

Ejercicios/Ejercicio1punto1/src/ejercicio1punto1/Ejercicio1punto1.java

```
1 package ejercicio1punto1;
2
3 import java.io.File;
4 import java.io.IOException;
5
6 public class Ejercicio1punto1 {
7
8     //Crea un fichero temporal utilizando el método createTempFile, sin
9     //especificar el directorio donde se crea.
10    //Escribe la ruta absoluta o completa para el fichero, una vez creado.
11    //Verifica que el fichero está en ese directorio.
12
13    public static void main(String[] args) {
14
15        // Crear un fichero temporal, sin especificar la ruta.
16        /*
17        try {
18            File temp = File.createTempFile("a.txt", "");
19            System.out.println("Directorio del fichero: " + temp.
20                getAbsolutePath());
21            temp.delete();
22        } catch (IOException ex) {
23            ex.printStackTrace();
24        }
25        */
26
27        try {
28            File directory = new File("C:\\Users\\Sergio\\Desktop\\
29                Estudios\\Acceso-a-Datos\\Ejercicios de repaso\\
30                Ejercicio1punto1");
31            File temp = File.createTempFile("hola", ".txt", directory);
32            System.out.println("Directorio del fichero: " + temp.
33                getAbsolutePath());
34
35            if(directory.exists()){
36                if(temp.getParentFile().equals(directory)){
37                    System.out.println("El fichero esta en el directorio
38                        correcto.");
39                } else {
40                    System.out.println("El fichero no esta en el
41                        directorio correcto.");
42                }
43                System.out.println("El directorio existe.");
44            } else {
45                System.out.println("El directorio no existe.");
46            }
47            temp.delete();
48        } catch (IOException ex) {
49            ex.printStackTrace();
50        }
51    }
52 }
```

No paths relativos, y menos que solo existan en tu ordenador. Si no se especifica se crear en directorio para fich temporales



45
46 }

Ejercicios/Ejercicio1punto10/src/ejercicio1punto10/Ejercicio1punto10.java

```
1 package ejercicio1punto10;
2
3 import java.io.FileReader;
4 import java.io.FileWriter;
5 import java.io.IOException;
6
7 public class Ejercicio1punto10 {
8
9     /**
10      * Averigua la codificación de texto con la que el programa creado
11      * para
12      * ambas actividades anteriores ha generado el fichero. En Linux
13      * puedes
14      * utilizar el comando file. Tanto en Linux como en Windows se puede
15      * abrir
16      * el fichero con un editor de texto y seleccionar la opción Guardar
17      * como,
18      * que normalmente muestra la codificación utilizada para el fichero y
19      * permite seleccionar una distinta para guardarlo. En la
20      * documentación de
21      * la clase String
22      * (https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/
23      * String.html) se explica que la cadena de caracteres se almacena
24      * internamente codificada en UTF-16. ¿Se ha cambiado la codificación
25      * en el
26      * fichero con respecto a la que tenía el String en memoria? Si es así
27      * , ¿en
28      * qué momento crees que puede haberse realizado esta recodificación
29      * en cada
30      * programa?
31      */
32     final static char[] arr = new char[5];
33
34     public static void main(String[] args) {
35         String cadena = "Hola, soy una espléndida y muy reseñable
36         secuencia de bytes.";
37         try (FileWriter fw = new FileWriter("prueba.txt", true)) {
38             fw.write(cadena);
39
40         } catch (IOException ex) {
41             System.out.printf("ERROR: escribiendo a fichero: %s\n", ex.
42             getMessage());
43         }
44
45         try (FileReader fr = new FileReader("prueba.txt")) {
46             int unCar;
```



```

37
38         while ((unCar = fr.read(arr)) != -1) {
39             String texto = new String(arr, 0, unCar);
40             System.out.println(texto);
41         }
42         System.out.printf("Codificacion: %s\n", fr.getEncoding());
43         System.out.println("¿Se ha cambiado la codificación en el
            fichero con respecto a la que tenía el String en memoria?")
            ;
44         System.out.printf("Si a %s\n", fr.getEncoding());
45         System.out.println(" ¿En qué momento crees que puede haberse
            realizado esta recodificación en cada programa?");
46         System.out.println("En la creacion del archivo.");
47     } catch (IOException ex) {
48         System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
            getMessage());
49     }
50 }
51
52 }

```

Ejercicios/Ejercicio1punto11/src/ejercicio1punto11/Ejercicio1punto11.java

```

1  package ejercicio1punto11;
2
3  import java.io.File;
4  import java.io.FileWriter;
5  import java.io.IOException;
6
7  /*
8  En realidad, lo que hace newLine es, básicamente, escribir un salto de
9  línea, '\n'. Prueba a modificar el programa
10 anterior para que utilice solo la clase FileWriter.
11 */
12 public class Ejercicio1punto11 {
13
14     private static final String NOM_FICH_SALIDA = "fichero.txt";
15
16     public static void main(String[] args) {
17         try (FileWriter fw = new FileWriter(NOM_FICH_SALIDA)) {
18             for (int i = 0; i < 10; i++) {
19                 fw.write("*".repeat(i)+"\n");
20             }
21         } catch (IOException e) {
22             System.out.println("Error de E/S: " + e.getMessage());
23         }
24     }
25 }

```

Ejercicios/Ejercicio1punto12/src/ejercicio1punto12/Ejercicio1punto12.java

```

1 package ejercicio1punto12;
2
3 import java.io.FileWriter;
4 import java.io.IOException;
5 import java.nio.charset.Charset;
6
7 public class Ejercicio1punto12 {
8
9     public static void main(String[] args) {
10
11         String fichero = "prueba.txt";
12
13         try(FileWriter w = new FileWriter(fichero, Charset.forName("ISO
14             -8859-1"))){
15
16             w.write("ñ y ç, y el símbolo del euro (€).");
17
18         } catch (IOException ex){
19             System.out.printf("ERROR: %s\n", ex.getMessage());
20         }
21     }
22
23 }

```

Falta leerlo, o verificar que está con esa codificación.

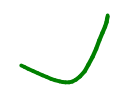


Ejercicios/Ejercicio1punto13/src/ejercicio1punto13/Ejercicio1punto13.java

```

1 package ejercicio1punto13;
2
3 import java.io.BufferedReader;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.InputStream;
7 import java.net.MalformedURLException;
8 import java.net.URL;
9
10 public class Ejercicio1punto13 {
11
12     public static void main(String[] args) {
13
14         String url = " https://www-curator.jsc.nasa.gov/antmet/mmc/nakhla.
15             pdf";
16         URL u = null;
17         try {
18             u = new URL(url);
19         } catch (MalformedURLException ex) {
20             System.out.printf("ERROR: leyendo la URL: %s\n", ex.getMessage
21                 ());
22         }
23         String nomFich = url.substring(url.lastIndexOf("/") + 1);
24     }
25 }

```



```

23     try(InputStream is = u.openConnection().getInputStream();
24         FileOutputStream w = new FileOutputStream(nomFich)){
25         int byteLeido;
26         while((byteLeido = is.read()) != -1){
27             w.write(byteLeido);
28         }
29     } catch(IOException ex){
30         System.out.printf("ERROR: %s\n", ex.getMessage());
31     }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46     }
47
48 }

```

Ejercicios/Ejercicio1punto14/src/clases/Empleado.java

```

1  package clases;
2
3  public class Empleado {
4
5      int numEmp;
6      String dni;
7      String nombre;
8      double salBrutoAnual;
9      boolean tParcial;
10
11     public Empleado(int numEmp, String dni, String nombre, double
12         salBrutoAnual, boolean tParcial) {
13         this.numEmp = numEmp;
14         this.dni = dni;
15         this.nombre = nombre;
16         this.salBrutoAnual = salBrutoAnual;
17         this.tParcial = tParcial;
18     }
19
20     // <editor-fold defaultstate="collapsed" desc=" Getter y Setter ">
21     public int getNumEmp() {
22         return numEmp;

```

```

22     }
23
24     public void setNumEmp(int numEmp) {
25         this.numEmp = numEmp;
26     }
27
28     public String getDni() {
29         return dni;
30     }
31
32     public void setDni(String dni) {
33         this.dni = dni;
34     }
35
36     public String getNombre() {
37         return nombre;
38     }
39
40     public void setNombre(String nombre) {
41         this.nombre = nombre;
42     }
43
44     public double getSalBrutoAnual() {
45         return salBrutoAnual;
46     }
47
48     public void setSalBrutoAnual(double salBrutoAnual) {
49         this.salBrutoAnual = salBrutoAnual;
50     }
51
52     public boolean istParcial() {
53         return tParcial;
54     }
55
56     public void settParcial(boolean tParcial) {
57         this.tParcial = tParcial;
58     }
59     // </editor-fold>
60
61     @Override
62     public String toString() {
63         return "Empleado{" + "numEmp=" + numEmp + ", dni=" + dni + ",
64             nombre=" + nombre + ", salBrutoAnual=" + salBrutoAnual + ",
65             tParcial=" + tParcial + '}';
66     }
67
68 }

```

Ejercicios/Ejercicio1punto14/src/ejercicio1punto14/Ejercicio1punto14.java

```

1 package ejercicio1punto14;
2
3 import clases.Empleado;
4 import java.io.DataInputStream;
5 import java.io.DataOutputStream;
6 import java.io.EOFException;
7 import java.io.FileInputStream;
8 import java.io.FileOutputStream;
9 import java.io.FileReader;
10 import java.io.IOException;
11 import java.io.OutputStream;
12 import java.util.ArrayList;
13 import java.util.List;
14
15 /**
16  * Todo lo que se pide en esta actividad debe estar en un mismo proyecto.
17  * Crea
18  * una clase Empleado, con atributos numEmp de tipo int, dni de tipo
19  * String,
20  * nombre de tipo String, salBrutoAnual de tipo double, y tParcial de tipo
21  * boolean. Crea un programa que cree tres objetos de la clase Empleado y
22  * escriba sus datos en un fichero, utilizando la clase DataOutputStream.
23  * Debes
24  * crear una nueva clase para ello, con un método main para que sea
25  * ejecutable.
26  * Crea un programa que lea un fichero con datos de empleados, utilizando
27  * la
28  * clase DataInputStream, y que a partir de ellos cree los
29  * correspondientes
30  * objetos de la clase Empleado, y que los introduzca en una lista (objeto
31  * de
32  * una clase cualquiera que implemente la interfaz List), y después itere
33  * sobre
34  * la lista para mostrar los datos de cada empleado. Debes crear una nueva
35  * clase
36  * para ello, con un método main para que sea ejecutable.
37  *
38  */
39 public class Ejercicio1punto14 {
40
41     public static void main(String[] args) {
42
43         /*
44         Empleado[] empl = {
45             new Empleado(1, "26352", "Paco", 1240.21, false),
46             new Empleado(2, "154663", "Antonio", 810.11, true),
47             new Empleado(3, "632421", "Maria", 2143.61, false)
48         };
49
50         try (var os = new FileOutputStream("empleados.txt"); var dos = new
51             DataOutputStream(os)) {
52             for (Empleado empleado : empl) {
53                 dos.writeInt(empleado.getNumEmp());
54                 dos.writeUTF(empleado.getDni());
55             }
56         }
57     }
58 }

```

```

45         dos.writeUTF(empleado.getNombre());
46         dos.writeDouble(empleado.getSalBrutoAnual());
47         dos.writeBoolean(empleado.istParcial());
48     }
49     } catch (IOException ex) {
50         System.out.printf("ERROR: %s", ex.getMessage());
51     }
52     */
53     List<Empleado> empleados = new ArrayList<>();
54     try (FileInputStream is = new FileInputStream("empleados.txt");
55         DataInputStream dis = new DataInputStream(is)) {
56         while (true) {
57             int numEmp = dis.readInt();
58             String dni = dis.readUTF();
59             String nombre = dis.readUTF();
60             double salBrutoAnual = dis.readDouble();
61             boolean tParcial = dis.readBoolean();
62             empleados.add(new Empleado(numEmp, dni, nombre,
63                                     salBrutoAnual, tParcial));
64         }
65     } catch (EOFException ex) {
66         // No es un error, simplemente indica que hemos terminado de
67         // leer el archivo
68     } catch (IOException ex) {
69         System.out.printf("ERROR: %s", ex.getMessage());
70     }
71     for (Empleado empleado : empleados) {
72         System.out.println(empleado);
73     }
74 }

```

Ejercicios/Ejercicio1punto15/src/clases/Empleado.java

```

1 package clases;
2
3 import java.io.Serializable;
4
5 public class Empleado implements Serializable {
6
7     int numEmp;
8     String dni;
9     String nombre;
10    double salBrutoAnual;
11    boolean tParcial;
12
13    public Empleado(int numEmp, String dni, String nombre, double
14                    salBrutoAnual, boolean tParcial) {
15        this.numEmp = numEmp;
16        this.dni = dni;
17        this.nombre = nombre;

```



```

17         this.salBrutoAnual = salBrutoAnual;
18         this.tParcial = tParcial;
19     }
20
21     // <editor-fold defaultstate="collapsed" desc=" Getter y Setter ">
22     public int getNumEmp() {
23         return numEmp;
24     }
25
26     public void setNumEmp(int numEmp) {
27         this.numEmp = numEmp;
28     }
29
30     public String getDni() {
31         return dni;
32     }
33
34     public void setDni(String dni) {
35         this.dni = dni;
36     }
37
38     public String getNombre() {
39         return nombre;
40     }
41
42     public void setNombre(String nombre) {
43         this.nombre = nombre;
44     }
45
46     public double getSalBrutoAnual() {
47         return salBrutoAnual;
48     }
49
50     public void setSalBrutoAnual(double salBrutoAnual) {
51         this.salBrutoAnual = salBrutoAnual;
52     }
53
54     public boolean istParcial() {
55         return tParcial;
56     }
57
58     public void settParcial(boolean tParcial) {
59         this.tParcial = tParcial;
60     }
61     // </editor-fold>
62
63     @Override
64     public String toString() {
65         return "Empleado{" + "numEmp=" + numEmp + ", dni=" + dni + ",
66             nombre=" + nombre + ", salBrutoAnual=" + salBrutoAnual + ",
67             tParcial=" + tParcial + '}';
68     }

```

69
70 }

Ejercicios/Ejercicio1punto15/src/ejercicio1punto15/Ejercicio1punto15.java

```
1 package ejercicio1punto15;
2
3 import clases.Empleado;
4 import java.io.EOFException;
5 import java.io.FileInputStream;
6 import java.io.FileOutputStream;
7 import java.io.IOException;
8 import java.io.ObjectInputStream;
9 import java.io.ObjectOutputStream;
10 import java.util.ArrayList;
11
12 public class Ejercicio1punto15 {
13
14     /*
15     Todo lo que se pide en esta actividad debe estar en un mismo proyecto.
16     Puedes empezar con una copia del proyecto
17     desarrollado para la actividad anterior.
18     Sobre la clase Empleado no se harán en principio cambios, salvo los
19     estrictamente necesarios para que se puedan
20     escribir instancias suyas utilizando la clase ObjectOutputStream.
21     El proyecto debe incluir un programa que cree tres objetos de la clase
22     Empleado y escriba sus datos en un fichero,
23     utilizando la clase ObjectOutputStream. Debe haber una clase para ello
24     , con un método main para que sea
25     ejecutable.
26     El proyecto debe incluir un programa que lea un fichero con objetos de
27     la clase Empleado, utilizando la clase
28     ObjectInputStream, los introduzca en una lista (objeto de una clase
29     cualquiera que implemente la interfaz
30     List), y después itere sobre la lista para mostrar los datos de cada
31     empleado. Debe haber una nueva clase para ello,
32     con un método main para que sea ejecutable.
33     */
34     public static void main(String[] args) {
35         ArrayList<Empleado> arr = new ArrayList<Empleado>();
36
37         Empleado[] empl = {
38             new Empleado(1, "26352", "Paco", 1240.21, false),
39             new Empleado(2, "154663", "Antonio", 810.11, true),
40             new Empleado(3, "632421", "Maria", 2143.61, false)
41         };
42
43         try (var fos = new FileOutputStream("empleados.txt"); var oos =
44             new ObjectOutputStream(fos); var fis = new FileInputStream("
45             empleados.txt"); var ois = new ObjectInputStream(fis)) {
46             // Escribir objetos Empleado en el archivo
47             for (Empleado empleado : empl) {
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```

39         oos.writeObject(empleado);
40     }
41
42     // Leer objetos Empleado del archivo y agregarlos a la lista
43     while (true) {
44         Empleado empleado = (Empleado) ois.readObject();
45         arr.add(empleado);
46     }
47 } catch (IOException | ClassNotFoundException ex) {
48     System.out.printf("ERROR: %s", ex.getMessage());
49 }
50
51 for (Empleado empleado : arr) {
52     System.out.println(empleado);
53 }
54 }
55
56 }

```

Ejercicios/Ejercicio1punto16/src/ejercicio1punto16/Ejercicio1punto16.java

```

1 package ejercicio1punto16;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7 import java.util.zip.ZipEntry;
8 import java.util.zip.ZipInputStream;
9 import java.util.zip.ZipOutputStream;
10
11 public class Ejercicio1punto16 {
12
13     /*
14     Las clases ZipInputStream y ZipOutputStream permiten, respectivamente,
15     leer y crear ficheros
16     comprimidos de tipo zip. Consulta la página https://www.thecoderscorner.com/team-blog/java-and-jvm/12-reading-azip-file-from-java-using-zipinputstream/ y las alcanzables a partir de los
17     enlaces que contiene para ver ejemplos de
18     generación de ficheros zip y de extracción de sus contenidos.
19     Crea un programa al que se le pase como parámetro de línea de comandos
20     la ruta de un directorio, y que cree en el
21     directorio de ejecución del programa un fichero zip con el mismo
22     nombre que el directorio y terminado en .zip. Si
23     no se le pasa una ruta de directorio que corresponda a un directorio
24     que exista, el programa debe mostrar un mensaje
25     de error y terminar su ejecución. El programa debe ser consistente con
26     el estilo de programación utilizado en general
27     hasta ahora. Por ejemplo: en lugar de utilizar un Logger, debe
28     escribir en la salida estándar y/o de error. En la

```

```

22 medida de lo posible, para trabajar con ficheros, debe utilizar la
23     clase java.io.File y no clases del paquete
24     java.nio.file.
25 Crea un programa al que se le pase como parámetro de línea de comandos
26     la ruta de un fichero zip y que lo
27     descomprima en el directorio de ejecución del programa el fichero zip.
28     El fichero podría ser uno generado por el programa anterior.
29     */
30 public static void main(String[] args) {
31     if (args.length == 0) {
32         System.out.println("Por favor, proporciona la ruta del
33             directorio como argumento.");
34         System.exit(0);
35     }
36
37     try (var fos = new FileOutputStream(args[0]); var zipOut = new
38         ZipOutputStream(fos)) {
39         // Creo los ficheros
40         agregarArchivoAlZIP("prueba.txt", "HOLA", zipOut);
41         agregarArchivoAlZIP("prueba2.txt", "Hoy es lunes", zipOut);
42
43         // Extraigo los ficheros
44         try (var fis = new FileInputStream(args[0]); var zipIn = new
45             ZipInputStream(fis)) {
46             ZipEntry entry;
47             while ((entry = zipIn.getNextEntry()) != null) {
48                 String fileInfo = entry.getName();
49                 File directorio = new File(fileInfo);
50                 try (var fos2 = new FileOutputStream(directorio)) {
51                     byte[] buffer = new byte[1024];
52                     int len;
53
54                     while ((len = zipIn.read(buffer)) > 0) {
55                         fos2.write(buffer, 0, len);
56                     }
57                     System.out.println("Archivo extraido: " + fileInfo
58                         );
59                 }
60             }
61         }
62     } catch (IOException ex) {
63         System.out.printf("ERROR: %s\n", ex.getMessage());
64     }
65 }
66
67 // Método para agregar un archivo al archivo ZIP
68 private static void agregarArchivoAlZIP(String nombreArchivo, String
69     contenido, ZipOutputStream zipOut) throws IOException {
70     // Creo el nombre del archivo que voy a introducir
71     ZipEntry zipEntry = new ZipEntry(nombreArchivo);
72     // Escribo sobre el nombre del fichero
73     zipOut.putNextEntry(zipEntry);

```

```

68         // Guardo el contenido en un array de bytes
69         byte[] bytes = contenido.getBytes();
70         // Escribo byte en el archivo, empiezo en la posicion 0 y llego
71         // hasta la longitud de bytes
72         zipOut.write(bytes, 0, bytes.length);
73
74         // Cierro la entrada
75         zipOut.closeEntry();
76     }
77 }

```

Ejercicios/Ejercicio1punto2/src/ejercicio1punto2/Ejercicio1punto2.java

```

1 package ejercicio1punto2;
2
3 import java.io.File;
4 import java.io.IOException;
5
6 public class Ejercicio1punto2 {
7
8     /*Crea un directorio, y dentro de él dos ficheros y un directorio.
9     Para cada uno los ficheros y directorios que has creado,
10    muestra su nombre, una vez obtenido mediante un método de la clase
11    File. Para el primer directorio que has creado,
12    muestra una lista con todos sus contenidos, mostrando para cada
13    fichero y directorio su nombre y si es un fichero o un
14    directorio.
15    Verifica que todo se ha creado correctamente. */
16
17    public static void main(String[] args) {
18
19        File directory = new File("C:\\Users\\Sergio\\Desktop\\Estudios\\
20        Acceso-a-Datos\\Tema 1\\Ejercicios de repaso\\Ejercicio1punto2
21        \\Ej");
22        System.out.println(directory.getName());
23
24        // Creacion de los directorios y archivos y monstrado nombres por
25        // consola.
26        if(directory.mkdir()){
27            System.out.println("Directorio ha sido creado exitosamente.");
28            try {
29                File directoryChild = new File("C:\\Users\\Sergio\\Desktop
30                \\Estudios\\Acceso-a-Datos\\Ejercicios de repaso\\
31                Ejercicio1punto2\\Ej\\Child");
32                directoryChild.mkdir();
33                System.out.println(directoryChild.getName());
34                File f = new File(directory, "hola.txt");
35                File f2 = new File(directory, "hola2.txt");
36                f.createNewFile();
37                System.out.println(f.getName());
38                f2.createNewFile();
39                System.out.println(f2.getName());
40            } catch (IOException e) {
41                e.printStackTrace();
42            }
43        }
44    }
45 }

```

Crearlos en el directorio actual

X

```

32         } catch (IOException ex) {
33             ex.printStackTrace();
34         }
35     } else {
36         System.out.println("Directorio no ha podido ser creado.");
37     }
38
39     //Recorrido para comprobar si es directorio o archivo
40     for(File f : directory.listFiles()){
41         if(f.isDirectory()){
42             System.out.println(f.getName() + " es Directorio.");
43         } else if (f.isFile()){
44             System.out.println(f.getName() + " es un Archivo.");
45         }
46     }
47
48
49 }
50
51 }

```

Ejercicios/Ejercicio1punto3/src/ejercicio1punto3/Ejercicio1punto3.java

```

1  package ejercicio1punto3;
2
3  import java.io.FileInputStream;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6
7  public class Ejercicio1punto3 {
8
9      /*
10     Crea y ejecuta el primer programa. Después crea y ejecuta el segundo
11     programa. Para que este último funcione, copia
12     antes en su directorio de trabajo el fichero generado por el primer
13     programa.
14     Prueba introduciendo el el texto del primer programa vocales
15     acentuadas y caracteres como ñ, ç, £, y otros que no
16     existan en inglés. Verifica si el texto se ha escrito correctamente en
17     el fichero y cómo el segundo programa muestra
18     los caracteres
19     */
20     public static void main(String[] args) {
21
22         /*
23         String cadena = "Hola, soy una secuencia de bytes.";
24         byte[] bytes = cadena.getBytes();
25         try (FileOutputStream fos = new FileOutputStream("fichero.txt")) {
26
27             for (byte unByte : bytes) {
28                 fos.write(unByte);
29             }
30         }
31         */
32     }
33 }

```

```

26     } catch (IOException ex) {
27         System.out.printf("ERROR: escribiendo a fichero: %s\n", ex.
            getMessage());
28     }*/
29     try (FileInputStream fis = new FileInputStream("fichero.txt")) {
30         int unByte;
31         while ((unByte = fis.read()) != -1) {
32             System.out.printf("%3d(%c)\n", unByte, (char) unByte);
33         }
34     } catch (IOException ex) {
35         System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
            getMessage());
36     }
37
38 }
39
40 }

```

Ejercicios/Ejercicio1punto4/src/ejercicio1punto4/Ejercicio1punto4.java

```

1  package ejercicio1punto4;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.FileOutputStream;
7  import java.io.IOException;
8
9  public class Ejercicio1punto4 {
10
11
12     /*
13     Crea un programa al que se le pase por parámetro de línea de comandos
        un nombre de fichero. Este puede tener un
14     path absoluto, lo que es útil para acceder a él si no está en el mismo
        directorio en que se ejecuta el programa. Por
15     ejemplo: /etc/fstab
16     Si el fichero no existe, se mostrará un mensaje de error y se
        terminará la ejecución del programa.
17     En otro caso, se creará una copia del fichero. El nombre de la copia
        será igual que el del fichero original, añadiendo
18     .bak al final. Esta copia se creará en el mismo directorio en el que
        se ejecuta el programa. No hay que hacer nada
19     especial para ello. Solo asegurarse de que se toma solo el nombre del
        fichero, y la ruta completa, para añadirle al final
20     .bak. Por ejemplo, si el fichero original que hay que copiar es /etc/
        fstab, entonces el nombre de la copia será
21     fstab.bak.
22     Para crear la copia, el programa debe crear un FileInputStream para
        leer del fichero de origen, byte a byte, y un
23     FileOutputStream para escribir en el fichero de destino (la copia),
        cada byte que se va leyendo del fichero de

```

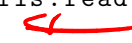
```

24     origen.
25     */
26
27     public static void main(String[] args) {
28
29         // Leo por args la direccion y el nombre del archivo.
30         if(args.length < 1){
31             System.out.println("No se ha encontrado el nombre del archivo
32             por argumentos.");
33             System.exit(0); return;
34         }
35
36         File archivo = new File(args[0]);
37
38         // Si el archivo no existe lo creo y cierro la ejecucion del
39         programa.
40         if(!archivo.exists()){
41             System.out.println("El fichero no existe.");
42             try {
43                 archivo.createNewFile();
44             } catch (IOException ex) {
45                 ex.printStackTrace();
46             }
47             System.exit(0); return;
48         } else { // Si existe uso el InputStream para guardarlo y creo
49             una copia
50             try(FileInputStream original = new FileInputStream(archivo);
51                 FileOutputStream copia = new FileOutputStream(archivo.
52                 getAbsolutePath() + ".bak")) { // Lo quiere de esta forma
53
54                 int byteLeido;
55
56                 while ((byteLeido = original.read()) != -1) { // Asigno
57                     byte a byte hasta que el archivo se quede sin byte
58                     copia.write(byteLeido);
59                 }
60
61                 System.out.printf("Copia de seguridad creada como: %s.bak\
62                 n", archivo.getAbsolutePath());
63
64                 } catch (FileNotFoundException ex) {
65                     ex.printStackTrace();
66                 } catch (IOException ex) {
67                     ex.printStackTrace();
68                 }
69             }
70         }
71     }
72 }

```

Crearlo en directorio actual, aunque el original en otro lugar.

Ejercicios/Ejercicio1punto5/src/ejercicio1punto5/Ejercicio1punto5.java

```
1 package ejercicio1punto5;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5
6 public class Ejercicio1punto5 {
7
8     private static final String NOM_FICH_ENTRADA = "fichero.txt";
9     public static final int LONG_BUFF = 5;
10
11     public static void main(String[] args) {
12
13         byte[] buff = new byte[LONG_BUFF];
14
15         try (FileInputStream fis = new FileInputStream(NOM_FICH_ENTRADA))
16         {
17             int nBytesLeidos;
18             while ((nBytesLeidos = fis.read(buff)) != -1) {
19                 for(byte b: buff){  No todos siempre, nBytesLeidos
20                     System.out.printf("%3d(%c)", b, (char) b);
21                 }
22                 System.out.printf(" | %d bytes leidos\n", nBytesLeidos);
23             }
24         } catch (IOException ex) {
25             System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
26                 getMessage());
27         }
28     }
```

Ejercicios/Ejercicio1punto6/src/ejercicio1punto6/Ejercicio1punto6.java

```
1 package ejercicio1punto6;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5
6 public class Ejercicio1punto6 {
7
8     public static void main(String[] args) {
9
10         if (args.length < 1) {
11             System.out.println("No se ha encontrado el nombre del archivo
12                 por argumentos.");
13             System.exit(0);
14         }
15
16         String fichero = args[0];
```

```

17     try (FileInputStream f = new FileInputStream(fichero)) {
18         int unByte;
19         byte contador = 0;
20         byte posicion = 0;
21         String texto = "";
22         while ((unByte = f.read()) != -1) {
23
24             if (contador == 0) {
25                 System.out.printf("%04X | ", posicion);
26             }
27
28             System.out.printf("%02X ", unByte);
29             contador++;
30             posicion++;
31             texto += (char) unByte;
32
33             if (contador == 16) {
34                 System.out.printf("|%s|\n", texto);
35                 texto = "";
36                 contador = 0;
37             }
38         }
39
40     } catch (IOException ex) {
41         ex.printStackTrace();
42     }
43
44 }
45
46 }


```

Ejercicios/Ejercicio1punto7/src/ejercicio1punto7/Ejercicio1punto7.java

```

1  package ejercicio1punto7;
2
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6
7  public class Ejercicio1punto7 {
8
9      /*
10      Crea y ejecuta el programa anterior que escribe el texto carácter a
11      carácter en un fichero. Después crea y ejecuta el
12      segundo programa. Para que este último funcione, copia antes en su
13      directorio de trabajo el fichero generado por el
14      primer programa. Verifica que el segundo programa escribe el texto
15      correctamente, también las vocales acentuadas y la letra ñ.
16      */
17      public static void main(String[] args) {
18
19      }
20
21 }

```



```

16     String cadena = "Hola, soy una espléndida y muy reseñable
17         secuencia de bytes.";
18     try (FileWriter fw = new FileWriter("prueba.txt")) {
19         for (int i = 0; i < cadena.length(); i++) {
20             fw.write(cadena.charAt(i));
21         }
22     } catch (IOException ex) {
23         System.out.printf("ERROR: escribiendo a fichero: %s\n", ex.
24             getMessage());
25     }
26
27     try (FileReader fr = new FileReader("prueba.txt")) {
28         int unCar;
29         while ((unCar = fr.read()) != -1) {
30             System.out.printf("%c\n", (char) unCar);
31         }
32     } catch (IOException ex) {
33         System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
34             getMessage());
35     }
36 }

```

Ejercicios/Ejercicio1punto8/src/ejercicio1punto8/Ejercicio1punto8.java

```

1 package ejercicio1punto8;
2
3 import java.io.FileReader;
4 import java.io.FileWriter;
5 import java.io.IOException;
6
7 public class Ejercicio1punto8 {
8
9     /*
10     Existe un constructor de FileWriter con un parámetro que permite
11     añadir contenido al final de un fichero en lugar
12     de sobrescribir sus contenidos. Modifica el programa anterior para que
13     lo use. Ejecútalo varias veces y verifica que se
14     añade texto al final cada vez que se ejecuta.
15     */
16     public static void main(String[] args) {
17
18         String cadena = "Hola, soy una espléndida y muy reseñable
19             secuencia de bytes.";
20         try (FileWriter fw = new FileWriter("prueba.txt", true)) {
21             for (int i = 0; i < cadena.length(); i++) {
22                 fw.write(cadena.charAt(i));
23             }
24         } catch (IOException ex) {
25             System.out.printf("ERROR: escribiendo a fichero: %s\n", ex.
26                 getMessage());
27         }
28     }
29 }

```

```

23     }
24
25     try (FileReader fr = new FileReader("prueba.txt")) {
26         int unCar;
27         while ((unCar = fr.read()) != -1) {
28             System.out.printf("%c", (char) unCar);
29         }
30     } catch (IOException ex) {
31         System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
            getMessage());
32     }
33
34 }
35
36 }

```

Ejercicios/Ejercicio1punto9/src/ejercicio1punto9/Ejercicio1punto9.java

```

1  package ejercicio1punto9;
2
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6
7  public class Ejercicio1punto9 {
8
9      /*
10     La clase FileWriter tiene métodos que permiten escribir un String de
11     una vez en un fichero. Cambia el primer
12     programa de ejemplo para que escriba directamente el String en el
13     fichero.
14     Cambia el segundo programa para que lea el fichero generado por el
15     anterior programa utilizando un array de char
16     con longitud 5. La longitud del array se debe definir como una
17     constante de clase (final static). El programa
18     leerá repetidas veces del fichero hacia el array. En la última lectura
19     , cuando se llegue al final del fichero, leerá un
20     número de caracteres inferior a la longitud del array, y así sabrá que
21     ha llegado al final del fichero. La salida del
22     programa será igual que la del programa anterior a partir del cual se
23     ha creado.
24     */
25     final static char[] arr = new char[5];
26
27     public static void main(String[] args) {
28         String cadena = "Hola, soy una espléndida y muy reseñable
29             secuencia de bytes.";
30         try (FileWriter fw = new FileWriter("prueba.txt", true)) {
31             fw.write(cadena);
32
33         } catch (IOException ex) {
34
35         }
36     }
37 }

```

```
26         System.out.printf("ERROR: escribiendo a fichero: %s\n", ex.
           getMessage());
27     }
28
29     try (FileReader fr = new FileReader("prueba.txt")) {
30         int unCar;
31
32         while ((unCar = fr.read(arr)) != -1) {
33             String texto = new String(arr, 0, unCar);
34             System.out.println(texto);
35         }
36     } catch (IOException ex) {
37         System.out.printf("ERROR: leyendo de fichero: %s\n", ex.
           getMessage());
38     }
39 }
40
41 }
```
