Cégep de Sainte-Foy – Hiver 2024
Introduction à la programmation – 420-W10-SF

Énoncé du Travail Pratique 2

7 novembre 2024

Préparé par
Pierre Poulin
Adapté par
Raphaël Paradis

1 Résumé

Créer une application « graphique » permettant de jouer à un jeu s'apparentant au jeu de carte, le 31 (mais avec des règles différentes). En fait, vous simulerez un jeu avec des cartes à jouer et tenterez d'obtenir le score de 31 en conservant un certain nombre de cartes à chaque distribution.

2 Conditions de réalisation

Valeur de la note finale	Type	Durée	Nombre de remises
15 %	Individuel	1 ½ semaine	1

3 Spécifications

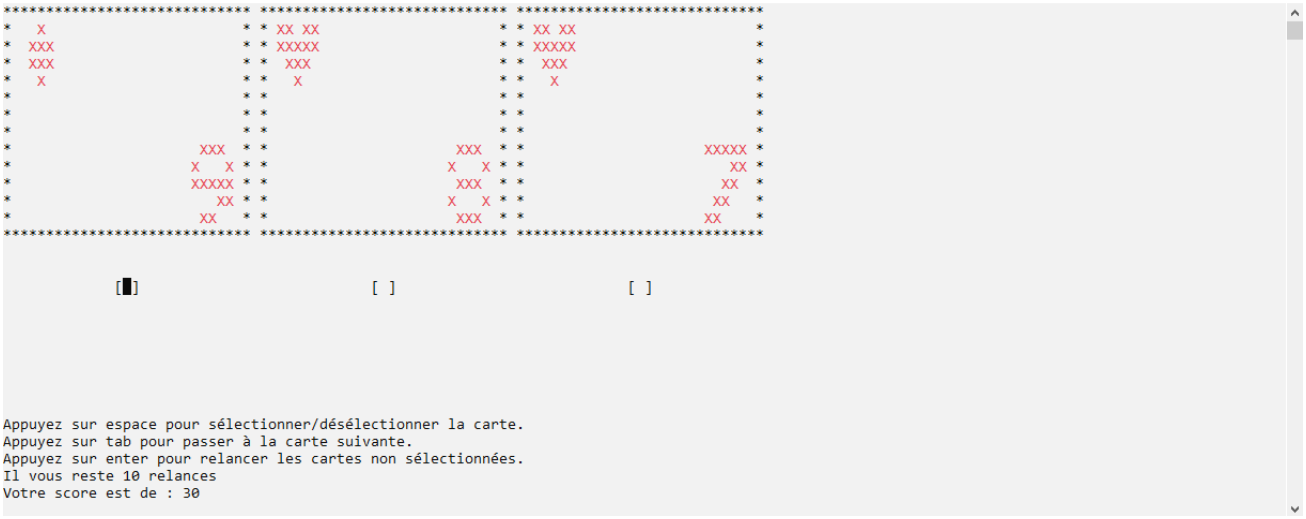
Cette section vise à vous guider durant la réalisation de ce travail. Réalisez chaque étape dans l'ordre et passez à l'étape suivante uniquement lorsque vous avez terminé l'étape actuelle.

3.1 Solution Visual Studio

Ouvrez la solution Visual Studio fournie avec cet énoncé. Elle est déjà configurée et prête à être exécutée. Vous êtes fortement encouragés(ées) à effectuer régulièrement des « commit » de votre projet sur GIT.

Vous ne pourrez pas exécuter correctement l'application au début car plusieurs fonctions doivent d'abord être créées.

Vous disposerez néanmoins d'une interface graphique rudimentaire (en *console*) similaire à celle-ci :



Il vous sera possible de sélectionner une carte pour la conserver en main en appuyant sur ***Espace*** et de passer à la carte suivante en appuyant sur ***Tab***. Quand vous aurez choisi toutes les cartes que vous souhaitez conserver, appuyez sur ***Enter*** pour tirer d'autres cartes.

Le projet de départ contient quelques fichiers. Vous n'avez pas à les modifier sauf le fichier `Game.cs` qui contient certaines fonctions à modifier et qui devra contenir les fonctions nécessaires pour réaliser ce travail.

3.2 La fonction `GetSuitFromFaceIndex`

Tel que son nom l'indique cette fonction a pour but de déterminer la suite (coeur, diamant, pique ou trèfle) associée à un index de carte. Les différentes cartes sont représentées par un nombre entier de 0@51 inclusivement. Les faces 0@12 sont les coeurs, les faces 13@25 sont les diamants, les faces 26@38 sont les piques tandis que celles de 39@51 sont les trèfles.

Voici la représentation des faces selon leur index :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Valeur	As	2	3	4	5	6	7	8	9	10	Valet	Reine	Roi	As	2	3	4	5	6	7	8	9	10	Valet	Reine	Roi
Suite	Cœurs ♥													Diamants ♦												
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
	As	2	3	4	5	6	7	8	9	10	Valet	Reine	Roi	As	2	3	4	5	6	7	8	9	10	Valet	Reine	Roi
	Piques ♠													Trèfles ♣												

3.3 La fonction `GetValueFromCardIndex`

Cette fonction a pour but de déterminer la valeur numérique de la carte, peu importe sa couleur. Par exemple, le « 3 » de coeur est représenté par la valeur numérique 2 tandis que le « 3 » de pique est représenté par la valeur numérique 28. Dans les deux cas, la fonction `GetValueFromCardIndex` doit retourner la même valeur car il s'agit dans les deux cas de « 3 » mais dans deux suites différentes.

3.4 La fonction `DrawFaces`

Cette fonction a pour objectif de distribuer des cartes au joueur. Elle possède trois paramètres :

`facesValues` : les 3 valeurs (index) de cartes que possède déjà le joueur. Ce tableau contient des valeurs entre 0 et 51 inclusivement.

`selectedFaces` :des booléens indiquant si le joueur veut garder la face(`true`) ou s'il souhaite l'échanger (`false`). Ce tableau a la même taille que `facesValues`. Au début de la partie, les 3 booléens sont initialisés par le code fourni à `false`.

`availableFaces` : des booléens indiquant si chacune des 52 cartes est disponible (`true`) ou non (`false`). Évidemment, il ne sera possible de distribuer que les cartes disponibles. Il vous faudra en tenir compte dans votre code. Lorsqu'une carte est rejetée par le joueur, elle redevient disponible lors des tours suivants.
Au début de la partie, les 52 booléens sont initialisés par le code fourni à `true`.

3.4.1 La fonction **`GetScoreFromCardValue`**

Cette fonction a pour but de donner le score associé à une carte. Vous avez des constantes fournies pour donner la position des cartes dans une suite, mais elles ne sont pas nécessairement égales aux scores de celles-ci. Voici la table de la valeur du score pour chaque carte :

Cartes	Valeur de score
Cartes de 2 à 10 (inclusivement)	La valeur sur la carte (donc un 5 donne un score de 5)
Faces (valet, reine et roi)	10
As	11

3.4.2 La fonction **`GetHandScore`**

Le but du 31 est d'obtenir un score de 31 (ou de s'en rapprocher) en additionnant la valeur des cartes dans notre main qui sont de la même suite. Par exemple, une main comprenant un 10, un roi et un as de pique donnerait le score de 31. Cependant, si l'as avait été de cœur par exemple, notre score aurait été seulement de 20 (le score du roi + le score du 10).

Si toutes nos cartes sont de suites différentes, notre score correspond à la valeur de la carte de notre main la plus haute.

Il existe quelques combinaisons spéciales qui donne un score spécifique :

1. Si toutes les cartes de notre main ont la même valeur, on obtient un score de 30 (ex. : avoir le 3 de pique, le 3 de cœur et le 3 de trèfle).
2. Si notre main ne comporte que des faces, on obtient un score de 28 (ex. : avoir le roi de cœur, le roi de pique et la reine de cœur).
3. Si toutes les cartes de notre main sont de la même couleur, on obtient un score de 24 (ex. : le 3 de cœur, le 5 de carreau et le 9 de cœur).
4. Si les valeurs de nos cartes forment une suite, on obtient un score de 26 (ex. : avoir le 3 de pique, le 4 de coeur et le 5 de pique). À noter que les suites des cartes ainsi que l'ordre des cartes dans votre affichage n'ont pas d'impact sur s'il y a une suite ou non.
5. Si les valeurs de nos cartes forment une suite et qu'elles sont toutes de la même couleur, on obtient un score de 28 (ex. : avoir le 5 de pique, le 6 de trèfle et le 7 de pique).
6. Si notre main est composée des trois faces différentes (en d'autres mots, d'une suite de faces), on obtient un score de 30 (ex. : avoir le valet de trèfle, la reine de cœur et le roi de pique).

À noter que cette fonction devrait toujours retourner le score le plus haut de notre main. Donc, si on a une combinaison spéciale dans notre main, on devrait comparer le score que celle-ci donne par rapport à l'addition des cartes de même suite de notre main et retourner le meilleur score. Par exemple, avoir le 10 de trèfle, le 9 de trèfle et le roi de trèfle devrait donner un score de 29 par addition plutôt qu'un score de 24 par combinaison de même couleur.

3.4.3 … et toutes les autres fonctions

Référez-vous aux tests unitaires fournis en commentaires pour déduire les autres fonctions que vous devrez coder. Vous devrez notamment coder les fonctions pour trouver si toutes les cartes en main ont la même valeur ou si elles comportent une suite, ainsi que d'autres fonctions utilitaires

> **Prudence**
> Pensez à faire du code efficace! Réutilisez vos fonctions autant que possible et faites du code qui s'adapterait facilement si on voulait jouer au 41 (même principe mais en essayant d'avoir un score de 41) par exemple. Vous serez évalués(ées) sur cet aspect.

3.5 Autres spécifications techniques

Description
L'usage du langage *C#* est obligatoire.
L'usage de l'environnement Visual Studio est obligatoire.

4 Modalités de remise

Remettre sur LÉA uniquement le fichier **`Game.cs`** contenant votre code.

N'incluez ni l'énoncé ni la grille de correction dans la remise. Ces fichiers sont aussi considérés comme inutiles. Une pénalité est appliquée si ces fichiers sont remis.

Je corrigerai avec les mêmes tests unitaires dont vous disposez. Vous devez donc vous assurer que tous les tests unitaires compilent. Si vous manquez de temps pour terminer le travail, assurez-vous de créer les fonctions utilisées par les tests même si elles n'effectuent pas le travail attendu.

Une pénalité de 15 % est appliqué en cas de retard de moins d'une journée. Au-delà de ce délai, le travail est refusé et la note de 0 % est automatiquement attribuée.

** Un projet qui ne compile pas ne sera pas corrigé et se méritera donc une note de zéro. **

5 Évaluation

Vous trouverez dans le fichier ci-joint la grille d'évaluation qui sera utilisée pour le travail pratique.