

Travail pratique #4

Pondération : 10%

Date de remise : La date de remise officielle est indiquée dans *Moodle*. **Aucun retard permis.**

Directives

- Ce travail doit être réalisé **individuellement**, soit sans l'aide d'une autre personne, sauf l'enseignant, ni de l'intelligence artificielle. La note 0, en plus d'un rapport de plagiat, sera attribuée pour un travail jugé trop semblable à celui produit par un autre étudiant ou une intelligence artificielle.
- Vous devez créer une classe nommée `Tp04` dans le *package*/répertoire `<votrenom>.tp04`. Les signatures des méthodes inscrites dans cet énoncé doivent être utilisées sans modifications dans votre programme. Il est cependant permis d'ajouter des méthodes supplémentaires au besoin. Une fois le programme complété et testé, vous devez remettre le fichier source (`.java`) dans *Moodle*, dans la section **TP#4**.
- Vous devez utiliser uniquement les notions enseignées dans le cours et respecter toutes les normes de programmation contenues dans le document intitulé *Normes de codage* se trouvant dans *Moodle*.

Barème de correction

	Maximum	Note obtenue
Code <i>Java</i> (10 méthodes)	50	
Documentation (/ * *)	10	
Exécution	40	
<i>Bonus : Utilisation de Git (At06 + Tp04)</i>	10	
Total	100	

Description des interactions

Vous devez écrire un programme qui permet de jouer à *Boutabou (Streams)* en solitaire au terminal. Le jeu consiste à placer des tuiles, soit des nombres tirés au hasard, dans une série de cases afin de créer les suites les plus longues possibles. Des points sont attribués selon la longueur des suites obtenues. Une copie des règles officielles ainsi qu'un exemple de feuille de pointage sont disponibles à la fin de ce document, soit aux pages 6 et 7 respectivement.

Dans le cadre de ce travail, nous proposons à l'utilisateur quatre variantes possibles. Il peut premièrement choisir entre une distribution des tuiles telle que décrite dans les règles, que nous appelons la distribution classique, ou des tuiles toutes différentes. La distribution classique comprend les nombres de 1 à 30, une deuxième copie des nombres de 11 à 19 et un *Joker* (0), soit quarante tuiles. Pour les tuiles toutes différentes, nous aurons simplement les nombres de 1 à 40 sans *Joker*. De plus, l'utilisateur peut choisir entre un pointage de base ou expert. La seule différence entre ces deux types de pointage est que pour les suites de 6, 12 et 17 tuiles, le pointage est moins élevé en mode expert. Par exemple, une suite de 12 tuiles vaut 35 points en mode de base et 20 points en mode expert.

En général, une exécution du programme se déroule comme suit. L'utilisateur choisit premièrement le type de pointage désiré, soit base ou expert, mais il peut aussi quitter le programme à cette étape. S'il entre un caractère invalide, le pointage de base est sélectionné. Ensuite, s'il n'a pas décidé de quitter, il choisit la distribution des tuiles, soit classique ou différentes. S'il entre un caractère invalide, la distribution classique est utilisée. La partie débute alors et consiste habituellement en la pige de vingt tuiles au hasard. Pour chacune, l'utilisateur indique dans quelle case il veut la placer avant de passer à la suivante. S'il indique une case invalide ou qui contient déjà un nombre, on lui demande une nouvelle position jusqu'à ce qu'il indique une case valide. La tuile 0 correspond au *Joker* et le programme demande à l'utilisateur par quel nombre la remplacer à la fin de la partie, soit après avoir placé la vingtième tuile. Une fois la partie terminée, on affiche les différentes suites créées avec le pointage correspondant en dessous. Enfin, on indique le score de l'utilisateur, soit le total des points de toutes les suites ainsi que le pointage maximal obtenu pour une seule suite, avant de lui permettre de faire une nouvelle partie ou de quitter. Lorsqu'il décide enfin de quitter, on lui rappelle son score maximal obtenu au cours d'une partie avant de lui souhaiter une bonne journée.

Travail à réaliser

La première étape consiste à enregistrer le code `Tp04.java`, attaché à ce document *PDF*, puis à ajouter le nom du *package* dans lequel vous l'avez déposé, normalement `<votrenom>.tp04`. Ensuite, vous devez compléter le programme principal ainsi que les neuf méthodes en suivant les indications fournies dans la suite de ce document et en traduisant les commentaires inscrits dans le fichier *Java*. Vous ne devez pas modifier les signatures des méthodes, ni le nom de la constante `NB_CASES`, que vous devez utiliser dans votre programme. Cette dernière sert à indiquer le nombre de cases à remplir lors d'une partie. Elle devrait ordinairement être initialisée à vingt pour une partie officielle, mais nous avons choisi une valeur de sept dans l'exemple d'exécution du terminal *Boutabou* à la page 5.

Lorsque vous allez tester votre programme, si vous voulez obtenir le même affichage que dans le terminal *Boutabou*, n'oubliez pas d'appeler `Outils.Aleatoire.setSeed()` une seule fois au début du programme principal, en lui fournissant la valeur inscrite en bleu dans l'en-tête du terminal. Vous devez aussi placer les tuiles dans un ordre précis dans un tableau, sinon la pige sera différente de l'exemple. L'ordre à utiliser pour la distribution classique est :

0, 1, 2, 3, ... , 28, 29, 30, 11, 12, 13, ... , 17, 18, 19

De plus, pour vous aider à tester les cinq méthodes qui ne produisent aucun affichage, nous avons inclus des tests unitaires au début du fichier *Java* fourni. Pour les exécuter, vous devez appuyer sur la flèche verte dans la marge. Pour exécuter votre programme, afin de reproduire l'exemple du terminal, utilisez la façon habituelle de le démarrer.

En résumé :

- Enregistrer le code de départ, attaché à ce *PDF*, et assurez-vous qu'il compile.
- Coder, en *Java* dans *Visual Studio Code*, le programme demandé (`.java`), soit les neuf méthodes et le programme principal. Votre code doit se trouver dans le *package* `<votrenom>.tp04` et respecter les normes de codage.
- Documenter les neuf méthodes (`/**`).
- Déposer le fichier dans *Moodle*, dans la section *TP#4*.

Programme principal : `void main(String[] args)`

Contient la boucle principale du programme qui permet au joueur de jouer plusieurs parties et de sélectionner le type de partie désiré. Tant que le joueur ne choisit pas de quitter, on lui permet de sélectionner le type de pointage et la distribution des tuiles pour sa prochaine partie. On doit alors initialiser deux tableaux selon ses choix avant d'appeler la méthode `jouerPartie()`. Cette dernière gère une partie complète et nous retourne le score final que l'on pourra comparer avec le record obtenu par le joueur. S'il a battu son record, on lui affiche un message comme celui à la ligne 32 du terminal *Boutabou*. On lui permet ensuite de faire une nouvelle partie ou de quitter. Lorsqu'il choisit de quitter, on lui rappelle son record avant de le saluer.

`int jouerPartie(Scanner cl, int[] tuiles, int[] points)`

Gère une partie complète de *Boutabou*. On commence par appeler `toursDeJeu()` pour permettre au joueur de remplir les cases du jeu à l'aide des tuiles tirées au hasard. Le tableau ainsi obtenu est passé à la méthode `changeJoker()` afin de remplacer le *Joker* par une valeur choisie par le joueur. L'appel de `calculScore()` renvoie un tableau contenant le pointage pour chaque suite trouvée. On peut alors extraire le nombre total de points et le nombre maximum de points obtenus pour une suite avant d'afficher le résultat final de la partie. Un exemple d'affichage est disponible dans le terminal *Boutabou* aux lignes 29 à 31. Le nombre total de points est retourné par cette fonction.

`int[] toursDeJeu(Scanner cl, int[] tuiles)`

Remplit toutes les cases d'un tableau à l'aide de valeurs tirées au hasard, selon les choix du joueur, afin de compléter une partie de *Boutabou*. Le nombre de cases à remplir dans une partie est spécifié par la constante `NB_CASES`. Tant que le tableau n'est pas plein, on affiche son état actuel, en remplaçant les cases vides par `__`, ainsi que les positions correspondantes de chaque élément. On tire ensuite une tuile à l'aide de la fonction `pigeTuile()`, puis on demande sa position d'insertion tant que le joueur n'en indique pas une valide. La fonction retourne le tableau ainsi rempli.

`void changeJoker(Scanner cl, int[] cases)`

Remplace le *Joker*, s'il est présent dans le tableau `cases`, reçu en paramètre, par la valeur choisie par le joueur. Cette méthode commence par rechercher le *Joker*, soit la valeur 0, dans le tableau `cases`, à l'aide de la fonction `trouve()`. S'il est présent, on affiche le tableau en substituant le *Joker* par `**` avant de demander la valeur de remplacement. Aucune validation n'est faite sur la valeur lue, elle doit simplement être écrite dans le tableau à la position du *Joker*. Dans le cas où le *Joker* est absent, la méthode retourne immédiatement, sans modifier le tableau.

`int[] calculScore(int[] cases, int[] points)`

Inscrit, dans un tableau, le nombre de points obtenus pour chaque suite croissante de nombres identifiée dans le tableau `cases`, reçu en paramètre. Pour ce faire, on débute par initialiser à zéro un tableau de scores qui aura la même taille que le tableau `cases`. On a aussi besoin d'une variable qui contient la longueur courante de la suite, que l'on initialise à un. Ensuite, pour chaque élément du tableau `cases`, on le compare à l'élément suivant. S'ils sont en ordre décroissant, c'est que l'on a atteint la fin d'une suite. Dans ce cas, on inscrit le pointage obtenu à la position correspondante dans le tableau des scores et on réinitialise la longueur courante de la suite à un. Dans le cas contraire, on ne fait qu'augmenter la longueur de la suite de un. À la fin de la fonction, on retourne le tableau des scores ainsi créé. Les tableaux `cases` et `points` ne doivent pas être modifiés par cette fonction.

Ci-dessous, nous illustrons le contenu du tableau des scores en considérant que les tableaux `points` et `cases` ont été reçus en paramètres. Nous avons identifié chaque suite en utilisant des couleurs différentes. On remarque que seul l'élément correspondant au dernier élément d'une suite contient une valeur autre que zéro. Par exemple, la suite en vert a une longueur de cinq, ce qui vaut sept points, soit le cinquième élément du tableau `points` à l'index quatre. Les deux premières suites ont une longueur de un, donc valent chacune zéro point.


	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
points	0	1	3	5	7	9	11	15	20
cases	3	2	1	6	4	5	5	6	8
Scores	0	0	0	1	0	0	0	0	7

int pigeTuile(int[] tuiles, boolean[] disponible)

Sélectionne une tuile au hasard, parmi celles disponibles, et retourne sa valeur. Le tableau `tuiles` contient les valeurs de chacune des tuiles et ne doit pas être modifié par la fonction. D'autre part, le tableau `disponible` contient des valeurs booléennes pour indiquer si une tuile est disponible (`true`), ou si elle a déjà été pignée (`false`). Il faut donc sélectionner des positions au hasard dans le tableau `disponible`, jusqu'à ce que l'on trouve une tuile disponible. `Outils.Aleatoire` doit être utilisé pour générer des valeurs aléatoires. Il faut enfin indiquer que la tuile a été pignée avant de retourner sa valeur inscrite dans la case correspondante du tableau `tuiles`.

void affiche(int[] tab, char sepC, char sepD, int val, String rem)

Affiche le contenu du tableau `tab`, reçu en paramètre, sur une seule ligne, en utilisant les séparateurs indiqués entre les éléments. Si deux éléments consécutifs sont en ordre croissant ou égaux, on utilise le caractère `sepC`, sinon on utilise `sepD`, pour les séparer. Chaque élément doit être affiché en utilisant trois colonnes et il faut mettre une espace devant le séparateur. Enfin, il est possible d'afficher toutes les occurrences d'une valeur (`val`) en utilisant une chaîne de remplacement (`rem`), reçue en paramètre. Le tableau ne doit pas être modifié par cette procédure. Ci-dessous, nous présentons trois exemples d'appels de la méthode ainsi que les résultats attendus au terminal.

 Affiche.java

```
1 public static void main(String[] args) {
2     int[] tab = new int[] {2, 3, 1, 6, 4, 5, 5, 6, 7};
3     affiche(tab, ' ', ' ', -1, "");
4     affiche(tab, '<', '>', -1, "");
5     affiche(tab, ':', '##', 6, "##");
6 }
```

Terminal : Affiche

```
1 2 3 1 6 4 5 5 6 7
2 2 < 3 > 1 < 6 > 4 < 5 < 5 < 6 < 7
3 2 : 3 : 1 : ## : 4 : 5 : 5 : ## : 7
```

int somme(int[] vecteur)

Calcule et retourne la somme des valeurs contenues dans le tableau `vecteur`, reçu en paramètre. Ce dernier ne doit pas être modifié par cette fonction.

int maximum(int[] vecteur)

Recherche la valeur maximale contenue dans le tableau `vecteur`. Cette fonction retourne la position de la première occurrence de la valeur maximale trouvée. Le tableau ne doit pas être modifié par cette fonction.

int trouve(int[] vecteur, int valeur)

Recherche la valeur reçue en paramètre dans le tableau `vecteur`. Cette fonction retourne la position de la première occurrence trouvée ou `-1` si la valeur est absente. Le tableau ne doit pas être modifié par cette fonction.

```

1 Sélectionnez le pointage de Base ou Expert ou Quitter : B
2 Distribution des chiffres, soit Classique ou Différents? C
3
4   1   2   3   4   5   6   7
5 Position d'ajout du nombre 3 > 1
6   3   -   -   -   -   -   -
7   1   2   3   4   5   6   7
8 Position d'ajout du nombre 13 > 3
9   3   -   -   13   -   -   -
10  1   2   3   4   5   6   7
11 Position d'ajout du nombre 11 > 7
12  3   -   -   13   -   -   -
13  1   2   3   4   5   6   7
14 Position d'ajout du nombre 13 > 2
15  3   -   13   -   13   -   -
16  1   2   3   4   5   6   7
17 Position d'ajout du nombre 30 > 8
18 Cette position est invalide, choisissez-en une autre > 4
19  3   -   13   -   13   -   30   -   -
20  1   2   3   4   5   6   7
21 Position d'ajout du nombre 18 > 5
22  3   -   13   -   13   -   30   -   18   -   -
23  1   2   3   4   5   6   7
24 Position d'ajout du nombre 0 > 3
25 Cette position est invalide, choisissez-en une autre > 4
26 Cette position est invalide, choisissez-en une autre > 6
27  3   -   13   -   13   -   30   -   18   -   **   -   11
28 En quelle valeur voulez-vous changer le JOKER? 3
29  3   -   13   -   13   -   30   /   18   /   3   -   11
30      5
31 Total de la partie = 6 (5)
32 Bravo, vous avez battu votre record! Nouveau score à battre : 006
33 Sélectionnez le pointage de Base ou Expert ou Quitter : e
34 Distribution des chiffres, soit Classique ou Différents? d
35
36   1   2   3   4   5   6   7
37 Position d'ajout du nombre 4 > 1
38   4   -   -   -   -   -   -
39   1   2   3   4   5   6   7
40 Position d'ajout du nombre 33 > 6
41   4   -   -   -   -   -   33   -
42   1   2   3   4   5   6   7
43 Position d'ajout du nombre 26 > 4
44   4   -   -   -   26   -   -   33   -
45   1   2   3   4   5   6   7
46 Position d'ajout du nombre 6 > 2
47   4   -   6   -   -   26   -   -   33   -
48   1   2   3   4   5   6   7
49 Position d'ajout du nombre 32 > 5
50   4   -   6   -   -   26   -   32   -   33   -
51   1   2   3   4   5   6   7
52 Position d'ajout du nombre 9 > 3
53   4   -   6   -   9   -   26   -   32   -   33   -
54   1   2   3   4   5   6   7
55 Position d'ajout du nombre 25 > 4
56 Cette position est invalide, choisissez-en une autre > 5
57 Cette position est invalide, choisissez-en une autre > 0
58 Cette position est invalide, choisissez-en une autre > 7
59   4   -   6   -   9   -   26   -   32   -   33   /   25
60      3
61 Total de la partie = 3 (3)
62 Sélectionnez le pointage de Base ou Expert ou Quitter : q
63 Votre record est : 006
64 Bonne journée

```

Jeu Boutabou (*Streams*¹) par Yoshihisa Itsubaki

MATERIEL DE JEU :

- 40 tuiles STREAMS numérotées de 1 à 30, les nombres 11 à 19 étant en double, et une tuile JOKER
- 1 bloc de feuilles de jeu
- 1 sac en tissu

BUT DU JEU :

Dans STREAMS vous allez devoir réaliser les plus longues suites de nombres possibles, plus la suite est longue plus vous marquez de points. Le gagnant est le joueur avec le plus grand nombre de points à la fin de la partie.

PREPARATION DU JEU :

1. Distribuez une feuille de jeu à chaque joueur.
2. Placez toutes les tuiles STREAMS dans le sac et mélangez les.
3. Choisissez un joueur qui sera responsable de la pioche des tuiles STREAMS.
4. La partie va se dérouler en vingt tours de jeu.

DEROULEMENT D'UN TOUR DE JEU :

5. Le joueur responsable de la pioche tire au hasard une tuile STREAMS dans le sac, il annonce à voix haute le nombre indiqué dessus ou s'il s'agit de la tuile JOKER, puis place celle-ci face visible sur la table.
6. Tous les joueurs inscrivent le nombre annoncé ou le symbole JOKER sur leurs feuilles de jeu. Ils sont libres de placer le nombre ou le JOKER où ils veulent dans une case libre de la grille. Ils essaient ainsi de constituer progressivement la ou les suites de nombres les plus longues possibles.
7. Dès que tous les joueurs ont inscrit le nombre sur leurs feuilles de jeu, un nouveau tour commence (5.).

FIN DE PARTIE :

8. A la fin du vingtième tour, lorsque la grille de chaque joueur est remplie, la partie s'arrête. Les joueurs comptent leurs scores en se reportant au tableau des scores inclus sur leurs feuilles de jeu.
9. La partie est remportée par le(s) joueur(s) avec le plus haut score.

Dans STREAMS, qu'est-ce qu'une suite?

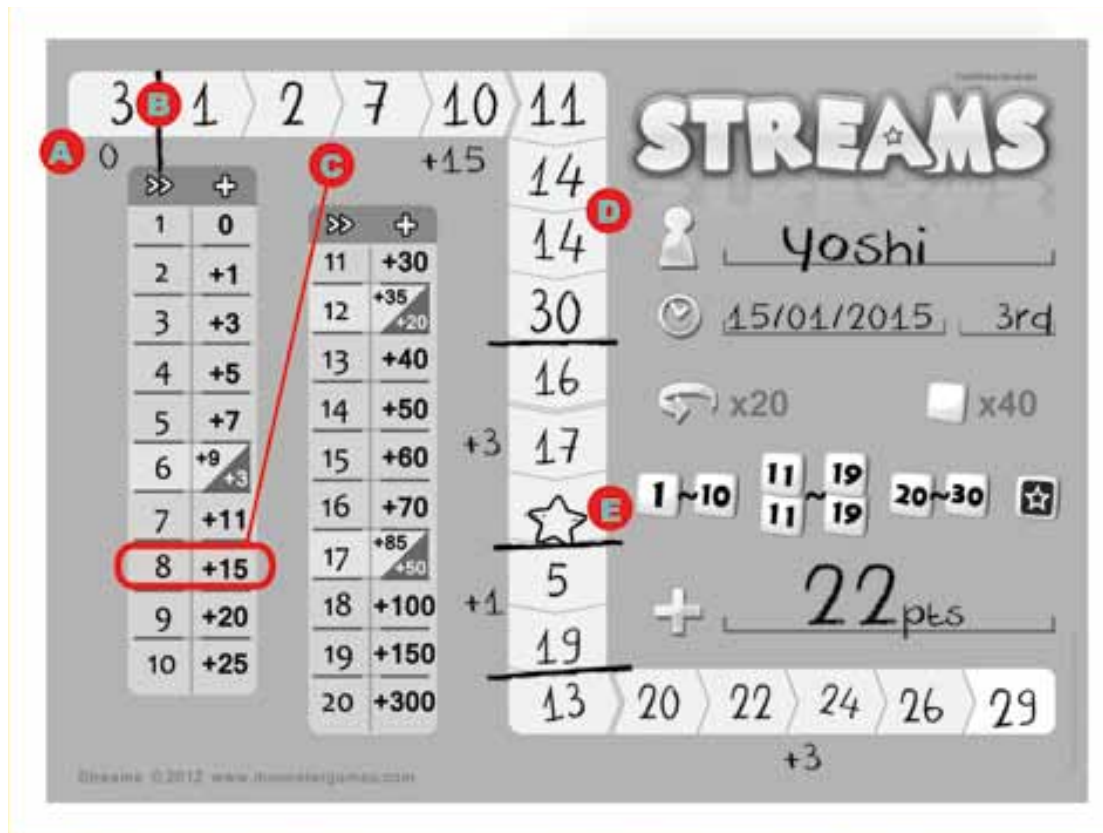
Une suite est une séquence de nombres allant du plus faible au plus élevé, ceci incluant :

- Les nombres qui ne sont pas consécutifs, mais dont les valeurs vont dans l'ordre croissant.
 - Les nombres identiques placés à la suite les uns des autres.
 - Le JOKER, qui doit être comptabilisé dans une seule suite et qui doit respecter les deux critères précédents.
- Par exemple, 5–12–X–14–17–21–21–27 est une suite de huit nombres valide.

DECOMPTE DES POINTS :

Une fois la partie terminée, les joueurs doivent identifier chaque suite figurant sur leur feuille. Chaque suite rapporte un nombre de points variant selon sa longueur. Pour les suites de 6, 12 et 17 nombres, prendre la valeur de gauche dans le tableau de référence pour le mode de base, ou celle de droite pour le mode expert. Le score final d'un joueur est le total de points rapporté par chacune des suites qu'il a pu constituer.

1. <https://boardgamegeek.com/boardgame/103814/streams>



- Exemple du décompte des points en mode expert — A. Un nombre seul ne rapporte pas de points -
 B. Séparez vos suites par un trait - C. Relevez le nombre de points correspondant à votre suite - D. Deux nombres identiques comptent chacun dans la longueur de la suite - E. Le JOKER est comptabilisé dans une seule suite