

Solution des exercices sur les tableaux et méthodes

Les méthodes

Exercice 1 : Écrivez une méthode qui permet de déterminer si une année est bissextile

```
import javax.swing.*;

public class Bissextile {

    public static void main(String args[]) {

        int annee;

        boolean bissextile;

        String message;

        annee = Integer.parseInt(JOptionPane.showInputDialog(

            "Entrez une année (4 chiffres)");

        bissextile = estBissextile(annee); // appel à la méthode

        if (bissextile)

            message = " est ";

        else

            message = " n'est pas ";

        JOptionPane.showMessageDialog(null,

            annee + message + "bissextile");

        System.exit(0);

    } // fin de la méthode main

    // méthode qui indique si une année est bissextile ou non
    // pour être bissextile, l'année doit se diviser par 400 ou,
    // si ce n'est pas un siècle, l'année doit se diviser par 4
    static boolean estBissextile(int an)

    {

        boolean bissextile;

        bissextile = (an % 400 == 0 || (an % 100 != 0 && an % 4 == 0));

        return bissextile;

    } // fin de la méthode estBissextile

} // fin de la classe
```

Exercice 2 : Écrivez une méthode qui permet de calculer le périmètre d'un rectangle

```
public class Perimetre {  
    public static void main(String args[]) {  
        int largeur = 5;  
        int hauteur = 3;  
        System.out.println("Le périmètre du rectangle est " +  
            calculerPerimetre(largeur, hauteur));  
  
    } // fin de la méthode main  
    // méthode qui calcule le périmètre d'un rectangle  
    // paramètres reçus : largeur et hauteur du rectangle  
    static int calculerPerimetre(int largeur, int hauteur)  
    {  
        return (hauteur + largeur) * 2;  
  
    } // fin de la méthode calculerPerimetre  
  
} // fin de la classe
```

Exercice 3 : Qu'affiche le programme suivant ?

```
public class Exercice3 {  
    {  
        public static void main (String args[])  
        {  
            int nombre = 10;  
            plus1(nombre);  
            System.out.println(nombre + " ");  
            plus1(nombre + 15);  
            System.out.println(nombre);  
        }  
        static void plus1(int z)  
        {  
            z++;  
            System.out.print(z + " ");  
        }  
    }  
}
```

Résultats affichés sur la console :

11 10
26 10

Exercice 4 : Qu'affiche le programme suivant ?

```
public class Exercice4 {  
    {  
        public static void main (String args[])  
        {  
            int nombre = 10;  
            nombre = plus1(nombre);  
            System.out.println(nombre);  
        }  
  
        static int plus1(int z)  
        {  
            z++;  
            System.out.print(z + " ");  
            return z;  
        }  
    }  
}
```

Résultats affichés sur la console :

11 11

Exercice 5 : Qu'affiche le programme suivant ?

```
public class Exercice5 {
{
    public static void main (String args[])
    {
        int a = 5,
            b = 6,
            c = 7;

        System.out.println("Avant l'appel de test");
        System.out.print("a = " + a + " b = " + b +
            " c = " + c + "\n\n");

        c = test(a, b);
        System.out.println("Après l'appel de test");
        System.out.print("a = " + a + " b = " + b +
            " c = " + c);
    }

    static int test(int x, int y)
    {
        int z;
        x += 10;
        y += 10;
        z = x + y;
        System.out.println("À l'intérieur de test");
        System.out.print("x = " + x + " y = " + y +
            " z = " + z + "\n\n");

        return z;
    }
}
```

Résultats affichés sur la console :

Avant l'appel de test
a = 5 b = 6 c = 7
À l'intérieur de test
x = 15 y = 16 z = 31
Après l'appel de test
a = 5 b = 6 c = 31

Exercice 6 : Complétez le programme suivant qui appelle la méthode *afficherMultiple*. La méthode affiche sur la console tous les multiples de 3 compris entre 0 et un nombre entier limite, reçu en paramètre.

```
public class Exercice6 {

    public static void main(String args[])
    {
        // appel de la méthode pour faire afficher
        // les multiples de 3 compris entre 0 et 12
        afficherMultiple(12);

    } // fin de la méthode main

    // méthode qui affiche les multiples de 3
    // compris entre 0 et limite
    static void afficherMultiple(int limite)
    {
        System.out.println("Multiples de 3 compris entre 0 et " + limite);
        for (int i = 0; i <= limite; i += 3)
            System.out.print(i + " ");

    } // fin de la méthode afficherMultiple

} // fin de la classe
```

Exercice 6B : Modifiez le programme précédent pour que la méthode reçoive également en paramètre le nombre dont on veut afficher les multiples.

```
public class Exercice6B {

    public static void main(String args[])
    {
        // appel de la méthode pour faire afficher
        // les multiples de 2 compris entre 0 et 8
        afficherMultiple(8, 2);

    } // fin de la méthode main
```

```
// méthode qui affiche les multiples de multiple
// compris entre 0 et limite

static void afficherMultiple(int limite, int multiple)
{
    System.out.println("Multiples de " + multiple +
                       " compris entre 0 et " + limite);
    for (int i = 0; i <= limite; i += multiple)
        System.out.print(i + " ");

} // fin de la méthode afficherMultiple

} // fin de la classe
```

Exercice 7 : Écrivez une méthode *estEntier* qui reçoit une chaîne de caractères et qui s'assure qu'il n'y a que des chiffres dans cette chaîne. Votre méthode devra retourner **true** ou **false**.

```
// la méthode retourne true si tous les caractères d'une chaîne reçue
// sont des chiffres, autrement elle retourne false
static boolean estEntier(String chaine)
{
    boolean valide = true;

    for (int i = 0; i < chaine.length() && valide; i++)
        if (! Character.isDigit(chaine.charAt(i)))
            valide = false;

    return valide;

} // fin de la méthode estEntier
```

Exercice 8 : Écrivez une méthode *retrancherTexte* qui reçoit deux chaînes de caractères en paramètres. Votre méthode devra chercher la seconde chaîne dans la première et en enlever la première occurrence. Elle retourne ensuite la chaîne transformée.

```
// la méthode retourne une chaîne formée de la première chaîne dans
// laquelle on a retranché la première occurrence de la deuxième chaîne
static String retrancherTexte(String chaine1, String chaine2)
{
    String resultat = chaine1;
    int posi;

    posi = chaine1.indexOf(chaine2);

    if (posi != -1)
        resultat = chaine1.substring(0, posi) +
                    chaine1.substring(posi + chaine2.length());

    return resultat;

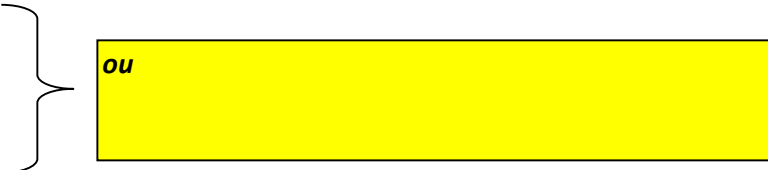
} // fin de la méthode retrancherTexte
```

Exercice 9 : Écrivez une méthode *rechercherPlusGrand* qui retourne le plus grand de trois nombres entiers reçus en paramètres.

```
// la méthode retourne le plus grand de trois nombres entiers
//
static int rechercherPlusGrand(int nb1, int nb2, int nb3)
{
    int max;

    max = Math.max(nb1, nb2);
    max = Math.max(max, nb3);

    return max;
} // fin de la méthode rechercherPlusGrand
```



Exercice 10 : Écrivez une méthode *calculerConsommation* qui reçoit en paramètres une distance en kilomètres ainsi que le nombre de litres consommés (deux nombres réels). La méthode retourne la consommation obtenue en litres par 100 kilomètres. Par exemple, pour une distance de 592.5 kilomètres et une consommation de 38.75 litres, on aura une consommation de 6.45 litres par 100 kilomètres.

```
// la méthode retourne la consommation en litres par 100 kilomètres
// d'après la distance parcourue et le nombre de litres consommés
static double calculerConsommation( double distance, double litres )
{
    return litres / distance * 100;
} // fin de la méthode calculerConsommation
```

Exercice 11 : Écrivez la méthode *obtenirNombreValide* qui doit lire et valider un nombre selon l'algorithme de la page 10 des notes de cours sur les algorithmes. La méthode retourne ensuite le nombre valide.

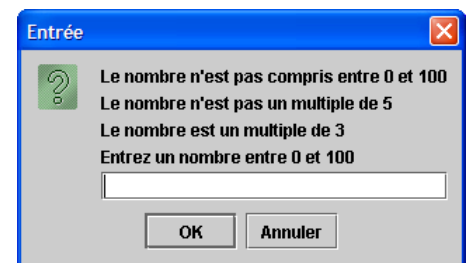
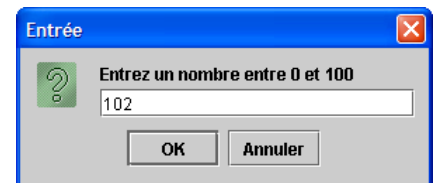
```
import javax.swing.*;

public class Exercice10 {

    public static void main(String args[])
    {
        JOptionPane.showMessageDialog(null,
            "Le nombre obtenu est " + obtenirNombreValide());
        System.exit(0);
    }

    static int obtenirNombreValide()
    {
        String message = "";
        boolean valide;
        int nombre;

        do
        {
            nombre = Integer.parseInt(JOptionPane.showInputDialog(
                message + "Entrez un nombre entre 0 et 100"));
            valide = true;
            message = "";
        }
        while (!valide);
    }
}
```



```
        if (nombre < 0 || nombre > 100)
        {
            valide = false;
            message = "Le nombre n'est pas compris entre 0 et 100\n";
        }

        if (nombre % 5 != 0)
        {
            valide = false;
            message += "Le nombre n'est pas un multiple de 5\n";
        }

        if (nombre % 3 == 0)
        {
            valide = false;
            message += "Le nombre est un multiple de 3\n";
        }

    } while (! valide);

    return nombre;
} // fin de la méthode obtenirNombreValide
}
```

Les tableaux et méthodes

Exemples :

```
double[] tabSalaires; // tableau tabSalaires de type double
    ou
double tabSalaires[];

int[] nombres; // tableau nombres de type int
    ou
int nombres[];

char codes[]; // tableau codes de type char

String noms[]; // tableau noms de type String
    ou
String[] noms;
```

Exercices : Déclarez et créez les tableaux demandés

```
// tableau destiné à contenir les 365 températures de l'année
double tabTempératures[] = new double[365];

// tableau destiné à contenir les adresses de 10 employés
String tabAdresses[] = new String[10];

// tableau destiné à contenir les lettres de l'alphabet
char tabLettres[] = new char[26];
```

Exercices 1 :

Référez-vous aux tableaux créés précédemment

Instruction	Explication
<code>tabSalaires[2] = 50000;</code>	Met à 50000 le 3 ^{ième} salaire du tableau <i>tabSalaires</i>
<code>int longueur = noms.length;</code>	Retourne la longueur du tableau <i>noms</i> (on s'attend à obtenir une longueur de 12)
<code>codes[9] = 'm';</code> ou <code>codes[codes.length - 1] = 'm';</code>	Initialise le dernier caractère du tableau <i>codes</i> à 'm'
<code>noms[1] = "Bob";</code>	Initialise le 2 ^{ième} nom du tableau <i>noms</i> à Bob

Exercices 2 :

Instruction	Explication
<code>int nbPoints[] = new int[50];</code>	Déclare et crée un tableau de 50 nombres entiers nommé <i>nbPoints</i>
<code>System.out.println(nbPoints[3]);</code>	Affiche le 4 ^{ième} élément du tableau <i>nbPoints</i>
<code>nbPoints[0] += 4;</code>	Additionne 4 au premier élément du tableau <i>nbPoints</i>
<code>lettres[lettres.length - 1] = 'S';</code>	Met la lettre ' S ' dans le dernier élément d'un tableau nommé <i>lettres</i> (on ne connaît pas le nombre d'éléments du tableau)

Exercices 3 :

Instruction	Explication
<code>int reponses[] = {3, 9, 2, 6, 8};</code>	Déclare et initialise un tableau nommé <i>reponses</i> avec les valeurs 3, 9, 2, 6 et 8
<code>String jours[] = {"DIM", "LUN", "MAR", "MER", "JEU", "VEN", "SAM"};</code>	Déclare et initialise un tableau nommé <i>jours</i> avec les valeurs DIM, LUN, MAR, MER, JEU, VEN et SAM
<code>boolean enPanne[] = {true, false, false};</code>	Déclare et initialise un tableau nommé <i>enPanne</i> avec les valeurs true, false et false
<code>double pourboires[] = {2.50, 8.75, 3.00, 0.75, 10.00};</code>	Déclare et initialise un tableau nommé <i>pourboires</i> avec les valeurs 2.50, 8.75, 3.00, 0.75 et 10.00

Exercice 1 : Pour chaque programme, indiquez ce qui sera affiché sur la console

	affichage à l'exécution
<pre>public class Exercice1A { public static void main(String args[]) { int tab[] = new int[5]; tab[0] = 10; for (int k = 1; k < tab.length; k++) tab[k] = 2 * tab[k - 1]; System.out.print("Le dernier vaut " + tab[4]); } }</pre>	Le dernier vaut 160

```

public class Exercice1B
{
    public static void main(String args[])
    {
        int tab[] = new int[10];

        int k = 2;

        tab[0] = 10;
        tab[1] = 2;

        for (int j = 1; j < 5; j++)
        {
            tab[k] = tab[k - 1] / 2;
            tab[k + 1] = tab[k] * (k + 1);
            k = k + 2;
        }

        System.out.print("****" +
            tab[tab[2]] + "****" + tab[9]);
    }
}

```

263

Exercice 2 : Écrivez une méthode qui reçoit en paramètre un tableau de nombres réels et qui retourne la plus grande valeur trouvée dans le tableau.

```

static double trouverMax(double tab[])

```

```

{

```

```

    double max = Double.MIN_VALUE;

```

```

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

```

```

        if (tab[i] > max)

```

```

            max = tab[i];

```

```

    return max;

```

```

} // fin de la méthode trouverMax

```

// on peut aussi initialiser max

// avec le premier élément

// et commencer la boucle de

// comparaison au deuxième élément

```
double max = tab[0];
```

```
for (int i = 1; i < tab.length; i++)
```

Exercice 3 : Écrivez une méthode qui reçoit en paramètre un tableau de nombres réels et qui retourne la position de la plus petite valeur trouvée dans le tableau.

```

static int trouverMin(double tab[])

```

```

{

```

```

    int positionMin = 0; // position du premier élément du tableau

```

```

    for (int i = 1; i < tab.length; i++)

```

```

        if (tab[i] < tab[positionMin])

```

```

            positionMin = i;

```

<code>return positionMin;</code>
<code>} // fin de la méthode trouverMin</code>

Exercice 4 : Écrivez une méthode qui reçoit en paramètres un tableau contenant le code sexe de chaque employé d'une compagnie (le code est 'F' pour une femme et 'H' pour un homme) et un tableau contenant leur âge respectif. La méthode doit retourner le nombre de femmes dont l'âge est inférieur à 30.

<code>static int compterJeunesFemmes(char tabSexe[], int tabAge[])</code>
<code>{</code>
<code>final char FEMME = 'F';</code>
<code>int compteur = 0;</code>
<code>for (int i = 0; i < tabSexe.length; i++)</code>
<code>if (tabSexe[i] == FEMME && tabAge[i] < 30)</code>
<code>compteur++;</code>
<code>return compteur;</code>
<code>} // fin de la méthode compterJeunesFemmes</code>

Exercice 5 : Complétez la méthode *main* et la méthode *rechercher* qui reçoit en paramètres un tableau de nombres entiers, le nombre d'éléments que contient le tableau ainsi que le nombre à chercher dans le tableau. La méthode doit retourner la position du nombre cherché dans le tableau ou -1 si le nombre n'est pas trouvé dans le tableau.

<code>import javax.swing.*;</code>
<code>import java.text.*;</code>
<code>public class Exercice5 {</code>
<code>public static void main(String args[]) {</code>
<code>final int NB_PROD = 8;</code>
<code>DecimalFormat cash = new DecimalFormat("0.00 \$");</code>
<code>int tabNoProd[] = { 234, 125, 657, 987, 213, 934, 678, 776};</code>
<code>double tabPrixProd[] = {45.99, 9.50, 5.75, 12.35, 9.75, 87.45, 56.99, 76.56};</code>
<code>int noProd,</code>
<code>qte,</code>
<code>posiProd; // position de noProd dans le tableau tabNoProd</code>
<code>double cout;</code>
<code>char reponse;</code>

do {
noProd = Integer.parseInt(JOptionPane.showInputDialog(
"Entrez le no du produit à acheter"));
qte = Integer.parseInt(JOptionPane.showInputDialog(
"Entrez la quantité désirée"));
posiProd = rechercher(tabNoProd, NB_PROD, noProd);
if (posiProd != -1)
{
cout = qte * tabPrixProd[posiProd];
JOptionPane.showMessageDialog(null,
"Le coût de cet achat est de " + cash.format(cout));
}
else
JOptionPane.showMessageDialog(null, "No de produit erroné");
reponse = JOptionPane.showInputDialog(
"Avez-vous un autre produit à acheter O/N ?").charAt(0);
} while (Character.toUpperCase(reponse) == 'O');
System.exit(0);
} // fin de la méthode main
static int rechercher(int tab[], int nbEl, int valeurCherchee)
{
int posi = -1;
boolean trouve = false;
for (int i = 0; i < nbEl && !trouve; i++)
if (tab[i] == valeurCherchee)
{
posi = i;
trouve = true;
}
return posi;
} // fin de la méthode rechercher
} // fin de la classe

Exercice 6 : Écrivez une méthode qui reçoit en paramètres un tableau contenant le code sexe de chaque employé d'une compagnie (le code est 'F' pour une femme et 'H' pour un homme) et un tableau contenant leur âge respectif. La méthode doit retourner l'âge de l'homme le plus jeune.

static int trouverAgeMinHomme(char tabSexe[], int tabAge[])
{
final char HOMME = 'H';
int ageMin = Integer.MAX_VALUE;
for (int i = 0; i < tabSexe.length; i++)
if (tabSexe[i] == HOMME && tabAge[i] < ageMin)
ageMin = tabAge[i];
return ageMin;
} // fin de la méthode trouverAgeMinHomme

Exercice 7 : Complétez le programme pour résoudre le problème suivant : on lance un dé 20 fois et on désire connaître le nombre de fois que chaque face du dé a été obtenue. Le lancer du dé est simulé à l'aide de la méthode Math.random()

```
import javax.swing.*;

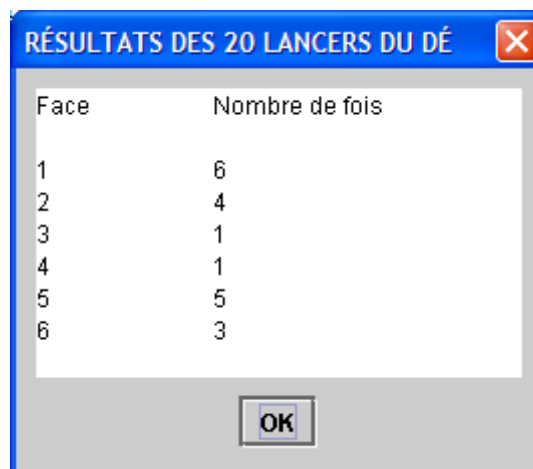
public class Exercice7 {

    public static void main(String args[]) {

        final int NB_LANCERS = 20;
        int tabNbFois[] = new int[6];           // tableau de compteurs
        jouerPartie(tabNbFois, NB_LANCERS);
        afficherResultats(tabNbFois, NB_LANCERS);
        System.exit(0);
    } // fin de la méthode main

    static void jouerPartie(int nbFois[], int nbLancers) {
        int face;
        for (int lancer = 1; lancer <= nbLancers; lancer++)
        {
            face = (int) (Math.random() * 6) + 1;
            nbFois[face - 1]++;
        }
    }

    static void afficherResultats(int nbFois[], int nbLancers) {
        JTextArea sortie = new JTextArea(5,5);
        sortie.append("Face\tNombre de fois\n\n");
        for (int face = 1; face <= 6; face++)
            sortie.append(face + "\t" + nbFois[face - 1] + "\n");
        JOptionPane.showMessageDialog(null, sortie,
            "RÉSULTATS DES " + nbLancers + " LANCERS DU DÉ",
            JOptionPane.PLAIN_MESSAGE);
    }
}
```



Exercice 8 : Modifiez le programme de l'exercice 7 en supposant qu'on veut jouer 10 parties de 20 lancers plutôt qu'une seule.

```
import javax.swing.*;

public class Exercice8 {

    public static void main(String args[]) {

        final int NB_LANCERS = 20;
        final int NB_PARTIES = 10;
        int tabNbFois[] = new int[6]; // tableau de compteurs

        for (int partie = 1; partie <= NB_PARTIES; partie++) {
            initialiser(tabNbFois); // remettre les compteurs à 0
            jouerPartie(tabNbFois, NB_LANCERS);
            afficherResultats(tabNbFois, NB_LANCERS, partie);
        }

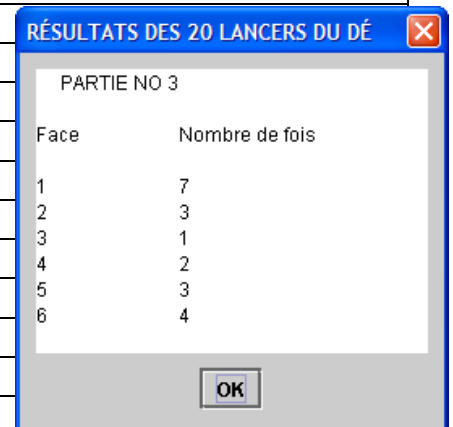
        System.exit(0);
    } // fin de la méthode main

    static void initialiser(int nbFois[]) {
        for (int i = 0; i < nbFois.length; i++)
            nbFois[i] = 0;
    } // fin de la méthode initialiser

    static void jouerPartie(int nbFois[], int nbLancers) {
        int face;
        for (int lancer = 1; lancer <= nbLancers; lancer++) {
            face = (int) (Math.random() * 6) + 1;
            nbFois[face - 1]++;
        }
    } // fin de la méthode jouerPartie

    static void afficherResultats(int nbFois[], int nbLancers, int noPartie) {
        JTextArea sortie = new JTextArea();
        sortie.append("PARTIE NO " + noPartie + "\n\n");
        sortie.append("Face\tNombre de fois\n\n");

        for (int face = 1; face <= 6; face++)
            sortie.append(face + "\t" + nbFois[face - 1] + "\n");
        JOptionPane.showMessageDialog(null, sortie,
            "RÉSULTATS DES " + nbLancers + " LANCERS DU DÉ",
            JOptionPane.PLAIN_MESSAGE);
    } // fin de la méthode afficherResultats
} // fin de la classe
```



Exercice 9 : Complétez le programme pour résoudre le problème suivant : on lit la note sur 60 (un entier) de 10 étudiants d'une classe en les mémorisant dans un tableau appelé **points**. À partir de ce tableau, on veut établir un tableau appelé **nbNotes** de dimension 7 et qui est composé de la façon suivante :

nbNotes[6] contient le nombre de notes égales à 60
 nbNotes[5] contient le nombre de notes de 50 à 59
 nbNotes[4] contient le nombre de notes de 40 à 49

 nbNotes[0] contient le nombre de notes de 0 à 9

```

import javax.swing.*;

public class Exercice9 {

    public static void main(String args[]) {

        final int NB_ETUD = 10;
        int points[] = new int[NB_ETUD];
        int nbNotes[] = new int[7]; // tableau des compteurs des notes
                                   // dans un intervalle

        remplirTableau(points, NB_ETUD);
        compterNbNotes(points, nbNotes, NB_ETUD);
        afficherRésultats(nbNotes, NB_ETUD);

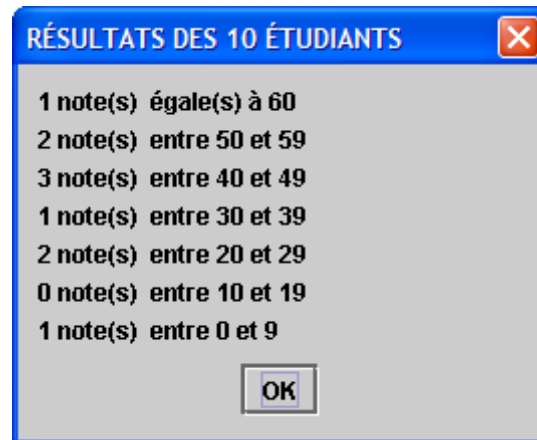
        System.exit(0);
    } // fin de la méthode main

    static void remplirTableau(int points[], int nbEtud) {
        for (int i = 0; i < nbEtud; i++)
            points[i] = Integer.parseInt(JOptionPane.showInputDialog(
                "Entrez la note de l'étudiant " + (i + 1)));
    } // fin de la méthode remplirTableau

    static void compterNbNotes(int points[], int nbNotes[], int nbEtud) {
        int intervalle;
        for (int i = 0; i < nbEtud; i++)
        {
            intervalle = points[i] / 10;
            nbNotes[intervalle]++;
        }
    }

    } // fin de la méthode compterNbNotes
  
```

static void afficherResultats(int nbNotes[], int nbEtud) {
String mots[] = {" entre 0 et 9", " entre 10 et 19",
" entre 20 et 29", " entre 30 et 39",
" entre 40 et 49", " entre 50 et 59",
" égale(s) à 60"};
String texte = "";
for (int i = nbNotes.length - 1; i >= 0; i--)
texte += nbNotes[i] + " note(s)" + mots[i] + "\n";
JOptionPane.showMessageDialog(null, texte,
"RÉSULTATS DES " + nbEtud + " ÉTUDIANTS",
JOptionPane.PLAIN_MESSAGE);
} // fin de la méthode afficherResultats
} // fin de la classe



Exercice 10 : Complétez le programme suivant qui permet de calculer et d'afficher le coût de l'achat d'un client (il y a plusieurs clients à traiter, mais on suppose qu'un client n'effectue qu'un seul achat) et qui va afficher (après avoir traité tous les clients de la journée) la quantité totale vendue pour chaque produit.

import javax.swing.*;
import java.text.*;
public class Exercice10 {
public static void main(String args[]) {
final int NB_PROD = 8;
int tabNoProd[] = { 234, 125, 657, 987, 213, 934, 678, 776};
double tabPrixProd[] = {45.99,9.50,5.75,12.35,9.75,87.45,56.99,76.56};
int tabQte[] = new int[NB_PROD];
traiterLesClients(tabNoProd, tabPrixProd, tabQte, NB_PROD);
afficherRésultats(tabNoProd, tabQte, NB_PROD);
System.exit(0);
} // fin de la méthode main
static void traiterLesClients(int tabNoProd[], double tabPrix[],
int tabQteTotale[], int nbProd) {
DecimalFormat cash = new DecimalFormat("0.00 \$");
int numero,
qte,
posiProd; // position du numéro dans le tableau tabNoProd
double cout;
char reponse;
do {
numero = Integer.parseInt(JOptionPane.showInputDialog(
"Entrez le numéro du produit à acheter :"));
qte = Integer.parseInt(JOptionPane.showInputDialog(
"Entrez la quantité désirée :"));
posiProd = rechercher(tabNoProd, nbProd, numero);
if (posiProd != -1)
{
cout = qte * tabPrix[posiProd];
JOptionPane.showMessageDialog(null,
"Le coût de cet achat est de " + cash.format(cout));
tabQteTotale[posiProd] += qte;
}
else
JOptionPane.showMessageDialog(null, "No de produit erroné");

```

    reponse = JOptionPane.showInputDialog(
        "Avez-vous un autre client à traiter O/N ?").charAt(0);
    reponse = Character.toUpperCase(reponse);
} while (reponse == 'O');
} // fin de la méthode traiterLesClients

static int rechercher(int tab[], int nbEl, int valeurCherchee)
{
    int posi = -1;
    boolean trouve = false;

    for (int i = 0; i < nbEl && !trouve; i++)
        if (tab[i] == valeurCherchee)
        {
            posi = i;
            trouve = true;
        }
    return posi;
} // fin de la méthode rechercher

static void afficherRésultats(int tabNoProd[], int tabQteTotale[],
    int nbProd) {
    JTextArea sortie = new JTextArea();
    sortie.append("Numéro du\tQuantité\nproduit\tttotale\n\n");

    for (int i = 0; i < nbProd; i++)
        sortie.append(tabNoProd[i] + "\t" + tabQteTotale[i] + "\n");

    JOptionPane.showMessageDialog(null, sortie,
        "RÉSULTATS DE LA JOURNÉE", JOptionPane.PLAIN_MESSAGE);
} // fin de la méthode afficherResultats
} // fin de la classe

```

