Méthodes et Outils pour la Programmation: Rapport de projet Planethacks UFR-ST Besançon

Pierre LIMBALLE, groupe TP2B
Quentin PEREZ, groupe TP2B

January 1, 2016

1 Sujet

1.1 Contexte

Dans le cadre du module: Méthodes et Outils pour la Programmation (MOP) la réalisation d'un projet nous a été demandé afin de mettre en application les connaissances théoriques acquisent tout au long du semestre.
La conception de l'application a pour objectifs:

- l'utilisation du langage de programmation Java en orienté objets
- l'application du paradigme MVC (Modèle - Vue - Contrôleur)
- l'approche du travail collaboratif par l'utilisation de logiciels de gestion de versions
- la manipulation d'éléments graphiques de la bibliothèque graphique Java Swing

1.2 But de l'application

La finalité de ce projet est la conception Java une application qui réalise une animation graphique en 2D représentant un ensemble de planètes et leurs satellites en orbite autour d'une ou plusieurs étoiles. L'utilisateur doit disposer des fonctionnalités suivantes:

- l'ajout et la suppression d'astres (étoiles ou satellites) dans l'application
- l'affichage des astres dans une fenetre graphique animée
- la possibilité de sauvegarder un système planétaire

La contrainte d'ergonomie est également importante quant à l'utilisation du logiciel.

2 Conception

2.1 Introduction

Planethacks est le nom choisi pour l'application. Du fait d'un travail en binôme l'utilisation du logiciel de gestion de version Git en lien avec le serveur d'hébergement GitHub fut priviligiée (sources GitHub de Planethacks: `https://github.com/qperez/planetacks`).
L'IDE (*Internal Development Environement*) IntelliJ IDEA fut utilisé durant toute la phase de développement et de tests et ce afin d'accélérer et faciliter le développement de l'application ainsi que son déboguage.

2.2 Structures de données au sein de l'application

La définition de classes permettant de structurer les données de notre programme fut la première étape de conception. Ainsi 3 classes nous permettent de modéliser un astre:

- Astre (classe abstraite)
- Etoile
- Satellite

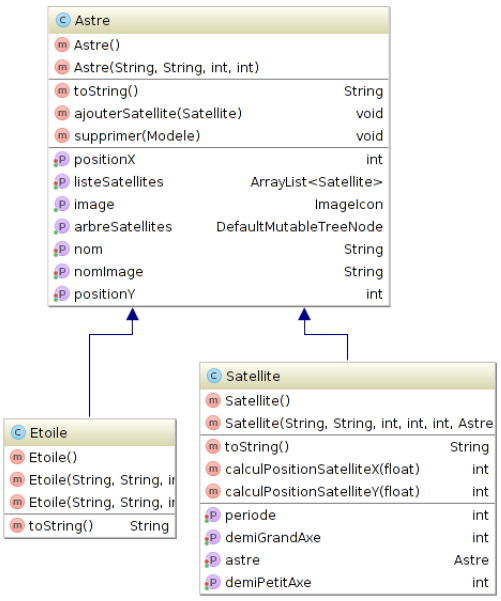
Planethacks repose sur une conception basée sur le paradigme MVC. Ce paradigme impose l'utilisation à minima de 3 classes (Modèle - Vue - Contrôleur) permettant de séparer les données, de l'affichage et des interactions utilisateur. Dans un souçis de respect du modèle MVC nous disposons donc des classes suivantes:

- Fenetre
- Modele
- Controleur (classe abstraite)
- ControleurMenu
- AffichageAstres

Afin de réaliser la fonctionnalité de sauvegarde une classe spécifique contenant les outils nécessaires fut créée: XMLTools.
Le point d'entré de l'application est assuré par la classe: Appli

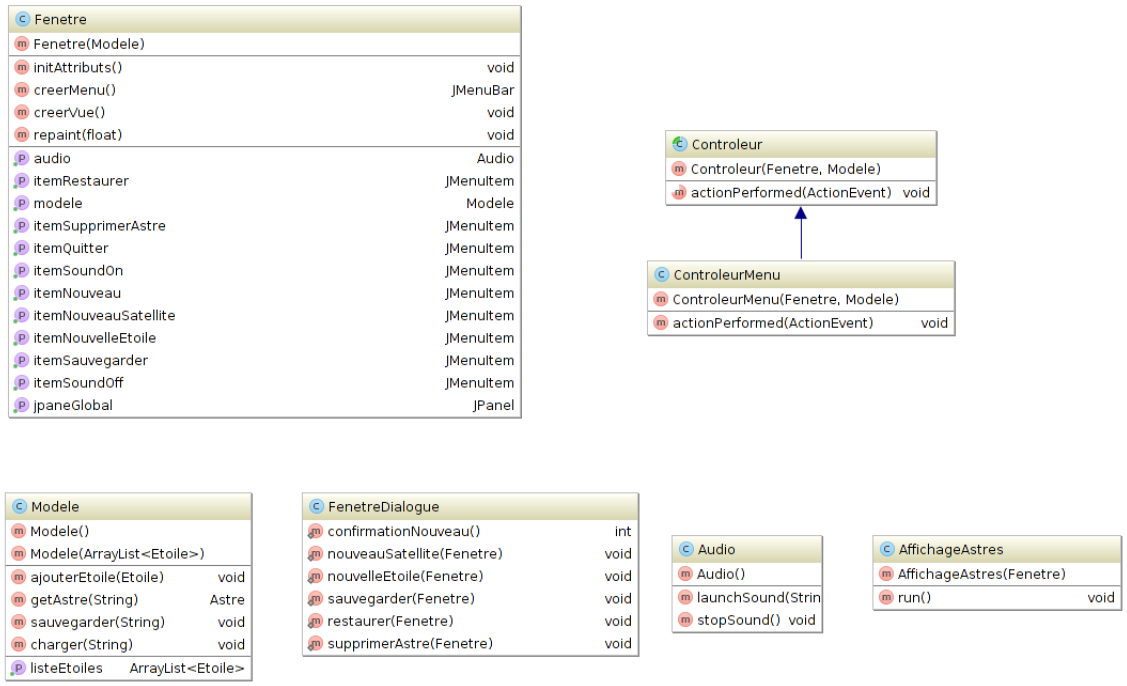
2.2.1 Modélisation d'un astre (Etoile ou Satellite

Le diagramme UML ci-dessous représente la modélisation choisie pour un astre:



Les méthodes : *calculPositionSatelliteX(float)* et *calculPositionSatelliteX(float)* présentent dans la classe Satellite permettent de calculer la position en X et en Y du Satellite en fonction du temps.

2.2.2 Architecture MVC



Algorithm 1 méthode repaint de la classe Fenetre

function REPAINT(t temps)
  Supprimer l'ensemble des composants graphique du JPanel
  for chaque Etoile e de la liste d'étoiles du Modele **do**
    $jlabastre \leftarrow$ nouveau JLabel avec l'image de e
    Afficher $jlabastre$ en fonction de la position X_e et Y_e
    Ajouter le $jlabastre$ au JPanel
    for chaque Satellite s de la liste des satellites de e **do**
      $jlabsat \leftarrow$ nouveau JLabel avec l'image de s
      Afficher $jlabsat$ en fonction de $s.calculPositionSatelliteX(t)$ et $s.calculPositionSatelliteY(t)$
      Ajouter le $jlabsat$ au JPanel
    end for
  end for
  Appelle de la méthode $repaint()$ sur le JPanel
end function

public void repaint(float t) jpaneGlobal.removeAll(); for (Etoile e : modele.getListeEtoiles()) JLabel jlabastre = new JLabel(e.getImage()); System.out.println("Astre = "+e.getNom()); System.out.println("X = "+e.getPositionX()); System.out.println("Y = "+e.getPositionY()); //jlabastre.sAfficher $jlabastre$etBorder(javax.swing.BorderFactory.createBevelBorder

```
jlabastre.setBounds(e.getPositionX(), e.getPositionY(), e.getImage().getIconWidth(),
e.getImage().getIconHeight());
    jpaneGlobal.add(jlabastre);
    for (Satellite s : e.getListeSatellites()) JLabel jlabsat = new JLabel(s.getImage());
jlabsat.setBounds(s.calculPositionSatelliteX(t), s.calculPositionSatelliteY(t), s.getImage().getIconWidth(),
s.getImage().getIconHeight()); jpaneGlobal.add(jlabsat);   jpaneGlobal.repaint();
```