Programación Basada en Eventos

La construcción de una GUI utiliza un modelo de **programación basado en eventos**.

En este modelo el **orden** en el cual se ejecutan las instrucciones de un programa va a quedar determinado por **eventos.**

Un evento es una **señal** de que algo ha ocurrido.

En esta materia consideraremos únicamente eventos generados por acciones del usuario al interactuar con la GUI.

Programación Basada en Eventos

Algunas componentes de una GUI van a ser **reactivas**, es decir tienen la capacidad de reaccionar ante las acciones del usuario.

Una componente reactiva están asociada a un **objeto fuente del evento** creado por el programador.

La reacción del sistema en respuesta a la acción del usuario va a quedar determinada por la clase a la que pertenece un **objeto oyente.**

El objeto oyente está ligado al objeto fuente de evento a través de una instrucción de **registración**.

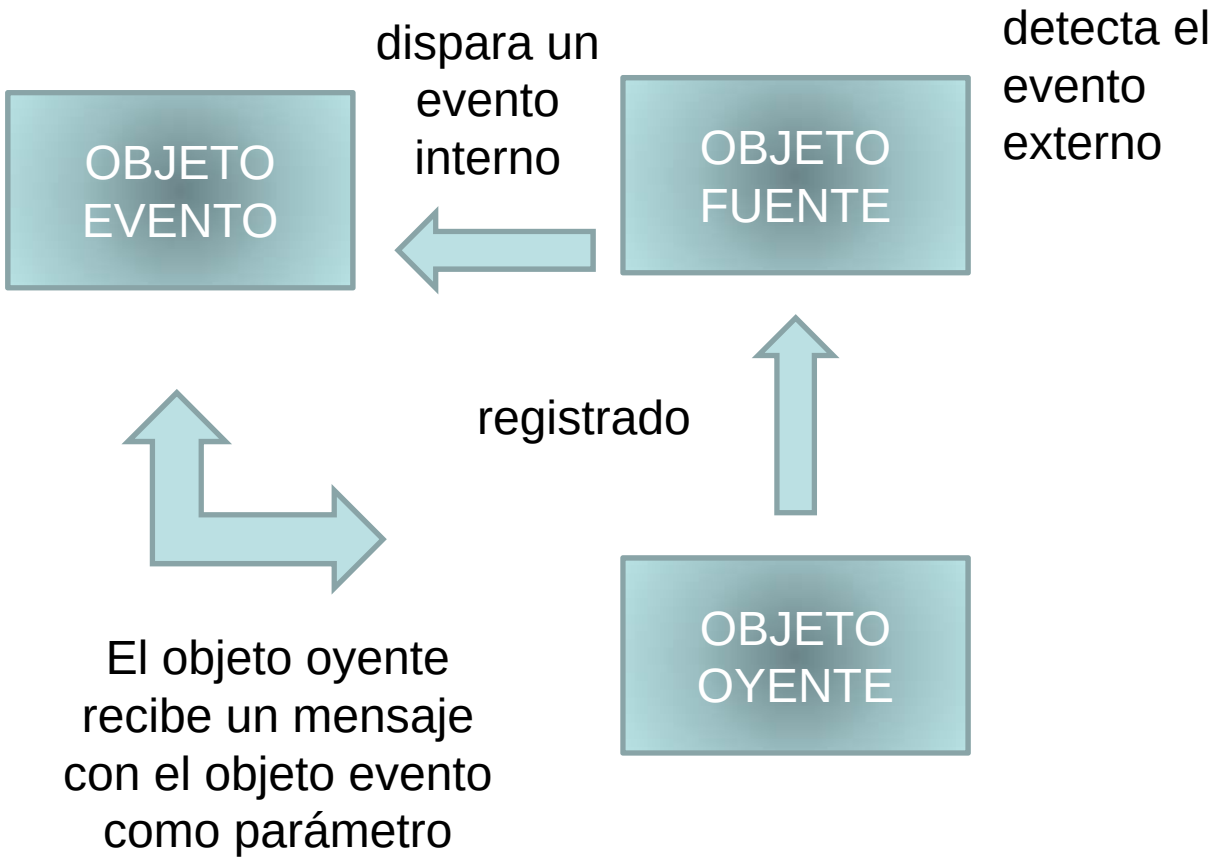
Programación Basada en Eventos

Un **objeto fuente de evento** tienen la capacidad de ***percibir*** un evento externo y ***disparar*** un evento interno, esto es, crear un **objeto evento de software**.

Este objeto evento de software es el argumento de un mensaje enviado al **objeto oyente**.

El método que se ejecuta en respuesta a este mensaje forma parte de una interface provista por Java y es implementado por el programador en la clase del oyente.

Programación Basada en Eventos



Interfaces Gráficas

La construcción de una GUI va a requerir

- Definir clases que **deriven** de las clases gráficas provistas por Java.
- Definir clases que **implementen** interfaces gráficas provistas por Java.
- Crear objetos de las clases provistas o de las clases que derivadas o implementadas
- Elegir un organizador de layout y especificar la apariencia de las componentes.
- Insertar las componentes en los contenedores

Interfaces Gráficas

La estructura de las GUI que hemos estamos definiendo consta entonces de:

- Instrucciones para importar paquetes gráficos.
- La definición de una clase que crea un frame de una clase que extiende a JFrame y lo hace visible.
- La definición de la clase que extiende a JFrame e incluye:
  - ❖ Atributos asociados a componentes de la GUI y otros vinculados a la aplicación
  - ❖ Un constructor
  - ❖ Clases internas que implementan interfaces y permiten crear oyentes

Interfaces Gráficas

Un constructor incluye instrucciones para:

- ✓ crear objetos ligados a componentes gráficas
- ✓ crear objetos oyente para las componente gráficas que sean objetos fuente de eventos y registrarlos
- ✓establecer el diagramado y los atributos de las componentes
- ✓insertar las componentes en los contenedores

Algunas de estas instrucciones pueden ser provistas por un método interno a la clase para favorecer la modularización de la clase.

Caso de Estudio: Cuenta Bancaria

Se desea modelar una cuenta bancaria sobre la que se realizan depósitos, extracciones y consultas de saldo.

La cuenta bancaria tiene asociado un código y un saldo. El código la identifica unívocamente y es fijo. El saldo aumenta cuando el titular de la cuenta efectúa un depósito y disminuye cuando realiza una extracción.

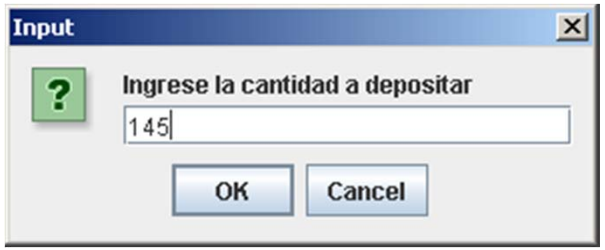
El usuario está autorizado a girar en descubierto hasta un monto máximo establecido.

Caso de Estudio: Cuenta Bancaria

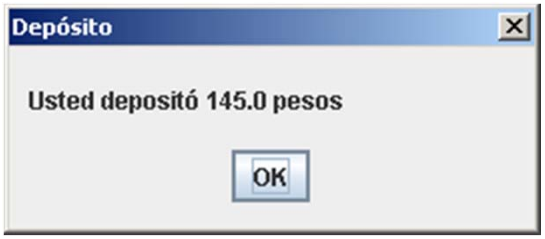
El usuario opera con su cuenta bancaria a través de una interfaz gráfica como la que sigue

Caso de Estudio: Cuenta Bancaria

*Si el usuario presiona el **botón** Depositar en pantalla aparece un **cuadro de diálogo** como el que sigue:*

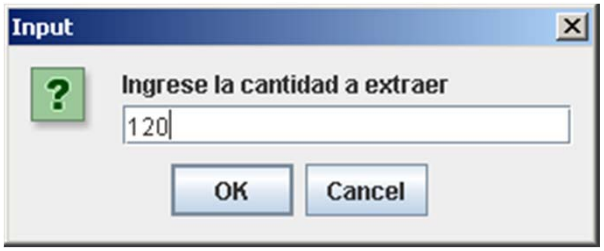


Si el usuario presiona el botón OK aparece en pantalla un...



Caso de Estudio: Cuenta Bancaria

*Análogamente si el usuario presiona el **botón** Extraer en pantalla aparece un **cuadro de diálogo** como el que sigue:*



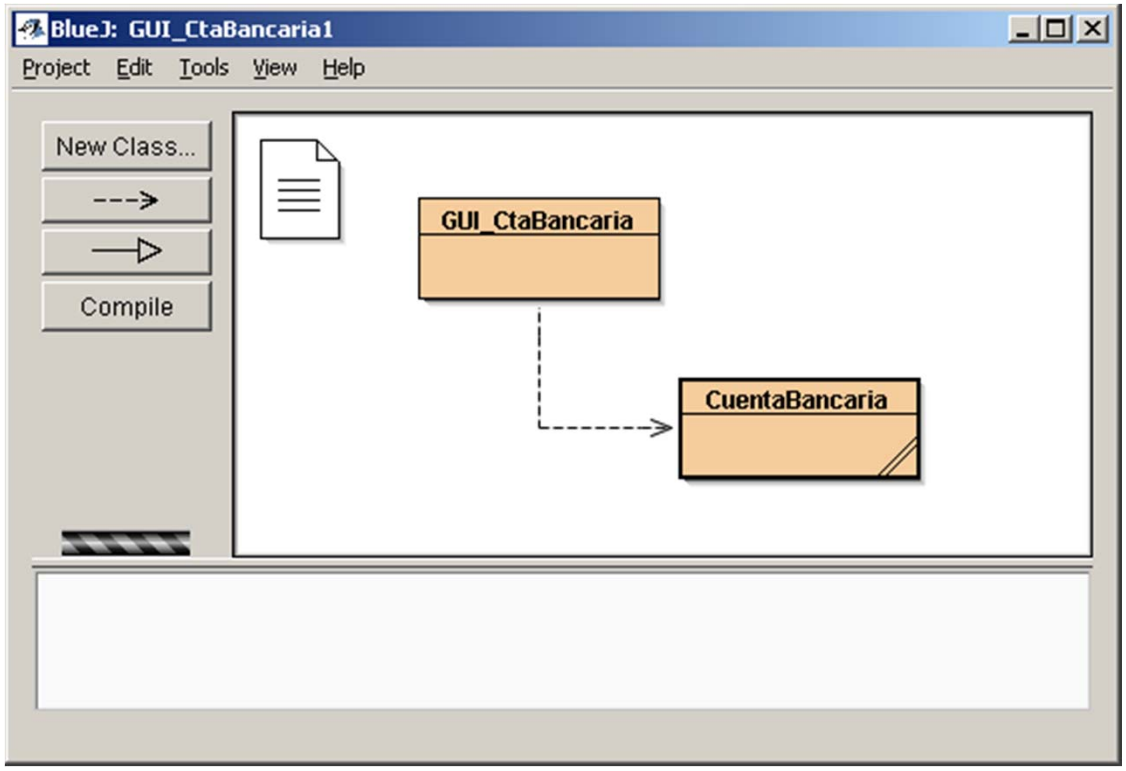
Si el usuario presiona el botón OK aparece en pantalla un...

Caso de Estudio: Cuenta Bancaria

*Si el usuario presiona el **botón** Consultar Saldos en pantalla aparece un*

O bien

Caso de Estudio: Cuenta Bancaria




```
class Cajero
public static void main(String[] args) {
   GUI_CtaBancaria unaCuenta = new GUI_CtaBancaria();
   unaCuenta.setVisible(true);
   }
}
```

Caso de Estudio: Cuenta Bancaria

```
import …
public class GUI_CtaBancaria extends JFrame {
    private CuentaBancaria cuenta;
    private Container contenedor;
    private JPanel panelAcciones, panelSaldo;
    private JButton botonConsultar,
                    botonExt, botonDep;

    public GUI_CtaBancaria() {
    …
    }
  …
}
```

Caso de Estudio: Cuenta Bancaria

- Crear la Cuenta Bancaria
- Capturar el panel de contenido
- Crear el panel de acciones y de consulta
- Crear cada uno de los tres botones
- Establecer la apariencia de los botones
- Crear los tres oyentes
- Registrar cada oyente a su botón
- Establecer la apariencia de los paneles
- Insertar los botones a los paneles correspondientes
- Insertar los paneles al panel de contenido

Caso de Estudio: Cuenta Bancaria

```
public GUI_CtaBancaria() {
    cuenta = new CuentaBancaria(3);
    contenedor = getContentPane();
    panelAcciones = new JPanel();
    panelSaldo = new JPanel();
    botonDep = new JButton();
    botonExt = new JButton();
    botonConsultar = new JButton();
    setSize(210, 210);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    armarGUI();
}
```

Caso de Estudio: Cuenta Bancaria

```
public void armarGUI() {
//Apariencia de los botones
 botonDep.setPreferredSize(new Dimension(124, 50));
 botonDep.setSize(150, 50);
 botonDep.setBorder(BorderFactory.createCompoundBorde(
     new LineBorder
   (new java.awt.Color(0, 0, 0), 1, false),null));
botonExt.setText("Extraer");
 botonExt.setPreferredSize(new Dimension(124, 50));
 botonExt.setSize(150, 50);
 botonExt.setBorder(BorderFactory.createCompoundBorder(
       new LineBorder(
      new java.awt.Color(0, 0, 0), 1, false),null));
 botonConsultar.setText("Consultar Saldo");
 botonConsultar.setPreferredSize(new Dimension(136, 30));
 botonConsultar.setSize(150, 30);
 botonConsultar.setBorder
(BorderFactory.createBevelBorder(BevelBorder.LOWERED));
}
```

Caso de Estudio: Cuenta Bancaria

```
public void armarGUI() {

//Crear oyentes
 OyenteDepositar oDepositar=new OyenteDepositar();
 OyenteExtraer oExtraer = new OyenteExtraer();
 OyenteConsultar oConsultar =new OyenteConsultar();

// Registrar oyentes
 botonDep.addActionListener(oDepositar);
 botonExt.addActionListener(oExtraer);
 botonConsultar.addActionListener(oConsultar);
}
```

Caso de Estudio: Cuenta Bancaria

```
public void armarGUI() {
…
 // Layout del panel contenedor
   contenedor.setLayout(new BorderLayout());

// Panel de Acciones
  panelAcciones.setBorder(
BorderFactory.createEtchedBorder(BevelBorder.LOWERED));
   panelAcciones.setPreferredSize(
       new Dimension(160, 130));
   panelAcciones.setSize(160, 125);


…
}
```

Caso de Estudio: Cuenta Bancaria

```
public void armarGUI() {
…
// Insertar botones a los paneles

   panelAcciones.add(botonDep);
   panelAcciones.add(botonExt);
   panelSaldo.add(botonConsultar);

// Insertar los paneles al contenedor
   contenedor.add(panelAcciones, BorderLayout.NORTH);
   contenedor.add(panelSaldo, BorderLayout.SOUTH);
}
```

Caso de Estudio: Cuenta Bancaria

```
private class OyenteDepositar implements ActionListener {
        public void actionPerformed(ActionEvent event){
  float dep;
  String deposito;
  JOptionPane dialogo = new JOptionPane();

  deposito = dialogo.showInputDialog
       ( "Ingrese la cantidad a depositar" );
  if ((deposito != null) && (deposito.length() > 0)){
    dep = Float.parseFloat(deposito);
    dialogo.showMessageDialog(null,
         "Usted depositó " + dep+ " pesos","Depósito",
            JOptionPane.PLAIN_MESSAGE );
      cuenta.depositar(dep);
  }
}

}
```

Caso de Estudio: Cuenta Bancaria

```
private class OyenteExtraer implements ActionListener {
        public void actionPerformed(ActionEvent event){
  float ext;
  String extraccion;
  JOptionPane dialogo = new JOptionPane();
  extraccion = dialogo.showInputDialog
          ( "Ingrese la cantidad a extraer" );
  if ((extraccion != null) && (extraccion.length() > 0)){
   ext = Float.parseFloat(extraccion);
   if (cuenta.puedeExtraer(ext)){
      JOptionPane.showMessageDialog( null,
                  "Usted extrajo "+ext+ " pesos",
                "Extracción", JOptionPane.PLAIN_MESSAGE );
      cuenta.extraer(ext) ;}
    else
      dialogo.showMessageDialog( null,
          "Usted NO puede extraer esa cantidad“,
"Advertencia", JOptionPane.WARNING_MESSAGE );
```

Caso de Estudio: Cuenta Bancaria

```
private class OyenteConsultar implements ActionListener {
  public void actionPerformed(ActionEvent event){
     JOptionPane dialogo = new JOptionPane();

     if (cuenta.obtenerSaldo()>=0)
            dialogo.showMessageDialog(null,
      "Usted tiene un saldo de " + cuenta.obtenerSaldo()+
      " pesos“,"SALDO“,JOptionPane.INFORMATION_MESSAGE );
     else
       dialogo.showMessageDialog(null,
          "Usted está en rojo en " +
          (-1)*cuenta.obtenerSaldo() + " pesos“,
            "SALDO", JOptionPane.ERROR_MESSAGE );
       }
   }
```

Caso de Estudio: Cuenta Bancaria

Conectamos la clase CuentaBancaria a una GUI sin modificar el código que está completamente encapsulado.

La clase CuentaBancaria es proveedora de servicios

La clase GUICuentaBancaria ***usa*** CuentaBancaria como una caja negra, conoce únicamente la interface y el contrato.

Podemos modificar la GUI sin cambiar la clase asociada y viceversa.

Maquina Expendedora

Una fábrica produce dos tipos diferentes de máquinas expendedoras de infusiones, M111 y R101.

Las máquinas del tipo M111 preparan ***café***, ***café con leche***, ***té***, ***té con leche*** *y* ***submarino***. *Tienen depósitos para los siguientes ingredientes:* ***café***, ***té***, ***leche*** *y* ***cacao***.

Las máquinas de tipo R101 preparan ***café***, ***té*** *y* ***café carioca*** *. Tienen depósitos para* ***café***, ***té***, ***crema*** *y* ***cacao***.

Maquina Expendedora

Los depósitos tienen las siguientes capacidades máximas:

Café	**1500**
Té	**1000**
Leche	**600**
Cacao	**600**
Crema	**600**

Además de la capacidad máxima de cada ingrediente, cada máquina mantiene registro de la cantidad disponible.

Maquina Expendedora

Cuando se habilita una máquina las cantidades disponibles comienzan con el valor máximo de cada ingrediente.

La cantidad disponible aumenta cuando se carga el depósito con un ingrediente específico y disminuye cada vez que se prepara una infusión.

El aumento es variable, aunque nunca se puede superar la capacidad máxima de cada ingrediente.

Si el valor que se intenta cargar, sumado al disponible, supera al máximo, se completa hasta el máximo y retorna el sobrante.

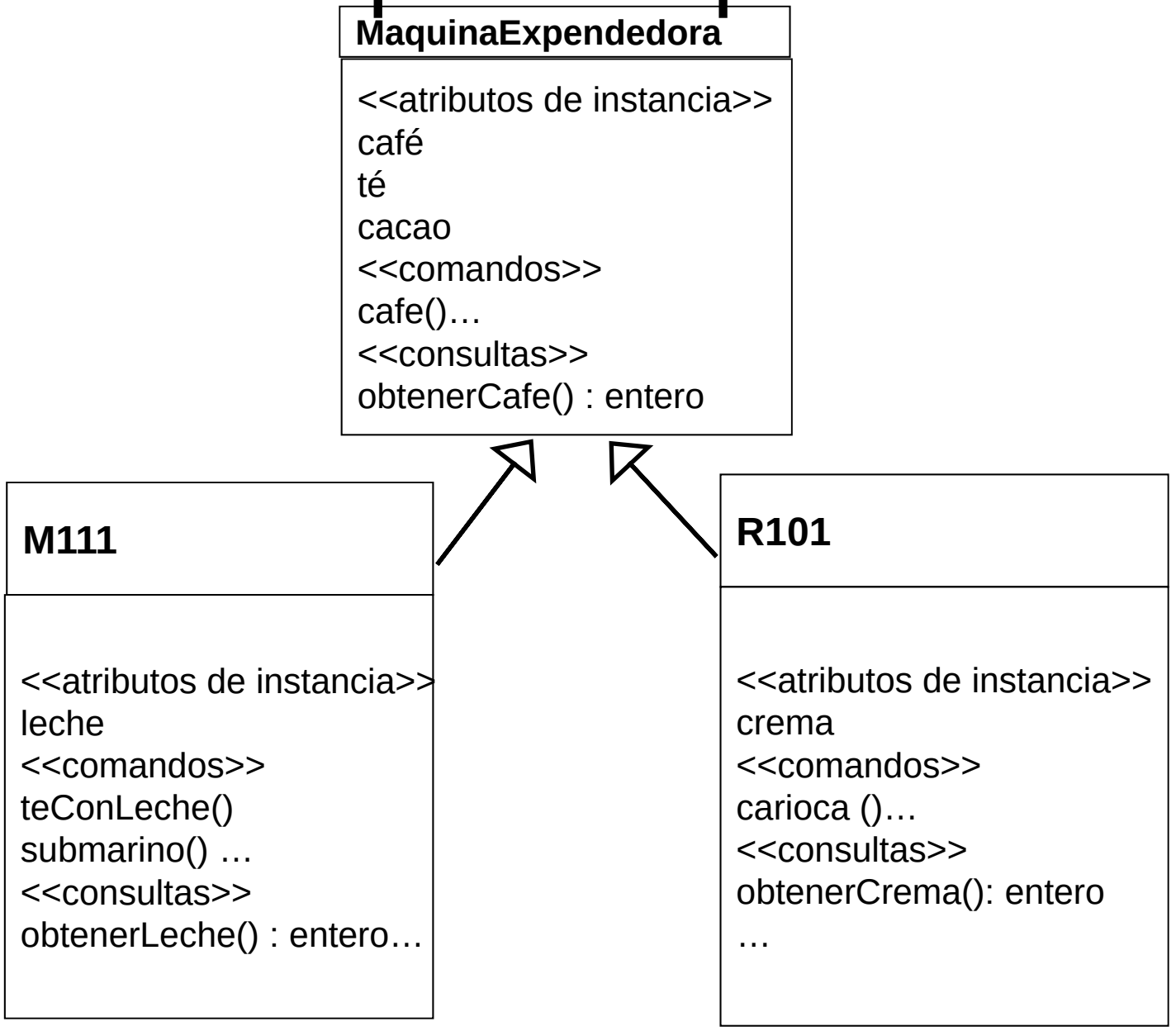
Maquina Expendedora

Cada vez que se solicita una infusión se reducen los ingredientes de acuerdo a la siguiente tabla:

	Café	Café con leche	Submarino	Té con leche	café carioca
Café	40	30			30
Leche		20	50	20	
Té				20	
Cacao			40		10
Crema					30

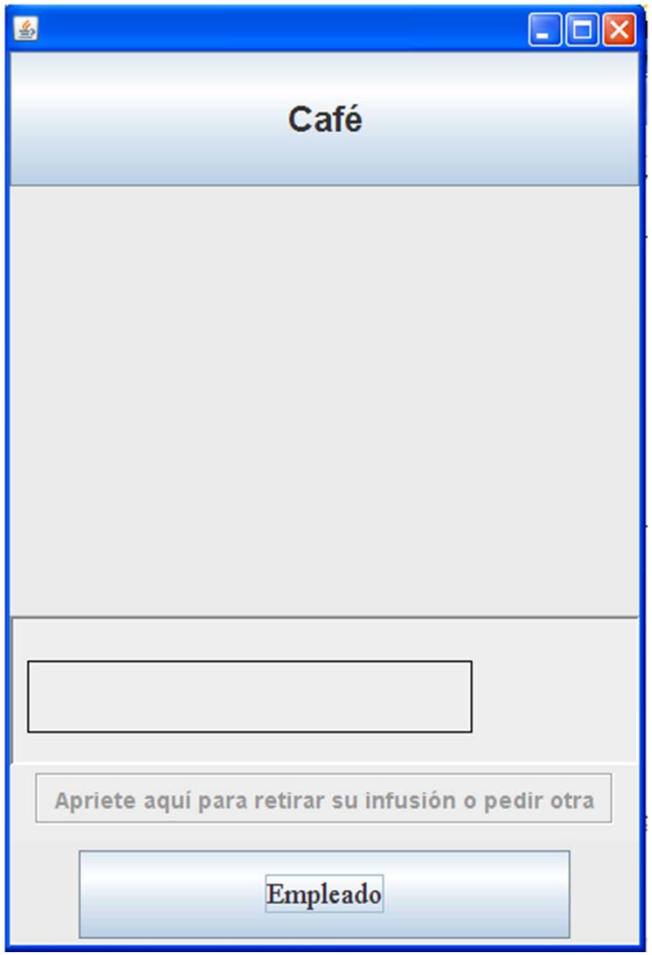
En el caso de la preparación de una taza de té, la máquina M111 utiliza 10 grs y la máquina R101 15 grs.

Maquina Expendedora



Maquina Expendedora

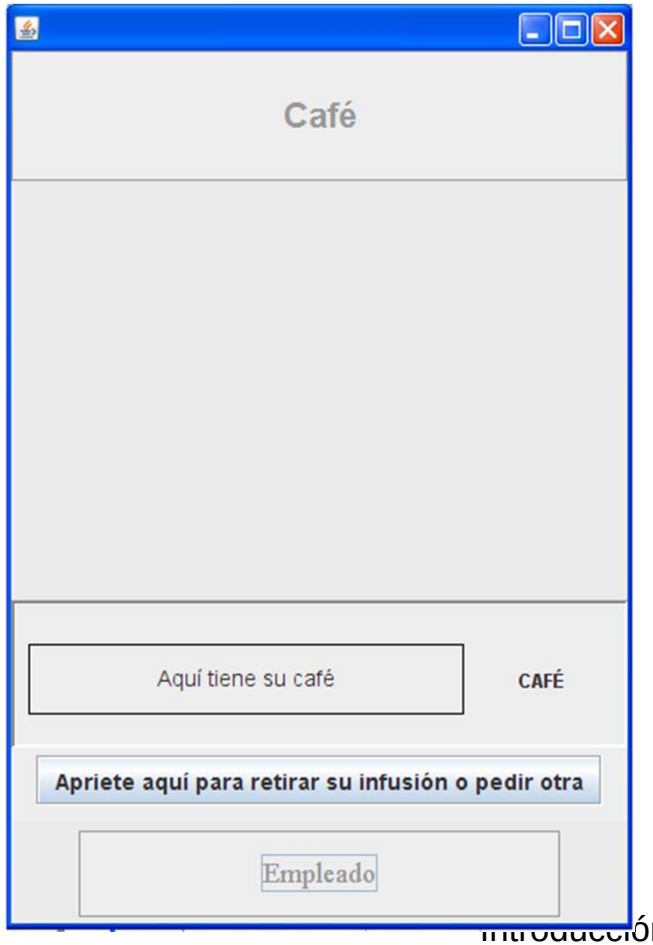
Comenzaremos implementando parcialmente una GUI para una máquina expendedora del modelo R101



Inialmente está activo el botón para preparar café y uno para el Empleado que permite cargar ingredientes

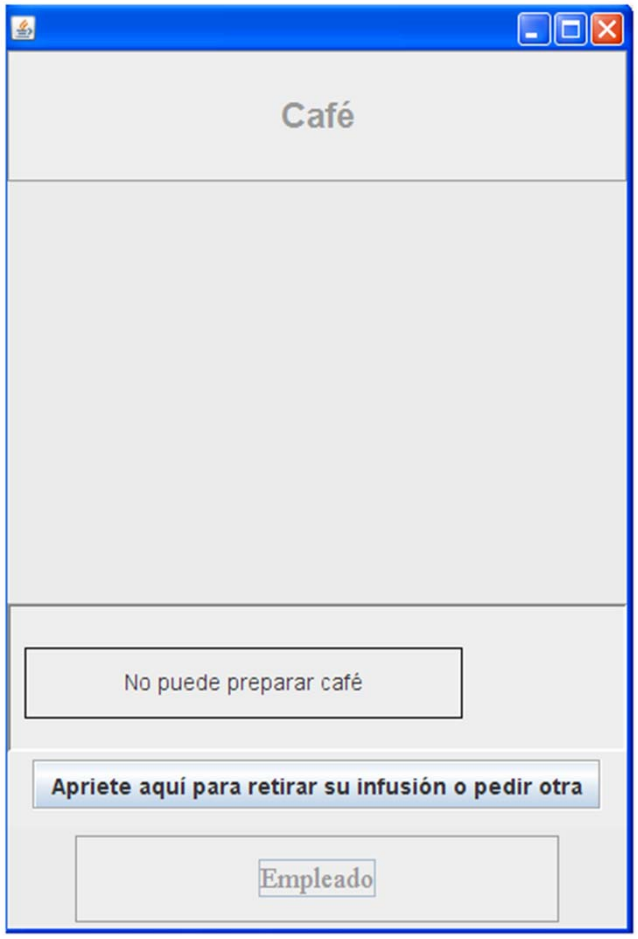
Maquina Expendedora

Si se oprime el botón Café aparece un cartel informativo y el único botón activo es el que debe seleccionarse al retirar el vaso



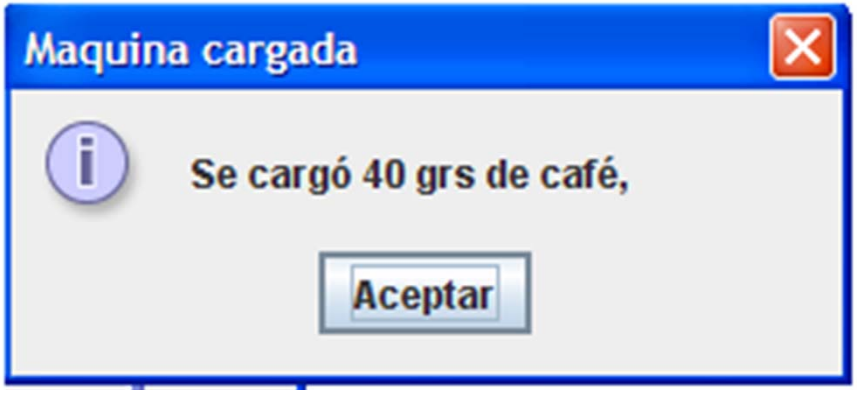
Maquina Expendedora

Si la cantidad de ingredientes no es suficiente se muestra un cartel y se activa el botón que permite volver al estado inicial.



Maquina Expendedora

Si se elige el botón Empleado se cargan todos los ingredientes hasta llegar al máximo y aparece un cartel informativo:



Al oprimir Aceptar vuelve a aparecer la pantalla inicial.

Maquina Expendedora

panelBotones

panelES

panelRetirar

panelEmpleado

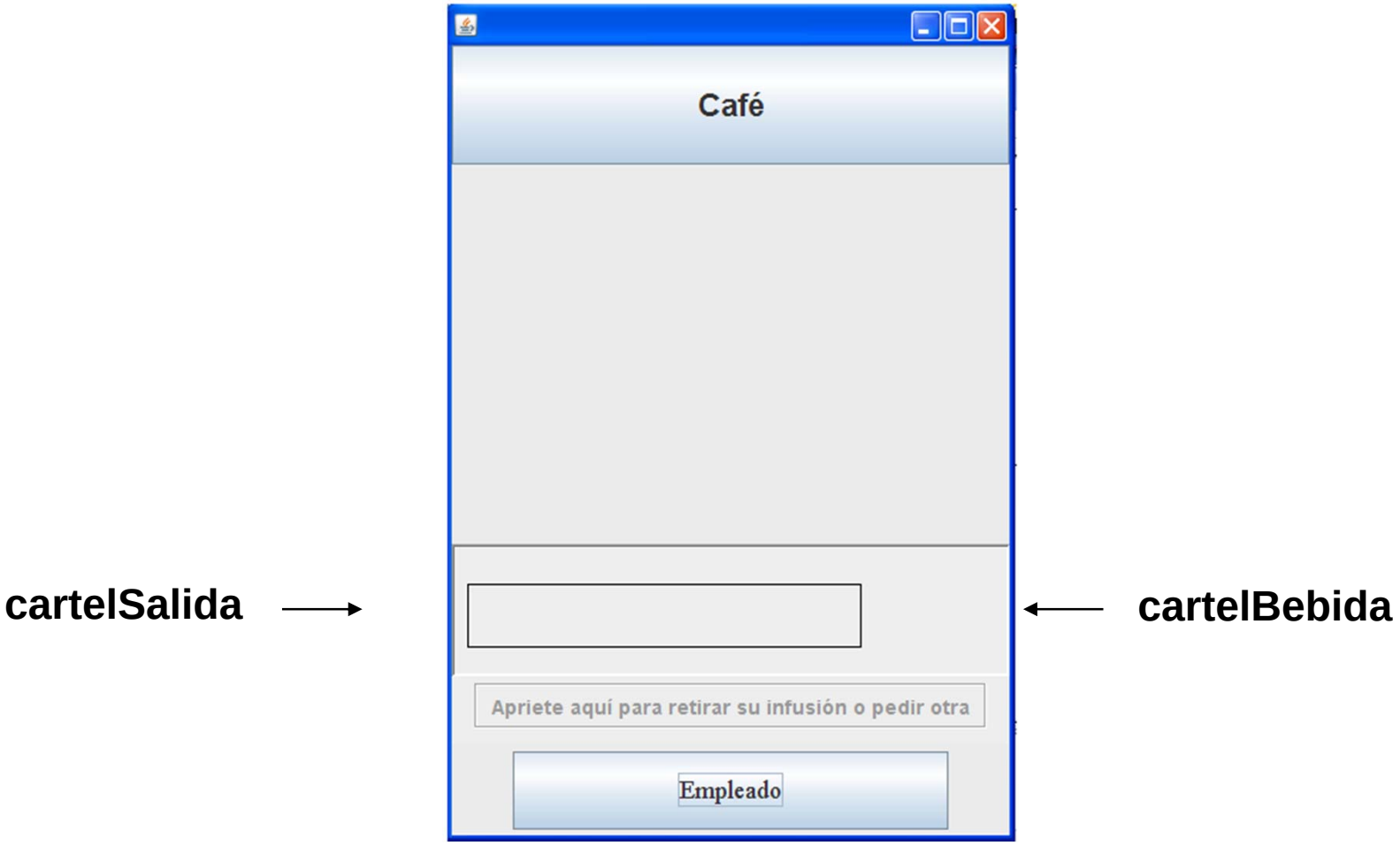
Maquina Expendedora

botonCafe →

botonRetirarInfusion →

botonEmpleado →

Maquina Expendedora



Maquina Expendedora

```
public class GUI_R101 extends JFrame {

    private R101  unaMaquina;
    private Container contenedor;
    private JPanel panelBotones, panelES,
            panelRetirar, panelEmpleado;
    private JLabel cartelSalida,cartelBebida;
    private JButton botonCafe,
                    botonRetirarInfusion,
                    botonEmpleado;

…
}
```

Maquina Expendedora

Un constructor incluye instrucciones para:

- ✓ capturar el panel de contenido
- ✓crear paneles, etiquetas y botones
- ✓ crear objetos oyente para los botones
- ✓establecer el diagramado y atributos de los paneles
- ✓establecer los atributos de etiquetas y botones
- ✓insertar botones y etiquetas en los paneles y los paneles en el panel de contenido

Maquina Expendedora

```
public GUI_R101()    {

    unaMaquina = new R101();

// Declaración de objetos gráficos
    contenedor = getContentPane();
    botonCafe = new JButton() ;
    botonRetirarInfusion = new JButton() ;
    botonEmpleado = new JButton() ;
    cartelSalida = new JLabel();
    cartelBebida = new Jlabel();
    panelBotones = new JPanel();
    panelES = new JPanel() ;
    panelRetirar = new JPanel() ;
    panelEmpleado = new JPanel() ;
    initGUI();
}
```

Maquina Expendedora

```
private void initGUI() {


   BoxLayout esteLayout = new
      BoxLayout(contenedor,BoxLayout.Y_AXIS) ;
  contenedor.setLayout(esteLayout);



…
}
```

Maquina Expendedora

```
private void initGUI() {


//Boton y oyente del café

  OyenteCafe oCafe = new OyenteCafe();
  botonCafe.setText("Café");
  botonCafe.setFont(new Font("Arial",1,22));
  botonCafe.addActionListener(oCafe);


…
}
```

Maquina Expendedora

```
private void initGUI() {

//Boton y oyente retirar

  OyenteRetirar oRetirar = new OyenteRetirar();
  botonRetirarInfusion.setText
  ("Apriete aquí para retirar su infusión o pedir otra");
   botonRetirarInfusion.setEnabled(false);
   botonRetirarInfusion.setFont(new
      Font("SansSerif",1,14));
   botonRetirarInfusion.setBorder
      (BorderFactory.createEtchedBorder(BevelBorder.LOWER
ED));
   botonRetirarInfusion.setPreferredSize
       (new Dimension(360, 32));
   botonRetirarInfusion.addActionListener(oRetirar);
```

Maquina Expendedora

```
private void initGUI() {

//Boton y oyente Empleado

OyenteEmpleado oEmpleado = new
      OyenteEmpleado();

 botonEmpleado.setText("Empleado");
 botonEmpleado.setPreferredSize(new Dimension(306, 55));
 botonEmpleado.setFont(new Font("Times New Roman",0,18));

 botonEmpleado.addActionListener(oEmpleado);

…
}
```

Maquina Expendedora

```
private void initGUI() {

//Diagramado del Panel de Botones

 GridLayout panelBotonesLayout = new
           GridLayout(4, 1) ;

  panelBotonesLayout.setHgap(5);
  panelBotonesLayout.setVgap(5);
  panelBotones.setLayout(panelBotonesLayout);
  panelBotones.setPreferredSize(new Dimension(392, 369));
  panelBotones.setSize(369, 250);
  panelBotones.setBackground(new Color(235,235,235));


…
}
```

Maquina Expendedora

```
private void initGUI() {
…
//Diagramado del Panel de ES

 panelES.setPreferredSize(new Dimension(392, 101));
 panelES.setSize(369, 51);
 panelES.setBorder
(BorderFactory.createBevelBorder(BevelBorder.LOWERED));


…
}
```

Maquina Expendedora

```
private void initGUI() {
…
// Apariencia carteles
 cartelSalida.setLayout(new FlowLayout());
 cartelSalida.setBorder (new LineBorder
       (new Color(0,0,0), 1, false));
 cartelSalida.setPreferredSize(new Dimension(277, 45));
 cartelSalida.setHorizontalAlignment
      (SwingConstants.CENTER);
 cartelSalida.setHorizontalTextPosition
      (SwingConstants.LEFT);
 cartelSalida.setFont(new Font("Arial",0,14));

 cartelBebida.setText("");
 cartelBebida.setHorizontalAlignment
      (SwingConstants.CENTER);
 cartelBebida.setPreferredSize(new Dimension(88, 88));
…}
```

Maquina Expendedora

```
private void initGUI() {
…

// Diagramado Paneles Retirar y Empleado
 panelRetirar.setPreferredSize(new Dimension(392, 50));


  panelEmpleado.setPreferredSize(new Dimension(392, 64));
  panelEmpleado.setBackground(new Color(235,235,235));


…
}
```

Maquina Expendedora

```
private void initGUI() {
…
//Insertar botones, carteles y paneles
  panelBotones.add(botonCafe);
  contenedor.add(panelBotones);

  panelES.add(cartelSalida);
  panelES.add(cartelBebida);
  contenedor.add(panelES);

  panelRetirar.add(botonRetirarInfusion);
  contenedor.add(panelRetirar);

  panelEmpleado.add(botonEmpleado);
  contenedor.add(panelEmpleado);
}
```

Maquina Expendedora

```
public class GUI_R101 extends JFrame {
public GUI_R101() {
…
}
private void initGUI() {
…
}
private void deshabilitarBotones(){
          botonCafe.setEnabled(false);
          botonRetirarInfusion.setEnabled(true);;
          botonEmpleado.setEnabled(false);
         }

private void habilitarBotones(){
          botonCafe.setEnabled(true);
          botonRetirarInfusion.setEnabled(false);;
          botonEmpleado.setEnabled(true);
         }
}
```

Maquina Expendedora

```
public class GUI_R101 extends JFrame {
…
class OyenteCafe implements ActionListener{

   public void actionPerformed(ActionEvent evt) {

   int cantVasos = unaMaquina.vasosCafe();
   if (cantVasos>=1){
      unaMaquina.cafe();
      cartelSalida.setText("Aquí tiene su café");
      cartelBebida.setText("CAFÉ");    }
   else {
      cartelSalida.setText("No puede preparar café");
      cartelBebida.setText(""); }
   deshabilitarBotones();
                                          }
}
…
}
```

Maquina Expendedora

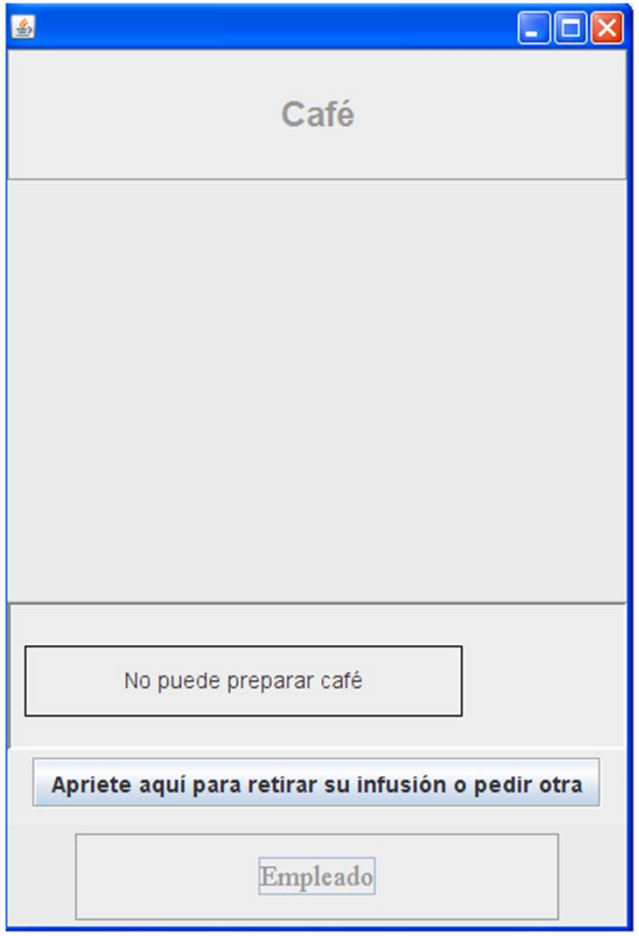
```
public class GUI_R101 extends JFrame {
…

class OyenteRetirar implements ActionListener{

    public void
    actionPerformed(ActionEvent evt) {

         cartelSalida.setText("");
         cartelBebida.setText("");
         habilitarBotones();
    }

}
…
}
```



Maquina Expendedora

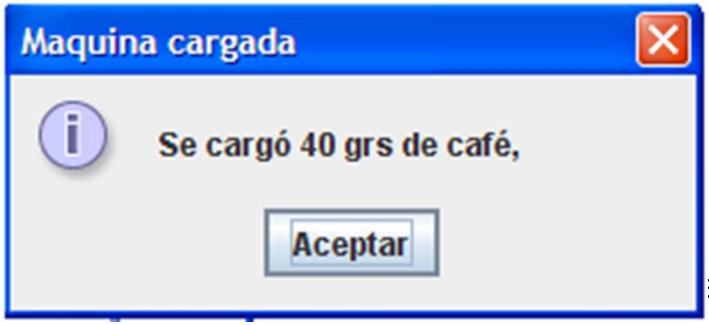
```
public class GUI_R101 extends JFrame {
…

class OyenteEmpleado implements ActionListener{

   public void actionPerformed(ActionEvent evt) {

      int cafeCargado;
      JOptionPane dialogo = new JOptionPane();
      cafeCargado = unaMaquina.obtenerMaxCafe()-
      unaMaquina.cargarCafe(unaMaquina.obtenerMaxCafe());

      dialogo.showMessageDialog(null,"Se cargó "+
      cafeCargado+ " grs de café, “,"Maquina cargada“ ,
      JOptionPane.INFORMATION_MESSAGE);          }
}
…
}
```



Maquina Expendedora

Complete la implementación con los botones que faltan

botonCafe →

botonTe →

botonCafeCarioca →

Maquina Expendedora

Implemente una GUI para la máquina R101 Plus

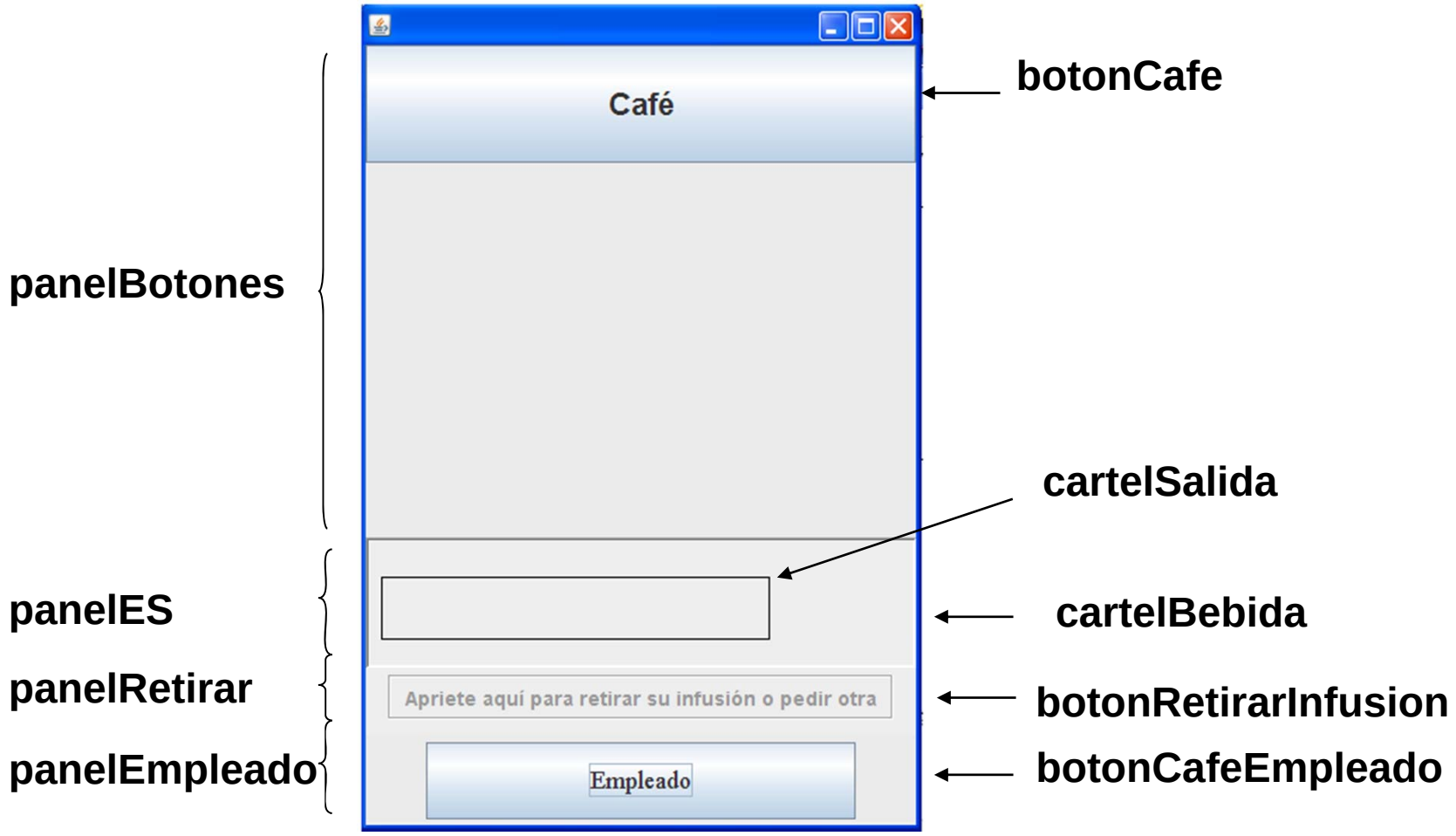
botonCafe →

botonTe →

botonCafeCarioca →

botonBahiano →

Maquina Expendedora



La GUI incluye diferentes tipos de componentes pero en este caso sólo los botones son reactivos.

Maquina Expendedora

```
public class GUI_R101 extends JFrame {
   private R101  unaMaquina;

   private Container contenedor;
   private JPanel panelBotones, panelES,
           panelRetirar, panelEmpleado;
   private JLabel cartelSalida,cartelBebida;
   private JButton botonCafe,
                   botonRetirarInfusion,botonEmpleado;
…
}
```