

Auteur : Pierre Bélisle

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 procédures et la manipulation de tableaux. Il vous est demandé de factoriser (découper en procédures) votre programme au maximum, en créant les différentes procédures 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 :


1. Obtenir des 1 (pointage = somme des 1)
2. Obtenir des 2 (pointage = somme des 2)
3. Obtenir des 3 (pointage = somme des 3)
4. Obtenir des 4 (pointage = somme des 4)
5. Obtenir des 5 (pointage = somme des 5)
6. Obtenir des 6 (pointage = somme des 6)
7. *Brelan* : Obtenir 3 dés identiques (pointage = somme des dés)
8. *Carré* : Obtenir 4 dés identiques (pointage = somme des dés)
9. *Courte séquence* : obtenir une séquence de quatre dés (15 points), par exemple : 2, 3, 4, 5
10. *Longue séquence* : obtenir une séquence de 5 dés (20 points)
11. *Roulement de surplus* : obtenir un ensemble dont la somme est maximale (pointage = somme des dés)

12. *Main pleine* : obtenir 3 dés identiques et 2 autres dés identiques (25 points)

13. *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 lors de la partie.

À la fin des 13 tours, le joueur obtient un boni 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 boni (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.

 Y U M		Partie 1		
BONNE CHANCE		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

***Contrairement à la grille plus haut, les objectifs sont présentés sur 2 colonnes : colonne de gauche pour les 6 premiers et colonne de droite pour les 7 seconds.*

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 (il apparaît pour le premier et le deuxième relancement de dés)

```

|o  o||o  o||o  o||o  o||o  o|
|o__o||o__o||__o||o__o||__o|

```

Choix possibles	

3	6

4	8

6	6

Surplus	20

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.

Lorsque le joueur a complété ses trois lancers, le programme lui demande de choisir le coup désiré parmi les coups possible (qui devraient être entre 0 et 13). Si l'utilisateur entre 0, la partie se termine à l'instant. La valeur entrée par l'utilisateur doit être validée (le nombre doit être entre 0 et 13 et la case ne doit pas être occupée) : le programme lui demande de rentrer une valeur valide si l'entrée est invalide.

```

-----
Choix possibles
-----
| 3 | 3 |
-----
| 5 | 20 |
-----
| Carre | 23 |
-----
| Surplus | 23 |
-----

```

(1 a 6) ou 10 = Brelan, 11 = Carre, 12 = Main pleine, 13 = Petite, 14 = Grosse, 15 = Surplus, 16 = Yum

Entrez le numero de case ou vous mettez les points : 15

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.

Il est important de mettre à jour les scores de cumul (sous total, bonus, total haut, total bas, total) tout au long de la partie.

4. Conception

L'organisation des données que nous vous **imposons** pour réaliser ce jeu est d'utiliser 3 tableaux qui seront des attributs (non static) de la classe YumEtud :

1. Un tableau de cinq cases, représentant les dés, qui contient des valeurs allant de 1 à 6.
2. Un tableau de 19 cases représentant la feuille de pointage. Toutes les cases de la feuille de pointage¹ 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. La première valeur n'est pas utilisée, comme elle correspond au cas où l'utilisateur quitte la partie.
3. Un tableau de 19 cases qui représente les pointages de la combinaison courante des dés pour chaque objectif. Ce dernier tableau servira à suggérer au joueur les coups possibles pour les dés joués, avec leur pointage respectif. Les valeurs de ce tableau seront actualisées à chaque nouvelle combinaison de dés (à chaque lancée).

Description par l'exemple

À titre d'exemple, voici le tableau de dés dans le cas où les valeurs des dés sont 4, 6, 4, 4 et 6 (l'indice de chaque case est indiqué au-dessus de sa valeur) :

1	2	3	4	5
4	6	4	4	6

Pour cette combinaison de dés, le tableau des pointages serait tel que ci-dessous :

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

Présumons que la feuille de pointage est (état de départ) :

¹ Sauf pour les cases représentant les totaux et les bonus, qui devront être initialisées à zéro

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

Cette dernière serait mise à jour tel que ci-dessous dans le cas où 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	<u>-1</u>	25	-1	-1	-1	-1	0	0

ou tel que ci-dessous 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	<u>-1</u>	-1	-1	-1	-1	-1	0	0

5. Procédures à réaliser

Voici certaines procédures pertinentes à la réalisation de votre programme.

5.1. Procédure pour réaliser le programme principal (main). Il s'agit de la procédure qui gère l'exécution du jeu.

5.2. Procédure pour rouler les dés choisis par le joueur :

Écrivez une procédure qui reçoit le tableau de dés et un entier (exemple : 124) qui représente les dés devant être roulés à nouveau dans le tableau. Cet entier est validé avant l'appel de la procédure.

Il faut parcourir itérativement chaque chiffre de cette variable dans une boucle et rouler la case correspondante dans le tableau de dés. La valeur de chaque dé est générée aléatoirement ($\text{Math.random()} * \text{NB_FACES} + 1$).

Considérer :

- *Un nombre entier positif modulo 10 donne son dernier chiffre (135%10 donne 5); et la partie entière du même nombre divisé par 10 enlève le dernier chiffre (135/10 donne 13.).*
- *Les cases vont de 0 à NB_DES-1, mais pas les chiffres du nombre (1 à NB_DES).*

5.3. Procédure pour gérer les trois lancers de dés

Cette procédure prend les 3 tableaux en paramètres. Cette procédure s'occupe de l'interaction avec le joueur lors des trois lancers possibles. Elle s'occupe ainsi de lancer les dés selon ses directives et de mettre à jour la grille des coups possibles. *Rappelons-nous que le joueur peut décider de garder tous les dés et de passer directement à la feuille de pointage.*

5.4. Procédures utilitaires

Pour arriver à suggérer les coups possibles, il faudra vérifier toutes les combinaisons de points possibles. Afin de permettre cet objectif, les procédures utilitaires suivantes doivent être réalisées. Des procédures additionnelles pourraient également être réalisées, à votre guise.

- **Procédure pour saisir et valider les dés choisis à rouler**
- **Procédure pour valider le choix de la case de la feuille de pointage**
- **Procédure pour générer les valeurs des dés**
- **Procédure qui retourne le nombre d'occurrences d'un dé** dans le tableau de dés (combien de 6 par exemple)
- **Procédure qui vérifie si vous avez une main pleine**
- **Procédure pour vérifier les suites**
- **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.
- **Procédure pour afficher la grille des coups possibles.** *Remarquez que seules les procédures d'affichage des dés et de la feuille de pointage vous sont fournies.*

5.5. Procédures de tests

Réaliser des procédures permettant de tester les procédures de votre application (tests unitaires). Ces procédures doivent également être commentées.

6. Réalisation et commentaires

- Mettre le nom des contributeurs pour chaque procédure dans le code.
- Il est interdit de dédier un-e étudiant-e à la **seule** écriture ou la **seule** révision des commentaires : l'étudiant qui réalise un code est celui qui commente ce même code. Les commentaires doivent ainsi être écrits 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 consulter le chargé de laboratoire aux séances ou par courriel. Cette infraction de nature académique détectée sera dénoncée au comité prévu pour ces cas.

7. Barème de correction

Exécution et tests

40%

Élément	Pondération
Déroulement du programme et interaction avec l'utilisateur correcte selon l'étape en cours	10
Fonctionnement de la relance des dés sélectionnés	5
Mise à jour à la bonne case de la grille selon la sélection	5
Calcul correct des points selon la sélection	10
Calcul correct du score final (en tenant compte du bonus)	5
Les procédures de tests valident bien le fonctionnement des procédures testées	5

Qualité de programmation

60%

Élément	Pondération
Organisation des sous-programmes. Le tout judicieusement distribué.	10
Organisation du code en sous-procédures pertinentes, il est facile de comprendre le mandat de chaque bloc de code, et comment il le réalise. Lié à l'efficacité du code	20
Lisibilité du code : les noms des éléments sont bien choisis selon les normes, l'indentation est claire, les retours de ligne et l'espacement bien utilisés	10
Qualité de la documentation : commentaires complets, faciles à comprendre, sans fautes d'orthographe. Les références sont clairement indiquées et bien citées.	20

Bon travail!