

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE TOLUCA

MANUAL DE PRÁCTICAS

TÓPICOS AVANZADOS DE PROGRAMACIÓN

M. EN C. IMELDA VERTTI GUZMAN

JULIO 2012

CONTENIDO

Práctica No.1. Manejo de Eventos	1
Objetivo	1
Introducción	1
Temas y subtemas correlacionados con la práctica	1
Material y equipo necesario:	1
Metodología.	1
Sugerencias didácticas.	2
Reporte del alumno. (Resultados)	3
Bibliografía preliminar	14
Práctica No.2. Creación de formularios	16
Objetivo	16
Introducción	16
Temas y subtemas correlacionados con la práctica	16
Material y equipo necesario:	16
Metodología.	16
Sugerencias didácticas.	18
Reporte del alumno. (Resultados)	18
Bibliografía preliminar	36
Práctica No.3. Aplicación de elementos de Interfaz de usuario como son botones, etiquetas y cajas de texto	37
Objetivo	37
Introducción	37
Temas y subtemas correlacionados con la práctica	37
Material y equipo necesario:	37
Metodología.	37
Sugerencias didácticas.	38
Reporte del alumno. (Resultados)	39
Bibliografía preliminar	41
Práctica No.4. Aplicación de elementos de Interfaz de usuario como son listas, radio, check buttons y menús	42
Objetivo	42
Introducción	42
Temas y subtemas correlacionados con la práctica	42
Material y equipo necesario:	42
Metodología.	43
Sugerencias didácticas.	44
Reporte del alumno. (Resultados)	44
Bibliografía preliminar	50
Práctica No.5. Implementación de elementos de Interfaz de usuario que incluyan el desarrollo de computación gráfica como son el dibujo de líneas, círculos, rectángulos, texto y polígonos	51
Objetivo	51
Introducción	51
Temas y subtemas correlacionados con la práctica	51
Material y equipo necesario:	51
Metodología.	51
Sugerencias didácticas.	52
Reporte del alumno. (Resultados)	53

Bibliografía preliminar	56
Práctica No.6. Utilización de componentes proporcionados por el lenguaje	57
Objetivo	57
Introducción	57
Temas y subtemas correlacionados con la práctica	57
Material y equipo necesario:	57
Metodología.	57
Sugerencias didácticas.	59
Reporte del alumno. (Resultados)	60
Bibliografía preliminar	65
Práctica No.7. Uso de librerías proporcionadas por el lenguaje	66
Objetivo	66
Introducción	66
Temas y subtemas correlacionados con la práctica	66
Material y equipo necesario:	66
Metodología.	66
Sugerencias didácticas.	71
Reporte del alumno. (Resultados)	71
Bibliografía preliminar	78
Práctica No.8. Creación de componentes definidas por el usuario	79
Objetivo	79
Introducción	79
Temas y subtemas correlacionados con la práctica	79
Material y equipo necesario:	79
Metodología.	79
Sugerencias didácticas.	80
Reporte del alumno. (Resultados)	80
Bibliografía preliminar	84
Práctica No.9. Creación y uso de paquetes/librerías definidas por el usuario	85
Objetivo	85
Introducción	85
Temas y subtemas correlacionados con la práctica	85
Material y equipo necesario:	85
Metodología.	85
Sugerencias didácticas.	87
Reporte del alumno. (Resultados)	87
Bibliografía preliminar	93
Práctica No.10. Comparación de un programa de flujo único contra uno de flujo múltiple	94
Objetivo	94
Introducción	94
Temas y subtemas correlacionados con la práctica	94
Material y equipo necesario:	94
Metodología.	94
Sugerencias didácticas.	98
Reporte del alumno. (Resultados)	98
Bibliografía preliminar	105
Práctica No.11. Creación de un programa que cree y controle hilos	106
Objetivo	106
Introducción	106

Temas y subtemas correlacionados con la práctica	106
Material y equipo necesario:	106
Metodología.	106
Sugerencias didácticas.	109
Reporte del alumno. (Resultados)	109
Bibliografía preliminar	113
Práctica No.12. Creación de un programa que Maneje la sincronización de hilos	114
Objetivo	114
Introducción	114
Temas y subtemas correlacionados con la práctica	114
Material y equipo necesario:	114
Metodología.	114
Sugerencias didácticas.	117
Reporte del alumno. (Resultados)	117
Bibliografía preliminar	121
Práctica No.13. Ejecución de un programa para dispositivo Móvil	122
Objetivo	122
Introducción	122
Temas y subtemas correlacionados con la práctica	122
Material y equipo necesario:	122
Metodología.	122
Sugerencias didácticas.	128
Reporte del alumno. (Resultados)	128
Bibliografía preliminar	139
Práctica No.14. Creación de un programa en un dispositivo móvil para el envío y recepción de información	140
Objetivo	140
Introducción	140
Temas y subtemas correlacionados con la práctica	140
Material y equipo necesario:	140
Metodología.	140
Sugerencias didácticas.	146
Reporte del alumno. (Resultados)	147
Bibliografía preliminar	152

1. PRÁCTICA No. 1

2. Nombre: MANEJO DE EVENTOS.

3. Objetivo: Identificar los eventos a los cuales los programas de software reaccionarán y como son tratados dichos eventos por estos programas a través de su código.

4. Introducción. El alumno ejecutará programas que manejen eventos, para que él o ella puedan identificar los distintos tipos de eventos, así como la creación, generación y propagación y los métodos para su control.

5. Temas y subtemas correlacionados con la práctica.

Tema I Eventos

- 1.1. Tipos de eventos
- 1.2. Generación y propagación de eventos
- 1.3. Métodos de control de eventos
- 1.4. Creación de eventos.

En esta práctica el alumno puede visualizar de una manera práctica el mecanismo de eventos en los programas java y observar como se relacionan entre sus tres partes, al ejecutar programas que lo emplean y analizar cómo interactúan los objetos de cada parte que conforman dicho mecanismo.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% de cada subtema del tema dado en la teoría de la asignatura.
- b. Hacer un repaso de el cómo compilar un programa Java aprendido en el curso anterior de programación orientada a objetos.

Puntos a desarrollar en la práctica:

1. Utilizando el libro de texto del autor Dietel buscar el capítulo en el que se trate el tema del manejo de eventos e identificar el nombre de la librería que contiene las clases que manejan eventos en cada paquete gráfico.
2. Elaborar un diagrama de bloques de las clases que manejan los eventos del paquete `java.awt.event`.
3. Crear un carpeta llamada `Pratica1`, en el directorio `C:\`.
4. Utilizando un editor de texto simple como el bloc de notas del sistema operativo, escribir en él los siguientes programas del capítulo mencionado en el punto 1:
 - i. `PruebaCampotexto.java`
 - ii. `PruebaBotonOpcion.java`
 - iii. `DemoTeclas.java`
 - iv. `RastreadorRaton.java`
5. Compilar cada programa con el comando `javac` del `jdk`
6. Ejecutar cada programa con el comando `java`.
7. Para cada programa puntualizar el evento al cual el programa reacciona y el método que da tratamiento o controla dicho evento, así como las clases involucradas.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

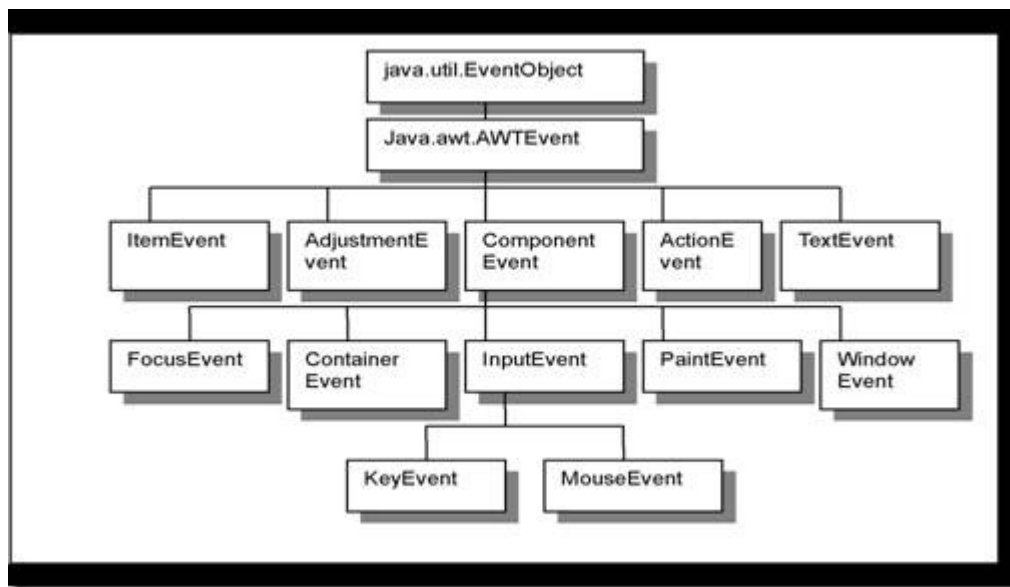
Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

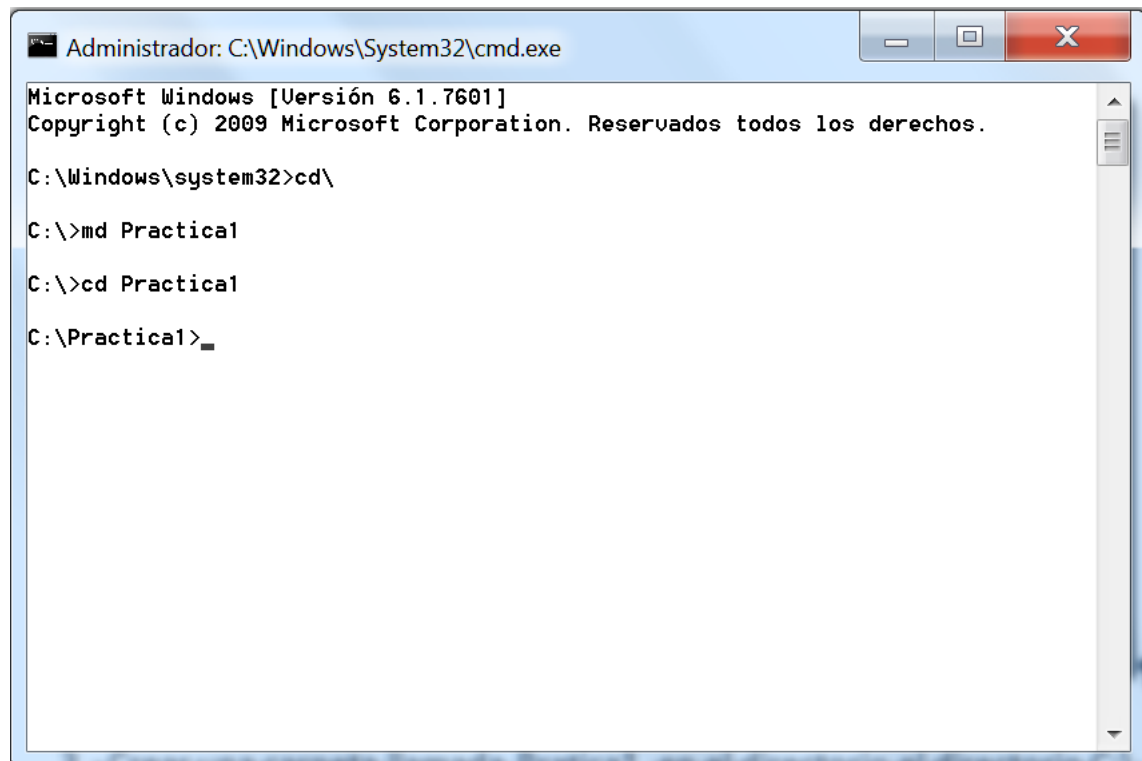
Resultados de la Práctica de acuerdo a lo indicado.

1. En el libro de texto “Como programar en Java”, en el capítulo 13, sección 4, se trata el tema de manejo de eventos. En el paquete `java.awt`, la librería `event` es la que contiene clases que manejan eventos para los componentes GUI tanto de AWT como de SWING, por otra parte en la librería `event` del paquete `javax.swing` se tienen otras clases que manejan eventos definidos para las clases Swing
2. Elaborar un diagrama de bloques de las clases que manejan los eventos del paquete `java.awt.event`.



3. Crear una carpeta llamada `Practica1`, en el directorio `C:\`.

Se ejecuta el comando `cmd` y con el comando `"cd\"`, se cambia al directorio principal y con el comando `"md Practica1"` se crea la carpeta y nuevamente con el comando `"cd Practica1"` nos posicionamos en ella:



```
Administrador: C:\Windows\System32\cmd.exe

Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Windows\system32>cd\

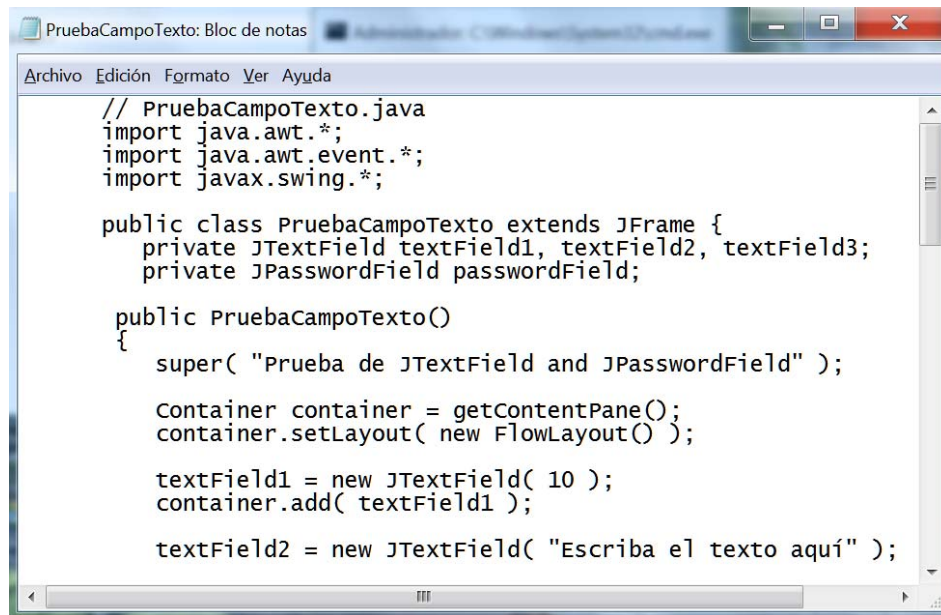
C:\>md Practica1

C:\>cd Practica1

C:\Practica1>_
```

4. Utilizando un editor de texto simple como el bloc de notas del sistema operativo, escribir en él los siguientes programas del capítulo mencionado en el punto 1.
5. Compilar cada programa con el comando javac del jdk
6. Ejecutar cada con el comando java.
7. Para cada programa puntualizar el evento al cual el programa reacciona y el método que da tratamiento o controla dicho evento, así como las clases involucradas.
 - i. PruebaCampoTexto.java

Utilizando el bloc de notas se crea el programa PruebaCampoTexto.java y se guarda en la carpeta c:\Practica1



```
// PruebaCampoTexto.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PruebaCampoTexto extends JFrame {
    private JTextField textField1, textField2, textField3;
    private JPasswordField passwordField;

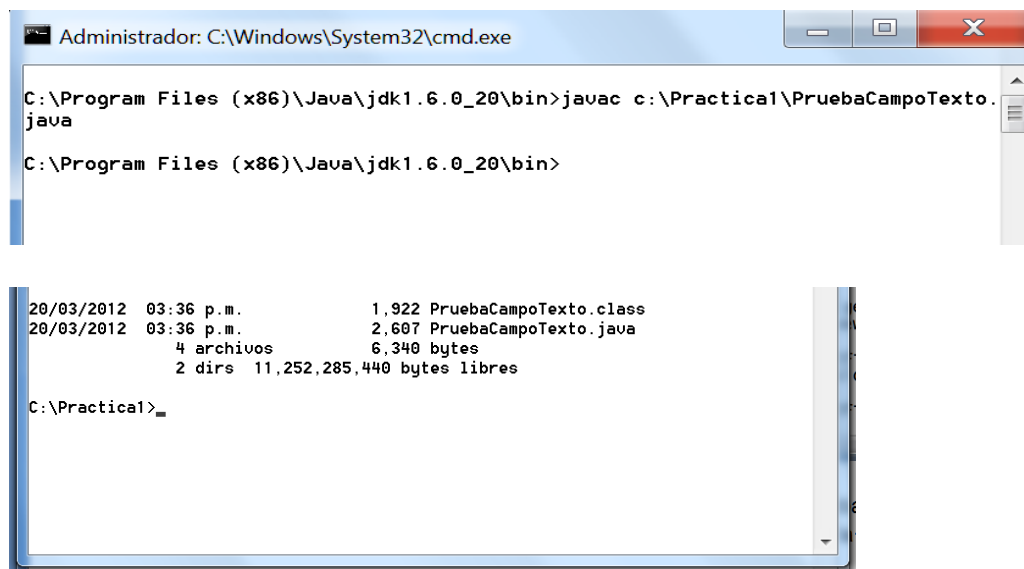
    public PruebaCampoTexto()
    {
        super( "Prueba de JTextField and JPasswordField" );

        Container container = getContentPane();
        container.setLayout( new FlowLayout() );

        textField1 = new JTextField( 10 );
        container.add( textField1 );

        textField2 = new JTextField( "Escriba el texto aquí" );
```

Se ejecuta el comando `javac PruebaCampoTexto.java` desde la carpeta bin del jdk y como puede observarse en la carpeta `c:\Practica1` se crea la clase `PruebaCampoTexto.class`



```
Administrador: C:\Windows\System32\cmd.exe

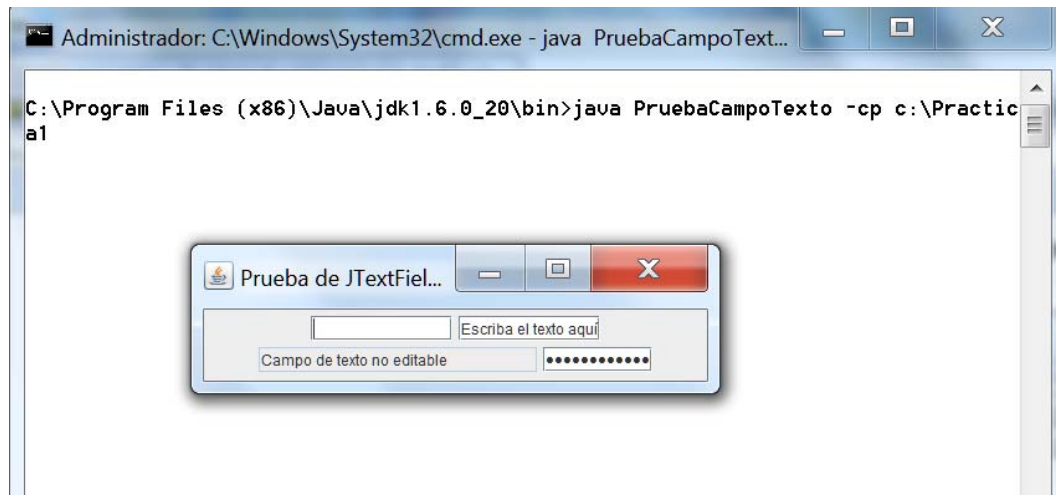
C:\Program Files (x86)\Java\jdk1.6.0_20\bin>javac c:\Practica1\PruebaCampoTexto.
java

C:\Program Files (x86)\Java\jdk1.6.0_20\bin>

20/03/2012  03:36 p.m.          1,922 PruebaCampoTexto.class
20/03/2012  03:36 p.m.          2,607 PruebaCampoTexto.java
           4 archivos          6,340 bytes
           2 dirs 11,252,285,440 bytes libres

C:\Practica1>_
```

Ahora se ejecuta el comando `java PruebaCampoTexto -cp c:\Practica1` desde la carpeta bin del jdk para ver el resultado.



En java el tratamiento de eventos participan tres objetos: el que recibe el evento, llamado evento origen o fuente; el objeto evento que es el que contiene la información del tipo de evento que ocurrió y una referencia sobre quién fue generado dicho evento y por último el objeto escuchador del evento, este objeto es el encargado de dar respuesta al evento.

Para este programa existen cuatro objetos origen que son los cuatro campos de texto. `textField1`, `textField2`, `textField3` y `passwordFiled`. Que se crean y añaden al formulario en las siguientes líneas de código:

```
textField1 = new JTextField( 10 );
container.add( textField1 );

textField2 = new JTextField( "Escriba el texto aquí" );
container.add( textField2 );

textField3 = new JTextField( "Campo de texto no editable", 20 );
textField3.setEditable( false );
container.add( textField3 );

passwordField = new JPasswordField( "Texto oculto" );
container.add( passwordField );
```

El objeto escuchador es una instancia de una clase privada llamada `TextFieldHandler` y es ligada a cada uno de los objetos origen, como puede observarse en las siguientes líneas de código:

```
TextFieldHandler handler = new TextFieldHandler();
textField1.addActionListener( handler );
textField2.addActionListener( handler );
textField3.addActionListener( handler );
passwordField.addActionListener( handler );
```

La clase TextFieldHandler implementa la interfaz ActionListener su método actionPerformed el cual tiene el objeto evento que es la instancia "event" de la clase "ActionEvent" la cual recibe el evento cuando el usuario oprime la tecla "Enter" en cualquiera de las cuatro cajas de Texto, y el método "getSource" de este objeto evento, es el que identificará cual objeto origen, es decir en cual de las cuatro cajas de texto, fue sobre el que se produjo el evento. Lo cual puede observarse en las siguientes líneas de código:

```
private class TextFieldHandler implements ActionListener {

    public void actionPerformed((ActionEvent event) )
    {
        String string = "";

        if ( event.getSource() == textField1 )
            string = "textField1: " + event.getActionCommand();

        else if ( event.getSource() == textField2 )
            string = "textField2: " + event.getActionCommand();

        else if ( event.getSource() == textField3 )
            string = "textField3: " + event.getActionCommand();

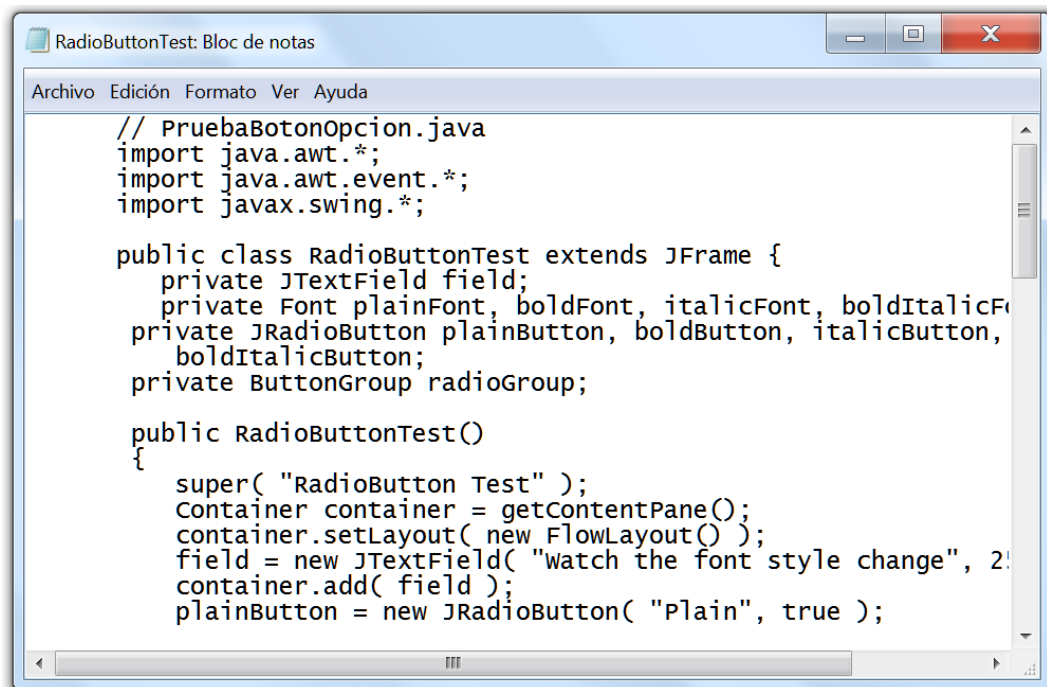
        else if ( event.getSource() == passwordField ) {
            string = "passwordField: " +
                new String( passwordField.getPassword() );
        }
        JOptionPane.showMessageDialog( null, string );
    } // Fin método actionPerformed

} // Fin private inner class TextFieldHandler

} //
```

ii. PruebaBotonOpcion.java

Utilizando el bloc de notas se crea el programa PruebaCampoTexto.java y se guarda en la carpeta c:\Practica1

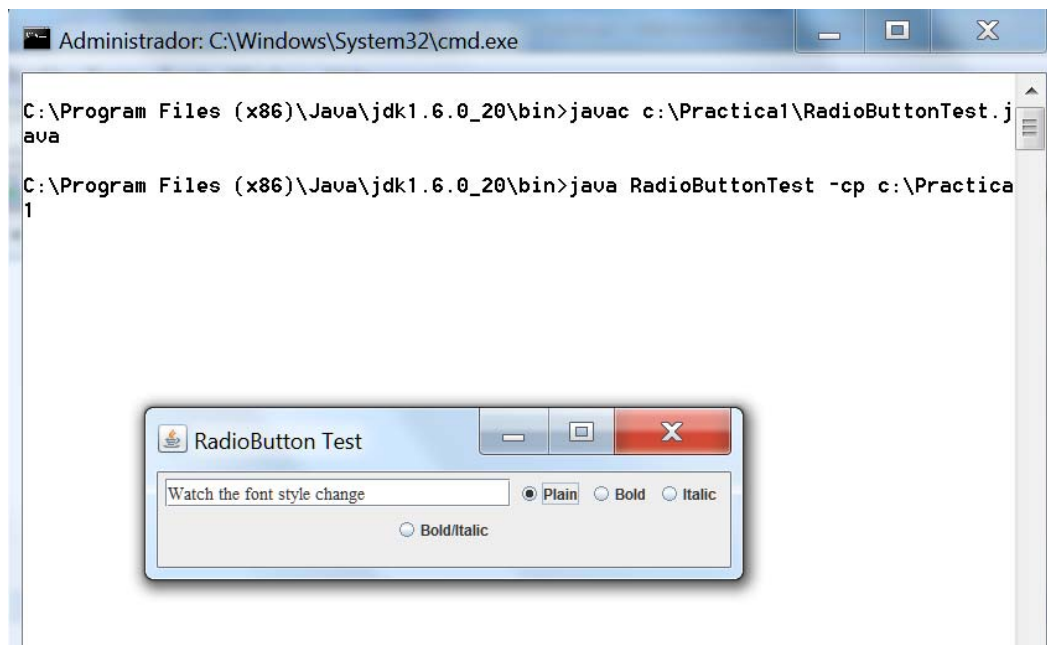


```
// PruebaBotonOpcion.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RadioButtonTest extends JFrame {
    private JTextField field;
    private Font plainFont, boldFont, italicFont, boldItalicFont;
    private JRadioButton plainButton, boldButton, italicButton,
        boldItalicButton;
    private ButtonGroup radioGroup;

    public RadioButtonTest()
    {
        super( "RadioButton Test" );
        Container container = getContentPane();
        container.setLayout( new FlowLayout() );
        field = new JTextField( "Watch the font style change", 20 );
        container.add( field );
        plainButton = new JRadioButton( "Plain", true );
```

Se ejecuta el comando `javac RadioButtonTest.java` desde la carpeta bin del jdk y el comando `java PruebaCampoTexto` desde la misma carpeta para ver el resultado.



Para este programa existen cuatro objetos origen que son los cuatro `JRadioButtons`: `plainButton`, `boldButton`, `italicButton` y `boldItalicButton`. Que se crean y añaden al formulario en las siguientes líneas de código:

```

plainButton = new JRadioButton( "Plain", true );
container.add( plainButton );
boldButton = new JRadioButton( "Bold", false );
container.add( boldButton );
italicButton = new JRadioButton( "Italic", false );
container.add( italicButton );
boldItalicButton = new JRadioButton( "Bold/Italic", false );
container.add( boldItalicButton );

```

Cada JRadioButton cambiará el tipo de fuente del campo de texto para lo cual es necesario crear cuatro instancias de la clase Font, lo cual se hace en las siguientes líneas de código:

```

plainFont = new Font( "Serif", Font.PLAIN, 14 );
boldFont = new Font( "Serif", Font.BOLD, 14 );
italicFont = new Font( "Serif", Font.ITALIC, 14 );
boldItalicFont = new Font( "Serif", Font.BOLD + Font.ITALIC, 14 );

```

El objeto escuchador es una instancia anónima de una clase privada llamada RadioButtonHandler para cada uno de los objetos origen, y cuyo constructor recibirá la fuente para cada tipo de JRadioButton como puede observarse en las siguientes líneas de código:

```

plainButton.addItemListener( new RadioButtonHandler( plainFont ) );
boldButton.addItemListener( new RadioButtonHandler( boldFont ) );
italicButton.addItemListener(
    new RadioButtonHandler( italicFont ) );
boldItalicButton.addItemListener(
    new RadioButtonHandler( boldItalicFont ) );

```

La clase RadioButtonHandler implementa la interfaz ItemListener su método itemStateChanged el cual tiene el objeto evento que es la instancia "event" de la clase "ItemEvent" la cual recibe el evento cuando el usuario selecciona cualquiera de los cuatro RadioButtons, el método "itemStateChanged" Cambiará el tipo de Font en el campo de Texto, este Font lo recibe el constructor de la clase RadioButtonHandler, lo cual puede observarse en las siguientes líneas de código:

```

private class RadioButtonHandler implements ItemListener {
    private Font font;

    public RadioButtonHandler( Font f )
    {
        font = f;
    }

    public void itemStateChanged( ItemEvent event )
    {

```

```

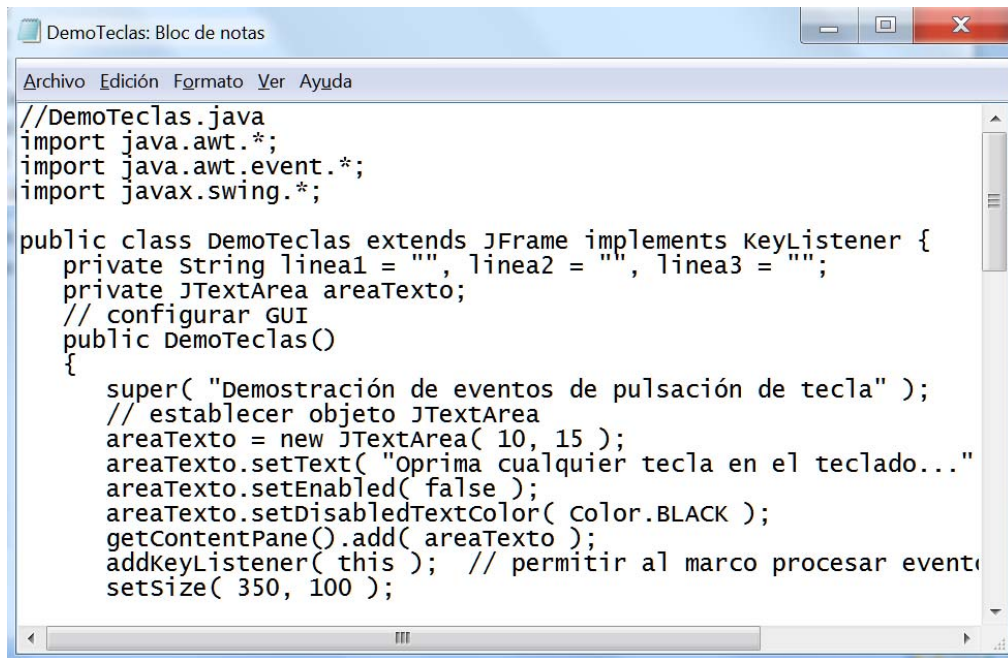
        field.setFont( font );
    }

} // Fin private inner class RadioButtonHandler

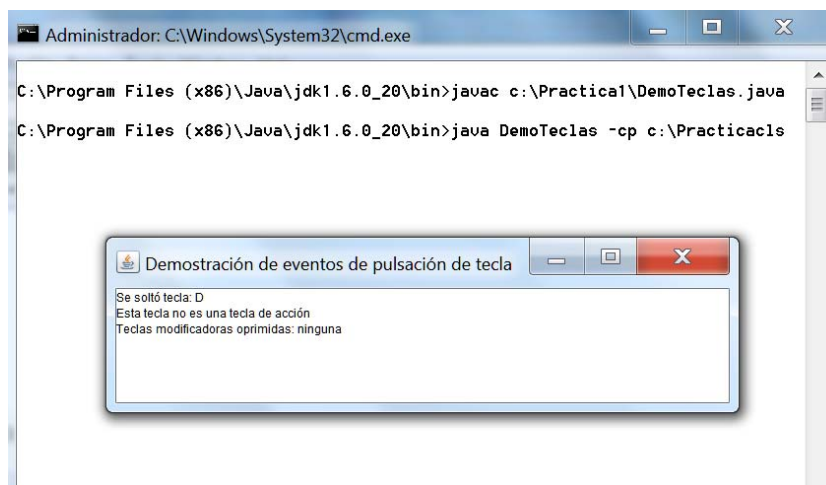
```

iii. DemoTeclas.java

Utilizando el bloc de notas se crea el programa DemoTeclas.java y se guarda en la carpeta c:\Practica1



Se ejecuta el comando javac DemoTeclas.java desde la carpeta bin del jdk y el comando java DemoTeclas desde la misma carpeta para ver el resultado.



Para este programa existe un solo objeto origen que es un área de texto. Que se crea y añade al formulario en las siguientes líneas de código:

```
areaTexto = new JTextArea( 10, 15 );
areaTexto.setText( "Oprima cualquier tecla en el teclado..." );
areaTexto.setEnabled( false );
areaTexto.setDisabledTextColor( Color.BLACK );
getContentPane().add( areaTexto );
```

El objeto escuchador es la misma clase DemoTeclas que implementa la interfaz KeyListener con sus métodos keyPressed, keyReleased y keyTyped, como puede observarse en las siguientes líneas de código:

```
public class DemoTeclas extends JFrame implements KeyListener {
    . . . .
    . . . .
    . . . .

    public void keyPressed( KeyEvent evento )
    {
        linea1 = "Se oprimió tecla: " + evento.getKeyText( evento.getKeyCode() );
        establecerLineas2y3( evento );
    }

    // manejar evento de liberación de cualquier tecla
    public void keyReleased( KeyEvent evento )
    {
        linea1 = "Se soltó tecla: " + evento.getKeyText( evento.getKeyCode() );
        establecerLineas2y3( evento );
    }

    // manejar evento de pulsación de una tecla de acción
    public void keyTyped( KeyEvent evento )
    {
        linea1 = "Se escribió tecla: " + evento.getKeyChar();
        establecerLineas2y3( evento );
    }
}
```

Estos métodos manejan el evento objeto el cual mandan como instancia al método establecerLineas2y3, este objeto evento utiliza el método "isActionKey" para identificar si el tipo de tecla que el usuario oprimió es una de acción y el método "getKeyModifiersText" para establecer si la tecla oprimida es o no una tecla modificadora. El código del método establecerLineas2y3 puede observarse a continuación:

```
private void establecerLineas2y3( KeyEvent evento )
{
    linea2 = "Esta tecla " + ( evento.isActionKey() ? "" : "no " ) +
        "es una tecla de acción";

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

```

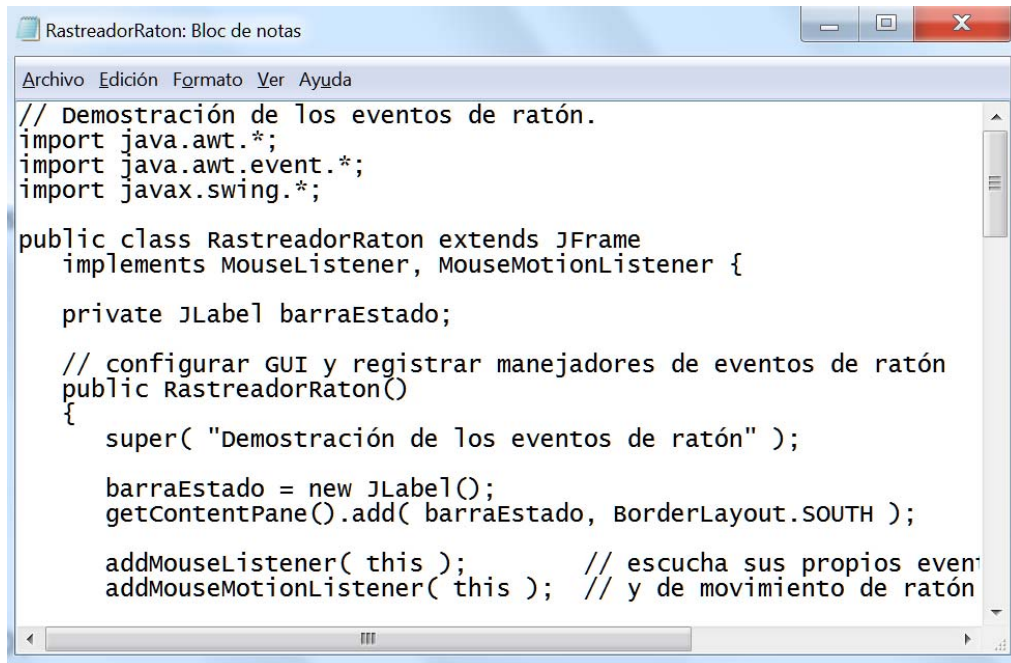
        linea3 = "Teclas modificadoras oprimidas: " +
            ( temp.equals( "" ) ? "ninguna" : temp );

        areaTexto.setText( linea1 + "\n" + linea2 + "\n" + linea3 + "\n" );
    }

```

iv. RastreadorRaton.java

Utilizando el bloc de notas se crea el programa RastreadorRaton.java y se guarda en la carpeta c:\Practica1



```

// Demostración de los eventos de ratón.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RastreadorRaton extends JFrame
    implements MouseListener, MouseMotionListener {

    private JLabel barraEstado;

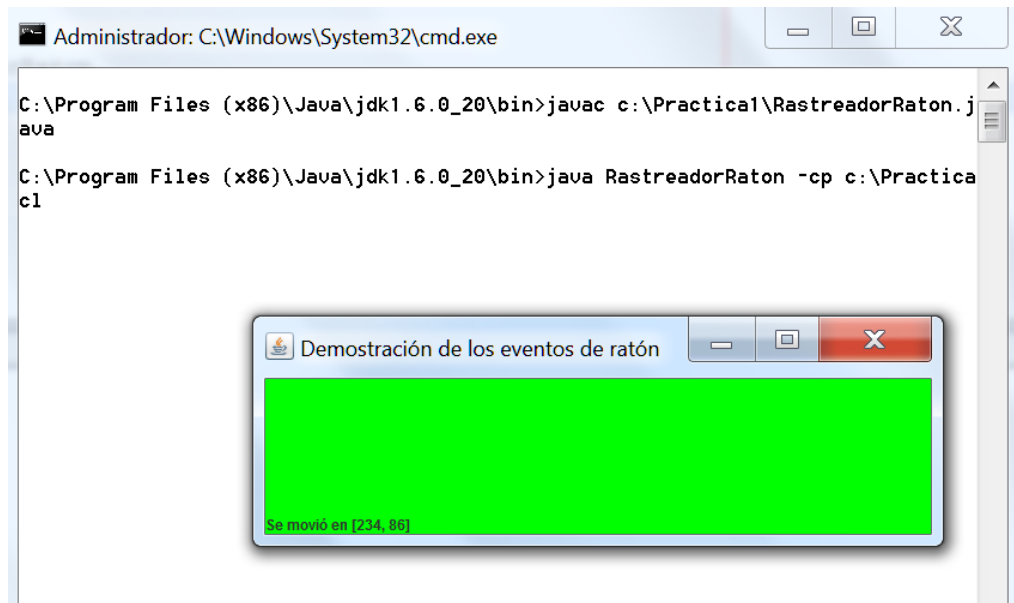
    // configurar GUI y registrar manejadores de eventos de ratón
    public RastreadorRaton()
    {
        super( "Demostración de los eventos de ratón" );

        barraEstado = new JLabel();
        getContentPane().add( barraEstado, BorderLayout.SOUTH );

        addMouseListener( this );      // escucha sus propios eventos
        addMouseMotionListener( this ); // y de movimiento de ratón
    }
}

```

Se ejecuta el comando `javac RastreadorRaton.java` desde la carpeta bin del jdk y el comando `java RastreadorRaton` desde la misma carpeta para ver el resultado.



Para este programa existe un objeto origen que es el mismo formulario `RastreadorRaton` y también es el objeto escuchador ya que implementa las interfaces `MouseListener` y `MouseMotionListener` con sus métodos: `mouseClicked`, `mousePressed`, `mouseReleased`, `mouseEntered`, `mouseExited`, y `mouseDragged`, `mouseMoved` respectivamente como puede observarse en las siguientes líneas de código:

```
public class RastreadorRaton extends JFrame
    implements MouseListener, MouseMotionListener {
    . . . .
    . . . .
    . . . .

    // Manejadores de eventos de MouseListener
    // manejar el evento cuando el botón del ratón se suelta inmediatamente
    // después de oprimir
    public void mouseClicked( MouseEvent evento )
    {
        barraEstado.setText( "Se hizo clic en [" + evento.getX() +
            ", " + evento.getY() + "]" );
    }

    // manejar evento cuando se oprime el botón del ratón
    public void mousePressed( MouseEvent evento )
    {
        barraEstado.setText( "Se oprimió en [" + evento.getX() +
            ", " + evento.getY() + "]" );
    }

    // manejar evento cuando se suelta el ratón después de arrastrar
    public void mouseReleased( MouseEvent evento )
    {
```

```

        barraEstado.setText( "Se soltó en [" + evento.getX() +
            ", " + evento.getY() + "]" );
    }

    // manejar el evento cuando el ratón entra al área
    public void mouseEntered( MouseEvent evento )
    {
        barraEstado.setText( "Ratón entro en [" + evento.getX() +
            ", " + evento.getY() + "]" );
        getContentPane().setBackground( Color.GREEN );
    }

    // manejar evento cuando el ratón sale del área
    public void mouseExited( MouseEvent evento )
    {
        barraEstado.setText( "Ratón fuera de la ventana" );
        getContentPane().setBackground( Color.WHITE );
    }

    // Manejadores de eventos de MouseMotionListener
    // manejar el evento cuando el usuario arrastra el ratón con el botón
    oprimido
    public void mouseDragged( MouseEvent evento )
    {
        barraEstado.setText( "Se arrastró en [" + evento.getX() +
            ", " + evento.getY() + "]" );
    }

    // manejar el evento cuando el usuario mueve el ratón
    public void mouseMoved( MouseEvent evento )
    {
        barraEstado.setText( "Se movió en [" + evento.getX() +
            ", " + evento.getY() + "]" );
    }
}

```

Estos métodos manejan el evento objeto MouseEvent el cual utiliza los métodos “getX” y “getY” para obtener las coordenadas del ratón y mostrarlas en una instancia de un etiqueta llamada “barraEstado”, que ha sido creada y colocada en la parte sur del Layout por defecto del JFrame. Como muestra las siguientes líneas de código

```

barraEstado = new JLabel();
getContentPane().add( barraEstado, BorderLayout.SOUTH );

```

10. Bibliografía preliminar

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 2

2. Nombre: CREACIÓN DE FORMULARIOS.

3. Objetivo: Aprender a crear formularios usando diferentes entornos de desarrollo.

4. Introducción. El alumno ejecutará un mismo programa que muestra un formulario en los dos entornos de desarrollo gratuitos más populares como son: Netbeans y Eclipse y en un entorno comercial como el JCreator LE y puntualizará si es que hay, las diferencias de cada uno de ellos en la creación de formularios.

5. Temas y subtemas correlacionados con la práctica.

Tema II Interfaz Gráfica de Usuario

2.1. Creación de interfaz grafica para usuarios

En esta práctica el alumno es introducido de una manera práctica, al subtema de Creación de interfaz gráfica para usuarios; específicamente en la creación de formularios, ampliando su conocimiento entre los diferentes entornos de desarrollo Java que puede utilizar.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el subtema del tema dado en la teoría de la asignatura.
- b. Haber instalado en el computador el Netbeasn Versión 6.9 o 7.0 , el Eclipse versión 3.7 y el JCreator LE Versión 5.0

Puntos a desarrollar en la práctica:

1. En cada entorno de desarrollo crear un proyecto llamado Practica2 y al último describir las diferencias entre ellos.
2. Para cada proyecto Practica2 de cada entorno de desarrollo agregar un archivo java llamado FormularioConBotones.java que tenga el siguiente código y al último describir las diferencias entre ellos:

```
import java.awt.*;
import javax.swing.*;

public class FormularioConBotones {
    private JFrame f;
    private JButton b1;
    private JButton b2;

    public FormularioConBotones() {
        f = new JFrame("Ejemplo de GUI con botones ");
        b1 = new JButton("Presioname");
        b2 = new JButton("No me presiones");
    }

    public void launchFrame() {
        f.getContentPane().setLayout(new FlowLayout());
        f.getContentPane().add(b1);
        f.getContentPane().add(b2);
        f.pack();
        f.setVisible(true);
    }

    public static void main(String args[]) {
        FormularioConBotones guiWindow = new FormularioConBotones();
        guiWindow.launchFrame();
    }
}
```

3. Compilar y ejecutar el programa en cada entorno de desarrollo y al último describir las diferencias entre ellos.
4. Para cada entorno de desarrollo crear formularios con los recursos que el mismo entorno proporciona y agregarle los mismos botones del

programa anterior. Compilar y ejecutar el programa y al último describir las diferencias entre ellos.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

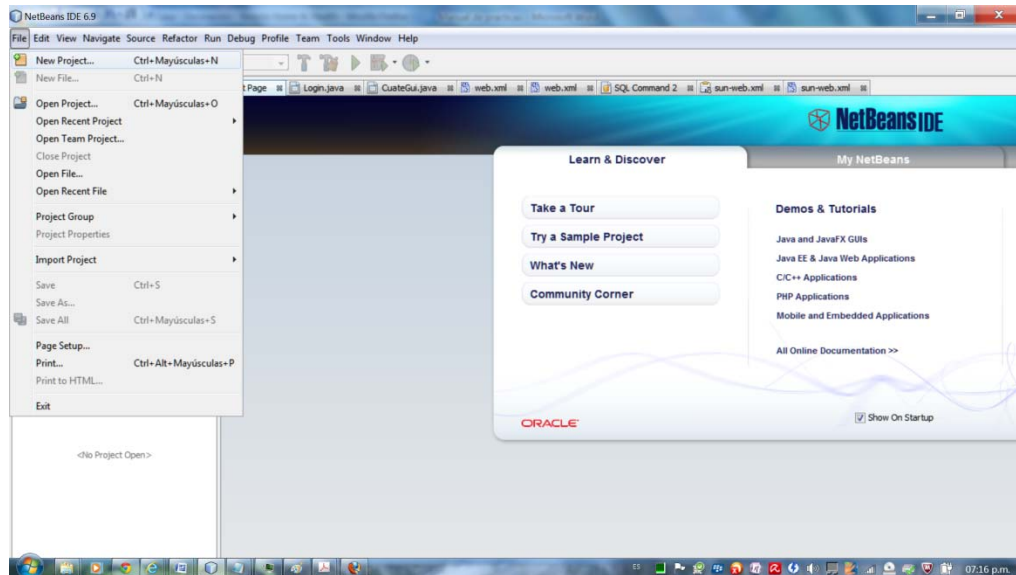
9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

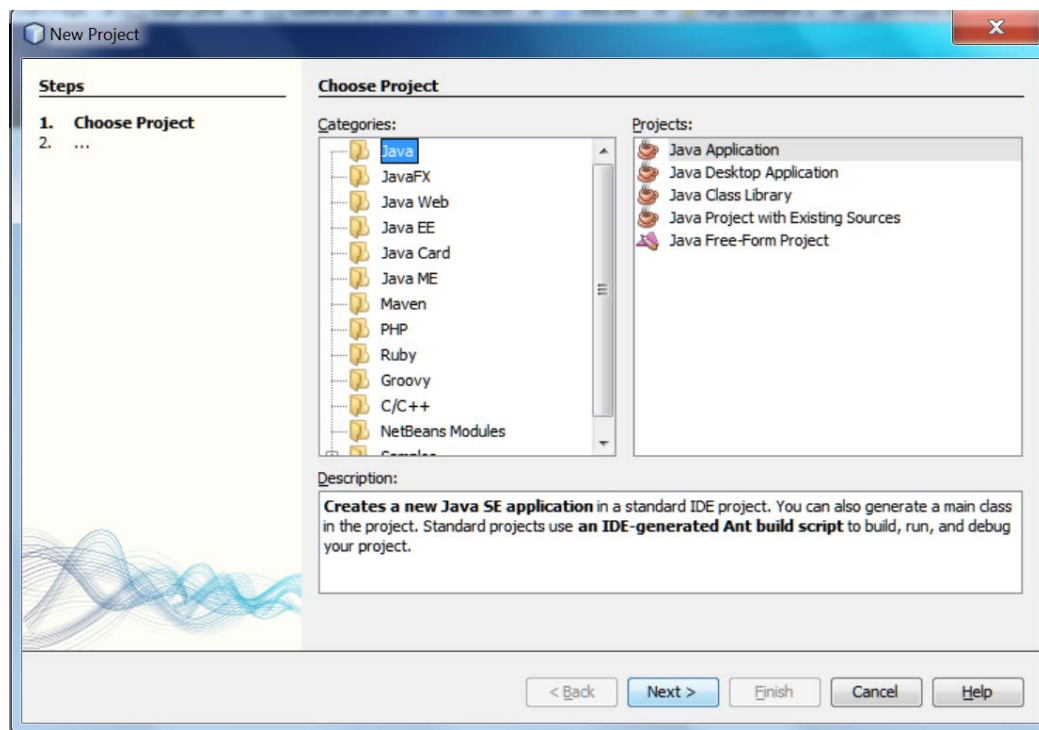
1. En cada entorno de desarrollo crear un proyecto llamado Practica2 y al último describir las diferencias entre ellos.

Netbeans .

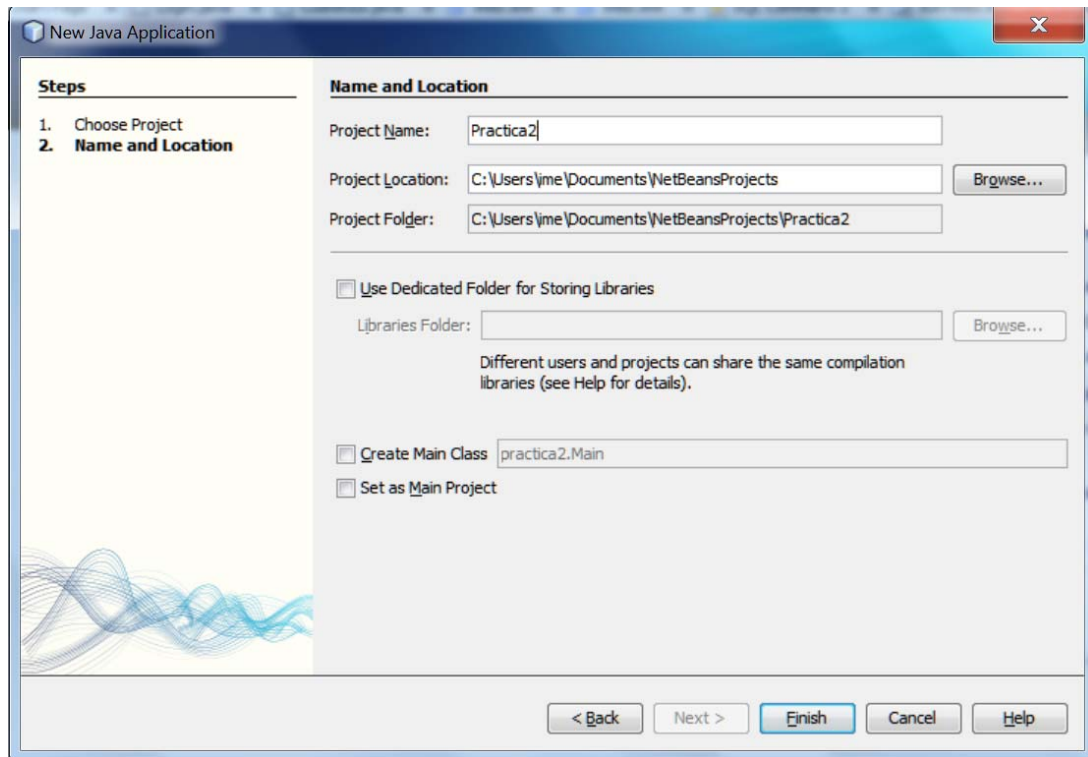
En el menú principal se selecciona el menú Archivo y la opción Nuevo Proyecto.



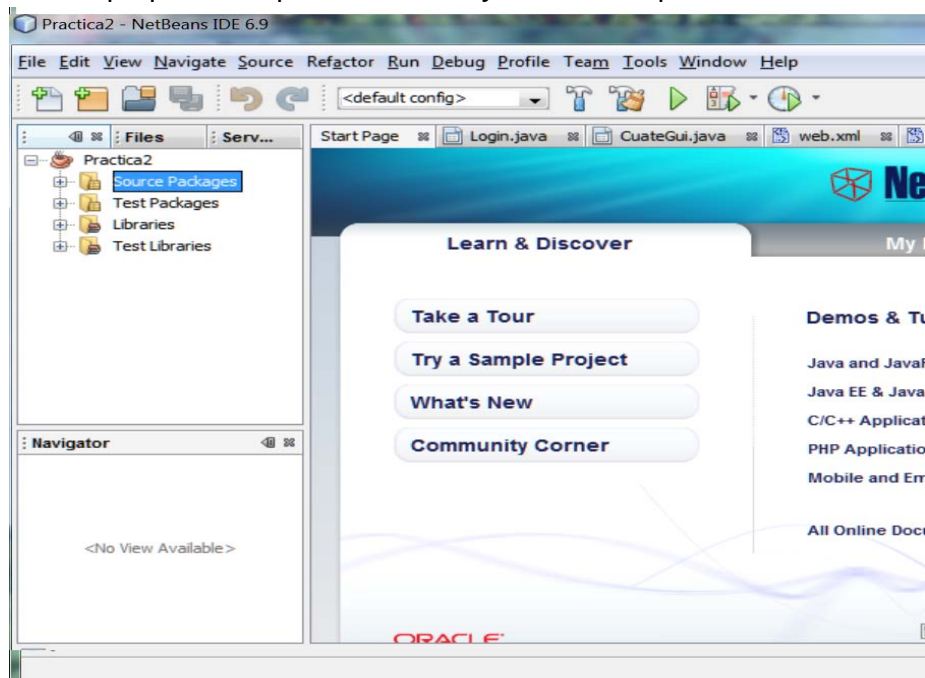
En la ventana de diálogo resultante, se selecciona un proyecto Java tipo Java Aplicación y se da un clic en siguiente



En la ventana de diálogo resultante, se coloca el nombre del proyecto en este caso Practica2 y se deselectiona Crear Clase Main y se da un clic en Terminar

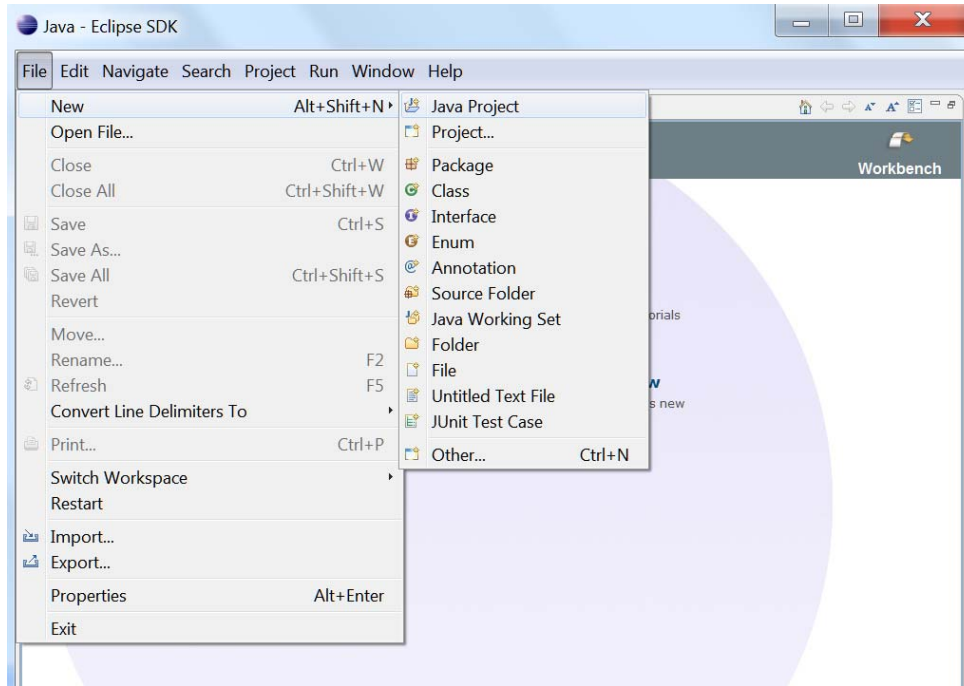


Una vez terminado se tiene el árbol del proyecto con sus carpetas de paquetes fuente, paquetes de prueba, librería y librerías de prueba

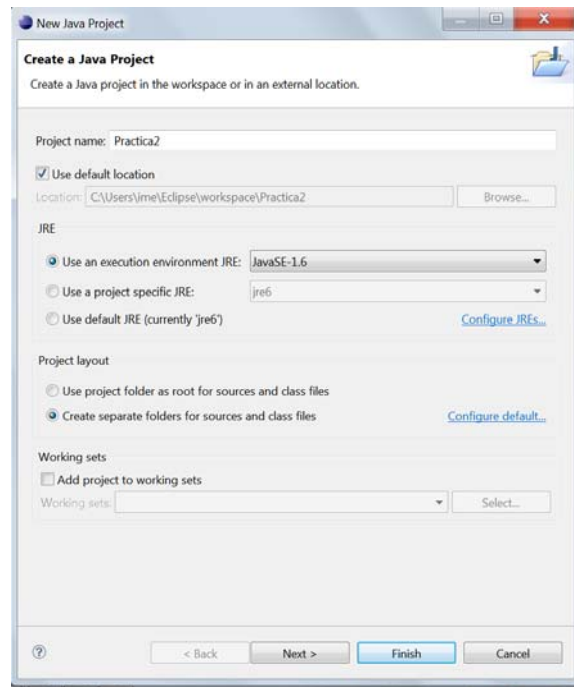


Eclipse .

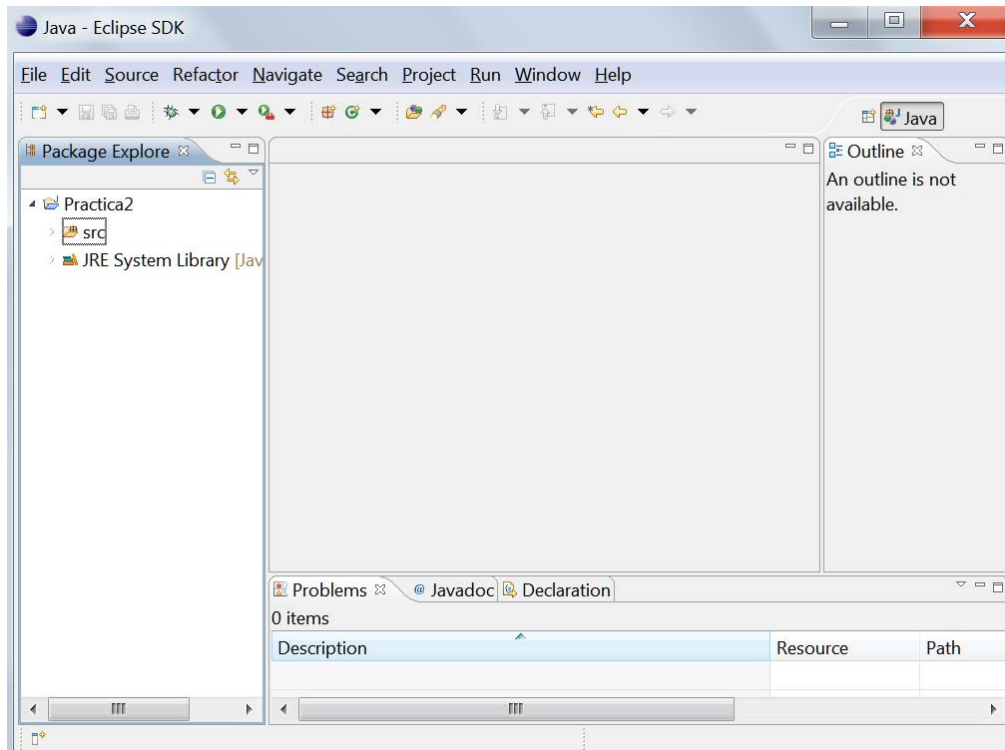
En el menú principal se selecciona el menú Archivo y la opción Nuevo Proyecto Java.



En la ventana de diálogo resultante, se coloca el nombre del proyecto en este caso Practica2 y se da un clic en Terminar

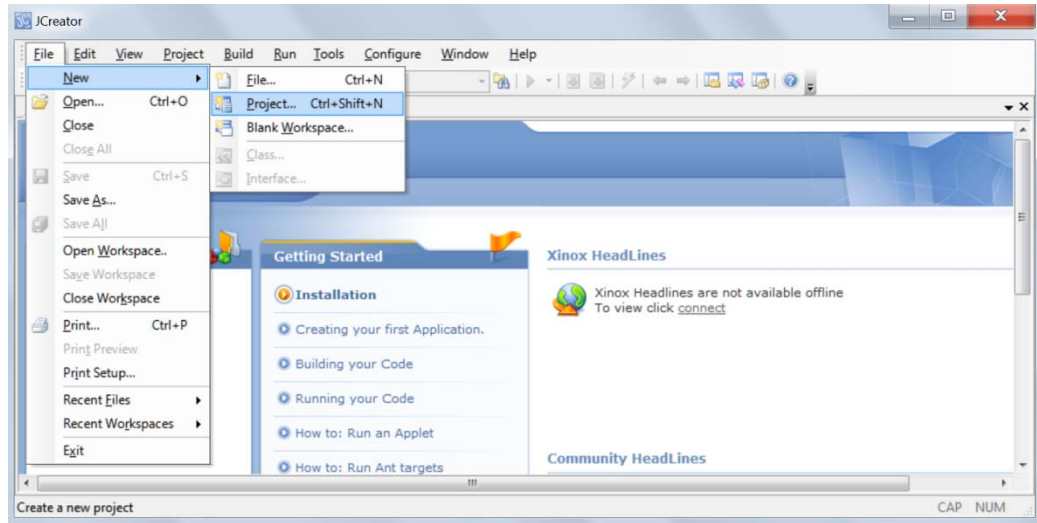


Una vez terminado y cerrando la ventana de bienvenida se puede observar el árbol del proyecto con sus carpetas de paquetes fuente, y la librería JRE System

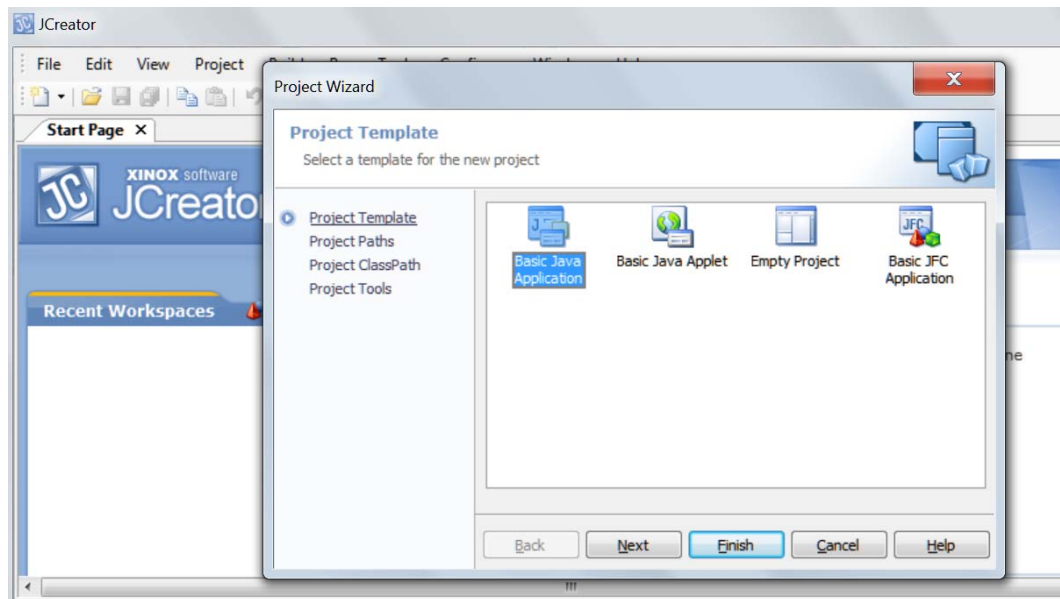


JCreator LE .

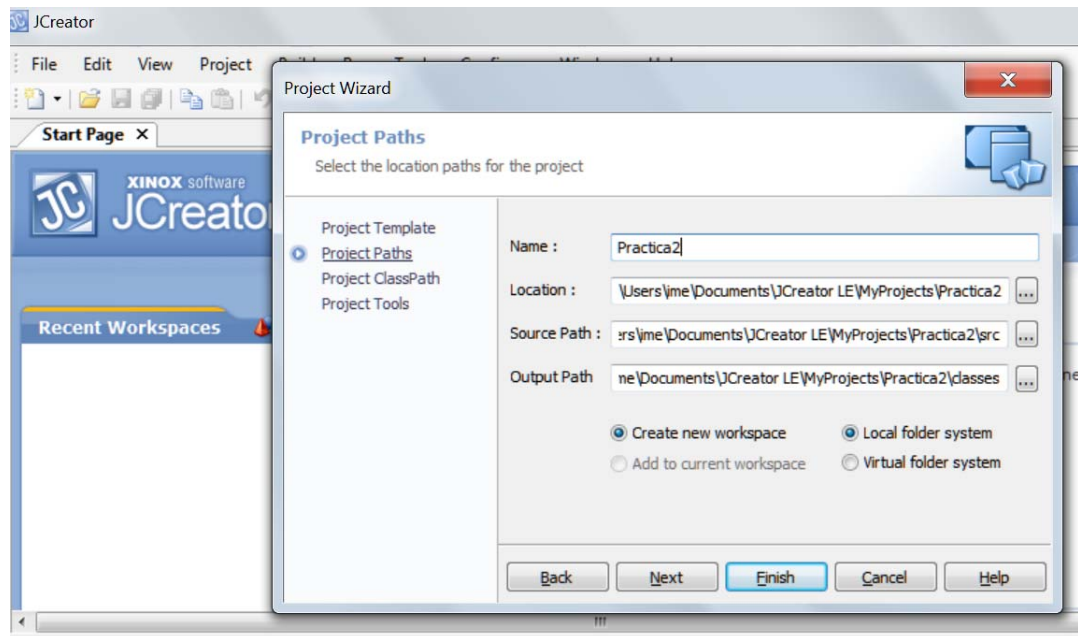
En el menú principal se selecciona el menú Archivo y la opción Nuevo Proyecto.



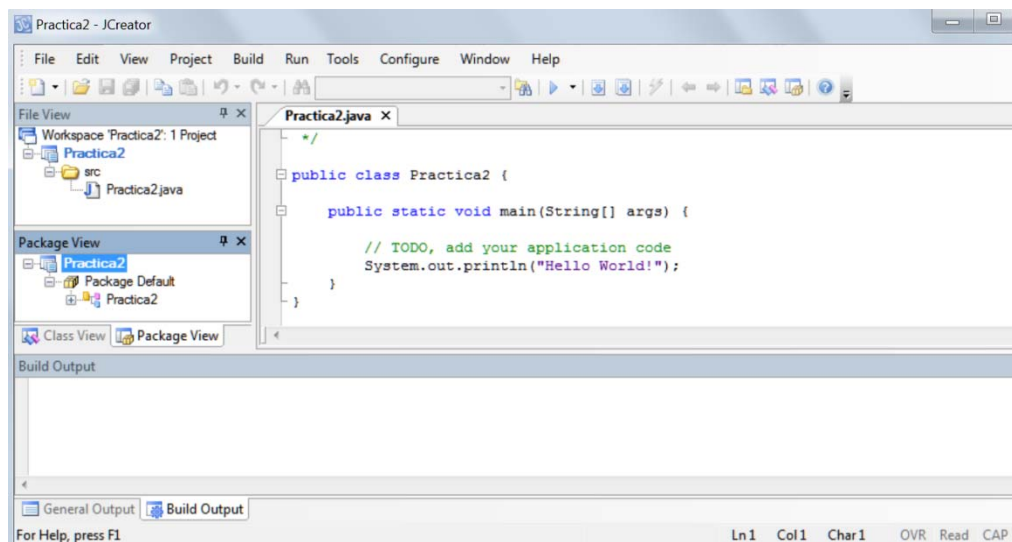
En la ventana de diálogo resultante, se selecciona un proyecto Java tipo Básico Aplicación Java y se da un clic en siguiente



En la ventana de diálogo resultante, se coloca el nombre del proyecto en este caso Practica2 y se da un clic en Terminar



Una vez terminado se tiene el árbol del espacio de trabajo del proyecto con sus carpetas de paquetes fuente, pero se ha creado aparte una clase java llamada Practica2.



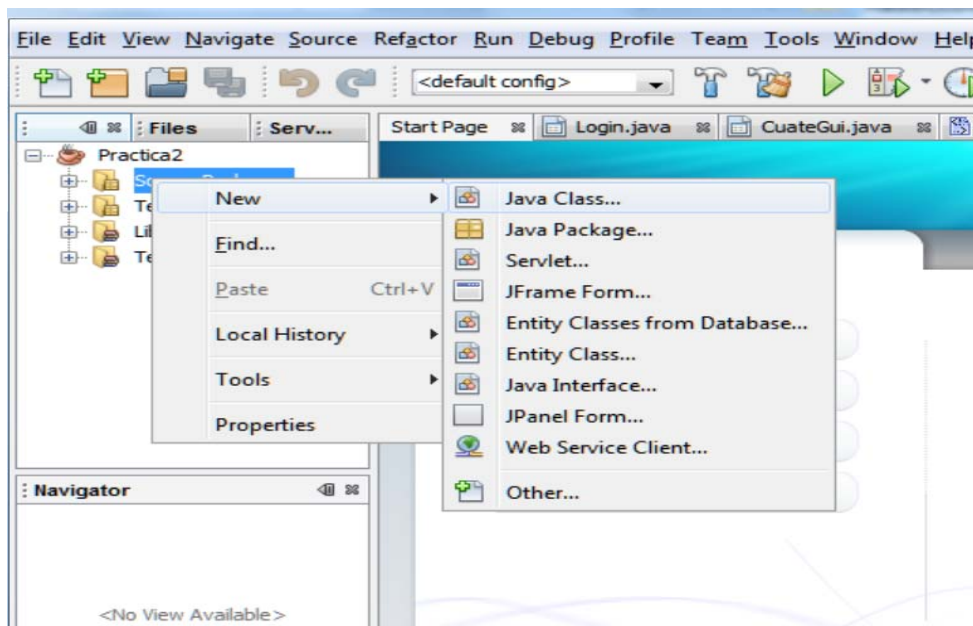
En cuanto a la creación de proyectos no parece existir grandes diferencias en cuanto a los pasos principales que hay que dar entre los tres entornos de desarrollo. Sin embargo el resultado varía entre los dos entornos libres, es decir, NetBeans y Eclipse y el JCreator ya que este genera un espacio de trabajo y aparte sino se solicita un proyecto vacío, se crea un clase por defecto con el mismo nombre del proyecto que podría actuar como clase principal de todo el

proyecto ya que cuenta con el método main() y si no se desea que esta clase sea la principal simplemente se selecciona del árbol y se elimina.

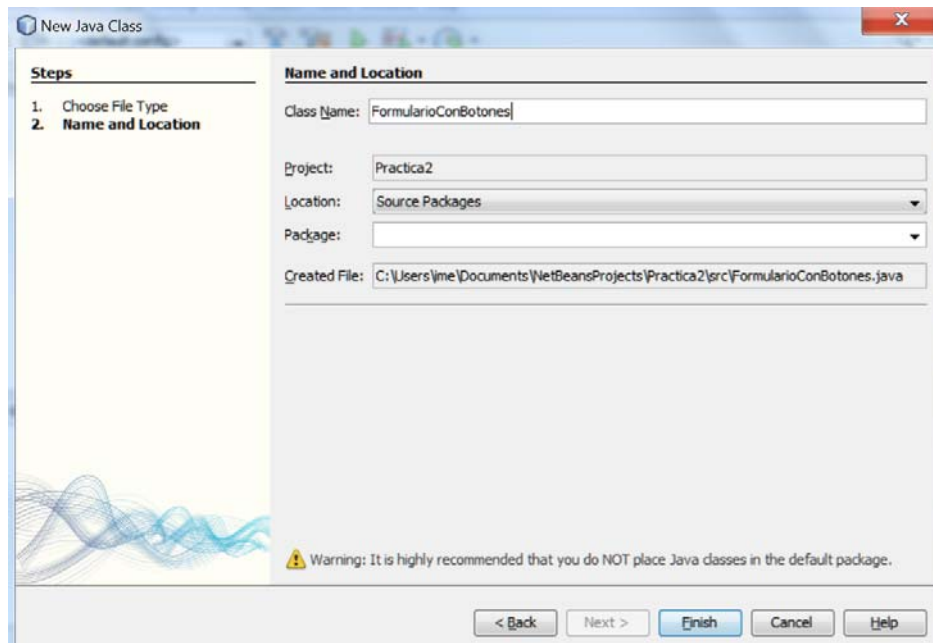
2. Para cada proyecto Practica2 de cada entorno de desarrollo agregar un archivo java llamado FormularioConBotones.java, con el código indicado y al último describir las diferencias entre ellos.

Netbeans .

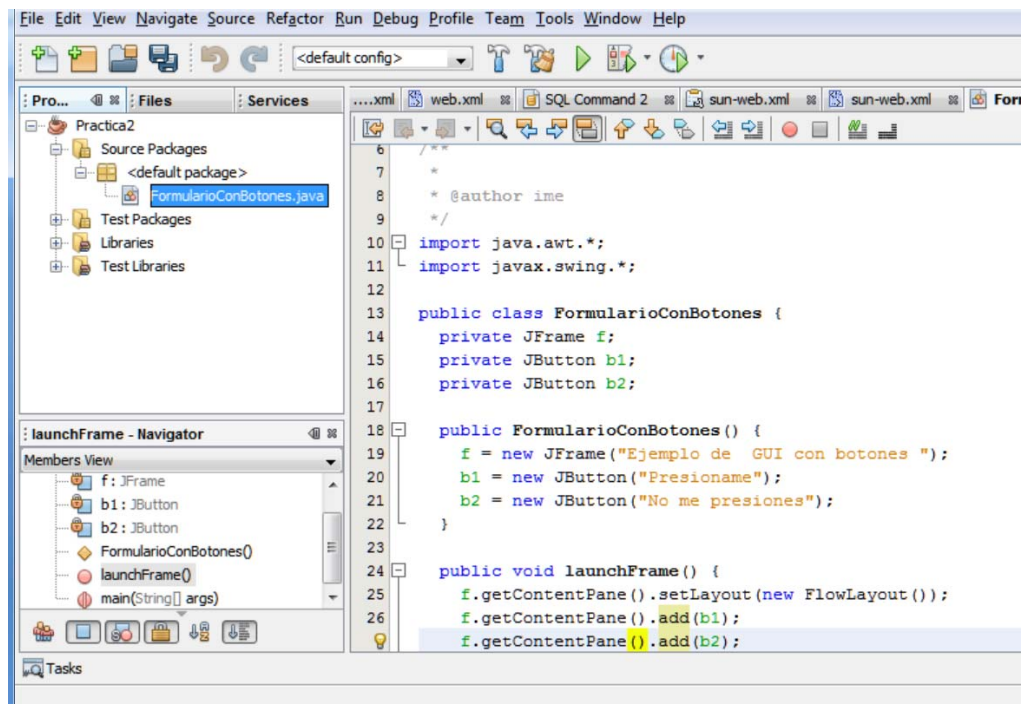
Se selecciona en el árbol del proyecto el paquete de fuentes y haciendo clic con el botón derecho del mouse, se solicita crear una nueva clase java:



En la ventana de diálogo resultante se pone el nombre de la clase "FormularioConBotones" y se selecciona Terminar:

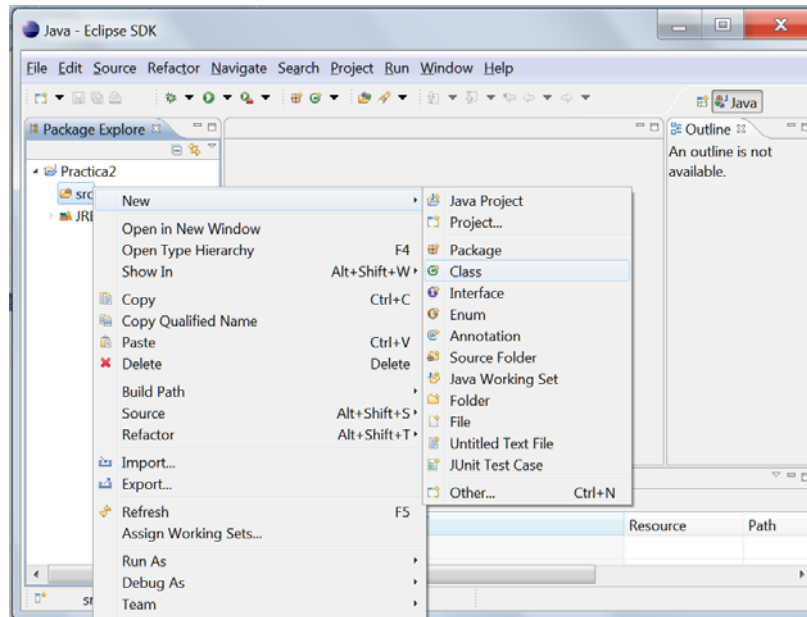


Y se le agrega el código proporcionado:

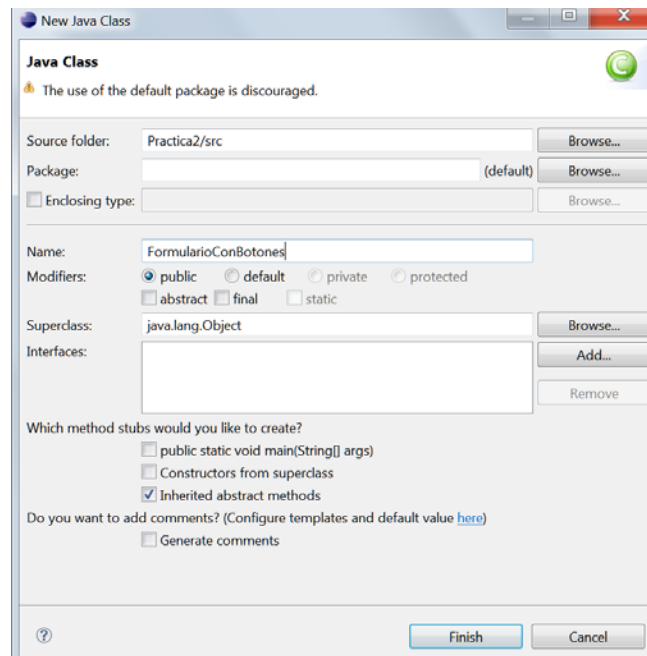


Eclipse.

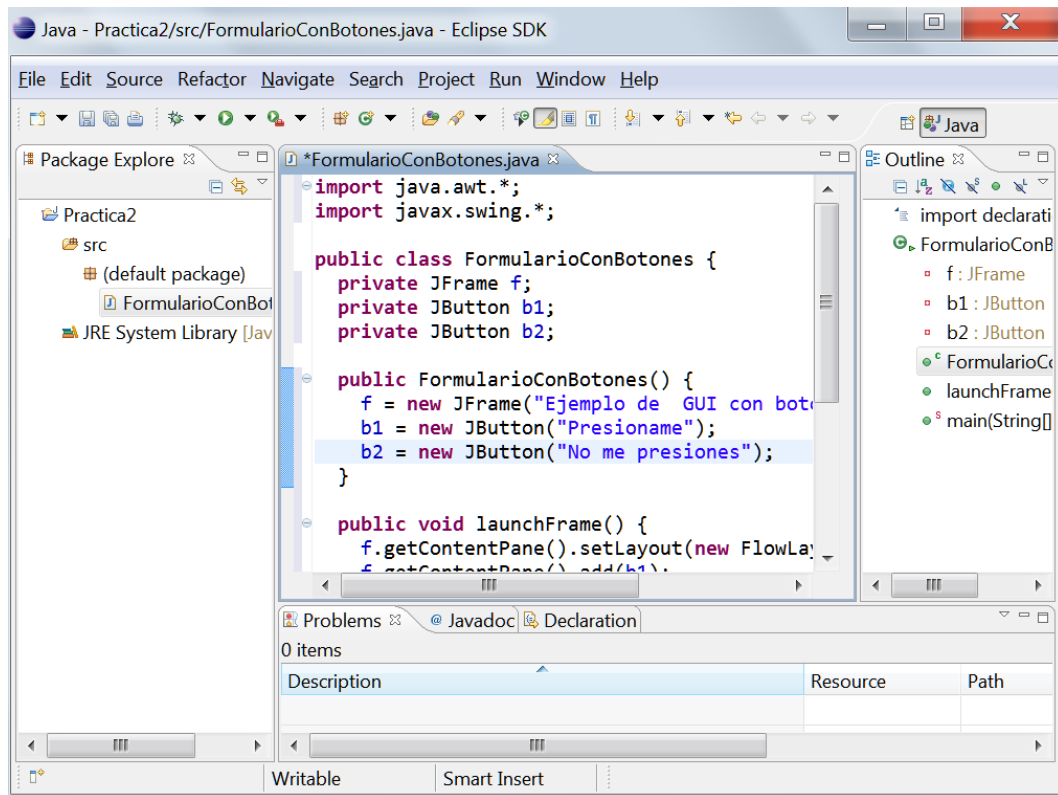
Se selecciona en el árbol del proyecto el paquete de fuentes y haciendo clic con el botón derecho del mouse, se solicita crear una nueva clase:



En la ventana de diálogo resultante se pone el nombre de la clase “FormularioConBotones” y se selecciona Terminar:

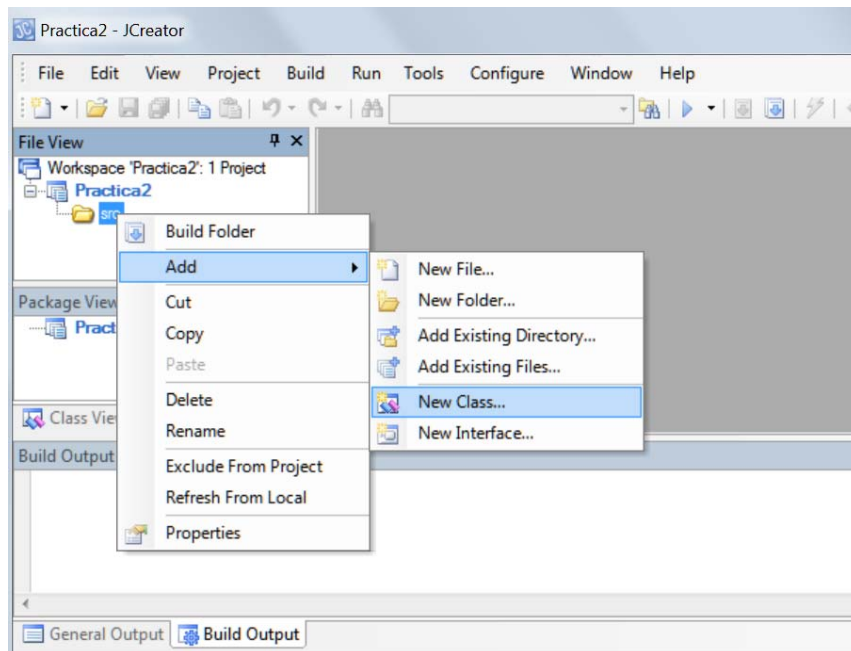


Y se le agrega el código proporcionado:

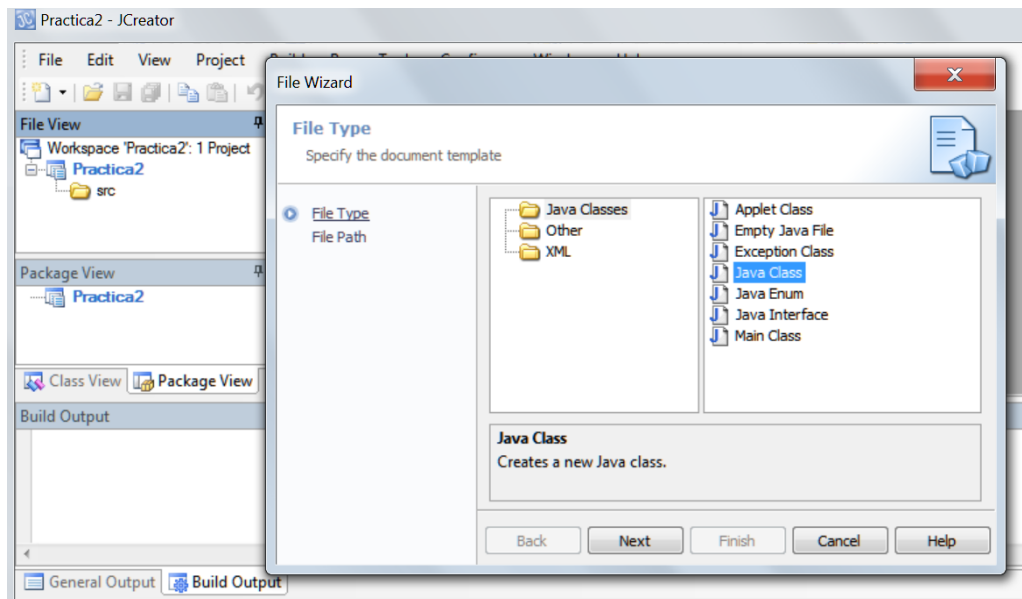


JCreator LE .

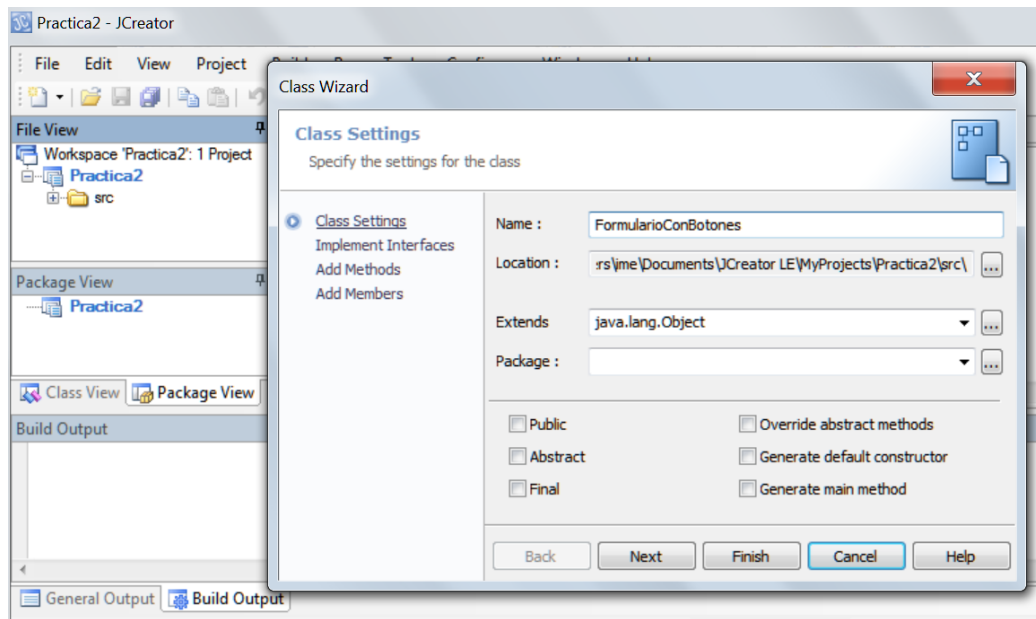
Se selecciona en el árbol del proyecto el paquete de fuentes y haciendo clic con el botón derecho del mouse, se solicita crear una nueva clase, (recordar que como ya existe previamente una clase con Main() llamada Practica2 se debe eliminar previamente):



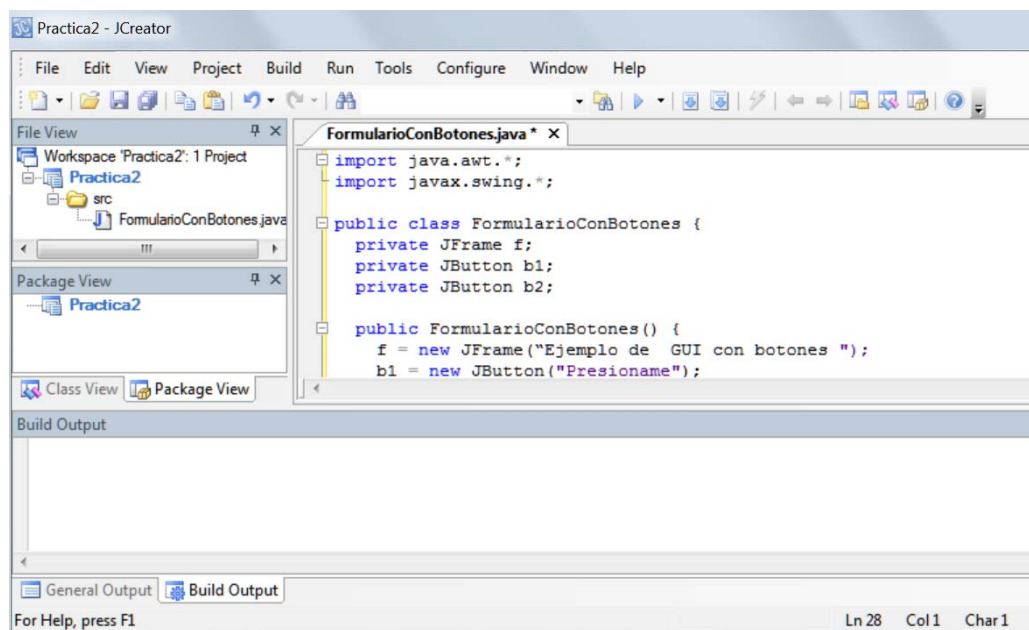
En la ventana de diálogo resultante se elije una clase Java o una Main() se da clic en siguiente:



En la ventana de diálogo resultante se pone el nombre de la clase "FormularioConBotones" y se selecciona Terminar:



Y se le agrega el código proporcionado:

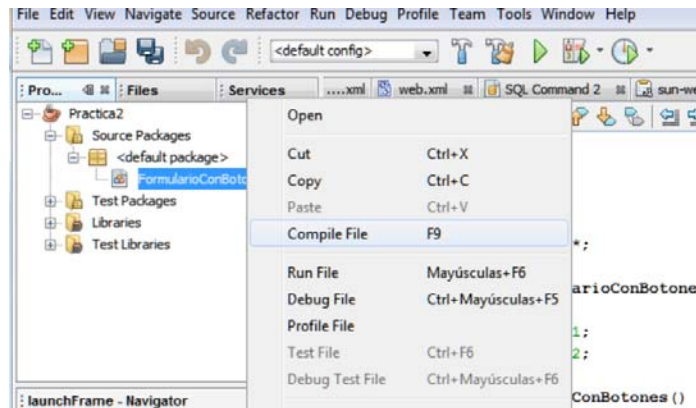


En cuanto a la creación de clases tanto NetBeans como Eclipse utilizan menos ventanas de diálogo que JCreator LE y además tienen la advertencia de que no se debe utilizar el paquete por defecto y aunque la clase se cree en este paquete lo muestra en su árbol, mientras que JCreator LE no.

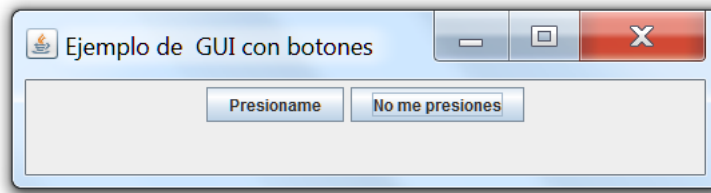
3. Compilar y ejecutar el programa en cada entorno de desarrollo y al último describir las diferencias entre ellos.

Netbeans .

En el árbol del proyecto, en el paquete de fuentes se selecciona el archivo correspondiente al programa FormularioConBotones.java y haciendo clic en el botón derecho del mouse se selecciona del menú desplegado la opción compilar o simplemente se oprime la tecla F9.

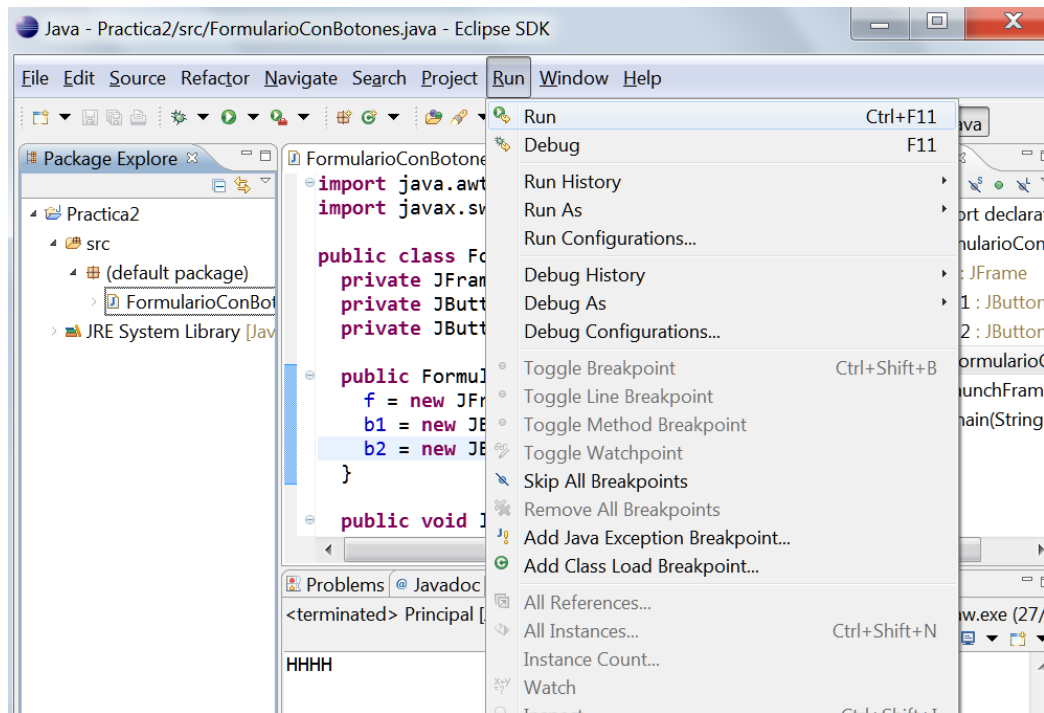


Nuevamente haciendo clic en el botón derecho del mouse se selecciona del menú desplegado la opción ejecutar o se oprimen a la vez las teclas “Mayúsculas y F6” y se observa el resultado.

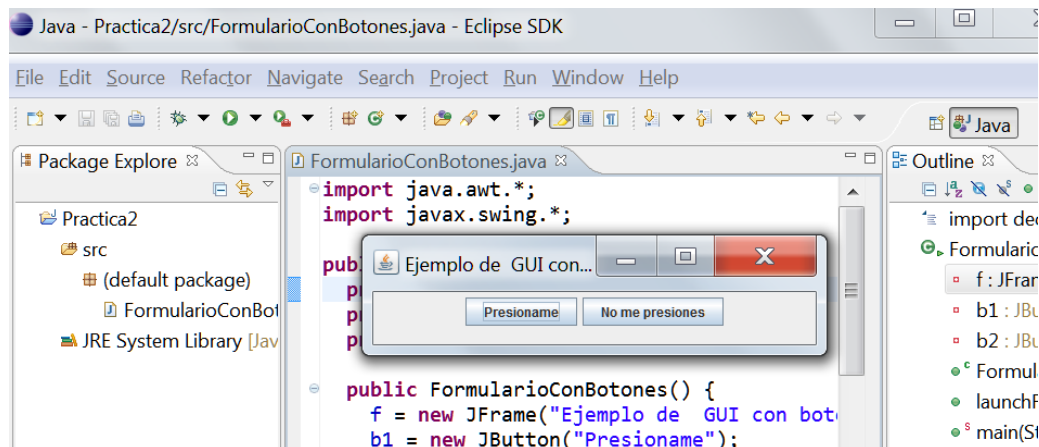


Eclipse .

Se selecciona del menú desplegado la opción Ejecutar o simplemente se hace clic en el botón ejecutar o se oprimen las teclas Ctrl F9, con lo cual se realiza la compilación y ejecución:



Y se observa el resultado.



JCreator .

En el árbol del proyecto, en el paquete de fuentes se selecciona el archivo correspondiente al programa FormularioConBotones.java y haciendo clic en el botón derecho del mouse se selecciona del menú desplegado la opción compilar o simplemente se oprime la tecla F9.

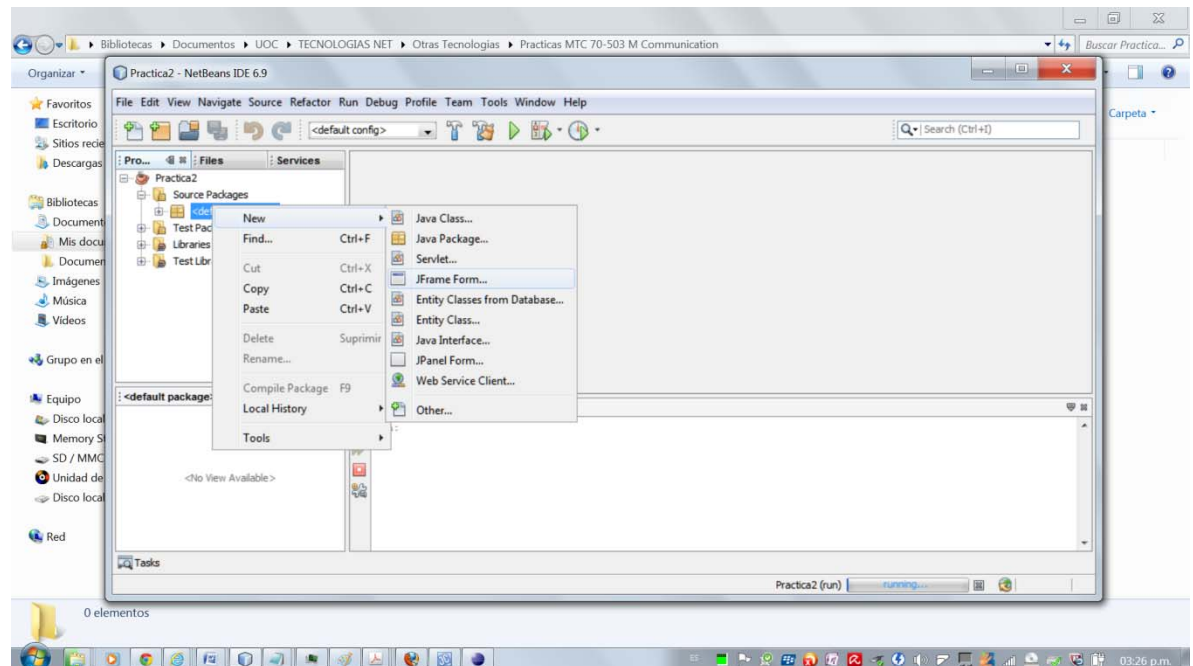
Nuevamente haciendo clic en el botón derecho del mouse se selecciona del menú desplegado la opción ejecutar o se oprimen a la vez las teclas “Mayúsculas y F6” y se observa el resultado.

En este paso se notan más diferencias, aunque en NetBeans se puede obviar la compilación y ejecutar directamente ya que la clase contiene el método `main()` la cual se compila previa ejecución; en Eclipse simplemente se ejecuta el programa y aunque existan otras clases con el método `main()` se ejecutará la que tenga el foco en ese momento.

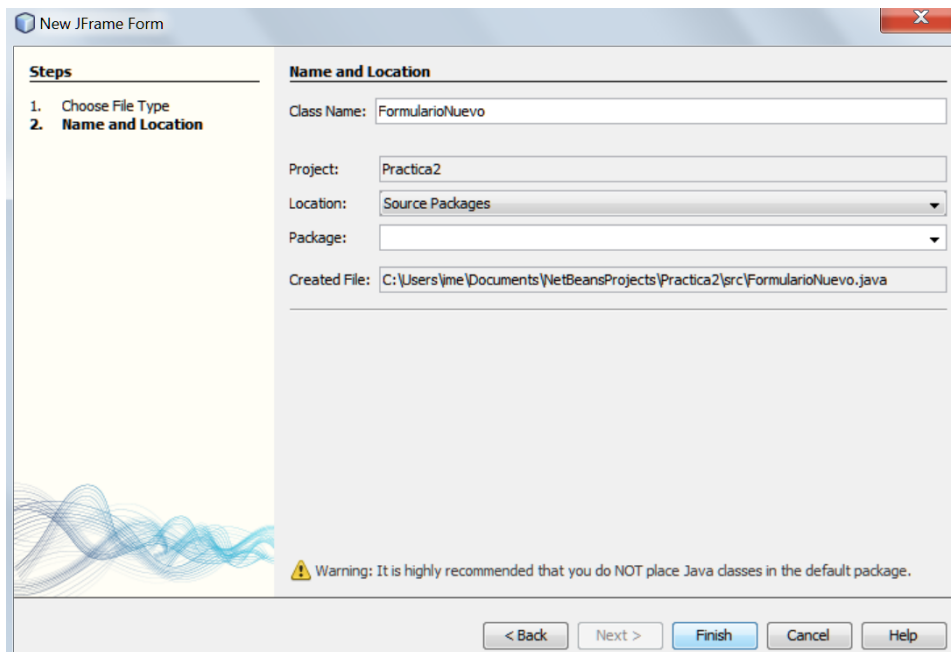
4. Para cada entorno de desarrollo crear formularios con los recursos que el mismo entorno proporciona y agregarle los mismos botones del programa anterior. Compilar y ejecutar el programa y al último describir las diferencias entre ellos.

Netbeans .

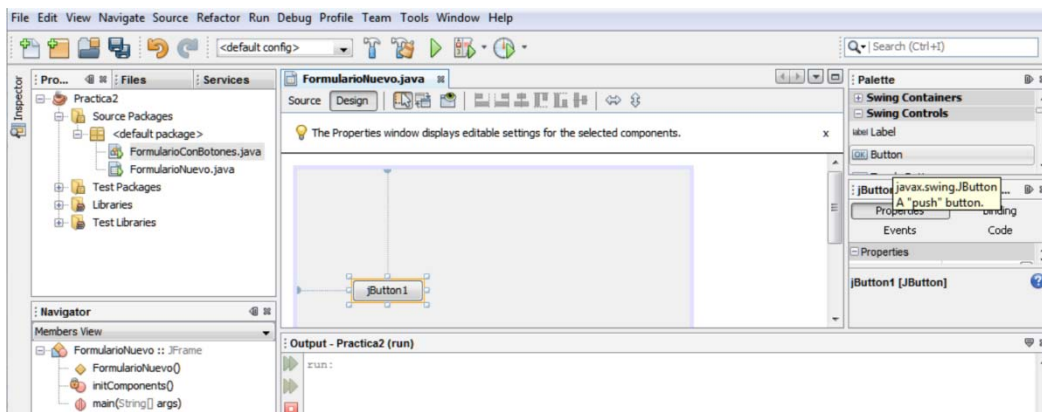
Con el botón derecho del mouse se elije nuevo formulario JFrame



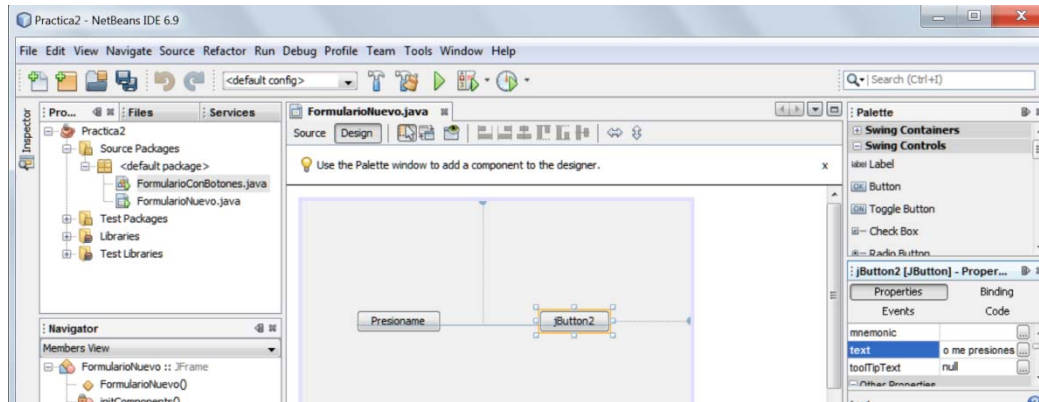
Y en la siguiente ventana de diálogo se le da un nombre y se da clic en Terminar



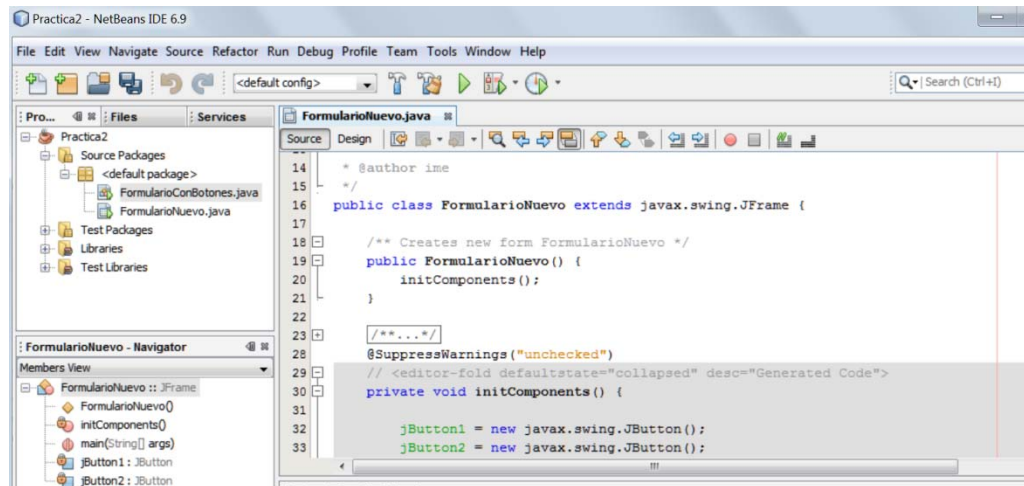
Con lo cual además de crearse el código se pasa a la vista de diseño y en esta podemos agregarle nuevos elementos gráficos a nuestra forma, simplemente seleccionándolos de la paleta y arrastrándolos al formulario:



Se selecciona cada botón y en la ventana de propiedades se hacen los cambios necesarios:



Y cambiando a vista de código fuente, puede observarse el código creado de manera automática:



Eclipse.

No proporciona la facilidad de crear una clase de tipo JFrame que añada el código básico y que pueda ser anexada al proyecto recién creado.

JCreator.

No proporciona la facilidad de crear una clase de tipo JFrame que añada el código básico y que pueda ser anexada al proyecto recién creado.

Ni Eclipse, ni JCreator proporcionan la facilidad de crear una clase de tipo JFrame que añada el código básico y que pueda ser anexada al proyecto

recién creado como lo hace Netbeans y tener una vista grafica que permita al usuario agregarle nuevos elementos gráficos al formulario recién creado.

En si Netbeans es un entorno más completo incluso para desarrolladores experimentados en el lenguaje, desde su primera instalación se le pueden anexar diversos módulos para aplicaciones web o móviles e incluso otros lenguajes. Por su parte Eclipse tiene la ventaja de ser gratuito y concentra al estudiante en la importancia del código que sea él y no el entorno el que lo escriba, lo mismo que JCreator LE, que tiene la desventaja de para que sea útil de una manera más profesional tiene un alto costo sobre todo para un estudiante.

10. Bibliografía preliminar

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 3

2. Nombre: **Aplicación de elementos de Interfaz de usuario como son botones, etiquetas y cajas de texto**

3. Objetivo: Comprender las diferencias y aplicación de elementos de interfaz de usuario como los botones y cajas de texto, y como se realiza el tratamiento de eventos con estos elementos.

4. Introducción. El alumno creará un programa combinando distintos componentes de interfaz de usuario que interactúen entre ellos a través de un evento, para que él o ella aprendan cual objeto escuchador es los más adecuado.

5. Temas y subtemas correlacionados con la práctica.

Tema II Interfaz Gráfica de Usuario

2.1. Creación de interfaz grafica para usuarios.

En esta práctica el alumno sigue practicando contenidos relacionados con el subtema de Creación de interfaz gráfica para usuarios; ampliando su dominio sobre los diferentes componentes gráficos que puede aplicar a los formularios como son botones, etiquetas y cajas de texto.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el subtema del tema dados en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. Crear un proyecto llamado Practica3, que tenga una clase java con método *"main"* llamada PruebaPractica3a y que cree una instancia de otra clase llamada Practica3a la cual herede de la clase **JFrame** e implemente la interfaz **ActionListener**, la cual debe crear una instancia de cada una de las siguientes clases: **JButton**, **JLabel** y **TextField**. La propiedad texto del botón deberá decir "Saludo", la de la Etiqueta deberá decir "Te saludo" mientras que la propiedad texto de la caja de texto deberá cambiar cada vez que el usuario haga un clic sobre el botón, así por ejemplo antes de oprimir el botón la caja de texto estará vacía y al hacer clic en el botón por primera vez dirá "1 vez", la segunda "2 veces", la tercera "3 veces" y así consecutivamente. El método *"actionPerformed"* de la interfaz es el que debe colocar el cambio en la caja de texto, se debe cambiar el **Layout** por defecto del **JFrame** a "FlowLayout". Ejecutar la clase PruebaPractica3a.
2. Dentro del proyecto Practica3 crear una clase java con método *"main"* llamada PruebaPractica3b y que cree una instancia de otra clase llamada Practica3b la cual herede de la clase **JFrame** e implemente la interfaz **ActionListener** y que cree las mismas instancias del programa anterior y tenga la misma funcionalidad, con la diferencia de que el usuario pueda utilizar la combinación de teclas "Alt S" para cambiar el texto en la caja de texto.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

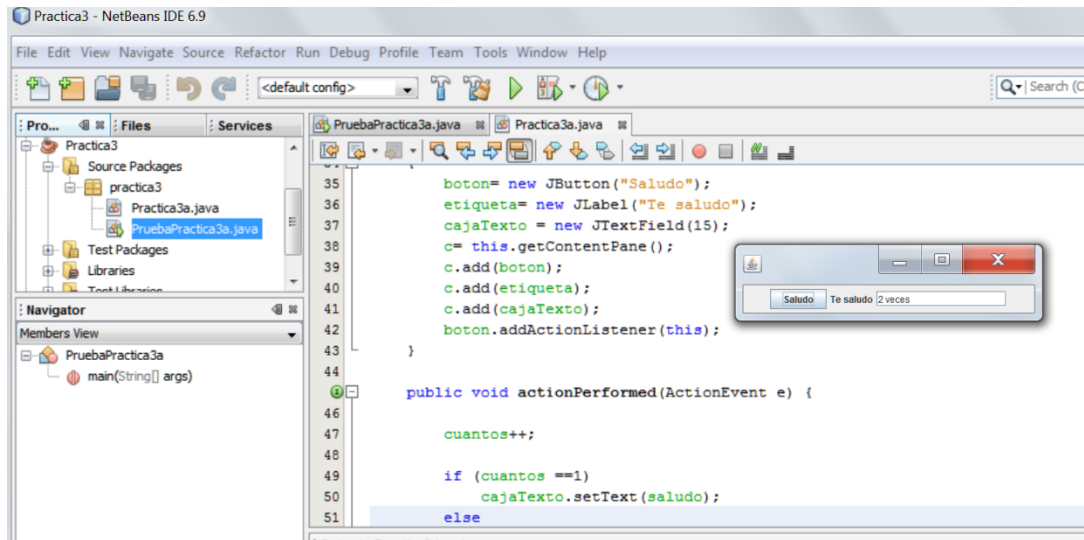
Resultados de la Práctica de acuerdo a lo indicado.

1. Crear un proyecto llamado Practica3, que tenga una clase java con método “*main*” llamada PruebaPractica3a y que cree una instancia de otra clase llamada Practica3a la cual herede de la clase **JFrame** e implemente la interfaz **ActionListener** , la cual debe crear una instancia de cada una de las siguientes clases: **JButton**, **JLabel** y **TextField**. La propiedad texto del botón deberá decir “Saludo”, la de la Etiqueta deberá decir “Te saludo” mientras que la propiedad texto de la caja de texto deberá cambiar cada vez que el usuario haga un clic sobre el botón, así por ejemplo antes de oprimir el botón la caja de texto estará vacía y al hacer clic en el botón por primera vez dirá “1 vez”, la segunda “2 veces”, la tercera “3 veces” y así consecutivamente. El método “*actionPerformed*” de la interfaz es el que debe colocar el cambio en la caja de texto, se debe cambiar el **Layout** por defecto del **JFrame** a “FlowLayout”. Ejecutar la clase PruebaPractica3a.

El código del método actionPerformed se muestra en las siguientes líneas:

```
public void actionPerformed(ActionEvent e) {  
  
    cuantos++;  
  
    if (cuantos ==1)  
        cajaTexto.setText(saludo);  
    else  
    {  
        saludo = Integer.toString(cuantos)+ " veces";  
        cajaTexto.setText(saludo);  
    }  
}
```

Y el resultado al ejecutar el programa y después de hacer dos veces clic en el botón se muestra a continuación:



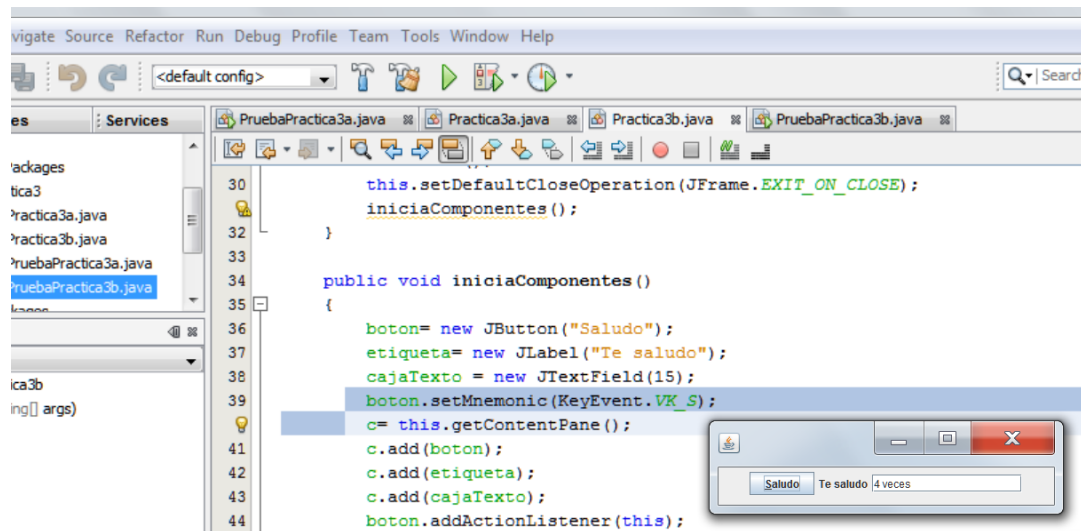
2. Dentro del proyecto Practica3 crear una clase java con método **“main”** llamada PruebaPractica3b y que cree una instancia de otra clase llamada Practica3b la cual herede de la clase **JFrame** e implemente la interfaz **ActionListener** y que cree las mismas instancias del programa anterior y tenga la misma funcionalidad, con la diferencia de que el usuario pueda utilizar la combinación de teclas “Alt S” para cambiar el texto en la caja de texto.

El código del método “actionPerformed” queda igual, lo único que cambia en el programa es que ahora el botón creado llama a su método “setMnemonic” que recibe como argumento el mnemónico correspondiente a la tecla que se desea que lance la acción.

```
boton.setMnemonic(KeyEvent.VK_S);
```

Esto da la ventaja de que tanto al oprimir las teclas “Alt s” como e hacer clic en el botón funcionan de la misma manera, es decir ambos eventos de teclado y de clic de ratón siguen utilizando la misma interfaz ActionListener y el mismo código en su método “actionPerformed”

El resultado de ejecutar PruebaPractica3b y oprimir la combinación “Alt s” dos veces y dos hacer clic en el botón se observa a continuación:



10. Bibliografía preliminar

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 4

2. Nombre: **Aplicación de elementos de Interfaz de usuario como son listas, radio, check buttons y menús**

3. Objetivo: Comprender las diferencias y aplicación de elementos de interfaz de usuario como listas, radio, check buttons y menús, y como se realiza el tratamiento de eventos con estos elementos.

4. Introducción. El alumno creará distintas clases que que contengan elementos gráficos y que reaccionen a distintos eventos; para que él o ella puedan identificar que objetos escuchadores son adecuados para cada componente gráfico de acuerdo al evento a manejar en él y como la crear los métodos para su control.

5. Temas y subtemas correlacionados con la práctica.

Tema II Interfaz Gráfica de Usuario

2.1. Creación de interfaz grafica para usuarios

En esta práctica el alumno sigue practicando contenidos relacionados con el subtema de Creación de interfaz gráfica para usuarios; ampliando su dominio sobre los diferentes componentes gráficos que puede aplicar a los formularios como son listas, radio, check buttons y menús realizando variaciones de ejemplos de los libros de texto y analizados en clase, para que no sólo aplique los conocimientos vistos en clase sino que vaya más allá reformando elementos e incorporando nuevos.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% de cada subtema del tema dado en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. Crear un proyecto llamado Practica4 que tenga una clase java con método **“main”** llamada **“PruebaPractica4”** y que cree una instancia de otra clase llamada **“Practica4a”** la cual herede de la clase **JFrame** y cree una instancia de la clases: **JLabel** y **JList**, la propiedad de Texto de la etiqueta dirá inicialmente **“Color negro”** y la fuente del texto debe ser por lo tanto de color negro y el **JList** debe contener los siguientes elementos: **“Negro”, “Azul”, “Cyan”, “Gris oscuro”, “Gris”, “Verde”, “Gris claro”, “Magenta”, “Naranja”, “Rosa”, “Rojo”, “Blanco”** y **“Amarillo”** y a esta lista se le debe añadir en el método **“addListSelectionListener”** una instancia de una clase anónima que implemente la interfaz **ListSelectionListener**, en cuyo método **“valueChanged”**, cada vez que el usuario seleccione un elemento distinto de la lista, la etiqueta cambiará de texto y color según corresponda. Ejecutar la clase **PruebaPractica4**.
2. Anexarle al proyecto Practica4, una clase llamada Practica4b la cual herede de la clase **JFrame** y cree una instancia de la clase **JLabel**, una instancia de la clase **ButtonGroup**, cinco de la clase **JRadioButton** y cinco instancias de la clase **Icon**, apartir de 5 gif de 5 animales distintos.
La clase Practica4b tendrá una clase interna llamada **“RadioButtonHandler”** la cual en su constructor recibirá una instancia del tipo **Icon**, y en su método **itemStateChanged** cambiará el icono de la etiqueta cada vez que se selecciona un distinto **JRadioButton**.
En la clase **PruebaPractica4**, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase **Practica4b** y se ejecutará el programa.
3. Al mismo proyecto Practica4, anexarle una clase llamada Practica4c, que tendrá una funcionalidad similar a la del punto anterior (punto 2), pero en lugar de utilizar instancias de la clase **JRadio Button** se deberá aplicar un menú con cinco opciones una por cada gif.

En la clase PruebaPractica4, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase Practica4c y se ejecutará el programa.

4. Agregar al proyecto Practica4 una clase llamada Practica4d la cual herede de la clase **JFrame** y cree una instancia de la clase **JLabel** con la propiedad Texto "Texto a probar cambio de estilo", una instancia de la clase **ButtonGroup**, dos de la clase **JCheckBox**, y en su método `addItemListener` cree una instancia de una clase interna `CheckBoxHandler`, que implemente la interfaz **ItemListener** y en el método `"itemStateChanged"` de ésta, se varíe el aspecto de la fuente de la etiqueta de "normal" a "negrita" o a "itálica" o a "negrita + itálica", dependiendo de las casillas de verificación seleccionadas.

En la clase PruebaPractica4, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase Practica4d y se ejecutará el programa.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

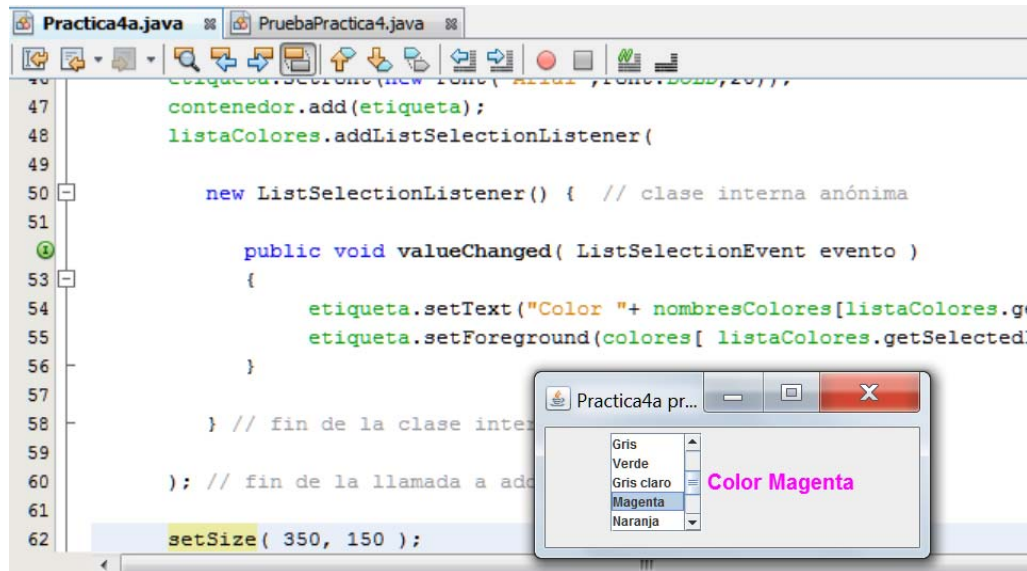
Resultados de la Práctica de acuerdo a lo indicado.

1. Crear un proyecto llamado Practica4 que tenga una clase java con método **“main”** llamada “PruebaPractica4” y que cree una instancia de otra clase llamada “Practica4a” la cual herede de la clase **JFrame** y cree una instancia de la clases: **JLabel** y **JList**, la propiedad de Texto de la etiqueta dirá inicialmente “Color negro” y la fuente del texto debe ser por lo tanto de color negro y el **JList** debe contener los siguientes elementos: "Negro", "Azul", "Cyan", "Gris oscuro", "Gris", "Verde", "Gris claro", "Magenta", "Naranja", "Rosa", "Rojo", "Blanco" y "Amarillo" y a esta lista se le debe añadir en el método “addListSelectionListener” una instancia de una clase anónima que implemente la interfaz **ListSelectionListener**, en cuyo método “valueChanged”, cada vez que el usuario seleccione un elemento distinto de la lista, la etiqueta cambiará de texto y color según corresponda. Ejecutar la clase PruebaPractica4.

La instancia anónima que implementa la interfaz ListSelectionListener con el método valueChanged y que se ponen como argumento al método addListSelection del objeto JList llamado listaColores, se muestra en las siguientes líneas:

```
listaColores.addListSelectionListener(  
    new ListSelectionListener() { // clase interna anónima  
        public void valueChanged( ListSelectionEvent evento )  
        {  
            etiqueta.setText("Color "+  
nombresColores[listaColores.getSelectedIndex()]);  
            etiqueta.setForeground(colores[ listaColores.getSelectedIndex() ]  
);  
        }  
    } //fin de la clase interna anónima  
);
```

Y el resultado al ejecutar la clase PruebaPractica4 y haber seleccionado el color magenta, se muestra a continuación:



2. Anexarle al proyecto Practica4, una clase llamada Practica4b la cual herede de la clase **JFrame** y cree una instancia de la clase **JLabel**, una instancia de la clase **ButtonGroup**, cinco de la clase **JRadioButton** y cinco instancias de la clase **Icon**, apartir de 5 gif de 5 animales distintos.

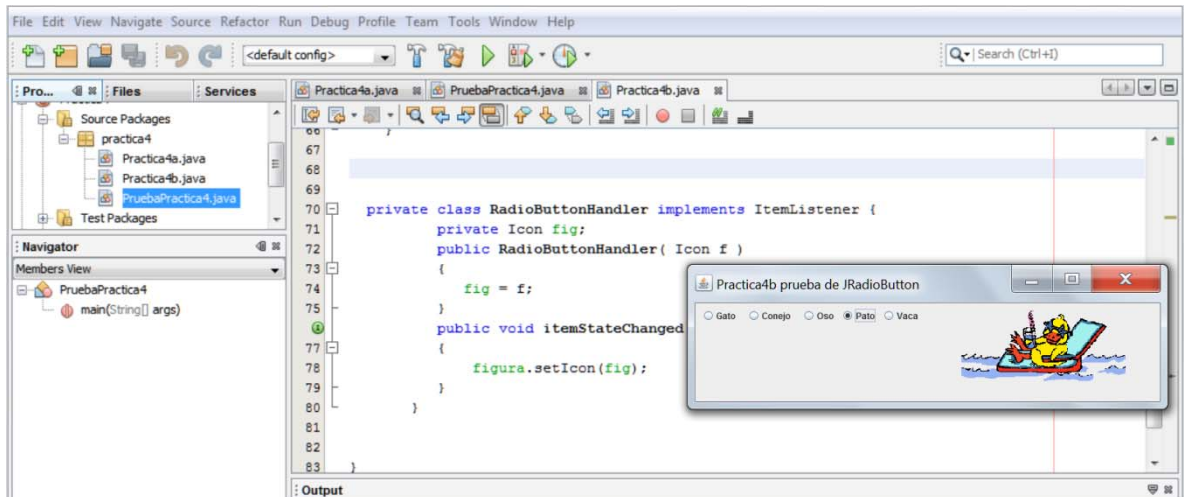
La clase Practica4b tendrá una clase interna llamada "RadioButtonHandler" la cual en su constructor recibirá una instancia del tipo **Icon**, y en su método `itemStateChanged` cambiará el icono de la etiqueta cada vez que se selecciona un distinto **JRadioButton**.

En la clase `PruebaPractica4`, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase `Practica4b` y se ejecutará el programa.

El código de clase interna `RadioButtonHandler` se muestra a continuación:

```
private class RadioButtonHandler implements ItemListener {
    private Icon fig;
    public RadioButtonHandler( Icon f )
    {
        fig = f;
    }
    public void itemStateChanged( ItemEvent event )
    {
        figura.setIcon(fig);
    }
}
```

Y el resultado al ejecutar la clase PruebaPractica4 y haber seleccionado el color magenta, se muestra a continuación:



3. Al mismo proyecto Practica4, anexarle una clase llamada Practica4c, que tendrá una funcionalidad similar a la del punto anterior (punto 2), pero en lugar de utilizar instancias de la clase JRadio Button se deberá aplicar un menú con cinco opciones una por cada gif.

En el caso del componente JMenuItem se requiere crear una clase que implemente no el `ItemListener`, sino el `ActionListener`, por lo que se hace una clase interna llamada `MenuHandler` que implemente esta interfaz. El código de clase interna `MenuHandler` se muestra a continuación y como puede observarse es casi idéntico al de la clase interna anterior:

```
private class MenuHandler implements ActionListener {
    private Icon fig;
    public MenuHandler( Icon f )
    {
        fig = f;
    }
    public void actionPerformed(ActionEvent e) {

        if (e.getSource()==gato)

            figura.setIcon(imagCat);

        if (e.getSource()==conejo)
```

```

        figura.setIcon(imagRabbit);

    if (e.getSource()==oso)
        figura.setIcon(imagBear);

    if (e.getSource()==pato)
        figura.setIcon(imagCuac);

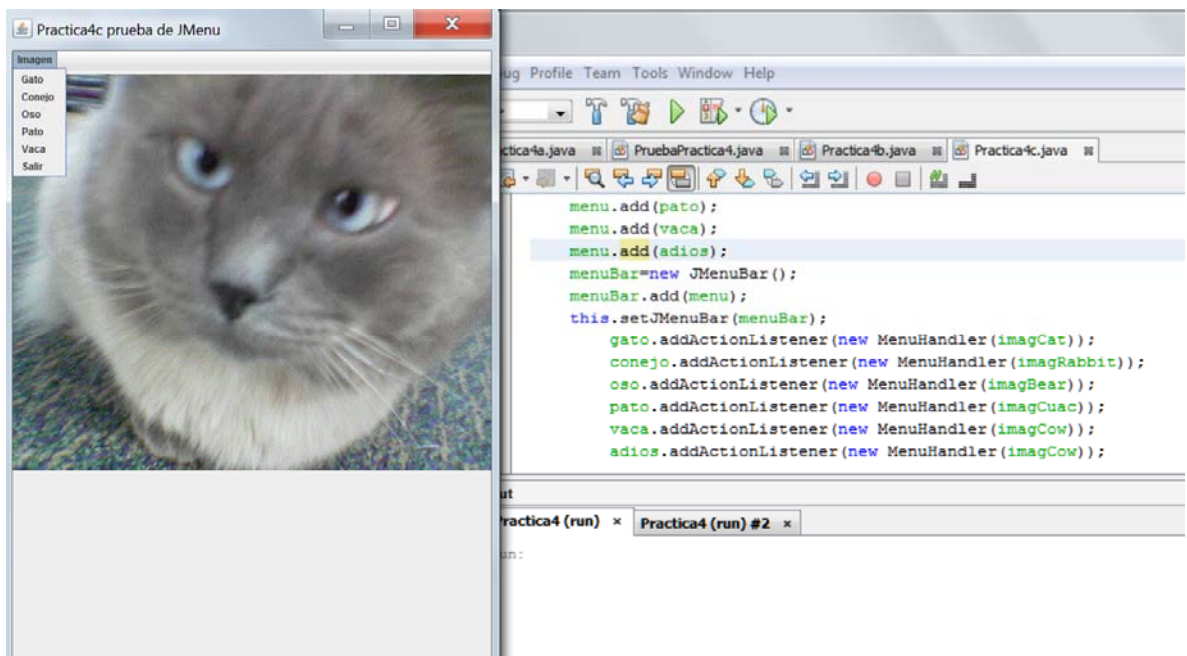
    if (e.getSource()==vaca)
        figura.setIcon(imagCow);

    if (e.getSource()==adios)
    {
        System.out.println("Adios.....");
        System.exit(0);
    }

}

```

Y el resultado al ejecutar la clase PruebaPractica4 y haber seleccionado la opción de menú gato , se muestra a continuación:



4. Agregar al proyecto Practica4 una clase llamada Practica4d la cual herede de la clase **JFrame** y cree una instancia de la clase **JLabel** con la propiedad Texto "Texto a probar cambio de estilo", una instancia de la clase **ButtonGroup**, dos de la clase **JCheckBox**, y en su método `addItemListener` cree una instancia de una clase interna `CheckBoxHandler`, que implemente la interfaz **ItemListener** y en el método *"itemStateChanged"* de ésta, se varíe el aspecto de la fuente de la etiqueta de "normal" a "negrita" o a "itálica" o a "negrita + itálica", dependiendo de las casillas de verificación seleccionadas.

En la clase `PruebaPractica4`, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase `Practica4d` y se ejecutará el programa.

El código de clase interna `CheckBoxHandler` se muestra a continuación:

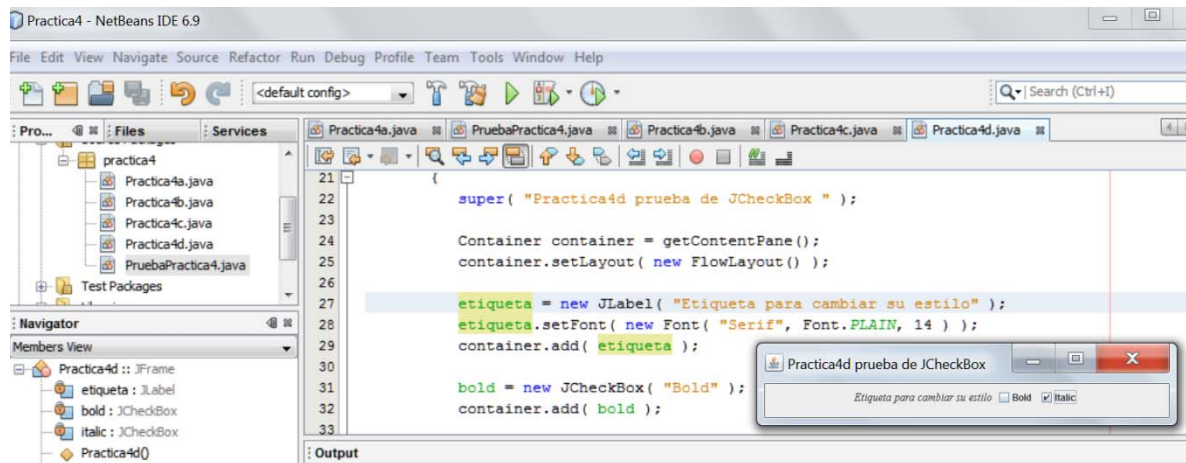
```
private class CheckBoxHandler implements ItemListener {
    private int valBold = Font.PLAIN;
    private int valItalic = Font.PLAIN;

    public void itemStateChanged( ItemEvent event )
    {
        if ( event.getSource() == bold )
            valBold = bold.isSelected() ? Font.BOLD : Font.PLAIN;

        if ( event.getSource() == italic )
            valItalic = italic.isSelected() ? Font.ITALIC : Font.PLAIN;

        etiqueta.setFont( new Font( "Serif", valBold + valItalic, 14 )
    );
    }
}
```

Y el resultado al ejecutar la clase `PruebaPractica4` y haber seleccionado la casilla de verificación `italic`, se muestra a continuación:



10. Bibliografía preliminar

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 5

2. Nombre: Implementación de elementos de Interfaz de usuario que incluyan el desarrollo de computación gráfica como son el dibujo de líneas, círculos, rectángulos, texto y polígonos

3. Objetivo: Implementar los métodos correspondientes para que elementos de Interfaz de usuario que lleven a cabo el dibujo de líneas, texto y polígonos.

4. Introducción. El alumno creará programas que incorpore componentes gráficos y eventos cuyo tratamiento se realice con métodos que creen dibujos.

5. Temas y subtemas correlacionados con la práctica.

Tema II Interfaz Gráfica de Usuario

2.2. Computación Grafica

En esta práctica el alumno sigue practicando contenidos relacionados con el subtema de Creación de interfaz gráfica para usuarios; pero integrando ahora el subtema de la computación gráfica, para que aprenda cómo estos elementos pueden interactuar entre sí.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% de cada subtema del tema dado en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. Crear un proyecto llamado Practica5 que tenga una clase java con método “*main*” llamada “PruebaPractica5” y que cree una instancia de otra clase llamada “Practica5a” la cual herede de la clase **JFrame** y cree tres instancias de la clases: **JPanel** y **JButton**, la propiedad de Texto de la primera instancia de la clase **JButton** será “Texto”, de la segunda “Círculo” y de la tercera “Cuadrado”. El **Layout** que se debe aplicar al **JFrame** debe ser **GridLayout** con una fila y tres columnas para colocar sobre el **JFrame** las tres instancias del **JPanel** y sobre cada panel irá un botón y una instancia de una clase interna llamada Dibujo que en su constructor recibe información sobre qué tipo de dibujo se debe trazar en su método “*paint*”. Ejecutar la clase PruebaPractica5.
2. Al mismo proyecto Practica5, anexarle una clase llamada Practica5b, que derive de la clase JFrame e implemente las interfaces ActionListener y MouseMotinListener y tenga un menú con opciones de colores azul, rojo, verde, amarillo y negro para cambiar el color de las líneas libres que serán dibujadas por el usuario cuando el usuario oprime y arrastre el mouse a través de la ventana.
En la clase PruebaPractica5, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase Practica5b y se ejecutará el programa.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. Crear un proyecto llamado Practica5 que tenga una clase java con método “*main*” llamada “PruebaPractica5” y que cree una instancia de otra clase llamada “Practica5a” la cual herede de la clase **JFrame** y cree tres instancias de la clases: **JPanel** y **JButton**, la propiedad de Texto de la primera instancia de la clase **JButton** será “Texto”, de la segunda “Círculo” y de la tercera “Cuadrado”. El **Layout** que se debe aplicar al **JFrame** debe ser **GridLayout** con una fila y tres columnas para colocar sobre el **JFrame** las tres instancias del **JPanel** y sobre cada panel irá un botón y una instancia de una clase interna llamada Dibujo que en su constructor recibe información sobre qué tipo de dibujo se debe trazar en su método “*paint*”. Ejecutar la clase PruebaPractica5.

El código de la clase Dibujo se muestra a continuación:

```
class Dibujo extends JPanel {  
  
    int d;  
    public Dibujo(int dibujo){  
        d = dibujo;  
    }  
  
    public void paint(Graphics g){  
  
        if (d==1)  
        {  
            g.setColor(Color.BLACK);  
            g.drawString("Hola", 50, 40);  
        }  
        if (d==2)  
        {  
            g.setColor(Color.red);  
            g.fillRect(46, 16, 40, 40);  
        }  
        if (d==3)  
        {
```

```

        g.setColor(Color.blue);
        g.fillOval(46, 16, 40, 40);
    }
}

```

Mientras que el código del método “actionPerformed” es el siguiente:

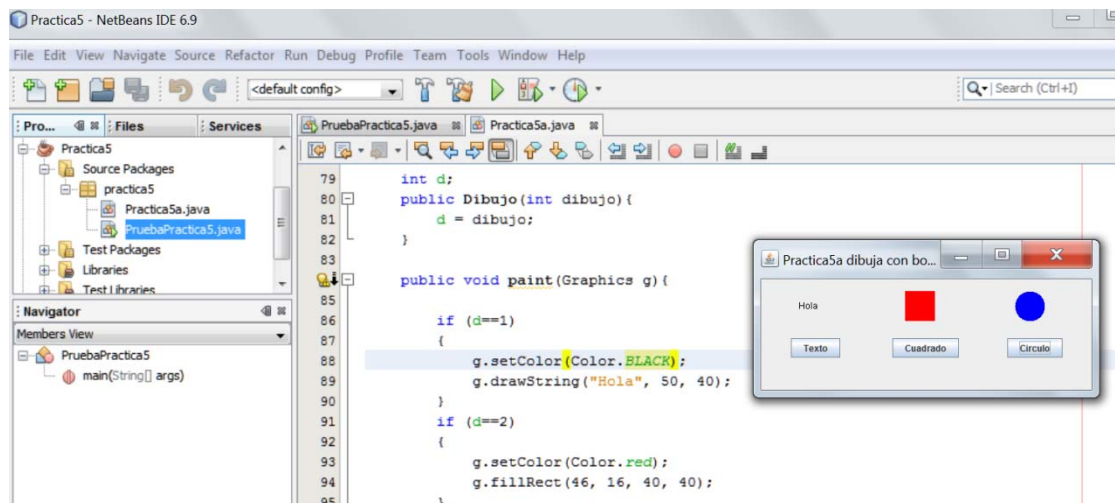
```

public void actionPerformed(ActionEvent e) {

    if (e.getSource()==btexto){
        celda_1_1.setVisible(true);
    }
    if (e.getSource()==brecta){
        celda_1_2.setVisible(true);
    }
    if (e.getSource()==bcircu){
        celda_1_3.setVisible(true);
    }
}

```

Y el resultado al ejecutar la clase PruebaPractica5 al haber hecho clic en cada uno de los tres botones, se muestra a continuación:



- Al mismo proyecto Practica5, anexarle una clase llamada Practica5b, que derive de la clase JFrame e implemente las interfaces ActionListener y MouseMotinListener y tenga un menú con opciones de colores azul, rojo, verde, amarillo y negro para cambiar el color de las líneas libres que serán dibujadas por el usuario cuando el usuario oprime y arrastre el mouse a través de la ventana.

En la clase PruebaPractica5, se eliminará la clase del punto anterior y en su lugar se creará una instancia de la clase Practica5b y se ejecutará el programa.

El código del método ya implementado "actionPerformed" de la interfaz ActionListener se muestra a continuación:

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==azul)
        color = colores[0];
        if (e.getSource()==rojo)
            color=colores[1];
    if (e.getSource()==verde)
        color=colores[2];
    if (e.getSource()==amarillo)
        color=colores[3];
    if (e.getSource()==negro)
        color=colores[4];
}
```

Y el código de los métodos ya implementados "mousePressed", "mouseDragged" y "mouseMoved", de la interfaz MouseMotionListener se muestran a continuación:

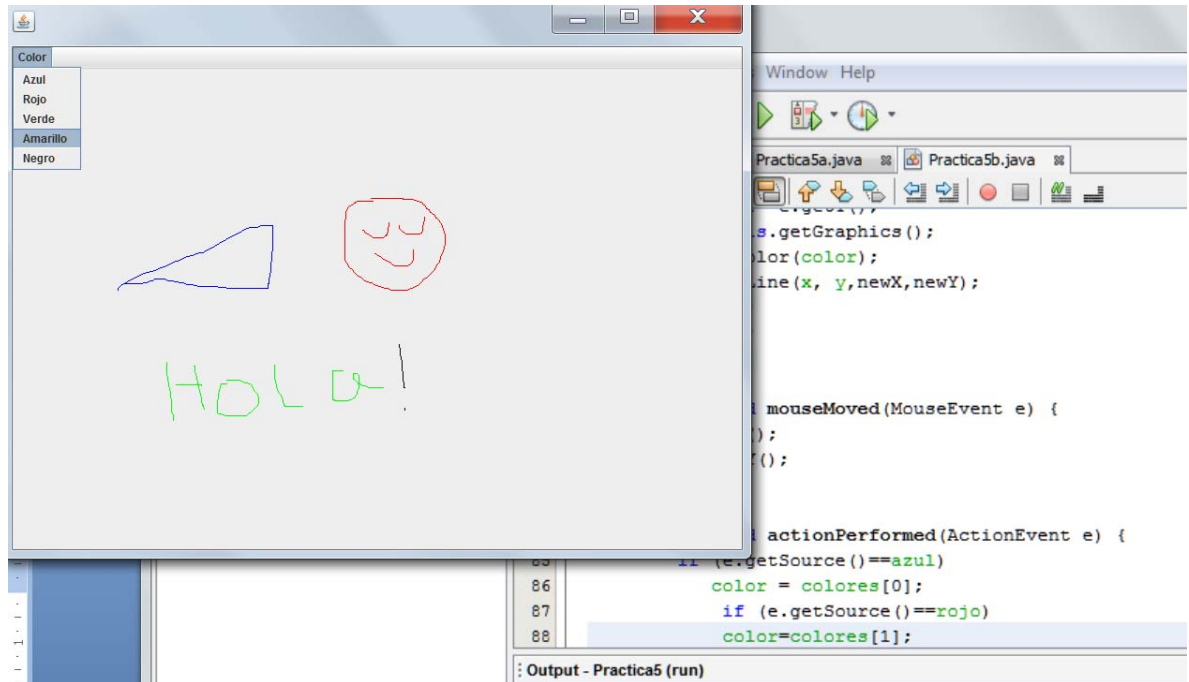
```
public void mousePressed(MouseEvent e) {
    g = this.getGraphics();
    g.setColor(color);
    x=e.getX();
    y= e.getY();
}

public void mouseDragged(MouseEvent e) {
    int newX=e.getX();
    int newY= e.getY();
    g = this.getGraphics();

    g.setColor(color);
    g.drawLine(x, y,newX,newY);
    x=newX;
    y=newY;
}

public void mouseMoved(MouseEvent e) {
    x=e.getX();
    y= e.getY();
}
```

Y el resultado al ejecutar la clase PruebaPractica5 al haber seleccionado diferentes colores para dibujar y hacer trazos con el botón del clic del mouse presionado y arrastrando a éste, se muestra a continuación:



10. Bibliografía preliminar

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 6

2. Nombre: Utilización de componentes proporcionados por el lenguaje

3. Objetivo: Conocer y aplicar las componentes que proporciona el lenguaje utilizado

4. Introducción. El alumno aplicará componentes tanto visuales como no visuales que proporciona el lenguaje y observará como estas pueden estar contenidas en los paquetes/librerías.

5. Temas y subtemas correlacionados con la práctica.

Tema III Componentes y librerías.

3.1 Definición conceptual de componentes, paquetes/librerías.

3.2 Uso de componentes (visuales y no visuales) proporcionados por el lenguaje.

El alumno comprenderá el significado de componentes y su diferencia con los paquetes/librerías, al aplicarlas en la práctica a desarrollar y podrá distinguir las características que difieren de componentes visuales y no visuales.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% los dos primeros subtemas del tema dado en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. Acceder a la página web
<http://docs.oracle.com/javase/1.4.2/docs/api/javax/swing/JFileChooser.html>
y responder lo siguiente:
 - a. ¿Es el JFileChooser es un componente o un paquete/librería?
 - b. Diga si JFileChooser es un componente visual o un paquete/librería visual o un componente no visual o un paquete/librería no visual.
 - c. Poniendo a JFileChooser como ejemplo explicar la diferencia entre un componente y un paquete/librería.
2. Crear un Proyecto que se llame Practica6 y crear un nueva clase llamada PruebaPractica6 que tenga el método "main" y que cree una instancia de la clase Practica6a cuyo código se muestra a continuación y ejecutar la clase PruebaPractica6

```
import javax.swing.JFileChooser;
import javax.swing.JFrame;

public class Practica6a extends JFrame {

    public Practica6a() {

        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Selecciona un archivo");
        this.getContentPane().add(fileChooser);
        fileChooser.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.pack();
        this.setVisible(true);

    }
}
```

3. Acceder a la página web
<http://docs.oracle.com/javase/1.4.2/docs/api/java/io/FileReader.html>
y responder lo siguiente:
 - a. ¿Es el FileReader es un componente o un paquete/librería?
 - b. Diga si FileReader es un componente visual o un paquete/librería visual o un componente no visual o un paquete/librería no visual.

4. Al proyecto Practica6 agregar una nueva clase llamada Practica6b cuyo código se muestra a continuación:

```
import java.io.FileReader;

public class Practica6b {
    public static void main(String[] argv) throws Exception {
        FileReader fr = new FileReader("texto.txt");

        int ch;
        do {
            ch = fr.read();
            if (ch != -1)
                System.out.print((char) ch);
        } while (ch != -1);
        System.out.println("");
        fr.close();
    }
}
```

Utilizando un editor de texto simple como el bloc de notas, escribir lo siguiente:

Este es un archivo de texto que sirve para probar el programa Practica6b

Y guardarlo en la carpeta del proyecto con el nombre de "texto.txt" y ejecutar el programa Practica6b.

5. Crear una clase llamada Practica6c que derive de **JFrame** implemente **ActionListener** utilice **JMenu**, **JMenuItem**, **JTextArea**, **JFileChooser** y **FileReader** y que a través de una opción del menú busque un archivo y despliegue su contenido en un área de texto.
- En la clase PruebaPractica6, se eliminará la clase Practica6a y en su lugar se creará una instancia de la clase Practica6c y se ejecutará el programa

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. Acceder a la página web

<http://docs.oracle.com/javase/1.4.2/docs/api/javax/swing/JFileChooser.html>

y responder lo siguiente:

a. ¿Es el JFileChooser es un componente o un paquete/librería?

R. JFileChosser es un componente que se deriva de la clase JComponent

b. Diga si JFileChooser es un componente visual o un paquete/librería visual o un componente no visual o un paquete/librería no visual.

R. JFileChosser es un componente visual ya pertenece al paquete `javax.swing`.

Poniendo a JFileChooser como ejemplo explicar la diferencia entre un componente y un paquete/librería.

Como se mencionó en los puntos anteriores JFileChooser es una componente que pertenece al paquete `javax.swing`, es decir un paquete/librería está compuesto por un conjunto de componentes y una componente es una clase predefinida que puede ser reutilizada en alguna aplicación que utilice componentes, en este caso gráficos o visuales.

2. Crear un Proyecto que se llame Practica6 y crear un nueva clase llamada PruebaPractica6 que tenga el método "main" y que cree una instancia de la clase Practica6a cuyo código se muestra a continuación y ejecutar la clase PruebaPractica6

```

import javax.swing.JFileChooser;
import javax.swing.JFrame;

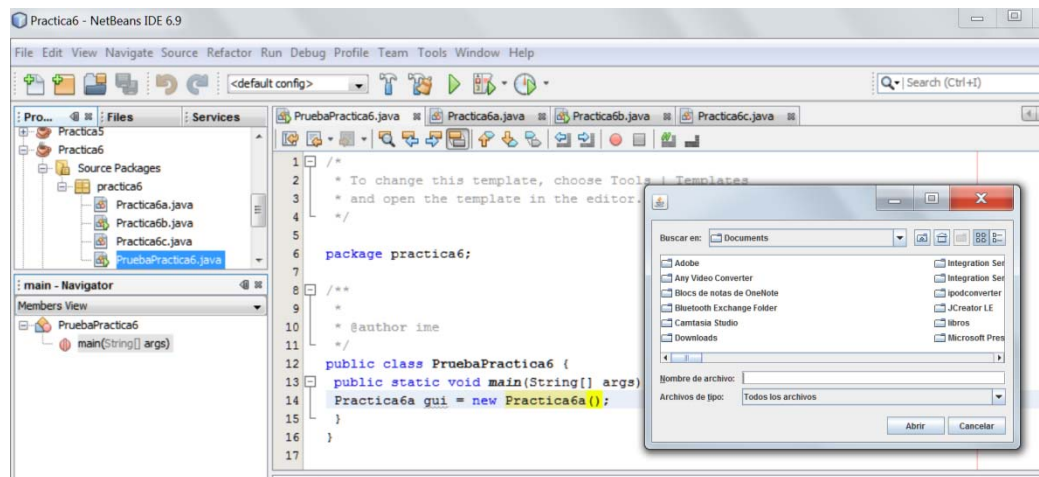
public class Practica6a extends JFrame {

    public Practica6a() {

        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Selecciona un archivo");
        this.getContentPane().add(fileChooser);
        fileChooser.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.pack();
        this.setVisible(true);
    }
}

```

Y el resultado al ejecutar la clase PruebaPractica6 después de haber creado la clase Practica6a, se muestra a continuación:



3. Acceder a la página web

<http://docs.oracle.com/javase/1.4.2/docs/api/java/io/FileReader.html> y responder lo siguiente:

- ¿Es el FileReader es un componente o un paquete/librería?
R. FileReader es una componente ya que pertenece al paquete java.io y deriva de la clase InputStreamReader
- Diga si FileReader es un componente visual o un paquete/librería visual o un componente no visual o un paquete/librería no visual.

R. **FileReader** es una componente no visual, ya que pertenece a un paquete que no maneja elementos visuales.

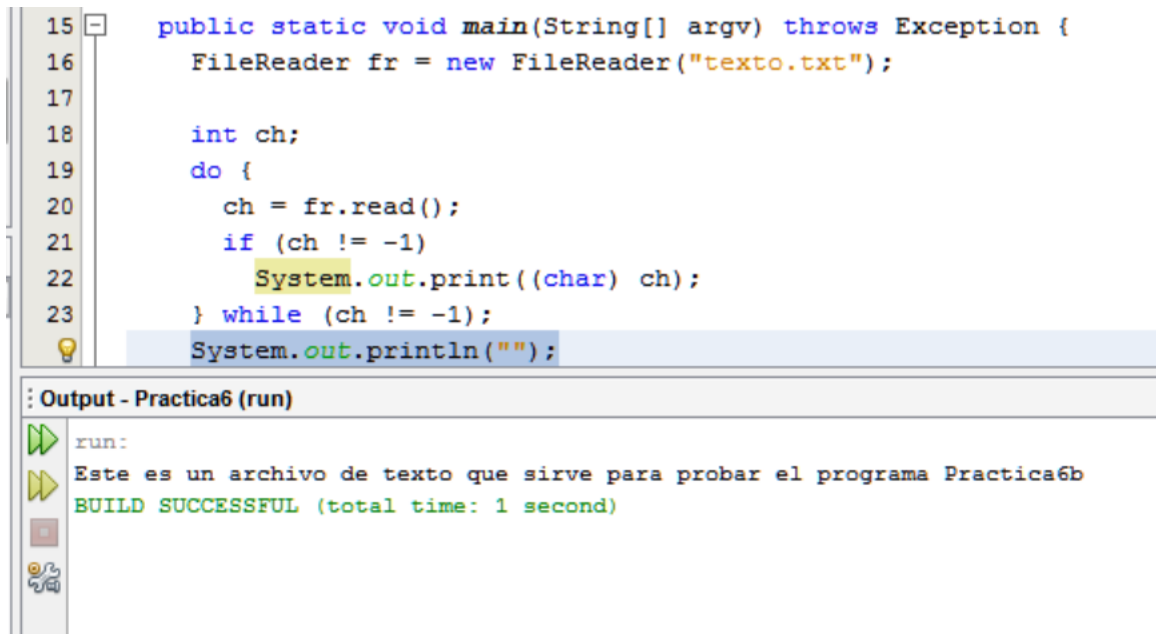
4. Al proyecto Practica6 agregar una nueva clase llamada Practica6b cuyo código se muestra a continuación:

```
import java.io.FileReader;

public class Practica6b {
    public static void main(String[] argv) throws Exception {
        FileReader fr = new FileReader("texto.txt");

        int ch;
        do {
            ch = fr.read();
            if (ch != -1)
                System.out.print((char) ch);
        } while (ch != -1);
        System.out.println("");
        fr.close();
    }
}
```

Y el resultado al ejecutar la clase Practica6b, se muestra a continuación:



The screenshot displays an IDE with two panels. The top panel shows the Java code for the `Practica6b` class, which reads the contents of `texto.txt` character by character and prints them. The bottom panel, titled "Output - Practica6 (run)", shows the execution results. It indicates that the program ran successfully, displaying the message "Este es un archivo de texto que sirve para probar el programa Practica6b" and confirming a "BUILD SUCCESSFUL (total time: 1 second)".

6. Crear una clase llamada Practica6c que derive de **JFrame** implemente **ActionListener** utilice **JMenu**, **JMenuItem**, **JTextArea**, **JFileChooser** y

FileReader y que a través de una opción del menú busque un archivo y despliegue su contenido en un área de texto.

En la clase PruebaPractica6, se eliminará la clase Practica6a y en su lugar se creará una instancia de la clase Practica6c y se ejecutará el programa

El código del método "actionPerformed" se muestra a continuación:

```
public void actionPerformed(ActionEvent e) {

    if (e.getSource()==salir)
    {
        System.out.println("Adios.....");
        System.exit(0);

    }

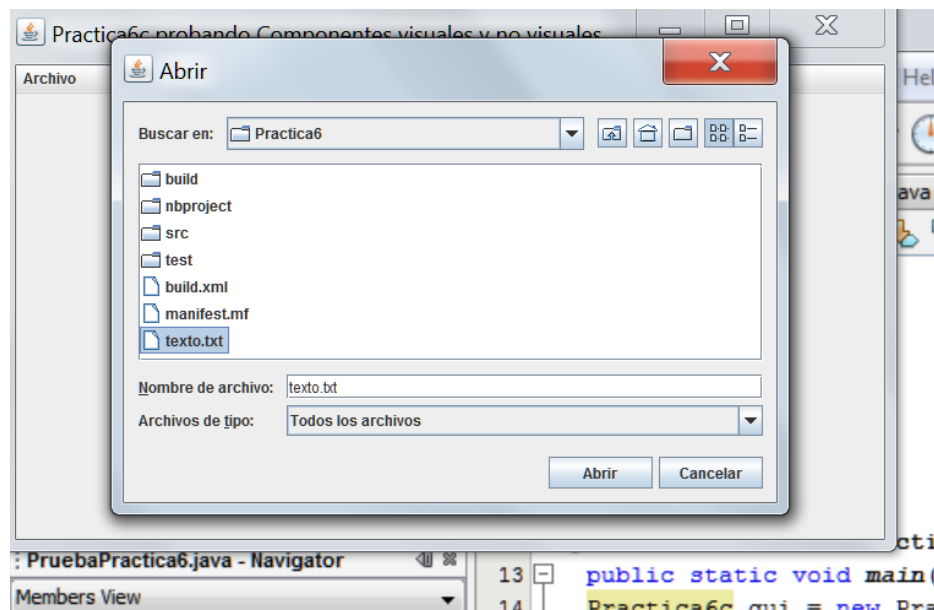
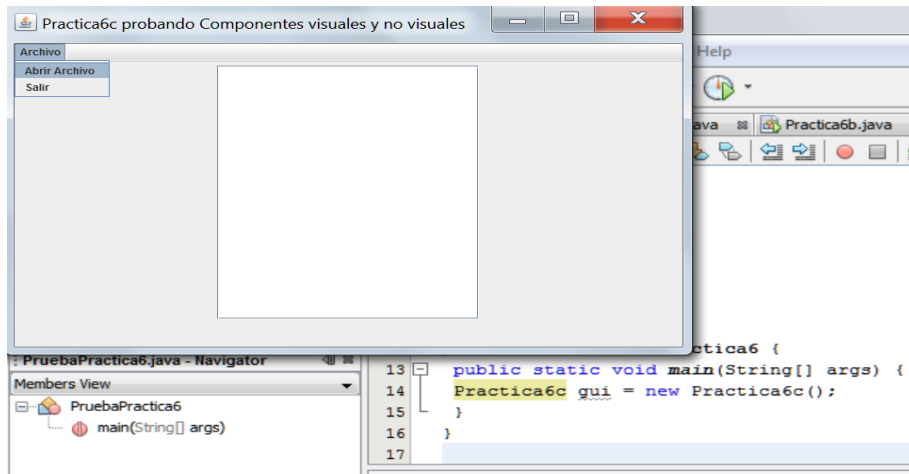
    if (e.getSource()==abrir)
    {
        cargaArchivo();
    }
}
```

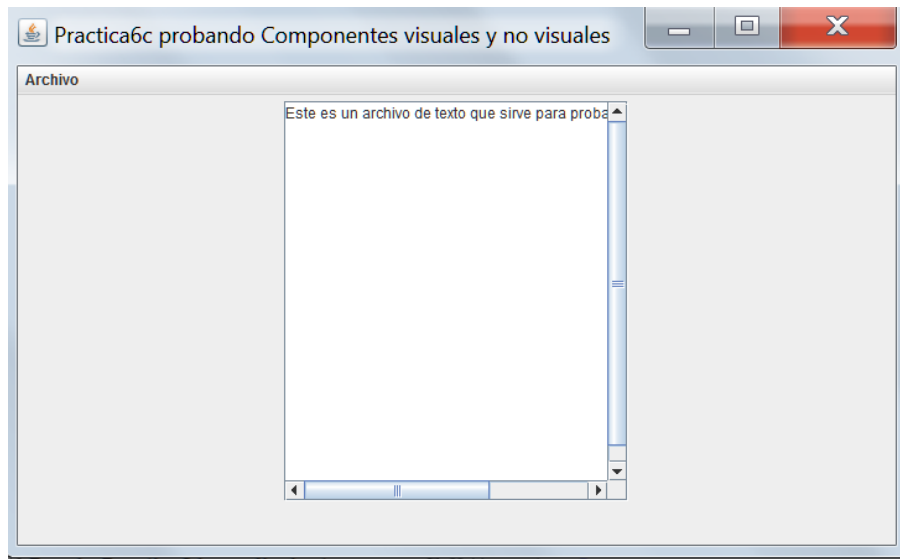
Y el código de la clase cargaArchivo es el siguiente:

```
private void cargaArchivo(){

try{
    JFileChooser fc=new JFileChooser(System.getProperty("user.dir"));
    fc.showOpenDialog(this);
    File Abrir=fc.getSelectedFile();
    if(Abrir!=null){
        Texto.setText("");
        FileReader Fichero=new FileReader(Abrir);
        BufferedReader leer=new BufferedReader(Fichero);
        while((Text=leer.readLine())!=null){
            Texto.append(Text+ "\n");
        }
        leer.close();
    }
}
catch(IOException ioe){
    System.out.println(ioe);
}
}
```

Y el resultado al ejecutar la clase PruebaPractica6, después de cambiar el código ahora creando una instancia de la clase Practica6c en lugar de Practica6a, se muestra a continuación:





10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 7

2. Nombre: Uso de librerías proporcionadas por el lenguaje

3. Objetivo: Aplicar algunas librerías que proporciona el lenguaje utilizado

4. Introducción. En esta práctica el alumno utilizará librerías gráficas y no gráficas que le permitan elaborar programas que se conecten con elementos ajenos al programa como una base de datos y poder consultar los datos contenidos en ella para tomar decisiones con respecto a lo que el programa hará, dependiendo de esa información.

5. Temas y subtemas correlacionados con la práctica.

Tema III Componentes y librerías

3.3 Uso de librerías proporcionadas por el lenguaje

El alumno aprenderá no sólo a utilizar librerías proporcionadas por el lenguaje sino a incorporar nuevas librerías que se han desarrollado específicamente por terceros para el lenguaje ampliando así su conocimiento del mismo

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el subtema del tema dado correspondiente en la teoría de la asignatura.
- b. Tener instalado el manejador de Base de Datos MySQL.
- c. Haber descargado e instalado el MySQL Connector/J de la página web <http://www.mysql.com/products/connector/>, en su computadora.
- d. Haber ejecutado el programa de MySQL en línea de comandos como administrador y crear un usuario y su correspondiente contraseña. Posteriormente crear una base de datos llamada Practica7:

```
mysql>create database Practica7;
```

Otorgarle acceso completo al usuario recién creado a la base de datos:

```
mysql>grant select, insert, update, alter, create, delete, drop on Practica7. * to
usuarioBD identified by 'clave'
```

donde “usuarioBD” y “clave” son los nombres que se le dieron al usuario y su respectiva contraseña recién creado.

Salir de MySQL y entrar nuevamente con el nombre de usuario y contraseña creados y hacer uso de la base de datos Practica7 y crear un tabla llamada usuarios con dos campos alfanúmericos de longitud 20 llamados nombre y password respectivamente e insertar unos cuantos registros:

```
mysql>use Practica7;
```

```
mysql>create table usuarios(nombre varchar(20),password varchar(20));
```

```
mysql>insert into usuarios(nombre,password)values ('Javier','aaaaaa');
```

```
.....
.....
```

Puntos a desarrollar en la práctica:

1. Con el entorno de desarrollo Netbeans crear un nuevo proyecto llamado Practica7 y agregar una nueva Clase llamada Practica7a y para hacer uso de la librería instalada con el MySQL Connector para aprender como

conectarse con la base de datos Practica7 y su tabla usuarios copiar el siguiente código:

```
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Sentencia{
    public static void main( String args[] ){
        Connection conexion = null;
        String bd="Practica7";
        String login ="imelda";
        String password="clave";
        String url = "jdbc:mysql://localhost/"+bd;

        try {
            Class.forName("com.mysql.jdbc.Connection");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Sentencia.class.getName()).log(Level.SEVERE, null, ex);}

        try{
            conexion = (Connection) DriverManager.getConnection(url, login, password);
            System.out.println("Se ha establecido la conexión con: "+url);
            Statement sentencia=conexion.createStatement();
            ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios");
            System.out.println("Se ha ejecutado la sentencia SQL, que tiene las siguientes características");
            System.out.println("Timeout de consulta: "+sentencia.getQueryTimeout()+" sg");
            System.out.println("Número máximo de registros: "+sentencia.getMaxRows());
            System.out.println("Tamaño máximo de campo: "+sentencia.getMaxFieldSize()+" bytes");
            System.out.println("Número de registros devueltos cada vez: " +sentencia.getFetchSize());

            //se liberan los recursos utilizados por la sentencia
            sentencia.close();
            System.out.println("Se ha cerrado la sentencia");
            conexion.close();
            System.out.println("Se ha cerrado la conexión con: "+url);
        } catch(SQLException ex){
            System.out.println("Se han dado excepciones SQLException\n");
            System.out.println("=====\n");
            //Pueden existir varias SQLException encadenadas
            while(ex!=null){
                System.out.println("SQLState :"+ex.getSQLState()+"\n");
                System.out.println("Mensaje :"+ex.getMessage()+"\n");
                System.out.println("Código de error :"+ex.getErrorCode()+"\n");
                ex=ex.getNextException();
                System.out.println("\n");
            }
        }
    }
}
```

Se debe cambiar los valores de las variables String login ="imelda"; y String password="clave"; con los valores correspondientes, es decir en lugar de "imelda" en la variable *login*, se debe poner el nombre del usuario y que va a tener derechos sobre la base de datos creada "Practica7" que se creó con

anterioridad como prerequisite en MySQL, lo mismo para la variable *password*.

En este código se debe importar la librería *java.sql*, la cual contiene clases para el manipulación de bases de datos. También se deben importar las clases: *java.util.logging.Level* y *java.util.logging.Logger*; para poder manejar las excepciones cuando nuestro programa a través de un *try* intente localizar la clase “*com.mysql.jdbc.Connection*” para establecer la conexión con MySQL como lo muestra las siguientes líneas de código:

```
try {
    Class.forName("com.mysql.jdbc.Connection");
} catch (ClassNotFoundException ex) {
    Logger.getLogger(Sentencia.class.getName()).log(Level.SEVERE, null, ex);}
```

Para realizar la conexión con la base de datos y la tabla en particular se hace a través de las siguientes líneas de código:

```
conexion = (Connection) DriverManager.getConnection(url, login, password);
System.out.println("Se ha establecido la conexión con: "+url);
Statement sentencia=conexion.createStatement();
ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios");
```

2. Dentro del proyecto *Practica7* agregar un nuevo formulario “*JFrame Form*” llamado *Login*, en la vista de diseño arrastrar de la paleta de la sección de controles swing dos etiquetas, dos cajas de texto y un botón de la sección de hacia el formulario, en esta etapa se está haciendo uso de la librería swing proporcionada por el lenguaje java que es el utilizado en el curso.
3. En el árbol del proyecto, haciendo clic en la rama de librerías, se expande y se observa que el proyecto tiene conectada a él las librerías correspondientes al *Jdk*. Al proyecto se le debe agregar la necesaria para conectarse con el manejador de base de datos *MySQL* para lo cual se debe hacer clic con el botón derecho del mouse sobre la carpeta de librerías y se selecciona del menú desplegado la opción *agregar librería*, de la ventana de diálogo desplegada se selecciona la librería *MySQL JDBC Driver*, la cual debe aparecer puesto que con anterioridad y requisito a la práctica se debió haber instalado el conector.

4. En la vista de diseño del formulario Login, seleccionar el botón y en la ventana de propiedades cambiar a eventos y hacer clic en el evento **actionPerformed** para añadir el código a este evento que será el siguiente:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    // TODO add your handling code here:  
    // declaracion de variables para manejar base de datos  
    Connection conexion = null;  
    String bd="Practica7";  
    String login ="imelda";  
    String password="clave";  
    String url = "jdbc:mysql://localhost/"+bd;  
    String name= jTextField1.getText().trim();  
    String pwd= jTextField2.getText().trim();  
    try {  
        try {  
            Class.forName("com.mysql.jdbc.Connection");  
        } catch (ClassNotFoundException ex) {  
            Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null,  
ex);  
        }  
        conexion = (Connection) DriverManager.getConnection(url, login,  
password);  
        System.out.println("Se ha establecido la conexión con: "+url);  
        Statement sentencia=conexion.createStatement();  
        ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios  
where nombre='"+name+"' AND password='"+pwd+"'");  
        System.out.println("Se ejecuto el query");  
        if (resultado.next()) {  
  
            //System.out.println(resultado.getString(resultado.findColumn("nombre")));  
  
            JOptionPane.showMessageDialog(this, "Bienvenida");  
  
        } else {  
            JOptionPane.showMessageDialog(this, "Alerta de Intruso, suelten a  
los perros");  
  
        }  
  
    }catch(SQLException ex){  
        System.out.println("Se han dado excepciones SQLException\n");  
        System.out.println("=====\n");  
        //Pueden existir varias SQLException encadenadas  
        while(ex!=null){  
            System.out.println("SQLState :"+ex.getSQLState()+"\n");  
            System.out.println("Mensaje :"+ex.getMessage()+"\n");  
            System.out.println("Código de error :"+ex.getErrorCode()+"\n");  
            ex=ex.getNextException();  
        }  
    }  
}
```

```

        System.out.println("\n");
    }
}

```

Recuerde que debe cambiar los valores de las variables String login="imelda"; y String password="clave"; con los valores correspondientes, es decir en lugar de "imelda" en la variable *login*, se debe poner el nombre del usuario y que va a tener derechos sobre la base de datos creada "Practica7" que se creó con anterioridad como prerequisite en MySQL, lo mismo para la variable *password*.

Y también importar lo siguiente:

```

import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

```

Para que el programa no marque errores

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. Con el entorno de desarrollo Netbeans crear un nuevo proyecto llamado Practica7 y agregar una nueva Clase llamada Practica7a y para hacer uso de la librería instalada con el MySQL Connector para aprender como conectarse con la base de datos Practica7 y su tabla usuarios copiar el siguiente código:

```
import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Sentencia{
    public static void main( String args[] ){
        Connection conexion = null;
        String bd="Practica7";
        String login ="imelda";
        String password="clave";
        String url = "jdbc:mysql://localhost/"+bd;

        try {
            Class.forName("com.mysql.jdbc.Connection");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Sentencia.class.getName()).log(Level.SEVERE, null, ex);}

        try{
            conexion = (Connection) DriverManager.getConnection(url, login, password);
            System.out.println("Se ha establecido la conexión con: "+url);
            Statement sentencia=conexion.createStatement();
            ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios");
            System.out.println("Se ha ejecutado la sentencia SQL, que tiene las siguientes
características");
            System.out.println("Timeout de consulta: "+sentencia.getQueryTimeout()+" sg");
            System.out.println("Número máximo de registros: "+sentencia.getMaxRows());
            System.out.println("Tamaño máximo de campo: "+sentencia.getMaxFieldSize()+"
bytes");
            System.out.println("Número de registros devueltos cada vez: "
+sentencia.getFetchSize());

            //se liberan los recursos utilizados por la sentencia
            sentencia.close();
            System.out.println("Se ha cerrado la sentencia");
            conexion.close();
            System.out.println("Se ha cerrado la conexión con: "+url);
        } catch (SQLException ex){
            System.out.println("Se han dado excepciones SQLException\n");
            System.out.println("=====\n");
            //Pueden existir varias SQLException encadenadas
            while(ex!=null){
                System.out.println("SQLState :"+ex.getSQLState()+"\n");
                System.out.println("Mensaje :"+ex.getMessage()+"\n");
                System.out.println("Código de error :"+ex.getErrorCode()+"\n");
                ex=ex.getNextException();
            }
        }
    }
}
```

```

        System.out.println("\n");
    }
}
}
}

```

En este código se debe importar la librería `java.sql`, la cual contiene clases para la manipulación de bases de datos. También se deben importar las clases: `java.util.logging.Level` y `java.util.logging.Logger`; para poder manejar las excepciones cuando nuestro programa a través de un `try` intente localizar la clase “`com.mysql.jdbc.Connection`” para establecer la conexión con MySQL como lo muestra las siguientes líneas de código:

```

try {
    Class.forName("com.mysql.jdbc.Connection");
} catch (ClassNotFoundException ex) {
    Logger.getLogger(Sentencia.class.getName()).log(Level.SEVERE, null, ex);}

```

Para realizar la conexión con la base de datos y la tabla en particular se hace a través de las siguientes líneas de código:

```

conexion = (Connection) DriverManager.getConnection(url, login, password);
System.out.println("Se ha establecido la conexión con: "+url);
Statement sentencia=conexion.createStatement();
ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios");

```

Al ejecutar el programa se muestra el siguiente resultado en la ventana de salida:

Output - Practica7 (run)

2. Dentro del proyecto Practica7 agregar un nuevo formulario “JFrame Form” llamado Login, en la vista de diseño arrastrar de la paleta de la sección de controles swing dos etiquetas, dos cajas de texto y un botón de la sección de hacia el formulario.

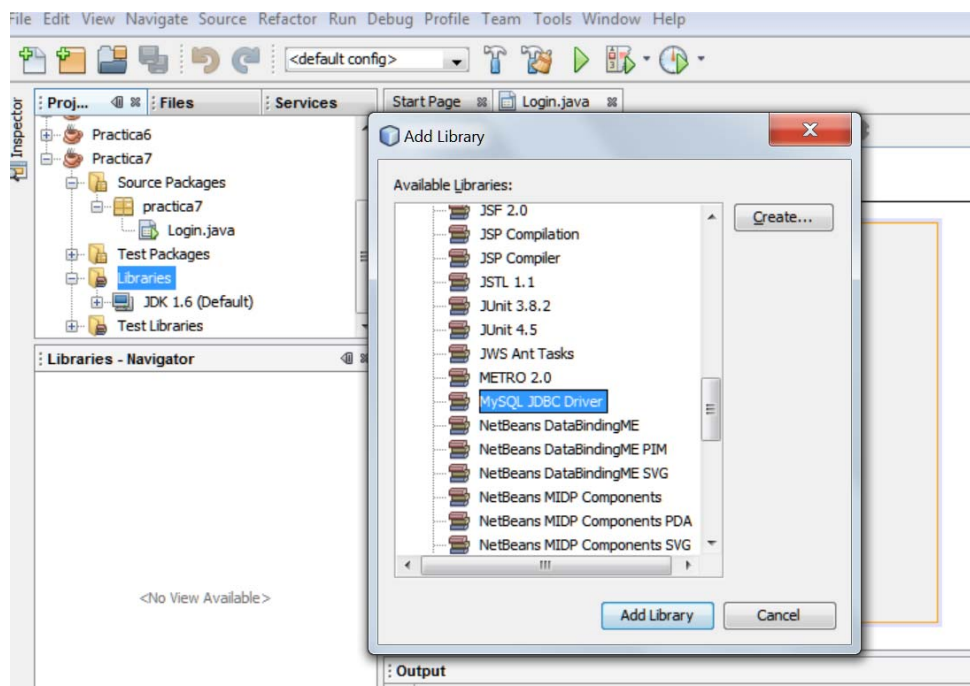


74

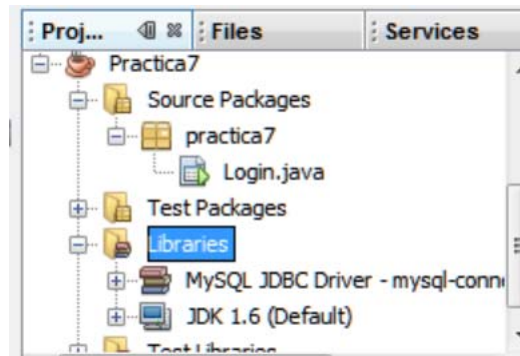
desarrollo; se aprovecha en esta práctica para ejercitar el uso de librerías proporcionadas por el lenguaje como lo es la librería swing que en entorno de diseño de Netbeans se puede apreciar de una manera gráfica. Y con la ventaja que nos deja un esqueleto de código fuente al cual se le puede agregar las sentencias necesarias para darle la funcionalidad requerida a la aplicación y en este caso al formulario.

3. En el árbol del proyecto, haciendo clic en la rama de librerías, se expande y se observa que el proyecto tiene conectada a él las librerías correspondientes al Jdk. Al proyecto se le debe agregar la necesaria para conectarse con el manejador de base de datos MySql para lo cual se debe hacer clic con el botón derecho del mouse sobre la carpeta de librerías y se selecciona del menú desplegado la opción agregar librería, de la ventana de diálogo desplegada se selecciona la librería MySQL JDBC Driver.

Al realizar el paso anterior se obtiene el siguiente resultado



Y en el árbol del proyecto en la carpeta de librerías aparece añadida la librería como puede observarse en la siguiente figura:



5. En la vista de diseño del formulario Login, seleccionar el botón y en la ventana de propiedades cambiar a eventos y hacer clic en el evento **actionPerformed** para añadir el código a este evento que será el siguiente:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    // declaracion de variables para manejar base de datos
    Connection conexion = null;
    String bd="Practica7";
    String login ="imelda";
    String password="clave";
    String url = "jdbc:mysql://localhost/"+bd;
    String name= jTextField1.getText().trim();
    String pwd= jTextField2.getText().trim();
    try {
        try {
            Class.forName("com.mysql.jdbc.Connection");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null,
ex);
        }
        conexion = (Connection) DriverManager.getConnection(url, login,
password);
        System.out.println("Se ha establecido la conexión con: "+url);
        Statement sentencia=conexion.createStatement();
        ResultSet resultado=sentencia.executeQuery("SELECT * FROM usuarios
where nombre='"+name+"' AND password='"+pwd+"'");
        System.out.println("Se ejecuto el query");
        if (resultado.next()) {

//System.out.println(resultado.getString(resultado.findColumn("nombre")));

            JOptionPane.showMessageDialog(this, "Bienvenida");

        } else {
            JOptionPane.showMessageDialog(this, "Alerta de Intruso, suelten a
los perros");
        }
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

```

    }
} catch(SQLException ex){
    System.out.println("Se han dado excepciones SQLException\n");
    System.out.println("=====\n");
    //Pueden existir varias SQLException encadenadas
    while(ex!=null){
        System.out.println("SQLState :"+ex.getSQLState()+"\n");
        System.out.println("Mensaje :"+ex.getMessage()+"\n");
        System.out.println("Código de error :"+ex.getErrorCode()+"\n");
        ex=ex.getNextException();
        System.out.println("\n");
    }
}
}
}

```

Recuerde que debe cambiar los valores de las variables String login="imelda"; y String password="clave"; con los valores correspondientes, es decir en lugar de "imelda" en la variable *login*, se debe poner el nombre del usuario y que va a tener derechos sobre la base de datos creada "Practica7" que se creó con anterioridad como prerequisite en MySQL, lo mismo para la variable *password*.

Y también importar lo siguiente:

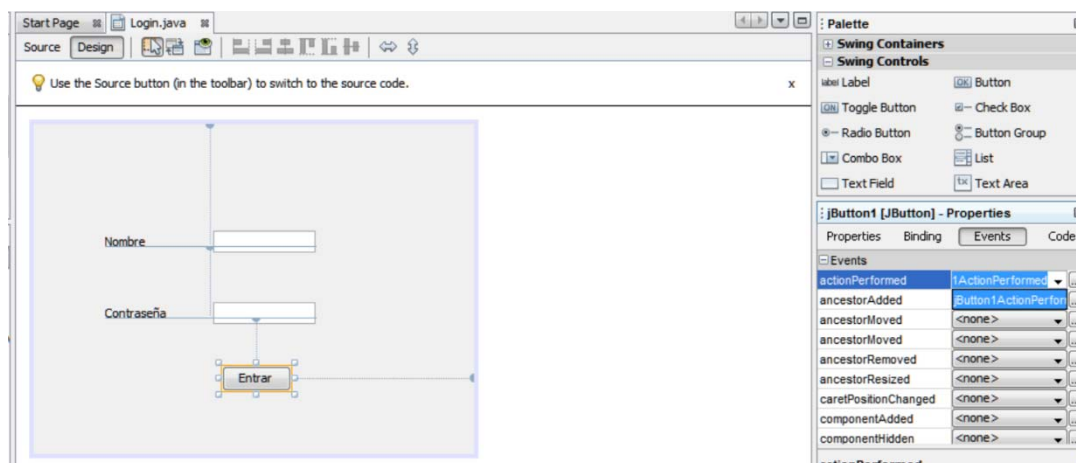
```

import java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

```

Para que el programa no marque errores

A continuación se muestra la agregación del evento actionPerformed

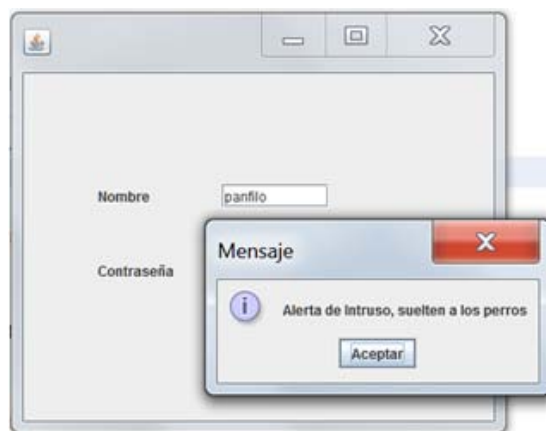


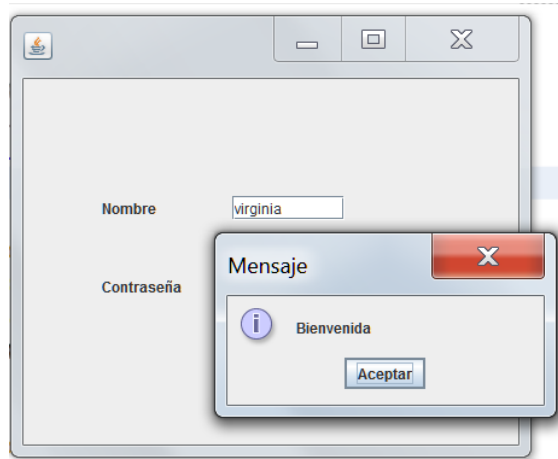
```

34  * always regenerated by the Form Editor.
35  */
36  @SuppressWarnings("unchecked")
37  Generated Code
98
99  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
100
101
102      // TODO add your handling code here:
103      // declaracion de variables para manejar base de datos
104      Connection conexion = null;
105      String bd="Practica7";
106      String login ="imelda";
107      String password="clave";
108      String url = "jdbc:mysql://localhost/"+bd;
109      String name= jTextField1.getText().trim();
110      String pwd= jTextField2.getText().trim();
111      try {
112          try {
113              Class.forName("com.mysql.jdbc.Connection");
114          } catch (ClassNotFoundException ex) {
115              Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);

```

Una vez realizados los cambios al programa se ejecuta y el resultado se muestra a continuación:





10. Bibliografía preliminar.

Bernard Van Haecke Jdbc 3.0: Java Database Connectivity, John Wiley & Sons Inc , 2002

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México,2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill,México, 2008

1. PRÁCTICA No. 8

2. Nombre: Creación de componentes definidas por el usuario

3. Objetivo: Conocer los principios básicos que proporciona el lenguaje utilizado para la creación de componentes definidas por el usuario

4. Introducción. En esta práctica el alumno desarrollará un programa que utilice componentes definidos por el usuario, que el mismo hará a partir de componentes bases.

5. Temas y subtemas correlacionados con la práctica.

Tema III Componentes y librerías.

3.4 Creación de componentes (visuales y no visuales) definidos por el usuario

El alumno a partir de los ejemplos vistos en clase creará componentes derivándolos de componentes bases tanto visuales como no visuales.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el subtema del tema dado en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. Crear un Proyecto llamado Practica8 y agregar una nueva clase llamada Practica8 la cual se debe derivar de la clase **JFrame** e implemente la interfaz **ActionListener** a la cual se le agregarán un toolbar, un panel, dos etiquetas y dos cajas de texto y un botón con la propiedad “Text” como “Agregar”.

También agregar una nueva clase llamada **ImprimirLista** la cual se debe derivar de la clase **JButton** e implementar la interfaz **ActionListener** que debe incluir un **gif** que represente una lista de elementos y agregar otra nueva clase llamada **Persona** que derive de la clase **Component**.

Y por último anexar una nueva clase llamada PruebaPráctica8 con el método “main()”, en el cual se cree un instancia de la clase Practica8

La clase Practica8 creará también una instancia de la clase **ImprimirLista** la cual al ser un componente visual será añadida al JFrame en un JToolBar. Al hacer clic en el botón “Agregar” se creará una nueva instancia de la clase **Persona** y se agregará a un **ArrayList** y al hacer clic en la componente visual definida por el usuario **ImprimirLista** se imprimirá en una nueva ventana la lista de las instancias de la componente no visual definida por el usuario **Persona**.

Ejecutar el programa PruebaPractica8.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. Crear un Proyecto llamado Practica8 y agregar una nueva clase llamada Practica8 la cual se debe derivar de la clase **JFrame** e implemente la interfaz **ActionListener** a la cual se le agregarán un toolbar, un panel, dos etiquetas y dos cajas de texto y un botón con la propiedad "Text" como "Agregar".

También agregar una nueva clase llamada **ImprimirLista** la cual se debe derivar de la clase **JButton** e implementar la interfaz **ActionListener** que debe incluir un *gif* que represente una lista de elementos y agregar otra nueva clase llamada **Persona** que derive de la clase **Component**.

Y por último anexar una nueva clase llamada PruebaPráctica8 con el método "main()", en el cual se cree un instancia de la clase Practica8

La clase Practica8 creará también una instancia de la clase **ImprimirLista** la cual al ser un componente visual será añadida al JFrame en un JToolBar. Al hacer clic en el botón "Agregar" se creará una nueva instancia de la clase **Persona** y se agregará a un **ArrayList** y al hacer clic en la componente visual definida por el usuario **ImprimirLista** se imprimirá en una nueva ventana la lista de las instancias de la componente no visual definida por el usuario **Persona**.

Ejecutar el programa PruebaPractica8.

El código de la componente no visual Persona se muestra a continuación a la cual se le dotan de métodos para obtener y poner los valores de sus propiedades nombre y edad:

```
public class Persona extends Component{

    private String nombre,edad;

    public Persona()
    {
        nombre="";
        edad="";
    }

    public void setNombre(String n)
    {
        nombre=n;
    }

    public void setEdad(String e)
    {
        edad=e;
    }
}
```

```

    }

    public String getNombre(){
        return nombre;
    }

    public String getEdad()
    {
        return edad;
    }
}

```

El código de la componente visual ImprimirLista se muestra a continuación, es importante recordar que JButton se deriva de JComponent que es una componente Visual:

```

public class ImprimirLista extends JButton implements ActionListener {
    private Icon imprimir;
    public ArrayList lista;
    private JFrame ventana;
    private JTextArea area;

    public ImprimirLista(ArrayList l){
        imprimir = new ImageIcon("lista.gif");
        this.setIcon(imprimir);
        area= new JTextArea();
        area.setEditable(false);
        lista=l;
        addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        ventana= new JFrame("Lista de personas");
        ventana.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        ventana.setVisible(true);
        ventana.setSize(400,200);
        ventana.getContentPane().add(area);
        String texto= "\t"+" Persona "+" Edad"+ "\n";

        Persona persona;
        for (int i=0; i<lista.size();i++){
            persona = (Persona) lista.get(i);
            texto+= "\n" + "\t" + persona.getNombre()+"\t" +
persona.getEdad();
        }
        area.setText(texto);
    }
}

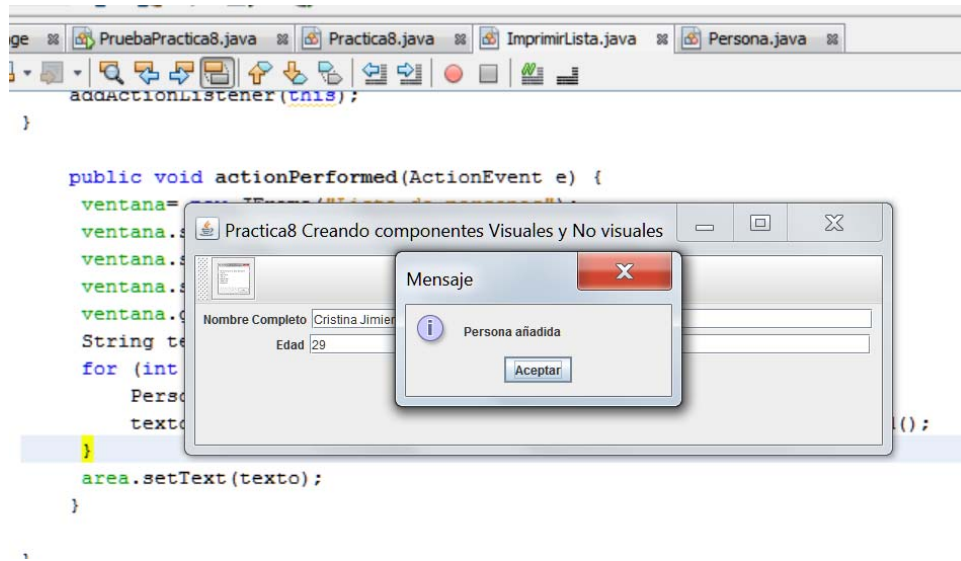
```

```
}
```

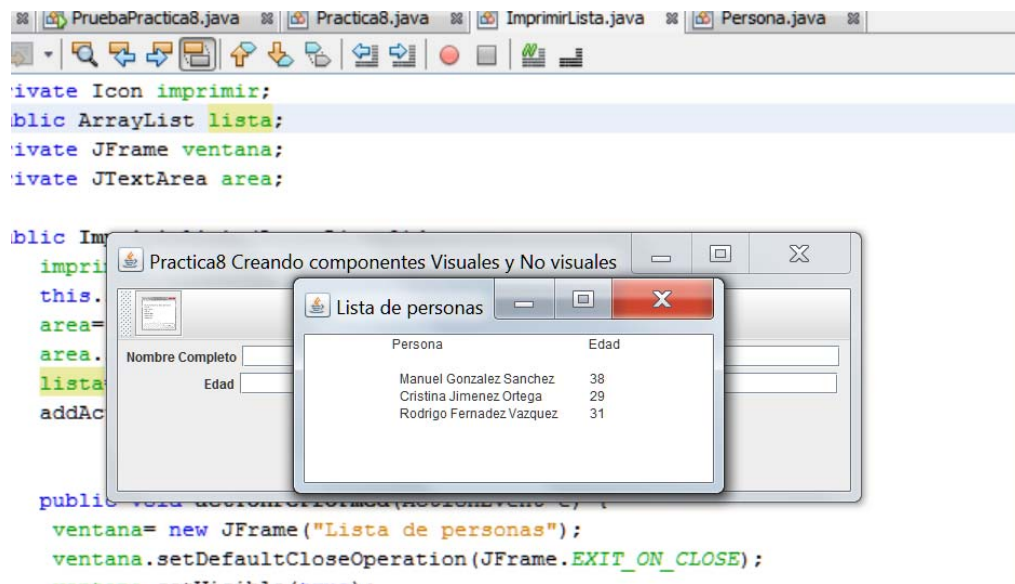
Tanto la componente ImprimirLista como la de Persona podrían ser añadidas a una librería para manipular datos de la persona, extendiendo por supuesto las propiedades o datos que se desean manipular de una persona, y podrían ser utilizadas en diferentes aplicaciones

Y por último al ejecutar el programa se obtienen los siguientes resultados:

Agregando personas a la lista



Oprimiendo el botón de lista de la barra de herramientas:



10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 9

2. Nombre: Creación y uso de paquetes/librerías definidas por el usuario

3. Objetivo: Conocer los principios básicos que proporciona el lenguaje utilizado para la creación de paquetes/librerías definidas por el usuario

4. Introducción. En esta práctica el alumno creará una librería de las componentes creadas en la práctica anterior y hará un programa que haga uso de ellas.

5. Temas y subtemas correlacionados con la práctica.

Tema III Componentes y librerías

3.5 Creación y uso de paquetes/librerías definidas por el usuario

El alumno podrá crear un paquete/librería de clases sin necesidad de utilizar un entorno de desarrollo en su lugar usará comandos del propio JDK y utilizarla en un programa que será creado en un entorno de desarrollo para aplicar ambos enfoques.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el subtema del tema dado en la teoría de la asignatura.
- b. Haber realizado en su totalidad la practica número ocho.

Puntos a desarrollar en la práctica:

1. Crear una carpeta llamada Pratica9, en el directorio C:\ y dentro de esta la carpeta persona, copiar en esta última los archivos ImprimirLista.class y Persona.class de la practica 8 que se encuentran en la subcarpeta build/clases/Practica8 del proyecto.

Desde la línea de comandos del sistema operativo y en la carpeta Practica9 crear la librería ejecutando el comando:

```
c:\Practica9> jar cvf MiLibreria.jar persona
```

2. En el entorno de desarrollo crear un proyecto llamado Practica9 y agregar una nueva clase llamada Practica9 que se derive de JFrame e implemente la interfaz **ActionListener** a la cual se le agregarán un toolbar, un panel, dos etiquetas y dos cajas de texto y un botón con la propiedad "Text" como "Agregar". También añadir una nueva clase llamada PruebaPractica9 con método main() dentro del cual se creará un instancia de la clase Practica9. En el menú horizontal del Netbeans elegir la opción herramientas y de este seleccionar librerías.
3. En la ventana de diálogo desplegado hacer clic en el botón de nueva librería y en la ventana desplegada poner el nombre de la librería como MyLibrary. Una vez hecho lo anterior hacer clic en el botón añadir jar/Carpeta y en la ventana de diálogo resultante seleccionar el jar en el directorio c:>Practica9 y hacer clic en aceptar en ambas ventanas de diálogo.
4. En el árbol del proyecto, haciendo clic en la rama de librerías, se expande y se observa que el proyecto tiene conectada a él las librerías correspondientes al Jdk. Al proyecto se le debe agregar la librería que se acaba de crear para lo cual se debe hacer clic con el botón derecho del mouse sobre la carpeta de librerías y se selecciona del menú desplegado la opción agregar librería, de la ventana de diálogo desplegada se selecciona la la librería MyLibrary.
5. En la clase Practica9 importar la librería creada:

```
import persona.*;
```

Y agregar el código correspondiente, es decir, la clase Practica9 creará también una instancia de la clase **ImprimirLista** la cual al ser un componente visual será añadida al JFrame en un JToolBar. Al hacer clic en el botón "Agregar" se creará una nueva instancia de la clase **Persona** y se agregará a un **ArrayList** y al hacer clic en la componente visual definida por el usuario **ImprimirLista** se imprimirá en una nueva ventana la lista de las instancias de la componente no visual definida por el usuario **Persona** similar a la clase Practica8 creada en la práctica anterior.

Ejecutar el programa PruebaPractica9.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

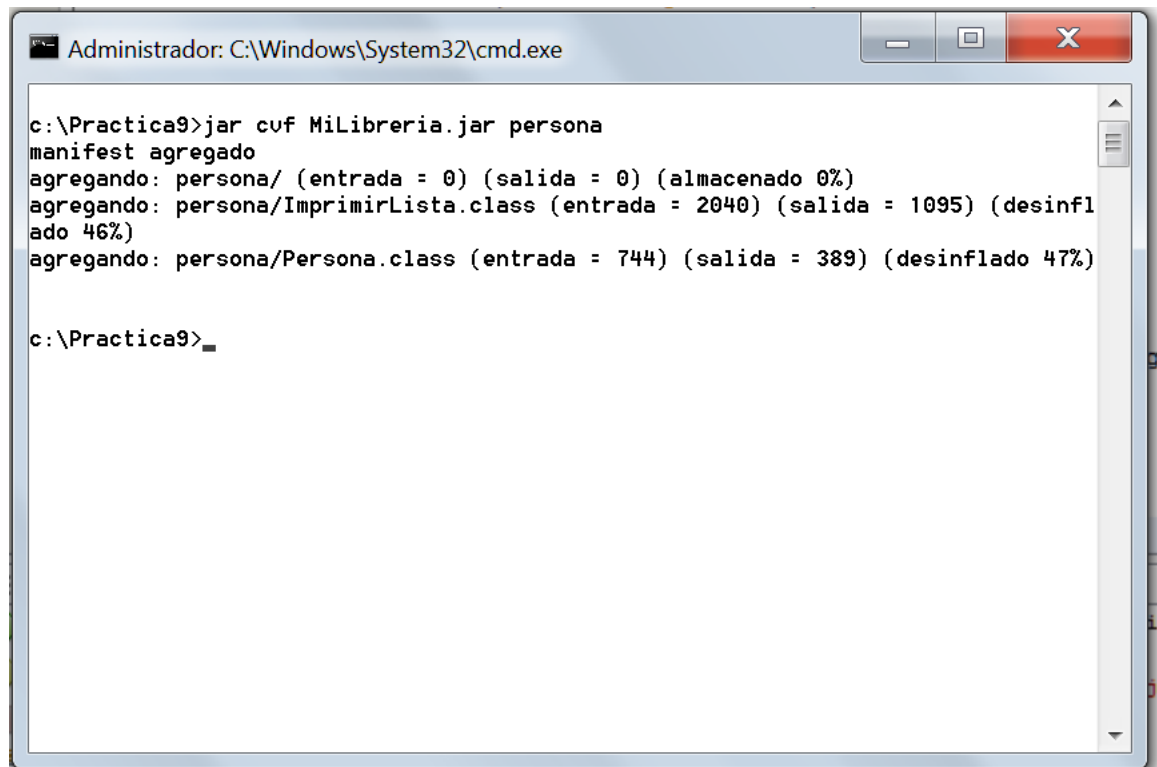
Resultados de la Práctica de acuerdo a lo indicado.

1. Crear una carpeta llamada Pratica9, en el directorio C:\ y dentro de esta la carpeta persona, copiar en esta última los archivos ImprimirLista.class y Persona.class de la practica 8 que se encuentran en la subcarpeta build/clases/Practica8 del proyecto.

Desde la línea de comandos del sistema operativo y en la carpeta Practica9 crear la librería ejecutando el comando:

```
c:\Practica9> jar cvf MiLibreria.jar persona
```

El resultado de realizar este paso se muestra a continuación:



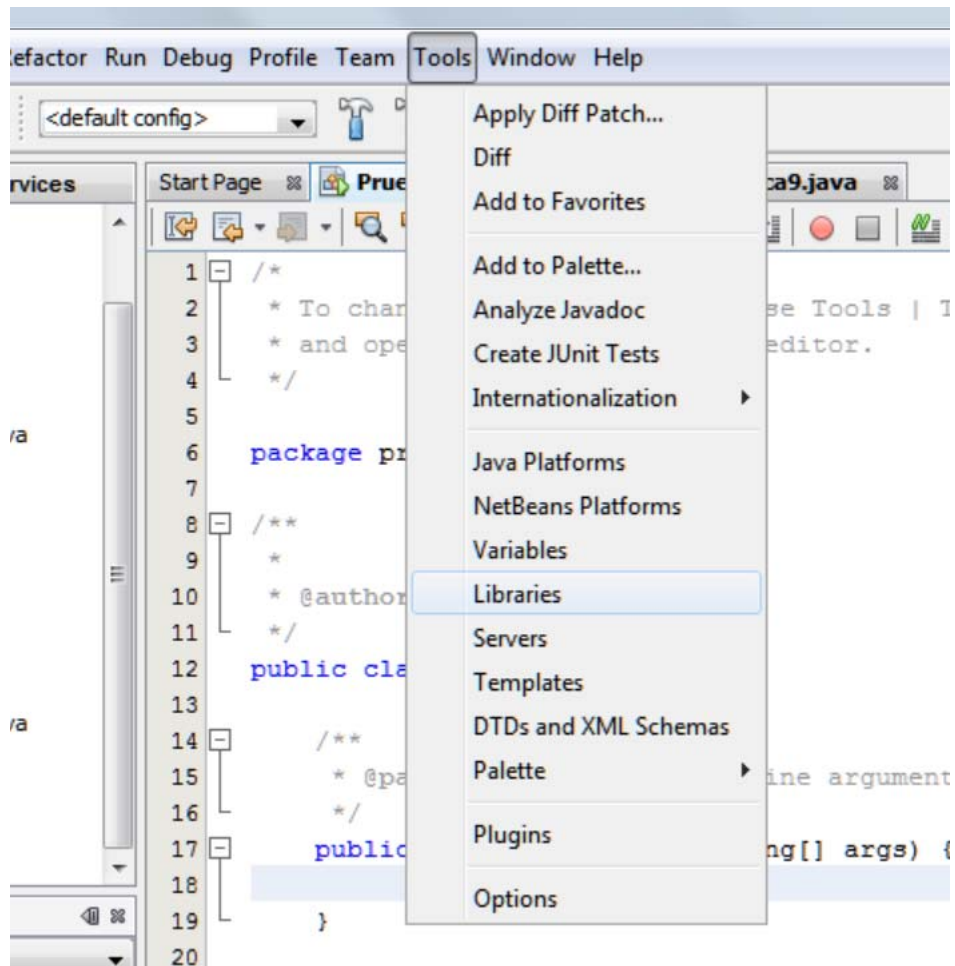
```
Administrador: C:\Windows\System32\cmd.exe

c:\Practica9>jar cvf MiLibreria.jar persona
manifest agregado
agregando: persona/ (entrada = 0) (salida = 0) (almacenado 0%)
agregando: persona/ImprimirLista.class (entrada = 2040) (salida = 1095) (desinflado 46%)
agregando: persona/Persona.class (entrada = 744) (salida = 389) (desinflado 47%)

c:\Practica9>
```

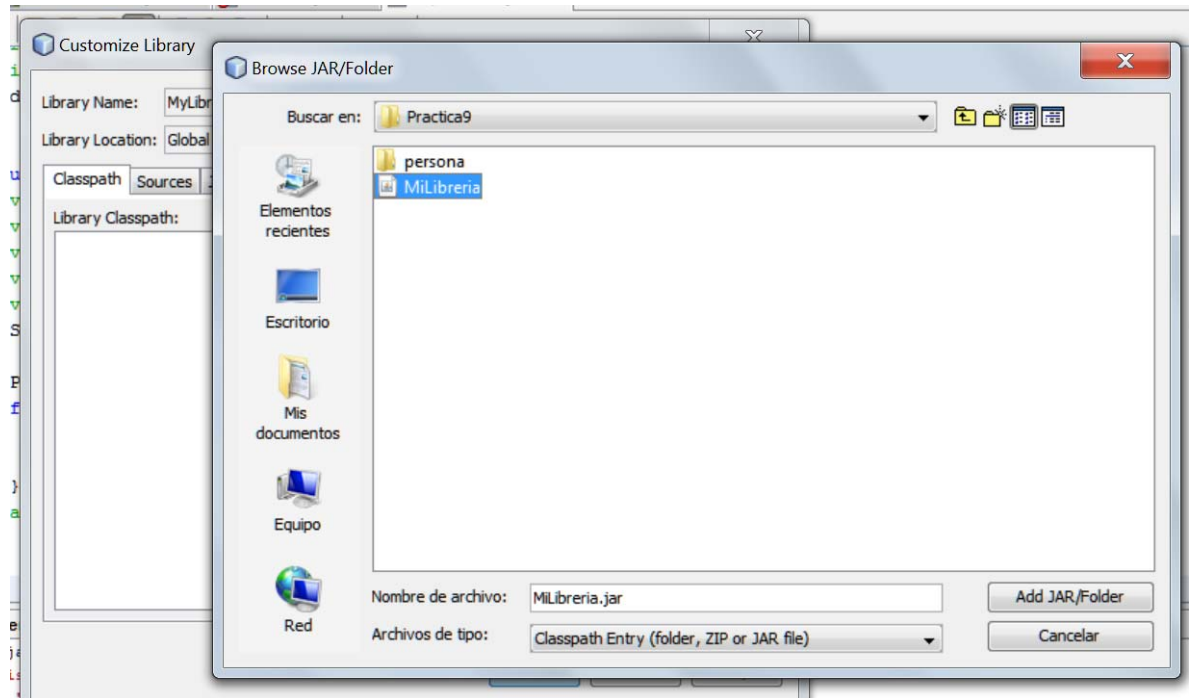
2. En el entorno de desarrollo crear un proyecto llamado Practica9 y agregar una nueva clase llamada Practica9 que se derive de JFrame e implemente la interfaz **ActionListener** a la cual se le agregarán un toolbar, un panel, dos etiquetas y dos cajas de texto y un botón con la propiedad "Text" como "Agregar". También añadir una nueva clase llamada PruebaPractica9 con método main() dentro del cual se creará un instancia de la clase Practica9. En el menú horizontal del Netbeans elegir la opción herramientas y de este seleccionar librerías.

Lo cual se puede observarse enseguida:



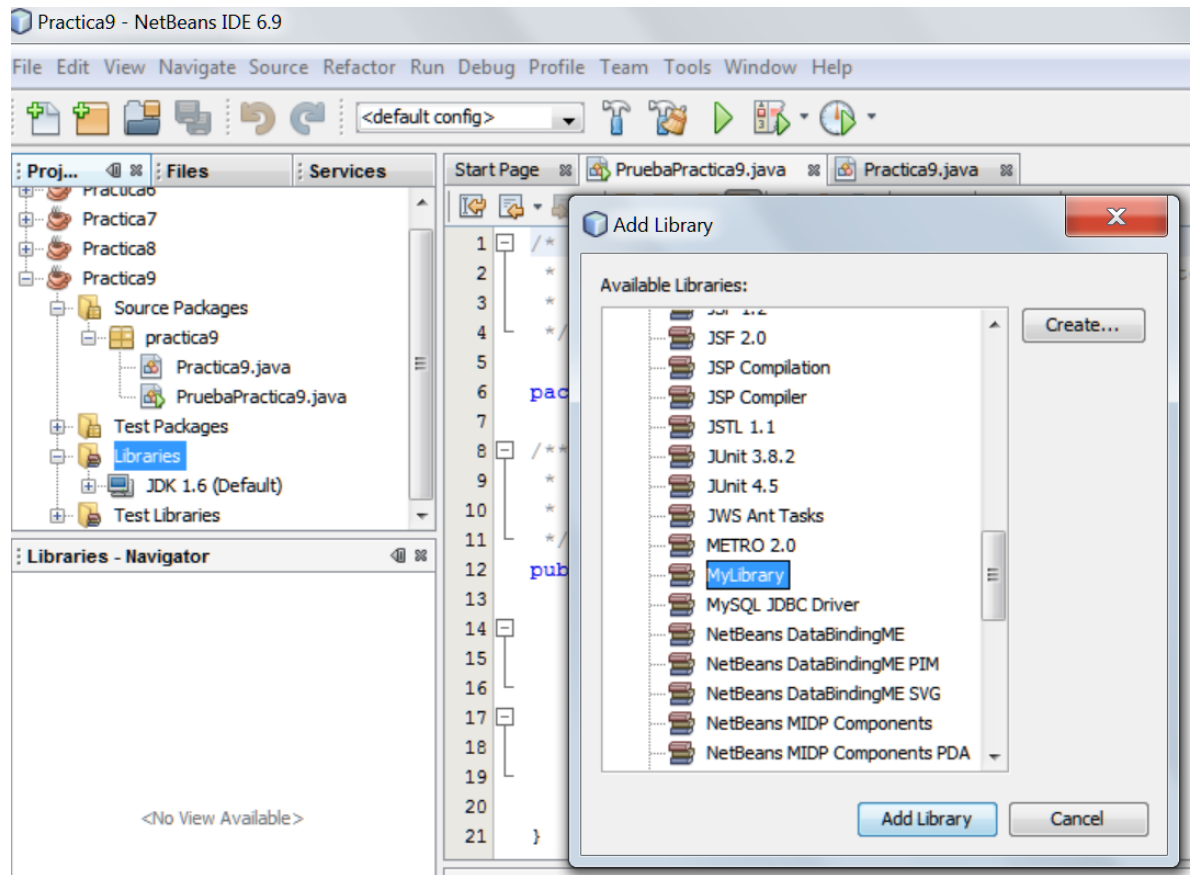
3. En la ventana de diálogo desplegado hacer clic en el botón de nueva librería y en la ventana desplegada poner el nombre de la librería como MyLibrary. Una vez hecho lo anterior hacer clic en el botón añadir jar/Carpeta y en la ventana de diálogo resultante seleccionar el jar en el directorio c:>Practica9 y hacer clic en aceptar en ambas ventanas de diálogo.

El resultado de realizar este paso se muestra a continuación:



4. En el árbol del proyecto, haciendo clic en la rama de librerías, se expande y se observa que el proyecto tiene conectada a él las librerías correspondientes al Jdk. Al proyecto se le debe agregar la librería que se acaba de crear para lo cual se debe hacer clic con el botón derecho del mouse sobre la carpeta de librerías y se selecciona del menú desplegado la opción agregar librería, de la ventana de diálogo desplegada se selecciona la la librería MyLibrary.

Lo cual se puede observarse enseguida:



5. En la clase Practica9 importar la librería creada:

```
import persona.*;
```

Y agregar el código correspondiente, es decir, la clase Practica9 creará también una instancia de la clase **ImprimirLista** la cual al ser un componente visual será añadida al JFrame en un JToolBar. Al hacer clic en el botón “Agregar” se creará una nueva instancia de la clase **Persona** y se agregará a un **ArrayList** y al hacer clic en la componente visual definida por el usuario **ImprimirLista** se imprimirá en una nueva ventana la lista de las instancias de la componente no visual definida por el usuario **Persona** similar a la clase Practica8 creada en la práctica anterior. Ejecutar el programa PruebaPractica9.

El código de la clase Practica9 se muestra enseguida:

```
import java.awt.event.*;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
```

```

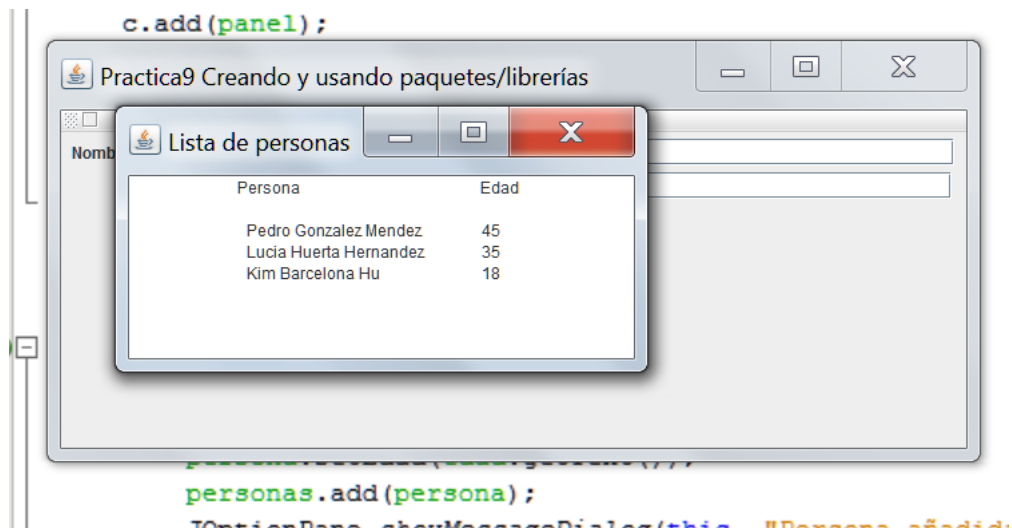
import persona.*;

public class Practica9 extends JFrame implements ActionListener{
private JLabel l1,l2;
private JTextField nombre, edad;
private JPanel panel;
private JToolBar toolbar;
private JButton agregar;
private ImprimirLista lista;
private ArrayList personas;
private Persona persona;
public Practica9(){
    super("Practica9 Creando y usando paquetes/librerías");
    Container c = this.getContentPane();
    personas= new ArrayList();
    l1= new JLabel("Nombre Completo");
    l2= new JLabel("Edad",JLabel.RIGHT);
    nombre = new JTextField(50);
    edad = new JTextField(50);
    panel = new JPanel();
    lista = new ImprimirLista(personas);
    agregar = new JButton("Agregar");
    agregar.addActionListener(this);
    toolbar = new JToolBar("Toolbar", JToolBar.HORIZONTAL);
    toolbar.add(lista);
    panel.add(l1);
    panel.add(nombre);
    panel.add(l2);
    panel.add(edad);
    panel.add(agregar);
    c.add(toolbar,BorderLayout.NORTH);
    c.add(panel);
    setSize(700,250);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    persona= new Persona();
    persona.setNombre(nombre.getText());
    persona.setEdad(edad.getText());
    personas.add(persona);
    JOptionPane.showMessageDialog(this, "Persona añadida");
    nombre.setText("");
    edad.setText("");
}
}

```

Y el resultado de ejecutar el programa PruebaPractica9 después de añadir a unas personas y hacer clic en el botón de la barra de herramientas puede observarse a continuación:



10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 10

2. Nombre: Comparación de un programa de flujo único contra uno de flujo múltiple.

3. Objetivo: • Comprender las diferencias entre la programación de flujo único y la programación concurrente

4. Introducción. En esta práctica el alumno ejecutará dos programas que simulan una competencia uno sin uso de hilos y otro aplicando hilos, y observará las diferencias en el resultado de la ejecución de ambos.

5. Temas y subtemas correlacionados con la práctica.

Tema IV Programación concurrente (MultiHilos)

4.1. Concepto de hilo.

4.2. Comparación de un programa de flujo único contra uno de flujo múltiple

El alumno ejecutará los programas propuestas y observará los resultados de la ejecución.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% los dos primeros subtemas del tema dado en la teoría de la asignatura.

Puntos a desarrollar en la práctica:

1. En el entorno de desarrollo crear un proyecto llamado Practica10 y agregar una nueva clase llamada Competencia que tenga el método Main y dentro de éste cree una instancia de una clase llamada Argentina y otra llamada Brasil. El código mostrará error ya que falta crear ambas clases.
2. Añada al proyecto una clase llamada Argentina y copie en ella el siguiente código:

```
public class Argentina {  
  
    private String id;  
    public Argentina(String str)  
    {  
        id=str;  
    }  
  
    public void Inicia()  
    {  
        Corre();  
    }  
    private void Corre()  
    {  
        for (int i= 0; i < 10;i++){  
            System.out.print(id);  
        }  
    }  
}
```

3. Añada al proyecto una clase llamada Brasil y copie en ella el siguiente código:

```
public class Brasil {  
  
    private String id;
```

```

    public Brasil(String str)
    {
        id=str;
    }

    public void Inicia()
    {
        Corre();
    }
    private void Corre()
    {
        for (int i= 0; i < 10;i++){
            System.out.print(id);
        }
    }
}

```

4. Ejecute la clase principal Competencia y observe los resultados, se supone que se simula una competencia entre el participante de Argentina versus el participante de Brasil.
5. Ya que en realidad no existe casi diferencia entre el código de la clase Argentina y el código de la clase Brasil, más que el nombre del constructor y de la clase añada una nueva clase al proyecto llamada Nadador y copie en ella el siguiente código:

```

public class Nadador {

    private String id;

    public Nadador(String n)
    {
        id=n;
    }
    public void Inicia()
    {

```

```

        Corre();
    }
    private void Corre()
    {
        for (int i= 0; i < 10;i++){
            System.out.print(id);
        }
    }
}

```

6. Ahora modifique la clase Competencia para que cree dos instancias de la clase Nadador con los mismos nombres de instancia que tenía y enviando como parámetro para la primera instancia la cadena “pibe” y para la segunda instancia “brasileiro”. Ejecute la clase y observe los resultados.
7. Modifique la clase Nadador para que ahora herede de la clase Thread , lo que significa que ahora es un hilo su nuevo código debe ser el siguiente:

```

public class Nadador extends Thread {

    private String id;

    public Nadador(String n)
    {
        id=n;
    }
    public void run()
    {
        int i=0;
        System.out.println("Comienza .. "+ id);
        while(i<5)
        {
            try

```

```

        {
            Thread.sleep(5000);
            System.out.println(id);
        }
        catch(InterruptedException ex)
        {
        }
        i++;
    }
    System.out.println("Termina .. " + id);
}
}

```

8. Modifique la nuevamente la clase principal Competencia y dado que ahora nadador es un hilo y ya no tiene el método Inicia(), cambie la llamada al método Start() en lugar de Inicia(), para ambas instancias y ejecute la clase y compare los resultados con la anterior ejecución.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. En el entorno de desarrollo crear un proyecto llamado Practica10 y agregar una nueva clase llamada Competencia que tenga el método Main y dentro de éste cree una instancia de una clase llamada Argentina y otra llamada Brasil. El código mostrará error ya que falta crear ambas clases.
2. Añada al proyecto una clase llamada Argentina y copie en ella el siguiente código:

```
public class Argentina {  
  
    private String id;  
    public Argentina(String str)  
    {  
        id=str;  
    }  
  
    public void Inicia()  
    {  
        Corre();  
    }  
    private void Corre()  
    {  
        for (int i= 0; i < 10;i++){  
            System.out.print(id);  
        }  
    }  
}
```

3. Añada al proyecto una clase llamada Brasil y copie en ella el siguiente código:

```
public class Brasil {
```

```

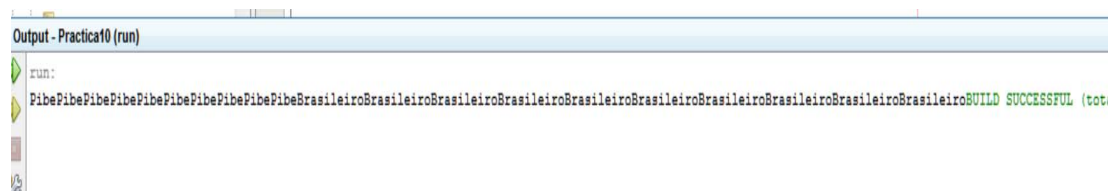
    private String id;
    public Brasil(String str)
    {
        id=str;
    }

    public void Inicia()
    {
        Corre();
    }
    private void Corre()
    {
        for (int i= 0; i < 10;i++){
            System.out.print(id);
        }
    }
}

```

4. Ejecute la clase principal Competencia y observe los resultados, se supone que se simula una competencia entre el participante de Argentina versus el participante de Brasil.

El resultado de realizar los cuatro pasos se muestra a continuación:



Ya que la ejecución es secuencial, es decir primero se ejecuta el método de la instancia “pibe” y luego el método de la instancia “brasileiro”, el lógica la salida, es decir, primero se despliega el ciclo “for” de la clase Argentina y después el ciclo “for” de la clase Brasil.

5. Ya que en realidad no existe casi diferencia entre el código de la clase Argentina y el código de la clase Brasil, más que el nombre del

constructor y de la clase añada una nueva clase al proyecto llamada Nadador y copie en ella el siguiente código:

```
public class Nadador {  
  
    private String id;  
  
    public Nadador(String n)  
    {  
        id=n;  
    }  
    public void Inicia()  
    {  
        Corre();  
    }  
    private void Corre()  
    {  
        for (int i= 0; i < 10;i++){  
            System.out.print(id);  
        }  
    }  
}
```

6. Ahora modifique la clase Competencia para que cree dos instancias de la clase Nadador con los mismos nombres de instancia que tenía y enviando como parámetro para la primera instancia la cadena “pibe” y para la segunda instancia “brasileiro”. Ejecute la clase y observe los resultados.

El resultado de realizar los dos pasos anteriores se muestra a continuación:

```
Page Competencia.java Argentina.java Brasil.java Nadador.java
C:\Users\ime\Docum

/**
 *
 */
public class Nadador extends Thread {

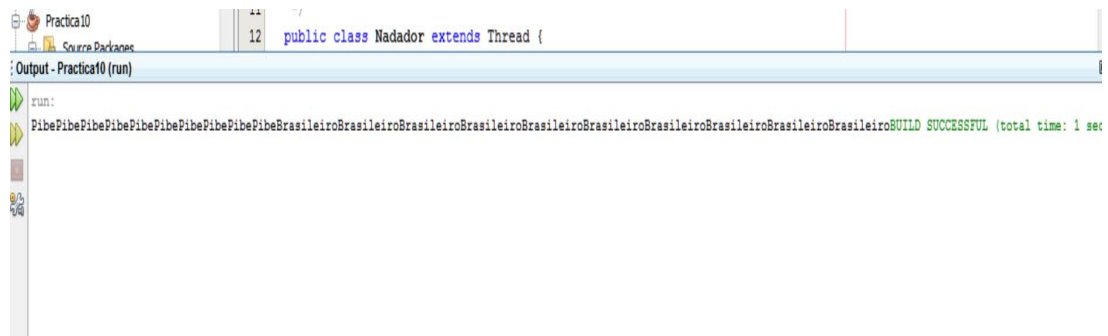
    private String id;

    public Nadador(String n)
    {
        id=n;
    }

    public void Inicia()
    {
        Corre();
    }

    private void Corre()
    {
        for (int i= 0; i < 10;i++){
            System.out.print(id);
        }
    }
}
```

```
8 /**
9  *
10  * @author ime
11  */
12 public class Competencia {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         // TODO code application logic here
19         Nadador pibe= new Nadador("Pibe");
20         Nadador brasileiro = new Nadador("Brasileiro");
21         pibe.Inicia();
22         brasileiro.Inicia();
23
24     }
25
26 }
27
```



7. Modifique la clase Nadador para que ahora herede de la clase Thread , lo que significa que ahora es un hilo su nuevo código debe ser el siguiente:

```
public class Nadador extends Thread {  
  
    private String id;  
  
    public Nadador(String n)  
    {  
        id=n;  
    }  
    public void run()  
    {  
        int i=0;  
        System.out.println("Comienza .. "+ id);  
        while(i<5)  
        {  
            try  
            {  
                Thread.sleep(5000);  
                System.out.println(id);  
            }  
            catch(InterruptedException ex)
```

```

        {
        }
        i++;
    }
    System.out.println("Termina .. " + id);
}
}

```

8. Modifique la nuevamente la clase principal Competencia y dado que ahora nadador es un hilo y ya no tiene el método Inicia(), cambie la llamada al método start() en lugar de Inicia(), para ambas instancias y ejecute la clase y compare los resultados con la anterior ejecución.

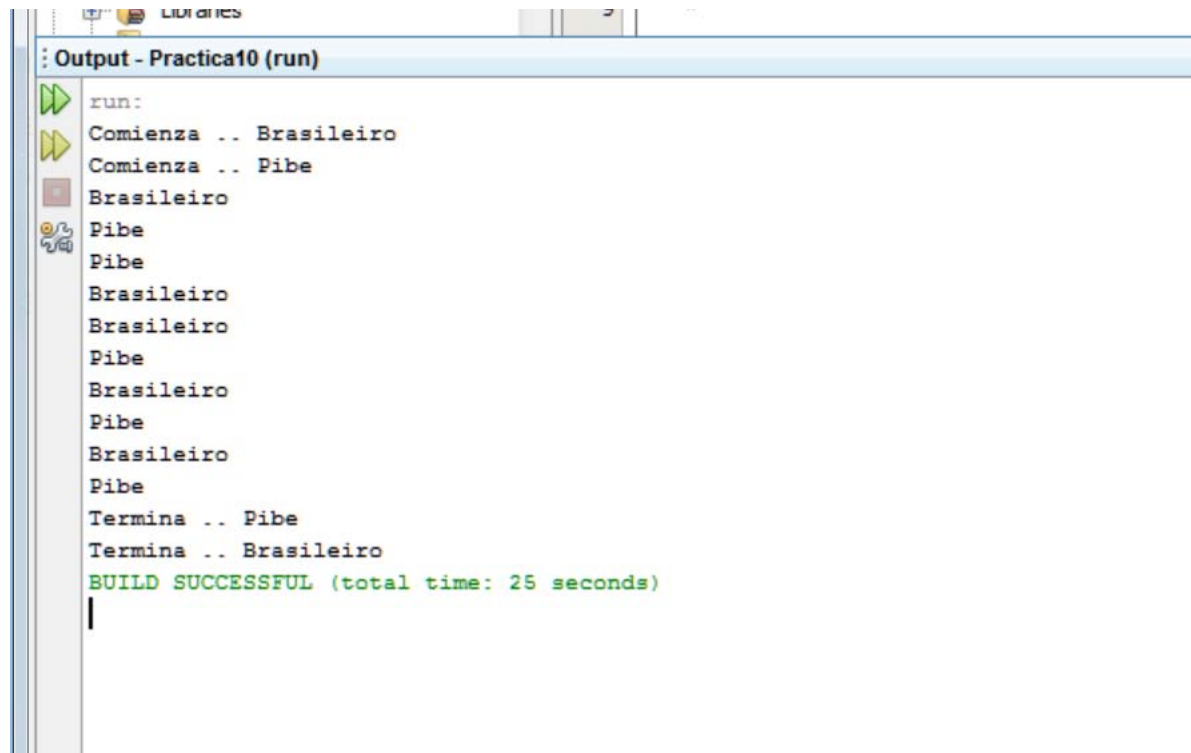
El resultado de realizar los dos pasos anteriores se muestra a continuación:

```

L  */
  public class Competencia {

      /**
       * @param args the command line arguments
       */
      public static void main(String[] args) {
          // TODO code application logic here
          Nadador pibe= new Nadador("Pibe");
          Nadador brasileiro = new Nadador("Brasileiro");
          pibe.start();
          brasileiro.start();
      }
  }
}

```



```
Output - Practica10 (run)
run:
Comienza .. Brasileiro
Comienza .. Pibe
Brasileiro
Pibe
Pibe
Brasileiro
Brasileiro
Pibe
Brasileiro
Pibe
Brasileiro
Pibe
Termina .. Pibe
Termina .. Brasileiro
BUILD SUCCESSFUL (total time: 25 seconds)
```

Aparte del hecho de que ahora los resultados salen uno por línea, se nota que a pesar de que se lanza primero en código de la clase Competencia la ejecución del método start(), de nadador de la instancia “pibe”, - el cual por cierto no se ve en la clase Nadador lo que significa que se hereda de la clase Thread-, la ejecución demuestra que inicia primero la instancia “brasileiro”, pero a pesar de iniciar primero, es la instancia “pibe” , la que termina antes. Además se ve la alternancia de ejecución entre los ciclos “for” de cada instancia, es decir, se observa cómo se ejecuta una vuelta de “brasileiro” y luego una de “pibe” y luego dos de “brasileiro” y así consecutivamente; esto es porque en realidad se han creado dos hilos de ejecución independientes del principal que es Competencia y todos ellos compiten por el procesador como si fuesen procesos independientes.

10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 11

2. Nombre: Creación de un programa que cree y controle hilos.

3. Objetivo: •Comprender las diferentes formas de proporcionadas por el lenguaje para la creación de un programa que utilice hilos

4. Introducción. En esta práctica se aplicarán las dos formas en las que en java se pueden crear hilos, utilizando una interfaz y heredando una clase.

5. Temas y subtemas correlacionados con la práctica.

Tema IV Programación concurrente (MultiHilos)

4.3. Creación y control de hilos.

El alumno ejecutará un programa que crea un hilo implementando la interfaz Runnable y después un programa que crea un hilo heredando de la clase Thread y comentará sobre las diferencias entre ambos.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el tercer subtema del tema dado en la teoría de la asignatura.

.

Puntos a desarrollar en la práctica:

1. En el entorno de desarrollo crear un proyecto llamado Practica11 y agregar una nueva clase llamada HiloRunnable y copie en ella el siguiente código:

```
public class HiloRunnable implements Runnable
{
    public void run()
    {
        for (int i = 0; i < 2; i++)
        {
            System.out.println("Hola soy el hilo hecho con la interfaz runnable");
            try {
                Thread.sleep((long)(Math.random() * 900));
            } catch (InterruptedException e) {}
        }
    }
}
```

2. Añada al proyecto una clase llamada HiloThread y copie en ella el siguiente código:

```
public class HiloThread extends Thread
{
    public void run()
    {
        for (int i = 0; i < 2; i++)
        {
            System.out.println("Hola soy el hilo hecho con la clase Thread");
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
    }
}
```

```

        }
    }
}

```

3. Explique las diferencias entre el código de la clase HiloThread y de la clase HiloRunnable.
4. Explique qué hace el método sleep y si es método de instancia o método de clase
5. Cree una clase principal, es decir, que tenga método Main(), llamada PruebaPractica11 y copie en ella el siguiente código:

```

public class PruebaPractica11 {

    public static void main(String[] args) {
        //Creamos un hilo utilizando la primera clase
        //y lo ponemos en marcha
        HiloThread ht = new HiloThread();
        ht.start();
        //Creamos un hilo utilizando la segunda clase
        //y lo ponemos en marcha
        Thread hr = new Thread(new HiloRunnable());
        hr.start();
    }
}

```

6. Utilizando como referencia el código del método Main() de la clase principal, explique la diferencia al crear las instancias de la clase HiloThread y de la clase HiloRunnable.
7. Ejecute la clase principal y comente el resultado.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. En el entorno de desarrollo crear un proyecto llamado Practica11 y agregar una nueva clase llamada HiloRunnable y copie en ella el siguiente código:

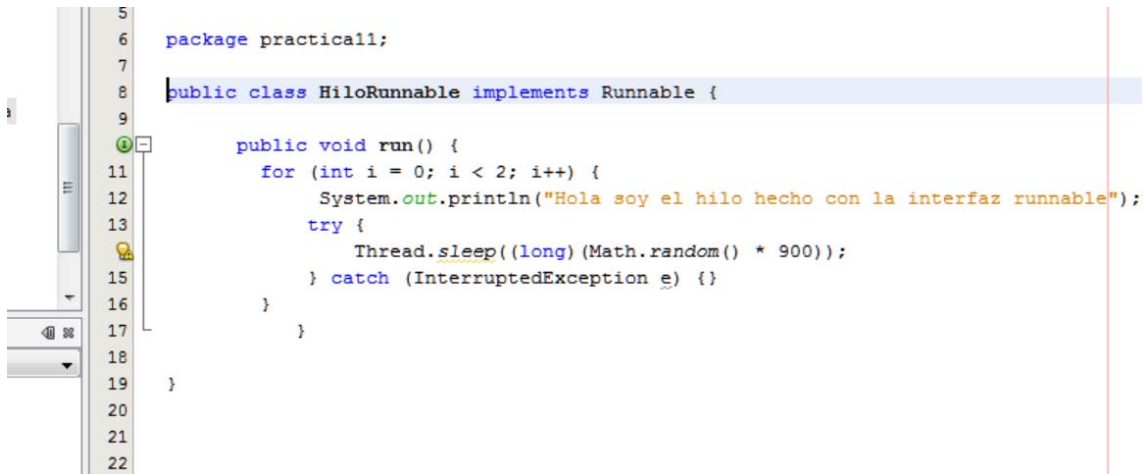
```
public class HiloRunnable implements Runnable
{
    public void run()
    {
        for (int i = 0; i < 2; i++)
        {
            System.out.println("Hola soy el hilo hecho con la interfaz runnable");
            try {
                Thread.sleep((long)(Math.random() * 900));
            }
        }
    }
}
```

```

    } catch (InterruptedException e) {}
}
}
}

```

El resultado de realizar este paso se muestra a continuación:



```

5
6 package practica11;
7
8 public class HiloRunnable implements Runnable {
9
10     public void run() {
11         for (int i = 0; i < 2; i++) {
12             System.out.println("Hola soy el hilo hecho con la interfaz runnable");
13             try {
14                 Thread.sleep((long) (Math.random() * 900));
15             } catch (InterruptedException e) {}
16         }
17     }
18 }
19
20
21
22

```

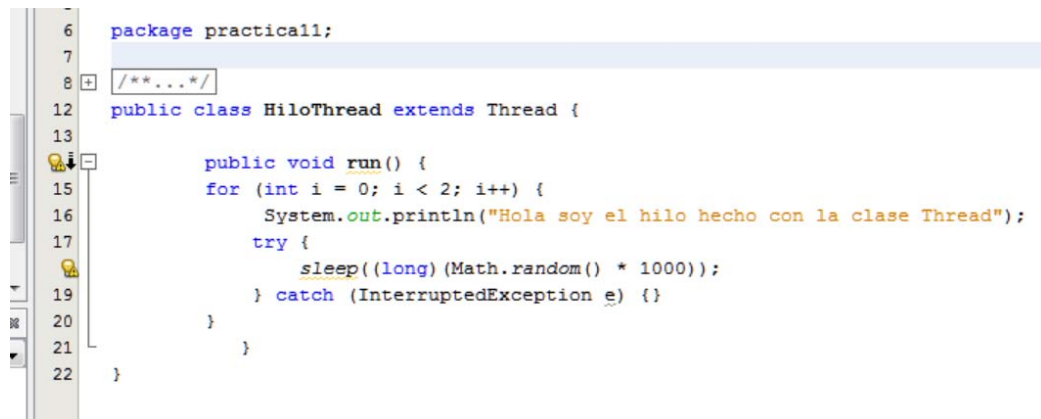
2. Añada al proyecto una clase llamada HiloThread y copie en ella el siguiente código:

```

public class HiloThread extends Thread
{
    public void run()
    {
        for (int i = 0; i < 2; i++)
        {
            System.out.println("Hola soy el hilo hecho con la clase Thread");
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
    }
}

```

El resultado de realizar este paso se muestra a continuación:



```
6 package practica11;
7
8 /**...*/
12 public class HiloThread extends Thread {
13
14     public void run() {
15         for (int i = 0; i < 2; i++) {
16             System.out.println("Hola soy el hilo hecho con la clase Thread");
17             try {
18                 sleep((long) (Math.random() * 1000));
19             } catch (InterruptedException e) {}
20         }
21     }
22 }
```

3. Explique las diferencias entre el código de la clase HiloThread y de la clase HiloRunnable.

Prácticamente no existe diferencia entre ambos métodos a excepción de la forma de llamar al método sleep, en la clase HiloThread dado que ésta se deriva de la clase Thread hereda sus métodos por lo que no es necesario utilizar la clase Thread para mandarlo a ejecutar.

4. Explique qué hace el método sleep y si es método de instancia o método de clase

El método sleep es un método de clase, como se puede ver en el código de la clase HiloRunnable, ya que no se requiere crear una instancia de la clase Thread para llamarlo, simplemente se usa la misma clase. Este método lo que hace es bloquear la ejecución del hilo por el tiempo en milisegundos estipulado a través de su parámetro.

5. Cree una clase principal, es decir que tenga método Main(), llamada PruebaPractica11 y copie en ella el siguiente código:

```
public class PruebaPractica11 {

    public static void main(String[] args) {
        //Creamos un hilo utilizando la primera clase
        //y lo ponemos en marcha
    }
}
```

```

        HiloThread ht = new HiloThread();
        ht.start();
    //Creamos un hilo utilizando la segunda clase
    //y lo ponemos en marcha
        Thread hr = new Thread(new HiloRunnable());
        hr.start();
    }
}

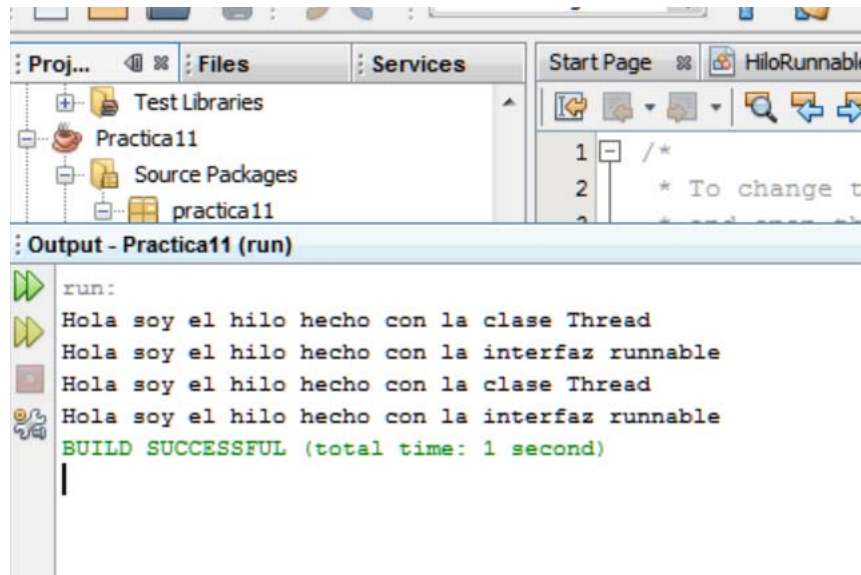
```

6. Utilizando como referencia el código del método Main() de la clase principal, explique la diferencia al crear las instancias de la clase HiloThread y de la clase HiloRunnable.

Dado que la clase HiloRunnable implementa la interfaz Runnable , es necesario crear un hilo para que pueda utilizar el método Run() en éste, por ello se crea una instancia de un hilo con el constructor Thread que recibe como parámetro una Instancia de una clase que implemente a la interfaz Runnable, mientras que en el caso de la clase HiloThread ya es un hilo, por eso sólo es necesario crear una instancia de esta clase.

7. Ejecute la clase principal y comente el resultado.

El resultado de realizar de este paso se muestra a continuación:



Nuevamente como en la práctica anterior, a pesar de que la instancia de la clase HiloThread se bloquee en total 200 milisegundos menos que la clase HiloRunnable, es la que termina primero, en esta ejecución aunque en otra el resultado puede variar. Esto se debe obviamente a que ambas instancias son procesos independientes y compiten por el procesador al igual que todos los demás programas que se estén ejecutando en el computador en dicho momento.

10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 12

2. Nombre: Creación de un programa que Maneje la sincronización de hilos.

3. Objetivo: •Comprender como funciona la sincronización entre hilos

4. Introducción. En esta práctica se ejecutará un programa en dos versiones diferentes que muestre primero como funciona cuando no se aplica la sincronización entre hilos y posteriormente el resultado de hacer la sincronización entre hilos.

5. Temas y subtemas correlacionados con la práctica.

Tema IV Programación concurrente (MultiHilos)

4.4. Sincronización de hilos.

El alumno a través de un programa que se ejecutará en dos versiones distintas : sin sincronización y con sincronización podrá comprender la importancia de aplicar la sincronización cuando un programa o aplicación así lo requiera..

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% el cuarto subtema del tema dado en la teoría de la asignatura.

.

Puntos a desarrollar en la práctica:

1. En el entorno de desarrollo crear un proyecto llamado Practica12 y agregar una nueva clase llamada Mensaje y copie en ella el siguiente código:

```
public class Mensaje
{
    static void ImprimeMensaje(String str1, String str2)
    {
        System.out.print(str1);
        try {
            Thread.sleep(500);
        } catch (InterruptedException ie) {}
        System.out.println(str2);
    }
}
```

2. Añada al proyecto una clase llamada Escribiente y copie en ella el siguiente código:

```
public class Escribiente extends Thread
{
    String str1, str2;

    public Escribiente(String str1, String str2)
    {
        this.str1 = str1;
        this.str2 = str2;
        start();
    }

    public void run()
    {
        Mensaje.ImprimeMensaje(str1, str2);
    }
}
```

3. Añada al proyecto una clase principal llamada PruebaPractica12 y copie en ella el siguiente código.

```
public class PruebaPractica12
{
    public static void main(String[] args)
    {
        new Escribiente("Hola ", "que tal");
        new Escribiente("Como estás ", "tú?");
        new Escribiente("Muchas ", "Gracias");
    }
}
```

4. Ejecute la clase principal y comente el resultado.
5. Modifique la clase Mensaje anteponiendo al método ImprimeMensaje() el calificador synchronized.

```
public class Mensaje
{
    synchronized static void ImprimeMensaje(String str1, String str2)
    {
        System.out.print(str1);
        try {
            Thread.sleep(500);
        } catch (InterruptedException ie) { }
        System.out.println(str2);
    }
}
```

6. Ejecute nuevamente la clase principal y compare los resultados obtenidos anteriormente y teniendo en cuenta lo aprendido en clase de teoría explique las diferencias.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. En el entorno de desarrollo crear un proyecto llamado Practica12 y agregar una nueva clase llamada Mensaje y copie en ella el siguiente código:

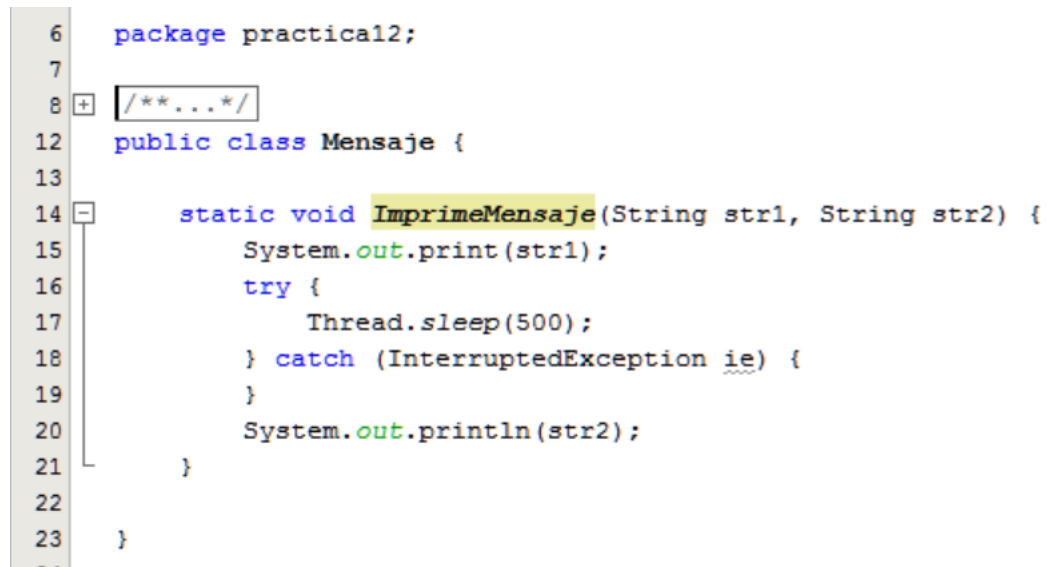
```
public class Mensaje
{
    static void ImprimeMensaje(String str1, String str2)
    {
        System.out.print(str1);
        try {
            Thread.sleep(500);
        } catch (InterruptedException ie) {}
    }
}
```

```

        System.out.println(str2);
    }
}

```

El resultado de realizar este paso se muestra a continuación:



```

6   package practical2;
7
8   /**...*/
12  public class Mensaje {
13
14      static void ImprimeMensaje(String str1, String str2) {
15          System.out.print(str1);
16          try {
17              Thread.sleep(500);
18          } catch (InterruptedException ie) {
19              }
20          System.out.println(str2);
21      }
22
23  }

```

2. Añada al proyecto una clase llamada Escribiente y copie en ella el siguiente código:

```

public class Escribiente extends Thread
{
    String str1, str2;

    public Escribiente(String str1, String str2)
    {
        this.str1 = str1;
        this.str2 = str2;
        start();
    }

    public void run()
    {
        Mensaje.ImprimeMensaje(str1, str2);
    }
}

```

El resultado de realizar este paso se muestra a continuación:

```
+ /**...*/
public class Escribiente extends Thread{

    String str1, str2;

    Escribiente(String str1, String str2) {
        this.str1 = str1;
        this.str2 = str2;
        start();
    }

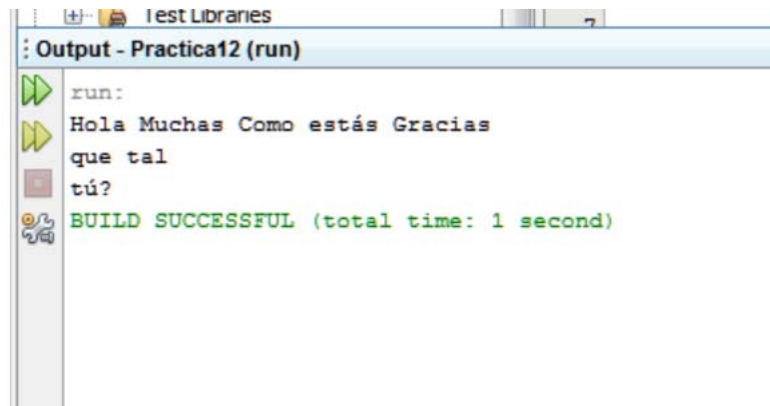
    public void run() {
        Mensaje.ImprimeMensaje(str1, str2);
    }
}
```

3. Añada al proyecto una clase principal llamada PruebaPractica12 y copie en ella el siguiente código.

```
public class PruebaPractica12
{
    public static void main(String[] args)
    {
        new Escribiente("Hola ", "que tal");
        new Escribiente("Como estás ", "tú?");
        new Escribiente("Muchas ", "Gracias");
    }
}
```

4. Ejecute la clase principal y comente el resultado.

El resultado de realizar este paso se muestra a continuación:



Como se observa en la ventana de salida del programa, a pesar de que el primer escritor envía las cadenas "Hola " y "que tal", en esta ejecución aparecen separadas entre las cadenas enviados por los otros dos escritores, esto se debe a que cada uno de ellos hace uso del mismo recurso que es método estático de la clase Mensaje, y que al ser ellos hilos son de ejecución independiente y el recurso compartido, es decir el método `ImprimeMensaje` se bloquea por un determinado tiempo, haciendo que se libere el recurso para otro hilo, en este caso el segundo escritor y luego nuevamente se bloquea por un determinado tiempo liberando nuevamente el recurso para ser usado por el último escritor. Por lo que ninguno de ellos lo aparta en su totalidad y las cadenas que componen el mensaje no están sincronizadas.

5. Modifique la clase Mensaje anteponiendo al método `ImprimeMensaje()` el calificador `synchronized`.

```
public class Mensaje
{
    synchronized static void ImprimeMensaje(String str1, String str2)
    {
        System.out.print(str1);
        try {
            Thread.sleep(500);
        }
    }
}
```

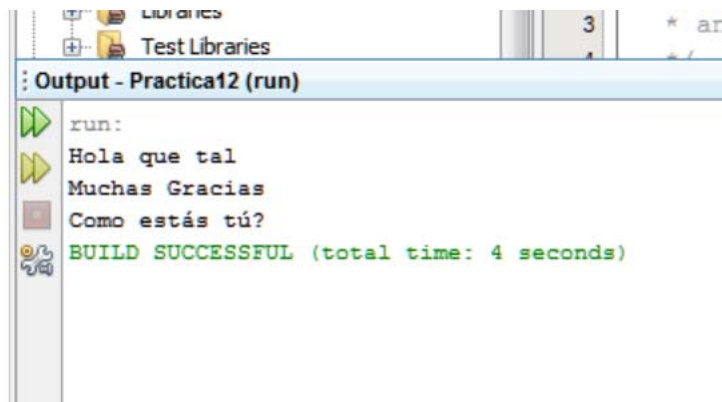
```

        } catch (InterruptedException ie) { }
        System.out.println(str2);
    }
}

```

6. Ejecute nuevamente la clase principal y compare los resultados obtenidos anteriormente y teniendo en cuenta lo aprendido en clase de teoría explique las diferencias.

El resultado de realizar este paso se muestra a continuación:



Como se muestra en la salida las dos cadenas que envía cada escribiente ahora si salen seguidas, esto se debe a que el calificador synchronized “amarra” el recurso Mensaje hasta que el escribiente en turno lo haya terminado de utilizar, haciendo con ello que las cadenas se sincronicen en su salida.

10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 13

2. Nombre: Ejecución de un programa para dispositivos Móvil.

3. Objetivo: Utilizar un entorno de desarrollo específico para la ejecución de un programa para un dispositivo móvil

4. Introducción. En esta práctica se ejecutará un programa utilizando un entorno de desarrollo en específico.

5. Temas y subtemas correlacionados con la práctica.

Tema V Programación de dispositivos móviles

- 5.1. Introducción
- 5.2. Dispositivos móviles
- 5.3. Clasificación y aplicaciones de los dispositivos móviles
- 5.4. Ambientes de las aplicaciones
- 5.5. Protocolo de Interfaz entre aplicaciones
- 5.6. Programación de teléfonos móviles y PDA

El alumno a través de un programa que se ejecutará en Netbeans comprenderá los principios básicos que involucra el desarrollo de aplicaciones móviles, como son los tipos de dispositivos, los ambientes de desarrollo y los protocolos de comunicación utilizados en las interfaces de los ambientes:

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

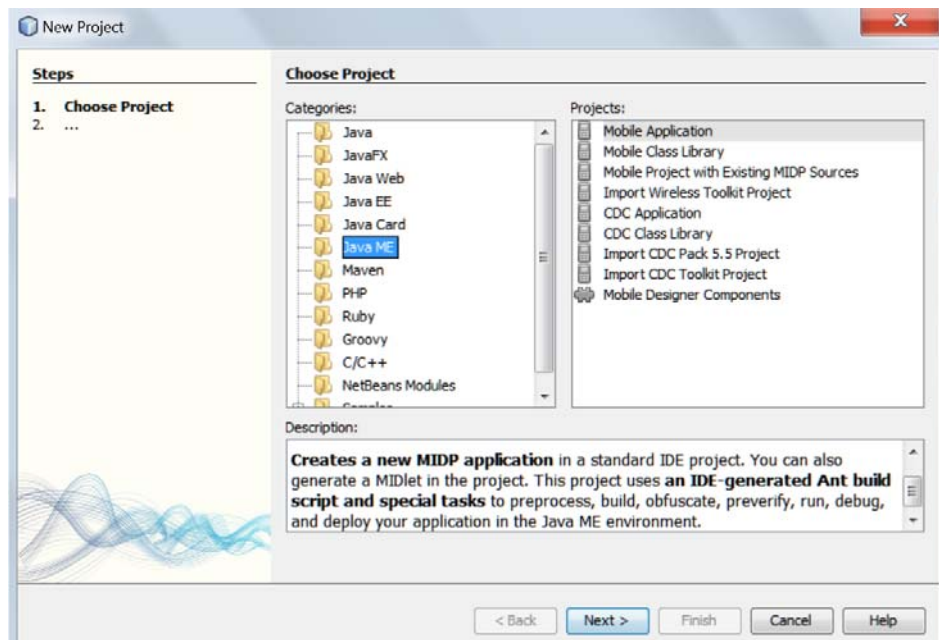
Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% los subtemas del tema dado en la teoría de la asignatura.
- b. Haber instalado el entorno de desarrollo para móviles J2ME dentro del Netbeans.

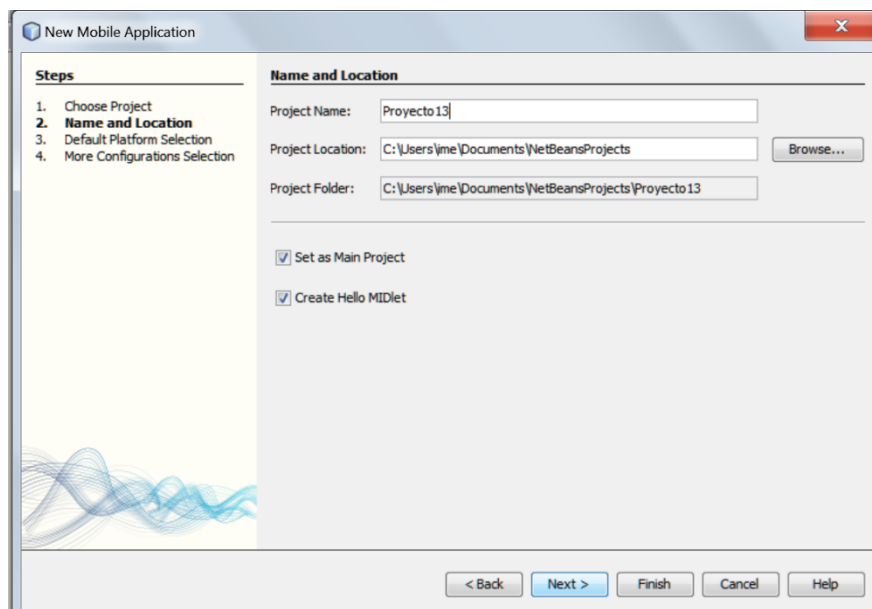
.

Puntos a desarrollar en la práctica:

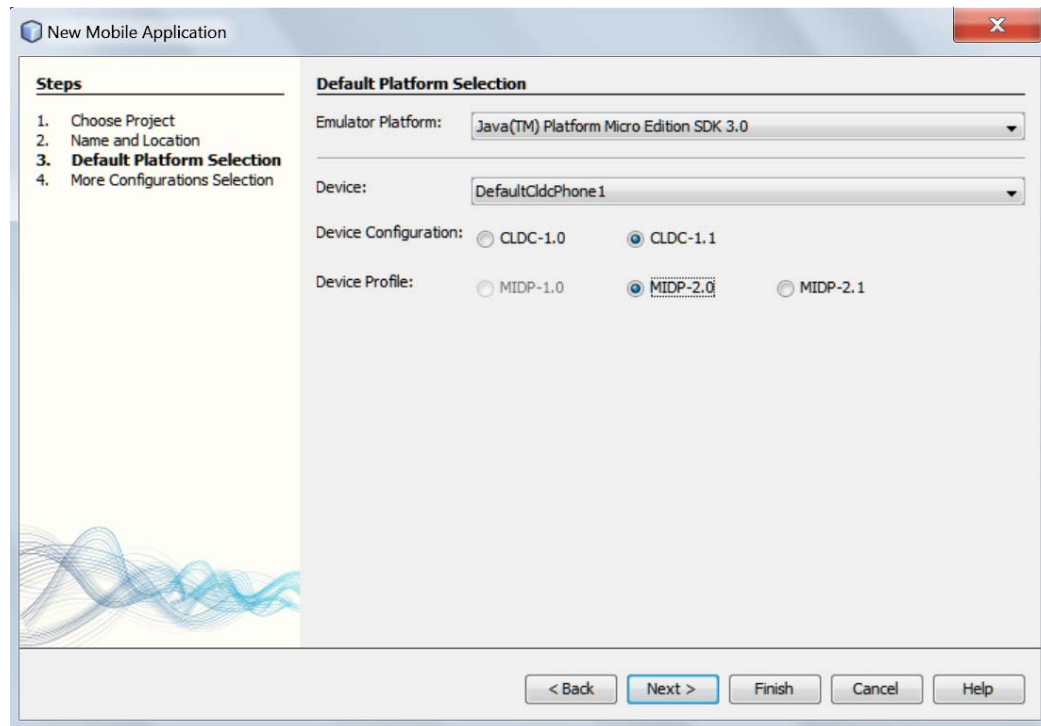
1. En Netbeans crear un proyecto llamado Practica13 con la categoría Java ME, como se muestra en la siguiente figura:



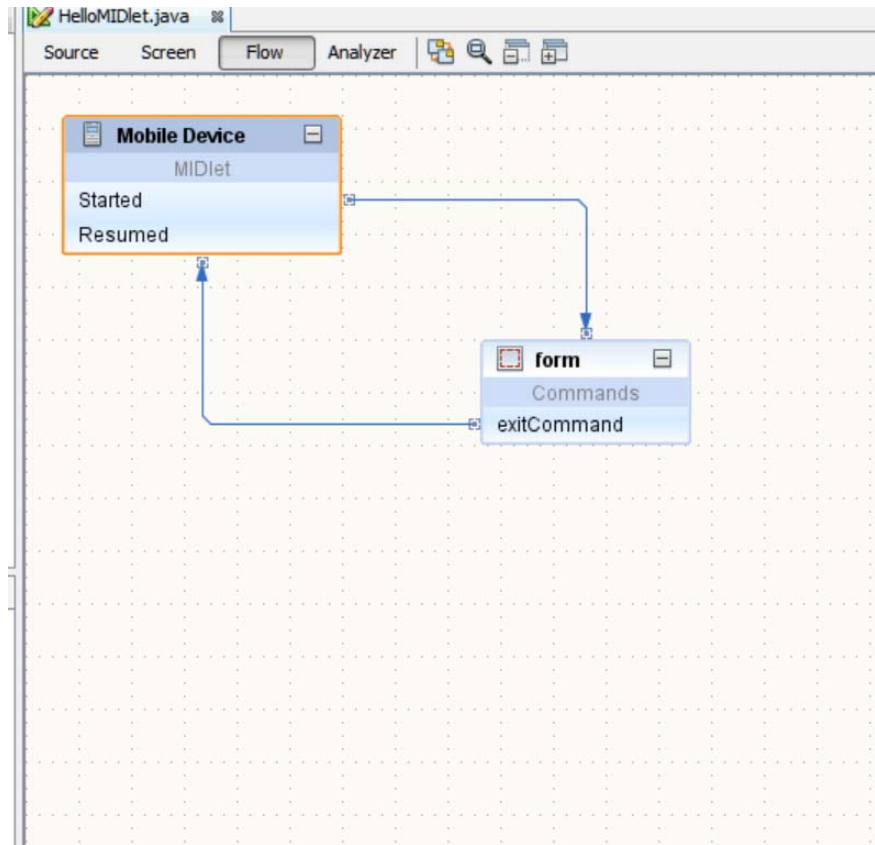
Con lo cual se activará la funcionalidad de J2ME y se desplegará la siguiente ventana de diálogo:



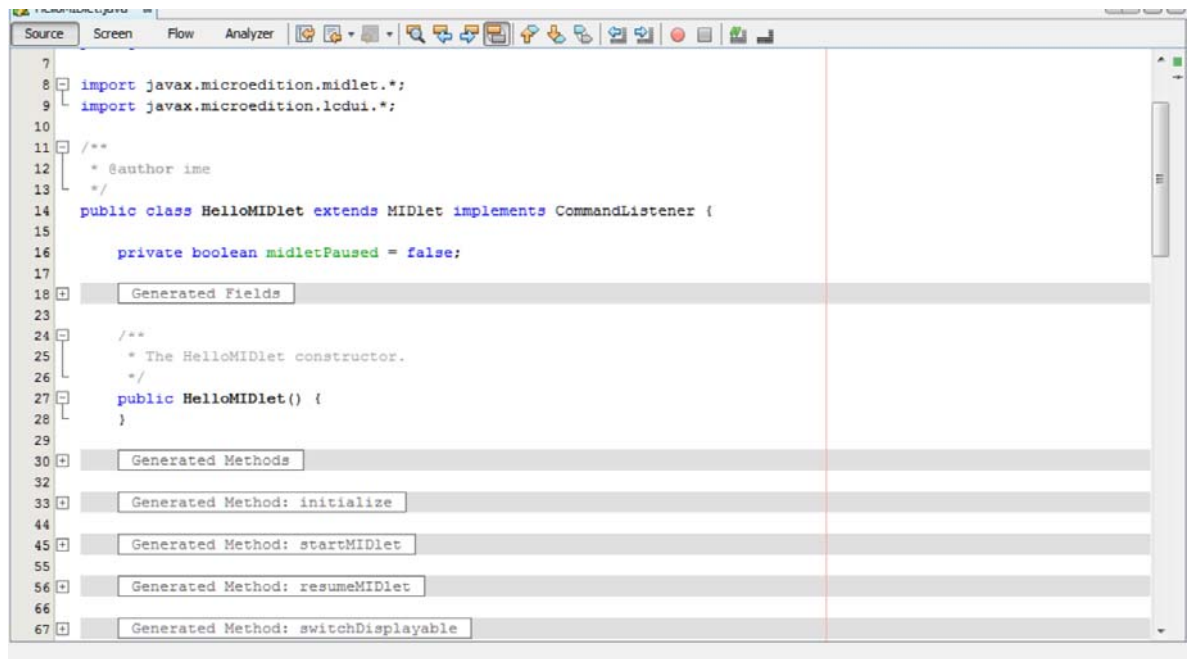
Se hace clic sobre el botón “Siguiente” para seleccionar la plataforma de emulación, el dispositivo, las versiones tanto de su configuración de CLDC como la del perfil MIDP, en este caso se dejan la configuración por defecto CLDC 1.1 y el perfil se elije el MIDP 2.0 y como dispositivo un teléfono por defecto Cldc y finalmente se hace clic sobre el botón “Terminar”



Como resultado se genera automáticamente el diagrama de flujo entre el Midlet del dispositivo y los comandos de la forma, en este caso el único comando es “exit”..



1. Haciendo clic sobre pantalla, se puede observa la pantalla del dispositivo y el elemento de cadena que tiene la etiqueta llamada Hello, haciendo clic sobre ésta se observa en el recuadro inferior derecho las propiedades del elemento, modifique la propiedad Text para que diga “ Hola, ¿cómo estás?”, en lugar de “Hello,World” y la de Label como Hola en lugar de “Hello” :
2. Ahora ejecute el programa y muestre el resultado.
3. Ejecute la clase principal y comente el resultado.
4. Cierre el emulador y haga clic sobre fuente, con lo cual se desplegará el archivo fuente y podrá ver el código generado del programa, como se muestra en la siguiente figura.



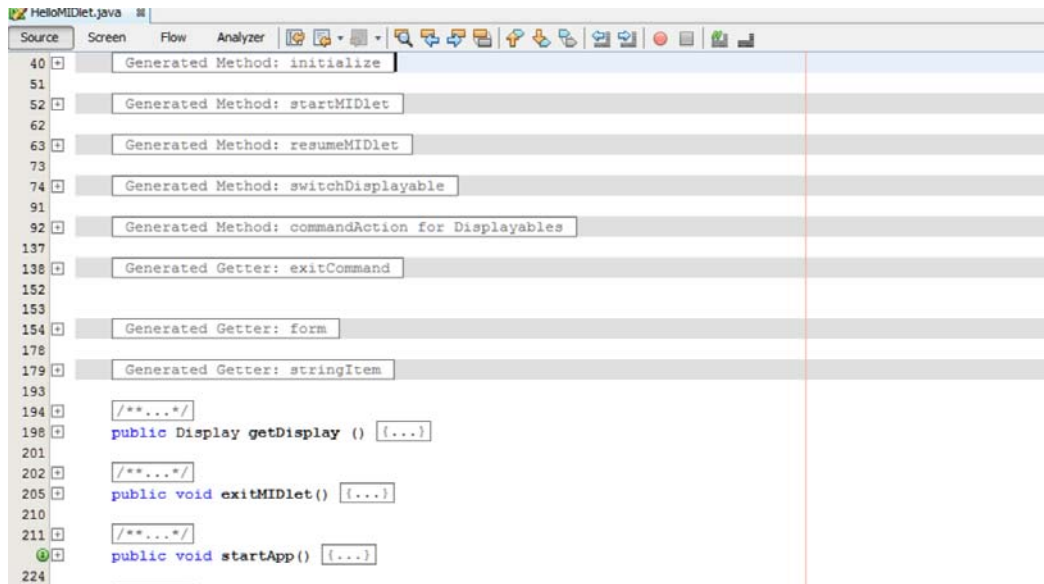
Abra la sección de campos y agregue las siguientes líneas de código:

```

TextField tFieldA = new TextField("Num1=", "1", 5, TextField.NUMERIC);
TextField tFieldB = new TextField("Num2=", "2", 5, TextField.NUMERIC);
StringItem resultItem = new StringItem("Resultado", "");
Command sumaComando = new Command("Suma", Command.SCREEN, 1);

```

5. Abra la sección de comandos de acción para despliegue



y en el método `commandAction`, modifique el código comentando el existente y luego añadiendo las siguientes líneas de código:

```
if (displayable == form)
{
    if (command == exitCommand)
    {
        exitMIDlet();
    }
    if(command == sumaComando)
    {
        try
        {
            int nA = Integer.parseInt(tFieldA.getString());
            int nB = Integer.parseInt(tFieldB.getString());
            int nResult = nA+ nB;
            resultItem.setText(Integer.toString(nResult));
        }
        catch (Exception e)
        {
            e.printStackTrace();
            resultItem.setLabel("Error:");
            resultItem.setText(e.toString());
        }
    }
}
```

6. Ahora abra el código generado para la forma y modifique el código del método `getForm()`, comentando las líneas existentes de código y añadiendo las siguientes:

```
form = new Form("Bienvenido", new Item[] { getStringItem() });
form.addCommand(getExitCommand());
form.setCommandListener(this);
form.addCommand(sumaComando);
form.append(tFieldA);
form.append(tFieldB);
form.append(resultItem);
```

7. Ejecute nuevamente el programa, muestre y comente el resultado

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

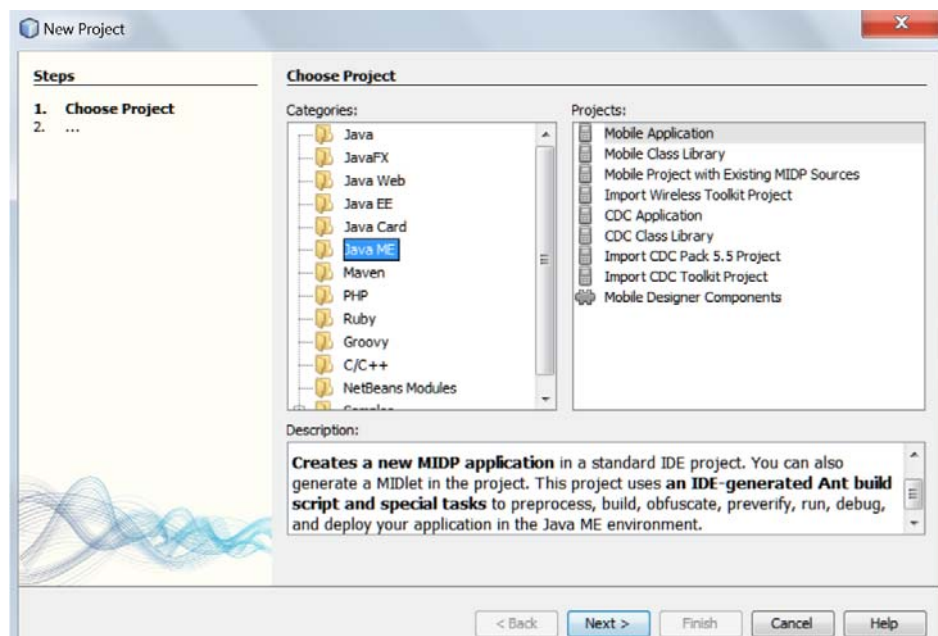
Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

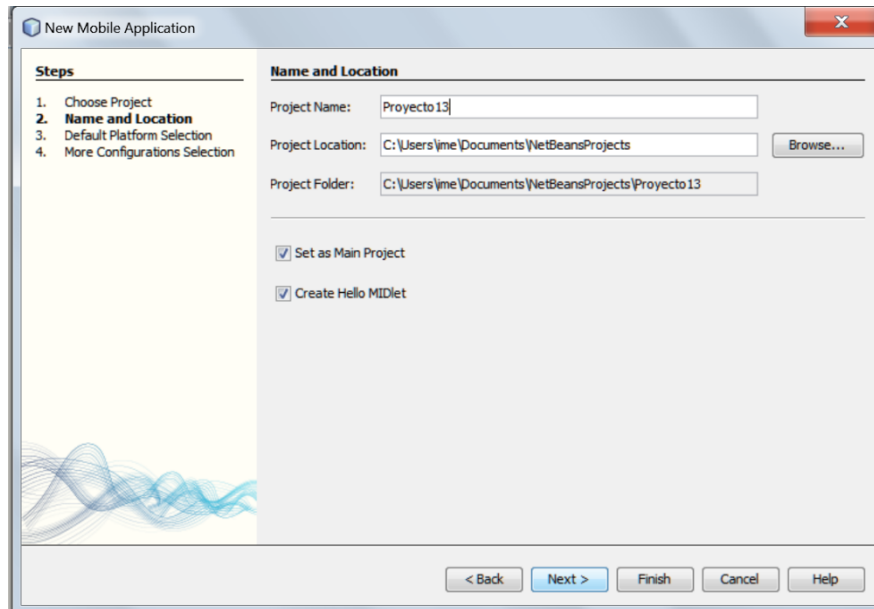
9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

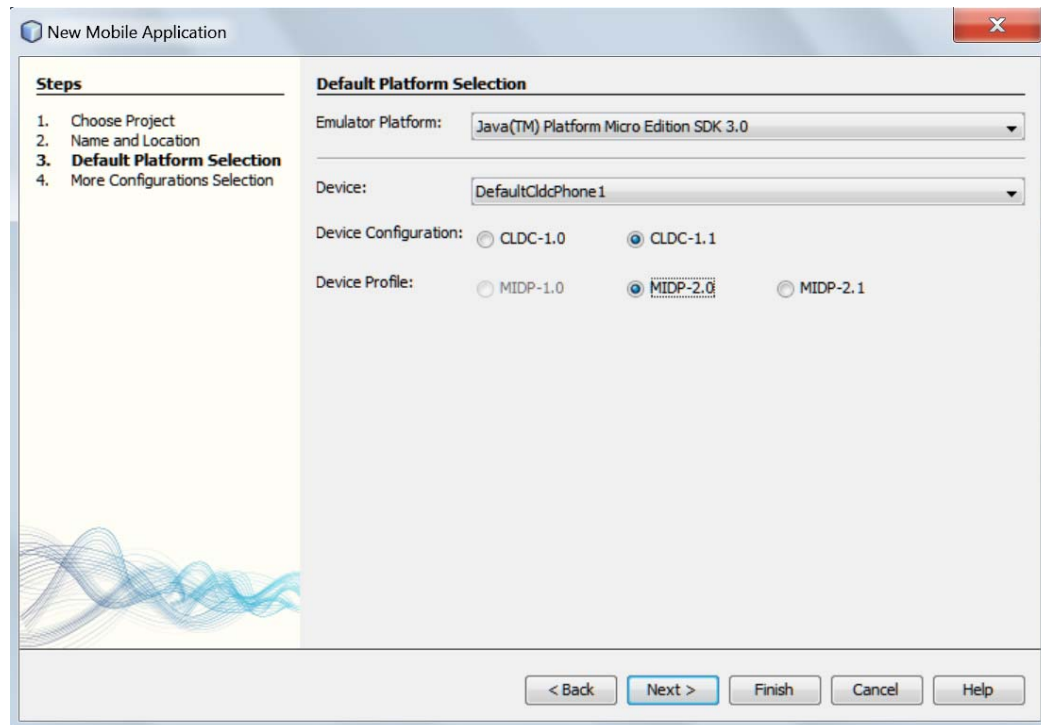
1. En Netbeans crear un proyecto llamado Practica13 con la categoría Java ME, como se muestra en la siguiente figura:



Con lo cual se activará la funcionalidad de J2ME y se desplegará la siguiente ventana de diálogo:



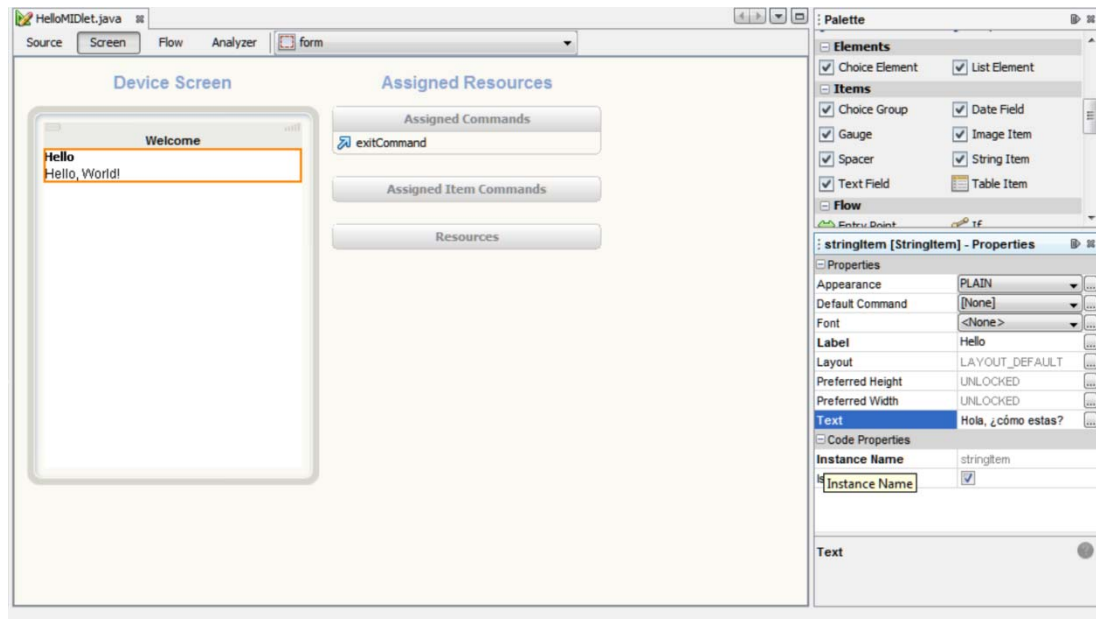
Se hace clic sobre el botón “Siguiente” para seleccionar la plataforma de emulación, el dispositivo, las versiones tanto de su configuración de CLDC como la del perfil MIDP, en este caso se dejan la configuración por defecto CLDC 1.1 y el perfil se elije el MIDP 2.0 y como dispositivo un teléfono por defecto Cdlc y finalmente se hace clic sobre el botón “Terminar”



Como resultado se genera automáticamente el diagrama de flujo entre el Midlet del dispositivo y los comandos de la forma, en este caso el único comando es “exit”.

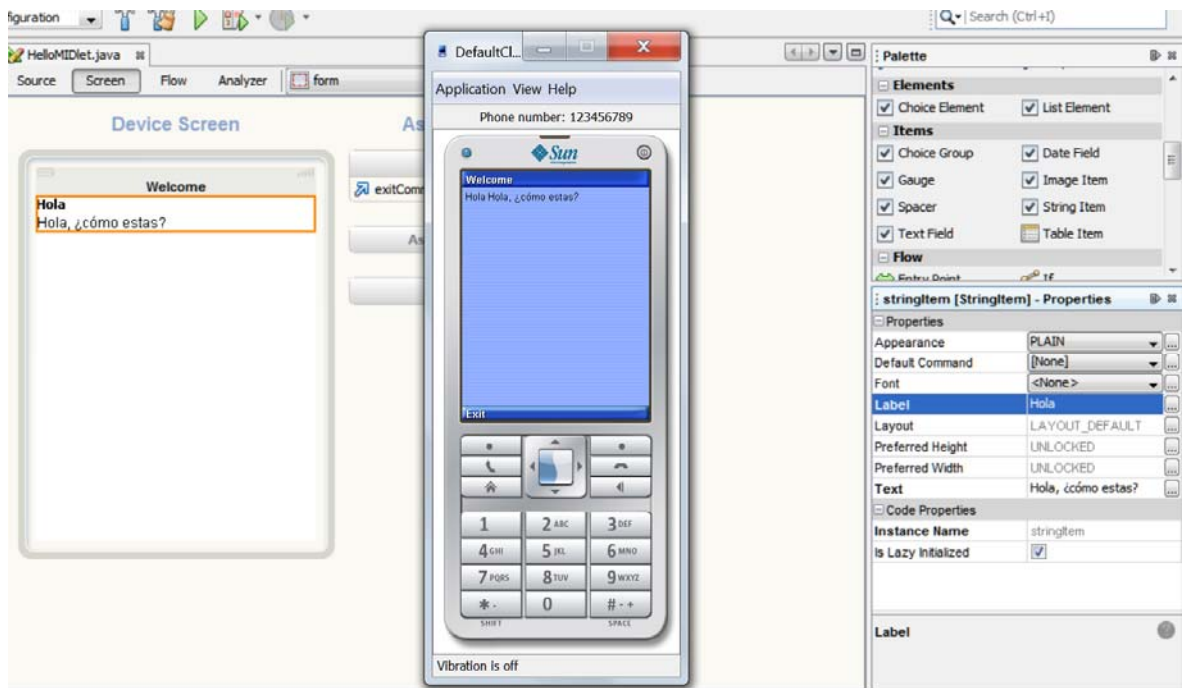
2. Haciendo clic sobre pantalla, se puede observa la pantalla del dispositivo y el elemento de cadena, haciendo clic sobre éste se observa en el recuadro inferior derecho las propiedades del elemento, modifique la propiedad Text para que diga “ Hola, ¿cómo estás?”, en lugar de “Hello,World” y la de Label como Hola en lugar de “Hello” :

El resultado de realizar este paso se muestra a continuación:

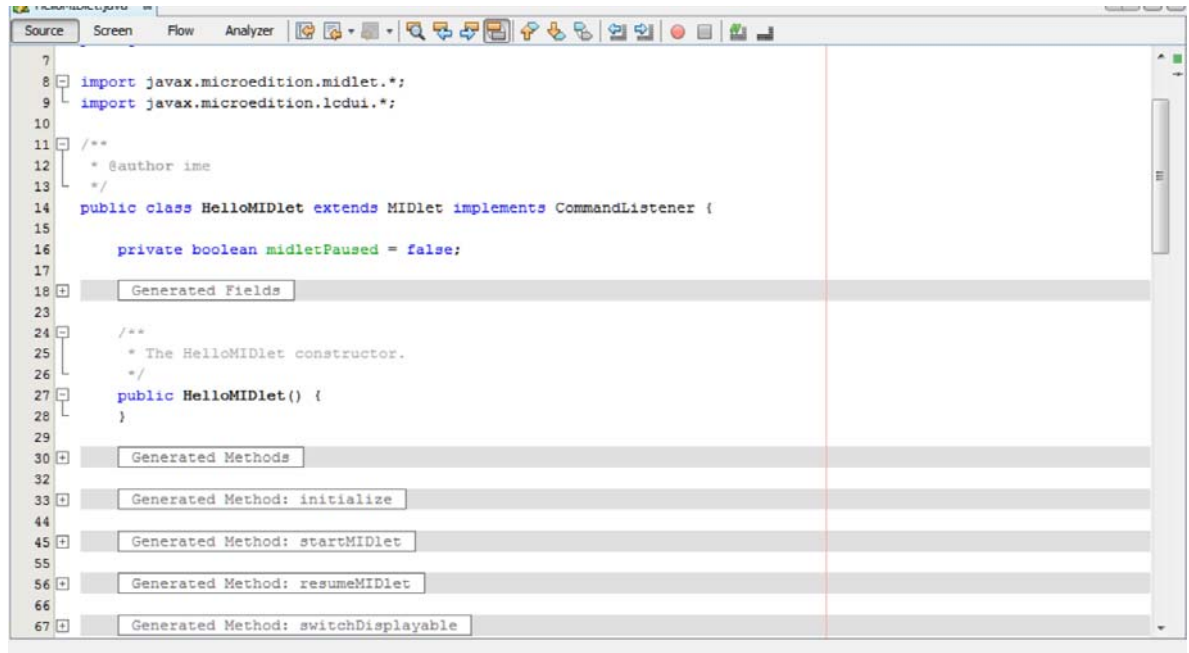


3. Ahora ejecute el programa y muestre el resultado

Al ejecutar el programa se levante el manejador para el dispositivo seleccionado, algo muy similar a cuando se levanta un servidor web, mostrando un emulador de un teléfono celular. En su pantalla se ve el elemento de cadena y en la esquina inferior izquierda el comando exit, como se muestra en la figura a continuación:



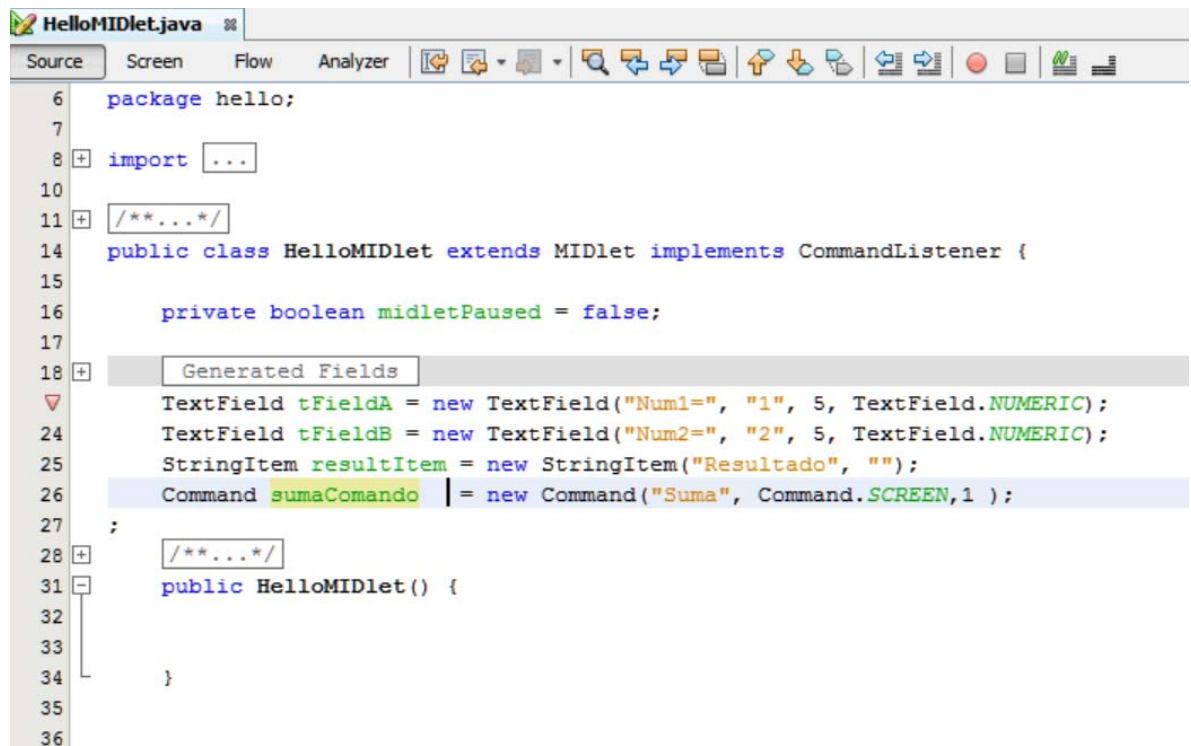
4. Cierre el emulador y haga clic sobre fuente, con lo cual se desplegará el archivo fuente y podrá ver el código del programa, como se muestra en la siguiente figura.



Abra la sección de campos y agregue las siguientes líneas de código:

```
TextField tFieldA = new TextField("Num1=", "1", 5, TextField.NUMERIC);
TextField tFieldB = new TextField("Num2=", "2", 5, TextField.NUMERIC);
StringItem resultItem = new StringItem("Resultado", "");
Command sumaComando = new Command("Suma", Command.SCREEN, 1);
```

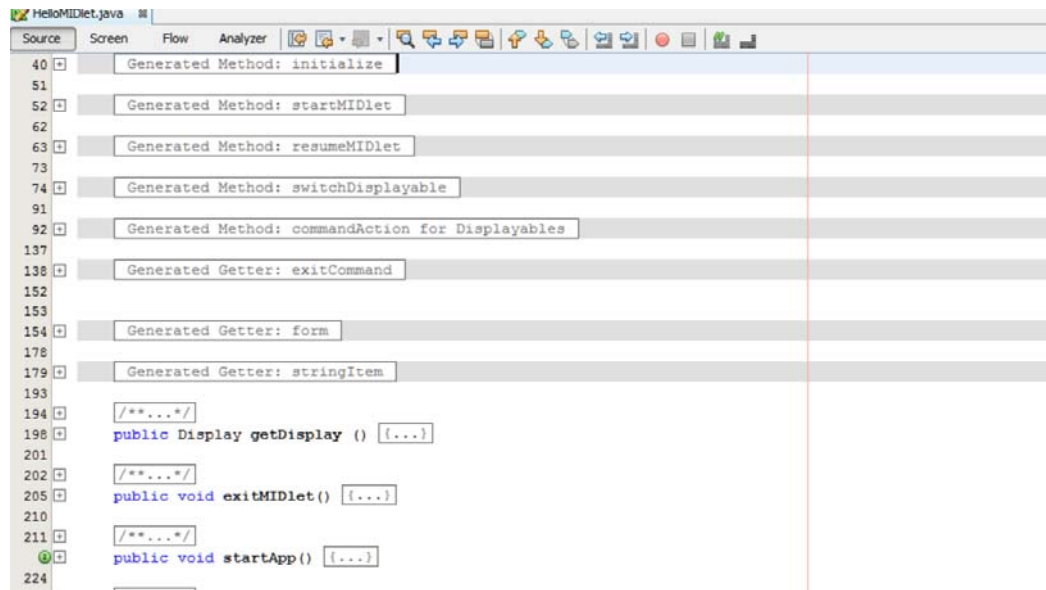
El resultado de realizar este paso se muestra a continuación:



```
6 package hello;
7
8 import ...
9
10 /**...*/
11
14 public class HelloMIDlet extends MIDlet implements CommandListener {
15
16     private boolean midletPaused = false;
17
18     Generated Fields
19     TextField tFieldA = new TextField("Num1=", "1", 5, TextField.NUMERIC);
20     TextField tFieldB = new TextField("Num2=", "2", 5, TextField.NUMERIC);
21     StringItem resultItem = new StringItem("Resultado", "");
22     Command sumaComando = new Command("Suma", Command.SCREEN, 1 );
23
24 ;
25
26 /**...*/
27
28 public HelloMIDlet() {
29
30
31
32
33
34 }
35
36
```

Con este paso lo que se hace en realidad es añadir dos cajas de texto con sus etiquetas incluidas llamadas Num1 y Num2 e inicialmente tendrá la primera el valor 1 y la segunda el valor dos, y por otra parte se agrega un nuevo elemento de cadena, que será utilizado para mostrar el resultado de ejecutar el comando sumaComando que también es agregado, y como se indica en su constructor agregado a la pantalla del teléfono, del lado derecho como lo indica el último parámetro .

5. Abra la sección de comandos de acción para despliegue



y en el método `commandAction`, modifique el código comentando el existente y luego añadiendo las siguientes líneas de código:

```

if (displayable == form)
{
    if (command == exitCommand)
    {
        exitMIDlet();
    }
    if (command == sumaComando)
    {
        try
        {
            int nA = Integer.parseInt(tFieldA.getString());
            int nB = Integer.parseInt(tFieldB.getString());
            int nResult = nA + nB;
            resultItem.setText(Integer.toString(nResult));
        }
        catch (Exception e)
        {
            e.printStackTrace();
            resultItem.setLabel("Error:");
            resultItem.setText(e.toString());
        }
    }
}

```

El resultado de realizar este paso se muestra a continuación:

```

73
74 + Generated Method: switchDisplayable
91
92 //<editor-fold defaultstate="collapsed" desc=" Generated Method: commandActi
93 /**...*/
94 public void commandAction(Command command, Displayable displayable) {
95     // write pre-action user code here
96     /*
97     if (displayable == form) {
98         if (command == exitCommand) {
99             // write pre-action user code here
100             exitMIDlet();
101             // write post-action user code here
102         }
103     */
104     if (displayable == form)
105     {
106         if (command == exitCommand)
107         {
108             exitMIDlet();
109         }
110         if (command == sumaComando)
111         {
112             try
113             {
114                 int nA = Integer.parseInt(tFieldA.getString());
115                 int nB = Integer.parseInt(tFieldB.getString());
116                 int nResult = nA+ nB;
117                 resultItem.setText(Integer.toString(nResult));
118             }
119             catch (Exception e)
120             {
121                 e.printStackTrace();
122                 resultItem.setLabel("Error:");
123                 resultItem.setText(e.toString());
124             }
125         }
126     }
127 }
128 //</editor-fold>
129
130 + Generated Getter: exitCommand
131
132
133 + Generated Getter: form

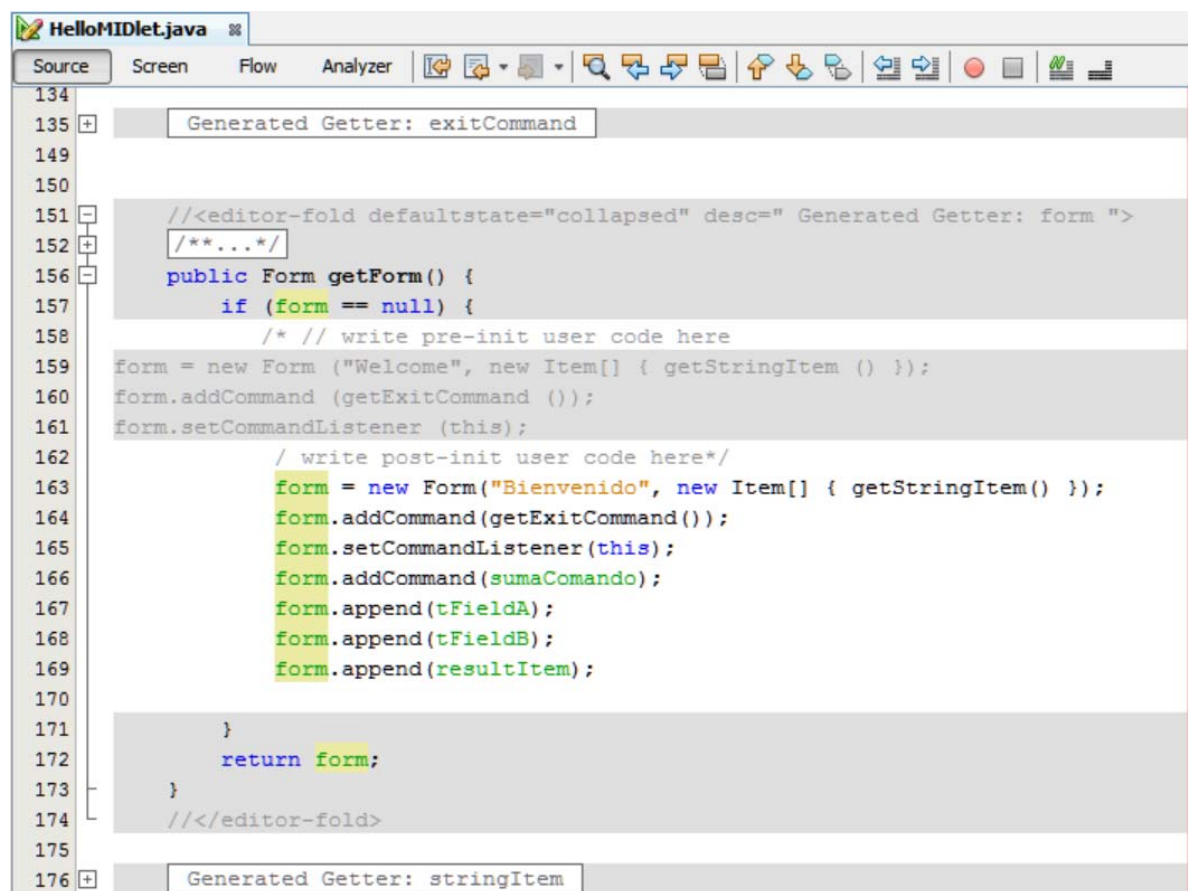
```

Una vez comentado el contenido original del método commandAction, se repite su código para incluir la acción efectuada al comando exit, pero ahora se agrega también la acción efectuada al comando suma.

6. Ahora abra el código generado para la forma y modifique el código del método `getForm()`, comentando las líneas existentes de código y añadiendo las siguientes:

```
form = new Form("Bienvenido", new Item[] { getStringItem() });
form.addCommand(getExitCommand());
form.setCommandListener(this);
form.addCommand(sumaComando);
form.append(tFieldA);
form.append(tFieldB);
form.append(resultItem);
```

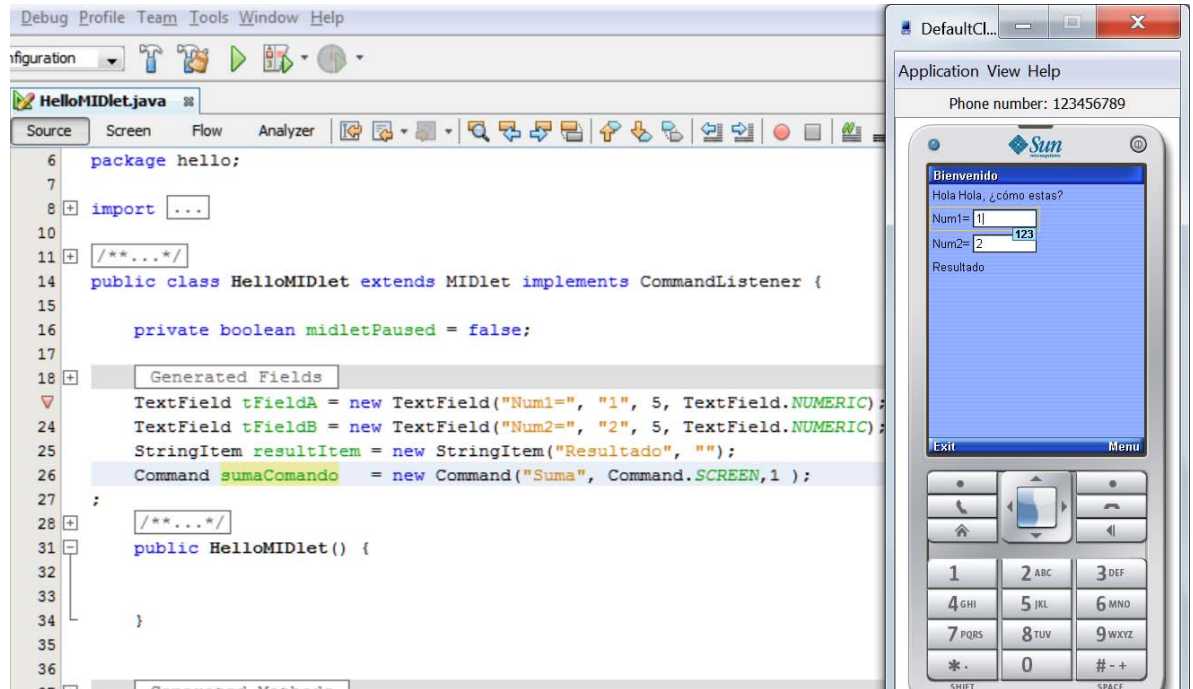
El resultado de realizar este paso se muestra a continuación:



Una vez puesta la acción al comando suma, falta solamente añadir a la forma las cajas de texto creadas y el comando suma.

7. Ejecute nuevamente el programa, muestre y comente el resultado

El resultado de realizar este paso se muestra a continuación:

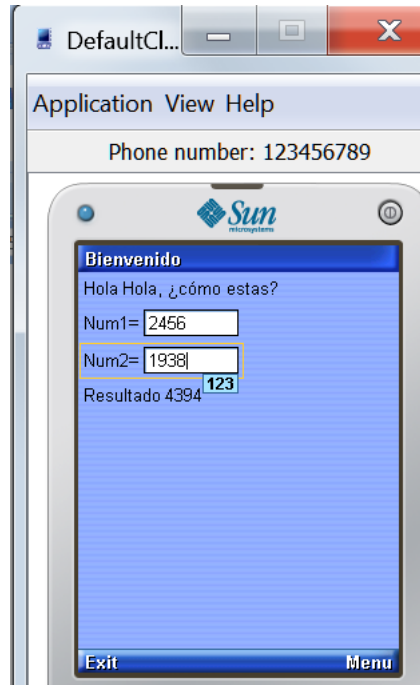


como puede observarse en la figura aparecen las dos cajas de texto creadas como numéricas y con los valores iniciales puestos, así como el elemento Resultado, sin ningún texto como se creó originalmente

Ahora bien la siguiente figura como se han modificado los valores de la caja de texto y como se selecciona la opción Menú para escoger el comando suma .



Y la figura a continuación muestra el resultado de seleccionar el comando Suma



La presente práctica ha mostrado como se pueden añadir elementos a un Midlet y comandos de acción desde código. Para comprender la programación, aparte de el cómo en un ambiente de aplicaciones móviles como J2ME, permite seleccionar de los diferentes tipos de dispositivos, de acuerdo a sus protocolos de interfaz como son las configuraciones y los perfiles

10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008

1. PRÁCTICA No. 14

2. Nombre: Creación de un programa en un dispositivo móvil para el envío y recepción de información.

3. Objetivo: Comprender los principios básicos de la comunicación entre dispositivos móviles dentro de una aplicación para un teléfono móvil.

4. Introducción. En esta práctica se ejecutará un programa que realice el envío y recepción de información simulando un chat entre dos teléfonos móviles emulados dentro de un entorno de desarrollo específico y siguiendo el esquema de cliente – servidor. El servidor como tal deberá ser ejecutado primero, ya que es el encargado de recibir la información, el cliente buscará el servidor y entonces le enviará información y una vez recibida el servidor responderá al cliente.

5. Temas y subtemas correlacionados con la práctica.

Tema V Programación de dispositivos móviles

5.6. Programación de teléfonos móviles y PDA

El alumno a través del entorno de desarrollo para móviles J2ME comprenderá los principios básicos que involucra la comunicación entre dos teléfonos móviles emulados dentro del entorno de desarrollo, utilizando sockets como medio de comunicación e hilos.

6. Material y equipo necesario:

- Computadora personal
- Compilador de Java.
- Editor de texto

7. Metodología.

Para la elaboración de esta práctica se deberán cumplir los siguientes requisitos:

- a. Haber cubierto el 100% todos los subtema del tema dado en la teoría de la asignatura.

- b. Haber elaborado la investigación solicitada en clase sobre la comunicación empleando sockets en java.
- c. Haber completado la práctica anterior.

. Puntos a desarrollar en la práctica:

1. En Netbeans crear un proyecto llamado Practica14Servidor como aplicación móvil Java ME móvil, deseleccionar la creación del Hello Midlet y la configuración CDLC 1.1 y perfil Mipp 2.0 y como dispositivo un DefaultFXPhone1.
2. Dentro del proyecto crear una nueva clase Midlet llamada ServidorMidlet, modifique la clase creado con el siguiente código:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.ServerSocketConnection;
import javax.microedition.io.SocketConnection;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Gauge;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.ItemCommandListener;
import javax.microedition.lcdui.Spacer;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;

public class ServidorMidlet extends MIDlet implements CommandListener, ItemCommandListener, Runnable {

    private Display display;
    private Form forma;
    private Command cmdSalir;
    private StringItem ipStr;
    private StringItem iniciaServidorStr;
    private Command cmdIniciar;
    private Form formMuestraIp;
    private Gauge gauge;
    private Spacer spacer;
    private ServerSocketConnection servSocketCon;
    private String Ip;
    private SocketConnection socketCon;
    private DataInputStream inputStream;
    private DataOutputStream outputStream;
    private Form formMensajes;
    private TextField cajaTextoMensj;
```

```

private Alert alerta;
private Command cmdEnviar;
private String mensjServidor;
private Enviar enviar;
public void startApp() {
    display = Display.getDisplay(this);
    forma = new Form("Iniciar al Servidor");
    iniciaServidorStr = new StringItem("Para iniciar Servidor elija \t", "Iniciar",
Item.BUTTON);
    forma.append(iniciaServidorStr);
    cmdIniciar = new Command("Iniciar", Command.ITEM, 8);
    iniciaServidorStr.setDefaultCommand(cmdIniciar);
    iniciaServidorStr.setItemCommandListener(this);
    spacer = new Spacer(200, 50);
    forma.append(spacer);
    gauge = new Gauge("Esperando al cliente... \n", false, Gauge.INDEFINITE,
Gauge.CONTINUOUS_RUNNING);
    gauge.setLayout(Item.LAYOUT_CENTER);
    forma.append(gauge);
    cmdSalir = new Command("Salir", Command.EXIT, 7);
    forma.addCommand(cmdSalir);
    forma.setCommandListener(this);
    display.setCurrent(forma);
    formMuestraIp = new Form("Servidor");
    ipStr = new StringItem("El cliente debe conectarse a este IP: \n\n", "Mi IP \n");
    ipStr.setLayout(Item.LAYOUT_CENTER);
    ipStr.setFont(Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD,
Font.SIZE_LARGE));
    formMuestraIp.append(ipStr);
    formMuestraIp.addCommand(cmdSalir);
    formMuestraIp.setCommandListener(this);
    formMensajes = new Form("Servidor manejando informacion...");
    cajaTextoMensj = new TextField("Mensaje a enviar", "", 160, TextField.ANY);
    formMensajes.append(cajaTextoMensj);
    cmdEnviar = new Command("Enviar", Command.OK, 4);
    formMensajes.addCommand(cmdEnviar);
    formMensajes.addCommand(cmdSalir);
    formMensajes.setCommandListener(this);
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
    if(c == cmdSalir){
        notifyDestroyed();
    }else if(c == cmdEnviar){
        mensjServidor = cajaTextoMensj.getString();
        enviar.send(mensjServidor);
    }
}

public void commandAction(Command c, Item item) {
    if(c == cmdIniciar){
        new Thread(this).start();
    }
}
}

```

```

public void run() {
    System.out.println("Ejecutando");
    try {
        servSocketCon = (ServerSocketConnection) Connector.open("socket://:60000");
        System.out.println("Abriendo socket...");
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    try {
        Ip = servSocketCon.getLocalAddress();
        System.out.println("IP: " + Ip);
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    formMuestraIp.append(Ip);
    display.setCurrent(formMuestraIp);

    try {
        System.out.println("Esperando por cliente...");
        socketCon = (SocketConnection) servSocketCon.acceptAndOpen();
        System.out.println("Conexión acepta y abierta...");
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    try {
        inputStream = socketCon.openDataInputStream();
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    try {
        outputStream = socketCon.openDataOutputStream();
    } catch (IOException ex) {
        ex.printStackTrace();
    }

    enviar = new Enviar(outputStream, cajaTextoMensj);
    enviar.start();
    Recibir msgReceiverClass = new Recibir(inputStream, formMensajes);
    msgReceiverClass.start();
    alerta = new Alert("Cliente conectado exitosamente.");
    display.setCurrent(alerta, formMensajes);
}

public void clear(){
    cajaTextoMensj.setString("");
}

public TextField getChatTxtFld() {
    return cajaTextoMensj;
}
}

```

3. Añada una nueva clase y nómbrala Enviar que herede de la clase Thread y que en su constructor reciba una instancia de la clase OutputStream y una

instancia de la clase `TextField` y que tenga un método sincronizado llamado `enviar` que reciba como parámetro un objeto de la clase `String` y llame al método `notify()` y que también su método `run()` sea sincronizado y como primera instrucción despliegue un mensaje que diga “Enviando” y tenga un ciclo infinito para escribir a través de un objeto de `OutputStream` como la clase vista en teoría.

4. Agregue una nueva clase y llámela `Recibir` que también herede de la clase `Thread` y que en su constructor reciba una instancia de la clase `InputStream` y una instancia de la clase `Form` y que su método `run()` sea sincronizado y como primera instrucción despliegue un mensaje que diga “Recibiendo” y tenga un ciclo infinito para recibir los mensajes a través de un objeto de `InputStream` y los coloque en un objeto `StringBuffer`, recuerde que esta clase fue vista en teoría.
5. .Abrir otra instancia de Netbeans y crear un nuevo proyecto llamado `Practica14Clente` como aplicación móvil Java ME sin crear clase `Hello Midlet` y configuración CDLC 1.1 y perfil Mipp 2.0 y como dispositivo un `DefaultCdlcPhone1`
6. Dentro del proyecto crear una nueva clase `Midlet` llamada `ServidorMidlet`, modifique la clase creado con el siguiente código:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.SocketConnection;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;

public class ClienteMidlet extends MIDlet implements CommandListener, Runnable {
    private Display display;
    private Form formIntroducirIp;
    private Command cmdSalir;
    private TextField txtFldIP;
    private Command cmdConectar;
    private Form formMensj;
    private TextField txtFldMensj;
    private Command cmdEnviar;
    private String mensj;
    private Enviar enviar;
    private String ipServ;
    private SocketConnection socketCon;
```

```

private DataInputStream inputStream;
private DataOutputStream outputStream;
private Alert alerta;

public void startApp() {
    display = Display.getDisplay(this);

    formIntroducirIp = new Form("Iniciando Cliente");
    txtFldIP = new TextField("Introduce el IP del servidor \n", "", 24, TextField.ANY);
    formIntroducirIp.append(txtFldIP);

    cmdSalir = new Command("Salir", Command.EXIT, 7);
    formIntroducirIp.addCommand(cmdSalir);
    cmdConectar = new Command("Conectar", Command.OK, 4);
    formIntroducirIp.addCommand(cmdConectar);

    formIntroducirIp.setCommandListener(this);
    display.setCurrent(formIntroducirIp);

    formMensj = new Form("Cliente mensajeando...");
    txtFldMensj = new TextField("Introduce mensaje", "", 160, TextField.ANY);
    formMensj.append(txtFldMensj);

    cmdEnviar = new Command("Enviar", Command.OK, 4);
    formMensj.addCommand(cmdEnviar);
    formMensj.addCommand(cmdSalir);
    formMensj.setCommandListener(this);
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
    if(c == cmdSalir){
        notifyDestroyed();
    }else if(c == cmdConectar){
        new Thread(this).start();
    }else if(c == cmdEnviar){
        mensj = txtFldMensj.getString();
        enviar.send(mensj);
    }
}

public void run() {
    ipServ = txtFldIP.getString();

    if (ipServ != null && ipServ.length() > 0) {
        try {
            socketCon = (SocketConnection) Connector.open("socket://" + ipServ +
":60000");

            inputStream = socketCon.openDataInputStream();
            outputStream = socketCon.openDataOutputStream();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

```

```

        enviar = new Enviar(outputStream, txtFldMensj);
        enviar.start();
        Recibir msgRecClass = new Recibir(inputStream, formMensj);
        msgRecClass.start();
        alerta = new Alert("Servidor conectado exitosamente.");
        display.setCurrent(alerta, formMensj);
    }
}
}

```

7. Al igual que en el proyecto anterior añada una nueva clase y nómbrala **Enviar** que herede de la clase **Thread** y que en su constructor reciba una instancia de la clase **OutputStream** y una instancia de la clase **TextField** y que tenga un método sincronizado llamado **enviar** que reciba como parámetro un objeto de la clase **String** y llame al método **notify()** y que también su método **run()** sea sincronizado y como primera instrucción despliegue un mensaje que diga “Enviando” y tenga un ciclo infinito para escribir a través de un objeto de **OutputStream** como la clase vista en teoría.
8. También agregue una nueva clase y llámela **Recibir** que también herede de la clase **Thread** y que en su constructor reciba una instancia de la clase **InputStream** y una instancia de la clase **Form** y que su método **run()** sea sincronizado y como primera instrucción despliegue un mensaje que diga “Recibiendo” y tenga un ciclo infinito para recibir los mensajes a través de un objeto de **InputStream** y los coloque en un objeto **StringBuffer**, recuerde que esta clase fue vista en teoría.
9. Ejecute el proyecto **Practica14Servidor** y después ejecute el proyecto **Práctica14Cliente** y muestre los resultados.

8. Sugerencias didácticas.

Asegurarse que el alumno utiliza las fuentes de consulta sugeridas en el programa de estudio para el desarrollo de la práctica.

Propiciar actividades de búsqueda, selección y análisis de información en distintas fuentes de consulta aparte de las sugeridas.

Propiciar que por sí mismo el estudiante vea y entienda la relación de los contenidos de temáticos con el desarrollo de la práctica.

Fomentar la participación del estudiante utilizando la discusión del desarrollo de la práctica que propicie el uso adecuado de conceptos, y de terminología de programación, para el desarrollo de sus competencias interpersonales

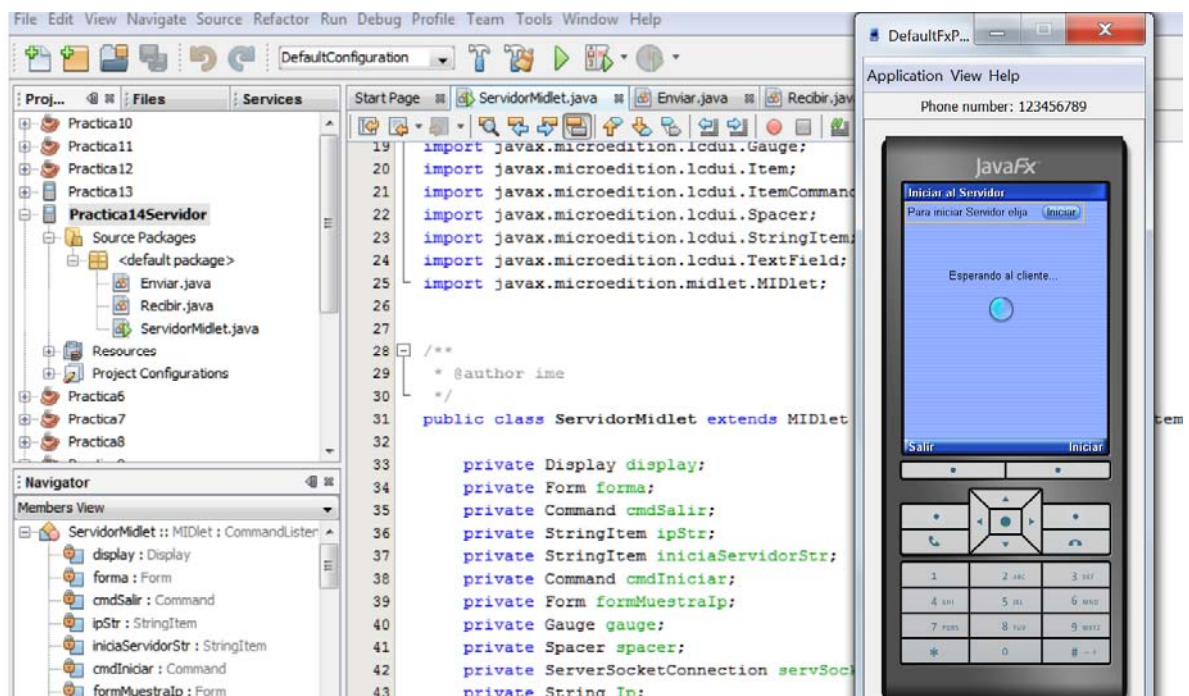
Fomentar el trabajo en equipo para contrarrestar los resultados obtenidos por cada uno de los elementos de cada equipo y realizar su análisis en forma grupal, para el desarrollo de sus competencias instrumentales, interpersonales y sistémicas.

9. Reporte del alumno. (Resultados)

Resultados de la Práctica de acuerdo a lo indicado.

1. Se creó un proyecto llamado Practica14Servidor como aplicación móvil Java ME móvil, seleccionar la creación del Hello Midlet y la configuración CDLC 1.1 y perfil Mipp 2.0 y como dispositivo un DefaultFXPhone1.
2. Dentro del proyecto se creó una nueva clase Midlet llamada ServidorMidlet, y se modificó conforme se solicitó:
3. Se añadió una nueva clase y nombrada Enviar y que herede de clase Thread como la clase vista en teoría.
4. Se añadió una nueva clase y llámela Recibir que también herede de la clase Thread como la vista en teoría.
5. Se abrió otra instancia de Netbeans y se creó un nuevo proyecto llamado Practica14Cliente como aplicación móvil Java ME sin crear clase Hello Midlet y configuración CDLC 1.1 y perfil Mipp 2.0 y como dispositivo un DefaultCdlcPhone1
6. Dentro del proyecto se creó una nueva clase Midlet llamada ServidorMidlet, con el código proporcionado.
7. Al igual que en el proyecto anterior se anexó una nueva clase Enviar.
8. Y también agregue una nueva clase Recibir.
9. Ejecute el proyecto Practica14Servidor y después ejecute el proyecto Práctica14Cliente y a continuación se muestran los resultados.

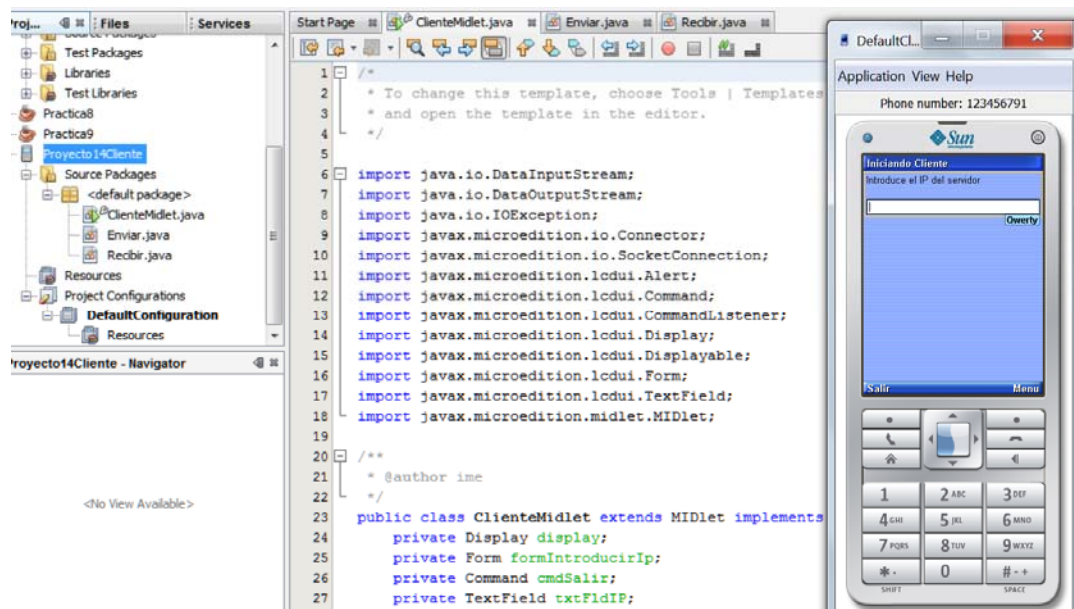
La siguiente figura muestra el inicio del servidor al ser ejecutado



Y al hacer clic sobre el botón Iniciar se observa la dirección Ip del dispositivo servidor:



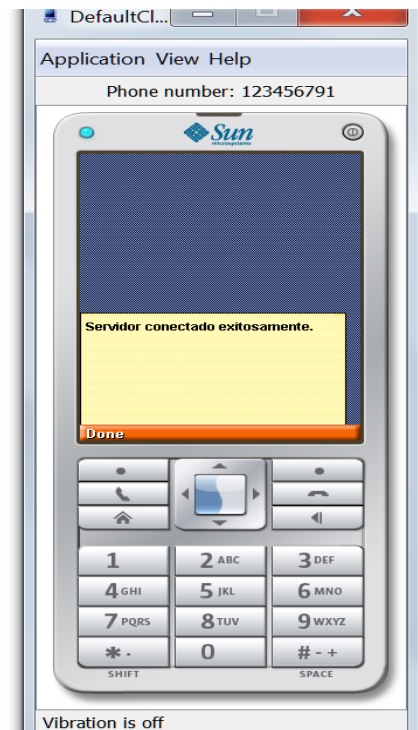
La siguiente figura muestra la ejecución del proyecto Practica14Cliente:



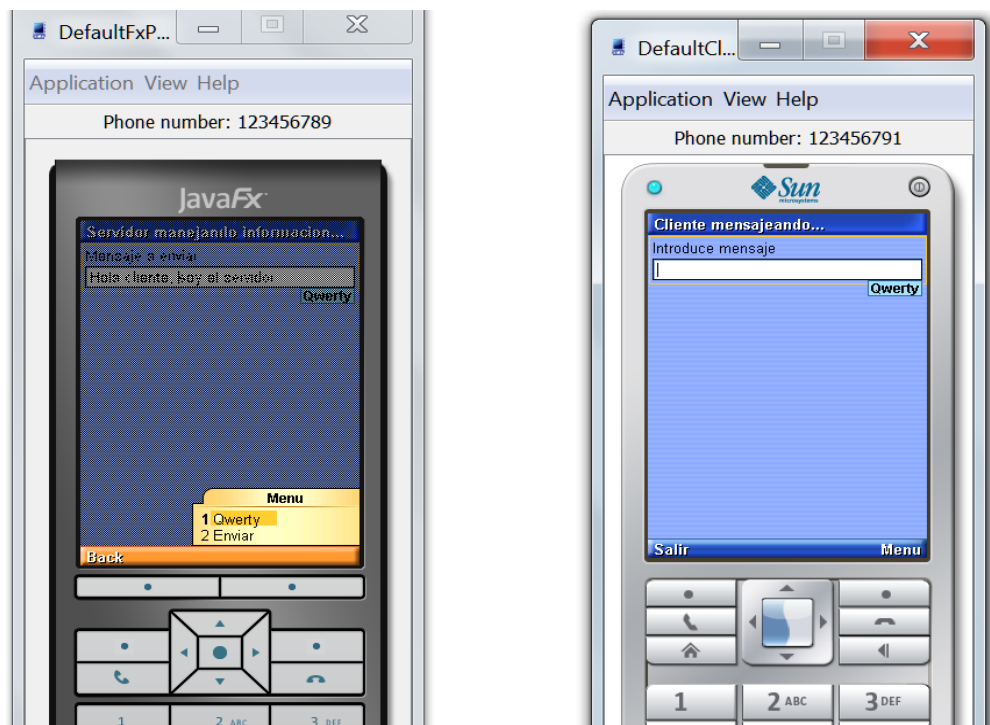
En la cual se solicita introducir la dirección Ip del servidor una vez introducida se solicita conectar



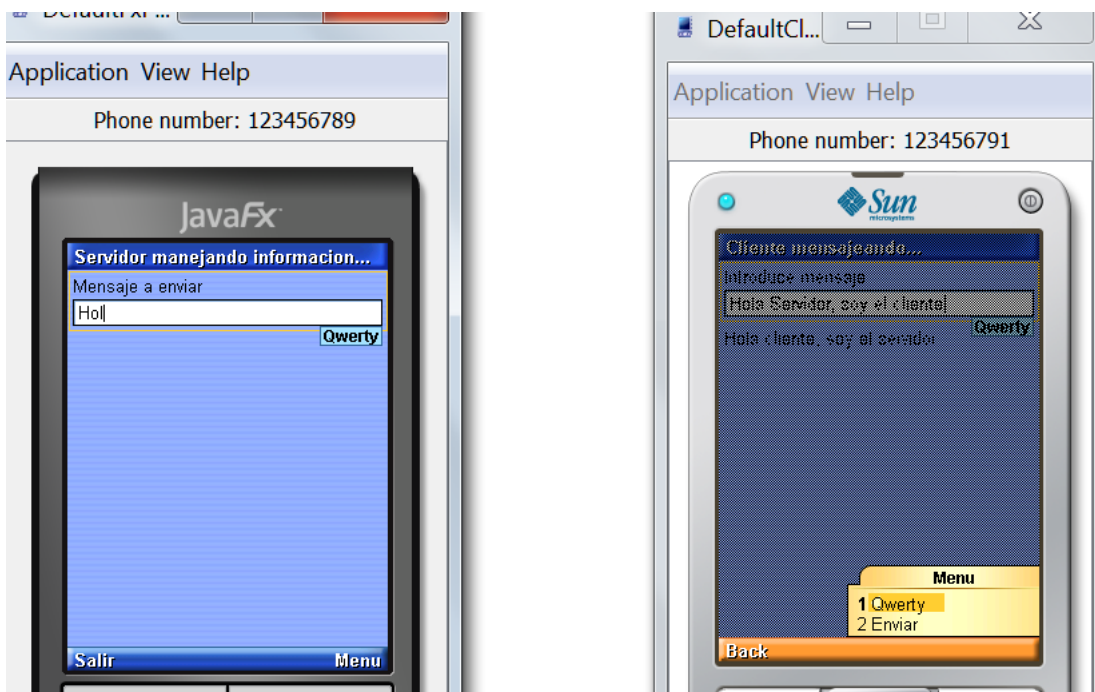
Una vez establecida la conexión, ambos dispositivos muestran mensajes de conexión exitosa:



Y están listos para enviar y recibir información, la siguiente figura muestra como el servidor se comunica con el cliente:



Y la siguiente como el mensaje ha sido recibido por el cliente, y está a punto de enviar la respuesta:



Y la última figura muestra la recepción del mensaje por parte del servidor.



10. Bibliografía preliminar.

Deitel Y Deitel, Java Cómo Programar, 7ª Edición, Prentice Hall, México, 2008

John Dean, Introducción A La Programación Con Java, 1, Mcgraw Hill, México, 2009

Herbert Schildt, Java Manual De Referencia 7a. Ed., 7ª Edición, Mcgraw Hill, México, 2008