Université Paris 13 - Licence d'informatique Année 2009-2010

# Projet de *Programmation Fonctionnelle* n° 1

## Blackjack

Le but de ce projet est de développer en OCaml une version du jeu Blackjack (vingt-et-un en français). Les questions de l'énoncé sont très détaillées au début, et moins vers la fin pour vous encourager à faire preuve d'originalité. Vous avez le droit d'utiliser n'importe quelle fonction définie dans la bibliothèque standard de OCaml. Il vous est impérativement demandé de programmer en style fonctionnel, aucune primitive impérative (boucles `for`, `while`, arrays, etc.) ne vous sera permise.

**Soutenance de projet.** Vous devrez rendre à la mi-mai (la date exacte vous sera communiquée ultérieurement) un fichier `.tar.gz` contentant :

- un rapport en format `pdf` détaillant précisément le travail effectué et les choix d'implémentation ;
- un fichier `.ml` contenant le code correspondant aux exercices de la section 2, en suivant rigoureusement l'énoncé, et sans aucune extension ;
- un ou plusieurs fichiers correspondant aux exercices de la section 3, permettant de construire un exécutable ;
- un fichier `README` expliquant les commandes à taper pour générer l'exécutable des exercices de la section 3 et le lancer.

Une soutenance de projet sera organisée fin mai. Vous devrez faire une démonstration de votre programme et une présentation de vos choix. Ces soutenances auront lieu sur les machines des salles de TP et sur votre compte. Il est impératif que la démonstration soit prête (programme déjà compilé et sans erreur) dès le début de la soutenance.

**Évaluation.** Le travail devra être fait en binôme, mais la soutenance est individuelle. Le travail sera évalué selon les critères suivants :

- qualité de la conception ;
- fonctionnalités réalisées ;
- qualité du code écrit, lisibilité (votre code doit être bien indenté, et imprimable sans remise en page) ;
- soutenance orale ;
- qualité du rapport.

On préférera un projet simple dont les choix de conception sont bien motivés et qui marche, à un projet trop ambitieux dont on ne comprend pas vraiment la structure et qui ne marche pas trop bien.

# 1 Le jeu

Le Blackjack est un jeu de cartes dans lequel le joueur tente de battre le casino, représenté par le croupier, en obtenant une main d'une valeur égale ou inférieure à 21 et supérieure à la main du croupier. Bien qu'il y ait plusieurs joueurs autour d'une table de Blackjack, la relation est toujours joueur contre casino, et jamais joueur contre joueur. Une table de blackjack est généralement composée d'un croupier (le donneur) et de un à sept joueurs. On utilise en général de un à huit jeux.

**Valeurs des cartes.** Chaque carte possède une valeur égale à son poids, sauf pour les as et les figures. Tous les dix et figures comptent pour 10. Les as peuvent être comptés pour 1 ou 11 selon le choix du joueur. Les couleurs ne comptent pas.

Par exemple, une main composée d'un 3 de carreau, un 6 de coeur et une dame de pique vaudra 19, une main composée d'un as de trèfles et un 10 de pique vaudra 21.

Lorsque les deux premières cartes d'un joueur sont un as et une carte d'une valeur de dix, faisant donc un total de 21, on dit que le joueur a un *Blackjack*. En général un blackjack est payé 3/2 de la mise. Lorsque le joueur et le donneur ont tous les deux un blackjack il y a égalite, et le joueur récupère sa mise (sans gain).

**Actions du joueur.** Chaque joueur fait sa mise initiale et puis reçoit deux cartes, face visible, distribuées une par une, dans le sens des aiguilles d'une montre. Le donneur reçoit une carte face visible et l'autre face cachée.

Une fois la donne effectuée, l'un après l'autre, les joueurs peuvent, s'ils le souhaitent, demander une ou plusieurs cartes supplémentaires jusqu'à ce qu'il souhaite arrêter ou bien lorsqu'il *saute* (la somme de ses cartes faisant plus de 21). Selon les règles internes du casino, le joueur peut avoir la possibilité de doubler sa mise initiale après avoir reçu ses 2 premières cartes. On appelle cela *doubler*. Dans ce cas le joueur ne reçoit qu'une seule carte supplémentaire.

Lorsque la main d'un joueur dépasse 21, on dit qu'il a sauté et il perd sa mise quel que soit le résultat de la main du donneur. Lorsqu'un joueur saute, ses cartes et sa mise sont aussitôt collectées par le croupier.

Si le joueur et le donneur possèdent chacun une main d'une même valeur sans avoir sauté, on dit qu'il y a égalité et le joueur ne perd pas sa mise, mais ne gagne rien non plus. Néanmoins un Blackjack sera toujours vainqueur face a une main de 21 (ex. 10 pique + 4 coeur +7 pique).

**Actions du donneur.** Une fois que tous les joueurs ont joué et si il reste des joueurs n'ayant pas sauté, le donneur decouvre sa carte face cacheé et se distribue des cartes, si nécéssaire, jusqu'à ce que sa main aie atteint un total de 17 ou plus. Après cela, si le donneur a sauté, il paie les joueurs restants, sinon il paie les joueurs ayant une main plus forte que la sienne.

# 2 Programmer les fonctions principales

La première partie du travail, à faire **impérativement avant de commencer à écrire le code**, consiste à bien réfléchir sur :
- les structures de données à utiliser ;
- les types à définir ;
- les fonctions principales dont vous aurez besoin ;
- les exceptions qui vous devrez gérez.

Cette partie est fondamentale, il vous est conseillé de faire un cahier de charges précis, de décider des choix de découpage du code avant de commencer l'implémentation.

**Exercice 1** D'abord on vous demande de définir des nouveaux types afin de gérer les données à utiliser. On vous suggère les types suivants (l'originalité sera quand même bien appréciée) :
- `nom` : le type des noms des joueurs ;
- `cartes` : pair d'un nombre $(1, \dots, 10)$ ou figure $(j, q, k)$ et un couleur ;
- `main` : une liste de cartes qui décrive la main d'un joueur ou du croupier ou bien le jeu ;
- `argent` : une représentation de l'argent (par exemple `Euro 25`) ;
- `croupier` : la description du croupier, c.à-d. de sa main ;
- `joueur` : la description d'un joueur, c.à-d. son nom, sa main, l'argent dans sa poche et sa mise sur la table ;
- `table` : la description d'une table, c.à-d. le croupier et la liste des joueurs.

**Exercice 2** Le but de cet exercice est donner les outils pour manipuler le jeu de cartes. On vous demande de suivre rigoureusement l'énoncé, en donnant le code pour les fonctions qui suivent.
- `jeux : int -> main` qui prend en entrée un entier `n` $\in [1, \dots, 8]$ et retourne en sortie un jeu de `n` $\times$ 52 cartes. La fonction devra soulever une exception dans le cas `n` $\notin [1, \dots, 8]$.
- `melanger : main -> main` qui mélange le jeu. Il faut utiliser un *random number generator* pour que les cartes soient mélanges au hasard.
- `distribution : main -> main -> (main, main)` qui prend en entrée le jeu `d` et la main `m` d'un joueur ou du croupier et donne à ce dernier la première carte parmi les cartes de jeu. Plus précisément, la fonction retourne une paire ayant comme premier élément le jeu privé d'une carte et comme deuxième élément la main du joueur augmentée avec la carte privée du jeu. La fonction devra soulever une exception dans le cas le jeu est vide.

**Exercice 3** On définie les fonctions pour calculer les gagnants d'une table.
- `value : main -> int list` qui prend en entrée une main et donne en sortie la liste de ses valeurs possibles, selon que les as valent 1 ou 11.
- `gagnant : table -> (nom*int) list` qui calcule les gagnants et les perdants d'une table, c.à-d. `gagnant` prend en entrée une table et retourne en sortie la liste des noms de joueurs accouplés avec un entier qui code une parmi les situations possibles : le joueur perd, le joueur gagne avec Blackjack, le joueur gagne sans Blackjack, il y a égalité entre joueur et croupier.

**Exercice 4** Let but de cet exercice est de programmer une fonction `croupier` qui simule le jeu du croupier. Cette fonction aura comme type `table -> bool`, étant donné une table elle retourne `true` si le croupier décide de demander une carte, `false` si il termine le jeu.

**Exercice 5** Jouer tout seul à une table de Blackjack c'est triste parfois ! On vous demande donc de programmer le jeu d'un joueur, qui pourra vous faire compagnie quand vous jouerez contre le croupier. On vous suggère de programmer trois fonctions distinctes :

- `mise : table -> argent`
- `double : table -> bool`
- `joueur : table -> bool`

La première décide la mise selon l'argent disponible et la mise des autres joueurs. La fonction `double` décide si doubler la mise initiale ayant considéré les deux premières cartes. La fonction `joueur` décide si demander une carte au croupier ou terminer son jeu. On détaille pas comment programmer ces fonctions pour vous encourager à faire preuve d'originalité. Il sera bien apprécié un joueur malin, dont sa décision de prendre ou pas une carte doit maximaliser les probabilités de gagner d'argent. Le joueur tiendra en compte au moins de :

- ses cartes,
- les cartes visibles sur la table,
- la mise de chaque joueur.

Il faut surtout avoir des joueurs honnêtes, qui ne voient que les cartes visibles sur la table.

# 3 Interface avec l'utilisateur

**Exercice 6** Écrire une interface textuelle qui dialogue avec un utilisateur. En gros, on s'attend le fonctionnement suivant :

- l'utilisateur d'abord choisit des options comme le nombre des joueurs (éventuellement leur noms), l'argent de départ, quels joueurs seront géres par lui même et quels joueurs seront gérés par le programme, le nombre de jeux, etc ;
- le programme crée une table (en particulier distribue les cartes aux joueurs et au croupier) et affiche la table à l'écran ;
- à son tour l'utilisateur peut regarder les cartes visibles sur la table et décider si doubler sa mise initiale, ou demander une ou plusieurs cartes supplémentaires ou s'arrêter ;
- quand tous les joueurs auront fait leur jeu, le programme fait jouer le croupier, en montrant ses cartes ;
- enfin la partie termine en distribuant l'argent sur la table selon les gagnants et les perdants ; une nouvelle partie peut commencer (si il y a encore des joueurs avec de l'argent !).

Attention ! il sera bien évalué la gestion des exceptions, par exemple dans le cas où l'utilisateur tape des caractères qui ne sont pas valides, ou cherche de miser plus argent qu'il en dispose, ou encore saute (c.à-d. dépasse la vingt-et-un), etc.