```java
package Ejercicio01;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;

public class U6E01 {

    static Scanner stdin = new Scanner(System.in); //para preguntar la provincia

    public static void main(String[] args) throws IOException  {

        //variables locales
        String provincia;                   //almacena la provincia
        int contador = 1;                   //para contar las lineas que vamos leyendo del fichero csv
        BufferedReader br = null;           //para leer texto usando buffer
        int totalHA=0;                      //almacena el total de HA de los lugares de la provincia
        boolean existeProvincia = false;    //sirve para ver que hay datos de la provincia

        //preguntamos por la provincia
        System.out.print("Espacios Naturales de la Provincia: ");
        provincia = stdin.nextLine();
        stdin.close(); //cerramos la lectura

        //transformamos la cadena

        //eliminamos los espacios
        provincia = provincia.trim();

        //ponemos la cadena en minuscula
        provincia = provincia.toLowerCase();

        try {
            //nos creamos un filereader envuelto en un buffer de lectura
            br = new BufferedReader(new FileReader("espacios1.csv"));

            //leemos la primera linea
            String linea = br.readLine();

            //mientras hayan lineas que leer
            while (linea != null) {
                contador++; //vamos contando lineas leidas
                //construimos un array con los campos de la linea. Separamos los campos con split
                String [] fields = linea.split(";");

                //si hay info de la provincia, la mostramos por pantalla
                if(fields[0].toLowerCase().contains(provincia) && fields.length == 5) {
                    existeProvincia = true;
```

```
                    System.out.print("Espacio: ");
                    System.out.printf("%35s", fields[1]);
                    System.out.print("\tTipo: ");
                    System.out.printf("%35s", fields[3]);
                    System.out.print("\tSuperficie(HA): ");
                    System.out.print(fields[4]);
                    System.out.println();
                    totalHA = totalHA + Integer.parseInt(fields[4]); //calculo de HA

                } //mostramos las lineas que no tienen el numero exacto de campos
                else if (fields.length != 5) System.out.println("  LINEA " + contador + " INCORRECTA " + linea);

                 //leemos la siguiente linea
                linea = br.readLine();
            }

        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        finally {  //cerramos cosas
           br.close();
         }

        //si hemos encontrado datos de la provincia, mostramos el total de hectareas
        if(existeProvincia) System.out.println("Total de superficie: " + totalHA + " HA");
        else System.out.println("No hemos encontrado nada sobre " + provincia);

      }//fin main

}//fin clase
```

```java
package Ejercicio02;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Calendar;

import javax.swing.JFileChooser;

public class U6E02A {


    public static void main(String[] args) throws IOException {
        //variables globales
        File file = leerFichero();                          //objeto File. Llamamos al metodo para seleccionar el fichero
        float[] cantidad = new float[256];                  //para almacenar las veces que aparece cada uno de los 256 bytes
        FileInputStream st = null;                                  //objeto FileInputStream
        BufferedInputStream bis = null;                         //objeto BufferedInputStream
        int bytes=0;                                        //almacena el total de bytes
        long tiempo;                                        //almacena el tiempo transcurrido
        int pos = 0;

        //rellenamos el array con ceros
        for(int i = 0; i < cantidad.length; i++) {
            cantidad[i] = 0;
        }

        //cogemos el tiempo de inicio
        tiempo= System.currentTimeMillis();
        System.out.println(tiempo);


        try {

            //creamos el FileInputStream
            st = new FileInputStream(file);

            //lo encapsulamos en un bufferedInputStream
            bis = new BufferedInputStream(st);

            //calculamos cuantos bytes tiene el archivo seleccionado
            bytes = bis.available();

            //leemos el primer byte
            int valor=bis.read();

            //mientras hay bytes que leer almacenamos en cada posicion del array las veces que aparece
            while(valor != -1) {
                cantidad[valor]++;
```

```
52                  valor = bis.read();
53              }
54
55
56          } catch (FileNotFoundException e) {
57              System.out.println(e.getMessage());
58          } catch (IOException e) {
59              System.out.println(e.getMessage());
60          }
61
62          finally {  //cerramos cosas
63              st.close();
64              bis.close();
65           }
66
67          //calculamos el histograma
68          for(int i = 0; i < cantidad.length; i++)cantidad[i] = cantidad[i]/bytes;
69
70          //paramos el cromo
71          tiempo = System.currentTimeMillis() - tiempo;
72
73
74          //mostramos el nombre del fichero, su peso y el tiempo de calculo del histograma
75          System.out.println("Fichero: " + file.getName());
76          System.out.print("Peso: " + bytes + " bytes. ");
77          System.out.printf("Proceso realizado en %5.4f segundos.",(float)tiempo/1000);
78          System.out.println();
79
80          //imprimimos el histograma
81          for(int i = 0; i < 32; i++) {
82              pos = i;
83              for(int j = 0; j < 8;j++) {
84                  System.out.printf("h[%3d]: %7.6f ",pos,cantidad[pos]);
85                  pos = pos + 32;
86              }
87              System.out.println();
88          }
89
90
91
92      }
93
94      //metodo para seleccionar un fichero
95      public static File leerFichero() {
96          //variables
97          File archivo;
98
99          //creamos el objeto
100         JFileChooser fd = new JFileChooser();
101
102         //mostramos por pantalla
```

```
103             fd.setDialogTitle("Selecciona el fichero a leer");
104 
105             fd.setSelectedFile(null); //no hay ninguno seleccionado
106 
107             int opcion = fd.showOpenDialog(null);
108 
109             if (opcion != JFileChooser.APPROVE_OPTION ) {
110                 archivo = null;
111                 return archivo;
112             }
113 
114             archivo = fd.getSelectedFile();
115             return archivo;
116 
117         }
118 
119     }
120 
```

```java
package Ejercicio02;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import javax.swing.JFileChooser;

public class U6E02B {


    public static void main(String[] args) throws IOException {
        //variables globales
        File file = leerFichero();                          //objeto File. Llamamos al metodo para seleccionar el fichero
        float[] cantidad = new float[256];                  //para almacenar las veces que aparece cada uno de los 256 bytes
        FileInputStream st=null;                                    //objeto FileInputStream
        BufferedInputStream bis=null;                           //objeto BufferedInputStream
        int bytes=0;                                        //almacena el total de bytes
        long tiempo;                                        //almacena el tiempo transcurrido
        int pos = 0;


        //rellenamos el array con ceros
        for(int i = 0; i < cantidad.length; i++) {
            cantidad[i] = 0;
        }

        //cogemos el tiempo de inicio
        tiempo= System.currentTimeMillis();


        try {

            //creamos el FileInputStream
            st = new FileInputStream(file);

            //lo encapsulamos en un bufferedInputStream
            bis = new BufferedInputStream(st);

            //calculamos cuantos bytes tiene el archivo seleccionado
            bytes = bis.available();

            //creamos el buffer
            byte [] buffer = new byte[64];

            //leemos el primer byte
            int valor=0;

            //mientras hay bytes que leer almacenamos en cada posicion del array las veces que aparece
```

```
52              while(0 < (valor=bis.read(buffer))) {
53                  for(int i = 0; i < valor; i++) {
54                      if (buffer[i] < 0) pos = buffer[i]+256;
55                      else pos = buffer[i];
56                      cantidad[pos]++;
57                  }
58              }
59
60
61
62          } catch (FileNotFoundException e) {
63              System.out.println(e.getMessage());
64          } catch (IOException e) {
65              System.out.println(e.getMessage());
66          }
67
68          finally {  //cerramos cosas
69              st.close();
70              bis.close();
71           }
72
73          //calculamos el histograma
74          for(int i = 0; i < cantidad.length; i++)cantidad[i] = cantidad[i]/bytes;
75
76          //paramos el cromo
77          tiempo = System.currentTimeMillis() - tiempo;
78
79
80          //mostramos el nombre del fichero, su peso y el tiempo de calculo del histograma
81          System.out.println("Fichero: " + file.getName());
82          System.out.print("Peso: " + bytes + " bytes. ");
83          System.out.printf("Proceso realizado en %5.4f segundos.",(float)tiempo/1000);
84          System.out.println();
85
86          //imprimimos el histograma
87          for(int i = 0; i < 32; i++) {
88              pos = i;
89              for(int j = 0; j < 8;j++) {
90                  System.out.printf("h[%3d]: %7.6f ",pos,cantidad[pos]);
91                  pos = pos + 32;
92              }
93              System.out.println();
94          }
95
96      }
97
98      //metodo para seleccionar un fichero
99      public static File leerFichero() {
100         //variables
101         File archivo;
102
```

```
103             //creamos el objeto
104             JFileChooser fd = new JFileChooser();
105 
106             //mostramos por pantalla
107             fd.setDialogTitle("Selecciona el fichero a leer");
108 
109             fd.setSelectedFile(null); //no hay ninguno seleccionado
110 
111             int opcion = fd.showOpenDialog(null);
112 
113             if (opcion != JFileChooser.APPROVE_OPTION ) {
114                 archivo = null;
115                 return archivo;
116             }
117 
118             archivo = fd.getSelectedFile();
119             return archivo;
120 
121         }
122 
123     }
124 
125 
```

```java
package Ejercicio02;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import javax.swing.JFileChooser;

public class U6E02C {


    public static void main(String[] args) throws IOException {
        //variables globales
        File file = leerFichero();                          //objeto File. Llamamos al metodo para seleccionar el fichero
        float[] cantidad = new float[256];                  //para almacenar las veces que aparece cada uno de los 256 bytes
        FileInputStream st=null;                                    //objeto FileInputStream
        BufferedInputStream bis=null;                           //objeto BufferedInputStream
        int bytes=0;                                        //almacena el total de bytes
        long tiempo;                                        //almacena el tiempo transcurrido
        int pos = 0;
        double entropia=0;


        //rellenamos el array con ceros
        for(int i = 0; i < cantidad.length; i++) {
            cantidad[i] = 0;
        }

        //cogemos el tiempo de inicio
        tiempo= System.currentTimeMillis();


        try {

            //creamos el FileInputStream
            st = new FileInputStream(file);

            //lo encapsulamos en un bufferedInputStream
            bis = new BufferedInputStream(st);

            //calculamos cuantos bytes tiene el archivo seleccionado
            bytes = bis.available();

            //creamos el buffer
            byte [] buffer = new byte[64];

            //leemos el primer byte
            int valor=0;

```

```
52              //mientras hay bytes que leer almacenamos en cada posicion del array las veces que aparece
53              while(0 < (valor=bis.read(buffer))) {
54                  for(int i = 0; i < valor; i++) {
55                      if (buffer[i] < 0) pos = buffer[i]+256;
56                      else pos = buffer[i];
57                      cantidad[pos]++;
58                  }
59              }
60
61          } catch (FileNotFoundException e) {
62              System.out.println(e.getMessage());
63          } catch (IOException e) {
64              System.out.println(e.getMessage());
65          }
66
67          finally {  //cerramos cosas
68              st.close();
69              bis.close();
70           }
71
72          //calculamos el histograma
73          for(int i = 0; i < cantidad.length; i++)cantidad[i] = cantidad[i]/bytes;
74
75          //calculamos la entropia
76          for(int i = 0; i < cantidad.length; i++) {
77              entropia = entropia + (cantidad[i]*(Math.log(1/cantidad[i])/Math.log(2)));
78          }
79
80          //paramos el cromo
81          tiempo = System.currentTimeMillis() - tiempo;
82
83
84          //mostramos el nombre del fichero, su peso y el tiempo de calculo del histograma
85          System.out.println("Fichero: " + file.getName());
86          System.out.print("Peso: " + bytes + " bytes. ");
87          System.out.printf("Proceso realizado en %5.4f segundos.",(float)tiempo/1000);
88          System.out.println();
89          System.out.printf("La entropia del fichero es: %7.6f bits", entropia);
90          System.out.println();
91          System.out.printf("La redundancia es %7.6f bits", 8-entropia);
92          System.out.println();
93          if(8-entropia < 0.1) System.out.println("Fichero comrprimido o aleatorio");
94          else if(8-entropia >= 0.1 && 8-entropia <= 3.5) System.out.println("Fichero parcialmente comprimido");
95          else System.out.println("Fichero no comprimido");
96
97
98
99
100     }
101
102     //metodo para seleccionar un fichero
```

```java
    public static File leerFichero() {
        //variables
        File archivo;

        //creamos el objeto
        JFileChooser fd = new JFileChooser();

        //mostramos por pantalla
        fd.setDialogTitle("Selecciona el fichero a leer");

        fd.setSelectedFile(null); //no hay ninguno seleccionado

        int opcion = fd.showOpenDialog(null);

        if (opcion != JFileChooser.APPROVE_OPTION ) {
            archivo = null;
            return archivo;
        }

        archivo = fd.getSelectedFile();
        return archivo;

    }

}
```

```java
package Ejercicio03;

import java.io.File;
import java.io.FileFilter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.attribute.UserPrincipal;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;


public class U6E03 {
    //variables globales
    public static String extension;     //almacena una extension de archivo

    public static FileFilter filter = new FileFilter() {
        public boolean accept(File file) {
                String tmp = file.getName().toLowerCase();
                if (tmp.endsWith(extension)){
                    return true;
                }
                    return false;
                }
            };



    public static void main(String[] args) throws IOException {

        //variables locales
        String carpetaActual;   //almacena la carpeta actual
        String carpeta;         //almacena una carpeta


        Scanner stdin = new Scanner(System.in); //para la entrada de datos

        boolean flag=false;

        File e = null;

        //carpeta actual
        carpetaActual = System.getProperty("user.dir");
        System.out.println("Carpeta actual: " + carpetaActual);

        //creamos el objeto File
        File d = new File(carpetaActual);
        File[] listado = d.listFiles();

        //preguntamos por la carpeta a examinar
```

```
52          System.out.print("Carpeta a examinar: ");
53          carpeta = stdin.nextLine();
54
55          //preguntamos por la extension
56          System.out.print("Explorar ficheros que acaban en: ");
57          extension = stdin.nextLine();
58
59          //vemos si existe la carpeta en el directorio actual
60          for(int i = 0; i < listado.length; i++) {
61              if(listado[i].isDirectory()) {
62                  if(listado[i].getName().equals(carpeta)) flag = true;
63                  e = new File(listado[i].getAbsolutePath());
64              }
65          }
66
67          //si la carpeta a examinar no existe
68          if (!flag) System.out.print("La carpeta no existe");
69          //si existe
70          else {
71              listado = e.listFiles();
72              //imprimimos la primera linea
73              System.out.println("[" + carpeta + "]");
74
75              //llamada a funcion
76              recorreFichero(listado,2);
77
78              //vemos si hay ficheros con la extension
79              File[] listaFicheros = e.listFiles(filter);
80              //si hay archivos con la extension
81              if(listaFicheros.length > 0) {
82                  for(int i = 0; i < listaFicheros.length; i++) {
83                      //imprimir el nombre del fichero
84                      System.out.print(listaFicheros[i].getName()+ " ");
85                      System.out.print(listaFicheros[i].length() + " bytes ");
86                      // imprimir la fecha de la ultima modificacion
87                      long millisec = listaFicheros[i].lastModified();
88                      Date dt = new Date(millisec);
89                      System.out.print(" Ul.Mod ");
90                      System.out.print(new SimpleDateFormat("dd-MM-yyyy hh:mm:ss ").format(dt));
91                      //imprimir los permisos
92                      System.out.print("Permisos (RWX):");
93                      if(listaFicheros[i].canRead())System.out.print("R");
94                      else System.out.print("-");
95                      if(listaFicheros[i].canWrite())System.out.print("W");
96                      else System.out.print("-");
97                      if(listaFicheros[i].canExecute())System.out.print("X");
98                      else System.out.print("-");
99                      //imprimir usuario
100                     Path p = listaFicheros[i].toPath();
101                     UserPrincipal userPrincipal = null;
102                     try {
```

```
                        userPrincipal = Files.getOwner(p);
                    } catch (IOException f) {
                        // TODO Bloque catch generado automáticamente
                        f.printStackTrace();
                    }
                    System.out.print(" Propietario: " + userPrincipal);

                }
            }

        }

    }

    public static void recorreFichero(File[] listado, int espacios) {
        File[] listadoTemporal = null;
        //corremos el directorio
        if(listado.length > 0) {
            for(int i = 0; i < listado.length; i++) {
                if(listado[i].isDirectory()) {
                    for(int e = 0; e < espacios; e++) System.out.print(" ");
                    System.out.println("[" + listado[i].getName() + "]");
                    File temp = new File(listado[i].getAbsolutePath());
                    listadoTemporal = temp.listFiles();
                    File[] archivosTemporal = temp.listFiles(filter);
                    if(archivosTemporal.length > 0) {
                        //hay ficheros con la extension
                        for(int j = 0; j < archivosTemporal.length; j++) {
                            for(int e = 0; e < espacios; e++) System.out.print(" ");
                            //imprimir nombre
                            System.out.print(archivosTemporal[j].getName()+ " ");
                            //imprimir peso
                            System.out.print(archivosTemporal[j].length() + " bytes ");
                            // imprimir fecha modificacion
                            long millisec = archivosTemporal[j].lastModified();
                            Date dt = new Date(millisec);
                            System.out.print(" Ul.Mod ");
                            System.out.print(new SimpleDateFormat("dd-MM-yyyy hh:mm:ss ").format(dt));
                            //imprimir permisos
                            System.out.print("Permisos (RWX):");
                            if(archivosTemporal[j].canRead())System.out.print("R");
                            else System.out.print("-");
                            if(archivosTemporal[j].canWrite())System.out.print("W");
                            else System.out.print("-");
                            if(archivosTemporal[j].canExecute())System.out.print("X");
                            else System.out.print("-");
                            //imprimir usuario
                            Path p = archivosTemporal[j].toPath();
                            UserPrincipal userPrincipal = null;
                            try {
                                userPrincipal = Files.getOwner(p);
```

```
154                             } catch (IOException e) {
155                                 // TODO Bloque catch generado automáticamente
156                                 e.printStackTrace();
157                             }
158                             System.out.print(" Propietario: " + userPrincipal);
159                             System.out.print("\n");
160                         }
161                     }
162                     recorreFichero(listadoTemporal,espacios+2);
163 
164                 }
165 
166             }
167         }
168     }
169 
170 }
171 
```

```java
package Ejercicio04;

import java.io.BufferedReader;
import java.io.Console;
import java.io.File;
import java.io.FileFilter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class U06E04 {

    //lo usaremos para filtrar archivos gps
    public static FileFilter filter = new FileFilter() {
        public boolean accept(File file) {
                String tmp = file.getName().toLowerCase();
                if (tmp.endsWith(".gps")){
                    return true;
                }
                    return false;
                }
            };

    public static void main(String[] args) {

        //variables locales
        String fichero; //almacena el fichero
        String origen ="origen";    //almacena la ciudad de origen
        String destino ="destino";  //almacena la ciudad de destino
        String carpetaActual;   //almacena la carpeta actual

        Scanner stdin = new Scanner(System.in); //entrada por teclado

        boolean flag=false;

        //preguntamos por el fichero, en este caso suponemos que lo tenemos en el directorio actual
        System.out.print("Fichero (.GPS): ");
        fichero = stdin.nextLine();

        //suponiendo que vamos a trabajar desde el actual directorio, lo almacenamos
        carpetaActual = System.getProperty("user.dir");

        //creamos el objeto File
        File d = new File(carpetaActual);

        //vemos si hay ficheros con la extension gps dentro del directorio
        File[] listaFicheros = d.listFiles(filter);
```

```
52
53          //si hay archivos gps, buscamos el que queremos
54          if(listaFicheros.length > 0) {
55              for(int i = 0; i < listaFicheros.length; i++) {
56                  if(listaFicheros[i].getName().toLowerCase().equals(fichero.toLowerCase())) flag=true;
57              }
58          }
59
60          //si no hemos encotrado nada finalizamos el programa
61          if(flag == false) {
62              System.out.println("No hemos encontrado el fichero " + fichero);
63              System.exit(0);
64          }
65
66          //si existe el fichero continuamos con el programa
67
68          //mientras el origen y el destino no esten vacios
69          while(!origen.isEmpty() && !destino.isEmpty()) {
70
71              //preguntamos por la ciudad de origen
72              System.out.print("Origen: ");
73              origen = stdin.nextLine();
74
75              //preguntamos por la ciudad de destino
76              System.out.print("Destino: ");
77              destino = stdin.nextLine();
78
79              //si origen y destino no estan vacias
80              if(!origen.isEmpty() && !destino.isEmpty()) {
81                  Grafo unGrafo = new Grafo(fichero);
82                  String cadena = unGrafo.caminoMasCorto(origen, destino);
83                  System.out.println(cadena);
84              }//fin del if
85
86          }//fin del while
87
88      }//fin del main
89
90  }//fin de la clase
91
92  //*****************************************clase Grafo*********************************************
93  class Grafo{
94
95      public static String[] datos;   //para almacenar los datos del fichero
96      public static int contador = 0;     //las posiciones del array de datos
97      public static int lineasTotales = 0; //lineas totales del fichero
98
99      //variables miembro
100     private int nV; //numero de vertices
101     private int nA; //numero de aristas
102     private int[][] tA; //tabla de adyacencias
```

```
103     private String[] nombreVertice; //nombre de los vertices
104
105     //constructor
106     public Grafo(String nomFichero) {
107
108         //declaramos una variable BufferedReader
109         BufferedReader br = null;
110
111         try {
112             //crear un objeto BufferedReader al que se le pasa
113             //un objeto FileReader con el nombre del fichero
114             br = new BufferedReader(new FileReader(nomFichero));
115
116
117             //leer la primera linea, guardando en un string
118             String texto = br.readLine();
119
120             //repetir mientras no se llegue al final del fichero
121             while(texto != null) {
122
123                 //contamos las lineas del fichero y leemos
124                 contador++;
125                 texto = br.readLine();
126
127
128             }//fin del while
129
130             //almacenamos las lineas totales leidas
131             lineasTotales = contador;
132
133             //conocido el numero de lineas construimos el array de datos
134             datos = new String[contador];
135
136             //ponemos el contador a cero
137             contador = 0;
138
139             //creamos un bufferedReader
140             //con el metodo br.mark y br.reset deberia haber reiniciado el buffer anterior
141             //pero me da error al aplicarlos
142             BufferedReader br2 = new BufferedReader(new FileReader(nomFichero));
143
144             //volvemos a correr el fichero rellenando de nuevo los datos
145             texto = br2.readLine();
146             while(texto != null) {
147
148                 datos[contador]=texto.trim();
149                 contador++;
150                 texto = br2.readLine();
151
152
153             } //fin del while
```

```
154
155             //ahora vamos rellenando las variables miembro
156             contador=0;
157
158             //la primera linea indica el numero de indices
159             nV = Integer.parseInt(datos[0]);
160             contador++;
161
162             //la segunda linea indica el numero de aristas
163             nA = Integer.parseInt(datos[1]);
164             contador++;
165
166             //creamos el array de vertices
167             nombreVertice = new String[nV];
168
169             //creamos el array de adyacentes
170             tA = new int[nV][nV];
171
172             //rellenamos el array de vertices
173             for(int i = 2; i < nV+2; i++) {
174                 int pos = datos[i].indexOf(" "); //el espacio separa el nº vertice del nombre de vertice
175                 nombreVertice[i-2] = datos[i].substring(pos+1);
176                 contador++;
177             }
178
179             //rellenamos la tabla de adyacentes
180             for(int i = contador; i < lineasTotales; i++) {
181
182                 //como son 3 campos, cogemos la primera y ultima posicion del separador
183                 int pos1 = datos[i].indexOf(" ");
184                 int pos2 = datos[i].lastIndexOf(" ");
185
186                 //volcamos cada campo en variables
187                 int temp1 = Integer.parseInt(datos[i].substring(0, pos1));
188                 int temp2 = Integer.parseInt(datos[i].substring(pos1+1, pos2));
189
190                 //los valores no pueden ser negativos o mayores que el numero de vertices
191                 if(temp1 >= nV || temp1 < 0 || temp2 >= nV || temp2 < 0) {
192                     throw new IllegalArgumentException ("Error formato en " + nomFichero + " linea " + i);
193                 }
194
195                 //el primer campo no puede igual que el segundo campo
196                 if(temp1 == temp2) {
197                     throw new IllegalArgumentException ("Error formato en " + nomFichero + " linea " + i);
198                 }
199
200                 //si esta todo correcto
201                 tA[temp1][temp2] = Integer.parseInt(datos[i].substring(pos2+1));
202                 tA[temp2][temp1] = Integer.parseInt(datos[i].substring(pos2+1));
203
204             }
```

```
205             //tratamiento de errores
206             } catch (FileNotFoundException e) {
207                 System.out.println("Error: Fichero no encontrado");
208                 System.out.println(e.getMessage());
209             } catch (Exception e) {
210                 System.out.println("Error de lectura del fichero");
211                 System.out.println(e.getMessage());
212             }
213             finally {
214                 try {
215                     if(br != null)
216                         br.close();
217                 } catch (Exception e) {
218                     System.out.println("Error al cerrar el fichero");
219                     System.out.println(e.getMessage());
220                 }
221             }
222 
223         } //fin del constructor
224 
225         //metodos
226 
227         public String caminoMasCorto(String origen, String destino) {
228 
229             //variables locales
230             boolean cOrigen = false;
231             boolean cDestino = false;
232             String cadena = "";
233             int d=0,o=0; //numero de vertice de origen y destino
234 
235             //comprobamos que la ciudad de origen estan dentro de los datos
236             for(int i = 0; i < nombreVertice.length; i++) {
237                 if(nombreVertice[i].toLowerCase().equals(origen.toLowerCase())) {
238                     cOrigen = true;
239                     o = i;
240                 }
241                 if(nombreVertice[i].toLowerCase().equals(destino.toLowerCase())) {
242                     cDestino = true;
243                     d = i;
244                 }
245             }
246 
247             if(cOrigen == false && cDestino == false) cadena = cadena + "ciudad de origen " + origen + " y destino " + destino +
            " no encontradas.";
248             else if(cOrigen == false) cadena = cadena + "ciudad de origen " + origen +  " no encontrada.";
249             else if(cDestino == false) cadena = cadena + "ciudad de destino " + destino +  " no encontrada.";
250             else if(origen.equals(destino)) cadena = cadena + origen +"(0) -> " + origen;
251             //llamada al metodo
252             else cadena = Dijsktra(o,d);
253 
254             return cadena;
```

```
255       } //fin del metodo
256
257       //metodo dijsktra
258       private String Dijsktra(int ori, int des) {
259           //variables locales
260           ArrayList S = new ArrayList();
261           int[] D = new int[nV];
262           int[] P = new int[nV];
263           int menos = 0;
264           //int pos = 0;
265           int tramo;
266           int distancia=0;
267           int w=-1;
268           int v=-1;
269           String camino = "";
270
271           //inicialmente S contiene el vertice origen(x)
272           S.add(ori);
273
274           //para cada vertice
275           for(int i = 0; i < nV; i++) {
276               //si i no es el origen
277               if(i != ori) {
278                   //si hay arista entre i y origen
279                   if(tA[ori][i] > 0) D[i] = tA[ori][i];
280                   //si no hay arista ponemos el maximo valor entero
281                   else D[i] = Integer.MAX_VALUE;
282               }
283           } //fin del for
284
285           //explorar el grafo
286           for(int i = 0; i < nV; i++) {
287               if(S.size() == nV) break;
288               //corremos el array de distancias, buscando la posicion que no esté en S y que tenga la menor distancia
289               menos = Integer.MAX_VALUE; //inicializamos la variable para ver que posicion tiene menos
290               for(int j = 0; j < nV; j++) {
291                   //comprobamos que j no está en S
292                   if(!S.contains(j)) {
293                   //comprobamos que tenga el valor mas bajo
294                       if(D[j] < menos) {
295                           menos = D[j];
296                           w = j; //asignamos a v la posicion en D[i]
297                       }
298                   }
299               }
300               //añadimos el vertice w al conjunto S
301               S.add(w);
302
303               //volvemos a correr V-S, o sea todos los vertices que no esten en S
304               for(int j = 0; j < nV; j++) {
305                   //comprobamos que j no está en S
```

```
306                 if(!S.contains(j)) {
307                 //comprobamos que hay una arista w - j
308                     if(tA[w][j] > 0) {
309                         v = j;
310                         if (D[v] + tA[w][v] < D[v]) {
311                             D[v] = D[w] + tA[w][v];
312                             P[v] = w;
313                         }
314                     }
315                 }
316             }
317
318
319
320         } //fin exploracion del grafo
321
322         //test de arrays
323         /*for(int i = 0; i<nV; i++) System.out.print(i + ":" + D[i] + " ");
324         System.out.println();
325         for(int i = 0; i<nV; i++) System.out.print(i + ":" + P[i] + " ");
326         System.out.println();*/
327
328         distancia = 0;
329
330         camino = nombreVertice[des];
331
332
333         v = des;
334
335
336         while (v != ori){
337             //en el vertice previo al destino se me va a 0, asi que añado esto
338             if(P[v]==0) {
339                 tramo = tA[v][ori];
340                 distancia = distancia + tramo;
341                 camino = nombreVertice[ori] + " (+" + tramo + ") -> " + camino;
342                 v=ori;
343             } else {
344                 tramo = tA[v][P[v]];
345                 distancia = distancia + tramo;
346                 camino = nombreVertice[P[v]] + " (+" + tramo + ") ->" + camino;
347                 v = P[v];
348             }
349         }
350
351         camino ="  " + camino + " Total: " + distancia;
352         return camino;
353
354
355
356     } //fin de dijsktra
```

```
357
358
359
360   } //fin de la clase grafo
361
```

```java
package Ejercicio05;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStream;

public class U06E05 {

    public static void main(String[] args) throws IOException {

        //variables locales
        //creamos variables finales que van a ser cada una de las letras del ADN
        final int a = 0;
        final int g = 1;
        final int t = 2;
        final int c = 3;

        int contador = 0;   //esta variable la uso para ir formando grupos de 4 letras
        int binario = 0;    //esta variable almacena el numero binario que vamos a ir cosntruyendo
        int[] caracteres = new int[4]; //este array ira almacenando los caracteres leidos, hasta formar un bloque de 4
        int ultimoByte = 0; //almacenamos el valor del ultimo byte cuando comprimimos.

        BufferedReader br = null; //declaramos una variable BufferedReader
        BufferedWriter bw = null; //declaramos una variable BufferedWriter

        OutputStream salida = null; //declaramos una variable outputstream
        FileInputStream st = null;  //declaramos una variable fileinputstream
        BufferedInputStream bis = null; //declaramos un bufferedInputStream
        FileWriter salida2 = null;

        String sinComprimir = "prueba.adn";
        String comprimido = "prueba.adc";

        //pendiente de pedir los ficheros de entrada y salida al usuario
        //en principio vamos al grano y vamos a trabajar con ficheros especificos

        //comprimir

        try {
            //crear un objeto BufferedReader al que se le pasa
            //un objeto FileReader con el nombre del fichero
            br = new BufferedReader(new FileReader(sinComprimir));
            File ficheroBinario = new File(comprimido);
```

```
52                  salida = new FileOutputStream(ficheroBinario);
53
54                  int charRead = br.read();
55
56                  //Ponemos el array a -1
57                  for(int i = 0; i <4; i++) {
58                      caracteres[i] = -1;
59                  }
60
61                  while (charRead != -1) {
62
63
64                      if(contador < caracteres.length) {
65                          //System.out.println(charRead);
66                          if(charRead == 65 || charRead == 97) caracteres[contador] = a;
67                          if(charRead == 71 || charRead == 103) caracteres[contador] = g;
68                          if(charRead == 84 || charRead == 116) caracteres[contador] = t;
69                          if(charRead == 67 || charRead == 99) caracteres[contador] = c;
70                          contador++;
71                          if(contador < caracteres.length) charRead = br.read();
72
73                      } else {
74                          //primera letra
75                          binario = caracteres[0];
76                          //segunda letra
77                          binario = binario << 2;
78                          binario = binario + caracteres[1];
79                          //tercera letra
80                          binario = binario << 2;
81                          binario = binario + caracteres[2];
82                          //ultima letra
83                          binario = binario << 2;
84                          binario = binario + caracteres[3];
85                          //sacamos la info
86                          salida.write(binario);
87                          //Ponemos el array a -1
88                          for(int i = 0; i <4; i++) {
89                              caracteres[i] = -1;
90                          }
91                          binario = 0;
92                          contador=0;
93                          charRead = br.read();
94                      }
95
96                      if (charRead == -1) {
97                          binario = 0;
98                          //primera letra
99                          if(caracteres[0] != -1) binario = binario + caracteres[0];
100                         else binario = binario + a; //
101
102                         //segunda letra
```

```
103                     if(caracteres[1] != -1) {
104                         binario = binario << 2;
105                         binario = binario + caracteres[1];
106                     }
107                     else {
108                         binario = binario << 2;
109                         binario = binario + a;
110                     }
111                     //tercera letra
112                     if(caracteres[2] != -1) {
113                         binario = binario << 2;
114                         binario = binario + caracteres[2];
115                     }
116                     else {
117                         binario = binario << 2;
118                         binario = binario + a;
119                     }
120                     //ultima letra
121                     if(caracteres[3] != -1) {
122                         binario = binario << 2;
123                         binario = binario + caracteres[3];
124                     }
125                     else {
126                         binario = binario << 2;
127                         binario = binario + a;
128                     }
129                     //sacamos la info
130                     //System.out.println(binario);
131                     salida.write(binario);
132 
133                     //añadimos el ultimo byte
134                     if (contador==0) {
135                         salida.write(4);
136                         ultimoByte = 4;
137                     }
138                     if (contador==1) {
139                         salida.write(1);
140                         ultimoByte = 1;
141                     }
142                     if (contador==2) {
143                         salida.write(2);
144                         ultimoByte = 2;
145                     }
146                     if (contador==3) {
147                         salida.write(3);
148                         ultimoByte = 3;
149                     }
150                 }
151 
152             }
153             //tratamiento de errores
```

```
154         } catch (FileNotFoundException e) {
155             System.out.println("Error: Fichero no encontrado");
156             System.out.println(e.getMessage());
157         } catch (Exception e) {
158             System.out.println("Error de lectura del fichero");
159             System.out.println(e.getMessage());
160         }
161             finally {
162                 try {
163                 if(br != null)
164                         br.close();
165             } catch (Exception e) {
166                     System.out.println("Error al cerrar el fichero");
167                     System.out.println(e.getMessage());
168             }
169         }
170 
171         //descomprimir
172 
173 try {
174 
175             //creamos el FileInputStream
176             st = new FileInputStream(comprimido);
177             //lo encapsulamos en un bufferedInputStream
178             bis = new BufferedInputStream(st);
179 
180             //creamso el fichero de salida
181             File archivo2 = new File("descomprimido.adn");
182             salida2 = new FileWriter(archivo2,true);
183 
184             //miramos cuantos bytes tiene el fichero comprimido
185             int totalBytes = st.available();
186             int cuentaBytes = 0; //para contar las veces que leemos 1 byte en el fichero comprimido
187 
188             //leemos el primer byte
189             int valor=bis.read();
190 
191             //hemos leido un byte, incrementamos el contador
192             cuentaBytes++;
193 
194             //mientras hay bytes que leer almacenamos en cada posicion del array las veces que aparece
195             while(valor != -1) {
196                 //vemos si hemos leido el penultimo byte
197                 if(cuentaBytes != totalBytes-1) {
198                     //pasamos el primer byte a binario
199                     String numBinario = Integer.toBinaryString(valor);
200                     //como los primeros ceros se ignoran, los ponemos nosotros
201                     while(numBinario.length()<8) {
202                         numBinario = "0" + numBinario;
203                     }
204                     //vamos leyendo los bits y trasnformandolos a caracter
```

```
                    for (int i = 0; i < 8; i=i+2) {
                        String trozo = numBinario.substring(i, i+2);
                        //System.out.println(trozo);
                        if(trozo.equals("00")) salida2.write("A");
                        if(trozo.equals("01")) salida2.write("G");
                        if(trozo.equals("10")) salida2.write("T");
                        if(trozo.equals("11")) salida2.write("C");
                    }

                    valor=bis.read();
                    cuentaBytes++;
                } else {
                    //pasamos el byte a binario
                    String numBinario = Integer.toBinaryString(valor);
                    //como los primeros ceros se ignoran, los ponemos nosotros
                    while(numBinario.length()<8) {
                        numBinario = "0" + numBinario;
                    }
                    //vamos leyendo los bits y trasnformandolos a caracter, pero solo los que nos marque el ultimo byte
                    ultimoByte = ultimoByte * 2;
                    if(ultimoByte == 8) ultimoByte = 0;
                    for (int i = 0; i < ultimoByte; i=i+2) {
                        String trozo = numBinario.substring(i, i+2);
                        //System.out.println(trozo);
                        if(trozo.equals("00")) salida2.write("A");
                        if(trozo.equals("01")) salida2.write("G");
                        if(trozo.equals("10")) salida2.write("T");
                        if(trozo.equals("11")) salida2.write("C");
                    }
                    valor=bis.read(); //leemos el ultimo byte, de control
                    valor=bis.read(); //volvemos a leer para forzar la salida
                    cuentaBytes++;
                }
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            st.close();
            bis.close();
            salida2.close();
        }

    }

}
```

```java
package Ejercicio06;

import java.util.Scanner;
import java.util.regex.*;


import java.io.File;
import java.io.FileNotFoundException;
class U06EO6 {
    public static void main(String[] args) throws FileNotFoundException{

        Pattern delimitador = Pattern.compile("<[^>\\n]+>");
        try {
            // inicializamos el objeto scaner
            Scanner scan = new Scanner(new File("index.html"));
            //inicializamos el delimitador
            scan.useDelimiter(delimitador);
            int contador = 0;
            //imrpimimos los tokens
            while(scan.hasNext()){
                String st = scan.next().trim();
                if(st.length()>0) System.out.println(st);
            }
            scan.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: Fichero no encontrado");
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("Error de lectura del fichero");
            System.out.println(e.getMessage());
        }

       }
    }


```