

**UNIVERSITY CENTER UAEM  
ATLACOMULCO**

**Bachelor Degree**

**Ingeniería en Computación**

**Subject**

**Paradigmas de la Programación**

**Teacher**

**Julio Alberto de la Teja López**

**Student**

**Alexis Valencia Martínez**

**Delivery Date**

**21/SEPTEMBER/2023**

## PROGRAM 1

### CODE

```
// Fig. 12.6: LabelFrame.java
// Componentes JLabel con texto e iconos.
import java.awt.FlowLayout; // especifica cómo se van a ordenar los
componentes
import javax.swing.JFrame; // proporciona las características básicas de una
ventana
import javax.swing.JLabel; // muestra texto e imágenes
import javax.swing.SwingConstants; // constantes comunes utilizadas con
Swing
import javax.swing.Icon; // interfaz utilizada para manipular imágenes
import javax.swing.ImageIcon; // carga las imágenes

public class LabelFrame extends JFrame {
    private JLabel etiqueta1; // JLabel sólo con texto
    private JLabel etiqueta2; // JLabel construida con texto y un icono
    private JLabel etiqueta3; // JLabel con texto adicional e icono

    // El constructor de LabelFrame agrega objetos JLabel a JFrame
    public LabelFrame() {

        super("Prueba de JLabel");
        setLayout(new FlowLayout()); // establece el esquema del marco

        // Constructor de JLabel con un argumento String
        etiqueta1 = new JLabel("Etiqueta con texto");
        etiqueta1.setToolTipText("Esta es etiqueta1");
        add(etiqueta1); // agrega etiqueta1 a JFrame

        // Constructor de JLabel con argumentos de cadena, Icono y
alineación
        Icon mapa = new ImageIcon(getClass().getResource("rtw_mapa.jpg"));
        etiqueta2 = new JLabel("Etiqueta con texto e icono", mapa,
            SwingConstants.LEFT);
        etiqueta2.setToolTipText("Esta es etiqueta2");
        add(etiqueta2); // agrega etiqueta2 a JFrame

        etiqueta3 = new JLabel(); // constructor de JLabel sin argumentos
        etiqueta3.setText("Etiqueta con icono y texto en la parte
inferior");
        etiqueta3.setIcon(mapa); // agrega icono a JLabel
        etiqueta3.setHorizontalTextPosition(SwingConstants.CENTER);
    }
}
```

```

        etiqueta3.setVerticalTextPosition(SwingConstants.BOTTOM);
        etiqueta3.setTooltipText("Esta es etiqueta3");
        add(etiqueta3); // agrega etiqueta3 a JFrame
    }
} // fin de la clase LabelFrame

```

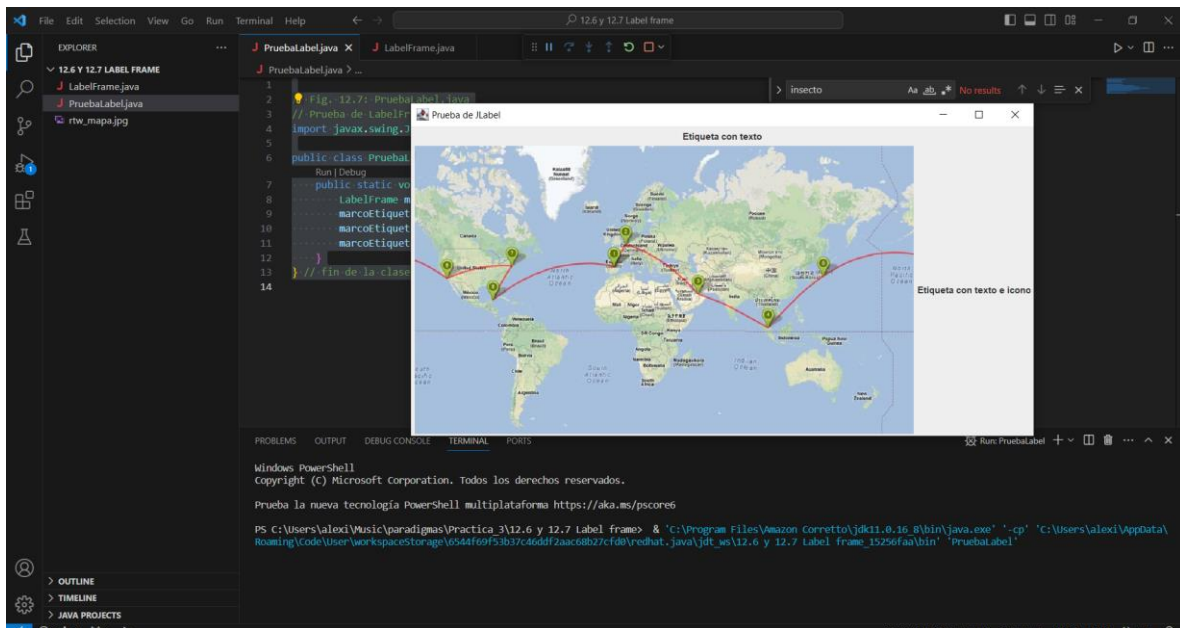
```

// Fig. 12.7: PruebaLabel.java
// Prueba de LabelFrame.
import javax.swing.JFrame;

public class PruebaLabel {
    public static void main(String[] args) {
        LabelFrame marcoEtiqueta = new LabelFrame();
        marcoEtiqueta.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoEtiqueta.setSize(260, 180);
        marcoEtiqueta.setVisible(true);
    }
} // fin de la clase PruebaLabel

```

## SALIDA



## PROGRAM 2

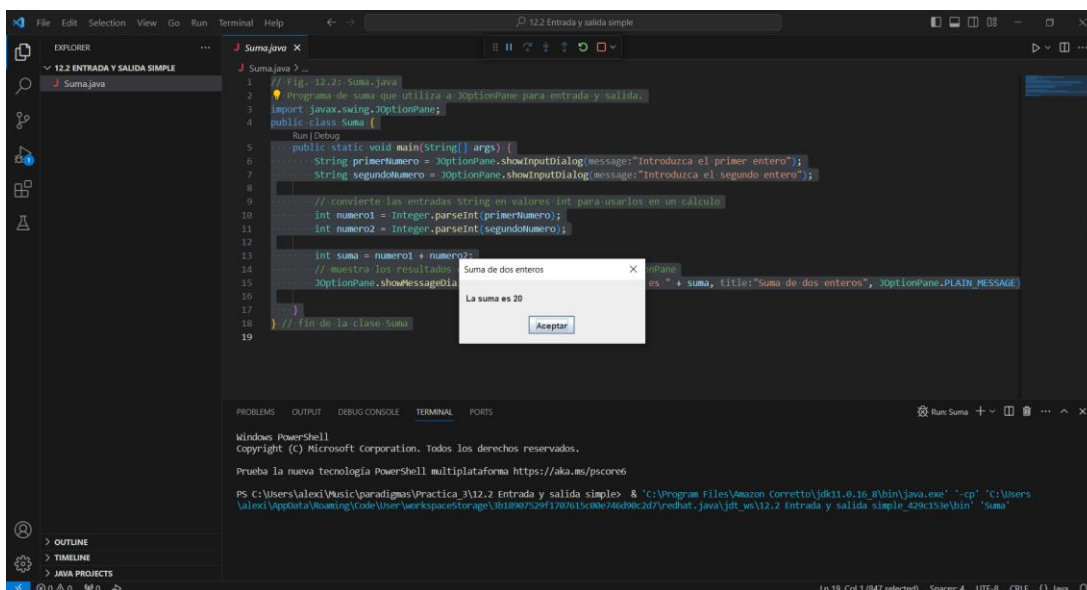
### CODE

```
// Fig. 12.2: Suma.java
// Programa de suma que utiliza a JOptionPane para entrada y salida.
import javax.swing.JOptionPane;
public class Suma {
    public static void main(String[] args) {
        String primerNumero = JOptionPane.showInputDialog("Introduzca el
primer entero");
        String segundoNumero = JOptionPane.showInputDialog("Introduzca el
segundo entero");

        // convierte las entradas String en valores int para usarlos en un
cálculo
        int numero1 = Integer.parseInt(primerNumero);
        int numero2 = Integer.parseInt(segundoNumero);

        int suma = numero1 + numero2;
        // muestra los resultados en un diálogo de mensajes de JOptionPane
        JOptionPane.showMessageDialog(null, "La suma es " + suma, "Suma de
dos enteros", JOptionPane.PLAIN_MESSAGE);
    }
} // fin de la clase Suma
```

### SALIDA



## PROGRAM 3

### CODE

```
// Fig. 12.9: CampoTextoMarco.java
// Los componentes JTextField y JPasswordField.
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JOptionPane;

public class CampoTextoMarco extends JFrame {
    private final JTextField campoTexto1; // campo de texto con tamaño fijo
    private final JTextField campoTexto2; // campo de texto con texto
    private final JTextField campoTexto3; // campo de texto con texto y
    tamaño
    private final JPasswordField campoContrasenia; // campo de contraseña
    con texto

    // El constructor de CampoTextoMarco agrega objetos JTextField a JFrame
    public CampoTextoMarco() {
        super("Prueba de JTextField y JPasswordField");
        setLayout(new FlowLayout());

        // construye campo de texto con 10 columnas
        campoTexto1 = new JTextField(10);
        add(campoTexto1); // agrega campoTexto1 a JFrame

        // construye campo de texto con texto predeterminado
        campoTexto2 = new JTextField("Escriba el texto aqui");
        add(campoTexto2); // agrega campoTexto2 a JFrame

        // construye campo de texto con texto predeterminado y 21 columnas
        campoTexto3 = new JTextField("Campo de texto no editable", 21);
        campoTexto3.setEditable(false); // deshabilita la edición
        add(campoTexto3); // agrega campoTexto3 a JFrame

        // construye campo de contraseña con texto predeterminado
        campoContrasenia = new JPasswordField("Texto oculto");
        add(campoContrasenia); // agrega campoContrasenia a JFrame

        // registra los manejadores de eventos
```

```

        ManejadorCampoTexto manejador = new ManejadorCampoTexto();
        campoTexto1.addActionListener(manejador);
        campoTexto2.addActionListener(manejador);
        campoTexto3.addActionListener(manejador);
        campoContrasenia.addActionListener(manejador);
    }

    // clase interna privada para el manejo de eventos
    private class ManejadorCampoTexto implements ActionListener {
        // procesa los eventos de campo de texto
        @Override
        public void actionPerformed(ActionEvent evento) {
            String cadena = "";

            // el usuario oprimió Intro en el objeto JTextField campoTexto1
            if (evento.getSource() == campoTexto1)
                cadena = String.format("campoTexto1: %s",
                    evento.getActionCommand());

            // el usuario oprimió Intro en el objeto JTextField campoTexto2
            else if (evento.getSource() == campoTexto2)
                cadena = String.format("campoTexto2: %s",
                    evento.getActionCommand());

            // el usuario oprimió Intro en el objeto JTextField campoTexto3
            else if (evento.getSource() == campoTexto3)
                cadena = String.format("campoTexto3: %s",
                    evento.getActionCommand());

            // el usuario oprimió Intro en el objeto JTextField
            campoContrasenia
            else if (evento.getSource() == campoContrasenia)
                cadena = String.format("campoContrasenia: %s",
                    evento.getActionCommand());

            // muestra el contenido del objeto JTextField
            JOptionPane.showMessageDialog(null, cadena);
        }
    } // fin de la clase interna privada ManejadorCampoTexto
} // fin de la clase CampoTextoMarco

```

```

// Prueba de CampoTextoMarco.
import javax.swing.JFrame;

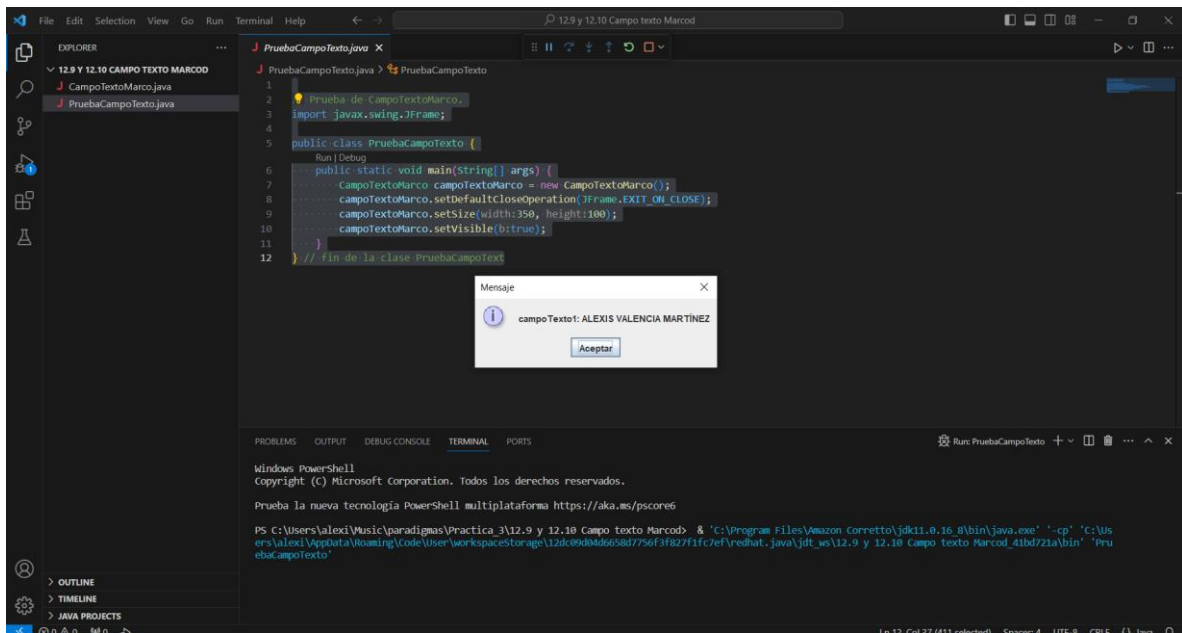
```

```

public class PruebaCampoTexto {
    public static void main(String[] args) {
        CampoTextoMarco campoTextoMarco = new CampoTextoMarco();
        campoTextoMarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        campoTextoMarco.setSize(350, 100);
        campoTextoMarco.setVisible(true);
    }
} // fin de la clase PruebaCampoText

```

## SALIDA



## PROGRAM 4

### CODE

```
// Fig. 12.15: MarcoBoton.java
// Botones de comando y eventos de acción.
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class MarcoBoton extends JFrame {
    private final JButton botonJButtonSimple; // botón con texto solamente
    private final JButton botonJButtonElegante; // botón con iconos

    // MarcoBoton agrega objetos JButton a JFrame
    public MarcoBoton() {
        super("Prueba de botones");
        setLayout(new FlowLayout());

        botonJButtonSimple = new JButton("Boton simple"); // botón con texto
        add(botonJButtonSimple); // agrega botonJButtonSimple a JFrame

        Icon boton1 = new
ImageIcon(getClass().getResource("ratonsimple.jpg"));
        Icon boton2 = new
ImageIcon(getClass().getResource("ratonelegante.jpg"));
        botonJButtonElegante = new JButton("Boton elegante", boton1); //
establece la imagen
        botonJButtonElegante.setRolloverIcon(boton2); // establece la imagen
de sustitución
        add(botonJButtonElegante); // agrega botonJButtonElegante a JFrame

        // crea nuevo ManejadorBoton para manejar los eventos de botón
        ManejadorBoton manejador = new ManejadorBoton();
        botonJButtonElegante.addActionListener(manejador);
        botonJButtonSimple.addActionListener(manejador);
    }

    // clase interna para manejar eventos de botón
    private class ManejadorBoton implements ActionListener {
```



```

        // maneja evento de botón
@Override
public void actionPerformed(ActionEvent evento) {
    JOptionPane.showMessageDialog(MarcoBoton.this, String.format(
        "Usted oprimio: %s", evento.getActionCommand()));
}
}
} // fin de la clase MarcoBoton

```

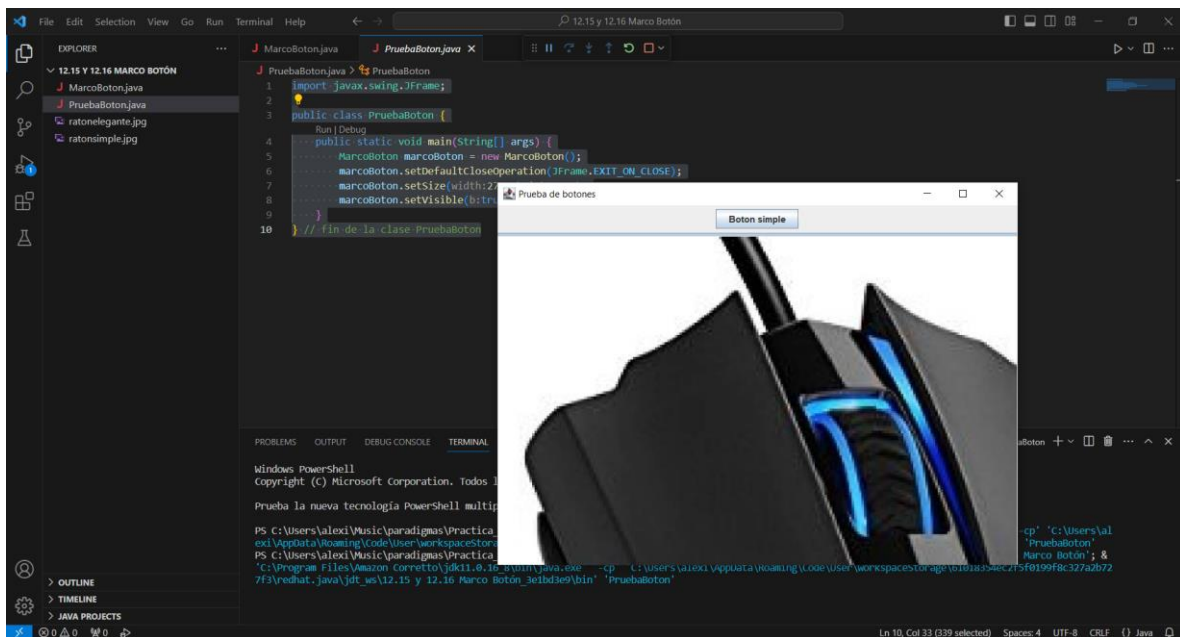
```

import javax.swing.JFrame;

public class PruebaBoton {
    public static void main(String[] args) {
        MarcoBoton marcoBoton = new MarcoBoton();
        marcoBoton.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoBoton.setSize(275, 110);
        marcoBoton.setVisible(true);
    }
} // fin de la clase PruebaBoton

```

## SALIDA



## PROGRAM 5

### CODE

```
// Fig. 12.17: MarcoCasillaVerificacion.java
// Botones JCheckBox y eventos de elementos.
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JCheckBox;

public class MarcoCasillaVerificacion extends JFrame {
    private JTextField campoTexto; // muestra el texto en tipos de letra
    cambiantes
    private JCheckBox negritaJCheckBox; // para seleccionar/deseleccionar
    negrita
    private JCheckBox cursivaJCheckBox; // para seleccionar/deseleccionar
    cursiva

    // El constructor de MarcoCasillaVerificacion agrega objetos JCheckBox a
    JFrame
    public MarcoCasillaVerificacion() {
        super("Prueba de JCheckBox");
        setLayout(new FlowLayout());

        // establece JTextField y su tipo de letra
        campoTexto = new JTextField("Observe como cambia el estilo de tipo
de letra", 20);
        campoTexto.setFont(new Font("Serif", Font.PLAIN, 14));
        add(campoTexto); // agrega campoTexto a JFrame

        negritaJCheckBox = new JCheckBox("Negrita");
        cursivaJCheckBox = new JCheckBox("Cursiva");
        add(negritaJCheckBox); // agrega casilla de verificación "negrita" a
JFrame
        add(cursivaJCheckBox); // agrega casilla de verificación "cursiva" a
JFrame

        // registra componentes de escucha para objetos JCheckBox
        ManejadorCheckBox manejador = new ManejadorCheckBox();
        negritaJCheckBox.addItemListener(manejador);
        cursivaJCheckBox.addItemListener(manejador);
    }
}
```

```

    }

    // clase interna privada para el manejo de eventos ItemListener
    private class ManejadorCheckBox implements ItemListener {

        // responde a los eventos de casilla de verificación
        @Override
        public void itemStateChanged(ItemEvent evento) {
            Font tipoletra = null; // almacena el nuevo objeto Font

            // determina cuáles objetos CheckBox están seleccionados y crea
            // el objeto Font
            if (negritaJCheckBox.isSelected() &&
                cursivaJCheckBox.isSelected())
                tipoletra = new Font("Serif", Font.BOLD + Font.ITALIC, 14);
            else if (negritaJCheckBox.isSelected())
                tipoletra = new Font("Serif", Font.BOLD, 14);
            else if (cursivaJCheckBox.isSelected())
                tipoletra = new Font("Serif", Font.ITALIC, 14);
            else
                tipoletra = new Font("Serif", Font.PLAIN, 14);

            campoTexto.setFont(tipoletra);
        }
    }
} // fin de la clase MarcoCasillaVerificacion

```

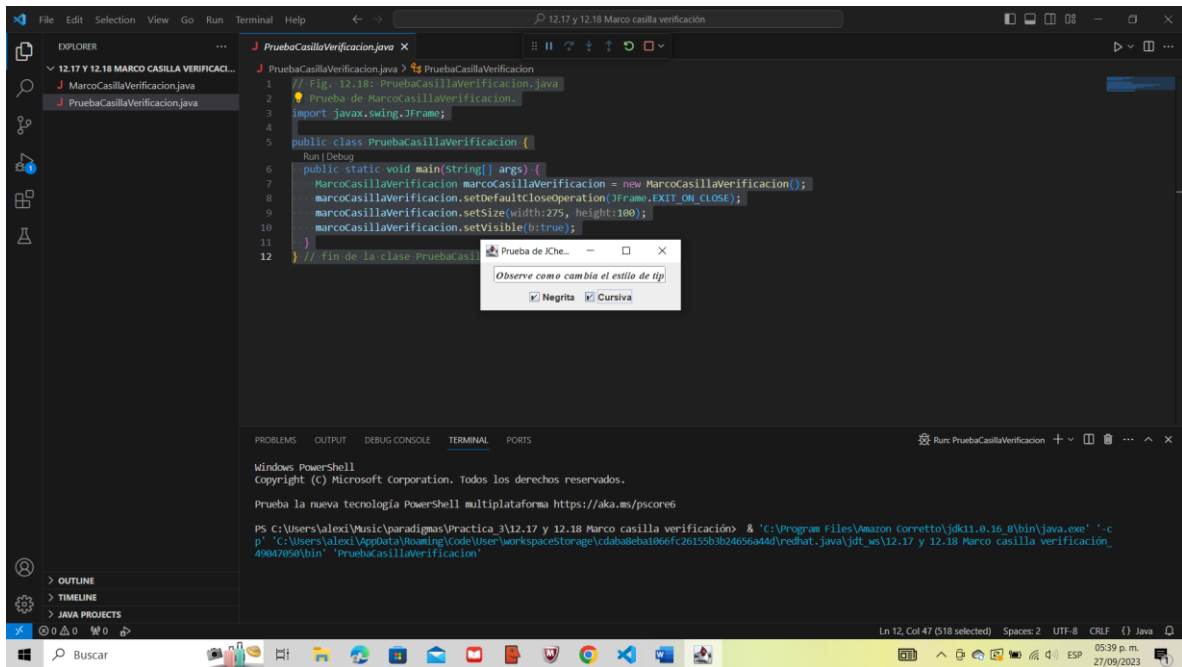
```

// Fig. 12.18: PruebaCasillaVerificacion.java
// Prueba de MarcoCasillaVerificacion.
import javax.swing.JFrame;

public class PruebaCasillaVerificacion {
    public static void main(String[] args) {
        MarcoCasillaVerificacion marcoCasillaVerificacion = new
        MarcoCasillaVerificacion();
        marcoCasillaVerificacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoCasillaVerificacion.setSize(275, 100);
        marcoCasillaVerificacion.setVisible(true);
    }
} // fin de la clase PruebaCasillaVerificacion

```

## SALIDA



The screenshot displays an IDE with the following components:

- EXPLORER:** Shows a project structure with files `MarcoCasillaVerificacion.java` and `PruebaCasillaVerificacion.java`.
- EDITOR:** Contains the source code for `PruebaCasillaVerificacion.java`. The code includes a comment about Figure 12.18, an import for `javax.swing.JFrame`, and a `main` method that creates and configures a `MarcoCasillaVerificacion` object.
- DEBUG CONSOLE:** Shows the output of the program execution, including the Windows PowerShell prompt and the command used to run the application.
- STATUS BAR:** Indicates the current line and column (Ln 12, Col 47) and the selected text.

```
1 // Fig. 12.18: PruebaCasillaVerificacion.java
2 // Prueba de MarcoCasillaVerificacion.
3 import javax.swing.JFrame;
4
5 public class PruebaCasillaVerificacion {
6
7     public static void main(String[] args) {
8         MarcoCasillaVerificacion marcoCasillaVerificacion = new MarcoCasillaVerificacion();
9         marcoCasillaVerificacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10        marcoCasillaVerificacion.setSize(width:275, height:100);
11        marcoCasillaVerificacion.setVisible(b:true);
12    } // fin de la clase PruebaCasillaVerificacion
```

Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\alexi\AppData\Local\Programs\Practica\_3\12.17 y 12.18 Marco casilla verificación & 'C:\Program Files\Amazon Corretto\jdk11.0.16\_8\bin\java.exe' '-c  
p' 'C:\Users\alexi\AppData\Local\Programs\Practica\_3\12.17 y 12.18 Marco casilla verificación\_49047650\bin' 'PruebaCasillaVerificacion'

## PROGRAM 6

### CODE

```
// Fig. 12.19: MarcoBotonOpcion.java
// Creación de botones de opción, usando ButtonGroup y JRadioButton.
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;

public class MarcoBotonOpcion extends JFrame {
    private final JTextField campoTexto; // se utiliza para mostrar los
    cambios en el tipo de letra
    private final Font tipoLetraSimple; // tipo de letra para texto simple
    private final Font tipoLetraNegrita; // tipo de letra para texto en
    negrita
    private final Font tipoLetraCursiva; // tipo de letra para texto en
    cursiva
    private final Font tipoLetraNegritaCursiva; // tipo de letra para texto
    en negrita y cursiva
    private final JRadioButton simpleJRadioButton; // selecciona texto
    simple
    private final JRadioButton negritaJRadioButton; // selecciona texto en
    negrita
    private final JRadioButton cursivaJRadioButton; // selecciona texto en
    cursiva
    private final JRadioButton negritaCursivaJRadioButton; // negrita y
    cursiva
    private ButtonGroup grupoOpciones; // contiene los botones de opción

    // El constructor de MarcoBotonOpcion agrega los objetos JRadioButton a
    JFrame
    public MarcoBotonOpcion() {
        super("Prueba de RadioButton");
        setLayout(new FlowLayout());

        campoTexto = new JTextField(" Observe el cambio en el estilo del
    tipo de letra", 25);
        add(campoTexto); // agrega campoTexto a JFrame
    }
}
```

```

        // crea los botones de opción
        simpleJRadioButton = new JRadioButton("Simple", true);
        negritaJRadioButton = new JRadioButton("Negrita", false);
        cursivaJRadioButton = new JRadioButton("Cursiva", false);
        negritaCursivaJRadioButton = new JRadioButton("Negrita/Cursiva",
false);

        add(simpleJRadioButton); // agrega botón simple a JFrame
        add(negritaJRadioButton); // agrega botón negrita a JFrame
        add(cursivaJRadioButton); // agrega botón cursiva a JFrame
        add(negritaCursivaJRadioButton); // agrega botón negrita y cursiva

        // crea una relación lógica entre los objetos JRadioButton
        grupoOpciones = new ButtonGroup(); // crea ButtonGroup
        grupoOpciones.add(simpleJRadioButton); // agrega simple al grupo
        grupoOpciones.add(negritaJRadioButton); // agrega negrita al grupo
        grupoOpciones.add(cursivaJRadioButton); // agrega cursiva al grupo
        grupoOpciones.add(negritaCursivaJRadioButton); // agrega negrita y
cursiva

        // crea objetos tipo de letra
        tipoLetraSimple = new Font("Serif", Font.PLAIN, 14);
        tipoLetraNegrita = new Font("Serif", Font.BOLD, 14);
        tipoLetraCursiva = new Font("Serif", Font.ITALIC, 14);
        tipoLetraNegritaCursiva = new Font("Serif", Font.BOLD + Font.ITALIC,
14);

        campoTexto.setFont(tipoLetraSimple);

        // registra eventos para los objetos JRadioButton
        simpleJRadioButton.addItemListener(
            new ManejadorBotonOpcion(tipoLetraSimple));
        negritaJRadioButton.addItemListener(
            new ManejadorBotonOpcion(tipoLetraNegrita));
        cursivaJRadioButton.addItemListener(
            new ManejadorBotonOpcion(tipoLetraCursiva));
        negritaCursivaJRadioButton.addItemListener(
            new ManejadorBotonOpcion(tipoLetraNegritaCursiva));
    }

    // clase interna privada para manejar eventos de botones de opción
    private class ManejadorBotonOpcion implements ItemListener {
        private Font tipoLetra; // tipo de letra asociado con este
componente de escucha

        public ManejadorBotonOpcion(Font f) {
            tipoLetra = f;

```

```

    }

    // maneja los eventos de botones de opción
    @Override
    public void itemStateChanged(ItemEvent evento) {
        campoTexto.setFont(tipoLetra);
    }
}
} // fin de la clase MarcoBotonOpcion

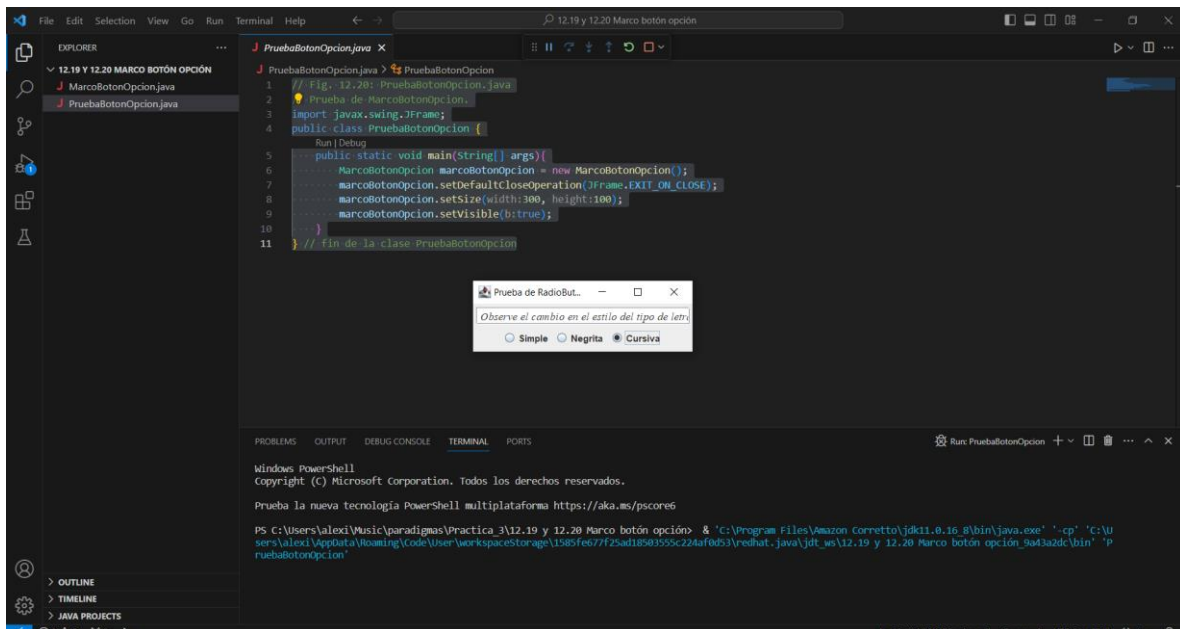
```

```

// Fig. 12.20: PruebaBotonOpcion.java
// Prueba de MarcoBotonOpcion.
import javax.swing.JFrame;
public class PruebaBotonOpcion {
    public static void main(String[] args){
        MarcoBotonOpcion marcoBotonOpcion = new MarcoBotonOpcion();
        marcoBotonOpcion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoBotonOpcion.setSize(300, 100);
        marcoBotonOpcion.setVisible(true);
    }
} // fin de la clase PruebaBotonOpcion

```

## SALIDA



## PROGRAM 7

### CODE

```
// Fig. 12.21: MarcoCuadroCombinado.java
// Objeto JComboBox que muestra una lista de nombres de imágenes.
import java.awt.FlowLayout;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.Icon;
import javax.swing.ImageIcon;

public class MarcoCuadroCombinado extends JFrame {
    private final JComboBox<String> imagenesJComboBox; // contiene los
nombres de los iconos
    private final JLabel etiqueta; // muestra el icono seleccionado

    private static final String nombres[] = { "gato1.jpg", "gato2.jpg",
"gato3.jpg", "gato4.jpg" };
    private final Icon[] iconos = {
        new ImageIcon(getClass().getResource(nombres[0])),
        new ImageIcon(getClass().getResource(nombres[1])),
        new ImageIcon(getClass().getResource(nombres[2])),
        new ImageIcon(getClass().getResource(nombres[3])) };

    // El constructor de MarcoCuadroCombinado agrega un objeto JComboBox a
JFrame
    public MarcoCuadroCombinado() {
        super("Prueba de JComboBox");
        setLayout(new FlowLayout()); // establece el esquema del marco

        imagenesJComboBox = new JComboBox<String>(nombres); // establece
JComboBox
        imagenesJComboBox.setMaximumRowCount(3); // muestra tres filas

        imagenesJComboBox.addItemListener(
            new ItemListener() // clase interna anónima
            {
                // maneja evento de JComboBox
                @Override
                public void itemStateChanged(ItemEvent evento) {
                    // determina si está seleccionado el elemento
                }
            }
        );
    }
}
```



```

        if (evento.getStateChange() == ItemEvent.SELECTED)
            etiqueta.setIcon(iconos[imagenesJComboBox.getSelectedIndex()]);
    }
} // fin de la clase interna anónima
); // fin de la llamada a addItemListener

add(imagenesJComboBox); // agrega cuadro combinado a JFrame
etiqueta = new JLabel(iconos[0]); // muestra el primer icono
add(etiqueta); // agrega etiqueta a JFrame
}
} // fin de la clase MarcoCuadroCombinado

```

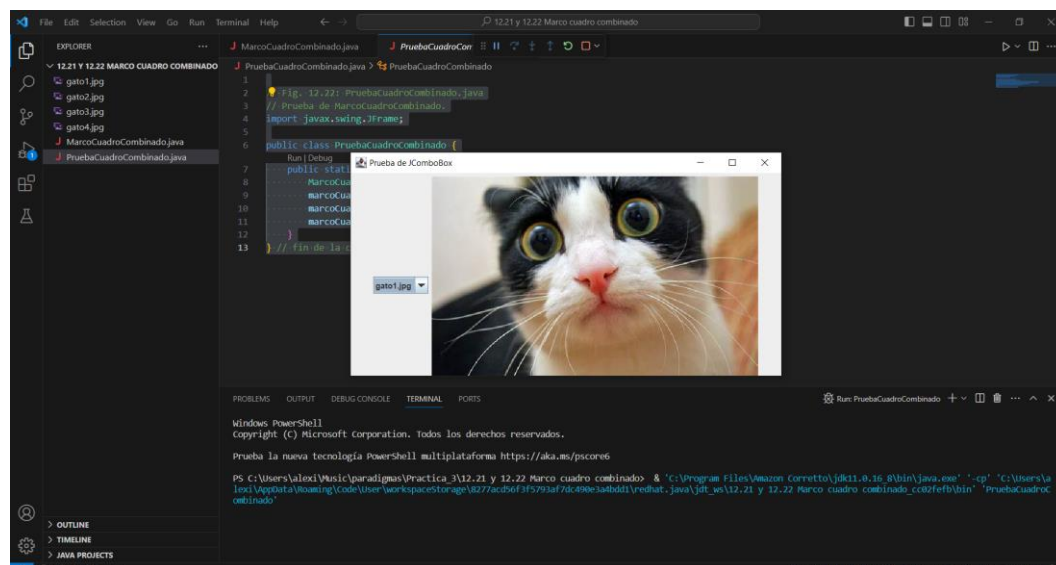
```

// Fig. 12.22: PruebaCuadroCombinado.java
// Prueba de MarcoCuadroCombinado.
import javax.swing.JFrame;

public class PruebaCuadroCombinado {
    public static void main(String[] args) {
        MarcoCuadroCombinado marcoCuadroCombinado = new
MarcoCuadroCombinado();
        marcoCuadroCombinado.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoCuadroCombinado.setSize(350, 150);
        marcoCuadroCombinado.setVisible(true);
    }
} // fin de la clase PruebaCuadroCombinado

```

## SALIDA



## PROGRAM 8

### CODE

```
// Fig. 12.23: MarcoLista.java
// Objeto JList que muestra una lista de colores.
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.ListSelectionModel;

public class MarcoLista extends JFrame {
    private final JList<String> listaJListColores; // lista para mostrar
    colores
    private static final String[] nombresColores = { "Negro", "Azul",
    "Cyan",
        "Gris oscuro", "Gris", "Verde", "Gris claro", "Magenta",
        "Naranja", "Rosa", "Rojo", "Blanco", "Amarillo" };
    private static final Color[] colores = { Color.BLACK, Color.BLUE,
    Color.CYAN, Color.DARK_GRAY, Color.GRAY, Color.GREEN,
    Color.LIGHT_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK,
    Color.RED, Color.WHITE, Color.YELLOW };

    // El constructor de MarcoLista agrega a JFrame el JScrollPane que
    contiene a
    // JList
    public MarcoLista() {
        super("Prueba de JList");
        setLayout(new FlowLayout());

        listaJListColores = new JList<String>(nombresColores); // lista de
    nombresColores
        listaJListColores.setVisibleRowCount(5); // muestra cinco filas a la
    vez

        // no permite selecciones múltiples
        listaJListColores.setSelectionMode(ListSelectionModel.SINGLE_SELECTI
    ON);

        // agrega al marco un objeto JScrollPane que contiene a JList
        add(new JScrollPane(listaJListColores));
    }
}
```

```
// Fig. 12.24: PruebaLista.java
// Selección de colores de un objeto JList.
import javax.swing.JFrame;

public class PruebaLista {
    public static void main(String[] args) {
        MarcoLista marcoLista = new MarcoLista(); // crea objeto MarcoLista
        marcoLista.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoLista.setSize(350, 150);
        marcoLista.setVisible(true);
    }
} // fin de la clase PruebaList
```

The screenshot shows an IDE with the following components:

- File Explorer (Left):** Shows a project structure with files like `12.23 y 12.24 MARCO LISTA`, `MarcoLista.java`, and `PruebaLista.java`.
- Code Editor (Center):** Displays the source code for `PruebaLista.java`. The code includes imports for `java.awt.Color` and `java.util.ArrayList`, and defines a `JList` with five color elements.
- Terminal (Bottom):** Shows the command prompt output, including the command to run the application and the resulting list of colors.
- Prueba de JList Window:** A small window titled "Prueba de JList" is open, showing a list of five color swatches: Verde, Gris claro, Magenta, Naranja, and Rojo.

## PROGRAM 9

### CODE

```
// Fig. 12.25: MarcoSeleccionMultiple.java
// Objeto JList que permite selecciones múltiples.
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JButton;
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;

public class MarcoSeleccionMultiple extends JFrame {
    private final JList<String> listaJListColores; // lista para guardar los
nombres los colores
    private final JList<String> listaJListCopia; // lista en la que se van a
copiarlos nombres de los colores
    private JButton botonJButtonCopiar; // botón para copiar los nombres
seleccionados
    private static final String[] nombresColores = { "Negro", "Azul",
"Cyan", "Gris oscuro", "Gris", "Verde",
        "Gris claro", "Magenta", "Naranja", "Rosa", "Rojo", "Blanco",
"Amarillo" };

    // Constructor de MarcoSeleccionMultiple
    public MarcoSeleccionMultiple() {
        super("Listas de seleccion multiple");
        setLayout(new FlowLayout());

        listaJListColores = new JList<String>(nombresColores); // lista de
nombres de colores
        listaJListColores.setVisibleRowCount(5); // muestra cinco filas
        listaJListColores.setSelectionMode(
            ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
        add(new JScrollPane(listaJListColores)); // agrega lista con panel
de desplazamiento

        botonJButtonCopiar = new JButton("Copiar >>>");
        botonJButtonCopiar.addActionListener(
            new ActionListener() // clase interna anónima
            {
                // maneja evento de botón
                @Override
```

```

        public void actionPerformed(ActionEvent evento) {
            // coloca los valores seleccionados en
            listaJListCopia
                listaJListCopia.setListData(
                    listaJListColores.getSelectedValuesList().to
                    Array(
                        new String[0]));
        }
    });

    add(botonJButtonCopiar); // agrega el botón copiar a JFrame

    listaJListCopia = new JList<String>(); // lista para guardar nombres
    de colores copiados
    listaJListCopia.setVisibleRowCount(5); // muestra 5 filas
    listaJListCopia.setFixedCellWidth(100); // establece la anchura
    listaJListCopia.setFixedCellHeight(15); // establece la altura
    listaJListCopia.setSelectionMode(
        ListSelectionModel.SINGLE_INTERVAL_SELECTION);
    add(new JScrollPane(listaJListCopia)); // agrega lista con panel de
    desplazamiento
    }
} // fin de la clase MarcoSeleccionMultiple

```

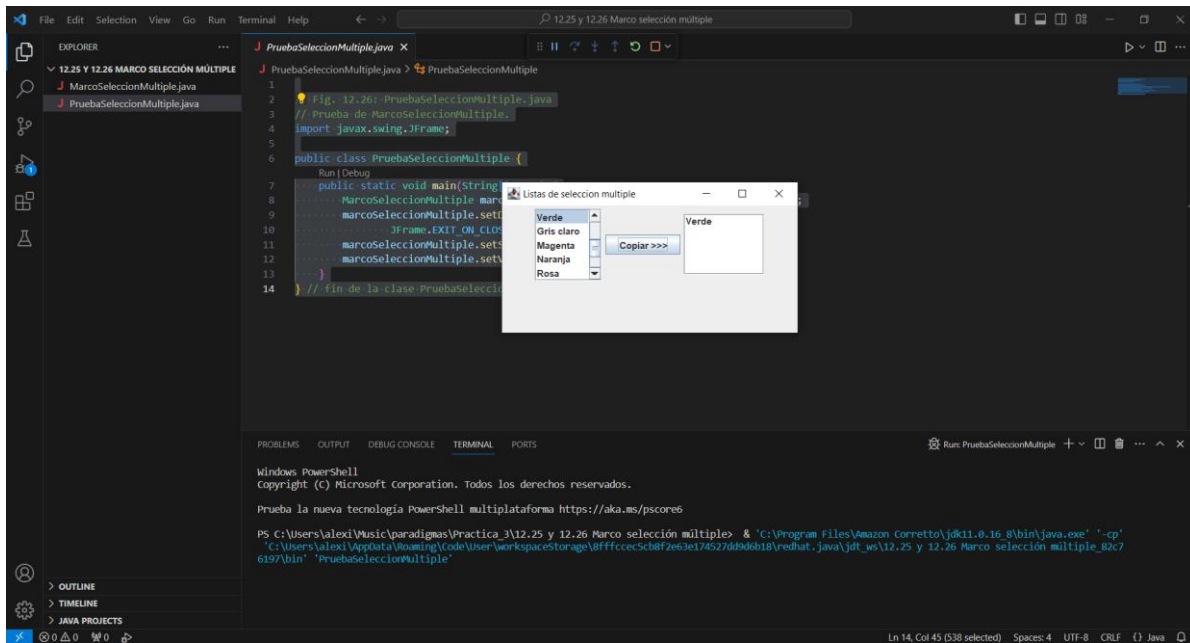
```

// Fig. 12.26: PruebaSeleccionMultiple.java
// Prueba de MarcoSeleccionMultiple.
import javax.swing.JFrame;

public class PruebaSeleccionMultiple {
    public static void main(String[] args) {
        MarcoSeleccionMultiple marcoSeleccionMultiple = new
        MarcoSeleccionMultiple();
        marcoSeleccionMultiple.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);
        marcoSeleccionMultiple.setSize(350, 140);
        marcoSeleccionMultiple.setVisible(true);
    }
} // fin de la clase PruebaSeleccionMultiple

```

## SALIDA



## PROGRAM 10

### CÓDIGO

```
// Fig. 12.28: MarcoRastreadorRaton.java
// Manejo de eventos de ratón.
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class MarcoRastreadorRaton extends JFrame {
    private final JPanel panelRaton; // panel en el que ocurrirán los
    eventos de ratón
    private final JLabel barraEstado; // muestra información de los
    eventos

    // El constructor de MarcoRastreadorRaton establece la GUI y
    // registra los manejadores de eventos de ratón
    public MarcoRastreadorRaton() {
        super("Demostracion de los eventos de raton");

        panelRaton = new JPanel();
        panelRaton.setBackground(Color.WHITE);
        add(panelRaton, BorderLayout.CENTER); // agrega el panel a
JFrame

        barraEstado = new JLabel("Raton fuera de JPanel");
        add(barraEstado, BorderLayout.SOUTH); // agrega etiqueta a
JFrame

        // crea y registra un componente de escucha para los eventos
de ratón y de su
        // movimiento
        ManejadorRaton manejador = new ManejadorRaton();
        panelRaton.addMouseListener(manejador);
        panelRaton.addMouseMotionListener(manejador);
    }

    private class ManejadorRaton implements MouseListener,
        MouseMotionListener {
```

```

        // Los manejadores de eventos de MouseListener
        // manejan el evento cuando se suelta el ratón justo después
de oprimir el botón
        @Override
        public void mouseClicked(MouseEvent evento) {
            barraEstado.setText(String.format("Se hizo clic en [%d,
%d]",
                                                evento.getX(), evento.getY()));
        }

        // maneja evento cuando se oprime el ratón
        @Override
        public void mousePressed(MouseEvent evento) {
            barraEstado.setText(String.format("Se oprimio en [%d,
%d]",
                                                evento.getX(), evento.getY()));
        }

        // maneja evento cuando se suelta el botón del ratón
        @Override
        public void mouseReleased(MouseEvent evento) {
            barraEstado.setText(String.format("Se solto en [%d,
%d]",
                                                evento.getX(), evento.getY()));
        }

        // maneja evento cuando el ratón entra al área
        @Override
        public void mouseEntered(MouseEvent evento) {
            barraEstado.setText(String.format("Raton entro en [%d,
%d]",
                                                evento.getX(), evento.getY()));
            panelRaton.setBackground(Color.GREEN);
        }

        // maneja evento cuando el ratón sale del área
        @Override
        public void mouseExited(MouseEvent evento) {
            barraEstado.setText("Raton fuera de JPanel");
            panelRaton.setBackground(Color.WHITE);
        }

        // Los manejadores de eventos de MouseMotionListener manejan
        // el evento cuando el usuario arrastra el ratón con el botón
oprimido

```



```

        @Override
        public void mouseDragged(MouseEvent evento) {
            barraEstado.setText(String.format("Se arrastro en [%d,
%d]",
                                            evento.getX(), evento.getY()));
        }

        // maneja evento cuando el usuario mueve el ratón
        @Override
        public void mouseMoved(MouseEvent evento) {
            barraEstado.setText(String.format("Se movio en [%d,
%d]",
                                            evento.getX(), evento.getY()));
        }
    } // fin de la clase interna ManejadorRaton
} // fin de la clase MarcoRastreadorRaton

```

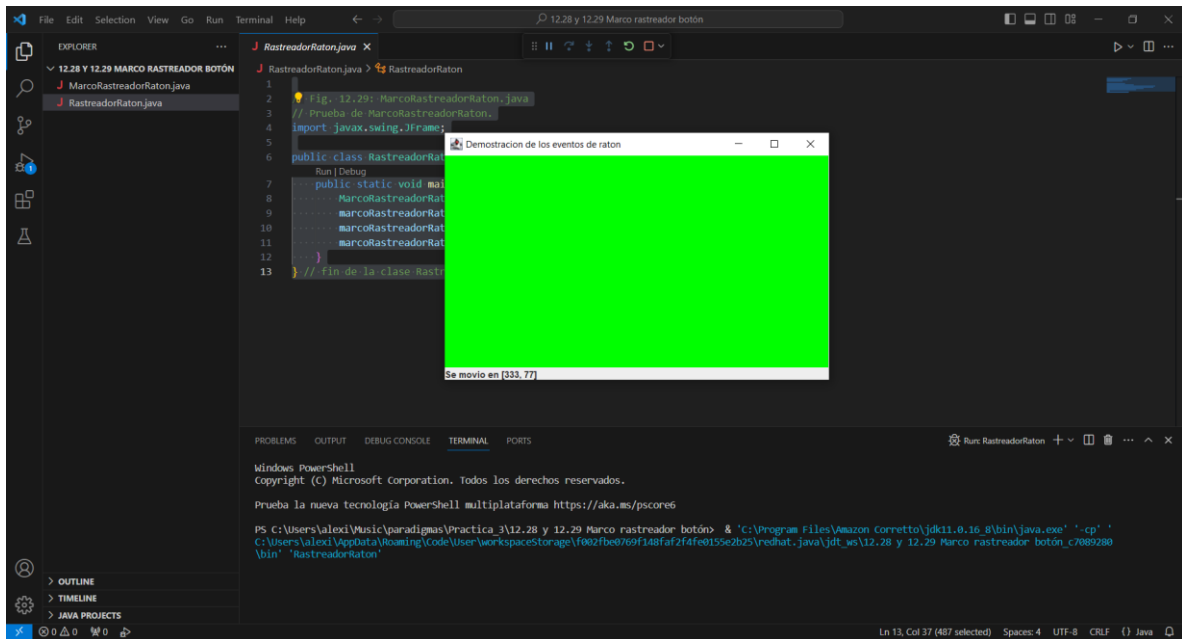
```

// Fig. 12.29: MarcoRastreadorRaton.java
// Prueba de MarcoRastreadorRaton.
import javax.swing.JFrame;

public class RastreadorRaton {
    public static void main(String[] args) {
        MarcoRastreadorRaton marcoRastreadorRaton = new
MarcoRastreadorRaton();
        marcoRastreadorRaton.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoRastreadorRaton.setSize(300, 100);
        marcoRastreadorRaton.setVisible(true);
    }
} // fin de la clase RastreadorRaton

```

## SALIDA



The screenshot shows an IDE with the following components:

- EXPLORER:** Lists files for '12.28 Y 12.29 MARCO RASTREADOR BOTÓN', including 'MarcoRastreadorRaton.java' and 'RastreadorRaton.java'.
- EDITOR:** Displays the code for 'RastreadorRaton.java'. The code includes a comment 'Fig. 12.29: MarcoRastreadorRaton.java', an import for 'javax.swing.JFrame', and a class definition for 'RastreadorRaton' with a 'main' method. A red squiggly line indicates an error at line 13.
- DEBUG CONSOLE:** Shows the output of the program, which is a solid green rectangle. The title bar of the window is 'Demostración de los eventos de ratón'. A status bar at the bottom of the window says 'Se movió en (335, 77)'.
- TERMINAL:** Displays the command prompt output, showing the execution of the Java program using 'java.exe'.

```
1 // Fig. 12.29: MarcoRastreadorRaton.java
2 // Prueba de MarcoRastreadorRaton.
3 import javax.swing.JFrame;
4
5
6 public class RastreadorRaton {
7     public static void main(String[] args) {
8         MarcoRastreadorRaton marcoRastreadorRaton =
9             new MarcoRastreadorRaton();
10        marcoRastreadorRaton.marcoRastreadorRaton();
11    }
12 }
13 // fin de la clase RastreadorRaton
```

Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\Users\alexi\Music\paradigmas\Practica\_3\12.28 y 12.29 Marco rastreador botón> & 'C:\Program Files\Amazon Corretto\jdk11.0.16.8\bin\java.exe' '-cp' 'C:\Users\alexi\AppData\Roaming\Code\User\workspaceStorage\1002fbd0f9f148f2f4fe0155c2b25\redhat\_java\jdk\_ws\12.28 y 12.29 Marco rastreador botón\_c7089280\bin' 'RastreadorRaton'

Ln 13, Col 37 (487 selected) Spaces: 4 UTF-8 CRLF {} Java

## PROGRAM 11

### CODE

```
// Fig. 12.32: DetallesRaton.java
// Prueba de MarcoDetallesRaton.
import javax.swing.JFrame;

public class DetallesRaton {
    public static void main(String[] args) {
        MarcoDetallesRaton marcoDetallesRaton = new MarcoDetallesRaton();
        marcoDetallesRaton.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoDetallesRaton.setSize(400, 150);
        marcoDetallesRaton.setVisible(true);
    }
} // fin de la clase DetallesRaton
```

```
// Fig. 12.31: MarcoDetallesRaton.java
// Demostración de los clics del ratón y cómo diferenciar los botones del
// mismo.
import java.awt.BorderLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class MarcoDetallesRaton extends JFrame {
    private String detalles; // String que se muestra en barraEstado
    private final JLabel barraEstado; // JLabel que aparece en la parte
    // inferior de la ventana

    // constructor establece String de la barra de título y registra
    // componente de
    // escucha del ratón
    public MarcoDetallesRaton() {
        super("Clics y botones del raton");

        barraEstado = new JLabel("Haga clic en el raton");
        add(barraEstado, BorderLayout.SOUTH);
        addMouseListener(new ManejadorClicRaton()); // agrega el manejador
    }

    // clase interna para manejar los eventos del ratón
```

```

private class ManejadorClicRaton extends MouseAdapter {
    // maneja evento de clic del ratón y determina cuál botón se oprimió
    @Override
    public void mouseClicked(MouseEvent evento) {
        int xPos = evento.getX(); // obtiene posición x del ratón
        int yPos = evento.getY(); // obtiene posición y del ratón

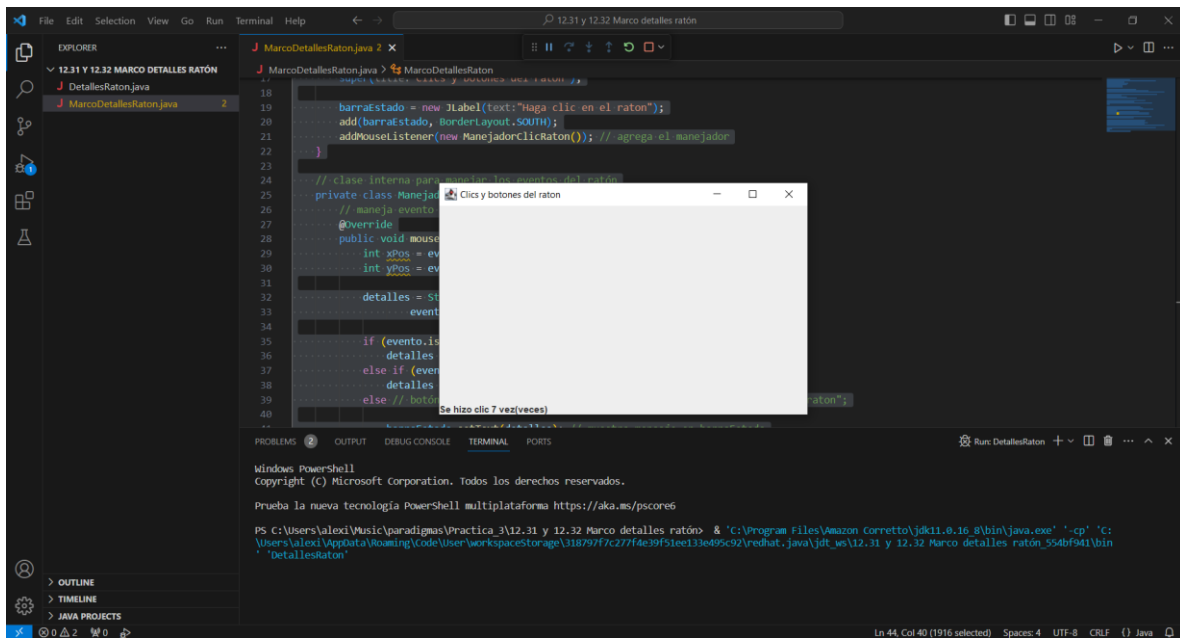
        detalles = String.format("Se hizo clic %d vez(veces)",
            evento.getClickCount());

        if (evento.isMetaDown()) // botón derecho del ratón
            detalles += " con el boton derecho del raton";
        else if (evento.isAltDown()) // botón central del ratón
            detalles += " con el boton central del raton";
        else // botón izquierdo del ratón
            detalles += " con el boton izquierdo del raton";

        barraEstado.setText(detalles); // muestra mensaje en barraEstado
    }
}
} // fin de la clase MarcoDetallesRaton

```

## SALIDA



## PROGRAM 12

### CODE

```
// Fig. 12.34: PanelDibujo.java
// Clase adaptadora que se usa para implementar manejadores de eventos.
import java.awt.Point;
import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.util.ArrayList;
import javax.swing.JPanel;

public class PanelDibujo extends JPanel {
    // lista de referencias Point
    private final ArrayList<Point> puntos = new ArrayList<>();

    // establece la GUI y registra el manejador de eventos del ratón
    public PanelDibujo() {
        // maneja evento de movimiento del ratón en el marco
        addMouseMotionListener(
            new MouseMotionAdapter() // clase interna anónima
            {
                // almacena las coordenadas de arrastre y vuelve a
                dibujar

                @Override
                public void mouseDragged(MouseEvent evento) {
                    puntos.add(evento.getPoint());
                    repaint(); // vuelve a dibujar JFrame
                }
            }
        );
    }

    // dibuja óvalos en un cuadro delimitador de 4 x 4, en las ubicaciones
    // especificadas en la ventana
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g); // borra el área de dibujo

        // dibuja todos los puntos
        for (Point punto : puntos)
            g.fillOval(punto.x, punto.y, 4, 4);
    }
} // fin de la clase PanelDibujo
```

```
// Fig. 12.35: Pintor.java
// Prueba de PanelDibujo.
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;

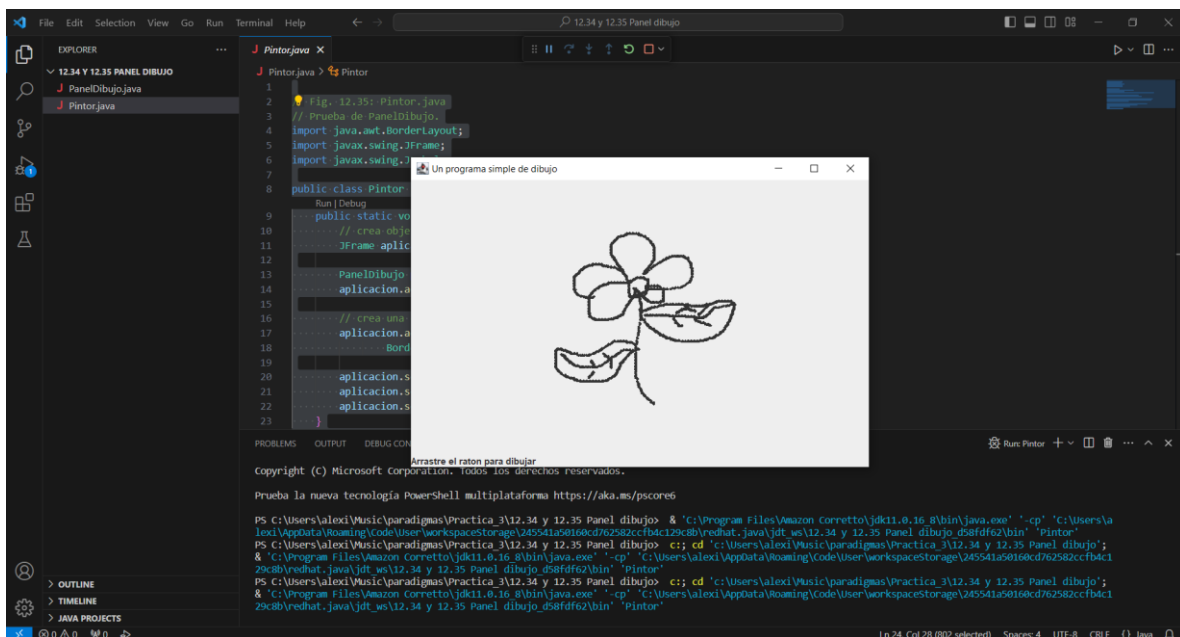
public class Pintor {
    public static void main(String[] args) {
        // crea objeto JFrame
        JFrame aplicacion = new JFrame("Un programa simple de dibujo");

        PanelDibujo panelDibujo = new PanelDibujo();
        aplicacion.add(panelDibujo, BorderLayout.CENTER);

        // crea una etiqueta y la coloca en la región SOUTH de BorderLayout
        aplicacion.add(new JLabel("Arrastre el raton para dibujar"),
            BorderLayout.SOUTH);

        aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        aplicacion.setSize(400, 200);
        aplicacion.setVisible(true);
    }
} // fin de la clase Pintor
```

## SALIDA



## PROGRAM 13

### CODE

```
// Fig. 12.37: DemoTeclas.java
// Prueba de MarcoDemoTeclas.
import javax.swing.JFrame;

public class DemoTeclas {
    public static void main(String[] args) {
        MarcoDemoTeclas marcoDemoTeclas = new MarcoDemoTeclas();
        marcoDemoTeclas.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoDemoTeclas.setSize(350, 100);
        marcoDemoTeclas.setVisible(true);
    }
} // fin de la clase DemoTeclas
```

```
// Fig. 12.36: MarcoDemoTeclas.java
// Manejo de eventos de teclas.
import java.awt.Color;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.JTextArea;

public class MarcoDemoTeclas extends JFrame implements KeyListener {
    private String linea1 = ""; // primera línea del área de texto
    private String linea2 = ""; // segunda línea del área de texto
    private String linea3 = ""; // tercera línea del área de texto
    private JTextArea areaTexto; // área de texto para mostrar la salida

    // constructor de MarcoDemoTeclas
    public MarcoDemoTeclas() {
        super("Demostracion de los eventos de pulsacion de teclas");

        areaTexto = new JTextArea(10, 15); // establece el objeto JTextArea
        areaTexto.setText("Oprima cualquier tecla en el teclado...");
        areaTexto.setEnabled(false);
        areaTexto.setDisabledTextColor(Color.BLACK);
        add(areaTexto); // agrega el área de texto a JFrame

        addKeyListener(this); // permite al marco procesar los eventos de
teclas
```

```

    }

    // maneja el evento de oprimir cualquier tecla
    @Override
    public void keyPressed(KeyEvent evento) {
        linea1 = String.format("Tecla oprimida: %s",
            KeyEvent.getKeyText(evento.getKeyCode())); // muestra la
tecla oprimida
        establecerLineas2y3(evento); // establece las líneas de salida dos y
tres
    }

    // maneja el evento de liberar cualquier tecla
    @Override
    public void keyReleased(KeyEvent evento) {
        linea1 = String.format("Tecla liberada: %s",
            KeyEvent.getKeyText(evento.getKeyCode())); // muestra la
tecla liberada
        establecerLineas2y3(evento); // establece las líneas de salida dos y
tres
    }

    // maneja el evento de oprimir una tecla de acción
    @Override
    public void keyTyped(KeyEvent evento) {
        linea1 = String.format("Tecla oprimida: %s", evento.getKeyChar());
        establecerLineas2y3(evento); // establece las líneas de salida dos y
tres
    }

    // establece las líneas de salida dos y tres
    private void establecerLineas2y3(KeyEvent evento) {
        linea2 = String.format("Esta tecla %s es una tecla de accion",
            (evento.isActionKey() ? "" : "no "));

        String temp = KeyEvent.getKeyModifiersText(evento.getModifiers());

        linea3 = String.format("Teclas modificadoras oprimidas: %s",
            (temp.equals("") ? "ninguna" : temp)); // imprime
modificadoras

        areaTexto.setText(String.format("%s\n%s\n%s\n",
            linea1, linea2, linea3)); // imprime tres líneas de texto
    }
} // fin de la clase MarcoDemoTeclas

```



## SALIDA

The screenshot shows an IDE with the following components:

- EXPLORER:** Shows a project named "12.36 Y 12.37 MARCO DEMO TECLAS" with files "DemoTeclas.java" and "MarcoDemoTeclas.java".
- EDITOR:** Displays the code for "MarcoDemoTeclas.java". The code includes imports for Java Swing and AWT, and defines a class "MarcoDemoTeclas" with a constructor and a text area.
- OUTPUT:** Shows the execution output of the program. It displays the text "Tecla liberada: A" and "Esta tecla no es una tecla de accion".
- TERMINAL:** Shows the command prompt output, including the command to run the program and the resulting output.

```
1 // Fig. 12.36: MarcoDemoTeclas.java
2 // Manejo de eventos de teclas.
3 import java.awt.Color;
4 import java.awt.event.KeyListener;
5 import java.awt.event.KeyEvent;
6 import javax.swing.JFrame;
7 import javax.swing.JTextArea;
8
9 public class MarcoDemoTeclas {
10     private String llave;
11     private String llaveModificadora;
12     private String llaveModificadoraOprimida;
13     private JTextArea areaTexto;
14
15     // constructor de la clase
16     public MarcoDemoTeclas() {
17         super("Marco Demo Teclas");
18
19         areaTexto = new JTextArea(10, 20);
20         areaTexto.setLineWrap(true);
21         areaTexto.setWrapStyleWord(true);
22         areaTexto.setDisabledTextColor(Color.BLACK);
23         add(areaTexto); // agrega el area de texto a JFrame
24     }
25 }
```

Output:

```
Tecla liberada: A
Esta tecla no es una tecla de accion
Teclas modificadoras oprimidas: ninguna
```

Terminal:

```
PS C:\Users\alexi\Music\paradigmas\Practica_3\12.36 y 12.37 Marco demo teclas> & 'C:\Program Files\Amazon Corretto\jdk11.0.16_8\bin\java.exe' '-cp' 'C:\Users\alexi\AppData\Roaming\Code\User\workspacestorage\25306ea0586d309b38fa113af6952aff\redhat.java\jdk_11\12.36 y 12.37 Marco demo teclas_e92fd8ae\bin' "DemoTeclas"
```

## PROGRAM 14

### CODE

```
// Fig. 12.40: DemoFlowLayout.java
// Prueba MarcoFlowLayout.
import javax.swing.JFrame;

public class DemoFlowLayout {
    public static void main(String[] args) {
        MarcoFlowLayout marcoFlowLayout = new MarcoFlowLayout();
        marcoFlowLayout.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoFlowLayout.setSize(300, 75);
        marcoFlowLayout.setVisible(true);
    }
} // fin de la clase DemoFlowLayou
```

```
// Fig. 12.39: MarcoFlowLayout.java
// FlowLayout permite que los componentes fluyan a través de varias líneas.
import java.awt.FlowLayout;
import java.awt.Container;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class MarcoFlowLayout extends JFrame {
    private final JButton botonJButtonIzquierda; // botón para establecer la
    alineación a la izquierda
    private final JButton botonJButtonCentro; // botón para establecer la
    alineación al centro
    private final JButton botonJButtonDerecha; // botón para establecer la
    alineación a la derecha
    private final FlowLayout esquema; // objeto esquema
    private final Container contenedor; // contenedor para establecer el
    esquema

    // establece la GUI y registra los componentes de escucha de botones
    public MarcoFlowLayout() {
        super("Demostracion de FlowLayout");

        esquema = new FlowLayout();
        contenedor = getContentPane(); // obtiene contenedor para esquema
    }
}
```

```

setLayout(esquema);

// establece botonJButtonIzquierda y registra componente de escucha
botonJButtonIzquierda = new JButton("Izquierda");
add(botonJButtonIzquierda); // agrega botón Izquierda al marco
botonJButtonIzquierda.addActionListener(
    new ActionListener() // clase interna anónima
    {
        // procesa evento de botonJButtonIzquierda
        @Override
        public void actionPerformed(ActionEvent evento) {
            esquema.setAlignment(FlowLayout.LEFT);

            // realinea los componentes adjuntos
            esquema.layoutContainer(contenedor);
        }
    });

// establece botonJButtonCentro y registra componente de escucha
botonJButtonCentro = new JButton("Centro");
add(botonJButtonCentro); // agrega botón Centro al marco
botonJButtonCentro.addActionListener(
    new ActionListener() // clase interna anónima
    {
        // procesa evento de botonJButtonCentro
        @Override
        public void actionPerformed(ActionEvent evento) {
            esquema.setAlignment(FlowLayout.CENTER);

            // realinea los componentes adjuntos
            esquema.layoutContainer(contenedor);
        }
    });

// establece botonJButtonDerecha y registra componente de escucha
botonJButtonDerecha = new JButton("Derecha");
add(botonJButtonDerecha); // agrega botón Derecha al marco
botonJButtonDerecha.addActionListener(
    new ActionListener() // clase interna anónima
    {
        // procesa evento de botonJButtonDerecha
        @Override
        public void actionPerformed(ActionEvent evento) {
            esquema.setAlignment(FlowLayout.RIGHT);

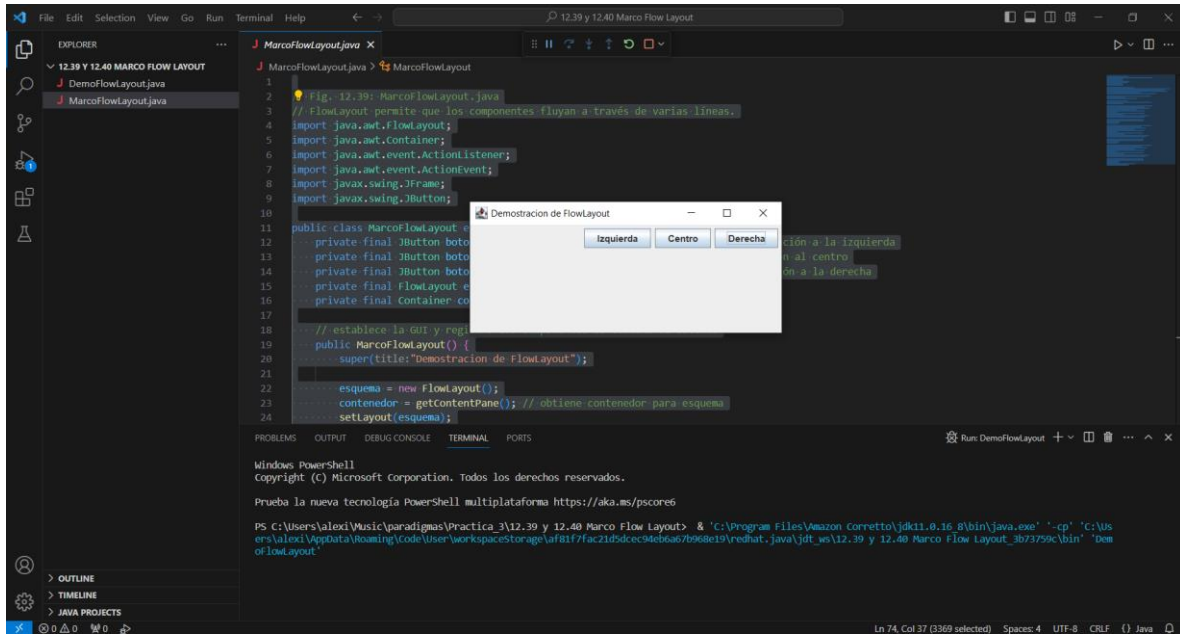
```

```

        // realinea los componentes adjuntos
        esquema.layoutContainer(contenedor);
    }
    });
} // fin del constructor de MarcoFlowLayout
} // fin de la clase MarcoFlowLayout

```

## SALIDA



## PROGRAM 15

### CODE

```
// Fig. 12.42: DemoBorderLayout.java
// Prueba de MarcoBorderLayout.
import javax.swing.JFrame;

public class DemoBorderLayout {
    public static void main(String[] args) {
        MarcoBorderLayout marcoBorderLayout = new MarcoBorderLayout();
        marcoBorderLayout.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoBorderLayout.setSize(300, 200);
        marcoBorderLayout.setVisible(true);
    }
} // fin de la clase DemoBorderLayout
```

```
// Fig. 12.41: MarcoBorderLayout.java
// BorderLayout que contiene cinco botones.
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class MarcoBorderLayout extends JFrame implements ActionListener {
    private final JButton botones[]; // arreglo de botones para ocultar
    // porciones
    private static final String nombres[] = { "Ocultar Norte", "Ocultar
    Sur",
        "Ocultar Este", "Ocultar Oeste", "Ocultar Centro" };
    private final BorderLayout esquema;

    // establece la GUI y el manejo de eventos
    public MarcoBorderLayout() {
        super("Demostracion de BorderLayout");

        esquema = new BorderLayout(5, 5); // espacios de 5 píxeles
        setLayout(esquema);
        botones = new JButton[nombres.length];

        // crea objetos JButton y registra componentes de escucha para ellos
        for (int cuenta = 0; cuenta < nombres.length; cuenta++) {
```



## PROGRAM 16

### CODE

```
// Fig. 12.44: DemoGridLayout.java
// Prueba de MarcoGridLayout.
import javax.swing.JFrame;

public class DemoGridLayout {
    public static void main(String[] args) {
        MarcoGridLayout marcoGridLayout = new MarcoGridLayout();
        marcoGridLayout.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoGridLayout.setSize(300, 200);
        marcoGridLayout.setVisible(true);
    }
} // fin de la clase DemoGridLayou
```

```
// Fig. 12.43: MarcoGridLayout.java
// GridLayout que contiene seis botones.
import java.awt.GridLayout;
import java.awt.Container;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class MarcoGridLayout extends JFrame implements ActionListener {
    private final JButton[] botones; // arreglo de botones
    private static final String[] nombres = { "uno", "dos", "tres",
"cuatro", "cinco", "seis" };
    private boolean alternar = true; // alterna entre dos esquemas
    private Container contenedor; // contenedor del marco
    private GridLayout cuadrícula1; // primer objeto GridLayout
    private GridLayout cuadrícula2; // segundo objeto GridLayout

    // constructor sin argumentos
    public MarcoGridLayout() {
        super("Demostracion de GridLayout");
        cuadrícula1 = new GridLayout(2, 3, 5, 5); // 2 por 3; espacios de 5
        cuadrícula2 = new GridLayout(3, 2); // 3 por 2; sin espacios
        contenedor = getContentPane();
        setLayout(cuadrícula1);
        botones = new JButton[nombres.length];
```

```

        for (int cuenta = 0; cuenta < nombres.length; cuenta++) {
            botones[cuenta] = new JButton(nombres[cuenta]);
            botones[cuenta].addActionListener(this); // registra componente
de escucha

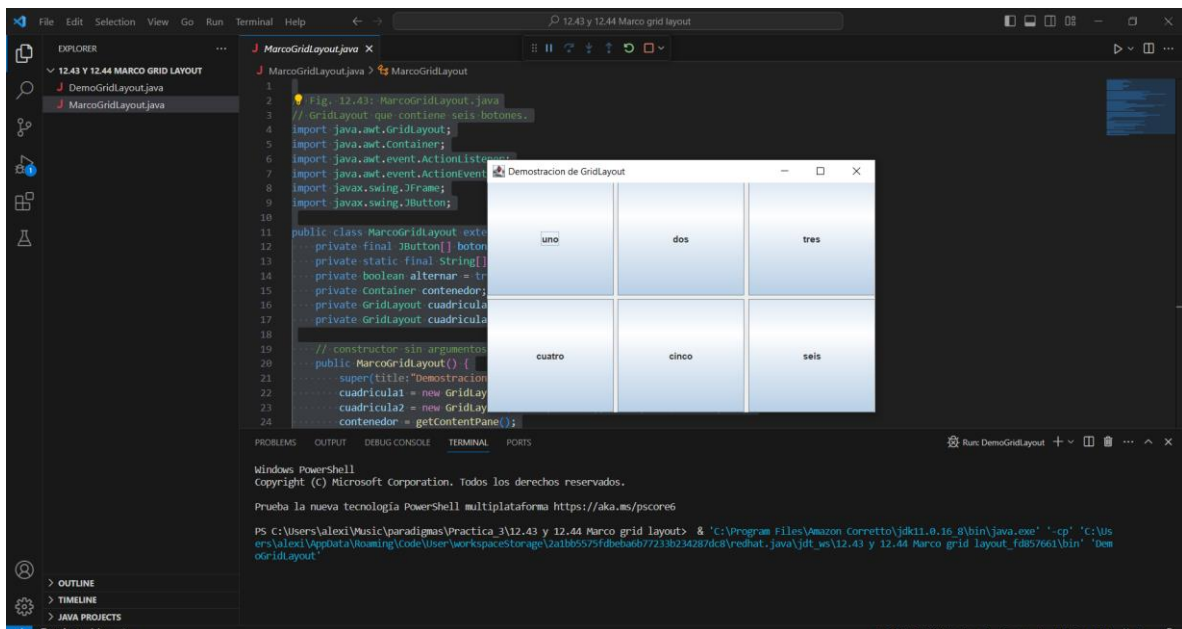
            add(botones[cuenta]); // agrega boton a objeto JFrame
        }
    }

    // maneja eventos de boton, alternando entre los esquemas
    @Override
    public void actionPerformed(ActionEvent evento) {
        if (alternar) // establece esquema con base en alternar
            contenedor.setLayout(cuadricula2);
        else
            contenedor.setLayout(cuadricula1);

        alternar = !alternar;
        contenedor.validate(); // redistribuye el contenedor
    }
} // fin de la clase MarcoGridLayout

```

SALIDA





## PROGRAM 17

### CODE

```
// Fig. 12.46: DemoPanel.java
// Prueba de MarcoPanel.
import javax.swing.JFrame;

public class DemoPanel extends JFrame {
    public static void main(String[] args) {
        MarcoPanel marcoPanel = new MarcoPanel();
        marcoPanel.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoPanel.setSize(450, 200);
        marcoPanel.setVisible(true);
    }
} // fin de la clase DemoPanel
```

```
// Fig. 12.45: MarcoPanel.java
// Uso de un objeto JPanel para ayudar a distribuir los componentes.
import java.awt.GridLayout;
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;

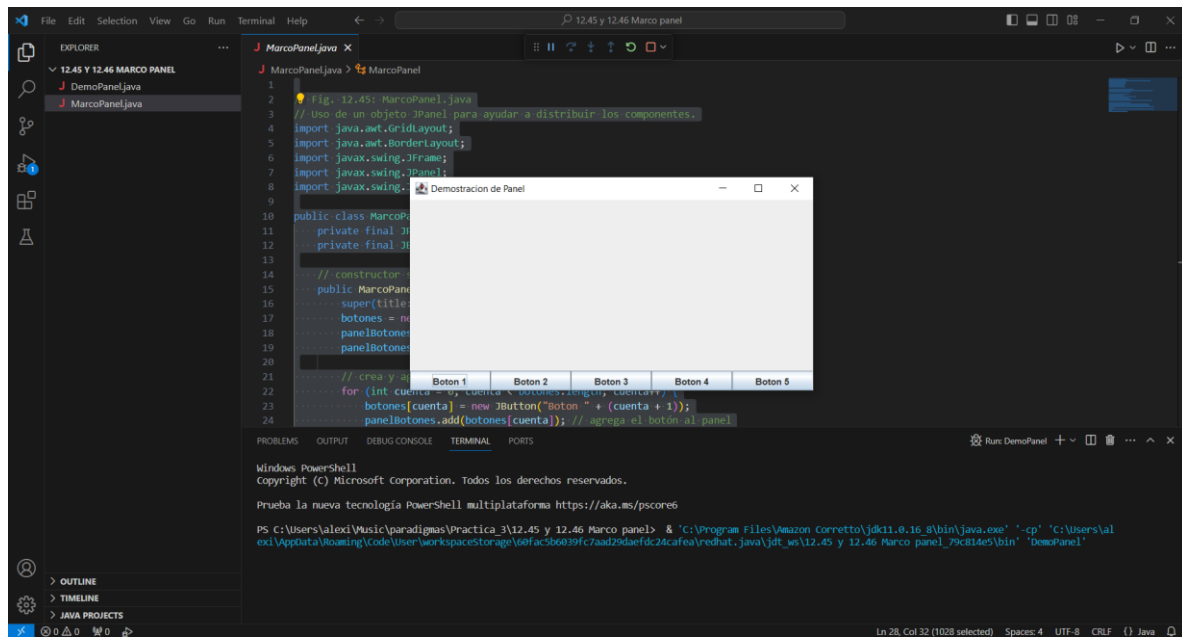
public class MarcoPanel extends JFrame {
    private final JPanel panelBotones; // panel que contiene los botones
    private final JButton[] botones;

    // constructor sin argumentos
    public MarcoPanel() {
        super("Demostracion de Panel");
        botones = new JButton[5];
        panelBotones = new JPanel();
        panelBotones.setLayout(new GridLayout(1, botones.length));

        // crea y agrega los botones
        for (int cuenta = 0; cuenta < botones.length; cuenta++) {
            botones[cuenta] = new JButton("Boton " + (cuenta + 1));
            panelBotones.add(botones[cuenta]); // agrega el botón al panel
        }
        add(panelBotones, BorderLayout.SOUTH); // agrega el panel a JFrame
    }
}
```

```
} // fin de la clase MarcoPanel
```

## SALIDA



## PROGRAM 18

### CODE

```
// Fig. 12.48: DemoAreaTexto.java
// Prueba de MarcoAreaTexto.
import javax.swing.JFrame;

public class DemoAreaTexto {
    public static void main(String[] args) {
        MarcoAreaTexto marcoAreaTexto = new MarcoAreaTexto();
        marcoAreaTexto.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoAreaTexto.setSize(425, 200);
        marcoAreaTexto.setVisible(true);
    }
} // fin de la clase DemoAreaTexto
```

```
// Fig. 12.47: MarcoAreaTexto.java
// Copia el texto seleccionado de un área JText a otra.
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.Box;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JButton;
import javax.swing.JScrollPane;

public class MarcoAreaTexto extends JFrame {
    private final JTextArea areaTexto1; // muestra cadena de demostración
    private final JTextArea areaTexto2; // el texto resaltado se copia aquí
    private final JButton botonCopiar; // inicia el copiado de texto

    // constructor sin argumentos
    public MarcoAreaTexto() {
        super("Demostracion de JTextArea");
        Box cuadro = Box.createHorizontalBox(); // crea un cuadro
        String demo = "Esta es una cadena de\ndemostracion para\n" +
            "ilustrar como copiar texto\nde un area de texto a \n" +
            "otra, usando un\nevento externo\n";

        areaTexto1 = new JTextArea(demo, 10, 15);
        cuadro.add(new JScrollPane(areaTexto1)); // agrega panel de
desplazamiento
```

```

        botonCopiar = new JButton("Copiar >>>"); // crea botón para copiar
        cuadro.add(botonCopiar); // agrega botón de copia al cuadro
        botonCopiar.addActionListener(
            new ActionListener() // clase interna anónima
            {
                // establece el texto en areaTexto2 con el texto
                // seleccionado de areaTexto1
                @Override
                public void actionPerformed(ActionEvent evento) {
                    areaTexto2.setText(areaTexto1.getSelectedText());
                }
            });

        areaTexto2 = new JTextArea(10, 15);
        areaTexto2.setEditable(false);
        cuadro.add(new JScrollPane(areaTexto2)); // agrega panel de
        desplazamiento

        add(cuadro); // agrega cuadro al marco
    }
} // fin de la clase MarcoAreaText

```

## SALIDA

