Anguenot Victor
Pouget Guillaume
Zniber Ismail

Université Paul Sabatier

Projet C++ M2 MAPI3

Compte rendu de la programmation du jeu de cartes : Le Rami

22 Novembre 2019

1 Les règles :

Paquet :

Jeu de 52 cartes.

But :

Etre le premier à poser toutes ses cartes, en une ou plusieurs fois.

Configuration :

Chaque joueur a 14 cartes. Les cartes qui ne sont pas distribuées sont placées face contre table au milieu de la table pour former la pioche.

Séquence de jeu :

1. Piocher une carte, soit dans la pioche, soit dans la défausse.
2. Le joueur peut placer une combinaison sur la table (mais n'est pas obligé), ou ajouter des cartes à une combinaison déjà posée sur la table.
3. Le joueur doit se défausser d'une carte, en l'ajoutant face visible au dessus de la défausse. Exception : si le joueur a posé toutes ses cartes, il n'a pas à se défausser.

   ATTENTION : si un joueur a pioché dans la défausse, il doit la poser directement sur le plateau

Combinaisons valables :

- Brelan : 3 cartes identiques
- Carré : 4 cartes identiques
- Séquence : au moins 3 cartes de même couleur qui se suivent.

Score :

Quand un joueur a terminé, on compte les points. On déduit du score les cartes éventuellement restantes dans la main des joueurs.

- AS : 1 point
- Cartes de 2 à 10 : nombre de points = numéro de la carte
- Valet, Dame, Roi : 10 points

2 Les classes :

2.1 La classe Carte :

2.1.1 protected :

- **int** valeur
- **std : :string** couleur

2.1.2 public :

- Constructeur par défaut :
  - **Carte** ()
- Constructeur :
  - **Carte**(**int** valeur, **std : :string** couleur)
- **getValeur** () {
  Obtenir le numéro de la carte
  }
- **setValeur** (**int** v) {
  Donner une valeur à une carte
  }
- **getCouleur** (){
  Obtenir la couleur de la carte
  }

- **setCouleur** (**std : :string** c) {
  Donner une couleur à une carte
  }
- **afficherCarte** () {
  Afficher une carte
  }

2.1.3 Les opérateurs :

La surcharge des opérateurs permet d'adapter les opérateurs mathématiques, afin de pouvoir comparer des objets de type Carte.

- ==
- >
- <
- ≥
- ≤

2.2 La classe Paquet :

Cette classe hérite de la classe Carte.

2.2.1 private :

- **std : :vector<Carte>** cartesPaquet

2.2.2 public :

- Constructeur par défaut :
  - **Paquet** ()
- Constructeur :
  - **Paquet** (**std : :vector<Carte>** c)

- **suppDerniereCarte** () {
  Supprime la dernière carte
  }
- **initPaquet** () {
  Initialise le paquet
  }
- **viderPile** (){
  Vide une pile de carte
  }
- **ajoutDevant** (**Carte** c) {
  Ajoute une carte à l'avant d'une pile
  }
- **melangerPaquet** () {
  Mélange le paquet
  }
- **nouveauPaquet** () {
  Création d'un nouveau paquet de carte
  }
- **piocheCarte** () {
  Pioche une carte
  Retourne objet de type carte

  }
- **ajouteCarte** (**Carte** carteAjoutee){
  Ajoute une carte
  }
- **supprimerCarte** (**Carte** c) {
  Supprime une carte
  }
- **viewerCartes** () {
  Permet l'affichage des cartes dans le terminal
  }

- **taillePaquet** () {
  Permet d'obtenir la taille d'un paquet
  Retourne objet de type entier
  }
- **getCartes** (**Carte** C) {
  Permet d'obtenir les valeurs et couleurs des cartes
  }
- **nbMelangesAleatoire** () {
  Nombre de mélanges des cartes
  Retourne objet de type entier
  }

2.3 La classe Joueur :

2.3.1 private :

- **std : :string** nom
- **int** score ;
- **int** choice ;

2.3.2 public :

- **Paquet** *hand {
  Correspond à la main du joueur.
  }
- Constructeur par défaut :
  - **Joueur** ()
- Constructeur :
  - **Joueur** (**std : :string** n)
  - **Joueur** (**Paquet** *h, **int** sco, **int** choi)
- Destructeur :
  - ~ **Joueur()**

- **setNom** (**std : :string** n){
  Permet d'ajouter un nom au joueur
  }
- **getNom** () {
  Permet d'obtenir le nom du joueur
  }
- **setScore** (**int**) {
  Permet de modifier le score du joueur
  }
- **getScore** () {
  Permet d'obtenir le score du joueur
  Retourne un objet de type entier
  }
- **setChoice** (**int**) {
  Permet de modifier le choix du joueur
  }
- **getChoice** () {
  Permet d'obtenir le choix du joueur
  Retourne un objet de type entier. 1 si le joueur décide de piocher dans la pioche, 2 si dans la défausse
  }
- **premiereMain** (**Paquet** *paquet) {
  Permet de définir combien de cartes chaque joueur possèdera au début de la partie lors de la distribution
  }
- **afficheMain** () {
  Affiche la main du joueur pendant son tour
  }
- **verifieCartes** () {
  Permet de savoir si un joueur possède des cartes dans sa main.
  Retourne un objet de type booléen. True si le joueur n'a plus de carte
  }

- **soustraireScore** () {
  A la fin de la partie, le joueur qui n'aura pas gagné verra son score retranché du nombre de points qu'il possède encore dans sa main
  }

2.4 La classe Hand :

2.4.1 private :

- **std : :vector<Paquet *>** pileDeCartes ;

2.4.2 public :

- Constructeur par défaut (pile vide) :
  - **Hand**() ;
- Constructeur :
  - Hand(**std : :vector<Paquet *> pile**)
- **suppDernierElement**() {
  Supprime le denier élèment de la pile }
- **ajouterPile**(**std : :vector<Carte>** c, **Joueur** *j) {
  Ajoute à la pile et augmente le score du joueur }
- **getPile**() {
  Retourne la pile
  }
- **ajouterPileExistante**(**std : :vector<Carte>** c, **Joueur** *j) {
  Ajoute une carte à une existante pile et augmente le score du joueur
  }
- **afficherPiles**() {
  Affiche toutes les piles déjà posées sur le tapis
  }

2.5 La classe Senario

2.5.1 private :

- **Hand** *cartesEnJeu
- **Paquet** *pioche
- **Paquet** *defausse

2.5.2 public :

- **Joueur** *joueur1
- **Joueur** *joueur2
- Constructeur par défaut :
  - **Scenario** (**Joueur** *j1, **Joueur** *j2, **Paquet** *pilePioche, **Paquet** *defausse)
- Destructeur
  - ~ **Scenario**()
- **sequenceTour**(**Joueur** *joueurActuel) {
  Sequence d'un tour de jeu :
  - **afficherMain**()
  - **tirerCarte**()
  - Si joueurActuel->choice == 1
    - **afficherTable**(joueurActuel)
    - **choixduJoueur**(joueurActuel)
  - **defausseFinTour**(joueurActuel)

  }
- **checkSequence**(**std : :vector**<**Carte**> carteSelection) {
  Vérifie si les cartes forment une séquence
  Retourne un objet de type entier
  }
- **tirerCarte**(**Joueur** *joueurActuel) {
  Tire la dernière carte de la pile ou de la defausse
  }

- **placerSequence**(**Joueur** *joueurActuel) {
  Place une séquence
  }
- **ajouterCartesPileExistante**(**Joueur** *joueurActuel) {
  Ajoute une carte à une pile déjà existante
  }
- **enleverCartes**(**std : :vector**<**Carte**> carte, **Joueur** *joueurActuel) {
  Enleve des cartes dans la main du joueur
  }
- **selectionCarte**(**Joueur** *joueurActuel) {
  Sélectionne une carte de la main du joueur
  Retourne un objet de type Carte
  }
- **jouer**() {
  }
- **afficherTable**(**Joueur** *joueurActuel) {
  Affiche la table
  }
- **defausseFinTour**(**Joueur** *joueurActuel) {
  Permet de se défausser d'une carte de son jeu
  }
- **modifierPile**(**Joueur** *joueurActuel) {
  Ajoute une carte à une pile déjà existante
  }
- **pileDefausse**(**Joueur** *joueurActuel) {
  Permet de piocher dans la defausse
  }
- **choixJoueur**(**Joueur** *joueurActuel) {
  Demande si le joueur veut poser quelque chose sur la table
  }

3 Le déroulement du jeu :

Le début du jeu :

Comme décrit dans le paragraphe précédent, le but d'une manche est d'être le premier à poser toutes ses cartes en respectant les combinaisons (suite, brelan ou carré).
Au début de chaque manche, chacun des joueurs dispose de 14 cartes qui resteront cachée des autres joueurs tout au long de la manche. Les autres cartes constituent la pioche et la défausse.

Au cours du jeu :

Durant le jeu, chaque joueur, peut piocher une carte de la pioche ou de la défausse. Si le joueur pioche dans la pioche, il a la possibilité de choisir entre :

- Poser tout ou une partie de ses cartes, sous forme de combinaisons, sur la table. Les figures possèdent un score de 10 points, alors que les cartes numérables un score correspondant à leur numéro et l'As un point.
- Poser une ou plusieurs de ses cartes sur son jeu déjà posé ou celui d'un autre joueur

Si le joueur pioche dans la défausse il doit poser cette carte durant le tour, sinon elle sera renvoyée dans celle-ci et il devra piocher dans la pioche.
Après chaque tour, le joueur devra se séparer d'une de ses cartes. Elle sera envoyée dans la défausse.

La fin du jeu :

Pour gagner une partie de Rami, le joueur doit être le premier à se débarrasser de toutes ses cartes. A la fin du jeu, les cartes restantes en main sont décomptées du score du perdant.