Patchwork

Nous avons divisé notre code en plusieurs fichiers, un fichier par classe et un main dans le fichier "SimpleGameController.java" qui appelle toutes les fonctions nécessaires au fonctionnement du jeu.

I Player

L'objet Player est utilisé pour représenter les deux joueurs de la partie de Patchwork. Chaque joueur a :
-un nombre de boutons
-un QuiltBoard qui représente la grille où il peut poser ses patchs.
-une liste contenant tous les patchs qu'il possède sur son QuiltBoard.
-une position sur le timeBoard qui permet de savoir où il se situe et de l'afficher

Parmi les méthodes nous avons:
-un constructeur qui initialise la postion à 0 et le nombre de boutons à 5 et qui initialise un nouveau QuiltBoard. -Des accesseur permettant d'accéder aux attributs et de leur assigner des valeurs.
-une méthode addPatchOwned qui permet d'ajouter un patch à la liste des patchs possédés par le joueur
-une réécriture de la méthode toString qui affiche la position, le nombre de boutons et le QuiltBoard dans la console.

Pour l'interface graphique:
-La méthode drawPlayer dessine le joueur sous forme d'un carré sur le timeBoard en fonction de sa position qu'on récupère de simpleGameData
-La méthode announcePlayerTurn annonce quel joueur doit jouer, on utilise simpleGameData pour savoir quel joueur doit jouer.

II QuiltBoard

La classe QuiltBoard est utilisé pour représenter la grille où le joueur peut poser ses patchs.

Chaque QuiltBoard a comme attributs:
-une matrice de int représentant par des 1 les emplacements occupés sur la grille des patchs.
-le nombre de boutons total des patchs positionnés sur le QuiltBoard.

Parmi les méthodes nous avons:
-une réécriture de la méthode toString qui affiche toute la grille avec des "X" sur les emplacements occupés et des numéros de lignes et de colonne ainsi que le nombre de boutons sur la grille.
-une méthode putPatch qui insère un patch dans le QuiltBoard à des coordonnées spécifiées en paramètre.
-une méthode ableToPut qui teste si un patch peut être positionné aux coordonnées spécifiées en paramètre et qui affiche "Illegal position" et retourne false sinon
-une méthode procedureToPutchPatch qui prend en paramètre un patch et qui demmande au joueur où et dans le sens il veut le poser, puis appelle les fonction chooseRotation, ableToPut,putPatch et addButtons
-une méthode chooseRotation qui demmande au joueur quelle rotation il veut appliquer au patch, puis qui en fonction de la réponse du joueur appelle les fonctions qui applique la rotation à la pièce.
-des méthodes rotateRight,rotateDown et rotateLeft qui applique des rotations au patch.
-une méthode addButtons qui ajoute les boutons d'un patch au QuiltBoard
-une méthode countScoreBlank qui compte le nombre de cases vides sur le QuiltBoard
-une méthode verifyBonus qui vérifie si le quiltBoard contient un carré 7 * 7.

Interface graphique:
-une méthode drawQuiltBoard qui dessine la grille sur l'écran, on récupère le quiltBoard du joueur dans simpleGameData pour savoir quelles pièces on doit afficher sur l'écran, on dessine un carré bleu sur les cases du quiltBoard qui valent 1 et des cases vides sinon

III Patch

Nous utilisons le record Patch pour représenter les patchs, il a comme attributs:
-buttons qui représente le nombre de boutons du patch
-price qui est le prix d'achat du patch
-timteStep le nombre de case dont un joueur avance à l'acquisition du patch
-height sa hauteur
-width sa largeur
-shape une matrice de taille height * width composé de 0 et 1 dont les 1 représentent la forme du patch
Les méthodes de Patch sont:
-la méthode printShape qui renvoie la forme du patch sous forme de chaine de caractère
-la méthode getShape qui renvoie la matrice shape
-une réécriture de la méthode toString qui affiche le nombre de boutons, prix, le timeStep et la forme du patch

Interface Graphique:
-La méthode getPlayerChoice qui attend un clic du joueur, récupère les coordonnées du clic, teste si le clic est sur patch sinon on attend un autre clic, dans le cas où le clic est sur le patch on retourne l'index du patch cliqué dans la liste des patchs, "-1" si la case est invalide et "-2" si on clique sur "skip turn"
-La méthode drawPatch dessine patch sur l'écran avec en paramètre le patch et les coordonnées x et y où on veut l'affiche, on parcourt le tableau shape et on dessine un carré pour chaque 1 rencontré
-la méthode procedureToPutPatch qui prend en paramètre le patch qui affiche les choix de la rotation et qui insère le patch final dans liste des patchs.
-La méthode chooseRotation affiche les choix des rotations et attend un clic.
-La méthode checkRotationRange vérifie si le clic est bon.
-La méthode drawRotation affiche le patch en fonction de la rotation qui lui est associée.

IV Le jeu

La classe jeu représente une partie de patchWork, ses attributs sont:
-timeBoard le plateau sur lequel les joueurs se déplace
-patches une liste de patch représentant les patchs pouvant être achetés lors de la partie
-la position du pion determinant les 3 patchs que l'on peut acheter
-mode un entier qui définit le mode de jeu normal ou simplifié
-bonusSquare7Owner qui représente le joueur ayant remporté bonus du carré 7 * 7
-player1 et player2, les deux joueurs de la partie -firstFinisher, le joueur qui a finit de parcourir le timeBoard en premier
-turn indique quel joueur doit jouer, 1 si c'est le tour du joueur 1 et 2 si c'est le tour du joueur 2

Comme méthodes de classe nous avons; -un constructeur qui initialise les deux joueurs, fait le choix mode, initalise la NTPos (le pion définissant l'achat des patchs) à 0, initialise la liste des patchs en fonction du mode, mélange cette liste puis initialise le time board avec les positions des boutons et des carrés 1 * 1.
-une méthode initSimplePatches qui initialise la liste de patch avec des patch simpled

-une méthode initPatch qui initialise la liste patch avec la liste complète des patchs du *Patchwork* -une méthode switchTurn qui passe au tour de l'autre joueur
-une méthode getTurnPlayer qui renvoie le joueur à qui c'est le tour
-une méthode printTimeBoard qui affiche le timeBoard
-une méthode getPlayerChoice qui demmande au joueur l'action qu'il veut faire durant ce tour
-une méthode choosePiece qui retourne le patch choisi par le joueur et qui le supprime de la liste des patchs à acheter
-une méthode buyPatch qui teste si le joueur peut acheter le patch et si le joueur a assez de boutons l'ajoute à sa liste de patch posséder et lui décompte le prix avant de retourner le patch acheter
-une méthode moveAfterSkipping qui déplace le joueur quand il veut passer son tour sans rien acheter
-une méthode moveAfterBuying qui déplace lejoueur qui déplace le joueur après l'achat d'un patch en fonction du timeStep
-une méthode updateButtonScore qui incrémente le nombre de boutons du joueur quand il passe sur un bouton
-une méthode play qui joue une tour en demmandant un choix d'action au joueur, appelle ensuite les fonction d'achat ou de déplacement sans achat, teste si le bonus 7 * 7 est atteint et actualise le joueur à qui c'est le tour.
-une méthode endGame qui renvoie true si les 2 joueurs ont parcouru l'intégralité du timeBoard
-une méthode tiedWinner qui renvoie le joueur qui a fini de parcourir le plateau en premier
-une méthode winner qui calcule le score des deux joueurs et qui affiche le gagnant
-une méthode announcePlayerTurn qui annonce à qui c'est le tour
-une méthode chooseMode qui demmande au joueur s'il veut jouer avec la version normale ou la version simplifiée.
-une réécriture de la méthode toString qui affiche le timeBoard et les deux joueurs

Interface Graphique:
-la méthode playGame qui lance le déroulement du jeu en rajoutant des affichages sur l'interface graphiques
-la méthode drawTimeBoard dessine le timeBoard vide avec ses 54 cases affichées en 7 rangées de 7 cases plus une ligne de 5 cases.
-La méthode chooseMode demmande à l'utilisateur de choisir un mode s'il choit le mode graphique la variable mode vaut 2.
-la methode memoryGame qui demmande le choix de la version si le choix une initialise une gameView de type ConsoleGameView si la variable version vaut 1 et sinon on initialise l'écran d'affichage ainsi qu'une gameView de type SimpleGameView.

V Le déroulement du jeu

Pour faire une partie de *Patchwork*:
-on affiche le message "Welcome to PatchWork!!!"
-on initialise les deux joueurs
-on initialise la partie avec le choix du mode, la liste des patchs à acheter, la position du pion et le timeBoard
-le déroulement du jeu se fait dans une boucle while qui teste si les deux joueurs ont parcouru l'intégralité du plateau
-dans cette boucle:
-on affiche le jeu

-on demmande au joueur de faire un choix et on appelle les fonctions adaptées
-on teste si un des deux joueurs a fini de parcourir le plateau et le cas échéant on le stocke dans la variable firstFinisher.
-à la fin du while on fait le calcul du score et on affiche le gagnant