Practica 2. Diseño e implementación de una aplicación con GUI y eventos diversos.

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

Equipo de cómputo: Laptop o PC

Software: Puedes utilizar cualquier IDE de java con versión del jdk 1.8.0

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.

En la fase de desarrollo se indican los pasos de la práctica, partiendo de lo más sencillo para que los estudiantes ejerciten y refuercen sus conocimientos, haciendo propuestas para que según las disponibilidad de tiempo puedan profundizar en algunas áreas de programación, puesto que la intensión de la práctica es que ésta pueda llagar a escalar a un proyecto. Las partes en las que el estudiante tiene que completar el código se indican con el símbolo '?':

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.

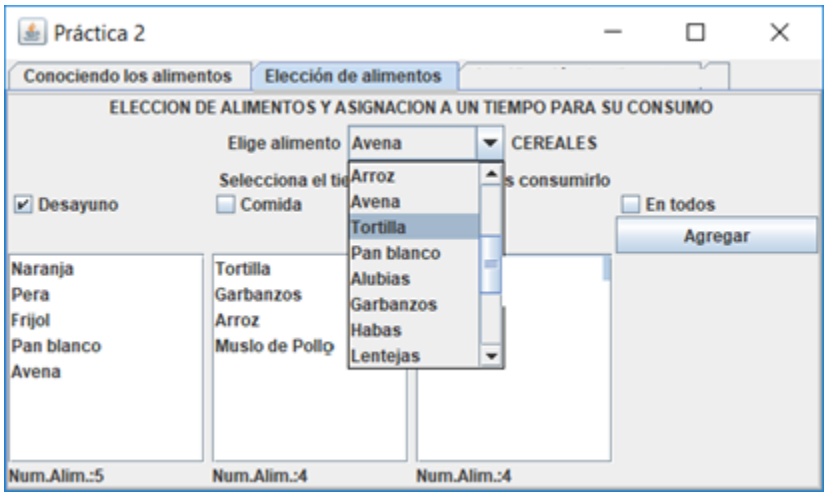
Aquí el número elementos deslizadores (Slider's) debe ser de acuerdo al número de tipo de alimentos a considerar, en este caso son cinco pero puede ser menos o más, cada vez que uno se desplace, su color correspondiente deben variar en altura y mostrar el valor seleccionado centrado en la parte superior.

Al hacer clic en verificar proporción se debe mostrar la información correspondiente que haga rectificar en caso de que el usuario no haya un número de raciones que estén de acuerdo con la información siguiente:

Vegetales y frutas deben ser de demanda libre, mientras más alto sean las reciones es mejor.

Cereales y Leguminosas deben de tener un número de raciones de doso más.
Los alimentos de origen animal deben estar en un rango de uno a cuatro
Las kilocalorías que aporta cada tipo de alimento por una ración son: verduras 25 KC ; frutas 60 KC ,Legumbres 70 KC; leguminosas 120 KC, y alimentos de origen animal 40 KC. Esto es de acuerdo al IMSS (http://www.imss.gob.mx/sites/all/statics/salud/guia-alimentos.pdf)

ii) Elección de los alimentos a consumir. Se parte de un conjunto de alimentos que se llegan a desplegar donde el elemento seleccionado muestra a su derecha el tipo de alimento a que pertenece. Luego se elige el o los tiempos en que el usuario desea consumir un elemento seleccionado, cuando se activa el cuadro de verificación '[ ] En todos' (los tiempos), los demás cuadros de verificación se activan como seleccionados y cuando uno de ellos se des-selecciona este cuadro referido también se deselecciona o bien cuando se cada tiempo es seleccionado en forma individual se activa como seleccionado '[ ] En todos'



Para agregar un elemento seleccionado a la lista de tiempo (desayuno, comida , cena o cualquier combinación de ellos) se pulsa la botón agregar y el alimento debe ser agregado en la lista correspondiente, debiéndose reflejar el número de los alimentos que cada lista tiene agregados.

Un alimento a agregado a una lista de algún tiempo no debe ser agregado más de una vez

Desarrollo de la práctica:

Sigue los pasos para la implementación, recordando anotar las repuestas a las preguntas planteadas en tu bitácora de seguimiento y qué cuando encuentres el símbolo te corresponde completar el código o instrucción indicado.

Dentro de tu 'IDE' de Java:

1. Creación de la clase base y su estructura
   a) Crea 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) Crea el constructor

```
public ManejoAlimentos() {
```

```
    super("Práctica 2");
    Container panel = getContentPane();
    JTabbedPane panelPrincipal = new JTabbedPane();
    panelPrincipal.addTab("Conociendo los alimentos ",panelConociendo());
    panelPrincipal.addTab("Elección de alimentos", panelAlimentos());
   // panelPrincipal.addTab("Clasificación de alimentos", grafico);
    panel.add(panelPrincipal);
    this.setVisible(true);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
```

"?" Crea los métodos usados en panelPrincipal.addTab("",…), que regresan como valor un JPanel, excepto el de grafico que por lo pronto se deja comentado

c) Crea un clase donde pruebes la interface, especificando su diemensión 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):

```
private final Color COLORES[] = {Color.GREEN, Color.YELLOW, Color.ORANGE, Color.WHITE, 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}; //kilocalrias por ración por tipo de alimento
    private JSlider racTipoAlim[];//Fija el numero de raciones por tipo de alimento
    private JLabel  kCalTipoAlim[];//Guarda las kilocalorais por tipo de alimento
    private JPanel  pKcalTipoAlim[];//Paneles para guardar las barras a desplegar
    private JLabel  kilocalorias;// Para mostrar el valor de las kilocalorias
```

b) Diseño propuesto para el panel

c) Su definición:

```
private JPanel panelConociendo()
  {/ panel de conociendo los alimentos
   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(0,1));
JLabel enc1= new JLabel("CONOCIMIENTO DE LAS KILOCALORIAS APORTADAS POR RACIONES POR TIPO DE ALIMENTO");
JLabel enc2= new JLabel("Elige el número las raciones de cada tipo de alimento que consideres debes de consumir");
```

"?" Te corresponde agregar los encabezados a '**panelSuperior**'

d) Siguiendo con la creación de elementos del panel

```
pKcalTipoAlim = new JPanel[TIPO_ALIMENTO.length];//Atributo
JPanel pGpoTipoAlim = new JPanel(); // sirve para agrupar a los JSlider y las JLabel (barras)
pGpoTipoAlim.setLayout(?); // Te toca especificar el tipo de distribución(administrador)
racTipoAlim = new JSlider[TIPO_ALIMENTO.length];
kCalTipoAlim  = new JLabel[TIPO_ALIMENTO.length];
verificar = new JButton("Verifica proporciòn"); // Debes  declararlo como atributo
kilocalorias = new JLabel("Total de kilocalorias: 0");
panelSur.add(kilocalorias);
panelSur.add(verificar);
panelSur.add(reAlimentacion);
```

e) Agregando todos los elementos al grupo de paneles, sustituye el valor que le corresponda donde está el signo de ?

```
for(int ta=0; ta < TIPO_ALIMENTO.length ; ta++)
     { JPanel pDatos = new JPanel();
       pDatos.setLayout(new BorderLayout());
       JLabel tit= new JLabel(TIPO_ALIMENTO[ta]);
       tit.setHorizontalAlignment(SwingConstants.CENTER);
       pKcalTipoAlim[ta]= new JPanel();
       racTipoAlim[ta] = new  JSlider();
       racTipoAlim[ta].setOrientation(?);
        racTipoAlim[ta].setPaintLabels(?);
        racTipoAlim[ta].setPaintTicks(?);
        racTipoAlim[ta].setMinimum(?);
        racTipoAlim[ta].setMaximum(?);
        racTipoAlim[ta].setValue(?);
        racTipoAlim[ta].setMajorTickSpacing(?);
        racTipoAlim[ta].setMinorTickSpacing(?);
        kCalTipoAlim[ta] = new JLabel(?);
        kCalTipoAlim[ta].setBackground(?);
        kCalTipoAlim[ta].setOpaque(?);
        kCalTipoAlim[ta].setVerticalAlignment(?);
        kCalTipoAlim[ta].setHorizontalAlignment(?);
        pKcalTipoAlim[ta].setLayout(?);
        pKcalTipoAlim[ta].setPreferredSize(?,?);
        pKcalTipoAlim[ta].add(?);
        pDatos.add(tit,?);
        pDatos.add(racTipoAlim[ta],?);
        pDatos.add(pKcalTipoAlim[ta],?);
        pGpoTipoAlim.add(pDatos);
     }
```

f) Ahora falta agregar todos los elementos al panel '**conocer**' y retornar su valor, aquí se muestran dos instrucciones la segunda precisamente consiste en agregar lo que va en el sur, a ti te toca el panel centro y norte

```
panelCentro.add(pGpoTipoAlim);
conocer.add(panelSur,BorderLayout.SOUTH);
```

3. Creación de la funcionalidad del panel "***Conociendo los alimentos***"

a) Agrega el administrador de eventos de tipo de acción (ActionListener) a la clase, haz que sea implementado por la clase "ManejoAlimentos", agrega el método necesario que implementa la acción: @Override public void actionPerformed(ActionEvent ae){}

b) Dentro del método "actionPerformed(ActionEvent ae)" , agrega el código:

```
Object prod= ae.getSource(); // Productor del evento
if(prod==verificar)
 { String mensaje= // agrega el texto necesario, como se muestra en el apartado de especificación
  JOptionPane.showMessageDialog(this,mensaje);
  }
```

c) Regístrale al botón "verificar" el administrador del evento acción.

d) 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) {}

e) Dentro del método stateChanged(ChangeEvent ce) agrega el siguiente código

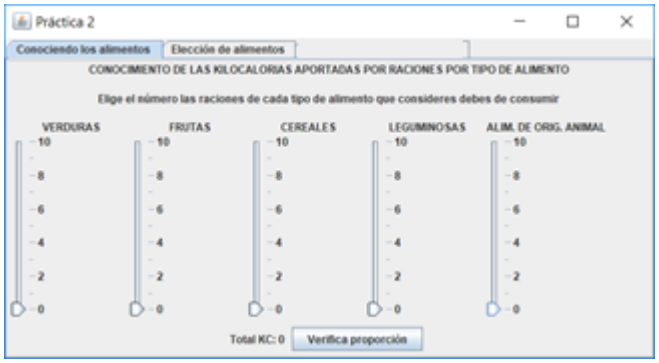
```
int totalkc=0;
     for(int ta=0; ta< COLORES.length; ta++)
     { int base= (racTipoAlim[ta].getHeight()-racTipoAlim[ta].getValue()*KCAL_TIPO_ALIM[ta])*racTipoAlim[ta].getHeight()/racTipoAlim[ta].getHeight();
       kCalTipoAlim[ta].setBounds(0,base,60,racTipoAlim[ta].getHeight()*racTipoAlim[ta].getValue()*KCAL_TIPO_ALIM[ta]/racTipoAlim[ta].getHeight());
       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();
    }
```

f) Describe en forma esquemática lo que hace el código mostrado en e), lo cual deberás entregar en tu reporte final.

g) Regístrale a cada elemento de tipo JSlider (racTipoAlim[ta]) el administrador de este tipo de evento.

4. Prueba de la funcionalidad del panel: ¨Conociendo los alimentos ¨.

a) Ejecuta la aplicación, al inicio debe verse como;



b) Varia el número de raciones de cada tipo de alimentos con cada deslizador (JSlider), debiéndose ver como se muestra en el apartado de la descripción . Graba las imágenes de ejecución en tu bitácora

c) Modifica el código para los casos en que el valor de las calorías rebase la altura de su JSlider.

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

```
public class Alimentos {
private String alimento; // Nombre del alimento
private int tipo; // identifcacion del tipo de alimento
public Alimentos(String alimento,int tipo)
{
   this.alimento=alimento;
   this.tipo = tipo;
}
public int tipo()
{return tipo;}
public String alimento()
{
 return alimento;
}
@Override
public String toString()
{
  return alimento;
}
public boolean equals(Alimentos ali)
{
  return alimento.equals(ali.alimento) && tipo == ali.tipo;
}
}
```

b) En la clase ‘**ManejoAlimento’** 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

```
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[2]= new Alimentos("Acelgas",0);
          alimentos[2]= new Alimentos("Tomate",0);
          alimentos[2]= new Alimentos("Platano",1);
          alimentos[2]= new Alimentos("Naranja",1);
```

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()"**

a) Declaración de los atributos necesarios a usar en la clase.

```
private final String TIEMPOSCOMIDA[] = {"Desayuno", "Comida", "Cena"};
    private JCheckBox eleccionTiempo[];
    private DefaultListModel <Alimentos> modelosListas[];
    private JList tiempos[];
    private JComboBox alimento;
    private JButton aceptar; //para agregar un alimnento
```

b) Especificación de los elementos a agregar en el método "**panelAlimentos()**":

```
JPanel pAlimentos = new JPanel();
        pAlimentos.setLayout(new BorderLayout());
        JPanel panelCentro = new JPanel();//Para la selección de alimentos y cuadros de verifcación
        JPanel panelCheck = new JPanel();//Para cuadros de verificación de tiempos de comidas
        panelCheck.setLayout(new GridLayout(1, 0));
        panelCentro.setLayout(new BorderLayout());
        JPanel panelSeleccion = new JPanel();//Para la selcción 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("");
        alimento = new JComboBox(alimentos);//creacion del cuadro combinado
        tipoAli.setText(TIPO_ALIMENTO[0]);// se muestra el primero

        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;
```

c) Guarda los cambios, pues antes de probarlo se continua con la creación del panel que se regresa en "creaPanelTiempos()"

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

```
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");
```

b) Creación de los atributos a utilizar. Te corresponde definir la longitud de los arreglos

```
eleccionTiempo = new JCheckBox[ ? ];
numAlimTie = new int[ ? ];
```

c) Definición del tipo de distribución para '**eleccion**' de tipo GridLayout(). Especifica su dimensión

```
eleccion.setLayout(new GridLayout( ?, ? ))
```

d) Creación de los elementos de 'cuadros de verificación'

```
for (int t = 0; t < eleccionTiempo.length-1; t++) {
            eleccionTiempo[t] = new JCheckBox(TIEMPOSCOMIDA[t]);
            eleccion.add(eleccionTiempo[t]);
            numAlimTie[t]=0;
        }
        eleccionTiempo[eleccionTiempo.length-1] = new JCheckBox("En todos");
        eleccion.add(eleccionTiempo[eleccionTiempo.length-1]);
```

e) Haz que se agregue el botón '**aceptar'** en el panel '**tiemposAlimento'** en el lugar que le corresponde

f) Se agrega a '**eleccion'** y se regresa el panel formado

```
tiemposAlimento.add(eleccion,BorderLayout.CENTER);
return tiemposAlimento;
```

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

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

b) Agregación de los elementos de tipo etiqueta que guardan el total de alimentos agregados a cada lista, completa lo que hace falta indicado por ?.

```
for (int t = 0; t < tiempos.length; t++) {
        numAlimTiem[t]=   ?
         ?.add(numAlimTiem[t]);
      }
```

c) Agregar los elementos y regreso del panel formado

```
pListas.add(listas,BorderLayout.NORTH);
pListas.add(numAlim,BorderLayout.SOUTH);
return pListas;
```

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)'

Para administrar el evento 'ItemEvent', substituye lo que le corresponde por '?' el siguiente código que va dentro de su método correspondiente:

//Se tiene que mostrar el nombre del tipo de alimento seleccionado en el JComboBox

```
JComboBox aliSel= (JComboBox)ie.getSource();
 int t=((Alimentos)aliSel.getSelectedItem()).tipo();
 tipoAli.setText( ? );
```

d) El evento de tipo acción es producido por dos tipos de elementos: por los de tipo JCheckBox

(elementos de arreglo eleccionTiempo[ ]) y el elemento de tipo JButton(llamado aceptar). Regístrale a dichos elementos al administrador de este tipo de evento.

e) Administración del evento producido por los elementos 'eleccionTiempo[ ]'. En el método que implementa al administrador de ActionEvent, agrega el siguiente código, completa lo que falta indicado por el '?'

```
// Actualiza elección del tiempo de comida
     boolean seleccion = true;
     for(int t=0;t<eleccionTiempo.length-1;t++)
       seleccion=  ?  && eleccionTiempo[t]. ?;
     eleccionTiempo[eleccionTiempo.length-1].setSelected(seleccion);
    }
```

f) Anota en tu bitácora de actividades una descripción de lo que entiendes realiza el código anterior.

g) Administración del evento producido por el botón identificado como 'aceptar'. En el método que implementa al administrador de ActionEvent, agrega el siguiente código.

```
if(prod==aceptar)
     { Alimentos aliSel = (Alimentos) alimento.getSelectedItem();
       for(int t=0;t<eleccionTiempo.length-1;t++)
       if(eleccionTiempo[t].isSelected())
       {   modelosListas[t].addElement(aliSel);
            numAlimTiem[t].setText("Num.Alim.:"+(++numAlimTie[t]));
            numTipoAli[aliSel.tipo()]++;

       }
     }
```

f) Guarda los cambios de tu proyecto

10. Prueba y afinación de la funcionalidad de 'Elección de alimentos'

a) Prueba agregando diferentes tipos de alimentos a diferentes tiempos(Desayuno, comida y/o cena), observaras que un mismo alimento se agrega más de una vez en un tiempo

b) Modifica el administrador de eventos de acción para que una vez que se ha agregado un alimento en un tiempo no se vuelva agregar en ese tiempo.

c) Prueba de nuevo y guarda el código y prueba de captura de pantalla en tu bitácora de seguimiento.

d) Ahora observa que cuando se seleccionan en forma individual un tiempo el de todos los tiempo no se activa en forma automática. Haz las modificaciones pertinentes para que se cumpla con lo especificado en la descripción. Guarda de nuevo en tu bitácora el código realizado

Sugerencias didácticas

Utiliza una bitácora de seguimiento de lo que vayas realizado

Usa un sistema de versiones para cada fase del programa que vayas creando

Comparte el seguimiento y resultados en un foro del grupo

Reporte del estudiante

El estudiante deberá entregar los siguientes productos

v) Código de la aplicación propuesta que funcione
vi) Un cuadro de los casos de prueba
vii) Resultados de los casos de prueba
viii) Bitácora de seguimiento con las anotaciones solicitadas durante el desarrollo de la práctica
ix) Conclusiones sobre los principales aspectos a considerar en una aplicación GUI

Bibliografía

4. Wu Thomas, Programación en Java, Mc. Graw-Hill, 2008
5. Deitel y Deitel. Como programar en Java. Prentice Hall.Septima Edición, 2008
6. http://www.imss.gob.mx/sites/all/statics/salud/guia-alimentos.pdf .Ultimo acceso 15/11/18