UNIVERSITÉ DE MONTPELLIER 2
PLACE EUGÈNE BATAILLON 34095 MONTPELLIER CEDEX 5
Licence informatique, Semestre 4 – 2012/2013

GLIN403 – Contrôle Continu n° 2

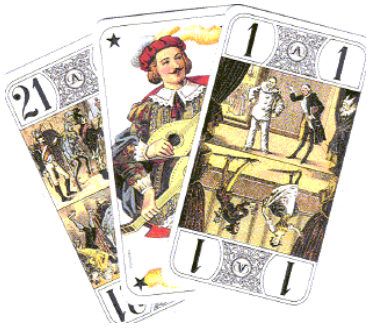
Tard au Tarot – Durée 3h – Documents autorisés
Alban MANCHERON

Avant toute chose, il convient de rappeler certaines règles et bons usages relatifs aux épreuves. Tout d'abord, le barème n'est fourni qu'à titre indicatif. Ensuite, il est bien entendu que les copies, écrans, messages des voisins et autres collègues ne font pas partie de l'ensemble des documents autorisés. De surcroît, seul l'enseignant est habilité à répondre à vos questions pendant l'épreuve. Il vous est par ailleurs conseillé de lire attentivement le sujet dans son intégralité. Seules les questions posées dans le premier quart d'heure de l'épreuve seront susceptibles d'être écoutées. Enfin, vous remettrez votre travail à la fin de la séance en déposant vos fichiers (dans une archive compressée) dans l'espace pédagogique. Pour cela il suffit de vous identifier sur votre espace numérique de travail, puis d'aller sur l'onglet *travaux* du module GLIN403 [1], de cliquer sur le travail associé à votre groupe, et enfin de cliquer sur *nouvelle soumission*. Il vous reste maintenant à parcourir vos fichiers et à ajouter votre archive compressée. Pensez bien à renseigner votre nom dans le texte de description du fichier.

Les cartes du tarot

Le jeu de tarot est apparu en France au tout début du XVI^{e} siècle. Il s'agit d'un jeu de 78 cartes réparties ainsi :
- 56 cartes réparties en 4 enseignes traditionnelles : trèfle ♣, carreau ♢, cœur ♡ et pique ♠. Chaque enseigne comporte 10 cartes numérotées de 1 à 10, un valet, un cavalier, une dame et un roi ;
- 21 cartes numérotées de 1 à 21, appelés « atouts » qui ont priorité sur les 56 cartes de couleurs. Le numéro indique la force de chaque atout, du plus fort (le 21), au plus faible (le 1, appelé « petit ») ;
- L'« excuse », est une carte marquée d'une étoile et représente un joueur de mandoline. Il s'agit d'une sorte de joker.

Le petit, le 21 d'atout, et l'excuse sont appelés des « Oudlers » ou « bouts ».



À chaque carte est associée un nombre de point :

- un bout vaut 4,5 points ;
- un roi vaut 4,5 points ;
- une dame vaut 3,5 points ;
- un cavalier vaut 2,5 points ;
- un valet vaut 1,5 points ;
- une autre carte vaut 0,5 points.

Le travail demandé consiste à modéliser en C++ certains aspects du tarot, sachant qu'aucune prédisposition à ce jeu n'est requise.

Représentation d'une carte [8 points]

Un carte du tarot est donc représentable par une classe **Carte** permettant de l'identifier. Une façon de faire est de considérer que chaque carte correspond à une valeur numérique *valeur* et une catégorie *categorie* .

1. Vous devez être inscrit à ce module ; si cela n'est pas le cas, demandez à votre enseignant de vous y inscrire.

Pour simplifier, on supposera que :
- l'excuse est l'atout 0 ;
- le valet est assimilé à la valeur 11 ;
- le cavalier est assimilé à la valeur 12 ;
- la dame est assimilée à la valeur 13 ;
- le roi est assimilé à la valeur 14.

Concernant les catégories, vous utiliserez l'énumération **Categorie** correspondant aux 4 couleurs (♣, ♢, ♡, ♠) plus l'atout définie au sein de la classe **Carte**.

Pour des raisons algorithmiques, vous trouverez la catégorie fictive supplémentaire *NB_CATEGORIES* à la fin de l'énumération.

Vous devez donc permettre de créer un objet **Carte** avec un constructeur à deux paramètres : sa valeur (-1 par défaut) et sa catégorie (*NB_CATEGORIES* par défaut).

Pour tout objet **Carte** créé ou modifié (hormis par le constructeur par défaut), il conviendra d'afficher un avertissement sur le flot de sortie *std* :: *cerr* si le couple *valeur*/ *categorie* n'est pas cohérent. Vous expliquerez dans un commentaire du fichier `.h` ce que cela implique.

Il vous est également demandé de définir les accesseurs en lecture et modification des deux attributs composant une carte. Ces accesseurs devront également afficher un avertissement sur le flot de sortie *std* :: *cerr* si le couple *valeur*/ *categorie* n'est pas cohérent.

Vous définirez une méthode *GetScore*() qui calcule et renvoie le nombre de points de la carte. Une carte incohérente compte pour 0 point (et vous afficherez encore une fois un avertissement sur le flot de sorte *std* :: *cerr*).

Enfin, vous devrez fournir des méthodes de comparaison avec une autre carte qui implémentent la relation d'ordre suivante :
- deux cartes cohérentes de deux catégories différentes sont ordonnées selon l'ordre des catégories (de la plus petite à la plus grande : ♣, ♢, ♡, ♠ puis atout) ;
- deux cartes incohérentes de deux catégories différentes sont ordonnées selon l'ordre des catégories (de la plus petite à la plus grande : ♣, ♢, ♡, ♠ puis atout) ;
- pour deux cartes cohérentes au sein d'une même catégorie, la carte la plus forte est celle dont la valeur est la plus élevée ;
- pour deux cartes incohérentes au sein d'une même catégorie, la carte la plus forte est celle dont la valeur est la plus élevée ;
- une carte cohérente est toujours plus petite qu'une carte incohérente.

Un jeu complet [8 points]

À partir de la classe **Carte**, il est possible de définir un jeu complet de tarot, mais également :
- un jeu traditionnel de 52 cartes (ensemble des cartes du tarot privé des cavaliers et des atouts) ;
- un jeu traditionnel de 54 cartes (jeu de 52 cartes + 2 cartes d'excuse) ;
- un jeu traditionnel de 32 cartes (jeu de 52 cartes privé des cartes de valeurs comprises entre 2 et 6) ;
- un jeu traditionnel de 34 cartes (jeu de 32 cartes + 2 cartes d'excuse) ;

⋮

Un jeu de carte est donc un objet de la classe **JeuDeCartes** représentant un tableau élastique (au sens des cours/TD/TP) de **Carte**.

Il vous est demandé de définir un constructeur permettant de créer un jeu vide, un jeu de n cartes incohérentes, un jeu traditionnel trié de 32, 34, 52, 54 et 78 cartes.

Outre les traditionnels accesseurs, vous devrez définir :
- une méthode permettant de vérifier si le jeu est trié ;
- une méthode permettant de vérifier si le jeu contient des cartes incohérentes ;
- une méthode permettant de mélanger le jeu de cartes ;
- une méthode permettant de couper le jeu de cartes.

Pour mélanger un jeu, un algorithme simple consiste à échanger chaque position du jeu avec une position prise au hasard dans le jeu. Pour cela, on pourra utiliser la méthode **int** *rand*(); définie dans la librairie *cstdlib* et qui renvoie un nombre aléatoire compris entre 0 et *RAND_MAX* ($\geq$ 32 767). Par exemple, l'instruction *rand*()%10 renvoie un entier compris entre 0 et 9.

Pour couper un jeu de carte, il faut choisir au hasard la position de coupe i, et effectuer i décalages circulaires de l'ensemble des cartes (un décalage circulaire signifie que la première carte devient la dernière).

Partie de cartes [4 points]

Les parties de cartes (tarot ou autre) sont généralement organisées en tours, et à chaque tour, chaque joueur défausse sa « main » (ensemble des cartes qui compose son jeu) d'une ou plusieurs cartes. Modéliser un joueur revient essentiellement à modéliser sa main et ses gains durant la partie. Il va de soi que la structure de donnée qui s'impose pour la modélisation de la main et ses gains est une liste de carte représentée par la classe **ListeDeCartes** et non un tableau de carte.

Un joueur est alors un objet qui a un pseudo (**string**), une main (**ListeDeCartes**), un gain (**ListeDeCartes**), et un score (**int**).

Il reste à définir un tableau élastique de joueurs **JoueursElast** et à simuler une partie de carte. Pour simplifier, vous dérouleriez une partie de « bataille » dans sa version la plus simple : à chaque tour, les joueurs posent la première carte de leur main et le joueur qui a la carte la plus forte gagne les cartes qui ont été posées. S'il y a plusieurs cartes gagnantes sur le tour, c'est celui qui a posé sa carte en dernier qui remporte le tour.

Celle-ci peut alors se décomposer comme suit :

1. Mélanger les cartes (cela ne se fait pas au tarot, sauf si le jeu est trié).
2. Couper les cartes ;
3. Distribuer équitablement (en nombre) les cartes aux joueurs, le reste de carte non distribué constituera le talon. Pour l'exercice, la distribution consistera simplement à partager les cartes entre les joueurs ;
4. Tant que les joueurs ont encore des cartes ;
   (a) Chaque joueur joue une carte en commençant par le gagnant du précédent tour ;
   (b) Le gagnant du tour place les cartes jouées dans son gain ;
5. Compter les points de chaque joueur et arrondir au multiple de 5 le plus proche ;
6. Rassembler les cartes des gains de chaque joueur et du talon.

Bonus [5 points]

Pour chaque classe de votre programme, vous êtes libres d'implémenter des programmes de tests afin de valider le comportement de toutes les méthodes. Vous prendrez le cas échéant le temps d'écrire en commentaire ce que vous testez.

Pour vous aider…

Les codes `C++` des fichiers d'entête, ainsi que les codes incomplets des fichiers d'implémentation et un programme principal vous sont fournis sur l'espace pédagogique. Il ne vous reste donc qu'à remplir les trous.

Bon courage…