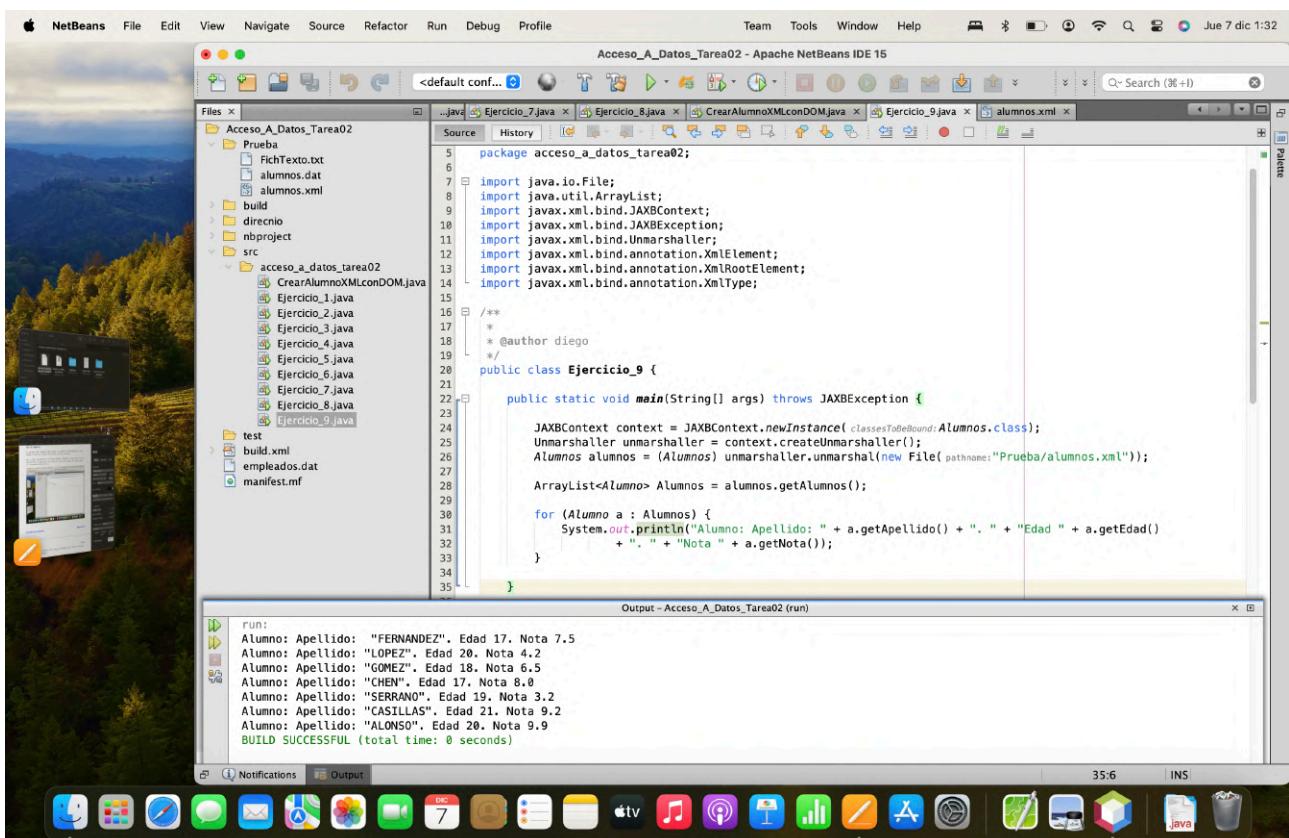


Tarea 2 para Acceso a Datos



The screenshot shows the Apache NetBeans IDE 15 interface. The left pane displays the project structure for 'Acceso_A_Datos_Tarea02'. The right pane shows the code editor with Java code for reading an XML file ('alumnos.xml') and printing its contents. The bottom pane shows the output window with the results of the execution.

```
package acceso_a_datos_tarea02;
import java.io.File;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

/**
 * 
 * @author diego
 */
public class Ejercicio_9 {

    public static void main(String[] args) throws JAXBException {
        JAXBContext context = JAXBContext.newInstance(classesToBeBound:Alumnos.class);
        Unmarshaller unmarshaller = context.createUnmarshaller();
        Alumnos alumnos = (Alumnos) unmarshaller.unmarshal(new File(pathname:"Prueba/alumnos.xml"));

        ArrayList<Alumno> Alumnos = alumnos.getAlumnos();

        for (Alumno a : Alumnos) {
            System.out.println("Alumno: Apellido: " + a.getApellido() + " " + "Edad " + a.getEdad()
                + " " + "Nota " + a.getNota());
        }
    }
}
```

Output - Acceso_A_Datos_Tarea02 (run)

```
run:
Alumno: Apellido: "FERNANDEZ", Edad 17, Nota 7.5
Alumno: Apellido: "LOPEZ", Edad 20, Nota 4.2
Alumno: Apellido: "GOMEZ", Edad 18, Nota 6.5
Alumno: Apellido: "CHEN", Edad 17, Nota 8.0
Alumno: Apellido: "SERRANO", Edad 19, Nota 3.2
Alumno: Apellido: "CASILLAS", Edad 21, Nota 9.2
Alumno: Apellido: "ALONSO", Edad 20, Nota 9.9
BUILD SUCCESSFUL (total time: 0 seconds)
```

Diego Manuel Carrasco Castañares

Para aplicar los conceptos obtenidos referente al manejo de ficheros: Crea un proyecto y en src irás creando cada uno de los ficheros .java correspondientes a cada apartado, es decir, el proyecto, en src, deben aparecer 10 ficheros .java.

Hay que entregar un documento explicativo de la tarea. Si no se entrega, la nota tendrá una bajada de 1 punto.

Obligatorio todos los apartados deben capturar las excepciones en las clases o ficheros que vas a crear. try-catch-finally

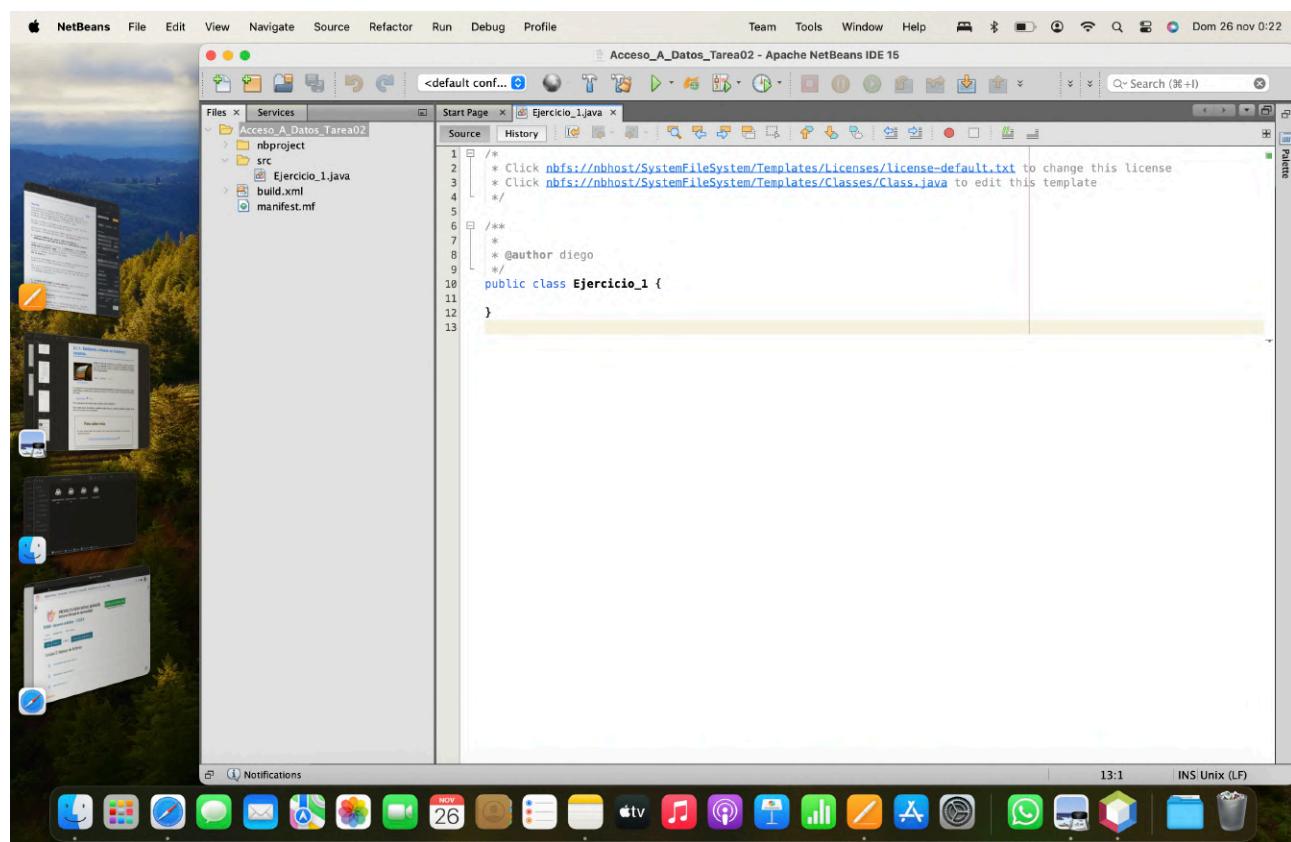
1. (1 punto) Muestra por consola todos los ficheros y subdirectorios que tiene el directorio donde está el proyecto.

También debes comprobar si existe un **directorio** llamado **Prueba**, **dondere está el proyecto (src)**. Si existe el directorio muestra un mensaje "El directorio prueba ya existe ". Y si no existe créalo y muestra el mensaje de que ha sido creado.

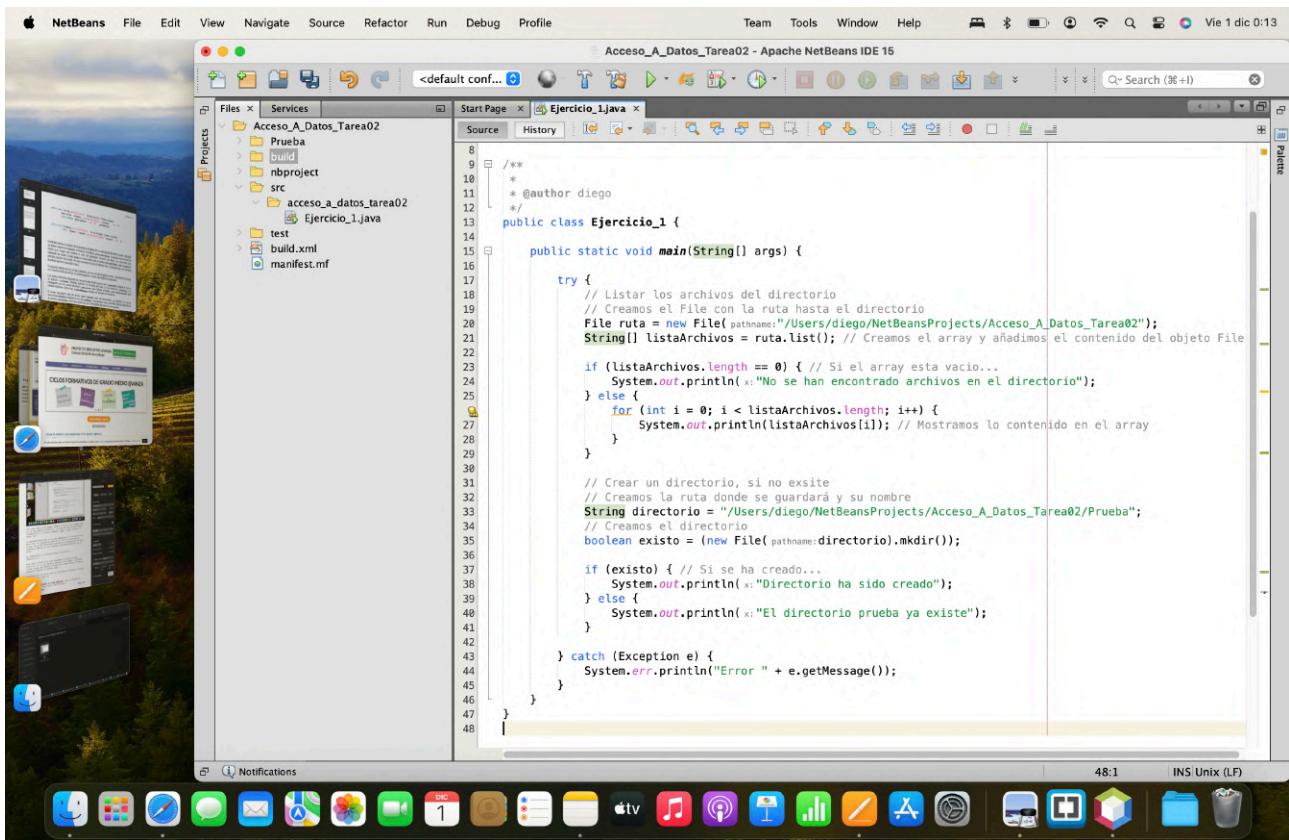
Con la clase io.

Lo primero que haremos será crear el proyecto en NetBeans al cual vamos a darle el nombre de Acceso_A_Datos_Tarea02.

Tal y como se indica en la tarea vamos a empezar creando una clase java dentro de src para el ejercicio que nos ocupa. En este caso la llamaremos Ejercicio_1.



Lo siguiente que haremos será crear el código, tal como se muestra en la siguiente imagen.



Como puede verse en la imagen creamos un objeto File llamado ruta con la ruta hasta el directorio, un array de tipo String llamado listaArchivos y le pasamos, mediante .list, lo que contiene el objeto File.

Lo siguiente será crear un if, que comprobara si la listaArchivos esta vacía (su longitud es igual a 0) y se mostrara el mensaje "No se han encontrado archivos en el directorio", en caso contrario, mediante un for, mostrara los datos contenidos en el array.

Con terminaremos la parte de mostrar los archivos y subdirectorios.

Para crear un directorio nuevo crearemos un String llamado directorio con la ruta donde se guardara y su nombre, un boolean llamado existe que creará, a través de un objeto File, el número directorio mediante el método .mkdir.

Mediante un if, si se ha creado el directorio, mostrará el mensaje "Directorio creado", en caso contrario mostrara el mensaje "El directorio Prueba ya existe".

Todo lo envolveremos en try - catch para contra las excepciones.

Con esto damos por finalizado este apartado

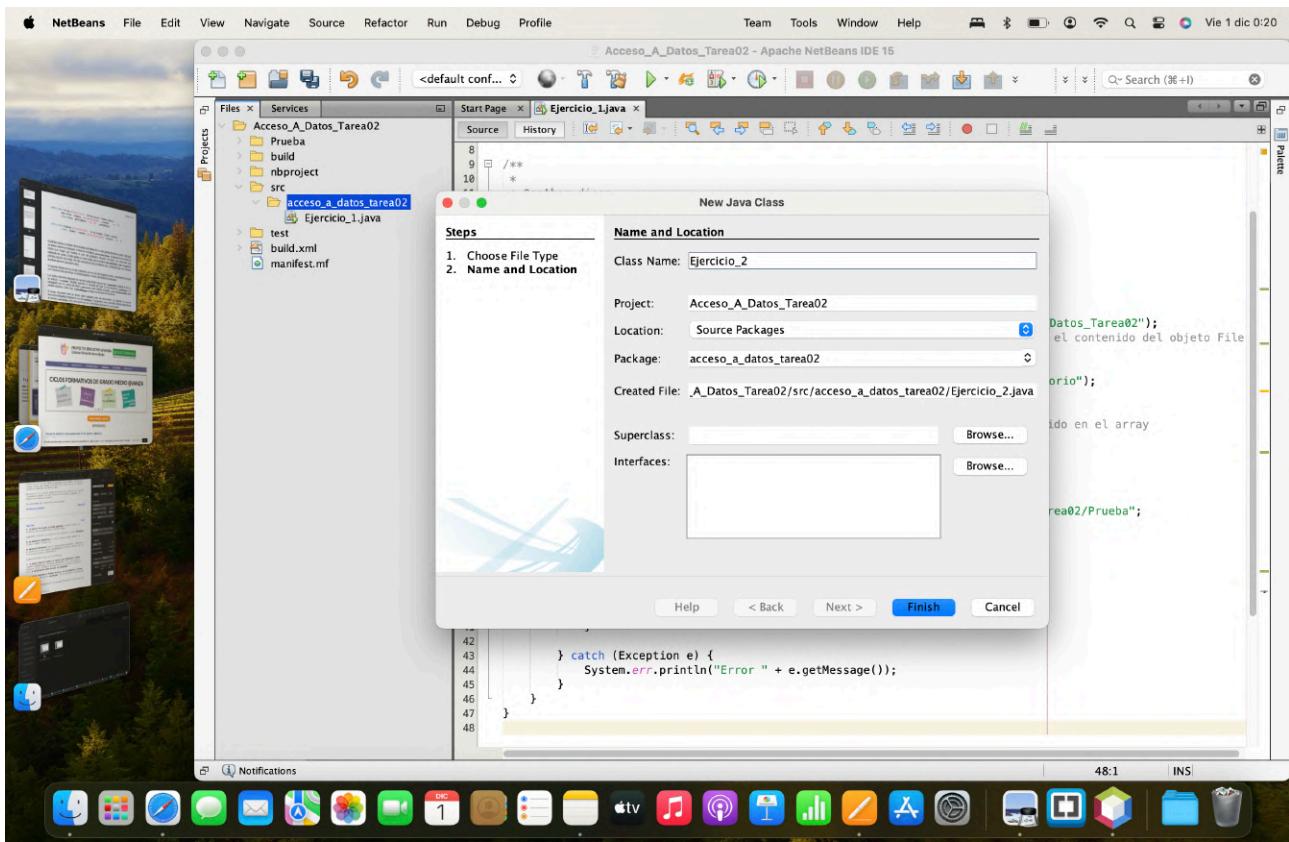
2. (1 punto) Utilizando la clase java.nio , para el manejo de ficheros, con la clase Files y objetos Path:

Comprueba si existe un directorio en el proyecto llamado **direcnio**.

Si no existe el directorio lo creas y dentro creas también un fichero llamado **fichero.txt**.

Si existe el directorio saca un mensaje de que existe, comprueba si existe un fichero dentro de direcnio el fichero **fichero.txt** y a continuación borra el directorio.

Lo primero que haremos será crear la clase java llamada **Ejercicio_2** dentro de nuestro proyecto.



Posteriormente crearemos el código.

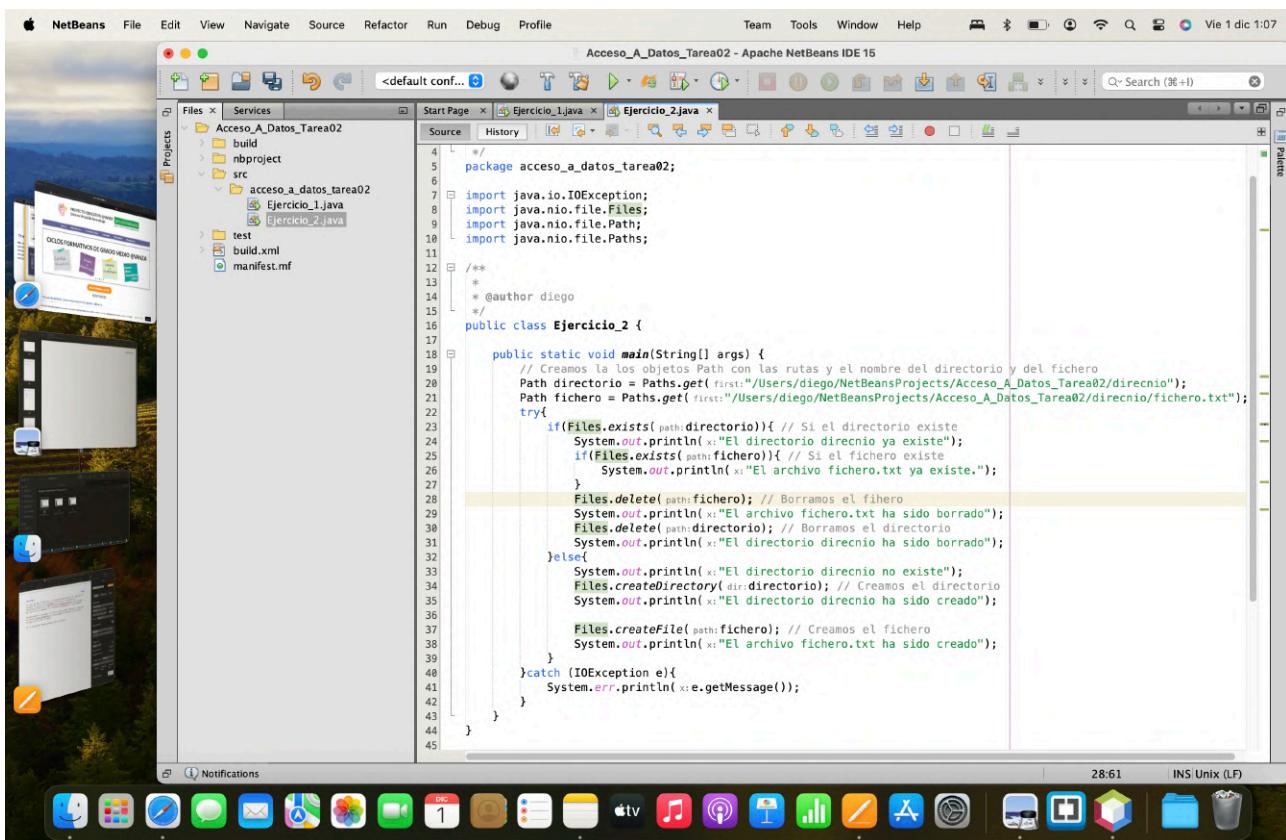
Lo primero que haremos será crear los objetos Path que contienen la ruta y el nombre del directorio y del fichero que queremos crear.

Mediante un if comprobaremos si el directorio existe, y si existe mostraremos por consola el mensaje "el directorio ya existe", comprobaremos si contiene el fichero que indica la tarea, y si lo contiene, mostraremos por consola el mensaje "El fichero ya existe" y borraremos en cascada mediante el método Files.delete (primero el fichero y luego el directorio) mostrando los mensajes de que el fichero y el directorio han sido borrados.

En caso de que el directorio no exista mostraremos por pantalla el mensaje "El directorio direcnio no existe", crearemos el directorio mediante el método `Files.createDirectory`, mostraremos por pantalla el mensaje "El directorio direcnio ha sido creado".

Posteriormente crearemos el fichero dentro del directorio mediante el método `Files.createFile` y mostraremos el mensaje "El archivo fichero.txt ha sido creado".

En la siguiente imagen podemos ver el código.



The screenshot shows the NetBeans IDE interface on a Mac OS X desktop. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The left sidebar shows a project structure with a main folder "Acceso_A_Datos_Tarea02" containing subfolders "build", "nbproject", and "src", which further contains "acceso_a_datos_tarea02" and "Ejercicio_1.java". Below these are "test", "build.xml", and "manifest.mf". The central workspace displays the source code for "Ejercicio_2.java". The code uses Java's NIO API to handle file paths and manipulate files and directories. The code is as follows:

```

4  /*
5   * To change this license header, choose License Headers in Project Properties.
6   * To change this template file, choose Tools | Templates
7   * and open the template in the editor.
8   */
9  package acceso_a_datos_tarea02;
10
11 import java.io.IOException;
12 import java.nio.file.Files;
13 import java.nio.file.Path;
14 import java.nio.file.Paths;
15
16 /**
17  * @author diego
18  */
19 public class Ejercicio_2 {
20
21     public static void main(String[] args) {
22         // Creamos la los objetos Path con las rutas y el nombre del directorio y del fichero
23         Path directorio = Paths.get(first:"/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/direcnio");
24         Path fichero = Paths.get(first:"/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/direcnio/fichero.txt");
25
26         try{
27             if(Files.exists(path:directorio)){ // Si el directorio existe
28                 System.out.println(x:"El directorio direcnio ya existe");
29                 if(Files.exists(path:fichero)){ // Si el fichero existe
30                     System.out.println(x:"El archivo fichero.txt ya existe.");
31                 }
32                 Files.delete(path:fichero); // Borramos el fichero
33                 System.out.println(x:"El archivo fichero.txt ha sido borrado");
34                 Files.delete(path:directorio); // Borramos el directorio
35                 System.out.println(x:"El directorio direcnio ha sido borrado");
36             }else{
37                 System.out.println(x:"El directorio direcnio no existe");
38                 Files.createDirectory(path:directorio); // Creamos el directorio
39                 System.out.println(x:"El directorio direcnio ha sido creado");
40
41                 Files.createFile(path:fichero); // Creamos el fichero
42                 System.out.println(x:"El archivo fichero.txt ha sido creado");
43             }
44         }catch (IOException e){
45             System.err.println(x:e.getMessage());
46         }
47     }
48 }

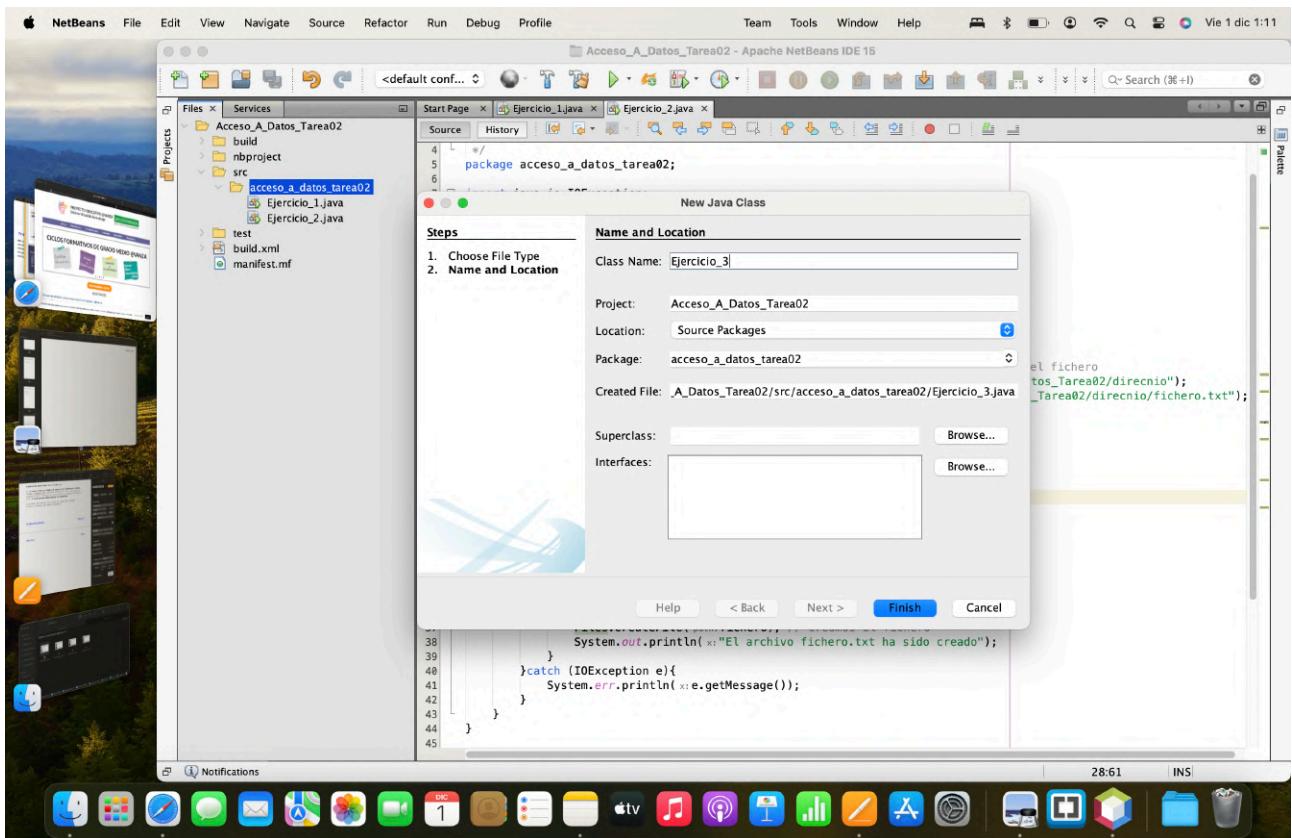
```

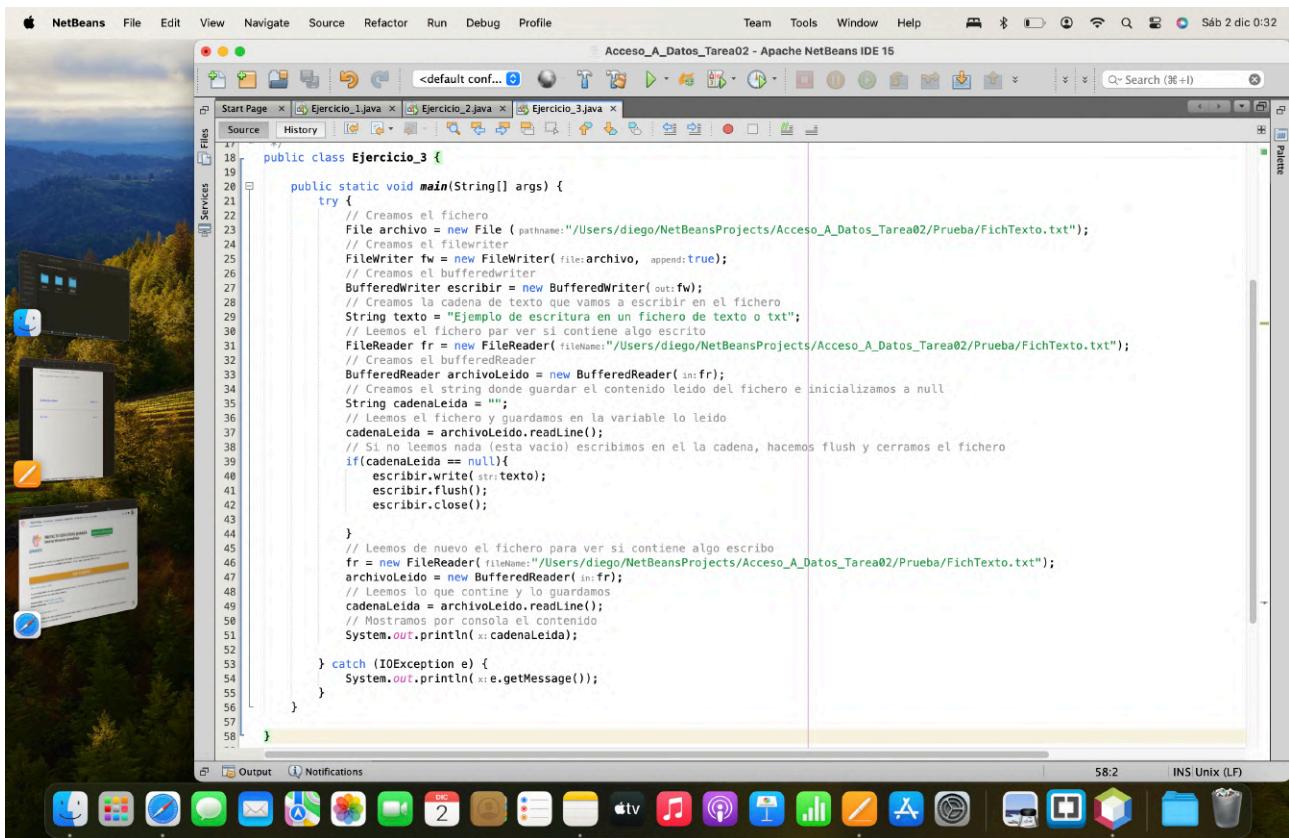
Con esto damos por finalizo este apartado.

A partir de ahora todos con la clase io

3. (1 punto) Crea un fichero de texto en el directorio Prueba, llamado FichTexto.txt, de forma secuencial, en el que vaya el siguiente texto: "Ejemplo de escritura en un fichero de texto o txt". A continuación debe mostrar su contenido.

Lo primero que haremos será crear la clase java llamada `Ejercicio_3` dentro de nuestro proyecto.





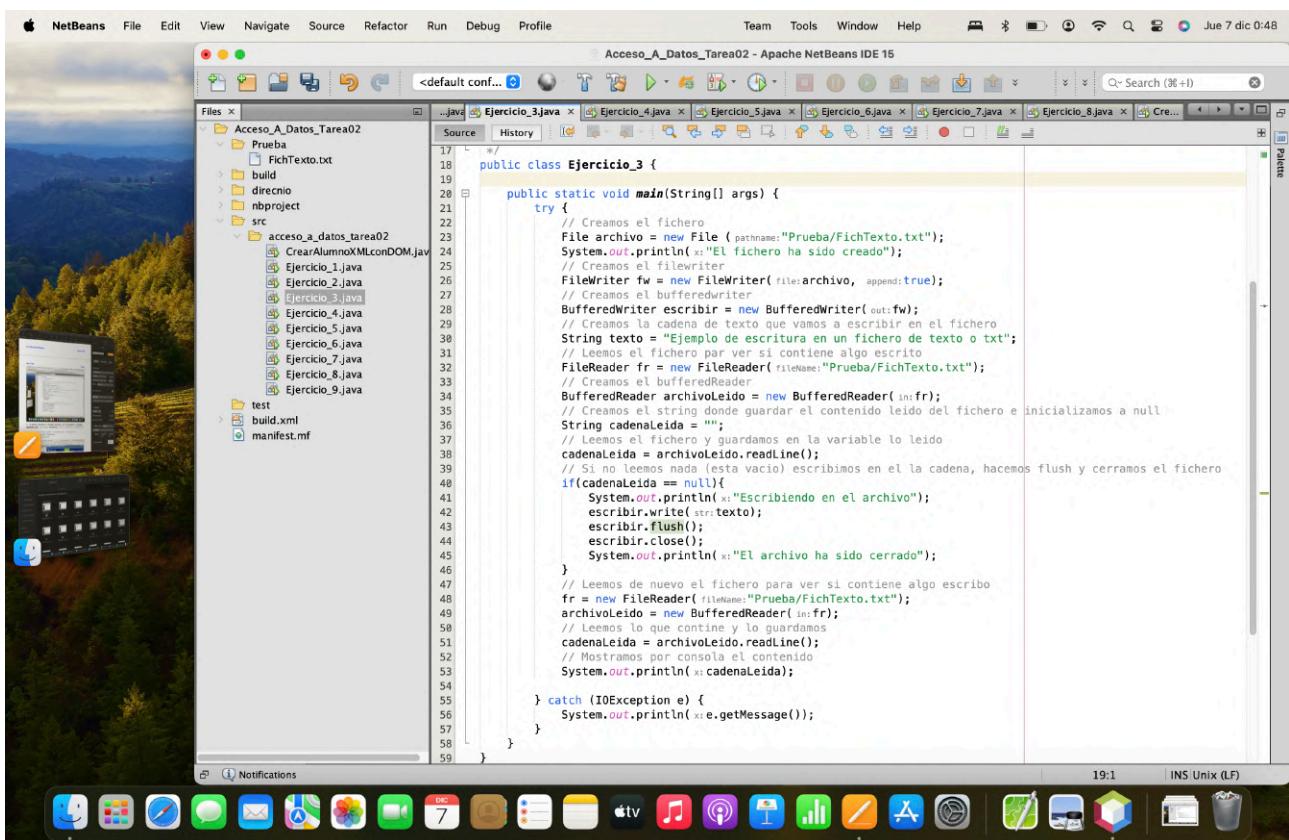
The screenshot shows the Apache NetBeans IDE 15 interface. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has various icons for file operations. The main window displays a Java code editor with the following code:

```

public class Ejercicio_3 {
    public static void main(String[] args) {
        try {
            // Creamos el fichero
            File archivo = new File (" pathname:""/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/Prueba/FichTexto.txt");
            // Creamos el filewriter
            FileWriter fw = new FileWriter(file:archivo, append:true);
            // Creamos el bufferedwriter
            BufferedWriter escribir = new BufferedWriter(out:fw);
            // Creamos la cadena de texto que vamos a escribir en el fichero
            String texto = "Ejemplo de escritura en un fichero de texto o txt";
            // Leemos el fichero par ver si contiene algo escrito
            FileReader fr = new FileReader(fileName:""/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/Prueba/FichTexto.txt");
            // Creamos el bufferedReader
            BufferedReader archivoLeido = new BufferedReader(in:fr);
            // Creamos el string donde guardar el contenido leido del fichero e inicializamos a null
            String cadenaLeida = "";
            // Leemos el fichero y guardamos en la variable lo leido
            cadenaLeida = archivoLeido.readLine();
            // Si no leemos nada (esta vacio) escribimos en el la cadena, hacemos flush y cerramos el fichero
            if(cadenaLeida == null){
                escribir.write(str:texto);
                escribir.flush();
                escribir.close();
            }
            // Leemos de nuevo el fichero para ver si contiene algo escrito
            fr = new FileReader(fileName:""/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/Prueba/FichTexto.txt");
            archivoLeido = new BufferedReader(in:fr);
            // Leemos lo que contiene y lo guardamos
            cadenaLeida = archivoLeido.readLine();
            // Mostramos por consola el contenido
            System.out.println(cadenaLeida);
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

The status bar at the bottom shows "58:2 INS Unix (LF)". The Mac OS X Dock is visible at the bottom.



This screenshot shows the same NetBeans IDE 15 environment as the first one, but with a different project structure in the left sidebar. The project tree shows a folder named "acceso_a_datos_tarea02" containing several Java files (Ejercicio_1.java, Ejercicio_2.java, Ejercicio_3.java, etc.) and other project files like build.xml and manifest.mf. The code in the editor is identical to the one shown in the first screenshot.

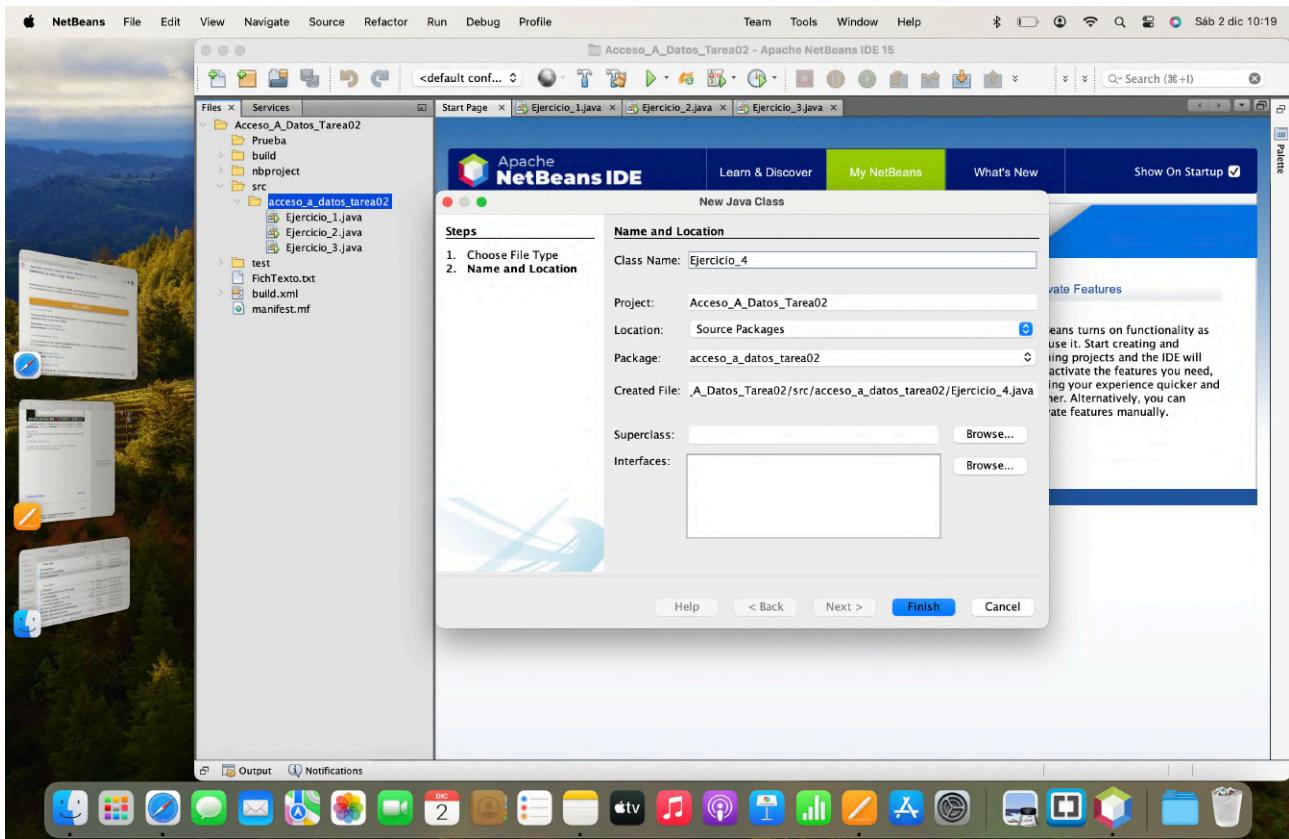
4. (1 punto) Escribe un fichero binario, en el proyecto, llamado empleados.dat, de manera secuencial. Con la siguiente información:

Departamento "Contabilidad", "Informática", "Dirección", "Análisis", "Finanzas", "Hardware"

Diego Manuel Carrasco Castañares

Página 7 de 30

Nº Empleados 3,10,2,5,4,8



Como puede apreciarse en la imagen anterior, lo primero que haremos será crear la clase java llamada Ejercicio_4 dentro de nuestro proyecto.

Crearemos un FileOutputStream llamado archivo que creara nuestro fichero empleados.dat y mostraremos por pantalla que se ha creado.

Posteriormente crearemos un DataOutputStream llamado escribir al que le pasaremos por parámetro el archivo.

Mediante un string llamado cadena guardaremos la cadena de texto tal como indica el ejercicio.

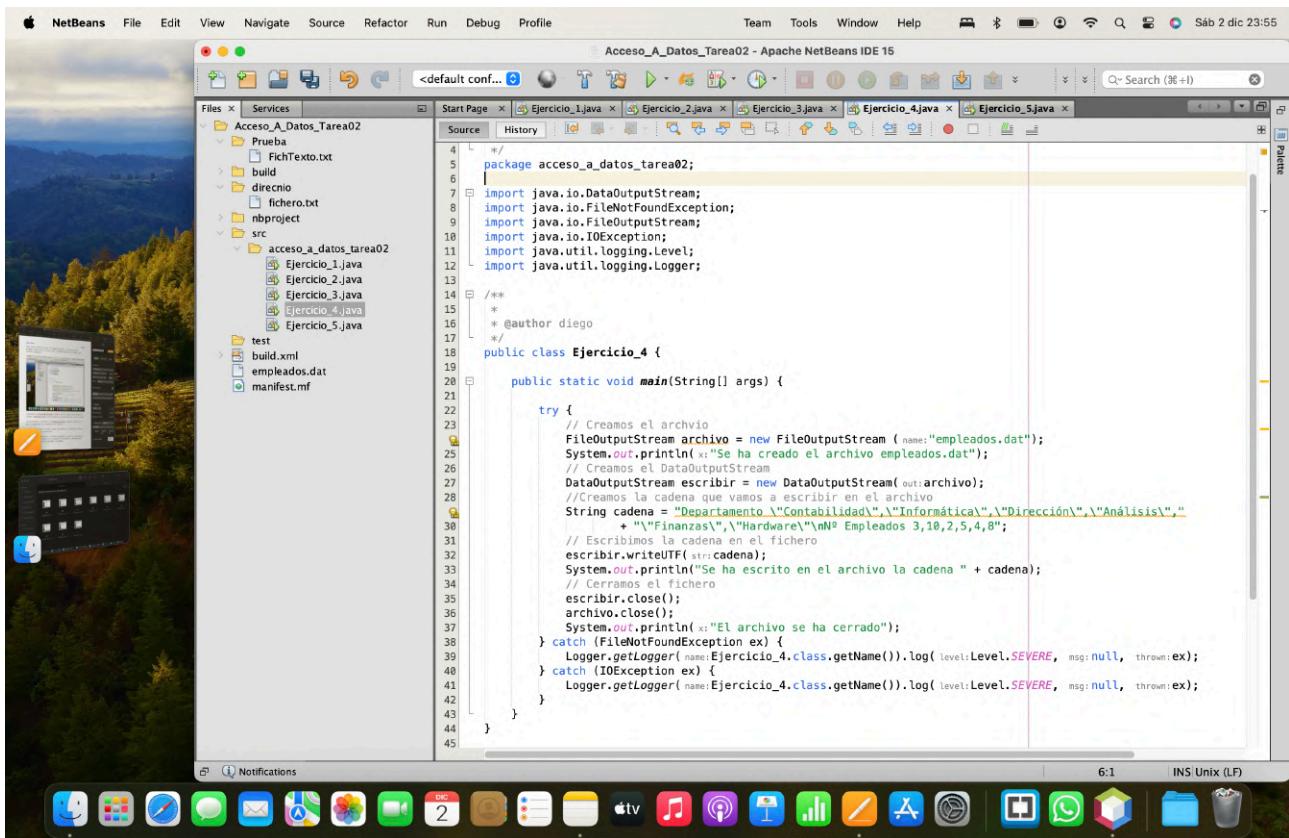
Posteriormente, mediante el método writeUTF al cual le pasaremos por parámetro la cadena, la guardará en escribir.

Lo que haremos será mostrar por pantalla que se ha guardado la cadena.

Por cerramos el DataOutputStream y el FileOutputStream.

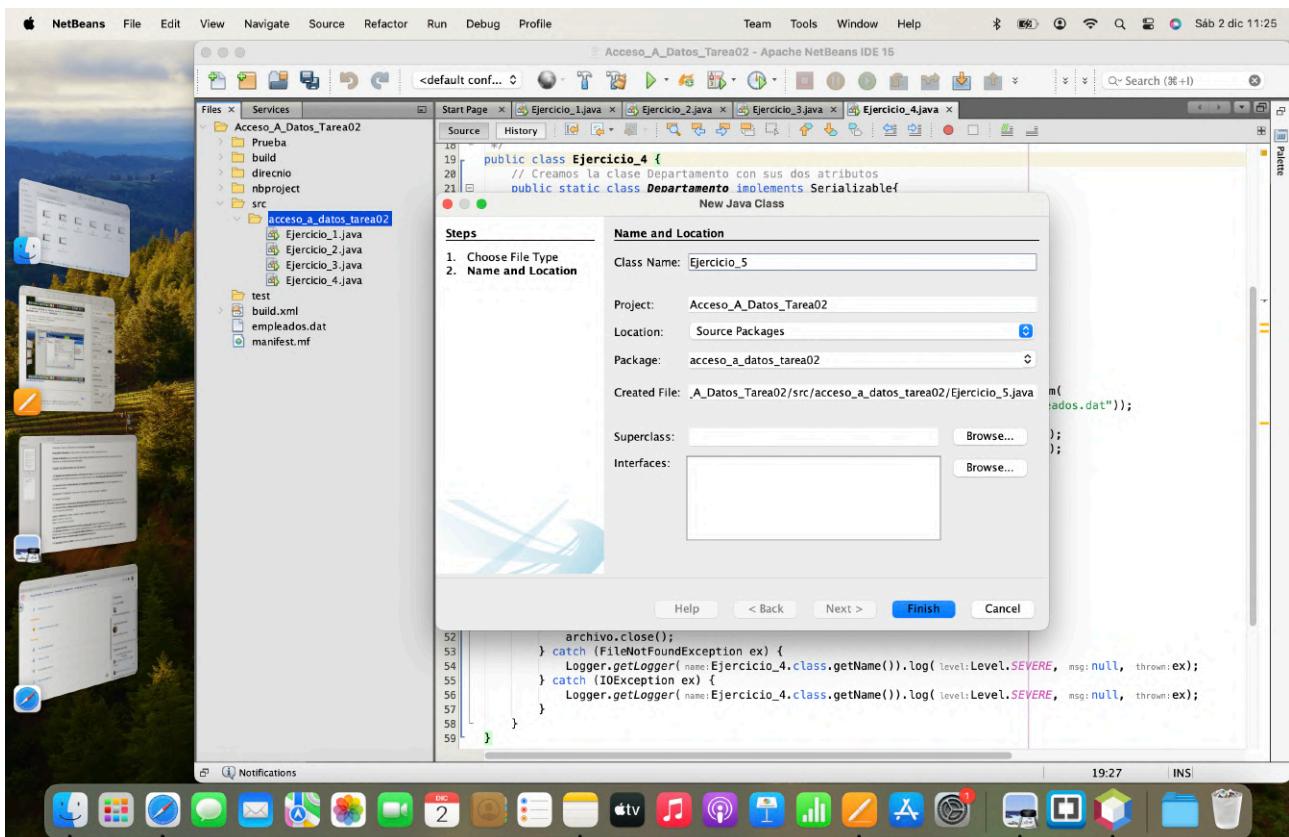
Todo el código lo envolvemos en un try - catch.

En la siguiente imagen podemos ver el código de esta clase.



5. (1 punto) Mostrar la información del fichero binario empleados.dat de forma secuencial creado anteriormente.

Departamento "Contabilidad", "Informática", "Dirección", "Análisis", "Finanzas", "Hardware"
Nº Empleados 3,10,2,5,4,8



Lo primero que haremos, al igual que en los ejercicios anteriores, será crear la clase java llamada Ejercicio_5 dentro de nuestro proyecto, como puede apreciarse en la imagen anterior.

Lo siguiente que haremos será crear el FileInputStream llamado fileIn que contendrá el archivo que le pasamos por parámetro.

Posteriormente crearemos el DataInputStream llamado inputStream al cual le pasaremos por parámetro el fileIn creado anteriormente.

Lo siguiente será crear un StringBuffer llamado cadenaLeida que contendrá lo que vayamos leyendo del archivo.

A continuación creamos un while que mientras que haya datos que leer nos mostrara por pantalla "Leyendo el archivo".

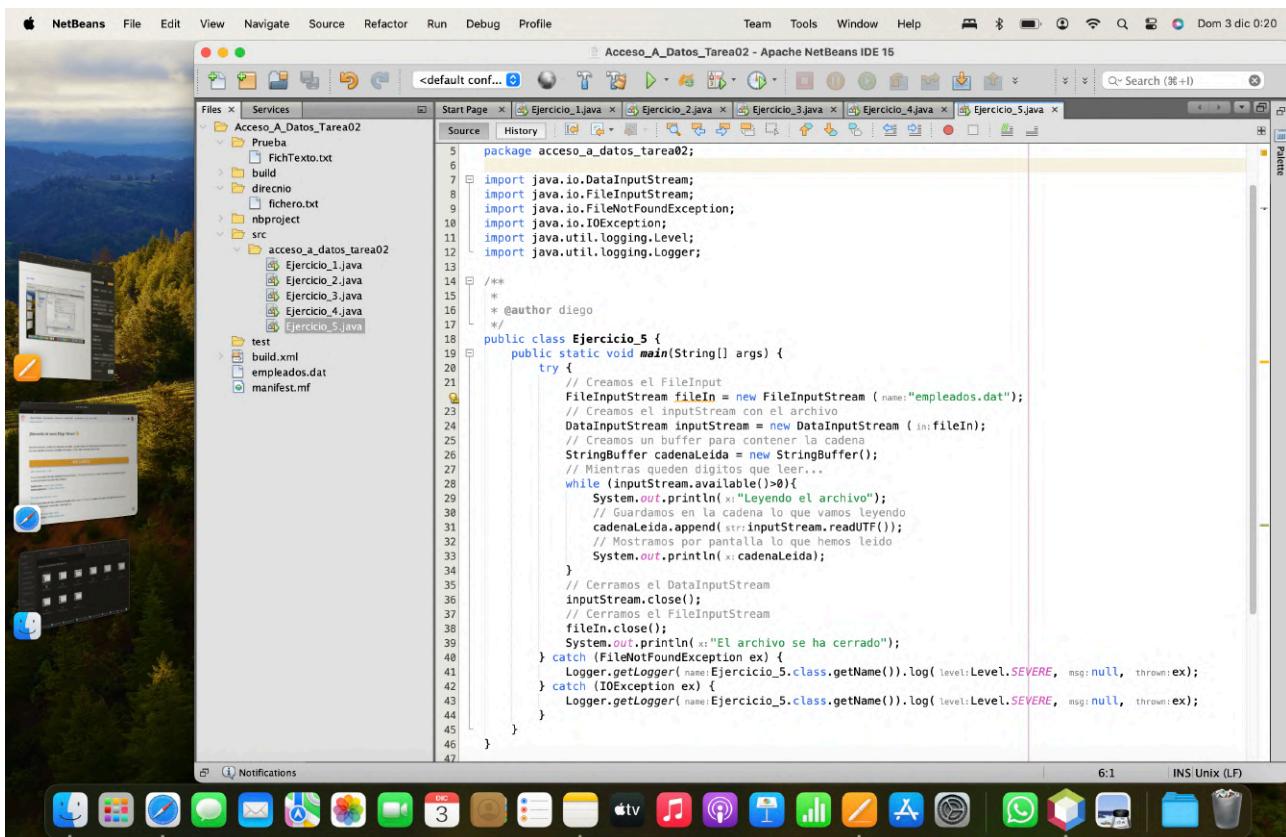
Posteriormente guardará en cadenaLeida mediante el método append lo que va leyendo mediante el método readUTF.

Lo siguiente será mostrar por pantalla los datos leídos que hemos almacenado en cadenaLeida.

Por último cerraremos el DataInputStream llamado inputStream y el FileInputStream llamado fileIn y mostraremos por pantalla que los hemos cerrado.

Todo esto será envuelto en un Try - Catch.

En la siguiente imagen podemos ver el código de esta clase.



```

  package acceso_a_datos_tarea02;
  import java.io.DataInputStream;
  import java.io.FileInputStream;
  import java.io.FileNotFoundException;
  import java.io.IOException;
  import java.util.logging.Level;
  import java.util.logging.Logger;
  /**
   * 
   * @author diego
   */
  public class Ejercicio_5 {
    public static void main(String[] args) {
        try {
            // Creamos el FileInputStream
            FileInputStream fileIn = new FileInputStream ("empleados.dat");
            // Creamos el inputstream con el archivo
            DataInputStream inputStream = new DataInputStream (fileIn);
            // Creamos un buffer para contener la cadena
            StringBuffer cadenaLeida = new StringBuffer();
            // Mientras queden dígitos que leer...
            while (inputStream.available()>0){
                System.out.println("Leyendo el archivo");
                // Guardamos en la cadena lo que vamos leyendo
                cadenaLeida.append(inputStream.readUTF());
                // Mostramos por pantalla lo que hemos leido
                System.out.println(cadenaLeida);
            }
            // Cerramos el DataInputStream
            inputStream.close();
            // Cerramos el FileInputStream
            fileIn.close();
            System.out.println("El archivo se ha cerrado");
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Ejercicio_5.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Ejercicio_5.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

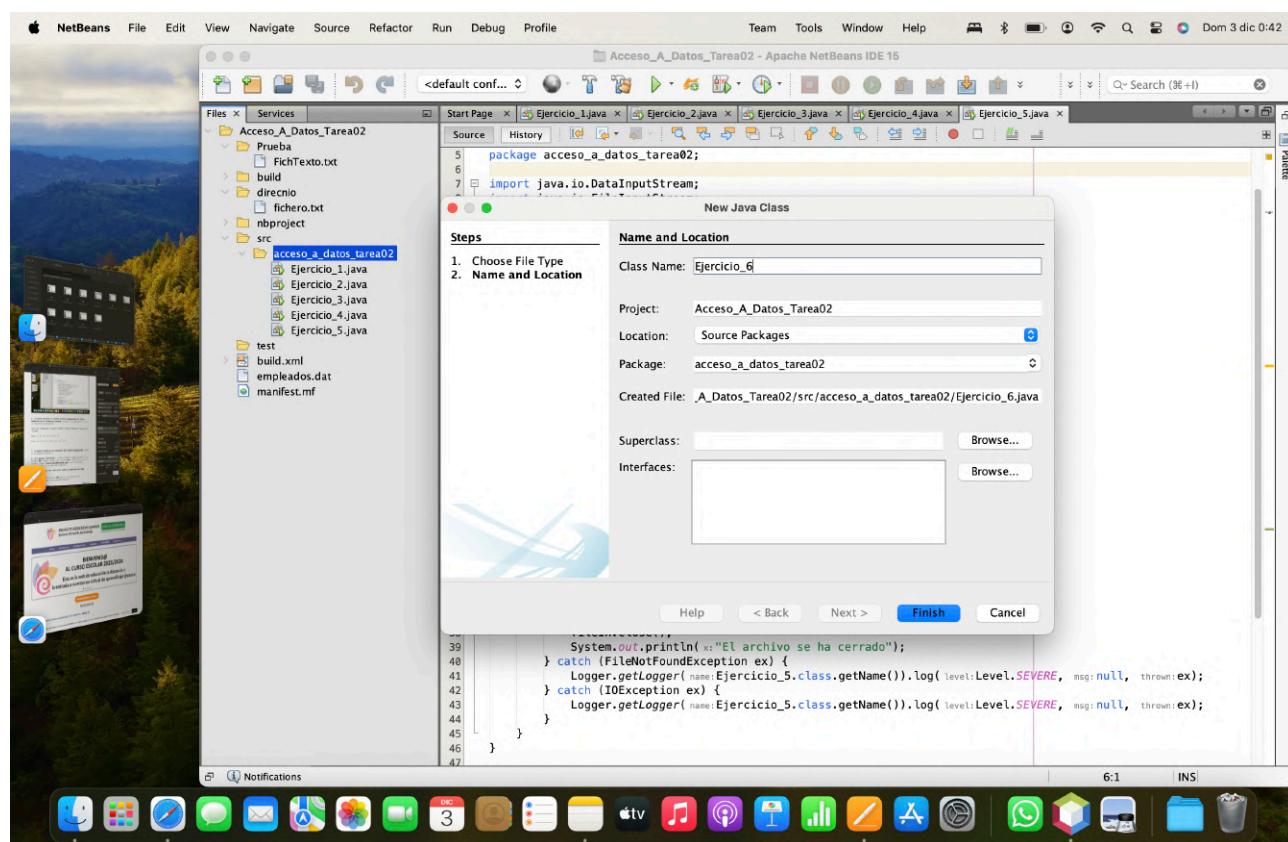
6. (1 punto) Escribe un fichero binario **alumnos.dat** de forma aleatoria en el directorio **prueba** (creado en el apartado 2). Con la siguiente información.

Apellido "FERNANDEZ", "LOPEZ", "GOMEZ", "CHEN", "SERRANO", "CASILLAS", "ALONSO"

Edad 17, 20, 18, 17, 19, 21, 20

Nota 7.5, 4.2, 6.5, 8.0, 3.2, 9.2, 9.9

Lo primero que haremos será crear la clase java llamada **Ejercicio_6** dentro de nuestro proyecto.



Posteriormente crearemos un RandomAccessFile llamado archivo.

A continuación crearemos un string llamado cadena que contendrá lo que queremos escribir en el archivo.

Lo siguiente que haremos será abrir el archivo pasándole la cadena, el nombre y el tipo de acceso que queremos en el.

Mediante un if comprobaremos si el archivo existe (comprobando que no tenga nada escrito mediante el método length).

Si no tiene contenido (que esto sucederá sino no existe) mostraremos por pantalla que el archivo ha sido creado y abierto.

Posteriormente nos situaremos al final del mismo (o mas bien de su contenido) mediante el método seek.

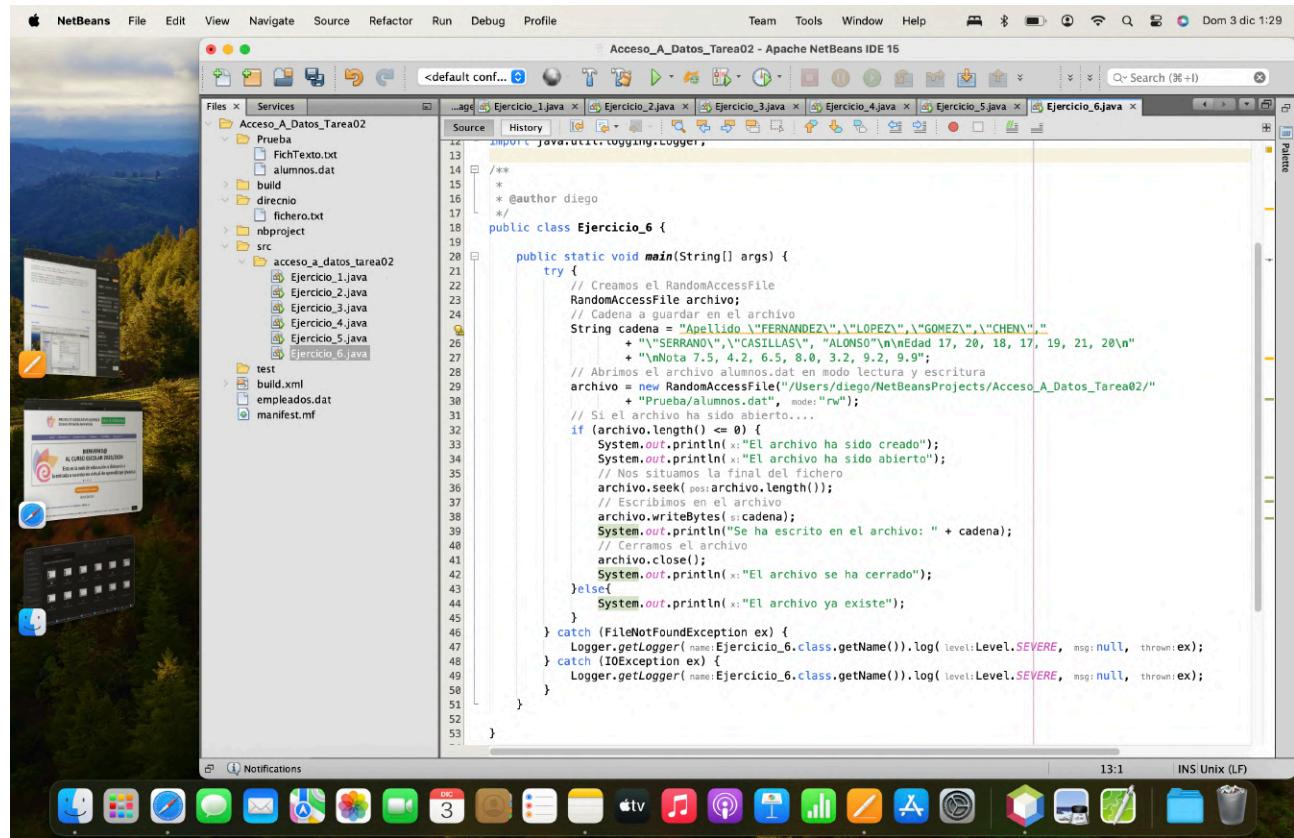
Una vez allí escribiremos en él la cadena creada.

Posteriormente mostraremos por pantalla que hemos escrito en el archivo mas la cadena que hemos escrito.

Una vez escrito en el, cerraremos el archivo y mostraremos por pantalla que lo hemos cerrado.

En la clausula else del if, que saltaría en caso de que el archivo tuviera contenido, mostraremos por pantalla que el archivo ya existe.

En la siguiente imagen podemos ver el código de esta clase.



```

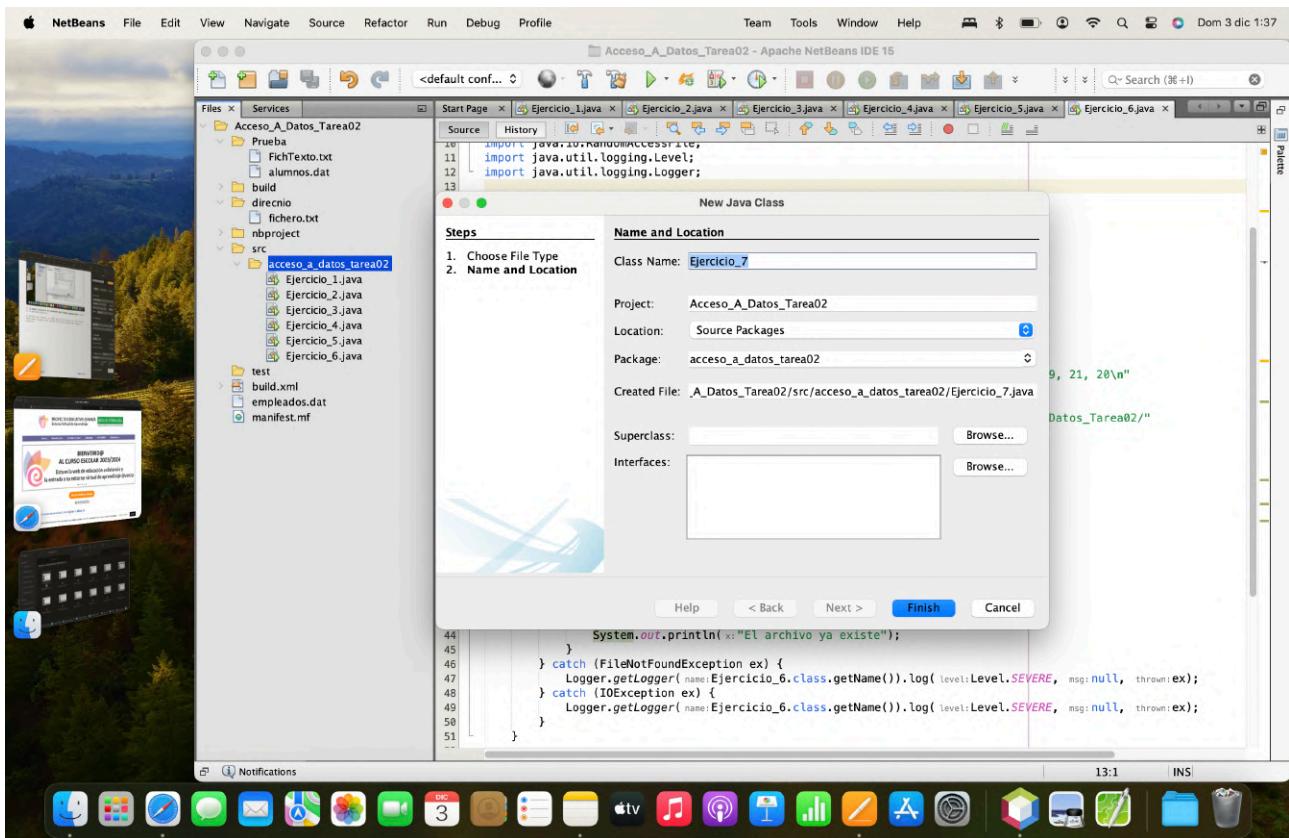
 1 package Ejercicio_6;
 2 
 3 import java.util.logging.Logger;
 4 import java.io.*;
 5 
 6 /**
 7  * 
 8  * @author diego
 9  */
10 public class Ejercicio_6 {
11 
12     public static void main(String[] args) {
13         try {
14             // Creamos el RandomAccessFile
15             RandomAccessFile archivo;
16             // Cadena a guardar en el archivo
17             String cadena = "Apellido \\"FERNANDEZ\\",\\"LOPEZ\\",\\"GOMEZ\\",\\"CHEN\\",\n" +
18                         + "\\"SERRANO\\", 7.5, 4.2, 6.5, 8.0, 3.2, 9.2, 9.9";
19             // Abrimos el archivo alumnos.dat en modo lectura y escritura
20             archivo = new RandomAccessFile("/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/" +
21                 + "Prueba/alumnos.dat", "rw");
22             // Si el archivo ha sido abierto...
23             if (archivo.length() <= 0) {
24                 System.out.println("El archivo ha sido creado");
25                 System.out.println("El archivo ha sido abierto");
26                 // Nos situamos la final del fichero
27                 archivo.seek(archivo.length());
28                 // Escribimos en el archivo
29                 archivo.writeBytes(cadena);
30                 System.out.println("Se ha escrito en el archivo: " + cadena);
31                 // Cerramos el archivo
32                 archivo.close();
33                 System.out.println("El archivo se ha cerrado");
34             } else {
35                 System.out.println("El archivo ya existe");
36             }
37         } catch (FileNotFoundException ex) {
38             Logger.getLogger(Ejercicio_6.class.getName()).log(Level.SEVERE, null, ex);
39         } catch (IOException ex) {
40             Logger.getLogger(Ejercicio_6.class.getName()).log(Level.SEVERE, null, ex);
41         }
42     }
43 }
44
45
46
47
48
49
50
51
52
53
}

```

7. (1 punto) Visualiza el contenido del fichero alumnos.dat creado en el apartado anterior.

Lo primero que haremos, al igual que en los ejercicios anteriores, será crear la clase java llamada Ejercicio_7 dentro de nuestro proyecto.

En la siguiente imagen puede verse como ha sido creada la clase.



Lo primero que haremos será crear el RandomAccessFile llamado archivo para acceder de forma aleatoria.

Posteriormente crearemos un array de bytes llamado array para almacenar los bytes que contiene el archivo.

Lo siguiente que haremos será crear un String llamado cadena que contendrá la cadena de texto a mostrar.

A continuación indicamos por consola que abrimos el archivo e instanciamos el RandomAccessFile con la ruta donde se encuentra el archivo.

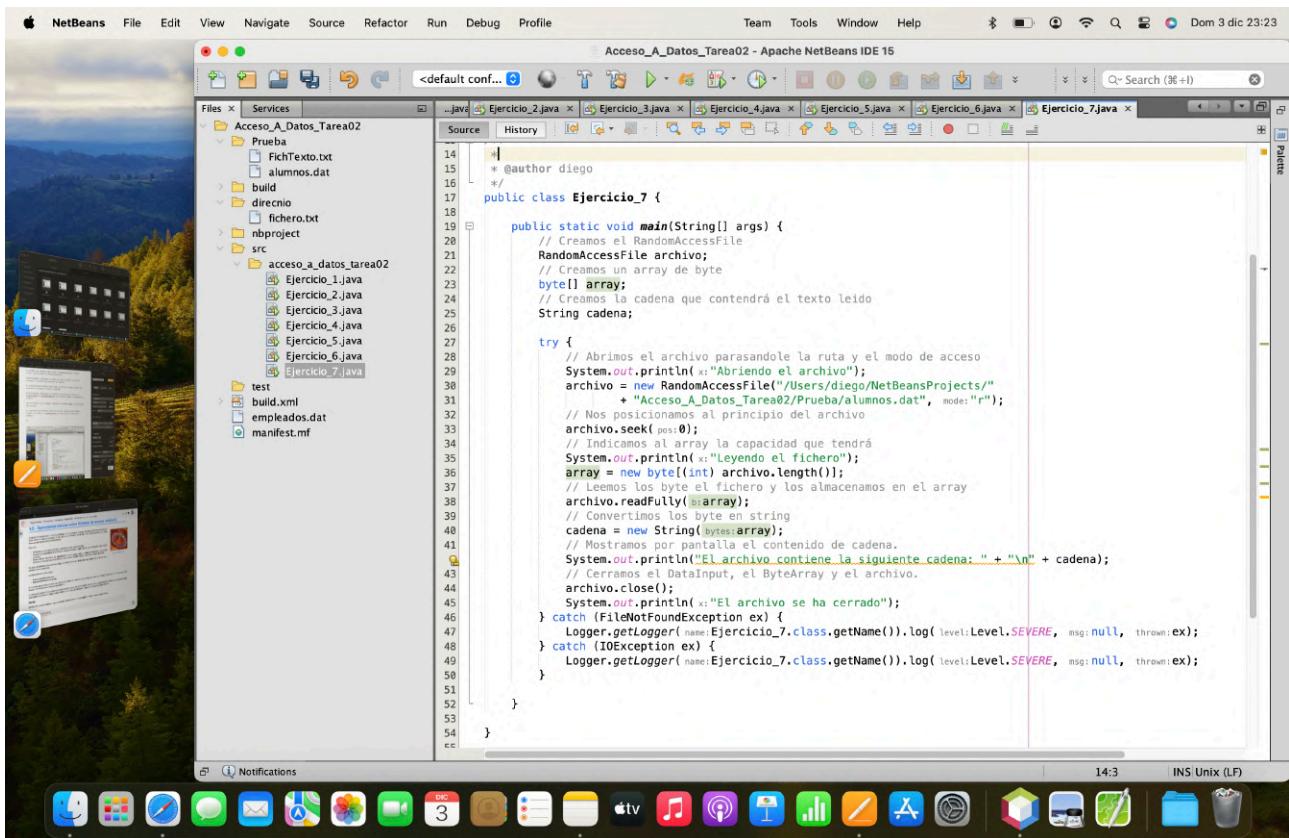
Posteriormente nos situamos, mediante el método seek, al principio del archivo.

Lo siguiente que haremos será mostrar por consola que estamos leyendo el archivo e le indicaremos al array la capacidad que contendrá, en nuestro caso, la longitud del archivo y leemos, mediante el método readFully, el archivo y guardamos lo leído en el array.

Posteriormente guardaremos en la variable cadena el contenido del array y mostraremos por consola lo que contiene dicha cadena.

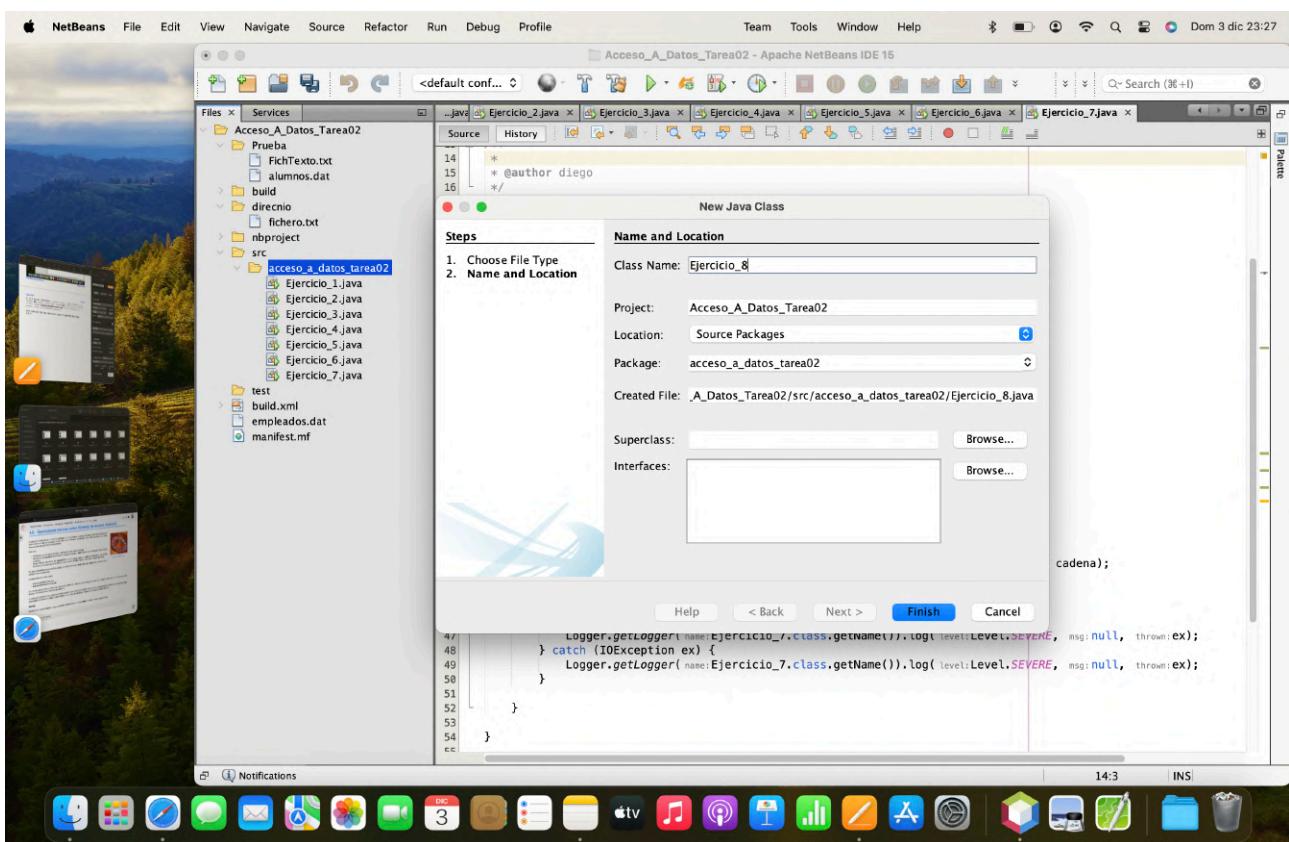
Lo último que haremos será cerrar el archivo e indicar que lo hemos cerrado.

Todo esto estará envuelto en un try - catch.



8. (1,5 puntos) Convierte el fichero binario (.dat) alumnos a un fichero XML llamado alumnos.xml, usando DOM. Para hacerlo crea el fichero llamado **CrearAlumnoXMLconDOM.java**. en el directorio prueba, creado en un apartado anterior.

Este ejercicio hay que realizarlo según el apartado del tema 6.4.3.



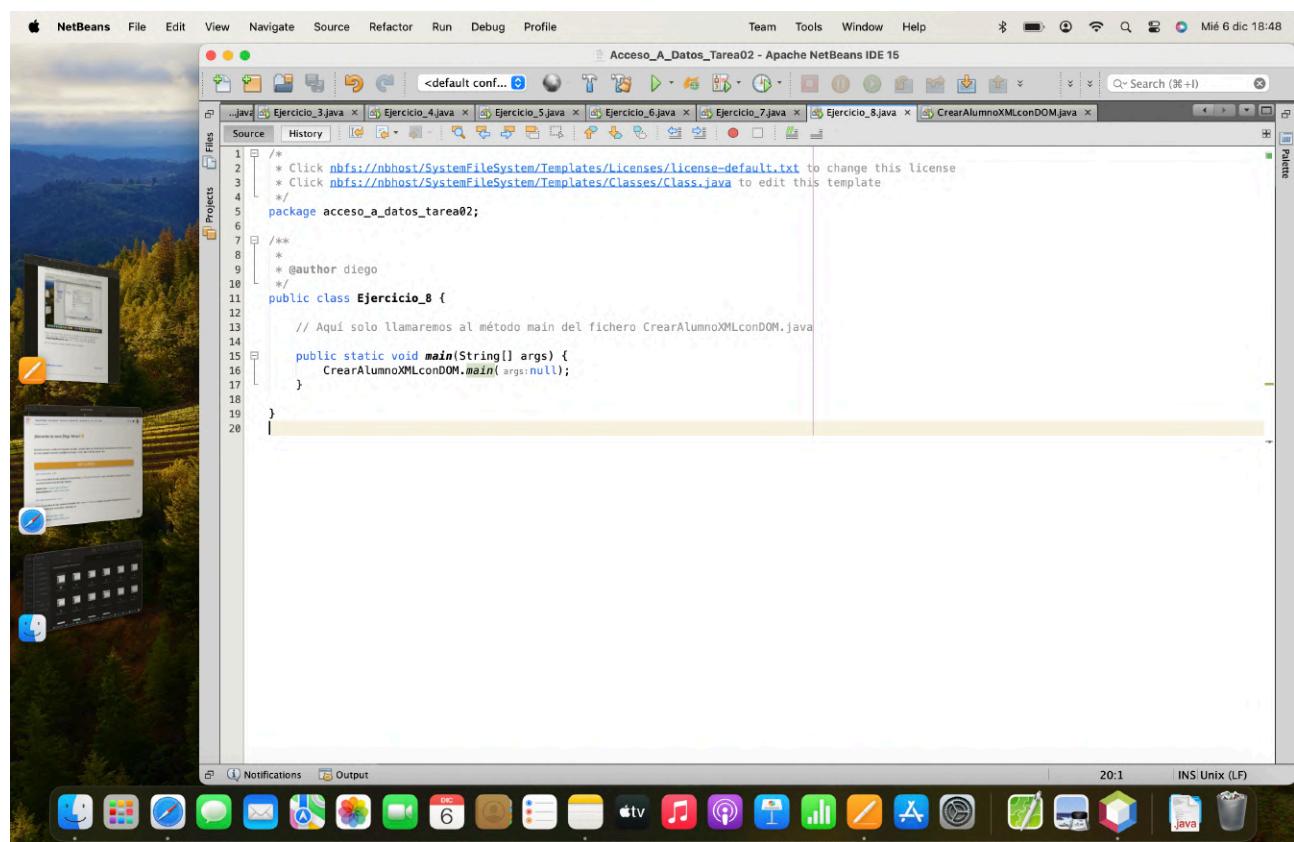
Lo primero que haremos, como puede apreciarse en la imagen anterior, al igual que en los ejercicios anteriores, será crear la clase java llamada Ejercicio_8 dentro de nuestro proyecto.

Esto lo haremos para seguir un poco la lógica que llevamos de crear un fichero java con el nombre de cada ejercicio.

Dentro de esta clase java solo llamaremos al método main de la clase java que crearemos a continuación a la que llamaremos (**CrearAlumnoXMLconDOM.java**) tal y como indica el enunciado.

Posteriormente crearemos la clase CrearAlumnoXMLconDOM.java, como se indica en el ejercicio.

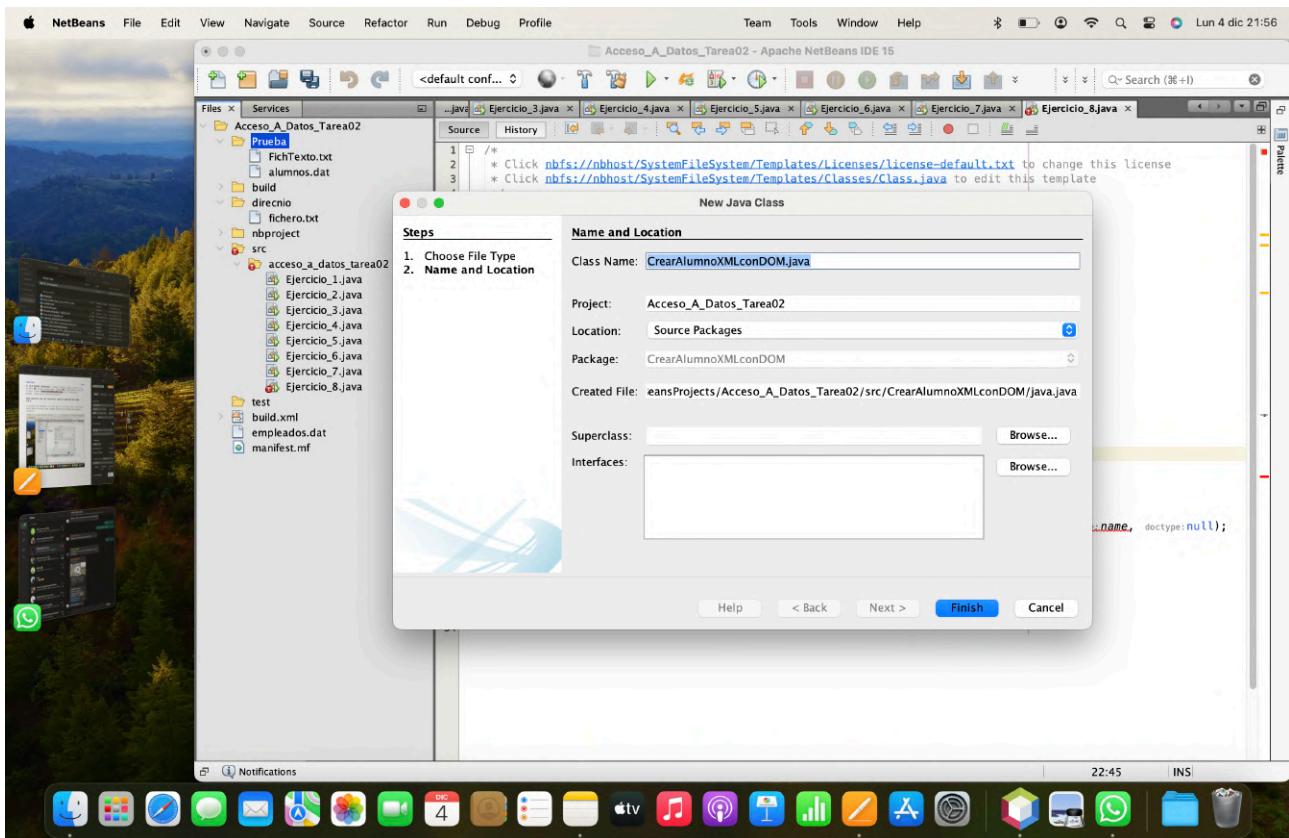
En la siguiente imagen podemos ver el código de esta clase.



The screenshot shows the Apache NetBeans IDE 15 interface. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a date/time indicator "Mié 6 dic 18:48". The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar shows a "Projects" view with several Java files listed: Ejercicio_3.java, Ejercicio_4.java, Ejercicio_5.java, Ejercicio_6.java, Ejercicio_7.java, Ejercicio_8.java, and CrearAlumnoXMLconDOM.java. The main editor area displays the code for Ejercicio_8.java:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package acceso_a_datos_tarea02;
6
7  /**
8   *
9   * @author diego
10  */
11 public class Ejercicio_8 {
12
13     // Aquí solo llamaremos al método main del fichero CrearAlumnoXMLconDOM.java
14
15     public static void main(String[] args) {
16         CrearAlumnoXMLconDOM.main(args);
17     }
18
19 }
20
```

Posteriormente, vamos a crear la clase java CrearAlumnoXMLconDOM.java, tal como indica el ejercicio, que contendrá toda la lógica para crear un archivo xml con DOM.



Lo primero que haremos en la clase `CrearAlumnoXMLconDOM.java`, será guardar el archivo `alumnos.dat` en arrays y así acceder a la información para poder incluirla en el xml. Para ello hemos copiado el código del ejercicio 7, comentando los sout que mostraban por pantalla lo que se estaba haciendo para dejar mas limpia la salida por consola.

Al tener ya el archivo `.dat` en un array de tipo string y por tanto, conocer la ubicación de cada dato, vamos a crear tres arrays de tipo string para guardar los apellidos, la edad y la nota, llamados `textoApe`, `textoEdad` y `textoNota` respectivamente.

Mediante bucles que recorren las posiciones exactas de donde están cada conjunto de datos vamos a asignar los campos que queremos del array `textoSepPrinc` en los tres arrays creados. También dejamos comentado los bucles `for` en los que se puede Mostar por consola el array creado.

A partir de aquí ya empezamos con el código para crear el xml.

Lo primero que haremos será declarar e instanciar un `DocumentBuilderFactory` llamado `factoría` mediante su método `newInstance`.

Posteriormente declaramos un `DocumentBuilder` llamado `builder` y lo inicializamos a null para posteriormente, envuelto en `try - catch`, inicializar el builder pasándole la factoría creada anteriormente mediante el método `newDocumentBuilder`

A continuación vamos a crear un Document llamado documento y lo vamos a inicializar pasándole el builder creado anteriormente mediante el método newDocument.

Lo siguiente que haremos será crear el elemento raíz de nuestro documento mediante un Element llamado alumnos y el método createElement para posteriormente, mediante el método appendChild, decirle que es hijo de documento.

Como los alumnos son siete, con sus siete edades y sus siete notas, vamos a envolver en un bucle for (de longitud siete), la creación de cada alumno.

Mediante tres Element llamados alumno, apellido, edad y notas respectivamente crearemos los nodos hijos.

Al Element alumno le decimos, mediante el método appendChild, que es hijo de alumnos.

El Element apellido, mediante el método setTextContent, al cual de pasamos el array textoApe y la posición a la que deseamos acceder (que es la posición donde se encuentra la variable i del bucle for), contendrá cada uno de los apellidos y posteriormente le diremos que es hijo de alumno mediante el método appendChild.

El Element edad, mediante el método setTextContent, al cual de pasamos el array textoEdad y la posición a la que deseamos acceder (que es la posición donde se encuentra la variable i del bucle for), contendrá cada una de las edades y posteriormente le diremos que es hijo de alumno mediante el método appendChild.

El Element nota, mediante el método setTextContent, al cual de pasamos el array textoNota y la posición a la que deseamos acceder (que es la posición donde se encuentra la variable i del bucle for), contendrá cada una de las notas y posteriormente le diremos que es hijo de alumno mediante el método appendChild.

Tras ello cerramos el bucle for.

Posteriormente declaramos un TransformerFactory llamado transformerfactory y lo inicializamos mediante el método newInstance.

A continuación declaramos un Transformer llamado transformar y lo inicializamos a null.

Lo siguiente que haremos será pasarle el transformerfactory, mediante el método newTransformer, al transformar instanciado a null anteriormente, envuelto en try - catch.

Tras ello, vamos a declarar un DOMSource llamado dom al cual le pasaremos el documento creado anteriormente.

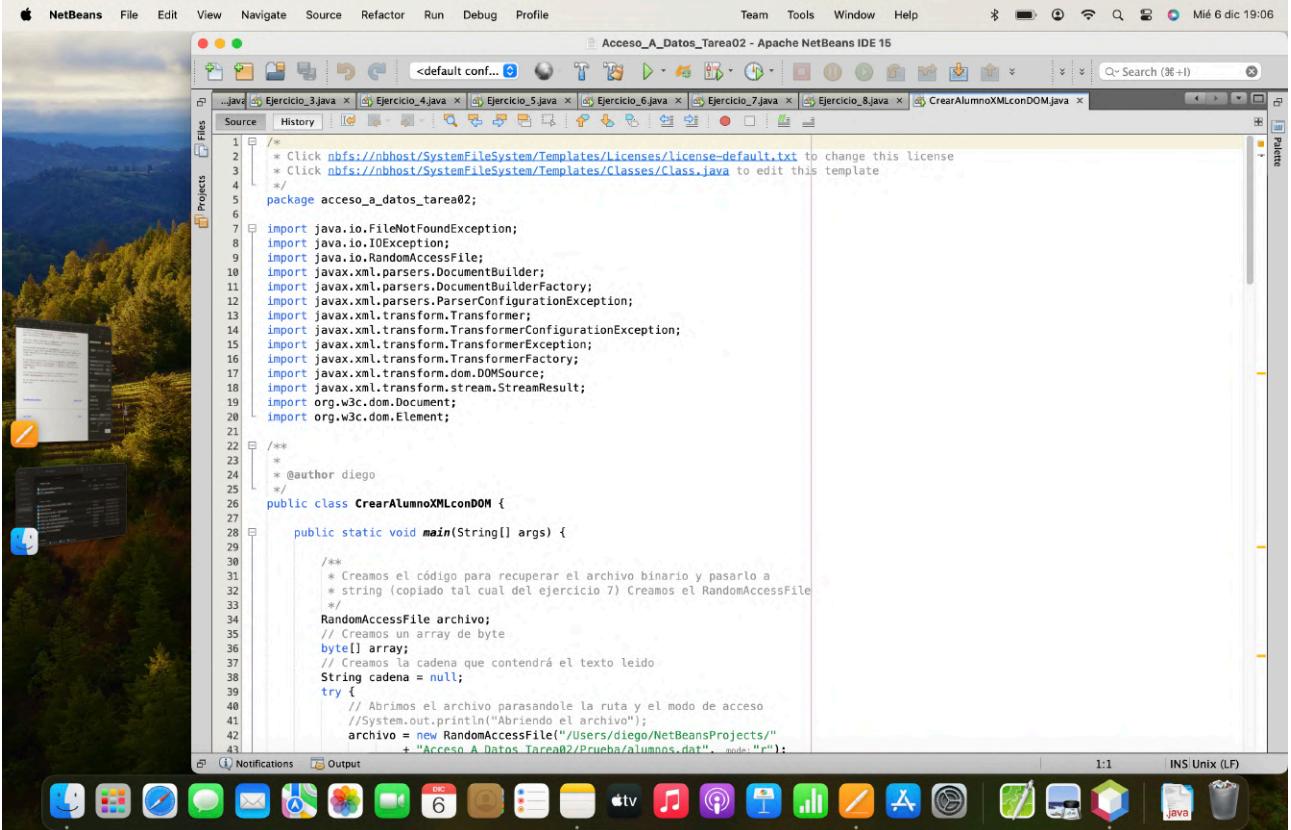
Posteriormente declaramos un `StreamResult` llamado `sr` al cual le pasaremos el path donde queremos que se guarde nuestro archivo xml creado seguido por el nombre que queremos darle al archivo y su extensión.

A continuación aplicaremos el método `transform` al transformer creado anteriormente y le pasaremos el source llamado `dom` y el `StreamResult` creados anteriormente, y todo ello, envuelto en un `try - catch`.

Por último mostramos por pantalla el mensaje "El fichero se ha creado correctamente" si todo ha salido bien.

En las siguientes imágenes se puede ver el código que contiene esta clase.

Con ellas, damos por finalizado este ejercicio.



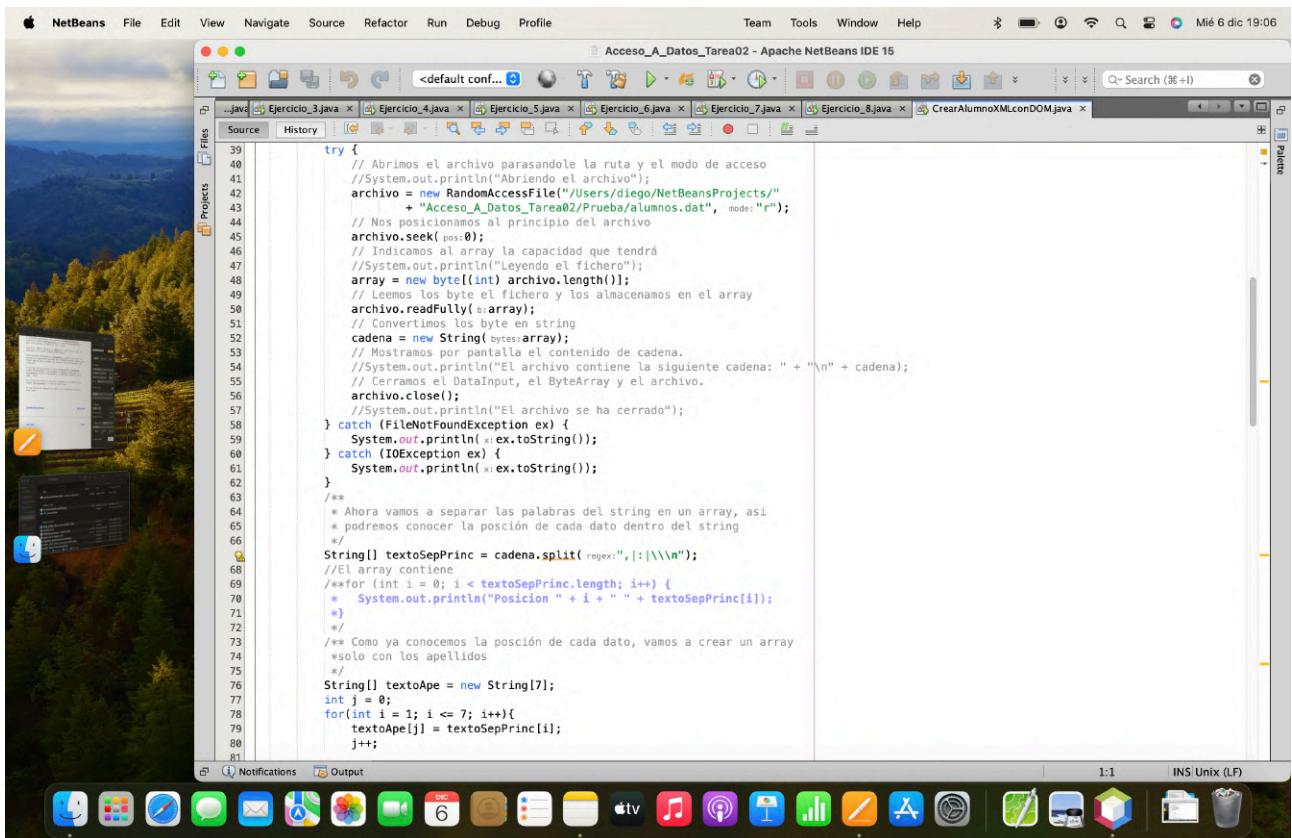
The screenshot shows the Apache NetBeans IDE interface. The title bar reads "NetBeans Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The main window displays a Java code editor with the following content:

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package acceso_a_datos_tarea02;
6
7  import java.io.FileNotFoundException;
8  import java.io.IOException;
9  import java.io.RandomAccessFile;
10 import javax.xml.parsers.DocumentBuilder;
11 import javax.xml.parsers.DocumentBuilderFactory;
12 import javax.xml.parsers.ParserConfigurationException;
13 import javax.xml.transform.Transformer;
14 import javax.xml.transform.TransformerConfigurationException;
15 import javax.xml.transform.TransformerException;
16 import javax.xml.transform.TransformerFactory;
17 import javax.xml.transform.dom.DOMSource;
18 import javax.xml.transform.stream.StreamResult;
19 import org.w3c.dom.Document;
20 import org.w3c.dom.Element;
21
22 /**
23  * @author diego
24  */
25 public class CrearAlumnoXMLconDOM {
26
27     public static void main(String[] args) {
28
29         /**
30          * Creamos el código para recuperar el archivo binario y pasarlo a
31          * string (copiado tal cual del ejercicio 7) Creamos el RandomAccessFile
32          */
33         RandomAccessFile archivo;
34         // Creamos un array de byte
35         byte[] array;
36         // Creamos la cadena que contendrá el texto leído
37         String cadena = null;
38         try {
39             // Abrimos el archivo pasándole la ruta y el modo de acceso
40             //System.out.println("Abriendo el archivo");
41             archivo = new RandomAccessFile("/Users/diego/NetBeansProjects/" +
42                 + "Acceso_A_Datos_Tarea02/Prueba/alumnos.dat", "r");
43         }

```

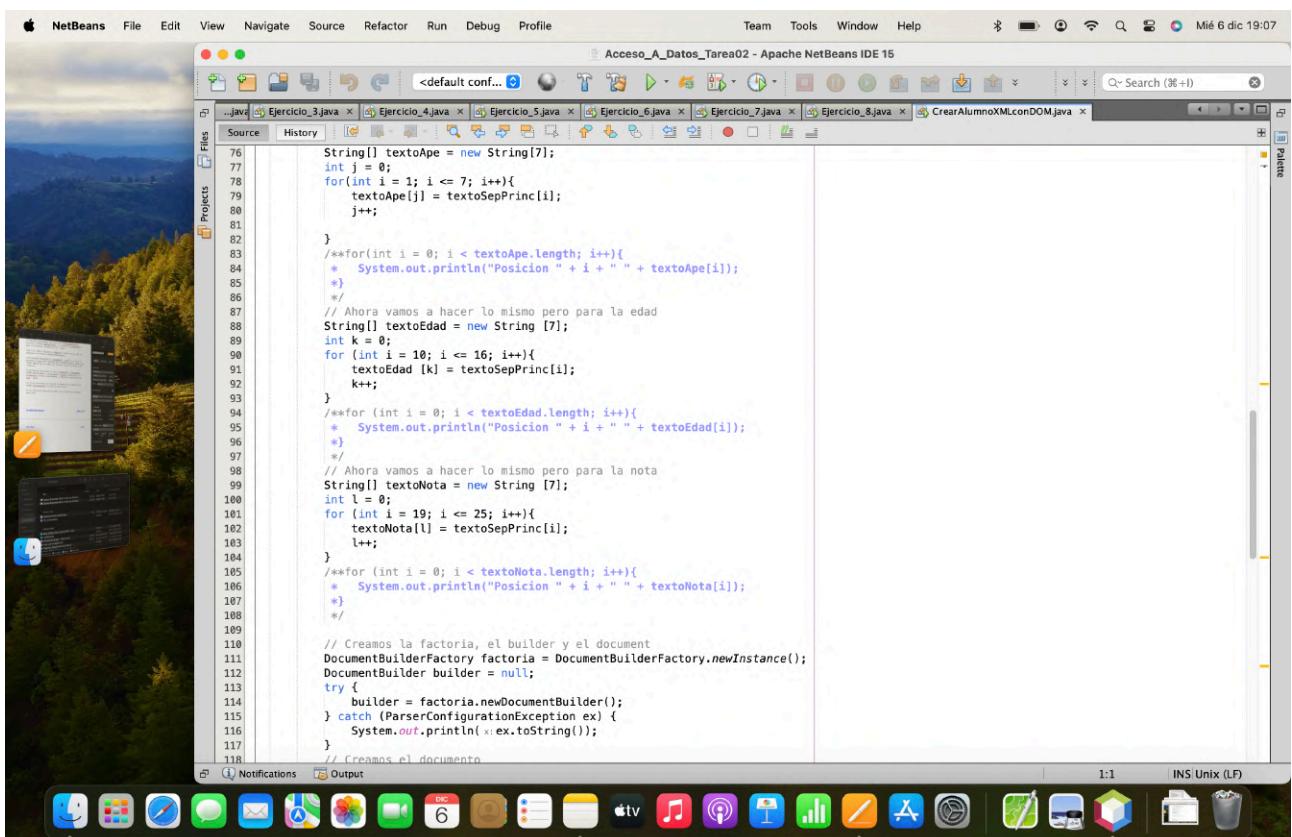
The code implements a `CrearAlumnoXMLconDOM` class with a `main` method. It uses `RandomAccessFile` to read binary data from a file named "alumnos.dat" and converts it into a `String`. This string is then used to create a `Document` object via `DocumentBuilder`. Finally, the `Transformer` is used to transform the `Document` into an `StreamResult`, which is then written to a file named "alumnos.xml".



```

try {
    // Abrimos el archivo pasandole la ruta y el modo de acceso
    //System.out.println("Abriendo el archivo");
    archivo = new RandomAccessFile("/Users/diego/NetBeansProjects/" +
        + "Acceso_A_Datos_Tarea02/Prueba/alumnos.dat", "r");
    // Nos posicionamos al principio del archivo
    archivo.seek(0);
    // Indicamos al array la capacidad que tendrá
    //System.out.println("Leyendo el fichero");
    array = new byte[(int) archivo.length()];
    // Leemos los byte del fichero y los almacenamos en el array
    archivo.readFully(array);
    // Convertimos los byte en string
    cadena = new String(array);
    // Mostramos por pantalla el contenido de cadena.
    //System.out.println("El archivo contiene la siguiente cadena: " + "\n" + cadena);
    // Cerramos el DataInputStream, el ByteArray y el archivo.
    archivo.close();
    //System.out.println("El archivo se ha cerrado");
} catch (FileNotFoundException ex) {
    System.out.println(ex.toString());
} catch (IOException ex) {
    System.out.println(ex.toString());
}
*/
* Ahora vamos a separar las palabras del string en un array, asi
* podremos conocer la posición de cada dato dentro del string
*/
String[] textoSepPrinc = cadena.split(regex:"|:\\\\n");
//El array contiene
/*for (int i = 0; i < textoSepPrinc.length; i++) {
    * System.out.println("Posicion " + i + " " + textoSepPrinc[i]);
}
*/
/** Como ya conocemos la posición de cada dato, vamos a crear un array
*solo con los apellidos
*/
String[] textoApe = new String[7];
int j = 0;
for(int i = 1; i <= 7; i++){
    textoApe[j] = textoSepPrinc[i];
    j++;
}
*/
**for(int i = 0; i < textoApe.length; i++){
    * System.out.println("Posicion " + i + " " + textoApe[i]);
}
*/
// Ahora vamos a hacer lo mismo pero para la edad
String[] textoEdad = new String [7];
int k = 0;
for (int i = 10; i <= 16; i++){
    textoEdad [k] = textoSepPrinc[i];
    k++;
}
*/
**for (int i = 0; i < textoEdad.length; i++){
    * System.out.println("Posicion " + i + " " + textoEdad[i]);
}
*/
// Ahora vamos a hacer lo mismo pero para la nota
String[] textoNota = new String [7];
int l = 0;
for (int i = 19; i <= 25; i++){
    textoNota[l] = textoSepPrinc[i];
    l++;
}
*/
**for (int i = 0; i < textoNota.length; i++){
    * System.out.println("Posicion " + i + " " + textoNota[i]);
}
*/
// Creamos la factoria, el builder y el document
DocumentBuilderFactory factoria = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = null;
try {
    builder = factoria.newDocumentBuilder();
} catch (ParserConfigurationException ex) {
    System.out.println(ex.toString());
}
// Creamos el documento

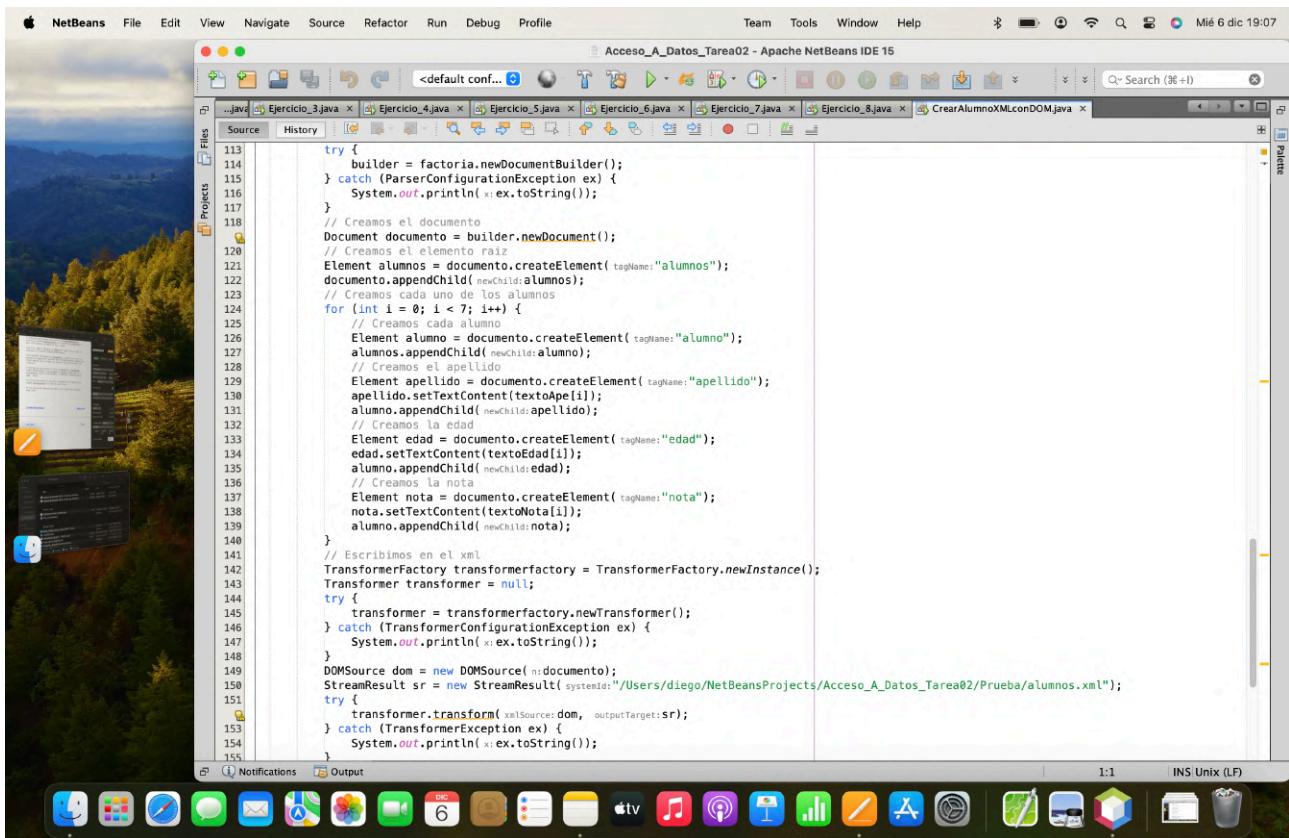
```



```

DocumentBuilderFactory factoria = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = null;
try {
    builder = factoria.newDocumentBuilder();
} catch (ParserConfigurationException ex) {
    System.out.println(ex.toString());
}
// Creamos el documento

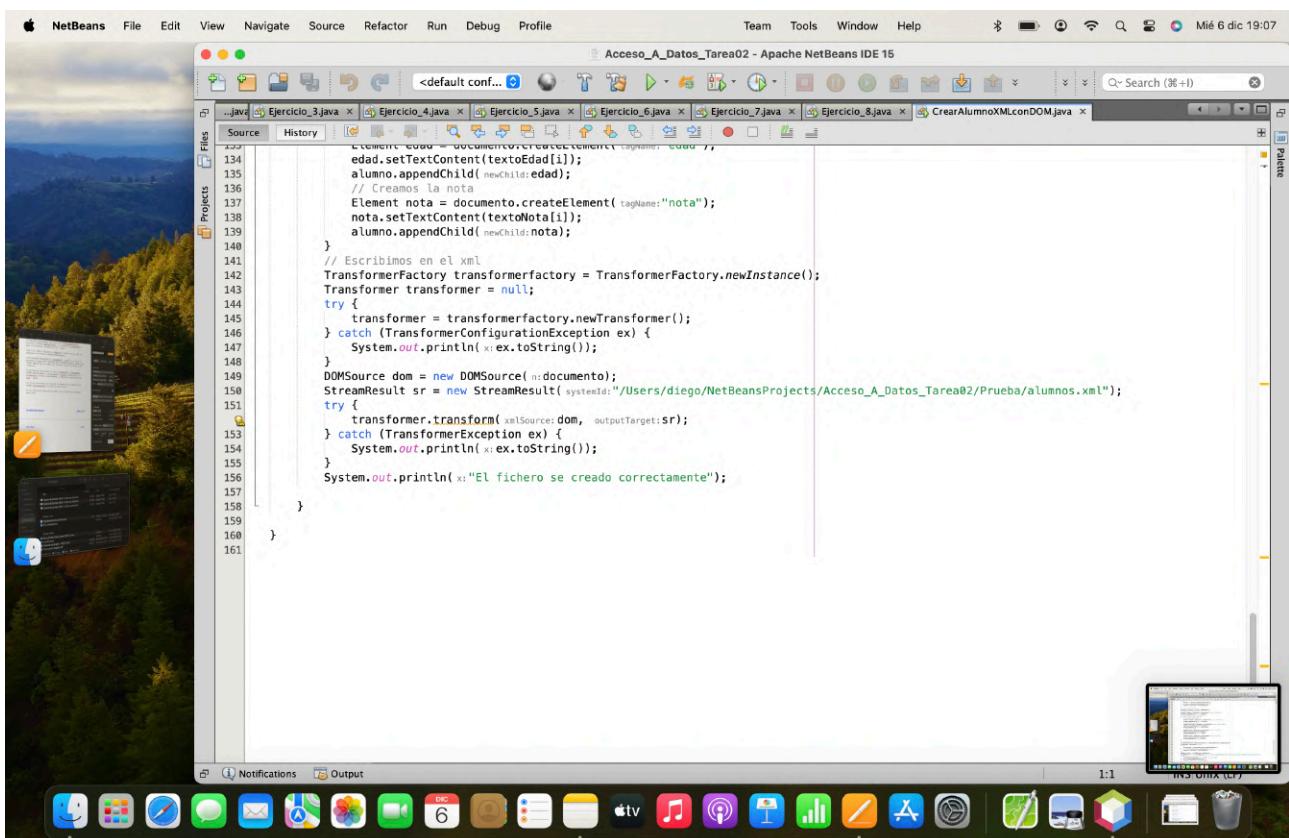
```



```

113     try {
114         builder = factoria.newDocumentBuilder();
115     } catch (ParserConfigurationException ex) {
116         System.out.println(x:ex.toString());
117     }
118     // Creamos el documento
119     Document documento = builder.newDocument();
120     // Creamos el elemento raiz
121     Element alumnos = documento.createElement(tagName:"alumnos");
122     documento.appendChild(newChild:alumnos);
123     // Creamos cada uno de los alumnos
124     for (int i = 0; i < 7; i++) {
125         // Creamos cada alumno
126         Element alumno = documento.createElement(tagName:"alumno");
127         alumnos.appendChild(newChild:alumno);
128         // Creamos el apellido
129         Element apellido = documento.createElement(tagName:"apellido");
130         apellido.setTextContent(textoApellido[i]);
131         alumno.appendChild(newChild:apellido);
132         // Creamos la edad
133         Element edad = documento.createElement(tagName:"edad");
134         edad.setTextContent(textoEdad[i]);
135         alumno.appendChild(newChild:edad);
136         // Creamos la nota
137         Element nota = documento.createElement(tagName:"nota");
138         nota.setTextContent(textoNota[i]);
139         alumno.appendChild(newChild:nota);
140     }
141     // Escribimos en el xml
142     TransformerFactory transformerfactory = TransformerFactory.newInstance();
143     Transformer transformer = null;
144     try {
145         transformer = transformerfactory.newTransformer();
146     } catch (TransformerConfigurationException ex) {
147         System.out.println(x:ex.toString());
148     }
149     DOMSource dom = new DOMSource(documento);
150     StreamResult sr = new StreamResult(systemId:"/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/Prueba/alumnos.xml");
151     try {
152         transformer.transform(dom, outputTarget:sr);
153     } catch (TransformerException ex) {
154         System.out.println(x:ex.toString());
155     }
156 }
157
158
159
160
161

```



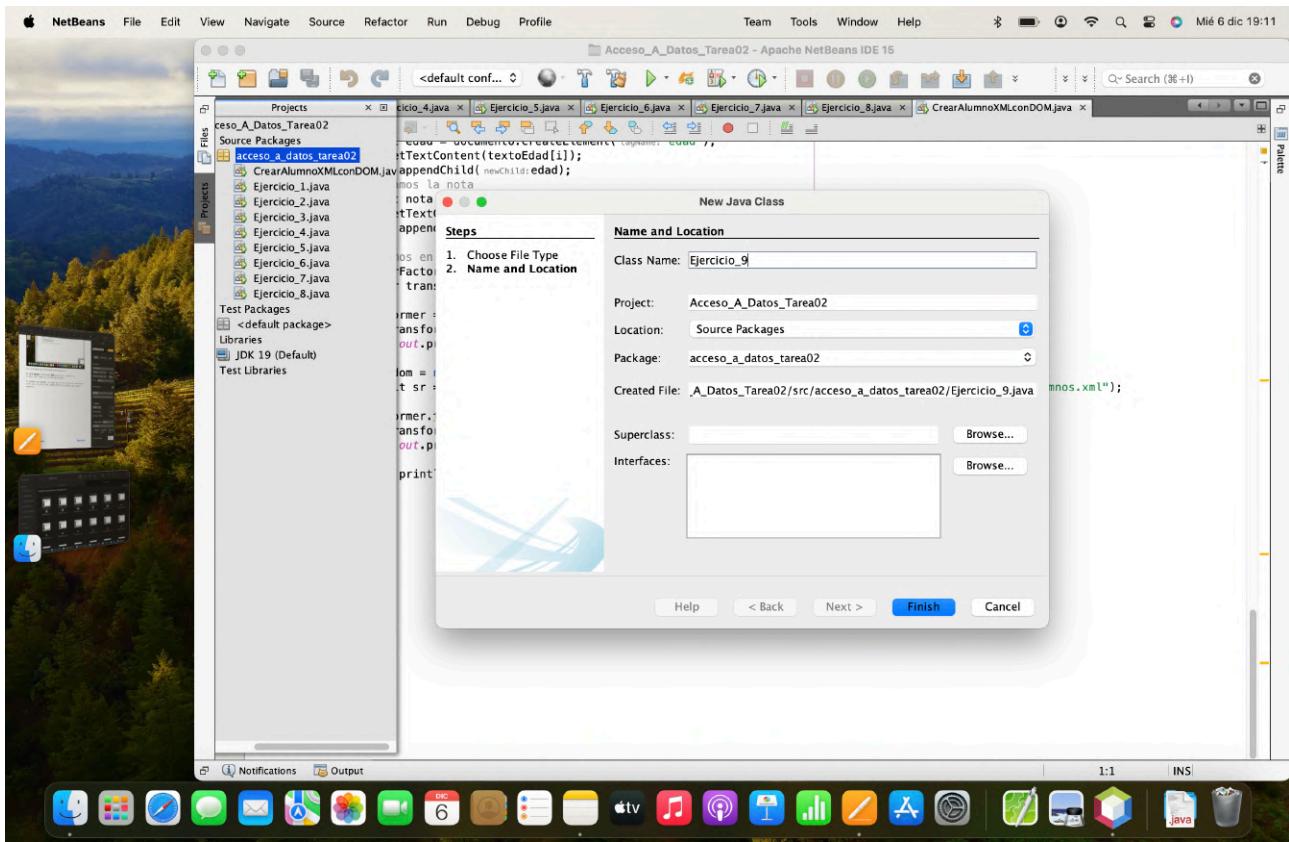
```

132         edad.setTextContent(textoEdad[i]);
133         alumno.appendChild(newChild:edad);
134         // Creamos la nota
135         Element nota = documento.createElement(tagName:"nota");
136         nota.setTextContent(textoNota[i]);
137         alumno.appendChild(newChild:nota);
138     }
139     // Escribimos en el xml
140     TransformerFactory transformerfactory = TransformerFactory.newInstance();
141     Transformer transformer = null;
142     try {
143         transformer = transformerfactory.newTransformer();
144     } catch (TransformerConfigurationException ex) {
145         System.out.println(x:ex.toString());
146     }
147     DOMSource dom = new DOMSource(documento);
148     StreamResult sr = new StreamResult(systemId:"/Users/diego/NetBeansProjects/Acceso_A_Datos_Tarea02/Prueba/alumnos.xml");
149     try {
150         transformer.transform(dom, outputTarget:sr);
151     } catch (TransformerException ex) {
152         System.out.println(x:ex.toString());
153     }
154     System.out.println(x:"El fichero se creado correctamente");
155
156
157
158
159
160
161

```

9. (1,5 puntos) Utilizando **JAXB** muestra por pantalla el fichero .xml creado en el apartado anterior.

Lo primero que haremos, al igual que en los ejercicios anteriores, será crear la clase java llamada Ejercicio_9 dentro de nuestro proyecto.



En nuestro caso vamos a crear todo el código en una sola clase java (Ejercicio_9) para mantener la consonancia del ejercicio pero lo ideal hubiera sido crear varias clases.

Dentro del main empezaremos declarando el JAXBContext llamado contexto mediante el método newInstance al cual le pasaremos el nombre de la clase que contiene el elemento principal del xml, en nuestro caso la clase Alumnos.

Posteriormente declararemos el Unmarshaller llamado unmarshaller, el cual instanciaremos mediante el método createUnmarshaller aplicado al context creado anteriormente.

A continuación declaramos un objeto de tipo Alumnos llamado alumnos, el cual instanciaremos haciendo un cast al unmarshaller creado anteriormente, al cual se le ha aplicado el método unmarshal y pasado a dicho método el objeto file que contiene el xml.

Luego volveremos al main para crear lo que nos falta ya que para ello necesitamos crear previamente las dos clases que alojaran el contenido del xml, y eso es los que haremos a continuación.

Vamos a crear una clase pública y estática llamada Alumnos. A esta clase le pondremos la notación @XmlRootElement y le diremos que corresponde al elemento alumnos del xml.

Dentro de esta clase crearemos un único atributo privado. Este atributo será un ArrayList de elementos alumno llamado alumnos. Crearemos su método constructor y los getter and setter. En el método get le pondremos la notación @XmlElement y le diremos que corresponde al elemento alumno del xml.

Ahora crearemos la clase Alumno, que como la clase creada anteriormente, será pública y estática. Le pondremos dos notaciones, una, al igual que la anterior, de tipo XmlRootElement y le diremos que corresponde al elemento alumno del xml y la otra del tipo XmlType y le diremos que el orden de los elementos que contiene serán primero apellido, posteriormente edad y por último nota.

Esta clase contendrá tres atributos privados, uno de tipo string llamado apellido, otro de tipo int llamado edad y otro de tipo double llamado nota.

Posteriormente crearemos el método constructor y los métodos getter and setter.

En el get del apellido pondremos la notación XmlElement y le diremos que corresponde al elemento apellido del xml.

En el get de la edad también le pondremos la notación XmlElement pero en este caso le diremos que corresponde al elemento edad del xml.

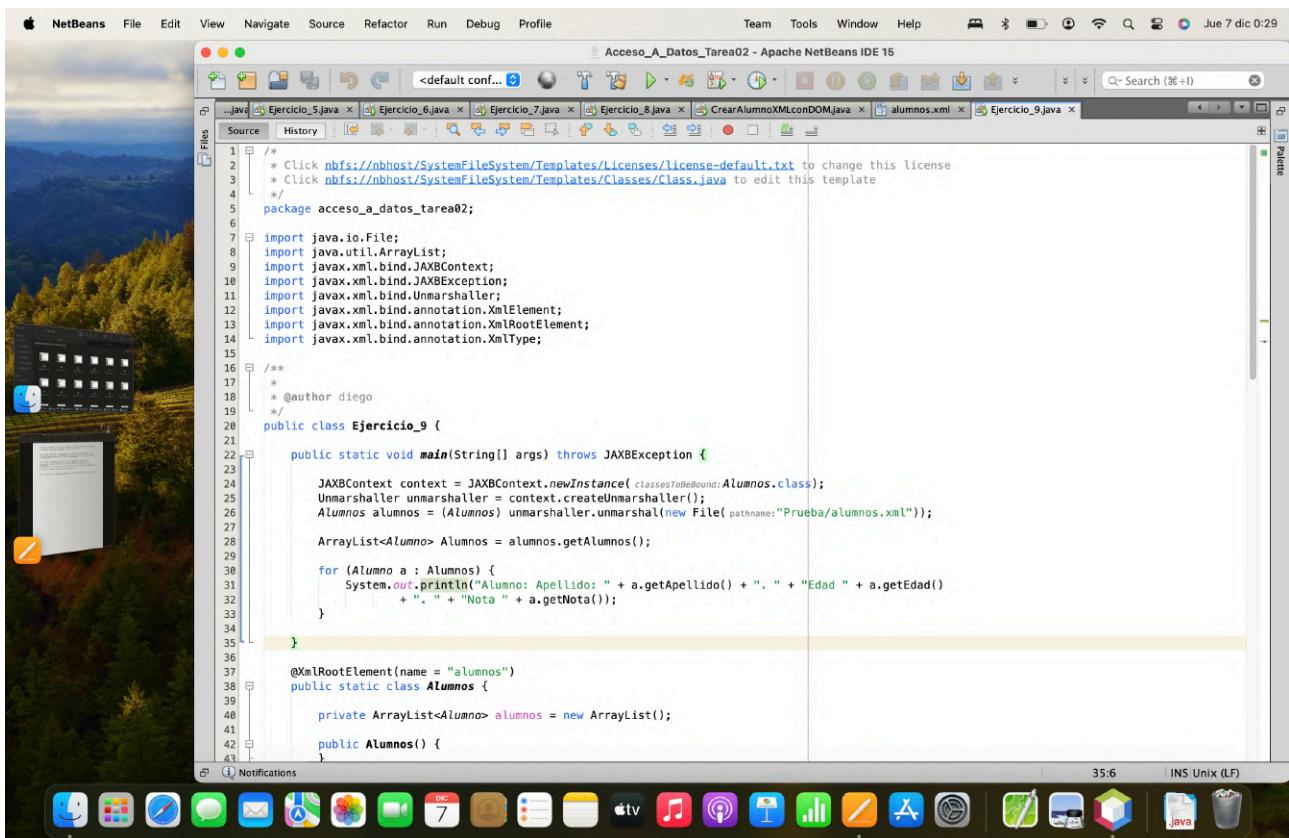
La última notación se la pondremos al get de la nota, será también una notación XmlElement y le diremos que corresponde al elemento nota del xml.

Con esto acabamos las clases alumnos y alumno por lo que volvemos al método main para terminar lo que nos falta.

Nos queda declarar un ArrayList de alumno llamado alumnos y lo instanciaremos mediante el método getAlumnos de la clase Alumnos.

Por último crearemos un bucle for - each para recorrer el ArrayList alumnos completo y mostraremos por pantalla una concatenación de cada alumno en la cual nos indicara el apellido, la edad y la nota de cada uno de ellos.

En las siguientes imágenes podemos ver el código de la aplicación.

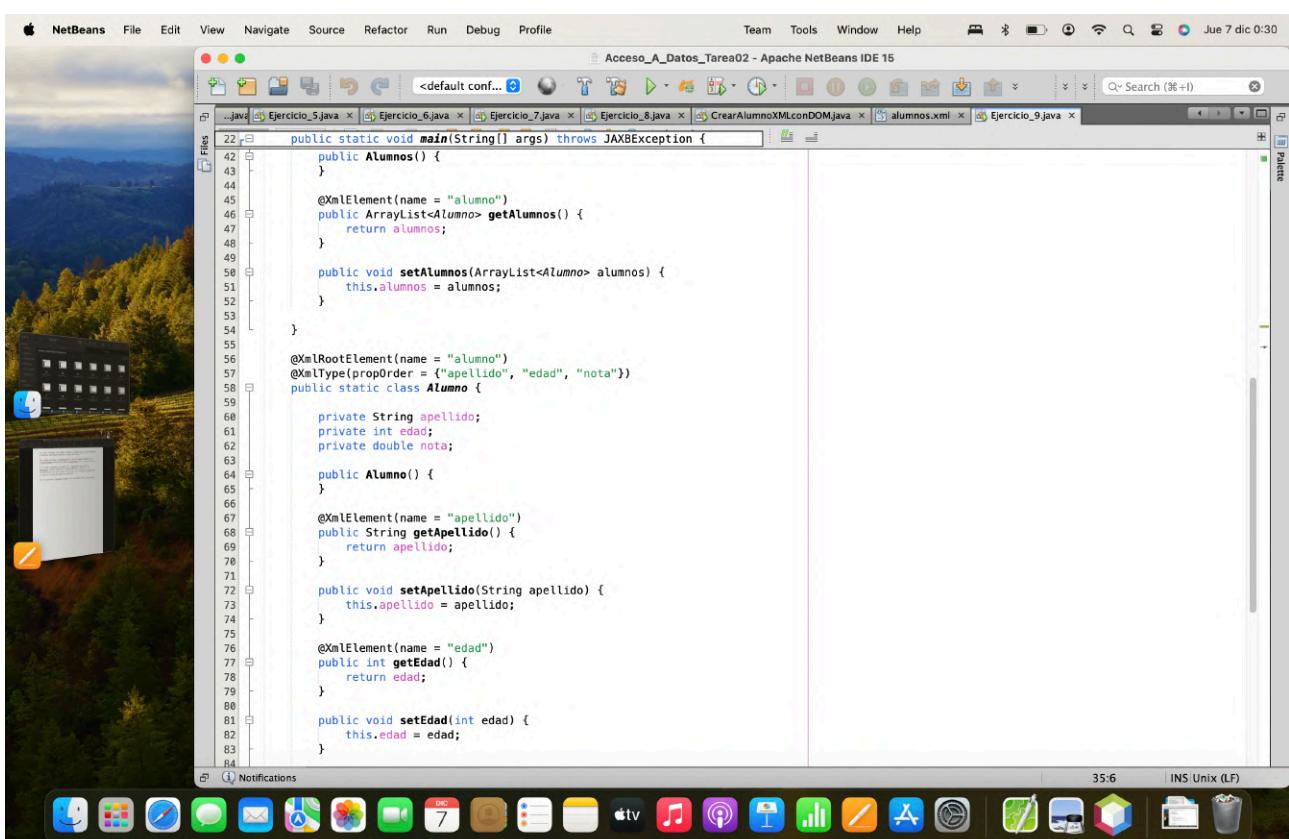


NetBeans IDE 15 - Acceso_A_Datos_Tarea02

```

1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/class.java to edit this template
4  */
5  package acceso_a_datos_tarea02;
6
7  import java.io.File;
8  import java.util.ArrayList;
9  import javax.xml.bind.JAXBContext;
10 import javax.xml.bind.JAXBException;
11 import javax.xml.bind.Unmarshaller;
12 import javax.xml.bind.annotation.XmlElement;
13 import javax.xml.bind.annotation.XmlRootElement;
14 import javax.xml.bind.annotation.XmlType;
15
16 /**
17 *
18 * @author diego
19 */
20 public class Ejercicio_9 {
21
22     public static void main(String[] args) throws JAXBException {
23
24         JAXBContext context = JAXBContext.newInstance(alumnos.class);
25         Unmarshaller unmarshaller = context.createUnmarshaller();
26         Alumnos alumnos = (Alumnos) unmarshaller.unmarshal(new File("Prueba/alumnos.xml"));
27
28         ArrayList<Alumno> Alumnos = alumnos.getAlumnos();
29
30         for (Alumno a : Alumnos) {
31             System.out.println("Alumno: Apellido: " + a.getApellido() + ". " + "Edad " + a.getEdad()
32                     + ". " + "Nota " + a.getNota());
33         }
34     }
35
36     @XmlRootElement(name = "alumnos")
37     public static class Alumnos {
38
39         private ArrayList<Alumno> alumnos = new ArrayList();
40
41         public Alumnos() {
42     }
43 }

```

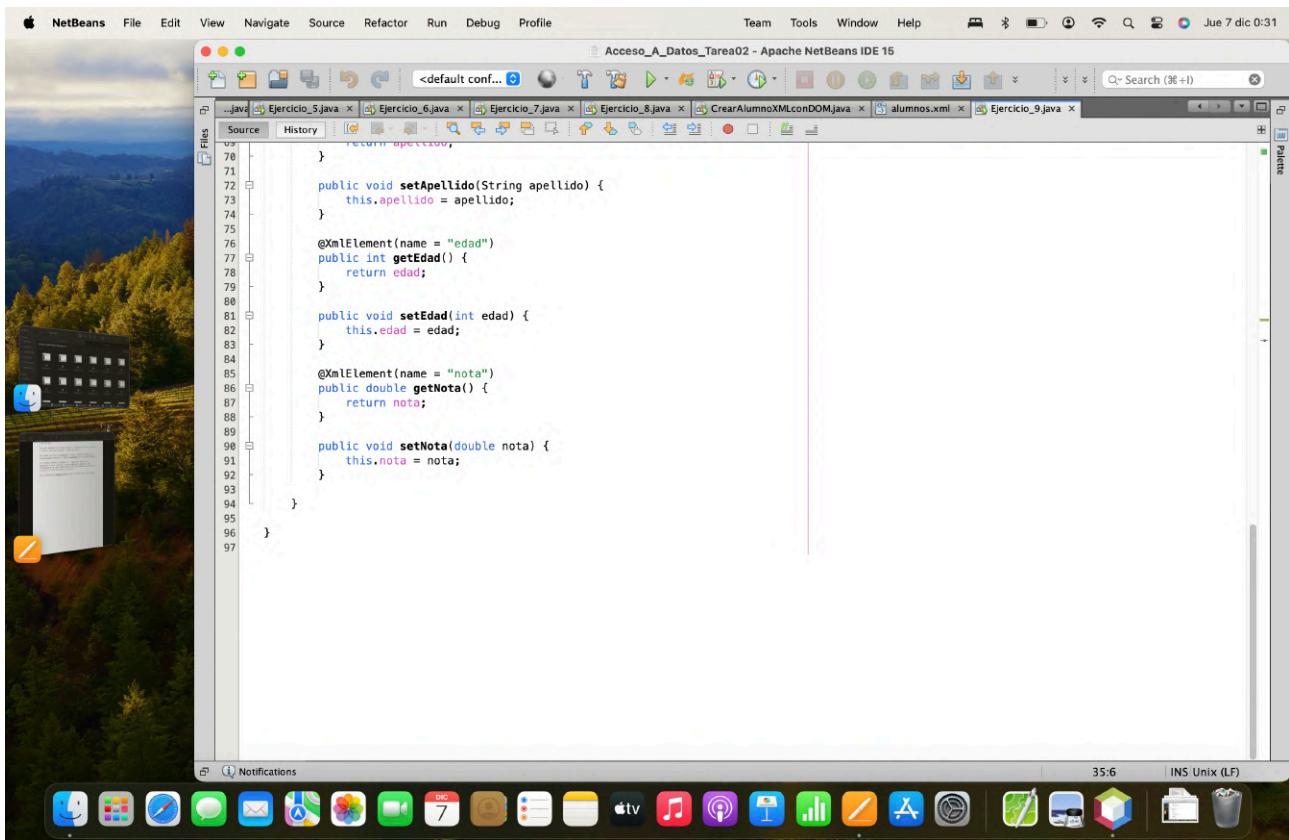


NetBeans IDE 15 - Acceso_A_Datos_Tarea02

```

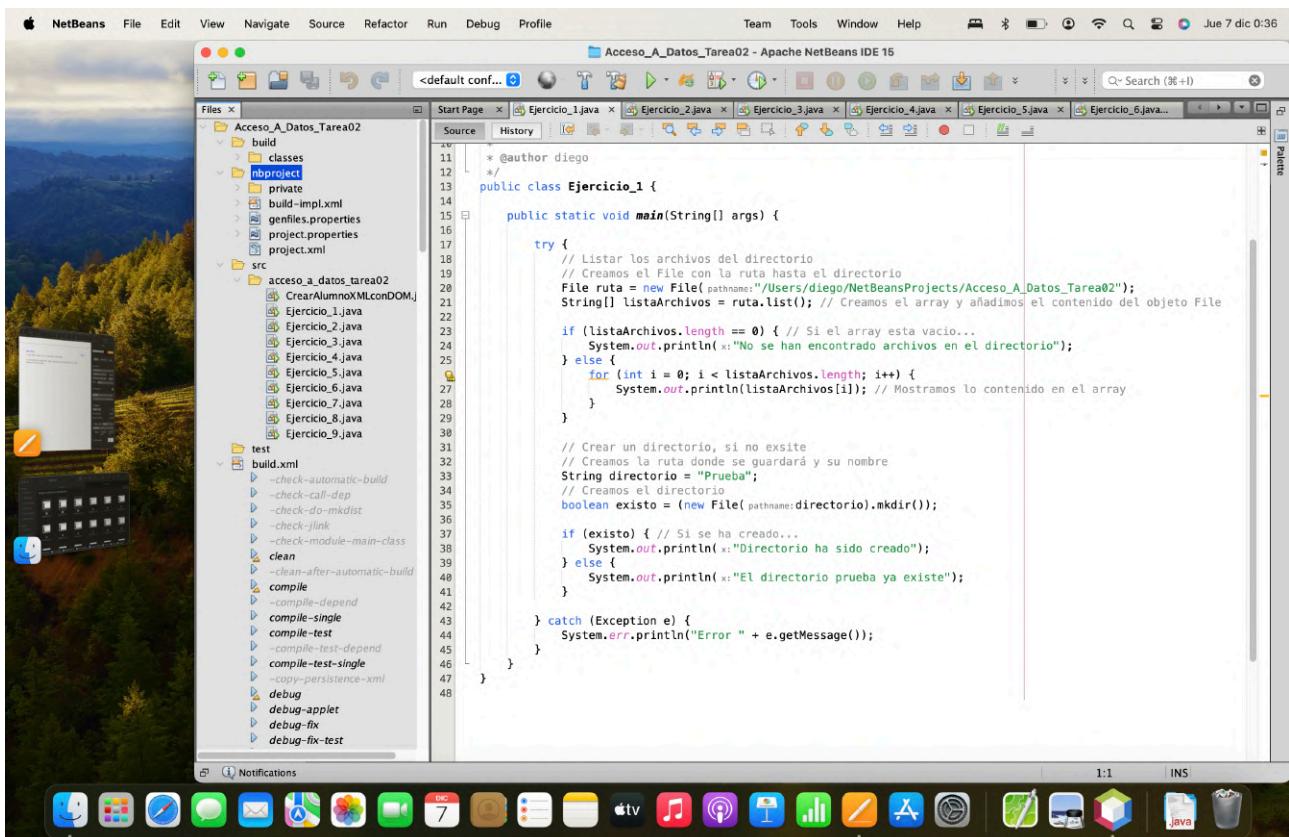
22     public static void main(String[] args) throws JAXBException {
23         public Alumnos() {
24     }
25
26         @XmlElement(name = "alumno")
27         public ArrayList<Alumno> getAlumnos() {
28             return alumnos;
29         }
30
31         public void setAlumnos(ArrayList<Alumno> alumnos) {
32             this.alumnos = alumnos;
33         }
34     }
35
36     @XmlRootElement(name = "alumno")
37     @XmlType(propOrder = {"apellido", "edad", "nota"})
38     public static class Alumno {
39
40         private String apellido;
41         private int edad;
42         private double nota;
43
44         public Alumno() {
45     }
46
47         @XmlElement(name = "apellido")
48         public String getApellido() {
49             return apellido;
50         }
51
52         public void setApellido(String apellido) {
53             this.apellido = apellido;
54         }
55
56         @XmlElement(name = "edad")
57         public int getEdad() {
58             return edad;
59         }
60
61         public void setEdad(int edad) {
62             this.edad = edad;
63         }
64
65         @XmlElement(name = "nota")
66         public double getNota() {
67             return nota;
68         }
69
70         public void setNota(double nota) {
71             this.nota = nota;
72         }
73
74     }
75 }

```



Y con esto damos por finalizada la tareas.

A continuación mostrare unas capturas de pantallas de cada ejercicio ejecutado.



The screenshot shows the NetBeans IDE interface with the title "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The left pane displays the project structure for "Acceso_A_Datos_Tarea02", which includes a "Prueba" folder, a "build" folder, an "nbproject" folder containing "private", "build-impl.xml", "genfiles.properties", "project.properties", and "project.xml", and a "src" folder containing "acceso_a_datos_tarea02" with files like "CrearAlumnoXMLconDOM.java", "Ejercicio_1.java", "Ejercicio_2.java", "Ejercicio_3.java", "Ejercicio_4.java", "Ejercicio_5.java", "Ejercicio_6.java", "Ejercicio_7.java", "Ejercicio_8.java", and "Ejercicio_9.java". It also includes "test", "build.xml", and "manifest.mf". The right pane shows the code for "Ejercicio_1.java". The code creates a directory named "Prueba" if it doesn't exist, lists files in the current directory, and prints them to the console. The bottom pane shows the "Output" window with the following text:

```
run:
manifest.mf
.DS_Store
test
build.xml
nbproject
build
src
Directorio ha sido creado
BUILD SUCCESSFUL (total time: 0 seconds)
```

The screenshot shows the Apache NetBeans IDE interface. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The left sidebar displays the project structure:

- Prueba
- build
- direcnio
- fichero.txt
- nbproject
- src
 - acceso_a_datos_tarea02
 - CrearAlumnoXMLConDOM.java
 - Ejercicio_1.java
 - Ejercicio_2.java**
 - Ejercicio_3.java
 - Ejercicio_4.java
 - Ejercicio_5.java
 - Ejercicio_6.java
 - Ejercicio_7.java
 - Ejercicio_8.java
 - Ejercicio_9.java
 - test
 - build.xml
 - manifest.mf

The main editor window shows the Java code for **Ejercicio_2.java**:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package acceso_a_datos_tarea02;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

/**
 *
 * @author diego
 */
public class Ejercicio_2 {

    public static void main(String[] args) {
        // Creamos los objetos Path con las rutas y el nombre del directorio y del fichero
        Path directorio = Paths.get(first:"direcnio");
        Path fichero = Paths.get(first:"direcnio/fichero.txt");
        try{
            if(Files.exists(path:directorio)){ // Si el directorio existe
                System.out.println("El directorio direcnio ya existe");
                if(Files.exists(path:fichero)){ // Si el fichero existe
                    System.out.println("El archivo fichero.txt ya existe.");
                }
                Files.delete(path:fichero); // Borramos el fichero
                System.out.println("El archivo fichero.txt ha sido borrado");
                Files.delete(path:directorio); // Borramos el directorio
            }
        }
    }
}
```

The output window at the bottom shows the run results:

```
run:
El directorio direcnio no existe
El directorio direcnio ha sido creado
El archivo fichero.txt ha sido creado
BUILD SUCCESSFUL (total time: 0 seconds)
```

The screenshot shows the Apache NetBeans IDE interface. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The left sidebar displays the project structure for "Acceso_A_Datos_Tarea02" with files like "Ejercicio_1.java" through "Ejercicio_9.java". The main editor window shows the code for "Ejercicio_2.java". The code creates a directory "direccio" and a file "fichero.txt", then deletes them. The output window shows the results of the run:

```

run:
El directorio direccio ya existe
El archivo fichero.txt ya existe.
El archivo fichero.txt ha sido borrado
El directorio direccio ha sido borrado
BUILD SUCCESSFUL (total time: 0 seconds)

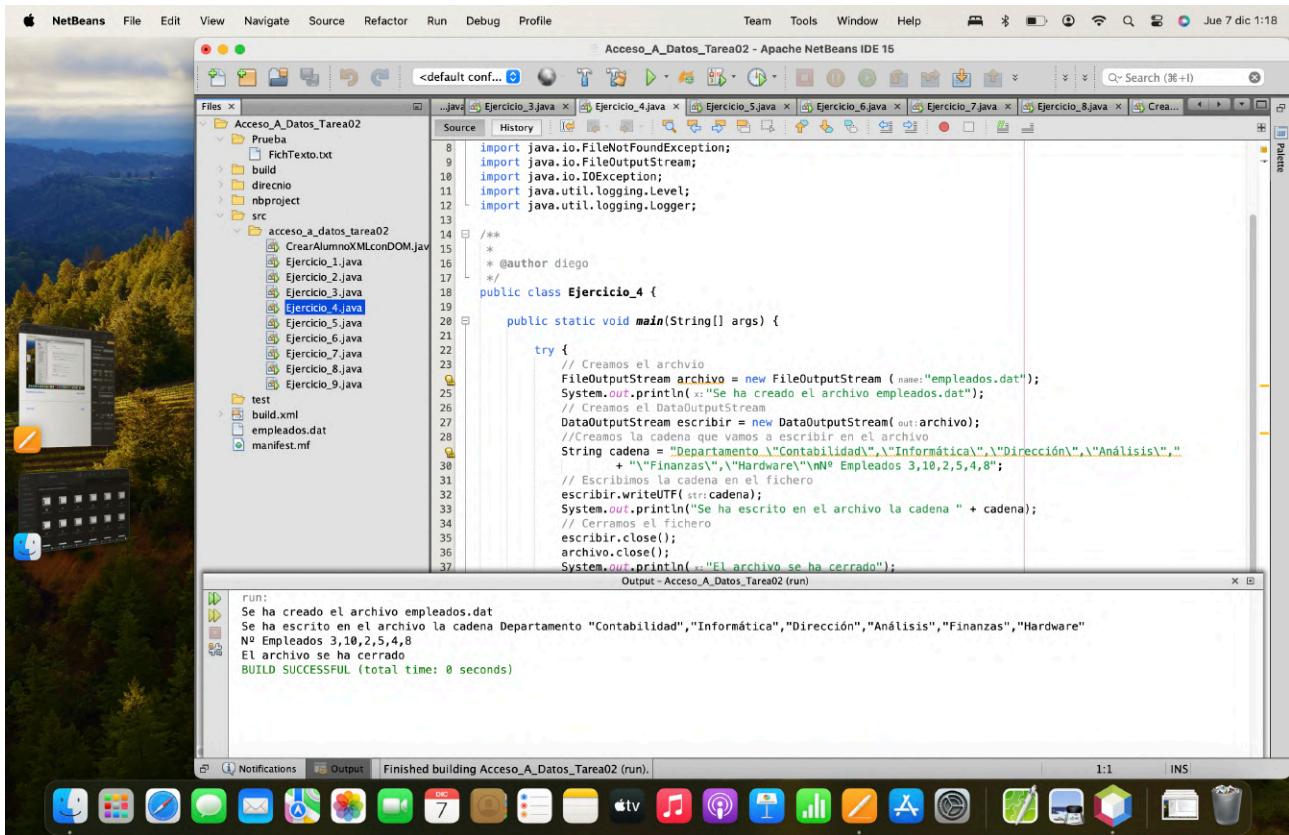
```

The screenshot shows the Apache NetBeans IDE interface. The title bar reads "Acceso_A_Datos_Tarea02 - Apache NetBeans IDE 15". The left sidebar displays the project structure for "Acceso_A_Datos_Tarea02" with files like "Ejercicio_3.java" through "Ejercicio_9.java". The main editor window shows the code for "Ejercicio_3.java". The code creates a file "FichTexto.txt", writes to it, and then reads it back. The output window shows the results of the run:

```

run:
El fichero ha sido creado
Escribiendo en el archivo
El archivo ha sido cerrado
Leyendo el fichero
Ejemplo de escritura en un fichero de texto o txt
BUILD SUCCESSFUL (total time: 0 seconds)

```



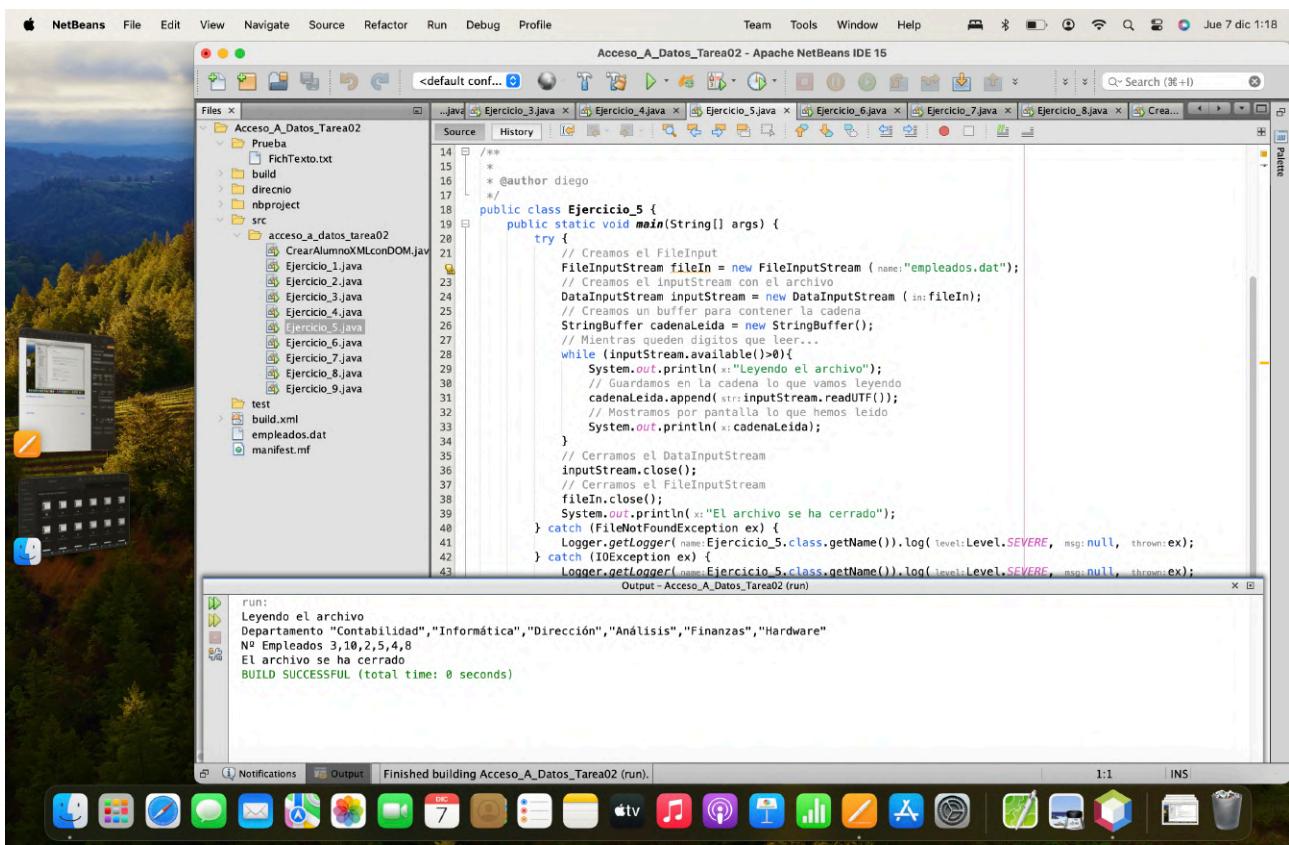
```

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Ejercicio_4 {
    public static void main(String[] args) {
        try {
            // Creamos el archivo
            FileOutputStream archivo = new FileOutputStream ("name:empleados.dat");
            System.out.println("Se ha creado el archivo empleados.dat");
            // Creamos el DataOutputStream
            DataOutputStream escribir = new DataOutputStream(archivo);
            //Creamos la cadena que vamos a escribir en el archivo
            String cadena = "Departamento \"Contabilidad\", \"Informática\", \"Dirección\", \"Análisis\", \""
                + "\"Finanzas\", \"Hardware\"\\n\"Nº Empleados 3,10,2,5,4,8";
            // Escribimos la cadena en el fichero
            escribir.writeUTF(cadena);
            System.out.println("Se ha escrito en el archivo la cadena " + cadena);
            // Cerramos el fichero
            escribir.close();
            archivo.close();
            System.out.println("El archivo se ha cerrado");
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Ejercicio_4.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Ejercicio_4.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

run:
 Se ha creado el archivo empleados.dat
 Se ha escrito en el archivo la cadena Departamento "Contabilidad", "Informática", "Dirección", "Análisis", "Finanzas", "Hardware"
 Nº Empleados 3,10,2,5,4,8
 El archivo se ha cerrado
 BUILD SUCCESSFUL (total time: 0 seconds)

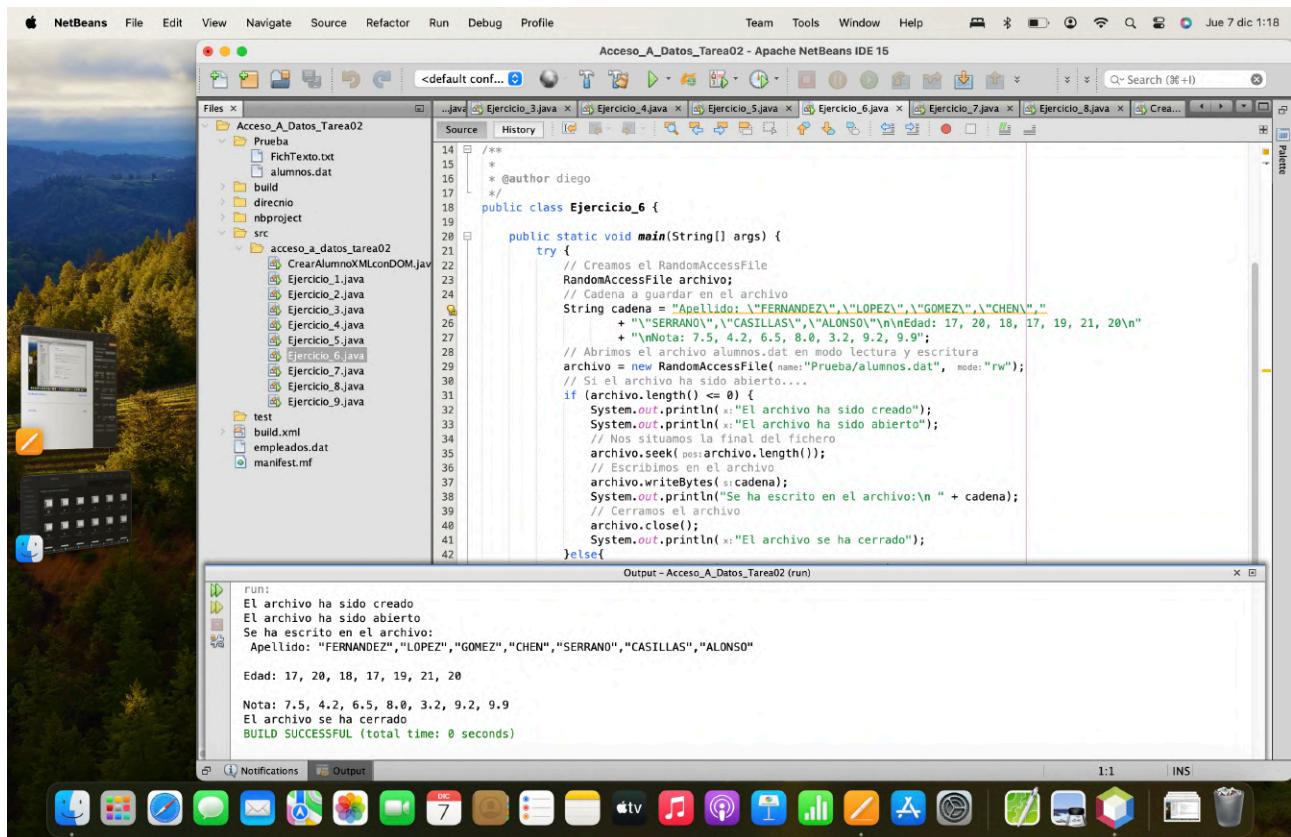


```

public class Ejercicio_5 {
    public static void main(String[] args) {
        try {
            // Creamos el FileInputStream
            FileInputStream fileIn = new FileInputStream ("name:empleados.dat");
            // Creamos el inputSteam con el archivo
            DataInputStream inputStream = new DataInputStream(fileIn);
            // Creamos un buffer para contener la cadena
            StringBuffer cadenaLeida = new StringBuffer();
            // Mientras queden dígitos que leer...
            while (inputStream.available() > 0) {
                System.out.println("Leyendo el archivo");
                // Guardamos en la cadena lo que vamos leyendo
                cadenaLeida.append(inputStream.readUTF());
                // Mostramos por pantalla lo que hemos leido
                System.out.println(cadenaLeida);
            }
            // Cerramos el DataInputStream
            inputStream.close();
            // Cerramos el FileInputStream
            fileIn.close();
            System.out.println("El archivo se ha cerrado");
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Ejercicio_5.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Ejercicio_5.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

run:
 Leyendo el archivo
 Departamento "Contabilidad", "Informática", "Dirección", "Análisis", "Finanzas", "Hardware"
 Nº Empleados 3,10,2,5,4,8
 El archivo se ha cerrado
 BUILD SUCCESSFUL (total time: 0 seconds)



```

14  /*
15  *
16  * @author diego
17  */
18 public class Ejercicio_6 {
19
20     public static void main(String[] args) {
21         try {
22             // Creamos el RandomAccessFile
23             RandomAccessFile archivo;
24             // Cadena a guardar en el archivo
25             String cadena = "Apellido: \"FERNANDEZ\", \"LOPEZ\", \"GOMEZ\", \"CHEN\", "
26                     + "\n\"SERRANO\", \"CASILLAS\", \"ALONSO\"\n\nEdad: 17, 20, 18, 17, 19, 21, 20\n";
27             // Abrimos el archivo alumnos.dat en modo lectura y escritura
28             archivo = new RandomAccessFile(name:"Prueba/alumnos.dat", mode:"rw");
29             // Si el archivo ha sido abierto...
30             if (archivo.length() <= 0) {
31                 System.out.println("El archivo ha sido creado");
32                 System.out.println("El archivo ha sido abierto");
33                 // Nos situamos la final del fichero
34                 archivo.seek( pos:archivo.length());
35                 // Escribimos en el archivo
36                 archivo.writeBytes( s:cadena);
37                 System.out.println("Se ha escrito en el archivo:\n " + cadena);
38                 // Cerramos el archivo
39                 archivo.close();
40                 System.out.println("El archivo se ha cerrado");
41             }else{
42
        }
    }
}

```

Output - Acceso_A_Datos_Tarea02 (run)

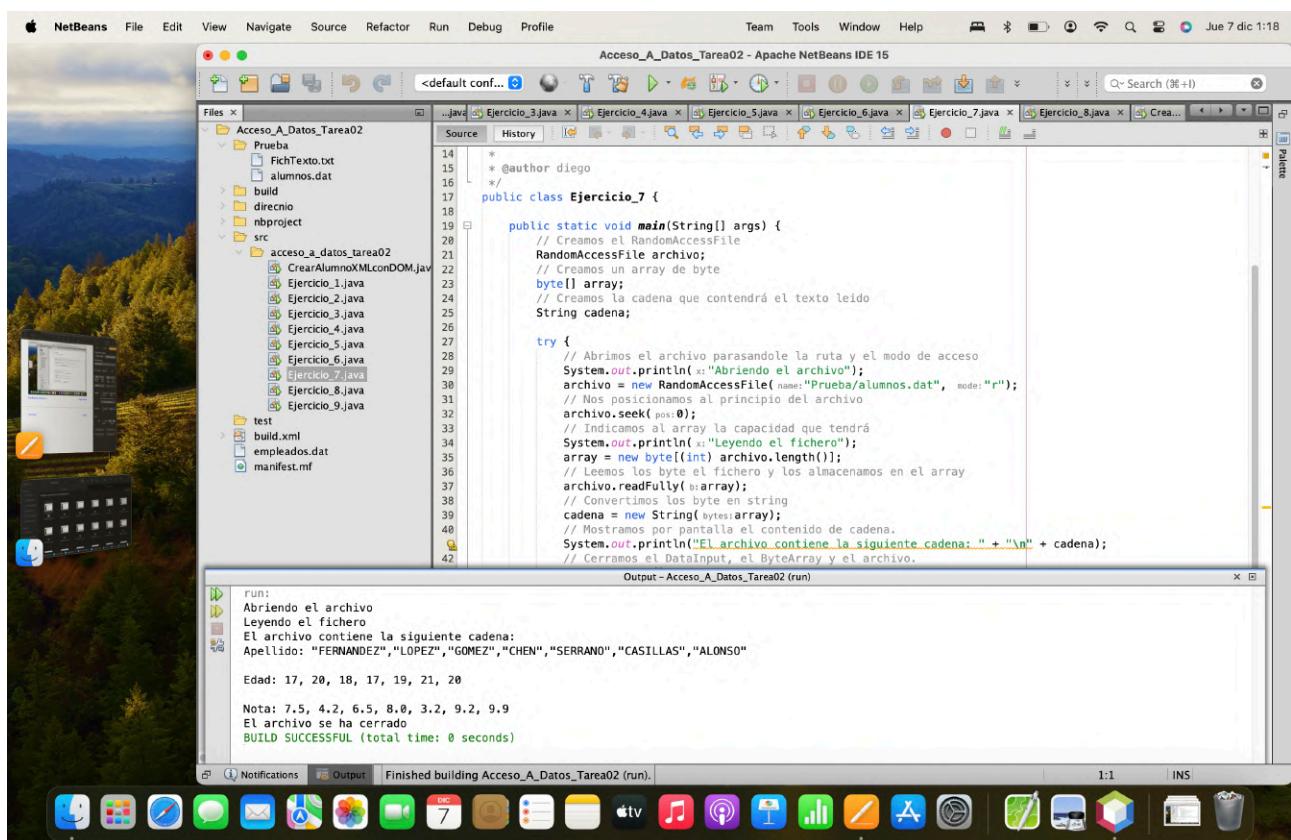
```

run:
El archivo ha sido creado
El archivo ha sido abierto
Se ha escrito en el archivo:
Apellido: "FERNANDEZ", "LOPEZ", "GOMEZ", "CHEN", "SERRANO", "CASILLAS", "ALONSO"

Edad: 17, 20, 18, 17, 19, 21, 20

Nota: 7.5, 4.2, 6.5, 8.0, 3.2, 9.2, 9.9
El archivo se ha cerrado
BUILD SUCCESSFUL (total time: 0 seconds)

```



```

14  /*
15  * @author diego
16  */
17 public class Ejercicio_7 {
18
19     public static void main(String[] args) {
20         // Creamos el RandomAccessfile
21         RandomAccessFile archivo;
22         // Creamos un array de byte
23         byte[] array;
24         // Creamos la cadena que contendrá el texto leido
25         String cadena;
26
27         try {
28             // Abrimos el archivo pasandole la ruta y el modo de acceso
29             System.out.println("Abriendo el archivo");
30             archivo = new RandomAccessFile(name:"Prueba/alumnos.dat", mode:"r");
31             // Nos posicionamos al principio del archivo
32             archivo.seek( pos:0);
33             // Indicamos al array la capacidad que tendrá
34             System.out.println("Leyendo el fichero");
35             array = new byte[(int) archivo.length()];
36             // Leemos los byte el fichero y los almacenamos en el array
37             archivo.readFully( array);
38             // Convertimos los byte en string
39             cadena = new String( bytes:array);
40             // Mostramos por pantalla el contenido de cadena.
41             System.out.println("El archivo contiene la siguiente cadena: " + "\n" + cadena);
42             // Cerramos el DataInput, el ByteArray y el archivo.
        }
    }
}

```

Output - Acceso_A_Datos_Tarea02 (run)

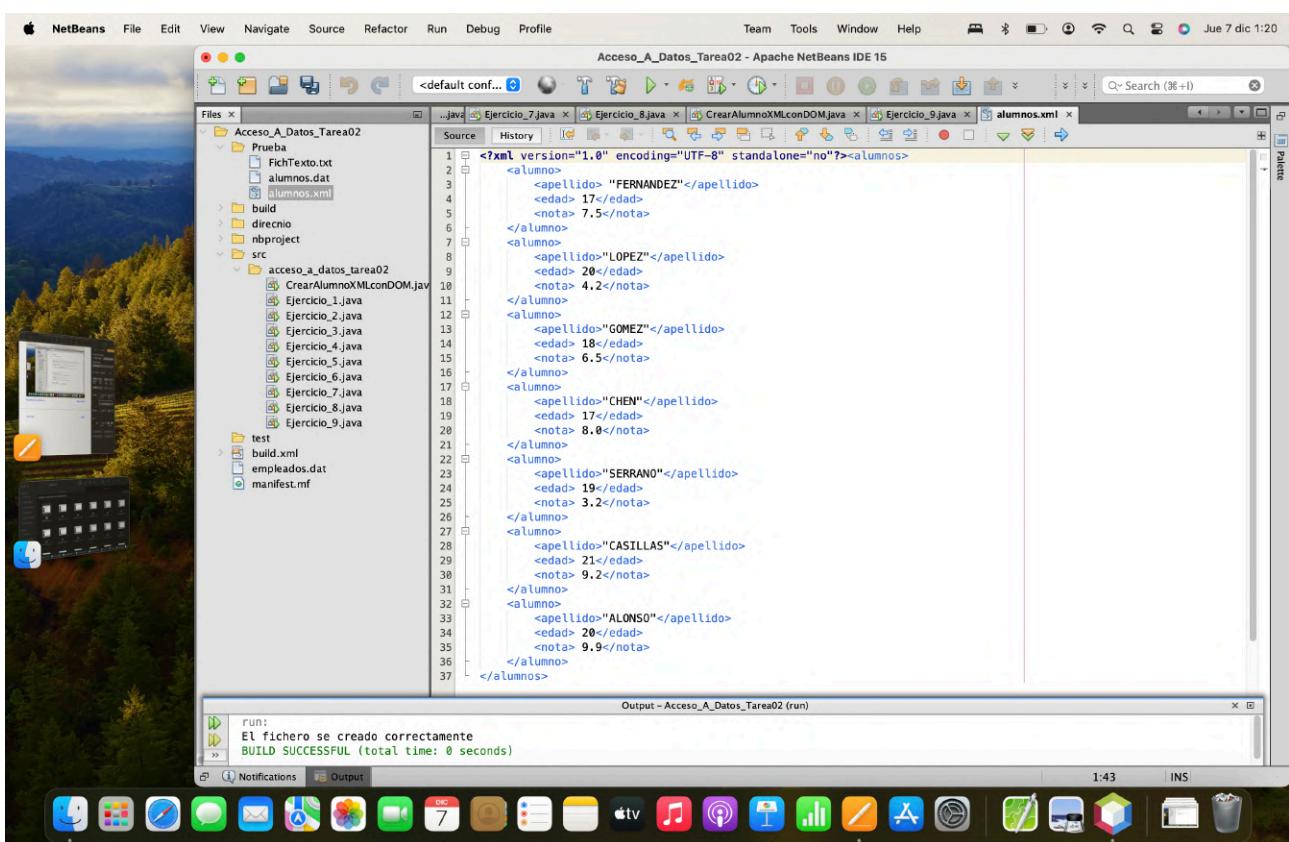
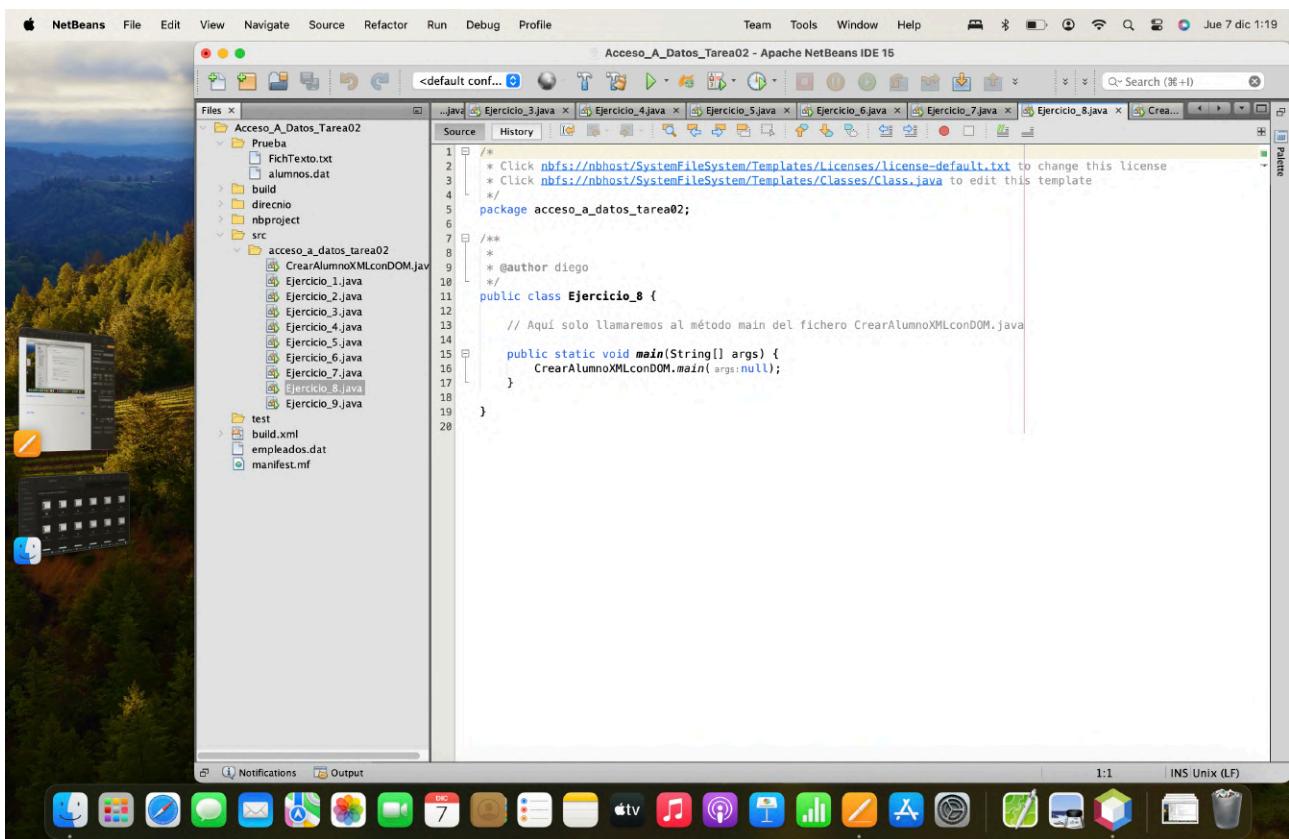
```

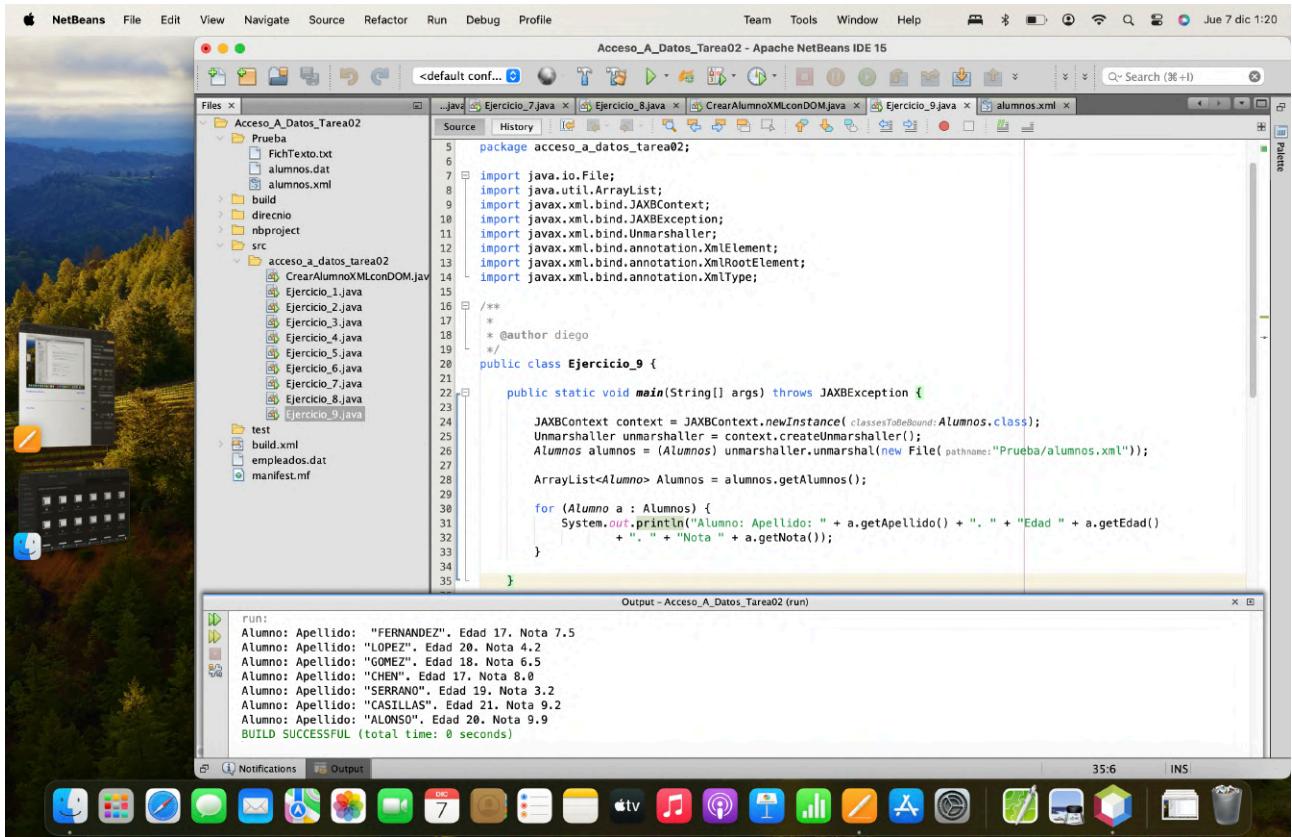
run:
Abriendo el archivo
Leyendo el fichero
El archivo contiene la siguiente cadena:
Apellido: "FERNANDEZ", "LOPEZ", "GOMEZ", "CHEN", "SERRANO", "CASILLAS", "ALONSO"

Edad: 17, 20, 18, 17, 19, 21, 20

Nota: 7.5, 4.2, 6.5, 8.0, 3.2, 9.2, 9.9
El archivo se ha cerrado
BUILD SUCCESSFUL (total time: 0 seconds)

```





Criterios de puntuación. Total 10 puntos.

La tarea consta de 9 actividades a desarrollar, del 1 al 7 valen 1 punto. El apartado 8 y 9 valen 1,5 puntos cada uno.

Debes crear un proyecto con cada fichero o clase correspondiente a cada apartado de la tarea. Debes elaborar un único documento (en formato pdf) donde expliques el desarrollo de la tarea, apartado por apartado. Debes enviar una carpeta comprimida con el proyecto y el fichero pdf. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_ADxx_Tareaxx_Entregaxx

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la primera unidad del MP de AD, debería nombrar esta tarea como...

sanchez_manas_begona_AD02_Tarea02_Entrega01