

**INSTRUCCIONES:**

- El EXAMEN PRÁCTICO consiste en un único ejercicio de 6.5 puntos a realizar completamente por todos, independientemente de que se hubiera aprobado el examen de FEBRERO.
- El EXAMEN PRÁCTICO se divide en tres apartados:
  - a) Diseño e implementación de las clases: 1.25 puntos.
  - b) Diseño de la interfaz gráfica: 0.75 puntos.
  - c) Implementación de la funcionalidad de las ventanas: 4.5 puntos.
- Para aquellos que no aprobaron el EXAMEN DE FEBRERO, el apartado a) de 1,25 puntos se ponderará sobre 6.5 puntos para obtener la nota correspondiente al examen práctico del PRIMER PARCIAL.
- Se puede utilizar cualquier tipo de material bibliográfico, como apuntes o libros, tanto en soporte papel como en soporte digital (pendrives, acceso a Internet, etc.).

**EVALUACIÓN PRIMER PARCIAL**

La puntuación total del examen TEÓRICO + PRÁCTICO será de 10 puntos. Parte teórica: 3.5 puntos. Parte Práctica: 6.5 puntos.

**EVALUACIÓN SEGUNDO PARCIAL**

La puntuación total del examen TEÓRICO + PRÁCTICO será de 10 puntos. Parte teórica: 3.5 puntos. Parte Práctica: 6.5 puntos.

**ENUNCIADO DEL EXAMEN PRÁCTICO**

La academia Masnota te ha encargado que diseñes e implementes una aplicación llamada **AcademiaMas** que se encargara de la gestión de la academia. La aplicación deberá permitir dar de alta alumnos y módulos y mostrar la información relativa a dichos alumnos.

**Diseño e implementación de clases: (1,25 puntos)**

**public class Módulo (0.25 puntos):**

- Atributos privados: **nombre (String)**, **horas (int)** y **unidades (int)**.
- **Constructor** por parámetros y métodos **get** and **set** para cada atributo.

**public class Alumno (1 punto):**

- Atributos privados: **nombre (String)**, **localidad (String)** y **modulos (ArrayList<Modulo>)**.
- **Constructor** por parámetros que recibe: nombre y localidad. Inicializa el atributo modulos como un ArrayList<Modulo> vacío.
- **Constructor** por parámetros que recibe: nombre, localidad y lista de modulos.
- Métodos **get** and **set** para los atributos nombre y localidad.
- Método **public void añadirModulo(Modulo a)** que permite añadir un módulo al ArrayList de módulos.
- Método **public String[] getNombreModulos()** que devuelve un array



de String con los nombres de los módulos a los que está apuntado el alumno.

- Método **public double getNumeroHoras()** que devuelve el número total de horas semanales en las que se encuentra matriculado.

### **Diseño de la interfaz gráfica: (0.75 puntos)**

La interfaz de la aplicación **AcademiaMas** constará de **4 ventanas** cuyo aspecto y componentes se especifican a continuación:

#### **Ventana MenuPrincipal (0.15 puntos):**

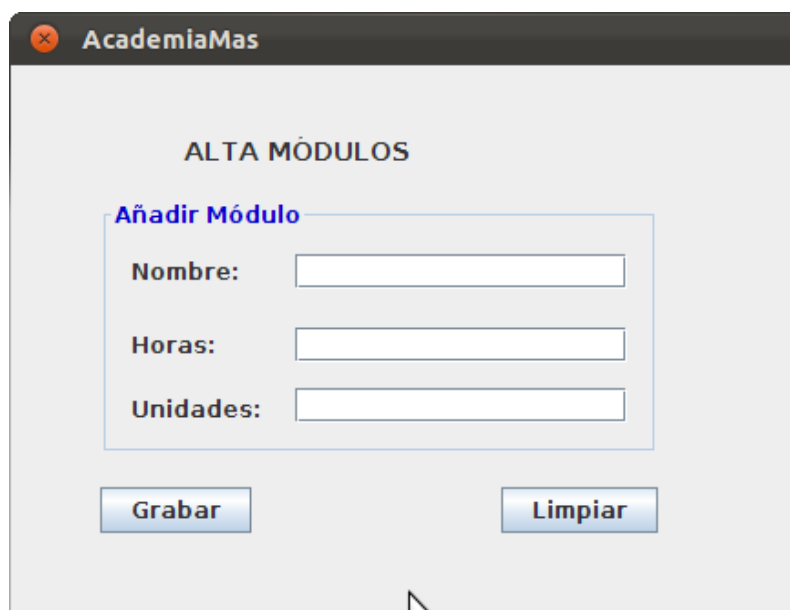
Será de tipo **JFrame** y contendrá dos menús:

- **Alumno** con las opciones Añadir alumno, Ver alumno, Guardar alumno y Salvar alumno.
- **Módulo** con la opción Añadir módulo, Guardar módulo y Salvar módulo.
- Además, la ventana poseerá tres botones con las mismas funcionalidades que las respectivas opciones de los menús. Cada ítem del menú deberá incluir un **acelerador** y un **mnemónico**.



#### **Ventana JDAñadirModulo (0.20 puntos):**

- Será de tipo **JDialog** y tendrá el siguiente aspecto:



### Ventana JDAñadirAlumno (0.20 puntos):

- Será de tipo **JDialog** y contendrá entre otros elementos, un **JComboBox** donde aparecerán todos los módulos que oferta la academia, y un **JList** donde irán apareciendo los módulos que vayamos seleccionando para el alumno que estemos dando de alta. Su aspecto será el siguiente:

The screenshot shows a Java Swing dialog box titled 'AcademiaMas'. Inside, there is a section titled 'ALTA ALUMNOS' with a sub-header 'Añadir Alumno'. Below this, there are three input fields: 'Nombre:', 'Localidad:', and 'Módulos:'. The 'Módulos:' field is a JComboBox. To the right of the 'Módulos:' field is a 'Quitar' button. Below the input fields is a large empty rectangular area, likely a JList. At the bottom of the dialog are two buttons: 'Grabar' and 'Limpiar'.

### Ventana JDVerAlumno (0.20 puntos):

- Será de tipo **JDialog** y contendrá entre otros elementos un **JList** donde se mostrarán los nombres de los módulos a los que esté apuntado el alumno que se ha buscado. Su aspecto será el siguiente:

The screenshot shows a Java Swing dialog box titled 'AcademiaMas'. Inside, there is a search section with the label 'Nombre a buscar:' followed by an input field and a 'Mostrar' button. Below this is a section titled 'Datos del alumno' with three input fields: 'Nombre:', 'Localidad:', and 'Horas:'. At the bottom of the dialog is a large empty rectangular area labeled 'Módulos:', which is intended to be a JList.



### **Implementación de la funcionalidad de la aplicación: (4.5 puntos)**

A continuación se describen las características de implementación de cada una de las ventanas de la aplicación:

#### **public class MenuPrincipal (1.5 puntos):**

- Atributos privados: **listaAlumnos** (ArrayList<Alumno>) y **listaModulos** (ArrayList<Modulo>) que almacenará respectivamente los alumnos y los módulos dados de alta en el sistema.
- Tendrá los métodos necesarios para poder acceder a los 2 atributos privados anteriores (añadir elemento a la lista, obtener elemento de la lista y obtener dimensión de la lista).
- Sus botones (así como la opción correspondiente en el menú) harán visible la ventana (JDialog) respectiva a la operación que representan.
- Las opciones de menú Cargar/Salvar deben usar serialización para guardar y recuperar la lista de alumnos y módulos.

#### **public class JDAñadirAlumno (1 punto):**

- Para gestionar respectivamente el JComboBox y el JList de la ventana, tendrá los atributos **DefaultComboBoxModel** y **DefaultListModel**. Además de un ArrayList<Modulo> para almacenar los módulos a los que está apuntado el alumno que se está dando de alta.
- Para poder acceder a los ArrayList de la clase **MenuPrincipal** será necesario crear un atributo de tipo MenuPrincipal que se llamará padre. La referencia la obtendremos a través del método **getParent()**. Este método devuelve una instancia de la clase de la que depende el **JDialog** como referencia a **Object** por lo que debemos hacer el casting correspondiente. Una vez obtenida esta referencia, se puede hacer uso de los atributos y métodos públicos de la clase MenuPrincipal. Ejemplo: *MenuPrincipal padre = (MenuPrincipal) this.getParent();*
- Al abrirse la ventana, el **JComboBox** contendrá todos los módulos que estén dados de alta en el sistema en ese momento. Cada vez que seleccionemos uno de esos módulos se añadirá al **JList**.
- El **botón Grabar** creará un objeto **Alumno** con los datos escritos en las cajas de texto así como las actividades incluidas en el **JList** y lo añadirá al atributo **listaClientes** de la clase MenuPrincipal. Luego llamará al método **dispose()**. Antes de crear el alumno, se deberá comprobar que no hay ningún dato vacío y que el número de horas matriculadas no supera las 30, si no es así se mostrará un mensaje de aviso y no se creará el alumno. No es obligatorio que el alumno se apunte a algún módulo.
- El **botón Limpiar** vaciará todas las cajas de texto incluyendo el **JList**.
- El **botón Quitar** eliminará del **JList** el módulo seleccionado. Si no hay ninguno se deberá mostrar un mensaje de aviso.



**public class JDAñadirMódulo (1 punto):**

- Para poder acceder a los ArrayList de la clase MenuPrincipal será necesario crear un atributo de tipo MenuPrincipal que se llamará padre y se creará llamando al método getParent().
- El **botón Grabar** creará un objeto **Modulo** con los datos escritos en las cajas de texto y lo añadirá al atributo **listaModulos** de la clase MenuPrincipal, luego llamará al método **dispose()**. Se deberá comprobar que los datos están cumplimentados y que el número de hora y unidades es un entero. En caso contrario no se creará el módulo y se dará un mensaje de aviso.
- El **botón Limpiar** vaciará todas las cajas de texto.

**public class JDVerAlumno (1 punto):**

- Para poder acceder a los ArrayList de la clase MenuPrincipal será necesario crear un atributo de tipo MenuPrincipal que se llamará padre y se creará llamando al método **getParent()**.
- Incluirá un atributo **DefaultListModel** para gestionar el contenido del **JList** de la ventana.
- El **botón Mostrar** buscará en listaAlumnos de la clase MenuPrincipal algún alumno con el **nombre** que se ha introducido. Si lo encuentra deberá mostrar sus datos en las cajas de texto correspondientes (dichas cajas no serán editables inicialmente) así como los nombres de los módulos a los que está apuntado el alumno en el JList. Además, deberá mostrar el número de horas totales en las que está matriculado. Si no se encuentra al alumno, se mostrará un mensaje apropiado.

