

Le 28 juin 2020 par Michel Mendi
Formation Technicien développeur du CNAM

Logiciel de gestion des patients



Sommaire

- I – Introduction
- II – Structure
- III – Détail de l'application
- IV – Installation
- V – Ressources externes
- VI – Dysfonctionnement
- VII – Amélioration
- VIII – Vécu du projet

I - Introduction

L'application M.A.S.H a pour objectif la prise de rendez-vous médical pour les patients. Celle-ci a pour fonction :

- Ajout, suppression, modification des informations personnelles des patients.
- Prise de rendez-vous médical: Ajout, suppression, modification des rendez-vous pour chaque patient.

The screenshot shows the M.A.S.H application interface. It is divided into two main sections: 'Ajouter un patient' and 'Ajouter examens'.

Ajouter un patient: This section contains form fields for 'Nom', 'Prénom', 'NSS', 'Date de Naissance', 'Date d'entrée', 'Date de sortie', and 'N° de téléphone'. There are also radio buttons for 'Masculin' and 'Feminin', and buttons for 'Ajouter', 'Modifier', 'Annuler', and 'Supprimer'. A blue arrow points from the 'Informations personnelles des patients' label to the 'Nom' field.

Ajouter examens: This section contains a 'Patient' dropdown, a 'Type d'examen' dropdown (currently set to 'Neurologie'), and a 'Date d'examen' field. There are buttons for 'Ajouter', 'Modifier', and 'Supprimer'. A blue arrow points from the 'Saisie des rendez-vous d'examens' label to the 'Date d'examen' field.

Liste des patients: A table on the right side of the 'Ajouter un patient' section displays a list of patients. A blue arrow points from the 'Liste des patients' label to this table.

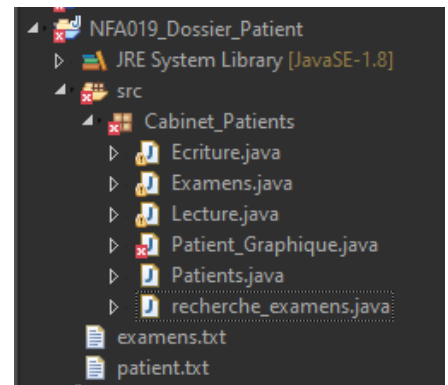
Liste des rendez-vous d'examens: A table on the right side of the 'Ajouter examens' section displays a list of appointments. A blue arrow points from the 'Liste des rendez-vous d'examens' label to this table.

Annotations: Four blue boxes with white text and arrows point to specific parts of the interface: 'Informations personnelles des patients' points to the 'Nom' field; 'Liste des patients' points to the patient list table; 'Saisie des rendez-vous d'examens' points to the 'Date d'examen' field; and 'Liste des rendez-vous d'examens' points to the appointment list table.

II -STRUCTURE

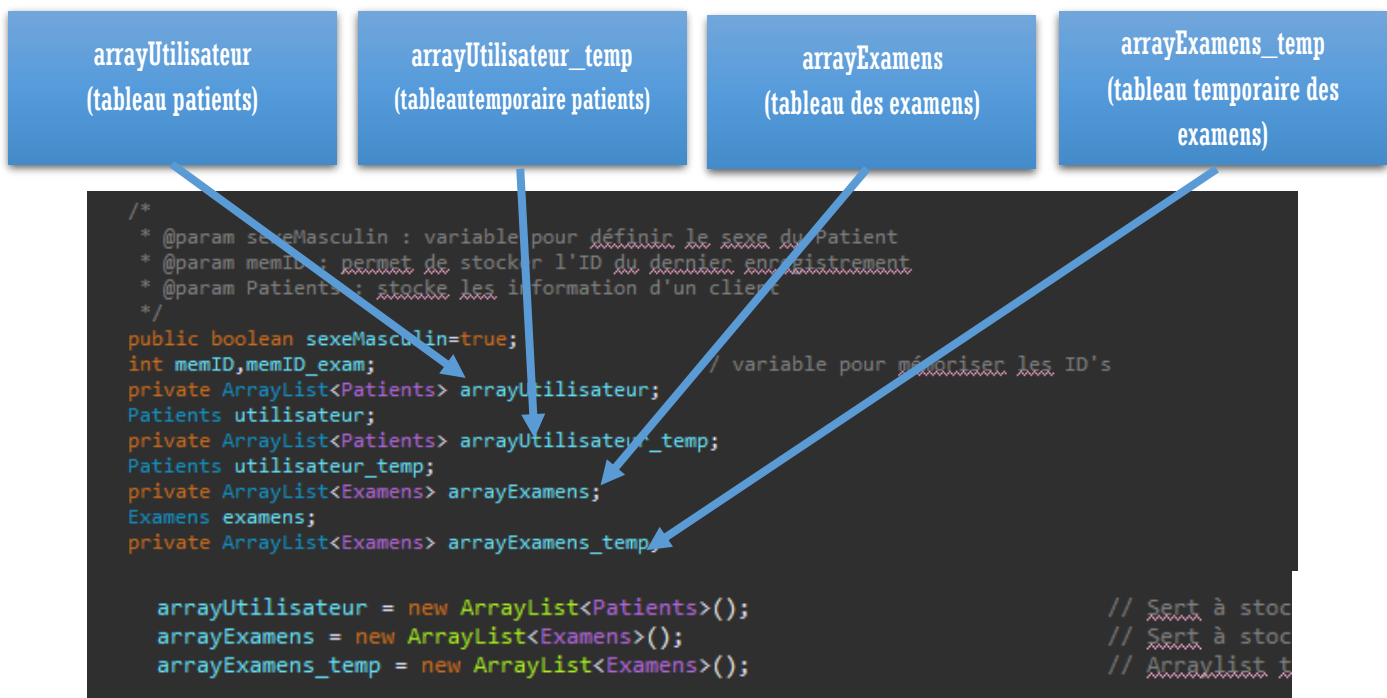
La structure du programme est décomposée de la façon suivante :

- **Patient_Graphique.java** : Programme principal.
- **Patients.java** : Constructeur qui stocke les informations personnelles des patients.
- **Recherche_examens.java** : Méthode qui recherche les examens correspondants à un patient sélectionné.
- **Lecture.java** : Méthode qui lit un fichier passé en paramètre. Ici, soit le fichier « patient.txt » qui contient la liste des patients, soit « examens.txt » qui contient les examens.
- **Ecriture.java** : Méthode qui écrit les informations correspondantes aux patients et aux examens.
- **Examens.txt** : Fichier texte qui contient les informations liées aux examens.
- **Patient.txt** : Fichier qui contient les informations liées aux patients.



III - DETAILS DE L'APPLICATION

Patient_Graphique.java



arrayUtilisateur_temp servira à manipuler les informations des patients afin de pouvoir afficher correctement celles-ci dans la zone d'affichage (JList). Une fois les informations manipulées

(modifications, suppressions), celle-ci seront stockées dans arrayUtilisateur afin de pouvoir les écrire dans le fichier texte pour les sauvegarder.

arrayExamens_temp aura la même fonction que ci-dessus (arrayUtilisateur_temp), mais applicable aux examens.

1 - Initialisation des éléments graphique (Swing)

```
/**
 * Initialise le contenu des fenêtres
 */
private void initialize() {

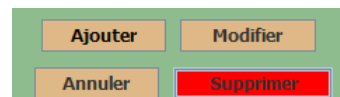
    arrayUtilisateur = new ArrayList<Patients>(); // Set à stocker les infos des Patients
    arrayExamens = new ArrayList<Examens>(); // Set à stocker les infos des examens
    arrayExamens_temp = new ArrayList<Examens>(); // ArrayList temporaire pour stocker les examens correspondant à un patient sélectionné

    frmMash = new JFrame();
    frmMash.setForeground(new Color(143, 188, 143));
    frmMash.setTitle("M.A.S.H");
    frmMash.setBackground(new Color(143, 188, 143));
    frmMash.getContentPane().setBackground(new Color(143, 188, 143));

    JPanel panelPatient = new JPanel(); // zone d'affichage pour les patients
    panelPatient.setBackground(new Color(143, 188, 143));
    JLabel labelNom = new JLabel("Nom"); // Label Nom
    JLabel labelPrenom = new JLabel("Pr\u00E9nom"); // Label prénom
    JLabel labelNSS = new JLabel("NSS"); // Label Numéro de sécurité social
    JLabel labelDateNaiss = new JLabel("Date de Naissance"); // Label Date de naissance
}
```

Cette partie permet de définir tous les éléments graphiques de l'interface utilisateur (zones d'affichages, boutons, champs de saisies).

On y définit aussi le comportement des boutons radios ainsi que leur animations (changement de couleur au passage de la souris).



```
JRadioButton btnMasculin = new JRadioButton("Masculin");
btnMasculin.setBackground(new Color(143, 188, 143));
JRadioButton btnFeminin = new JRadioButton("Feminin");
btnFeminin.setBackground(new Color(143, 188, 143));
```

```
btnSupprimerPatient.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        btnSupprimerPatient.setBackground(Color.RED);
    }
    @Override
    public void mouseExited(MouseEvent e) {
        btnSupprimerPatient.setBackground(new Color(222, 184, 135));
    }
});
```

L'affichage des listes des patients et des examens se font par les Jlist (fenêtre d'affichage) :

- listPatients qui permet d'afficher la liste de tous les patients.
- panelExamens qui permet d'afficher les examens programmés correspondant aux patients sélectionnés.

Jlist -> panelExam
Affichage des informations
concernant les examens

Jlist -> listPatients
Affichage des infos des
patients

The screenshot shows the 'Application patient / examens' window. On the left, there's a form to 'Ajouter un patient' with fields for Nom, Prénom, NSS, Date de Naissance, Date d'entrée, Date de sortie, and N° de téléphone. Below these are radio buttons for 'Masculin' and 'Feminin', and buttons for 'Ajouter', 'Modifier', 'Annuler', and 'Supprimer'. A message 'Le champs Nom est vide' is displayed. In the center, there's a group photo of the M.A.S.H. team. On the right, a list of patients is displayed with columns for patient ID, name, gender, NSS, birth date, entry date, and exit date. Below the patient list, there's a section for 'Ajouter examens' with a dropdown for 'Type d'examen' (currently set to 'Neurologie') and a date field for 'Date d'examen'. Buttons for 'Ajouter', 'Modifier', and 'Supprimer' are also present. A list of exams is displayed on the right side of this section.

2 - Partie gestion de saisies des informations

La saisie nécessite de vérifier que les informations soient conformes.

Pour cela un certain nombre de contrôles sont effectués tels que :

- Ne pas avoir de champs vide.
- Respecter un format, comme par exemple, saisir la date en format jj/mm/aaaa.
- Cohérence de l'information, par exemple, une date de naissance ne doit pas être supérieure à la date et heure du jour (on ne peut pas enregistrer quelqu'un qui n'est pas né).

Dans notre programme les vérifications sont :

- Nom et prénom : pas de chiffre, ni de caractères spéciaux tel que #,~, \$ etc..
- NSS : uniquement des chiffres. (Par manque de temps je n'ai pas vérifié le nombre de chiffres)
- Date de naissance : vérification du format qui doit être jj/mm/aaaa.
Vérification de la date qui ne doit pas être supérieure à maintenant (jour , heure).
- Date d'entrée : vérification du format qui doit être jj/mm/aaaa.
Vérification de la date qui ne doit pas être inférieure à maintenant (jour , heure) car on ne peut pas avant la date du jour.
- Date de sortie : vérification du format qui doit être jj/mm/aaaa.
Vérification de la date qui ne doit pas être inférieure à la date d'entrée (jour , heure) car on ne peut sortir avant d'être arrivé.

- Date d'examens : vérification du format qui doit être jj/mm/aaaa.
Vérification de la date qui ne doit pas être inférieure à maintenant (jour , heure) car on ne peut pas prendre un rendez-vous pour hier.

```

textFieldDateNaissance.addFocusListener(new FocusAdapter() {
    @Override
    public void focusLost(FocusEvent e) {
        saisieOk=true;
        textFieldMessErreur.setVisible(false);
        if (textFieldDateNaissance.getText().equals("")) {
            textFieldMessErreur.setVisible(true);
            textFieldMessErreur.setText("Le champs date de naissance est vide");
            saisieOk=false;
        }
        else {
            textFieldMessErreur.setVisible(false);
            String verif = textFieldDateNaissance.getText();
            if(!verif.matches("^([0-2][0-9]|3[0-1])/([0-1-9]|1[0-2])/[0-9]{4}$")){
                textFieldMessErreur.setVisible(true);
                textFieldMessErreur.setText("Le champs date de naissance n'est pas au bon format");
                saisieOk=false;
            }
            else {
                // Date saisie par l'utilisateur
                SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
                Date dateFormat = null;

                try {
                    dateFormat = sdf.parse(verif);
                } catch (ParseException pe) {
                    pe.printStackTrace();
                }

                Date dateJour = new Date();

                if (dateFormat.after(dateJour)) {

                    textFieldMessErreur.setVisible(true);
                    textFieldMessErreur.setText("La date doit être inférieure ou égale à la date du jour");
                    saisieOk=false;
                }
            }
        }
    }
});

```

Champs vide
interdit

Doit respecter le
format jj/mm/aaaa

Vérifie si la date
de naissance est
bien inférieure à
maintenant

Exemple de code pour la validation de la saisie d'un patient.

```

btnValiderExamen.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        if (saisieOk) {
            textFieldMessErreur.setVisible(false);

            if (textFieldNom.getText().equals("") || textFieldPatientExam.getText().equals("")) {
                // S'il n'y a aucun nom on sort de la méthode
                textFieldMessErreur.setVisible(true);
                textFieldMessErreur.setText("ATTENTION: Il n'y a pas de nom de sélectionné, veuillez
                choisir un Patient.");
                return;
            }
            arrayExamens.clear();
            Lecture.LitFichierExamens(arrayExamens,"examens.txt");

            /*
             * récupération des informations du formulaire dans 'utilisateur'
             * puis ajouter dans la Jlist "listPatients"
             */

            examens.setId_Patient_Exam(memID+1);
            //examens.memID;
            examens.setId_Patient((examens.getId_Patient()));
            examens.setExamIsDel(1);

            Object type= comboBoxExamen.getSelectedItem();
            examens.setTypeExamens(type);
            examens.setDateExamens(getTextFieldDateExam().getText());
            arrayExamens.add(examens);
            arrayExamens_temp.add(examens);
            panelExam.setListData(arrayExamens_temp.toArray());

            memID =arrayExamens.get(arrayExamens.size() - 1).getId_Patient_Exam();

            /*
             * Enregistrement des données de l'objet 'utilisateur'
             */

            try {
                File f=new File("examens.txt");
                FileWriter fw = new FileWriter(f,false);
                BufferedWriter buffer = new BufferedWriter(fw);

                for(Examens elem: arrayExamens) {
                    buffer.write(elem.toString());
                    buffer.newLine();
                    buffer.flush();
                }
                buffer.close();
            }catch (IOException ex) {
                System.out.println("ERREUR: "+ex);
            }
            else {
                textFieldMessErreur.setVisible(true);
                textFieldMessErreur.setText("Examen non enregistré. Il y a une erreur");
                saisieOk=false;
            }
        }
    }
});

```

Champs vide interdit

Recupère la liste de tous les examens

- Ajoute +1 à l'ID.
- Recupère les informations saisies dans le formulaire des patients.
- Ajoute l'ensemble de l'ArrayList temporaire.
- Recupère l'ID

Examen non enregistré. Il y a une erreur

Enregistrement de l'ArrayList Examens dans le fichier « examens.txt ». On affiche un message d'erreur au cas où l'enregistrement se passe mal

Patients.java

Constructeur qui stocke les informations personnelles relatives aux patients.

```
public Patients(){
    identifiant=0;
    nom = "";
    prenom = "";
    NSS="";
    dateEntree = " ";
    dateSortie = " ";
    telephone = " ";
    TextSexe="";
    patientIsDel=1;
}
```

Recherche_examens.java

Méthode qui affiche les examens soit en totalité, soit ceux correspondant à un patient sélectionné. Elle gère aussi la suppression des examens en mettant un '0' plutôt que de le supprimer définitivement, ce qui pourrait permettre de le réactiver ultérieurement.

```
public static <arrayExamens> ArrayList<Examens> recherche(ArrayList<Examens> arrayExamens, Object ID_Local, int supprime){
    Examens examens_temp;
    ArrayList<Examens> arrayExamens_temp=new ArrayList<Examens>();

    //arrayExamens_temp.clear();
    for (int i=0; i < arrayExamens.size();i++) {
        examens_temp = arrayExamens.get(i);

        /*
        * Si l'ID_Local est égale à -1 alors on affiche tous les examens
        */
        if (ID_Local.equals(-1)) {
            examens_temp.setId_Patient_Exam(examens_temp.getId_Patient_Exam());
            examens_temp.setId_exam(examens_temp.getId_exam());
            examens_temp.setExamIsDel(examens_temp.getExamIsDel());

            Object type= examens_temp.getTypeExamens();
            examens_temp.setTypeExamens(type);
            examens_temp.setDateExamens(examens_temp.getDateExamens());
            //panelExam.setSelectedIndex(i);
            arrayExamens_temp.add(examens_temp);
        }

        /*
        * sinon, affiche uniquement les examens correspondant au Patient
        */
        else {
            if (ID_Local.equals(examens_temp.getId_Patient())) { //si l'ID du patient sélectionné est égale à l'ID du fichier examens
                if (supprime==1) { // si supprime==1 on met un 0 pour
                    arrayExamens.get(i).setExamIsDel(0);
                }

                /*
                * on stocke dans 'exam_temp' les données du fichier 'examens.txt'
                * qui correspondent à la section du patient.
                */
                examens_temp.setId_Patient_Exam(examens_temp.getId_Patient_Exam());
                examens_temp.setId_exam(examens_temp.getId_exam());
                examens_temp.setExamIsDel(examens_temp.getExamIsDel());

                Object type= examens_temp.getTypeExamens();
                examens_temp.setTypeExamens(type);
                examens_temp.setDateExamens(examens_temp.getDateExamens());
                //panelExam.setSelectedIndex(i);
                arrayExamens_temp.add(examens_temp);
            }
        }
    }
    return arrayExamens_temp;
}
```

Parcours de l'arrayList 'arrayExamens' et stocke dans arrayExamens_temp tous les examens afin de les afficher

Pour la suppression d'examens, on met un '0' sur toutes les lignes correspondantes au patient sélectionné dans la liste.

Lecture.java

Cette méthode permet de lire le fichier passé en paramètre.

Ensuite, le programme découpera le fichier afin de récupérer les différents champs (nom, prénom,...) et stockera le résultat dans l'objet 'utilisateur'.

```
public class Lecture {  
    public static <arrayUtilisateur> ArrayList<Patients> litFichierPatients(ArrayList<Patients> arrayUtilisateur, String NomFichier) {  
        Patients utilisateur;  
  
        int memID;  
        File fichier = null;  
        FileReader fr = null;  
        BufferedReader br = null;  
  
        /*  
        * Lecture du fichier 'patient.txt'  
        */  
        try {  
            fichier = new File (NomFichier);  
            fr = new FileReader (fichier);  
            br = new BufferedReader(fr);  
            String line;  
            if (NomFichier=="patient.txt") {  
                // Boucle de lecture du fichier  
                while((line=br.readLine())!=null)  
                {  
                    String[] splitArray = null;  
                    splitArray = line.split(",");  
  
                    // on mets les infos lues dans l'array 'utilisateur'  
                    System.out.println("Supprimé "+splitArray[1]);  
  
                    int verif = Integer.parseInt(splitArray[1]);  
                    if (verif!=0) { //Si le patient est supprimé on ne l'affiche pas  
                        utilisateur = new Patients();  
                        utilisateur.setID(Integer.parseInt(splitArray[0]));  
                        utilisateur.setPatientIsDel(Integer.parseInt(splitArray[1]));  
                        utilisateur.setNom(splitArray[2]);  
                        utilisateur.setPrenom(splitArray[3]);  
                        utilisateur.setTextSexe(splitArray[4]);  
                        utilisateur.setNSS(splitArray[5]);  
                        utilisateur.setDateNaissance(splitArray[6]);  
                        utilisateur.setDateEntree(splitArray[7]);  
                        utilisateur.setDateSortie(splitArray[8]);  
                        utilisateur.setTelephone(splitArray[9]);  
                        arrayUtilisateur.add(utilisateur);  
                        memID= utilisateur.setID(Integer.parseInt(splitArray[0])); // on met le  
                        System.out.println("memID: "+memID);  
                    }  
                }  
            }  
        }  
        catch (IOException ex){  
            System.out.println(" Il y a une erreur de lecture du fichier");  
        }  
        return arrayUtilisateur;  
    }  
}
```

-Ouverture du fichier + mise en buffer.

-Boucle qui parcourt le fichier ligne par ligne.
- Puis on sépare (split) les champs (nom, prénom....).
-On stocke ces informations dans arrayUtilisateur

Ecriture.java

Cette méthode écrit dans le fichier au format txt les informations liées aux clients.

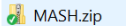
```
public class Ecriture {  
    public static void EcriturePatients(ArrayList<Patients> arrayUtilisateur, ArrayList listPatients ) {  
        /*  
        * Enregistrement des données de l'objet 'utilisateur'  
        */  
        try {  
            File f=new File("patient.txt");  
            FileWriter fw = new FileWriter(f,false);  
            BufferedWriter buffer = new BufferedWriter(fw);  
  
            for (int i = 0;i<listPatients.getModel().getSize();i++) {  
                buffer.write(listPatients.getModel().getElementAt(i).toString());  
                buffer.newLine();  
                buffer.flush();  
            }  
            buffer.close();  
        }catch (IOException ex) {  
        }  
        finally {  
        }  
    }  
}
```

-Boucle qui parcourt le liste 'listPatients' ligne par ligne.
-Puis on écrit celles-ci dans le fichier.

IV - INSTALLATION

Pour l'installation de l'application, veuillez suivre les étapes suivantes :

1 -Créer un répertoire d'installation sur votre disque dur.

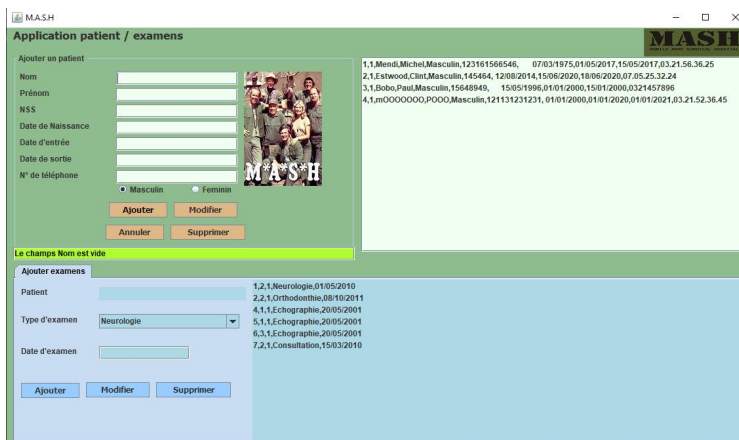
2 - Dans ce répertoire, copier le fichier 'MASH.ZIP'. 

3 - Ensuite, décompresser le fichier 'MASH.ZIP'.

4 - Vous obtenez le résultat suivant :

Nom
exams.txt
MASH.jar
mash.jpg
MASH.zip
Mash-logo2.jpg
patient.txt

5 -Il suffit de faire un double clic sur le fichier 'MASH.jar ' .
L'application se lance.



V – RESSOURCES EXTERNES

Montre quelques bases pour l'utilisation de WindowsBuilder :

https://koor.fr/Java/TutorialSwing/first_application.wp

Ressource sur différents sujets (bouton, fenêtres, etc) : <https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/23108-creez-votre-premiere-fenetre>

De nombreux sites ont été utilisés pour résoudre des situations. Il serait trop long d'énumérer ici tous ces sites.

VI - DYSFONCTIONNEMENTS

Les couleurs de l'interface devaient être uniquement en tons verts, mais la fenêtre de l'onglet 'ajouter examens' est restée en bleu. Après avoir parcouru tout le code, aucun `setBackground` avec un code couleur bleu n'y figurait. Incompréhensible !

La gestion des photos n'a pas été faite.

Il faudrait supprimé l'ID et la valeur de suppression de l'affichage.
Avoir un entête de colonne (nom,prénom,etc).

```
0,1,Mendi,Michel,Masculin,123161566546, 07/03/1975,01/05/2017,15/05/2017,03,21,56,36,25
2,1,Estwood,Clint,Masculin,145464, 12/08/2014,15/06/2020,10/06/2020,07,05,25,32,24
3,1,Bobo,Paul,Masculin,15648949, 15/05/1996,01/01/2000,15/01/2000,03,21,45,78,96
4,1,mOOOOOOO,POOO,Masculin,121131231231, 01/01/2000,01/01/2020,01/01/2021,03,21,52,36,45
```

```
1,2,1,Neurologie,01/05/2010
2,2,1,Orthodontie,08/10/2011
7,2,1,Consultation,15/03/2010
```

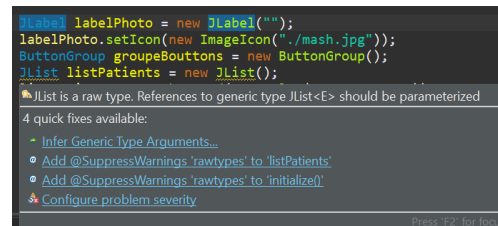
Idem pour les examens.

Difference entre l'exécution dans Eclipse et l'exécutable JAR. Par exemple dans l'exécutable la modification des patient ne fonctionne pas alors que dans Eclipse cela fonctionne. Je ne comprends pas pourquoi.

Message d'erreur : JList is a raw type. References to generic type JList<E> should be parameterized.

Je n'ai pas bien compris cette erreur malgré les explications du site :

<https://adiguba.developpez.com/tutoriels/java/7/> et par manque de temps j'ai dû laissé le code comme il est.



✚ VII - AMELIORATIONS ✚

Recherche : avoir une fonction de recherche (*pas fait par manque de temps*).

Avoir la possibilité de choisir tous les examens, par exemple avec un bouton 'voir tout'.

Mettre le nom de la personne en face du rendez-vous d'examens.

Avoir des thèmes d'affichages afin de changer l'aspect visuel.

Pré remplir la date d'entrée avec la date du jour.

Gérer le format du numéro de sécurité social.

Pouvoir gérer l'ordre d'affichage des colonnes.

Vérifier, lors de la création d'un nouveau dossier patient qu'il n'existe pas déjà. Idem pour les rendez-vous d'examens.

Avoir plus d'infos comme, le numéro de mobile, la mutuelle, etc...

Avoir le nom du médecin qui s'occupe de l'examen.

Lorsque l'on quitte l'application et que des données ont été modifié, il faudrait proposer une sauvegarde.

✚ VIII – VECU DU PROJET ✚

J'ai éprouvé beaucoup de difficultés sur ce projet car au démarrage je n'arrivais à assimiler le concept de la programmation orienté objet, ni les constructeurs, ni les arrayList, etc.

De plus, je n'arrivais pas à m'y retrouver dans le code de WindowsBuilder.

Etant perdu, au départ, je ne savais pas comment structurer mon programme (difficile de structurer quelque chose que l'on ne maîtrise pas), j'ai tout développé dans le Main 'Patient_Graphique.java' et ensuite j'ai divisé celui-ci avec différentes méthodes (Examens.java, recherche_examens.java, etc).

Je me suis, du coup, perdu à plusieurs reprises et ai dû refaire plusieurs parties de mon code.

J'en ai retiré qu'il est important de prendre plus de temps dans l'étude de la structure de base. Il aurait fallu que je puisse mieux comprendre le fonctionnement du Swing, des constructeurs, des arrayList afin de mieux savoir comment structurer mon programme.

J'ai maintenant mieux compris le fonctionnement de la POO et comment diviser le programme en méthodes.

