ECOLE CENTRALE DE NANTES

SMP2 - TP4

Le but de ce TP est de créer un ensemble de classes permettant à une communauté de communes de gérer un PLU (Plan Local d'Urbanisme) simplifié et plus particulièrement lui permettant d'implanter un nouveau SI pour gérer son cadastre de manière adéquate en incluant les nouvelles directives du PLU.

1 PRÉAMBULE - CRÉATION DU REPOSITORY

Pour faire les groupes pour le TP noté vous devez suivre le lien donné sur hippocampus.

Donnez à votre groupe le nom des différents membres séparés par des - (NOM1-NOM2-NOM3-NOM4). Créez un fichier README.md qui comprend à minima les noms des 4 membres du groupe et importez ce projet comme un projet C++ sous repl.it.
Vous rendrez le code via github et les diagrammes de classes, descriptions des choix de conception, algorithmes et jeux d'essais via codepost.io.
Les classes à coder sont décrites ci-après.

2 CLASSE POINT2D

Cette classe doit représenter un point dans un plan cartésien à deux dimensions. La figure 3.1 décrit les éléments attendus dans cette classe. Vous surchargerez l'opérateur << pour permettre l'affichage d'un `Point2D` à l'écran.

3 CLASSE POLYGONE

Un Polygone est composé l'une liste de sommets qui sont des `Point2D`. La figure 3.1 décrit les éléments attendus dans cette classe. Vous surchargerez l'opérateur << pour permettre

l'affichage d'un Polygone à l'écran.

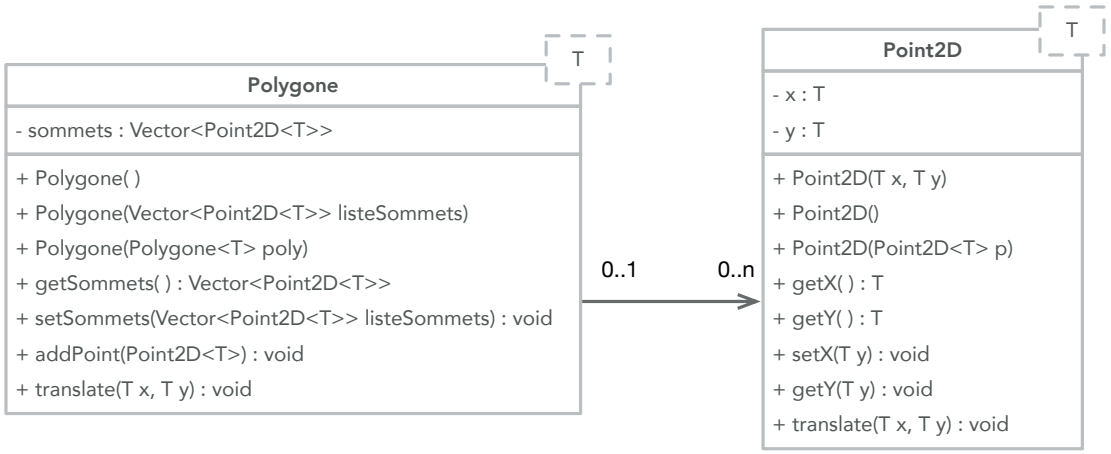

FIGURE 3.1 – Diagramme des classes Point2D et Polygone

Montrer par des jeux d'essais la création par recopie et la translation de `Polygone` de votre choix. Si un polygone est translaté, les Polygones construits par recopie ne doivent pas être translatés.

3.1 CLASSE PARCELLE

Une Parcelle est caractérisée par :

- Un type (string) qui sera défini dans la partie suivante.
- Un numéro (entier)
- Un propriétaire (chaîne de caractères)
- Une surface en m2(réel)
- Un pourcentage de surface constructible de la parcelle (entier)
- Une forme définie par un Polygone<int, float>

La figure 3.2 décrit les éléments attendus dans cette classe. La classe Parcelle est abstraite car elle possède la méthode `setType()` qui est virtuelle pure.
Vous surchargerez l'opérateur << pour permettre l'affichage des informations d'une parcelle à l'écran.
On va supposer dans un premier temps les formes des parcelles données par des polygones non croisés. Il faudra lever une exception si le polygone décrivant la parcelle est croisé, mais ce n'est pas demandé pour le moment. En revanche vous devez lever des exceptions si la surface d'une parcelle est négative ou nulle.
La surface d'un polygone dont les coordonnées sont entières ou réelles est calculée à partir de la liste des coordonnées des points constituant les sommets du Polygone **dans le sens trigonométrique** à partir de la formule :

$$surface = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)$$

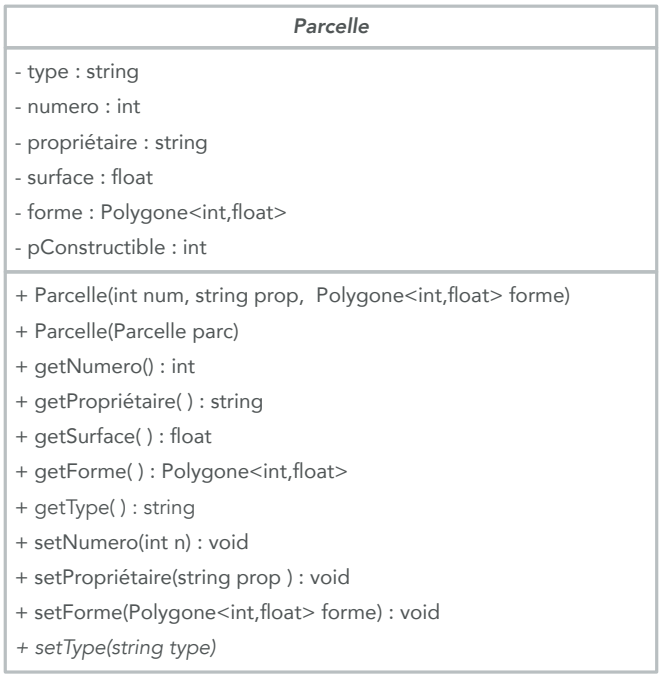

FIGURE 3.2 – Diagramme de classe de la classe Parcelle

avec $\{(x_n, y_n)\}_{i=0^{n-1}}$ les sommets consécutifs du polygone et $(x_n, y_n) = (x_0, y_0)$.
C'est ainsi à la construction d'un Polygone vous devrez calculer la valeur à affecter à l'attribut `surface`. La surface doit aussi être modifiée si la forme de la Parcelle change.

4 TYPES DE PARCELLES

Le PLU n'a pas qu'une sorte de Parcelle et prévoit de découper les parcelles de son cadastre suivants les règles suivantes :

- Zones Urbaines (ZU),
- Zones à Urbaniser (ZAU),
- Zones Agricoles (ZA),
- Zones Naturelles et Forestières (ZN).

De plus vous créerez une Classe abstraite `Constructible` avec une fonction virtuelle pure `surfaceConstructible`

- Les ZU et ZAU sont constructibles.
- Les ZU ont comme caractéristique supplémentaire par rapport à une ZAU d'avoir déjà une surface (en m2) construite. Pour les ZU et ZAU vous écrirez une méthode `surfaceConstructible` permettant de calculer la surface constructible qui donnera la surface constructible totale pour une ZAU et la surface constructible restante pour une ZU.
- Les ZA sont des ZN qui ont comme caractéristiques supplémentaires le type de culture cultivée (chaîne de caractères).

— Sur une ZA, un agriculteur peut construire des bâtiments agricoles à condition que la surface construite ne dépasse pas 10% de la surface de la ZA et 200m2 au maximum (chiffres fictifs).
— Les ZN ne sont pas constructibles.

Faites un diagramme de classe dans votre rapport qui prend en compte ces nouvelles informations. Puis implémentez les nouvelles classes ZU, ZAU, ZA et ZN selon la conception de votre diagramme de classes. Vous surchargerez « pour ZU, ZAU, ZA et ZN en indiquant de plus de quelle classe il s'agit et en adaptant la sortie pour écrire les informations pertinentes pour chaque classe.
La figure 4.1 donne un exemple de sortie d'écran attendu pour les différents types de parcelles.

```
Parcelle n°43 :
        Type : ZU
        Polygone : [0;30] [60;100] [0;100]
        Propriétaire : bdaerz
        Surface : 2100.0
        % constructible : 55 %
        surface construite : 479.12198
        surface à construire restante : 675.87805
Parcelle n°14 :
        Type : ZAU
        Polygone : [0;30] [80;30] [80;100] [60;100]
        Propriétaire : okrrhu
        Surface : 3500.0
        % constructible : 5 %
Parcelle n°92 :
        Type : ZA
        Polygone : [0;0] [80;0] [80;30] [0;30]
        Propriétaire : hnbnor
        Surface : 2400.0
        Type culture : znnp
Parcelle n°67 :
        Type : ZN
        Polygone : [80;0] [100;0] [100;100] [80;100]
        Propriétaire : tihdbw
        Surface : 2000.0
```

FIGURE 4.1 – Exemple d'écritures à l'écran d'informations sur des Parcelles de différents Types

Les noms et les pourcentages constructibles peuvent être totalement aléatoires. Illustrez par des jeux d'essais l'implémentation de vos parcelles.

5 CARTE

Une `Carte` est caractérisée par une liste de Parcelles et une surface totale. Une carte est construite à partir des informations lues dans un fichier passé en paramètre qui suit le format suivant :

— Une Parcelle est définie par deux lignes :

    - La première contient au moins : typeParcelle numéro propriétaire
    - La deuxième contient la liste des points définissant la forme de la Parcelle
  - Pour une ZU : typeParcelle numéro propriétaire pConstructible surfaceConstruite
  - Pour une ZAU : typeParcelle numéro propriétaire pConstructible
  - Pour une ZA : typeParcelle numéro propriétaire typeCulture
  - Pour une ZN : typeParcelle numéro propriétaire

Vous trouverez deux exemples de fichiers de sauvegarde dans `Parcelle_short.txt` et `Parcelle.txt` (figure 5.1).

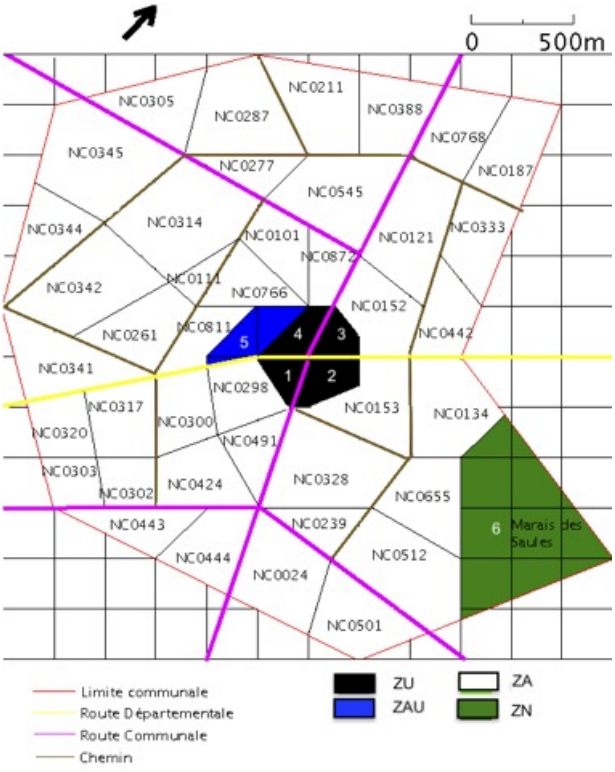

FIGURE 5.1 – Carte décrite par le fichier `Parcelle.txt`

Vous devez aussi permettre la sauvegarde d'une Carte dans un fichier de sauvegarde. Complétez votre diagramme de classe précédent avec la classe `Carte`.

6 UN PEU D'ALGORITHMIQUE - BONUS

Nous avons laissé de côté le test de la bonne conformation des polygones qui constituent les formes des Parcelles. Pour cette partie, ajoutez les classes et méthodes qui vous semblent nécessaires et décrivez vos choix.

6.1 Levée d'exception pour un polygone dont les points ne sont pas dans le sens trigonométrique

Pour vérifier qu'une liste de point est dans le sens trigonométrique, cherchez le point en bas à gauche. Le produit scalaire des vecteurs formés par ce point (O) et deux points suivants dans la liste P_i et P_{i+1}. Ce produit scalaire doit être positif $\vec{OP_i}.\vec{OP_{i+1}} > 0$.
Écrivez cet algorithme et implémentez-le. Vous lèverez une exception partout où c'est nécessaire si les points d'une forme ne sont pas dans le sens trigonométriques.

6.2 Levée d'exception pour un polygone croisé

Pour vérifier que le polygone est croisé, vous pouvez pour chaque côté du Polygone vérifier l'intersection de ce côté avec tous les autres côtés du polygone. Si une intersection existe, le polygone est croisé est vous devez lever une exception. Cette exception est susceptible d'être levée à la création d'un Polygone ou à l'ajout d'un point dans un polygone existant.
Pour cette question vous commencerez par écrire les algorithmes vous permettant de déterminer si un polygone est croisé avant de coder ! Les algorithmes sont à mettre dans le compte rendu.
Vous illustrerez les levées d'exceptions par des jeux d'essais.