Rapport du Projet

Tout d’abords nous avions mis du temps pour bien lire et comprendre l’objectif de ce projet ainsi que le code de base fournis afin de savoir ce que nous allons faire plus tard.

Le code initial contenait de nombreuses responsabilités au sein d'une même classe (FirefighterBoard), ce qui violait plusieurs principes SOLID, notamment la responsabilité unique et l'ouverture à l'extension. Ce rapport détaille les étapes de refactorisation entreprises pour améliorer la conception en respectant les principes SOLID, avec des interfaces et des classes dédiées pour chaque fonction spécifique.

L'objectif était de diviser les responsabilités afin de :

1. Respecter les principes SOLID et faciliter la maintenabilité.
2. Rendre le système plus modulaire et extensible.
3. Faciliter la testabilité en isolant les comportements en petites classes.

Afin de commencer avec ce TP on a créé plusieurs classes et deux interfaces afin de simplifier le code et de faciliter la modification et l’implémentation plus tard notamment avec les interfaces qui utilisent le patron de conception Strategy.

Pour la gestion des incendies on a créé une classe FireManager qui est responsable de la gestion des positions des incendies, la propagation des incendies sur le plateau et enfin l’extinction des incendies à des positions spécifiques.

De plus cette classe utilise le code de la propagation des incendies de la Startegy FireSpreadStrategy une interface que nous avions créée, ce qui nous permet de modifier plus tard la manière dont l’incendie se propage simplement en ajoutant simplement de nouvelles classes qui implémentent cette interface Strategy sans avoir à modifier FireManager.

La gestion des pompiers a été extraite dans une classe distincte appelée FirefighterManager. Cette classe est responsable de :

1. La gestion des positions des pompiers.
2. Le déplacement des pompiers vers les incendies.
3. L’extinction des incendies à proximité des pompiers.

Pour gérer le déplacement des pompiers vers les incendies, une stratégie (FirefighterMovementStrategy) a été introduite cette stratégie comme celle de l’incendie est aussi une interface. Cela permet de facilement ajuster le comportement de mouvement des pompiers en changeant la stratégie.

Les deux interfaces donc sont conforment aux principes OCP mais aussi LSP. (Chadi Mansour)

On a enfin ajouté une nouvelle classe NeighborManager pour centraliser et simplifier la logique de gestion des voisins dans la grille. Elle fournit une méthode pour récupérer les voisins de toute position donnée, ce qui était auparavant dispersé dans le code de FirefighterBoard. Cette classe stocke les voisins de chaque position et permet un accès rapide aux positions adjacentes. La classe est instanciée avec la grille et calcule les voisins une seule fois, optimisant ainsi les performances pour une grille de taille fixe. Cela nous permets de mettre la logique de calcule des voisins dans une seule classe et si besoin de changer cette logique il nous faudrait juste changer la classe NeighborManager et rien d’autre.

Apres tous les changements qu’on a faits on a refactorisé la classe FirefighterBoard en enlève les parties qui ont été substitué par les nouvelles classes et respecte mieux les principes SOLID.

Ces améliorations rendent le projet prêt pour une éventuelle extension et facilitent également l’écriture de tests unitaires.

Dans une prochaine étape nous allons tout d’abord créé des classes tests pour vérifier que tout se comporte comme on le souhaite et puis nous allons commencer à étendre le modèle avec les fonctionnalités de la tache 2 du projet.

Chadi Mansour et Chahine Rami