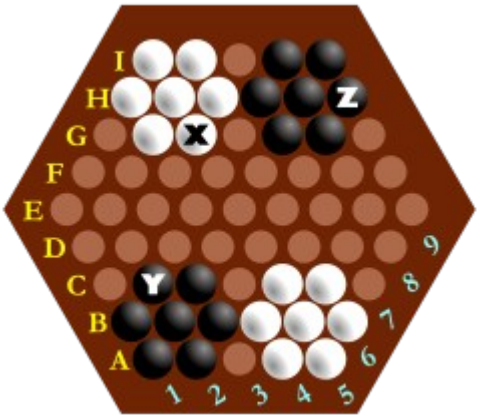
ABALONE-DEV4

Viho Juste/Gardeur Nicolas

Table des matières

ABALONE-DEV4

Viho Juste/Gardeur Nicolas



But du jeu

Être le premier à avoir éjecté les 6 billes adverses hors de l'hexagone, bille après bille.

Classes et Methode

Marble :

-color : Color
-position : Position

Methode:

-getColor():Color
+Marble(Color color)
+getPosition(): Position
+setPosition(Position Position):void

Description:

Cette classe représente une bille du jeu composer d'une enum color pour representer la couleur de la bille (Noir ou Blanc).

La bille aura une position dans la plateau de jeu d'ou l'atribitut secondaire position de la classe Position qui represente une position de coordonnée (x,y) d'un tableau à deux dimensions.

En guise de fonction nous avons des accesseurs et mutateur avec un constructeur somme toute classique .

ABALONE-DEV4

Viho Juste/Gardeur Nicolas

Player :

-playerColor : Color
-_marble[]: Marble
-_playerId: char
-_playerState: State

methode :

+getPlayerColor(): Color
+getMarble(indice int): Marble
+move(Direction direction, Marble marble...):void
+remove(Marble marble...):void
+Player(Color color)
+getId(): char
+getState(): State

Description:

La classe player représente un joueur dans le jeu nous aurons deux instance de cette classe pour representer deux joueurs differents.
PlayerColor : pour faire refference à la couleur du joueur.
Marble[] : Un tableau de bill contenant 14 bille par joueur
playerId : definit l'identifiant du joueur
playerState : definit l'état du joueur au sein du jeu (WIN ⇒ si gagné, FAIL ⇒ si perdu , NEUTRAL ⇒ etat de base du joueur).
Outre les accesseurs il y a la fonction move qui va déplacer les billes du joueur et remove pour retirer la bille du joueur si il perd.

PosAbaPro :

-_row : char
-_column : integer

methode:

+PosAbaPro(char ligne, int colonne)
+getLigne():char
+getColonne(): int

Description:

Elle represent la façon dont l'encodage des entrée en console seront effectuer row de type char pour la ligne et column de type int pour la colonne. Le but au final est de créer un tableau d'objet de typePosAbaPro qui vont regroupé toute les entrées de coup du joueur qui vont ensuite être converti en position 2D pour savoir exactement dans notre implémentation de quel marble il s'agit grâce à la fonction convert dans game le tableau PosAbaPro de coup (char X,int Y) va être convertie en (int X,int Y) pour savoir sur quel marble dans notre board qu'il faudra appliquer un deplacement.

Comme fonction nous avons uniquement contructeur et accesseur

Game :

ABALONE-DEV4

Viho Juste/Gardeur Nicolas

-_board : Board
-_currentPlayer : player
-_opponentPlayer : Player

methode:

+Game()
+initialized():void
+convert():void
+move(Position position, Direction direction):void
+isOver(): bool
+winner():char

+switchPlayer():void

Description:

C'est la classe assurant la gestion et bon fonctionnement du jeu tout en respectant les régles.

Comme attribut :

board : le plateau du jeu ou afficher les grilles
currentPlayer : l'instance du joueur courant
opponentPlayer : l'instance du joueur adverse
Comme fonction nous avons un constructeur de game une fonction initialized qui initialise le jeu.
La fonction convert retourne un tableau de Position des input abaPro pour appliquer les mouvements move applique le déplacement.
isOver retourne vrai si le jeu est finit donc si on a un gagnant et false si il y en a pas.

Board :

-_square[][] : Square

methode:

+getSquare(): Square
+Board()
+isInside(Position Position): bool
+setSquareType(Position position)

Description:

Board est le plateau de jeu une representation de l'ensemble des cases

getSquare retourne les cases du plateau
Board représente le plateau de jeu
isInside va retourner un boolean true si je suis toujours sur le plateau de jeu après mouvement
setSquareType va permettre de modifier l'état de la case sur le plateau

Position :

-_x : int
-_y : int

methode:

+getRow(): int
+getColumn(): int
+Position(int_row,int_column)

Description:

getRow retourne la ligne
getColumn retourne la colonne
Position position en deux dimension d'une case dans le plateau

Square :

-position : Position
-marble : Marble
-squareType : Type

methode:

+Square(Marble marble, squareType Type, position Position)
+getMarble(): Marble
+getSquareType(): Type
+getPosition(): Position
+marbleIsPresent(Position position): bool
+setSquareType(Type squareType)

Description:

La classe Square represente une case dans le plateau de jeu

Square créer un objet square qui représente une case
getMarble retourne la bille
getPosition retourne la position (x,y) de la case
marbleIsPresent retourne un boolean true si la bille et a cette case
setSquareType modifie l'état du square

Enumération Color :

WHITE
BLACK

Description:

Couleur des billes des deux joueurs

Enumération Direction :

RIGHT
LEFT
RIGHT UP
RIGHT DOWN
LEFT UP
LEFT DOWN

Description:

contient les directions possible pour effectuer un deplacement dans le jeu

ABALONE-DEV4

Viho Juste/Gardeur Nicolas

Enumeration Type :

EMPTY
BUSY

Description:

Contient le type de la case EMPTY ⇒ pour une case sans marble et BUSY ⇒ pour une case avec marble

Enumeration State :

WIN
FAIL
NEUTRAL

Description:

Une façon d'implémenter l'etat du joueur durant le jeu

ABALONE-DEV4

Viho Juste/Gardeur Nicolas

Aperçu UML