TP1 – Tours de Hanoi

Dominique Bégin,
Dominique Septembre et
Gérémy Sorlini

Remis à Benjamin Lelemin le 25 Septembre 2014

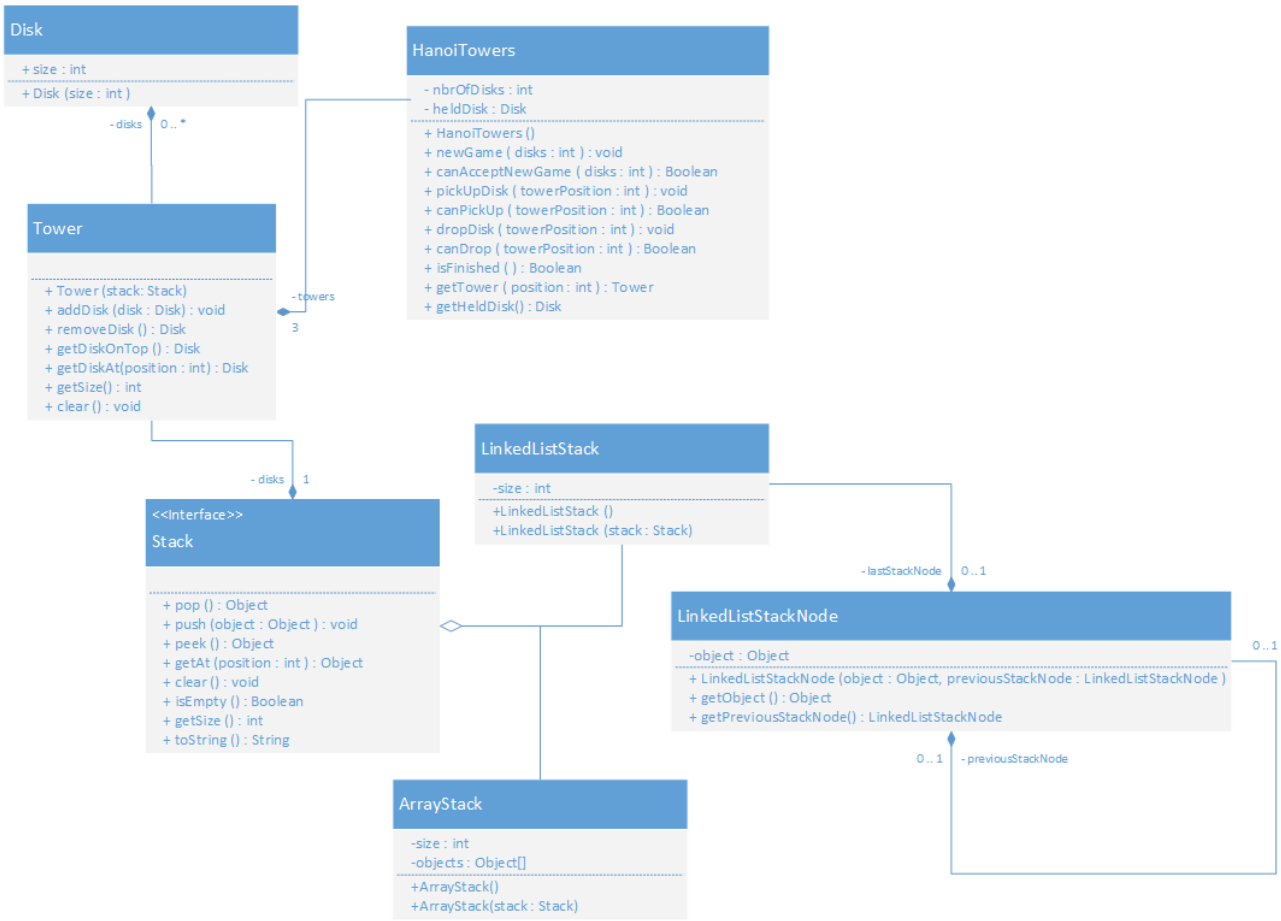
Table des Matières

- Progression des "Users Story" : Page 4
- Diagramme UML des classes : Page 5
- Particularitées liées à l'architecture : Page 6

Progression des "Users Story"

No.	Description	Progression
1	En tant qu'utilisateur, je veux pouvoir commencer une nouvelle partie de Hanoi Towers.	100.00%
2	En tant qu'utilisateur, je veux pouvoir choisir le nombre de disques sur la première tour lorsque je commence une nouvelle partie. Le nombre de disques maximal doit être six (6) et le nombre minimal doit être trois (3)	100.00%
3	En tant qu'utilisateur, je veux pouvoir déplacer un disque d'une tour à l'autre.	100.00%
4	En tant qu'utilisateur, je veux être restraint au niveau des déplacements de disques en fonction des règles du jeu Hanoi.	100.00%
5	En tant qu'utilisateur, je veux que voir, à l'aide de l'interface graphique, les disques sur les trois tours de Hanoi.	100.00%
6	En tant qu'utilisateur, je veux que chaque déplacement d'un disque se reflète sur la représentation graphique des trois (3) tours.	100.00%
7	En tant qu'utilisateur, je veux que l'on me félicite lorsque je réussis à déplacer tout les disques de la première à la dernière tour.	100.00%
8	En tant qu'utilisateur, je veux que l'on me demande si je veux faire une nouvelle partie une fois que j'en ai terminé une. Sinon, l'application doit quitter.	100.00%

Diagramme De Classes



Particularitées liés a l'architecture

Le programme est construit autour de la classe "HanoiTowers" qui est la "classe maitresse" du jeu. Elle est éssentiellement un ensemble de méthodes nécéssaire éffectuer les movements librement, ainsi que de vérifier la légalitée des coups.

Elle possède trois attributs; les trois tours, un integer correspondant au nombre de disques utilisés pour la partie, ainsi qu'un pointeur de disque pour le disque pris par le joueur, nommé held disk. L'importance de cet attribut sera souligné dans un instant.

Ensuite, au niveau des méthodes, on retrouve "newGame", permettant, comme le nom le suggère, de commencer une nouvelle partie, et ensuite, les méthodes de déplacement; "pickUpDisk" et "dropDisk", séparés grâce à l'atribut held disk. "pickUpDisk" retire simplement un disque d'une tour est le fait pointer par heldDisk, alors que dropDisk ajoute le "heldDisk" à une tour, ce qui permet de séparer les méthodes pour respecter le "Single Responsability Principle", puis attribuant "NULL" à l'attribut "heldDisk" puisque le disque n'est plus tenu. Enfin, nous retrouvons les méthodes de vérification des coups légaux, soit "canDrop" et "canPickUp". Ces méthodes sont séparés des méthodes de déplacement, encore une fois sous le SRP. Elle sont en fait appelés par le contrôleur d'interface à chaque déplacement, pourque celui-ci puisse déterminer les mouvements légaux et ainsi choisir quel bouton doit être actif ou non. Enfin, on retrouvera "isFinished", qui retourne, en booléen, si le joueur a gagné ou non, méthode qui est, une fois de plus, appelé par l'interface, après chaque déplacement.

Ainsi, HanoiTowers est un regroupement de méthodes et d'attribut, ainsi que de "stacks" (sous la forme de "Towers"), qui permet à n'importe quel interface d'intégrer le jeu de manière logique et versatile. Il pourrait facilement être extrait et être utilisé dans d'autre interfaces, comme par exemple un menu de console. Il est même possible de faire une "partie mentale" à partir d'une suite de méthodes dans le code (un vrai délice pour les tests unitaires!), ce qui nous donne une architecture versatile et symbiotique.