titre;
    private String categorie;
    private int nbre_Disposible;
    private String nom_Emprunteur;
    private String anniversaire_Emprunteur;
    private LocalDate date_emprunt;
//Listes ArrayList
    public static ArrayList<Bibliotheque> liste_livre_admin = new ArrayList<>(10000);
    public static ArrayList<Bibliotheque> liste_emprunteur = new ArrayList<>(10000);
//Getters
    public String getAuteur() { return auteur; }
    public String getTitre() { return titre; }
    public String getCategorie() { return categorie; }
    public int getNbre_Disposible() { return nbre_Disposible; }
    public String getNom_Emprunteur() { return nom_Emprunteur; }
    public String getAnniversaire_Emprunteur() { return anniversaire_Emprunteur; }
    public LocalDate getDate_emprunt() { return date_emprunt; }
//Setter
    public void setNbre_Disposible(int nbre_Disposible) { this.nbre_Disposible = nbre_Disposible; }
// Construteurs
    public Bibliotheque(String auteur, String titre, String categorie, int nbre_Disposible) {...}
    public Bibliotheque(String nom_Emprunteur, String anniversaire_Emprunteur, String titre, String auteur,LocalDate date_emprunt) {...}
}
```

- Public Bibliotheque() {}

```
//Constructeur vide
    public Bibliotheque(){}
```

Ce constructeur est utilisé pour utilser pour utiliser des instances de la classe *Bibliotheque* tel que les liste : *liste_livre_admin* et *liste_emprunteur*

- Variables

```
//Variables
    private String auteur;
    private String titre;
    private String categorie;
    private int nbre_Disposible;
    private String nom_Emprunteur;
    private String anniversaire_Emprunteur;
    private LocalDate date_emprunt;
```

C'est variable des informations sur les livres et les utilisateurs

- Liste

```
//Listes ArrayList
    public static ArrayList<Bibliotheque> liste_livre_admin = new ArrayList<>(10000);
    public static ArrayList<Bibliotheque> liste_emprunteur = new ArrayList<>(10000);
```

La liste *Liste_livre_admin* conserve les informations relatives aux livres comme l'auteur, le titre, la catégorie et le nombre disponible.

La liste *liste_emprunteur* conserve les informations sur les emprunteurs cet à dire le nom, la date de naissance, le livre emprunter, la catégorie du livre, l'auteur du livre et la date d'emprunt.

- Getters

Les getters permettent d'obtenir les valeurs de chaque attribut de la classe

- Setter SetNbre_Disponible()

Elle permet de modifier le nombre de livre disponible elle est utilisée au moment d'emprunter un livre et au moment de rendre un livre.

- Constructeurs

Le premier constructeur prend en argument l'auteur, le titre, la catégorie et le nombre disponible

La seconde prend en charge le nom de l'emprunteur, sa date de naissance, le titre de l'œuvre, l'auteur et la date d'emprunt. Ces constructeurs sont utilisés pour crées des Objets utilisé dans les fonctions que nous verrons ci-après

- **Classe Administateur**

```
import java.io.*;

public class Administrateur extends Bibliotheque {

    public Administrateur() {}

    public void afficher_Livres(String nomFichier) {...}

    public void afficher_Livres_NonDisponibles(String nomFichier){...}

    public void afficher_Emprunteur(String nomFichier){...}

    public void ajouter_Livre(Bibliotheque livre){...}

    public void retirer_Livre(String titre){...}

}
```

- Public Administrateur (){}

Ce constructeur est utilisé pour utiliser pour utiliser des méthodes de la classe dans les autres classes

- Public void afficher_livres(String nomFichier)

```java
public void afficher_Livres(String nomFichier) {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(nomFichier));
        String ligne;
        while ((ligne = reader.readLine()) != null) { // Tant que la ligne n'est pas vide
            System.out.println(ligne); //Affiche la ligne
        }
    } catch (IOException e) { //Exception pouvant se produire lors de la lecture du fichier
        e.printStackTrace();//Affiche les détails de l'exception.
    }
}
```

L'argument prit en charge par la méthode *afficher_Livres* est le non du ficher texte. Elle lit le contenu du fichier et affiche son contenue. Le *FileReader* est responsable de lire le fichier spécifié par *nomFichier*, et le *BufferedReader* est utilisé pour lire le contenu de manière plus efficace et ligne par ligne. La boucle while La boucle continue de s'exécuter tant que *readLine()* renvoie une ligne non nulle, a chaque tour elle affiche le contenue de la ligne avec *System.out.println(ligne)*. Si une exception de type IOException se produit lors de la lecture du fichier en cas de problème d'accès au fichier, de fichier introuvable le code captures cette exception et en affiche les détails avec *e.printStackTrace()*

- public afficher_Livres_NonDisponibles

```java
public void afficher_Livres_NonDisponibles(String nomFichier){
    Parametre_liste parametre_liste = new Parametre_liste();
    parametre_liste.enregistrer_LivresNonDispo("ListeLivresAdmin.txt", nomFichier);
    afficher_Fichier(nomFichier);
}
```

L'argument prit en charge par la méthode *afficher_Livres* est le nom du ficher texte. Ensuite elle crée un objet de la classe Parametre_liste *parametre_liste* puis utilise la méthode *enregistrer_LivresNonDispo* de la classe *Parametre_liste* pour remplir le fichier et la méthode *afficher_Fichier* pour afficher le fichier.

- Public ajouter_Livre

```java
public void ajouter_Livre(Bibliotheque livre){
    Parametre_liste parametre_liste = new Parametre_liste();
    liste_livre_admin.add(livre);
    parametre_liste.enregistrer_FichierTexteAdmin(livre);
    parametre_liste.enregistrer_FichierTexteLecteur(livre);
    parametre_liste.enregistrer_LivresNonDispo("ListeLivresAdmin.txt","ListLivreNonDispo.txt");
}
```

La méthode a pour paramètre d'entré un objet de la classe Bibliothèque du nom de livre. Elle crée un objet de la classe Parametre_liste *parametre_liste* puis utilise les méthodes *enregistrer_FichierTexteAdmin, enregistrer_FichierTexteLecteur* et *enregistrer_LivreNonDispo* de la classe Parametre_liste.

- Public void retirer_Livre

```java
public void retirer_Livre(String titre){
    File fichier = new File("ListeLivresAdmin.txt");
    File fichierTemporaire = new File("FichierTestTemporaire.txt");
    try {
        BufferedReader reader = new BufferedReader(new FileReader(fichier));
        BufferedWriter writer = new BufferedWriter(new FileWriter(fichierTemporaire));
        String line;
        boolean supprime = false;
        while ((line = reader.readLine()) != null) {
            if (line.contains("Titre : " + titre)) {
                supprime = true;
                for (int i = 0; i < 4; i++) {
                    reader.readLine();
                }
                continue;
            }
            writer.write(line);
            writer.newLine();
        }

        if (supprime) {
            writer.close();
            reader.close();
            if (fichier.delete()) {
                if (!fichierTemporaire.renameTo(fichier)) {
                    System.out.println("Impossible de renommer le fichier temporaire.");
                }
            } else {
                System.out.println("Impossible de supprimer le fichier original.");
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

La méthode prend en paramètre d'entré le titre du livre à supprimer. Ensuite elle crée deux Objets *fichier* et *fichierTemporaire* puis lis ligne par ligne les deux fichiers. Si elle trouve une ligne contenant le titre d'un livre à retirer, elle supprime les informations du livre. Elle écrit le reste du fichier *fichier* dans le fichier temporaire avec la condition if(supprime). Le catch gère les exceptions dans le cas où l'un des fichiers n'existe pas.

- **Parametre_liste**

```java
import java.io.*;

public class Parametre_liste extends Bibliotheque{

    public Parametre_liste() {}

    public void enregistrer_FichierTexteAdmin(Bibliotheque livre ) {...}

    public void enregistrer_FichierTexteLecteur(Bibliotheque livre) {...}

    public static void enregistrer_FichierTexteEmprunteur(Bibliotheque livre) {...}

    public void enregistrer_LivresNonDispo(String fichierOrigine, String fichierDestination) {...}
}
```

- Public Parametre_liste()

Ce constructeur est utilisé pour utiliser pour utiliser des méthodes de la classe dans les autres classes

- public void enregistrer_FichierTexteAdmin(Bibliotheque livre)

```
public void enregistrer_FichierTexteAdmin(Bibliotheque livre ) {
    String ListeLivres = "ListeLivresAdmin.txt";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(ListeLivres, true))) {
        writer.write("Titre : " + livre.getTitre());
        writer.newLine();
        writer.write("Auteur : " + livre.getAuteur());
        writer.newLine();
        writer.write("Catégorie : " + livre.getCategorie());
        writer.newLine();
        writer.write("Nombre disponible : " + livre.getNbre_Disposible());
        writer.newLine();
        writer.newLine();//Séparation
        System.out.println("Données enregistrées dans " + ListeLivres);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

La méthode *enregistrer_FichierTexteAdmin* prend un objet *Bibliotheque* (représentant un livre) en entrée et enregistre ses informations dans un fichier texte appelé *ListeLivresAdmin.txt* . Si le fichier n'existe pas, il est créé. Elle gère également les exceptions liées à la manipulation de fichiers, notamment les erreurs. Une fois les données enregistrées, elle affiche un message de confirmation.

- Public void enregistrer_FichierTexteLecteur

```
public void enregistrer_FichierTexteLecteur(Bibliotheque livre) {
    String ListeLivres = "ListeLivresLecteur.txt";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(ListeLivres, true))) {
        writer.write("Titre : " + livre.getTitre());
        writer.newLine();
        writer.write("Auteur : " + livre.getAuteur());
        writer.newLine();
        writer.write("Catégorie : " + livre.getCategorie());
        writer.newLine();
        writer.newLine();//Séparation
        System.out.println("Données enregistrées dans " + ListeLivres);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

La méthode *enregistrer_FichierTexteLecteur* prend un objet *Bibliotheque* (représentant un livre) en entrée et enregistre ses informations dans un fichier texte appelé *ListeLivresLecteur.txt* . Si le fichier n'existe pas, il est créé. Elle gère également les exceptions liées à la manipulation de fichiers. Une fois les données enregistrées, elle affiche un message de confirmation.

- public static void enregistrer_FichierTexteEmprunteur(Bibliotheque livre)

```
public static void enregistrer_FichierTexteEmprunteur(Bibliotheque livre) {
    String ListeLivres = "ListeLivresEmprunteur.txt";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(ListeLivres, true))) {
        writer.write("Nom de l'emprunteur : " + livre.getNom_Emprunteur());
        writer.newLine();
        writer.write("Date de naissance : " + livre.getAnniversaire_Emprunteur());
        writer.newLine();
        writer.write("Titre : " + livre.getTitre());
        writer.newLine();
        writer.write("Auteur : " + livre.getAuteur());
        writer.newLine();
        writer.write("Date d'emprunt : " + livre.getDate_emprunt());
        writer.newLine();
        writer.newLine();//Séparation
        System.out.println("Données enregistrées dans " + ListeLivres);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

La méthode *enregistrer_FichierTexteEmprunteur* prend un objet *Bibliotheque* (représentant un livre emprunté) en entrée et enregistre ses informations dans un fichier texte appelé *ListeLivresEmprunteur.txt* . Si le fichier n'existe pas, il est créé. Elle gère également les exceptions liées à la manipulation de fichiers. Une fois les données enregistrées, elle affiche un message de confirmation.

- Pu

```
public void enregistrer_LivresNonDispo(String fichierOrigine, String fichierDestination) {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(fichierOrigine));
        BufferedWriter writer = new BufferedWriter(new FileWriter(fichierDestination));
        String ligne;
        boolean livre_Disponible = true;
        while ((ligne = reader.readLine()) != null) {
            if (ligne.contains("Nombre disponible : 0")) {
                livre_Disponible = false;
                continue;
            }
            if (!livre_Disponible) {
                writer.write(ligne);
                writer.newLine();
            }
            if (ligne.startsWith("Catégorie : ")) {
                livre_Disponible = true; // Réinitialise le drapeau
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

En résumé, cette méthode parcourt un fichier source ligne par ligne, identifie les livres non disponibles en cherchant *"Nombre disponible : 0",* et copie les informations de ces livres dans un fichier de destination. Elle utilise un indicateur *(livre_Disponible)* pour suivre si un livre est disponible ou non. Les flux de lecture et d'écriture sont correctement fermés, et les exceptions sont gérées pour éviter les erreurs liées aux opérations de lecture/écriture de fichiers.

- **Classe Principale**

```java
import java.util.Scanner;

public class Principale {
//Variable de connexion
    public static String user;
    public static String login;
//Methode principale
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("\t\tBienvenue dans la Bibliotheque\n\t\t\tVeillez vous connecter\n");
        do {
            System.out.print("Entrez votre nom : ");
            user = scanner.next();
            if (Gestion_Exeception.nom_impossible(user)){
                System.out.println("Le nom ne doit pas contenir des chiffres ou des caractères speciaux");
            }
        }while (Gestion_Exeception.nom_impossible(user));
        System.out.print("Entrez votre Date de naissance au format ");
        System.err.print("AAAA-MM-JJ : ");
        login = scanner.next();

        if ((login.equals("2005")) && (user.equals("admin") || user.equals("Admin"))){
            Fonction_Principale.page_administrateur();
        }else {
            System.out.print("Une erreur c'est produite\nEntrez a nouveau votre Date de naissance au format AAAA-MM-JJ : ");
            Gestion_Exeception.date_impossible(scanner);
            Fonction_Principale.page_utilisateur();
        }
    }

}
```

Il s'agit de la classe Principale au démarrage du programme elle demande d'entrer le nom et la date de naissance. Si nom = « admin » et date de naissance = « 2005 » elle ouvre la méthode *page_administrateur* de la classe fonction principale Sinon la méthode *page_utili* de la classe *Fonction_Principale*.

- **Classe Fonction_Principale**

```java
import java.util.Scanner;

public class Fonction_Principale {

    public static void page_utilisateur(){
        System.out.println("\n\n\t\t\tBienvenue "+ Principale.user+"\n");
        Fonctions_page_utili.fonction_Utilisateur();
    }

    public static void  page_administrateur(){
        Scanner scanner = new Scanner(System.in);
        String mdp;
        System.out.println("\n\n\t\tVous etes sur la page administrateur\n");
        System.out.print("Entrez le mot de passe pour avoir acces aux fonction\nATTENTION VOUS AVEZ SEULEMENT DROIT A 3 ESSAIS\n\nEntrez le code : ");
        for (int i = 1; i <= 3; i++){
            mdp = scanner.next();
            if(mdp.equals("bibli23")){
                System.out.println("\nLe mot de passe est CORRECT");
                Fonctions_page_admin.fonction_Administrateur();
                break;
            }else {
                System.err.print("Le mot de passe est INCORRECT");
                if (i < 3){
                    System.out.print(" Il vous reste "+(3-i)+"/3 essais\nEntrez le code : ");
                }
            }
        }
    }

}
```

- Public static void page_utilisatateur()

Cette méthode récupère la variable user de la classe principale et dirige à la méthode *fonction_Utilisateur()* de la classe *Fonctions_page_utili*

- Public static void page_administrateur()

Cette méthode est une mesure de sécurité pour connecter l'administrateur. Si la mesure de sécurité est passé la méthode nous conduis à la méthode *fonction_Administrateur()* de la classe *Fonctions_page_admin*

- **Classe Fonctions_page_admin**

```java
import java.util.Scanner;
import java.util.InputMismatchException;


public class Fonctions_page_admin {

    public static void fonction_Administrateur(){
        boolean boucle = true;
        while (boucle){
            int choix;
            Scanner scanner = new Scanner(System.in);
            Administrateur administrateur = new Administrateur();
            System.out.println("""
                                Vous etes sur la page administrateur

                                        Fonction disponible :

                    1 -> Afficher la liste des livres        2 -> Afficher la liste des livres non disponible
                    3 -> Afficher la liste des emprunteurs   4 -> Ajouter un livre
                    5 -> Retirer un livre
                    0 -> Arreter le programme
                    """);
            System.out.print("Entrer le chiffre de ce que vous voulez pour effectuer une action");
            try {
                System.out.print("\nchoix : ");
                choix = scanner.nextInt();
                switch (choix) {
                    case 0 -> boucle = false;
                    case 1 -> {
                        System.out.println("\n\nVoici la liste des livres\n");
                        administrateur.afficher_Fichier("ListeLivresAdmin.txt");
                    }
                    case 2 -> {
                        System.out.println("\n\nVoici la liste des livres non disponible\n");
                        administrateur.afficher_Livres_NonDisponibles("ListLivreNonDispo.txt");
                    }
                    case 3 -> {
                        System.out.println("\n\nVoici la liste des emprunteurs\n");
                        administrateur.afficher_Fichier("ListeLivresEmprunteur.txt");
                    }
                    case 4 -> {
                        String titre;
                        String auteur;
                        String categorie;
                        int nbre_disponible;
                        System.out.println("\n\nVous allez ajoujer un livre\n");
                        System.out.print("Titre : ");
                        titre = scanner.next();
                        System.out.print("Auteur : ");
                        auteur = scanner.next();
                        System.out.print("Categorie : ");
                        categorie = scanner.next();
                        System.out.print("Nombre disponible : ");
                        nbre_disponible = scanner.nextInt();
                        Bibliotheque nouveau_livre = new Bibliotheque(auteur, titre, categorie, nbre_disponible);
                        administrateur.ajouter_Livre(nouveau_livre);
                        System.out.println("Le livre a ete correctement ajoute");
```

```
                }
                case 5 -> {
                    String titre;
                    System.out.println("Veillez le titre du livre a supprimer");
                    System.out.print("Titre : ");
                    titre = scanner.next();
                    administrateur.retirer_Livre(titre);
                    System.out.println("Le livre "+titre+" a ete supprime");
                }
                default -> System.err.println("\nVeillez entre un des nombres");
            }
        }catch (InputMismatchException e){
            System.out.println("Veillez entre un entier valide");
        }

    }

}
}
```

Il s'agit de l'interface de l'administrateur (Voir guide d'utilisation)

- **Classe Fonctions_page_utili**

```
import java.time.LocalDate;
import java.util.Scanner;
import java.util.InputMismatchException;

public class Fonctions_page_utili {

    public static void fonction_Utilisateur(){
        boolean boucle = true;
        while (boucle){
            int choix;
            Scanner scanner = new Scanner(System.in);
            Lecteurs lecteurs = new Lecteurs();
            System.out.println("""
                                Vous etes sur la page utilisateur

                                     Fonction disponible :

                    1 -> Afficher la liste des livres disponible  2 -> Rechercher un livre
                    3 -> Emprunter un livre                       4 -> Rendre un livre
                    0 -> Arreter le programme
                    """);
            System.out.print("Entrer le chiffre de ce que vous voulez pour effectuer une action");
            try {
                System.out.print("\nchoix : ");
                choix = scanner.nextInt();
                switch (choix) {
                    case 0 -> boucle = false;
                    case 1 -> {
                        System.out.println("\n\nVoici la liste des livres\n");
                        lecteurs.afficher_Livres("ListeLivresLecteur.txt");
```

```
                }
                case 2 -> {
                    String recherche;
                    System.out.println("Vous pouvez rechercher le un livre veillez entrer le titre");
                    System.out.print("Titre : ");
                    recherche = scanner.next();
                    Lecteurs.recherche_livre("ListeLivresLecteur.txt",recherche);
                }
                case 3 -> {
                    String titre;
                    String auteur;
                    LocalDate date_aujourdhu = LocalDate.now();
                    System.out.println("Entrez les information du livre a emprunter : ");
                    System.out.print("Titre : ");
                    titre = scanner.next();
                    System.out.print("Auteur : ");
                    auteur = scanner.next();
                    Bibliotheque emprunteur = new Bibliotheque(Principale.user,Principale.login,titre,auteur,date_aujourdhu);
                    if (Gestion_Exeception.verification("ListeLivresLecteur.txt",titre) && Gestion_Exeception.verification("ListeLivresLecteur.txt",titre)){
                        Lecteurs.ajouter_Emprunteur(emprunteur);
                    }
                }
                case 4 -> {
                    String titre;
                    System.out.println("Entrez le titre du livre a rendre : ");
                    System.out.print("Titre : ");
                    titre = scanner.next();
                    Lecteurs.rendre_livre(Lecteurs.recherche_livre(Bibliotheque.liste_emprunteur));
                    System.out.println("Vous avez rendu "+titre);
                }

                default -> System.err.println("\nVeillez entre un des nombres");
            }
        }catch (InputMismatchException e){
            System.out.println("Veillez entre un entier valide");
        }
    }
}
}
```

Il s'agit de l'interface de l'utilisateur (Voir guide d'utilisation)

- **Classe Gestion_Execption**

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeParseException;
import java.util.Scanner;


public class Gestion_Exeception {
//S'assure que la date est entré au format Année-Mois-Jour
    public static LocalDate format_Date(Scanner scanner) {...}
//S'assure que la date d'anniversaire entré est antérieur à la date du jour
    public static void date_impossible(Scanner scanner){...}
//S'assure que le non de l'utilisateur entré ne contient pas de caractères spéciaux
    public static boolean nom_impossible(String nom) { return !nom.matches("^[a-zA-Z]+$"); }
//S'assure qu'une recherche est dans un fichier text
    public  static boolean verification(String nom_Fichier, String donne_verifier){...}
}
```

- public static LocalDate format_Date(Scanner scanner)

```java
public static LocalDate format_Date(Scanner scanner) {
    LocalDate date = null;
    boolean valide = false;
    int i = 0;
    while (!valide) {
        try {
            if (i > 0){
                System.out.print("Entrez votre Date de naissance au format AAAA-MM-JJ : ");
            }
            i++;
            String date_entrer = scanner.next();
            date = LocalDate.parse(date_entrer);
            valide = true;
        } catch (DateTimeParseException e) {
            System.out.println("Format de date incorrect");
        }
    }
    return date;
}
```

Cette méthode prend la date entrée et vérifie si elle respecte le format avec *date = LocalDate.parse(date_entrer)* si la date est valide la boucle s'arrête sinon elle affiche un message d'erreur jusqu'à ce que la date soit correcte.

- public static void date_impossible(Scanner scanner)

```java
public static void date_impossible(Scanner scanner){
    LocalDate.now();
    LocalDate date_aujourdhui;
    do {
        LocalDate date_entree = format_Date(scanner);
        date_aujourdhui = LocalDate.now();
        if (date_entree.isAfter(date_aujourdhui)){
            System.out.println("Vous avez entree un date plus avance que la date d'aujourd'hui");
            System.out.print("Entrez une date possible au forma AAAA-MM-JJ : ");
        }else {
            break;
        }
    }while (true);
}
```

Cette méthode vérifie si la date entrer est antérieur à la date d'aujourd'hui. Elle récupère la date du jour avec *date_aujourdhui = LocalDate.now()* Si ce n'est pas le cas elle demande d'entrer à nouveau la date.

- public static boolean nom_impossible(String nom)

```java
public static boolean nom_impossible(String nom){
    return !nom.matches("^[a-zA-Z]+$");
}
```

Cette méthode s'assure que le non de l'utilisateur entré ne contient pas de caractères spéciaux

- public static boolean verification(String nom_Fichier, String donne_verifier)

```java
public  static boolean verification(String nom_Fichier, String donne_verifier){
    try {
        BufferedReader reader = new BufferedReader(new FileReader(nom_Fichier));
        String ligne;
        while ((ligne = reader.readLine()) != null) {
            if (ligne.contains(donne_verifier)) {
                reader.close();
                return true;
            }
        }
        reader.close(); //ferme le fichier si aucune correspondance n'a été trouvée.
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}
```

La méthode ouvre un fichier spécifié, lit son contenu ligne par ligne, et vérifie si la chaîne de caractères *donne_verifier* est présente dans l'une des lignes. Si une correspondance est trouvée, elle retourne *true*, sinon, elle retourne false. Le fichier est correctement fermé dans les deux cas, et les exceptions liées à la lecture de fichiers sont gérées pour éviter les erreurs.

Exemples des différentes erreurs possible.

- Caractère spéciale dans le nom

```
Entrez votre nom : Eliel12
Le nom ne doit pas contenir des chiffres ou des caractères speciaux F
```

- Format de la date incorrecte

```
Entrez a nouveau votre Date de naissance au format AAAA-MM-JJ : 12-06-2005
Format de date incorrect
Entrez votre Date de naissance au format AAAA-MM-JJ :
```

- Date de naissance impossible

```
Entrez votre Date de naissance au format AAAA-MM-JJ : 2055-05-01
Vous avez entree un date plus avance que la date d'aujourd'hui
Entrez une date possible au forma AAAA-MM-JJ :
```

- Entrer un autre chose que l’entier demandé

```
            Bienvenue Eliel


                    Vous etes sur la page utilisateur

                            Fonction disponible :

1 -> Afficher la liste des livres disponible  2 -> Rechercher un livre
3 -> Emprunter un livre                       4 -> Rendre un livre
0 -> Arreter le programme

Entrer le chiffre de ce que vous voulez pour effectuer une action
choix : a
Veillez entre un entier valide
```