IUT Sénart-Fontainebleau, UPEC — Florent Madelaine
CPOO
IUT Info 2A — 2025-26

Séquence 1 (FA)

---

1 Introduction

Programme

CM: (mardi 3/02) rappels de première année sur les diagrammes de classe (DC) les diagrammes objet (DO) et les diagrammes de séquence (DS).

TD: (mardi 3/02) mise en oeuvre (exo Fichier).

TP: (mercredi 4/02) conception (métro), le matin puis compréhension et amélioration (parcelle), l'après-midi.

Concepts à connaître sur les diagrammes de classe • diagramme de classes d'analyse • diagramme de classes de conception • classe • instance • attribut • méthode • visibilités et leurs notations • propriété statique • propriété dérivée • association • agrégation (ensemble) • composition • héritage • classe abstraite • dépendance (use, create, call) • interface • réalisation • paquetage

Concepts à connaître sur les diagrammes de séquence • ligne de vie • message synchrone et message de retour (appel de méthode) • message asynchrone • message de création • message de destruction • message de séquence système vs message de séquence détaillé

Les notions

Dans la mesure du possible, dans chaque exercice proposé dans le cadre de ce cours, j'indique la notion correspondante

1. usage : synopsis et DCU
2. structure : DC, DO.
3. comportement : DSS et DC.
4. Abstraction : DC avec classes abstraites et interfaces, API, couplage faible.
5. Patron : design patterns (patrons de conception), reconnaître et mettre en oeuvre quelques réponses élégantes apportées à des problèmes récurrents.

2 Les exercices

Fichier

Exercice 1 (structure)**.**

1. Dessiner le diagramme de classe correspondant au code donné en annexe.
2. Dessiner le diagramme objet correspondant au code donné dans le Main en annexe avant le premier appel à countFiles.

Expliquer le changement qui se produit dans ce diagramme objet quand on a terminé de lire la dernière ligne de ce fichier.

On attend donc un DC et un DO.

Réponse. 1.

2.

Exercice 2 (structure)**.** Dessiner le diagramme de séquence correspondant au premier appel de countFiles.

Expliquer la différence avec le diagramme de séquence correspondant au second appel de countFiles.

On attend donc un DSS et des explications, voir deux DSS et des explications.

Réponse.

Métro

Cet exercice est à faire sur papier ou avec plantUML.

Exercice 3 (structure)**.** On souhaite modéliser les lignes de métro d'une grande métropole comme Paris et sa région.

Proposer un diagramme de classe sans méthodes mais détaillant bien les associations et permettant de gérer les concepts suivants.

Il y a plusieurs *lignes* dans la ville. Une ligne comprend des **liaisons** entre des *stations*. Le terme ligne est un peu abusif puisque la ligne peut avoir des embranchements (comme la ligne R à Moret-Veneux-les-Sablons vers Montargis ou Montereau). Un *parcours* d'une station A à une station B peut être *sans changements* ou bien *comporter des changements de lignes*.

Proposer aussi une partie d'un diagramme objet illustrant le fonctionnement de votre diagramme de classe (et donc cohérent avec ce dernier) pour le parcours suivant. On suggère d'avoir une classe liaison entre deux stations. *Ligne R de Melun à Gare de Lyon. Changement pour le RER D à Gare de Lyon (en passant par Maison Alfort) pour Vert de Maison.*

On attend donc un DC et un DO.

Réponse. Notez qu'on peut aller de Melun à Vers de Maison par le RER D (voir 1). A priori il vaut mieux prendre la ligne R puis changer à Gare de Lyon.

Notez qu'un projet de parcours (a un début et une fin) n'est pas un parcours (qui lui connais les liaisons entre les deux).

Le changement n'est peut-être pas pour maintenant mais c'est parfois pratique (surtout quand on ne peut pas faire autrement).

- Classe Ligne
- Classe Station
- Classe Liaison : classe associative, station1, station2, ligne. (on peut ajouter une note soulignant qu'il y a un d'ordre dans ne liaison).
- Classe abstraite parcours comporte deux associations vers Station (début et fin) et un agrégat de liaisons. On peut ajouter une note qui stipule que les liaisons sont ordonnées. Pour souligner le fait qu'on peut itérer, on peut ajouter des méthodes. Typiquement, on peut avoir une méthode qui retourne un Iterateur de liaison (voir patron itérateur). Finalement Deux classes instanciant parcours.

Figure 1: RER D.

- Classe (concrète) parcours direct : comme la classe abstraite, mais en plus une association vers une ligne. On peut ajouter une note que les liaisons sont toutes de cette ligne.
- Classe (concrète) parcours multiple : classe associative, début, fin mais qui est en plus un agrégat de parcours directs
  On peut ajouter une note qui stipule que les parcours directs sont ordonnées. Pour souligner le fait qu'on peut itérer, on peut ajouter des méthodes. Typiquement, on peut avoir une méthode qui retourne un Iterateur de parcours directs (voir patron itérateur).

Parcelles

Exercice 4 (structure)**.** On s'intéresse à un système permettant de gérer un jardin, composé de parcelles rectangulaires.

Chaque jardin est initialement une seule parcelle, c'est-à-dire un rectangle d'une certaine taille. À tout instant, une parcelle peut être sous-divisée en 2 (verticalement ou horizontalement) pour en former deux nouvelles. La parcelle initiale existe toujours et contient dorénavant deux sous-parcelles. À l'inverse, on peut aussi réunir deux sous-parcelles plus tard (à condition qu'elles soient bien issue de la même parcelle), ce qui revient à faire disparaître ses sous-parcelles.

Les jardins pouvant devenir complexes, à l'instar d'une gestionnaire de fichiers, il faudra pouvoir prévoir un système adapté permettant de naviguer et d'occulter des détails ou de ls révéler.

À tout instant, on doit pouvoir indiquer qu'on réalise une action sur une parcelle. Il en existe trois types :

- gérer les légumes (semer, transplanter, récolter, arracher);
- travailler le sol (double-bêchage, bêchage, sarclage, désherbage, amendement de la terre, traitement avec un purin); et,

- observer (maladie, ravageur, etc).

1. Proposer un diagramme de classe (notez qu'il s'agit du diagramme de classe du modèle de l'application, on ne souhaite pas avoir de classe correspondant à la vue ou au controleur).
2. Vous pouvez pour vous aider donner un diagramme d'objet illustrant l'exemple de la figure

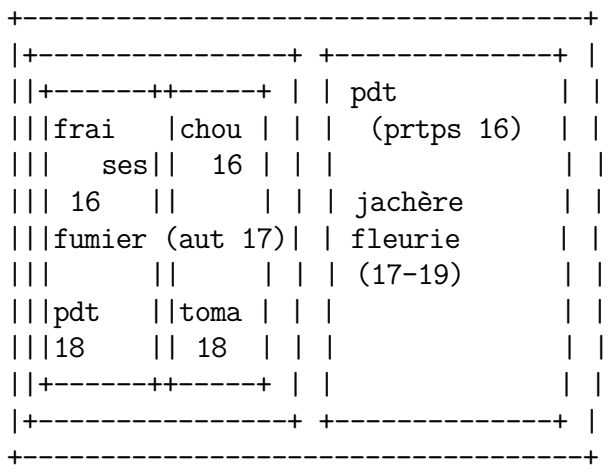
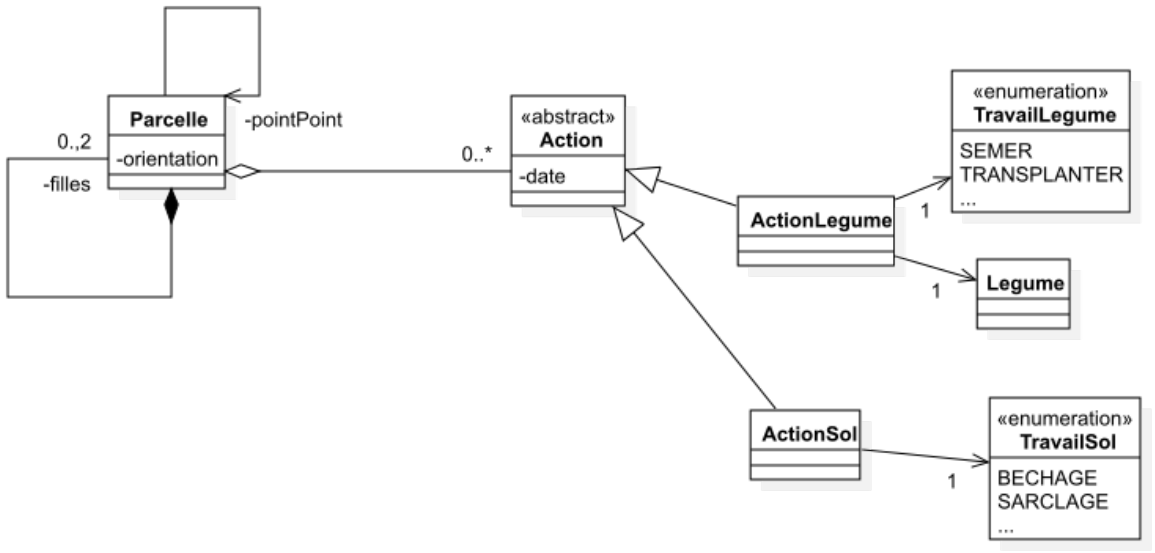


Figure 2: Esquisse d'un jardin (NB. prtps veut dire Printemps, aut veut dire Automne et hvrs veut dire hivers; pdt dénote un tubercule cher à Parmentier).

Réponse.

On se contente des classes et de leur structure (on ne mentionne pas les méthodes pour l'instant, qu'on verra plus en détails dans d'autres exercices).

Je n'ai pas modélisé le troisième type d'action, mais c'est similaire.



Exercice 5 (comportement)**.** Un jardinier interagit avec un système de jardins, qui est une version proche mais simplifiée de celui entrevu dans des exercices précédents.

Pour simplifier, il y aura donc un seul jardin et les parcelles de ce dernier ne seront pas sauvées si elles sont fermées. Il y aura un seul acteur qui pourra à la fois changer les parcelles et pour chaque parcelle ajouter ou enlever un légumes .

On a les classes suivantes dont on ne connaît pas encore toutes les méthodes (voir Figure 3 pour le diagramme de classe).

- Classe VueJardin. Elle contient un attribut de type parcelle qui correspond à la parcelle courante que le jardinier regarde et une méthode `SetParcelle` qui prend en argument une parcelle et va mettre à jour cet attribut.
- Classe Parcelle. Attributs : un booléen `root` vrai ssi c'est la parcelle principale; une association vers deux sous-parcelles `sp0` et `sp1`; une orientation `split` (H ou V) (les sous-parcelles seront gauche et droite ou haut et bas en fonction de l'orientation, l'attribut vaut `Null` si il n'y a pas de sous-parcelle); un légume (un type énuméré). La méthode `getFirst` retourne la parcelle gauche si split vaut V, ou haute si split vaut H. La méthode `reset` va détruire les sous-parcelles.

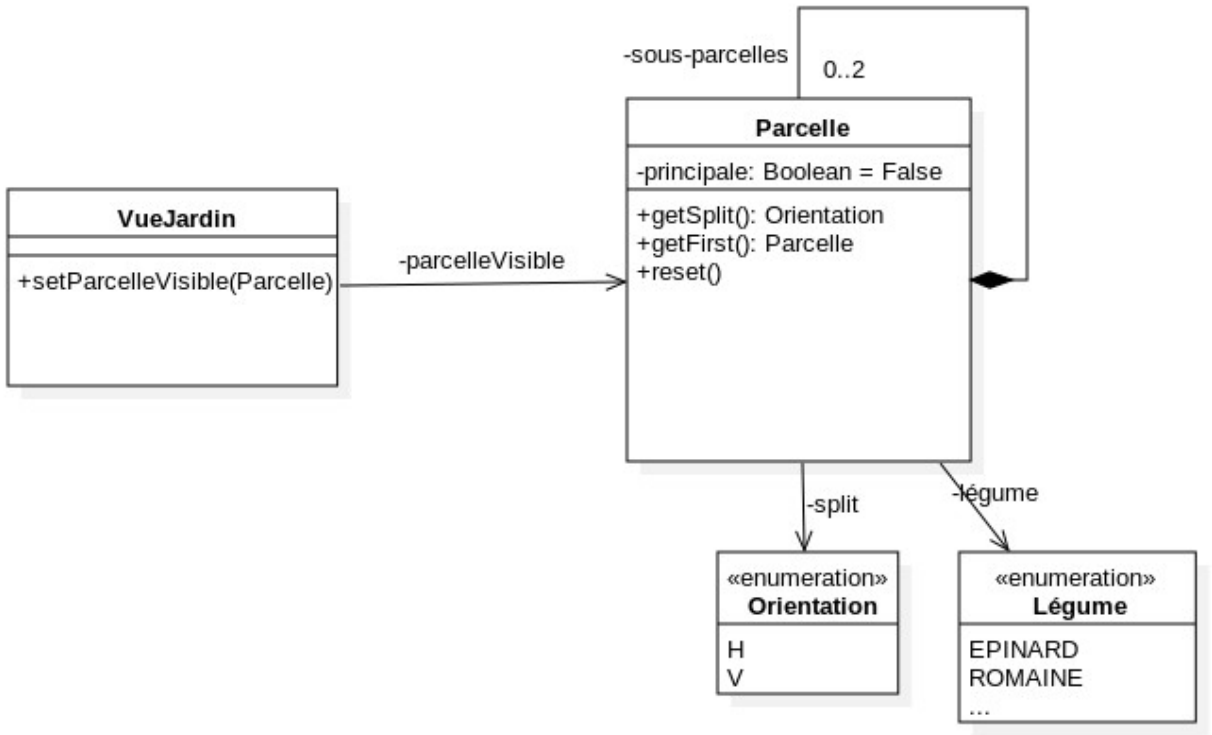


Figure 3: Diagramme de classe partiel

Question 1. On considère le diagramme de séquence système de la Figure 4. Notez que P0 est la parcelle racine du jardin. Donnez un synopsis listant ce que Papé14 a fait sur l'appli. Vous pouvez illustrer le synopsis par un dessin des parcelles.

Question 2. Nous disposons du synopsis ci-dessous. Proposez un diagramme de séquence système correspondant à ce synopsis.

Si une méthode est nécessaire dans une classe, indiquez sa signature et donnez une brève description de son comportement si le nom n'était pas assez clair.

Synopsis

- Le jardinier Papé14 visionne la parcelle initiale de son jardin.
- Il voit que cette parcelle est coupée verticalement en deux sous-parcelles.
- Il sélectionne et navigue dans la parcelle droite. Cette parcelle est coupée verticalement en deux.
- Il sélectionne la partie droite et voit qu'elle n'est pas découpée.
- Il la partage en 2 verticalement.
- Il change d'avis et réunit ces deux parcelles en une seule parcelle.
- Il la partage en 2 horizontalement.
- Il navigue successivement sur la parcelle initiale.
- Il sélectionne la parcelle gauche et voit qu'elle contient de la romaine (une salade).
- Il enlève la romaine.
- Il ajoute des épinards.

Réponse. Question 1.

Synopsis

- Le jardinier Papé14 visionne la parcelle initiale de son jardin.
- Il voit que cette parcelle est coupée verticalement en deux sous-parcelles.
- Il sélectionne la parcelle gauche.
- Il voit que cette parcelle est coupée horizontalement en deux.
- Il réunit ces deux parcelles en une seule parcelle.

Question 2. DSS TO DO (mais largement inspiré de la question précédente).

Il faut ajouter `Split(Orientation)` à la classe parcelle qui va créer des sous-parcelles.

Il faut pouvoir naviguer à droite ou en bas avec `getSecond()`.

Pour la navigation, il faut pouvoir revenir en arrière. Il faut donc une méthode `getUp()` qui retourne la parcelle mère. Il faut ajouter un attribut `up` de type Parcelle pour représenter la parcelle mère.

Il faut des méthodes `GetLégume(), ResetLégume(), SetLégume()`.

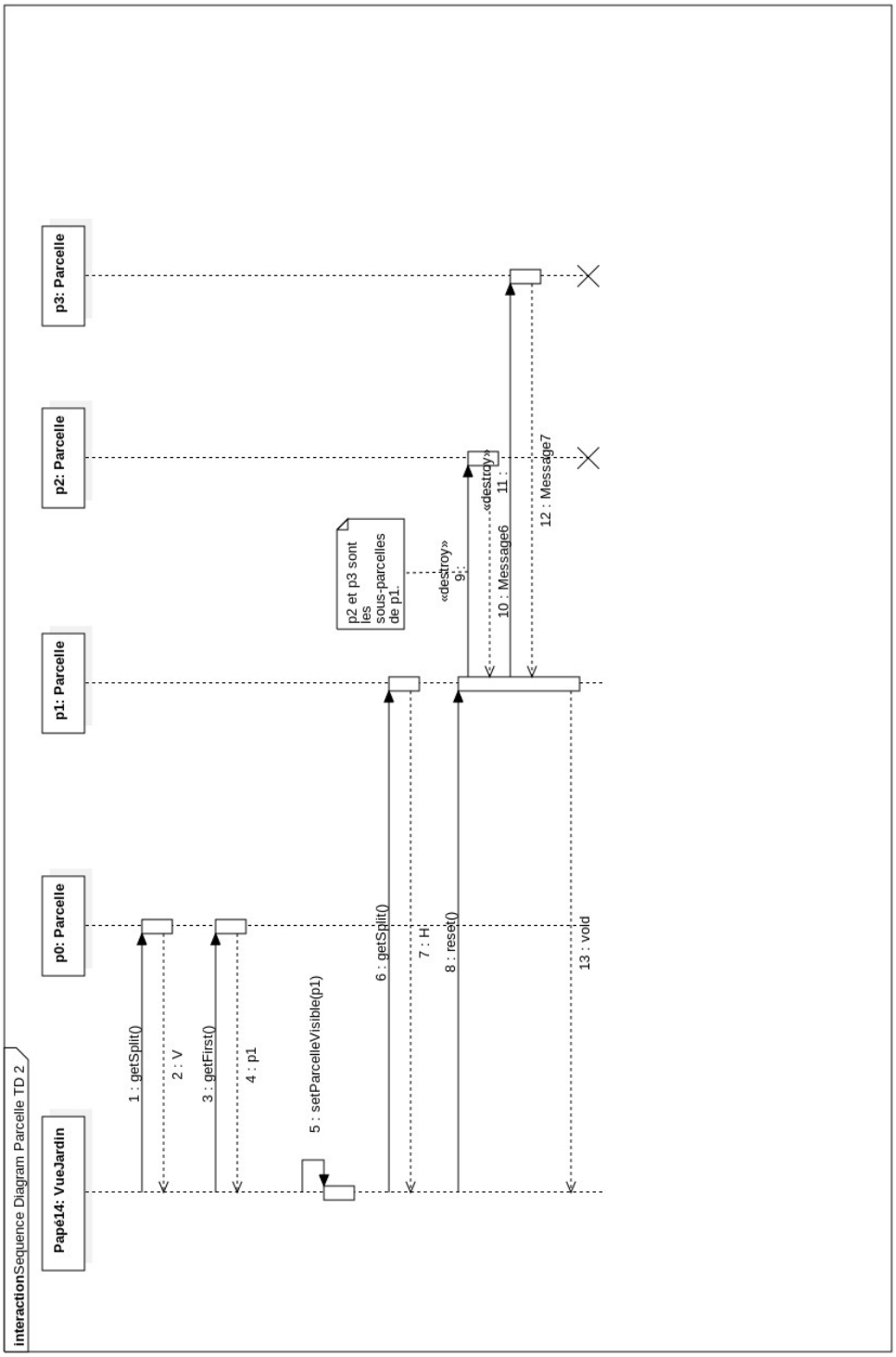


Figure 4: Diagramme de séquence système.