isep
École d'ingénieurs du numérique

ISEP 2018-2019

JAVA – Domi'Nations

Carla Deruelle, Anaïs Perrier

I. Rappel du Sujet

1. Présentation générale du jeu

Pour ce tout nouveau projet de JAVA, il s'agissait de réaliser un jeu de dominos baptisé Domi'Nations. Domi'Nations est jeu de placement et de plateau pour 2 à 4 joueurs. Le but du jeu est de constituer un plateau de 5 cases par 5 en utilisant des dominos.

Les dominos sont composés de deux types de terrains (qui peuvent tout à fait être les mêmes) ainsi que d'un nombre de couronnes pouvant aller de 0 à 3. Nous distinguons 6 types de terrains différents en tout. Les joueurs constituent leur plateau personnel en choisissant des dominos par un système de pioche et en les plaçant autour de leur château à côté de dominos de même type de terrain. Les couronnes et les types de terrains serviront par la suite à déterminer le score d'un joueur. Le joueur qui obtient le plus grand score gagne la partie.

2. Déroulement et règles du jeu

Le jeu est constitué de 48 dominos, de 4 rois et de quatre châteaux.

- Pour un jeu à 2 joueurs : 24 dominos sur les 48 seront nécessaires, ainsi que 4 rois et 2 châteaux (un joueur possèdera donc 2 rois) ;
- Pour un jeu à 3 joueurs : 36 dominos sur les 48 sont nécessaires, ainsi que 3 rois et 3 châteaux ;
- Pour un jeu à 4 joueurs : les 48 dominos, les 4 rois et les 4 châteaux sont nécessaires.

La partie commence par la constitution de la pioche de dominos. Pour un jeu à 4 joueurs, il faudra prendre 4 dominos au hasard parmi les 48 (la pioche sera constituée de 3 dominos pour un jeu à 3 joueurs et de 4 dominos pour un jeu à 2 joueurs). Les dominos sélectionnés pour la pioche sont disposés face cachée. Ils possèdent tous un numéro sur leur face cachée. Il faut alors les trier du plus petit au plus grand puis les retourner face visible.

Un des joueurs tire au hasard les rois et les sort les uns après les autres. Le joueur dont le roi a été tiré en premier choisit en exclusivité sur quel domino se placer pour pouvoir ensuite le disposer autour de son château. Place ensuite au joueur dont le roi a été tiré en deuxième position et ainsi de suite. Chaque joueur ayant maintenant choisi un domino, il faut de nouveau piocher un nombre de domino selon le nombre de joueurs. Une fois les dominos piochés, il faut les trier par ordre croissant puis les retourner face visible. A ce moment-là, le joueur ayant choisi comme premier domino, le domino de plus petit numéro, est le premier à choisir son deuxième domino. Il peut alors placer son premier domino obligatoirement autour de son château. C'est ensuite

au joueur ayant choisi le deuxième domino de plus petit numéro à choisir son second domino. Ainsi de suite. Cette même opération se répète jusqu'à ce qu'il n'y ait plus de dominos à piocher. Le jeu est alors terminé.

Pour le premier tour, il est obligatoire de placer son domino autour du château. Lors des tours suivants, les dominos peuvent être placés autour du château s'il reste de la place, mais ils peuvent aussi être placés à côté de dominos ayant le même type de terrain. Si un joueur ne peut pas placer le domino qu'il a pioché à côté d'un autre domino, qu'il ne veut tout simplement pas le jouer pour une histoire de stratégie, ou qu'il ne peut pas, parce que le placer lui ferait un plateau de plus de 5 cases par 5, il défausse son domino (qu'il ne pourra plus réutiliser).

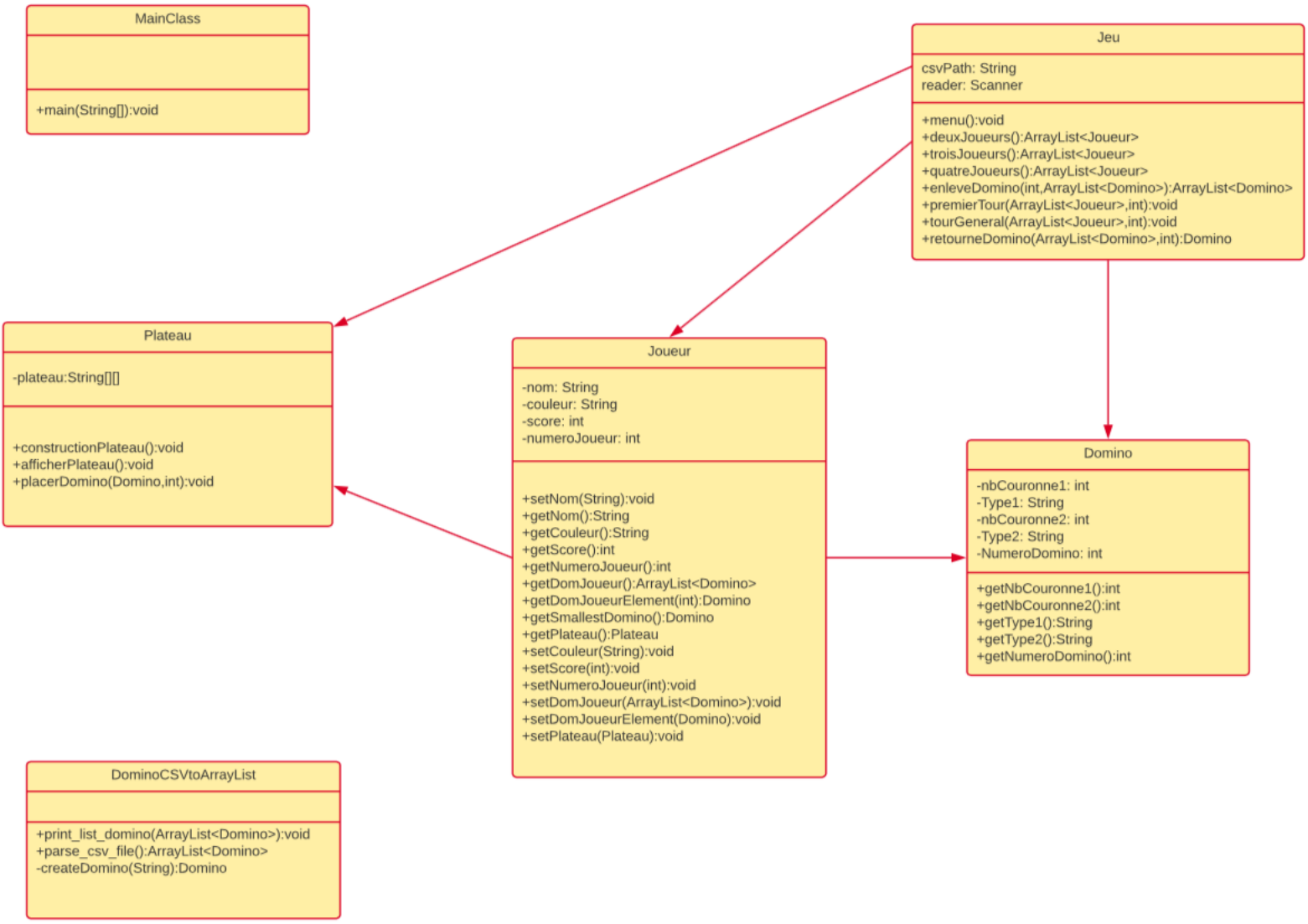
Le joueur qui a gagné est celui qui a le plus grand score. Pour compter les points, il suffit de compter le nombre de couronnes qui sont présentes à l'intérieur d'un même territoire (cases adjacentes) avec le même type de terrain. Une fois le nombre de couronne compté, il faut multiplier ce nombre avec le nombre de cases qui composent le territoire et le tour est joué !

3. Fonctionnalités attendues

- Récupération des dominos à partir d'un fichier CSV ;
- Implémentation des fonctionnalités de base : initialisation du jeu, tours de jeu, calcul des points et identification du vainqueur ;
- Développement de l'interface graphique rendant le jeu jouable pour 2 à 4 joueurs humains en tour par tour ;
- Implémentation d'une IA capable de jour selon les règles du jeu.

II. Modélisation du jeu

1. Diagramme UML



2. Classes du programme

Le programme Domi'Nations que nous avons codé repose sur une structure de code orientée objet par opposition aux méthodes procédurales. Ainsi, le code repose sur une répartition des méthodes par classes. Certaines de ces classes sont ensuite instanciées, dans notre cas ceci est le cas pour les classes Joueur et Domino qui sont utilisées tout au long du programme. Cette philosophie permet une économie non négligeable de code pour les structures qui se répètent.

a) MainClass

La classe Main sert de lancement pour notre programme, c'est elle qui lance la classe « Jeu » qui mêle toutes les autres classes. C'est pourquoi, nous avons peu de contenu dans cette classe. Nous n'avons malheureusement pas pu réaliser l'interface graphique de notre jeu, du coup nous avons décidé de montrer dès le début les règles du jeu aux joueurs, car nous partons du principe que les utilisateurs ne les connaissent pas forcément. Ensuite nous appelons seulement notre classe « Jeu » qui lance notre jeu.

b) DominoCSVtoArrayList

La classe DominoCSVtoArrayList est primordiale. En effet c'est la classe qui sert à lire le fichier CSV et à créer une liste de domino à partir de ce fichier CSV.
Elle comporte pour ce faire une méthode appelée parse_csv_file qui s'occupe de manière générale à parcourir le fichier CSV en le lisant lignes par lignes et remplir la liste de dominos appelée listDom avec ce qui est lu dans le fichier CSV. La méthode parse_csv_file, s'occupe tout d'abord de lire le fichier. C'est le BufferedReader qui permet de lire un fichier. Chaque ligne lue du fichier CSV par le BufferedReader est stockée dans la variable dominoLine. Une boucle while est ensuite utilisée pour dire que tant qu'il y a des lignes à lire dans dominoLine, java les lit, crée un domino (grâce à la méthode createDomino décrite juste après) et les ajoute à listDom (qui est la liste qui contiendra tous les dominos après l'opération). S'il n'y a plus de ligne à lire, il s'arrête. Si le fichier CSV n'est pas trouvé ou s'il est égal à null, le programme gère les exceptions en renvoyant un message d'erreur.

La méthode createDomino permet de créer des dominos à partir de ce qui est lu dans le fichier CSV. C'est pour cela qu'elle prend en arguments dominoLine. Grâce à splitData, cette méthode sépare les lignes contenues par dominoLine correctement. C'est à dire que java va aller voir ce qu'il y a dans dominoLine, et grâce à splitData, il va comprendre que ce qui est lu en premier (avant la virgule) est le nombre de couronne sur le terrain 1, ce qui est lu en deuxième est le type de terrain 1, en troisième le nombre de couronne sur le terrain 2, en quatrième le type de terrain 2 et enfin en cinquième le numéro du domino. d est un objet créé dans la classe Domino qui a donc comme attributs ces cinq éléments.

La méthode print_list_domino contenue dans cette classe nous servait à vérifier que le fichier CSV avait bien été lu et que la liste de domino s'affichait correctement dans la console.

c) Domino

Domino comporte un constructeur qui associe à Domino les attributs suivants :

- nbCouronne1
- nbCouronne2

- Type1
- Type2
- NumeroDomino

  Domino comporte également les *Getters* des attributs précédents.

d) Plateau

La classe plateau sert à créer le plateau mais aussi à demander au joueur où qu'il veut placer son domino sur son propre plateau et réaliser sa demande.

Pour ce faire, cette classe comprend tout d'abord une méthode constructionPlateau, qui comme son nom l'indique, permet de construire un plateau à deux dimensions de cinq cases par cinq. Les cases du plateau sont remplies tout d'abord par des zéros pour indiquer qu'aucun domino n'est encore posé au début du jeu. Une seule case est forcée par java à contenir un C (pour Château) au début du jeu. Il s'agit de la case en haut à gauche. Chaque joueur aura donc au début du jeu son plateau vide (remplir de zéros) avec son château positionné en haut à gauche.

La seconde méthode, afficherPlateau permet tout simplement au joueur de visualiser son plateau lorsque c'est à son tour, puis à chaque fois qu'il pose la première moitié de son domino, puis la seconde moitié. Elle est grandement utilisée dans la classe Jeu.

La troisième méthode placerDomino s'occupe de demander au joueur où il veut placer son domino, de faire les vérifications nécessaires puis de placer effectivement le domino sur le plateau si tout est bon. Cette fonction prend en argument un domino et typeTour (expliqué par la suite). Nous avons décidé de décomposer le domino en deux parties : la face1 (qui correspondra au premier type de terrain) et la face2 (qui correspondra au second type de terrain). Chaque face (ou case) du domino va être posée sur le plateau en entrant des coordonnées (x1, y1 pour la première case et x2, y2 pour la seconde case). Ensuite vient une boucle do while qui s'occupe de répéter les mêmes opérations tant que les conditions ne sont pas vérifiées. Dans un premier temps, java vérifie s'il s'agit du premier tour de jeu grâce à un if, parce que si c'est le cas, les conditions à vérifier seront différentes que celles pour les autres tours de jeu. En effet, si c'est le premier tour qui est en jeu, java vérifie que la première case du domino posée par le joueur est bien à côté du château, sinon, par un else, java indique au joueur qu'il doit saisir de nouveau la ligne de la première case de son domino. Par un second if, java vérifie que le premier couple de coordonnées données (x1 ou y1) par le joueur se situe bel et bien dans les limites du plateau 5 par 5 (donc les coordonnées entrées par les joueurs doivent être entre 0 et 4). Sinon, un message qui dit que les coordonnées entrées ne sont pas valides est affiché et indique au joueur qu'il doit saisir de nouveau la ligne de la première case de son domino. Ensuite, pour le premier tour, une quatrième condition permet de vérifier que le domino posé n'est pas sur le château (en x1 = 0 et en y1 = 0). Si c'est le cas, un message qui dit que les coordonnées entrées ne sont pas valides est affiché et indique au joueur qu'il doit saisir de nouveau la ligne de la première case de son domino. Enfin, une dernière condition sert à vérifier

que les deux cases données constituent bel et bien un domino (c'est-à-dire qu'elles sont adjacentes et non superposées). Sinon, java redemande la ligne de la deuxième case du domino. Si toutes ces conditions sont vérifiées, la première case du domino est placée sur le plateau et java peut enfin demander la ligne puis la colonne de la deuxième case du domino au joueur tout en vérifiant toujours si les coordonnées données sont bien dans les limites du plateau et si elles sont bien adjacentes au couple de coordonnées correspondant à la première case.

e) Joueur

La classe Joueur comporte un constructeur qui associe à un joueur des attributs : un nom, une couleur, un score, un plateau, un numéro de joueur et une liste de domino, ou alors un appel à un élément particulier de la liste de dominos. A ces attributs on y ajoute des getters et des setters afin de pouvoir renvoyer ou modifier les valeurs des attributs.

f) Jeu

La classe « Jeu » représente notre plus grosse classe, en effet, c'est elle qui comporte l'algorithme nécessaire au bon déroulement du jeu.

Cette classe s'exécute sous la forme d'un menu, nous avons donc divisé la première partie de l'algorithme en fonction du nombre de joueurs, ce fonctionnement nous semblait plus clair pour comprendre notre programme. Ainsi on demande aux utilisateurs : « combien de joueurs êtes-vous ? » et à partir de la réponse le programme exécute un cas différent du menu, par exemple s'il y a deux joueurs, on se retrouve automatique dans « case 2 : » qui fait alors appel à différentes fonctions. La première fonction appelée est deuxJoueurs, troisJoueurs ou quatreJoueurs (selon le nombre de joueurs), et à partir de ce moment-là nous définissons toutes les caractéristiques du joueur en fonction de ses goûts, comme le nom de son personnage ou la couleur de son roi (il a le choix entre quatre couleurs proposées), de plus, nous assignons à chaque joueur un score (initialisé à 0), un plateau, une liste qui se remplira en fonction des dominos que le joueur aura choisi et un numéro de joueur.
A partir de ces informations, nous créons une liste de joueurs (listeJoueur) qui va servir de base de données, et qui va permettre pour certaines fonctions d'appeler des données précises des joueurs.
Ensuite nous appelons la fonction « enleveDomino() » qui va permettre d'enlever aléatoirement des dominos en fonction du nombre de joueurs. Pour rappel, listDom est une liste qui regroupe tous les dominos, ainsi si nous avons 2 joueurs, nous enlevons au hasard des dominos, jusqu'à ce que la liste de dominos (listDom) soit égale à 24, ou à 36 dans le cas où il y aurait 3 joueurs.

Après nous appelons la première grosse boucle, qui est « premierTour ». Tout d'abord deux nouvelles listes sont créées : listeJoueurBis, qui contient le nombre de participants, et dominoCourant, qui représente les dominos piochés à chaque tour. Ces deux nouvelles listes permettent de pouvoir être modifiées sans toucher aux listes principales qui doivent garder leurs caractéristiques.
Ensuite nous distinguons deux cas pour tirer les dominos, si listDom contient 36 dominos (donc 3 joueurs), 3 dominos sont sortis au hasard et sont ajoutés à la liste dominoCourant et sont en même temps supprimés de la liste listDom pour éviter de tirer à nouveau ces dominos. Le deuxième cas est le même pour 2 ou 4 joueurs, car dans les deux cas il faut tirer 4 dominos, le fonctionnement est le même.
Par la suite, un des joueurs est choisi au hasard pour commencer à joueur, puis les dominos sont affichés, avec leurs caractéristiques (terrains et nombre de couronnes), grâce à la fonction retourneDomino() qui les renvoie en fonction du numéro du domino.
Le joueur sélectionne alors un des dominos, et le programme vérifie que la réponse du joueur correspond bien à un des dominos piochés, et ensuite le plateau propre au joueur sort en console et il ainsi peut placer son domino à côté du chateau. On sort alors de la boucle lorsque tous les joueurs ont joué, autrement dit lorsque listeJoueurBis.isEmpty().
On passe alors à la boucle tourGeneral(), qui a pour seules différences : la contrainte en moins de devoir placer son domino à côté du château, et que le tour général se fini lorsqu'il n'y a plus de dominos, donc lorsque listDom.isEmpty().