

Sommaire

- **I - Interfaces Homme-Machines**
 - **1.1 Contexte**
 - **1.2 Personnas & user story**
 - **1.3 Diagramme des Cas d'utilisation**
 - **1.4 Analyse du diagramme de cas d'utilisation**
- **II – Conception Orientée Objet**
 - **2.1 Diagramme de paquetage**
 - **2.2 Analyse du diagramme de paquetage**
 - **2.3 Diagramme de classes**
 - **2.4 Analyse du diagramme de classes**
 - **2.4.1 Le cœur de l'application : les astres**
 - **2.4.2 Les fabriques d'astres**
 - **2.4.3 Les constellations**
 - **2.4.4 Le Manager et la Carte**
 - **2.4.5 Les classes utilitaires**
 - **2.5 Diagrammes de séquences**

I - Interfaces Homme-Machines

1.1 Contexte :

Nous cherchons à concevoir une application nommée *Stellar*. Elle porterait sur le thème de l'astronomie.

Nous avons conscience qu'il existe aujourd'hui de nombreuses applications sur l'astronomie, et c'est pourquoi nous avons voulu faire une application originale, utile et pratique, qui offre deux possibilités. Cette application se décompose donc en deux parties : une partie à objectif informatif et une partie à objectif créatif sous forme d'un éditeur.

Tout d'abord, on aurait une carte du ciel sur laquelle serait placé des étoiles et planètes, ou plus vaguement des *astres*. On aurait la possibilité d'obtenir différentes informations sur une étoile spécifique, telle que sa température, sa masse, son nom, sa constellation (si elle en a une), sa date de découverte, de même pour les planètes avec sa galaxie, etc.

L'application a également été pensée ergonomie, c'est pourquoi il serait possible de rechercher un astre en fonction de divers filtres (dont un système de favoris qui permet de sauvegarder les astres préférés de l'utilisateur).

La seconde partie consiste à ce que l'utilisateur puisse créer les étoiles et planètes qu'il souhaite (qu'elles existent ou non, il est libre de faire ce qu'il souhaite). Il peut ainsi les placer sur la carte, relier des étoiles entre elles pour créer des constellations, les supprimer... Il pourra aussi enregistrer son travail, puis le charger et le reprendre plus tard. Bien sûr, il lui sera possible d'exporter sa carte en image, afin qu'il puisse s'en resservir dans un projet s'il le souhaite, il peut aussi imprimer sa carte créée afin de pouvoir l'afficher chez lui !


Cette application permet à n'importe quel utilisateur s'intéressant à l'astronomie (de façon amateur essentiellement) de pouvoir obtenir des informations sur les étoiles.

Les développeurs qui ont des besoins spécifiques lors de création de jeux, d'univers, peuvent utiliser le côté éditeur de l'application afin de se servir du ciel créé dans des projets. Le caractère initial de l'éditeur peut également être contourné pour être utilisé comme logiciel de dessin de formes géométriques (on peut imaginer les étoiles comme des points, et les traits comme des segments).

Les personnes s'intéressant à l'astronomie et souhaitant apprendre sur les étoiles et planètes pourront utiliser le côté informatif de l'application pour s'instruire sur les astres qui peuplent notre galaxie, notamment lors de balades nocturnes car l'application ne nécessite aucune connexion internet.

Les écoliers (primaire) sont les cibles typiques de cette application, car elle se présente sous forme ludique et laisse libre cours à l'imagination. C'est donc une bonne application pour découvrir l'astronomie et apprendre en s'amusant, en utilisant toutes les fonctionnalités mises à disposition !

1.2 Personas & user story



Vincent Time

Je me renouvelle au contact de la nature et des gens

Age : 33 ans

Profession : sans emploi

Revenus : RSA

Situation : marié et père de deux enfants.

Ancien fonctionnaire, Vincent suit une formation de paysagiste pour se réorienter. Il adore les randonnées, treks et trails d'endurance, et son contact avec la nature est une source de vitalité, c'est aussi pour cela qu'il souhaite changer de métier.

Internet : fréquence basse d'utilisation

Sites préférés :

- photographesdumonde.com
- tela-botanica.org

Applications préférées :

- Ecobalade
- Stellarium

Passe-temps :

- Voyager, découvrir des paysages, des personnes, des cultures...
- Apprendre à connaître le monde qui l'entoure (plantes, animaux)
- Balades nocturnes avec sa famille

Profil technique :

- Assez bonne maîtrise des outils bureautiques (suite Microsoft).
- Aucune notion de programmation.
- Possède un téléphone Android Huawei P9, une tablette Samsung et un ordinateur sous Windows 10.
- Passe environ 1h/j sur les écrans.
- Se forme en continu sur les métiers d'extérieur.
- Aime partager ses connaissances avec sa famille lors de randonnées.

C'est lors d'une randonnée, la nuit, que Vincent regarda le ciel nocturne et réalisa la beauté de notre galaxie. Il chercha alors à obtenir des informations sur les étoiles et planètes, afin de les retenir et de pouvoir se repérer lors de ses futures randonnées. Il lui faudrait donc une application portative (sur sa tablette) capable de lui permettre de mémoriser ces données. Il se tourna alors vers *Stellar*, qui ne nécessite aucune connexion internet, simple d'utilisation. Elle lui permettra d'apprendre des choses sur des étoiles et planètes et de dessiner les constellations qui vont lui servir à se repérer.



Sam Convient

Il faut préférer ce qui est impossible mais vraisemblable à ce qui est possible, mais incroyable.

Age : 24 ans

Profession : Développeur

Revenus : 31 000 brut/an

Situation : en couple, sans enfant.

Sam travaille dans la même entreprise depuis maintenant 3 ans. C'est un développeur de petits jeux, mais il lui arrive de faire du design. En dehors de son travail, il aime se former sur les nouvelles technologies et réalise de la veille.

Internet : utilisation quotidienne et intense

Sites préférés :

- docs.oracle.com
- stackoverflow.com

Applications préférées :

- Perfect Piano
- Duolingo

Passe-temps :

- Programmer et apprendre de nouveaux langages, apprendre des langues.
- Jouer du piano (depuis 6 mois), la musique plus généralement.

Profil technique :

- Maîtrise de nombreux langages de programmation orientés objets (Java, C++).
- Connaissance des langages du Web (HTML/CSS, PHP).
- Possède un téléphone Android Samsung Galaxy S10, une tablette graphique Microsoft et deux ordinateurs sous Linux.
- Passe 8h/j sur les écrans.
- Se forme en continu sur les nouveaux langages et nouvelles technologies.
- Améliore son anglais et apprend le Russe.

Un client demande à Sam de réaliser un petit jeu 2D en Java, de type RPG. Il devra incorporer un cycle jour/nuit pour ce projet, or Sam n'a aucune idée de l'image de fond à utiliser pour cela. Il va devoir créer une image de ciel nocturne de toutes pièces, et se tourne alors sur *Stellar*, qui va lui permettre de réaliser ce dont il a besoin, de manière assez simple :





Lucie Fer

Plus tard quand je serais grande, je serais maîtresse et j'enseignerais les maths !

Age : 8 ans

Profession : Sans emploi

Revenus : Aucun

Situation : Ecolière (CM1)

Lucie est encore écolière de primaire, mais cela ne l'empêche pas d'être très bonne élève. Elle a d'ailleurs sauté la classe de CE2. Toujours joviale, elle souhaiterait plus tard devenir maîtresse et enseigner les mathématiques. Elle aime aussi l'astronomie.

Internet : utilisation exceptionnelle sous autorisation des parents

Sites préférés :

- coloriage.info
- apprendremagie.com

Applications préférées :

- Google Chrome
- LibreOffice

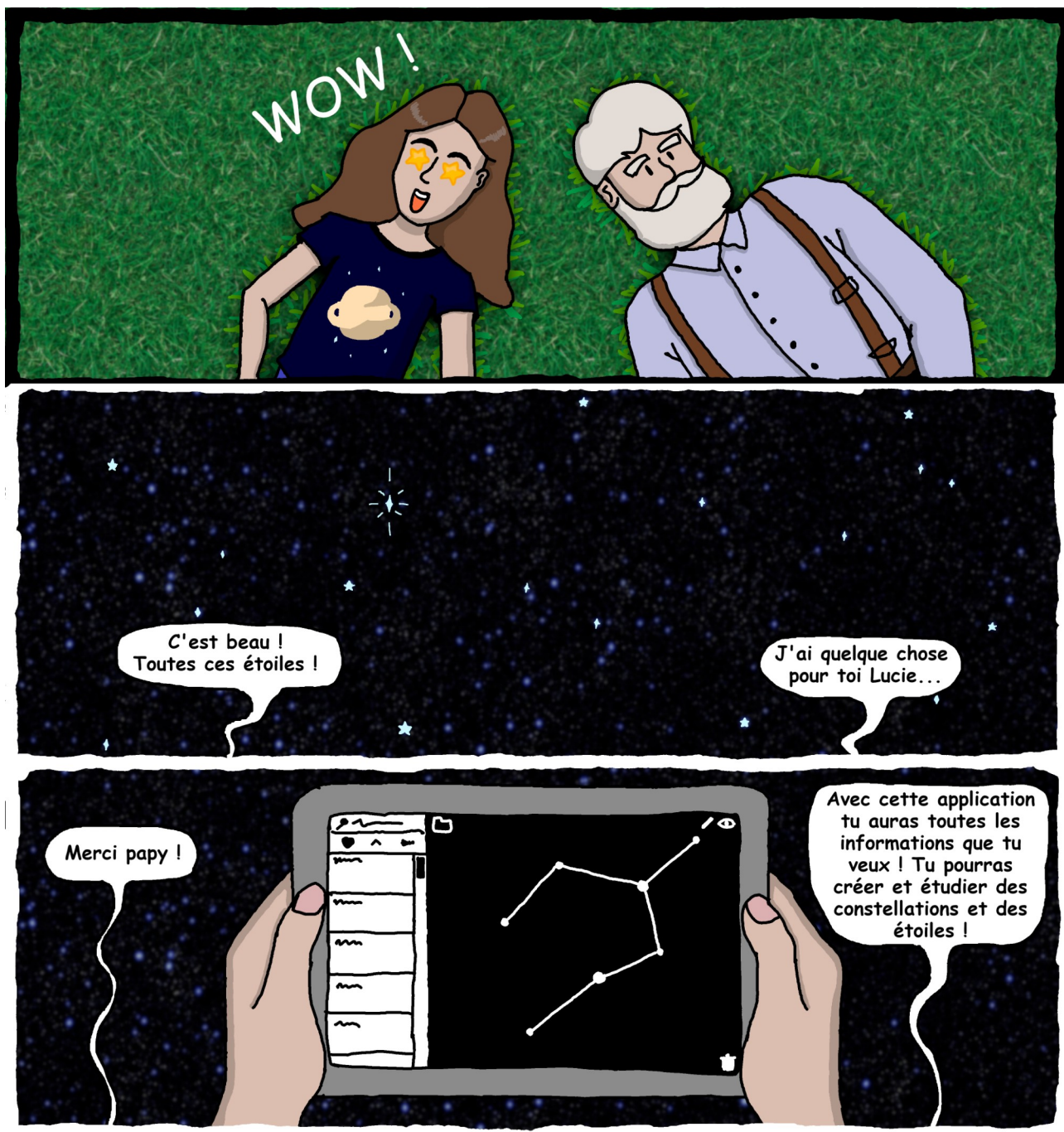
Passe-temps :

- Jouer avec ses ami.e.s (pendant les récréations).
- Apprendre à faire des tours de magie.
- Les coloriages et mots-croisés.
- Les sorties en famille.

Profil technique :

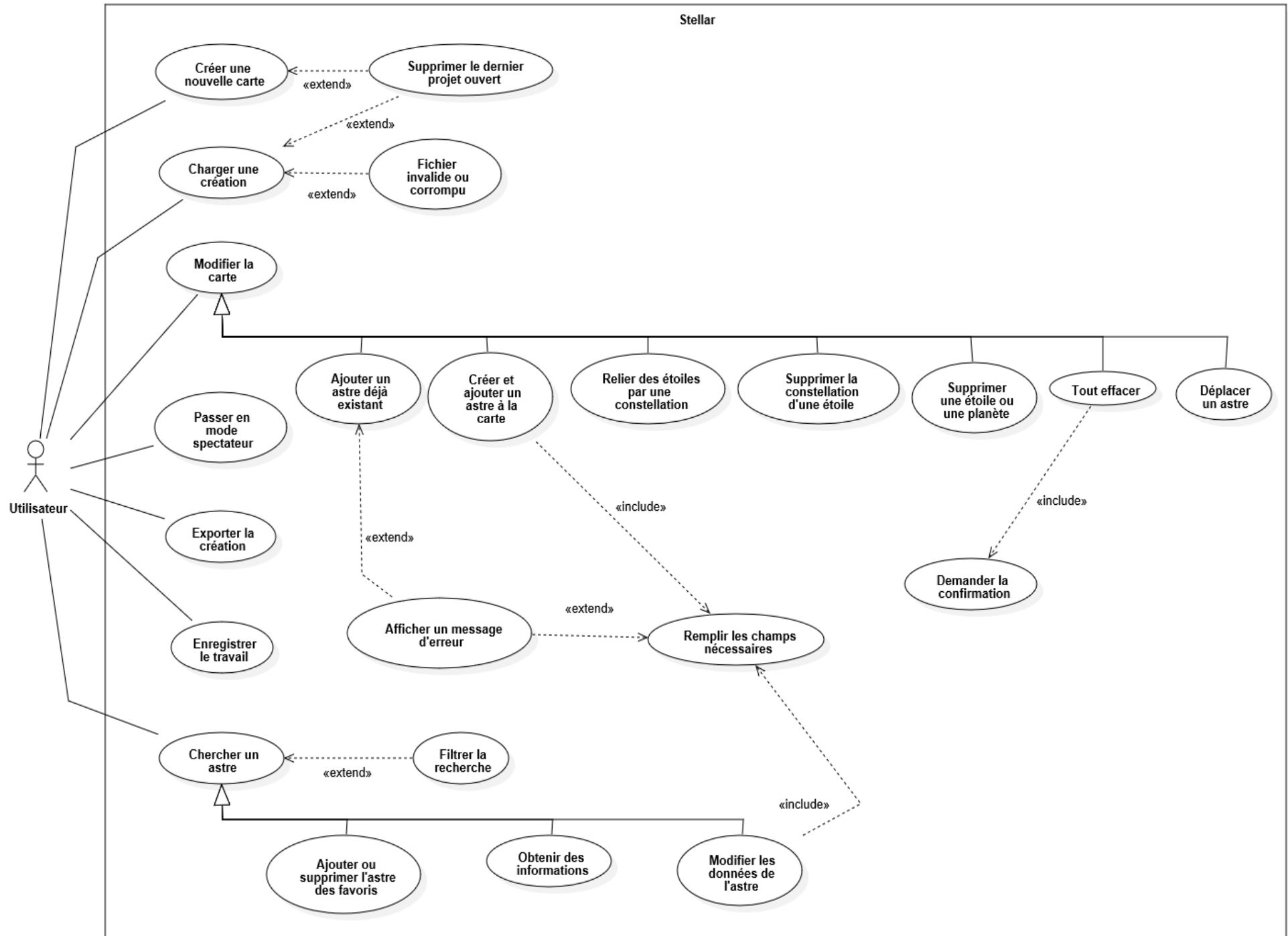
- Maîtrise la langue française et aussi l'Allemand (père d'origine Allemande).
- Connaît les rudiments des mathématiques (calculs, fractions, géométrie).
- A participé à plusieurs concours organisés par son école, dont le "Kangourou des mathématiques" qu'elle a remporté.
- Ne possède pas beaucoup de connaissances en informatique.
- S'intéresse à tout ce qui touche les sciences, comme le travail de sa mère (comptabilité).

Lors d'une soirée en extérieur avec son grand-père, Lucie regarde le ciel et se demande à quelle distance se trouve Sirius de chez nous... Alors son grand-père sort sa tablette avec l'application *Stellar*, de manière à ce que Lucie puisse obtenir toutes les informations qu'elle souhaite sur ces astres. Comme l'application ne nécessite aucune connexion internet, les parents n'ont pas besoin d'avoir recours à un contrôle parental !



Note : les visages de ces personnes ont été générés aléatoirement via le site <https://thispersondoesnotexist.com/>
Ils n'existent donc pas en vrai, il n'y a donc pas de droit d'image / de copyright.

1.3 Diagramme des Cas d'utilisation



1.4 Analyse du diagramme de cas d'utilisation

Nom	Charger une création
Objectifs	Charger un fichier de sauvegarde contenant les données de la création d'une carte (étoiles et planètes créées, informations sur celles-ci, constellations...).
Acteurs principaux	Utilisateur
Acteurs secondaires	Aucun
Conditions initiales	-L'utilisateur doit posséder un projet de l'application qu'il aura du enregistrer précédemment (qui contient des données de sauvegarde).
Scénario d'utilisation	<p>-L'utilisateur sélectionne « Ouvrir un fichier » depuis le menu dossier, ou réalise le raccourci clavier Ctrl + O.</p> <p>-Un message s'affiche comme quoi l'utilisateur peut ouvrir un projet, mais celui lui fera perdre les données du projet actuel, et lui conseille d'enregistrer le projet actuel avant d'en ouvrir un nouveau. L'utilisateur peut souhaiter de continuer et donc d'ouvrir le fichier, le cas « Supprimer le dernier projet ouvert » est alors réalisé. Si non, conditions de fin 2.</p> <p>-L'utilisateur est invité à parcourir les fichiers depuis son explorateur et sélectionne le fichier souhaité.</p> <p>-Si l'application détecte que le fichier ne contient pas les données d'un projet, ou si le fichier est corrompu, ou impossible à lire ou ouvrir, alors un message d'erreur s'affiche disant que le fichier n'a pu être traité (voir conditions de fin 3). Le cas « Fichier invalide ou corrompu » est réalisé.</p> <p>-S'il n'y a pas de problème, le projet est chargé en mémoire puis ouvert depuis l'application (conditions de fin 1).</p>
Conditions de fin	<p>1) Le projet a bien été chargé, l'utilisateur peut retrouver son travail (ses étoiles, planètes et sa carte), et travailler dessus.</p> <p>2) L'utilisateur a préféré annuler l'ouverture pour ne pas perdre son travail actuel. Il doit alors d'abord réaliser le cas « Enregistrer le travail » puis de nouveau réaliser le cas « Charger une création ».</p> <p>3) L'utilisateur a indiqué un mauvais fichier ou alors le fichier était corrompu ou déjà ouvert. Il doit alors veiller à fermer les fenêtres susceptibles d'utiliser ce document, et vérifier le fichier qu'il envoie au programme, puis de nouveau réaliser le cas « Charger une création ».</p>

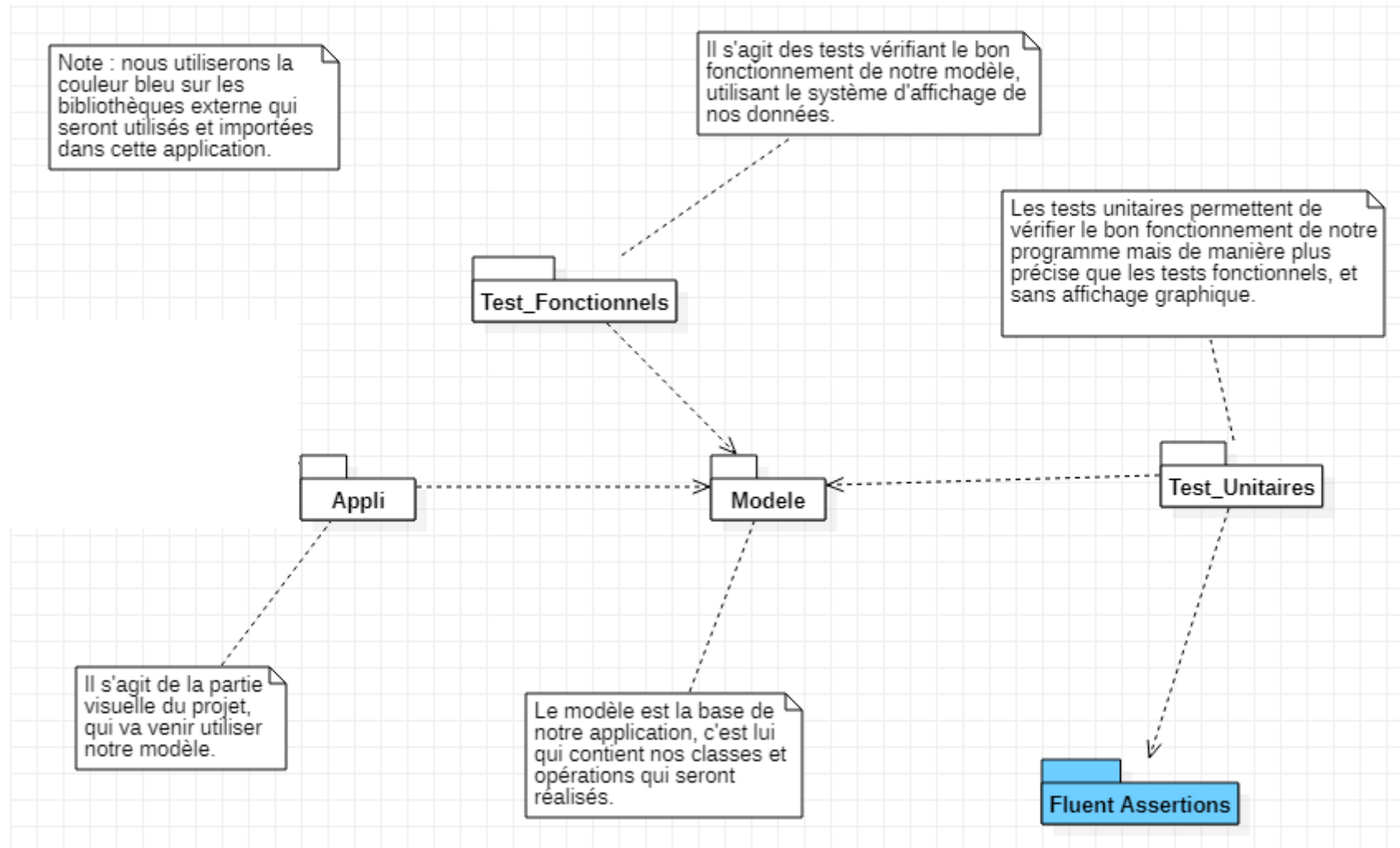
Nom	Supprimer une étoile ou une planète
Objectifs	Supprimer une étoile ou une planète de la carte, et les constellations (arêtes) qui lui sont reliées (si elle en a).
Acteurs principaux	L'utilisateur
Acteurs secondaires	Aucun
Conditions initiales	-L'utilisateur doit avoir au moins une étoile ou planète sur sa carte.
Scénario d'utilisation	<p>-L'utilisateur clique l'outil « suppression » (la gomme, en haut à droite).</p> <p>-Il peut annuler sa suppression, en cliquant de nouveau sur l'outil gomme (conditions de fin 2).</p> <p>-Il peut cliquer sur l'étoile ou la planète à supprimer. Si c'est une étoile et qu'elle possède des arêtes (= constellations) qui lui sont attachées, alors le premier clic sur l'étoile supprimera ces arêtes. Le second clic supprimera l'étoile. Si l'étoile n'a aucune arête, un clic suffira à supprimer l'étoile de</p>

	la carte. S'il s'agit d'une planète il suffira d'un seul clic également. L'utilisateur peut de nouveau cliquer sur l'outil gomme afin de quitter le mode suppression (conditions de fin 1).
Conditions de fin	1) L'étoile ou planète est alors effacée de la carte, mais aussi de la liste de gauche (elle disparaîtra) s'il s'agissait d'un astre personnalisé (= créé par l'utilisateur). Si c'est un astre intégré à l'application, il sera toujours visible dans la liste de gauche et pourra être de nouveau placé sur la carte. 2) L'utilisateur a annulé son action en désélectionnant l'outil gomme. S'il veut la supprimer, il a juste à réaliser de nouveau le cas « Supprimer une étoile ou une planète ».

Nom	Créer et ajouter un astre à la carte
Objectifs	Créer un astre personnalisé, avec le nom et informations que l'utilisateur souhaite, et le placer sur la carte.
Acteurs principaux	L'utilisateur.
Acteurs secondaires	Aucun.
Conditions initiales	Aucune.
Scénario d'utilisation	-L'utilisateur clique sur le bouton « Ajouter » (symbole étoile s'il veut ajouter une étoile, symbole rond s'il veut ajouter une planète) en haut à droite de la carte. -Il clique ensuite sur une zone de la carte pour pouvoir ajouter son étoile ou sa planète qui sera placée à cette position. Un pop-up s'ouvre et le cas « Remplir les champs nécessaires » doit être réalisé. Pour cela, l'utilisateur doit remplir les champs obligatoires notés d'un astérisque. S'il clique sur la croix ou sur « Annuler », alors l'astre sera supprimé de la carte et l'ajout ne sera pas réalisé (conditions de fin 2). -Sinon, il peut cliquer une « Valider ». S'il a choisi un nom d'astre déjà existant dans le logiciel, alors le cas « Afficher un message d'erreur » est réalisé. Dans ce cas, un pop-up s'affiche et lui demande de changer de nom. S'il clique sur la croix ou sur « Annuler », alors l'astre sera supprimé de la carte et l'ajout ne sera pas réalisé (conditions de fin 3). -Sinon, l'ajout se fait correctement (conditions de fin 1).
Conditions de fin	1) L'ajout a été réalisé sur la carte. L'astre a été créé et est maintenant visible depuis la liste de gauche. Il peut modifier les champs de celui-ci s'il le souhaite. 2) L'utilisateur a annulé son ajout. L'astre n'a pas été ajouté. S'il veut ajouter un astre, il doit alors réaliser de nouveau le cas « Créer et ajouter un astre à la carte ». 3) L'utilisateur a annulé son ajout suite à la réalisation du cas « Afficher un message d'erreur ». Il peut alors vérifier depuis la liste (en utilisant la barre de recherche), si le nom de l'astre déjà existant est un astre personnalisé. Si non, il lui faut trouver un autre nom pour son étoile qui ne peut pas avoir le même nom qu'une étoile déjà présente dans le système, et réaliser de nouveau le cas « Créer et ajouter un astre à la carte ». Si oui, alors il peut réaliser le cas « Supprimer une étoile ou une planète », qui supprimera son astre personnalisé précédent, puis le cas « Créer et ajouter un astre à la carte », qui créera un astre avec le nom qu'il souhaitait lui donner.

II – Conception Orientée Objet

2.1 Diagramme de paquetage



2.2 Analyse du diagramme de paquetage

Nous utiliserons dans ce modèle une bibliothèque de classes, nommée *Modele*, qui contiendra les classes, méthodes, interfaces, énumérations et autres opérations qui seront utilisées par notre *appli*.

L'*appli* justement sera en réalité la vue, sur laquelle il sera possible de naviguer, et de réaliser des actions qui auront été justement définies dans le *Modele* comme expliqué précédemment. Il s'agit de l'aspect visuel de notre projet.

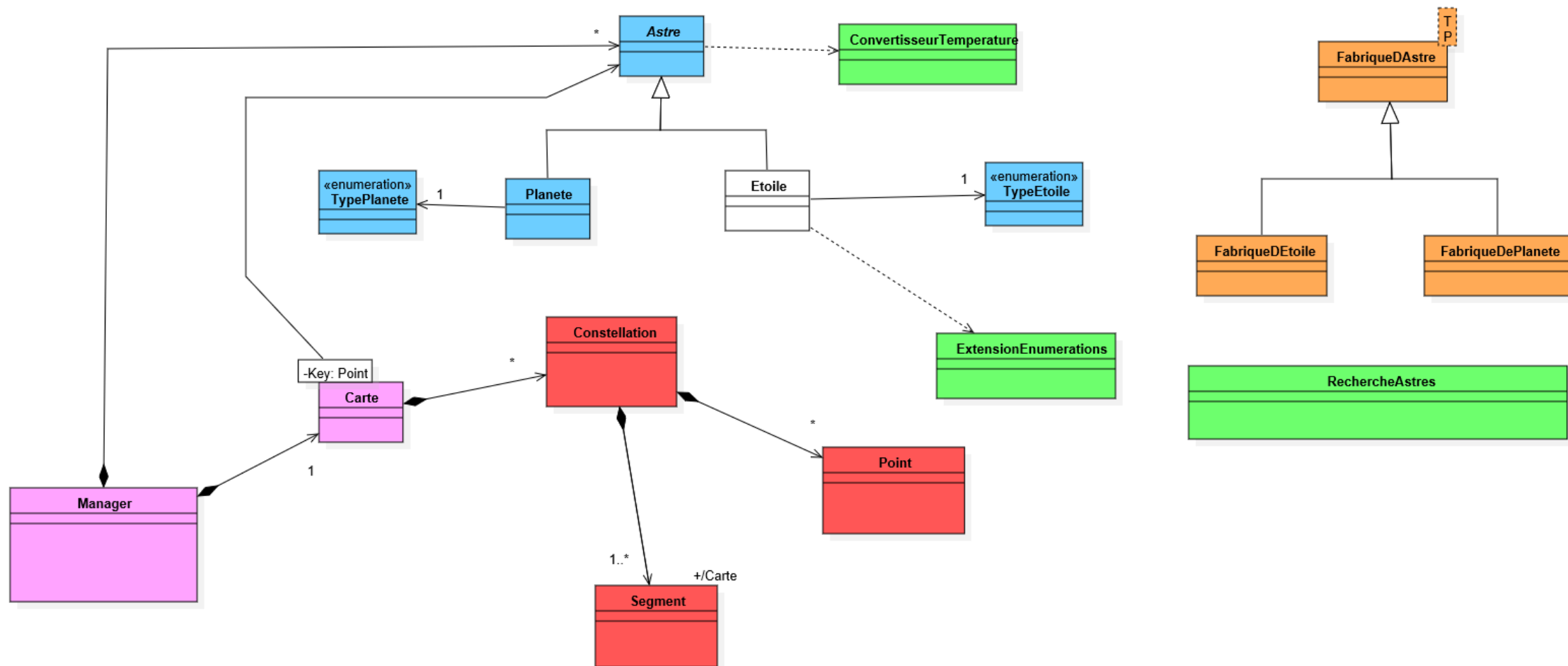
Les tests fonctionnels se déroulent sur application console, et permettent de tester les différentes méthodes, collections, et opérations définies dans le *Modele*. Ils utilisent le système d'affichage, ce qui permet de vérifier le bon fonctionnement de notre *Modele*. Ils se décomposent en plusieurs classes qui contiennent elles-mêmes plusieurs méthodes, permettant de vérifier le fonctionnement d'éléments spécifiques.

Les tests unitaires sont similaires aux tests fonctionnels, mais n'utilisent pas d'interface console. Ils vérifient simplement la cohérence des données, contenues dans des instances, classes, après des instanciations ou opérations, telle que la suppression, l'ajout, la modification, etc. de notre *Modele*. Ces tests se décomposent également en plusieurs classes, permettant chacune de tester diverses instanciations ou opérations.

Les tests unitaires utiliseront *FluentAssertion*, qui est un *Nuget* (une bibliothèque externe), qui permet de réaliser des tests de manière plus lisse, plus facile à lire, à créer, et à comprendre.

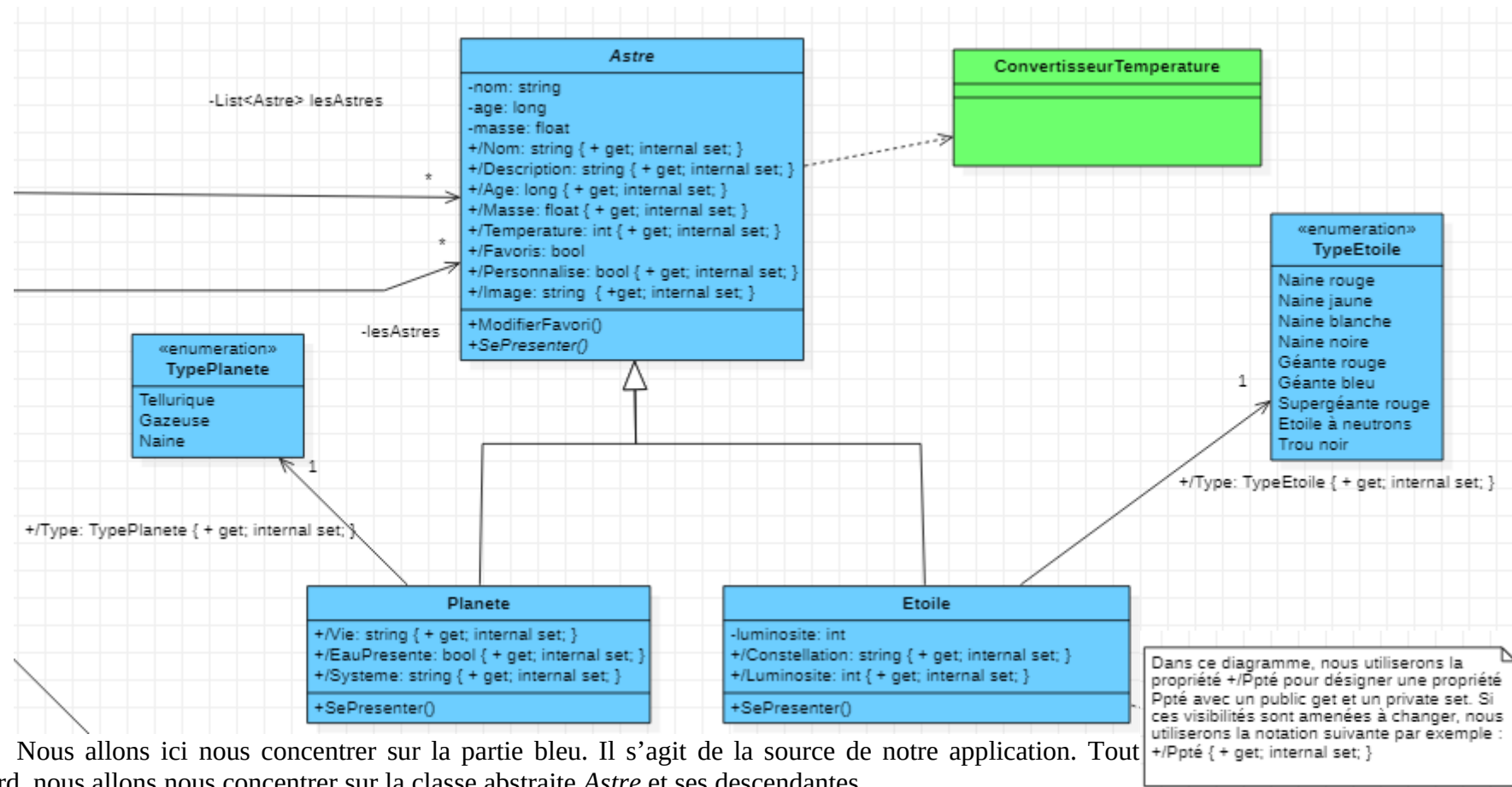
2.3 Le diagramme de classes

Il est possible de voir ici notre diagramme de classe de manière simpliste (les méthodes, attributs, propriétés ont volontairement été enlevés). Les différentes parties colorées vont être détaillées sur la suite de ce document.



2.4 Analyse du diagramme de classes

2.4.1 Le cœur de l'application : les Astres



Nous allons ici nous concentrer sur la partie bleu. Il s'agit de la source de notre application. Tout d'abord, nous allons nous concentrer sur la classe abstraite *Astre* et ses descendantes.

Un astre est un objet quelconque de l'Univers, il est caractérisé par un nom, un age (qui peut être assez grand), une masse, d'une température, et éventuellement d'une description. Dans le cadre de cet application nous auront deux types d'astres : ceux déjà présents dans le logiciel, et ceux créés par l'utilisateur. Il y aura donc un booléen qui indiquera si un astre est personnalisé ou non.

Un astre pourra être noté comme étant dans les favoris de l'utilisateur, d'où le booléen permettant d'indiquer cela, il y aura donc une méthode permettant de changer l'état du favori de l'astre. Il possédera également une image.

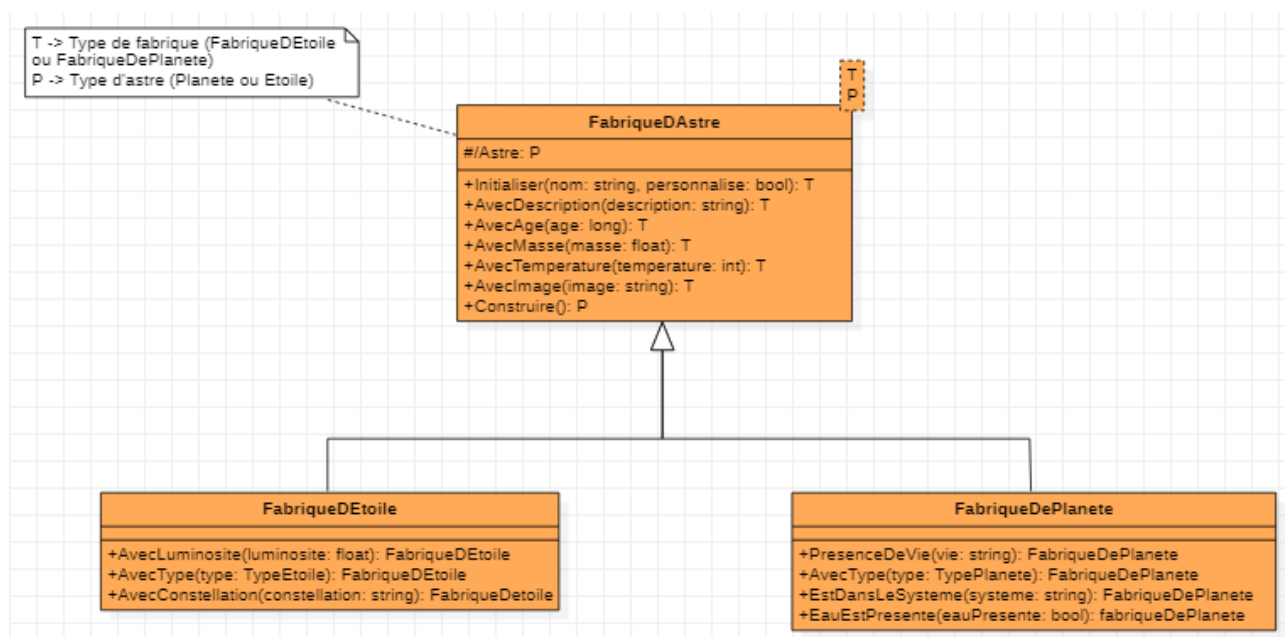
Or un Astre est un objet assez abstrait, d'où l'intérêt ici de ne pas pouvoir en instancier un. En revanche, il y a deux liens d'héritage qui partent d'*Astre*, et permettent de spécifier qu'il peut s'agit d'une étoile ou d'une planète. Chacune possède ses propres caractéristiques, notamment une luminosité et une constellation pour l'étoile, ainsi qu'une indication sur la présence d'eau ou de vie, et le système stellaire dans le cas d'une planète.

Chaque spécification de l'Astre possède un Type. Une étoile peut être une naine blanche, une naine jaune, un trou noir..., il y a donc une énumération qui permet d'énumérer ces différents types. Dans le cas de la planète, c'est similaire. Elle peut être tellurique, gazeuse ou naine.

Chaque astre pourra se présenter, et donner son type.

Avec cette configuration il est facile de maintenir le code et de rajouter de nouvelles spécifications d'astres par la suite.

2.4.2 Les fabriques d'Astres



Nos constructeurs d'astres étaient assez gros suite à l'héritage, et nous voulions pouvoir créer des astres de manière un peu plus fluide, un peu plus lisible et propre, car on a beaucoup de paramètres dans nos constructeurs.

Pour cela, nous avons réalisé le patron de conception correspondant à la fabrique. Nous avons également une relation d'héritage entre les fabriques, un astre restant un objet abstrait, il nous faut spécifier si nous allons créer une étoile ou une planète. Par conséquent, la *FabriqueDAstre* sera utilisée par la *FabriqueDEtoile*, et par la *FabriqueDePlanete*.

La fabrique d'astre utilise donc la généricité. Elle prend deux types *T* et *P* lors de l'instanciation. *T* correspond au type de fabrique (donc dans notre cas, on veut soit avoir une

FabriqueDEtoile, soit une *FabriqueDePlanete*), et *P* au type d'astre que l'on souhaite créer (donc dans la même logique, soit une *Etoile*, soit une *Planete*).

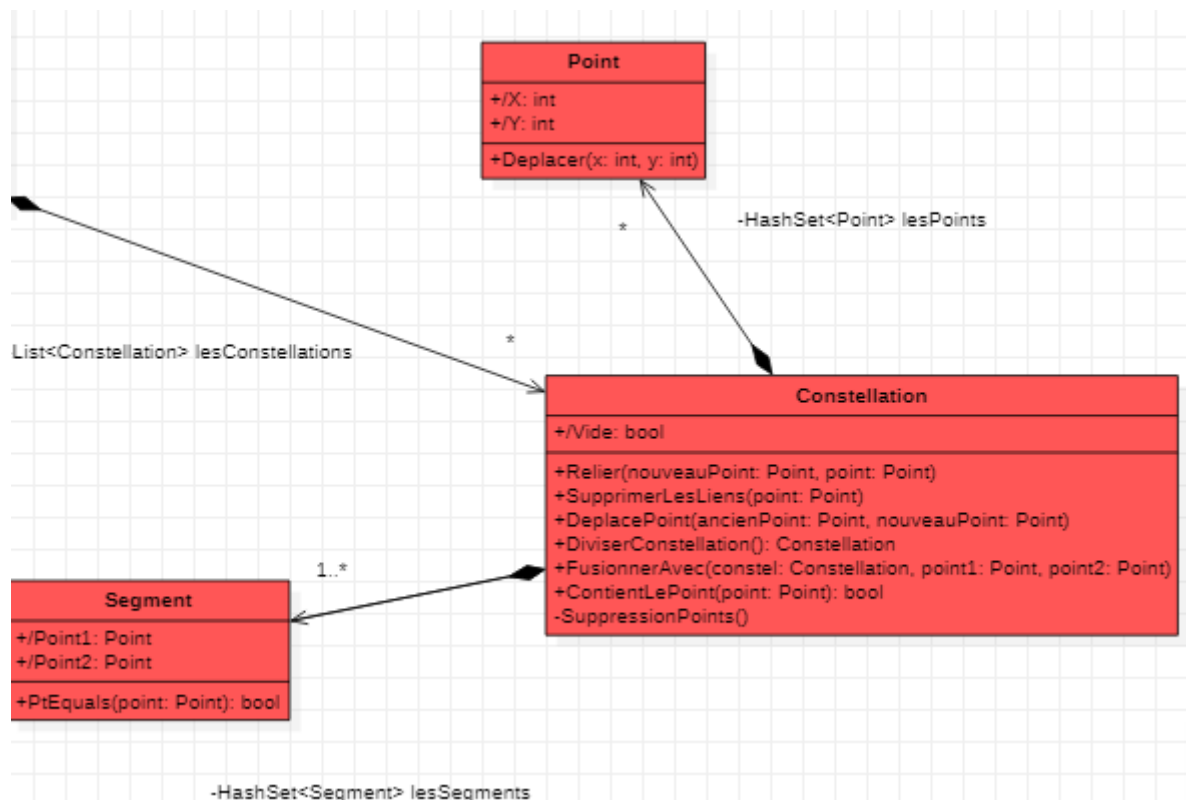
L'astre contenue dans la fabrique d'astre est protégée afin que ses classes filles y ait accès. Il est toujours de type *P* (*Etoile* ou *Planete*).

Par la suite, on retrouve une méthode d'initialisation qui permet d'instancier un astre en fonction de son type générique *P*. Concrètement on vient créer soit une étoile soit une planète. Il existe diverses méthodes permettant d'ajouter des paramètres à notre astre *P*, que l'on retourne sous forme de fabrique *T*. Cela permet de chaîner les méthodes entre elles, et faire des appels successifs d'ajouts de paramètres, de manière plus épurée qu'avec un gros constructeur.

Les classes *FabriqueDEtoile* et *FabriqueDePlanete* permettent d'ajouter les méthodes propres aux attributs de l'astre en question. Il ne sera possible de renseigner la constellation que pour une étoile par exemple.

Enfin la méthode de construction dans la fabrique d'astres permet de retourner cette fois-ci notre astre créé.

2.4.3 Les constellations



Il doit être possible dans notre application de créer des constellations. Ces constellations peuvent donc s'apparenter à un graphe : c'est un ensemble de points reliés entre eux par des segments.

Nous avons donc créé une classe *Point*. Un point est caractérisé par ses coordonnées en abscisses et en ordonnées, et il est possible de le déplacer.

Un *Segment* est caractérisé par deux points : le point de départ et le point d'arrivée. On

considère que deux segments sont égaux s'ils possèdent les deux mêmes points, mais pas forcément dans le même ordre (le segment n'est pas orienté). Il possède également une méthode permettant de savoir si un point est contenu dans le segment.

Enfin une *Constellation* est composée d'une collection de points (sous forme de *HashSet* pour éviter les doublons), et de segments (de même que pour les points).

Une constellation est composée au minimum de deux points et d'un segment.

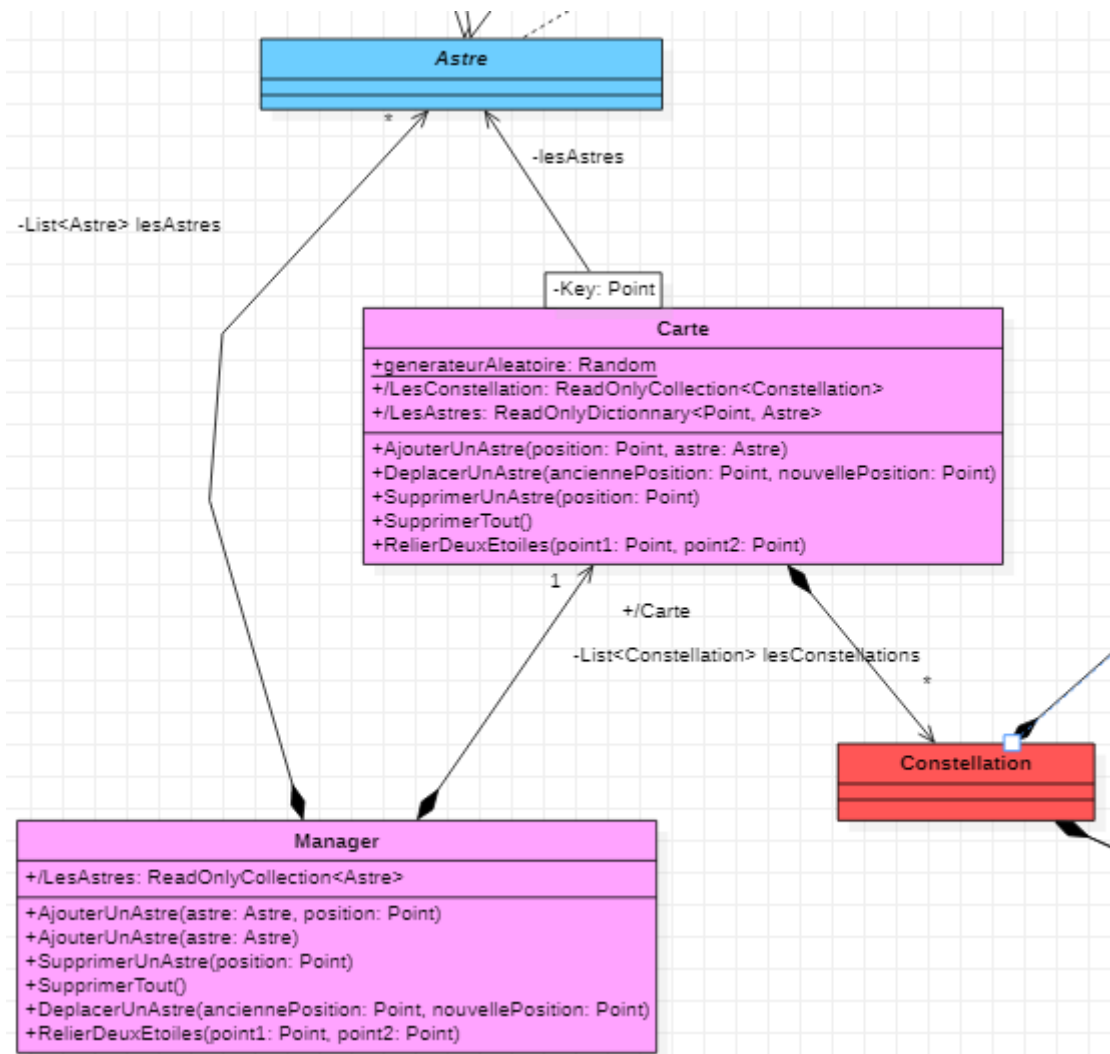
L'utilisateur doit pouvoir relier des étoiles entre elles, et même déplacer les points. Cette classe possède donc diverses méthodes, notamment une lui permettant de relier deux points, de déplacer un point, de savoir si un point est contenu dans la constellation.

Lorsque que deux points sont reliés, il peut arriver que deux constellations soient amenées à être fusionnées ensembles, d'où la méthode présente ici.

Enfin, il doit être possible de supprimer les segments d'un point en particulier, et donc effacer ce point de la collection (la méthode *SupprimerLesliens*). Il y a une méthode permettant d'effacer les points qui ne sont reliés à rien.

Pour finir, la méthode *DiviserConstellation* permet de réaliser un parcours en largeur sur la constellation afin de ne garder qu'une composante connexe, et retourne une nouvelle constellation contenant l'autre composante connexe.

2.4.4 Le Manager et la Carte



La classe *Carte* est la classe qui correspond à la partie éditeur de notre application (elle contient les données de cette partie de l'application). C'est elle qui est chargée de modifier les données concernant les constellations, points, qui sont ajoutés / supprimés / déplacés sur la carte.

Elle contient une liste de constellations, conformément à ce qui a été dit précédemment. Elle possède ainsi une méthode pour relier des étoiles. Attention, il est possible de ne relier que des étoiles, pas des planètes.

Elle est également caractérisée par un dictionnaire clé/valeur avec pour clé un *Point*, et pour valeur un *Astre*, ce qui permet de rapidement retrouver un astre à partir de sa position sur la carte.

Il est donc possible d'ajouter un astre sur la carte, avec un nouveau point de position. Il est également possible de déplacer un astre à une nouvelle position. Un astre peut aussi être supprimé, ce qui peut entraîner des modifications des constellations. Enfin, il est possible de tout effacer.

La classe *Manager* est la classe centrale de notre *Modele*. C'est elle qui gère la navigation dans notre fenêtre en appelant les méthodes nécessaires. Elle contient également les données de l'application car elle possède une liste d'astres, ce sont les astres contenus dans l'application mais

aussi ceux qui ont été créés par l'utilisateur. Elle possède également une Carte sur laquelle l'utilisateur pour ajouter et modifier des données.

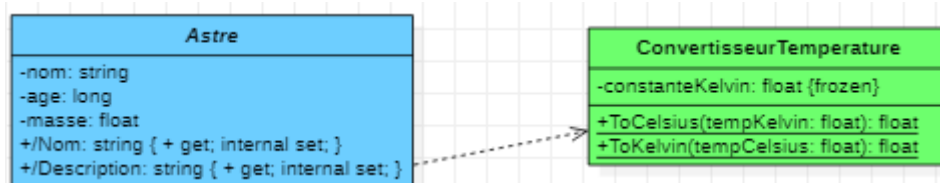
Il est donc naturellement possible de modifier cette liste d'astres en ajoutant des données, en supprimant, en en modifiant certaines.

Le Manager sera aussi là pour déléguer. La plupart des actions effectués lors de l'ajout d'un astre ou de sa suppression, le fait de relier des étoiles, etc. auront un impact sur la carte qui sera modifiée à son tour.

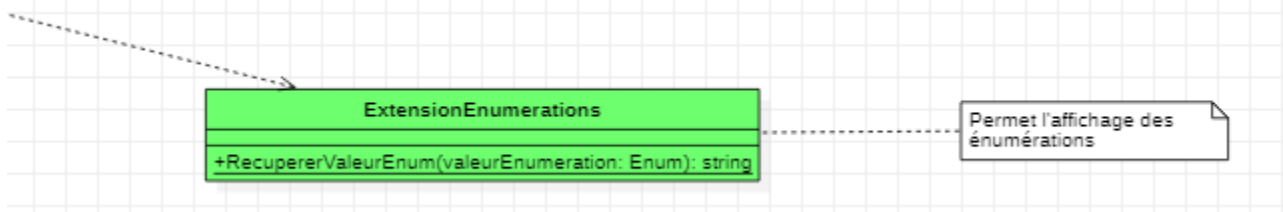
On observe donc ici un patron de conception bien précis : la façade.

La façade ici, la Manager, gère les données principales et délègue ses actions à la carte. Il s'agit de la pièce maîtresse de l'application. A aucun moment les sous-classes appelés par le Manager ne font référence à la façade, elles ne le connaissent même pas. Elle permet de simplifier un système de gestion complexe en une seule classe qui va venir gérer le code de manière autonome.

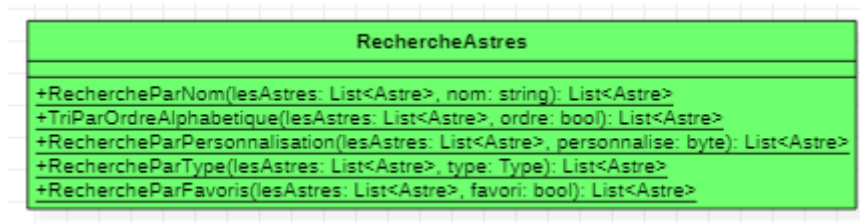
2.4.5 Les classes utilitaires



La classe *ConvertisseurTemperature* permet de convertir la température en différentes unités, et permet d'obtenir un rendu et un affichage plus cohérent. Il est utilisé au sein de la classe *Astre*.



La classe *ExtensionEnumerations* est aussi une classe utilitaire permettant de récupérer une énumération amis avec des espaces, et donc de pouvoir l'afficher de manière à avoir un meilleur rendu textuel, et d'avoir des caractères spéciaux et espaces.



Enfin la classe *RechercheAstres* permet d'effectuer divers tris et filtrages. Elle sera utilisée dans l'application pour pouvoir trier les différents astres en fonction des envies de l'utilisateur.