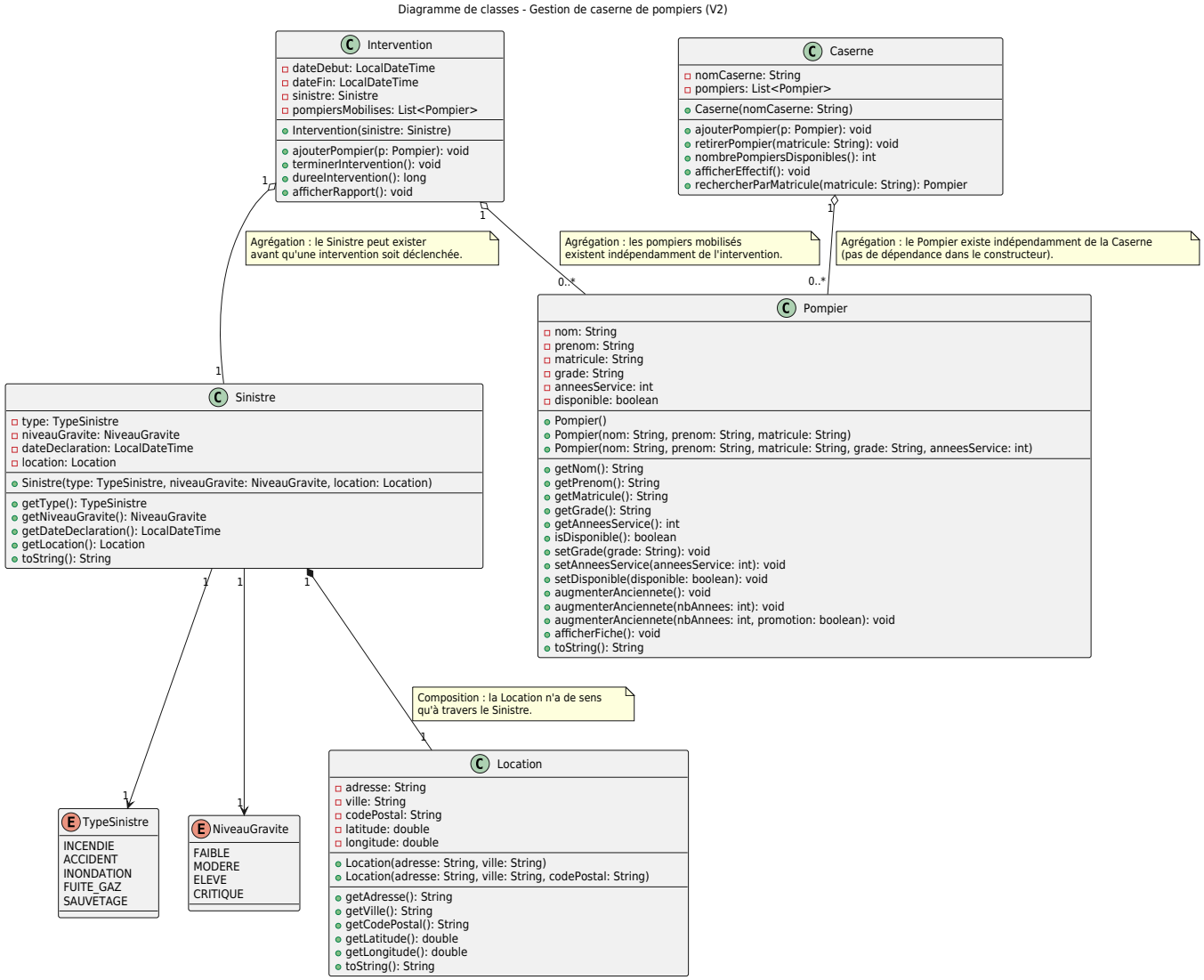
TD0 : Révisions

Encapsulation, construction

Associations



Descriptif des méthodes - Location, Sinistre, Intervention

Location

Constructeurs

- `Location(adresse, ville)` : constructeur simplifié, `codePostal` initialisé à “”**, latitude/longitude** à 0.0.
- `Location(adresse, ville, codePostal)` : constructeur complet (sans coordonnées GPS pour rester simple).

Getters

Un par attribut (`getAdresse()`, `getVille()`, `getCodePostal()`, `getLatitude()`, `getLongitude()`). Pas de setters : une fois créée, une localisation ne change pas.

toString()

Retourne une chaîne lisible, ex : “12 rue des Lilas, 75001 Paris”.

Sinistre

Constructeur

- `Sinistre(type, niveauGravite, location)` : initialise `dateDeclaration` automatiquement à LocalDateTime.now().

Getters

- `getType()`, `getNiveauGravite()`, `getDateDeclaration()`, `getLocation()`.

Pas de setters : un sinistre déclaré ne change pas de nature.

toString()

Doit afficher une synthèse : type, gravité, date, localisation. Ex :

[INCENDIE] Gravité : ELEVE - déclaré le 21/08/2026 à 10:32 - 12 rue des Lilas, Paris

---

Intervention

Constructeur

- `Intervention(sinistre)` : initialise `dateDebut` à LocalDateTime.now(), `dateFin` à null, `pompiersMobilises` à une liste vide.

ajouterPompier(Pompier p): void

Ajoute un pompier à la liste `pompiersMobilises`, mais uniquement s'il est disponible (p.isDisponible()). Si non disponible, afficher un message d'erreur (ou lever une exception si vous voulez aller plus loin). Une fois ajouté, on peut aussi le passer à disponible = false.

terminerIntervention(): void

- Fixe `dateFin` à LocalDateTime.now().
- Repasse chaque pompier de `pompiersMobilises` à disponible = true.
- Éventuellement afficher un message de confirmation.

dureeIntervention(): long

Calcule la durée en minutes entre `dateDebut` et `dateFin` (utiliser Duration.between(...).toMinutes()). Si `dateFin == null`, on peut soit calculer par rapport à LocalDateTime.now() (intervention en cours), soit renvoyer -1 ou lever une exception.

afficherRapport(): void

Affiche une synthèse complète de l'intervention : sinistre concerné, dates début/fin, durée, liste des pompiers mobilisés (nom, prénom, matricule).

Points de vigilance

- **Cohérence disponibilité pompier ↔ intervention** : si un pompier est mobilisé, il ne devrait pas pouvoir être ajouté à une deuxième intervention en même temps. C'est `ajouterPompier` qui doit porter cette règle métier (pas juste l'ajout brut à la liste) — bon exemple de logique métier encapsulée dans une méthode plutôt que laissée à l'appelant.
- **dateFin nullable** : premier contact avec la gestion d'un état “optionnel”/“pas encore renseigné”. Peut aussi ouvrir une discussion sur Optional<LocalDateTime> si vous voulez aller plus loin (probablement hors scope 2 SIO à ce stade).
- **Immutabilité de Location et Sinistre** : contraste intéressant avec Pompier qui a des setters. Bonne question à poser en classe : “Pourquoi ces classes n'ont pas de setters, à votre avis ?”