Exercice : Conception avec DDD

Exercice 1 – DDD : base

On souhaite réaliser un mini logiciel de jeu de simulation d'une ville simplifiée. La ville est composée des quartiers de quatre types : 1) Quartiers résidentiels contenant les habitants de la ville. Ce type de quartier a une densité de population variable ; 2) Quartiers commerciaux contenant les entreprises. Ce type de quartier a une densité d'entreprises variable ; 3) Quartiers de services contenant les services publics. Ce type de quartier a une couverture fixe : nombre de quartiers résidentiels voisons servis ; 4) Quartiers de divertissement contenant les lieux de divertissement. Ce type de quartier a niveau de capacité variable : nombre d'habitants pouvant en profiter.

Pour simplifier, chaque quartier est « atomique », c'est-à-dire que l'on n'a pas besoin d'y construire concrètement des bâtiments : on n'a qu'à gérer le quartier « en entier ». Les habitants des quartiers résidentiels se rendent au travail dans les quartiers commerciaux tous les jours dans la semaine. Il y a aussi les habitants travaillant dans les services publics et les lieux de divertissement qui se rendent ainsi au travail dans ces deux types de quartiers. De plus, tous les habitants vont dans les quartiers de divertissement, une fois dans la semaine et une fois chaque week-end.

Tous les déplacements dans la ville se font par métro ou à pieds (sans besoin de route). On ne peut avoir maximum qu'une seule station de métro dans un quartier. L'ensemble du quartier est donc desservi par cette station. S'il n'y a pas de station de métro dans un quartier, les habitants (ou travailleurs) du quartier peuvent emprunter la station de métro dans un quartier voisin et aller dans ce quartier à pieds sans besoin de route. Mais cela aura deux conséquences : 1) un temps de trajet supplémentaire 2) la surcharge de la station du quartier voisin.

Le joueur construit les quartiers, d'y place les stations de métro, et forme les lignes de métro en reliant les stations. On peut avoir des stations partagées par deux ou plusieurs lignes dans les quartiers de haute densité. La construction et le fonctionnement des stations et lignes de métro coûtent de l'argent. La construction initiale de tous les types de quartiers coûte de l'argent. Le fonctionnement des quartiers de services coûte également de l'argent. Le fonctionnement des autres types de quartiers ne coûte rien.

Les quartiers résidentiels, commerciaux et de divertissement produisent des impôts dont le montant dépend de la densité ou niveau du quartier. En revanche, la prospérité (augmentation de densité) des quartiers résidentiels et commerciaux est automatique (le joueur ne peut pas le contrôler) et dépend de l'accessibilité par le métro, du temps moyen de transport quotidien des habitants, et de la satisfaction des habitants (couverts par les services et les divertissements), etc. Vous pouvez proposer d'autres facteurs / mesures de satisfaction. En revanche, le joueur peut augmenter le niveau de capacité des quartiers de divertissement et cela coûte bien sûr de l'argent. Le joueur essaie de trouver un équilibre financier en ayant une prospérité maximale de l'ensemble de la ville.

Vous travaillez pour produire les éléments suivants :

- ✓ Une description des termes importants du sujet (langage ubiquitaire)
- ✓ Une description des règles de fonctionnement du système à développer (langage ubiquitaire)
- ✓ Un ensemble de classes correspondant aux termes qui sera corrigé en séance par l'enseignant
- ✓ Catégoriser les classes (avec un peu d'explication pour justifier votre solution)
  - Les classes entités
  - Les classes objets de valeur
  - Les différents agrégats dont pour chacun
    - La liste des classes à l'intérieur de la frontière
    - L'entité racine de l'agrégat

Exercice 2 – DDD : notions avancées

Reprenez la solution basique de conception proposée par l'enseignant. Enrichissez et détaillez la solution en considérant les aspects suivants :

Services : Réfléchissez aux fonctionnalités indispensables qui permettent au fonctionnement du système. Ces sont les activités (traitements) sur les objets (entité et objet de valeur) à réaliser dans les classes isolées.

Factories : Quelles sont les créations d'objets nécessaires ? Avec quelle mise en œuvre : simple factory ou builder ?

Repositories : Pour récupérer les objets créées, quelles sont les classes du type « respository » nécessaires ? Quelles sont les méthodes de recherche ?

Méthodologies de travail :

Ordre de conception des éléments détaillés :

1) D'abord, déterminer les nouvelles classes à ajouter ;

2) Ensuite, définir les liens entre les classes ;

3) Mentionnez les méthodes fonctionnelles dans les nouvelles classes et celles existantes ;

4) Définir les attributs des classes.

Vous pouvez aussi réaliser 3) et 4) en même temps. **Pas besoin** d'indiquer les méthodes getters/setters/toString et les constructeurs.

Vous portez une attention particulière sur la **navigabilité** entre les objets (associations) dans votre conception.