Sujet et corrigé CF1 2008

Imaginez la palette d'un peintre : elle permet au peintre de composer ses couleurs à partir de celles sur la palette. Il mélange par exemple la moitié (50%) de bleu et la moitié de rouge (50%) et obtient du violet. Nous modélisons, dans ce sujet, la palette d'un peintre composée de 12 couleurs (vert amande, vert, vert émeraude, bleu, indigo, violet, pourpre, rouge, sanguine, orange, soleil, jaune) et les proportions du mélange. Dans la suite du sujet, une couleur sur la palette est appelée une « composante » et le mélange obtenu est appelé « couleur »

Pour une couleur, le nom des composantes n'est en fait pas stocké. Seuls les pourcentages des composantes utilisées sont stockés dans un tableau de 12 réels. L'ordre des composantes est connu : la case 0 correspond au vert amande, la case 1 correspond au vert, la case 2 correspond au vert émeraude, la case 3 correspond au bleu, la case 4 correspond au indigo, la case 5 correspond au violet, la case 6 correspond au pourpre, la case 7 correspond au rouge, la case 8 correspond au sanguine, la case 9 correspond au orange, la case 10 correspond au soleil, la case 11 correspond au jaune.

Ci-dessous un exemple : si le peintre a mélangé, en proportions égales, du vert amande, du vert, du bleu et du jaune, le tableau contiendra les valeurs suivantes :

indice	0	1	2	3	4	5	6	7	8	9	10	11
composante associée	vert amande	vert	vert émeraude	bleu	indigo	violet	pourpre	rouge	sanguine	orange	soleil	jaune
contenu du tableau	25.0	25.0	0	25.0	0	0	0	0	0	0	0	25.0

La somme des valeurs contenues dans le tableau doit être exactement égale à 100.

La méthode nomCouleur(...) permettant d'associer le libellé d'une composante avec son indice, devra être appelée par les méthodes des questions 2 et 3. Elle reçoit en paramètre un entier compris entre 0 et 11, et renvoie en résultat une chaîne de caractères. Remarque : Il n'est pas nécessaire de recopier cette méthode sur votre copie.

```
static String nomCouleur (int indice) {
        switch (indice) {
                case 0  : return "vert amande";
                case 1  : return "vert";
                case 2  : return "vert emeraude";
                case 3  : return "bleu";
                case 4  : return "indigo";
                case 5  : return "violet";
                case 6  : return "pourpre";
                case 7  : return "rouge";
                case 8  : return "sanguine";
                case 9  : return "orange";
                case 10 : return "soleil";
                case 11 : return "jaune";
                default : return "";
        }
}
```

Questions

Question 1 : définition des structures de données

Définir un tableau de 12 réels nommé `couleur`. Pour cela vous devez écrire sur une feuille séparée une méthode « main() ». Vous écrirez la création du tableau dans cette méthode.

Pour chacune des questions suivantes, écrire la méthode demandée sur votre copie puis compléter le programme principal (main) permettant d'appeler cette méthode sur la feuille séparée.

Question 2 : initialiser les composantes d'une couleur

Écrire une méthode permettant de saisir (lire) les composantes d'une couleur. Les composantes doivent être lues une par une. Pour chaque composante, la question posée doit donner le nom de la composante (e.g., «vert amande»), et non son indice (e.g., 0). Vous devez pour cela faire appel à la méthode nomCouleur(...).

Le total des pourcentages est calculé au fur et à mesure de la saisie, et les règles suivantes doivent être respectées :

- Dès que le total atteint 100, la saisie s'arrête et la suite du tableau n'est pas saisie.
- Si le total dépasse 100 (la dernière valeur saisie est trop élevée), cette dernière valeur est diminuée afin que le total fasse bien 100. L'utilisateur est informé par l'affichage d'un message de la modification de cette valeur.
- Si, lorsque toutes les valeurs des composantes ont été saisies, le total reste inférieur à 100, la dernière valeur est augmentée afin que le total fasse bien 100. L'utilisateur est informé par l'affichage d'un message de la modification de cette valeur.

Voici 3 scénarios de saisie de composantes :

Scénario 1 :

```
Saisissez le taux des couleurs (vert amande, vert, vert émeraude, bleu, indigo, violet,
pourpre, rouge, sanguine, orange, soleil, jaune), le total doit faire 100

Entrez le taux de la couleur vert amande : 50

Entrez le taux de la couleur vert : 50

Taux de couleur total = 100.0

100% atteint. La suite du tableau est laissée à 0%
```

Scénario 2 :

```
Saisissez le taux des couleurs (vert amande, vert, vert émeraude, bleu, indigo, violet,
pourpre, rouge, sanguine, orange, soleil, jaune), le total doit faire 100
```

```
Entrez le taux de la couleur vert amande : 50

Entrez le taux de la couleur vert : 55

Taux de couleur total = 105.0

Vous aviez dépassé les 100%.  La dernière valeur a été recalculée à 50.0

100% atteint. La suite du tableau est laissée à 0%
```

Scénario 3 :

```
Saisissez le taux des couleurs (vert amande, vert, vert émeraude, bleu, indigo, violet,
pourpre, rouge, sanguine, orange, soleil, jaune), le total doit faire 100

Entrez le taux de la couleur vert amande : 50

Entrez le taux de la couleur vert : 0

Entrez le taux de la couleur vert emeraude : 0

Entrez le taux de la couleur bleu : 0

Entrez le taux de la couleur indigo : 0

Entrez le taux de la couleur violet : 0

Entrez le taux de la couleur pourpre : 0

Entrez le taux de la couleur rouge : 0

Entrez le taux de la couleur sanguine : 0

Entrez le taux de la couleur orange : 0

Entrez le taux de la couleur soleil : 0

Entrez le taux de la couleur jaune : 0

Taux de couleur total = 50.0

Vous n'aviez pas atteint 100%. La dernière valeur a été recalculée à 50.0
```

2.1 Écrire la méthode permettant de saisir la couleur. Le prototype (l'entête) de la méthode est le suivant :

```
static void saisirCouleur(double [] t)
```

2.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour saisir le tableau `couleur` défini à la question 1.

Question 3 : afficher les composantes d'une couleur

L'affichage de la composition d'une couleur doit être précédé du message « Composition de la couleur : » suivi directement (sur la même ligne) des pourcentages des composantes. Seules les composantes dont le pourcentage n'est pas nul sont affichées. L'affichage indique le nom de la composante (faire un appel à la méthode nomCouleur(...)) , suivi entre parenthèses de son pourcentage.

L'affichage doit se terminer par un retour à la ligne.

Voici un exemple d'affichage de composantes :

```
Composition de la couleur : vert amande (25.0%) vert (25.0%) bleu
(25.0%) jaune (25.0%)
```

3.1 Écrire une méthode permettant d'afficher (écrire) les composantes de la couleur. Le prototype (l'entête) de la méthode est le suivant :

```
static void afficherCouleur(double[] tab)
```

3.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour afficher le tableau `couleur` défini à la question 1.

Question 4 : Couleur froide ou couleur chaude

Il est possible de classifier les couleurs en couleurs froides et couleurs chaudes. Dans notre exemple, les composantes froides sont le vert amande, le vert, le vert émeraude, le bleu, l'indigo, et le violet (soit la première moitié du tableau); les composantes chaudes sont le pourpre, le rouge, le sanguine, l'orange, le soleil et le jaune (soit la deuxième moitié du tableau).

Une couleur froide contient majoritairement des composantes froides (somme des pourcentages de composantes froides est supérieure (ou égale) à la somme des composantes chaudes). Respectivement, une couleur chaude contient majoritairement des composantes chaudes.

Dans l'exemple donné en présentation du sujet, la couleur est froide (25% de vert émeraude + 25% de vert + 25 % de bleu de composantes froides, 25% de jaune soit de 25% composantes chaudes).

Voici un exemple d'affichage :

```
Taux de composantes froides : 75.0, taux de composantes chaudes : 25.0
C'est une couleur froide
```

4.1 Écrire la méthode permettant d'évaluer et d'indiquer si une couleur est chaude ou froide. Le prototype (l'entête) de la méthode est le suivant :

```
static void estChaudFroid(double[] tab)
```

4.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour afficher si le tableau `couleur` correspond à une couleur froide ou chaude.

Question 5 : Composante dominante

Il est possible de calculer la « composante » dominante. Pour cela, un « barycentre » est calculé : il s'agit de faire la somme des pourcentages des composantes pondérés par le poids des composantes, correspondant à l'indice + 1. Le total obtenu est ensuite divisé par 100 puis arrondi à l'entier le plus proche. Il faut enfin soustraire 1 au résultat pour obtenir l'indice de la case (compté à partir de 0 et non de 1). Sur l'exemple de la page 2, la formule à programmer doit permettre de faire le calcul suivant :

1 * 25.0 + 2 * 25.0 + 3 * 0 + 4*25.0 + 5*0 + 6*0 + 7*0 + 8*0 + 9*0 + 10*0 + 12 * 25.0 = 475.0

475.0/ 100 = 4.75 arrondi à 5

5 correspond à la couleur d'indice 5 - 1= 4, à savoir « indigo ».

La fonction `static long` Math.round(`double a`) devra être utilisée pour arrondir et convertir le résultat en un nombre entier. On peut par exemple écrire :

```
int indice;
indice = (int) Math.round(3.14);
```

Voici un exemple d'affichage :

```
La composante dominante est : indigo.
```

5.1 Écrire la méthode permettant d'évaluer et d'afficher la composante dominante. Le prototype (l'entête) de la méthode est le suivant :

```
static void calculerDominante(double[] t)
```

5.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour donner la composante dominante du tableau `couleur`.

Question 6 : basique exclusif

Les composantes basiques sont celles situées dans les cases d'indice impair (vert, bleu, violet, rouge, orange, jaune). Une couleur est exclusivement composée de composantes basiques si elle ne contient pas de composante dérivée soit ni vert amande, ni vert émeraude, ni indigo, ni pourpre, ni sanguine, ni soleil (indices pairs).

Voici un exemple d'affichage pour une couleur qui n'est pas « basique exclusif » :

```
La couleur n'est pas exprimée exclusivement en composantes basiques.
```

6.1 Écrire la méthode permettant d'évaluer si une couleur contient exclusivement des composantes basiques. Cette méthode retourne `true` si c'est le cas, `false` sinon. Le prototype (l'entête) de la méthode est le suivant:

```
static boolean estBasiqueExclusif(double[] t)
```

6.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir pour savoir si le tableau `couleur` contient exclusivement des composantes basiques et afficher un message approprié.

Question 7 : Transformer en basique exclusif

Il est possible de transformer une couleur qui contient des composantes dérivées pour qu'elle utilise uniquement des composantes basiques et devienne ainsi « basique exclusif ». Pour cela, chaque composante dérivée est répartie pour moitié sur ses deux composantes basiques voisines :

- le pourcentage de vert amande est réparti pour moitié sur le vert (case suivante), et pour moitié sur le jaune (case « précédente », si on imagine que le tableau est cyclique) ;
- le pourcentage de vert émeraude est réparti pour moitié sur le bleu (case suivante), et pour moitié sur le vert (case précédente) ;
- le pourcentage de indigo est réparti pour moitié sur le violet (case suivante), et pour moitié sur le bleu (case précédente) ;
- le pourcentage de pourpre est réparti pour moitié sur le rouge (case suivante), et pour moitié sur le violet (case précédente) ;
- le pourcentage de sanguine est réparti pour moitié sur le orange (case suivante), et pour moitié sur le rouge (case précédente) ;
- le pourcentage de soleil est réparti pour moitié sur le jaune (case suivante), et pour moitié sur le orange (case précédente).

NB: Si « i » est l'indice de la case contenant une composante dérivée :

- (i+1) est l'indice de la case suivante (contenant une composante basique) ;
- (i+11)%12 est l'indice de la case précédente (contenant une composante basique).

7.1 Écrire la méthode permettant de tranformer une couleur en « basique exclusif ».

7.2 Compléter le programme principal (main) afin d'appeler la méthode que vous venez de définir uniquement si la couleur n'était pas basique exclusif. Afficher ensuite la nouvelle composition de la couleur après transformation (en appelant la méthode d'affichage de la question 4).

Exemple d'affichage :

```
Composition après transformation

Composition de la couleur : vert (37.5%) bleu (25.0%) jaune
(37.5%)
```

Corrigé CF1 2008

```
class Couleurs {
        public static void main (String arg[]){

                double [] couleur = new double [12] ;
                // une couleur est composée de 12 couleurs
                // le poids (%) de chaque couleur est stocké dans un tableau de réels
```

```
                // saisir la couleur
                saisirCouleur(couleur);

                // afficher la couleur - associer un indice a un nom de couleur
                afficherCouleur(couleur);

                // couleurs froides = vert, bleu, violet = 1ere moitié de tableau
                // couleurs chaudes = rouge, orange, jaune = 2eme moitié de tableau
                // couleur chaude ou froide
                estChaudFroid(couleur);

                // calculer couleur dominante == barycentre
                calculerDominante(couleur) ;

                // primaire = bleu, rouge, jaune. Case d'indice impair
                // teste si la couleur est composée que de couleurs primaires
                boolean primaire;
                primaire = estBasiqueExclusif(couleur);

                // transformer en primaire exclusif si pas déjà en primaire exclusif

                if (primaire == false){
                        System.out.println("La couleur n'est pas exprimée
exclusivement en couleurs primaires. Elle va être transformée");
                        transformerBasiqueExclusif(couleur);
                        System.out.println("Composition après transformation");
                        afficherCouleur(couleur);
                }else{
                        System.out.println("La couleur est exprimée en couleurs
primaires exclusivement");
                }
        }

        static String nomCouleur (int indice) {
                switch (indice) {
                        case 0  : return "vert amande";
                        case 1  : return "vert";
                        case 2  : return "vert emeraude";
                        case 3  : return "bleu";
                        case 4  : return "indigo";
                        case 5  : return "violet";
                        case 6  : return "pourpre";
                        case 7  : return "rouge";
                        case 8  : return "sanguine";
                        case 9  : return "orange";
                        case 10 : return "soleil";
                        case 11 : return "jaune";
```

```
                default : return "";
            }

        }

        static void saisirCouleur(double [] t){ // Question 2
            // saisir tant que total < 100
            int i,j ;
            double total = 0;
            System.out.println("Saisissez le taux des couleurs (vert amande, vert, vert émeraude, bleu, indigo, violet, pourpre, rouge, sanguine, orange, soleil, jaune), le total doit faire 100");
            total = 0 ;
            i = 0;
            while ((i < t.length) && (total < 100)) {
                System.out.print("Entrez le taux de la couleur "+nomCouleur(i)+" : ") ;
                t[i] = Lire.d();
                total = total + t[i] ;
                i++;
            }

            System.out.println("Taux de couleur total = "+ total);

            if (total != 100){
                //if (total > 100) {
                if (total > 100) {
                    System.out.print("Vous avez dépassé les 100%. ");
                }else {
                    System.out.print("Vous n'avez pas atteint 100%. ");

                }
                t[i-1] = t[i-1] - (total -100);
                System.out.println(" La dernière valeur a été recalculée à "+t[i-1]);
                total = 100 ;

            }
            if (i< t.length) {
                    System.out.println("100% atteint. La suite du tableau est laissée à 0%");
            }
        }

        static void afficherCouleur(double [] tab){// Question 3
            int i ;
            System.out.print("Composition de la couleur : ");
```

```
		for (i=0; i < tab.length; i++) {

			if (tab[i] != 0) {
				System.out.print(nomCouleur(i)+"(" + tab[i]+ ")");
			}
		}
		System.out.println();
	}

	static void estChaudFroid(double[] tab){ // Question 4
		int i;
		double totalChaud = 0, totalFroid = 0;

		for (i=0; i < tab.length ; i++){
			if (i<=5) {
				totalFroid = totalFroid + tab[i] ;
			}else{
				totalChaud = totalChaud + tab[i];
			}
		}

		System.out.println("Taux de composantes froides : "+totalFroid+", taux de
composantes chaudes : "+totalChaud);

		if (totalFroid > totalChaud) {
			System.out.println("C'est une couleur froide");
		}else{
			System.out.println("C'est une couleur chaude");
		}
	}

	static void calculerDominante(double[] t){  // Question 5
		int i ;
		double barycentre = 0;
		for (i=0; i< t.length; i++){
			barycentre = barycentre + t[i] * (i+1);
		}
		barycentre = barycentre / 100 ;
		int indiceCouleurDominante = (int) Math.round(barycentre) -1 ;

		System.out.println("La couleur dominante est le "+
nomCouleur(indiceCouleurDominante));
	}

	static boolean estBasiqueExclusif(double[] t){ // Question 6
		int i ;
```

```
		double totalPrimaire = 0;
		for (i=1; i < t.length ; i=i+2){
			totalPrimaire=totalPrimaire + t[i];
		}
		if (totalPrimaire == 100){
			return true;
		}else{
			return false;
		}
	}

	static void transformerBasiqueExclusif(double[] t){  // Question 7
		int i ;
		// reverser les couleurs aux indices pairs (secondaires) sur la case après
(+1) et la case avant (en fait (case +5)%6)
		for (i=0; i< t.length; i=i+2){
			// moitié du taux de la couleur secondaire dans la case suivante
			t[i+1] = t[i+1] + t[i]/2;
			t[(i+5)%6] = t[(i+5)%6] + t[i]/2 ;
			t[i] = 0 ;
		}
	}
}
```

Sujet et corrigé CF1 2009

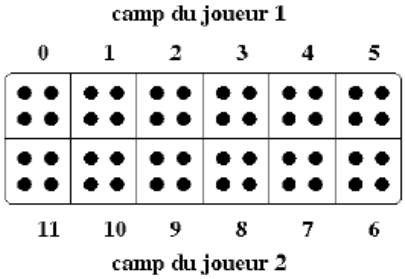
Programmation du jeu d'awélé

L'« awélé » ou « awalé » est un jeu de société d'origine africaine, considéré comme le jeu d'échecs africain.

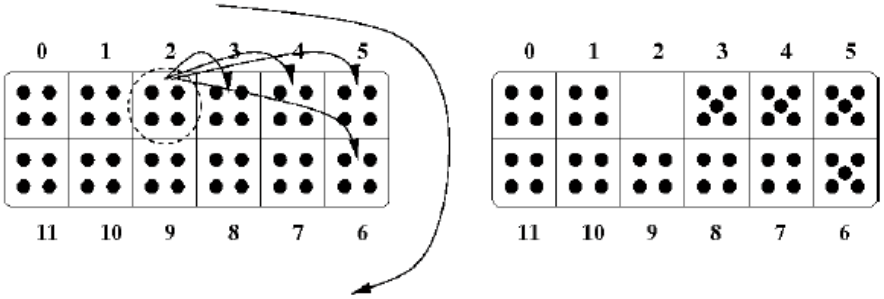
Nous vous proposons de réaliser un programme Java, permettant à deux joueurs humains de jouer à une version simplifiée de l'awélé, le programme servant d'arbitre et affichant les différentes étapes du jeu.

Règles du jeu

- Le jeu est composé d'une planche de bois creusée et de 48 graines. Les trous sont répartis en deux rangées de 6 trous chacune.
- Au départ, on répartit les graines dans les trous à raison de 4 graines par trou.
- Les joueurs sont l'un en face de l'autre, avec une rangée devant chaque joueur. Cette rangée sera son camp. Le joueur 1 commence la partie.



- Un tour se joue de la façon suivante :
  - Le premier joueur prend toutes les graines d'un des trous de son camp puis il les égraine dans toutes les cases qui suivent sur sa rangée puis sur celle de son adversaire dans le sens des aiguilles d'une montre (une graine dans chaque trou **après** celui où il a récupéré les graines). Dans l'exemple suivant, le joueur 1 joue la case n°2 et répartit ses 4 graines dans les trous suivants (3,4,5 et 6).



  - Si sa dernière graine tombe dans un trou du camp adverse et qu'il y a maintenant 2 ou 3 graines dans ce trou, le joueur récupère ces 2 ou 3 graines et les met de côté. Ensuite il regarde la case précédente : si elle est dans le camp adverse et contient deux ou trois graines, il récupère ces graines, et ainsi de suite jusqu'à ce qu'il arrive à son camp ou jusqu'à ce qu'il y ait un nombre de graines différent de 2 ou 3.
- Le but du jeu est de récupérer un maximum de graines.
- La partie s'arrête quand l'un des joueurs a capturé au moins 25 graines (le gagnant est alors celui qui aura le plus de graines) ou si un des joueurs est bloqué (dans ce cas, son adversaire aura gagné).

Questions

Question 1

Pour représenter le plateau de jeu, nous vous proposons d'utiliser un tableau d'entiers de 12 cases appelé **`plancheDeJeu`**, dont chaque case représente l'état d'un trou de l'awelé. En début de partie, chaque case contient 4 graines. De plus, les scores des deux joueurs seront représentés par deux variables entières, **`scoreJoueur1`** et **`scoreJoueur2`**, valant 0 au début de la partie. Enfin, pour savoir à qui est le tour de jouer, une variable entière **`joueurCourant`** représente le numéro du joueur (1 ou 2); elle vaut 1 au début de la partie.

Donnez les instructions permettant de définir le tableau **`plancheDeJeu`** et les autres variables, et de les initialiser.

Question 2

La méthode **`prochainJoueur`** permet de changer de joueur à chaque tour de jeu en l'utilisant de la manière suivante : `prochainJoueur(1)` renvoie la valeur 2 et `prochainJoueur(2)` renvoie la valeur 1.

```
static int prochainJoueur(int joueurCourant) { //à compléter }
```

Complétez le corps de la méthode.

Question 3

La méthode **`affiche`** permet d'afficher l'état du jeu. Elle prend en paramètres un tableau de jeu d'awélé et des variables de score telles que définies à la question 1.

```
static void affiche(int[] jeu, int scoreJ1, int scoreJ2) { //à compléter }
```

L'affichage doit ressembler à l'exemple ci-dessous (en cours de partie) :

```
|0|0|0|0|1|12|    Joueur 1, score : 5
=============
|0|2|9|1|10|1|    Joueur 2, score : 7
```

Complétez le corps de la méthode.

Question 4

La méthode **`choisirUnCoup`** permet d'interagir avec le joueur dont le numéro est passé en paramètre, en lui demandant de choisir un numéro de case pour jouer. Le coup est possible d'une part si le numéro de case est compris entre 0 et 5 pour le joueur 1 et entre 6 et 11 pour le joueur 2, et d'autre part si le trou correspondant n'est pas vide. Si le numéro donné par le joueur correspond à un coup impossible, alors la méthode affiche un message d'erreur et redemande à nouveau un numéro de case au joueur. Lorsque le coup est possible, la méthode renvoie la valeur entrée par le joueur.

```
static int choisirUnCoup(int[] jeu, int numJoueur) { //à compléter }
```

```
|1|0|0|0|0|13|
============
|1|0|9|1|10|1|
Voilà l'état du tableau de jeu
après l'appel de cette méthode :
|2|1|1|1|1|1|
============
|2|1|10|2|11|3|
```

Une autre situation dans laquelle le joueur 2 joue sa case numéro 7 qui contient 8 graines. La méthode renvoie la valeur 3 (indice du tableau de la dernière graine distribuée).

```
|1|2|1|8|0|1|
============
|0|4|8|1|8|8|
Voilà l'état du tableau de jeu
après l'appel de cette méthode :
|2|3|2|9|0|1|
============
|1|5|9|2|0|8|
```

Complétez le corps de la méthode.

Question 8

Après avoir joué un coup, il faut vérifier si le joueur peut prendre des graines. C'est le cas si le trou dans lequel il a déposé la dernière graine (paramètre **`numCasePrise`**) est dans le camp de son adversaire et contient 2 ou 3 graines. La méthode **`peutPrendreGraines`** renvoie vrai si c'est le cas, faux sinon.

```
static boolean peutPrendreGraines(int[] jeu, int numJoueur, int numCasePrise) {
      //à compléter
}
```

Complétez le corps de la méthode.

Question 9

Si le joueur peut prendre des graines, alors on appellera la méthode **`prendreGraines`**, qui permet de récupérer les graines comme indiqué dans la règle du jeu : tant qu'il y a 2 ou 3 graines on prend les graines et on recommence le processus dans la case précédente, jusqu'à sortir du camp de l'adversaire ou trouver une case qui contient plus de 3 graines ou moins de 2. Les paramètres sont les mêmes que pour la question 8. La méthode renvoie le nombre de graines capturées.

```
static int prendreGraines(int[] jeu, int numJoueur, int numCasePrise) {
      //à compléter
}
```

Voici l'exemple d'une situation de prise de graines :

```
Suite au coup du joueur 2 donné en
exemple à la question 7, le tableau
de jeu est dans l'état suivant :
|2|3|2|8|0|1|
=============
|1|5|9|2|9|0|
La variable numCasePrise vaut 2. Après
l'appel de la méthode prendreGraines,
le tableau est dans l'état suivant:
|0|0|0|8|0|1|
=============
|1|5|9|2|9|0|
La méthode renvoie la valeur 7, égale
au nombre de graines capturées.
```

Complétez le corps de la méthode.

Question 10

Le déroulement du jeu peut être programmé dans la méthode **main**. Celle-ci contient le code d'in tialisation demandé à la question 1. Ensuite, vous devez en utilisant les méthodes définies dans l questions précédentes (que vous pouvez utiliser même si vous ne les avez pas terminées) :

- afficher l'état du jeu,
- choisir un coup,
- jouer un coup,
- vérifier si la position atteinte permet de prendre des graines, si c'est le cas ajouter au sco du joueur courant le nombre de graines prises puis changer de joueur.

Tout cela se répète tant qu'il n'y a pas de gagnant. Lorsqu'il y a un gagnant, vous devez afficher u dernière fois le contenu de la planche de jeu ainsi qu'un message désignant le gagnant.

```
public static void main(String[] args) {
      //à compléter
}
```

Complétez le corps de la méthode.

Corrigé CF1 2009

```
public class CF1_Awele {

    /** Question 2 */
    static int prochainJoueur(int joueurCourant) {
        if (joueurCourant == 1)
            return 2;
        return 1;
    }

    /** Question 3 */
    static void affiche(int[] jeu, int scoreJ1, int scoreJ2) {
        for (int i = 0; i < 6; i++)
            // affichage des cases 0 à 5
            System.out.print("|" + jeu[i]);
        System.out.println("|\tJoueur 1, score : " + scoreJ1);
        System.out.println("=============");
        for (int i = 11; i > 5; i--)
            // affichage des cases 11 à 6
            System.out.print("|" + jeu[i]);
        System.out.println("|\tJoueur 2, score : " + scoreJ2);
    }

    /** Question 4 */
    static int choisirUnCoup(int[] jeu, int numJoueur) {
        int coup=-1;    //valeur initiale quelconque
        boolean coupValide = false;
        while (!coupValide) { // tant que le coup est invalide on boucle
            System.out.println("Joueur " + numJoueur
                    + ", quel est votre coup ?");
            coup = Lire.i(); // lire le coup du joueur
            if (numJoueur == 1 && coup >= 0 && coup <= 5 && jeu[coup] != 0)
                coupValide = true;
            if (numJoueur == 2 && coup >= 6 && coup <= 11 && jeu[coup] != 0)
                coupValide = true;
            if (!coupValide) // si erreur, on affiche un message
                System.out.println("Coup impossible ! Recommencez.");
        }
        return coup;
    }

    /** Question 5 */
    static boolean estBloque(int[] jeu, int numJoueur) {
        int somme = 0;
        int debut, fin;
```

```
	if (numJoueur == 1) { // calcul des indices de début et fin
		debut = 0;
		fin = 5;
	} else {
		debut = 6;
		fin = 11;
	}
	for (int i = debut; i <= fin; i++)
		// calcul de la somme des graines
		somme += jeu[i];
	return (somme == 0);
}

/** Question 6 */
static int gagnant(int[] jeu, int scoreJ1, int scoreJ2) {
	if (scoreJ1 > 25 || estBloque(jeu, 2)) // le joueur 1 gagne
		return 1;
	if (scoreJ2 > 25 || estBloque(jeu, 1)) // le joueur 2 gagne
		return 2;
	return 0; // si personne ne gagne, on retourne 0
}

/** Question 7 */
static int jouerUnCoup(int[] jeu, int numCase) {
	int graines = jeu[numCase]; // on récupère les graines de la case jouée
	jeu[numCase] = 0; // on vide la case
	// et on distribue
	int i = 1;
	int indiceDepot=0;
	while (graines > 0) {
		indiceDepot = numCase + i;
		jeu[indiceDepot]++; // on dépose une graine
		graines--;
		i++; // case suivante
		// quand on arrive à la fin du tableau on revient au début
		if (numCase + i == 12)
			i = -numCase;
	}
	return indiceDepot;
}

/** Question 8 */
static boolean peutPrendreGraines(int[] jeu, int numJoueur, int numCasePrise) {
	if (numJoueur == 1) {
		if ((jeu[numCasePrise] == 2 || jeu[numCasePrise] == 3)
				&& (numCasePrise >= 6))
			return true; // le joueur 1 peut prendre
```

```
        } else {
            if ((jeu[numCasePrise] == 2 || jeu[numCasePrise] == 3)
                    && (numCasePrise <= 5))
                return true; // le joueur 2 peut prendre
        }
        return false;
    }

    /** Question 9 */
    static int prendreGraines(int[] jeu, int numJoueur, int numCasePrise) {
        int nbGrainesPrises = 0; // compteur de graines prises
        int fin;
        // calcul de l'indice de fin de tableau pour prendre
        if (numJoueur == 1) {
            fin = 6;
        } else {
            fin = 0;
        }
        // prise dans chaque case dans le sens inverse du jeu
        // tant qu'on peut prendre
        while ((jeu[numCasePrise] == 2 || jeu[numCasePrise] == 3)
                && (numCasePrise >= fin)) {
            // somme des graines prises
            nbGrainesPrises = nbGrainesPrises + jeu[numCasePrise];
            jeu[numCasePrise] = 0; // on vide la case
            numCasePrise--;
        }
        return nbGrainesPrises;
    }

    public static void main(String[] args) {
        /** Question 1 */
        int[] plancheDeJeu = new int[12];
        int scoreJoueur1 = 0;
        int scoreJoueur2 = 0;
        int joueurCourant = 1; // le joueur 1 commence la partie
        // toutes les cases de la planches contiennent 4 graines
        for (int i = 0; i < plancheDeJeu.length; i++)
            plancheDeJeu[i] = 4;

        /** Question 10 */
        int coup, casePrise, grainesPrises;
        do {
            affiche(plancheDeJeu, scoreJoueur1, scoreJoueur2);
            coup = choisirUnCoup(plancheDeJeu, joueurCourant);
            casePrise = jouerUnCoup(plancheDeJeu, coup);
```

```
                //peut on prendre des graines ? si oui on les prend et on
                //modifie le score du joueur courant
                if (peutPrendreGraines(plancheDeJeu, joueurCourant, casePrise)) {
                    grainesPrises = prendreGraines(plancheDeJeu, joueurCourant, casePrise);
                    if (joueurCourant == 1)
                        scoreJoueur1+=grainesPrises;
                    else
                        scoreJoueur2+=grainesPrises;
                }
                //changement de joueur
                joueurCourant = prochainJoueur(joueurCourant);

            //on répete tant qu'il n'y a pas de gagnant
            } while (gagnant(plancheDeJeu, scoreJoueur1, scoreJoueur2) == 0);

            affiche(plancheDeJeu, scoreJoueur1, scoreJoueur2);

            System.out.println("Le gagnant est le joueur "
                    + gagnant(plancheDeJeu, scoreJoueur1, scoreJoueur2));
        }

}
```

Sujet et corrigé CF2 2009

Programmation du jeu de Nim

Le jeu de Nim est un jeu ancien dont il existe de nombreuses variantes, par exemple le jeu de Marienbad ou l'un des duels du maître des jeux de Fort Boyard.

Nous vous proposons de réaliser un programme Java, permettant à deux joueurs humains de jouer au jeu de Nim, le programme servant d'arbitre et affichant les différentes étapes du jeu.

Illustration 1: Une variante du jeu de Nim

Règles du jeu

- Dans la version que nous proposons, deux joueurs humains s'affrontent tour à tour.
- Le plateau de jeu est constitué initialement de 30 bâtonnets posés sur une seule ligne.
- Chaque joueur décide lors de son tour de prendre 1,2 ou 3 bâtonnets, n'importe où sur la ligne.
- Celui des deux joueurs qui prend le dernier bâtonnet perd la partie.

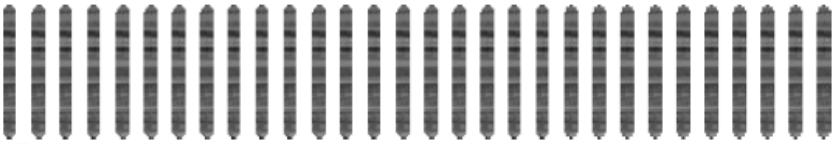


Illustration 2: Jeu de Nim à 30 bâtonnets

Questions

Question 1

Pour représenter le plateau de jeu, nous vous proposons d'utiliser un tableau de booléens de 30 cases appelé `plateauDeJeu`, dont chaque case représente la présence (ou l'absence) d'un bâtonnet. En début de partie, tous les bâtonnets sont présents donc toutes les cases du tableau contiennent la valeur `true`. De plus, pour savoir à qui est le tour de jouer, une variable entière `joueurCourant` représente le numéro du joueur (1 ou 2); elle vaut 1 au début de la partie.

Donnez les instructions permettant de définir les différentes variables du jeu et de les initialiser.

Question 2

La méthode `prochainJoueur()` permet de changer de joueur à chaque tour de jeu. Elle renvoie la valeur 2 lorsque le paramètre `numJoueur` vaut 1, et renvoie 1 dans les autres cas.

```
static int prochainJoueur(int numJoueur) {...}
```

Complétez le corps de la méthode.

Question 3

La méthode `affiche()` permet de montrer à l'écran l'état du plateau de jeu passé en paramètre. Voici un exemple d'affichage, en cours de partie :

```
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 26 27 28 29
|   |   | |     | | |  |  |     |  |     |  |  |  |  |  |  |           |  |  |
```

La rangée du haut contient les indices des cases pouvant contenir des bâtonnets, la rangée du bas montre la présence des bâtonnets par un caractère « | » et leur absence par un caractère d'espacement. Notez que chaque case est séparée de la suivante par un espace. De plus, pour permettre l'alignement de l'affichage des indices et des bâtonnets, les bâtonnets ou espaces de la rangée du bas sont séparés par un espace supplémentaire lorsque les valeurs des indices sont supérieures à 10 (deux chiffres).

```
static void affiche(boolean[] plateauDeJeu) {...}
```

Complétez le corps de la méthode.

Question 4

La méthode `choisirUnCoup()` permet à un joueur de préciser l'indice d'un bâtonnet qu'il souhaite retirer du plateau de jeu passé en paramètre. Cette méthode demande au joueur le numéro de la case dans laquelle il veut retirer un bâtonnet. Pour que le coup soit valide, il faut que la valeur fournie soit comprise entre 0 et la taille du plateau de jeu – 1 et que cette case ne soit pas déjà vide. Tant que le coup n'est pas valide, la méthode indique que le coup est invalide et demande un autre numéro de case. Enfin, cette méthode retourne le numéro de la case valide choisie par le joueur.

Voici un exemple en cours de partie :

```
(affichage de l'état du jeu avant l'appel de la méthode)
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 26 27 28 29
|   |   | |     | | |  |  |     |  |     |  |  |  |  |  |  |           |  |  |
Quel est votre coup ?
1
Coup impossible ! Recommencez.
31
Coup impossible ! Recommencez.
2
(dans cet exemple, la méthode choisirUnCoup() renverra la valeur 2)
```

```
static int choisirUnCoup(boolean[] plateauDeJeu) {...}
```

Complétez le corps de la méthode.

Question 5

La méthode `objetsRestants()` renvoie le nombre de bâtonnets restant en jeu dans le plateau de jeu fourni en paramètre.

```
static int objetsRestants(boolean[] plateauDeJeu) {...}
```

Complétez le corps de la méthode.

Question 6

La méthode `gagnant()` permet de savoir si l'un des deux joueurs a gagné la partie. Il y a un gagnant s'il ne reste plus aucun bâtonnet sur le plateau de jeu passé en paramètre. Cette méthode renvoie la valeur `true` s'il y a un gagnant et la valeur `false` sinon.

```
static boolean gagnant(boolean[] plateauDeJeu) {...}
```

Complétez le corps de la méthode.

Question 7

La méthode `min()` renvoie le plus petit des deux paramètres `a` et `b`.

```
static int min(int a, int b) {...}
```

Complétez le corps de la méthode.

Question 8

La méthode `jouerUnTour()` utilise certaines des méthodes écrites précédemment pour demander au joueur combien de bâtonnets il souhaite prendre, puis, pour chaque bâtonnet à enlever, lui faire choisir un bâtonnet valide et le retirer du plateau de jeu.

Le nombre de bâtonnets que le joueur peut retirer doit être compris entre 1 et le minimum des deux nombres suivants : 3 et le nombre restant de bâtonnets. Par exemple, s'il reste 10 bâtonnets, le joueur peut choisir de retirer entre 1 et 3 bâtonnets. S'il reste 2 bâtonnets, le joueur peut retirer 1 ou 2 bâtonnets. Tant que le joueur ne donne pas un nombre de bâtonnets à retirer valide, la méthode affiche un message d'erreur et redemande la saisie d'une valeur correcte.

Voici un exemple en cours de partie :

```
Combien d'objets voulez-vous prendre ?
19
Valeur impossible ! Recommencez.
3
Objet n°1
Quel est votre coup ?
26
Objet n°2
Quel est votre coup ?
25
Objet n°3
Quel est votre coup ?
24
```

```
static void jouerUnTour(boolean[] plateauDeJeu) {...}
```

Complétez le corps de la méthode.

Question 9

Dans cette question, vous devez intégrer l'ensemble des codes écrits précédemment pour réaliser le jeu de Nim.

Voici un exemple de début de partie :

```
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 26 27 28 29
| | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Joueur n°1 c'est à vous !
Combien d'objets voulez-vous prendre ?
3
Objet n°1
Quel est votre coup ?
1
Objet n°2
Quel est votre coup ?
3
Objet n°3
Quel est votre coup ?
13
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 26 27 28 29
|   |   | | | | | | | |  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Joueur n°2 c'est à vous !
Combien d'objets voulez-vous prendre ?
19
Valeur impossible ! Recommencez.
3
...
```

Voici un exemple de fin de partie :

```
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 26 27 28 29
                      |              |

Joueur n°1 c'est à vous !
Combien d'objets voulez-vous prendre ?
3
Valeur impossible ! Recommencez.
1
Objet n°1
Quel est votre coup ?
10
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 26 27 28 29
                                    |

Joueur n°2 c'est à vous !
Combien d'objets voulez-vous prendre ?
1
Objet n°1
Quel est votre coup ?
15
Joueur n°1, vous avez gagné !
```

Écrivez le programme principal du jeu de Nim, en utilisant les codes et méthodes écrits aux questions précédentes.

Corrigé CF2 2009

```
public class CF2_2009_Nim {

//Q2 - 1 point
  static int prochainJoueur(int numJoueur) {
     if (numJoueur == 1)     //0,5 (if)
        return 2;
     return 1;                  //0,5 (return)
  }

//Q3 - 2,5 points
  static void affiche(boolean[] plateauDeJeu) {
     for (int i = 0; i < plateauDeJeu.length; i++)   //0,5
        System.out.print(i + " ");
     System.out.println();

     for (int i = 0; i < plateauDeJeu.length; i++) { //0,5
```

```
        if (plateauDeJeu[i])                    //0,5
            System.out.print("| ");
        else                                    //0,5
            System.out.print("  ");
        if (i >= 10)                            //0,5 (cas 2 chiffres)
            System.out.print(" ");
    }
    System.out.println();
  }

//Q4 - 3 points
  static int choisirUnCoup(boolean[] plateauDeJeu) {
    boolean coupValide = false;                 // variables 0,5
    int coup = -1;

    System.out.println("Quel est votre coup ?");   //0,5

    while (!coupValide) {                           //0,5
      coup = Lire.i();                              //0,5
      if (coup >= 0 && coup < plateauDeJeu.length &&     //0,5
                              plateauDeJeu[coup])
        coupValide = true;
      else
        System.out.println("Coup impossible ! Recommencez.");
    }
    return coup;                                    //0,5
  }

//Q5 - 1,5 points
  static int objetsRestants(boolean[] plateauDeJeu) {
    int nb = 0;
    for (int i = 0; i < plateauDeJeu.length; i++) {       //0,5
      if (plateauDeJeu[i])                                    //0,5
        nb++;
    }
    return nb;                                                      //0,5
  }

//Q6 - 1 point
  static int min(int a, int b) {
    if (a > b)                                      //0,5 (test)
      return b;
    else
      return a;                                     //0,5 (return)
  }

//Q7 - 1 point
```

```
//si pas d'utilisation de la m,thode objetsRestants, ne pas p,naliser mais ne
//pas compter plus de points non plus
  static boolean gagnant(boolean[] plateauDeJeu) {
     return (objetsRestants(plateauDeJeu) == 0); //0,5 (test) + 0,5 (return)
  }

//Q8 - 4,5 points
  static void jouerUnTour(boolean[] plateauDeJeu) {
     boolean coupValide = false;                    //0,5 variables
     int nb = -1;

     System.out.println("Combien d'objets voulez-vous prendre ?");   //0,5

     while (!coupValide) {                          //0,5
        nb = Lire.i();                              //0,5
        if (nb >= 1 && nb <= min(3, objetsRestants(plateauDeJeu)))  //0,5
           coupValide = true;
        else
           System.out.println("Valeur impossible ! Recommencez.");  //0,5
     }

     int coup;
     for (int i = 0; i < nb; i++) {                 //0,5
        System.out.println("Objet nø" + (i + 1));   //0,5
        coup = choisirUnCoup(plateauDeJeu);         //0,5
        plateauDeJeu[coup] = false;                 //0,5 (compter 0,5 si
                                                    //fait en Q4)
     }
  }


  public static void main(String[] args) {
//Q1 - 2,5 points
     boolean[] plateauDeJeu = new boolean[30];      //1
     int numJoueur = 1;                             //0,5
     for (int i = 0; i < plateauDeJeu.length; i++)  //0,5
        plateauDeJeu[i] = true;                     //0,5

//Q9 - 3 points
     while (!gagnant(plateauDeJeu)) {               //0,5
        affiche(plateauDeJeu);                      //0,5
        System.out.println("\nJoueur nø" + numJoueur
                                    + " c'est … vous !");   //0,5
        jouerUnTour(plateauDeJeu);                  //0,5
        numJoueur = prochainJoueur(numJoueur);      //0,5
     }
     System.out.println("Joueur nø" + numJoueur
```

```
                                        + ", vous avez gagn, !"); // 0,5
    }

}
```