

UD1: Confección GUI con Java

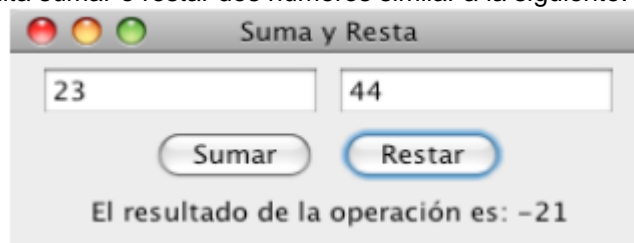
LISTA EJERCICIOS

IMPORTANTE:

- Modificar las plantillas de Netbeans (Templates) para que salga vuestro nombre y apellidos en cada clase de Java. Se requiere modificar TODO tipo de clases que vayáis a utilizar.
- Se creará un proyecto para cada ejercicio, nombrando cada ejercicio como *EjerNº*, por ejemplo, en el caso del primer ejercicio sería: *Ejer1*
- Finalmente se entregará una carpeta comprimida con todos los proyectos llamada: **UD1_Ejercicios_ NombreApellidosTusIniciales.zip**. Por ejemplo, para mí sería:
UD1_Ejercicios_AdriánPR.zip
- Cada proyecto debe ser importado a la carpeta, por tanto, cada proyecto estará así mismo en formato .zip.
- Cada ejercicio podrá contener los paquetes y clases que sean necesarias.

Ejercicio 1: Operaciones sencillas

Crea un programa que permita sumar o restar dos números similar a la siguiente:



- Configurar las acciones de los botones al ser pulsados, y el resultado debe actualizarse en un label (etiqueta) concreto.

Ejercicio 2: Convertidor € - Pesetas

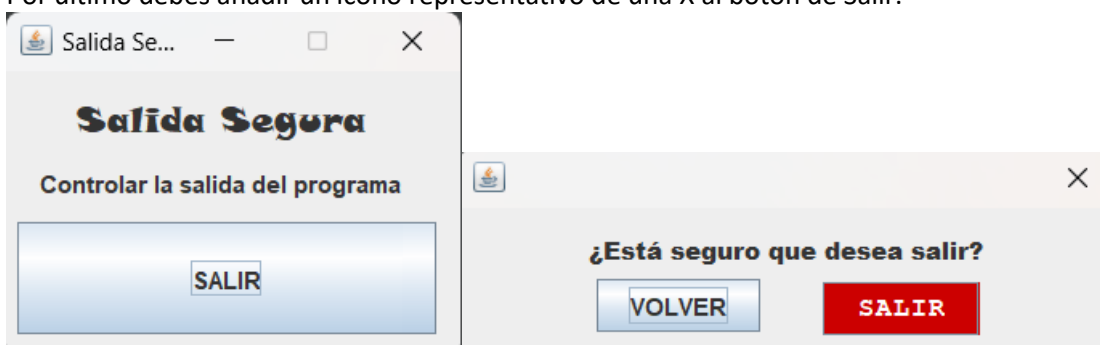
Construir una ventana de 300x400 píxeles con el título “**Conversor de Euros a Pesetas**”:

- La ventana debe mostrar una etiqueta con el nombre del programa, y debajo un botón, por ejemplo, “Ingresa Cantidad” que abre una ventana auxiliar. Esta ventana auxiliar tendrá un label con el texto: “Cantidad en euros” y un cuadro de texto donde introducir el importe. Finalmente tendrá un botón, por ejemplo “Convertir a pesetas”.
- Cuando se pulse “Convertir a pesetas”, se debe volver a la pantalla principal y que se muestre el resultado en pesetas.
- Tenéis libertad de cómo habilitar y/o deshabilitar cada parte de la interfaz según tu gusto.

Ejercicio 3: Salir Seguro

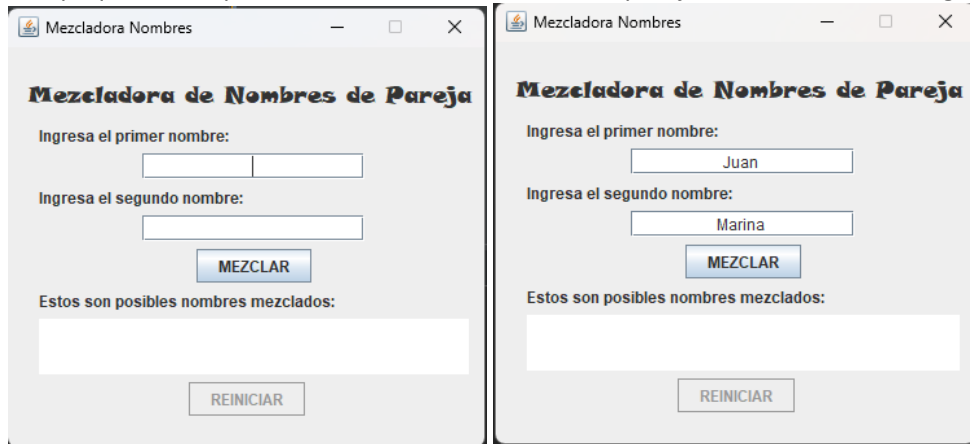
Construir un programa que controle la salida segura de él mismo:

- Muestre una ventana con un único botón de salir, y que el botón de cierre no haga nada, es decir que no se cierre, aunque se pulse:
- La ventana de salida debe tener dos botones, uno para volver y otro para salir definitivamente del programa. Así mismo su botón de cierre debe estar deshabilitado u oculto.
- Por último debes añadir un icono representativo de una X al botón de Salir.

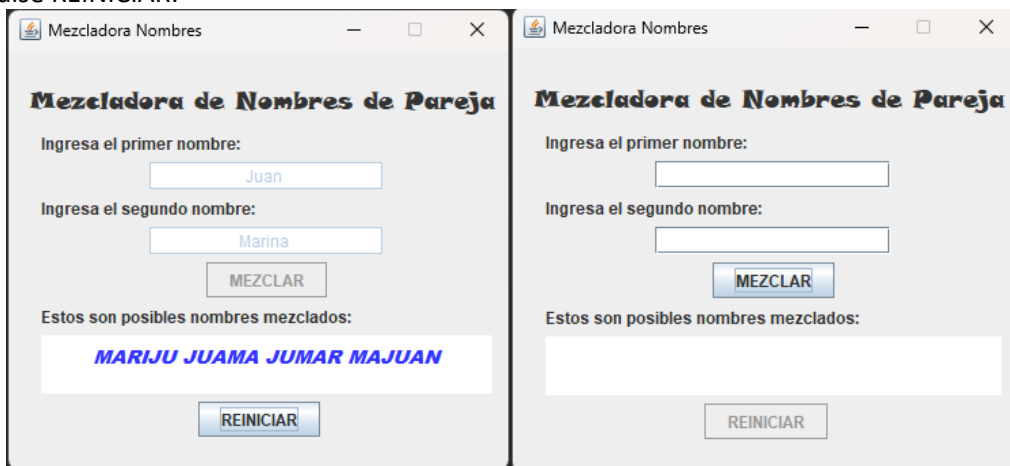


Ejercicio 4: Mezcladora de nombres

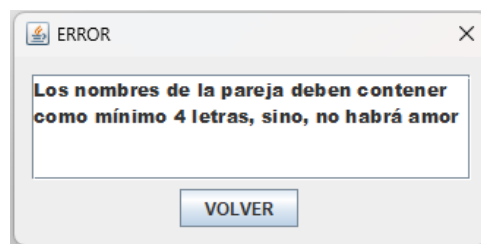
Vamos a crear una pequeña GUI para mezclar los nombres de una pareja, la interfaz será la siguiente:



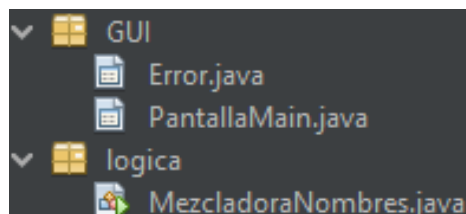
- Inicialmente estará deshabilitado la zona de resultado y el botón de REINICIAR
- Al pulsar MEZCLAR, se recogerán los valores de los nombres introducidos, y se deshabilitarán hasta que se pulse REINICIAR:



- Aparecerán **CUATRO** o más resultados de nombres nuevos para sus hijos, que serán mezcla de sus nombres.
- Al pulsar REINICIAR se reseteará de nuevo todo.
- En caso de que se introduzcan nombres con MENOS DE CUATRO caracteres se nos lanzará una nueva ventana de ERROR informándonos de ello y se reseteará todo de nuevo para introducir nombres más largos:



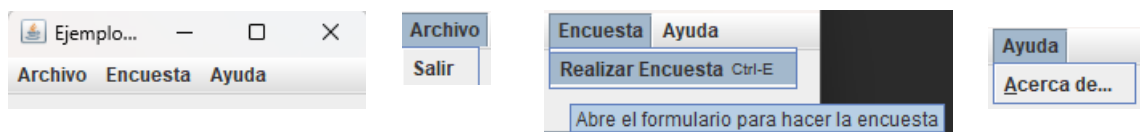
SEPARA la parte de interfaz gráfica de usuario de la lógica del programa, para ello crea la siguiente estructura:



Ejercicio 5: App Ministerio

Vamos a realizar la siguiente App que simula una pequeña parte de otra app mayor.

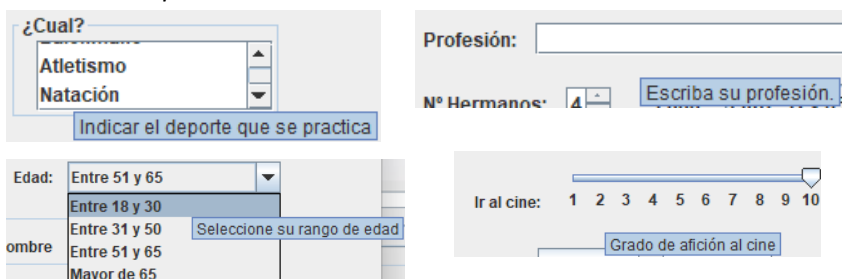
Se va a tener una barra de menú con 3 opciones: Archivo Encuesta Ayuda



El menú de Archivo -> Salir, simplemente saldrá de la aplicación.

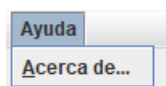
El menú de Encuesta -> Realizar Encuesta, abrirá una nueva ventana para recoger información:

- TODOS los campos del formulario deberán tener una *nota informativa* sobre lo requerido para cada *campo al pasar por encima el ratón*.



- Al pulsar “Aceptar” todos los datos de la persona se almacenan (muestran) en una **tabla** que aparecerá en la ventana de principal. Se irán añadiendo Personas a la tabla según rellenen la encuesta.
- Se debe rellenar obligatoriamente el campo de “Profesión”, si no debe avisar del error y volver. Si se pulsa “Cancelar” simplemente vuelve atrás.

El menú de Ayuda -> Acerca de ..., abrirá una nueva ventana de información sobre la app. Por ejemplo:



- Esta ventana informativa tendrá desactivadas las opciones de maximización, minimización, etc.
- Debéis personarla a vuestro gusto, colocando una imagen “corporativa” y rellenando la información de forma similar.



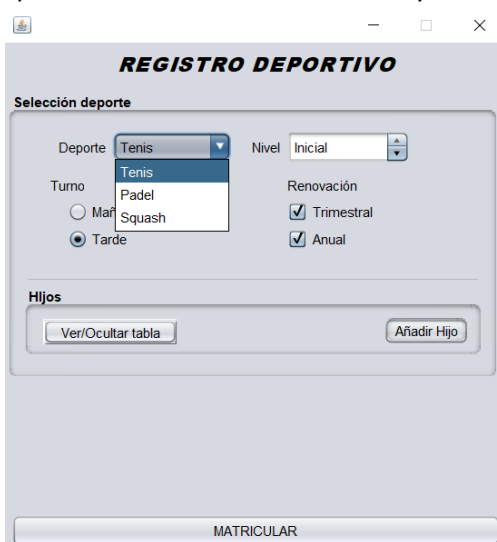
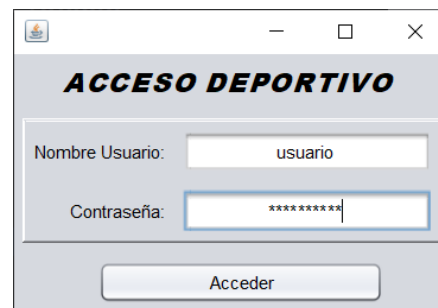
Se debe dividir el proyecto en mínimo en los siguientes paquetes:

- **gui**, dónde a su vez contenga una separación de otras dos carpetas según tipos de ventanas, principal y secundarias (que serán modales en este caso)
- **lógica**, que contendrá el control y lógica del programa, su ejecución y el almacenamiento de los datos de las personas.
- **dto**, para la creación de las clases de objetos de transferencia que necesitéis.

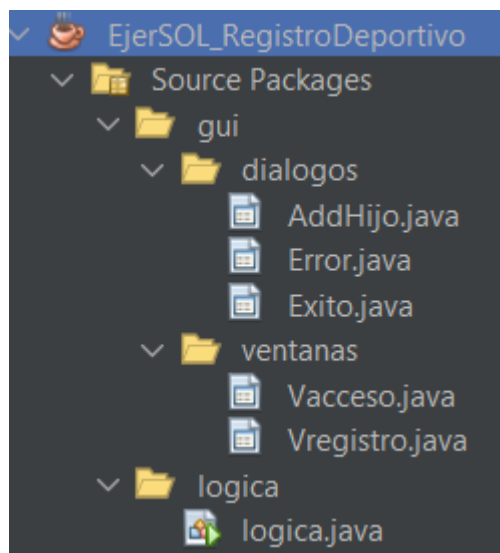
Ejercicio 6: Registro deportivo

Desarrollar la siguiente aplicación de acceso a una plataforma de registro deportivo de un club de deportes, dónde se puede entrenar y jugar a tenis, pádel y/o squash.

- La ventana de acceso será la inicial, y solicitará el nombre de usuario y una contraseña (dónde NO se podrá ver los caracteres que se están escribiendo). Si los campos están vacíos o no son correctos, saldrá un aviso de error.
- La ventana de registro deportivo tendrá un desplegable para seleccionar el deporte, una lista con los niveles (*Inicial, Medio, Avanzado y Profesional*), un turno (a escoger **sólo uno** entre mañana o tarde), y opciones de renovación trimestral y anual.



- Se podrá añadir hijos desde un botón que nos llevará a una ventana para rellenar los datos del hijo. Los datos añadirán una nueva fila a la tabla de hijos del registro. Además, existirá un botón para **ver/ocultar** la tabla de hijos.



NOTAS IMPORTANTES:

- Se usará un *JPaswordField* para la contraseña
- Para el **nivel** se usará un *JSpinner*, igual que para la **fecha**, pero la fecha se mostrará en formato *día/mes/año*
- El **turno** deberá o ser mañana o tarde, pero nunca ambos al mismo tiempo
- Se debe seguir la estructura de directorios y nombres de archivos de la imagen obligatoriamente.
- Todos los posibles errores** que se produzcan abrirán un diálogo de **Error** y explicarán brevemente el motivo
- Cuando se realice correctamente la matrícula se informará con el diálogo de **Éxito**
- Añadir directorio **dto** para la gestión y almacenamiento en tabla de los **hijos**.

Ejercicio 7: Paneles de colores

Crear una interfaz como la siguiente dónde se deberán usar 3 paneles:

1. Un panel de fondo general con libre disposición, dónde se podrá colocar el botón “Restablecer” dónde se considere oportuno.
2. Un panel interior que contenga los botones de cambio de color. La disposición será por tabla, con dos filas y dos columnas para los botones, y con el espaciado suficiente para ver el color de fondo.
3. Un panel interior que contenga el listado del tipo de zona a cambiar de color. Tendrá una disposición de caja, en la que los elementos que se coloquen se irán posicionando verticalmente.

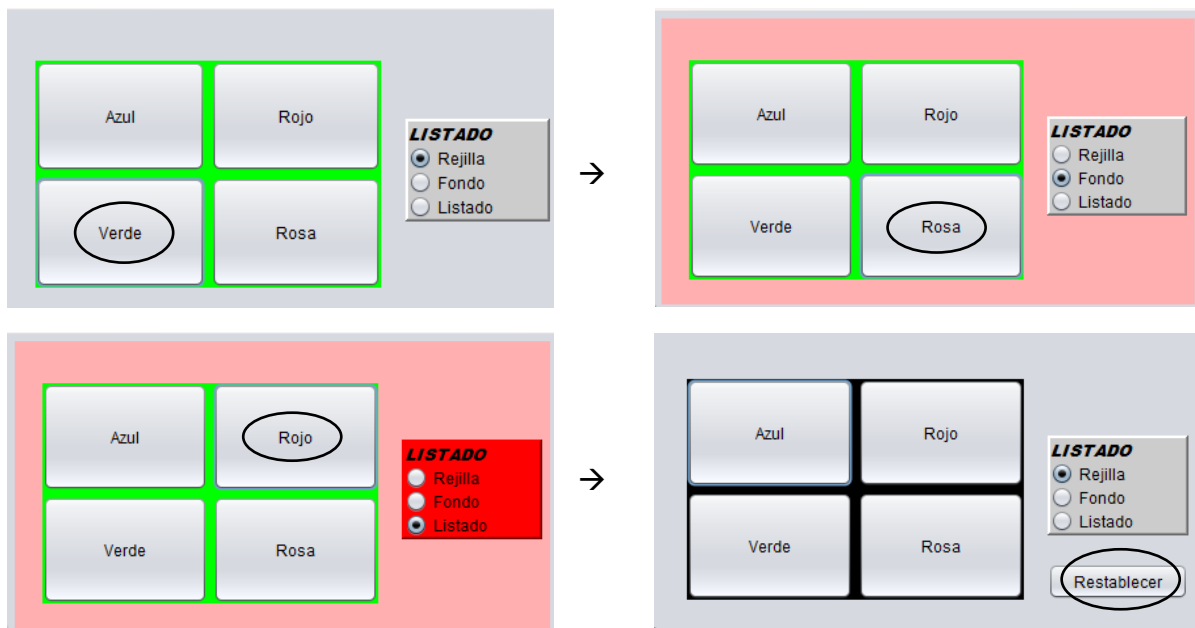


Cuando se pulse en alguno de los botones de cambio de color, la zona que esté activa en el listado de opciones cambiará a ese color. Se podrá ir cambiando la zona de cambio de color según sea:

- “Rejilla”, espacio de separación entre los botones de cambio de color.
- “Fondo”, panel de fondo general.
- “Listado”, cajetín del listado de colores.

Al pulsar el botón de “Restablecer” se volverá a la gama de colores por defecto, es decir la original.

Por ejemplo, se presenta una posible sucesión de cambios:



Ejercicio 8: Juego de paneles

Crear una interfaz que realice un cambio entre paneles.

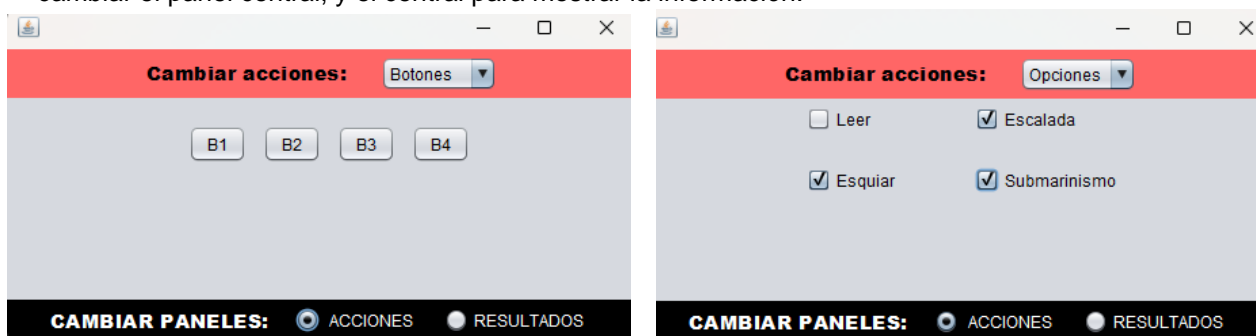
Para ello vamos a usar 2 tipos de disposiciones:

1. *BorderLayout* para los menús de cambios
2. *CardLayout* para ir cambiando entre paneles.

En la presentación inicial se mostrará la presentación tal cual la imagen. Consta de dos partes, un panel central ("Juego de Paneles") que será el que cambie, que ocupar todo el centro al fondo, que será el que gestione el cambio de los paneles del centro.

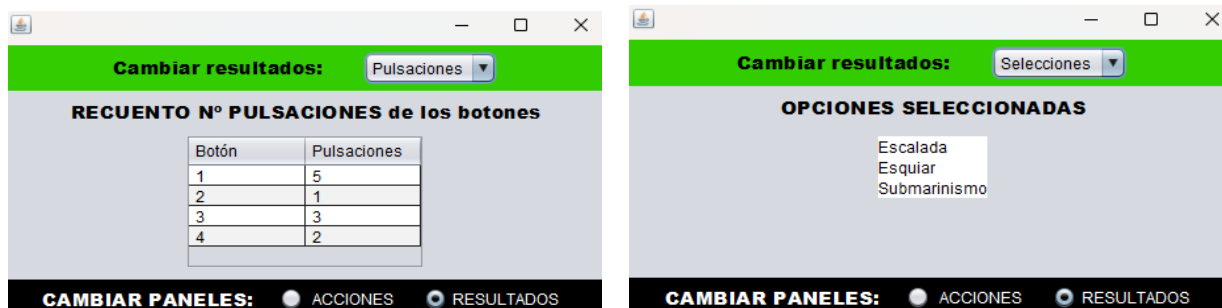
El usuario después podrá escoger entre ACCIONES o RESULTADOS, y ambos son otros dos paneles con la misma distribución que la anterior, es decir, el panel central a su vez irá cambiando entre un panel de "Acciones" y otro de "Resultados":

- a) Panel ACCIONES: Constituido por dos paneles, uno superior y otro central. El superior servirá para cambiar el panel central, y el central para mostrar la información:



El panel superior tendrá una lista para cambiar los diferentes subpaneles (subPanelBotones y subPanelOpciones), el central acciones que se irán recogiendo para posteriormente guardarlas en el panel de RESULTADOS.

- b) Panel de RESULTADOS: su funcionamiento es idéntico al de ACCIONES



Inicialmente el panel de "resultados" estará sin rellenar. La tabla sólo con los botones y lista de opciones vacía.

En el subpanel de *pulsaciones* mostrará una tabla con las veces que se ha pulsado cada botón, mientras que el subpanel *selecciones* mostrará una lista con las opciones seleccionadas.

Ambos subpaneles, se irán actualizando conforme se realicen más acciones. En el caso de las opciones, si se desmarcan, también deben eliminarse/añadirse a lista. Esta lista deberá ser etiquetas que se van añadiendo/eliminando de un contenedor (otro panel).

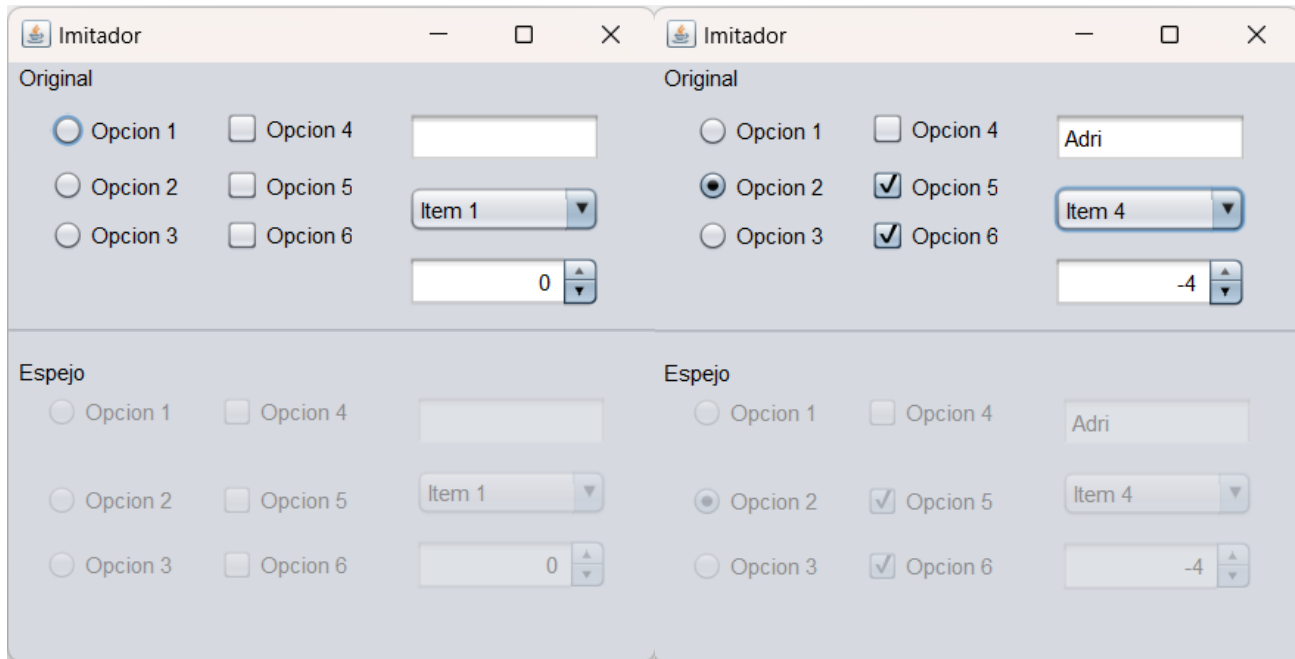
IMPORTANTE:

- ✓ Cada vez que se cambie de panel se requiere:
 - **refrescar los paneles**, es preciso crear un método para ello donde se usen adecuadamente los métodos *repaint()* y *revalidate()*. Investiga sobre ellos y anota con comentarios en el propio código cómo funcionan exactamente.
 - **Cambiar panel**, se requiere que el panel padre vacíe todos sus elementos y después añada el nuevo panel hijo. Investiga acerca de cómo hacerlo con los métodos *removeAll()* y *add(panel)*
- ✓ El truco del ejercicio está en tener siempre un panel padre con disposición *BorderLayout*, y dos paneles hijos:
 - Un panel CENTRAL que es el que contiene los subpaneles, con una disposición *CardLayout*.
 - Un panel de CAMBIO, que es el que gestiona el cambio de subpaneles. Su disposición variará según lo que necesitemos.

Ejercicio 9: Imitador

Vamos a crear una pequeña aplicación gráfica que “imita”, es decir duplica las acciones como si fuera un **espejo**.

Tendremos dos pares de conjuntos de componentes separados mediante un *separador*, y cuando pulsamos en un componente o escribimos en un campo, se cambiará en su correspondiente del otro conjunto:



Vamos a usar los siguientes componentes: *JTextField*, *JRadioButton*, *JCheckBox*, *JComboBox* y *JSpinner*.

El truco de este ejercicio es buscar y saber el evento correcto que gestiona lo que necesitamos.

IMPORTANTE:

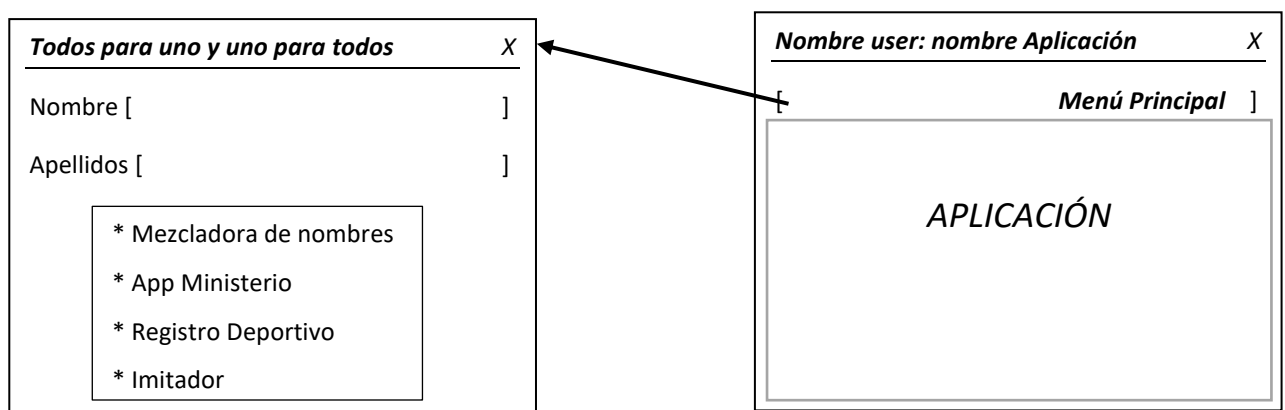
Sólo podéis modificar de un lado, el otro conjunto de componentes NO se puede tocar, es decir no es bidireccional.

Ejercicio 10: Todos para uno y uno para todos

Vamos a **mejorar los ejercicios anteriores y crear una interfaz de selección** de algunas de nuestras aplicaciones, para ello:

- Crear una **interfaz de selección** en la que se pueda escoger entre las aplicaciones:
Mezcladora de nombres, App Ministerio, Registro deportivo e Imitador
 - Debe cargarse directamente desde la lógica
 - Se podrá escoger mediante un icono representativo y su texto (Ej: botón más un label)
 - Debe obtener el nombre y apellidos del usuario, que después se corresponderá con el título de ventana concatenado con el nombre de la app. Por ejemplo:
Pepito Grillo Fernández: Mezcladora de nombres
 - El título de la ventana de principal será siempre: “*Todos para uno y uno para todos*”, pero cuando se esté en otra aplicación cambiará como se ha descrito anteriormente.
 - Una vez cargada la app en concreto, se le debe añadir a cada una un botón llamado “**Menú Principal**” que **SIEMPRE debe estar visible en la parte superior derecha** como si fuese una barra horizontal.
- Vamos a hacer uso de **JOptionPane** para TODOS los diálogos de error, mensajes informativos, que pidan una pequeña información, etc. Se tendrán que sustituir todas los JDialog de ese tipo por la construcción correcta de este tipo de paneles.
- Para los ejercicios con tablas ahora deberemos usar **AbstractTableModel** para cada conjunto de registros que se visualicen en una tabla, es decir deberá separar el modelo de tabla de la vista. Se llamará **tableModels**.
- Se debe mantener la jerarquía de carpetas: **dto, gui, lógica**.
La **gui** puede jerarquizarse según necesitéis, pero se debe tener una separación de ventanas, diálogos y modelos de tabla.
- La lógica del programa será la que en todo momento gestione los cambios entre las diferentes aplicaciones.

UN POSIBLE EJEMPLO DE DISEÑO:



Pero cada uno que lo adapte como más le interese.