Rapport de projet Domino et Carcassone

Bonjour, voici la présentation du projet de Yohan Leconte et Ronen Shay.

Nous nous sommes redistribué les différentes parties du cahier des charges qu'on a pu réaliser. Commençons donc par le menu qui a pour seul but de choisir le jeu qu'on veut sélectionner. Le menu est un JFrame assez simple décomposé en deux parties par un gridlayout auquel j'ai rajouté deux boutons: le bouton Carcassonne et le bouton Domino. Quand on appuie sur le bouton carcassonne ça lance un nouveau jframe qui est le jeu du carcassonne. Et quand on appuie sur le bouton domino ça réactualise le jframe de menu avec deux bouton l'un qui lance la version terminal du jeu domino et l'autre qui lance la partie interface graphique du jeu domino après avoir fermé le jframe du menu.

Ensuite nous avons décomposé Domino en 4 classes: Jeu.java, joueur.java,triple.java et Tuile.java.

L'objet triple n'est qu'un tableau de int de taille 3. L'objet Tuile.java est la représentation d'un domino il est constitué de 6 attributs: 4 attributs triple (nommé haut,bas,gauche,droite) 1 attribut Domino (qui est un un objet local Jframe dont je parlerais lors de la partie dédié a la partie graphique) et 1 attributs boolean vraieTuile qui détermine si c'est une tuile remplis ou vide. L'objet Joueur est constitué de 4 attributs: 1 attribut string nommé pseudos, 1 attribut int nommé score , 1 attribut Tuile nommé deck et un attribut boolean ia qui a pour but de déterminer si le joueur est un une ia ou un joueur(donc true si c'est une ia et false si c'est un humain). Et le dernier est en quelque sorte le lanceur du jeu il s’appelle jeu.java composé de 4 attributs: 1 attribut tableau de tableau de Tuile nommé plateau, 1 attribut tableau de Joueur nommé participant, 1 attribut qui est un int nommé tailleSac et un JFrame frame qui servira pour l'interface graphique seulement.

Nous allons donc commencer par décrire les différences classes et leur utilité pour le jeu de domino terminal fonctionne. La classe tuile a 2 constructeurs (un pour les tuile remplie et un pour les tuiles vides) et 2 méthodes: rotation qui tourne tous le domino d'un cran sur la droite et une méthode afficher qui affiche une tuile dans le terminal(en plus des getters et des setters). La classe Joueur posséde 2 constructeurs (l'un est pour la version terminal et l'autre pour la version graphique)

le premier ne concerne que le terminal est il utilise 2 scanner afin de demandé au joueur dans l'interface si le joueur créé est une ia ou un joueur (en écrivant 0 on obtient une ia et en écrivant 1 on obtient un joueur) et de demander le joueur son pseudos. garce au scanner on remplit l'attribut pseudos et ia du joueur et l'attribut deck est mis directement null et le score initialisé a 0.La classe joueur est composé aussi de 3 méthodes qui sont rotate() qui fais la rotation de la tuile de l'attribut deck, une méthode piocher() qui crée une tuile avec des valeurs aléatoires allant de 0 a 2 et qui l'attribut au deck du joueur et enfin la méthode afficher() qui crée une sorte de page qui explique l'état du joueur donc son score, son pseudos et montre ça tuile si il en a une dans le terminal. Nous avons aussi la class triple qui possède un constructeur et 2 méthodes l'une qui se nomme somme qui renvoie la somme des 3 entiers du triple et une méthode equalTriple(triple a) qui renvoie vrai si les triples de l'objet récurrent sont les mêmes et dans le bon sens que celui de a et faux dans le cas inverse.Ensuite nous avons la classe jeu qui fait tous les tests et qui fait le déroulement d'une partie je vais donc présenter ces méthodes et ensuite le constructeur qui fait tous le déroulement du jeu. Jeu.java possède 7 méthodes: plateauRempli() qui vérifie si le plateau est remplie ou non cette méthode permet de laisser les joueur tant qu'il que le plateau est pas encore complet. Une méthode affichage() qui affiche dans le terminal en fonction de comment est remplie le tableau de tableau de tuile (l'attribut plateau). La méthode estPossible(int i,int j) qui vérifie que si i et j sont bien dans le plateau (donc entre 0 et 8), renvoie faux si ce n'est pas le cas. La méthode TuileAutour(int i,int j) qui renvoie aussi un boolean, il renvoie vrai si il y a une tuile dans le plateau au dessus,en dessous, a droite ou a gauche et qui si ce n'est pas le renvoie faux. La méthode jouable(int i,int j, joueur A) qui vérifie que si on place la tuile dans le plateau[i][j] avec la tuile que le joueur A possède dans son attribut deck est jouable. c'est a dire il regarde si les tuiles autour de la tuile placé (si on la place) on les même triples que les tuiles autour (on utilise la méthode equaltriple() de triple pour cela.La méthode editScore(int i,int j,joueur a) qui vérifie ou se situe la tour de la tuile posée et édit le score du joueur a l'aide la méthode somme de la class triple. La méthode jeuxIa(joueur a) qui parcourt tous le plateau a l'aide de 2 boucles qui vérifie a chaque emplacement si y a pas une tuile a l'emplacement en question, si y a une tuile autour et si c'est jouable. si c'est le cas on place la tuile du l'ia sur le plateau on édit son score et on print dans le terminal qu'il a jouer. on refait toute ces opérations de recherche 4 fois car on utilise rotation sur la tuile 4 fois pour que chaque possibilité soit essayer, si on trouve aucune possibilité on set le deck de l'ia a null et on écrit dans le terminal qu'il a pas jouer. Maintenant passons au constructeur de jeu qui déroule le jeu. le constructeur jeu a

un boolean en argument si ce boolean est false le il lance la version interface graphique du jeu et si c'est un boolean true il lance la version terminal. la premiére chose que fait le constructeur c'est initialisé le plateau a un nouveau de tuile de taille [9][9] avec une tuile crée avec des valeurs aléatoires de 0 a 2 qui sera placé dans le plateau en [4][4].Ensuite il demandera a l'aide du terminal le nombre de joueur et un scanner récupérera l'entier récupéré, a l'aide de cette entier on peut créer le tableau de joueur car on connais la taille du tableau et donc on enchaîne en créant autant de joueur demandé avec le constructeur qui demandera au joueur si se sont des ia ou des joueurs et leur pseudo.Après nous demandons la taille du sac dans le terminal et on prend cette entier pour le placer dans l'attribut tailleSac de la classe jeu.

aprés ces formalités nous entrons enfin dans la partie jouable, en effet a présent les joueurs jouent chacun a leur tour tant que le taillesac est supérieur a 0 et que le plateau n'est pas encore rempli. en suite il vérifie pour chaque joueur si c'est une ia ou un joueur a l'aide de l'attribut ia. si c'est une ia on utilisela méthode piocher que l'ia posséde une tuile et tailleSac s'incrémente de -1 et après la méthode jeuxIa() fait le reste du boulot. quand il s'agit d'un joueur on le fait piocher aussi en faisant moins au tailleSac, ensuite on affiche les informations du joueurs, juste en dessous l’affichage du plateau et encore en dessous le terminal explique les options du joueurs qui sont jouer ça tuile en écrivant j, défaussez ça tuile en écrivant d ou tourner ça tuile sur la droite en écrivant r. Si il la tourne il peut encore refaire les trois possibilité décrite au dessus, si il écrit aucun des trois choix il sera écrit dans le terminal qu’il n'a écrit aucun des choix connu et lui demandera d'écrire tant qu'il n'a pas écrit d j ou r.Si il écrit j, alors le jeux demandera sur quel ligne et sur quel colonne il veut jouer et si il écrit a un endroit il ou ne peut pas jouer ou où il y a déjà une tuile il a le choix de défaussez en écriavant d remettre les coordonnés en réécrivant j ou tourner ça tuile avec r. si il tourne ça tuile elle s'affichera dans le terminal a chaque rotation.si il joue et que toute les les conditions sont réunis pour qu'il joue alors on place ça tuile on édite le score du joueur et on set son deck a null et c'est au joueur suivant. Voici le déroulement d'un partie de Domino terminal passons a la version graphique.

avant d'expliquer le déroulement de la partie graphique nous devons voir la class local domino qui se situe dans tuile.java. cette classe domino est un jpanel qui est décomposé en un gridlayout de 5,5 pour modélise le domino de la tuile créé dans le

constructeur, il prend comme argument une tuile.Si c'est une tuile vide il remplacera toute les valeurs par un "x".c'est pour cela que que nous avons deux constructeur de tuile: un pour les tuiles remplies et un autre pour les tuiles vides.d'ailleurs la méthode rotation de tuile affecte aussi le domino.Les autres changements qui arrive avec l'interface graphique sont situés dans jeu.class. Il y a 4méthodes dédiés a l'interface graphique uniquement se sont : la méthode tour(int a) qui commence par removeall() de l'attribut frame et qui a pour rôle de s'occuper des tour de chaque joueurs. en effet si c'est une ia il fera comme la version terminal sans rien montré et relancera la même méthode tour mais avec le joueur suivant.Quand c'est un humain qui aura accés au frame qui sera découpé en deux par deux jpanels: sur le jpanel de il y aura une modélisations du plateau déjà gauche créé par la méthode remplir(Jpanel a) qui se réactualisera a chaque fois que ça sera au joueur suivant et le jpanel de droite est une sorte d'affichage des options du joueurs créé par la méthode profilplayer(joueur a,JPanel b) qui modifie le jpanel en un borderlayout(). ce borderLayout a le pseudos du joueur sur la partie NORTH du borderlayout et son score dans la partie SOUTH, la modélisation du domino que le joueur posséde au CENTER, un bouton jouer au WEST et un bouton Defaussez a EAST. Le joueur pourra interargir sur les boutons jouer et défaussez. si il clique sur défaussez la méthode se relance et ça sera au joueur suivant. Et si ils cliquent sur jouer le jpanel droit du frame va modifier le NOTH en un JAreaText ou il devra écrire la colonne ou il veut écrire pareille pour la partie SOUTH mais pour les lignes et la partie CENTER se change en un gridLayout 2,1 ou on a la modélisation du domino est un nouveau bouton nommé rotation qui va faire le tourner le domino a chaque clique. le WEST reste le bouton qui marchera que si les coordonnés donner par le joueur son jouable est pas un endroit ou il y a deja une tuile il passera aprés au joueur suivant. J'ai donc parlé de ce que faisais les méthodes remplir() et la méthode profilplayer mais il reste la méthode creatjoeur(int a, Joueur[] b) cette méthode permet de configuré le frame de jeu en trois en un gridlayout(3,1) aprés avoir removeAll() . c'est trois espaces sont 2 boutons (un avec marqué ia et l'autre avec marqué humain) et un JAreaText pour mettre son pseudos. quand tu cliques sur le bouton humain tu prends le pseudos écrit et cela crée le joueur avec le pseudos écrit ça définit si c'est un humain ou un joueur en fonction du bouton appuyer et cela se répéte pour le nombre de joueur sélectionner.

Parlons a présent du lancement du jeux en mode interface graphique. Quand on lance la version graphique alors notre jframe est composé de deux JSlider : l'un qui va de l'entier 1 a 5 afin de sélectionner le nombre de joueur et le deuxiéme allant de

1 a 100 qui représente la taille du sac . et il est composé d'un bouton qui va permettre de passer a l'étape suivante aprés avoir mis les valeurs des jslider dans Nombre de participant et tailleSac. aprés cela le jframe removeAll() et utilise la méthode creatjoeur() pour créer les joueurs et quand cette méthode sera terminé la méthode tour() est lancé jusqu'à ce que la partie soit terminé. ce qui représente l'intégralité de la partie.

Nous avons ensuite réalisé le jeu Carcassone. Pour cela, nous avons d'abord crée une interface Lieu que nous reprenons ensuite dans les classes Routes, Champs, Ville et Abbaye.

Nous avons également crée une classe Tuile qui possède 5 attributs de type Lieu pour le haut, le bas, la gauche, la droite et la milieu de la Tuile. Cela nous a permis de vérifier lorsqu'on ajoute une nouvelle tuile au plateau, que si on a par exemple un route sur le coté gauche de la tuile qu'on rajoute, il y en a également une sur le coté droit de la tuile à coté de laquelle on veut la placer. La tuile a également pour attribut les images qui la représentent, l'image courante, un Jpanel qui contient l'image courant afin de la représenter graphiquement, les coordonnées de la tuile dans le plateau, 4 boutons respectivement à gauche, à droite, en haut et en bas afin de placer d'autres tuiles à coté, le plateau de jeu et le potentiel meeple placé sur la tuile. La classe contient une méthode pour entourer la tuile avec les boutons s'il n'y a pas de tuiles ces emplacements, des méthodes tourneDroite et tourneGauche qui permettent de faire pivoter les tuiles, ainsi que les fonctions bouton et poseTuile qui permettent d'ajouter les bons Listeners aux boutons selon la tuile piochée et ainsi d'ajouter une nouvelle tuile au plateau lorsqu'on appuie sur un des boutons placer entourant la tuile.

Afin de représenter les tour du jeu, nous avons crée les classes Plateau qui possède comme attribut un tableau de 2 dimension pour contenir les Tuiles posés au fur et à mesure du jeu. Une fois tous ces éléments mis en place nous avons enfin pu commencer l'implémentation graphique du jeu. C'est la classe Jeu (qui est donc la classe principale) qui va s'occuper de la partie graphique du jeu. Nous avons décidé d'utiliser le Composant Jpane avec principalement des BorderLayout ainsi qu'un

GridLayout pour représenter le plateau de jeu. La fenêtre est composée de 2 parties : une pour le plateau, et une pour la pioche.

Les méthodes seront légèrement différentes selon si c'est contre une IA ou un autre joueur que l'on souhaite jouer. Pour cela, nous avons une variable booléenne «ia» qui spécifie si l'on joue bien contre une IA ou un humain. La variable statique joueur permet quand à elle de savoir c'est au tour de qui de jouer. Cette variable est mise à jour à chaque nouveau tour de jeu. Le programme débute par la méthode initPile qui va initialiser la pioche avec les cartes du jeu. C'est ensuite la méthode build qui est appelé. Cette dernière va mettre en place tout l'environnement graphique que l'on va utiliser tout au long du jeu. Elle va ainsi créer les principaux Jpane et Layout sur les quelles seront posés les tuiles. On initie un GridLayout qui représentera le plateau de jeu, ainsi qu'un BorderLayout qui représente la pioche avec la carte piochée en son centre et les boutons qui permettent la rotation de cette carte à l'est et à l'ouest, grâce aux listeners qui vont appeler les méthodes tourneDroite et tourneGauche sur la tuile courante. Nous avons ensuite la méthode piocher qui comme son nom l'indique va aller chercher une carte aléatoire dans la pile et remplacer l'ancienne tuile courante par celle qu'on vient de piocher en mettant à jour les variables concernées. Elle va également appeler la méthode pose qui va activer les listeners des boutons placer et mettre à jour le joueur courant. Concernant le placement des meeple, nous avons crée une classe Meeple qui implémente une boite de dialogue dans laquelle joueur pourra choisir s'il veut placer un meeple et si oui, où il veut le placer sur la tuile. Cette boite se lance dès qu'un joueur place une tuile, puis on superpose les deux images en une nouvelle qu'on place sur le plateau Si c'est une IA qui joue, elle va utiliser les méthodes statiques de la classe Robot pour d'abord faire une boucle en cherchant toutes les possibilités de placer une tuile, puis choisi une possibilité au hasard parmi celles-là. Enfin, la classe Robot possède une dernière méthode statique qui permet à l'IA de choisir si elle veut placer un meeple et si oui, où elle veut le placer. Pour cela, nous utilisons un Random.nextBool pour le booléen qui détermine si oui ou non elle veut placer un meeple et nextInt qui indique l'emplacement de ce dernier.

Au niveau du cahier des charges le menu respecte bien le choix des jeux, on peut choisir le nombre de joueur dans la version terminal et graphique on peut aussi choisir se sont des humains ou des ia. nous possédons aussi un mode graphique et texte pour domino et une version graphique de Carcassone.