11.A. Introducción a las GUI.

Sitio: Aula Virtual CIERD (CIDEAD)
Curso: Programación_DAM
Libro: 11.A. Introducción a las GUI.
Imprimido por: LUIS PUJOL
Día: miércoles, 20 de mayo de 2020, 09:19

Tabla de contenidos

1 Introducción.

Hoy en día las **interfaces** de los programas son cada vez más sofisticadas y atractivas para el usuario. Son intuitivas y cada vez más fáciles de usar: pantallas táctiles, etc.

Sin embargo, no siempre ha sido así. No hace muchos años, antes de que surgieran y se popularizaran las interfaces gráficas de usuario para que el usuario interactuara con el sistema operativo con sistemas como Windows, etc., se trabajaba en **modo consola**, o **modo carácter**, es decir, se le daban las ordenes al ordenador con comandos por teclado, de hecho, por entonces no existía el ratón.

De manera que con el tiempo, con la idea de simplificar el uso de los ordenadores para extender el uso a usuarios de todo tipo, se ha convertido en una práctica habitual utilizar interfaces gráficas de usuario (GUI en inglés). Con ellas el usuario puede interactuar y establecer un contacto más fácil e intuitivo con el ordenador.

También verás otras definiciones de interfaz, como la que define una interfaz como un dispositivo que permite comunicar dos sistemas que no hablan el mismo lenguaje. También se emplea el término interfaz para definir el juego de conexiones y dispositivos que hacen posible la comunicación entre dos sistemas.

Aquí en este módulo, cuando hablamos de interfaz nos referimos a la cara visible de los programas tal y como se presenta a los usuarios para que interactúen con la máquina. La interfaz gráfica implica la presencia de un monitor de ordenador, en el que veremos la interfaz constituida por una serie de **menús e iconos que representan las opciones que el usuario** puede tomar dentro del sistema.

Las interfaces gráficas de usuario proporcionan al usuario ventanas, cuadros de diálogo, barras de herramientas, botones, listas desplegables y muchos otros elementos. Las aplicaciones son conducidas por eventos y se desarrollan haciendo uso de las clases que para ello nos ofrece el API de Java.

En 1981 aparecieron los primeros ordenadores personales, los llamados PC, pero hasta 1993 no se generalizaron las interfaces gráficas de usuario. El escritorio del sistema operativo Windows de Microsoft y su sistema de ventanas sobre la pantalla se ha estandarizado y universalizado, pero fueron los ordenadores Macintosh de la compañía Apple los pioneros en introducir las interfaces gráficas de usuario.

Para saber más

En el siguiente enlace puedes ver la evolución en las interfaces gráficas de diverso software, entre ellos, el de las GUI de MAC OS, o de Windows en las sucesivas versiones

Galería de interfaces gráficas

Autoevaluación

Señala la opción correcta acerca de las interfaces gráficas de usuario:

- ○ Son sinónimos de ficheros de texto.
- ○ Las ventanas de una aplicación no serían un ejemplo de elemento de interfaz gráfica de usuario.
- ○ Surgen con la idea de facilitar la interactuación del usuario con la máquina.
- ○ Ninguna es correcta.

2 Librerías de Java para desarrollar GUI.

Hemos visto que la interfaz gráfica de usuario es la parte del programa que permite al usuario interaccionar con él. Pero, ¿cómo la podemos crear en Java?

El API de Java proporciona una librería de clases para el desarrollo de interfaces gráficas de usuario (en realidad son dos: AWT y Swing). Esas librerías se engloban bajo los nombres de **AWT** y **Swing**, que a su vez forman parte de las **Java Foundation Classes o JFC.**

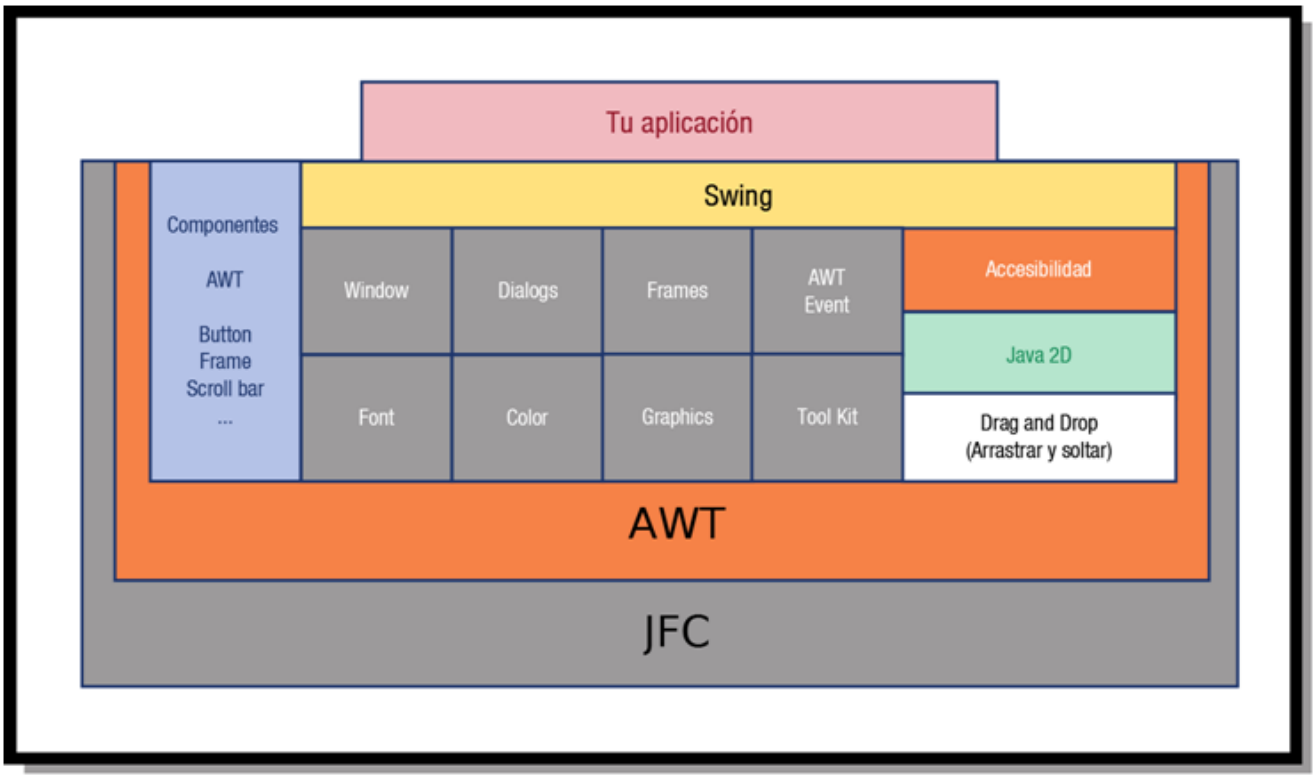


Imagen extraída de curso Programación del MECD.

Entre las clases de las **JFC** hay un grupo de elementos importante **que ayuda a la construcción de interfaces gráficas de usuario (GUI)** en Java.

Los elementos que componen las JFC son:

- **Componentes Swing**: encontramos componentes tales como botones, cuadros de texto, ventanas o elementos de menú.

- **Soporte de diferentes aspectos y comportamientos** (**Look and Feel**): permite la elección de diferentes apariencias de entorno. Por ejemplo, el mismo programa puede adquirir un aspecto **Metal** Java (multiplataforma, igual en cualquier entorno), **Motif** (el aspecto estándar para entornos Unix) o **Windows** (para entornos Windows). De esta forma, el aspecto de la aplicación puede ser independiente de la plataforma, lo cual está muy bien para un lenguaje que lleva la seña de identidad de multiplataforma, y se puede elegir el que se desee en cada momento.

- **Interfaz de programación Java 2D**: permite incorporar gráficos en dos dimensiones, texto e imágenes de alta calidad.

- **Soporte de arrastrar y soltar** (Drag and Drop) entre aplicaciones Java y aplicaciones nativas. Es decir, se implementa un portapapeles. Llamamos aplicaciones nativas a las que están desarrolladas

en un entorno y una plataforma concretos (por ejemplo Windows o Unix), y sólo funcionarán en esa plataforma.

- **Soporte de impresión.**

- **Soporte sonido:** captura, reproducción y procesamiento de datos de audio y MIDI.

- **Soporte de dispositivos de entrada distintos del teclado**, para japonés, chino, etc

- **Soporte de funciones de Accesibilidad, para crear interfaces para discapacitados:** permite el uso de tecnologías como los lectores de pantallas o las pantallas Braille adaptadas a las personas discapacitadas.

Con estas librerías, Java proporciona un conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en las que se ejecuten.

Autoevaluación

Señala la opción incorrecta. JFC **consta de los siguientes elementos:**

- Componentes Swing.
- Soporte de diversos "look and feel".
- Soporte de impresión.
- Interfaz de programación Java 3D.

2.1 AWT.

Cuando apareció Java, los componentes gráficos disponibles para el desarrollo de GUI se encontraban en la librería denominada Abstract Window Toolkit (**AWT**).

Las clases AWT se desarrollaron usando código nativo (o sea, código asociado a una plataforma concreta). Eso **dificultaba la portabilidad** de las aplicaciones. Al usar código nativo, para poder conservar la portabilidad era necesario restringir la funcionalidad a los mínimos comunes a todas las plataformas donde se pretendía usar AWT. Como consecuencia, AWT es una librería con una funcionalidad muy pobre.

La estructura básica de la librería gira en torno a componentes y contenedores.
Los contenedores contienen componentes y son componentes a su vez, de forma que los eventos pueden tratarse tanto en contenedores como en componentes.

AWT es adecuada para interfaces gráficas sencillas, pero no para proyectos más complejos.
Más tarde, cuando apareció Java 2 surgió una librería más robusta, versátil y flexible: **Swing**. AWT aún sigue estando disponible, de hecho se usa por los componentes de Swing.

Puedes ver la lista de los principales componentes de la clase AWT en la siguiente tabla:

Principales clases AWT	
NOMBRE DE LA CLASE AWT	**UTILIDAD DEL COMPONENTE**
Applet	**Ventana para una applet que se incluye en una página web.**
Button	**Crea un botón de acción.**
Canvas	**Crea un área de trabajo en la que se puede dibujar.** Es el único componente AWT que no tiene un equivalente Swing.
Checkbox	**Crea una casilla de verificación**.
Label	**Crea una etiqueta**.
Menu	**Crea un menú**.
ComboBox	**Crea una lista desplegable**.
List	**Crea un cuadro de lista.**
Frame	**Crea un marco para las ventanas de aplicación.**
Dialog	**Crea un cuadro de diálogo.**
Panel	**Crea un área de trabajo que puede contener otros controles o componentes.**
PopupMenu	**Crea un menú emergente.**
RadioButton	**Crea un botón de radio.**
ScrollBar	**Crea una barra de desplazamiento.**
ScrollPane	**Crea un cuadro de desplazamiento.**
TextArea	**Crea un área de texto de dos dimensiones.**
TextField	**Crea un cuadro de texto de una dimensión.**

Principales clases AWT	
NOMBRE DE LA CLASE AWT	**UTILIDAD DEL COMPONENTE**
Window	**Crea una ventana.**

Para saber más

Página oficial de Sun – Oracle sobre AWT (en inglés).

AWT (Abstract Window Toolkit)

AWT sigue siendo imprescindible, ya que todos **los componentes Swing se construyen haciendo uso de clases de AWT**. De hecho, como puedes comprobar en el API, todos los componentes Swing, como por ejemplo `JButton` (es la clase Swing que usamos para crear cualquier botón de acción en una ventana), derivan de la clase `JComponent`, que a su vez deriva de la clase AWT `Container`.

Las clases asociadas a cada uno de los componentes AWT se encuentran en el paquete `java.awt.`
Las clases relacionadas con el manejo de eventos en AWT están en el paquete `java.awt.event`.

AWT fue la primera forma de construir las ventanas en Java, pero:

- limitaba la portabilidad,
- restringía la funcionalidad y
- requería demasiados recursos.

Autoevaluación

AWT **está indicado para proyectos muy grandes y de gran complejidad.**

- ○ Verdadero.
- ○ Falso.

2.2 Swing.

Cuando se vio que era necesario mejorar las características que ofrecía AWT, distintas empresas empezaron a sacar sus controles propios para mejorar algunas de las características de AWT. Así, Netscape sacó una librería de clases llamada **Internet Foundation Classes** para usar con Java, y eso obligó a Sun (todavía no adquirida por Oracle) a reaccionar para adaptar el lenguaje a las nuevas necesidades.

Se desarrolló en colaboración con Netscape todo el conjunto de componentes Swing que se añadieron a la JFC.

Swing es una librería de Java para la generación del GUI en aplicaciones.

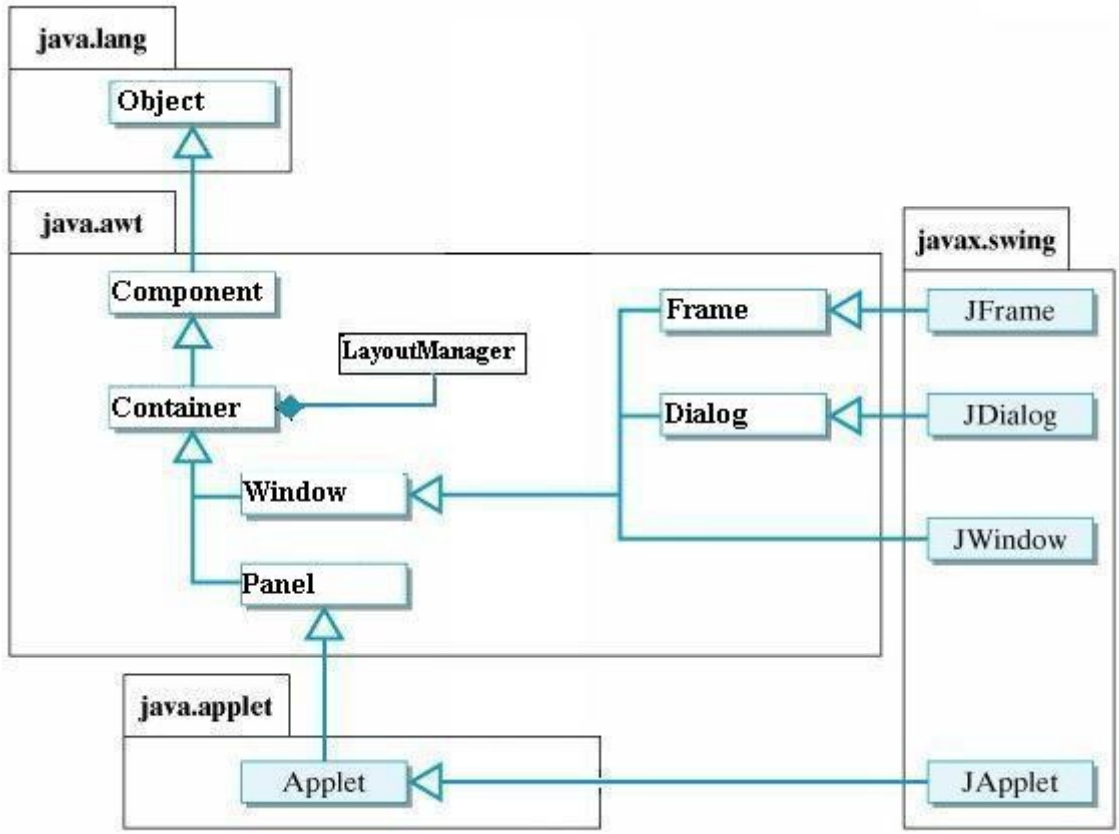


Imagen extraída de curso Programación del MECD.

Swing se apoya sobre AWT y añade `JComponents`. La arquitectura de los componentes de Swing facilita la personalización de apariencia y comportamiento, si lo comparamos con los componentes AWT.

Debes conocer

A continuación tienes una enlace con la lista de los controles Swing más habituales:

Controles Swing

Por cada componente AWT (excepto `Canvas`) existe un componente Swing equivalente, cuyo nombre empieza por J, que permite más funcionalidad siendo menos pesado. Así, por ejemplo, para el componente AWT `Button` existe el equivalente Swing `JButton`, que permite como funcionalidad

adicional la de crear botones con distintas formas (rectangulares, circulares, etc), incluir imágenes en el botón, tener distintas representaciones para un mismo botón según esté seleccionado, o bajo el cursor, etc.

La razón por la que no existe `JCanvas` es que los paneles de la clase `JPanel` ya soportan todo lo que el componente `Canvas` de AWT soportaba. No se consideró necesario añadir un componente Swing `JCanvas` por separado.

Algunas características más de Swing, podemos mencionar:

- Es independiente de la arquitectura (metodología no nativa propia de Java)

- Proporciona todo lo necesario para la creación de entornos gráficos, tales como diseño de menús, botones, cuadros de texto, manipulación de eventos, etc**.**

- **Los componentes Swing no necesitan una ventana propia del sistema operativo cada uno**, sino que son visualizados dentro de la ventana que los contiene mediante métodos gráficos, por lo que requieren bastantes menos recursos.

- **Las clases Swing están completamente escritas en Java**, **con lo que la portabilidad es total**, a la vez que no hay obligación de restringir la funcionalidad a los mínimos comunes de todas las plataformas

- Por ello las clase Swing aportan una considerable gama de funciones que haciendo uso de la funcionalidad básica propia de AWT aumentan las posibilidades de diseño de interfaces gráficas.

- Debido a sus características, los componentes AWT se llaman componentes **“de peso pesado”** por la gran cantidad de recursos del sistema que usan, y los componentes **Swing** se llaman componentes **“de peso ligero”** por no necesitar su propia ventana del sistema operativo y por tanto consumir muchos menos recursos.

- Aunque todos los componentes Swing derivan de componentes AWT y de hecho se pueden mezclar en una misma aplicación componentes de ambos tipos, se desaconseja hacerlo. Es preferible desarrollar aplicaciones enteramente Swing, que requieren menos recursos y son más portables.

3 Creación de interfaces gráficos de usuario utilizando asistentes y herramientas del entorno integrado.

Crear aplicaciones que incluyan interfaces gráficos de usuario, con Eclpse o con NetBeans, es muy sencillo debido a las facilidades que nos proporcionan sus asistentes y herramientas.

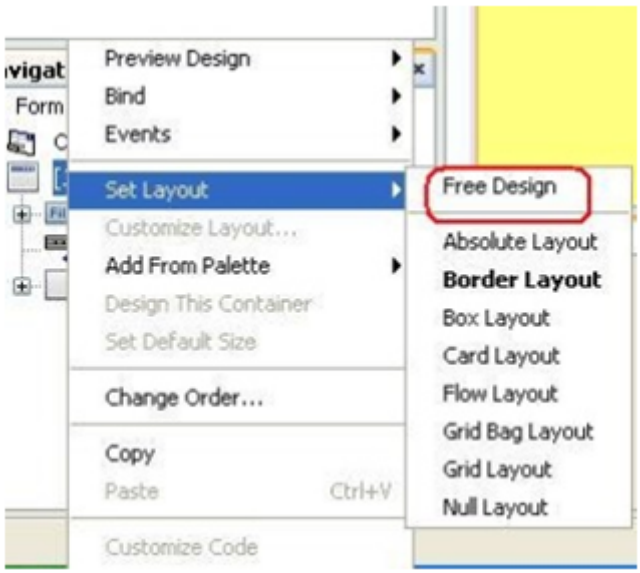
El IDE de programación nos proporciona un diseñador para crear aplicaciones de una manera fácil, sin tener que preocuparnos demasiado del código. Simplemente nos debemos centrar en el funcionamiento de las mismas.

Un programador experimentado, probablemente no utilizará los asistentes, sino que escribirá, casi siempre, todo el código de la aplicación. De este modo, podrá indicar por código la ubicación de los diferentes controles, de su interfaz gráfica, en el contenedor (panel, marco, etc.) en el que se ubiquen.

Pero en el caso de programadores novatos, o simplemente si queremos ahorrar tiempo y esfuerzo, tenemos la opción de aprovecharnos de las capacidades que nos brinda un entorno de desarrollo visual para diseñar e implementar formularios gráficos.

Algunas de las características de Eclipse (instalando el plugin Window Builder) o de NetBeans, que ayudan a la generación rápida de aplicaciones, y en particular de interfaces son:

- Modo de diseño libre (Free Design): permite mover libremente los componentes de interfaz de usuario sobre el panel o marco, sin atenerse a uno de los layouts por defecto.



- Independencia de plataforma: diseñando de la manera más fácil, es decir, arrastrando y soltando componentes desde la paleta al área de diseño visual, Eclipse y NetBeans sugieren y ajustan alineación, posición, y dimensión de los componentes, de manera que se adapten para cualquier plataforma, y en tiempo de ejecución el resultado sea el óptimo, sin importar el sistema operativo donde se ejecute la aplicación.

- Soporte de internacionalización. Se pueden internacionalizar las aplicaciones Swing, aportando las traducciones de cadenas de caracteres, imágenes, etc., sin necesidad de tener que reconstruir el proyecto, sin tener que compilarlo según el país al que vaya dirigido. Se puede utilizar esta características empleando ficheros de recursos (ResourceBundle files). En ellos, deberíamos aportar todos los textos visibles, como el texto de etiquetas, campos de texto, botones, etc.

NetBeans proporciona una asistente de internacionalización desde el menú de herramientas (Tools).

Debes saber

Echa un vistazo al siguiente video para ver como instalar en tu entorno Eclipse el plugin Window Builder:

Autoevaluación

Eclipse y NetBeans ayuda al programador de modo que pueda concentrarse en implementar la lógica de negocio que se tenga que ejecutar por un evento dado.

- Verdadero.
- Falso.

3.1 Diseñando con asistentes.

Los pasos para crear y ejecutar aplicaciones, se pueden resumir en:

- Crear un proyecto.
- Construir la interfaz con el usuario.
- Añadir funcionalidad, en base a la lógica de negocio que se requiera.
- Ejecutar el programa.

Veamos un ejemplo con estos pasos en Eclipse:

Veamos un ejemplo con estos pasos en NetBeans:

Hola Mundo en Entorno Grafico NetBeans - Swing Metodo 1