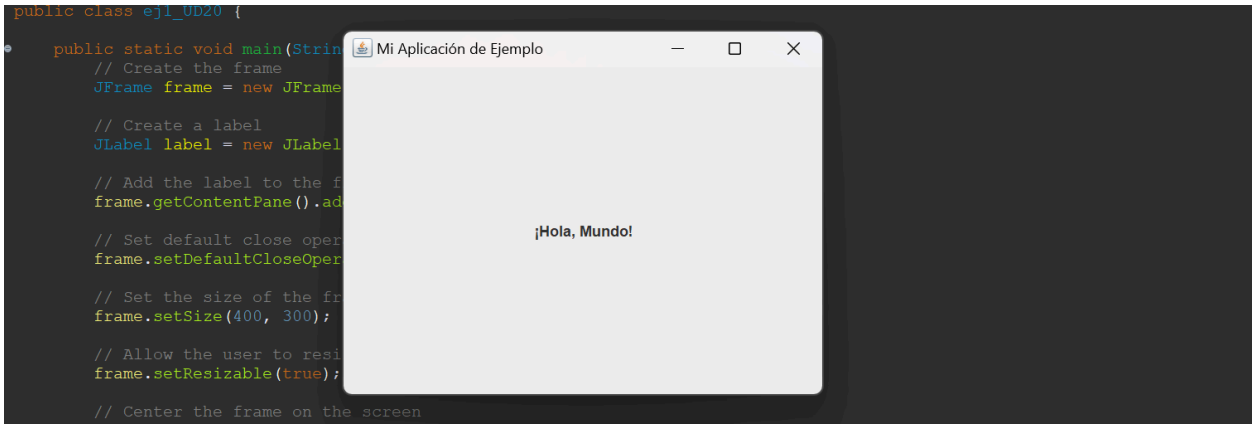
T020_Tareas

Ejercicios_UD20_maven · isabelmvi/Java-Techtalent-2024-local@f37e4f0 (github.com)

Ejercicio 1

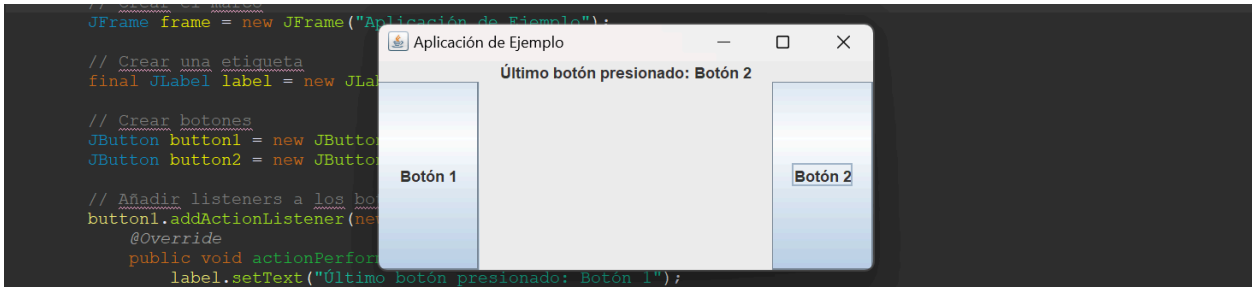
Intenta escribir una aplicación con interfaz gráfica en la que se construya una ventana con título y marco que tenga los controles básicos (es decir, restaurar, maximizar y cerrar) y que al pulsar sobre el aspa de la ventana (cerrar) se salga completamente de la aplicación. La ventana contendrá una etiqueta y el usuario debe poder cambiar su tamaño.



Este código crea una aplicación Java con una ventana (`JFrame`) que muestra una etiqueta centrada con el texto "¡Hola, Mundo!" (`JLabel`). Utiliza `BorderLayout` para centrar la etiqueta, establece el tamaño de la ventana, permite su redimensionamiento, y maneja el evento de cierre de la ventana con un `WindowAdapter` para finalizar la aplicación. La ventana se centra en la pantalla y se hace visible al usuario.

Ejercicio 2

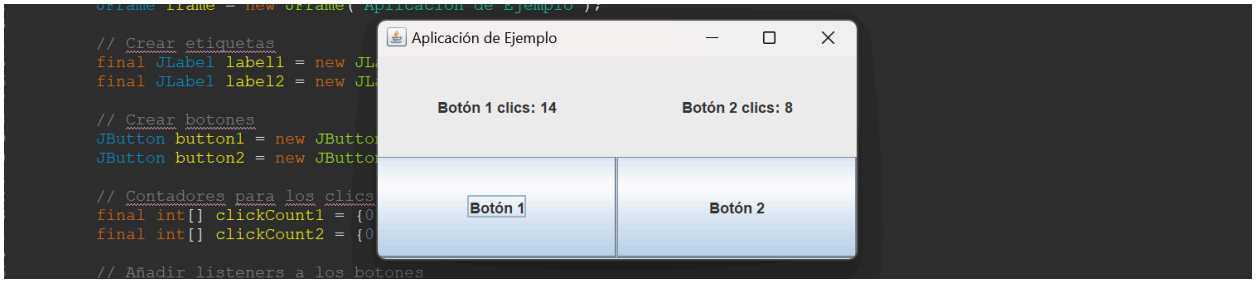
Escribe una aplicación gráfica con una ventana que tenga una etiqueta y dos botones de operación. El comportamiento de la aplicación debe reflejar en el texto de la etiqueta cuál es el último botón en el que el usuario ha hecho clic.



Este código crea una aplicación Java con una ventana (`JFrame`) que muestra una etiqueta y dos botones. La etiqueta inicializa con el texto "Presiona un botón" y cambia para mostrar cuál botón fue presionado usando `ActionListener` en cada botón. Se usa `BorderLayout` para posicionar la etiqueta en la parte superior y los botones a los lados. La ventana se configura para permitir el redimensionamiento, se centra en la pantalla, y se cierra adecuadamente cuando el usuario decide salir.

Ejercicio 3

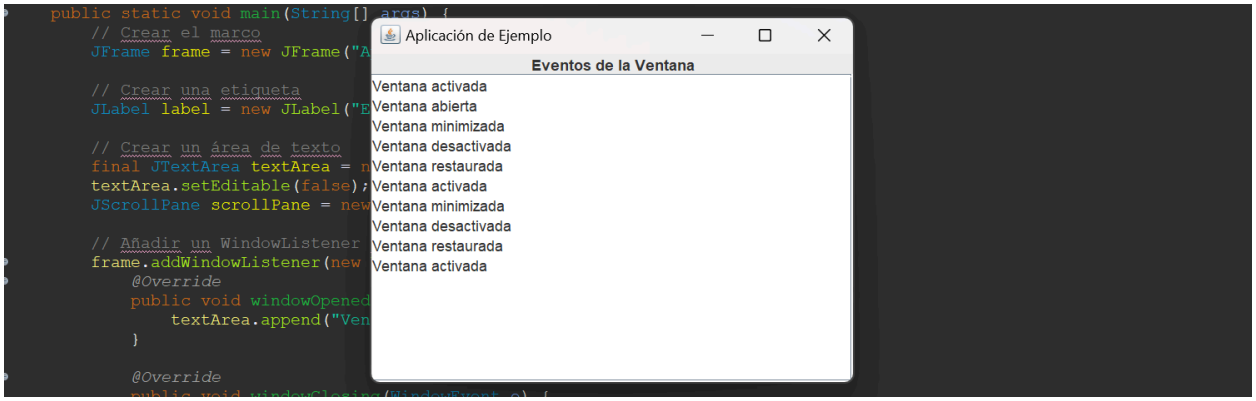
Intenta escribir una aplicación gráfica con una ventana que tenga dos etiquetas y dos botones de operación. El comportamiento de la aplicación debe reflejar en el texto de las etiquetas el número de veces que el usuario ha hecho clic en cada uno de los botones.



Este código crea una aplicación Java con una ventana (`JFrame`) que muestra dos etiquetas y dos botones, contando las veces que cada botón es presionado. Las etiquetas se actualizan con los contadores de clics usando `ActionListener` en cada botón. Se utiliza un diseño de cuadrícula (`GridLayout`) para organizar las etiquetas y botones en una matriz de 2x2. La ventana se configura para permitir el redimensionamiento, se centra en la pantalla, y se cierra adecuadamente cuando el usuario decide salir.

Ejercicio 4

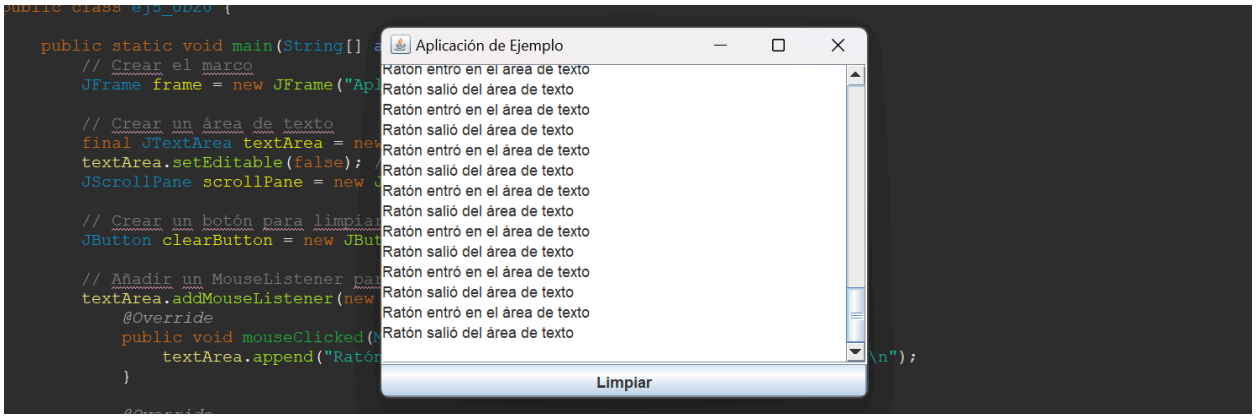
Intenta escribir una aplicación gráfica con una ventana que tenga una etiqueta y un área de texto. La aplicación debe reflejar en el área de texto todos los eventos de ventana que se produzcan por la creación de la ventana o por las interacciones del usuario.



Este código crea una aplicación Java con una ventana (`JFrame`) que muestra una etiqueta y un área de texto que registra eventos de la ventana. Utiliza `WindowListener` para capturar y registrar eventos como abrir, cerrar, minimizar y restaurar la ventana. Se usa `BorderLayout` para posicionar la etiqueta en la parte superior y el área de texto en el centro con un `JScrollPane` para la funcionalidad de desplazamiento. La ventana se configura para permitir el redimensionamiento, se centra en la pantalla y se cierra adecuadamente cuando el usuario decide salir.

Ejercicio 5

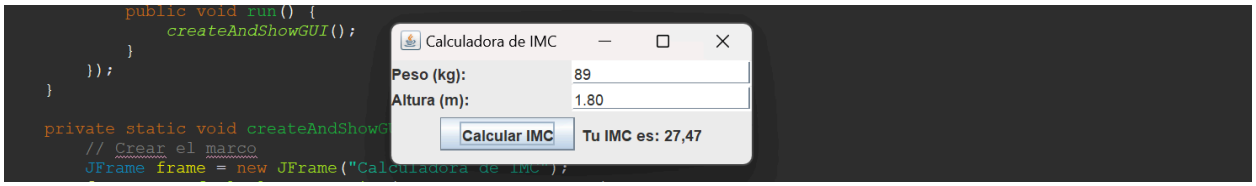
Escribe una aplicación gráfica con una ventana que tenga un botón y un área de texto. La aplicación debe reflejar en el área de texto los principales eventos de ratón que se produzcan sobre dicha área por las interacciones del usuario. Haciendo clic en el botón se limpiará el contenido del área de texto.



Este código crea una aplicación Java con una ventana (`JFrame`) que contiene un área de texto (`JTextArea`) y un botón para limpiar el área de texto. Se añade un `MouseListener` al área de texto para capturar eventos de ratón, como clics, presiones y liberaciones, así como la entrada y salida del ratón del área de texto. También se añade un `ActionListener` al botón para borrar el contenido del área de texto cuando se presiona. La ventana utiliza un diseño de borde (`BorderLayout`) para organizar el área de texto en el centro y el botón en la parte inferior. Se permite el redimensionamiento de la ventana y se cierra adecuadamente cuando el usuario decide salir.

Ejercicio 6

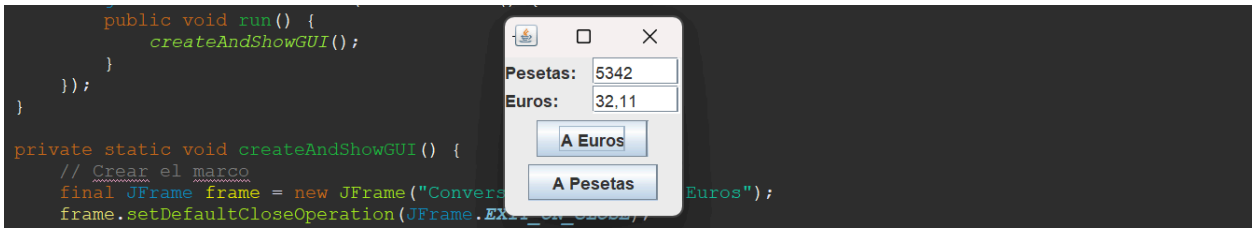
Intenta escribir una aplicación gráfica que permita calcular el índice de masa corporal. Os recuerdo que este índice se calcula dividiendo el peso de una persona en kilos por el cuadrado de su altura en metros.



Este código crea una aplicación Java que calcula el Índice de Masa Corporal (IMC) a partir del peso y la altura ingresados por el usuario. Utiliza una ventana (`JFrame`) con dos paneles: uno para ingresar los datos y otro para mostrar el resultado del cálculo. Se configuran etiquetas para el peso y la altura, campos de texto para ingresar los valores, un botón para calcular el IMC y un `JLabel` para mostrar el resultado. Se utiliza un diseño de bordes (`BorderLayout`) para organizar los paneles y se establecen los tamaños y visibilidad del marco. Cuando el usuario presiona el botón de cálculo, se extraen los valores de los campos de texto, se calcula el IMC y se muestra en el `JLabel` de resultados. La aplicación se cierra adecuadamente cuando el usuario decide salir.

Ejercicio 7

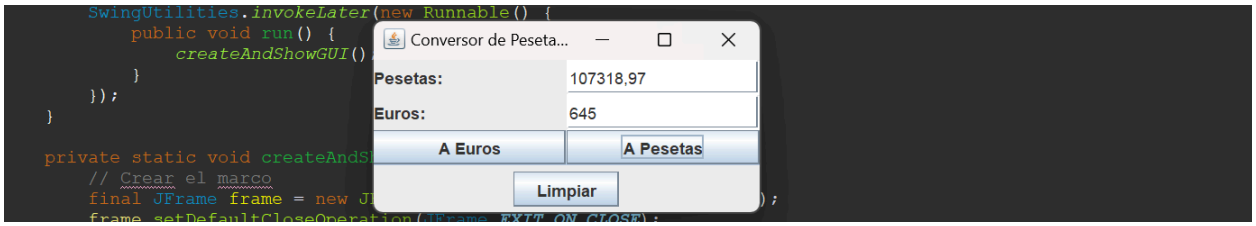
Intenta escribir una aplicación gráfica que permita de forma sencilla realizar el cambio de pesetas a euros (y viceversa). Por si acaso, os recuerdo que la tasa de cambio que se aplica es 1 euro a 166,386 pesetas. En todo momento, el usuario debe estar informado de la conversión que se está realizando.

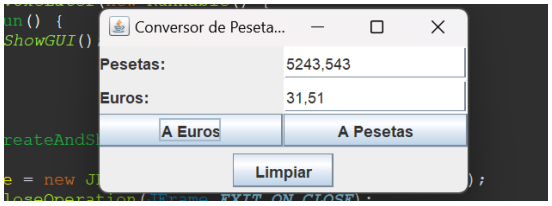


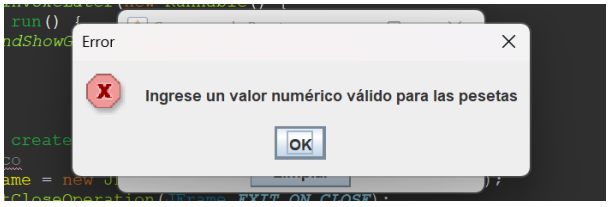
Este código crea una aplicación Java con una ventana (`JFrame`) que permite convertir entre pesetas y euros. Se utilizan dos paneles: uno para la entrada de datos y otro para mostrar los resultados. En el panel de entrada, hay etiquetas y campos de texto para ingresar la cantidad en pesetas y euros. En el panel de resultados, hay botones para realizar las conversiones. La ventana se configura para tener un tamaño inicial de 300x150 píxeles y se muestra al usuario. Los eventos de clic de los botones de conversión actualizan los campos de texto con los resultados de la conversión. Además, se manejan errores de entrada mediante ventanas emergentes de error. La aplicación se ejecuta en el hilo de eventos de Swing usando `SwingUtilities.invokeLater()`.

Ejercicio 8

Intenta mejorar la aplicación gráfica del ejercicio anterior. Además de las funcionalidades anteriores, se debe permitir operar con los botones desde el teclado y hay que añadir un botón que permita borrar los campos de datos y resultado. También para darle mayor robustez, puedes incluir un control de errores que avise mediante una ventana emergente si se ha introducido un número en formato erróneo, evitando que la aplicación se detenga.







Este código crea una aplicación Java que convierte entre pesetas y euros. Utiliza `JFrame` como la ventana principal y `JPanel` para organizar los componentes de entrada y resultado. Hay etiquetas, campos de texto y botones para ingresar valores y realizar conversiones. Además de la interacción con el ratón, se permite la interacción con los botones usando el teclado (la tecla Enter). Cuando se ingresa un valor numérico y se presiona Enter, se realiza la conversión correspondiente y se muestra el resultado. Si se ingresan valores no numéricos, se muestra un mensaje de error usando `JOptionPane`. La aplicación se cierra correctamente al salir.

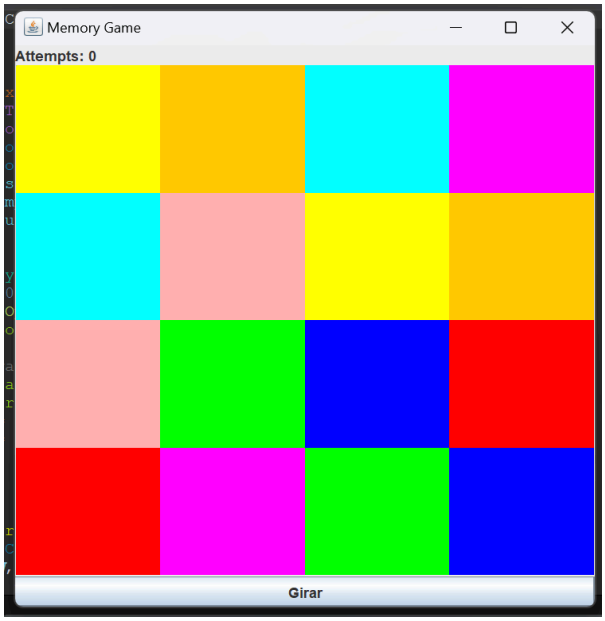
Ejercicio 9

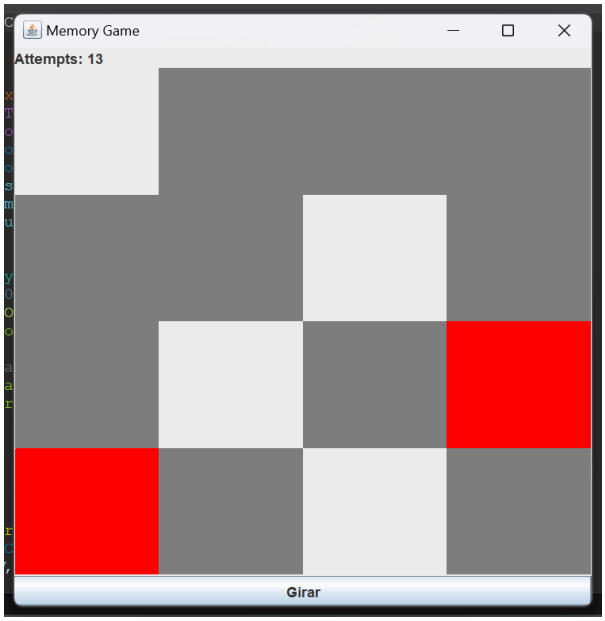
Crea dentro de la aplicación Java de la unidad un nuevo JFrameForm que simule un juego de memoria con al menos 16 cartas. Deberá realizar las siguientes operaciones:

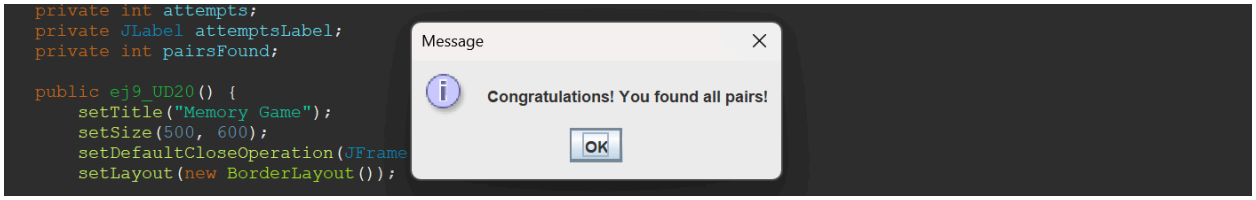
- Las cartas estarán implementadas con botones de tipo ToggleButton y estarán perfectamente distribuidas en la pantalla.
- Al pulsar la carta, ésta cambiará de color (habrá una pareja de cartas de cada color).
- El jugador únicamente podrá mostrar dos cartas cada vez.
- Si las dos cartas pulsadas son del mismo color, desaparecerán de la pantalla.
- Si las dos cartas pulsadas son de diferente color, volverán a su color original.

RETOS ADICIONALES:

- En lugar de utilizar colores de botones, utiliza imágenes.
- Incluye una etiqueta que vaya contabilizando los intentos y que muestre un mensaje cuando el jugador acaba el juego.
- Si el jugador desbloquea todas las cartas, se mostrará un mensaje (MessageDialog) de enhorabuena.







Este código implementa un juego de memoria en Java utilizando componentes Swing. La aplicación crea una ventana (`JFrame`) titulada "Memory Game" con un panel de cartas que contiene botones de alternancia (`JToggleButton`). Cada botón representa una carta con un color aleatorio, y al hacer clic en una carta, se muestra su color. El objetivo del juego es encontrar todas las parejas de cartas con el mismo color. Se proporciona un botón "Girar" para ocultar los colores de todas las cartas después de memorizarlas. Además, se lleva un seguimiento del número de intentos realizados y se muestra un mensaje de felicitación cuando se encuentran todas las parejas. La ventana es redimensionable, se cierra correctamente cuando se presiona el botón de cierre y se centra en la pantalla al abrirse.