

PRÁCTICA 7.- Creación de Componentes

Objetivo

Crear un nuevo componente de java empezando desde cero y crear un nuevo componente a partir de uno ya existente apoyándose en la herencia.

Objetivos específicos:

- Crear un componente nuevo usando herencia y mostrar su uso en una aplicación.
- Crear un componente nuevo desde cero y mostrar su uso en una aplicación.

Introducción

El desarrollo actual de aplicaciones se ha visto apoyado de manera importante con el uso de las interfaces gráficas de desarrollo (IDE) que permiten incorporar componentes a formularios, modificar sus propiedades de manera rápida y definir a qué eventos responder y cómo debe ser la respuesta. Esto hace que dichas aplicaciones resulten relativamente fáciles de usar y agradables a la vista.

Con el desarrollo de esta práctica, se dotará al alumno de una herramienta extremadamente importante, ya que le permitirá diseñar sus propios componentes desde cero e incorporarlos a la paleta de componentes o bien, si un componente actual requiere un comportamiento diferente al que tiene programado o le falta alguna característica, es posible conservar todo lo bueno que hace y solo modificar o añadir lo necesario. Este nuevo componente generado, también es posible incorporarlo a la paleta de componentes y dejarlo disponible de manera permanente para cualquier otra aplicación en el futuro.

Se presenta la construcción de tres componentes buscando mostrar tres de los posibles casos a la hora de crear un componente. Como ya se mencionó arriba, puede ser solo añadir alguna propiedad o comportamiento que le falte y por lo tanto el nuevo componente generado hereda todo del componente original; puede ser que se requiera cambiar algo de lo que ya tiene; el nuevo componente puede ser el resultado de agrupar varios componentes individuales y entonces se definen características y funcionamiento para el conjunto; otra posibilidad es la de crear un componente definiendo sus propiedades y comportamientos desde cero.

Correlación con los temas y subtemas del programa de estudio vigente

Se relaciona directamente con los temas de la Unidad II ya que se hace uso de la interface gráfica de desarrollo y se generan aplicaciones gráficas.

Con la Unidad III tiene una gran relación dado que se utilizan componentes que ya existen en el entorno de desarrollo, se crean nuevos componentes y adicionalmente, se utilizan librerías ya definidas en el ambiente. Específicamente con los puntos 3.2 y 3.4 del temario.

Material y equipo necesario

- Computadora o laptop.
- Software requerido: NetBeans 7.2 en adelante.

Metodología

Parte Uno Desarrollo de un Componente integrando varios ya existentes.

Se desarrolla un componente integrando componentes ya existentes sin modificar las propiedades de ninguno de ellos y solamente programando los eventos que nos interesa controlar de uno o algunos de todos.

Se creará un nuevo componente que integra un *JLabel* para la salida del resultado, tres *JLabel* de identificación y tres cajas de texto. Este componente permitirá que al teclear cualquier caracter en cualquiera de las tres cajas de texto, se vea escrito inmediatamente en la etiqueta primera *Name*.

El nuevo componente tendrá esta imagen:

Paso 1. Abra un nuevo proyecto con nombre *Practica7_ ComponenteNombre*. Cuidado que la casilla de verificación del Create Main Class esté desmarcada.

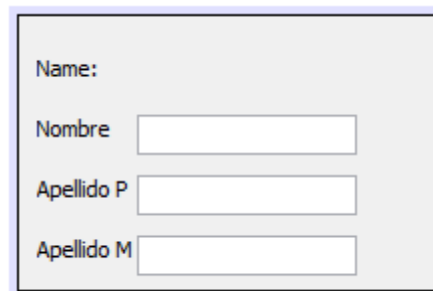
Paso 2. Añada un nuevo *JPanel Form* con nombre *MiNombre*.

Paso 3. Seguramente le marca error en la declaración de la clase. Para corregir este error lleva el cursor al pequeño globo rojo en el borde izquierdo del código, haga clic sobre él y acepte la sugerencia de implementar todos los métodos abstractos de la interface Customizer que incorporamos.

Una vez hecho lo anterior, modifique la declaración de la clase con lo siguiente:

```
public class MiNombre extends javax.swing.JPanel implements java.beans.Customizer {  
    private Object bean;  
    /** Creates new customizer MiNombre */  
    public MiNombre() {  
        initComponents();  
    }  
    public void setObject(Object bean) {  
        this.bean = bean;  
    }  
}
```

Paso 4. Muestre el diseño de la forma haciendo clic en el botón *design* del IDE. Enseguida añada los componentes individuales para construir el componente nuevo tal como se aprecia en la imagen siguiente:

A screenshot of a Java Swing window titled "Name:". Inside the window, there are three text input fields. The first field is labeled "Nombre", the second is labeled "Apellido P", and the third is labeled "Apellido M". The labels are in a dark blue font, and the input fields are white with a light gray border.

Como ya fue mencionado se incorporan 4 etiquetas y tres cajas de texto. Los nombres para estos componentes son:

jLabel1...jlbName

jLabel1...jlbNombre

jLabel1...jlbApePat

jLabel1...jlbApeMat

jTextField1...jtfNombre

jTextField1...jtfApePat

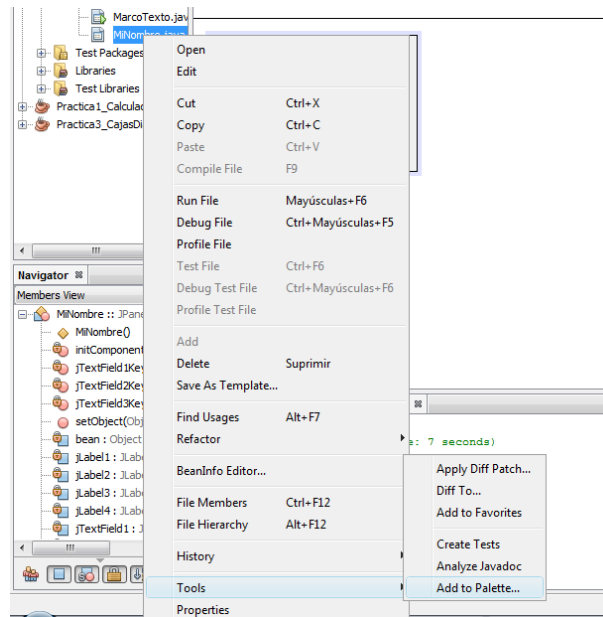
jTextField1...jtfApeMat

Paso 5. Se programarán los eventos `KeyTyped` de las tres cajas de texto. A estas alturas del curso ya debe saber realizar este paso. El código para estos eventos es el siguiente:

```
private void jtfNombreKeyTyped(java.awt.event.KeyEvent evt) {  
    jlName.setText("Name: " + jtfNombre.getText() + " " + jtfApePat.getText() + " "  
+ jtfApeMat.getText());  
}  
  
private void jtfApePatKeyTyped(java.awt.event.KeyEvent evt) {  
    jlName.setText("Name: " + jtfNombre.getText() + " " + jtfApePat.getText() + " "  
+ jtfApeMat.getText());  
}  
  
private void jtfApeMatKeyTyped(java.awt.event.KeyEvent evt) {  
    jlName.setText("Name: " + jtfNombre.getText() + " " + jtfApePat.getText() + " "  
+ jtfApeMat.getText());  
}
```

Paso 6. Este paso es bien importante para poder continuar. DEBE compilarse nuestro componente. Una manera de hacerlo es haciendo clic en el botón de la barra de herramientas que tiene un martillo y una escoba. No debe presentar errores de compilación.

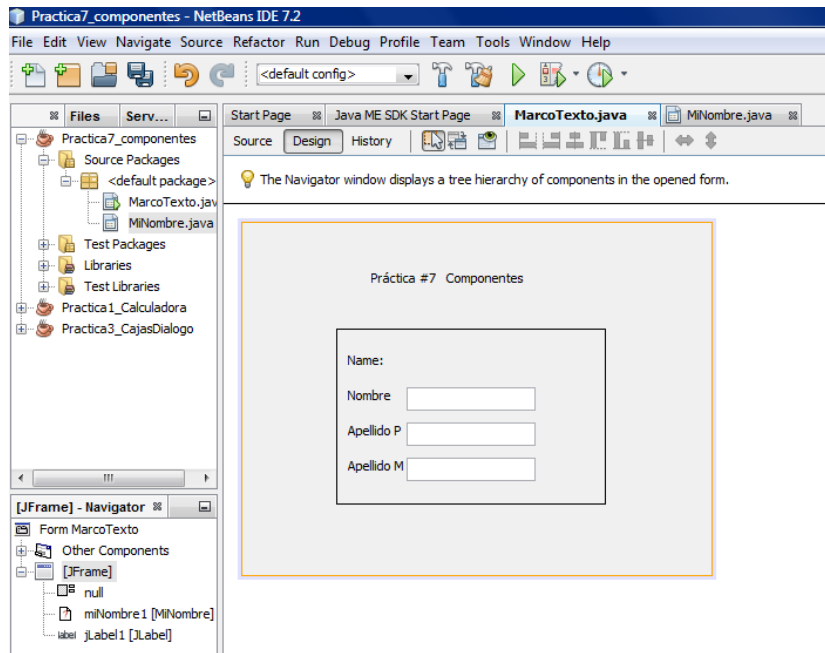
Paso 7. Vamos a añadir nuestro componente a la paleta de componentes. Para realizar esta tarea vamos a la ventana de proyectos, seleccionamos *MiNombre.java* y con botón derecho abrimos el menú emergente. En la parte inferior de la ventana que se abrió, está *Tools* y se selecciona. Se abre una nueva ventana y escogemos *Add to Palette*. Si aparece una nueva ventana, debemos escoger en cuál categoría queremos que aparezca nuestro componente creado. Puede escoger Beans, aunque también es posible crear categorías nuevas con el hecho de dar clic con botón derecho en la paleta de componentes y se muestra la opción de *New Category*.



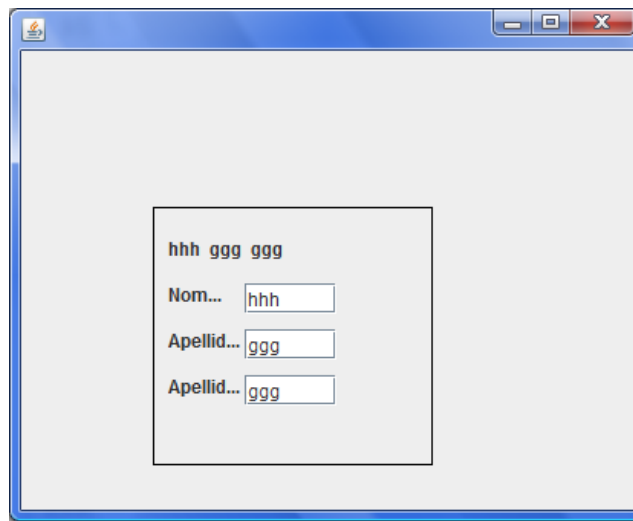
Paso 8. A continuación revisamos si nuestro componente se añadió correctamente a la paleta de componentes. Revisamos la categoría seleccionada para que lo contenga. Si no aparece, es necesario refrescar la paleta. Este refresco de paleta se logra con botón derecho sobre ella y buscar la opción *Refresh Palette*. De preferencia cuide que el cursor esté sobre una línea con el nombre de alguna de las categorías para que el menú contextual corresponda al de la paleta.

Paso 9. Es momento de probar nuestro componente, para ello añadimos un *JFrame Form* al proyecto asignándole por nombre *MarcoTexto*.

Paso 10. Sobre nuestro formulario *MarcoTexto* poner un componente *MiNombre* tomado de la paleta de componentes. La imagen debe parecerse a la siguiente:



Se presenta una pantalla durante la ejecución del programa anterior:



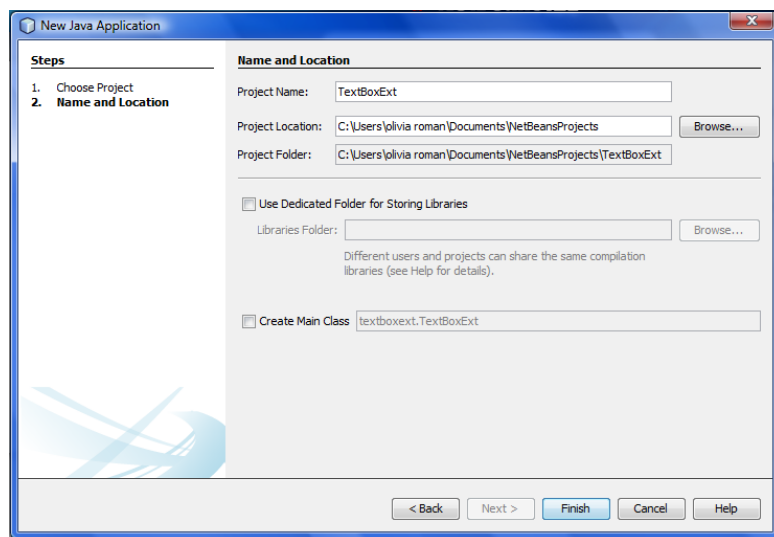
Observe que los caracteres que aparecen en las cajas de texto se encuentran integrados en el texto de la etiqueta superior. Cada vez que cualquiera de las cajas de texto se modifica, el texto también lo hace.

Parte Dos. Componente nuevo adaptando a un componente existente

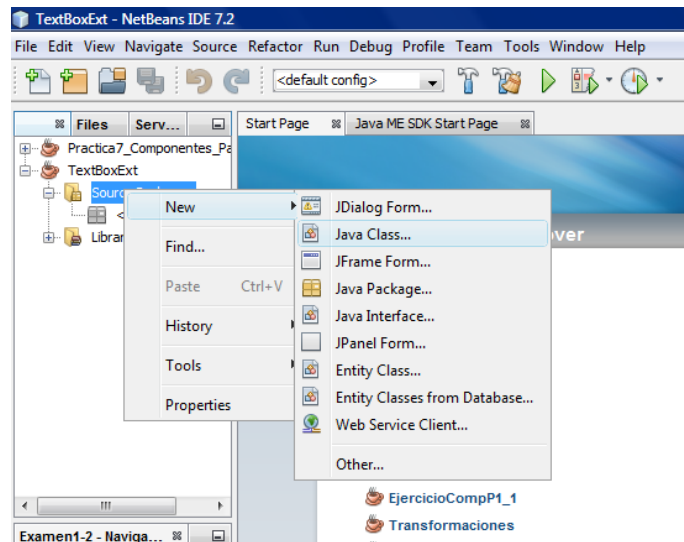
A continuación procederemos a ver cómo modificar el comportamiento de un componente ya existente, adecuando aquellas propiedades o comportamientos conforme a las necesidades en la aplicación, o bien, añadiendo características y comportamientos adicionales, pero conservando todo lo demás en su estado normal.

Para esta parte, haremos que una caja de texto que todos conocemos, cambie su color al recibir el foco, de manera que al estar varias cajas de texto presentes, se pueda identificar fácilmente la que está seleccionada. Otro comportamiento que se puede programar es que cuando reciba un valor numérico negativo, lo detecte y lo ponga de color rojo. Una manera de indicar que hay saldo rojo. Este segundo comportamiento para el nuevo componente se dejará al alumno.

Paso 1. Abrimos nuevo proyecto con nombre TextBoxExt, desmarcamos la casilla del Create Main Class.



Paso 2. Añadimos una *Java Class* al proyecto con nombre *TextBoxEx*.



El código que se añade automáticamente es el siguiente:

```
public class TextBoxEx {  
  
}
```

Paso 3. Modificar la declaración de la clase para que herede del componente *JTextField*. En este momento, TODOS los objetos que se crean usando la clase nueva *TextBoxEx*, TENDRÁN exactamente el mismo comportamiento y diseño que los objetos producidos con la clase *JTextField*. Ahora bien, la clase nueva *TextBoxEx*, a ésta SI la podemos modificar.

```
public class TextBoxEx extends javax.swing.JTextField {  
  
}
```

Paso 4. Se crea el constructor: dentro de las dos llaves de la clase, se abre menú contextual con botón derecho, seleccionar insertar código (*Insert Code*), luego en la siguiente ventana escoger constructor, se añade automáticamente el código siguiente:

```
public TextBoxEx() {  
  
}
```


Paso 5. Llamar al constructor de la clase padre escribiendo `super()` dentro del constructor

```
public TextBoxEx() {  
    super();  
}
```

Paso 6. Debemos tener presente que queremos que la caja de texto tenga un comportamiento diferente cuando se activa, por lo que será necesario redefinir los eventos que nos interesen en cada caso en particular y para este componente se trata de los eventos de obtener el foco y perder el foco, *FocusGained* y *FocusLost*, ya que cuando se gana el foco debe tomar un color diferente y cuando lo pierde debe regresar a su color original. La manera en que lo haremos será declarando el Listener (escuchador) y redefiniendo los dos eventos mencionados. Todo este código lo vamos a teclear en esta ocasión.

Declaramos el escuchador tipo *FocusAdapter* dentro del constructor de la clase así como los dos eventos que nos interesan: *focusGained* y *focusLost*

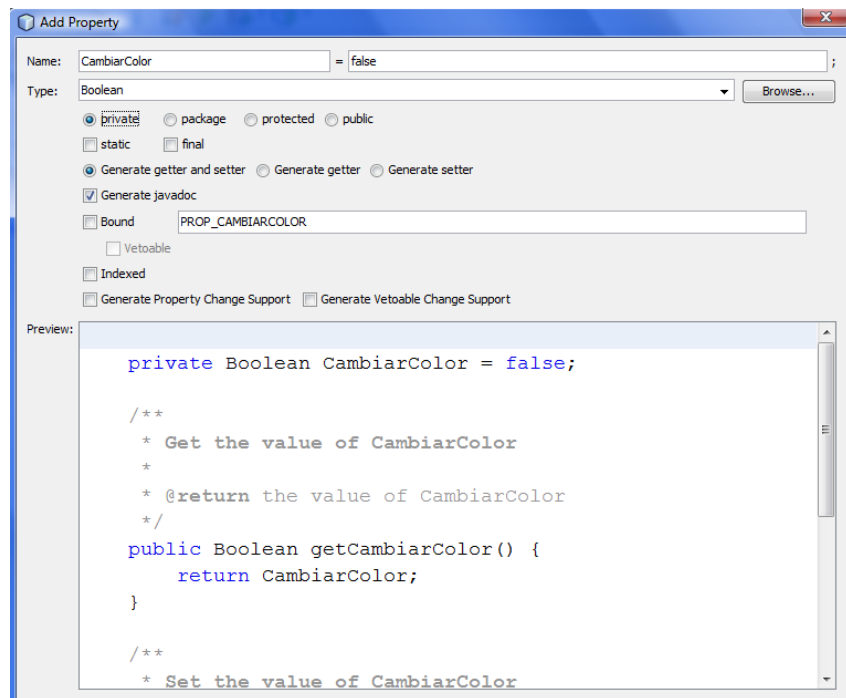
```
public TextBoxEx() {  
    super();  
    java.awt.event.FocusAdapter fl = new java.awt.event.FocusAdapter() {  
        public void focusGained(java.awt.event.FocusEvent evt){  
            jtfTextBoxExFocusGained(evt);  
        }  
        public void focusLost(java.awt.event.FocusEvent evt){  
            jtfTextBoxExFocusLost(evt);  
        }  
    };  
    addFocusListener(fl);  
}
```

Los dos métodos dentro de los eventos aparecerán marcados como errores ya que aún no han sido declarados. Ahí es donde haremos que la caja de texto modifique su comportamiento que ha heredado de la caja de texto original.

Es muy importante escribir la instrucción para añadir el escuchador como parte de la clase con la sentencia `addFocusListener(fl)`.

Paso 7. Vamos a declarar una nueva propiedad que permita al usuario decidir si quiere que la caja cambie de color o no. Esta propiedad será de tipo boolean y la llamaremos *CambiarColor*.

Con el menú contextual escogemos Insertar Código y enseguida, seleccionamos añadir propiedad. Esto nos llevará a una nueva pantalla como la que se muestra.



Es importante hacer clic en el botón *private* (en este caso es el especificador que necesitamos) aunque se vea seleccionado, para hacer que se activen los cambios y podamos ver en la ventana *preview* el código tal y cómo se añadirá.

Se añadió el siguiente código

```
private Boolean CambiarColor = false;

public Boolean getCambiarColor() {
    return CambiarColor;
}

public void setCambiarColor(Boolean CambiarColor) {
    this.CambiarColor = CambiarColor;
}
```

Paso 8. Procedemos a declarar los dos métodos de respuesta a los eventos de foco que tenemos pendientes. Escribimos fuera del constructor lo siguiente:

```
private void jtfTextBoxExFocusGained(java.awt.event.FocusEvent evt){  
    if(getCambiarColor())  
        setBackground(Color.pink);  
    else  
        setBackground(Color.WHITE);  
}  
private void jtfTextBoxExFocusLost(java.awt.event.FocusEvent evt){  
    setBackground(Color.white);  
}
```

Paso 9. Si por alguna razón cuando escribimos código o bien pegamos código hay algún elemento que no es reconocido, por lo general subrayado en color rojo, se puede intentar su corrección tomando del menú contextual la opción *Fix Imports* (arreglar importaciones). Puede haber sucedido con la palabra `Color` dentro del `setBackground(Color.pink)`. Tal como tenemos el código NO ESTÁ BIEN ya que si la caja de texto tuviera un color de fondo diferente del blanco, después de haber sido seleccionada y pintada de rojo, el método la pintará de blanco y no del color que tenía. Para corregir esto es conveniente declarar una variable de tipo `Color` para que se guarde en ella el color antes de cambiarlo y así poder asignarlo después de que la caja pierde el foco. Añada esta línea antes del constructor de la clase.

```
private java.awt.Color colorOriginal = Color.white;  
// inicializada en blanco
```

Modificamos los métodos `jtfTextBoxExFocusGained()` y `jtfTextBoxExFocusLost()` de la siguiente manera, haciendo uso de la nueva variable `colorOriginal`:

```
private void jtfTextBoxExFocusGained(java.awt.event.FocusEvent evt){  
    if(getCambiarColor()){  
        colorOriginal = getBackground();  
        setBackground(Color.pink);  
    }  
    else  
        setBackground(colorOriginal);  
}
```

```

    }

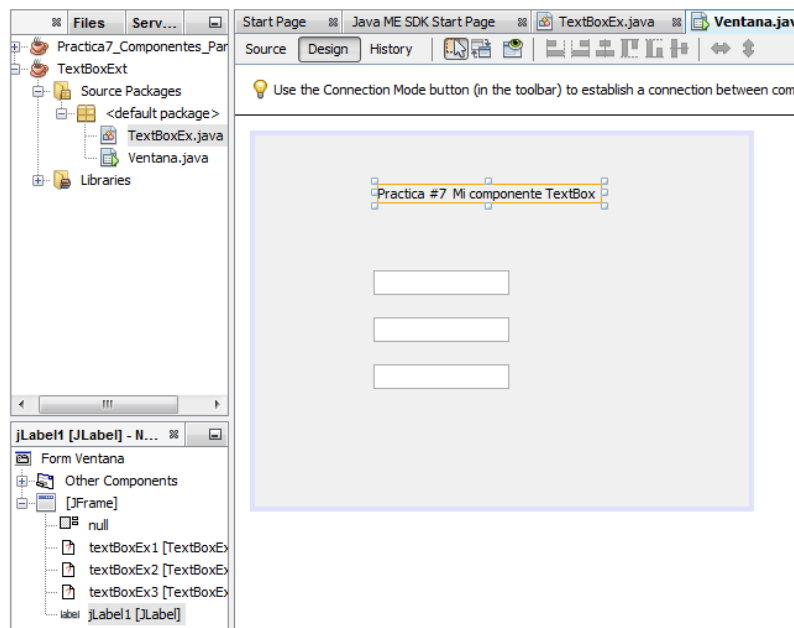
    private void jtfTextBoxExFocusLost(java.awt.event.FocusEvent evt){
        setBackground(colorOriginal);
    }
}

```

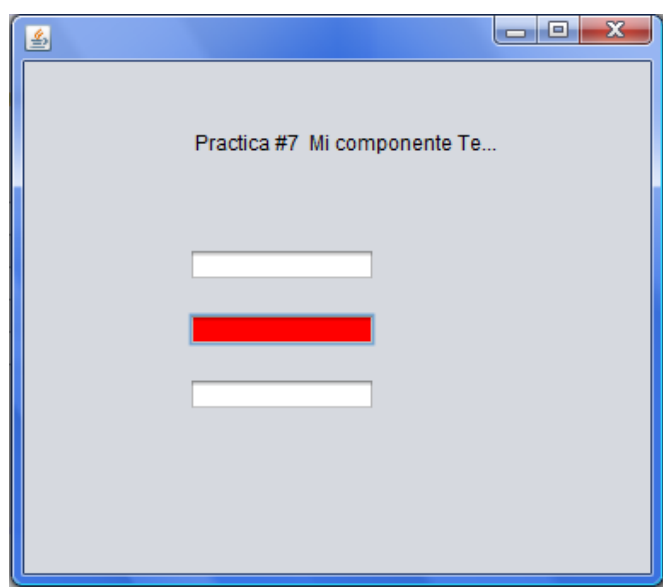
Paso 10. Compilamos nuestro componente haciendo clic en el botón con la imagen del martillo y la escoba en la barra de herramientas. Enseguida lo incorporamos a la paleta de componentes tal y como se realizó en la parte anterior. Ir a la ventana proyectos y sobre el nombre de la clase *TextBoxEx.java*, con el botón derecho activar el menú contextual correspondiente y seleccionar *Tools* y enseguida *addPalette*. Decidir una de las categorías existentes (Beans) o crear una categoría nueva *MisComponentes* y aceptar.

Paso 11. Vamos a probar el nuevo componente creado *TextBoxEx*, así que regresamos a la ventana proyectos y sobre la palabra *Source Packages*, activar menú contextual y seleccionar *new*, luego en la nueva ventana seleccionamos *JFrame Form* y aceptamos.

Sobre nuestra nueva ventana YA podemos probar nuestro componente creado el cual debe aparecer en la paleta de componentes. Añadimos tres *TextBoxEx* sobre nuestra forma. Si no aparece realice un *Refresh Palette* con botón derecho sobre la ventana de componentes.

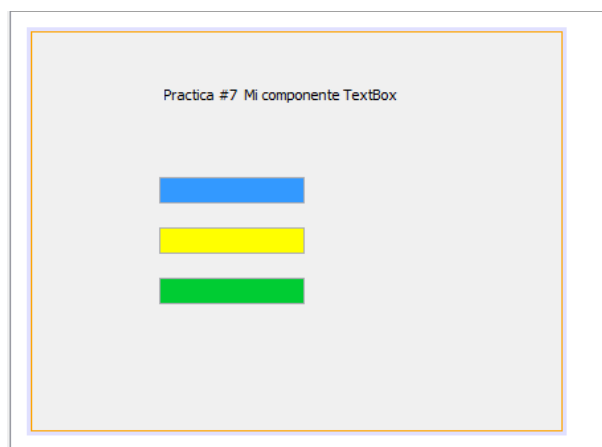


Si se ejecuta el programa en este momento, las cajas de texto no cambiarán de color al ser enfocadas debido a que la propiedad *cambiarColor* tiene valor *false* por defecto. Es necesario seleccionar cada una de las cajas e ir a la ventana de propiedades y cambiar el valor a *true* de la propiedad mencionada. Una vez hecho lo anterior podrá verse el comportamiento deseado.

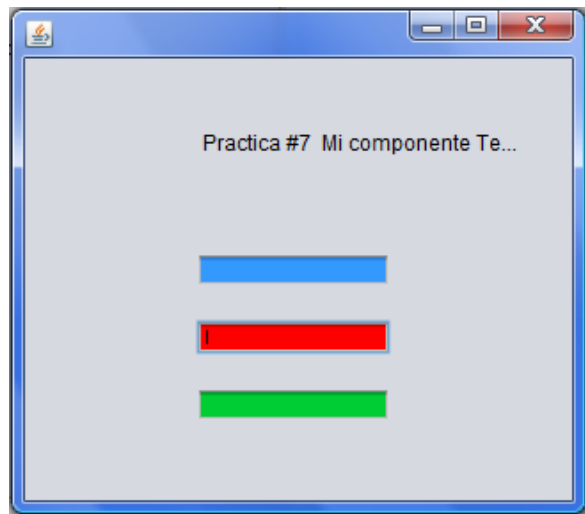
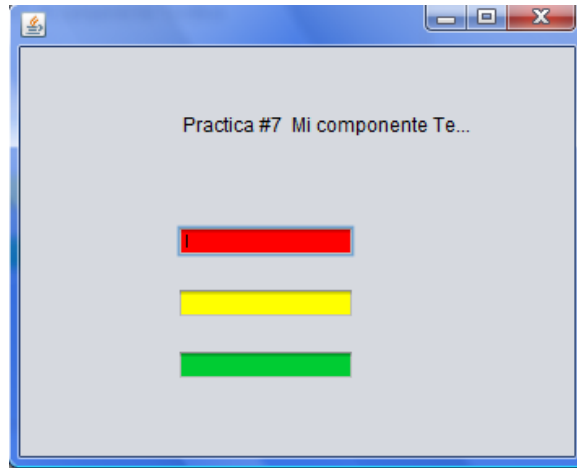


Paso 12. En caso de que las cajas tuvieran un color diferente al blanco, al perder el foco regresarían a su color original.

Cambie los colores (propiedad *background*) de las tres cajas de texto solo para probar que restablece el color original.



Una vez que se ejecuta el programa, la primera caja al quedar seleccionada automáticamente, su color es el rojo. Cuando pierde el foco, regresa al azul y la que tiene el foco ahora es la que presenta el color rojo.

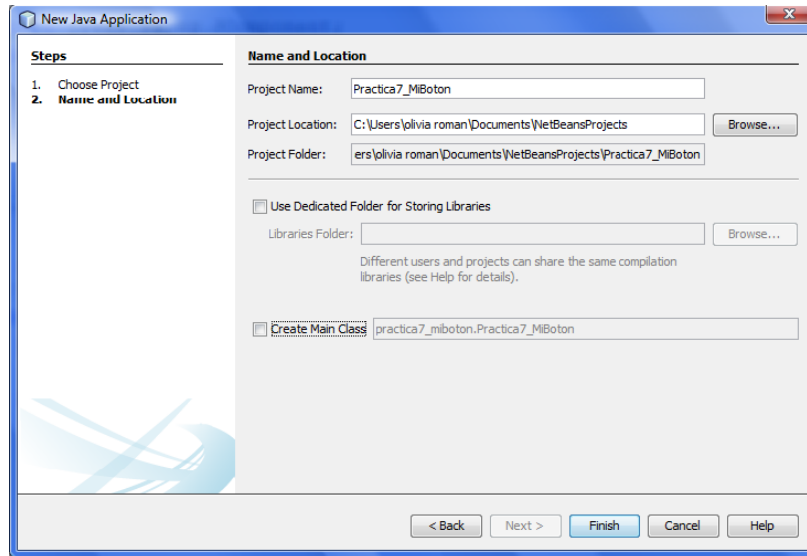


Parte Tres. Componentes creados desde cero

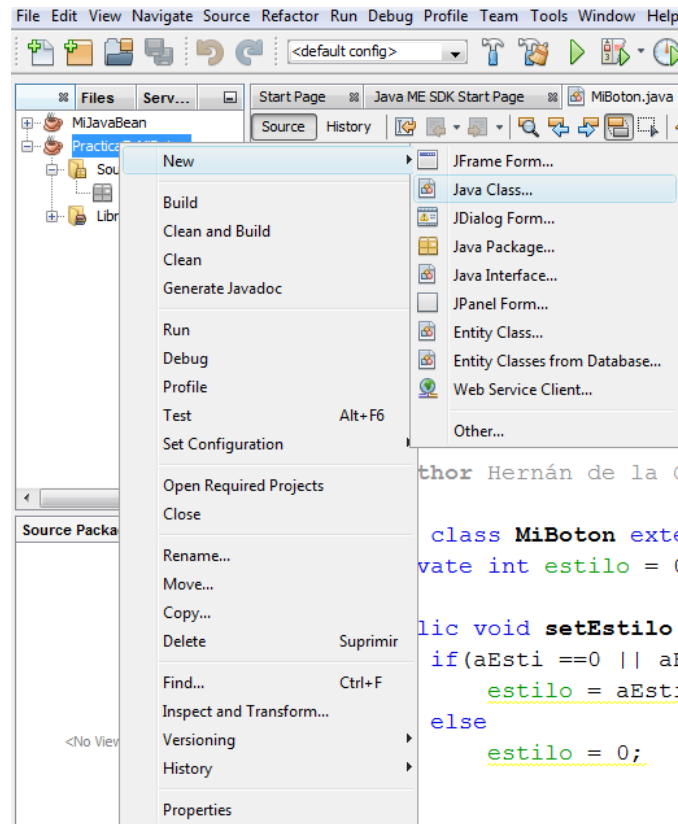
Se procederá a presentar la creación de un componente heredando directamente de la clase *JComponent* que es la clase de la que se derivan todos los componentes de la paleta de componentes.

Se diseñará un botón 3D que permita diferenciar cuando está presionado de cuando no lo está. Para ello se deberán dibujar las líneas que lo componen para cada caso. Se recurre a una propiedad de nombre estilo que define el cómo deberá pintarse. Dicha variable toma valores de cero o uno.

Paso 1. Se empieza un nuevo proyecto con nombre *Practica7_Boton3D* cuidando de desmarcar el *checkbox* de la función *main*.



Paso 2. En la ventana de proyectos, activar el menú contextual sobre la palabra *Source Packages*, escoger *new* y enseguida *Java Class*. Asignar el nombre *MiBoton* a la clase y aceptar.



Después de asignar el nombre y dar clic en el botón Finish, se nos muestra el siguiente código:

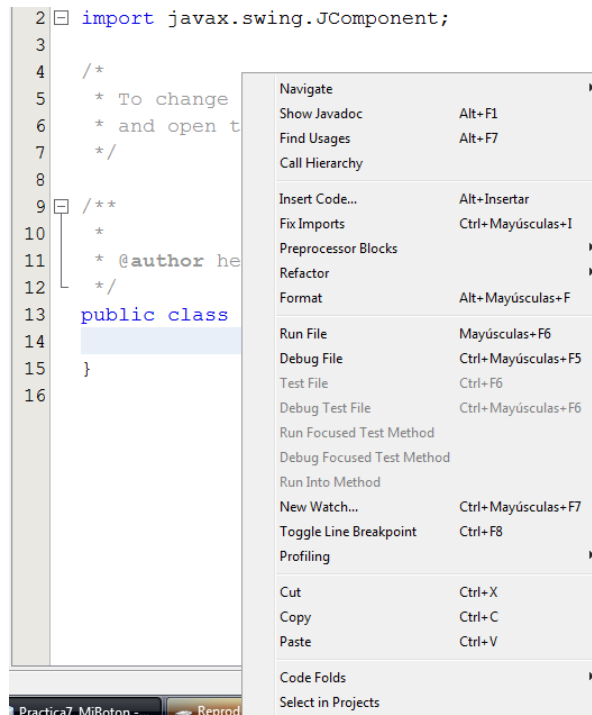
```
public class MiBoton {  
  
}
```

Paso 3. Hacemos que la clase nueva *MiBoton* herede de la clase superior *JComponent* por medio de la palabra reservada *extends*.

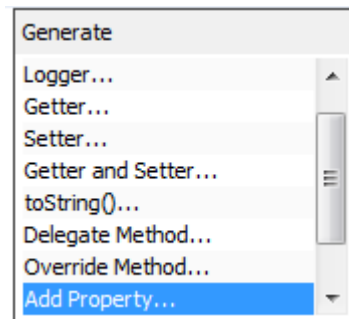
```
public class MiBoton extends JComponent{  
  
}
```

Es posible que la palabra *JComponent* esté subrayada como error, para corregir active menú contextual con el botón derecho y seleccione la opción *Fix Imports*. Con esto deberán añadirse las librerías necesarias.

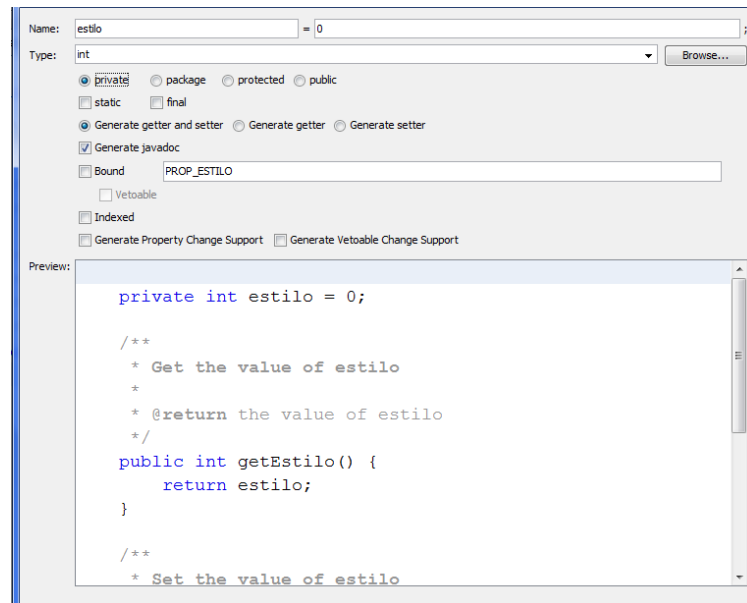
Paso 4. Se creará una nueva propiedad de nombre *estilo*, de tipo *int* y con valor inicial de cero. Para ello dentro del espacio de la clase active el menú contextual nuevamente pero esta vez seleccionamos *Insert Code*.



Se despliega una nueva ventana en la que se selecciona *Add Property*.



Se abre una nueva ventana en la que se deberá dar el nombre a la propiedad (estilo), su tipo de dato (int), su valor inicial (cero) y su ámbito de aplicación (private).



Se habrá añadido mas código referente a la propiedad estilo, sus métodos *setEstilo()* y *getEstilo()*. Dichos métodos serán definidos enseguida.

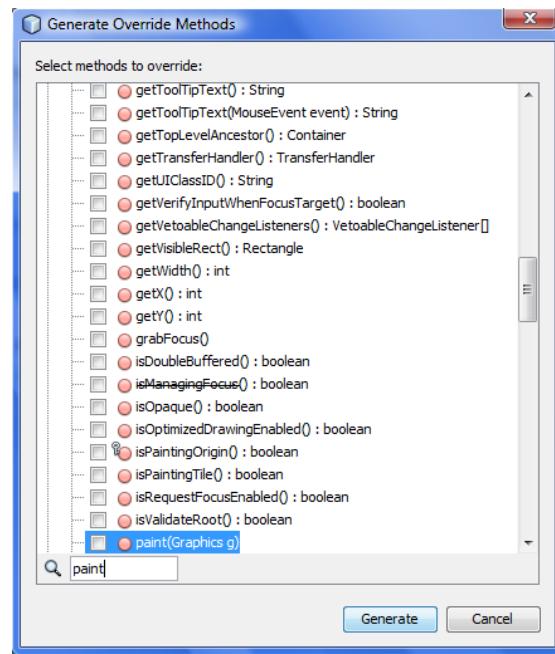
Paso 5. Se escribirá el código correspondiente a los dos métodos de la propiedad estilo. Teclear para que quede de la manera siguiente:

```
public class MiBoton extends JComponent{
    private int estilo = 0;

    public void setEstilo(int aEsti){
        if(aEsti ==0 || aEsti ==1)
            estilo = aEsti;
        else
            estilo = 0;
    }
    public int getEstilo(){
        return estilo;
    }
}
```

Paso 6. Se sobrescribirá el método *paint()* que hereda de la clase *JComponent*, para definir cómo queremos que se vea nuestro componente. El componente nuevo se podrá dibujar de dos maneras diferentes, cada una correspondiendo a un valor de la propiedad estilo.

Para realizar esto, debemos sobrescribir el método *paint()*, por lo que en un espacio arriba de la última llave de la clase, invocar al menú contextual por los medios ya conocidos y seleccionar nuevamente Insert Code. Cuando la nueva ventana se despliega seleccionamos *Override Method*. Nos aparece la ventana siguiente:



Para localizar el método *paint()* de una manera más rápida empiece a teclear la palabra *paint* y se irán mostrando los métodos que empiezan con p, luego los que tienen pa y así sucesivamente. MARCAR la casilla del método *paint()* y clic en *Generate*. El código que se añade se verá como sigue:

```
@Override
public void paint(Graphics g) {
    super.paint(g);
}
```

Paso 7. Procederemos a teclear las instrucciones que permiten dibujar el componente de dos maneras diferentes dependiendo del valor de la propiedad estilo. El código debe quedar como se muestra:

```
public void paint(Graphics g){
    int a = getWidth()-1;
    int h = getHeight()-1;
```

```

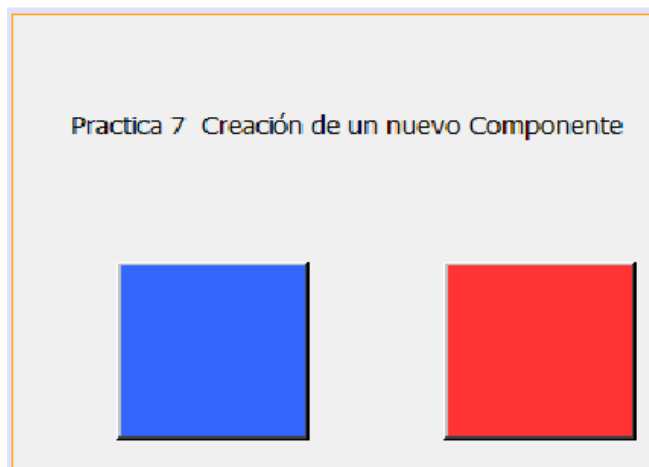
super.paint(g);
g.fillRect(0, 0, a, h);
Color auxi = g.getColor();
if(getEstilo()==0){
    g.setColor(Color.BLACK);
    g.drawLine(0, h, a, h);
    g.drawLine(a, h, a, 0);
    g.setColor(Color.DARK_GRAY);
    g.drawLine(0, h-1, a-1, h-1);
    g.drawLine(a-1, h-1, a-1, 0);
    g.setColor(Color.WHITE);
    g.drawLine(0, 0, 0, h-1);
    g.drawLine(0, 0, a-1, 0);
    g.setColor(Color.lightGray);
    g.drawLine(1, 1, 1, h-2);
    g.drawLine(1, 1, a-2, 1);
}
else {
    g.setColor(Color.white);
    g.drawLine(0, h, a, h);
    g.drawLine(a, h, a, 0);
    g.setColor(Color.LIGHT_GRAY);
    g.drawLine(0, h-1, a-1, h-1);
    g.drawLine(a-1, h-1, a-1, 0);
    g.setColor(Color.BLACK);
    g.drawLine(0, 0, 0, h-1);
    g.drawLine(0, 0, h-1, 0);
    g.setColor(Color.DARK_GRAY);
    g.drawLine(1, 1, 1, h-2);
    g.drawLine(1, 1, a-2, 1);
}
g.setColor(auxi);
}

```

Considere que el componente tendrá el tamaño que el usuario decida al momento de ponerlo en un formulario, por lo que entonces el tamaño no es un valor constante. Para resolver esto se pregunta el ancho y alto del componente por medio de los métodos *getWidth()* y *getHeight()*. Los valores se guardan en las variables *a* y *h*. Así entonces tenemos las dimensiones del componente y ya podemos trazar las líneas que lo definen considerando las cuatro esquinas del componente. Esquinas superiores (0,0) y (a,0); esquinas inferiores (0,h) y (a,h).

Paso 8. Se compila el componente y se añade a la paleta de componentes de la misma manera a como ya se ha trabajado en los casos anteriores de esta práctica.

Paso 9. Añadir un *JFrame Form* al proyecto de la manera como se ha indicado anteriormente. Poner una etiqueta y adecuar el texto a como se muestra en la siguiente figura, y añadir dos componentes *MiBoton* sobre el formulario. Cambie el color de la propiedad foreground diferente del negro para que se aprecien los cambios en el estilo.



Antes de ejecutar el programa se observan las líneas negras en la parte de abajo y en el costado de la derecha. Si cambia la propiedad estilo a valor de uno, podrá ver que se mueven a la parte de arriba y costado izquierdo.

El cambio en el estilo se hace EN TIEMPO DE DISEÑO, antes de ejecutarse, cambiando el valor de la propiedad estilo. Como ejercicio deberá programarse el evento *mouseClicked* para que con cada clic sobre el componente cambie el valor de la propiedad estilo y se repinte cambiado.

Hacer que el componente tome su color de manera aleatoria, para ello genere tres números aleatorios con rango de valor de 0 a 255. Cada número se asociará a un

color R,G,B (rojo,verde,azul) y con estos valores cambiar la propiedad foreground del componente. De esta manera cada componente tendrá un color diferente.

Sugerencias Didácticas

La manera recomendada para abordar el tema de creación de componentes, es primero dejar en claro las diferentes formas de generar un componente nuevo, remarcando las diferencias entre sí.

El primer método presentado resulta el más sencillo dado que solo se integran componentes existentes y se trabajan los eventos a los que responderá en conjunto.

Luego se puede presentar cualquiera de las dos maneras adicionales de crear componentes.

Reporte del alumno (resultados)

El alumno deberá presentar evidencias suficientes de haber realizado la práctica.

Deberá realizar las tareas que se mencionan en los apartados, tales como el habilitar el cambio del texto a color rojo cuando se introduzca una cifra con valor negativo en la caja de texto.

Bibliografía preliminar

1. Java The Complete Reference Eighth Edition
Herbert Schildt

Oracle Press 2011
2. Java 2 Interfaces Gráficas y Aplicaciones para Internet
Francisco Javier Ceballos
Alfaomega- RAMA 2ª. Edición 2007
3. En la página oficial de Java:
docs.oracle.com/javase/tutorial/java/