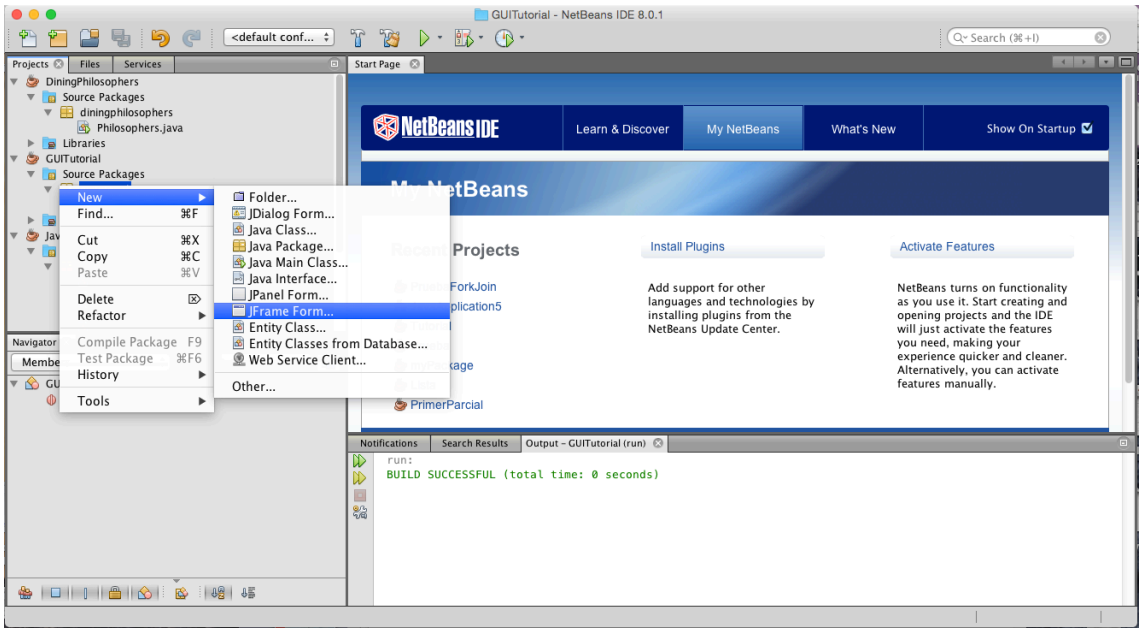
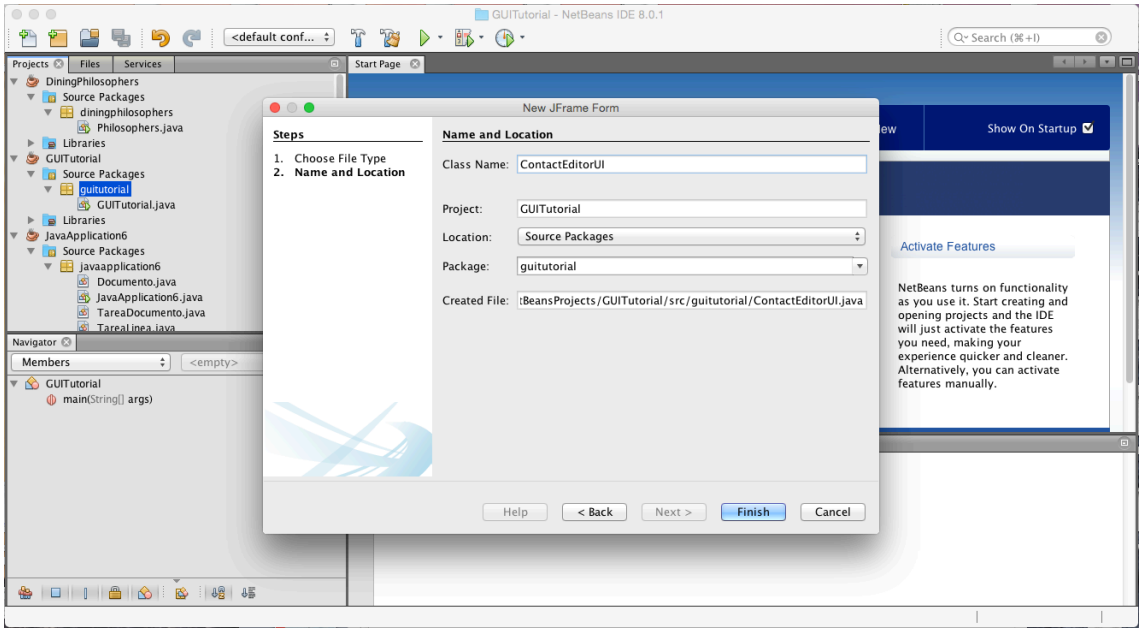
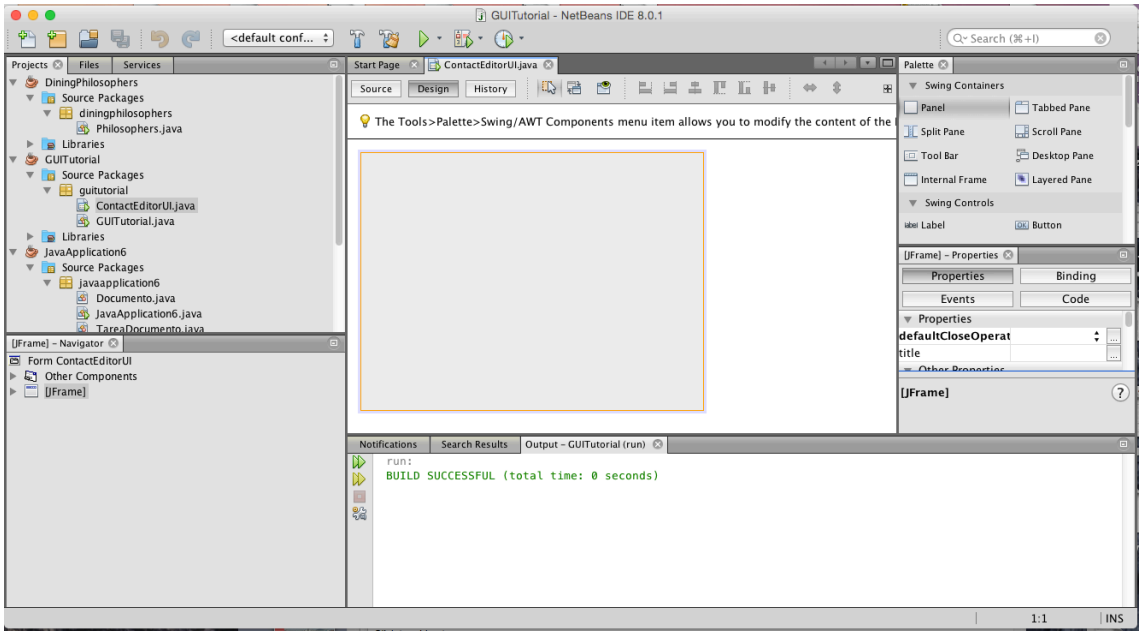
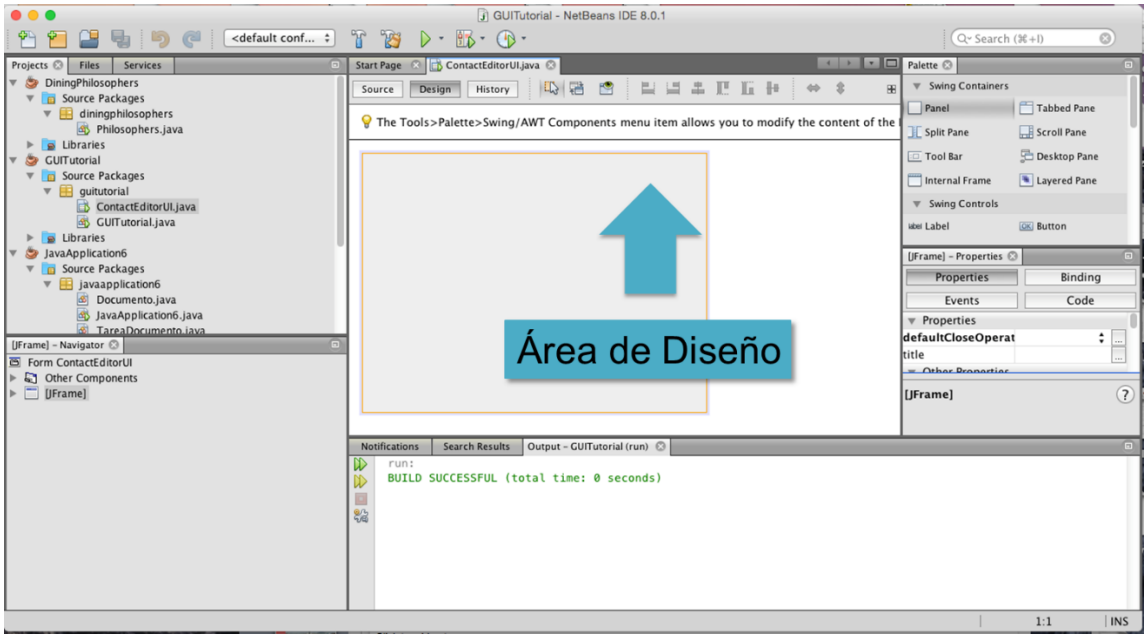
Java – Interfaces Gráficos de Usuario

Presentación interfaz de NetBeans para desarrollo de GUIs



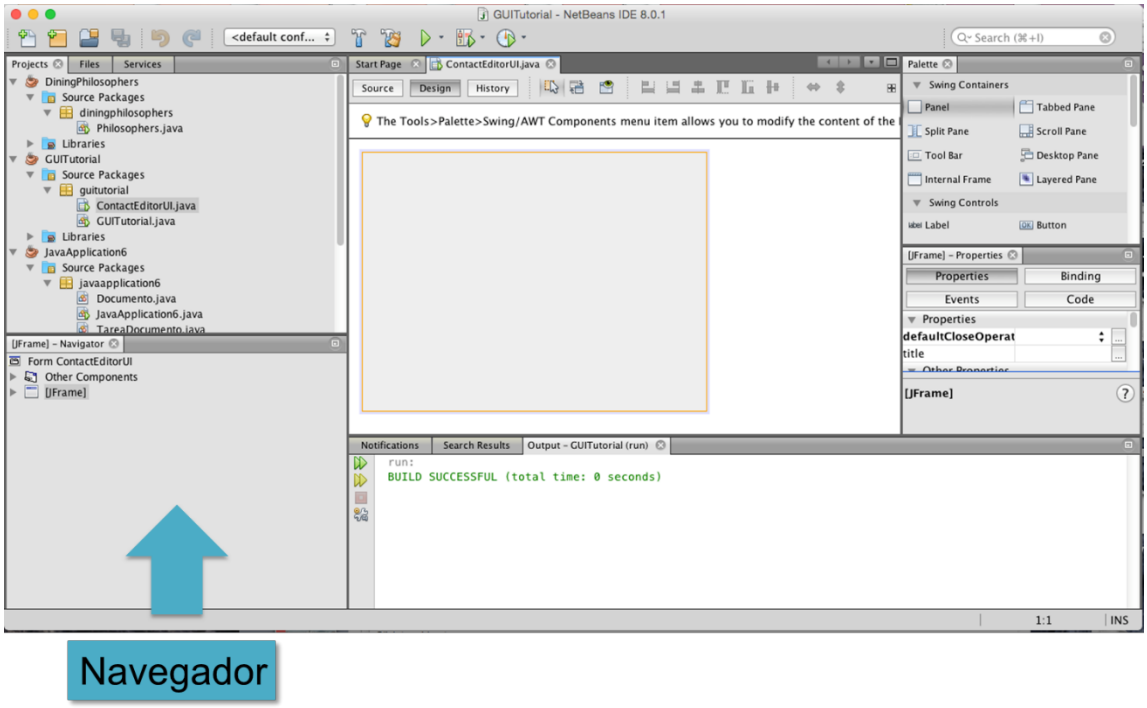




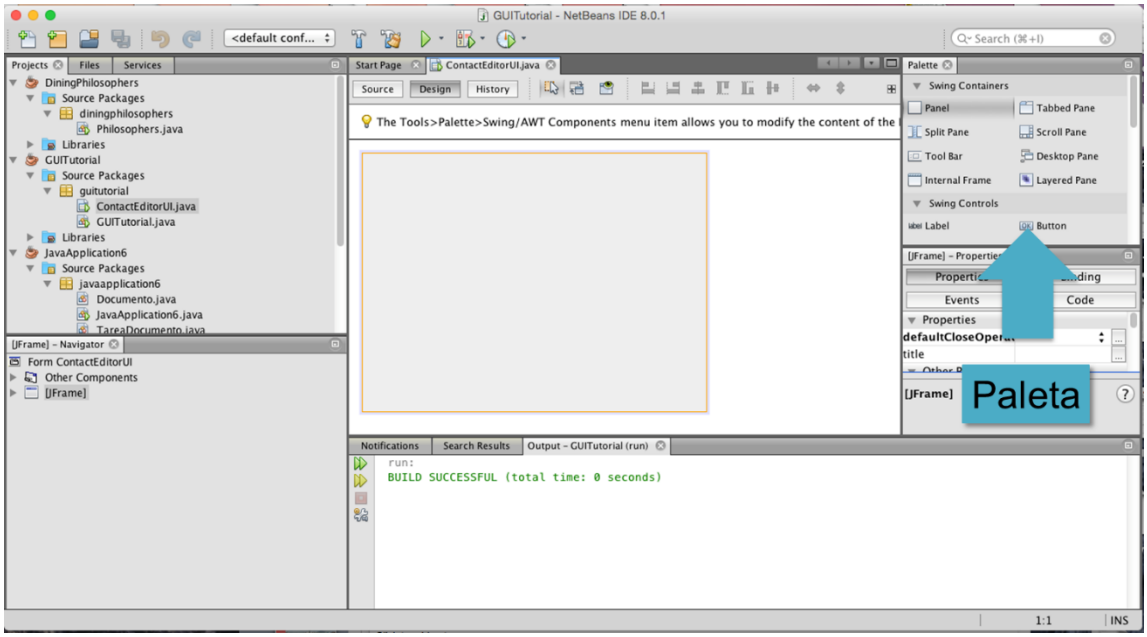


- La barra de herramientas asociada contiene 3 botones principales:
    - *Source* – permite ver el código fuente,
    - *Design* – vista gráfica de los componentes GUI,
    - *History* – te permite acceder al historial local de cambios producidos en el fichero.
- Los botones adicionales te permiten acceder a comandos muy habituales como:
    - Elección entre los modos *Selection* y *Connection*,
    - Alineación de componentes,
    - Establecer el comportamiento de auto-redimensionamiento de los componentes

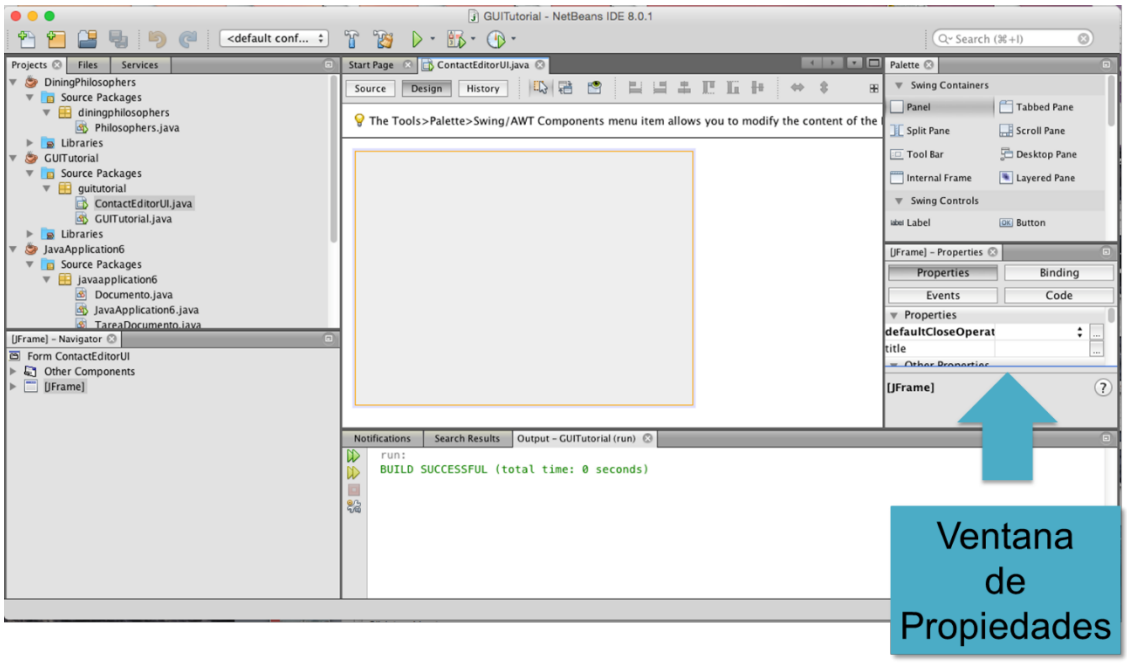
- Previsualización de los componentes.



- Proporciona una representación de todos los componentes, tanto visuales como no-visuales de la aplicación en modo árbol jerárquico
- Proporciona realimentación visual acerca de qué componente del árbol se está editando en cada momento
- También permite organizar los componentes en los paneles disponibles



- Una lista personalizable de componentes que contiene *tabs* tanto para componentes JFC/Swing, AWT, y JavaBeans, así como gestores de disposición de componentes componentes.
- Puedes crear, eliminar o reorganizar las categorías que se muestran en la paleta mediante el *customizer*.



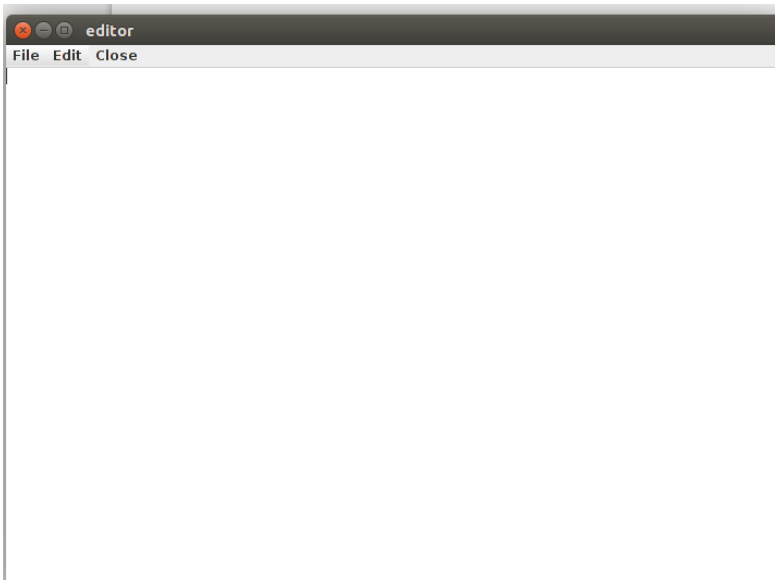
- Muestra las propiedades del documento actualmente seleccionado en el Constructor de GUI, la ventana de navegación, la ventana de proyectos o la ventana de archivos

---

Ejercicios a realizar

Ejercicio 1

Un editor de texto es un programa de ordenador que permite crear y modificar ficheros de texto. Está formado por un área rectangular destinada a la edición de texto y una zona superior con diferentes funcionalidades agrupadas en diferentes menús.



Para implementar el editor de texto básico hay que emplear las clases del paquete ***javax.swing*** *JTextArea*, *JMenuBar*, *JMenu* y *JMenuItems*.

La gestión de los eventos se hará implementando la interface *ActionListener* en la clase principal del editor.

El editor debe tener una barra de menú con dos menús (**File** y **Edit**) y un botón (**Close**).

El menú de fichero (**File**) tiene las siguientes opciones:

-*New*: Crea una nueva área de texto vacía

-*Open*: Muestra una ventana de diálogo para seleccionar un fichero que será leído y cargado en la area de texto, eliminando lo que hubiera previamente.

-*Save*: Muestra una ventana de dialogo para seleccionar en qué fichero se quiere almacenar el texto.

-*Print*: Imprime el contenido del área de texto.

El menú de edición (**Edit**) tiene las siguientes opciones:

-*Cut*: Elimina todo el texto del *JTextArea* y lo almacena en el portapapeles.

-*Copy*: Copia todo el texto del *JTextArea* al portapapeles.

-*Paste*: Pega el contenido del portapapeles al *JtextArea*.

Y como último elemento de la barra de menú hay que incluir un botón para cerrar la aplicación (**Close**).

Ejercicio 2

Al trabajar en un texto, es necesario disponer de las funcionalidades deshacer y rehacer de las acciones que se ejecutaron durante la sesión de edición. Los comandos Deshacer (***Undo***) y Rehacer (***Redo***) hay que implementarlos para el editor de texto gráfico.

Estas dos nuevas funcionalidades se incluirán en el menú ***Edit***. Para ello, se facilita el siguiente fragmento de código para añadir los *Listener* necesarios.

```
private class UndoEditListener implements UndoableEditListener {

		@Override
		public void undoableEditHappened(UndoableEditEvent e) {

			undoMgr__.addEdit(e.getEdit()); // remember the edit
		}
}

private class UndoActionListener implements ActionListener {

		private String type;

		public UndoActionListener(String type) {

			this.type = type;
		}

		@Override
		public void actionPerformed(ActionEvent e) {

			switch (type) {

				case "UNDO":
					if (! undoMgr__.canUndo()) {
```

```
                                        editor__.requestFocusInWindow();
                                        return; // no edits to undo
                                    }

                                    undoMgr__.undo();
                                    break;

                                case "REDO":
                                    if (! undoMgr__.canRedo()) {

                                        editor__.requestFocusInWindow();
                                        return; // no edits to redo
                                    }

                                    undoMgr__.redo();
                            }

                            editor__.requestFocusInWindow();
                        }
}
```

Ejercicio 3

Los diálogos son ventanas secundarias que aparecen sobre una ventana padre, primaria. Se utilizan para presentar información adicional o controles, incluyendo las preferencias y propiedades, o para presentar mensajes o preguntas.

En el editor de texto gráfico se requiere añadir dos ventanas de diálogo:

1ª) Ventana de diálogo que muestre una advertencia antes de cerrar la aplicación con la opción del menú ***Close***.

2ª) Ventana de diálogo a medida en una nueva opción del menú ***File*** que sea "*About*" que muestre información del fabricante: un texto con el creador del editor de texto gráfico y la versión, un mapa con la ubicación de la UPNA y un botón para cerrar la ventana.

Ejercicio 4

Un procesador de texto es un editor de texto al que se le puede añadir formato al texto. El formato de texto permite emplear diferentes tipografías, tamaños de letra, colores, tipos de párrafos, efectos artísticos, e incluso insertar imágenes y otros elementos externos.

En este ejercicio se limita las opciones de estilo a implementar a las siguientes: *bold*, *italic* y *underline*. Para ello hay que crear un nuevo menú denominado ***Format*** con los 3 *JmenuItem* anteriores.

El almacenamiento (*Save*) y recuperación del texto de un fichero (*Open*) debe modificarse para poder almacenar y recuperar el texto con su formato.

Ejercicio 5

HTML son las sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), que hace referencia al lenguaje para elaborar páginas web. Es un estándar que define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, vídeos, juegos, entre otros.

En este ejercicio hay que implementar un editor de HTML con posibilidad de scroll junto a una zona de visualización del HTML que se refresca con cualquier cambio en la zona de edición, tal y como se muestra en la siguiente captura.

Ejercicio 6

Los mensajes de error en ventanas de dialogo son una forma de mostrar errores de funcionamiento pero deben ser entendibles por un usuario y no pueden incluir detalles internos de implementación por cuestión de seguridad. Así, las aplicaciones requieren otras formas de registrar la actividad del usuario y de la aplicación. El equipo de soporte de cualquier aplicación necesita conocer los últimos archivos abiertos, últimos archivos modificados, los últimos comandos ejecutados, estado de la JVM, el volcado del stacktrace, etc., para poder dar respuesta a incidencias o problemas.

Un log de aplicaciones graba cronológicamente las operaciones durante el funcionamiento de la aplicación. Su función forma parte de la lógica de la aplicación. Por lo tanto, no debería estar detenida durante el funcionamiento de la misma.

La mayoría de los registros se almacenan en texto sin formato o en XML. De esta forma, el log puede ser fácilmente leído y procesado.

El nivel del LOG define la granularidad y prioridad de la información registrada. Los mas usados suelen ser DEBUG, INFO y ERROR.

Para proporcionar capacidad de generar logs de forma avanzada hay que integrar **LOG4J**, que es un proyecto de Apache, para generar el log de la aplicación del editor de texto gráfico con múltiples opciones de configuración. En este ejercicio, el editor de texto gráfico debe generar algunas líneas de log que deben aparecer en la consola.