MODELISATION SIMWARE – BALJ17058609 – BALDO JEROME

Table des matières

PARTIE 1 – IDENTIFICATION PATRONS GRASP

1. simwar.core.GestionnaireIntentions

Cette classe utilise indirection et fabrique pure. En effet, il y a un découplage de la responsabilité sur les intentions permet d'alléger le controleur simWar. Ce découplage est accentué par l'ajout de la fabrique pure. En effet c'est une délégation des intentions dans une autre classe. Les intentions sont gérées séparément.

2. simwar.core.SimWar

Pour cette classe, il y a une utilisation du contrôleur. En effet SIMWAR est la pièce centrale qui déclenche toutes les autres classes. En centralisant au niveau du SimWar, le déroulé des événements on obtient une meilleure maitrise sur le processus mais aussi une plus grande clarté.

3. simwar.core.environnement.Environnement

Pour cette classe, il y a une utilisation du patron GRASP fabrique pure et expert. En effet, Environnement est une classe abstraite qui évite l'utilisation des « if » pour cibler le bon environnement (comportement différent). D'autre part, l'avantage de l'utilisation de l'expert est de permettre de concentrer la responsabilité sur environnement pour la génération des obstacles. Ainsi gestionnaireEnvironnement est soulagé de cette responsabilité. Il y a plus de clarté dans l'architecture.

4. simwar.operation.OperationImplMouvement

C'est le patron polymorphisme qui est utilisé pour gérer les différentes actions. Grâce à cela il n'y a pas de if pour créer les bons objets surtout au vu de la quantité à disposition (tous les héritages liés à OperationImplCombat et OperationImplMouvement).

5. simwar.operation.Radar

L'intérêt d'utiliser le patron expert est qu'il émet à la demande. La logique de localiser les véhicules est encapsulé dans le radar et non dans SimWarVehicule.

6. simwar.vehicule.Intention

C'est un contrôleur qui permet de configurer les caractéristiques des véhicules, des armes et des actions à faire. Il permet de séparer le contrôle du véhicule de la configuration.

7. simwar.vehicule.SIMWarVehicule

Cette classe est à la fois un patron forte cohésion et un expert. En effet il possède toutes les fonctionnalités propres à sa classe. Par exemple il sait quelles sont ses armes (getArme()) mais aussi il a la responsabilité d'appeler le radar pour avoir les positions. Ainsi il rassemble un maximum de fonctionnalités propres à sa classe (patron forte cohésion) mais aussi il est en mesure de donner ses informations ou de se situer (patron expert)

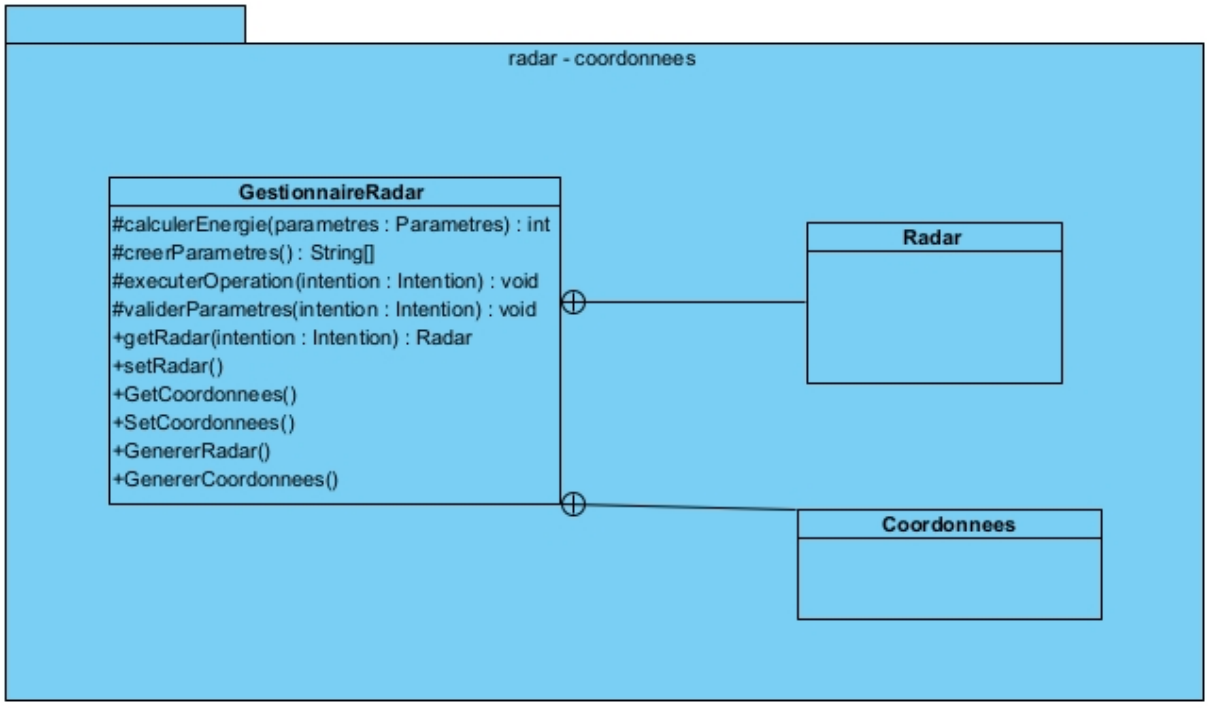
PARTIE 2 – IDENTIFICATIONS

1. Identification de deux erreurs

Identifier deux erreurs de conception dans le projet (pas de programmation). Fournir une solution via UML et un court texte explicatif.

Pour cette partie, mon objectif a été d'identifier les classes ou package n'utilisant pas au maximum le faible couplage et la forte cohésion. J'en ai donc identifié deux erreurs :

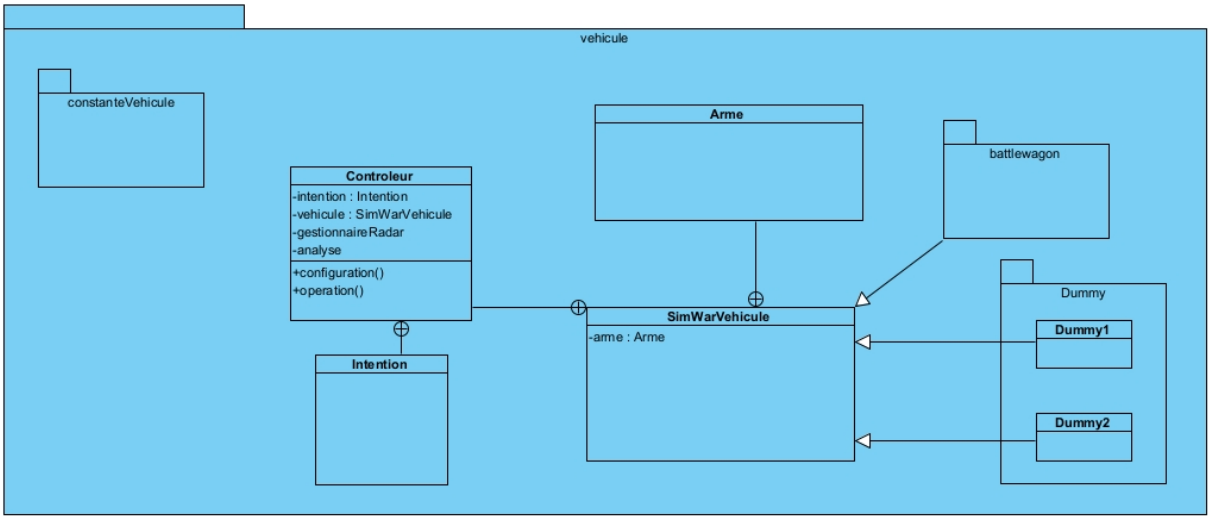
a) Première erreur – Radar



La classe Radar est mis dans le package Operation. Les méthodes qui lui sont liés, sont dispatchés dans le package où dans d'autres. L'idéal serait d'isoler Radar dans un package. Celui-ci sera constitué d'un GestionnaireRadar, du Radar mais aussi de la classe Coordonnees. GestionnaireRadar serait l'indirection tandis que Coordonnees et Radar seraient toujours des experts. Toutes les classes souhaitant interagir avec Radar ou Coordonnees devront passé par le gestionnaire. Celui-ci déclenchera des méthodes qui interagiront avec Radar et Coordonnees. GestionnaireRadar devrait reprendre les autres méthodes.

b) Deuxième erreur – le manque d'un contrôleur pour le package vehicule

Le package vehicule ne contient pas de contrôleur définir les paramètres. Il serait judicieux d'en créer un pour alléger la classe SimWarVehicule mais aussi détacher la logique de paramétrages de l'objet en lui-même (qui sera alors totalement un expert). L'intérêt est aussi d'isoler les 2 phases Opération et OperationImpl. Le controleur possèdera un SimWarVehicule mais aussi un objet Intention (je n'ai pas pu le signifier sur l'UML). Controleur reprendra toutes les méthodes de configuration, paramétrage et changement. SimWarVehicule quant à lui, n'aura que les assesseurs et les mutateurs.



2. Identification de la section complexe

La section la plus complexe est celle de vehicule. En effet ce package possède des dépendances dans tous les autres packages. La classe centrale de ce package est SimWarVehicule. Celle-ci est un expert mais qui comprend la gestion des intentions mais aussi des armes, de la position sur le radar. De plus cette classe appelle les opérations et l'implémentation des opérations.

3. Identification de la section la mieux conçue

Pour moi le package sur le core est le mieux conçu de l'application. En effet on peut déjà constater une grande clarté dans l'architecture. Cela est dû à un faible couplage et une forte cohésion dans la structure. L'élément central est la classe SimWar. Celui-ci est de type contrôleur. Il interagit avec les gestionnaires qui sont des indirections et des experts (d'ailleurs ils portent tous le mot gestionnaire dans le nom de classe). SimWar va déclencher les phases de jeux successives et appeler les différents gestionnaires. Par ailleurs comme évoqué dans la partie 1, le gestionnaireEnvironnement va appeler une fabrique pure (Environnement).

Cependant on peut noter que coordonnées n'a pas sa place dans ce package. Il devrait être avec Radar.