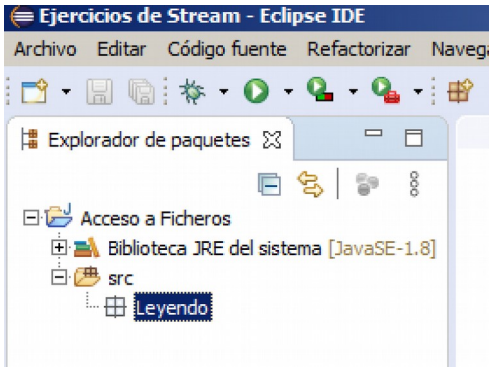
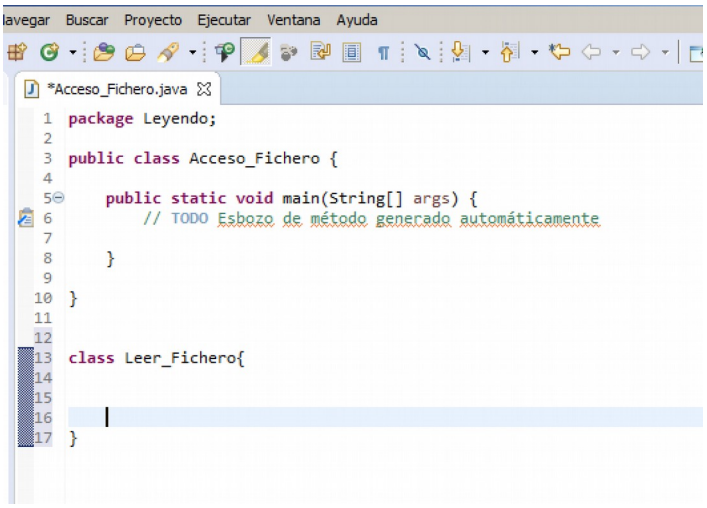
1.- Leer la información de un fichero de texto almacenado en el escritorio.

Ejercicio 1: Leer un fichero almacenado en nuestro equipo y mostrar su contenido en pantalla

Creamos la siguiente estructura:





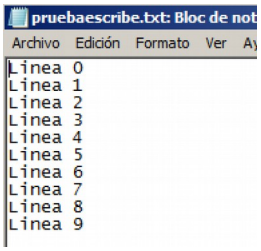

```
package Leyendo;

public class Acceso_Fichero {

	public static void main(String[] args) {
		Leer_Fichero accediendo=new Leer_Fichero();
		accediendo.lee();
	}
}
class Leer_Fichero{

	public void lee() {


	}
	}
```

2.- Como escribir caracteres en un fichero (Writer).

Ejercicio 2: Crea un fichero de texto y almacenalo en un directorio de nuestro equipo

```
package escribiendo;

public class Escribir_fichero {
		public static void main (String[] arg) {
			Escribiendo accede_es= new Escribiendo();
			accede_es.escribir();
		}

}

class Escribiendo{

	public void escribir() {

	}

}
```

Ejercicio 2.1: Crea un fichero llamado prueba_escritura con el siguiente contenido y guardalo en un directorio del equipo.




```
public class EscribeFichero
{
   public static void main(String[] args)
   {

   }
}
```

Ejercicio 2.3: Realiza un programa que dado el nombre y la ruta de un fichero , cree una copia . Ejemplo: fichero original* *prueba.txt* *, fichero resultante* *prueba_copia.txt

```
public class Copia_Ficheros {

		public static void main(String[] args) {
			Copia_Ficheros copiafichero= new Copia_Ficheros();
			copiafichero.copia();

	}

public void copia() {

}
```

3.- Manejo de archivos. Acceso a Ficheros (Buffers)

Ejercicio 3: Leer un archivo almacenado en nuestro equipo mediante Buffers y muestralo por la consola

```
package Leyendo;
public class Acceso_Fichero {

	public static void main(String[] args) {
		Leer_Fichero accediendo=new Leer_Fichero();
		accediendo.lee();
	}

}

class Leer_Fichero{

	public void lee() {

	}

}
```

4.- Manejo de archivos. Streams Byte.

Ejercicio 4:

Este ejemplo consistirá en leer un archivo cualquiera (en este caso lo haremos con un archivo de imagen) y escribiremos una copia en una carpeta cualquiera.

Para realizar esto, es necesario conocer cuantos bytes forma parte de la imagen. Esto lo haremos con un contador (a cada vuelta del bucle, lee un byte).

Una vez conocido los bytes que compone la imagen, con esos bytes debemos crear una imagen con ellos.

Para ello, necesitamos almacenar todos esos bytes en algún sitio para posteriormente crear el archivo con esos bytes.
Usaremos un Array para almacenar los bytes.

```
package leer_escribir_bytes;

public class Lectura_Escritura_bytes {

	public static void main(String[] args) {
		int contador=0; // cuenta el número de bytes que contiene la imagen.
		//Creamos un array para almacenar los bytes de la imagen.
		int datos_entrada[]=new int[2963860];
		// Leemos la información de la imagen para ello empezamos por instanciar la clase FileImputStream
		//Como esta clase lanza IOException debe de ir dentro de un bloque try_catch
		try {

			//Creamos un bucle while para leer la imagen byte a byte

			//Lee el archivo de imagen byte a byte que le indicamos en el Stream
```

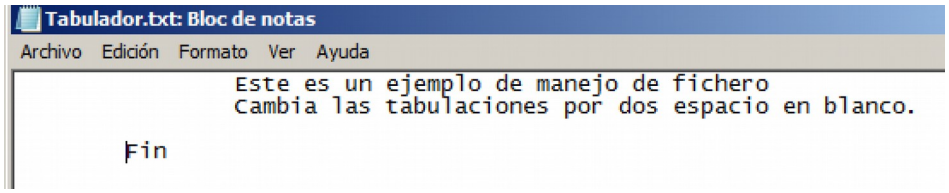
```
                // Cuando llegue al final del archivo read() devuelve -1
                // El método read() devuelve -1 cuando llega al final del fichero, pero este -1 no es ningún dato.
                // Controlo que el -1 no se guarde en el array


        //Una vez almacenado el fichero, hacemos una llamada a crear el fichero.
        // Creamos un método para crear el fichero. Este método recibe como parámetro el array que contiene todos los byte del fichero.


                        //Creamos la instancia de FileOutputStream
                        // Para simplificar, creamos el nuevo fichero en el mismo directorio y cambiamos el nombre del fichero.

                //Escribimos un fichero nuevo de bytes


package leer_escribir_bytes;

public class Lectura_Escritura_bytes {

        public static void main(String[] args) {

        }
// Creamos un método para crear el fichero. Este método recibe como parámetro el array que contiene todos los byte del fichero.
        }
}
```

	Flujos con Bytes	Flujos con caracteres
Entrada de datos	• ImputStream ◦ ByteArrayInputStream ◦ FileInputStream ◦ FilterInputStream ▪ BufferedInputStream ▪ DataInputStream ▪ LineNumberInputStream ▪ PushbackInputStream ◦ ObjectImputStream ◦ PipedInputStream ◦ SequenceInputStream ◦ StringBufferInputStream	• Reader ◦ BufferedReader ▪ LineNumberReader ◦ CharArrayReader ◦ FilterReader ▪ PushbackReader ◦ InputStreamReader ▪ FileReader ◦ PipedReader ◦ StringReader
Salida de datos	• OutputStream ◦ ByteArrayOutputStream ◦ FileOutputStream ◦ FilterOutputStream ▪ BufferedOutputStream ▪ DataOutputStream ▪ PrintStream ◦ ObjectOutputStream ◦ PipedOutputStream	• Writer ◦ BufferedWriter ◦ CharArrayWriter ◦ FilterWriter ◦ OutputStreamWriter ▪ FileWriter ◦ PipedWriter ◦ PrintWriter ◦ StringWriter

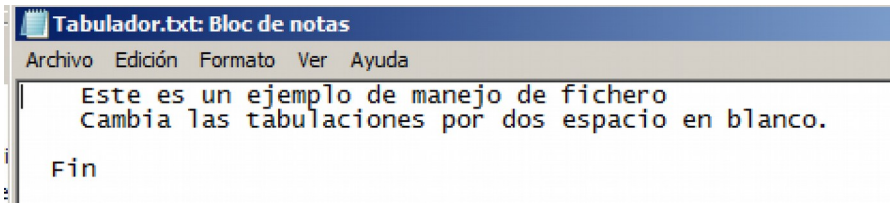
5.- Procesamiento de ficheros de texto.

Ejercicio 5: Realiza un programa que cambie cada tabulador de un fichero por dos espacios en blanco.

Fichero de entrada

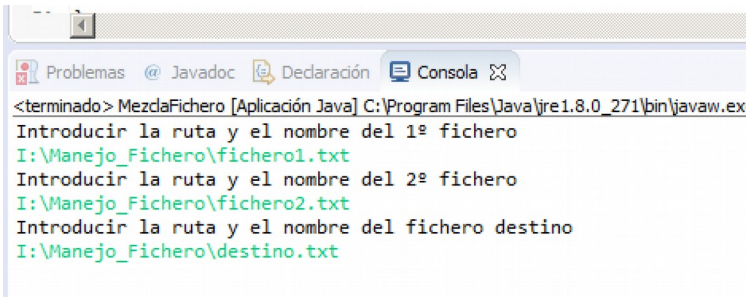


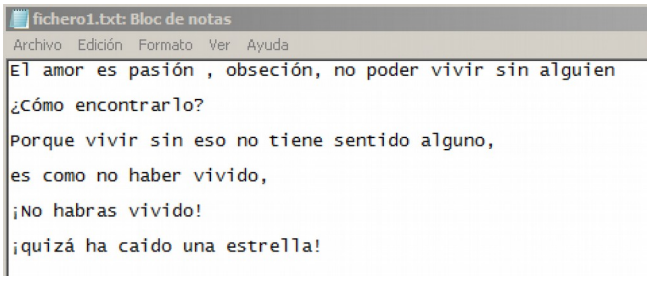
Fichero de salida

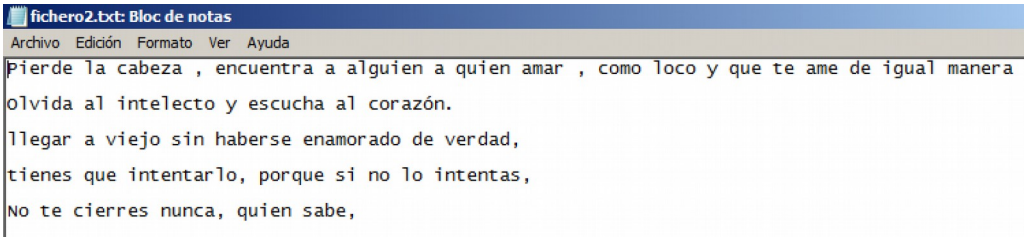



```
public class Tabulador {

        public static void main(String[] args) {

                }
        }
```

EJERCICIO 6: Escribe un programa que guarde en un fichero el contenido de otros dos ficheros, de tal forma que en el fichero resultante aparezcan las líneas de los primeros dos ficheros mezcladas, es decir, la primera línea será del primer fichero, la segunda será del segundo fichero, la tercera será la siguiente del
primer fichero, etc.
Los nombres de los dos ficheros origen y el nombre del fichero destino se deben pasar como argumentos en la línea de comandos.
Hay que tener en cuenta que los ficheros de donde se van cogiendo las líneas pueden tener tamaños diferentes.








```
public class MezclaFichero {

	public static void main(String[] args) {

		System.out.println("Introducir la ruta y el nombre del 1º fichero");

		System.out.println("Introducir la ruta y el nombre del 2º fichero");

		System.out.println("Introducir la ruta y el nombre del fichero destino");


		try {

		} catch (IOException ioe) {
			System.out.println("Se ha producido un error de lectura/escritura");
			System.err.println(ioe.getMessage());
		}
	}
}
```

Ejercicio 7 (Serialización): Convertir un objeto java en una sucesión de bytes, guardarlo en el HD del equipo y posteriormente rescatar el objeto del disco duro.

SOLUCIÓN

```
package Serialización;

import java.util.*;

import java.io.*;

public class Serializando {

	public static void main(String[] args) {
```

```
                    Administrador jefe=new Administrador("Juan", 80000, 2005,12,15);
                    jefe.setIncentivo(5000);
                    Empleado[] personal=new Empleado[3];
                    personal[0]=jefe;
                    personal[1]=new Empleado("Ana", 25000, 2008, 10,1);
                    personal[2]=new Empleado("Luis", 18000, 2012, 9,15);

/**Construimos un flujo de datos de tipo ObjectOutputStream (flujo de salida).
El constructor de ObjectOutputStream pide un argumento de tipo OutputStream
Le pasamos un objeto FileOutputStrem para especificar donde vamos a escribir esta sucesión de bytes del objeto Empleado*/
                    try {
                            ObjectOutputStream escribiendo_fichero = new ObjectOutputStream(new FileOutputStream("I:\\
EjercicioStream\\Ejer1Serializacion.dat"));
                            //Le he dado la extensión dat al fichero que se va a crear, la extensión da igual.
                            //Una vez que hemos creado el flujo de dato de salida al exterior,ahora tenemos que escribir ese objeto.
                            //El objeto a escribir es del tipo "Empleado[]" y se llama personal.
                            // ¿Cuál es el método de ObjectOutputStream que nos permite realizar todo esto?
                            //Visitamos la API y vemos que "writeObject(Object objeto)" escribe un objeto especificado en el
ObjectOutputStream
                            escribiendo_fichero.writeObject(personal);
                            //Cerramos flujo
                            escribiendo_fichero.close();

        //Una vez que hemos archivado el objeto ahora lo vamos a rescatar y lo vamos a mostrar por pantalla.
        // Creamos un objeto de tipo "ObjectInputStream" (flujo de entrada) lo llamamos "recupera_fichero"
                            ObjectInputStream recupera_fichero= new ObjectInputStream(new FileInputStream("I:\\
EjercicioStream\\Ejer1Serializacion.dat"));
                    // Con esto creamos el flujo de entrada.

// Ahora usamos el método readObject de la clase ObjectInputStream para leer ese objeto.
//Importante: ¿Qué estamos leyendo?. Estamos leyendo un array, al leerlo, necesito almacenarlo en otro array por supuesto de tipo
Empleado
// Si consultamos la API, vemos que el métode readObjet lee un objeto desde ObjectinputStream y devuelve un Object
// Esto me indica que tengo dos opciones:
1) Me creo un objeto de tipo Object para almacenarlo o bien,
 2) Si trato un objeto de otro tipo,debo de realizar un casting.

                            Empleado[] personal_recuperado=(Empleado[])recupera_fichero.readObject();
        // cerramos el flujo ObjectInputStream

                            recupera_fichero.close();
        // Ahora vamos a imprimir la información de dicho objeto. Lo hacemos mediante un "for-each" que recorra el array
recuperado.

                            for(Empleado e: personal_recuperado) {
                                    System.out.println(e);
                            }

                    }catch(Exception e) {


                    }

        }

}

//-----------------------------------------------------------------------------------------------

class Empleado implements Serializable {// indicamos que los objetos Empleados son susceptibles de ser convertidos a bytes.
```

```
	public Empleado(String n, double s, int año, int mes, int dia){

		nombre=n;
		sueldo=s;
		GregorianCalendar calendario=new GregorianCalendar(año, mes-1,dia);
		fechaContrato=calendario.getTime();
	}

	public String getNombre() {
		return nombre;
	}

	public double getSueldo() {
		return sueldo;
	}

	public Date getFechaContrato() {
		return fechaContrato;
	}

	public void subirSueldo(double porcentaje){

		double aumento=sueldo*porcentaje/100;
		sueldo+=aumento;

	}

	public String toString(){

		return "El Nombre es =" + nombre + ",y su sueldo es =" + sueldo + ", fecha de contrato=" + fechaContrato;

	}

	private String nombre;
	private double sueldo;
	private Date fechaContrato;

}

//-----------------------------------------------------------------------------------------

class Administrador extends Empleado{

	public Administrador(String n, double s, int año, int mes, int dia){
		super(n,s,año,mes,dia);
		incentivo=0;

	}
	public double getSueldo(){
		double sueldoBase=super.getSueldo();
		return sueldoBase + incentivo;

	}
	public void setIncentivo(double b){
		incentivo=b;
	}
	public String toString(){
		return super.toString() + " Incentivo=" + incentivo;

	}
	private double incentivo;

}
```

Ejercicio 8 (Serialización) Realiza un programa en JAVA en el que le pidas al usuario las notas de las 6 asignaturas del Ciclo de DAM y las guarde en un fichero. Posteriormente leerá el fichero y te calculará la nota media del curso.
Nota 1: Cada una de las asignaturas serán un objeto que se encuentran en un array de 6 posiciones, y cuyos atributos serán el nombre y la nota.
Nota 2: Con el constructor podrás asignar directamente el nombre de la asignatura al crear el objeto. En cambio, el atributo nota, será el usuario quien lo introduzca mediante un método que controle que la nota tenga un valor entre 0 y 10.
Ejemplo de ejecución:

```
INTRODUCIR LAS CALIFICACIONES DE CADA ASIGNATURA
Introduce la nota de Programacion :      11
nota no valida
Introduce la nota de Programacion :      9
Introduce la nota de Lenguajes de Marcas :      4
Introduce la nota de Bases de Datos :      8
Introduce la nota de Entornos de Desarrollo :      7
Introduce la nota de Sistemas Informaticos :      8
Introduce la nota de Formacion y Orientacion Laboral :      6
***** Notas almacenadas en el array *****
Creamos el fichero notas.obj
********Volcando el array al fichero *********

***** Volcado finalizado con Exito *****
Leyendo fichero
 tamaño del fichero= 329
Creado flujo Entrada de objeto
 Creada el array de recuperacion
Programacion   9.0
Lenguajes de Marcas   4.0
Bases de Datos   8.0
Entornos de Desarrollo   7.0
Sistemas Informaticos   8.0
Formacion y Orientacion Laboral   6.0
********   CALCULANDO LA MEDIA ************
Su calificación media del curso es: 7.0
```

```
package Serializa4;

public class Serializa {

        // Método que rellena el array de objetos Cursos con el atributo nombre solamente
         public static void rellenarArray(Curso[] asignatura) {
                 System.out.println("INTRODUCIR LAS CALIFICACIONES DE CADA ASIGNATURA");

           }
        //metodo que crea el fichero y almacena los datos en el mismo
//-----------------------------------------------------------------------------------------------
         public static void crearFichero(Curso[] asignatura) {
                 //Declaramos un flujo salida de tipo fichero


    } finally {

       try {
         if (flujoSalidaFichero != null) {
```

```
                flujoSalidaFichero.close();
            }
        } catch (Exception e) {
            System.out.println(" Error cerrar flujo"+e.getMessage());
        }
    }
  }// Fin crear fichero

        //Metodo para escribir el array de objetos en el fichero. Se le pasa el flujo de objeto y el array
//-----------------------------------------------------------------------------------------------
        public static void escribirFichero(ObjectOutputStream FlujoObjSalida, Curso[] asignatura){

    System.out.println("********Volcando el array al fichero *********"+ "\n");

  }
//-------------------------------------------------------------------------
        //Metodo para leer el fichero de objeto e imprimirlo por pantalla

        public static void leerFichero() {
                //Este método leera el fichero de objeto y lo mostrará por pantalla.
                System.out.println("Leyendo fichero");
                //Cremos un flujo de entrada de Fichdero


        }

//-------------------------------------------------------------------------

        public static void calculaMedia(Curso[] asignatura_rec) {
                double media=0, mediaFinal;
                System.out.println("********   CALCULANDO LA MEDIA ************");

                }
        }


//-----------------------------------------------------------------------------------------
        public static void main(String[] args) {
                //Creamos un array de objetos "Curos" de tamaño igual al número de asignaturo
    Curso[] asignatura = new Curso[6];
    //Llamamos a un método para introducir los datos con las calificaciones de cada asignatura
    rellenarArray(asignatura);

    crearFichero(asignatura);
    leerFichero();

        }

}// Fin clase Serializa
//-----------------------------------------------------------------------------------------
class Curso implements Serializable{

        private String nombre;
        private double nota;

  public Curso(String nombre) {
    this.nombre = nombre;
```

```
        establecerNota();
    }

    public String getNombre() {

    }

    public double getNota() {

    }

    public  void  establecerNota() {


    }
}
```

Manejo de archivos y directorios

Ejercicio 9: Escribir un método que reciba las rutas correspondientes a dos directorios y copie recursivamente (subdirectorios y ficheros) el contenido del directorio origen en el directorio destino.

```
package Directorios;

public class Main {

    public static void main(String[] args) {
}
```

```
package Directorios;

public class Ficheros {

    // Métodos
    public void copiarDirectorio(File dOrigen, File dDestino) throws IOException{

        // Si es un directorio entro a realizar la copia

        // Si el directorio no existe lo genera
       // Almaceno los fichos en un array de Strings
    }

    public void copiarFichero(File fOrigen, File fDestino) throws IOException{
        }

        in.close();
        out.close();

        System.out.println("Copiado " + fOrigen.getName());

    }
}
```