DÉPARTEMENT RÉSEAUX ET TÉLÉCOMMUNICATIONS

Rapport de stage

Rapport de stage rédigé par Marc DUGUE
PROMOTION 2010 2011

Lu et approuvé par :

Observatoire de la côte d'azur
Laboratoire Hippolyte FIZEAU
Parc Valrose
06108 Nice cedex 02
Tél. : 04 92 07 63 22

Responsable en entreprise : M. ABE Lyu
Lyu.Abe@unice.fr

Tutrice à l'IUT : Mme Céline Theys

Dominante stage : Informatique

Université de NICE SOPHIA-ANTIPOLIS
Institut Universitaire de Technologie
Département Réseaux et Télécommunications
650, route des Colles
06560 Valbonne
Tel 04.93.95.51.70
Fax 04.93.95.51.89
iutsoph.unice.fr

Mise en place d'outils d'aide au pilotage et au traitement de données pour l'expérience ASTEP (Antarctique/Dôme C)

Stage effectué du 11 avril au 17 juin 2011

Volume 1/1

Rapport rédigé par :

M. Marc DUGUE, étudiant en IUT département R&T, 2ème année

Copies du présent document :

1 copie à M. ABE
1 copie à l'IUT
1 copie à l'étudiant

Remerciements

Je souhaite remercier l'ensemble de l'équipe du laboratoire Fizeau ainsi que l'Observatoire de la Côte d'Azur pour m'avoir accueilli et soutenu au cours de ces deux mois de stage, et plus particulièrement M. Abe pour m'avoir donné la chance de pouvoir mener à bien ce projet.

Je tiens également à remercier l'ensemble de l'équipe pédagogique de l'IUT ainsi que ma tutrice à l'IUT, Mme Theys pour les connaissances et l'aide qu'ils m'ont apportées.

Table des matières :

Présentation de l'entreprise :

L'unité Fizeau est une des 4 Unités Mixes de Recherche de l'Observatoire de la Côte d 'Azur. Celui-ci est le résultat de la fusion entre l'Observatoire de Nice, fondé en 1881 par Raphaël Bischoffsheim et le CERGA, structure de recherche inter-universitaire implantée à Grasse et Calern.

Il s'agit d'un établissement à caractère administratif dirigé par M. Farrokh Vakili, et se compose d'environ 450 personnes.

Il est implanté sur 7 sites :

- le Mont Gros et le parc Valrose à Nice
- Villefranche sur mer
- Sophia Antipolis
- Roquevignon à Grasse
- le Plateau de Calern à Caussols et Cipières
- Rustrel (Laboratoire Souterrain à Bas Bruit)

Ses missions sont :

- Contribuer au progrès de la connaissance par le recherche fondamentale et appliquée
- Contribuer à la formation initiale et continue (étudiants et chercheurs)
- Contribuer à la diffusion de la culture scientifique

Dans le cadre de mon stage, j'ai été accueilli dans l'équipe de Recherche en physique Instrumentale (RPI) du laboratoire Fizeau.

Le principal objectif de l'équipe de RPI est de développer des outils matériels ainsi que des logiciels susceptibles d'améliorer les performances des méthodes d'observations.
Cela inclut de nombreux domaines tels que la résolution angulaire, la résolution et la couverture spectrale, la stabilité et la couverture temporelle, la sensibilité....
Le service de Recherche en physique Instrumentale s'occupe de l'élaboration conceptuelle jusqu'au traitement des données sans oublier la réalisation d'instruments.
Il n'existe malheureusement pas à l'heure actuelle d'organigramme détaillé de l'équipe.

Présentation du projet :

Durant ces deux mois de stages passés au sein de l'équipe de Recherche en physique Instrumentale (RPI), j'ai été amené à réaliser un outil de génération de "fichier d'observation" destiné au pilotage du télescope ASTEP400. Ce télescope de 40 cm est installé depuis 2009 en Antarctique sur la base Franco-Italienne de Concordia au Dôme C. Ces fichiers d'observation sont destinés à être décryptés par le logiciel de contrôle du télescope qui va exécuter une série d'actions (pointage d'une zone du ciel donnée, reconnaissance du champ stellaire, enregistrement cyclique, etc.) pendant toute la période hivernale australe. Cette programmation du télescope est nécessaire car l'instrument fonctionne 24 heures sur 24.

Ces fichiers comportent tous les paramètres nécessaires à la configuration de l'instrument et des séquences d'observation. Ces paramètres étaient auparavant générés à la main, mais ils requièrent un formatage bien défini pour pouvoir être décryptés par le programme de pilotage du télescope. Les risques de commettre une erreur de saisie étaient grandes, rendant le fichier illisible et constituant une source d'interruption de la séquence prévue.

Afin de simplifier cette étape de génération des fichiers et de réduire le risque d'erreur, il était devenu nécessaire de procéder à la création d'un outil dédié. La solution la plus pertinente trouvée était de réaliser un programme visant tout d'abord à présenter un interface graphique conviviale, utilisable par tous, afin qu'un maximum de personnes puissent proposer des observations pour cet instrument. Un point très important du projet était que le programme devait pouvoir être évolutif, car en effet, l'expérience ASTEP400 est encore en évolution et il est très probable que la syntaxe des fichiers d'observation va devoir être modifiée d'une manière ou d'une autre à l'avenir.

Abstract

I spent these two months working on a project for the ASTEP400 telescope in the RPI team at the Fizeau Laboratory of the Observatoire de la Côte d'Azur. My objective was to create a tool to automate the generation of observations files of this telescope installed in Antarctica at Dome C. This 40cm telescope is installed since 2009 in Antarctica, at the French-Italian Concordia Station at Dome C. These files are translated by the ASTEP control software in order to control the telescope in a series of actions (pointing a direction in the sky, recognize the star field, record images, etc.) during all the austral winter seaso

Indeed, those files were created manually before this project, so, there was a risk that they were wrong, because those files needs to be well formatted, it was a waste of time and an input error was possible. To solve this problem, the creation of a program convenient and easy to use was needed.

To meet this need, I had to establish a program that include a Graphical User Interface, to manage the input of data, and in a second time, i had to generate an obsevation file from the graphical interface. This file had to be complete, error free and easily usable. A very important point of the project was that the program needed to be evolutive, because the ASTEP400 experiment is still evolving and it is highly probable that the syntax of these observing files will need some changes in a way or in another.

I. Étude et organisation du projet :

Avant de pouvoir commencer la réalisation du projet, il était nécessaire de déterminer la façon dont allait être réalisé le projet, ainsi que les principaux intérêts du projet.
C'est pour cette raison que la première partie du projet a consisté a prendre en main les différents documents a disposition ainsi qu'à approfondir mes connaissances sur l'expérience.

1. Objectif du projet :

Le principal objectif du projet n'était pas de proposer au terme des deux mois de stage un projet complet et terminé, mais bien de débuter la création d'un programme allant pouvoir évoluer au cours des années à venir. Le but qu'il fallait ainsi garder à l'esprit était entre autre de permettre l'ajout ou la modification de paramètres d'observation facilement.

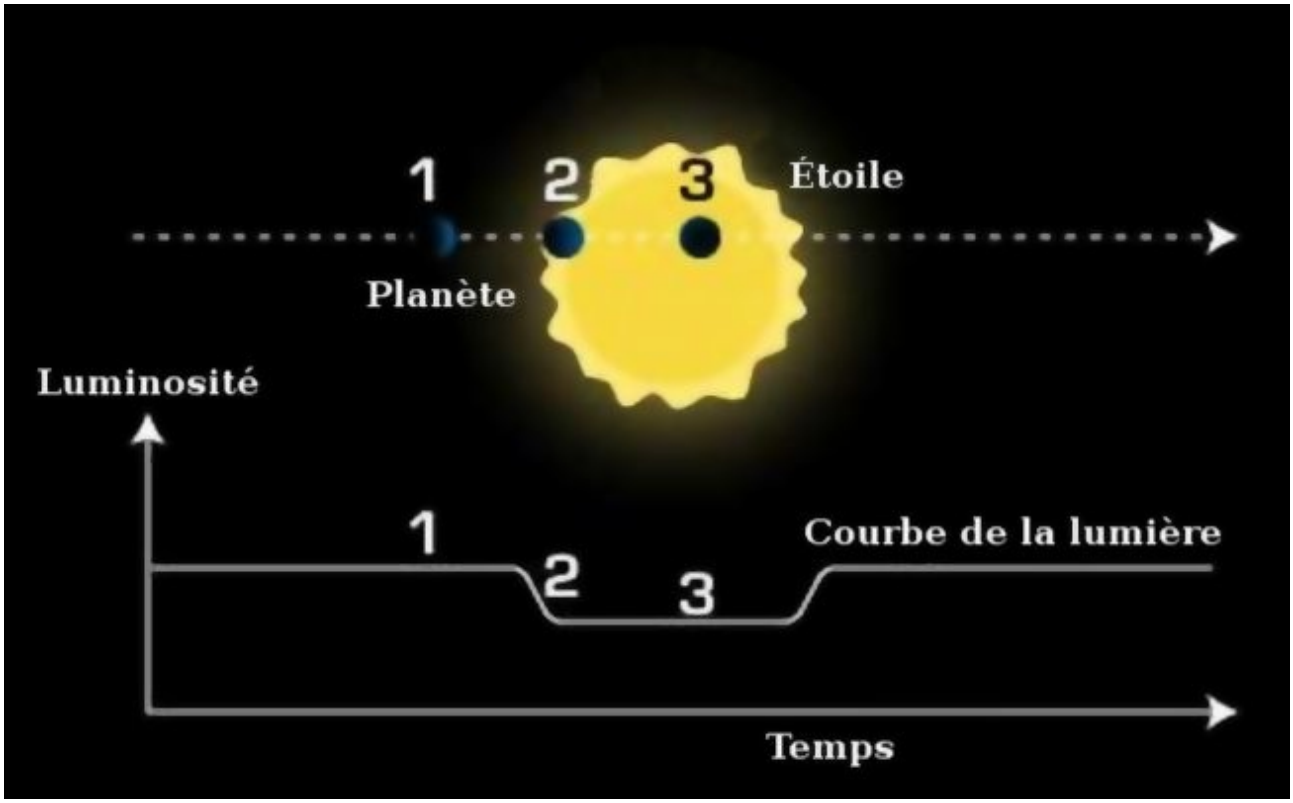
2. Principe de fonctionnement d'une observation :

L'objectif de ce programme est d'automatiser la création de fichiers d'observations destinés à être lus par le télescope ASTEP400. Ce télescope a été installé en Antarctique au Dôme C en 2009 pour un programme visant à détecter et caractériser des planètes extra-solaires par la méthode des transits (passage de la planète devant son étoile). Les observations peuvent être programmées en fonction de conditions particulières: fenêtres temporelles particulières, périodicité dans le temps, observation conditionnelle en fonction de la hauteur du soleil sur ou sous l'horizon, etc.

Le principe de fonctionnement du procédé est le suivant :

Afin de procéder à l'observation d'un certain champ, on utilise une étoile de repère, appelée étoile guide afin de centrer le champ d'observation. Lors de cette observation, on mesure la luminosité (appelée flux) de certaines étoiles, le but étant de mesurer les variations de celle-ci au cours du temps.

Si une diminution de ce l'intensité de ce flux est remarquée, on est alors peut-être en présence d'un transit, à savoir le passage d'une planète devant cet étoile. Cette méthode est également un moyen utilisé périodiquement afin de de déterminer la période orbitale de cette planète. En effet, il suffit de repérer l'intervalle entre les transits pour avoir une idée de cette période orbitale.



Variation de la luminosité d'une étoile lors du transit de la planète

Cette méthode est une méthode de détection des exoplanètes, c'est à dire des planètes orbitant autour d'une étoile autre que du Soleil.

3. Choix du langage informatique :

Afin de simplifier la création manuelle des fichiers d'observations et rendre la création de ceux-ci plus conviviale, il a été décidé de créer un programme, proposant une interface graphique, ayant pour but de recueillir les données saisies par les utilisateurs.
Une fois ces données récupérées, le programme devait être capable de générer le fichier texte final complet et sans erreur, directement utilisable par le télescope.

Il a donc fallu dans un premier temps décider quel langage informatique utiliser afin de coder ce générateur de script. Étant donné que ce programme devait pouvoir être utilisable par un grand nombre de chercheurs afin que ceux-ci propose leurs propres observations, il était nécessaire d'utiliser un langage compatible avec le maximum de système d’exploitation possible, notre choix s'est donc logiquement porté sur l'utilisation du langage JAVA.

4. Logiciels et documents a disposition :

Station de travail et OS :

Pour ce projet, la majorité du travail s'est effectué sur un ordinateur sous Windows 7 directement sur mon lieu de travail.

Documents :

Tout au long de ce projet, j'ai pu avoir accès à des documents qui m'ont guidés et servi de référence pour aboutir au résultat final attendu :

Je disposais en effet du manuel d'utilisateur ASTEP400 comportant de nombreuses informations sur le formatage du fichier final, celui-ci devait en effet correspondre exactement à la lettre près au type de programme attendu par ASTEP400
Voici à quoi celui-ci devait ressembler :

```
[FIELD_NAME] <Canopus>
[FIELD_TYPE] <TRANSIT>
[FIELD_TIME_UTC_BOUNDARIES] <Boundary min: 2009.12.07 14h00m00.000s, Boundary max: 2010.12.31
23h59m59.000s>
[MOUNT_ABS_POSITION] <ALT: 0.00000, AZ: 0.00000>
[MOUNT_TRACKING_TYPE] <SIDERAL>
[SCIENCE_STAR_CRD] <RA: 06 23 57.1, DEC: -52 41 44.3, ICRS>
[SCIENCE_STAR_INFO] <pmRA: 19.990, pmDE: 23.670, Parallax: 10.430, RadVel: 20.50>
[GUIDE_STAR_CRD] <RA: 06 23 57.1, DEC: -52 41 44.3, ICRS>
[GUIDE_STAR_INFO] <pmRA: 19.990, pmDE: 23.670, Parallax: 10.430, RadVel: 20.50>
[FIELD_RECOGNITION] <NO, ExpTime_ms=500>
[STORE_ROI] <NO, xc=256, yc=256, dx=128, dy=128, DataFolder="D:\DataASTEP400\Microlensing\">
  [OBS_NAME] <Flats>
  [TRACK_STAR] <NO, Exp_ms=2000, DetectionThreshold=-1>
  [OBS_SERVOGUIDINGSETPOINT] <YES>
  [OBS_SERVOFOCUS] <NO>
  [OBS_COND_SUNHEIGHT] <Sun height min (deg): none, Sun height max (deg): none>
  [OBS_COND_UTC] <UTC min: none, UTC max: none>
  [OBS_COND_PERIOD] <Period: 000d01h00m00.000s, Duration: 000d00h10m00.000s>
    [REPEAT_COUNT] <-1>
    [SYSTEM] <FlatSequence 10 0 15000 "8Mhz" 1>
  [OBS_NAME] <Science>
  [TRACK_STAR] <YES, Exp_ms=2000, DetectionThreshold=800>
  [OBS_SERVOGUIDINGSETPOINT] <YES>
  [OBS_SERVOFOCUS] <NO>
  [OBS_COND_SUNHEIGHT] <Sun height min (deg): none, Sun height max (deg): none>
  [OBS_COND_UTC] <UTC min: none, UTC max: none>
  [OBS_COND_PERIOD] <Period: 000d00h00m00.000s, Duration: 000d00h00m00.000s>
    [REPEAT_COUNT] <-1>
    [CAMS] <Acquisition 15000 "Science" "8Mhz" 0 1>
    [CAMS] <Acquisition 15000 "Science" "8Mhz" 0 1>
    [CAMS] <Acquisition 15000 "Science" "8Mhz" 0 1>
    [CAMS] <Acquisition 15000 "Science" "8Mhz" 0 1>
    [CAMS] <Acquisition 15000 "Science" "8Mhz" 0 1>
    [CAMS] <Acquisition 15000 "Dark" "8Mhz" 0 1>
[FIELD_END]
```

Extrait du manuel d'utilisateur ASTEP400

Ce fichier comporte deux sortes de données :

- Les caractères fixes, qui permettent de décrire le champ que le télescope attend :
  Ex : [FIELD_NAME]...
- Les données variables saisies par l'utilisateur, pouvant être une chaîne de caractère , un nombre entier, un choix parmi plusieurs possibilités :
  Ex : <Canopus>, (YES/NO)...

De plus, ce document m'a également permis de bénéficier d'une description précise de la syntaxe de chaque champ, ainsi que des valeurs prédéfinies dont l'utilisateur devait pouvoir avoir le choix :

8.2.1 FIELD_NAME

Description: spécifie un nom ou un identificateur pour ce champ. La valeur de ce descripteur n'a pas d'impact sur le fonctionnement du programme (purement indicatif).

Format: chaîne de 256 caractères au maximum pouvant contenir des espaces.

8.2.2 FIELD_TYPE

Description: définit le type d'observation qui sera indiqué dans le header FITS des images. La valeur de ce descripteur n'a pour l'instant pas d'impact sur le fonctionnement du programme.

Format: chaîne de 128 caractères au maximum dont on utilisera les valeurs prédéfinies ci-dessous.
Valeurs possibles:

TRANSIT_SRCH	Recherche de transit dans un champ
TRANSIT_OBJ	Observation de transit connu
MICROLENSING	Observation de microlentille gravitationnelle
CALIBRATION	Obervation de calibration (flat fields, séquences de darks,...)

Exemple de description pour les deux premiers champs

Logiciel et ressources utilisés :

Éclipse :

Il m'a été donné la possibilité de choisir les logiciels que je souhaitais afin de développer au mieux ce projet. Déjà familier de l'interface du logiciel Éclipse et satisfait des nombreuses fonctions qu'il propose, mon choix s'est porté sur celui-ci.

Éclipse est un *IDE (Integrated Development Environment)* , c'est à dire un outil disposant notamment d'un compilateur, d'un éditeur de texte, d'une interface graphique, dédié au développement de programmes, il s'agit de plus d'un logiciel libre.

JavaDoc :

La JavaDoc est une documentation regroupant les classes et méthodes de l'édition standard de Java. C'est un outil dont je me suis très souvent servi, sa consultation m'a en effet permis de connaître les syntaxes et utilisations des différentes classes et méthodes dont je pouvais avoir besoin pour mener à terme mon projet.

5. Organisation du programme :

Afin de programmer les différents aspects de ce projet, j'ai séparé celui-ci en différentes classes afin de simplifier sa réalisation, ainsi, le programme se compose de plusieurs classes dont les principaux objectifs sont les suivants :

- Classes destinées à automatiser la création des différents composants graphiques : JLabel (composant contenant un texte), JComboBox (liste déroulante) , JTextField (zone de texte) ...

- Classes « d'écoutes », qui ont pour but d'attendre que l'utilisateur interagisse avec un certain élément (bouton Valider, nouvelle observation...)

- Classes possédant un aspect utilitaire, comme convertir le format d'une date, ou bien gérer une collection...

- Classe « Main »

Le résultat souhaité devait être évolutif, à savoir par exemple l'ajout de paramètres devait pouvoir être accompli facilement, il était important de pouvoir modifier simplement ce programme si besoin est.

Voici les principales étapes de la création du fichier final :

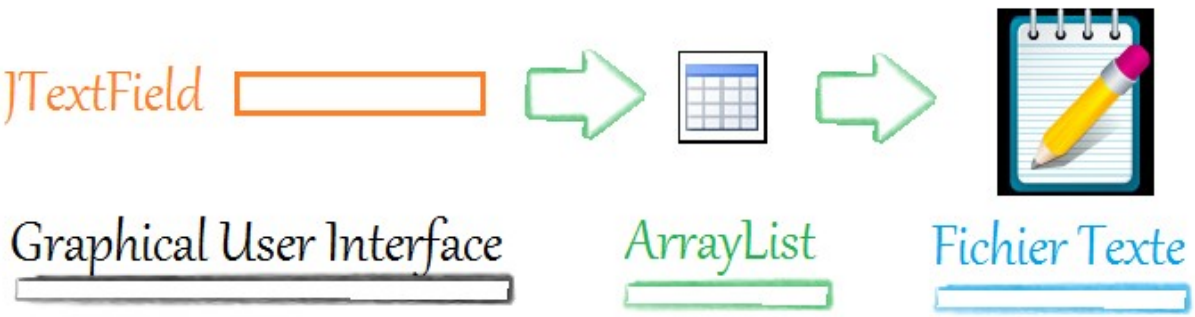


Schéma montrant le principe de fonctionnement souhaité du programme :

Conclusion :

Après avoir posé les bases de l'organisation ainsi que du fonctionnement du programme, il ne restait plus qu'a passer à l'étape suivante du projet, à savoir la réalisation du programme, qui devint alors plus facile du fait de cette étude préliminaire.

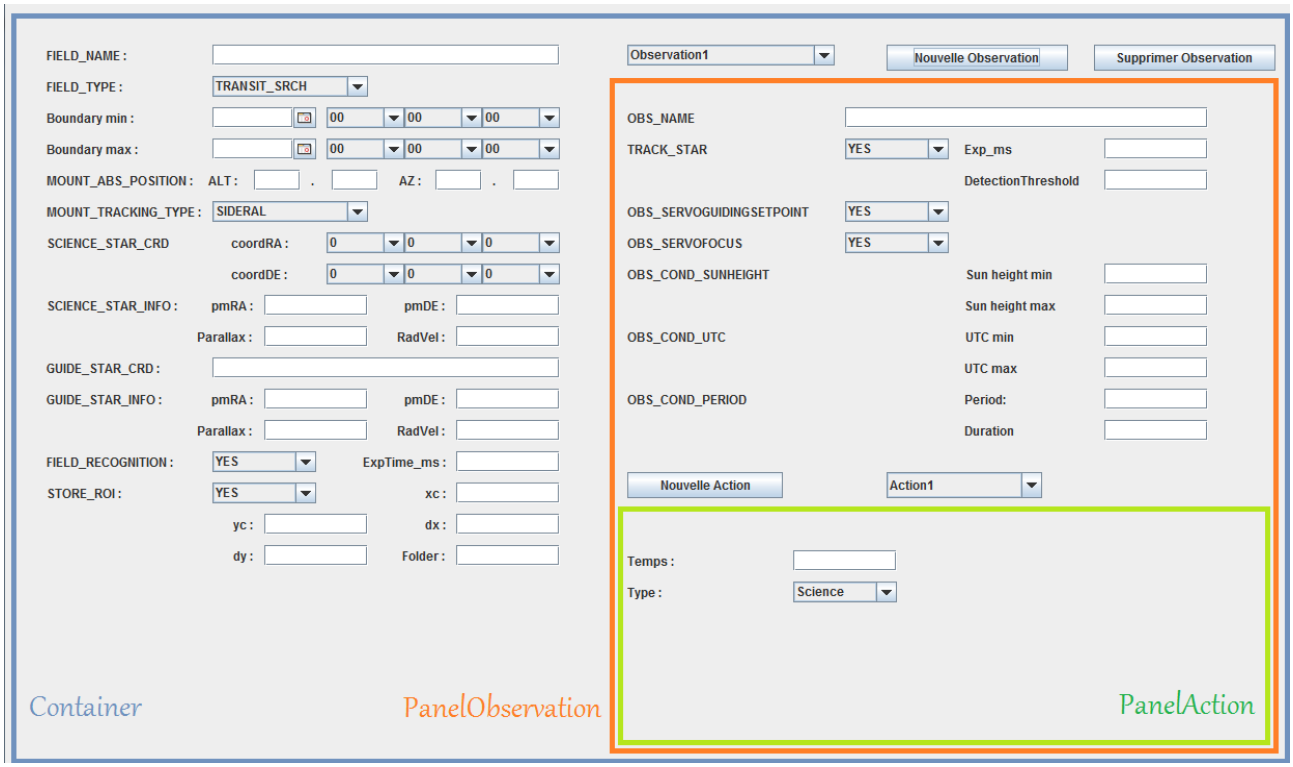
II. Création de l'interface graphique

Après avoir bien analysé et étudié les objectifs du projet, j'ai choisi, de par le nombre important de données à remplir pour l'utilisateur, et donc de composants graphiques à ajouter, de créer des classes dédiées à la création des différents composants graphiques et panels.

1. Création des composants graphiques :

Afin de mieux comprendre les éléments expliqués plus tard, il est important de bien pouvoir visualiser a quoi correspondent les différents aspects graphiques du programme réalisé

Voici ce a quoi ressemble l'interface graphique :



Le carré vert correspondant à la zone ou sont fixés les différents PanelAction, l'orange à la zone où sont fixés les différents PanelObservation, et le carré bleu correspondant au panel Container, englobant les autre panels.

Presque tous les composants graphiques utilisés pour recueillir les données à écrire dans le le fichier texte final sont créés avec une classe intitulée « *Créer....* »
En effet, ces classes permettent la création rapide et intuitive des différents composants graphiques nécessaires, les ajoutent à un JPanel, les positionnent et gère leur dimensions :

Ces classes ont pour but de simplifier la création des différents composants, mais également d'automatiser l'ajout de ces composants dans une ArrayList. Les valeurs des différentes données remplies dans ces composants graphiques sont ensuite récupérées et stockées dans un tableau afin

d'être ré-utilisées lors de la création du fichier.

Une des raisons de la création de ces classes est également que le programme devait être pensé de manière évolutive, car il ne faisait aucun doute que ce programme allait subir des modifications plus tard.

Exemple :

```
// Création des JLabel ainsi que des JTextfield utilisés pour receuillir les données /

new CreerJTextFieldDansContainer("FIELD_NAME :",40,40,150,335);
```

Ici, un JLabel contenant la chaîne de texte « FIELD_NAME » positionnée en (40,40) dans le JPanel Container ainsi qu'un JTextField sont créés grâce à la fonction *CreerJTextFieldDansPanelObservation*.

Les chiffres 150 et 335 correspondent respectivement à la taille en x du JLabel et la taille en x du JTextField. Le JTextField ainsi crée est ajouté à l'ArrayList *ArrayListTextField*, collection contenant presque tous les JTextField du programme.

```
public CreerJTextFieldDansContainer(String unLabel, int unx, int uny
        , int uneTaillexLabel, int uneTaillexTextField)
{
    Label=new JLabel(unLabel);
    Textfield= new JFormattedTextField();

    Main.Container.add(Label);
    Label.setBounds(unx,uny,uneTaillexLabel,20);

    Main.Container.add(Textfield);
    Textfield.setBounds(unx+uneTaillexLabel+10,uny,uneTaillexTextField,20);

    // Chaque JTextField crée est ajouté à l'ArrayList ArrayListTextField et incrèmente /
    // le NbJTextField qui sera utilisé pour définir le nombre d'objets présents /
    // dans l'ArrayList /

    Main.ArrayListTextField.add(Textfield);
    BoutonValider.NbJTextField++;
    NbJTextFieldDansContainer++;


}
```

Voici le code d'un constructeur de la classe CreerJTextFieldDansContainer. :

On y voit la création d'un Label, d'une zone de texte ainsi que leur ajout au Panel *Container*. On peut remarquer l'incrémentation des variables *NbJTextField* correspondant au nombre de JTextField total existant, et *NbJTextFieldDansContainer*, correspondant comme son nom l'indique, au nombre de JTextField dans le JPanel *Container*.

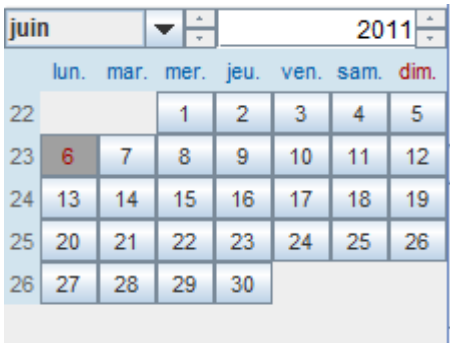
On constate également que ce principe a permis de simplifier la constitution des ArrayList : ainsi, tout objet créé par cette classe est automatiquement ajouté à la fin de l'ArrayList *ArrayListTextField* et, de ce fait, l'ajout manuel de chaque composant dans l'ArrayList n'est pas nécessaire, cela est gérée directement lors de la création de l'objet.

Le même principe a été appliqué pour les classes :

- CreerJTextFieldDansPanelObservation
- CreerJComboBoxDansContainer
- CreerJComboBoxDansPanelObservation

Celles-ci sont en effet similaires à la différence qu'elles concernent l'ajout soit d'un JTextField, soit d'un JComboBox, dans *PanelObservation*, ou *Container* : container étant le JPanel « principal», contenant les JPanel *PanelObservation* et *PanelAction*.

Afin de permettre à l'utilisateur de disposer d'un moyen convivial de choisir les dates de début et de fin de chaque observation, je me suis servi du calendrier *JDateChooser* qui est un logiciel libre :



2. Création de la frame :

La classe Main, permet principalement la création des différents composants graphiques et boutons, ainsi que de la frame *Frame1*, contenant les différents JPanel :

```
    static JFrame Frame1 = new JFrame();
    static JPanel Container = new JPanel();

                ...
public static void main(String[] args)
{

                ...

 // Paramètrage de la Frame

 Frame1.setResizable(false);
 Frame1.setTitle("GUI java");
 Frame1.setSize(1250, 900);
 Frame1.setLocationRelativeTo(null);
 Frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 Container.setLayout(null);
```

On y voit ici l'initialisation et le paramétrage de la frame ainsi que la création du JPanel *Container*.

```
new CreerJTextFieldDansContainer("FIELD_NAME :",40,40,150,335);
```

Ici, on voit par exemple que la classe CreerJTextFieldDansContainer est utilisée afin de créer une JTextField plus facilement, et surtout plus rapidement.

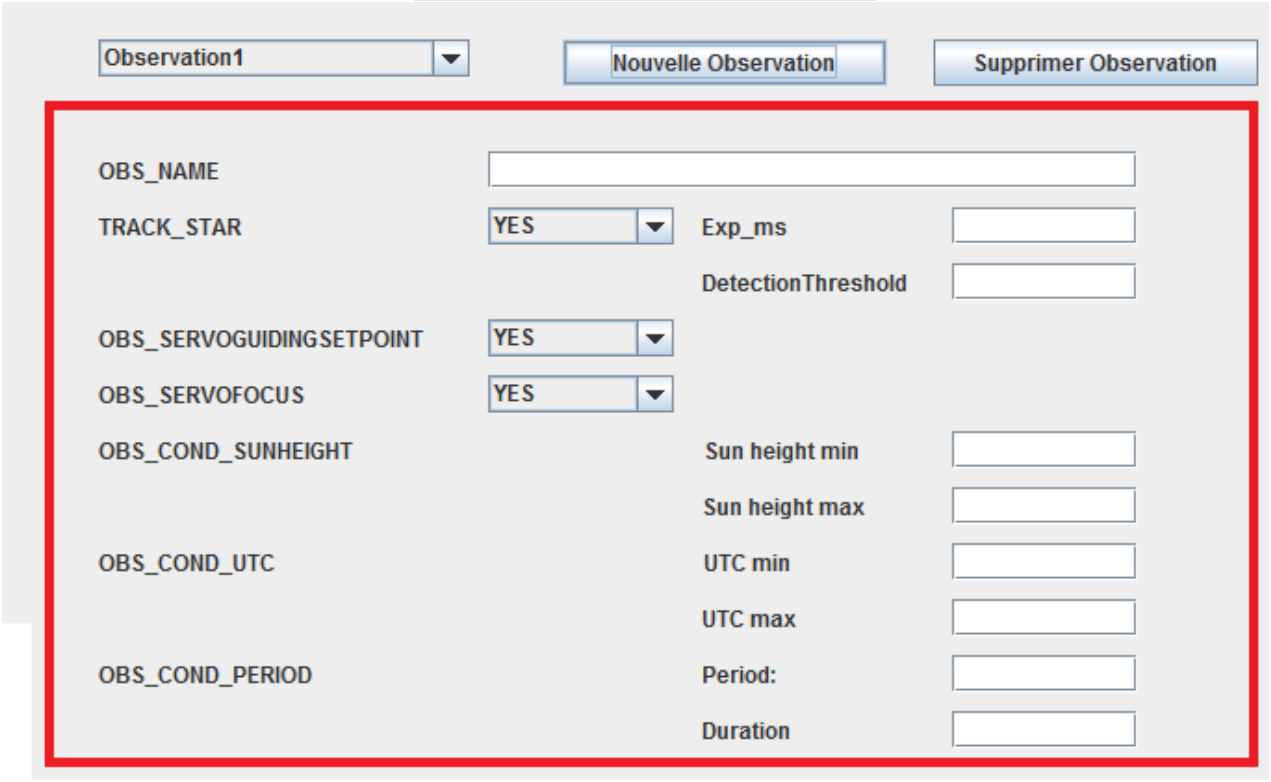
3. Création d'une nouvelle observation :

A chaque fichier veut être associées plusieurs observations : il a donc fallu permettre un moyen de créer celles-ci par l'implémentation d'une fonction supplémentaire :

La classe NouvelleObservation :

Cette classe a pour objectif, lors d'un clic sur le bouton « Nouvelle Observation », de créer un nouveau panel, contenant tous les composants graphiques nécessaires à la complétude des informations concernant cette nouvelle observation. Celui-ci gère également l'ajout de ces « panel » et des données saisies dans les ArrayList concernées.

Création d'une nouvelle observation



JPanel crée

Création du bouton Observation :

Dans la classe « Main » :

```
public class Main
{  …

   static JButton Observation = new JButton("Nouvelle Observation");
   …
   public static void main(String[] args)
   {  …

      // Ajout du bouton "Créer Observat

      Container.add(Observation);
      Observation.setBounds(850,40,175,2)

       …

   }
```

On voit ici que le JButton *Observation* est crée, puis ajouté au JPanel *Container*, et enfin placé et redimensionné dans celui-ci.

Principe de la mise sur écoute du bouton « Nouvelle Observation » :

```
public class NouvelleObservation
{  …

   class ObservationButtonListener implements ActionListener
   {

       public void actionPerformed(ActionEvent e)
       {

                    • • •


        }


public NouvelleObservation(JButton unBoutonObservation)
{
    unBoutonObservation.addActionListener(new ObservationButtonListener());
}
```

Ainsi, lors de l'instanciation de la classe « Nouvelle Observation » avec en paramètre le bouton *Observation*, un nouveau panel *PanelObservation* est crée, contenant de nouveaux composants graphiques créés à partir de d'une classe de type « *Créer ... DansPanelObservation* » .

Chaque clic entraînera la création d'un panel et réservera un nombre de places dans chaque ArrayList concernées égal au nombre d'éléments du type contenu par cet ArrayList présents dans une observation :

```
public void actionPerformed(ActionEvent e)
{   NbJTextFieldDansUnPanelObservation=0;
    NbJComboBoxDansUnPanelObservation=0;


  if (ComboBoxObs==null)
  {
      VObservation = new Vector();
      VObservation.addElement("Observation"+i);

      PanelObservation = new JPanel();
      BoutonAction= new JButton("Nouvelle Action");
      ComboBoxObs = new JComboBox(VObservation);

      new CreerJTextFieldDansPanelObservation("OBS_NAME",0,0,200,350);
      NbJTextFieldDansUnPanelObservation++;
      new CreerJTextFieldDansPanelObservation("Exp_ms",325,30,125,100);
      NbJTextFieldDansUnPanelObservation++:
```

• • •

```
       Main.Container.add(ComboBoxObs);
       Main.Container.add(PanelObservation);

       PanelObservation.setLayout(null);
       PanelObservation.setBounds(600,100,650,700);

       PanelObservation.add(BoutonAction);
       BoutonAction.setBounds(0,350,150,25);


       Main.ArrayListPanelObservation.add(PanelObservation);

       NbJPanelObservation++;
       NbObservation++;
       i++;

       ComboBoxObs.setBounds(600,40,200,20);
       ComboBoxObs.setModel(new DefaultComboBoxModel(VObservation));
```

Voici un extrait de NouvelleObservation, on y voit la création des différents composants graphiques, ainsi que du panel *PanelObservation*.

La condition if (*ComboBoxObs*==null) a pour objectif de vérifier que la liste déroulante permettant de sélectionner les différentes observations existe . Si elle n'existe pas, alors il y a création d'un « vector » *Observation* ainsi que d'une JComboBox *ComboBoxObs*. Si elle existe, il y a alors ajout d'un nouvel élément « Observation » + i, i étant incrémenté à chaque nouvelle observation.
Viens ensuite l'ajout dans le panel Container de la liste déroulante et de *PanelObservation* ainsi que le placement de ceux-ci et l'ajout du panel dans l'ArrayList *ArrayListPanelObservation*.

On peut également remarquer qu'à chaque nouvelle observation, les variables comptant le nombre de JTextField et de JComboBox dans *PanelObservation* sont remises à 0. Cela a pour objectif de véritablement comptabiliser le nombre de JTextField et JComboBox par *PanelObservation*, sinon, en effet ces variables comptabiliseraient le nombre total des éléments de chaque type contenu dans toutes les observations, et non pas d'une seule observation, ce qui fausserait les calculs exécutés plus tard.

4. Création d'une nouvelle action :

Le même principe a été appliqué pour la création d'une nouvelle action, à la différence qu'ici une nouvelle ArrayList a été utilisée pour le stockage des données recueillies pour chaque nouvelle action, en effet, étant donné qu'à chaque observation devait pouvoir être associées plusieurs actions, il devenait plus simple de séparer ces données afin de les traiter indépendamment.

```
NbJTextFieldDansUnPanelAction=0;

PanelAction = new JPanel();
Main.ArrayListPanelObservation.get(ObservationSelectionee.NumObservationInt-1)
.add(PanelAction);
PanelAction.setLayout(null);
PanelAction.setBounds(0,400,650,250);

PositionActionaRajouter=0;
PositionActionaRajouter=NouvelleObservation.TabNbActionParObs[0];

if (ObservationSelectionee.NumObservationInt!=1)
    for (i=1;i<=ObservationSelectionee.NumObservationInt-1;i++)
        PositionActionaRajouter=PositionActionaRajouter +
        NouvelleObservation.TabNbActionParObs[i];

new CreerJTextFieldDansPanelAction("Temps :",0,25,150,100);
new CreerJComboBoxDansPanelAction ("Type :",0,55,150,100,NouvelleObservation.VTypeAcquisition);
```

On voit ici qu'un « panel » *PanelAction* est crée, puis ajouté à l'observation auquel il correspond, puis positionné dans ce *PanelObservation*.

Après cela, si l'observation sélectionnée n'est pas la première observation, on utilise une boucle jusqu'au numéro de cet observation en ajoutant chaque fois le contenu de l'index correspondant du tableau *TabNbActionParObs*.
Ce tableau permet de stocker pour chaque observation, le nombre d'actions que celle-ci contient,

ainsi, la case index 0 du tableau contient le nombres d'action que l'observation numéro 1 comprend. L'utilisation d'une boucle itérative ne pose dans ce programme pas de problème en effet, le nombre de créations d'actions / d'observations demandés ne sera jamais trop important et donc ne risque pas d'utiliser trop de ressources.

Cela permettra de retrouver l'index d'une certaine action dans l'ArrayList contenant les différentes action, prenons par exemple :

Index	0	1	2	3
Nb Actions	3	4	3	2

N° PanelObs	1	2	3	4

Ici la 1ère observation contient 3 actions, la 2ème en contient 4, la 3ème en contient 3, et la 4ème observation contient 2 actions.

De cette manière, on pourra facilement retrouver l'index correspondant à l'action sélectionnée : en effet, si l'on veut retrouver l'index de l'action numéro 3 de l'observation 3, il suffit d'ajouter le nombre d'actions des panel 1 et 2 soit 3 + 4 = 7 puis de rajouter le numéro de l'action sélectionnée, à savoir 3 :

l'index de l'action numéro 2 de l'observation 3 dans l'ArrayList *l'ArrayListPanelAction* est donc (7 + 3) = 10.

Ainsi, si l'on veut rajouter une nouvelle action pour l'observation numéro 3, celle ci sera ajouté à l'index 11 de l'ArrayList, et le tableau sera mis a jour par l'opération :

```
NouvelleObservation.TabNbActionParObs[ObservationSelectionee.NumObservationInt-1]++;
```

Conclusion :

L'automatisation de la création des différents éléments graphiques m'a fait gagner un temps considérable et m'a également permis d'assurer la possibilité d'intégrer de nouveaux paramètres d'observations à tout moment par l'ajout simple d'une ou deux lignes de code. Cependant, il restait encore à créer des méthodes destinés à la gestion de toutes les données envoyées par l'utilisateur.

III. Analyse et traitement des données

Après la création de l'aspect graphique du logiciel, j'ai dû mettre en place des méthodes afin de pouvoir analyser et utiliser les interactions ainsi que les données saisies par les utilisateurs du logiciel. Cela signifiait la mise sur écoute des différents boutons et des listes déroulantes, ainsi que le traitement des données désirées par l'utilisateur.

1. Sélection d'une observation :

Afin de pouvoir naviguer entre les différentes observations, une JComboBox a été utilisée, celle-ci permet en effet d'afficher le panel correspondant à l'observation sélectionnée :

Mise sur écoute de la liste déroulante

```
public class ObservationSelectionee
{
int i;
String NumObservationString;
static int NumObservationInt=1;

    class ObservationComboBoxListener implements ItemListener
    {
        public void itemStateChanged(ItemEvent e)
        {

                                ...


        }
    }

    public ObservationSelectionee(JComboBox uneJComboBox)
    {
        uneJComboBox.addItemListener(new ObservationComboBoxListener());
    }
}
```

Action effectuée en cas de changement de sélection :

```
    for(i=0;i<NouvelleObservation.NbJPanelAction-1;i++){
        Main.ArrayListPanelAction.get(i).setVisible(false);
    }


NumObservationString=NouvelleObservation.ComboBoxObs.getSelectedItem().toString().substring(11,12);

NumObservationInt = Integer.parseInt(NumObservationString);


Main.ArrayListPanelObservation.get(NumObservationInt-1).setVisible(true);
```

On voit ainsi que lors de la sélection d'une observation, tous les PanelObservation sont rendus non-visibles par setVisible(false), puis le numéro de l'observation sélectionnée est obtenu en récupérant la lettre comprise entre les positions 11 et 12 de la String sélectionnée dans la liste déroulante (toutes les String contenues dans la liste déroulante étant de la forme « Observation+i »).
Celle-ci est ensuite convertie en entier par Integer.ParseInt(UneChaineDeCaractère) . Elle est enfin utilisée afin de rendre visible le panel de l'observation choisie, qui est contenue dans l'ArrayList à la position de ce chiffre auquel on soustrait 1 (le 1er index étant 0 et non pas 1) .

2. Suppression d'une observation :

Afin de permettre de pouvoir supprimer une observation qui aurait été créé en trop, il a fallu mettre au point une fonction permettant de supprimer celle-ci facilement, donc en passant par l'interface graphique : ainsi, il a été décidé de rajouter un bouton « Supprimer » qui lors de son utilisation, permettrait d'effacer l'observation sélectionnée, aussi bien graphiquement que dans le code, voici le principe de fonctionnement :

On met sous écoute le bouton Supprimer :

Dans Main :

Un nouvel Objet de type SupprimerObservation est crée avec comme paramètre le bouton SupprimerObservation :

```
new SupprimerObservation(SupprimerObservation);
```

Dans SupprimerObservation :

```
public class SupprimerObservation
{
    class SupprimerButtonObservationListener implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {

                        ...

        }

    }

    public SupprimerObservation(JButton unBoutonObservation)
    {
        unBoutonObservation.addActionListener(new SupprimerButtonObservationListener());
    }

}
```

Puis, dans la classe SupprimerObservation, action à effectuer lorsque le bouton « Supprimer » est utilisé :

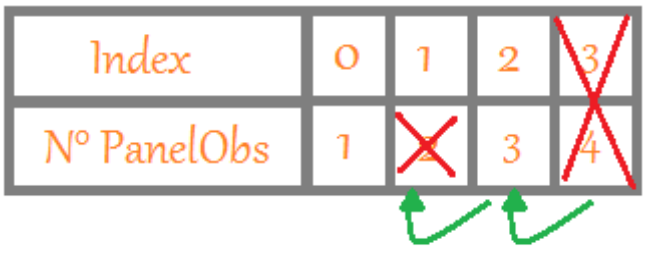
```
public void actionPerformed(ActionEvent e)
{
    int j;

    if(ObservationSelectionee.NumObservationInt!=NouvelleObservation.NbJPanelObservation)
    {
        for(j=ObservationSelectionee.NumObservationInt-1;
        j<NouvelleObservation.NbJPanelObservation-1;j++)
        {
            JPanel PanelTemp = new JPanel();

            PanelTemp = Main.ArrayListPanelObservation.get(j+1);
            PanelTemp.setBounds(600,100,650,700);

            Main.ArrayListPanelObservation.get(j).setVisible(false);
            Main.Container.add(PanelTemp);

            Main.ArrayListPanelObservation.set(j,PanelTemp);
            Main.ArrayListPanelObservation.get(j).setVisible(true);
            NouvelleObservation.ComboBoxObs.setSelectedItem(0);
        }
    }
```

Comme on peut le voir, si le bouton « Supprimer » est pressé, le programme réalise un test « if » afin de voir si l'observation sélectionnée lors de la demande de suppression est la dernière observation crée.
Si celle-ci n'est pas la dernière observation crée, alors une boucle sera réalisé afin de récupérer le contenu des différents panel des observation dont le nombre est supérieur ou égal à celle sélectionnée.

Si celle-ci est la dernière observation créé, alors, ce panel est rendu non visible :

```
else
{
   Main.ArrayListPanelObservation.get(ObservationSelectionee.NumObservationInt-1).setVisible(false);
}
```

L'opération consiste en fait à réaliser une sorte de « décalage vers la gauche » afin de supprimer l'observation en trop. Voici un schéma montrant *ArrayListPanelObservation*, ainsi que le numéro des panel d'observation contenus dans cette ArrayList en fonction de l'index :



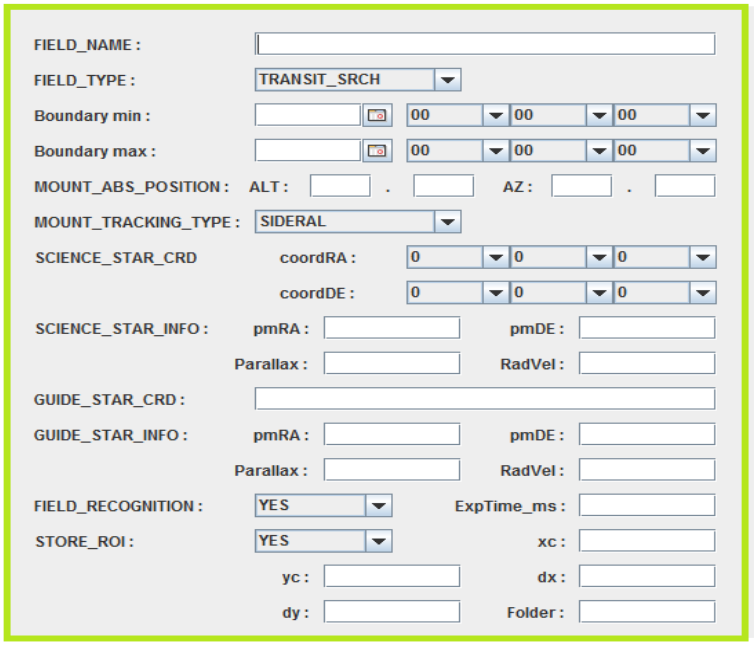
Ici, afin de supprimer *PanelObservation* numéro 2, on va décaler vers la gauche les panels 3 et 4, de manières à ce que le *PanelObservation* numéro 2 soit écrasé, puis on supprimera le dernier panel.

```
Main.ArrayListPanelObservation.remove(NouvelleObservation.NbJPanelObservation-1);
```

Après cela, il faudra encore supprimer les JTextField et JComboBox présents dans les ArrayList et que comportaient ces panels :

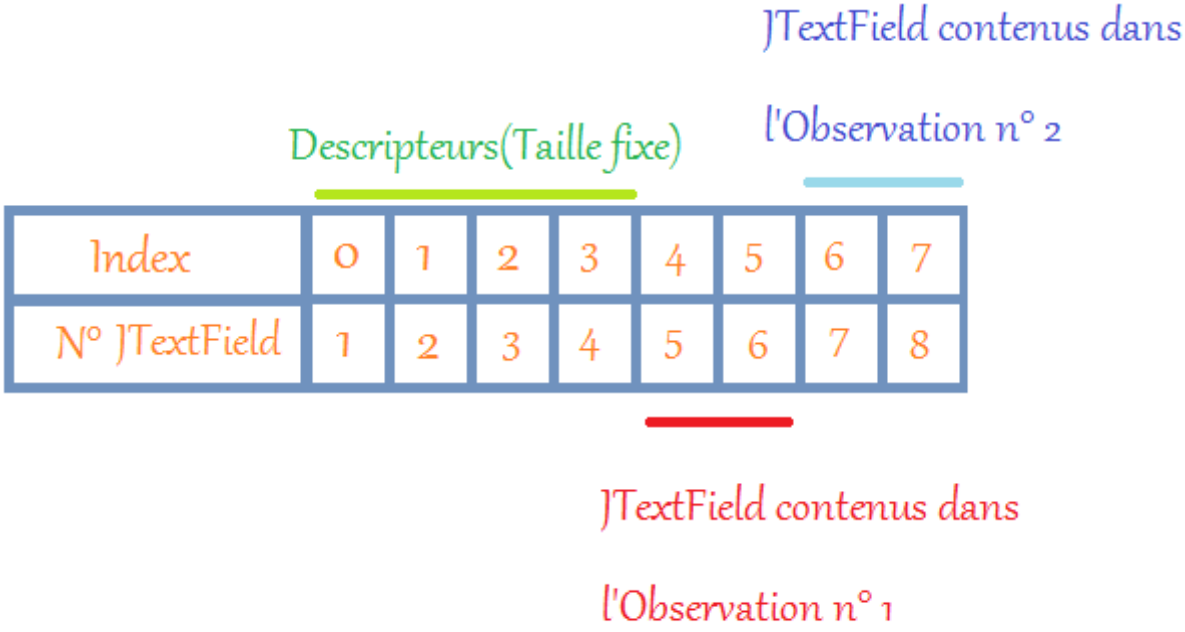
```
for(j=0;j<NouvelleObservation.NbJTextFieldDansUnPanelObservation;j++)
{
Main.ArrayListTextField.remove(CreerJTextFieldDansContainer.NbJTextFieldDescripteurs-1+
        (ObservationSelectionee.NumObservationInt-1)*
        NouvelleObservation.NbJTextFieldDansUnPanelObservation+1);
}

for(j=0;j<NouvelleObservation.NbJComboBoxDansUnPanelObservation;j++)
{
Main.ArrayListComboBox.remove(CreerJComboBoxDansContainer.NbJComboBoxDescripteurs-1+
        (ObservationSelectionee.NumObservationInt-1)*
        NouvelleObservation.NbJComboBoxDansUnPanelObservation+1);
}
```

Rappel : Il y a dans le véritable programme 20 JTextField descripteurs, et 18 JComboBox descripteurs, leur index dans l'ArrayList est fixe, au contraire des JTextField contenues dans les différentes observations, qui varient en fonction du nombre de JTextField par observation :



Descripteurs

Explication de l'opération :



Prenons pour un exemple l'ArrayList : *ArrayListTextField*.
Prenons un nombre de JTextField utilisé pour la partie réservée aux descripteurs égal à 4 JTextField.
Supposons que chaque observation contienne 2 JTextField, et que l'on veuille supprimer l'observation numéro 2 :
Dans notre exemple cela signifie supprimer les JTextField présents dans les index numéro 6 et 7 de l'ArrayList, c'est à dire, ces indices correspondent au :

Nombre de descripteurs - 1 + (Numéro de l'observation - 1) * Nombre de JTextField par observation + 1, on utilise une boucle afin de répéter la suppression autant de fois qu'il y a de JTextField par observation.

NbJTextFieldDescripteurs-1+ (NumObservationInt-1 NbJTextFieldDansUnPanelObservation)+1*

Vérification :
(4-1)+(2-1)*2+1=6 : Index de la 1ère JTextField à supprimer.

3. Utilisations des ArrayList :

Déclaration des différentes ArrayList :

```
public class Main
{

   static ArrayList<JTextField> ArrayListTextField;
   static ArrayList<JComboBox> ArrayListComboBox;
   static ArrayList<JComboBox> ArrayListComboBoxActions;
   static ArrayList<JPanel> ArrayListPanelObservation;
   static ArrayList<JPanel> ArrayListPanelAction;
```

Instanciation des ArrayList :

```
public static void main(String[] args)
{
    // Création des ArrayList utilisés pour stocker les JTextField, JComboBox et JPanel/

    ArrayListTextField = new ArrayList<JTextField>();
    ArrayListComboBox = new ArrayList<JComboBox>();
    ArrayListPanelObservation = new ArrayList<JPanel>();
    ArrayListPanelAction = new ArrayList<JPanel>();
    ArrayListComboBoxActions = new ArrayList<JComboBox>();
```

Rôle des ArrayList :

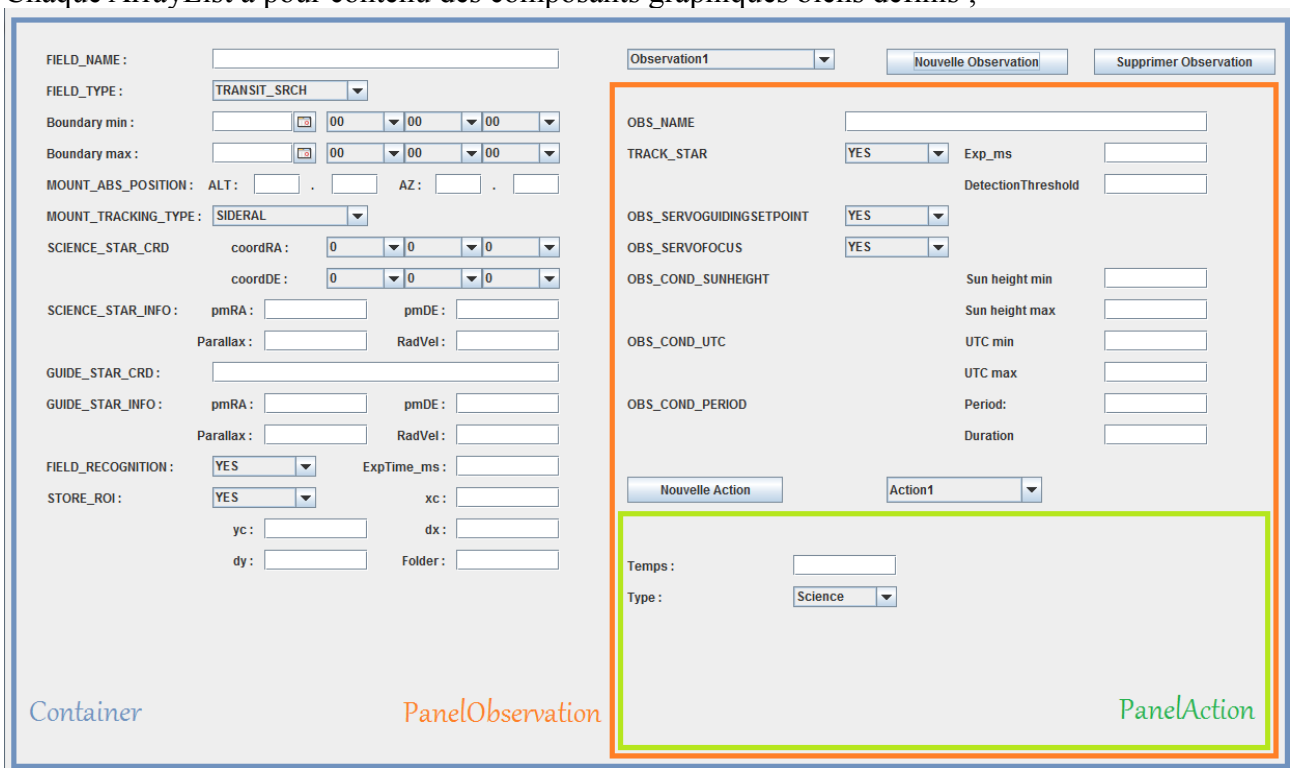
Les ArrayList sont des collections, elles peuvent contenir n'importe quel type d'objet et s'agrandissent automatiquement lorsque le nombre d'éléments stocké dépasse la taille de l'ArrayList.

Les différentes ArrayList créés ci-dessus ont chacune un rôle différent :

Alors que les ArrayList *ArrayListTextField* et *ArrayListComboBox* ont pour but de récupérer les informations saisies grâce à l'interface graphique afin d'être utilisées dans la génération du code final, l'intérêt des autres ArrayList s'avèrent différent : en effet, *ArrayListPanelObservation* et *ArrayListPanelAction* servent quant à elle à stocker les différents panels crées lors de la génération de nouvelles observations ou actions.

Contenu des différentes ArrayList :

Chaque ArrayList a pour contenu des composants graphiques biens définis ;



A l'exception des JTextField et JComboBox de *PanelAction*, qui elles sont contenues dans *ArrayListTextFieldDansPanelActions* et *ArrayListComboBoxDansPanelActions*, toutes les autres JTextField sont contenues dans *ArrayListTextField* et les JComboBox dans *ArrayListComboBox*.

Quant aux différents PanelObservation et PanelAction, ils sont respectivement contenus dans *ArrayListPanelObservation* et *ArrayListPanelAction*.

Accès au contenu et écriture du fichier :

Chaque contenu de ces zone de textes et listes déroulantes est ensuite extrait des ArrayList par un ArrayList.get(unIndex).

Ces données extraites sont ensuite insérées dans un tableau lors d'un clique sur le bouton Validation :

```
// Création d'un tableau récuperant le contenu des différents JTextField /
// stockés dans l'ArrayList ArrayListTextField /

for(k=0; k<NbJTextField; k++) {
TempTextField=Main.ArrayListTextField.get(k).getText();
TabGetTextField[CptTabTextField]=TempTextField;
CptTabTextField++; }
```

Ici, un exemple est donné avec *ArrayListTextField* : toutes les valeurs contenues dans les JTextField sont récupérées une par une grâce à la méthode getText puis stockées dans le tableau *TabGetTextField*.

Écriture du fichier d'observation :

Une fois celles-ci récupérées dans le tableau, il devient alors possible de les utiliser pour créer le fichier texte :

Écriture dans le flux de caractères formatés :

```
String FichierCible="c:\\A\\fichier_observation.txt";
File fichier = new File(FichierCible);


out = new PrintWriter(new FileWriter(fichier));
out.write("[FIELD_NAME] <");                // Ecrit [FIELD_NAME] < dans le fichier
```

Ici, il y a écriture dans le fichier cible (« c:\\A\\fichier_observation.txt ») des caractères :
« [FIELD_NAME] < »

Utilisation des valeurs contenues dans les différents tableaux :

On a ici créé une nouvelle classe afin de rendre plus simple l'accès aux données du tableau :

```
public class AjoutValeurJTextfield
{

    public AjoutValeurJTextfield()
    {
       // Ecrit dans le fichier le contenu de TabGetTextField[cptAjoutChampTextField], repésentant /
       // le contenu du JTextField n°cptAjoutChampTextField transformé en String /

       BoutonValider.out.write(BoutonValider.TabGetTextField[BoutonValider.CptAjoutChampTextField]);
       BoutonValider.CptAjoutChampTextField++;

    }
}
```

Ainsi, lorsque l'on fait appel à *AjoutValeurJTextField*, celle-ci écrit directement à la suite du fichier la valeur contenue dans la case numéro *CptAjoutChampTextField* du tableau *TabGetTextField*, puis incrémente la valeur du Cpt.

Autrement dit, cette classe permet à chaque fois d'ajouter le contenu de la case suivante du tableau *TabGetTextField*.

```
new AjoutValeurJTextfield();
out.write(">");
out.println();
```

Ici donc, *AjoutValeurJTextField* ajoutera dans le fichier le contenu de la case 0 du tableau, *CptAjoutChampTextField* étant initialisé à 0.
out.println() permet de faire un retour à la ligne dans le fichier texte.

Voici donc ce qu'on obtient dans fichier_observation.txt en écrivant « unNom » dans le premier JTextField et en validant :

```
[FIELD_NAME] <unNom>
```

Le même principe s'applique dans le cas de l'ajout d'une JComboBox : on fera appel à *AjoutValeurJComboBox*, qui permettra d'écrire à la suite du fichier, la valeur sélectionnée dans la Combobox stockée dans *TabGetComboBox* :

```
public class AjoutValeurJComboBox
{
    public AjoutValeurJComboBox()
    {
       // Ecrit dans le fichier le contenu de TabGetComboBox[cptAjoutChampComboBox], repésentant /
       // le contenu du  JComboBox n°cptAjoutChampComboBox transformé en String /

       BoutonValider.out.write(BoutonValider.TabGetComboBox[BoutonValider.CptAjoutChampComboBox]);
       BoutonValider.CptAjoutChampComboBox++;
    }
}
```

Après l'écriture des différents descripteurs, une boucle est exécuté eafin d'écrire les NbObservation créés :

```
for(k=0;k<NouvelleObservation.NbObservation;k++)
{
    out.write(" [OBS_NAME] <");
    new AjoutValeurJTextfield();
    out.write(">");
    BoutonValider.out.println();
```

...

Viens ensuite l'écriture des différentes actions :

```
        CptNumeroObservation++;
        for(l=0;l<NouvelleObservation.TabNbActionParObs[CptNumeroObservation-1];l++)
        {
            out.write("  [CAMS] <Acquisition ");
            out.write(Main.ArrayListTextFieldDansPanelActions.get(CptNuméroAction)
                    .getText());
            out.write(Main.ArrayListComboBoxDansPanelActions.get(CptNuméroAction)
                    .getSelectedItem().toString());
            out.println();
            CptNuméroAction++;
        }

    }
```

Ici, pour chaque observation on utilise le nombre d'actions qui y est associé : *TabNbActionParObs*[index] contenant le nombre d'actions pour chacune des observations (ainsi *TabNbActionParObs*[0] contient le nombre d'actions que présente la 1ère observation)

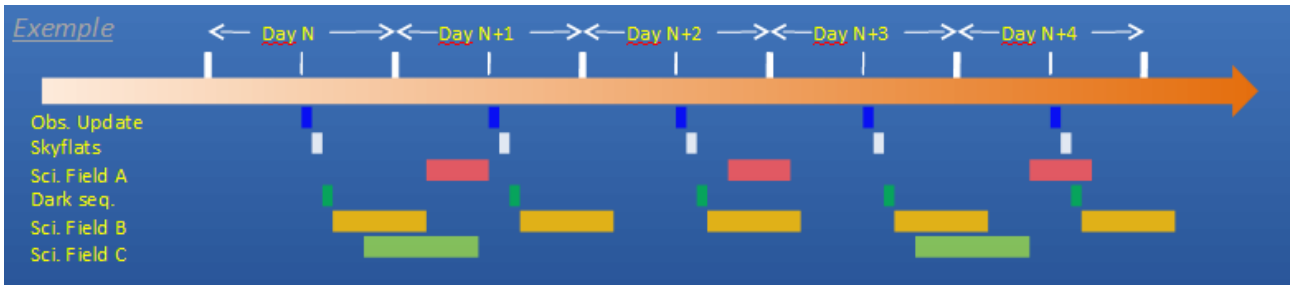
On écrit ainsi le contenu des n JTextField suivantes de l'ArrayList *ArrayListTextFieldDansPanelAction,* n étant le nombre de JTextField contenus par action * le nombre d'actions demandées pour cette observation, puis on incrémente le numéro de l'observation et donc de l'index du tableau *TabNbActionParObs.* Le même raisonnement s'applique pour l'écriture du contenu des JComboBox de chaque action, mais avec l'ArrayList *ArrayListComboBoxDansPanelAction.*

Conclusion :

L'exploitation des différentes données saisies ainsi que des interactions des utilisateurs fonctionne donc, il est en effet possible de se servir de l'interface graphique afin de pouvoir générer le fichier d'observation dans un format fixe et bien déterminé. L'utilisation des ArrayList assure, quant à elle, la simplicité d'ajout d'éléments dans le cadre d'une évolution futur.

Possibilités d'évolutions :

Des améliorations sont réalisables afin de rendre cet outil plus pratique, il est en effet possible de rajouter une représentation graphique temporelle des différentes observations afin de vérifier si aucune ne s'effectuent en même temps, cela pourrait se représenter sous cette forme par exemple :



Cela permettrait également d'établir des priorités mais également de mieux visualiser l'ensemble des programmations effectués.

Conclusion

Le développement de ce logiciel aura permis un gain de temps considérable dans la génération du fichier d'observation, et son développement pensé sur le long terme permettra à tout le monde de le modifier simplement.

Ce stage a été l'occasion d'approfondir mes connaissances en programmation, et ce principalement en JAVA, mais m'aura également permis de développer des méthodes de travail et d'organisation plus avancées, nécessaires au bon déroulement de la réalisation d'un projet sur plusieurs mois.

Je suis pour ma part très satisfait d'avoir pu travailler sur un projet utile et évolutif, qui m'aura permis de consolider mes connaissances dans un domaine que j'apprécie particulièrement.

Ce stage termine ainsi mes deux années de formation à l'IUT en m'accordant la chance de posséder à présent une expérience dans le monde de l'entreprise, qui me sera sans aucun doute, très utile au cours des années à venir.

Annexe 1 :

Voici en annexe, l'intégralité du programme :

ActionSelectionee

```
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.JComboBox;
import javax.swing.JTextField;


public class ActionSelectionee {

	static int NumActionInt=1;

	int i;
	int IndexPanelActionARecuperer=0;
	String NumActionString;

	// Code effectuée lorsque qu'une Action est séléctionnée /

	class ActionComboBoxListener implements ItemListener
	{
		public void itemStateChanged(ItemEvent e)
		{	int i;

				for(i=0;i<NouvelleObservation.NbJPanelAction-1;i++)
				{
					Main.ArrayListPanelAction.get(i)
					.setVisible(false);
				}

			// Récupération du numéro de l'action séléctionnée /

			NumActionString=Main.ArrayListComboBoxActions.get
			(ObservationSelectionee.NumObservationInt-
			1).getSelectedItem().toString().substring(6,7);

			NumActionInt = Integer.parseInt(NumActionString);

			// Calcul du numéro de l'index du panel séléctionné /

			for (i=0;i<ObservationSelectionee.NumObservationInt-1;i++)
			{		IndexPanelActionARecuperer=IndexPanelActionARecuperer
					+NouvelleObservation.TabNbActionParObs[i];
			}

			IndexPanelActionARecuperer=IndexPanelActionARecuperer+NumActionInt;

			Main.ArrayListPanelAction.get(IndexPanelActionARecuperer-
			1).setVisible(true);
			JTextField TextFieldtest = new JTextField();
			Main.Container.add(TextFieldtest);
```

```
            TextFieldtest.setBounds(0,25,150,20);

            TextFieldtest.setText(IndexPanelActionARecuperer-
            1+"!"+NumActionInt);


            IndexPanelActionARecuperer=0;

            }



      }
      public ActionSelectionee(JComboBox uneJComboBox)
      {
            uneJComboBox.addItemListener(new ActionComboBoxListener());
      }
}
```

Annexe 2 :

AjoutValeurJComboBox

```
public class AjoutValeurJComboBox
{
      public AjoutValeurJComboBox()
      {
         // Ecrit dans le fichier le contenu de
         // TabGetComboBox[cptAjoutChampComboBox], repésentant /
         // le contenu du  JComboBox n°cptAjoutChampComboBox transformé en
         // String /

         BoutonValider.out.write(
         BoutonValider.TabGetComboBox[BoutonValider.CptAjoutChampComboBox]);
         BoutonValider.CptAjoutChampComboBox++;
      }
}
```

Annexe 3 :

AjoutValeurJTextField

```
public class AjoutValeurJTextfield
{

      public AjoutValeurJTextfield()
      {
         // Ecrit dans le fichier le contenu de
         // TabGetTextField[cptAjoutChampTextField], repésentant /
         // le contenu du JTextField n°cptAjoutChampTextField transformé en
         // String /


BoutonValider.out.write(BoutonValider.TabGetTextField[BoutonValider.CptAjoutCham
pTextField]);
         BoutonValider.CptAjoutChampTextField++;

      }
}
```

Annexe 4 :

BoutonValider :

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JOptionPane;
import javax.swing.JButton;


public class BoutonValider
{
	static int NbJTextField=0;
	static int NbJComboBox=0;
	static int CptAjoutChampTextField=0;
	static int CptAjoutChampComboBox=0;
	static PrintWriter out  ;
	static String TabGetTextField[] = new String[500];
	static String TabGetTextFieldPanelActions[] = new String[500];
	static String TabGetComboBox[] = new String[500];
	static String TabGetComboBoxPanelActions[] = new String[500];
	static int CptAjoutChampTextFieldPanelActions=0;


	String TempTextField;
	String TempComboBox;
	String FichierCible="c:\\A\\fichier_observation.txt";

	int k;
	int l;

	int CptNumeroObservation=0;
	int CptNuméroAction=0;

	int CptTabTextField=0;
	int CptTabTextFieldPanelActions=0;
	int CptTabComboBox=0;
	int CptTabComboBoxPanelActions=0;

	File fichier = new File(FichierCible);


	class BoutonListener implements ActionListener
	{
		public void actionPerformed(ActionEvent e)
		{
			// Si pas de dates choisies, affiche une erreur /

			if((Main.DateChooserMax.getCalendar()==null)||
(Main.DateChooserMin.getCalendar()==null)||
(Main.DateChooserMax.getCalendar().before(Main.DateChooserMin.getCalendar())))
			{
```

```
                                    JOptionPane.showMessageDialog(Main.Container,
                                                "Dates choisies incorrectes !",
                                                "
Erreur",JOptionPane.ERROR_MESSAGE);

                        }

                        else
                        {

                    // Création d'un tableau récuperant le contenu /
                    // des différents JTextField stockés dans l'ArrayList /
                    // ArrayListTextField /

                    for(k=0; k<NbJTextField; k++) {
                    TempTextField=Main.ArrayListTextField.get(k).getText();
                    TabGetTextField[CptTabTextField]=TempTextField;
                    CptTabTextField++; }

                    // Création d'un tableau récuperant le contenu des /
                    // différents JCombo stockés dans l'ArrayList /
                    // ArrayListComboBox /

                    for(k=0; k<NbJComboBox; k++) {
                    TempComboBox=Main.ArrayListComboBox.get(k)
                   .getSelectedItem().toString();

                    TabGetComboBox[CptTabComboBox]=TempComboBox;
                    CptTabComboBox++; }

                    for(k=0; k<NouvelleAction.NbJTextFieldDansUnPanelAction;k++) {

                          TempTextField=Main.
                          ArrayListTextFieldDansPanelActions.get(k).getText();

                          TabGetTextFieldPanelActions[CptTabTextFieldPanelActions]
                          =TempTextField;

                          CptTabTextFieldPanelActions++; }

                    for(k=0; k<NouvelleAction.NbJComboBoxDansUnPanelAction;k++) {

                          TempComboBox=Main.ArrayListComboBoxDansPanelActions
                          .get(k).getSelectedItem().toString();

                          TabGetComboBoxPanelActions[CptTabComboBoxPanelActions]=
                          TempComboBox;

                          CptTabComboBoxPanelActions++; }

                    try
                    {

                        // Ecriture du fichier /

                        out = new PrintWriter(new FileWriter(fichier));

                        // Ecrit [FIELD_NAME] < dans le fichier

                        out.write("[FIELD_NAME] <");
```

```
// Ecrit la valeur String de la première JTextfield /
// contenu dans TabGetTextField[0] puis incrémente le cpt /
// de celui-ci  /

new AjoutValeurJTextfield();
out.write(">");
out.println();        //Retour à la ligne dans le fichier /
out.write("[FIELD_TYPE] <");

// Ecrit la valeur String de la première JComboBox /
// contenu dans TabGetComboBox[0] puis incrémente le cpt /
// de celui-ci /

new AjoutValeurJComboBox();
out.write(">");
out.println() ;

out.write("[FIELD_TIME_UTC_BOUNDARIES] <Boundary min: ");
new ConversionDate(Main.DateChooserMin);

out.write(ConversionDate.DateConvertie+" ");
new AjoutValeurJComboBox();
out.write("h");
new AjoutValeurJComboBox();
out.write("m");
new AjoutValeurJComboBox();
out.write("s");

out.write(", Boundary max: ");
new ConversionDate(Main.DateChooserMax);
out.write(ConversionDate.DateConvertie+" ");
new AjoutValeurJComboBox();
out.write("h");
new AjoutValeurJComboBox();
out.write("m");
new AjoutValeurJComboBox();
out.write(".000");
out.write("s");
out.write(">");
BoutonValider.out.println() ;

out.write("[MOUNT_ABS_POSITION] <ALT: ");
new AjoutValeurJTextfield();
out.write(".");

// Formatage de ALT /

int temp =
Integer.parseInt(TabGetTextField[CptAjoutChampTextField]);

if(temp<10)
      out.write(TabGetTextField[CptAjoutChampTextField]
      +"0000");

else if(temp<100)
      out.write(TabGetTextField[CptAjoutChampTextField]
      +"000");

else if(temp<1000)
      out.write(TabGetTextField[CptAjoutChampTextField]
```

```
      +"00");

else if(temp<10000)
      out.write(TabGetTextField[CptAjoutChampTextField]
      +"0");

else if(temp>=10000){
      TabGetTextField[CptAjoutChampTextField]=
      TabGetTextField[CptAjoutChampTextField]
      .substring(0, 5);
      out.write(TabGetTextField[CptAjoutChampTextField]);
}

CptAjoutChampTextField++;


out.write(", AZ: ");
new AjoutValeurJTextfield();
out.write(".");

// Formatage de Az //

temp =
Integer.parseInt(TabGetTextField[CptAjoutChampTextField]);

if(temp<10)
         out.write(TabGetTextField[CptAjoutChampTextField]
         +"0000");

   else if(temp<100)
         out.write(TabGetTextField[CptAjoutChampTextField]
         +"000");

   else if(temp<1000)
         out.write(TabGetTextField[CptAjoutChampTextField]
         +"00");

   else if(temp<10000)
         out.write(TabGetTextField[CptAjoutChampTextField]
         +"0");

   else if(temp>=10000){

          TabGetTextField[CptAjoutChampTextField]=
         TabGetTextField[CptAjoutChampTextField]
         .substring(0, 5);

        out.write(TabGetTextField[CptAjoutChampTextField]);
      }

CptAjoutChampTextField++;


out.write(">");
BoutonValider.out.println() ;

out.write("[MOUNT_TRACKING_TYPE] <");
new AjoutValeurJComboBox();
out.write(">");
BoutonValider.out.println();
```

```
                        out.write("[SCIENCE_STAR_CRD] <RA: ");
                        new AjoutValeurJComboBox();
                        out.write(" ");
                        new AjoutValeurJComboBox();
                        out.write(" ");
                        new AjoutValeurJComboBox();
                        out.write(", DEC: ");
                        new AjoutValeurJComboBox();
                        out.write(" ");
                        new AjoutValeurJComboBox();
                        out.write(" ");
                        new AjoutValeurJComboBox();
                        out.write(">");
                        BoutonValider.out.println();

                        out.write("[SCIENCE_STAR_INFO] <pmRA: ");
                        new AjoutValeurJTextfield();
                        out.write(", pmDE: ");
                        new AjoutValeurJTextfield();
                        out.write(", Parallax: ");
                        new AjoutValeurJTextfield();
                        out.write(", RadVel: ");
                        new AjoutValeurJTextfield();
                        out.write(">");
                        BoutonValider.out.println();

                        out.write("[GUIDE_STAR_CRD] <RA: ");
                        new AjoutValeurJTextfield();
                        out.write(">");
                        BoutonValider.out.println();

                        out.write("[GUIDE_STAR_INFO] <pmRA: ");
                        new AjoutValeurJTextfield();
                        out.write(", pmDE: ");
                        new AjoutValeurJTextfield();
                        out.write(", Parallax: ");
                        new AjoutValeurJTextfield();
                        out.write(", RadVel: ");
                        new AjoutValeurJTextfield();
                        out.write(">");
                        BoutonValider.out.println();

                        out.write("[FIELD_RECOGNITION] <");

                        // Ecrit ExpTime_ms dans le fichier, seulement /
                        // si "YES" séléctionné /

                        if
((BoutonValider.TabGetComboBox[BoutonValider.CptAjoutChampComboBox])
                                        .equals("YES"))
                        {
                              new AjoutValeurJComboBox();
                              out.write(", ExpTime_ms=");
                              new AjoutValeurJTextfield();
                              out.write(">");
                              BoutonValider.out.println();
                        }

                        else
                        {
                              new AjoutValeurJComboBox();
```

```
                        out.write(">");
                        BoutonValider.out.println();

      CreerJTextFieldDansContainer.NbJTextFieldDescripteurs--;

                  }

                  out.write("[STORE_ROI] <");

                  // Ecrit les champs Xc... Data Folder dans le /
                  // fichier, seulement si /
                  //"YES" séléctionné /

                  if
((BoutonValider.TabGetComboBox[BoutonValider.CptAjoutChampComboBox])
                        .equals("YES"))
                  {
                        new AjoutValeurJComboBox();
                        out.write(", xc=");
                        new AjoutValeurJTextfield();
                        out.write(", yc=");
                        new AjoutValeurJTextfield();
                        out.write(", dx=");
                        new AjoutValeurJTextfield();
                        out.write(", dy=");
                        new AjoutValeurJTextfield();
                        out.write(", DataFolder=\"");
                        new AjoutValeurJTextfield();
                        out.write("\">");
                        BoutonValider.out.println();
                  }

                  else
                  {
                        new AjoutValeurJComboBox();
                        out.write(">");
                        BoutonValider.out.println();

CreerJTextFieldDansContainer.NbJTextFieldDescripteurs=
CreerJTextFieldDansContainer.NbJTextFieldDescripteurs-5;
                  }



                  // Ecrit ExpTime_ms dans le fichier, seulement /
                  // si "YES" séléctionné /

                  for(k=0;k<NouvelleObservation.NbObservation;k++)
                  {
                        out.write(" [OBS_NAME] <");
                        new AjoutValeurJTextfield();
                        out.write(">");
                        BoutonValider.out.println();

                        out.write(" [TRACK_STAR] <");
                        new AjoutValeurJComboBox();
                        BoutonValider.out.write(", Exp_ms=");
                        new AjoutValeurJTextfield();
                        BoutonValider.out.write(", DetectionThreshold=");
                        new AjoutValeurJTextfield();
```

```
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();

                        out.write(" [OBS_SERVOGUIDINGSETPOINT] <");
                        new AjoutValeurJComboBox();
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();

                        out.write(" [OBS_SERVOFOCUS] <");
                        new AjoutValeurJComboBox();
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();

                        out.write(" [OBS_COND_SUNHEIGHT] <Sun height min
(deg): ");
                        new AjoutValeurJTextfield();
                        out.write(", Sun height max (deg): ");
                        new AjoutValeurJTextfield();
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();


                        out.write(" [OBS_COND_UTC] <UTC min: ");
                        new AjoutValeurJTextfield();
                        out.write(",UTC max: ");
                        new AjoutValeurJTextfield();
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();

                        out.write(" [OBS_COND_PERIOD] <Period:");
                        new AjoutValeurJTextfield();
                        out.write(", Duration:");
                        new AjoutValeurJTextfield();
                        BoutonValider.out.write(">");
                        BoutonValider.out.println();
                        CptNumeroObservation++;

for(l=0;l<NouvelleObservation.TabNbActionParObs[CptNumeroObservation-1];l++)
                        {
                              out.write("  [CAMS] <Acquisition ");

out.write(Main.ArrayListTextFieldDansPanelActions.get(CptNuméroAction)
                                                .getText());

out.write(Main.ArrayListComboBoxDansPanelActions.get(CptNuméroAction)
                                                .getSelectedItem().toString());
                              out.println();
                              CptNuméroAction++;
                        }

                  }




                  CptTabTextFieldPanelActions=0;
                  CptAjoutChampTextFieldPanelActions=0;
                  CptAjoutChampTextField=0;
                  CptAjoutChampComboBox=0;
```

```
                    CptTabTextField=0;
                    CptTabComboBox=0;
                    CptNuméroAction=0;

                    out.close();   //Ferme le flux du fichier, sauvegardant
                                   ainsi les données.


                    JOptionPane.showMessageDialog(Main.Container, "Fichier
d'observation crée avec succès dans : "+FichierCible," Fichier crée
",JOptionPane.PLAIN_MESSAGE);


                }

                  catch(IOException e1)
                  {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                  }
            }}
      }

      public BoutonValider(JButton unBoutonValider)
      {
         unBoutonValider.addActionListener(new BoutonListener());


      }
                                        }
```

Annexe 5 :

ConversionDate :

```
import java.util.Calendar;
import com.toedter.calendar.JDateChooser;


public class ConversionDate
{

   static Calendar calMyDate;
   static JDateChooser uneDate;
   static String DateConvertie;

   // Convertie une date choisie avec le calendrier JDateChooser en date /
   // au format String si sous la forme AAAA.MM.JJ /

   public ConversionDate(JDateChooser uneDate)
   {
         calMyDate=uneDate.getCalendar();
         int FormatMois;
         int FormatJour;

         FormatMois=calMyDate.get(Calendar.MONTH)+1;
         FormatJour=calMyDate.get(Calendar.DATE);



         String Jour=String.valueOf((calMyDate.get(Calendar.DATE)));
       String Mois=String.valueOf((calMyDate.get(Calendar.MONTH))+1);
       String Annee=String.valueOf((calMyDate.get(Calendar.YEAR)));

       // Si numéro mois séléctionné <10 : ex: Septembre (mois 9) => converti
       // en 09 /

       if(FormatMois<10)
         {Mois="0"+FormatMois;}
       if(FormatJour<10)
         {Jour="0"+FormatJour;}

       DateConvertie = Annee+"."+Mois+"."+Jour;
   }
}
```

Annexe 6 :

CreerJComboBoxDansContainer :

```
import java.util.Vector;
import javax.swing.JComboBox;
import javax.swing.JLabel;


public class CreerJComboBoxDansContainer
{
	static JLabel Label;
	static JComboBox ComboBox;
	int x;
	int y;
	int TaillexLabel;
	int TaillexComboBox;
	int i;
	static int NbJComboBoxDescripteurs;
	Vector Heure = new Vector();
	Vector Minute = new Vector();
	Vector Seconde = new Vector();



	// Crée 3 JComboBox Heure, Minute et Seconde avec taille fixe positionné /
	// en (x,y)/

	public CreerJComboBoxDansContainer(int unx, int uny)
	{
    x=unx;
	y=uny;

	for (i=0;i<24;i++){
		if (i<10)
		{
			Heure.addElement("0"+i);
		}
		else
		{
			Heure.addElement(i);
		}
	}

	for (i=0;i<60;i++){
		if (i<10)
		{
			Minute.addElement("0"+i);
		}
		else
		{
			Minute.addElement(i);
		}
	}
```

```
      for (i=0;i<60;i++){
            if (i<10)
            {
                  Seconde.addElement("0"+i);
            }
            else
            {
                  Seconde.addElement(i);
            }

      }



    ComboBox= new JComboBox(Heure);
    Main.ArrayListComboBox.add(ComboBox);
    Main.Container.add(ComboBox);
    ComboBox.setBounds(x,y,75,20);

    ComboBox= new JComboBox(Minute);
    Main.ArrayListComboBox.add(ComboBox);
    Main.Container.add(ComboBox);
    ComboBox.setBounds(x+75,y,75,20);

    ComboBox= new JComboBox(Seconde);
    Main.ArrayListComboBox.add(ComboBox);
    Main.Container.add(ComboBox);
    ComboBox.setBounds(x+150,y,75,20);

      // Chaque JComboBox crée est ajouté à l'ArrayList ArrayListComboBox et /
      // incrèmente le NbJComboBox qui sera utilisé pour définir le /
      // nombre d'objets présents dans l'ArrayList /

    BoutonValider.NbJComboBox=BoutonValider.NbJComboBox+3;
      }


      // Crée 3 JComboBox Heure, Minute et Seconde accompagné d'un Label
      // avec une taille X paramètrable /

      public CreerJComboBoxDansContainer(String unLabel,int unx, int uny, int
uneTaillexLabel)
      {
          x=unx;
            y=uny;
            TaillexLabel=uneTaillexLabel;

            for (i=0;i<24;i++){
            Heure.addElement(i);}

            for (i=0;i<60;i++){
            Minute.addElement(i);}

            for (i=0;i<60;i++){
            Seconde.addElement(i);}

            Label=new JLabel(unLabel);
          Main.Container.add(Label);
            Label.setBounds(x,y,TaillexLabel,20);

          ComboBox= new JComboBox(Heure);
```

```
            Main.ArrayListComboBox.add(ComboBox);
            Main.Container.add(ComboBox);
            ComboBox.setBounds(x+TaillexLabel+10,y,75,20);

            ComboBox= new JComboBox(Minute);
            Main.ArrayListComboBox.add(ComboBox);
            Main.Container.add(ComboBox);
            ComboBox.setBounds(x+75+TaillexLabel+10,y,75,20);

            ComboBox= new JComboBox(Seconde);
            Main.ArrayListComboBox.add(ComboBox);
            Main.Container.add(ComboBox);
            ComboBox.setBounds(x+150+TaillexLabel+10,y,75,20);

              // Chaque JComboBox crée est ajouté à l'ArrayList ArrayListComboBox
              // et incrèmente le NbJComboBox qui sera utilisé pour définir le /
              // nombre d'objets présents dans l'ArrayList /

            BoutonValider.NbJComboBox=BoutonValider.NbJComboBox+3;
            NbJComboBoxDescripteurs++;
      }


      // Crée une JComboBox et un Label entièrement paramètrables

      public CreerJComboBoxDansContainer(String unLabel, int unx, int uny, int
uneTaillexLabel,int uneTaillexComboBox, Vector unChoix)
      {
            Label=new JLabel(unLabel);
            ComboBox= new JComboBox(unChoix);

            x=unx;
           y=uny;

            TaillexLabel=uneTaillexLabel;
            TaillexComboBox=uneTaillexComboBox;
            Main.Container.add(Label);
            Label.setBounds(x,y,TaillexLabel,20);

            Main.Container.add(ComboBox);
            ComboBox.setBounds(x+TaillexLabel+10,y,TaillexComboBox,20);

              // Chaque JComboBox crée est ajouté à l'ArrayList ArrayListComboBox
              // et incrèmente le NbJComboBox qui sera utilisé pour définir le
              // nombre d'objets présents dans l'ArrayList /

            Main.ArrayListComboBox.add(ComboBox);
            BoutonValider.NbJComboBox=BoutonValider.NbJComboBox+1;
            NbJComboBoxDescripteurs++;
      }

}
```

Annexe 7 :

CreerJComboBoxDansPanelAction :

```
import java.util.Vector;
import javax.swing.JComboBox;
import javax.swing.JLabel;


public class CreerJComboBoxDansPanelAction {
      static JLabel Label;
      static JComboBox ComboBox;
      int x;
      int y;
      int TaillexLabel;
      int TaillexComboBox;
      int i;

      public CreerJComboBoxDansPanelAction (String unLabel, int unx, int uny,
int uneTaillexLabel,int uneTaillexComboBox, Vector unChoix)
      {

      Label=new JLabel(unLabel);
      ComboBox= new JComboBox(unChoix);

      x=unx;
    y=uny;

    TaillexLabel=uneTaillexLabel;
    TaillexComboBox=uneTaillexComboBox;
    NouvelleAction.PanelAction.add(Label);
    Label.setBounds(x,y,TaillexLabel,20);

    NouvelleAction.PanelAction.add(ComboBox);
    ComboBox.setBounds(x+TaillexLabel+10,y,TaillexComboBox,20);

      // Chaque JComboBox crée est ajouté à l'ArrayList ArrayListComboBox et /
      // incrèmente le NbJComboBox qui sera utilisé pour définir le /
      // nombre d'objets présents dans l'ArrayList /


      if(NouvelleObservation.PremièreActionDeObservationCree==0)


Main.ArrayListComboBoxDansPanelActions.add(NouvelleAction.PositionActionaRajoute
r,ComboBox);
      else
            Main.ArrayListComboBoxDansPanelActions.add(ComboBox);


    NouvelleAction.NbJComboBoxDansUnPanelAction++;


      }
}
```

Annexe 8 :

CreerJComboBoxDansPanelObservation :

```
import javax.swing.JComboBox;
import javax.swing.JLabel;


public class CreerJComboBoxDansPanelObservation {

	static JLabel Label;
	static JComboBox ComboBoxYesNo;
	int x;
	int y;
	int TaillexLabel;
	int TaillexComboBox;
	int i;
	static int NbJComboBoxDansPanel=0;

	// Crée une JComboBox YES/NO
	public CreerJComboBoxDansPanelObservation (String unLabel, int unx, int
uny, int uneTaillexLabel)
	{
		ComboBoxYesNo = new JComboBox(Main.VYES_NO);

		Label=new JLabel(unLabel);

		x=unx;
	    y=uny;
	    TaillexLabel=uneTaillexLabel;

	    NouvelleObservation.PanelObservation.add(Label);
	    Label.setBounds(x,y,TaillexLabel,20);

	    NouvelleObservation.PanelObservation.add(ComboBoxYesNo);
	    ComboBoxYesNo.setBounds(x+TaillexLabel+10,y,100,20);

	    Main.ArrayListComboBox.add(ComboBoxYesNo);
	    BoutonValider.NbJComboBox=BoutonValider.NbJComboBox+1;
	    NbJComboBoxDansPanel++;

	}

}
```

Annexe 9 :

CreerJTextFieldDansContainer :

```
import java.text.NumberFormat;
import javax.swing.JComboBox;
import javax.swing.JFormattedTextField;
import javax.swing.JLabel;
import javax.swing.text.MaskFormatter;


public class CreerJTextFieldDansContainer
{
	static JLabel Label;
	static JComboBox ComboBox;
	static JFormattedTextField Textfield;

	int x;
	int y;
	int TaillexLabel;
	int TaillexTextField;
	int TaillexComboBox;
	static int NbJTextFieldDescripteurs=0;
	static MaskFormatter mask;


	public CreerJTextFieldDansContainer(String unLabel, int unx, int uny
			, int uneTaillexLabel, int uneTaillexTextField)
	{
		Label=new JLabel(unLabel);
		Textfield= new JFormattedTextField();

		Main.Container.add(Label);
		Label.setBounds(unx,uny,uneTaillexLabel,20);

		Main.Container.add(Textfield);

	Textfield.setBounds(unx+uneTaillexLabel+10,uny,uneTaillexTextField,20);

		// Chaque JTextField crée est ajouté à l'ArrayList /
		// ArrayListTextField et incrèmente le NbJTextField qui sera /
		// utilisé pour définir le nombre d'objets présents /
		// dans l'ArrayList /

		Main.ArrayListTextField.add(Textfield);
		BoutonValider.NbJTextField++;
		NbJTextFieldDescripteurs++;


	}



	// Crée un JLabel ainsi qu'un JTextField dans Container paramètrable /
	// à une position(x,y) /
```

```
	public CreerJTextFieldDansContainer(String unLabel, int unx, int uny, int
uneTaillexLabel,int uneTaillexTextField,NumberFormat unNombreFormat)
	{
		Label=new JLabel(unLabel);
		Textfield= new JFormattedTextField(unNombreFormat);

		x=unx;
		y=uny;
		TaillexLabel=uneTaillexLabel;
		TaillexTextField=uneTaillexTextField;
		Main.Container.add(Label);
		Label.setBounds(x,y,TaillexLabel,20);

		Main.Container.add(Textfield);
		Textfield.setBounds(x+TaillexLabel+10,y,TaillexTextField,20);

		// Chaque JTextField crée est ajouté à l'ArrayList /
		// ArrayListTextField et incrèmente le NbJTextField qui sera /
		// utilisé pour définir le nombre d'objets présents /
		// dans l'ArrayList /

		Main.ArrayListTextField.add(Textfield);
		BoutonValider.NbJTextField=BoutonValider.NbJTextField+1;
		NbJTextFieldDescripteurs++;


	}

}
```

Annexe 10 :

CreerJTextFieldDansPanelAction :

```
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JTextField;


public class CreerJTextFieldDansPanelAction {

	static JLabel Label;
	static JTextField Textfield;
	static JComboBox ComboBox;

	int x;
	int y;
	int TaillexLabel;
	int TaillexTextField;
	int TaillexComboBox;

	// Crée un JLabel ainsi qu'un JTextField dans Container paramètrable à une
	// position(x,y)

	public CreerJTextFieldDansPanelAction(String unLabel, int unx, int uny,
int uneTaillexLabel,int uneTaillexTextField )
	{
		Label=new JLabel(unLabel);
		Textfield= new JTextField(30);

		x=unx;
		y=uny;
		TaillexLabel=uneTaillexLabel;
		TaillexTextField=uneTaillexTextField;

		NouvelleAction.PanelAction.add(Label);
		Label.setBounds(x,y,TaillexLabel,20);

		NouvelleAction.PanelAction.add(Textfield);
		Textfield.setBounds(x+TaillexLabel+10,y,TaillexTextField,20);

		// Chaque JTextField crée est ajouté à l'ArrayList /
		// ArrayListTextField et incrèmente le NbJTextField qui /
		// sera utilisé pour définir le nombre d'objets présents /
		// dans l'ArrayList /


		if(NouvelleObservation.PremièreActionDeObservationCree==0)
			Main.ArrayListTextFieldDansPanelActions.
			add(NouvelleAction.PositionActionaRajouter,Textfield);
		else
			Main.ArrayListTextFieldDansPanelActions.add(Textfield);
			NouvelleAction.NbJTextFieldDansUnPanelAction++;
	}
}
```

Annexe 11 :

CreerJTextFieldDansPanelObservation :

```
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JTextField;


public class CreerJTextFieldDansPanelObservation {

            static JLabel Label;
            static JTextField Textfield;
            static JComboBox ComboBox;

            int x;
            int y;
            int TaillexLabel;
            int TaillexTextField;
            int TaillexComboBox;
            static int NbTotalJTexFieldDansPanel=0;


            // Crée un JLabel ainsi qu'un JTextField dans Container /
            // paramètrable à une position(x,y) /

            public CreerJTextFieldDansPanelObservation(String unLabel, int unx,
int uny, int uneTaillexLabel,int uneTaillexTextField )
            {
                  Label=new JLabel(unLabel);
                  Textfield= new JTextField(30);

                  x=unx;
                  y=uny;
                  TaillexLabel=uneTaillexLabel;
                  TaillexTextField=uneTaillexTextField;

                  NouvelleObservation.PanelObservation.add(Label);
                  Label.setBounds(x,y,TaillexLabel,20);

                  NouvelleObservation.PanelObservation.add(Textfield);
                  Textfield.setBounds(x+TaillexLabel+10,y,TaillexTextField,20);

                  // Chaque JTextField crée est ajouté à l'ArrayList /
                  // ArrayListTextField et incrèmente /
                  // le NbJTextField qui sera utilisé pour définir le nombre /
                  // d'objets présents dans l'ArrayList /

                  Main.ArrayListTextField.add(Textfield);
                  BoutonValider.NbJTextField++;
                  NbTotalJTexFieldDansPanel++;
            }
      }
```

Annexe 12 :

Main :

```
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Vector;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import com.toedter.calendar.JDateChooser;



public class Main
{

   static ArrayList<JTextField> ArrayListTextField;
   static ArrayList<JComboBox> ArrayListComboBox;
   static ArrayList<JComboBox> ArrayListComboBoxActions;
   static ArrayList<JPanel> ArrayListPanelObservation;
   static ArrayList<JPanel> ArrayListPanelAction;

   static ArrayList<JTextField> ArrayListTextFieldDansPanelActions;
   static ArrayList<JComboBox> ArrayListComboBoxDansPanelActions;

   static JFrame Frame1 = new JFrame();
   static JPanel Container = new JPanel();
   static JButton Valider = new JButton("Valider");
   static JButton Observation = new JButton("Nouvelle Observation");
   static JButton SupprimerObservation = new JButton("Supprimer Observation");
   static JButton Debug = new JButton("Debug");
   static JDateChooser DateChooserMin = new JDateChooser();
   static JDateChooser DateChooserMax = new JDateChooser();
   static JLabel LabelDateMin= new JLabel("Boundary min :");
   static JLabel LabelDateMax= new JLabel("Boundary max :");
   static Vector VYES_NO;


   public static void main(String[] args)
   {
         // Création des ArrayList utilisés pour stocker les JTextField, /
         // JComboBox et JPanel/

         ArrayListTextField = new ArrayList<JTextField>();
         ArrayListTextFieldDansPanelActions = new ArrayList<JTextField>();
         ArrayListComboBox = new ArrayList<JComboBox>();
         ArrayListPanelObservation = new ArrayList<JPanel>();
         ArrayListPanelAction = new ArrayList<JPanel>();
         ArrayListComboBoxActions = new ArrayList<JComboBox>();
         ArrayListComboBoxDansPanelActions = new ArrayList<JComboBox>();

         // Création des Vectors utilisés pour remplir les JComboBox /
```

```
Vector VFIELD_TYPE = new Vector(10);
VFIELD_TYPE.addElement("TRANSIT_SRCH");
VFIELD_TYPE.addElement("TRANSIT_OBJ");
VFIELD_TYPE.addElement("MICROLENSING");
VFIELD_TYPE.addElement("CALIBRATION");

Vector VMOUNT_TRACKING_TYPE = new Vector(10);
VMOUNT_TRACKING_TYPE.addElement("SIDERAL");
VMOUNT_TRACKING_TYPE.addElement("MOON");
VMOUNT_TRACKING_TYPE.addElement("SUN");
VMOUNT_TRACKING_TYPE.addElement("STOP");

VYES_NO = new Vector(10);
VYES_NO.addElement("YES");
VYES_NO.addElement("NO");

// Paramètrage de la Frame /

Frame1.setResizable(false);
Frame1.setTitle("GUI java");
Frame1.setSize(1250, 900);
Frame1.setLocationRelativeTo(null);
Frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Container.setLayout(null);



// Création des JLabel ainsi que des JTextfield utilisés pour /
// receuillir les données /

new CreerJTextFieldDansContainer("FIELD_NAME :",40,40,150,335);
NumberFormat nombreFormat = NumberFormat.getNumberInstance();
new CreerJTextFieldDansContainer("MOUNT_ABS_POSITION
ALT :",40,160,190,45,nombreFormat);
new CreerJTextFieldDansContainer(".",295,160,10,45);
new CreerJTextFieldDansContainer("AZ :",380,160,25,45);
new CreerJTextFieldDansContainer(".",470,160,10,45);
new CreerJTextFieldDansContainer("SCIENCE_STAR_INFO
pmRA :",40,280,200,100);
new CreerJTextFieldDansContainer("pmDE :",385,280,40,100);
new CreerJTextFieldDansContainer("Parallax :",185,310,55,100);
new CreerJTextFieldDansContainer(" RadVel :",375,310,50,100);
new CreerJTextFieldDansContainer("GUIDE_STAR_CRD :",40,340,150,335);
new CreerJTextFieldDansContainer("GUIDE_STAR_INF
pmRA:",40,370,200,100);
new CreerJTextFieldDansContainer("pmDE :",385,370,40,100);
new CreerJTextFieldDansContainer("Parallax :",185,400,55,100);
new CreerJTextFieldDansContainer(" RadVel :",375,400,50,100);
new CreerJTextFieldDansContainer("ExpTime_ms :",345,430,80,100);
new CreerJTextFieldDansContainer("          xc :",375,460,50,100);
new CreerJTextFieldDansContainer("yc :",220,490,20,100);
new CreerJTextFieldDansContainer("dx :",405,490,20,100);
new CreerJTextFieldDansContainer("dy :",220,520,20,100);
new CreerJTextFieldDansContainer(" Folder :",380,520,45,100);
```

```
        // Création des JComboBox ainsi que des JTextfield utilisés pour /
        // receuillir les données /

        new CreerJComboBoxDansContainer("FIELD_TYPE :",40,70,150
        ,150,VFIELD_TYPE);
        new CreerJComboBoxDansContainer(310,100);
        new CreerJComboBoxDansContainer(310,130);
        new CreerJComboBoxDansContainer("MOUNT_TRACKING_TYPE :",40,190,150,
        150,VMOUNT_TRACKING_TYPE);
        new CreerJComboBoxDansContainer("SCIENCE_STAR_CRD
        coordRA : ",40,220,260);
        new CreerJComboBoxDansContainer(" coordDE : ",215,250,85);
        new CreerJComboBoxDansContainer("FIELD_RECOGNITION :",40,430,150
        ,100,VYES_NO);
        new CreerJComboBoxDansContainer("STORE_ROI :",40,460,150,100,VYES_NO);



        // Ajout du calendrier et des JComboxBox destinés aux horaires /

        Container.add(DateChooserMin);
        Container.add(LabelDateMin);
        LabelDateMin.setBounds(40,100,150,20);
        DateChooserMin.setBounds(200,100,100,20);

        Container.add(DateChooserMax);
        Container.add(LabelDateMax);
        LabelDateMax.setBounds(40,130,150,20);
        DateChooserMax.setBounds(200,130,100,20);

        // Ajout du bouton "Valider" /

        Container.add(Valider);
        Valider.setBounds(150,800,150,25);

        Container.add(Debug);
        Debug.setBounds(450,800,150,25);

        // Ajout du bouton "Créer Observation" /

        Container.add(Observation);
        Observation.setBounds(850,40,175,25);

        // Ajout du bouton "Supprimer Observation" /

        Container.add(SupprimerObservation);
        SupprimerObservation.setBounds(1050,40,175,25);


        //Lancement des procédures d'écoute /

        new BoutonValider(Valider);
        new NouvelleObservation(Observation);
        new SupprimerObservation(SupprimerObservation);
        new Debug(Debug);

        Frame1.setContentPane(Container);
        Frame1.setVisible(true);
}
```

```
}
```

Annexe 13 :

NouvelleAction :

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JPanel;
import javax.swing.JTextField;


public class NouvelleAction {

static JTextField TextFieldAction;
static JComboBox ComboBoxAct;
static Vector VNumeroAction;
static int i=1;
static JPanel PanelAction;
static int PositionActionaRajouter;
static int NbJTextFieldDansUnPanelAction=0;
static int NbJComboBoxDansUnPanelAction=0;


      class ActionButtonListener implements ActionListener
      {
            // Action effectuée lorsque l'on clique sur le bouton /
            // Nouvelle Action /

            public void actionPerformed(ActionEvent a)
            {
                  NbJTextFieldDansUnPanelAction=0;
                  NbJComboBoxDansUnPanelAction=0;

                  PanelAction = new JPanel();

                  // Récupération de l'observation actuellement séléctionnée /


Main.ArrayListPanelObservation.get(ObservationSelectionee.NumObservationInt-1)
                  .add(PanelAction);

                  PanelAction.setLayout(null);
                  PanelAction.setBounds(0,400,650,250);

                  PositionActionaRajouter=0;

      PositionActionaRajouter=NouvelleObservation.TabNbActionParObs[0];

                  // Calcul de l'index de ArrayListPanelAction auquel rajouter
                  // ce nouveau panel /
```

```
                if (ObservationSelectionee.NumObservationInt!=1)
                      for (i=1;i<=ObservationSelectionee.NumObservationInt-
                           1;i++)

                            PositionActionaRajouter=PositionActionaRajouter +
                            NouvelleObservation.TabNbActionParObs[i];


                new CreerJTextFieldDansPanelAction("Temps :",0,25,150,100);
                new CreerJComboBoxDansPanelAction("Type :",0,55,150,
                100,NouvelleObservation.VTypeAcquisition);

                Main.ArrayListPanelAction.add(PositionActionaRajouter,
                PanelAction) ;

                NouvelleObservation.NbJPanelAction++;

                int j;

                i=Main.ArrayListComboBoxActions.get
                (ObservationSelectionee.NumObservationInt-1).getItemCount()+1;

                VNumeroAction= new Vector();

                for(j=1;j<=i;j++)
                VNumeroAction.addElement("Action"+j);

                // Incrémentation du nombre d'action pour cette observation /

                NouvelleObservation.TabNbActionParObs[
                ObservationSelectionee.NumObservationInt-1]++;

                Main.ArrayListComboBoxActions.get(ObservationSelectionee.
                NumObservationInt-1).setModel(new DefaultComboBoxModel
                (VNumeroAction));


          }
    }


    public NouvelleAction(JButton unBoutonAction)
    {
          unBoutonAction.addActionListener(new ActionButtonListener());
    }

}
```

Annexe 14 :

NouvelleObservation :

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;

import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JPanel;

public class NouvelleObservation
{
	static JComboBox ComboBoxObs;
	static JComboBox ComboBoxAct;
	static JButton BoutonAction;
	static Vector VObservation;
	static Vector VAction;
	static Vector VTypeAcquisition;
	static int i=1;
	static JPanel PanelObservation;
	static JPanel PanelAction;
	static String TabGetPanelObservation[] = new String[50];
	static int TabNbActionParObs[] = new int[100];
	static int NbJPanelObservation=0;
	static int NbJPanelAction=0;
	static int NbObservation=0;
	static int NbJTextFieldDansUnPanelObservation=0;
	static int NbJComboBoxDansUnPanelObservation=0;
	static int declencheur=0;
	static int PremièreActionDeObservationCree =0;



	class ObservationButtonListener implements ActionListener
	{

		public void actionPerformed(ActionEvent e)
		{	NbJTextFieldDansUnPanelObservation=0;
			NbJComboBoxDansUnPanelObservation=0;

			VAction = new Vector();
			VAction.addElement("Action1");

			int k;

			// Initialise le nombre d'actions par observation à 1 pour /
			// toutes les observations /

			if (declencheur==0)
			{
			for (k=0;k<100;k++)
				TabNbActionParObs[k]=1;
			}
```

```
declencheur++;

// declencheur permet à cet opération de ne se dérouler
// qu'une seule fois /

VTypeAcquisition = new Vector();
VTypeAcquisition.addElement("Science");
VTypeAcquisition.addElement("Dark");
VTypeAcquisition.addElement("Bias");


// Si la ComboBoxObs n'existe pas, autrement dit, si aucune /
// observation crée /

if (ComboBoxObs==null)
{
      VObservation = new Vector();
      VObservation.addElement("Observation"+i);

      PanelObservation = new JPanel();
      BoutonAction= new JButton("Nouvelle Action");
      ComboBoxObs = new JComboBox(VObservation);

      // Ajout des différents composants graphiques d'une /
      // observation /

      new CreerJTextFieldDansPanelObservation("OBS_NAME",
      0,0,200,350);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation("Exp_ms",
      325,30,125,100);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation(
      "DetectionThreshold",325,60,125,100);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation(
      "OBS_COND_SUNHEIGHT Sun height min",0,150,450,100);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation("Sun height
      max",325,180,125,100);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation("OBS_COND_UTC
      UTC min",0,210,450,100);

      NbJTextFieldDansUnPanelObservation++;

      new CreerJTextFieldDansPanelObservation("UTC
      max",325,240,125,100);
```

```
NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("OBS_COND_PERIOD
Period:",0,270,450,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("Duration",
325,300,125,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJComboBoxDansPanelObservation("TRACK_STAR",
0,30,200);

NbJComboBoxDansUnPanelObservation++;

new CreerJComboBoxDansPanelObservation(
"OBS_SERVOGUIDINGSETPOINT",0,90,200);

NbJComboBoxDansUnPanelObservation++;

new CreerJComboBoxDansPanelObservation("OBS_SERVOFOCUS",
0,120,200);

NbJComboBoxDansUnPanelObservation++;


Main.Container.add(ComboBoxObs);
Main.Container.add(PanelObservation);

PanelObservation.setLayout(null);
PanelObservation.setBounds(600,100,650,700);

PanelObservation.add(BoutonAction);
BoutonAction.setBounds(0,350,150,25);


Main.ArrayListPanelObservation.add(PanelObservation);

NbJPanelObservation++;
NbObservation++;
i++;

ComboBoxObs.setBounds(600,40,200,20);
ComboBoxObs.setModel(new
DefaultComboBoxModel(VObservation));

// Création de la 1ère action pour l'observation venant
// d'être crée /

NouvelleAction.PanelAction = new JPanel();
PanelObservation.add(NouvelleAction.PanelAction);
NouvelleAction.PanelAction.setLayout(null);
NouvelleAction.PanelAction.setBounds(0,400,650,250);

PremièreActionDeObservationCree=1;

new CreerJTextFieldDansPanelAction("Temps :"
,0,25,150,100);
```

```
            new CreerJComboBoxDansPanelAction
            ("Type :",0,55,150,100,VTypeAcquisition);

            PremièreActionDeObservationCree=0;

            NouvelleAction.NbJTextFieldDansUnPanelAction++;
            NouvelleAction.NbJComboBoxDansUnPanelAction++;

            ComboBoxAct = new JComboBox(VAction);
            NouvelleObservation.PanelObservation.add(ComboBoxAct);
            ComboBoxAct.setBounds(250,350,150,25);
            Main.ArrayListComboBoxActions.add(ComboBoxAct);

            Main.ArrayListPanelAction.add
            (NouvelleAction.PanelAction);

            NbJPanelAction++;

            // Ecoute des différents boutons /

            new ObservationSelectionee(ComboBoxObs);
            new ActionSelectionee(ComboBoxAct);
            new NouvelleAction(BoutonAction);

      }

      // Si au moins une observation a été crée, pas de création /
      // d'une nouvelle liste déroulante/

      else
      {
            VObservation.addElement("Observation"+i);

            PanelObservation = new JPanel();
            Main.Container.add(PanelObservation);
            BoutonAction= new JButton("Nouvelle Action");

            PanelObservation.setLayout(null);
            PanelObservation.setBounds(600,100,650,700);

            new CreerJTextFieldDansPanelObservation("OBS_NAME",
            0,0,200,350);

            NbJTextFieldDansUnPanelObservation++;

            new CreerJTextFieldDansPanelObservation("Exp_ms",
            325,30,125,100);

            NbJTextFieldDansUnPanelObservation++;

            new CreerJTextFieldDansPanelObservation(
            "DetectionThreshold",325,60,125,100);

            NbJTextFieldDansUnPanelObservation++;

            new CreerJTextFieldDansPanelObservation(
            "OBS_COND_SUNHEIGHT Sun height min",0,150,450,100);

            NbJTextFieldDansUnPanelObservation++;
```

```
new CreerJTextFieldDansPanelObservation("Sun height
max",325,180,125,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("OBS_COND_UTC
UTC min",0,210,450,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("UTC
max",325,240,125,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("OBS_COND_PERIOD
Period:",0,270,450,100);

NbJTextFieldDansUnPanelObservation++;

new CreerJTextFieldDansPanelObservation("Duration",
325,300,125,100);

NbJTextFieldDansUnPanelObservation++;


new CreerJComboBoxDansPanelObservation("TRACK_STAR",
0,30,200);

NbJComboBoxDansUnPanelObservation++;

new CreerJComboBoxDansPanelObservation("
OBS_SERVOGUIDINGSETPOINT",0,90,200);

NbJComboBoxDansUnPanelObservation++;

new CreerJComboBoxDansPanelObservation("OBS_SERVOFOCUS",
0,120,200);

NbJComboBoxDansUnPanelObservation++;

Main.ArrayListPanelObservation.add(PanelObservation);

NbJPanelObservation++;
NbObservation++;
i++;

ComboBoxObs.setModel(new DefaultComboBoxModel(
VObservation));

// Création de la 1ère action pour l'observation venant
// d'être crée /

PanelObservation.add(BoutonAction);
BoutonAction.setBounds(0,350,150,25);

NouvelleAction.PanelAction = new JPanel();
PanelObservation.add(NouvelleAction.PanelAction);
NouvelleAction.PanelAction.setLayout(null);
NouvelleAction.PanelAction.setBounds(0,400,650,250);
```

```
                        PremièreActionDeObservationCree=1;

                        new CreerJTextFieldDansPanelAction("Temps :",
                        0,25,150,100);

                        new CreerJComboBoxDansPanelAction("Type :",
                        0,55,150,100,VTypeAcquisition);

                        PremièreActionDeObservationCree=0;

                        NouvelleAction.NbJTextFieldDansUnPanelAction++;
                        NouvelleAction.NbJComboBoxDansUnPanelAction++;

                        ComboBoxAct = new JComboBox(VAction);
                        NouvelleObservation.PanelObservation.add(ComboBoxAct);
                        ComboBoxAct.setBounds(250,350,150,25);
                        Main.ArrayListComboBoxActions.add(ComboBoxAct);


                        Main.ArrayListPanelAction.add(Main.ArrayListPanelAction.
                        size(),NouvelleAction.PanelAction);

                        NbJPanelAction++;

                        new ObservationSelectionee(ComboBoxObs);
                        new ActionSelectionee(ComboBoxAct);
                        new NouvelleAction(BoutonAction);
                    }
        }

        }

        public NouvelleObservation(JButton unBoutonObservation)
        {
          unBoutonObservation.addActionListener(new ObservationButtonListener());
        }
}
```

Annexe 15 :

ObservationSelectionee :

```
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.JComboBox;

public class ObservationSelectionee
{
int i;
String NumObservationString;
static int NumObservationInt=1;

	class ObservationComboBoxListener implements ItemListener
	{
		// Action effectué lorsqu'une observation est séléctionnée /

		public void itemStateChanged(ItemEvent e)
		{
			// Rend non visible tous les panel d'observation

			for(i=0;i<NouvelleObservation.NbJPanelObservation-1;i++){
			Main.ArrayListPanelObservation.get(i).setVisible(false);
		}

			for(i=0;i<NouvelleObservation.NbJPanelAction-1;i++){
				Main.ArrayListPanelAction.get(i).setVisible(false);
			}


		NumObservationString=NouvelleObservation.ComboBoxObs
		.getSelectedItem().toString().substring(11,12);

		NumObservationInt = Integer.parseInt(NumObservationString);

		// Rend visible uniquement le panel de l'observation séléctionée

		Main.ArrayListPanelObservation.get(NumObservationInt-
		1).setVisible(true);



		}
	}

	public ObservationSelectionee(JComboBox uneJComboBox)
	{
		uneJComboBox.addItemListener(new ObservationComboBoxListener());
	}
}
```

Annexe 16 :

SupprimerObservation :

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JPanel;


public class SupprimerObservation {

	class SupprimerButtonObservationListener implements ActionListener
	{
		public void actionPerformed(ActionEvent e)
		{
			int j;

			// Vérifie si l'observation séléctionnée actuellement est la
			// dernière observation crée /

			if(ObservationSelectionee.NumObservationInt!
			=NouvelleObservation.NbJPanelObservation)
			{
				for(j=ObservationSelectionee.NumObservationInt-
				1;j<NouvelleObservation.NbJPanelObservation-1;j++)

				{
					JPanel PanelTemp = new JPanel();

					PanelTemp =
					Main.ArrayListPanelObservation.get(j+1);

					PanelTemp.setBounds(600,100,650,700);


					Main.ArrayListPanelObservation.get(j)
					.setVisible(false);

					Main.Container.add(PanelTemp);
					Main.ArrayListPanelObservation.set(j,PanelTemp);

					Main.ArrayListPanelObservation.get(j)
					.setVisible(true);

					NouvelleObservation.ComboBoxObs
					.setSelectedItem(0);
				}
			}

			else
			{
			   Main.ArrayListPanelObservation.get(ObservationSelectionee.
			   NumObservationInt-1).setVisible(false);                  }

		Main.ArrayListPanelObservation.remove
```

```
            (NouvelleObservation.NbJPanelObservation-1);

            for(j=0;j<NouvelleObservation.NbJTextFieldDansUnPanelObservation;
            j++)

            {
                  Main.ArrayListTextField.remove
                  (CreerJTextFieldDansContainer.NbJTextFieldDescripteurs-1+
                  (ObservationSelectionee.NumObservationInt-1)*
                  NouvelleObservation.NbJTextFieldDansUnPanelObservation+1);
            }

            for(j=0;j<NouvelleObservation.NbJComboBoxDansUnPanelObservation;j++)
            {
                  Main.ArrayListComboBox.remove
                  (CreerJComboBoxDansContainer.NbJComboBoxDescripteurs-1+
                  (ObservationSelectionee.NumObservationInt-1)*
                  NouvelleObservation.NbJComboBoxDansUnPanelObservation+1));
            }


                  NouvelleObservation.VObservation.remove
                  (NouvelleObservation.VObservation.size()-1);

                  NouvelleObservation.ComboBoxObs.setModel(new
                  DefaultComboBoxModel(NouvelleObservation.VObservation));

                  NouvelleObservation.ComboBoxObs.revalidate();
                  NouvelleObservation.ComboBoxObs.repaint();


                  NouvelleObservation.NbObservation--;
                  NouvelleObservation.i--;

                  BoutonValider.NbJTextField=BoutonValider.NbJTextField-
                  NouvelleObservation.NbJTextFieldDansUnPanelObservation;

                  BoutonValider.NbJComboBox=BoutonValider.NbJComboBox-
                  NouvelleObservation.NbJComboBoxDansUnPanelObservation;

                  NouvelleObservation.NbJPanelObservation--;

            }

      }

      public SupprimerObservation(JButton unBoutonObservation)
      {
            unBoutonObservation.addActionListener(new
            SupprimerButtonObservationListener());
      }

}
```