Rapport d'analyse
Programmation Objet Avancée
Projet : Chasse au trésor

François Poguet
Enzo Costantini
TP1A

Avril 2020

1 Diagramme de classe

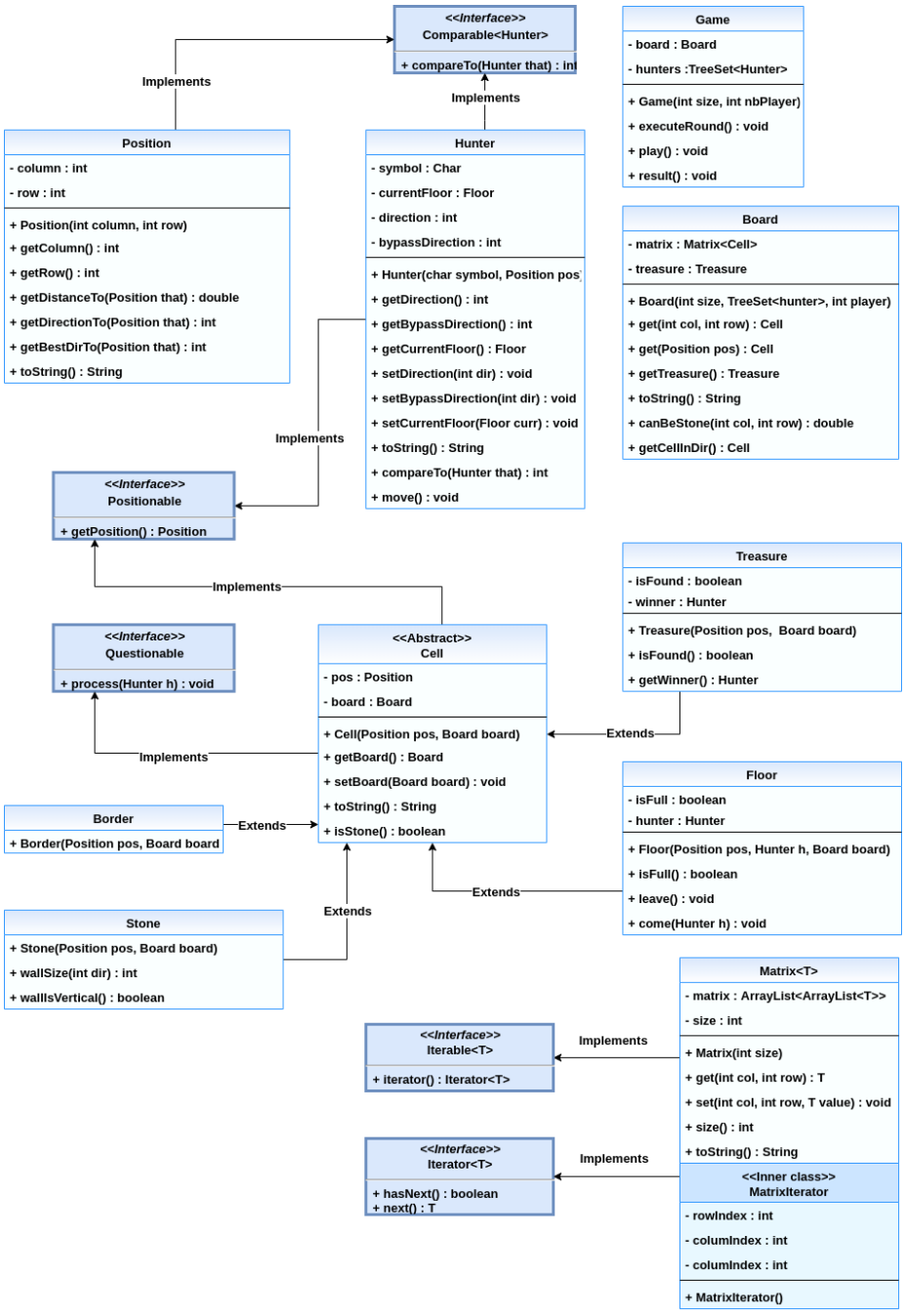

FIGURE 1 – Diagramme de classes de l'application en mode console

2 Informations complémentaires

Formalisme du diagramme de classes

Le diagramme de classes ci-dessus suit les conventions suivantes :

1. Les méthodes d'une interface sont implicitement présentes dans les classes qui l'implémentent.
2. Les méthodes d'une classe mère sont implicitement présentes dans ses classes filles.

Ex : Chaque classe héritant de la classe *Cell* intègre la méthode *process()* qui devra être réimplémentée.

Choix d'implémentations

La collection Matrix<T>

Nous avons choisi d'implémenter nous-même une collection pour représenter un tableau à deux dimmensions. L'avantage de cette collection générique est qu'elle sera adaptée au mieux aux besoins de l'application. Pour cela, des ArrayList seront utilisées, car une fois la matrice créée, les opérations principales seront des recherches et non des ajouts/suppressions.

La position d'un hunter

Bien que la classe *Hunter* implémente l'interface *Positionable*, le hunter ne stock pas une position en tant que telle. Il possède une case courrante, qui elle, est caractérisée par une position.

Un TreeSet de hunters

Dans la classe *Game*, nous avons préféré stocker les joueurs dans un TreeSet, sachant que les hunters sont comparables par leurs positions, et que deux hunters ne peuvent être sur la même case, ce choix nous semble judicieux. Cela nous permettra, entre-autres, de simplifier l'attribution des positions aux joueurs.

Les murs et leur contournement

Pour l'implémentation des murs, nous avons réfléchis à une éventuelle classe *Mur* qui regrouperais plusieurs cases de type *Stone*, mais nous ne pensons pas que cela soit vraiment utile.
Pour contourner les murs, les hunters possèdes une "direction de déviation" qui leur permettra de les contourner au mieux.