Université du Québec
École de technologie supérieure

INF111 : A2021 Travail pratique 1

Auteur : Pierre Bélisle
Amélioration : Patrice Boucher
Travail en équipe d'au maximum 4 personnes.

1. Objectifs

Ce travail a pour principal objectif de mettre en pratique la syntaxe Java et les différentes notions enseignées jusqu'à présent : les variables, les boucles (*while, do-while, for*), les sélections (*if-else, switch, ?:*), l'utilisation de fonctions existantes, l'implémentation de fonctions et la manipulation de tableaux. Il vous est demandé de factoriser (découper en fonctions) votre programme au maximum, en créant les différentes fonctions nécessaires à la réalisation du problème énoncé plus loin.

2. Description générale : Le jeu de Yum

Vous devez écrire un programme permettant à un utilisateur de jouer au jeu de *Yum*. Yum est un jeu de dés qui se joue en 13 tours. À chaque tour, le joueur essaie d'obtenir une combinaison stratégique de ses 5 dés afin de remplir, avec un maximum de points, une case de son choix d'une grille de pointage. Chacune des 13 cases comporte un objectif particulier :

- Obtenir des 1 (pointage = somme des 1)
- Obtenir des 2 (pointage = somme des 2)
- …
- Obtenir des 6 (pointage = somme des 6)
- *Brelan* : Obtenir 3 dés identiques (pointage = somme des dés)
- *Carré* : Obtenir 4 dés identiques (pointage = somme des dés)
- *Courte séquence* : obtenir une séquence de quatres dés (15 points), par exemple : 2, 3, 4, 5
- *Longue séquence* : obtenir une séquence de 5 dés (20 points)
- *Roulement de surplus* : obtenir un ensemble dont la somme est maximale (pointage = somme des dés)
- *Main plain :* obtenir 3 dés identiques et 2 autres dés identiques (25 points)
- *YUM* : obtenir 5 dés identiques (30 points).

À chaque tour, le joueur dispose de trois lancers possibles. À chacun des 2 lancers suivant le premier lancer, il peut décider de relancer les dés de son choix afin d'espérer atteindre un objectif particulier. Dans le cas où l'utilisateur désire garder tous les dés pour ce tour, il entre 0. Chaque objectif, c'est-à-dire chaque case, peut être rempli qu'une seule fois leur de la partie. À la fin des 13 tours, le joueurs obtient un bonis de 25 points s'il a réussi à obtenir un cumul d'au moins 63 points dans ses 6 premiers objectifs. Son pointage total correspond au cumul des pointages de tous les objectifs et du bonis (si applicable). *Au besoin, vous pouvez consulter la version originale des règles du jeu dans le document Yum.pdf et une démonstration vidéo* ici. Assurez vous de comprendre cette feuille de pointage avant de poursuivre.

YUM

BONNE CHANCE

Partie 1

Nom :
Date :

JEU DE DÉS	Score maximal	1	2	3
1	5			
2	10			
3	15			
4	20			
5	25			
6	30			
Sous-total	-			
Boni (63 et plus)	25			
Total Partie Supérieure	-			
3 pareils	**Total des dés**			
4 pareils	**Total des dés**			
Courte séquence de 4	15			
Longue séquence de 5	20			
Roulement de surplus	**Total des dés**			
Main Pleine	25			
YUM (5 pareils)	30			
Total Partie Inférieure	-			
Total Partie Supérieure	-			
TOTAL	-			
TOTAL - Partie	-			

3. Mandat

Ici, nous présumons que vous connaissez le jeu.

Votre programme doit permettre de jouer au Yum. À chaque tour, vous devez afficher la feuille de pointage à jour et, pour chacun des 3 essais, les dés roulés et une grille de points possibles. .

Grille de pointage au départ

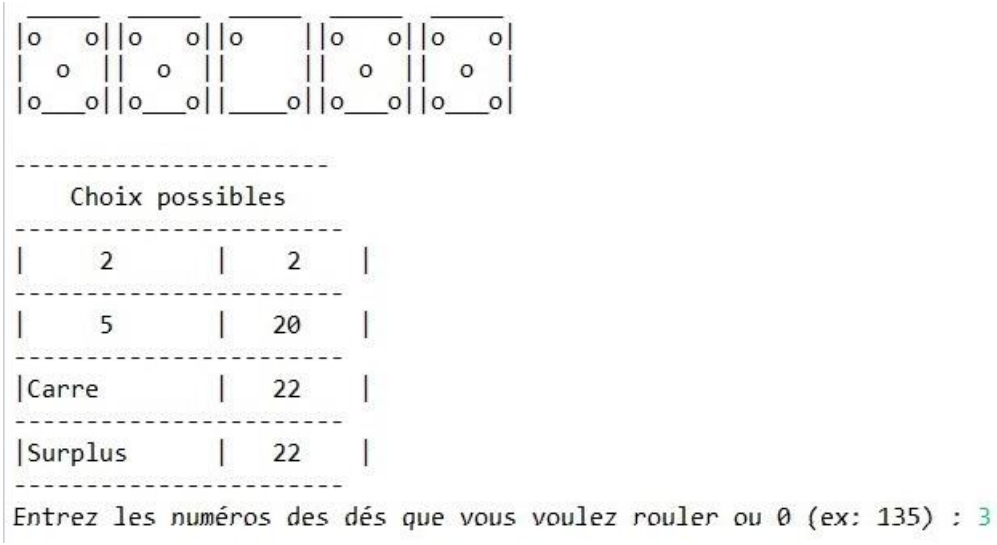
```
                GRILLE DE POINTAGE
-------------------------------------------------------
    1       |       |        |Brelan       |       |
--------------------          ------------------------
    2       |       |        |Carre        |       |
--------------------          ------------------------
    3       |       |        |Main pleine  |       |
--------------------          ------------------------
    4       |       |        |Petite suite |       |
--------------------          ------------------------
    5       |       |        |Grosse suite |       |
--------------------          ------------------------
    6       |       |        |Surplus      |       |
--------------------          ------------------------
Sous total  |   0   |        |YUM          |       |
--------------------          ------------------------
Bonus       |   0   |        |Total bas    |   0   |
--------------------          ------------------------
total haut  |   0   |        |Total        |   0   |
--------------------          ------------------------
```

L'affichage des dés et de la grille des coups possibles (premier essai)

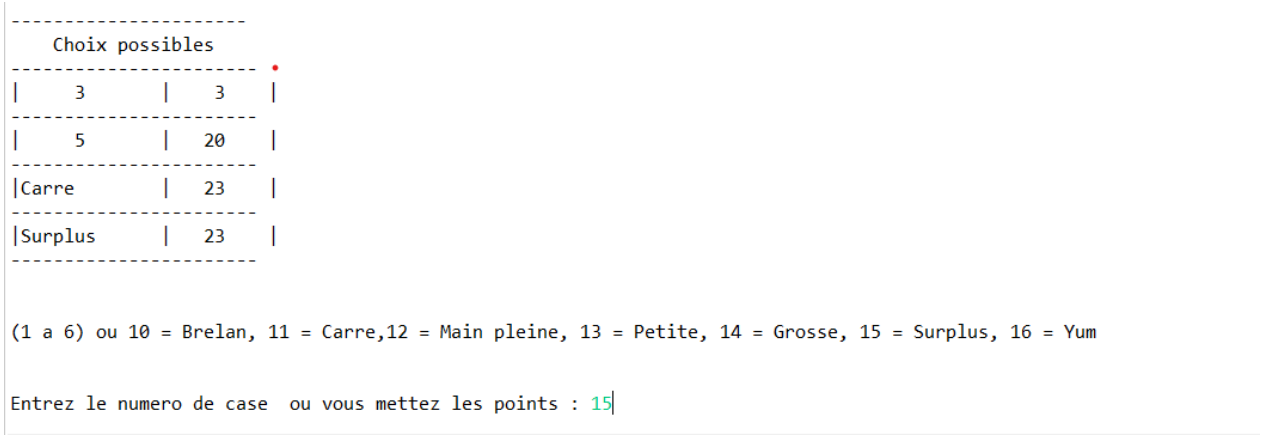
```
 _____  _____  _____  _____  _____
|o   o||o   o||o    ||o   o||o   o|
|  o  ||  o  ||  o  ||     ||o   o|
|o___o||o___o||____o||o___o||o___o|

----------------------
    Choix possibles
----------------------
|     3      |    3   |
----------------------
|     4      |    4   |
----------------------
|     5      |   10   |
----------------------
|     6      |    6   |
----------------------
|Surplus     |   23   |
----------------------
Entrez les numéros des dés que vous voulez rouler ou 0 (ex: 135) : 345
```

Par exemple, 345 indique de rouler à nouveau les dés 3, 4 et 5 (qui ont actuellement les valeurs 3, 6 et 3), 12345 indique de rouler tous les 5 dés,14 indique de rouler seulement les dés 1 et 4. L'entier entré doit être validé : c'est-à-dire que chaque chiffre doit être entre 1 à NB_DES et il ne doit pas y avoir de doublon.

L'affichage des dés et de la grille des coups possibles (deuxième essai)



L'affichage des dés et de la grille des coups possibles (dernier essai)



L'utilisateur doit décider quelle case il choisit dans vos suggestions. Il y aura de la validation à effectuer : le nombre doit être entre 0 et 16 et la case ne doit pas être occupée[1]). Si l'utilisateur entre 0, la partie se termine à l'instant.

[1] On ne validera pas les erreurs de type : l'utilisateur entre un caractère, ça plante.

La partie se termine lorsque les 13 cases sont remplies ou que l'utilisateur a annulé en entrant zéro comme numéro de case pour la feuille de pointage. Dans les deux cas, votre programme demandera si l'utilisateur veut jouer une autre partie. Vous devrez valider la réponse.

4. Conception

L'organisation des données que nous vous **imposons** pour réaliser ce jeu est d'utiliser 3 tableaux. Un tableau de cinq cases, représentant les dés, qui contiendront des valeurs allant de 1 à 6. Deux tableaux de 19 cases représentant la feuille de pointage et une grille des coups possibles (Brelan (3 pareils), carré (4 pareils), main pleine (full), etc.). Ce dernier tableau servira à suggérer au joueur les coups possibles pour les dés joués.

Toutes les cases de la feuille de pointage[2] seront initialisées avec la valeur -1 puisque zéro est un pointage possible. Cette valeur bidon servira à détecter si une case a déjà été choisie ou non.

Toutes les cases de la grille des coups possibles seront initialisées avec des zéros à chacun des 13 tours et cela pour chaque tour. Elles seront remplies éventuellement avec les possibilités que donnent les dés pour un tour. Pour faciliter la correspondance entre les deux tableaux, il y aura des cases inutilisées dans la grille des coups possibles.

Description par l'exemple

Voici un exemple de nos trois tableaux. Le premier est un exemple de tableau de dés après 3 essais, le deuxième est la grille de possibilités à partir du tableau de dés et ensuite on montre différentes options pour la feuille de pointage selon le choix du joueur (cela pour un tour seulement).

(après 3 essais)

0	1	2	3	4
4	6	4	4	6

Grille des coups possibles précédent)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	0	0	12	0	12	0	0	0	0	0	25	0	0	24	0	0	0

Feuille de pointage au départ.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	-1	-1	-1	0	0	0	-1	-1	-1	-1	-1	-1	-1	0	0

Feuille de pointage si l'utilisateur choisit la main pleine.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	-1	-1	-1	0	0	0	-1	-1	**25**	-1	-1	-1	-1	0	0

[2] Sauf pour les cases représentant les totaux et les bonus, qui devront être initialisées à zéro

Feuille de pointage si l'utilisateur choisit les 4.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	**12**	-1	-1	0	0	0	-1	-1	-1	-1	-1	-1	-1	0	0

Feuille de pointage si l'utilisateur choisit le carré.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	-1	-1	-1	0	0	0	-1	**24**	-1	-1	-1	-1	-1	0	0

Feuille de pointage si l'utilisateur choisit les 6.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	-1	-1	**12**	0	0	0	-1	-1	-1	-1	-1	-1	-1	0	0

Feuille de pointage si l'utilisateur choisit le SURPLUS.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-1	-1	-1	-1	-1	-1	0	0	0	-1	-1	-1	-1	-1	**24**	-1	0	0

*** Après chaque essai, la grille des possibilités est calculée et affichée. La feuille de pointage nécessite le choix du joueur quant à elle avant d'être mise à jour et affiché.

*** IMPORTANT : Ces variables doivent être déclarées dans le programme principal. Pas de déclarations **static** de ces trois tableaux.

5. Algorithmes

Voici différentes descriptions d'algorithme, parfois en français et parfois en pseudocode. D'autres algorithmes de base vous apparaîtront nécessaires mais ne sont pas décrits (la somme d'un tableau sans les -1 par exemple).

5.1 Algorithme pour rouler les dés choisit par le joueur :

Écrivez un sous-programme qui reçoit le tableau de dés et un entier qui sera déjà validé avant l'appel (pas besoin ici).

L'entier représente les dés à rouler. Par exemple, 12345 signifie de rouler les 5 dés tandis que la valeur 14 roule seulement les dés 1 et 4. Chaque chiffre doit être entre 1 à NB_DES. Il n'y doit pas y avoir de doublon.

Il faudra donc une boucle pour chaque chiffre de cette variable et rouler la bonne case dans le tableau de dés.

Considérer :

- Un nombre entier positif modulo 10 donne son dernier chiffre et la partie entière du même nombre divisé par 10 enlève le dernier chiffre. Ex : 135%10 donne 5 et 135/10 donne 13.

- Les cases vont de 0 à NB_DES -1 mais pas les chiffres du nombre (1 à NB_DES).
- 523 est une combinaison valable, les dés 2, 3 et 5 seront roulés.

5.2 Algorithme pour gérer les trois essais

Vous aurez besoin de recevoir les 3 tableaux en paramètres.

Rappelons-nous que le joueur peut décider de garder tous les dés et de passer directement à la feuille de pointage. L'algorithme présenté favorise la boucle et sort le dernier essai de la boucle pour éviter une sélection dans la boucle.

Pseudocode
Début

Il faut un itérateur pour compter le nombre d'essais.

Nous suggérons un booléen à faux qui sera mis à vrai si le joueur garde tous les dés (il entrera 0).

Initialiser une variable à 12345, pour rouler les 5 dés. Elle sera transmise à la procédure qui roule les dés.

Tantque les 2 premiers essais ne sont pas faits que le booléen est à faux

Effectuer le choix aléatoire des dés selon la valeur de la variable (12345 c'est tous les dés).

Afficher les dés

Calculer les possibilités

Afficher les possibilités.

Obtenir le choix du joueur, une valeur sans doublon où chaque chiffre représente le dé à rouler. (Mettez cette valeur dans votre variable pour rouler les dés au prochain tour de boucle).

Si la valeur saisie est 0, c'est qu'il garde tous les dés alors mettre le booléen à vrai. Ce qui terminera la boucle au prochain tour.

Incrémenter l'itérateur

Fin de la boucle pour les essais

Si le joueur ne garde pas tout (booléen à faux) et qu'il a terminé ses deux essais précédents, il reste à rouler les dés choisis une dernière fois, calculer et afficher les possibilités du dernier tour.

Fin

5.2 Algorithme pour trouver les coups possibles :

Pour arriver à suggérer les coups possibles, il faudra vérifier toutes les combinaisons de points possibles. Voici l'algorithme à implémenter mais décrit en français. Vous devez le comprendre et le traduire.

Ayez encore en tête avant de commencer que vous aurez à écrire plusieurs petites fonctions :

- **Une fonction qui retourne le nombre d'occurrences d'un dé** dans le tableau de dés (combien de 6 par exemple).

- **Une fonction qui retourne si vous avez une main pleine :** dans l'algorithme décrit après, comprenez qu'elle est appelée seulement s'il y a déjà eu trois dés identiques. Il reste simplement à voir s'il y a un autre dé avec 2 occurrences.

- **Une fonction pour vérifier les suites :** Compter le nombre d'occurrences de chaque possibilité de 1 à 6. Chaque variante de la petite suite contient un 3 et un 4. Il reste à regarder s'il y a un 1 et 2 ou un 2 et un 5 ou un 5 et un 6.

  Pour la grosse suite, il faut une seule occurrence de 1,2,3,4,5 ou de 2,3,4,5 et 6.

- **Une fonction pour générer les dés** : au départ mettez des valeurs fixes (ex : [6 6 6 6 6], [1 6 5 3 4], …) pour tester différents cas. Testez un cas à la fois. Par exemple vérifier que le pointage de 1 à 6 fonctionne. Ensuite, le pointage du brelan, le pointage du carré, … Lorsque chaque case de la feuille de pointage se remplie correctement vous modifierez la fonction pour qu'elle génère les valeurs aléatoirement (Math.random() * NB_FACES + 1).

- **Une procédure pour saisir et valider les dés choisis à rouler :** Chaque chiffre entre 1 et 5 sans doublon. À valider avant de rouler les dés.

- **Une procédure pour valider le choix de la case de la feuille de pointage :** L'utilisateur devra décider quelle case il choisit dans vos suggestions. Il y aura de la validation à effectuer. (un nombre entre 0 et 13 et que la case n'est

pas déjà prise[3]). À cet endroit, si l'utilisateur entre zéro c'est qu'il désire annuler la partie.

- **Une procédure pour appliquer le pointage à la feuille de pointage :** Selon le choix du joueur, il faudra mettre les points dans le tableau et calculer le sous-total, le bonus, et les totaux.

- Nous vous fournissons les procédures d'affichage des dés et de la feuille de pointage, **vous devez écrire la procédure d'affichage pour la grille de coups possibles** (vous n'afficherez que les cases qui sont différentes de zéro de ce tableau).

Voici donc enfin l'algorithme à implémenter pour générer les coups possibles qui seront affichés ensuite. Cet algorithme suit le moment où joueur a roulé les dés pour un essai (décrit 5.1). Nous avons besoin du tableau de dés, de la feuille de pointage et de la grille des coups possibles. Il précède l'affichage de cette grille de coups possibles.

Début :

Mettre la grille de coups possibles à zéro.

Pour chacune des valeurs D possibles (1 à 6), il faut vérifier le nombre d'occurrences[4] de la valeur D dans le tableau de dés. Si le nombre d'occurrences de D (par exemple 2) dans le tableau de dés est égal à zéro, on ne fait rien. Si ce n'est pas le cas alors on doit agir différemment selon le nombre d'occurrences de D.

S'il y en a cinq, c'est un Yum. Si c'est le premier (il y a -1 dans la feuille de pointage), on met 30 points dans la case YUM de la grille des coups .

Il faut mettre dans la case associée à D, de la section supérieure, combien de points elle rapporte (si la case est libre bien sûr).

Il faut regarder si le nombre d'occurrences est plus grand ou égal à trois. Si c'est le cas et que la case BRELAN est libre dans la feuille de pointage, on met la **somme des dés** dans la case appropriée de la grille. Nous vérifierons aussi si c'est un MAIN_PLEINE et que la case MAIN_PLEINE est libre dans la feuille de pointage. Si c'est le cas on met 25 points dans la case appropriée dans la grille.

Il faut regarder si le nombre d'occurrences de D est égal à quatre et que la case CARRÉ est libre dans la feuille de pointage, on met la **somme des dés** dans la grille.

La boucle est finie…mais ce n'est pas tout.

---

[3] On ne validera pas les erreurs de type (L'utilisateur entre un caractère, ça plante).
[4] Utilisez la fonction prévue.

On regarde maintenant les suites. S'il y a (une petite suite et que la case PETITE est libre dans la feuille, on met 15 points dans la grille. Si c'est une grosse et que la case GROSSE est libre dans la feuille, on met 20 points dans la grille. Finalement on met la **somme des dés** dans la case SURPLUS de la feuille de pointage.

Fin

Cet algorithme peut mais **ne doit pas** être implémenté dans une seule fonction.

5.3 Programme principal :

La partie se termine lorsque les 13 cases sont remplies ou que l'utilisateur a annulé en entrant zéro comme numéro de case pour la feuille de pointage. Dans les deux cas, votre programme demandera si l'utilisateur veut jouer une autre partie. Vous devrez valider la réponse.

Pseudocode

Début

Tant que l'utilisateur veut jouer

Mettre un booléen à **faux** pour l'option d'annulation.
Mettre un itérateur à 1;

Tant que la partie n'est pas finie et que i <= *NB_TOUR*

Afficher la grille de pointage

Récupérer le résultat des 5 dés choisis
par l'utilisateur après 3 lancés (5.2)

Obtenir et valider le choix de l'utilisateur pour la feuille de pointage (0 pour annuler la partie)

Si l'utilisateur a choisi 0
L'annulation est mis à vraie

Sinon
On applique les points à la grille de jeu (SOUS_TOTAL_HAUT, TOTAL_HAUT, TOTAL_BAS, GRAND_*TOTAL*). C'est ici qu'on vérifie et applique le bonus s'il y a lieu.

Fin Si

Incrémenter l'itérateur

Fin de la boucle de la partie

Afficher la grille une dernière fois

Demander si l'utilisateur veut jouer encore et valider sa réponse avant de recommencer la boucle. Aucun autre choix que 'o' ou 'n'.

Fin de la boucle principale

Afficher un message de fin

Fin de l'algorithme principal

6. Développement du programme :

6.1 Rappel des approches :

Si vous prenez une approche ascendante, vous écrivez un sous-programme avant de l'appeler et de le tester.

Pour l'approche descendante, vous appelez le sous-programme avant de l'écrire, vous l'écrivez et le testez.

6.2 Suivez les étapes de développement que nous vous avons présentées en classe (descendant - ascendante). Quelques conseils :

1. Faites la liste des exigences des sous-problèmes (1 à la fois).
2. Bien comprendre tous les cas possibles des algorithmes proposés.
3. Découpez le problème en plus petits problèmes[5].
4. Décidez du meilleur ordre d'implémentation pour vous (ascendant ou descendant?)
5. Codez
6. Testez
7. Déboguez (Un bogue peut venir du test comme du sous-programme testé).

6.3 Truc : Faites fonctionner votre programme pour un tour et lorsqu'il fonctionne mettez les boucles pour 13 tours. Lorsque tout fonctionne ajoutez la boucle qui permet de jouer plusieurs parties et la fonction de validation de la réponse. Autrement dit, attendez à la fin pour implémenter cela.

[5] On en a déjà fait beaucoup pour vous☺

6.4 Prenez note

Un bon travail donne environ 40 heures de par élève travail (incluant le temps pour comprendre la théorie). Ce travail au total devrait donner environ 700 à 800 lignes (code et commentaires). Vous êtes en équipe pour un peu plus de 2 semaines, c'est peu (seulement 6 heures en labo). N'attendez pas pour commencer.

On s'attend à du code de qualité et une exécution fonctionnelle.

7. Contraintes de l'enseignant

- La note du travail sera la note moyenne que l'équipe obtient à l'examen intra I.

- N'oubliez pas de mettre le nom des contributeurs pour chaque sous-programme dans le code.

- Il est interdit de dédier un-e étudiant-e à la **seule** écriture ou la **seule** révision des commentaires. En d'autres mots, il-elle n'écrit pas une ligne de code mais il-elle écrit ou révise seulement les commentaires à la fin, c'est 0/100 pour la contribution. Les commentaires doivent être écrit au fur et à mesure que le code est écrit.

- Les tests aussi doivent être commentés.

- Il n'est pas permis de copier-coller ni de consulter le code ne provenant pas d'un membre de votre groupe. En cas de problème, n'hésitez pas à me consulter aux séances ou par courriel. Cette infraction de nature académique détectée sera dénoncée au comité prévu pour ces cas.

8. Barème de correction

Exécution et tests 40%

- Doit respecter le comportement décrit dans l'énoncé.

Qualité de programmation 60%

- Liste non-exhaustive des erreurs pénalisées:

1. Nom de fonctions et méthodes non représentative.
2. Nom de variables et constantes non représentative.
3. Aération et/ou indentation laissent à désirer
4. Découpage en sous programmes insuffisants.
5. Répétition inutile de code dû au manque de sous programmes.
6. Répétition inutile de code dû à l'incompréhension de l'utilité du paramétrage.
7. Constantes non définies.
8. Constantes non utilisées lorsque possible (même dans les commentaires).
9. Constantes en minuscules.
10. Commentaire d'en-tête de programme manquant (explication, auteurs et version).
11. Commentaires des constantes manquants.
12. Commentaires des variables manquants.
13. Commentaires de déclaration des sous-programmes manquants (API).
14. Commentaires non judicieux ou inutile.
15. Commentaires manquants sur la stratégie employée dans chaque sous-programme dont l'algorithme n'est pas évident et nécessite réflexion (explique comment).
16. Commentaires mal disposés (ex : en bout de ligne).
17. Non utilisation d'un sous-programme lorsque c'est possible ou exigé.
18. Code inutile.
19. Affichage dans une fonction de calcul.
20. Qualité du français dans les commentaires.
21. Non-respect des droits d'auteurs (mettre toutes les références incluant les vôtres).
22. Non utilisation de boucle lorsque possible.
23. Autres (s'il y a une pratique non énumérée qui n'a pas de sens).

Comme vous pouvez le constater, le respect des normes de programmation Java est le critère le plus important.

C'est en programmant qu'on apprend à programmer!

Bon travail!

Instructions

YUM® Classique

But du jeu

Compter le plus grand nombre de points en formant des combinaisons de dés. Les joueurs doivent tenter de compléter les combinaisons de dés inscrites sur leur fiche de pointage en accumulant le plus de points possible. Pour ce faire, chaque joueur a le droit de lancer les dés jusqu'à trois fois par tour de jeu. Si une combinaison n'est pas complétée après le troisième lancer, le joueur peut décider de comptabiliser son pointage de la façon la plus avantageuse pour lui.

Préparation du jeu

Chaque joueur prend une fiche de pointage et la place près de lui sur la table. On détermine au sort qui sera le premier à jouer. Les joueurs joueront à tour de rôle dans le sens des aiguilles d'une montre. On place les dés dans le gobelet.

Déroulement du jeu

Le premier joueur lance les dés. En fonction des chiffres qu'il obtient lors de ce premier lancer, il décide quelle combinaison il désire faire. Il laisse donc sur la table les dés qu'il pourra utiliser pour la combinaison choisie et lance les autres dés. Il peut également choisir de lancer de nouveau les cinq dés. Il peut lancer les dés un maximum de trois fois par tour de jeu. Si la combinaison qu'il a choisie est complétée au deuxième lancer, le joueur peut comptabiliser ses points et ne pas effectuer son troisième lancer. C'est alors au tour du joueur suivant. S'il arrive que le joueur ne puisse compléter une combinaison même après trois lancers, il doit alors comptabiliser son pointage de la façon la plus avantageuse pour lui.

Exemple :
En essayant de compléter une Main pleine (trois pareils et une paire), un joueur termine son tour avec trois 6, un 2 et un 3. Il peut additionner les trois 6 et inscrire un pointage de 18 à la sixième case du «Jeu de dés 1, 2, 3, 4, 5, 6 » ou à la case 3 pareils, sur la fiche de pointage.

Un joueur ne peut faire qu'une combinaison par tour de jeu. Ainsi, s'il obtient par exemple une Main pleine à son troisième lancer de dés et que cette dernière a déjà été inscrite sur sa feuille de pointage, il doit comptabiliser son pointage de la façon la plus avantageuse pour lui.

Exemple : Trois 6 et deux 5 peuvent être comptabilisés pour former un pointage de 28 à la case Roulement de surplus OU les trois 6 peuvent être comptabilisés pour un total de 18 à la sixième case du « Jeu de dés 1, 2, 3, 4, 5, 6 », OU les deux 5 peuvent être comptabilisés pour un total de 10 à la cinquième case dans cette même section.

Si ces alternatives ont déjà été comptées, le joueur doit alors inscrire un pointage de 0 dans une autre case de son choix. Lorsque tous les joueurs ont complété leurs treize combinaisons, les scores sont comptabilisés. Le joueur ayant obtenu le plus haut score remporte la partie.

Partie supérieure

• JEU DE DÉS 1, 2, 3, 4, 5, 6

Le nombre total de pareils en trois lancers, par exemple :

(3 + 3 + 3) **compte 9 points** dans la case des 3.

• BONI

Si un joueur **compte 63 points** ou plus dans le «Jeu de dés 1, 2, 3, 4, 5, 6», il obtient un **boni de 25 points.**

Partie inférieure

• 3 PAREILS

Trois dés assortis, le joueur compte le total de tous les dés (3 pareils + les deux autres dés).

(4 + 4 + 4 + 5 + 6) **compte 23 points** dans la case **3 PAREILS.**

• 4 PAREILS

Quatre dés assortis, le joueur compte le total de tous les dés (4 pareils + l'autre dé).

(2 + 2 + 2 + 2 + 4) **compte 12 points** dans la case **4 PAREILS.**

• COURTE SÉQUENCE DE 4

Compte 15 points pour toute série de 4 numéros consécutifs, par exemple :

compte **15 points** dans la case **COURTE SÉQUENCE DE 4.**

• LONGUE SÉQUENCE DE 5

Compte 20 points pour toute série de 5 numéros consécutifs, par exemple :

compte 20 points dans la case **LONGUE SÉQUENCE DE 5.**

• ROULEMENT DE SURPLUS

Compte n'importe quelle combinaison de dés au moment qu'il juge le plus opportun. Le joueur compte le total de tous les dés affichés, par exemple :

(1 + 2 + 4 + 4 + 6) **compte 17 points** dans la case **ROULEMENT DE SURPLUS.**

• MAIN PLEINE

Compte 25 points pour trois pareils et une autre paire, par exemple :

compte 25 points dans la case **MAIN PLEINE.**

• YUM

Compte 30 points pour cinq chiffres pareils, par exemple :

compte 30 points dans la case **YUM.**

Bonne partie!