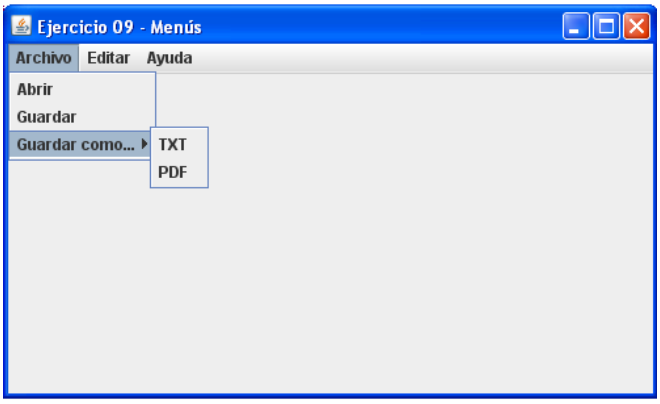
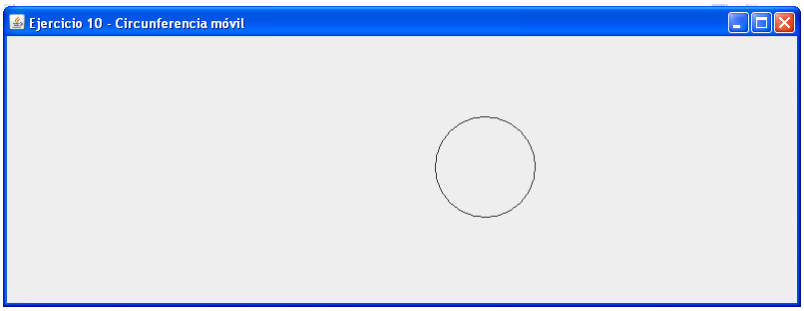
Ejercicios de interfaces de usuario

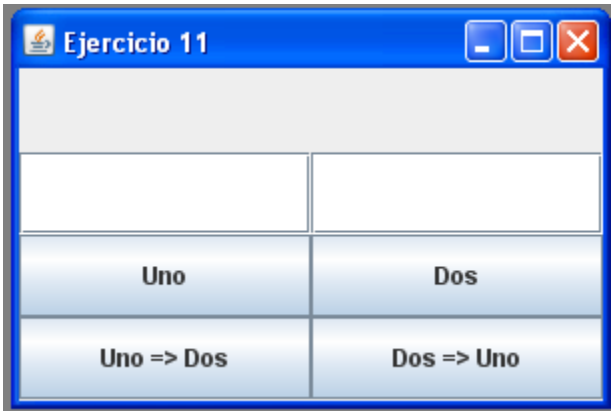
1. Crear y visualizar una ventana que se titule “Mi primera aplicación” de tamaño (400x200) ubicada en el punto (100,200), que se pueda cerrar con el aspa.
2. Igual que el anterior, pero en este caso la clase Ej02 es un JFrame, es decir, hereda de JFrame
3. Igual que el anterior, pero el título de la ventana se pregunta previamente por teclado (clase Scanner). Ubicar la clase en el centro de la pantalla, sea cual sea su resolución. PISTA: Ver API de Toolkit (métodos getToolkit() y getScreenSize(). Atributos width y heigth de Dimension)
4. Crear una ventana de 200x200, que partiendo desde el centro de la pantalla se mueva 200 puntos a la derecha. Pista: Hacer un bucle vacío para implementar un pequeño retraso entre cada actualización de la posición de la ventana.
5. Crear una ventana que, partiendo desde el centro de la ventana, tome una dirección al azar entre (NE,SE,NW,SW) y rebote en los bordes de la pantalla de forma infinita.
6. Crear una ventana que represente la bandera de España. Probar con la de Italia también.
7. Crear una ventana con fondo azul que diga “Hola mundo” en blanco.
8. Igual que el anterior, pero el texto se le pide al usuario antes de la aparición de la ventana.
9. Realizar una ventana que contenga un menú con los elementos “Archivo”, “Editar” y “Ayuda”. “Archivo”, tiene los submenús “Abrir”, “Guardar” y “Guardar como...”. Éste último a su vez despliega un submenú con las opciones “TXT” y “PDF”. El menú editar contiene los elementos “Cortar”, “Copiar”, “Pegar”.



10. Crear una aplicación que, utilizando la clase Graphics, haga avanzar una circunferencia de izquierda a derecha y se pare.

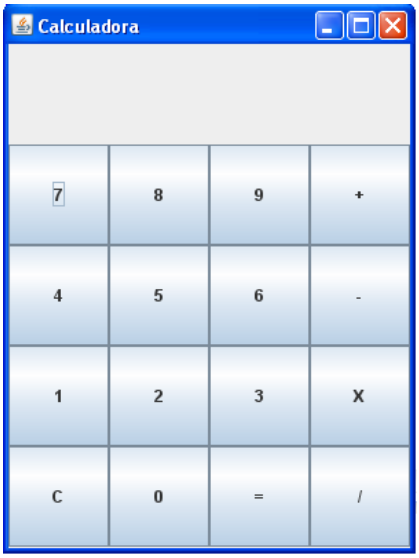


11. Crear una interfaz con el siguiente aspecto.

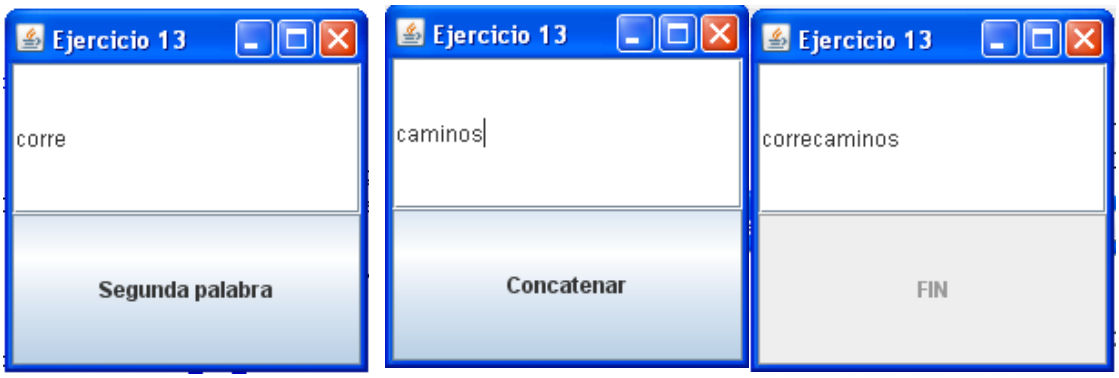


Cuando se pincha sobre “uno” aparece en el JLabel de arriba la palabra escrita en el primer JTextField. Cuando se pincha en “dos”, la palabra del segundo JTextField. Al pinchar en “uno=>dos” o “dos=>uno” aparecen ambas concatenadas en el orden indicado.

12. Crear la UI de una calculadora básica de enteros (con suma, resta, multiplicación y división entera), con el siguiente aspecto.

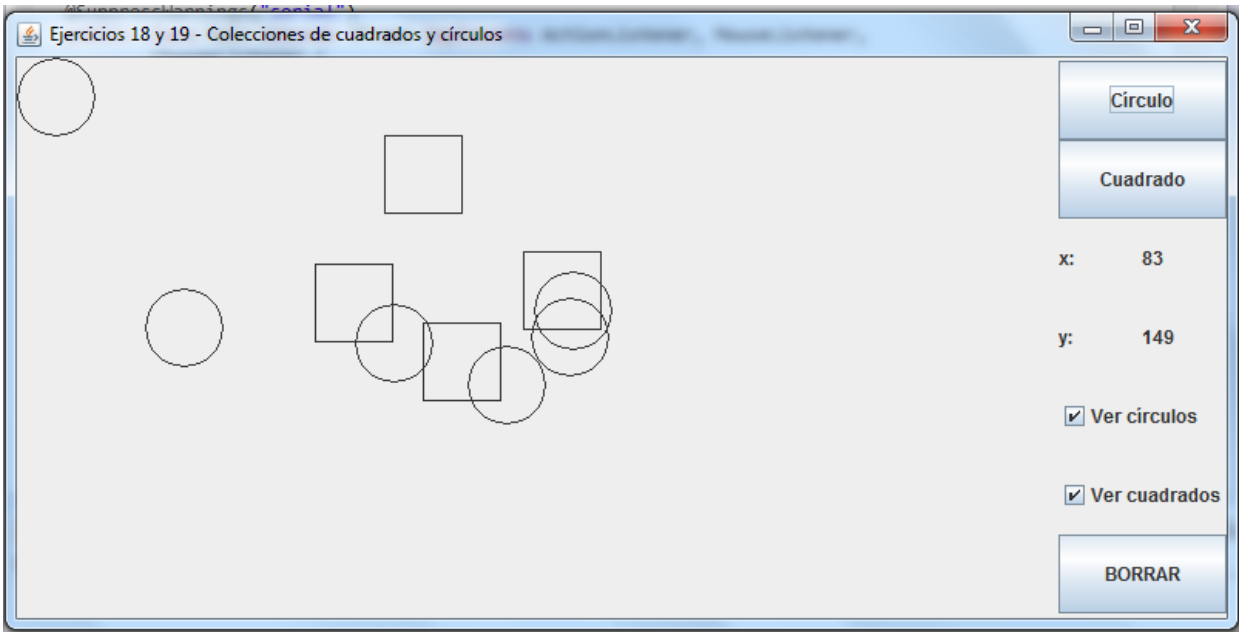


13. Hacer una UI que tenga un JTextField y un JButton etiquetado como “Segunda palabra”. La UI implementará una máquina de estados sencilla en la que la primera pulsación sobre el botón provocará que se memorice lo que hubiera escrito en el JTextField, se borre y el botón cambie su etiqueta por “Concatenación”. Llegada esta situación, si se vuelve a pulsar el botón, el programa mostrará en el JTextField la primera palabra que introdujimos concatenada con la palabra actual, y además se desactivará el botón (método setEnabled(boolean)), y se reetiquetará como “FIN”



14. Completar el ejercicio 12 utilizando una máquina de estados, para que la calculadora se comporte de la siguiente forma intuitiva. En un principio la calculadora muestra el "display" en blanco y queda a la espera de introducir el primer operando. Cuando se pulsa un número se concatena por la derecha con el número que hubiera escrito en la pantalla. Cuando se pulsa sobre una operación se memoriza el número introducido se borra el número del display y aparece el signo de la operación, pudiéndose cambiar éste varias veces hasta que se empieza a introducir el primer dígito del segundo operando. Cuando se pulsa "=", aparece el resultado de la operación, quedando la calculadora preparada para que se vuelva a escribir el primer dígito del primer operando de otra operación. Si en cualquier momento se pulsa "C" la pantalla se borra y la calculadora queda a la espera de la introducción del primer operando. NOTA: En esta calculadora básica no se admite el ENCADENAMIENTO de OPERACIONES. Cada vez que se obtiene un resultado se comienza una nueva operación, no pudiéndose reutilizar este resultado obtenido.
15. Hacer el juego del buscaminas para un tablero de 9x9 y 10 minas al azar. Utilizar un BORDER_LAYOUT en cuya parte superior está el botón del reset, y en el centro el resto de los 81 botones.

    Sugerencias: crear una clase Mina que herede de JButton, responsable de saber en qué posición está y cuántas minas tiene a su alrededor (desde 0 hasta 8, si tiene 9, ella misma es una mina). Hacer una clase CampoDeMinas que herede de JPanel y que mantenga una array 9x9 de "Mina" Cuando se pulsa un botón, pueden ocurrir dos cosas. Si no hay mina, se cambia el label del botón por el número de minas que tiene alrededor, y se desactiva el botón. Si hay mina se descubren las posiciones de las minas de todo el tablero.
16. Mejorar el programa anterior para que cuando se haga clic sobre una casilla con cero minas alrededor, despeje todas las casillas a su alrededor, y pida recursivamente, a aquéllas que también tengan cero minas, que se despejen. PISTA: para evitar recursividades infinitas, asegurarse de "marcar" el botón que se está limpiando (por ejemplo desactivándolo) antes de la llamada recursiva
17. Hacer una aplicación que en un JFrame de 500x500 genere un rectángulo de posición y tamaño determinados al azar (el tamaño oscilará entre 50x50 y 100x100). La aplicación permitirá hacer clic sobre ella. Si el clic ocurre dentro de los límites del rectángulo, se rellenará de color, si no no ocurre nada.
18. Hacer una aplicación swing con el siguiente aspecto:



Cuando se hace clic sobre el lienzo principal, el JLabel etiquetado como “x” cambia su texto por la coordenada x del lienzo sobre la que se ha hecho el clic. El JLabel “y” hace lo propio para la coordenada vertical. Cuando se presiona sobre el botón “cuadrado” se dibuja un cuadrado sobre el lienzo de 50x50, y ocurre algo similar para círculo. Todos estos cuadrados y círculos se irán añadiendo al lienzo. Cuando se pulsa el botón “borrar” se borra el lienzo.

19. Mejorar el programa anterior añadiendo un par de checkboxes que me permitan ver sólo los círculos o sólo los cuadrados que haya dibujado. **PISTA**; Usar JCheckBox, implementando una clase anónima “ChangeListener”, a la que suscribirse a eventos de cambio del estado del checkbox mediante addChangeListener(..). Implementar stateChanged(ChangeEvent) para reaccionar a cambios en los checkboxes y comprobar mediante el método “isSelected()” de los checkboxes si están activados o no.