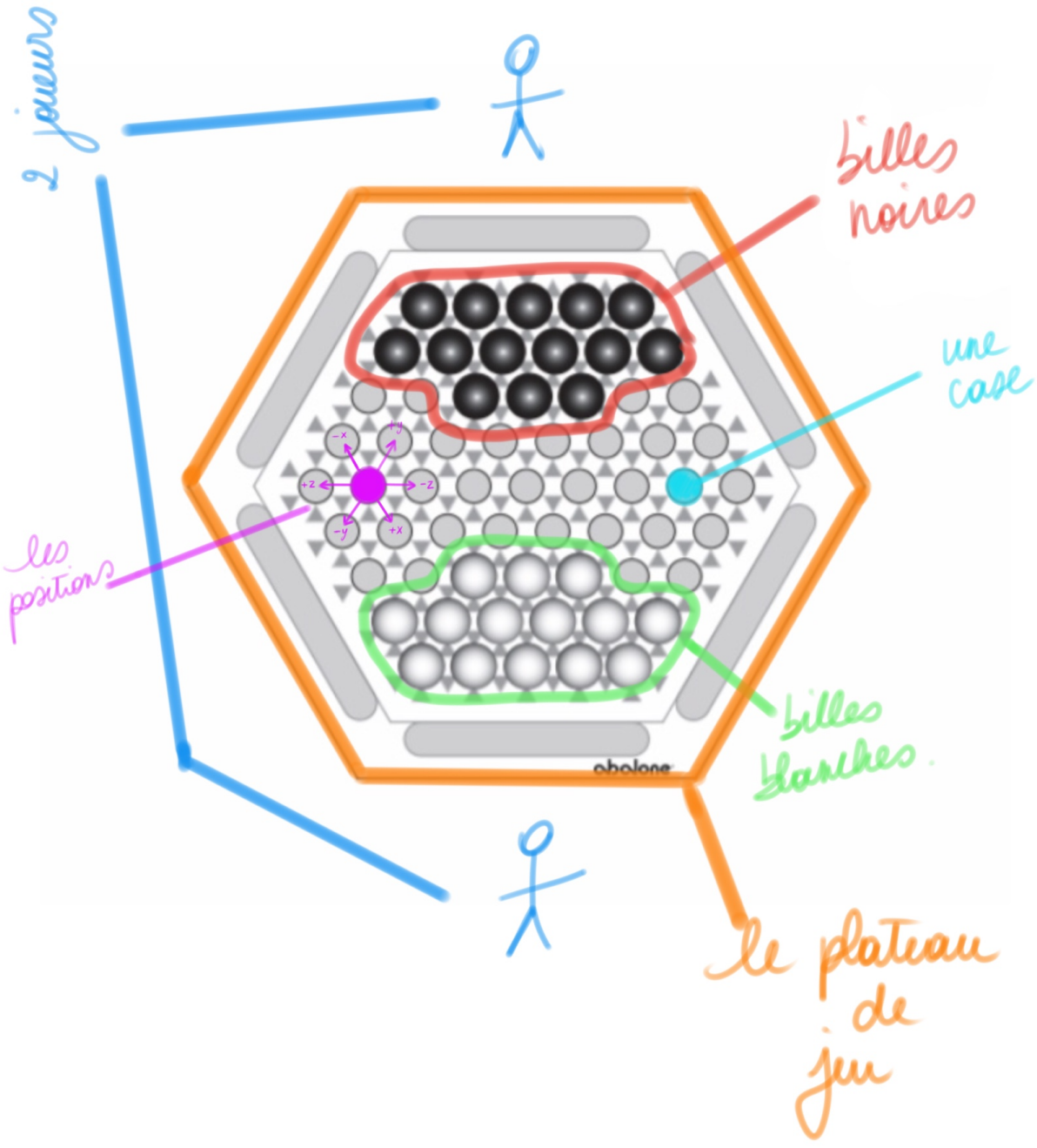
RapportRemise2

Donc pour commencer, nous allons parler de nos classes. Les éléments fondamentaux pour l'éxécution du jeu Abalone sont :

CLASSES

- Le support de jeu c'est-à-dire le plateau héxagonale.
- Des billes(noire ou blanche) à déplacer
- Les 2 joueurs pour effectuer une partie
- Les positions des billes à changer
- Les «cases» de notre support de jeu(libres ou pas)
- Le jeu qui montre le déroulement du jeu

ABALONE


2 joueurs
billes noires
une case
les positions
billes blanches
le plateau de jeu
abalone

ATTRIBUTS

- Marble :
  - ColorMarble color → la couleur de la bille (noire ou blanche)
  - AbaPro aba → le déplacement d'une bille par rapport à une case
- Player :
  - PlayerColor colorMarble → qui définit la couleur du joueur(donc de la bille) càd noire ou blanche
  - marble[ ]: Marble → est un tableau dans lequel on stocke toutes nos billes
  - Status playerStatus → le statut du joueur en fin de partie (gagnant, perdant ou neutre)
  - Direction → une énumération représentant les directions dans lesquelles le joueur va pouvoir déplacer sa bille (N,S,W,E,NE,NO,SE,SO)
- Game :
  - Board board → le plateau de jeu
  - Player player1 & Player player2 → ce sont les 2 joueurs participant à la partie(joueur 1 et joueur 2
- Board :
  - Case [][] board → un tableau à 2 dimensions qui représente le plateau de jeu, un tableau de cases
  - marble Marble → une bille
- Case :
  - Color color → couleur de la case faisant appel à l'énumération (case possédant une bille donc ayant la couleur noire ou blanche ou étant une case vide càd sans bille donc empty)
- Position:
  - x,y&z → c'est le déplacement d'une bille par une position donnée

- AbaPro :
  - int column et char row→ déplacement dans le tableau grâce aux lignes et colonnes

---

METHODES

- **Marble :**
  - Marble (ColorMarble:color)

  Un constructeur de la classe Marble qui crée une boule d'une certaine couleur

  - ColorMarble getColorMarble();

  Un getteur de la classe Marble retournant la couleur de la bille de type ColorMarble

  - AbaPro setPositionMarble( ) ;

  Méthode modifiant l'état de la bille (si elle est éjectée ou non)

  - AbaPro getPositionMarble();

  Getteur de la position de la bille de notre jeu

- **Player :**
  - ColorMarble getPlayerColor( );

  Getteur de la couleur des billes du joueur

  - move(Direction direction,Marble marble);

  Méthode qui permet au joueur d'effectuer un mouvement sous base d'une bille et direction donnée

  - Marble getMarble( );

  Getteur de la bille

  - remove(Marble marble);

Méthode qui retire de notre tableau de billes la bille éjectée du plateau de jeu

- `Player(ColorMarble marble)`

Constructeur de Player avec comme paramètre la couleur des billes qu'il possèdera

- `int scorePlayer(Marble[ ]marble);`

Méthode qui calcule le score du joueur sous base de 2 paramètres (le joueur et son nombre total de billes)

- `PlayerStatus getStatus( );`

Getteur du statut du joueur càd si il est gagnant, perdant ou neutre(càd statut durant la partie)

- `PlayerStatus setStatus( );`

Setteur du statut du joueur

- **Game :**
  - `void restartGame( );`

  Méthode qui demande au joueur si il veut recommencer

  - `void newGame();`

  Méthode qui recommence la partie

  - `Game()`

  Constructeur de Game, construit une game

  - `initialiseGame( );`

  Méthode qui initialise la partie de jeu

- **Board :**
  - `Board(Marble marble, Case[][]board)`

  Constructeur du board sous base de billes et de son tableau de cases

  - `Marble getMarble( );`

  Getteur de mes billes du board

- Status isInside (Case[][] board);

  Méthode qui retourne si à une case du tableau est vide ou non → (dont l'énum Status)

- **AbaPro :**
  - int getColumn();

    Getteur de la colonne

  - char getRow( );

    Getteur de la ligne

  - AbaPro(char row,int column)

    Constructeur de l'AbaPro

- **Case :**
  - Color getColor();

    Getteur de color

  - Color setColor();

    Setteur de color

- **Position :**
  - int getX( );

    Getteur de la variable x

  - int getY( );

    Getteur de la variable y

  - int getZ( );

    Getteur de la variable z

  - Position(int x,int y,int z);

    Constructeur qui crée une position qui va pouvoir être utiliser pour déplacer notre bille

  - Position checkNeighbor(Direction direction)

Méthode vérifiant si une bille possède des voisins et à quelle position avant le déplacement de notre bille

- `sumito( );`

Méthode regroupant les différents cas lorsqu'un joueur décide de pousser les billes de son adversaire (par supériorité numérique)

- `pac( );`

Méthode reprenant le cas d'une égalité numériques de billes entre les 2 adversaires

---

POUR LA REMISE DU JEU EN CONSOLE, NOUS AVONS EFFECTUER QUELQUES CHANGEMENTS :

Tout d'abord, un gros changement apporté est que l'on a supprimer la classe Marble car ceci apportait de la redondance. Nous avons créer la classe Marble dans le but de dire qu'une bille a une certaine couleur et pour pouvoir dire qu'un joueur a un nombre de billes. Cependant, nous avons eu de la redondance car dans la classe case nous avons dit qu'elle possède un attribut color donc notre choix de créer une classe marble était inutile sachant que nous pouvons calculer combien de billes a chaque joueur par un simple compteur initialiser à 14 et qui serait décrémentée lors de l'éjection d'une bille.

Les méthodes donc de Player associer à une liste de billes est maintenant faite via un compteur qui est cmptscore qui lorsqu'une bille est éjectée du plateau on décrémente notre compteur initialiser à 14.

Aussi, nous avons déplacer les méthodes de mouvements des billes dans la classe Game car nous considérons que les mouvements se font durant le jeu et donc serait mieux placer dans la classe Game que dans la classe board. Aussi, dans les méthodes de mouvements nous avons souvent besoin en paramètres d'un joueur et la classe Game possède cet attribut contrairement à la classe Board.

Dans la classe Position, nous avons ajouter des opérateurs pour par la suite additionner,comparer,soustraire des positions entre elles.

Et enfin, de nombreuses méthodes au bon fonctionnement du jeu ont été implémentées.

Temps estimé pour l'ensemble de ce travail : 85h de travail consacré pour cette remise en console