```java
package modelo;

import com.coti.tools.*;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import static java.nio.file.Files.readAllLines;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Comparator;

public class Profesteca implements Serializable{
    ArrayList<Profesor> listaProfesores = new ArrayList<>();
    private final String ficheroDatos = "PROFESORES";
    private final String ficheroBinario = "profesores.bin";
    private final String ficheroTexto = "profesores.txt";

    /**
    * Intenta importar los archivos .bin, en caso de que no lo consiga
    * intetará importar los .txt.
    * @throws java.io.IOException
    */
    public void importarDatos() throws IOException{
        if(importarBinario()){
        }else
            importarTexto();
    }

    /**
    * 1º Ruta al fichero profesores.txt que se encuentra en la carpeta
PROFESORES
    * 2º Si el archivo de la ruta existe:
    *      - Trata de crear un ArrayList que contenga todo el archivo .txt a
través
    *        de la función readAllLines de Rutas.java
    *      - For desde el pricipio hasta el final del array separandolo por
#,
    *        y metiendo los datos en el factory de profesor
    *        Si profesor no está vacio se agrega a la lista de profesores,
    *        si la carga de profesor produce un error, no se agrega
    * Return: Si no ha habido problemas con la carga o el fichero retorna
true.
     * @return
     * @throws java.io.IOException
    */
    public boolean importarTexto() throws IOException{

        Path ruta =
Rutas.pathToFileInFolderOnDesktop(ficheroDatos,ficheroTexto);
        if(ruta.toFile().exists()){
            try{
                ArrayList<String> lineas = (ArrayList<String>)
readAllLines(ruta);
```

```
                for(String i : lineas){
                    String[] campos = i.split("#");
                    Profesor profesor = Profesor.FactoryProfesor(campos);
                    if(profesor =≠ null){
                        listaProfesores.add(profesor);
                    }else{
                        System.out.printf("Error en la carga en la linea %s",
i); //Esto puede estar bien o no, ya que es control de erorres.
                    }
                }
            }catch( IOException exeption){
                System.out.printf("Error en fichero %s",
ruta.toFile().getAbsolutePath());
                return false;
            }
        }else{
            System.out.printf("Error en el fichero %s,",
ruta.toFile().getAbsolutePath());
            return false;
        }
        System.out.printf("Importados un total de %d profesores.\n",
listaProfesores.size());
        return true;
    }

    /**
    * 1º Ruta al fichero profesores.bin que se encuentra en la carperta
PROFESORES
    * 2º Tratar de leer el archivo a través del metodo fis-bis-ois
    * Return: en caso de que se lea y se guarde correctamente devuelve true,
en
    * caso contrario devuelve false.
     * @return
    */
    public boolean importarBinario(){

        Path ruta =
Rutas.pathToFileInFolderOnDesktop(ficheroDatos,ficheroBinario);
        try {
            FileInputStream fis = new FileInputStream(ruta.toFile());
            BufferedInputStream bis = new BufferedInputStream(fis);
            ObjectInputStream ois = new ObjectInputStream(bis);
            listaProfesores = (ArrayList<Profesor>) ois.readObject();
            ois.close();
        } catch (IOException | ClassNotFoundException ex) {
            return false;
        }

        return true;
    }

    /**
    * Añade a la lista de profesores el nuevo profesor dado de alta.
     * @param nombre
     * @param edad
     * @param asignaturas
    */
    public void altaProfesor(String nombre, int edad, ArrayList<String>
asignaturas){

```

```
106         listaProfesores.add(new Profesor(nombre,edad,asignaturas));
107     }
108 
109     /**
110     * Busca el nombre del profesor en la lista.
111      * @param nombre
112      * @return
113     */
114     public boolean existeProfesor(String nombre){
115 
116         for(Profesor profesor : listaProfesores ){
117             if(nombre.equalsIgnoreCase(profesor.nombre))
118                 return true;
119         }
120 
121         return false;
122     }
123 
124     /**
125      * Eliminar un profesor de la lista
126      * 1° Comprobar si existe el profesor
127      *    - Si existe guardar indice
128      *    - Si no existe return false
129      * 2° Eliminar a través de remove el profesor marcado por el indice
guardado.
130      * @param nombre
131      * @return
132      */
133     public boolean darBajaProfesor(String nombre){
134         int indiceABorrar=-1;
135         if(existeProfesor(nombre))
136         {
137             for(int i = 0; i < listaProfesores.size(); i ++){
138                 if(nombre.equalsIgnoreCase(listaProfesores.get(i).nombre))
139                     indiceABorrar = i;
140             }
141         }else{
142             return false;
143         }
144 
145         listaProfesores.remove(indiceABorrar);
146         return true;
147     }
148 
149     /**
150      * A traves de los parámetros recoges el nombre del profesor y la opcion
de modificar
151      * el nombre y la edad
152      * Si quiere modificar el nombre realiza un for del arraylist buscando el
nombre
153      * del profesor y setear el nombre con el nuevo atributo
154      * Si quiere modificar la edad realiza un for del arraylist buscando el
nombre
155      * del profesor y setear la edad con el nuevo atributo parseado.
156      * @param nombre
157      * @param opcion
158      * @param nuevoAtributo
159      */
160     public void modificarAtributoProfesor(String nombre, String opcion,
String nuevoAtributo){
```

```
161         if(opcion.equals("n")){
162             for(Profesor profesor : listaProfesores){
163                 if(profesor.getNombre().equalsIgnoreCase(nombre)){
164                     profesor.setNombre(nuevoAtributo);
165                 }
166             }
167         }else{
168             for(Profesor profesor : listaProfesores){
169                 if(profesor.getNombre().equalsIgnoreCase(nombre)){
170                     profesor.setEdad(Integer.parseInt(nuevoAtributo));
171                 }
172             }
173         }
174     }
175 
176     /**
177      * Consultar asignaturas de un profesor
178      * Buscar al profesor en la arraylist
179      * Con un string vacio realizar:
180      *      for de las asignaturas del profesor marcado
181      *      metes en el string las asignaturas que hay en el array
    asignaturas del profesor
182      * @param nombre
183      * @return
184      */
185     public String consultarAsignaturasProfesor(String nombre){
186 
187         String consulta="";
188         for(Profesor profesor : listaProfesores){
189             if(profesor.getNombre().equalsIgnoreCase(nombre)){
190                 for(String asignaturas : profesor.asignaturas){
191                     consulta = consulta + asignaturas + "\n";
192                 }
193             }
194         }
195 
196         return consulta;
197     }
198 
199     /**
200      * Ordenar Profesores por nombre.
201      */
202     public void ordenarProfesores(){
203         listaProfesores.sort(Comparator.comparing(Profesor::getNombre));
204     }
205 
206     /**
207      * Crea un String[][] del tamaño de la lista cargandola con cada
    profesor(String).
208      * @return
209      */
210     public String[][] arrayListToMatrix() {
211         //ordenarProfesores();
212         String [][] matrizProfesores = new String[listaProfesores.size()][];
213         for (int i = 0; i < listaProfesores.size(); i++) {
214             matrizProfesores[i] =
    listaProfesores.get(i).toString().split("#");
215         }
216         return matrizProfesores;
217     }
```

```
218     public String[][] listadoProfesores(){
219        String[][] matrizProfesores = arrayListToMatrix();
220        return matrizProfesores;
221     }
222 
223     /**
224     * 1º Path a la ruta del binario y crear archivo en esa ruta
225     * 2º Escribir en el archivo a través de fos-bos-oos.
226     */
227     public void exportarBinario() {
228 
229         Path p =
Rutas.pathToFileInFolderOnDesktop(ficheroDatos,ficheroBinario);
230         File f = p.toFile();
231 
232         try {
233             FileOutputStream fos = new FileOutputStream(f);
234             BufferedOutputStream bos = new BufferedOutputStream(fos);
235             ObjectOutputStream oos = new ObjectOutputStream(bos);
236             oos.writeObject(listaProfesores);
237             oos.close();
238         } catch (IOException e) {
239         }
240     }
241 }
242 
```