Sistema de gestión de cursos

Entrega 1. IPC Curso 2017/2018

Se pretende desarrollar una aplicación de escritorio para la gestión de cursos y alumnos de una academia. La aplicación debe permitir hacer un mantenimiento sobre la información de los alumnos: **alta y baja**[1]. De cada alumno se manejarán los siguientes datos:

- DNI
- Nombre
- Edad
- Dirección
- Fecha de alta
- Fotografía

El alumno no se podrá dar de baja si existen matrículas del mismo, deben borrarse primero las matrículas de ese alumno. Esta restricción la deben controlar mediante código en su programa

Por otra parte se desea gestionar también la información de los cursos que imparte la academia. Al igual que antes se desea poder **dar de alta un curso y poder borrarlo**. Para los cursos interesa la siguiente información:

- Título del curso
- Profesor asignado
- Número máximo de alumnos
- Fecha de inicio
- Fecha de fin
- Hora
- Días de la semana que se imparte
- Aula

El curso no se podrá dar de baja si existen matrículas del mismo, deben borrarse primero las matrículas de ese curso. Esta restricción la deben controlar mediante código en su programa

Por último se desea gestionar la matrícula de alumnos a los cursos que se imparten. La funcionalidad requerida es:

- Matricular alumno a curso: El programa permitirá seleccionar un alumno y un curso. Se comprobará si el alumno no está matriculado en otro curso que se imparta en las mismas horas y días, también que el curso no se exceda el número máximo de alumnos.
- Borrar alumno matriculado de curso: A partir de un curso seleccionado se mostrarán todos los alumnos que se encuentran matriculados, se podrá seleccionar uno de ellos y borrarlo de la matrícula.
- Listado de cursos: Se obtendrá en pantalla un listado de todos los cursos disponibles. Se podrá seleccionar uno de ellos y consultar la lista de alumnos matriculados.

Para la matrícula se almacenará la fecha y las referencias al curso y al alumno.

[1] No se contemplan las modificaciones de información

Persitencia de los datos

La información de la academia se proporciona en un archivo XML (**`academia.xml`**) que debe ubicarse en el home del usuario, en sistemas Windows en:

`C:\Usuarios\NombreUsuario`. Donde NombreUsuario es el usuario con el que hizo el inicio de sesión

Para acceder al mismo se utiliza un jar (accesoaBD.jar) que simula el acceso a una base de datos, en este caso la base de datos es el propio archivo XML. Inicialmente el archivo XML contiene dos alumnos, un curso y dos matrículas.

No deben manipular directamente el archivo xml, el acceso es mediante la API contenida en el archivo jar que debe importar en su proyecto, explicado más adelante.

La clase `Academia` es la clase raíz para la persistencia en XML. La clase no es directamente manipulable en el programa, se accede a sus atributos como hemos mencionado anteriormente mediante una API. Con respecto a la información almacenada por ésta:

- alumnos contiene la colección de alumnos, estén o no matriculados en un curso.
- cursos contiene la colección de cursos, tengan o no alumnos matriculados
- matriculas contiene todas las matrículas, para alumno, curso.

```
@XmlRootElement(name = "academia")
public class Academia {

    private ArrayList<Alumno> alumnos = new ArrayList<>();
    private ArrayList<Curso>  cursos = new ArrayList<>();
    private ArrayList<Matricula> matriculas = new ArrayList<>();

    …

}
```

El resto de las clases del modelo que **se importarán automáticamente en el proyecto** al añadir el jar:

```
@XmlType(propOrder={"dni","nombre", "direccion", "edad",
"fechadealta","foto"})
public class Alumno {
    private String dni;
    private String nombre;
    private int edad;
    private String direccion;
    private LocalDate fechadealta;
    private Image foto;

…
}
```

Observe que desde Alumno no se pueden obtener los cursos en los que está matriculado, para ellos se utiliza la clase matrícula.

```
public class Matricula {
    private LocalDate fecha;
```

```
    private Curso curso;
    private Alumno alumno;
…
}
```

La clase Curso contiene la siguiente información

```
public class Curso {
    private String titulodelcurso;
    private String profesorAsignado;
    private int numeroMaximodeAlumnos;
    private LocalDate fechainicio;
    private LocalDate fechafin;
    private LocalTime hora;
    private ArrayList<Dias> diasimparte;
    private String aula;
…
}
```

Por último `Dias` es un tipo enumerado

```
public enum Dias {
    Lunes, Martes, Miercoles,Jueves,Viernes,Sabado,Domingo

}
```

Al igual que antes dese curso no pueden obtenerse los alumnos que están matriculados, para ello debe usarse la clase matrícula.

Con respecto a la API que se ofrece, ésta tiene los siguientes métodos

Tabla 1 Métodos públicos de la clase AccesoaBD

`public accesoaBD()`	Instancia la clase, comprueba si el archivo xml se encuentra en la máquina, si no existe crea uno vacío
`public List<Matricula> getMatriculas()`	Devuelve todas las matrículas almacenadas en el archivo XML
`public List<Matricula> getMatriculasDeCurso(String nombreCurso)`	Devuelve las matrículas de un curso de acuerdo al nombre
`public List<Matricula> getMatriculasDeCurso(Curso c)`	Devuelve las matrículas del curso pasado como parámetro
`public List<Alumno> getAlumnosDeCurso(String nombreCurso)`	Obtiene los alumnos de un curso de título `nombreCurso`
`public List<Alumno> getAlumnosDeCurso(Curso c)`	Obtiene los alumnos del curso c
`public List<Curso> getCursos()`	Retorna todos los cursos de la academia
`public List<Alumno> getAlumnos()`	Devuelve todos los alumnos de la academia
`public Alumno getAlumnoByDNI(String dni)`	Devuelve el alumno identificado por su dni
`public Curso getCursoByNombre(String nombre)`	Devuelve un curso con ese nombre o título
`public void salvar()`	Salva en XML los cursos, alumnos y matrículas

Inclusión del jar en el proyecto

Cree un nuevo proyecto NetBeans de nombre por ejemplo TestLibreria. Utilice el explorador de archivos para crear un directorio dentro de su proyecto y dentro de `Source Packages` (en algunos sistemas se muestra `src`), llámelo por ejemplo `lib`. Los proyectos de NetBeans se almacenan por defecto en la carpeta `Documentos\NetBeansProjects` dentro del home del usuario, aunque obviamente la ubicación puede ser cambiada, al igual que en la figura inferior. Copie el archivo jar dentro de lib.

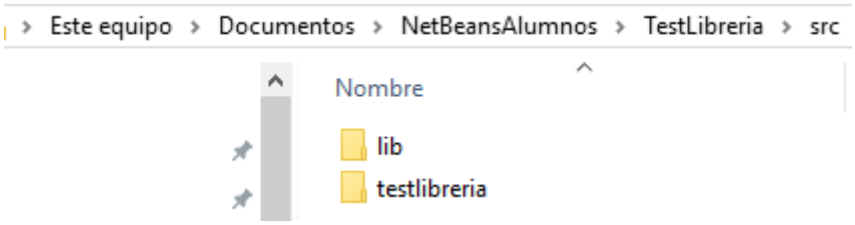

Figura 1 Carpeta lib dentro del fuente del nuevo proyecto

Seleccione el directorio Libraries que se muestra en el explorador del proyecto y pulse el botón derecho del ratón -> Add JAR/Folder. Busque la ubicación dentro de su proyecto del jar: accesoaBD.jar.

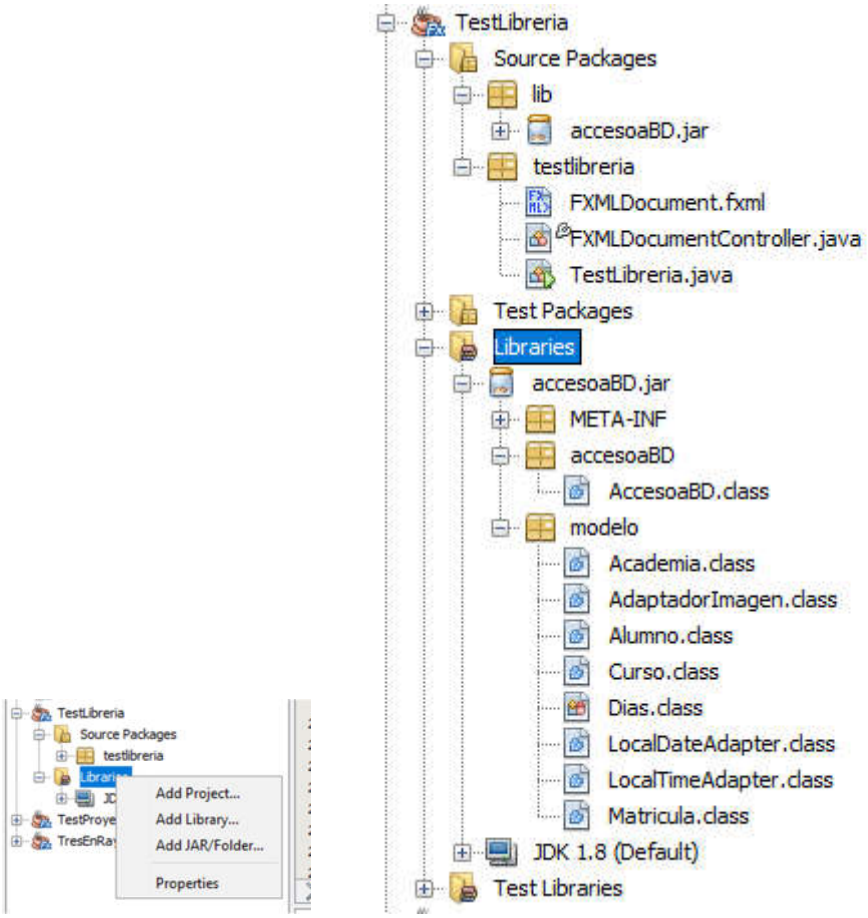

Figura 2 Incorporación de la librería al proyecto

En la figura se muestra la librería una vez incorporada al proyecto. Observe que también aparece dentro del subdirectorio lib. Ahora el proyecto lo puede cambiar de máquina llevándose consigo el archivo jar.

Haciendo doble-click sobre cualquiera de las clases (`Matricula.class`) se pueden ver sus atributos y el perfil de sus métodos, tal como se muestra en la figura siguiente.

```
package modelo;

import java.time.LocalDate;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

public class Matricula {

    private LocalDate fecha;
    private Curso curso;
    private Alumno alumno;

    public Matricula() {
        Compiled Code
    }

    public Matricula(LocalDate fecha, Curso curso, Alumno alumno) {
        Compiled Code
    }

    @XmlJavaTypeAdapter(value = LocalDateAdapter.class)
    public LocalDate getFecha() {
        Compiled Code
    }

    public void setFecha(LocalDate fecha) {
        Compiled Code
    }

    public Curso getCurso() {
        Compiled Code
    }
```

Figura 3 Vista parcial de la clase Matrícula

Para acceder a los métodos de la librería desde su programa debe utilizar algo parecido a lo siguiente:

```
AccesoaBD acceso = new AccesoaBD(); // instanciar la clase de acceso
    // acceder a la colección de alumnos
ArrayList<Alumno> listaalumnos = (ArrayList<Alumno>) acceso.getAlumnos();
       …/…
Alumno alumno= ….; // crea un nuevo alumno
    // incluir el alumno en la lista
listalumnos.add(alumno);
   // salvar en la bd/xml
acceso.salvar();
```

Figura 4 Ejemplo de utilización sin interfaz gráfica

La clase `AdaptadorImagen.class` se utiliza para salvar en el `xml` la imagen de un alumno, no necesita utilizarla directamente. Lo mismo pasa con `LocalTimeAdapter` y `LocalDateAdapter` se emplean para almacenar respectivamente los tipos `LocalTime` y `LocalDate`.

Cargar imágenes en la clase Alumno

Dado que el programa exige crear nuevos alumnos y uno de sus campos es un campo `Image`, a continuación mostramos diversas formas de leer una imagen del disco duro y mostrarla en un `ImageView`

```
// Opción 1 de carga: la imagen está dentro del directorio src en un subdirectorio resources
   Image image = new Image("/resources/foto1.png");
   imagenView.setImage(image);
// Opción 2 de carga: misma situación que antes
   Image imagen1 = new Image(Entrega1IPC_2018.class.
                       getResourceAsStream("/resources/foto2.jpg"));
        //nombre de la clase principal
   imagenView.setImage(imagen1);
// Opción 3 de carga: está en un directorio del disco duro
   String item = "c:/imagenes/foto1.png";
   File imageFile = new File(item);
   String fileLocation = imageFile.toURI().toString();
   Image imagen2 = new Image(fileLocation);
   imagenView.setImage(imagen2);
```

Utilización con componentes de la interfaz gráfica

Dado que el contenido del archivo xml deberá visualizarlo en las interfaces gráficas de su aplicación, debe utilizar listas observables para envolver el contenido del archivo xml, por ejemplo los alumnos que la API los devuelve como una List<Alumno>. La colección observable debe ser una `ObservableList`, de modo que los cambios que se hagan en ésta sean trasmitidos a la Lista. **No utilice en ningún caso un ObservableArrayList ya que los cambios no se transmiten a la lista componente.**

```
ObservableList<Alumno> alumnosObservable;

AccesoaBD acceso = new AccesoaBD(); // instanciar la clase de acceso
    // acceder a la colección de alumnos para visualizarlos por ejemplo

alumnosObservable = FXCollections.observableList(acceso.getAlumnos());
// si está vinculado con un TableView
tablaAlumnos.setItems(alumnosObservable);
        …/…
Alumno alumno= ….; // crea un nuevo alumno
    // incluir el alumno en la lista observable
alumnosObservable.add(alumno);
   // salvar en la bd/xml, en ese momento o posteriormente antes de salir
acceso.salvar();
```

Figura 5 Acceso a la API para mostrar en la interfaz de usuario

Una vez que haya obtenido del archivo xml las tres listas: alumnos, cursos y matrículas debe manipularlas en su programa teniendo en cuenta las dos restricciones que se mencionaron al principio del documento (no se puede borrar un alumno si existen matrículas del mismo, y no se puede borrar un curso si existen matrículas del curso)

Las modificaciones (añadir, borrar) las debe hacer desde las respectivas `ObservableList`, antes de salir del programa debe llamar al método `salvar()` de la API, observe que no necesita parámetros.

Instrucciones de entrega

- Exporta el proyecto NetBeans en un ZIP (https://media.upv.es/player/?id=a0bfd620-021e-11e6-851a-656f7e06a374) y súbelo a la tarea de poliformaT correspondiente. Cada grupo de alumnos hará una sola entrega, incluyendo en el campo de comentarios los nombres de los dos alumnos.
- La fecha de entrega para todos los grupos es el día **22 de Abril de 2018**

Evaluación

- Aquellos proyectos que no compilen o que no muestren la pantalla principal al arrancar se calificarán con un cero.
- Se deberán incluir los diálogos de confirmación, errores, etc. que se considere necesario.
- Para evaluar el diseño de la interfaz de la aplicación se tendrán en cuenta las directrices estudiadas en clase de teoría.
- Debe ser posible redimensionar la pantalla principal, cuyos controles se ajustarán adecuadamente para usar el espacio disponible (usa los contenedores vistos en clase).