Solución de la tarea para AD07.

La solución a la tarea pasa por tres fases que el alumnado debe completar, a saber:

Primera fase.

En este primer apartado hay que crear el componente que accede a la base de datos para cargar los contenidos de la tabla matrículas. Al componente se le añaden las propiedades correspondientes a los campos de la tabla, a saber:

- **DNI**: String que representa el DNI del alumno.
- **NombreModulo**: propiedad de tipo String que representa el módulo del que se matricula.
- **Curso**: String que representa el curso académico.
- **Nota**: valor numérico de tipo double que representa la nota obtenida por el alumno en el módulo.

Así como todos los atributos y clases privados necesarios para implementar el componente que serán:

La clase matrícula:

```
public class Matricula
    {
        String DNI;
        String NombreModulo;
        String Curso;
        String Nota;
    public Matricula()
        {}
        public Matricula(String nDNI, String nNombreModulo, String nCurso, double nNota)
        {
           this.DNI = nDNI;
           this.NombreModulo = nNombreModulo;
           this.Curso = nCurso;
           this.Nota = nNota;
     }
 }
```

Un vector de matrículas:

```
private Vector Matriculas=new Vector();
```

y un valor booleano que representará el modo (todos los alumnos o solo uno);

```
boolean modo;
```

Una vez hecho esto se pasaría a la segunda fase.

Segunda fase.

En esta fase se añade el código que implementa la funcionalidad del componente.

Se añaden los métodos

- **recargarFilas()**: carga el contenido total de la tabla matrículas en el vector de matrículas.
- **recargarDNI()**: recarga la estructura interna del componente con las matrículas de un DNI en particular.
- **seleccionarFila(i)**: recupera en las propiedades del componente el registro número i del vector.
- **addMatricula()**: añade un registro nuevo a la base de datos con la información almacenada en las propiedades del componente.

También se crearán las clases para gestionar el evento que lanza el componente. Será necesario implementar una clase que herede de **EventObjet** y otra que herede de **EventListener**. Además en el componente se debe añadir el código necesario para:

- Registrar al oyente que será el formulario principal del proyecto de prueba.
- Añadir el código a **ActionListener** necesario para lanzar el evento cuando coincida la hora actual con la hora de la alarma activa.

Para que sea posible conocer en el oyente el modo se añade como propiedad al evento y se pasa en el constructor.

```
private ModoModificadoListener receptor;
    public class ModoModificadoEvent extends java.util.EventObject
    {
        // constructor
        public boolean modo;
        public ModoModificadoEvent(Object source, boolean nModo)
        {
            super(source);
        DNI;
            modo = nModo;
        }
    }
    //Define la interfaz para el nuevo tipo de evento
    public interface ModoModificadoListener extends EventListener
    {
        public void capturarModoModificado(ModoModificadoEvent ev);
    }
    public void addModoModificadoListener(ModoModificadoListener receptor)
    {
        this.receptor = receptor;
    }
    public void removeModoModificadoListener(ModoModificadoListener receptor)
    {
        this.receptor=null;
    }
  }
```

Tercera fase.

Ahora es el momento de compilar y construir el proyecto para comprobar que las clases se han creado sin errores. El resultado lo podemos encontrar en un paquete .jar en el directorio build del proyecto.

Cuarta fase.

Crear un proyecto de prueba. Se añade el paquete con el componente a las bibliotecas del proyecto y podremos usarlo como una clase más. Generamos una clase que implemente la interfaz para que pueda convertirse en oyente y probamos los diferentes métodos.

```
public class AccedeBD implements ModoModificadoListener {
    MatriculaBean matricula;
    public AccedeBD()
    {
         matricula = new MatriculaBean();
         matricula.addModoModificadoListener((ModoModificadoListener) this);
    }
    public void listado()
    {
        for(int i=0; i<4; i++)
        {
            matricula.seleccionarFila(i);
            System.out.println("Matrícula " + i + "\n\tDNI:" + matricula.getDNI());
            System.out.println("\tNombre: " + matricula.getNombreModulo());
            System.out.println("\tApellidos: " + matricula.getCurso());
            System.out.println("\tDireccion: " + matricula.getNota());
        }
    }
    public void listadoDNI(String nDNI) throws ClassNotFoundException
    {
        matricula.recargarDNI(nDNI);
        for(int i=0; i<4; i++)
        {
            matricula.seleccionarFila(i);
            System.out.println("Matrícula " + i + "\n\tDNI:" + matricula.getDNI());
            System.out.println("\tNombre: " + matricula.getNombreModulo());
            System.out.println("\tApellidos: " + matricula.getCurso());
            System.out.println("\tDireccion: " + matricula.getNota());
        }
    }
    void anade()
    {
        matricula.setDNI("98765432A");
        matricula.setNombreModulo("Acceso a datos");
        matricula.setCurso("11-12");
        matricula.setNota(7.25);
        try {
            matricula.addMatricula();
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(AccedeBD.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    public void capturarModoModificado(ModoModificadoEvent ev)
    {
        if(ev.modo)
            System.out.println("Se ha cargado toda la tabla");
        else
```

```
                System.out.println("Se ha cargado la matricula de un alumno");
        }
    }
```

Para resolver la tarea propuesta se aporta como solución el proyecto para crear el componente y el proyecto con la aplicación de prueba junto con el código de creación de la base de datos en el siguiente enlace:

Tarea. (3.23 MB).

Si lo prefieres también lo puedes descargar en formato zip aquí:

Solución a la tarea. (3.23 MB)