Université de Paris

Projet de Programmation - Algorithmes de colonies de fourmis

réalisé par
Idil SAGLAM ;
22015094

16 mai 2022

Table des matières

1 Introduction générale

Ce rapport est le document du travail de Projet de Programmation du Licence 2 Informatique Générale à l'Université de Paris . Le but de ce projet est d'implémenter un algorithme de colonies de fourmis en Java.

2 Etude préalable

Dans ce premier chapitre, nous allons nous intéresser tout d'abord au contexte générale du sujet.

2.1 Cadre général du projet

2.1.1 Objectif

Le but de ce projet est d'implémenter un algorithme de colonies de fourmis.Un algorithme de colonies de fourmis est un algorithme qui comme son nom l'indique, s'inspire du comportement des fourmis, et ce pour le plus souvent résoudre des problèmes d'optimisations. Tout ceci vient du constat suivant : quand des fourmis ont trouvé de la nourriture et doivent la rapporter à leur nid, elles empruntent souvent le chemin le plus court.

• Des fourmis partent en éclaireuses pour trouver de la nourriture, explorant au hasard.
• Si une fourmi trouve de la nourriture, elle rentre au nid selon le trajet qu'elle a suivi. Elle laisse des phéromones sur son trajet.
• Ces phéromones sont très attirantes pour les autres fourmis. Si elles se trouvent a proximité de la piste de phéromones, une autre fourmi qui cherche de la nourriture aura envie de la suivre.
• Au retour, toutes les fourmis renforceront ainsi la piste, en augmentant les phéromones présents sur la piste.
• Si deux chemins permettent d'atteindre la nourriture, celui le plus court sera parcouru par plus de fourmis que le plus long. Ainsi, il sera davantage renforcé.
• Le chemin le plus long finira par disparaître car les phéromones se dissipent.
• Finalement, le système se stabilisera sur une solution optimale (ou plutôt "presque" optimale).

2.2 Solution proposée et travail demandé

Le but est de créer une carte, composée de cellules, qui serait le terrain, dans lequel on pourra simuler l'algorithme de colonies de fourmis. Une des cellules sera le nid d'où partent les fourmis, une autre cellule la nourriture.
Les attendus sont les suivants :
• (Obligatoire) On pourra avant l'exécution de l'algorithme modifier le terrain et rajouter des obstacles, et enregistrer ces cartes.

Solution proposée : L'utilisateur peut modifier la carte, ajouter des obstacles et enregistrer ses cartes sous forme json et utilisateur peut charger une carte présauvegardé depuis un fichier json.

FIGURE 1.1.1 - Choix de l'action

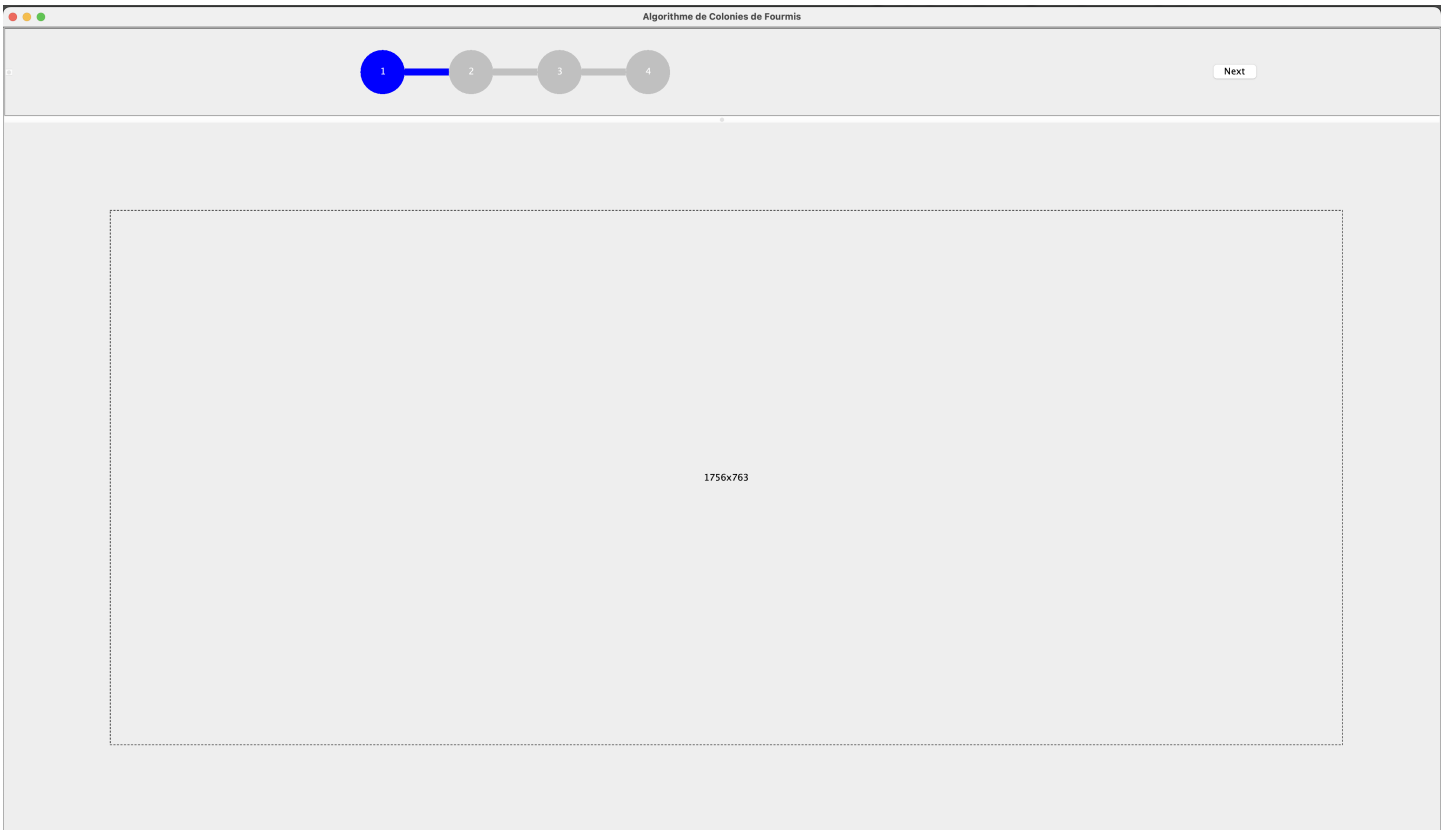


FIGURE 1.1.2 - Sélection la taille du chemin

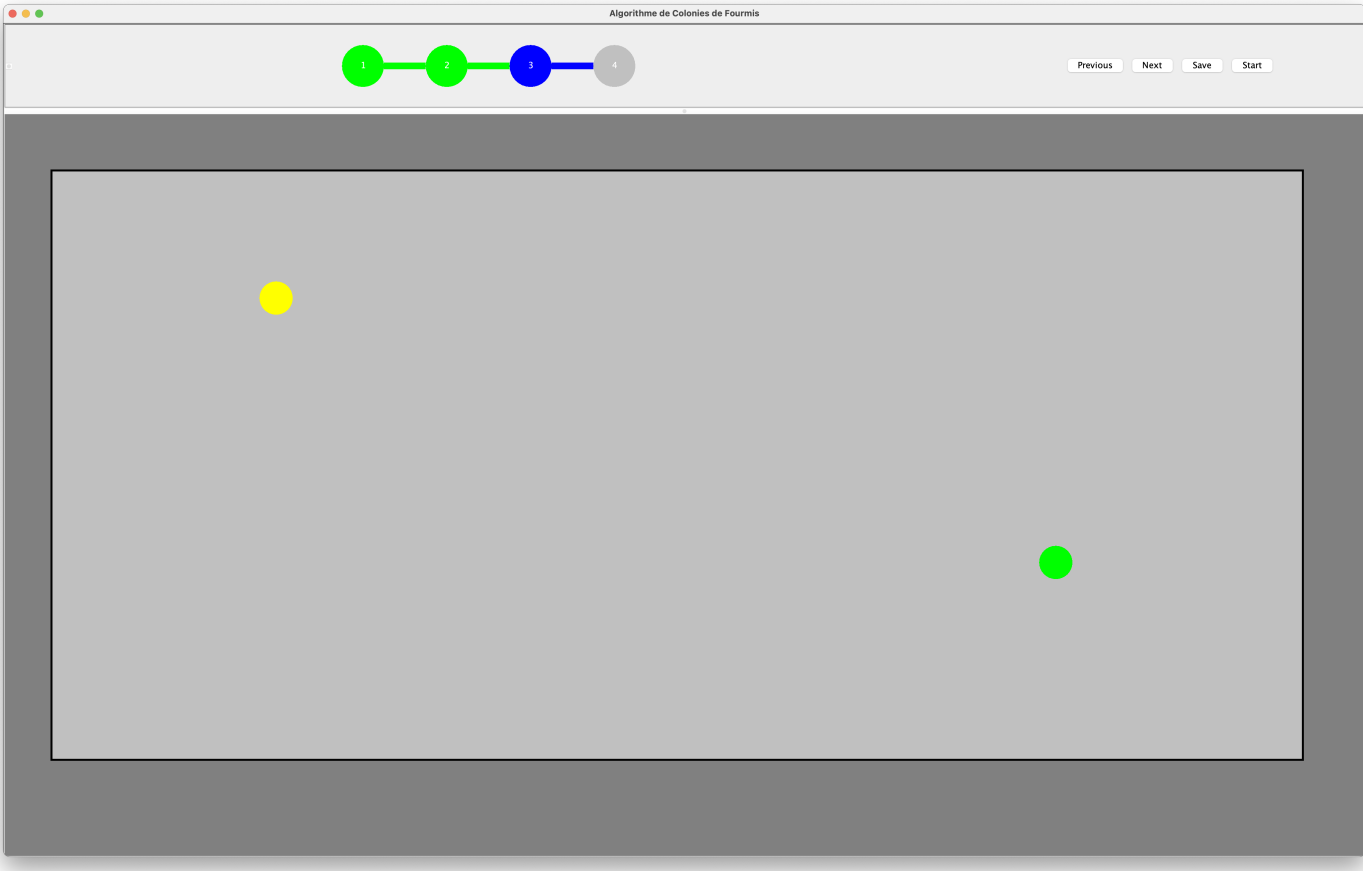


FIGURE 1.1.3 -. Séléction des points entrée et de sortie

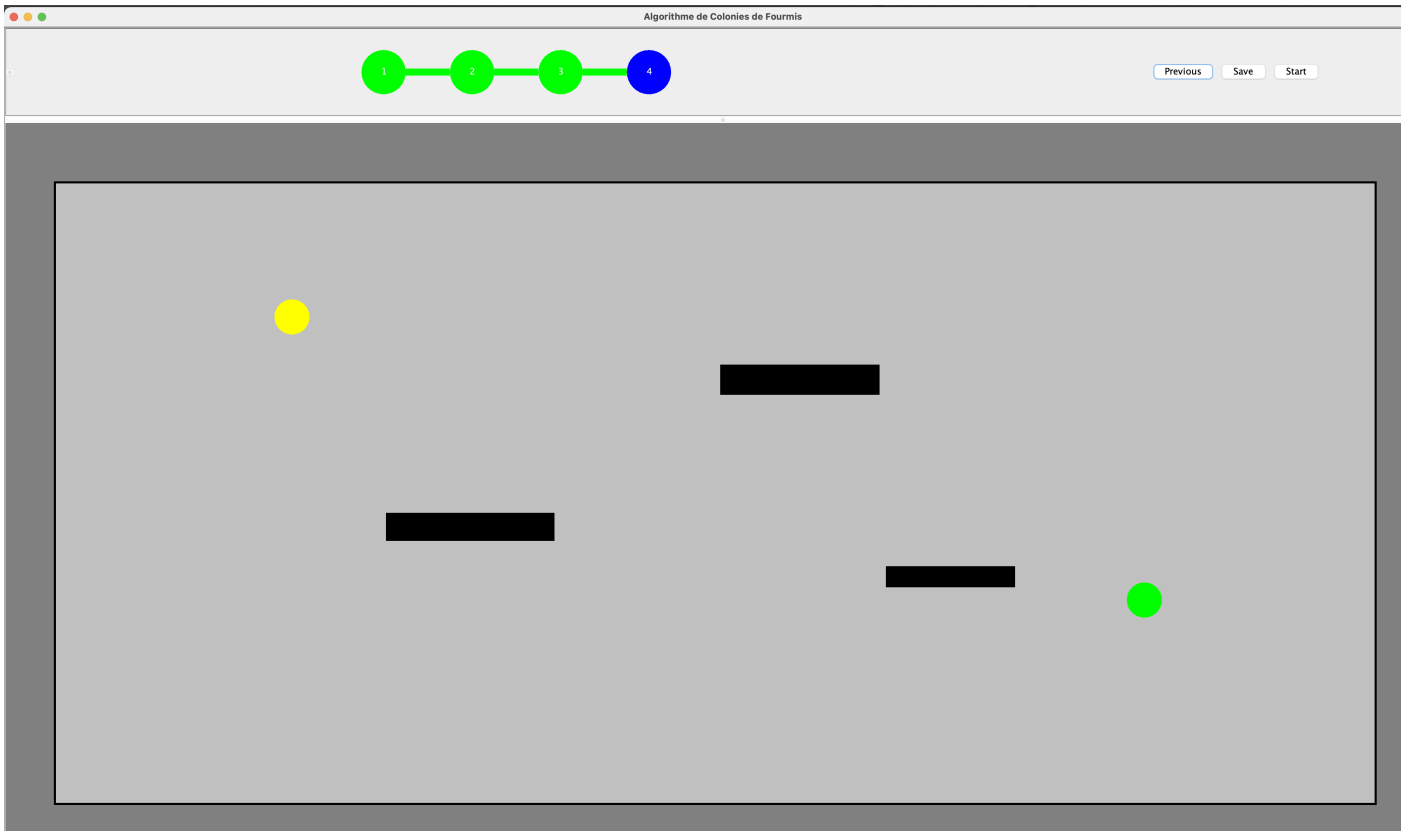


FIGURE 1.1.4 -Ajout des obstacles

• (Obligatoire) Le programme devra utiliser correctement le principe de l'orienté objet.
Solution proposée : Pour garder l'encapsulaion optimale, les classes internes ont implémentées. Le projet est divisé en deux sous modules avec une découpage faible entre les modules.

• (Obligatoire) Une façon de représenter le chemin choisi par les fourmis, et les fourmis.
Solution proposée :

FIGURE 1.1.5 -.Le chemin choisi et les fourmis

• (Souhaitable) Une vision dynamique du programme, qu'on puisse avoir un film de l'exploration, dans lequel on puisse mettre pause, etc. . .
Solution proposée :
Pour mettre pause le programme, pausable thread pool executor est implémentée.

• (Souhaitable) Des informations lors du clic sur les différentes cellules
Solution proposée : Quand on pose le souris sur un élément les informations de ce dernier sera affichée sous forme de tool tip(info-bulle)

2.3 Conclusion

Dans ce chapitre, j'ai présenté le contexte général du projet Ceci nous a permis de comprendre les besoins d'utilisateurs.

Le prochain chapitre est consacré à la présentation des besoins fonctionnels et non fonctionnels.

3 Analyse et spécifiation des besoins

3.1 Capture des Besoins

L'étape de l'analyse des besoins est très importante puisque la réussite de toute application dépend de la qualité de son étude. Il faut donc bien déterminer les fonctions attendues par le système.

3.2 Analyse de besoins

Les besoins sont divisés en deux catégories, à savoir les besoins fonctionnels et les besoins non fonctionnels.

3.2.1 Besoins fonctionnels

Ce sont les actions et les réactions que le système doit faire suite à une demande d'un acteur principal. Tenant compte de la nature de l'application, on distingue les besoins par acteurs :

— **Utilisateur :** L'application doit permettre à l'utilisateur de

1. Créer ou enregistrer une carte
2. Déterminer point d'entrée et point de sortie pour les fourmis
3. Mettre les obstacles
4. Arrêter et continuer la visualisation de l'algorithme.
5. Changer les couleurs, la densité des phérormones, le nombre de fourmis etc.

3.2.2 Besoins non fonctionnels

Les besoins non fonctionnels correspondent à la manipulation de l'application et précisent l'environnement de l'application.

— **L'extensibilité :** L'architecture de l'application permettra l'évolution et la maintenance (l'ajout ou la suppression ou le mise à jour) au niveau de ses différents modules d'une manière fléxible.

— **L'ergonomie et la convivialité :** L'application fournira une interface conviviale et simple à utiliser et qui ne requiert aucun prérequis, donc elle pourra être exploitable par tout type d'utilisateurs (même les non informaticiens)

4 Conception

Dans ce chapitre, je vais aborder la tâche plus importante dans l'élaboration de ce travail, à savoir la tâche de conception. En effet, je présente, en premier lieu, l'architecture générale de mon application afin d'en extraire les différents modules qui la composent.

4.1 K-means clustering

Dans cette projet, j'ai utilisé l'algorithme K-means clustering. J'ai suivi les étapes suivantes :

1. Isoler les phéromones dans la même côté que le point d'arrivée.
2. Calculer les tailles des intervalles pour chaque cluster ((max-min)/nbCluster)
3. Créer les clusters
4. Sélectionner le meilleur cluster : celui avec la valeur intensité totale / distance maximale
5. Répéter jusqu'à il y a un seul cluster non vide (Dans ce cas, il y a soit un seul phéromone dans la cluster ou les phéromones sont très proches)

4.2 Architecture physique

Avant de détailler l'architecture de mon application, il est recommendé d'avoir une vue globale sur les différentes architectures types existantes.

4.2.1 Description des architecture types

Les principales architectures à décrire dans ce paragraphe sont à savoir : **MVC**

MVC

MVC (Model-View-Controller ou Modèle-Vue-Contrôleur) est un modèle dans la conception de logiciels. Il met l'accent sur la séparation entre la logique métier et l'affichage du logiciel. Cette «séparation des préoccupations» permet une meilleure répartition du travail et une maintenance améliorée.

Les 3 parties du modèle de conception de logiciel MVC peuvent être décrites comme suit :

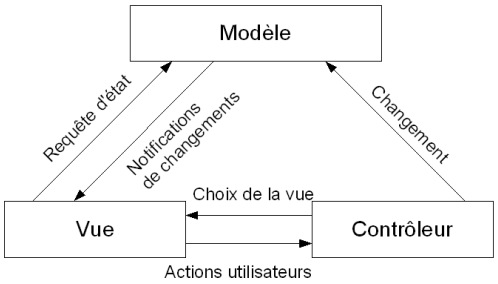
1. **Model (modèle) :** gère les données et la logique métier.
2. **View (vue) :** gère la disposition et l'affichage.
3. **Controller (contrôleur) :** achemine les commandes des parties "model" et "view".

L'avantage d'utiliser MVC

— Diminution de la complexité lors de la conception ;
— Possibilité de réutilisation de code dans d'autres applications ;
— Conception claire et efficace grâce à la séparation des données de la vue et du contrôleur ;
— Plus de facilité pour les tests unitaires.

Désavantage d'utiliser MVC

— Augmentation de la complexité lors de l'implantation ;
— Architecture complexe pour des petits projets ;



5 Conception détaillée

Dans cette partie, je vais entamer la description des détails conceptuels relatifs à Algorithmes de colonies de fourmis Ainsi, je commence par détailler le modèle.

5.1 Conception du modèle

La figure ci-dessous repésente quelques diagramme de classes des services et des templates à la couche modèle du projet.

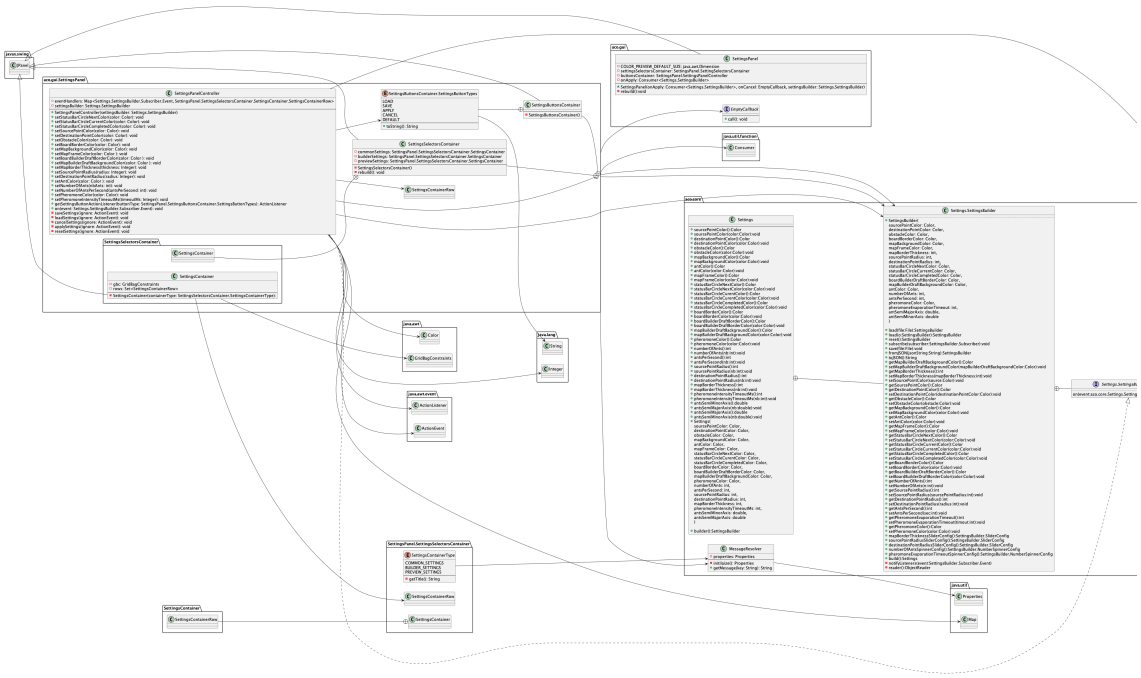


FIGURE : **Le diagramme de la classe du projet**

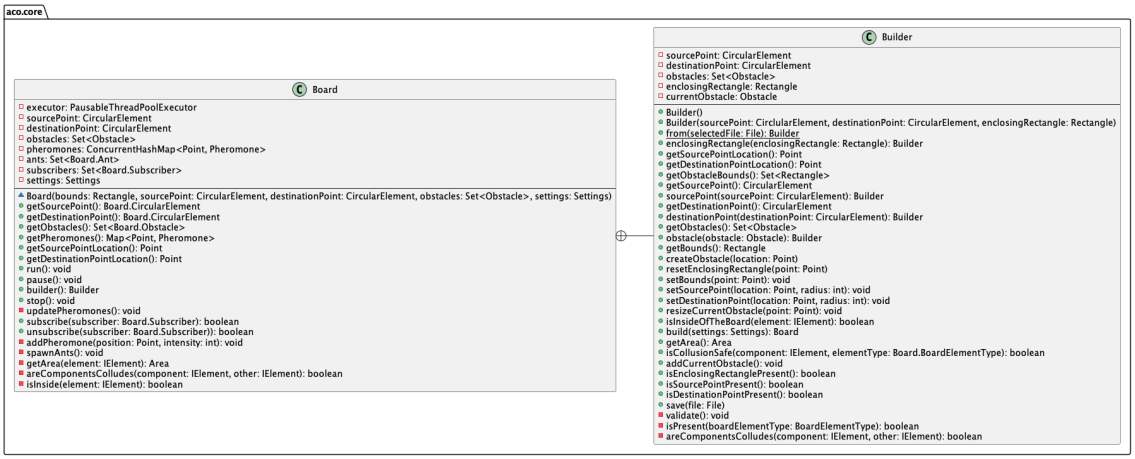


FIGURE : **Le diagramme de la classe Builder**

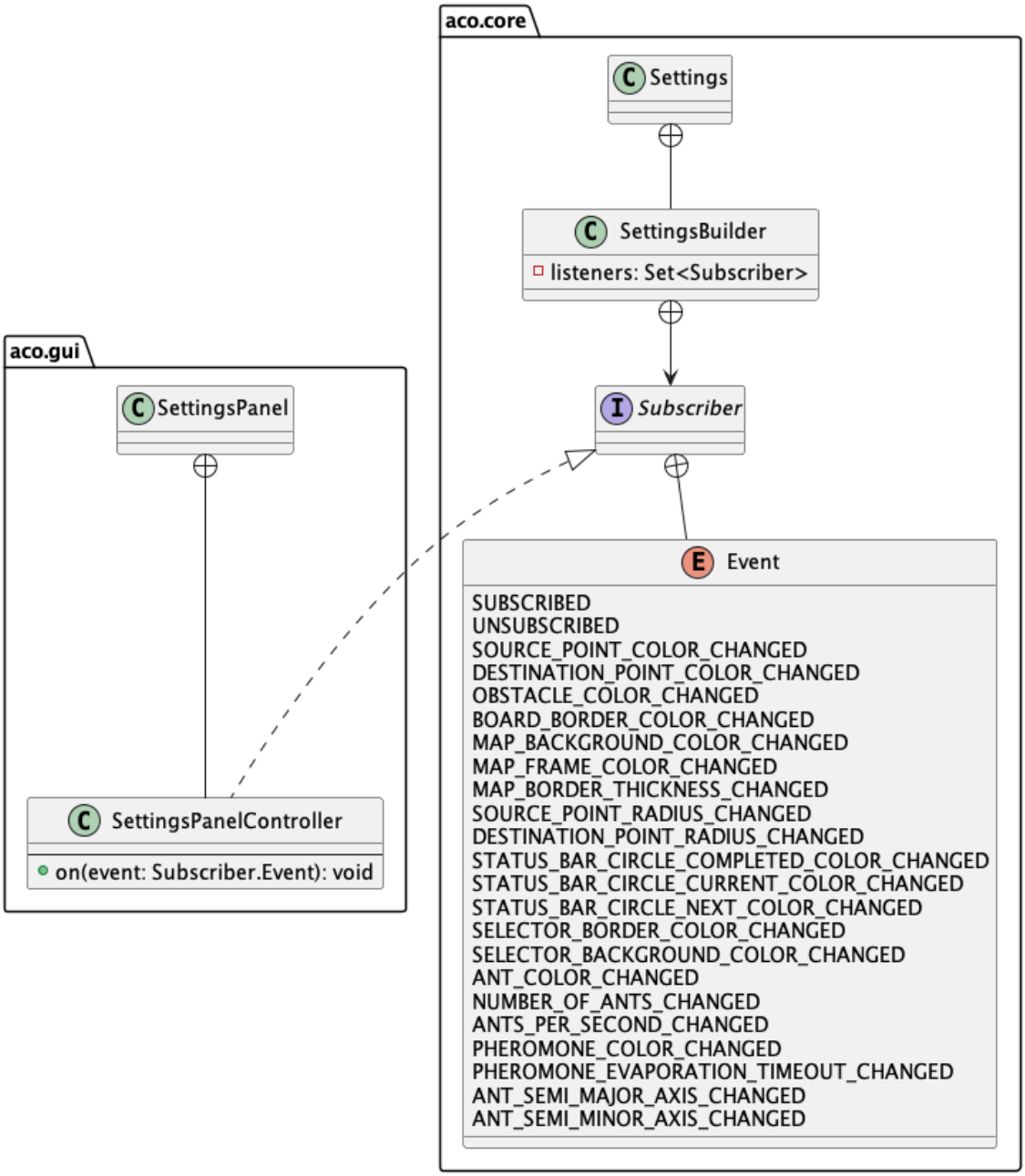


FIGURE : **Le diagramme montrant de patron de conception Observer**

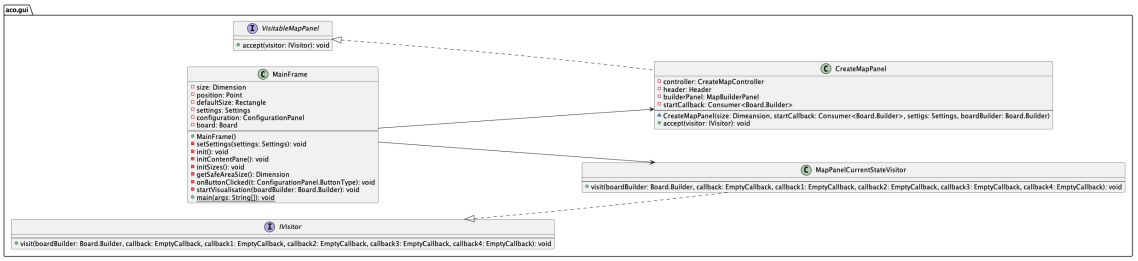


FIGURE : **Le diagramme montrant de patron de conception Visitor**

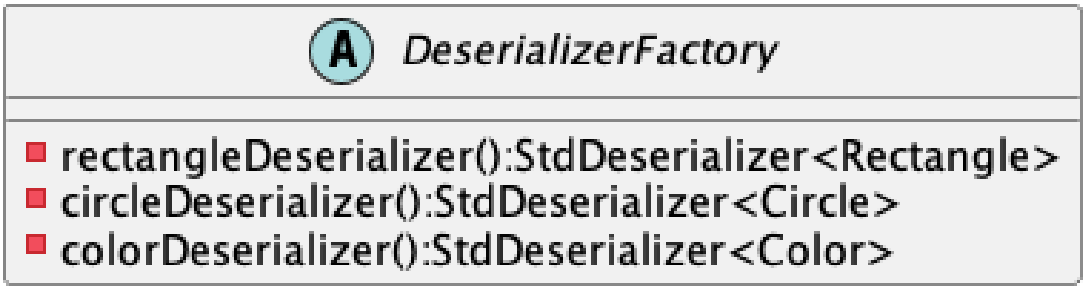


FIGURE : **Le diagramme de classe DeserializerFactory**

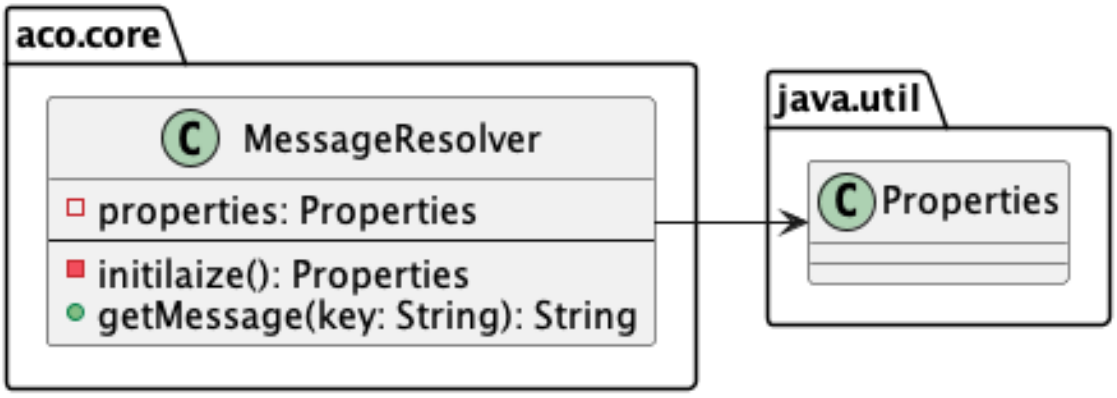


FIGURE : **Le diagramme de classe MessageResolver**

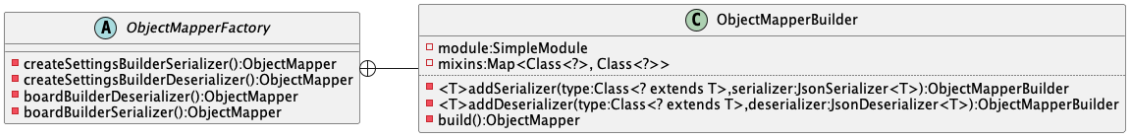


FIGURE : **Le diagramme de classe ObjectMapperFactory**


aco.core
Settings
+ sourcePointColor():Color
+ sourcePointColor(color:Color):void
+ destinationPointColor():Color
+ destinationPointColor(color:Color):void
+ obstacleColor():Color
+ obstacleColor(color:Color):void
+ mapBackgroundColor():Color
+ mapBackgroundColor(color:Color):void
+ antColor():Color
+ antColor(color:Color):void
+ mapFrameColor():Color
+ mapFrameColor(color:Color):void
+ statusBarCircleNextColor():Color
+ statusBarCircleNextColor(color:Color):void
+ statusBarCircleCurentColor():Color
+ statusBarCircleCurentColor(color:Color):void
+ statusBarCircleCompletedColor():Color
+ statusBarCircleCompletedColor(color:Color):void
+ boardBorderColor():Color
+ boardBorderColor(color:Color):void
+ boardBuilderDraftBorderColor():Color
+ boardBuilderDraftBorderColor(color:Color):void
+ mapBuilderDraftBackgroundColor():Color
+ mapBuilderDraftBackgroundColor(color:Color):void
+ pheromoneColor():Color
+ pheromoneColor(color:Color):void
+ numberOfAnts():int
+ numberOfAnts(nb:int):void
+ antsPerSecond():int
+ antsPerSecond(nb:int):void
+ sourcePointRadius():int
+ sourcePointRadius(nb:int):void
+ destinationPointRadius():int
+ destinationPointRadius(nb:int):void
+ mapBorderThickness():int
+ mapBorderThickness(nb:int):void
+ pheromoneIntensityTimeoutMs():int
+ pheromoneIntensityTimeoutMs(nb:int):void
+ antsSemiMinorAxis():double
+ antsSemiMajorAxis(nb:double):void
+ antsSemiMajorAxis():double
+ antsSemiMinorAxis(nb:double):void
+ Settings(
sourcePointColor: Color,
destinationPointColor: Color,
obstacleColor: Color,
mapBackgroundColor: Color,
antColor: Color,
mapFrameColor: Color,
statusBarCircleNextColor: Color,
statusBarCircleCurentColor: Color,
statusBarCircleCompletedColor: Color,
boardBorderColor: Color,
boardBuilderDraftBorderColor: Color,
mapBuilderDraftBackgroundColor: Color,
pheromoneColor: Color,
numberOfAnts: int,
antsPerSecond: int,
sourcePointRadius: int,
destinationPointRadius: int,
mapBorderThickness: int,
pheromoneIntensityTimeoutMs: int,
antsSemiMinorAxis: double,
antsSemiMajorAxis: double
)
+ builder():SettingsBuilder
Settings.SettingsBuilder
+ SettingsBuilder(
sourcePointColor: Color,
destinationPointColor: Color,
obstacleColor: Color,
boardBorderColor: Color,
mapBackgroundColor: Color,
mapFrameColor: Color,
mapBorderThickness: int,
sourcePointRadius: int,
destinationPointRadius: int,
statusBarCircleNextColor: Color,
statusBarCircleCurrentColor: Color,
statusBarCircleCompletedColor: Color,
boardBuilderDraftBorderColor: Color,
mapBuilderDraftBackgroundColor: Color,
antColor: Color,
numberOfAnts: int,
antsPerSecond: int,
pheromoneColor: Color,
pheromoneEvaporationTimeout: int,
antSemiMajorAxis: double,
antSemiMinorAxis: double
)
+ load(file:File):SettingsBuilder
+ load(o:SettingsBuilder):SettingsBuilder
+ reset():SettingsBuilder
+ subscribe(subscriber:SettingsBuilder.Subscriber):void
+ save(file:File):void
+ fromJSON(jsonString:String):SettingsBuilder
+ toJSON():String
+ getMapBuilderDraftBackgroundColor():Color
+ setMapBuilderDraftBackgroundColor(mapBuilderDraftBackgroundColor:Color):void
+ getMapBorderThickness():int
+ setMapBorderThickness(mapBorderThickness:int):void
+ setSourcePointColor(source:Color):void
+ getSourcePointColor():Color
+ getDestinationPointColor():Color
+ setDestinationPointColor(destinationPointColor:Color):void
+ getObstacleColor():Color
+ setObstacleColor(obstacle:Color):void
+ getMapBackgroundColor():Color
+ setMapBackgroundColor(color:Color):void
+ getAntColor():Color
+ setAntColor(color:Color):void
+ getMapFrameColor():Color
+ setMapFrameColor(color:Color):void
+ getStatusBarCircleNextColor():Color
+ setStatusBarCircleNextColor(color:Color):void
+ getStatusBarCircleCurrentColor():Color
+ setStatusBarCircleCurrentColor(color:Color):void
+ getStatusBarCircleCompletedColor():Color
+ setStatusBarCircleCompletedColor(color:Color):void
+ getBoardBorderColor():Color
+ setBoardBorderColor(color:Color):void
+ getBoardBuilderDraftBorderColor():Color
+ setBoardBuilderDraftBorderColor(color:Color):void
+ getNumberOfAnts():int
+ setNumberOfAnts(n:int):void
+ getSourcePointRadius():int
+ setSourcePointRadius(sourcePointRadius:int):void
+ getDestinationPointRadius():int
+ setDestinationPointRadius(radius:int):void
+ getAntsPerSecond():int
+ setAntsPerSecond(sec:int):void
+ getPheromoneEvaporationTimeout():int
+ setPheromoneEvaporationTimeout(timout:int):void
+ getPheromoneColor():Color
+ setPheromoneColor(color:Color):void
+ mapBorderThicknessSliderConfig():SettingsBuilder.SliderConfig
+ sourcePointRadiusSliderConfig():SettingsBuilder.SliderConfig
+ destinationPointRadiusSliderConfig():SettingsBuilder.SliderConfig
+ numberOfAntsSpinnerConfig():SettingsBuilder.NumberSpinnerConfig
+ pheromoneEvaporationTimeoutSpinnerConfig():SettingsBuilder.NumberSpinnerConfig
+ build():Settings
- notifyListeners(event:SettingsBuilder.Subscriber.Event)
- reader():ObjectReader
Settings.SettingsBuilder.Subscriber
on(event:aco.core.Settings.SettingsBuilder.Subscriber.Event):
Settings.SettingsBuilder.Subscriber.Event
SUBSCRIBED,
UNSUBSCRIBED,
SOURCE_POINT_COLOR_CHANGED,
DESTINATION_POINT_COLOR_CHANGED,
OBSTACLE_COLOR_CHANGED,
BOARD_BORDER_COLOR_CHANGED,
MAP_BACKGROUND_COLOR_CHANGED,
MAP_FRAME_COLOR_CHANGED,
MAP_BORDER_THICKNESS_CHANGED,
SOURCE_POINT_RADIUS_CHANGED,
DESTINATION_POINT_RADIUS_CHANGED,
STATUS_BAR_CIRCLE_COMPLETED_COLOR_CHANGED,
STATUS_BAR_CIRCLE_CURRENT_COLOR_CHANGED,
STATUS_BAR_CIRCLE_NEXT_COLOR_CHANGED,
SELECTOR_BORDER_COLOR_CHANGED,
SELECTOR_BACKGROUND_COLOR_CHANGED,
ANT_COLOR_CHANGED,
NUMBER_OF_ANTS_CHANGED,
ANTS_PER_SECOND_CHANGED,
PHEROMONE_COLOR_CHANGED,
PHEROMONE_EVAPORATION_TIMEOUT_CHANGED,
ANT_SEMI_MAJOR_AXIS_CHANGED,
ANT_SEMI_MINOR_AXIS_CHANGED;

FIGURE : **Le diagramme de classe Settings**

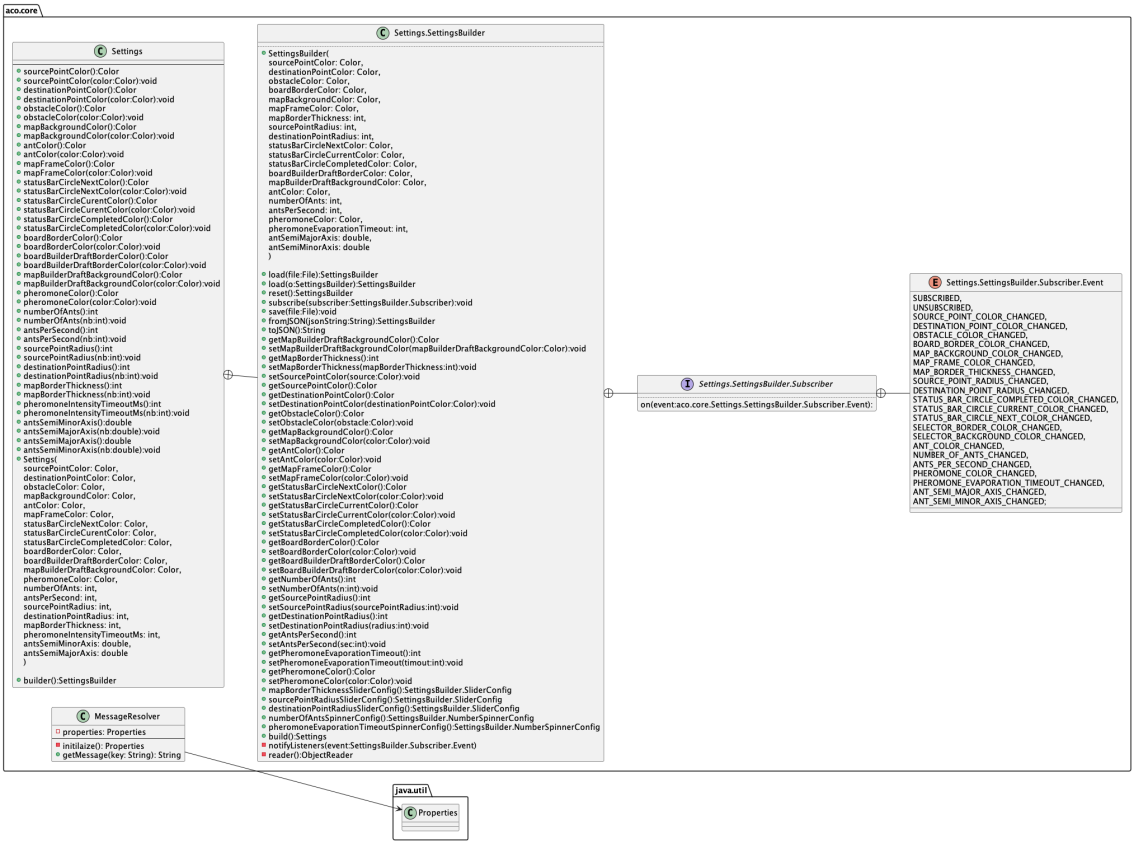


FIGURE : **Le diagramme du module core**

6 Conclusion générale et perspectives

Dans le présent rapport, j'ai détaillé les étapes par lesquelles j'ai passé pour concevoir et développer ma solution. Pour aboutir à ce résultat, j'ai tout d'abord commencé par présenter le cadre générale de mon travail. Puis, j'ai présenté les différents besoins et les exigences relevées. Ensuite, j'ai abordé la phase de conception qui nous a expliqué l'architecture de l'application. Finalement, l'étape de réalisation, au cours de laquelle j'ai présenté mon application.

Durant ce projet, j'ai été confrontés à plusiseurs problèmes et obstacles au niveau de développement et au niveau d'organisation. En effet, ce projet devrait être réalisé en groupe de 5 mais mon groupe m'a exclus.

Comme perspectives des travaux futurs, je propose d'enrichir cette program en s'intéressant à certains points. Pour étentre ma solution, je peux l'encrichir par d'autres fonctionnalités comme ajouter des nouveux algorithmes au programme.