Piéces d'échiquier:

Les pièces d'échiquier sont représentés par des chaines de deux caractères. Le premier charactére représente **le type de la pièce** et le deuxiéme **sa couleur**.

ex : une tour noire "TN"

lettre	couleur
N	Noir
B	Blanc

lettre	type
K	Roi
Q	Reine
F	Fou
C	Cavalier
T	Tour
P	Pion

Échiquier:

l'échiquier est définie par une liste (de 8 liste (de 8 liste (qui seront soit vide, soit contenant une string de 2 caractère(un pion))))et un int qui sera le numéro du tour

ex : - echiquier = [[[“TB”],[],[“KN”],[],[],[],[],[]], [[],[],[],[],[“QB”],[],[],[]], [[],[],[],[],[],[],[],[]], [[],[],[],[],[],[],[],[]], [[],[],[],[],[],[“KB”],[],[]], [[],[],[],[],[],[],[],[]], [[‘PB’],[],[],[],[],[],[],[]], [[],[],[],[],[],[],[],[]]], 0]

Les cordonnées d'une pièce

les cordonnées d'une pièce sont définie par un tuple (contenant une lettre majuscule(A B C D E F G H)et un chifre(1 2 3 4 5 6 7 8))

ex: (“B5”)

##Les mouvements de pièces

le mouvement d'une piece est définie par un tuple de tuple contenant les coordonées de départs et d'arrivées.

ex: - ((“A”,“5”),(“B,”6"))

etat_partie

variable pouvant 1 string parmit 5 string diferante : “partie_en cours” “victoire_J1” “victoire_J2” “pat” “égalité”

Les Class

class Moteur

description : cette class gère la partie.

`variable echiquier` :L'echiquier qui représente le status de la partie.

`variable etat_partie` :donne l'état de la partie sous la forme de variable : si il y a pat, échec, échec et mat, égalité ou rien.

`méthode get-Echiqier() : echiqier` :renvois echiquier.

`methode get_etat-partie` :renvoie etat_partie

`methode lancement(string) : rien` :place tout les paramètres dans leur état de base

`methode coupValide(mouvement) : boolean` :Renvois True si le mouvement présenté est possible, sinon renvois False.

`methode gestionCoupValider(mouvement) : echiqier`

effectue toutes les modifications nécéssaire à l'application du coup. soit: - modifier l'échiquier - tour +1 - stocker l'ancien echiquier - verifier si etat_partie a besoin d'etre modifier

`méthode connaitre_tour(echiquier) : "Blanc" ou "Noir"` :permet de connaître à qui c'est le tour actuellement (sert par exemple à savoir qui est en échec).

`méthode est_en_danger(coordonées_de_pièce): liste de tuples` :permet de connaitre quels pièces peuvent manger la pièce analysé. Renvoie une liste coordonées des pièces qui peuvent manger la pièce mise en paramètre (ex : [(0,7),(7,7)])

`méthode getNextPiece(coordonées_de_pièce, direction_a_analyser): coordonées de la pièce la plus proche dans la direction choisi` :permet de savoir où se situe la pièce la plus proche dans la direction choisi. La direction est choisi selon ces chiffres : x,y. Exemple, pour aller dans la diagonale de droite vers le bas, on entre (-1,1).

class Interface

description : cette class gére l'interface utilisateur.

`1- demarage(self)` :Renvoie la composition de la partie (nombre de joueurs/robots et leur couleur) et appelle init du moteur avec ce paramètre.

`2- generationEchiquier(self,echiquier)` :Affiche l'echiquier actuel

`3- partieFini(etat_partie)` :Vérifie si la partie est fini, et si oui quelle fin en la demandant au moteur

`4- appelerRobot(self)` :vérifie si c'est le tour du robot et si oui, demande un coup au robot et l'enrengistrer dans une variable coup_joueur

`5- demanderCoup(self)` :Renvoie le coup du joueur si c'est son tour et l'enrengistrer dans une variable coup_joueur

`6- verifierCoup(self,coup_joueur)` :Appelles la fonction du moteur qui sert à vérifier si le coup en paramètre est valide. Sinon, appelle une fonction qui affiche un message d'erreur. Appelle aussi une fonction qui "traduit" le mouvement (ex : (("A5"),("B6"))) en tuple de coups (ex : ((0,5),(1,6))).

`7- traducteurHumainMachine(self,coup_joueur)` :Traduit les coordonées entrée par le joueur pour la machine (ex : ("A5")) en tuple de coups (ex : (0,5))). Le processus est répété autant de fois que nécéssaire.

`8- messageErreur` :Est appellée quand le coup_joueur est invalide. Affiche en conséquence un message d'erreur.

`9- modifierEchiquier(self,coup_valide)` :Envoie coup_valide au moteur puis appelle *mouvement*.

`10- mouvement(self,coup_traduit)` :fait les changements sur l'échiquier graphique