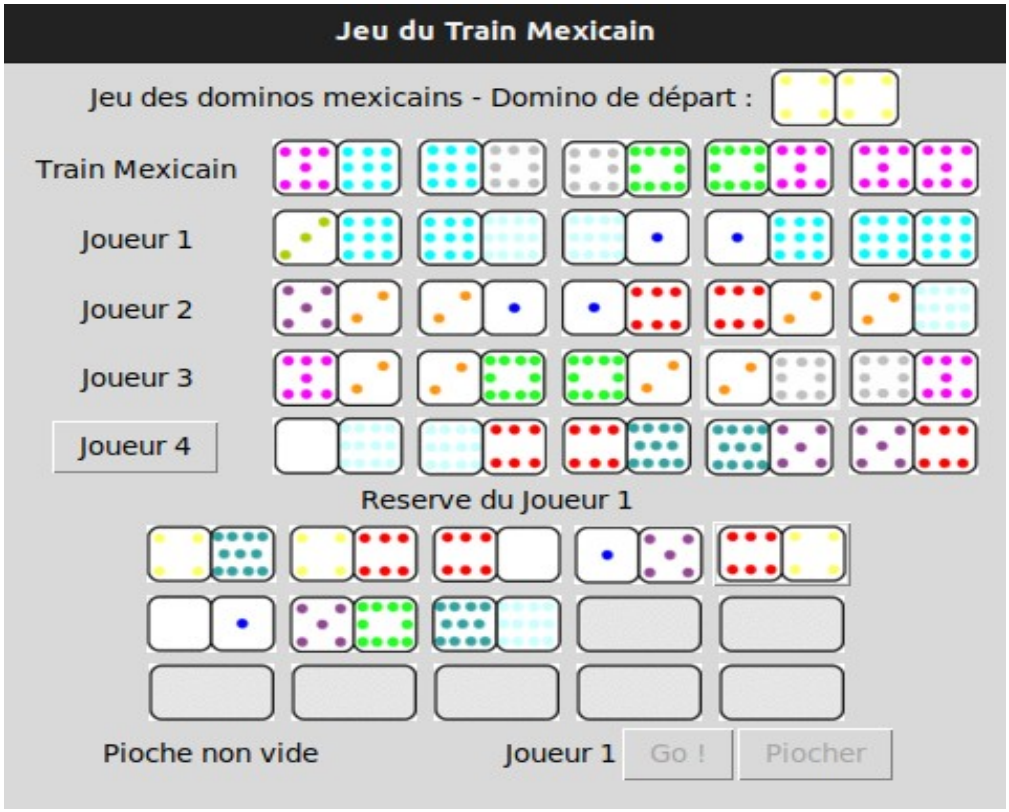
# Projet Jeu Mexicain

Explications du fonctionnement




## Semaine 5

***Nom : KATCHALA MELE*** ***Prénom : Abdoulaye***

***Groupe : TP 1-A*** ***Année : 2022-2023***

# Modélisations

Les classes de ce jeu sont écrites chacune dans leurs propres fichiers. Le jeu est modélisé par la classe ***JeuMexicain***, les joueurs par la classe ***Joueur***, la vue et le contrôleur par les classes ***VueTrain et Contrôleur***.

# Explication brève des fonctions

## *JeuMexicain*

La classe du jeu mexicain a pour attributs le ***__dominoDepart, __trainMexicain, et __pioche*** qui représentent respectivement le domino de départ, le train mexicain (une Pile), et la pioche (une Pile).

Le constructeur de cette classe crée aussi tous les dominos nécessaires pour le jeu, les mélange puis les empile dans la pioche.

```
# Création des dominos
dominos = []
for i in range(13) :
    for j in range(13) :
        dominos.append(Domino(i,j))

# Melange des dominos
for i in range(1000) :
    a,b = randint(0,168),randint(0,168)
    dominos[a],dominos[b] = dominos[b],dominos[a]

# Empile les pioches
for domino in dominos :
    self.__pioche.empiler(domino)
```

Les méthodes de cette classe sont :

- **est_choisi_depart(self)** qui teste si le domino de départ a été choisi
- **dernier_domino(self)** qui renvoie le domino au sommet du train mexicain

- **pose_domino(self,domino)** qui empile le domino en paramètre dans le train mexicain
- **pioche_est_vide(self)** qui teste si la pioche est vide ou pas
- **pioche(self)** qui permet de piocher dans la pioche en retournant le dernier domino de la pioche tout en le dépilant
- **train_a_demarre(self)** qui teste si le train a démarre c'est à dire le train mexicain n'est pas vide
- **domino_depart(self)** qui retourne le domino de départ afin de l'afficher
- **choisi_domino_depart(self,domino)** qui permet de choisir le domino de départ en affectant à celui ci le domino en paramètre.
- **str_dominos(self)** retourne la liste des strings des cinq derniers dominos du train mexicain. Cette fonction crée une nouvelle pile dans laquelle elle met les cinq premiers domino du train. Ensuite elle les remet en prenant leurs strings avec la méthode __str__ de la classe domino. Par finir elle retourne la liste de ces strings. Elle me permet de gérer le débordement dans l'affichage du train.
- **img_dominos(self)** retourne la liste des coordonnées d'image représentant les cinq derniers dominos du train mexicain. Cette fonction agit comme str_dominos mais elle prend cette fois ci les coordonnées des images qui correspondent au couple (FaceA, FaceB) du domino. En effet les images sont stockés dans une matrice.
- **empile_pioche(self,domino)** empile le domino passé en paramètre dans la pioche. Elle me permet de recommencer le jeu en empilant tous les dominos des joueurs dans la pioche.
- **recommencer(self)** empile tous les dominos du train mexicain dans la pioche plus celui de départ. Je l'ai écrite afin d'avoir la possibilité de recommencer le jeu.

### *Joueur*

La classe des joueurs possède 3 attributs : ***__train, __reserve, __jeu et __open*** qui représentent respectivement le train du joueur (une Pile), la réserve du joueur, le jeu mexicain et un booléen qui dit si le joueur est

ouvert. Un joueur est ouvert si il pioche; ainsi les autres joueurs peuvent mettre leur dominos dans son train.

Les méthodes de cette classe sont :

- **est_open(self)** qui dit si le joueur est ouver ou pas.

- **train_a_demarre(self)** qui teste si le train a démarré.

- **dernier_domino(self)** retourne le dernier domino du train du joueur.

- **est_posable_train(self,domino)** teste si le domino en paramètre est posable sur le train du joueur ou pas tout en vérifiant si le train a démarré.

- **possible_train(self,domino)** a le même travail que le précedent mais vérifie aussi pour l'inverse du domino.

- **est_posable_train_mexicain(self,domino)** teste si le domino en paramètre est posable sur le train mexicain ou pas tout en vérifiant si le train mexicain a démarré.

- **possible_train_mexicain(self,domino)** a le même travail que le précedent mais vérifie aussi pour l'inverse du domino.

- **possible_train_joueurs(self,domino,joueurs)** parcours joueurs (qui est un dictionnaire contenant les joueurs ouverts) pour voir si on peut placer domino dans un des trains des joueurs. Il retourne un booléen qui confirme.

- **nbre_domino_reserve(self)** retourne le nombre de domino dans la réserve du joueur.

- **nieme_domino(self,n)** retourne le n-ième domino de la réserve du joueur.

- **plus_grand_domino_double(self)** parcours la réserve du joueur pour trouver le plus grand domino qui est double et retourner son indice. Si il ne le trouve pas il retourne -1. Cette fonction me permet de choisir le domino de départ.

- **PDST(self)** retourne l’indice du premier domino posable ou inversement posable sur le train du joueur. Cette fonction parcours la réserve et retourne l’indice du domino aussitôt qu’il l’ait retrouvé. Si il ne trouve pas de domino posable ou inversement posable, il retourne -1. Elle permet de faire jouer les joueurs artificiels.

- **PDSTM(self)** retourne l’indice du premier domino posable ou inversement posable sur le train mexicain. Elle procède comme la fonction précédente et permet de faire jouer les joueurs artificiels.

- **PDSTJ(self,joueurs)** retourne un couple contenant le joueur ouvert et l’indice du premier domino posable sur le train du joueur ouvert. Pour chaque domino dans la reserve elle parcours les joueurs ouvert pour voir si il y’a compatibilité. Si oui elle retourne directe.

- **pose_domino_train(self,domino)** empile le domino en paramètre dans le train du joueur afin de l’ajouter.

- **pose_domino_train_mexicain(self,domino)** empile le domino en paramètre dans le train mexicain afin de l’ajouter.

- **pioche(self)** permet au joueur de piocher en appelant la méthode pioche du jeu. Dès que le joueur pioche, il devient ouvert.

- **str_dominos(self)** retourne la liste des strings des cinq derniers dominos de la pile. Elle procède comme la méthode de la classe JeuMexicain et me permet de gérer le débordement dans l’affichage du train du joueur

- **img_dominos(self)** retourne la liste des coordonnées d’image représentant les cinq derniers dominos du train mexicain. Elle procède comme la méthode de la classe JeuMexicain.

- **supprimer_domino(self,ind)** supprime le domino à l’indice ind de la réserve.

- **donner_domino_depart(self)** appelle la méthode de la classe qui permet de trouver l’indice du plus grand domino double dans la réserve qui va représenter le domino départ dans le jeu. Si elle

trouve le domino, elle appelle la méthode choisi_domino_depart du jeu afin d'affecter le domino choisi au dominoDepart et elle supprime le domino de la réserve afin d'éviter une duplication. Si non le joueur pioche.

- **dominos_posables(self,joueurs)** parcours la réserve afin de retourner la liste d'indices des dominos posables ou inversement posable sur le train du joueur, sur le train mexicain ou sur l'un des trains des joueurs (en appelant la fonction possible_train_joueurs). Cette fonction me permet d'ajouter une commande aux boutons qui représentent les dominos posables.

- **jouer(self,joueurs)** permet aux joueurs artificiels de jouer. Elle regarde si il y'a un domino posable ou inversement posable sur le train du joueur. Si oui il dépose tout en regardant si c'est son inverse qui est valide ou non. Si le domino posé est double, il recommence. Si il n'y a pas de domino posable il regarde si il y'a un qui est posable ou inversement posable sur le train du mexicain. Si oui il le dépose et applique le meme principe, si non il regarde dans les joueurs ouverts. Si il trouve pas il pioche. Cette fonction permet aux joueur artificiel de jouer. J'ai favorisé plus le train du joueur que le train du mexicain et le train des joueurs ouverts. Elle retourne un booléen qui dit si le joueur a pioché ou pas. Ce booléen me permet de savoir si le jeu est en boucle

- **a_gagne(self)** qui teste si le joueur a gagné c'est à dire si sa réserve est vide.

- **recommencer(self)** empile tous les dominos du train du joueur et de la réserve dans la pioche. Je l'ai écrite afin d'avoir la possibilité de recommencer le jeu.

- **permuter(self,ind)** permute le domino à l'indice ind dans la réserve. Elle me permet de gérer le double clic.

### *VueTrain*

Le constructeur de la vue prend en paramètre le jeu et les joueurs. La classe possède comme attributs ***.__fen*** (la fenêtre), ***.__jeu*** (le jeu), ***.__joueurs*** (liste des 4 joueurs), **.__images** (matrice contenant les images), ***.__label6*** (pour afficher le nom de la reserve), ***.__label7*** (pour

afficher l'etat de la pioche), ***.__btns_choix*** (boutons pour choisir le train où deposer) ***.__btn_depart*** (pour afficher le dominoDepart), ***.__btns_train*** (liste de listes de boutons pour les 5 derniers dominos du train mexicain et des joueurs), ***.__btn_go*** (Bouton permettant au joueur de jouer), et ***.__btn_piocher*** (bouton permettant au joueur de piocher).

Le constructeur crée une frame pour le bouton départ, une frame pour l'affichage des trains, une frame pour l'affichage de la réserve, et une frame pour l'affichage du bouton go et piocher. Il se charge aussi de la création de tous les images, les labels et les boutons sans leur mettre les commandes et en leur mettant comme image le vide.

Les méthodes de cette classe sont :

- **affiche_btn_depart(self)** permet d'afficher le dominoDepart. Elle prend les coordonnées de l'image puis le met sur le bouton.
- **les_boutons(self)** renvoie le bouton GO, le bouton piocher, la liste des boutons de la réserve et la liste des boutons de choix afin que le contrôleur puisse leur mettre leurs commandes.
- **fen(self)** retourne la fenêtre pour lancer la boucle d'évènement.
- **valide_domino(self,ind,num_joueur,train)** prend en paramètre l'indice du domino posable ou inversement posable, le numero du joueur qui joue et le train dans lequel on met le domino. Elle prend d'abord le dominos pour verifier si il est posable sur le train (True) ou si c'est son inverse qui est posable (False). Si c'est l'inverse il va avertir le joueur.
- **met_a_jour_affichage(self,num_joueur)** prend en paramètre l'indice du joueur dans la liste des joueurs et permet de mettre à jour l'affichage après chaque tour en mettant à jour l'affichage des train, de la réserve et du label7 qui dit au joueur de jouer et si la pioche est vide.
- **met_a_jour_train(self,ind)** prend en paramètre l'indice du joueur et permet de mettre à jour l'affichage du train de celui ci. Elle récupère la liste des coordonnées des images représentant les 5 derniers dominos du train du joueur en appelant sa méthode img_domino puis met les images sur les 5 boutons.
- **met_a_jour_train_mexicain(self)** met à jour l'affichage du train mexicain. Elle récupère la liste des coordonnées des images représentant les 5 derniers dominos du train mexicain et met les images sur ses 5 boutons

- **met_a_jour_reserve(self,num_joueur)** prend en paramètre l'indice du joueur dans la liste des joueurs et permet de mettre à jour l'affichage de la réserve de celui ci. Cette fonction gère le débordement en créant des boutons si nécessaire et elle parcours les boutons de la réserve et met au bouton l'image du domino se trouvant à la même indice.
- **jeu_fini(self,gagnant)** gère la fin du jeu en déclarant le gagnant ou match nul dans le label7. Elle change le label, met à jour les trains du gagnant si il y en a un, désactive les boutons de la réserve et transforme le bouton GO en ''Recommencer ?''.
- **recommencer(self)** remet les boutons de la réserve à la normale et détruit les surplus de boutons c'est à dire si il y a plus de 15 boutons, elle les remets à 15 boutons. Elle remet aussi l'image du bouton départ à vide.

### *Contrôleur*

Cette classe permet de jouer au jeu et de le contrôler. Son constructeur crée le jeu et la liste des 4 joueurs, ensuite elle crée la vue avec les 2 précédents comme paramètre. Elle sauvegarde les boutons en les récupérant chez la vue et leur met leurs commandes qui sont des méthodes de la classe. Elle partage les dominos, choisi le domino de Départ, met à jour les joueurs ouverts, demande à la vu de se mettre à jour et lance la boucle d'évènement. Elle initialise aussi les attributs **self.__boucle** (qui dit si le jeu est en boucle)**, self.__ tour** (qui désigne le joueur qui joue), **self.__nbre_humain** (le nombre de joueurs humains) et **self.__opened** ( un dictionnaire qui contient les joueurs ouverts à chaque tour). Le jeu est en boucle si la pioche est vide et que tous les joueurs piochent.

Les méthodes de cette classe sont :

- **partage_domino(self)** qui partage les dominos au joueur c'est à dire chaque joueur pioche dix fois.
- **choisi_domino_depart(self)** permet de choisir le domino de départ du jeu. Elle choisit au hasard le joueur qui choisit le domino de départ. Elle lui demande de donner le domino de départ en appelant sa méthode donner_domino_depart. Tant que le domino de départ n'est pas choisi elle demande au prochain joueur.
- **met_commande_a_jour(self,num_joueur)** met les commandes des différents boutons à jour en changeant les différents

paramètres de fonctions. Elle met aussi le double clic sur les boutons de la reserve.

- **met_a_jour_open(self)** met à jour les joueurs ouverts. Il vide le dictionnaire et parcours les joueurs pour y mettre ceux qui sont ouverts.
- **controle_GO(self,num_joueur)** il s'agit de la commande associé au bouton GO. elle prend en paramètre l'indice du joueur dans la liste des joueurs. Elle affiche le bouton de départ, puis demande au joueur l'indice des dominos posables sur le train mexicain, sur son train ou sur les trains des joueurs ouverts. il parcours ces indices et lève les bouton à ces indice dans la réserve et leurs met la commande *controle_reserve*. Si il n'y a eu aucun indice il active le bouton piocher qui permet au joueur de piocher.
- **controle_piocher(self)** il s'agit de la commande associé au bouton piocher. Elle commence par mettre à jour self.__boucle. Elle dit au joueur de piocher, active le bouton go et désactive le bouton pioche. Ensuite elle met à jour les joueurs ouverts et verifie si c'est à un joueur humain de jouer. Si non, elle demande aux joueurs artificiels de jouer tour à tour en vérifiant à chaque tour si le joueur a gagné. Si un joueur a gagné, elle arrête le jeu. Si non elle met à jour l'affichage et vérifie si le jeu est en boucle.
- **controle_reserve(self,ind,num_joueur)** il s'agit de la commande affecté au bouton à l'indice ind de la réserve à l'appuie du bouton go. elle prend en paramètre l'indice du domino posable sur un des trains et l'indice du joueur dans la liste des joueurs. Elle commence par enlever les commandes de tous les boutons et ensuite prend le domino passée en paramètre. Elle active les boutons du choix des trains pour lesquelles le domino est posable ou inversement.
- **controle_train(self,ind,num_joueur,train)** Il s'agit de la commande mise aux boutons de choix de train. Elle prend en paramètre l'indice du domino à poser, le numéro du joueur et le numéro qui représente le train sur lequel on va mettre le domino. Elle demande à la vue si le domino est valide. Si oui, elle prend le domino et regarde sur quelle train il est posable, elle l'ajoute dans le train, elle remet les commandes à rien, supprime le domino et elle vérifie si le joueur a gagné. Si non elle demande au prochain joueur de jouer ou aux joueurs artificiels de jouer tour à tour en vérifiant si le joueur a gagné. Si oui elle arrête le jeu, si non elle

met à jour l'affichage. Elle vérifie aussi si le joueur a déposé un domino double. Dans ce cas elle va pas mettre à jour self.__tour.

- **double_clic(self,ind,num_joueur)** gère le double clic en demandant au joueur de permuter son domino à l'indice ind dans sa réserve et met à jour la vue.
- **jeu_fini(self,gagnant)** dit à la vue de des activer les boutons et remet la commande du bouton Go à recommencer
- **recommencer(self)** demande aux joueurs de remettre tous les dominos dans la pioche du jeu et demande à la vue de réactiver les boutons. Elle partage les dominos, choisi le domino de départ et met au bouton Go sa commande.

## Fonctionnement du jeu

L'exécution du fichier Controleur.py permet de jouer au jeu mexicain. Mettez le nombre de joueurs en paramètre de l'appel du controleur. Ainsi, le Controleur va créer le jeu, les joueurs et il demande à la vue de créer les composants de l'affichage. Ensuite il partage les dominos aux joueurs puis demande à un joueur choisi au hasard de donner le domino de départ. Il va attendre que le joueur qui doit joueur appuie sur le bouton GO.

Quand le joueur appuie sur le bouton Go, le dominoDepart va être affiché et Les boutons Go va être désactivé.

Si le joueur a des dominos posables dans sa réserve, les boutons associés à ces dominos vont se distinguer avec leur relief. Ainsi, un clique sur un de ces boutons va permettre au joueur de déposer son domino sur le train correspondant en lui demandant dans quel train mettre le domino. Si le domino n'est pas valide et que c'est son inverse qui doit être déposé, il demandra au joueur de permuter le domino. Ainsi un clique sur le bouton du train deposera le domino et mettra à jour self.__tour qui demandra au prochain joueur de jouer. Si le prochain joueur n'est pas humain, il dira aux joueurs artificiels de jouer, rendra le bouton Go activé et mettra à jour l'affichage.

Si le joueur n'a aucun domino posable, le bouton piocher va être activé. Un clique sur ce bouton fera piocher le joueur, mettra à jour self.__tour qui demandra au prochain joueur de jouer. Si le prochain joueur n'est pas humain, il dira aux joueurs artificiels de jouer, rendra le bouton Go activé et le bouton Piocher désactivé et mettra à jour l'affichage. Ainsi

de suite jusqu’à ce que un joueur vide sa réserve ou que le jeu soit en boucle, dans ce cas le jeu va s’arrêter en affichant le gagnant et en demandant au joueur si il vaut recommencer