# 

# Programmation orientée objet

## Compte rendu



Jérôme ALIBERT  
Sébastien RIGAUD

Enseignant : Baudouin Dafflon

Sommaire

[Modification du Code 3](#_Toc472436192)

[Ajout 3](#_Toc472436193)

[Suppression 3](#_Toc472436194)

[Modification 3](#_Toc472436195)

[TODO 3](#_Toc472436196)

# Modification du Code

## Ajout

AbstractAgent.java : Créer une classe abstraite d’agent permettant au code d’évoluer et de créer une fabrique d’agent abstrait.

AgentFactory.java : Création d’une fabrique d’agent permettant de rompre les liens entre ces agents et les classes qui les utilises.

AgentsInteractionFactory.java : Création d’une fabrique d’interaction permettant de rompre les liens entre ces interactions et les classes qui les utilises.

BehavioursFactory.java : Création d’une fabrique de comportement permettant de rompre les liens entre ces comportements et les agents qui les utilises.

RoleFactory.java : Création d’une fabrique de rôle permettant de rompre les liens entre les rôles et les classes qui les utilises.

Rolable.java : Déplacement des comportements des rôles dans un interface permettant d’imposer des comportements aux classes l’implémentant.

AgentWorkable.java : Déplacement des comportements des agents dans un interface permettant d’imposer des comportements aux classes l’implémentant.

Interactable.java : Déplacement des comportements des interactions dans un interface permettant d’imposer des comportements aux classes l’implémentant.

Classes filles d’AbstractRole.java

* + EaterRole.java
  + HeavyHotBodyRole.java
  + MusicalRole.java
  + PainterRole.java
  + SunRole.java

## Suppression

Agent.java : Suppression de la méthode « remove » non utilisée

Spinner.java : Cette classe n’est jamais utilisé et a donc été supprimée

## Modification

AbstractRole.java : Renommer la classe Role.java en AbstractRole.java permettant de l’utiliser dans une fabrique et une meilleure maintenance d’application de code.

Agent.java : Classe fille de AbstractAgent.java et implémentation de l’interface AgentWorkable.java.

DrawPanel.java (fonction paintAgentsInformation) : suppression de la duplication de code (les catch répétitif). Optimisation des exceptions, toutes les exceptions attrapées afficher leur stack trace, utilisation de la classe mère Exception à la place.

# TODO

AbstractRole.java : Supprimer les noms pour différentier les rôles et utiliser les classes filles déjà créées.

Déplacer de nombreux comportement dans des interfaces pour créer des « contrats » avec les classes les implémentant. Ceci permettra d’optimiser le code et de facilité les évolutions.