Wetzler Alexandre
Sabaly Aurélien

Novembre 2015

Compte rendu TP N°2
Les Tableaux

Calculs dans un tableau.

Calculs moyenne et écart-type

```java
public static void addition(int tab[]) {
    int total = 0;

    for (int i = 0; i < tab.length; i++) {
        total += tab[i];
    }

    double ecartType = 0;
    double moyenne = total / tab.length + 1;
    int diff = 0;

    for (int i = 0; i < tab.length; i++) {
        diff = diff + (int) ((tab[i] - moyenne) * (tab[i] - moyenne));
    }

    double variance = diff / tab.length;
    ecartType = Math.sqrt(variance);
    System.out.println("la moyenne est de " + moyenne + " et l'écart type est de " + ecartType);
}
```

On récupère à l'aide d'une boucle tout les nombre que désire saisir l'utilisateur. On les stocks dans un tableau. On réalise ensuite le calcul de l'écart-type, de la moyenne et on réalise également la variance. Car la variance est nécéssaire pour le calcul de l'écart-type.

Nombres premiers

```java
public static int[] PremiersListe() {
    // declaration et allocation du tableau
    boolean[] premiers = new boolean[TAILLE];

    // initialiser toutes les cases a true (non barre)
    for (int i = 0; i < TAILLE; i++) {
        premiers[i] = true;
    }
    premiers[0] = false;

    int i = 2;
    while (i < TAILLE) {
        // on commence par le permier multiple superieur au nombre courant (i)
        int j = i * 2;
        while (j < TAILLE) {
            // on barre ce multiple
            premiers[j] = false;
            // passage au multiple suivant
            j += i;
        }
        // nombre suivant
        i++;
    }
    int count = 0;
    // affichage du tableau pour verification
    int j = 0;
    for (int k = 0; k < TAILLE; k++) {
        if (premiers[k] == true) {
            count++;
        }
    }
     int tab[] = new int[count];
     for (int k = 0; k < TAILLE; k++) {
         if (premiers[k] == true) {
             tab[j] = k;
             j++;
         }
     }
     System.out.println(count);
     PrintTab(tab);
     return tab;
}
```

Cette fonction nous permet de générer un tableau contenant les N premiers nombres premiers. Ainsi à partir de cela on a juste à récupérer le tableau puis le fournir à la fonction précédente. Qui va l'additionner et nous fournir la moyenne et l'écart-type de ces nombres.
Ici taille est une constante de classe permettant de savoir la taille des tableaux. Evitant ainsi les erreurs au moment de la compilation.

```java
static final int TAILLE = 100;
```

Recherches dans un tableau

Les multiples communs

```
public static int[] multiples(int q, int nb) {
    int count = 0;
    int num = q;
    int tempo[] = new int[nb];
    for (int i = 0; i < nb; i++) {
        tempo[i] = num;
        num = num + q;
        if (count == nb) {
            return tempo;
        } else {
            count++;
        }
    }
    return tempo;
}
```

Cette fonction « multiples » reçoit deux paramètres. Q qui est le nombre dont nous cherchons les multiples et nb qui est le nombre de multiples que l'on veut avoir. On crée ainsi un tableau de taille nb contenant nb multiples de q. Tableau qui sera ensuite retourné à l'utilisateur.

Comparaison de Tableaux

```
public static int[] commun(int T1[], int T2[]) {
    int tempo[] = new int[NOMBRE];
    int k = 0;

    for (int i = 0; i < NOMBRE; i++) {
        for (int j = 0; j < NOMBRE; j++) {
            if (T1[i] == T2[j]) {
                //la valeur que l'on en commun
                tempo[k]=T1[i];
                // l'indice dans le premier tableau
                tempo[k+1]= i;
                //l'indice dans le second tableau
                tempo[k+2]= j;
                k=k+3;
                //mettre toutes les données dans un seul tableau.
                System.out.print("nombre : " + T1[i]+ " en T1 : " + i);
                System.out.println(" et en T2 : " + j);
            }
        }
    }
    return tempo;
}
```

Cette fonction reçoit en paramètre deux tableaux, ici nommé T1 et T2. A partir de ces deux tableaux un troisième va être généré.

Celui-ci contiendra :

- En 0 le nombre commun au deux tableaux
- En 1 l'indice de ce nombre dans la tableau T1
- En 2 l'indice de ce nombre dans le tableau T2

Ainsi le tableau fonctionne par groupe de trois cases. Cette option a été prise car la consigne demandait de créer un tableau qui comportent les éléments. La meilleure solution aurait été de faire un tableau du type d'une classe. Classe que l'on aurait crée avec trois attributs, nombre, indice T1, indice T2.

Insertion de p nombres dans un tableau

```
public static void saisi(){
    System.out.println("Veuillez saisir la quantité de nombre que vous allez entrer");
    Scanner sc = new Scanner(System.in);
    int count=0;
    int p = sc.nextInt();
    int tab[]= new int[p];
    for (int i=0; i< p-1 ; i ++){
        int val = sc.nextInt();
        if(checkTab(tab,val)==false){
            tab[count] = val;
            Arrays.sort(tab);
            count++;
        } else {
            System.out.println("le nombre que vous avez saisr est déjà dans le tableau");
        }

    }
}
```

Voici la fonction checkTab utilisée pour vérifier si une valeur est présente dans un tableau. Cet algorithme gère les doublons ainsi que l'ordonnancement à chaque insertion dans le tableau.

```
public static boolean checkTab(int[] tab, int val){
    for(int i=0; i<tab.length;i++){
        if(tab[i]==val){
            return true;
        }
    }
    return false;
}
```

Cette fonction scan le tableau et nous dit si la valeur est déjà présente dans le tableau.

Retirer un nombre d'un tableau

Comme l'énoncé peut laisser penser à deux algorithme. J'ai réalisé ces deux algorithmes. Un où l'on supprime une occurrence de ce nombre à un indice précis et un second où l'on supprime tout les occurrences de ce nombre.

V1 : Suppression de toutes les occurrences

```
public static void retirerDoublon(int[] tab) {
    System.out.println("Quel nombre voulez vous supprimer du tableau ?");
    Scanner sc = new Scanner(System.in);
    int count = 0;
    int aRetirer = sc.nextInt();
    for (int i = 0; i < tab.length; i++) {
        if (tab[i] == aRetirer && tab[i] != 0) {
            for (int k = i; k < tab.length - 1; k++) {
                tab[k] = tab[k + 1];
            }
            tab[tab.length - 1] = 0;
        }
    }
    printTab(tab);
}
```

V2 : Suppression d'une occurrence

```
public static void retirerNombre(int[] tab,int indice, int nombre) {
  if(tab[indice]==nombre){
      for (int i = indice ; i < tab.length-1; i++) {
          tab[i] = tab[i+1];
      }
      tab[tab.length-1]=0;
      printTab(tab);
  } else {
      System.out.println("les valeurs fournit ne sont pas bonnes.");
      System.out.println("Il ny' a pas " + nombre + "à l'indice " + indice);
  }
}
```

Saisie utilisateur + retirer doublon

```java
public static void retirerDoublon() {
    System.out.println("Veuillez saisir la quantité de nombre que vous allez entrer");
    Scanner sc = new Scanner(System.in);
    int count = 0;
    int p = sc.nextInt();
    int tab[] = new int[p];
    for (int i = 0; i < p; i++) {
        System.out.println("nombre " + (i+1));
        tab[count] = sc.nextInt();
        count++;
    }
    for (int i = 0; i < tab.length; i++) {
        int tempo = tab[i];
        for (int k = i+1; k < tab.length; k++) {
            if (tab[k] == tempo) {
                for (int q = k; q < tab.length - 1; q++) {
                    tab[q] = tab[q + 1];
                    tab[tab.length-1]=0;
                }
            }
        }
    }
   printTab(tab);
}
```

Classer les mots dans un tableau

Classer les mots dans un tableau

```
public static void orderWordTab(String tab[]) {
    boolean inversion;
    do {
        inversion = false;

        for (int i = 0; i < tab.length - 1; i++) {
            if (tab[i].compareTo(tab[i + 1]) > 0) {
                String tmp = tab[i];
                tab[i] = tab[i + 1];
                tab[i + 1] = tmp;
                inversion = true;
            }
        }
    } while (inversion);
    printTab(tab);
}
```

On récupère un tableau en paramètre. On réalise ensuite un tri (tri à bulle) sur celui-ci.

Comparaison de deux tableaux, affichage mot commun

```java
public static void commun(String T1[], String T2[]) {
    String tempo[] = new String[15];
    int q = 0;
    for (int i = 0; i < T1.length; i++) {
        for (int k = 0; k < T2.length; k++) {
            if (T1[i] == T2[k]) {
                //la valeur que l'on en commun
                tempo[q] = T1[i];
                // l'indice dans le premier tableau
                tempo[q + 1] = Integer.toString(i);
                //l'indice dans le second tableau
                tempo[q + 2] = Integer.toString(k);
                q = q + 3;
                //mettre toutes les données dans un seul tableau.
                System.out.print("nombre : " + T1[i] + " en T1 : " + i);
                System.out.println(" et en T2 : " + k);
            }
        }
    }
}
```

Comme dans le petit 2 on range dans un tableau par niveau de 3 bien que cela ne soit pas demandé ici. J'ai utilisé l'opérateur == pour comparer mes chaines de caractères mais j'aurai pu aussi utiliser compareTo() . Il nous affiche ainsi à la fin les mots en commun dans ces deux tableaux ainsi que leur indice respectif

Matrices

Ajouter deux matrices (de même dimension)

```java
public static int[][] add(int[][] A, int[][] B) {
    int m = A.length;
    int n = A[0].length;
    int[][] C = new int[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
    return C;
}
```

Multiplication de deux matrices

```java
public static int[][] multiply(int[][] A, int[][] B) {
    int mA = A.length;
    int nA = A[0].length;
    int mB = B.length;
    int nB = B[0].length;
    if (nA != mB) {
        throw new RuntimeException("La dimension de la matric n'est pas bonne");
    }
    int[][] C = new int[mA][nB];
    for (int i = 0; i < mA; i++) {
        for (int j = 0; j < nB; j++) {
            for (int k = 0; k < nA; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}
```

Pour la multiplication on récupère les dimensions et on vérifie que leur dimension sont bonnes. Car une multiplication de matrice axb*cxd, il faut que b et c soient égaux. Ainsi si elles sont différentes ici cela ne fonctionnera pas.

Main

```
public static void main(String[] args) {
    int MAT1[][] = {{1, 2, 1},
                    {2, 1, 2},
                    {1, 2, 1}};

    int MAT2[][] = {{1, 2, 1},
                    {2, 1, 2},
                    {1, 2, 1}};

    int MAT3[][] = {{1, 2, 1},
                    {2, 1, 2},
                    {1, 2, 1}};

    MAT3 = add(MAT1, MAT2);
    PrintTab(MAT3);
    MAT3 = multiply(MAT1, MAT2);
    PrintTab(MAT3);
}
```

Main permettant de faire fonctionner cette classe matrice.

Tris dans un tableau

Tri par sélection

```
static void tri_selection(int tab[], int taille) {
    int indice_max;

    // à chaque tour de boucle, on va déplacer le plus grand élément
    // vers la fin du tableau, on diminue donc à chaque fois sa taille
    // car le dernier élément est obligatoirement correctement
    // placé (et n'a donc plus besoin d'être parcouru/déplacé)
    for (; taille > 1; taille--) // tant qu'il reste des éléments non triés
    {
        indice_max = max(tab, taille);

        echanger(tab, taille - 1, indice_max); // on échange le dernier élément avec le plus grand
    }
    PrintTab(tab);
}
```

Tri à bulles

```
public static void tri_bulles(int tableau[]) {
    int longueur = tableau.length;
    boolean inversion;

    do {
        inversion = false;

        for (int i = 0; i < longueur - 1; i++) {
            if (tableau[i] > tableau[i + 1]) {
                echanger(tableau, i, i + 1);
                inversion = true;
            }
        }
    } while (inversion);
    PrintTab(tableau);
}

    // Échange deux éléments d'un tableau
static void echanger(int tab[], int x, int y) {
    int tmp;

    tmp = tab[x];
    tab[x] = tab[y];
    tab[y] = tmp;
}
```

Ici on va inverser tout les valeurs afin de les remettre au bon endroit.

Tri par insertion

```
static void inserer(int element_a_inserer, int tab[], int taille_gauche) {
    int j;
    for (j = taille_gauche; j > 0 && tab[j - 1] > element_a_inserer; j--) {
        tab[j] = tab[j - 1];
    }
    tab[j] = element_a_inserer;
}

static void tri_insertion(int tab[], int taille) {
    int i;
    for (i = 1; i < taille; ++i) {
        inserer(tab[i], tab, i);
    }

    PrintTab(tab);
}
```

Tri par Shell

```
public static void tri_shell(int tableau[]) {
    int longueur = tableau.length;
    int n = 0;

    while (n < longueur) {
        n = 3 * n + 1;
    }

    while (n != 0) {
        n = n / 3;
        for (int i = n; i < longueur; i++) {
            int valeur = tableau[i];
            int j = i;

            while ((j > (n - 1)) && (tableau[j - n] > valeur)) {
                tableau[j] = tableau[j - n];
                j = j - n;
            }
            tableau[j] = valeur;
        }
    }
    PrintTab(tableau);
}
```

Main

```
public static void main(String[] args) {
    int[] tableau = generationTableau(TAILLEMAX);
    System.out.println("tableau initial non-trié");
    PrintTab(tableau);
    System.out.println("\n");
    System.out.println("tableau resultat du tri fusion");
    long temps1 = System.currentTimeMillis();
    triFusion(tableau);
    long temps2 = System.currentTimeMillis();
    long res = temps2 - temps1;
    System.out.println("le calcul du tri fusion a mis ->" + res + "ms\n\n");

    System.out.println("tableau résultat du tri par shell");
    temps1 = System.currentTimeMillis();
    tri_shell(tableau);
    temps2 = System.currentTimeMillis();
    res = temps2 - temps1;
    System.out.println("le calcul du tri shell a mis ->" + res + "ms\n\n");

    System.out.println("tableau résultat du tri à bulles");
    tri_bulles(tableau);
    System.out.println("tableau résultat du tri par insertion");
    tri_insertion(tableau, TAILLEMAX);
    System.out.println("tableau résultat du tri par selection");
    tri_selection(tableau, TAILLEMAX);

}
```

Ici on exécute nos différents tris sur un tableau générer de manière aléatoire.
Et on calcul également le temps d'exécution de ce tri.