

Examen de Algorithmique et Programmation JAVA

Durée : 1h30

Aucun document autorisé

Mobiles interdits

Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives, ainsi que les noms donnés aux variables, l'emploi à bon escient des majuscules et la bonne indentation rentrent pour une part importante dans l'appréciation du travail.

1 Tableau

Écrivez la méthode statique `décompte(int []t)` qui donne le nombre d'occurrences de chaque nombre contenu dans le tableau `t`. Cette méthode renvoie un tableau indiquant, à l'indice `i`, le nombre d'occurrences du nombre `i` dans `t`.

Par exemple : pour un tableau `t` qui contient les valeurs `[3,1,0,3,5,6,5]`, l'exécution de la méthode `décompte(t)` renvoie le tableau `[1,1,0,2,0,2,1]` car le nombre 0 et le nombre 1 sont présents 1 fois chacun, le nombre 2 n'est pas présent, le nombre 3 est présent 2 fois, le nombre 4 n'est pas présent, le nombre 5 est présent 2 fois, et le nombre 6 est présent une seule fois.

```
public static int[] décompte(int []t) {
    int max= 0;
    for (int i = 0; i < t.length; i++) {
        max = t[i] > max ? t[i] : max;
    }
    int[] occ = new int[max];
    for (int i = 0; i < t.length; i++) {
        int v = t[i];
        occ[v]++;
    }
    return occ;
}
```

2 Fichiers

Un professeur souhaite corriger les rendus de TD de ses élèves. Les logins des élèves sont dans un fichier de nom `listeEleve.txt`. On supposera que chaque élève a rendu un fichier de nom `TP<login>.java` (exemple : `TPtoto.java` pour un élève de login `toto`).

Le professeur inscrit les notes de ses élèves dans un fichier `notes.txt`. Il y écrit chaque nom de login suivi de la note pour cet élève. La note attribuée à un élève est un entier tiré au hasard entre 0 et 20, sauf s'il n'y a aucun rendu (dans ce cas, on met 0).

On rappelle qu'on ouvre un fichier en lecture ou en écriture grâce aux classes `FileInputStream` ou `FileOutputStream`, et que l'exception `FileNotFoundException` est émise si le fichier ne peut être ouvert pour une raison ou une autre.

D'autre part, les méthodes `readUTF()` et `readInt()` (resp. `writeUTF()` et `writeInt()`) de la classe `DataInputStream` (resp. `DataOutputStream`) permettent de lire (resp. écrire) une chaîne de caractères et un entier dans le fichier.

Dans une classe `Correction`, écrivez la méthode `main` qui remplit le fichier de notes `notes.txt` selon les indications précédentes.

```
import java.io.*;

class Correction {
    public static void main(String[] argv) {
        DataInputStream listeIs = null, tpIs = null;
        DataOutputStream notesOs = null;
        java.util.Random gen = new java.util.Random();
        try {
            listeIs = new DataInputStream(new FileInputStream("listeEleve.txt"));
            notesOs = new DataOutputStream(new FileOutputStream("notes.txt"));
            while (true) {
                String login = listeIs.readUTF();
                notesOs.writeUTF(login);
                try {
                    tpIs = new DataInputStream(new FileInputStream("TP"+login+".java"));
                    notesOs.writeInt(gen.nextInt(21));
                    tpIs.close();
                }
                catch (FileNotFoundException e) {
                    notesOs.writeInt(0);
                }
            }
        }
        catch (EOFException e) { /* rien à faire */ }
        catch (IOException e) {
            System.err.println("Erreur à la correction : "+e.getMessage());
        }
        finally {
            if (listeIs != null)
                listeIs.close();
            if (tpIs != null)
                tpIs.close();
            if (notesOs != null)
                notesOs.close();
        }
    }
}
```

3 Héritage

Soient les déclarations de classes et d'interfaces Java suivantes :

```
abstract class ClassA {
    public void methode(){System.out.println(1);}
}
class ClassB extends ClassA {
}
abstract class ClassC extends ClassA implements Interface1 {
```

```

    public void methode(){System.out.println(2);}
    public void methode(int x){System.out.println(3);}
}

class ClassD extends ClassC {
    private void methode(){System.out.println(4);}
}
class ClassE extends ClassC {
    public void methode(String s){System.out.println(5);}
    public void methode(int x){System.out.println(6);}
}
Interface Interface1 {
    public void methode(int x);
}

```

1. Dessinez le graphe d'héritage des classes et de l'interface précédentes.
2. Pour chacune des lignes du code suivant, indiquez s'il y a une erreur, d'où vient l'erreur et, si la ligne est censée afficher quelque chose, indiquez ce qu'elle affiche.

```

1 : ClassA c1 = new ClassB();
2 : ClassA c2 = new ClassC();
3 : ClassD c3 = new ClassD();
4 : ClassB c4 = new ClassD();
5 : ClassC c5 = new ClassE();
6 : c1.methode();
7 : c2.methode();
8 : c3.methode();
9 : c3.methode(3);
10 : c4.methode();
11 : c5.methode();
12 : c5.methode(3);

```

```

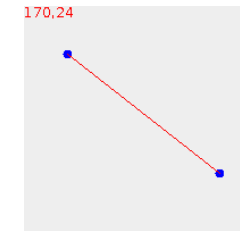
ClassA c1 = new ClassB();
ClassA c2 = new ClassC(); //classe abstraite
ClassD c3 = new ClassD();
ClassB c4 = new ClassD(); //mauvais type
ClassC c5 = new ClassE();
c1.methode(); //1
c2.methode(); //bug à cause de la déclaration de c2 qui n'a pas fonctionné
c3.methode(); //2
c3.methode(3); //3
c4.methode(); //bug à cause de la déclaration de c4
c5.methode(); //2
c5.methode(3); //6

```

4 Swing

On souhaite écrire une petite application graphique qui permet à l'utilisateur de dessiner deux points dans un carré, de les relier par une droite et d'afficher la distance entre les deux points. L'image ci-dessous donne une idée du rendu de l'application.

Le carré est représenté par un `JPanel`, et la méthode `public void paintComponent(Graphics g)` (que vous devrez écrire), vous permettra de dessiner dans le graphique `g` associé. Le point de coordonnées $(0,0)$ est le point nord-ouest du graphique.



`Graphics` possède en particulier les méthodes `fillOval`, `drawLine` et `drawString` dont vous aurez besoin. La documentation Java de ces méthodes est donnée en annexe.

On rappelle que l'interface `MouseListener` est l'auditeur des événements souris, et la méthode `public void mousePressed(MouseEvent e)` est déclenchée lorsqu'on appuie sur un des boutons de la souris. Les méthodes `getX()` et `getY()` d'un événement de type `MouseEvent` donnent les coordonnées du point (x, y) sur lequel l'utilisateur a cliqué.

Écrivez la classe `Carré` qui hérite de `JPanel` et implémente `MouseListener` et qui permet à l'utilisateur de tracer une droite entre 2 points sélectionnés à la souris, et d'afficher la distance entre ces points. **Notez bien que la droite et la distance ne s'affichent qu'après le second clic de souris.** Un 3ème clic de souris efface le carré.

On rappelle que la distance dans le plan cartésien entre deux points a et b de coordonnées (x_a, y_a) et (x_b, y_b) est égale à $\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$.

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.text.*;

```

```

public class Carré extends JPanel implements MouseListener {
    private int xA, yA, xB, yB; //coordonnées des 2 pts
    private int nbPoints;      //nb de pts dans le Carré
    private int size;          //longueur du côté du Carré

    public Carré(int size) {
        this.nbPoints = 0;
        this.size = size;
        this.setPreferredSize(new Dimension(size, size));
        addMouseListener(this);
    }

    /*
     * Rôle : calcule la distance entre les pts (xA,yA) et (xB,yB)
     * et l'affiche dans le coin sud-ouest du Carré courant
     */
    private String distance(int xA, int yA, int xB, int yB) {
        double d = Math.sqrt((xB-xA)*(xB-xA)+(yB-yA)*(yB-yA));
        NumberFormat df = new DecimalFormat("#.##");
        return df.format(d);
    }

    /**
     * Rôle : affiche/dessine le contenu du Carré
     */
}

```

```

public void paintComponent(Graphics g) {
    g.setColor(Color.blue);
    switch (this.nbPoints) {
        case 1 : //dessiner le 1er point
            g.fillOval(xA, yA, 8, 8); break;
        case 2: //dessiner le 2ème point
            g.fillOval(xB, yB, 8, 8);
            g.setColor(Color.red);
            //tracer la ligne qui les relie
            g.drawLine(xA+4, yA+4, xB+4, yB+4);
            //écrire la distance entre les 2 points
            g.drawString(distance(xA, yA, xB, yB), 0, this.size);
            break;
        default :
            //effacer le carré
            g.clearRect(0, 0, this.size, this.size);
    }
}

/**
 * Rôle : callback lorsqu'un bouton de souris est enfoncé
 * Mémoire les coordonnées du point sélectionné dans le
 * Carré
 */
public void mousePressed(MouseEvent e) {
    switch (++this.nbPoints) {
        case 1 : xA = e.getX(); yA = e.getY();
            break;
        case 2 : xB = e.getX(); yB = e.getY();
            break;
        case 3 : this.nbPoints=0;
    }
    //redessiner le Carré
    repaint();
}

public void mouseExited(MouseEvent me) { /* rien */ }
public void mouseReleased(MouseEvent me) { /* rien */ }
public void mouseEntered(MouseEvent me) { /* rien */ }
public void mouseClicked(MouseEvent me) { /* rien */ }
} //fin classe Carré

```

Annexe

fillOval

```
public void fillOval(int x, int y, int width, int height)
```

Fills an oval bounded by the specified rectangle with the current color.

Parameters:

x - the x coordinate of the upper left corner of the oval to be filled.
y - the y coordinate of the upper left corner of the oval to be filled.
width - the width of the oval to be filled.
height - the height of the oval to be filled.

drawLine

```
public void drawLine(int x1, int y1, int x2, int y2)
```

Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

Parameters:

x1 - the first point's x coordinate.
y1 - the first point's y coordinate.
x2 - the second point's x coordinate.
y2 - the second point's y coordinate.

drawString

```
public void drawString(String str, int x, int y)
```

Draws the text given by the specified string, using this graphics context's current font and color. The baseline of the leftmost character is at position (x, y) in this graphics context's coordinate system.

Parameters:

str - the string to be drawn.
x - the x coordinate.
y - the y coordinate.