Une image contenant texte, clipart

Description générée automatiquement

**Documentation du Projet Java**



**08 /01/2021**

**Réalisé par :**

**Abdel OUEDRAOGO & Rhode DAY**

Remerciements

Nous souhaitons remercier nos enseignants Raùl MAZO et Sylvain GUERIN qui nous ont permis de travailler sur ce projet très intéressant afin de concrétiser notre apprentissage à l’école. Merci d’avoir pris le temps de dispenser ce cours tout en ayant à cœur notre compréhension.

Présentation du Projet

Ce document dresse les travaux effectués concernant le projet Java Mission to Mars.

Dans ce projet, on devait construire un outil de simulation qui nous aidera dans notre mission vers Mars. La mission est d'envoyer une liste d'articles (Habitats, bunkers, vivres et rovers) à Mars, mais nous devons d'abord exécuter des simulations pour choisir la bonne flotte de fusées. deux prototypes de fusées avaient déjà été conçus, et nous, nous devrions concevoir et exécuter quelques simulations pour aider à décider quel type utiliser.

ButLa mission se compose de 2 phases :

La phase 1 est destinée à envoyer des équipements de construction et des matériaux de construction pour aider à construire la colonie.

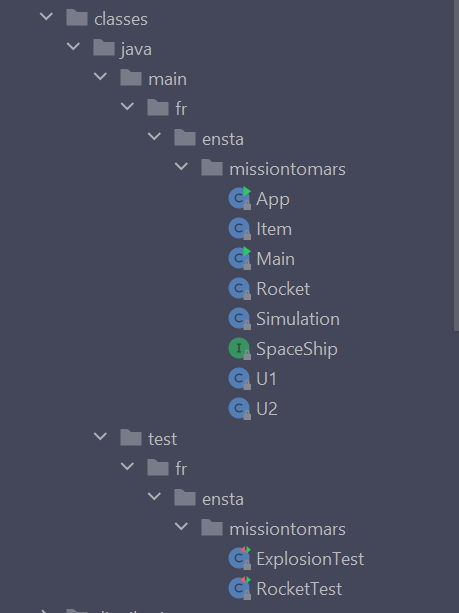
La phase 2 est destinée à envoyer la colonie d'humains avec quelques ressources alimentaires.

ObjectifsNotre travail consistait à faire des simulations et à tester les deux types de fusées pour chaque phase séparément. Et aussi construire un outil d’aide à la décision à travers le logiciel.

Méthodologie

Pour la réalisation du projet, nous avons organiser notre collaboration en présentiel. Nous nous accordions sur les horaires et nous nous retrouvions tous les deux durant la période de congés pour avancer pas à pas jusqu’à atteindre l’interface graphique. Un dépôt GitHub a été tout de même créé pour que chacun puisse garder un œil sur le projet et faire des ajouts en cas d’inspiration. (<https://github.com/Abdel-Oued/Mission_to_MARS> )

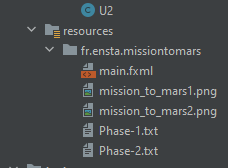
Structure du projet

Un dossier avec les différentes classes

* App
* Item
* Main
* Rocket
* Simulation
* SpaceShip
* U1
* U2

Un dossier Test avec les classes

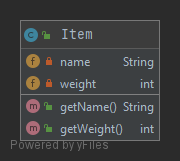
* ExplosionTest
* RocketTest

Un dossier ressource qui contient la partie JavaFX

* main.fxml

Présentation des classes

**Classe Item**Classe des Items à transporter qui permet de récupérer des valeurs comme le nom et le poids d’un Item.



**Interface SpaceShip**Représentation d’un engin spatial. L’ensemble des fonctionnalités sera implémenté par la classe Rocket.

**Classe Rocket**Classe qui implémente l’interface SpaceShip.

**Classe U1**Classe des rockets de types U1. Les chances d’explosions au lancement et à l’atterrissage ont été redéfinies de sortes à ce qu’elles ne soient plus linéaires.

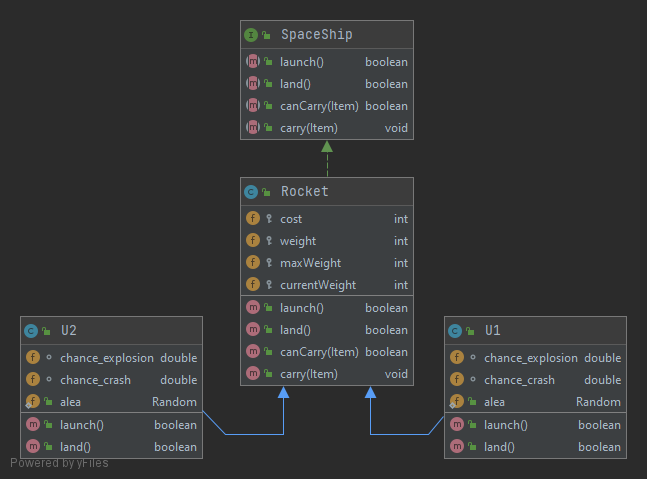
Chance of launch explosion = 5% \* (cargo carried / cargo limit)5

Chance of landing crash = 1% \* (cargo carried / cargo limit)5.

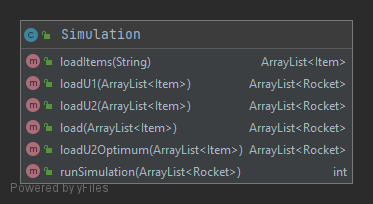
**Classe U2**Classe des rockets de types U1. Les chances d’explosions au lancement et à l’atterrissage ont été redéfinies de sortes à ce qu’elles ne soient plus linéaires.

Chance of launch explosion = 4% \* (cargo carried / cargo limit)5

Chance of landing crash = 8% \* (cargo carried / cargo limit)5.



**Classe Simulation**Cette classe définit les méthodes propres à la simulation. On y retrouve différentes méthodes de chargements des rockets dont il est important d’expliquer le principe.



* loadItems

Créer à partir du fichier texte qui lui est passé en paramètre une liste des items d’un phase donnée. On parcourt chaque ligne du fichier et on extrait le nom et le poids de l’item sur cette ligne.

* loadU1

Cette méthode crée un covoit de rocket U1 pour les deux phases. Les rockets sont chargées, avec les items de la liste retournée par loadItems, tant qu’elles n’ont pas atteint leur poids maximal. Il n’y a pas d’optimisation quant à la manière dont le chargement est effectué.

* loadU2

Pareil que loadU1 mais avec des rockets de type U2.

* load

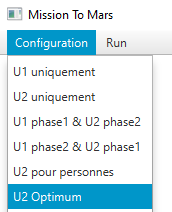
Dans cette méthode, on fait la différence entre les items à transporter. Les rockets U2 sont dédiées aux personnes et aux gros items (dont le poids > 5 tonnes) tandis que les rockets U1 se chargent du reste. Aussi, les rockets U1 n’excèdent jamais leurs poids maximal – 3 tonnes.

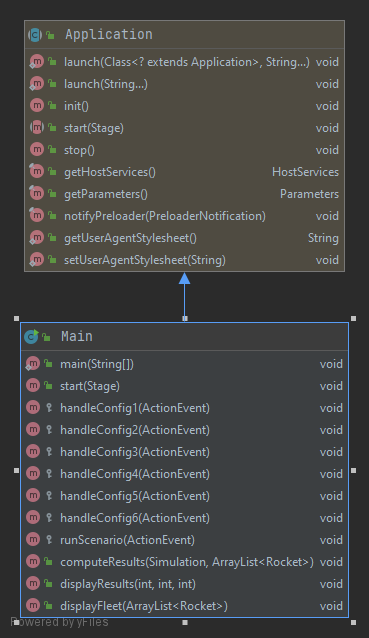
* loadU2Optimum

Là, des rockets U2 sont employées pour les deux phases mais il y a une optimisation sur l’algorithme de chargement. Il est fait de sorte qu'aucune rocket ne soit pleine ; une marge de sécurité de 3 tonnes est laissée. De plus, une rocket transportant des personnes ne transportera que des personnes.

**Classe Main**

C’est la classe qui s’occupe de l’interface graphique et du lancement des simulation à proprement parler. Elle définit les méthodes relatives à l’interface graphique (start, handlers). L’utilisateur choisi le scénario avec lequel il souhaite effectuer la simulation et en fonction de son choix, différentes méthodes sont exécutées. Les choix sont les suivants :





* runScenario

Cette méthode appelle les fonctions de chargement adéquates en fonction du choix de l’utilisateur. Ensuite, elle appelle successivement les méthodes computeResults, displayResults et displayFleet.

* computeResults

Methode qui appelle la méthode runSimulation un certain nombre (nbrSimulation) de fois puis calcule les statistiques qui en découle. Comme statistiques, nous avons retenu les pourcentages de lancements totalement réussis et le pourcentage de fois où il y a eu un seul échec sur le lancement.

* displayResults

Affiche les résultats de la simulation.

* displayFleet

Affiche les rockets nécessaires à la mission, chacune avec le poids d’items qu’elle transporte.

Tests

**Classe RocketTest**

Cette classe test les fonctions de la classe Rocket.

**Classe ExplosionTest**

Cette classe de test a pour but de vérifier que les chances d’explosion sont bien respectées. Elle crée deux rocket U1 et U2, la charge et la lance un certain nombre de fois pour estimer ces chances d’explosion.

Présentation de l’interface utilisateur

Permet à l’utilisateur de pouvoir choisir les différents scénarios dans la barre de menu Configuration.

Affiche les résultats de la simulation à savoir : pourcentage de réussite, pourcentage d’avoir un seul échec et budget en cas de réussite



