TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE OAXACA

TÓPICOS AVANZADOS DE PROGRAMACIÓN

PRACTICA 2. DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN CON GUI Y EVENTOS DIVERSOS.

PRESENTA ESTUDIANTE DE LA CARRERA INGENIERÍA EN SISTEMAS COMPUTACIONALES:

ZARATE CARREÑO JOSÉ VALENTÍN

DOCENTE: HERNANDEZ ABREGO ANAYANSI

GRUPO: 4SA **HORA: 9:00- 10:00**

Oaxaca de Juárez, Oax, 26 de Febrero de 2020.

COMPETENCIA A DESARROLLAR

Desarrolla programas para interactuar con el usuario de una manera amigable, utilizando GUI (Interfaz Gráfica de Usuario) manipuladas a través de eventos.

INTRODUCCIÓN

En una aplicación con elementos de tipo GUI se tienen que manejar adecuadamente sus propiedades que le dan una cierta presentación y funcionalidad. Aquí se destacan la manera de adecuar las propiedades de los componentes GUI para cumplir con la vista y funcionalidad requerida en una aplicación que se espera sea de utilidad a los estudiantes, la cual busca establecer una relación de continuidad funcional con la primera práctica. Esta práctica tiene como producto una aplicación que trata sobre tipos de alimentos y su consumo. En su diseño se destaca la forma de organizar la funcionalidad en tres grupos de paneles mediante un contenedor "JTabbedPane" manipulando sus elementos a través del manejo de varios tipos de eventos que llegan a generar los diversos componentes GUI involucrados. También se hace uso de los tipos básicos de administradores de presentación y de un manejo de elementos de índole gráfica (Graphics) que combina la forma de crear imágenes de tipo rectángulos con componentes GUI con una distribución de los elementos hecha a la medida de estos.

CORRELACIÓN CON LOS TEMAS Y APLICACIÓN EN EL CONTEXTO

Se vinculan los cuatro temas del curso, se abarca el primer tema con el diseño de la GUI y su implementación se relaciona con otros temas sobre los tipos de eventos y su manejo con elementos GUI y gráficos usados en la aplicación propuesta. El contexto a donde se aplica es dentro del ámbito de java usando los elementos de GUI contenidos en el paquete swing y los eventos contenidos en el paquete awt de java, así como elementos que producen gráficos a fin de introducirse en aspectos de graficación.

MATERIAL Y EQUIPO NECESARIO

- Acceso al api de java (internet requerido)
- Equipo de cómputo: Laptop o PC
- Software: cualquier “IDE” de java, versión del jdk 1.8.

METODOLOGÍA

Se parte de la descripción de la aplicación a implementar dividido en su interfaz y la funcionalidad que se espera de ésta. En la descripción se presenta el diseño de la GUI que comprende su distribución y los elementos a usar. En la funcionalidad se incluyen los eventos que se utilizarán para llevarla a cabo.

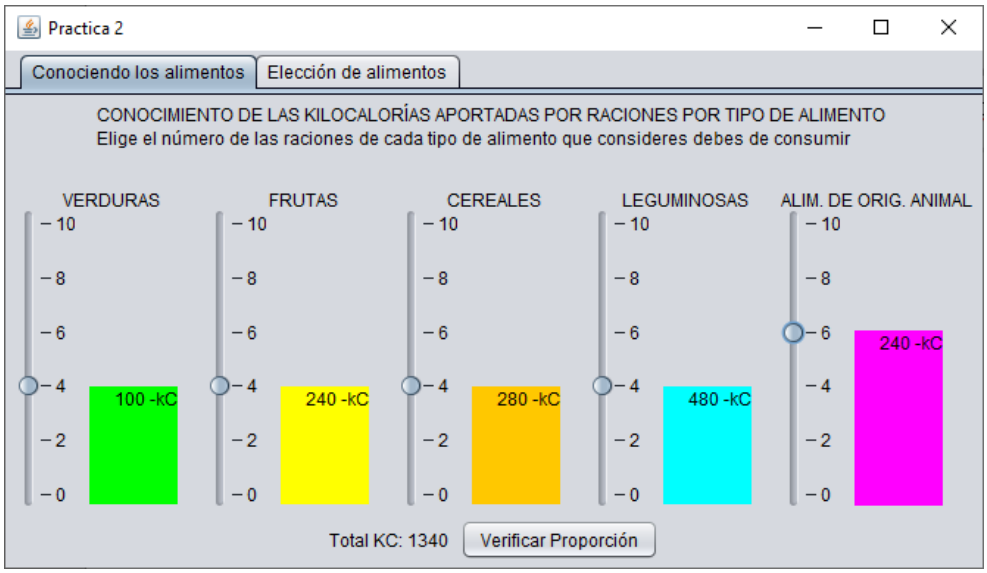
DESCRIPCIÓN

La aplicación a desarrollar trata sobre el manejo de información sobre los tipos de alimentos y su consumo de acuerdo a recomendaciones del IMSS. La cual se divide en dos partes, en la primera se pretende detectar el grado de conocimiento que tiene un usuario sobre el número kilocalorías de acuerdo al número de raciones que aportan los tipos de alimentos que éste cree deben consumirse a fin de orientarlo para la segunda parte que es donde se seleccionan los alimentos a consumir en diferentes tiempos (desayuno, comida y cena). A continuación se muestra el diseño de la GUI donde se parecían los elementos utilizados. i) Conocimiento de las kilocalorías que aportan los tipos de alimentos a consumir. Se tiene que ubicar en un tipo de alimento y desplazar el cursor para seleccionar el número de raciones que el usuario pretenda consumir, al hacerlo se debe mostrar una barra de un color determinado indicando en la su parte superior las kilocalorías (KC) que aporta el número de raciones elegido. En la parte inferior se debe mostrar el total de kilocalorías correspondiente a la suma de todos los grupos, actualizándose cada vez que se cambie un valor. Aquí el número elementos deslizadores (Slider’s) debe ser de acuerdo al número de tipo de alimentos a considerar, en este caso son cinco pero puede ser menos o más, cada vez que uno se desplace, su color

correspondiente deben variar en altura y mostrar el valor seleccionado centrado en la parte superior.

DESARROLLO DE LA PRÁCTICA

1. Creación de la clase base y su estructura
   a) Crear un proyecto simple de java, agrega un paquete, puedes llamarlo 'alimentos', dentro de él crea una clase de java, llámala ManejoAlimentos y herédala de JFrame.
   b) Crear el constructor de la clase ManejoAlimentos.
   c) Crear una clase donde pruebes la interface, especificando su dimensión y posición.
2. Diseño y creación de los elementos del panel que regresa el método panelConociendo()
   a) Declaración de atributos comunes usados a nivel de clase (puedes cambiar los colores):
   b) Diseño propuesto para el panel:



3. Creación de la funcionalidad del panel "Conociendo los alimentos"
   a) Se agrega el administrador de eventos de tipo de acción (ActionListener) a la clase implementado por la clase "ManejoAlimentos", se agrega el método necesario que implementa la acción: @Override public void actionPerformed(ActionEvent ae){}.

b) Se registra el botón “verificar” el administrador del evento acción.

c) Se agrega el administrador de eventos de tipo de ‘cambio de estado’ (ChangeListener) a la clase, haz que sea implementado por la clase “ManejoAlimentos”, agrega al método necesario que lo implementa : @Override public void stateChanged(ChangeEvent ce) {}

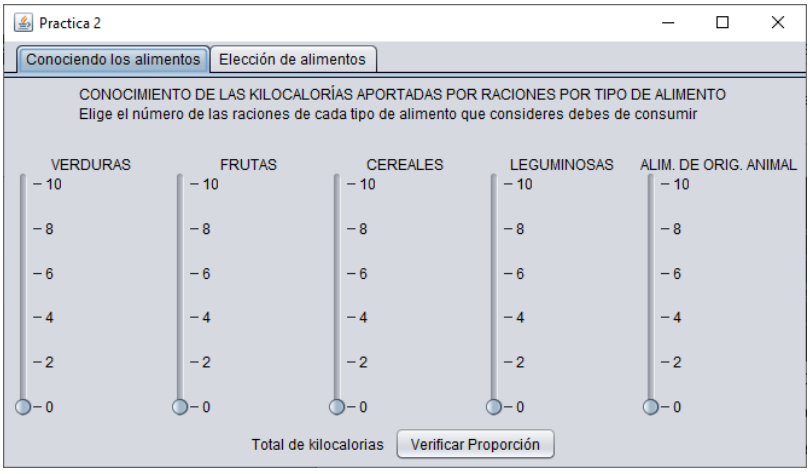
```
@Override
public void stateChanged(ChangeEvent ce) {
    int totalKc = 0;

    for (int ta = 0; ta < COLORES.length; ta++) {
        int base = (racTipoAlim[ta].getHeight() - racTipoAlim[ta].getValue() *
                KCAL_TIPO_ALIM[ta] * racTipoAlim[ta].getHeight() / racTipoAlim[ta].getHeight());
        int y = 195;
        KcalTipoAlim[ta].setBounds(0, y - (racTipoAlim[ta].getValue() * 19), 60, racTipoAlim[ta].getValue() * 20);

        KcalTipoAlim[ta].setText(" " + racTipoAlim[ta].getValue() * KCAL_TIPO_ALIM[ta] + " -kC");
        totalKc += racTipoAlim[ta].getValue() * KCAL_TIPO_ALIM[ta];
    }
    Kilocalorias.setText("Total KC: " + totalKc + " ");
    this.repaint();
}
```

En este fragmento de código lo que hace es hacer el cálculo para cada tipo de alimento esto a partir del array en el que se asignan las calorías que contiene cada alimento y a partir de ahí hacer el cálculo para finalmente obtener una suma total de las calorías.

d) Se registra a cada elemento de tipo JSlider (racTipoAlim[ta]) el administrador de este tipo de evento.

4. Se prueba el funcionamiento del panel “Conociendo Alimentos”.

a) Ejecutamos la aplicación y se ve como a continuación.



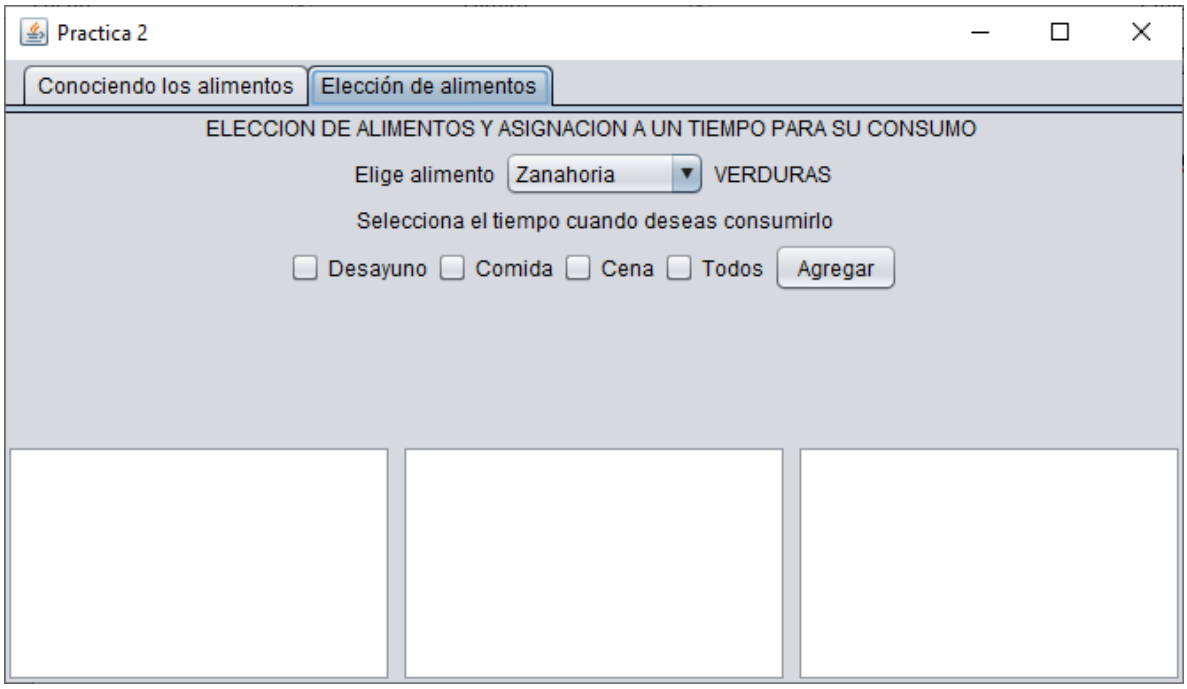
5. Creación de elementos de información necesarios a utilizar en panel “Elección de elementos”.
    a) Creación de una clase de alimentos para manejar su información
    b) En la clase ‘ManejoAlimento’ se declara un arreglo de la clase Alimentos, llámale “alimentos”.
    c) Crea un método que sirva para agregar valores al arreglo, aquí esta una parte, agrégale más datos.
    d) En el constructor de la clase “ManejoAlimento’, crea el arreglo “alimentos” y luego llama al método “cargarDatos()”.

6. Diseño de la presentación principal del panel “Elección de elementos” Este panel mencionado está integrado por tres partes, una de ellas ,la principal, comprende a las otras puesto que es la que abarca la presentación mostrada en la descripción y es formado en el método “panelAlimentos()”.
7. Creación del panel que integra a los cuadros de verificación que selecciona a cada tiempo de comida, esto se forma y regresa en el método “creaPanelTiempos()”
    a) La parte inicial define los elementos.
    b) Definición del tipo de distribución para ‘eleccion’ de tipo GridLayout(). Especifica su dimensión.
    c) Creación de los elementos de “cuadros de verificación”.
8. Creación de las listas que reciben los alimentos seleccionados.
    a) Creación de los paneles que lo conforman y de las listas con sus modelos.
    b) Agregación de los elementos de tipo etiqueta que guardan el total de alimentos agregados a cada lista.
    c) Agregar los elementos y regreso del panel formado.

9. Funcionalidad de los elementos del panel “Elección de elementos”. Esto se controla con dos tipos de eventos, de tipo ItemEvent y ActionEvent .

a) Administradores de los eventos. Para el caso de acción ya se definió anteriormente y para el caso de ItemEvent también hay que implementarlo en la clase mediante 'ItemListener' con su correspondiente método: @Override public void itemStateChanged(ItemEvent ie) {}
b) Registro del administrador 'ItemListener' en el elemento 'alimento' de tipo JComboBox`. Regístralo mediante alimento.addItemListener(this); después de haber creado a este elemento.
c) Implementación del método "itemStateChanged(ItemEvent ie)".

10. Prueba y afinación de la funcionalidad de "Elección de alimentos"
    a) Finalmente la interfaz queda de la siguiente manera.



CÓDIGO DE LA APLICACIÓN

Clase ManejoAlimentos:

```java
package alimentos;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class ManejoAlimentos extends JFrame implements
            ActionListener, ChangeListener, ItemListener {

    private final Color COLORES[] = {Color.GREEN, Color.yellow, Color.ORANGE, Color.CYAN, Color.magenta};
    private final String[] TIPO_ALIMENTO = {"VERDURAS", "FRUTAS", "CEREALES",
             "LEGUMINOSAS", "ALIM. DE ORIG. ANIMAL"};
    private final int[] KCAL_TIPO_ALIM = {25, 60, 70, 120, 40};
    private JSlider racTipoAlim[];
    private JLabel KcalTipoAlim[] = new JLabel[5];
    private JPanel pKalTipoAlim[];
    private JLabel Kilocalorias;
    private JButton verificar;

    // atributos para el panel Alimentos
    private final String TIEMPOSCOMIDA[] = {"Desayuno", "Comida", "Cena", "Todos"};
    private JCheckBox eleccionTiempo[];
    private DefaultListModel<Alimentos> modelosListas[];
    private JList tiempos[] = new JList[1];
    private JComboBox alimento;
    private JButton aceptar;
    private JLabel tipoAli;
    private JLabel nAlimentos;
```

```java
    private Alimentos alimentos[] = new Alimentos[22];
    private int[] numAlimTie;
    private JLabel numAlimTiem[];
    private JLabel itemsDesayuno = new JLabel(), itemsComida = new JLabel(), itemsCena = new JLabel();

    public ManejoAlimentos() {
        super("Practica 2");
        Container panel = getContentPane();
        JTabbedPane panelPrincipal = new JTabbedPane();
        panelPrincipal.addTab("Conociendo los alimentos", panelConociendo());
        panelPrincipal.addTab("Elección de alimentos", panelAlimentos());

        panel.add(panelPrincipal);
        this.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
```

```java
    private void cargarDatos() {
        alimentos = new Alimentos[22];

        alimentos[0] = new Alimentos("Zanahoria", 0);
        alimentos[1] = new Alimentos("Lechuga", 0);
        alimentos[2] = new Alimentos("Espinaca", 0);
        alimentos[3] = new Alimentos("Acelgas", 0);
        alimentos[4] = new Alimentos("Tomate", 0);
        alimentos[5] = new Alimentos("Platano", 1);
        alimentos[6] = new Alimentos("Naranja", 1);
        alimentos[7] = new Alimentos("Sandia", 1);
        alimentos[8] = new Alimentos("Pera", 1);
        alimentos[9] = new Alimentos("Pepino", 1);
        alimentos[10] = new Alimentos("Melón", 1);
        alimentos[11] = new Alimentos("Arroz", 2);
        alimentos[12] = new Alimentos("Tortilla", 2);
        alimentos[13] = new Alimentos("Jamón", 4);
        alimentos[14] = new Alimentos("Pan Blanco", 2);
        alimentos[15] = new Alimentos("Frijol", 2);
        alimentos[16] = new Alimentos("Muslo de Pollo", 4);
        alimentos[17] = new Alimentos("Sardina", 4);
        alimentos[18] = new Alimentos("Garbanzos", 2);
        alimentos[19] = new Alimentos("Habas", 2);
        alimentos[20] = new Alimentos("Lentejas", 0);
        alimentos[21] = new Alimentos("Pasta", 2);

    }
```

```java
private JPanel panelConociendo() {

    JPanel conocer = new JPanel();
    conocer.setLayout(new BorderLayout());
    JPanel panelSuperior = new JPanel();
    panelSuperior.setLayout(new BorderLayout());
    JPanel panelCentro = new JPanel();

    JPanel panelSur = new JPanel();
    JPanel panelTitulo = new JPanel();
    JPanel panelEncabezados = new JPanel();

    panelTitulo.setLayout(new GridLayout(3, 2));
    JLabel enc1 = new JLabel("CONOCIMIENTO DE LAS KILOCALORÍAS APORTADAS POR RACIONES POR TIPO DE ALIMENTO ");
    JLabel enc2 = new JLabel("Elige el número de las raciones de cada tipo de alimento que consideres debes de consumir");

    panelTitulo.add(enc1);
    panelTitulo.add(enc2);
    panelEncabezados.add(panelTitulo);

    panelSuperior.add(panelEncabezados, BorderLayout.CENTER);

    pKalTipoAlim = new JPanel[TIPO_ALIMENTO.length];
    JPanel pGpoTipoAlim = new JPanel();
    pGpoTipoAlim.setLayout(new GridLayout(0, 5));
    racTipoAlim = new JSlider[TIPO_ALIMENTO.length];
    verificar = new JButton("Verificar Proporción");
    Kilocalorias = new JLabel("Total de kilocalorias ", 0);
    panelSur.add(Kilocalorias);
    panelSur.add(verificar);
```

```
        for (int i = 0; i < TIPO_ALIMENTO.length; i++) {
            JPanel pdatos = new JPanel();
            pdatos.setLayout(new BorderLayout());
            JLabel tit = new JLabel(TIPO_ALIMENTO[i]);
            tit.setHorizontalAlignment(SwingConstants.CENTER);
            pKalTipoAlim[i] = new JPanel();
            racTipoAlim[i] = new JSlider();
            racTipoAlim[i].setOrientation(SwingConstants.VERTICAL);//
            racTipoAlim[i].setPaintLabels(true);
            racTipoAlim[i].setPaintTicks(true);
            racTipoAlim[i].setMinimum(0);
            racTipoAlim[i].setMaximum(10);
            racTipoAlim[i].setValue(0);
            racTipoAlim[i].setMajorTickSpacing(2);
            racTipoAlim[i].setMinorTickSpacing(0);
            racTipoAlim[i].addChangeListener(this);

            KcalTipoAlim[i] = new JLabel(KCAL_TIPO_ALIM[i] + "");
            KcalTipoAlim[i].setBackground(COLORES[i]);
            KcalTipoAlim[i].setOpaque(true);
            KcalTipoAlim[i].setVerticalAlignment(SwingConstants.TOP);
            KcalTipoAlim[i].setHorizontalAlignment(SwingConstants.RIGHT);
            pKalTipoAlim[i].setLayout(null);
            pKalTipoAlim[i].setPreferredSize(new Dimension(80, 80));
            pKalTipoAlim[i].add(KcalTipoAlim[i]);

            pdatos.add(tit, BorderLayout.NORTH);
            pdatos.add(racTipoAlim[i], BorderLayout.WEST);
            pdatos.add(pKalTipoAlim[i], BorderLayout.EAST);
            pGpoTipoAlim.add(pdatos);
        }
```

```
        verificar.addActionListener(this);
        panelCentro.add(pGpoTipoAlim);//bien
        conocer.add(panelSuperior, BorderLayout.NORTH);//bien
        conocer.add(panelCentro, BorderLayout.CENTER);//bien
        conocer.add(panelSur, BorderLayout.SOUTH);//bien
        return conocer;
    }

```

```java
public JPanel panelAlimentos() {
    JPanel pAlimentos = new JPanel();
    pAlimentos.setLayout(new BorderLayout());
    JPanel panelCentro = new JPanel();//Para la seleccion de alimentos y cuadros de verifcacion
    JPanel panelCheck = new JPanel();//Para cuadros de verificacion de tiempos de comidas
    panelCheck.setLayout(new GridLayout(1, 0));
    panelCentro.setLayout(new BorderLayout());
    JPanel panelSeleccion = new JPanel();//Para la selccion de los alimentos
    JLabel titulo = new JLabel("ELECCION DE ALIMENTOS Y ASIGNACION A UN TIEMPO PARA SU CONSUMO ");
    titulo.setHorizontalAlignment(JLabel.CENTER);
    JLabel tAlimento = new JLabel("Elige alimento");
    aceptar = new JButton("Aceptar");
    tipoAli = new JLabel("");
    cargarDatos();
    alimento = new JComboBox(alimentos);
    tipoAli.setText(TIPO_ALIMENTO[0]);// se rnuestra el primer°
    alimento.addItemListener(this);
    JPanel panelTiempos = new JPanel();
    panelTiempos.add(new JPanel());
    panelSeleccion.add(tAlimento);
    panelSeleccion.add(alimento);
    panelSeleccion.add(tipoAli);
    panelCentro.add(panelSeleccion, BorderLayout.NORTH);
    panelCentro.add(creaPanelTiempos(), BorderLayout.CENTER);
    pAlimentos.add(titulo, BorderLayout.NORTH);
    pAlimentos.add(panelCentro, BorderLayout.CENTER);
    panelCentro.add(creaListas(), BorderLayout.SOUTH);
    return pAlimentos;
}
```

```java
public JPanel creaPanelTiempos() {
    JPanel tiemposAlimento = new JPanel();
    tiemposAlimento.setLayout(new BorderLayout());
    JLabel indicacion = new JLabel("Selecciona el tiempo cuando deseas consumirlo");
    indicacion.setHorizontalAlignment(JLabel.CENTER);
    tiemposAlimento.add(indicacion, BorderLayout.NORTH);
    JPanel eleccion = new JPanel();
    aceptar = new JButton("Agregar");
    eleccionTiempo = new JCheckBox[TIEMPOSCOMIDA.length];
    numAlimTie = new int[10];
    eleccion.setLayout(new FlowLayout());

    for (int i = 0; i < eleccionTiempo.length; i++) {
        eleccionTiempo[i] = new JCheckBox(TIEMPOSCOMIDA[i]);
        eleccion.add(eleccionTiempo[i]);
        eleccionTiempo[i].addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                for (int i = 0; i < eleccionTiempo.length; i++) {
                    if (e.getSource() == eleccionTiempo[3]) {
                        if (eleccionTiempo[3].isSelected()) {
                            eleccionTiempo[i].setSelected(true);
                        } else {
                            eleccionTiempo[i].setSelected(false);
                        }
                    }

                }
            }
        });
    }
```

```
        aceptar.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                JButton evento = (JButton) event.getSource();
                if (evento == aceptar) {

                    if (eleccionTiempo[0].isSelected()) {
                        if (!modelosListas[0].contains(((Alimentos) alimento.getSelectedItem()))) {
                            modelosListas[0].addElement(((Alimentos) alimento.getSelectedItem()));

                            numAlimTiem[0].setText("Num. Alim :" + modelosListas[0].getSize());
                        }
                    }
                    if (eleccionTiempo[1].isSelected()) {
                        if (!modelosListas[1].contains(((Alimentos) alimento.getSelectedItem()))) {
                            modelosListas[1].addElement(((Alimentos) alimento.getSelectedItem()));

                            numAlimTiem[1].setText("Num. Alim : " + modelosListas[1].getSize());

                        }
                    }
                    if (eleccionTiempo[2].isSelected()) {
                        if (!modelosListas[2].contains(((Alimentos) alimento.getSelectedItem()))) {
                            modelosListas[2].addElement(((Alimentos) alimento.getSelectedItem()));

                            numAlimTiem[2].setText("Num. Alim : " + modelosListas[2].getSize());
                        }
                    }
                }
            }
        });
```

```
        eleccion.add(aceptar);

        tiemposAlimento.add(indicacion, BorderLayout.NORTH);
        tiemposAlimento.add(eleccion, BorderLayout.CENTER);

        return tiemposAlimento;
    }

    private JPanel creaListas() {
        JPanel psur = new JPanel();
        psur.setLayout(new GridLayout(1, 3));
        JPanel pListas = new JPanel();
        JPanel listas = new JPanel();
        JPanel numAlim = new JPanel();
        tiempos = new JList[TIEMPOSCOMIDA.length - 1];
        numAlimTiem = new JLabel[TIEMPOSCOMIDA.length - 1];
        modelosListas = new DefaultListModel[TIEMPOSCOMIDA.length - 1];
        GridLayout dList = new GridLayout(0, TIEMPOSCOMIDA.length - 1, 5, 0);
        pListas.setLayout(new BorderLayout());
        listas.setLayout(dList);
        numAlim.setLayout(dList);
        //muestra los Jlist
        for (int t = 0; t < tiempos.length; t++) {
            modelosListas[t] = new DefaultListModel();
            tiempos[t] = new JList(modelosListas[t]);
            listas.add(new JScrollPane(tiempos[t]));
        }

        for (int i = 0; i < tiempos.length; i++) {
            numAlimTiem[i] = new JLabel();
            numAlim.add(numAlimTiem[i]);
        }
```

```
        psur.add(numAlim);

        pListas.add(listas, BorderLayout.NORTH);
        pListas.add(psur, BorderLayout.SOUTH);

        return pListas;

    }

    @Override
    public void actionPerformed(ActionEvent ae) {
        JButton comprobar = (JButton) ae.getSource();
        if (verificar == comprobar) {
            if ((racTipoAlim[0].getValue() < 6 || racTipoAlim[1].getValue() < 6)
                && racTipoAlim[2].getValue() < 4 && racTipoAlim[3].getValue() < 5 && racTipoAlim[4].getValue() < 5) {
    JOptionPane.showMessageDialog(null, "Necesitas comer mas:\n Frutas\n Berduras\n Leguminosas\n cereales\nAlimentos de Origen Animal  "
                );
            } else if ((racTipoAlim[0].getValue() == 5) || (racTipoAlim[1].getValue() == 5)
                && (racTipoAlim[2].getValue() == 3) && (racTipoAlim[3].getValue() == 4) && (racTipoAlim[4].getValue() == 4)) {
                JOptionPane.showMessageDialog(null, "Estas comiendo Bien "
                );
            } else if (racTipoAlim[0].getValue() > 6 || racTipoAlim[1].getValue() > 6
                    && racTipoAlim[2].getValue() > 4 && racTipoAlim[3].getValue() > 5 && racTipoAlim[4].getValue() > 5) {
    JOptionPane.showMessageDialog(null, "Estas comiendo de mas:\nLeguminosas\n cereales\nAlimentos de Origen Animal  \nVerduras y Frutas BIEN"
                );
            }

        }
    }
```

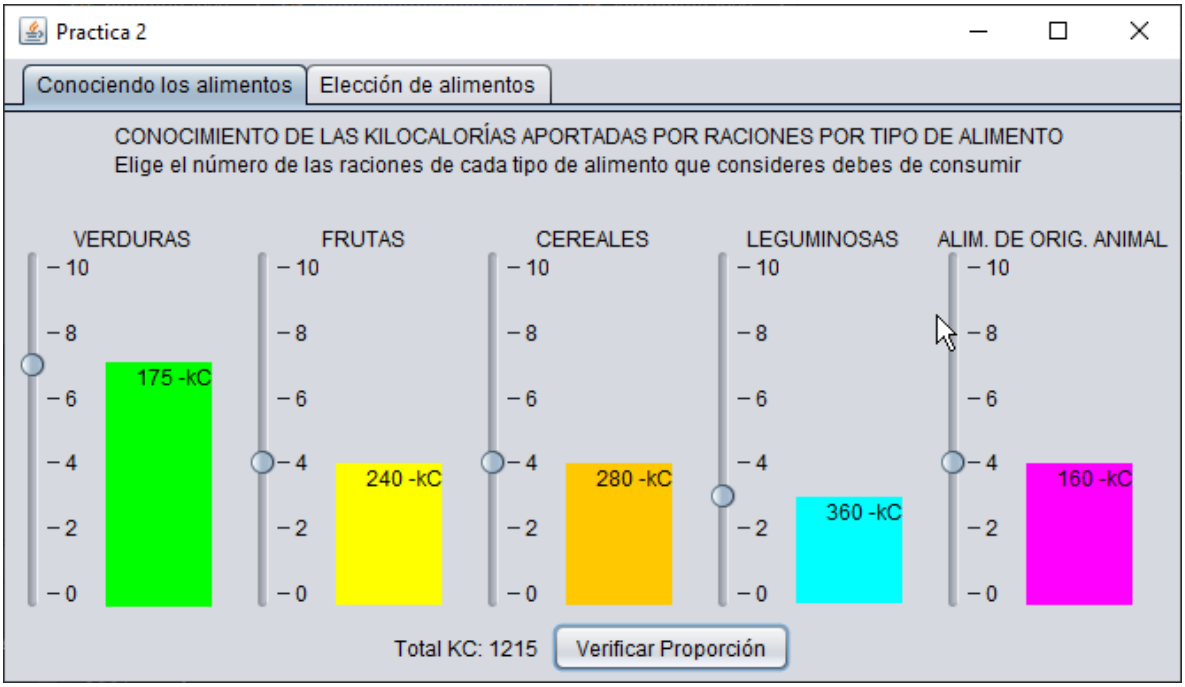
```
    @Override
    public void stateChanged(ChangeEvent ce) {
        int totalKc = 0;

        for (int ta = 0; ta < COLORES.length; ta++) {
            int base = (racTipoAlim[ta].getHeight() - racTipoAlim[ta].getValue() *
                    KCAL_TIPO_ALIM[ta] * racTipoAlim[ta].getHeight() / racTipoAlim[ta].getHeight());
            int y = 195;
            KcalTipoAlim[ta].setBounds(0, y - (racTipoAlim[ta].getValue() * 19), 60, racTipoAlim[ta].getValue() * 20);

            KcalTipoAlim[ta].setText(" " + racTipoAlim[ta].getValue() * KCAL_TIPO_ALIM[ta] + " -kC");
            totalKc += racTipoAlim[ta].getValue() * KCAL_TIPO_ALIM[ta];
        }
        Kilocalorias.setText("Total KC: " + totalKc + " ");
        this.repaint();
    }

    @Override
    public void itemStateChanged(ItemEvent ie) {
        JComboBox aliSel = (JComboBox) ie.getSource();
        int t = ((Alimentos) aliSel.getSelectedItem()).GetTipo();

        tipoAli.setText(TIPO_ALIMENTO[t]);

    }
```

```java
package alimentos;

public class Alimentos {

    private String alimento;
    private int tipo;

    public Alimentos(String alimento, int tipo){
    this.alimento=alimento;
    this.tipo=tipo;

    }
    public int GetTipo(){

    return tipo;
    }
    public String GetAlimento(){

    return alimento;
    }

    @Override
    public String toString(){
    return alimento;
    }
    public boolean equals(Alimentos ali){
    return alimento.equals(ali.alimento)&& tipo== ali.tipo;
    }
}
```

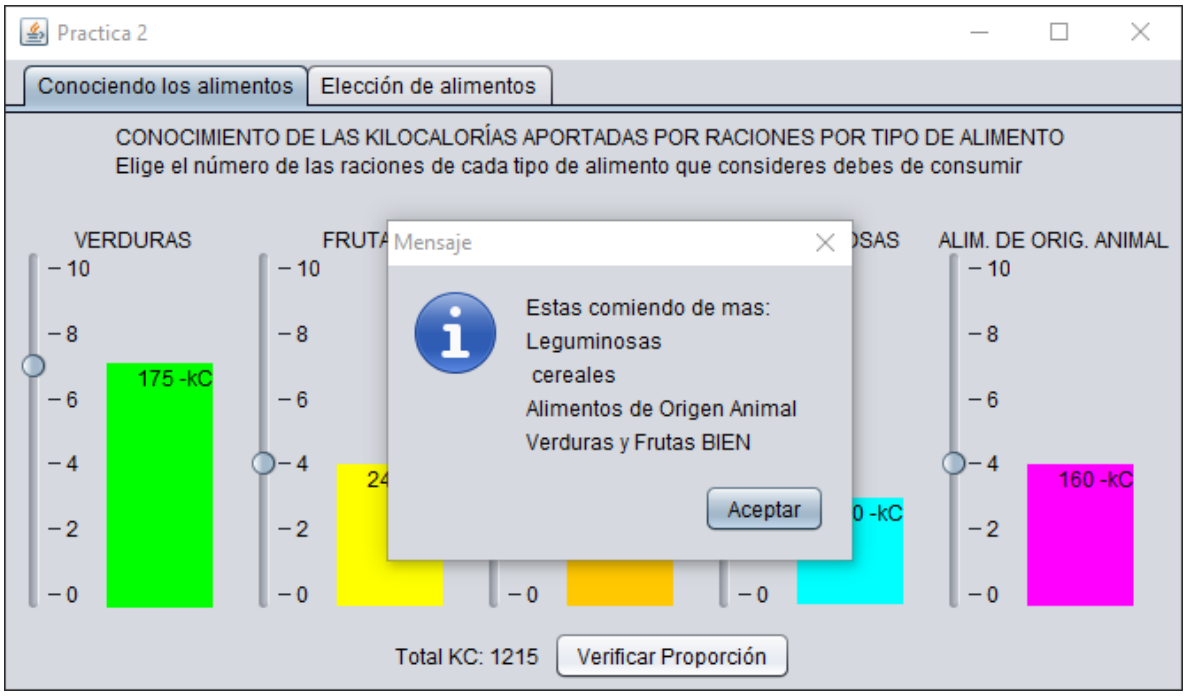
```java
package alimentos;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.UIManager;
public class pruebas {
    Run | Debug
    public static void main(String[] args) {
    try {
            for (UIManager.LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(ManejoAlimentos.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            Logger.getLogger(ManejoAlimentos.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            Logger.getLogger(ManejoAlimentos.class.getName()).log(Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            Logger.getLogger(ManejoAlimentos.class.getName()).log(Level.SEVERE, null, ex);
        }
       ManejoAlimentos alimentos = new ManejoAlimentos();
        alimentos.setSize(680, 390);
        alimentos.setVisible(true);

    }
}
```

RESULTADOS DE LOS CASOS DE PRUEBA

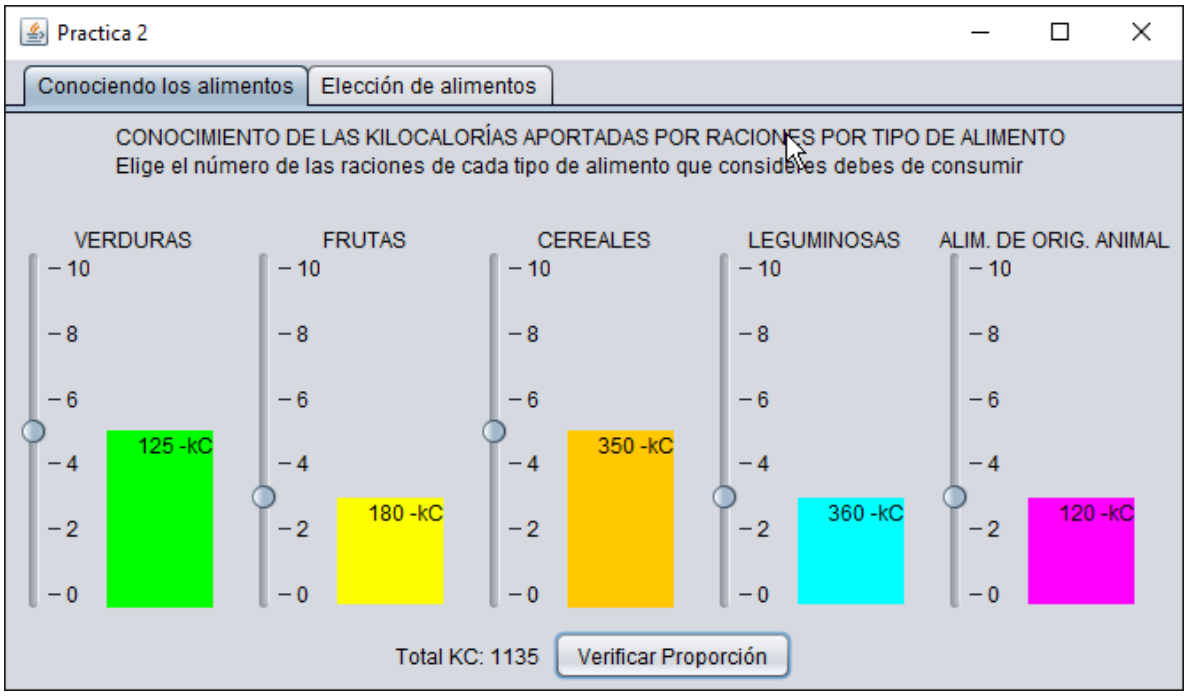
1. Prueba para el panel 1:

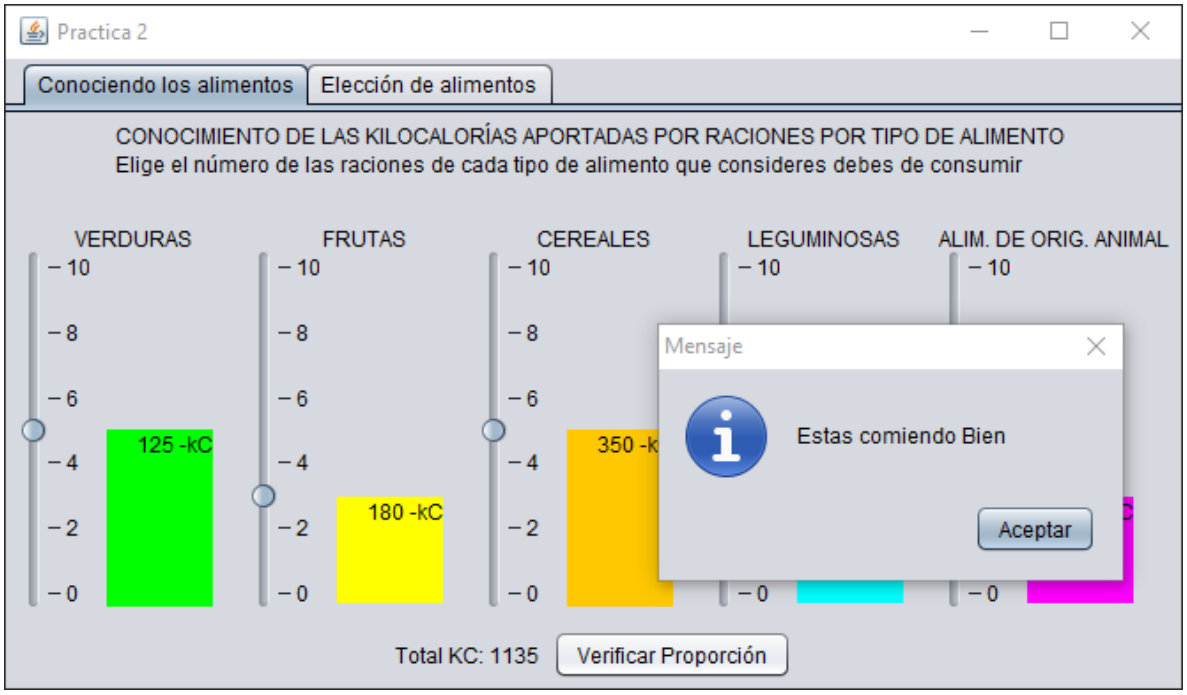


a) Cuando se da clic en el botón verificar, nos arroja el siguiente mensaje según a la cantidad de calorías que consumes de acuerdo al cálculo realizado para cada tipo de alimento.

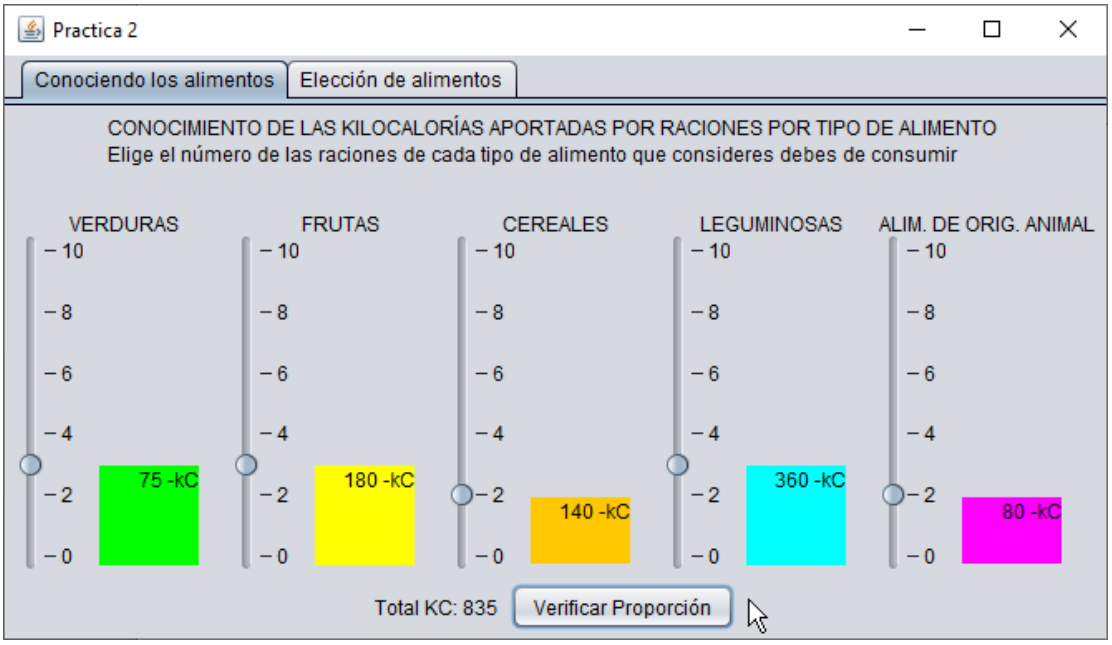


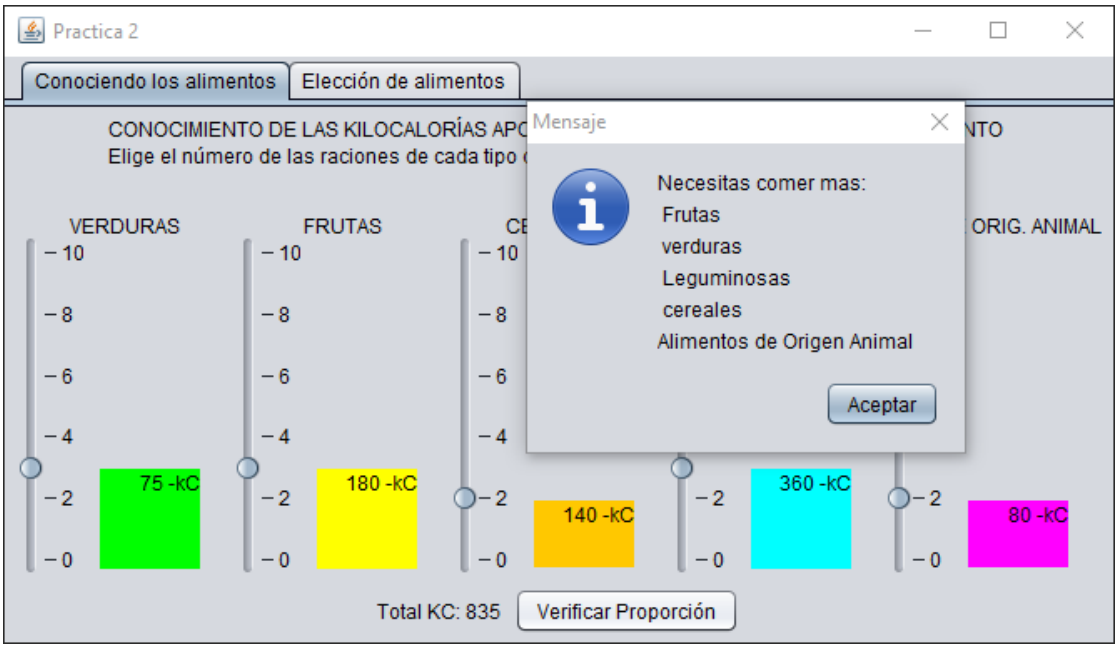
2. Para otro de los casos como se muestra a continuación:





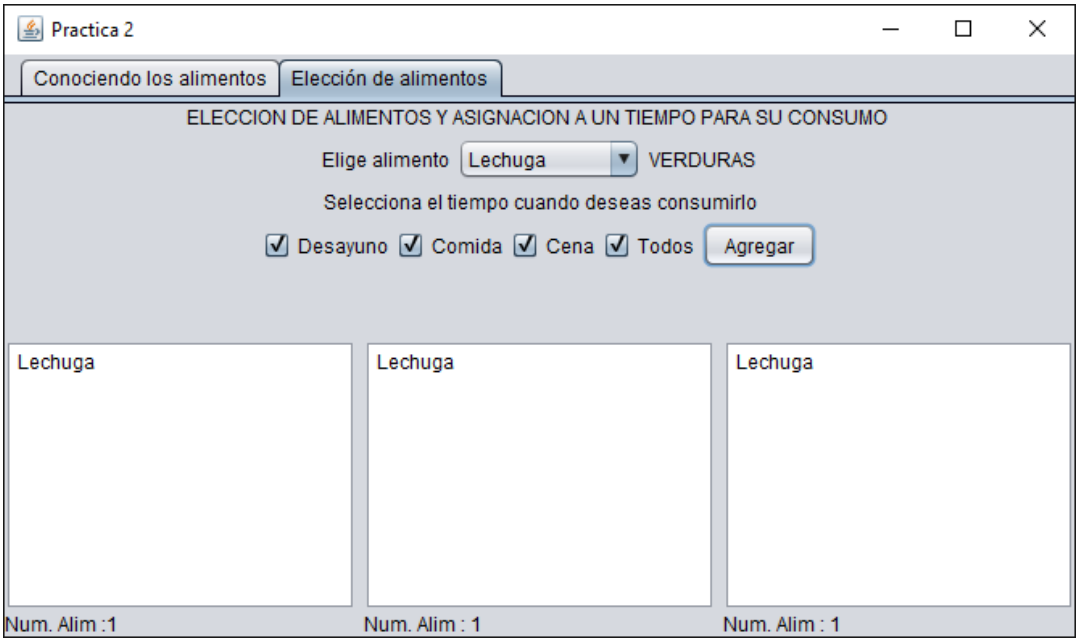
3. Para el tercer caso nos saldrá los siguiente:





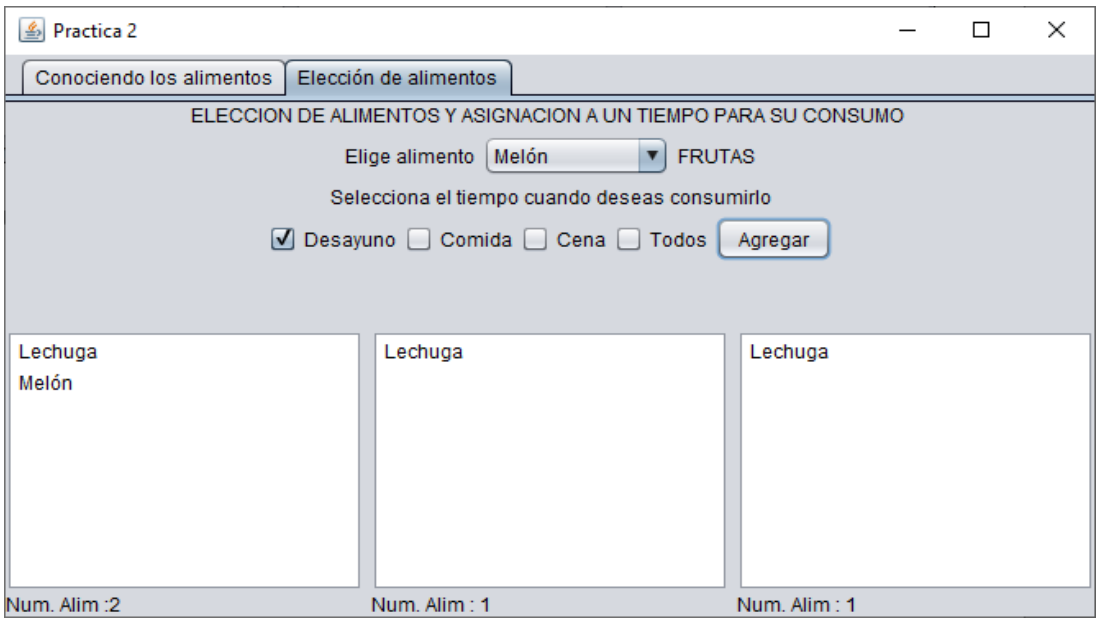
1. Prueba para el panel 2:

   a) Se agregan los alimentos a las listas como se muestra a continuación: (en este caso se activa la el combobox Todos).

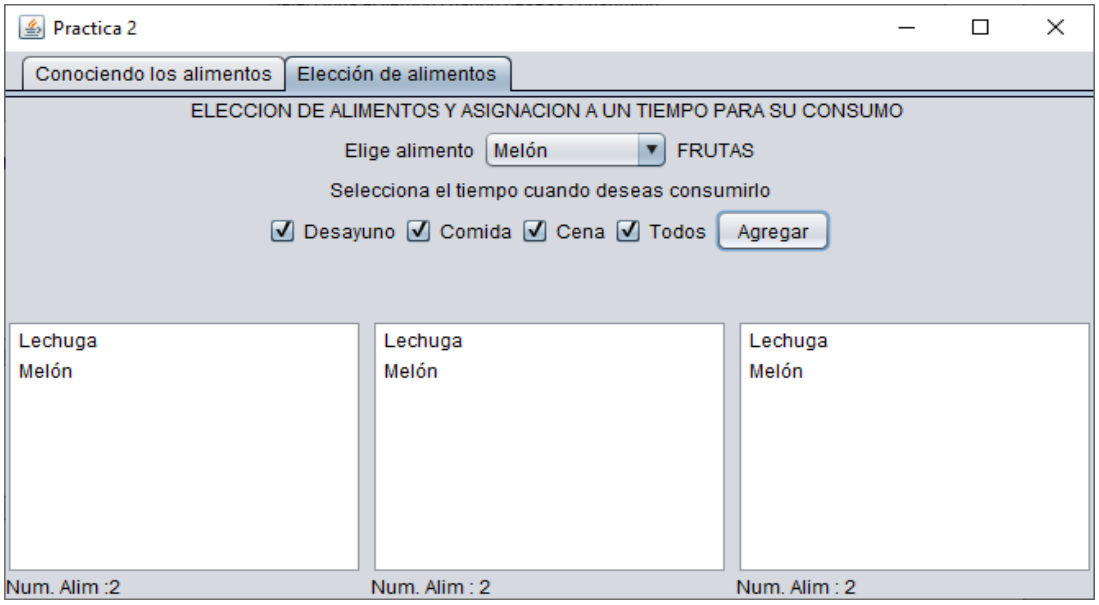


   Nota: para el caso de agregar el mismo alimento otra vez no se podrá.

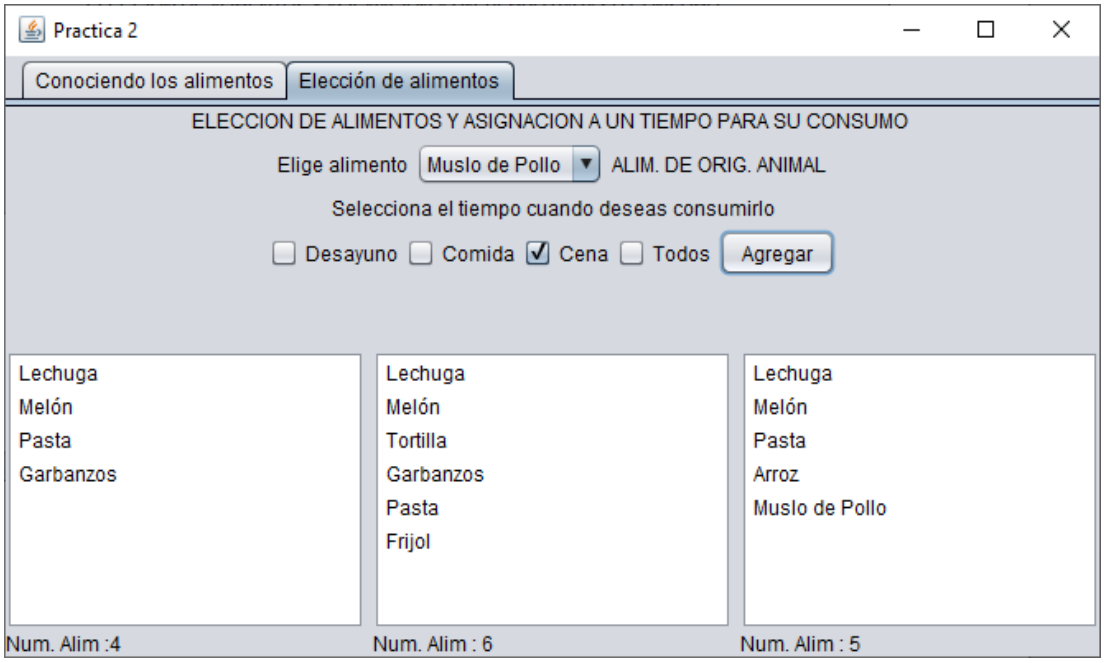
   b) Para el caso de agregar un alimento para solo una lista:



c) Para el caso de agregar ese alimento para todos (Tomando en cuenta que no se agregará a la primer lista porque ya estaba agregado):



d) Una prueba general del programa podría quedar de la siguiente manera:



Los elementos de cada lista se cuentan para cada una de las listas, así mismo el alimento sólo se puede agregar una vez en cada una de las listas, esto nos permite no repetir n veces el alimento en una misma lista.

Conclusiones sobre los principales aspectos a considerar en una aplicación GUI

En conclusión el conocimiento de estos puntos clave, nos permitirán enfocarnos mejor al estudio de la materia. También Las Interfaces de usuario, como vínculo de inmersión del hombre en el entorno de trabajo tecnológico actual, realzan su importancia en el desarrollo de nuevos productos, más eficaces, eficientes e interactivos, que es lo que el mercado demanda. Los puntos, cómo los históricos y evolutivos, deben ser abordados de manera más investigativa, recordemos que "conocer el pasado nos proyecta al futuro". Otras puntualizaciones de clasificación obligarán a que investiguemos y propongamos, nuevas distribuciones clasificatorias, útiles a futuro en una carrera de desarrollo de software.