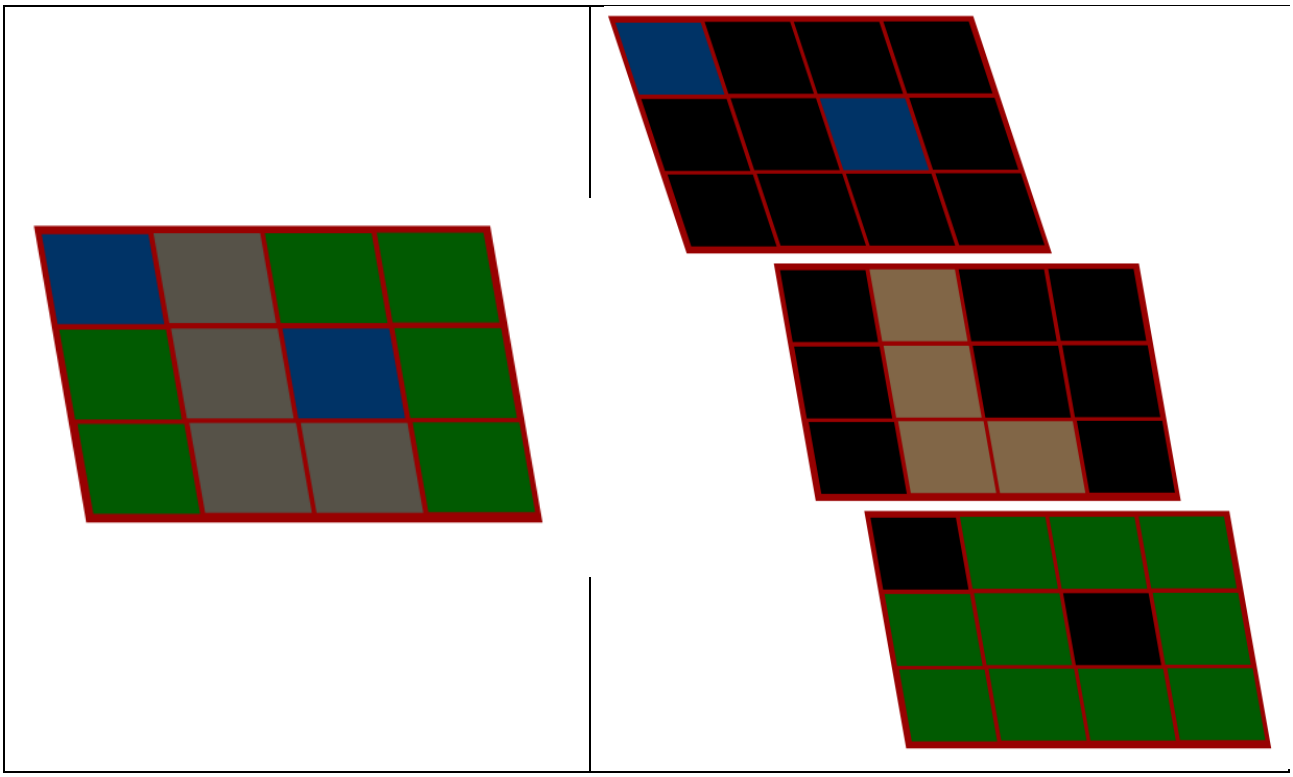
TD 2 – Layered BattleField

L'objectif de ce TD est de créer un champ de bataille un superposant des niveaux de terrains.

Dès lors on peut imaginer un terrain sur lequel nous allons disposer des pièces d'eau, des routes et de la plaine. L'idée est de séparer les différents types de terrains :

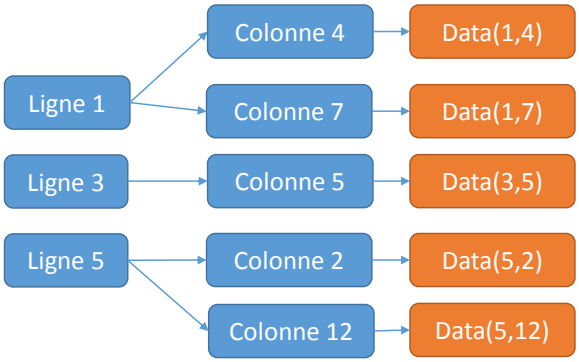


Pour tous le TD :

- chaque classe devra être commentée en JavaDoc
- chaque classe devra être soumise à un test unitaire
- mettre en place vos propres classes d'exceptions pour gérer les erreurs

Exercice 1 (Généricité & Collection) : création d'une matrice creuse générique – *la classe BattleFieldMatrix*

Proposer et implémenter une structure de donnée représentant une matrice creuse :



Les composants de la matrice (Data sur le dessin) doivent être un argument générique.

La classe décrivant cette matrice devra disposer (au moins) des accesseurs :

- *GenericData get(numeroLigne,numeroColonne)*
- *set(numeroLigne,numeroColonne,GenericData data)*
- *exists(numeroLigne,numeroColonne)*

Exercice 2 (Héritage & Généricité) : création d'un élément de terrain – *la classe BattleFieldPieceWise*

Un élément de terrain est une classe abstraite. Un élément de terrain dispose d'une position (X,Y) avec les « setter » et « getter » de cette position. Cette classe abstraite à deux fonctions abstraites :

- *boolean isCompatible(BattleFieldPieceWise with)*
- *isDestroyable()*

Faire en sorte que la classe générique ***BattleFieldMatrix*** (Exercice1) dépende uniquement de classe du type **BattleFieldPieceWise.** Autrement dit, l'argument générique doit être du type **BattleFieldPieceWise**.

Exercice 3 (Héritage & Introspection) : création de plusieurs implémentations de *la classe BattleFieldPieceWise*.

L'objectif ici est de créer les éléments de terrain qui héritent de la classe ***BattleFieldPieceWise*** et donc implémentent les deux fonctions abstraites *isCompatible* et *isDestroyable.*

Nous nous intéresserons à trois éléments de terrains (au moins) :

- Pièce d'eau : ***BattleFieldWater***
- De la plaine : ***BattleFieldLandScape***
- Une route ***: BattleFieldRoad***

Contraintes :

- ***BattleFieldWater***
  - n'est pas destructible
  - est incompatible avec de la plaine ou une route
- ***BattleFieldLandScape***
  - est destructible
  - est incompatible est une pièce d'eau
  - est compatible avec une route
- ***BattleFieldRoad***
  - est destructible
  - est incompatible est une pièce d'eau
  - est compatible avec une plaine

La fonction héritée *isCompatible* permet de savoir si la classe implémentant cette fonction est compatible avec une classe dérivée de la classe ***BattleFieldPieceWise***. En utilisant l'introspection et pour chaque classe ***BattleFieldWater, BattleFieldLandScape***, ***BattleFieldRoad,*** implémenter la fonction *isCompatible* pour respecter les contraintes décrites ci-dessus.

Exercice 4 – Allez plus loin (facultatif)

Vous pouvez maintenant créer une classe permettant de regrouper tous les niveaux de terrains.