Trabajo Práctico 6

Ejercicio 1: Crea una aplicación GUI en Java que gestione una lista de productos. La interfaz debe contener los siguientes elementos:

1. Un ComboBox (*JComboBox*) que permita al usuario seleccionar una categoría de producto de una lista predefinida (por ejemplo, electrónica, ropa, alimentos, etc.).
2. Un botón "Agregar producto" (*JButton*) que permita al usuario agregar un nuevo producto a la lista. Validar campos con datos o vacíos, dentro de un mismo panel.
3. Una tabla (*JTable*) que muestre los productos agregados, con columnas para el nombre del producto, la categoría y el precio.

Cuando el usuario seleccione una categoría en el ComboBox y haga clic en el botón "Agregar producto", el programa debe permitir al usuario ingresar el nombre y el precio del producto en campos de texto separados. Luego, debe agregar el producto a la tabla con la categoría seleccionada.

Además, la tabla debe ser actualizada automáticamente cada vez que se agrega un nuevo producto. **Subir proyecto a GitHub y enviar link del repositorio.**

Ejercicio 2: El supermercado "DeTodo S.A." nos ha pedido crear una demo del módulo productos que permita: dar de alta, bajas y modificar productos del establecimiento y una serie de consultas de dichos productos por: rubro, nombre y precios. De cada Producto, interesa conocer: código (numérico), descripción, precio, stock y rubro (comestible, limpieza y perfumería son los únicos permitidos). Para dar solución a este requerimiento, lo haremos implementando un `TreeSet` de Productos, en donde haremos que se ordenen por código a medida que se van ingresando.

Es importante que repartan las tareas entre los miembros del equipo, de tal forma que: uno crea repositorio en **GitHub**; los demás clonan, luego cada uno hace una de las clases que forman parte del proyecto y suben al repositorio los cambios.

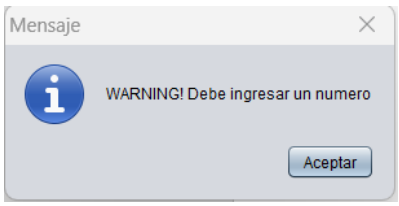
Utilice `JFrame`, `JDesktopPane`, `JInternalFrame`, `JMenu`, `JMenuBar`, `JMenuItem`, `JTable`, `JComboBox`, `JSpinner`, y `JPanel`. *Validar campos con datos o vacíos, dentro de un mismo panel. Valide los NumberFormatExeption al cargar. Recuerde el uso de eventos como: FocusLost y use requestFocus para mostrar un dialogo de ingreso incorrecto, y pedir reingreso.*

LABORATORIO 1

Armar una barra de menú que contenga: dos JMenu para Administración y Consultas. Del primero se invoca la vista para gestionar los productos. Del segundo JMenu, se extienden opciones para Consultar por Nombre, por Precio, y por Rubro.

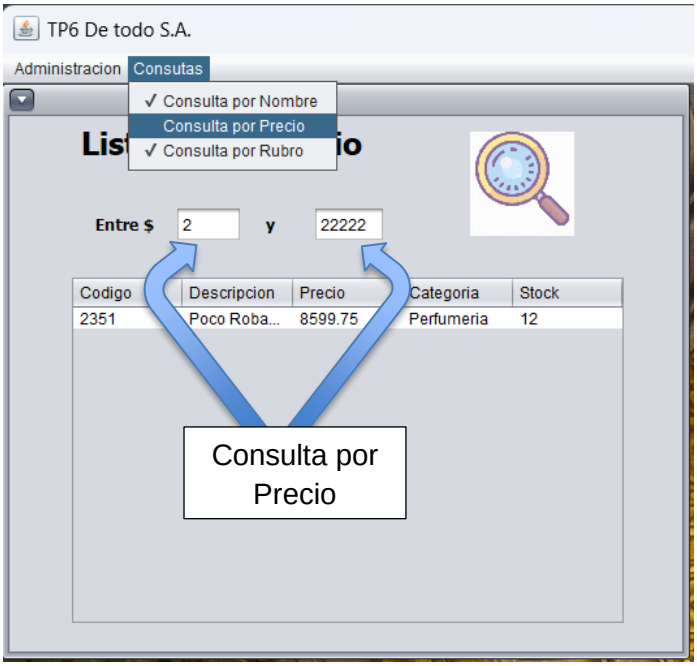
A continuación, mostramos las vistas sugeridas para el menú general, la gestión de productos y para las consultas solicitadas.

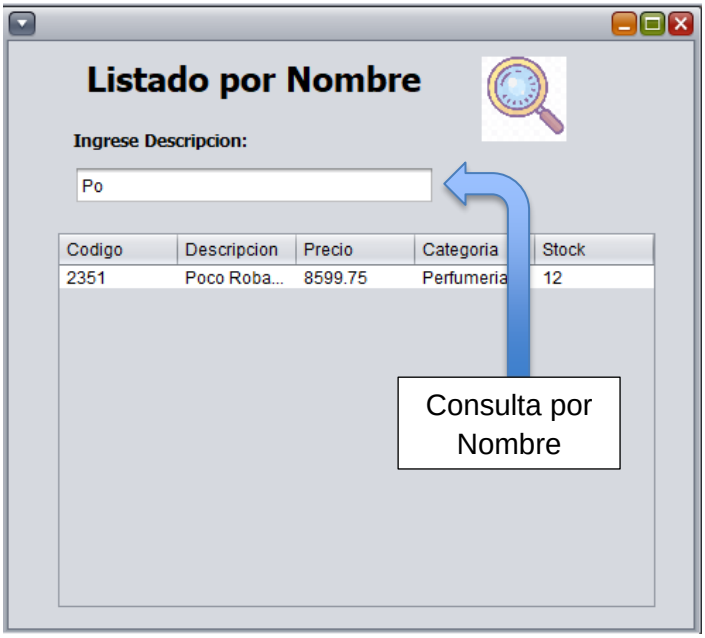
Note que el botón de Guardar se habilita (setEnabled) según se lo permita, luego de oprimir Nuevo; o bien Actualizar, Buscar, y Eliminar con el evento *MouseClicked* en la tabla.



Controle el formato de ingreso de números o letras. Use los *ShowMessageDialog* para comunicar lo validado.

LABORATORIO 1






```
//El método regresa true si todos los campos estan llenos
//false si alguno esta vacio
public boolean validarCamposVacios(JPanel jPanel){...10 lines }
public void vaciarCampos(JPanel jPanel){
    JComboBox combo = null;
    for (Component c : jPanel.getComponents()) {
       if(c instanceof JTextField){
          JTextField caja=(JTextField)c;
          caja.setText("");
       }
    }
    for(int i=0; i<jPanel.getComponents().length;i++){
       if(jPanel.getComponents()[i]instanceof JTextField){
          JTextField caja=(JTextField)jPanel.getComponents()[i];
          caja.setText("");
       }
       if(jPanel.getComponents()[i]instanceof JComboBox){
          combo =(JComboBox)jPanel.getComponents()[i];
          combo.setSelectedIndex(-1);
       }
    }
}
```

Validar campos con datos o vacíos, dentro de un mismo panel.

Las celdas de cualquier tabla no deben ser editables. No debe haber botones.

Sugerencia: Valerse de los eventos *keyReleased, keyPressed, MouseClicked, o keyTiped* para blanquear la tabla, borrar y redibujarla ante el evento de una tecla o mouse presionado. Puede usar *ItemStateChanged* en caso de los **`JComboBox`**.