R1.01 - Initiation au développement

Sujet: SHUFFLE - mélanger les tableaux

Instructeur: Joseph AZAR - joseph.azar@univ-fcomte.fr

Problème:

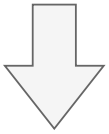
Nous devons créer un programme qui nous aide à jouer au jeu de loterie. Parmi les nombres allant de 1 à 42, nous devons en choisir 7. Dans ce petit jeu, le programme java va choisir ces 7 nombres pour nous. Pour ce faire, nous allons implémenter une fonction qui nous permet de mélanger un tableau et d'en renvoyer une partie (7 éléments dans ce jeu).

Mélange de tableaux - Shuffle

Étant donné un tableau, nous devons le mélanger au hasard. Toutes les permutations possibles des éléments du tableau doivent être également susceptibles d'être produites après le mélange.

Tableau d'origine :

1	2	3	4	5



SHUFFLE

Un exemple de sortie :

3	1	5	2	4

Approches pour résoudre ce problème :

Approche 1 : Utilisation d'un tableau auxiliaire
Approche 2 : Algorithme de Fisher Yates

Tout d'abord, nous discutons d'un algorithme de base qui nécessite de l'espace supplémentaire pour mélanger le tableau. Ensuite, nous discutons de l'algorithme de Fisher Yates qui mélange le tableau sur place.

Préparons le squelette de notre programme JAVA !

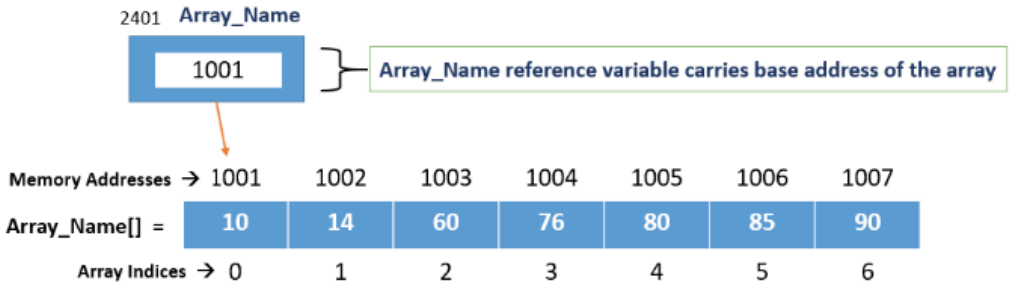
Création de tableaux

```
int[] loterie = new int[N];
```

Que se passe t-il ici?

Cette instruction fait deux choses : (1) elle crée un tableau en utilisant new int[N] et (2) elle assigne la référence du tableau nouvellement créé à la variable loterie.
La déclaration d'une variable de tableau, la création d'un tableau et l'attribution de la référence du tableau à la variable peuvent être combinées dans l'instruction ci-dessus.

```
int Array_Name = new int[7];
```



Approche 1 : utilisation d'un tableau auxiliaire

Algorithme:

1. Créez un tableau auxiliaire.

2. Tant qu'il y a plus d'éléments dans le tableau donné :
    2.1. Choisissez un élément dans le tableau donné à l'aide de la fonction random.
    2.2. Supprimez cet élément du tableau et ajoutez-le au tableau auxiliaire.

3. Renvoyez le tableau auxiliaire.

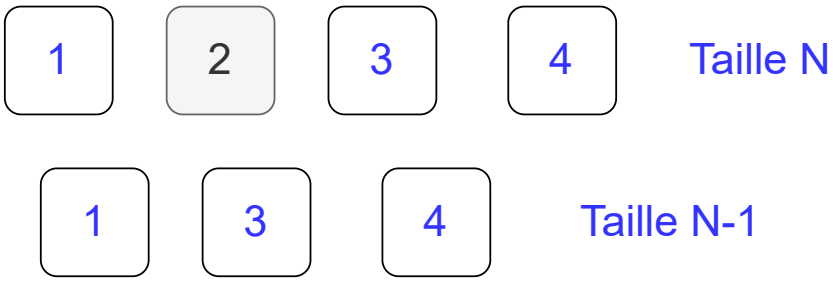
Exemple:

Tableau donné array = [1,2,3,4,5]
Tableau auxiliaire aux array = [ ]

it 1: rand(array) = 3, array = [1,2,4,5], aux array = [3]
it 2: rand(array) = 1, array = [2,4,5], aux array = [3,1]
it 3: rand(array) = 4, array = [2,5], aux array = [3, 1, 4]
it 4: rand(array) = 5, array = [2], aux array = [3, 1, 4, 5]
it 5: rand(array) = 2, array = [ ], aux array = [3, 1, 4, 5, 2]

Supprimer un élément d'un tableau

Supprimer un élément:




```java
/**
 * Fonction pour supprimer l'élément.
 * @param arr
 * @param N
 * @param index
 */
static int[] supprimerElement(int[] arr, int N, int index)
{
    // Si le tableau est vide
    // ou l'index n'est pas dans la plage du tableau
    // renvoie le tableau d'origine
    if (arr == null || index < 0
            || index >= N) {
        return arr;
    }
    // Créer un autre tableau de taille un de moins
    int[] autreTab = new int[N - 1];
    // Copiez les éléments sauf l'index
    // du tableau d'origine à l'autre tableau
    for (int i = 0, k = 0; i <= N-1; i++) {
        // si l'index est l'index de l'élément de suppression
        if (i == index) {
            continue;
        }
        // si l'index n'est pas l'index de l'élément de suppression
        autreTab[k++] = arr[i];
    }
    // retourne le tableau résultant
    return autreTab;
}
```

Approche 2 : Algorithme de Fisher Yates

L'algorithme de Fisher Yates offre une légère amélioration par rapport à l'approche précédente. Il mélange le tableau sur place, c'est-à-dire qu'il ne nécessite pas d'espace supplémentaire.

Algorithme:

1. pour i allant de n-1 à 1
    1.1 j = entier aléatoire tel que 0 <= j <= i
    1.2 échanger a[j] et a[i]

Exemple:

Étant donné, arr = [1,2,3,4,5]

pour i allant de 4 à 1 :
i = 4, arr[i] = 5, soit j = 2, puis échanger arr[4] avec arr[2] => arr = [1,2,5,4,3]
i = 3, arr[i] = 4, soit j = 1, puis échanger arr[3] avec arr[1] => arr = [1,4,5,2,3]
i = 2, arr[i] = 5, soit j = 0, puis échanger arr[2] avec arr[0] => arr = [5,4,1,2,3]
i = 1, arr[i] = 4, soit j = 1, puis échanger arr[1] avec arr[1] => arr = [5,4,1,2,3]

Cela donne un tableau mélangé = [5,4,1,2,3]

Implémentation JAVA complète

```java
import java.util.Random;
public class Shuffle {
    /**
     * Initialiser un tableau de loterie.
     * Après l'exécution, arr est nécessairement différent.
     * @param arr
     * @param N
     */
    static void initLoterie(int[] arr, int N){
        for (int i = 0; i < N; i++){
            arr[i] = i+1;
        }
    }
    /**
     * Afficher les éléments.
     * @param arr
     * @param N
     */
    static void afficherElems(int[] arr, int N){
        for (int i = 0; i < N; i++){
            System.out.print(arr[i]);
            System.out.print(" ");
        }
        System.out.println("");
    }

    /**
     * Fonction pour supprimer l'élément.
     * @param arr
     * @param N
     * @param index
     */
    static int[] supprimerElement(int[] arr, int N, int index)
    {
        // Si le tableau est vide
        // ou l'index n'est pas dans la plage du tableau
        // renvoie le tableau d'origine
        if (arr == null || index < 0
                || index >= N) {
            return arr;
        }
        // Créer un autre tableau de taille un de moins
        int[] autreTab = new int[N - 1];
        // Copiez les éléments sauf l'index
        // du tableau d'origine à l'autre tableau
        for (int i = 0, k = 0; i <= N-1; i++) {
            // si l'index est l'index de l'élément de suppression
            if (i == index) {
                continue;
            }
            // si l'index n'est pas l'index de l'élément de suppression
            autreTab[k++] = arr[i];
        }
        // retourne le tableau résultant
        return autreTab;
    }
```

```
    /**
     * Mélange un tableau donné et renvoie un tableau mélangé..
     * Après l'exécution, arr est nécessairement différent.
     * @param arr
     * @param N
     */
    static int[] shuffle_aux(int[] arr, int N){
        int[] aux_array = new int[N];
        int newSize = N;
        int aux_array_index = 0;
        while(newSize != 0){
            // Le nextInt(int n) est utilisé pour obtenir
            // un nombre aléatoire
            // entre 0(inclusif) et le nombre passé dans
            // cet argument(n), exclusif.
            int num = new Random().nextInt(newSize);
            aux_array[aux_array_index] = arr[num];
            arr = supprimerElement(arr,newSize,num);
            newSize--;
            aux_array_index++;
        }
        return aux_array;
    }

    /**
     * Renvoyer les 7 premiers éléments.
     * @param arr
     */
    static int[] premSeptElems(int[] arr){
        int[] tab = new int[7];
        for (int i = 0; i < 7; i++){
            tab[i] = arr[i];
        }
        return tab;
    }
    /**
     * Mélange un tableau donné en utilisant l'approche Fisher Yates.
     * Après l'exécution, arr est nécessairement différent.
     * @param arr
     * @param N
     */
    static int[] shuffle_fyates(int[] arr, int N){
        for(int i = N-1; i > 0; i--){
            int num = new Random().nextInt(i+1);
            int temp = arr[num];
            arr[num] = arr[i];
            arr[i] = temp;
        }
        return arr;
    }
```

```
    public static void main(String args[]){
        int N = 42; // nombre maximum dans le jeu de loterie
        int[] loterie = new int[N];
        initLoterie(loterie,N);
        afficherElems(loterie,N);
        //int[] shuffleTab = shuffle_aux(loterie,N);
        int[] shuffleTab = shuffle_fyates(loterie,N);
        afficherElems(shuffleTab,N);
        int[] mesNumeros = premSeptElems(shuffleTab);
        afficherElems(mesNumeros,7);
    }
}
```