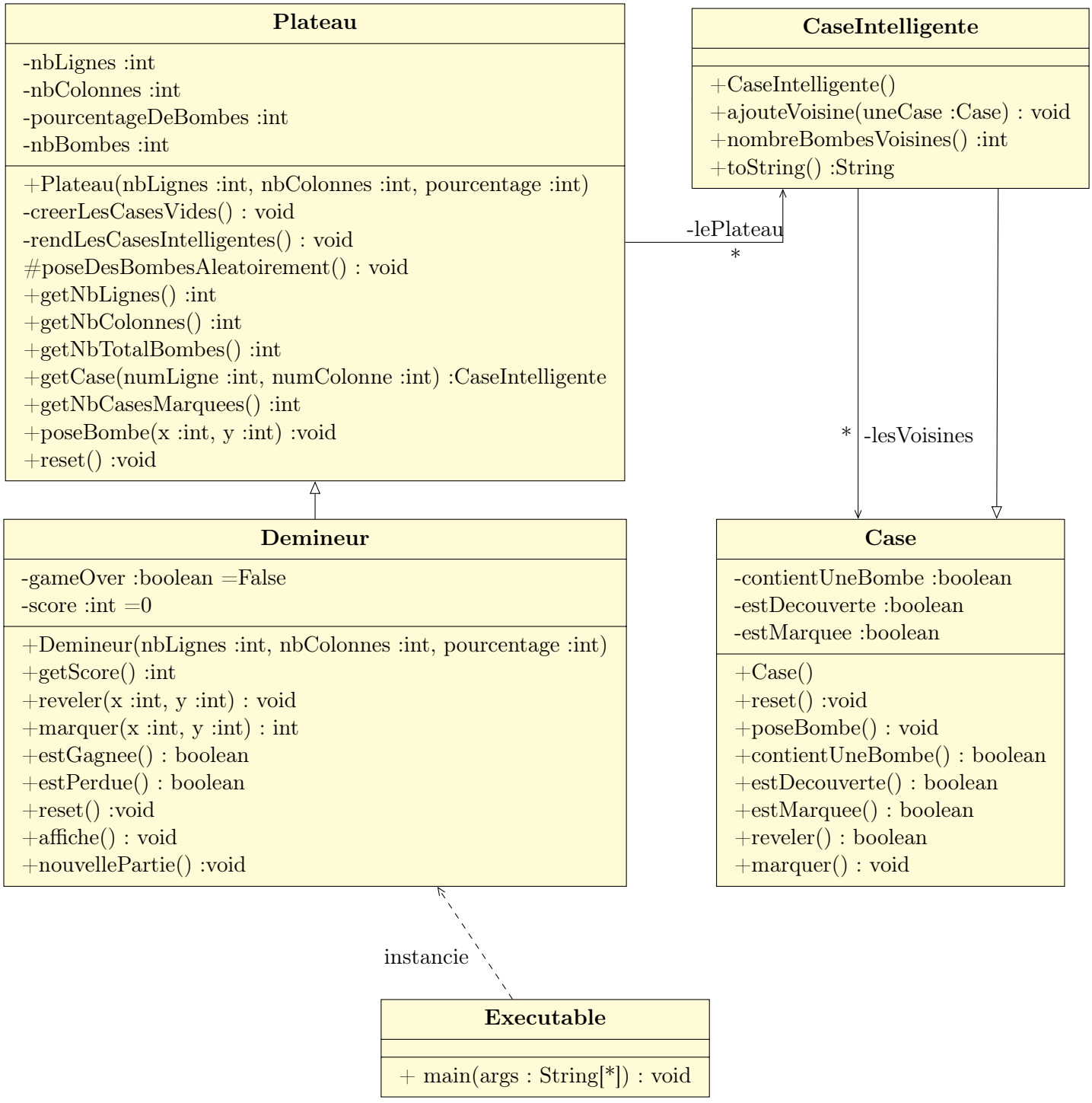
PROJET N°6 *Demineur*

Objectif

Développer une petite application (le jeu du démineur) dont on donne le diagramme de classes

Après avoir étudié plusieurs propositions, le client fournit un diagramme de classes de l'application.



Quelques précisions

- Dans la classe `CaseIntelligente`, la méthode `toString()` renvoie ce qui doit être affichée sur la case. Par exemple
  - `" "` si la case n'est ni révélée ni marquée
  - `"?"` si la case a été marquée
  - Si la case est révélée :
    - `"@"` si elle contient une bombe
    - `"0"` si elle ne contient pas de bombe et aucun de ses voisins ne contient une bombe
    - `"3"` si elle ne contient pas de bombe et trois de ses voisins contiennent une bombe.
- La méthode `reset()` de la classe `Plateau` permet de remettre toutes les cases dans leur état initial (non révélée, non marquées et non bombée). Celle de la classe `Demineur` remet en plus le score à zéro et gameOver à false.
- Dans la classe `Plateau`, la méthode `rendLesCasesIntelligentes()` permet de "donner" à chaque case toutes ses cases voisines. Par exemple, sur un plateau 5×5 :
  - Les cases voisines de `this.getCase(0,0)` sont `this.getCase(1,0)`, `this.getCase(1,1)` et `this.getCase(0,1)`
  - La case `this.getCase(2,3)` possède 8 cases voisines.
  - La case `this.getCase(3, 4)` en possède 5.

Exercice 1 *Implémentation du squelette de l'application*

Implémente ce diagramme de classes en Java dans un dépot git en portant une attention particulière aux points suivants :

☐ les fichiers sont correctement rangés dans des dossiers `src`, `bin` et `doc`

☐ Le code est documenté et la documentation est à jour dans le dossier `doc`

☐ Dans un premier temps, les méthodes autre que les constructeurs et les getteurs ne font RIEN.

☐ Le depôt git est correctement géré. En particulier, les dossiers `bin` et `doc` ne doivent pas être versionnés.

☐ Les commits ont du sens

Exercice 2 *Ajout des fonctionnalités*

Implémente petit à petit les différentes méthodes du projet en portant une attention particulière aux points suivants :

☐ Le code respecte les bonnes pratiques

☐ Les méthodes sont codées unes à unes. Chaque méthode est testée et validée avant de commencer à coder la suivante.

☐ Chaque méthode codée donne lieu à un commit.

Exercice 3 *On peut jouer ?*

3.1 Vérifier que tu peux jouer en exécutant la classe `Executable` fournie par le client.

3.2 Vérifier que ton application se "connecte" à la partie graphique fournie par le client. Pour cela, commente les premières lignes de l'`Executable` et décommente la dernière ligne. Compile puis exécute.

Exercice 4 *Pour aller plus loin*

4.1 Ajoute de la couleur aux numéros, mets des images de drapeaux à la place des '?'

4.2 Ajoute d'autres fonctionnalités à inventer ...