Thibaud Dirou

Projet : Pub Management

Description générale :

La totalité du code est rangée dans 4 packages, « bar », « cardGames », « lauchers » et « people ».

Le programme se lance avec le fichier « Laucher.java ».

Dans le package « people » sont rangés les classes et les méthodes décrites dans partie 1 de l'énoncé

Le package « cardGames » contient les classes nécessaires à la partie de belotte et au tournoi.

Le package « bar » contient de nombreuses variables correspondant aux éléments contenus dans le bar (boissons, tables, liste de personnes…)

Bien que les classes et méthodes correspondant aux personnages, aux interactions, à la partie de Belotte et au tournoi ont été codés. Elles ne fonctionnent que manuellement lorsqu'on écrit soit même les interactions voulues dans « laucher.java » puis qu'on lance le programme.

Je n'ai pas réussi à rendre le code autonome et à mettre en place une interface permettant à l'utilisateur d'interagir directement avec le programme par manque de temps (le projet était bien plus long que je l'avais anticipé)

Je vais maintenant décrire le fonctionnement du programme sur les Parties 1 à 3 de l'énoncé correspondant à ce que j'ai eu le temps de réaliser.

Partie 1 : Gestion du bar

La plupart des éléments nécessaires à cette partie se trouvent dans le package « people » :

- Les classes de personnages :

1. Bartender
2. Boss
3. Client
4. Human
5. Provider
6. Server

- Une classe de couleur « ConsoleColors » permettant de colorer les paroles de chaque type de personnage différemment

- Une interface « Worker » contenant une méthode commune aux employés du bar.

Le fichier BarElements.java dans le package bar est aussi requis au fonctionnement de cette Partie, il contient de nombreux éléments comme : la liste des boissons ainsi que le degré d'alcoolémie correspondant, le prix et le nombre de ces éléments dans le stock du barman.

Il contient aussi la caisse du bar, plusieurs ArrayList de personnage et d'autres variables.

La plupart des méthodes implantées dans les classes personnages correspondent à celles de l'énoncé.

Partie 2 : Jeu de belotte

Une partie de belotte (classe BelotteGame) est initialisée avec une instance de la classe Table, deux instances de Team et utilise 32 instances de la classe Carte.

- Classe table : il y a 3 tables dans BarElements, la table 2 et 3 sont libres d'accès aux clients et la table 1 est réservée aux parties de belotte. Cette classe possède une variable ArrayList peopleSitting contenant les clients assit à la table.
- Classe Team : Cette classe possède deux variables membre (membre1 et membre2) ainsi qu'une variable point permettant de comptabiliser les points gagnés par cette équipe lors qu'un tournoi.
- Classe Carte : possède une hauteur (allant de 8 à l'as) et un symbole (cœur, pique…). Les valeurs en points des cartes dans le jeu de belotte sont rangées dans deux Hashmap (un pour les valeurs de l'atout, l'autre pour les valeurs hors atout)

Dans une partie de belotte, les deux joueurs de la même équipe jouent face à face, c'est pourquoi les équipes sont crées en fonction de la place des joueurs à la table.

Le paquet de carte est une instance de java.util.stack() contenant les 32 instances de Carte.

Le déroulement de la partie est contenue dans la playGame(). CAD :

- Le réinitialisation des points et du paquet de carte après chaque round : resetRound()
- La distribution et le choix de l'atout : Distribution ()
- La recherche d'annonces dans le jeu de chaque joueur bonusChecking()
- Le choix de la carte à jouer à chaque round pour chaque joueur playMove()
- Le compte des points du round : roundScoring()
- Le calcul des points totaux et désignation du vainqueur : endGame()

Cette méthode est donc une grosse boucle while vérifiant qu'aucun des scores totaux n'ai dépassé 1010, lors que cet événement à lieu, la méthode endGame() est lancée.

Les choix (choix de l'atout, choix des cartes à jouer) sont réalisés dans une classe « Player ».

Cette classe contient 3 méthodes principales :

- acceptTrump() qui renvoi un boolean correspondant à l'acceptation ou non de l'atout au sommet du paquet lors de la première partie de la distribution.
- chooseTrump() qui renvoi un String correspondant au symbole choisi si aucun des joueurs n'a accepté l'atout du sommet du paquet
- play() qui renvoi une Carte correspondant à la carte choisi en fonction des cartes déjà posées sur le plateau. Cette fonction est codée de façon que les joueurs respectent les règles du jeu lorsqu'ils choisissent leurs cartes. (Choisir en priorité une carte du même symbole que celui demandé, sinon de l'atout, sinon une carte au hasard)

Ces 3 méthodes possèdent 3 sous méthodes appelés ou non en fonction de la variable intelligenceLevel du Player.

- Si intelligenceLevel = 0, les fonction bad play, badAcceptTrump,badChooseTrump sont utilisée
- Si intelligenceLevel = 1, averagePlay…, averageAccept…
- Si intelligenceLevel =2, goodPlay…

Ce qui permet d'avoir des NPC de différents niveaux, malheureusement je n'ai pas eu le temps de créer 3 IA différentes pour chaque niveau. Seul la fonction AcceptTrump possède 3 niveaux d'intelligence différents.

L'équipe vainqueur de chaque round est annoncée le score du round et, à la fin, le nom des deux vainqueurs de l'équipe vainqueur sont annoncés avec le score final.

Partie 3 : Tournoi

L'instance de la classe Tournoi est initialisé avec une ArrayList de Team en argument.

Le tournoi se déroule de la façon suivante :

Certains clients s'inscrivent sur une liste de participants par équipe de deux puis chaque équipe joue contre chaque équipe du tournoi.

A chaque nouvelle partie du tournoi, les 4 joueurs vont s'assoir à la Table 1 dans à la place adéquate pour être ne face de son partenaire.

Ensuite, une instance de BeloteGame est créée, le barman annonce tout haut les participant à la partie et le jeu de belotte vu précédemment est lancé.

A chaque fois qu'une équipe gagne la partie, elle gagne 1 point qui vient incrémenter la variable « points » de l'instance de la classe Team. (Je n'ai pas compris le comptage des points de l'énoncé)

Lorsque toutes les parties du tournoi ont été jouées, l'équipe ayant le plus de point est annoncée vainqueur et ses membres reçoivent des récompenses (argent et popularité), et expriment leur joie en poussant leur cri.

Conclusion :

Si j'avais eu le temps, j'aurais rendu le programme automatique en créant des listes de toutes les actions possibles pour chaque personnage. Et j'aurais appelé ces actions au hasard avec random tout le long du programme qui tournerais en boucle dans une fonction while true.

J'aurais permis à l'utilisateur de créer un personnage avec lequel il aurait pu réaliser des actions via l'entrée Scanner Kheyboard.

J'aurais aussi probablement essayé d'utiliser le multithreading permettant à l'utilisateur d'interagir avec le programme pendant le déroulement du programme en fond.