Tecnológico Nacional de México
Campus Orizaba

Tópicos Avanzados de Programación

Ingeniería en Sistemas Computacionales

Tema:
Componentes y Librerías

Integrantes:

Maciel Villa Valeria – 21010985
Muñoz Hernández Vania Lizeth – 21011009
Romero Ovando Karyme Michelle – 21011037

Grupo:
4g2A

Fecha de entrega: 18/Abril /2023

2.1 Definición conceptual de componentes, paquetes/librerías.

1. Componente:

Es una clase abstracta que representa todo lo que tiene una posición, un tamaño, puede ser pintado en pantalla y puede recibir eventos.

Un componente es el elemento básico de una interfaz gráfica. Los componentes permiten al usuario interactuar con la aplicación y proporcionar información desde el programa al usuario sobre el estado de la aplicación.

Cabe mencionar que los componentes nunca se encuentran de forma aislada, sino agrupados dentro de contenedores. Los contenedores contienen y organizan la situación de los componentes; además, son en sí mismos componentes y como tales pueden ser situados dentro de otros contenedores.

La clase padre Component no se puede utilizar directamente. Es una clase abstracta, que proporciona algunos métodos útiles para sus subclases.

- Component: superclase de todas las clases de interfaz gráfica.
- Container: para agrupar componentes.
- JComponent: superclase de todos los componentes de Swing que se dibujan directamente en los lienzos (canvas). Sus subclases son los elementos básicos de la GUI.
- JFrame: ventana que no está contenida en otras ventanas.
- JDialog: cuadro de diálogo.
- JApplet: subclase de Applet para crear applets tipo Swing.
- JPanel: contenedor invisible que mantiene componentes de interfaz y que se puede anidar, colocándose en otros paneles o en ventanas. También sirve de lienzo.
- Graphics: clase abstracta que proporciona contextos gráficos donde dibujar cadenas de texto, líneas y otras formas sencillas.
- Color: color de los componentes gráficos.
- Font: aspecto de los caracteres.
- FontMetrics: clase abstracta para propiedades de las fuentes.

Categorías de clases:

- ✓ Contenedores:
  JFrame, JApplet, JWindow, JDialog

- ✓ Componentes intermedios:
  JPanel, JScrollPane

- ✓ Componentes:
  JLabel, JBbutton, JTextField, JTextArea, ...

- ✓ Clases de soporte:
  Graphics, Color, Font, ...

2. Paquetes/librerías:

Los paquetes en Java son una manera de organizar nuestras clases, ya sea por finalidad, por su relación con la herencia que tienen, etc.

Conforme empieza a crecer un desarrollo de Software surge la necesidad de reutilizar ciertos componentes que ya han sido escritos, así como dar cierta estructura para mantener una organización de código; al igual que otros lenguajes, esta organización se lleva a cabo mediante librerías, denominadas "packages" en el mundo Java .

- AWT: es una biblioteca pesada, heavy weight de componentes. La creación, visualización y gestión de los elementos gráficos depende del SO.
- Swing: es una biblioteca ligera light weight de componentes. Es Java quien visualiza y gestiona la interacción del usuario sobre los elementos de la interface gráfica.
- JavaFX: permite crear interfaces gráficas de usuario tanto para aplicaciones de escritorio como para la web y dispositivos móviles.

2.2 Uso de librerías proporcionadas por el lenguaje.

Java es un lenguaje de programación desarrollado para una multitud de plataformas y procesadores. Consideremos los dos tipos de aplicaciones gráficas más comunes. Modelos de Frames y Applets, se pueden construir usando cualquiera de las dos galerías de componentes visuales, son:

- *JAVA AWT:* Es la librería visual más antigua de java usando esta librería, se podrán construir los tres tipos de programas más comunes como son FRAME, WINDOW y APPLET.
- *JAVA SWING:* Es la librería de componentes visuales más nueva que proporciona java, usando esta librería se podrán construir los tres tipos de programas o aplicaciones que son JFRAME, WINDOW Y JAPPLET.

Un applet es un programa en java que se mandan a una máquina o PC remota para que los ejecuten o lo corra, cuando este applet de llegada a las máquinas remotas vía browser, dicho browser es quien activa la máquina virtual de java que da la orden de compilación y ejecución, es decir java programa. applet.

Entonces es importante que la máquina virtual de java, que se encuentra en la PC remota, tenga capacidad de incluir todas las librerías de java, como la de match, la de AWT, la de lang.etc.

Existen diferentes librerías en java, entre las cuales se encuentra.

→ **`Java.lang`**

Colección de tipo básico siempre importados a cualquier unidad de compilación. Aquí están las declaraciones de objetos, clases, wrappers.

Interfaces Clases:

- Cloneables Boolean
- Comparable Byte
- Runnable Character
- ClassLoader
- Compiler
- Double
- Float
- InheritableThreadLocal
- Interger
- Long
- Math
- Number
- Object
- System
- Thread
- Void String, etc.

→ **`Java.io`**

Archivos de stream y acceso aleatorio. Librería estándar de entrada y salida.

Interfaces Clases:

- DataInput BufferedInputStream
- DataOutput BufferedOutputStream
- Externalizable BufferedReader
- Filefilter Bufferedwrite
- FilenameFilter ByteArrayInputStream
- OdjectInput ByteArrayOutputStream
- Serializable DataOutputStream
- File
- InputStream reader
- Writer,etc.

→ **Java.net**

Librería que apoya interfaces con telnet y URL.

Interfaces Clases:

- ContentHandlerFactory Authenticator
- DatagramSocketImplFactory ContentHandler
- FileNameMap DatagramPacket
- SocketOptions DatagramSocketImpl
- URLStreamHanlerFactory HttpURKConnection URL, etc.

→ **Java.util**

Clase como de diccionarios, tabla de hash, stack, técnica de codificación hora, fecha, etc.

Interfaces Clases:

- Collection AdstractCollection
- Comparator AdstracList
- Enumeration AdstrectMap
- EventListener AdstrectSecquentialList
- Interator AdstractSet
- List ArreyList
- Observer Collection
- SortedSet EventObject
- Random Stack
- Timer
- Vector
- Date,etc.

→ **Java.Awt**

Abstract Windowing Toolkit que proporciona una capa abstracta que permita llevar una aplicación en java de un sistema de ventanas a otro. Contiene clases para componentes básicos de la interfaz, tales como eventos, colores, tipos de letra, botones, campos de texto.

Estructura del awt.

La estructura de la versión actual del AWT en la plataforma Java 2 se puede resumir en los puntos siguientes:

- Los contenedores contienen componentes, que son los controladores básicos.

- No se usan posiciones fijas de los componentes, si no están situados a través de una disposición controlado (layouts)
- El común denominador de más bajo nivel se acerca al teclado, ratón y manejo de eventos.
- Alto nivel de abstracción respecto al entorno de ventanas en que se ejecute la aplicación (no hay áreas clientes, ni llamadas a X).
- La arquitectura de la aplicación es dependiente del entorno de ventanas, en vez de tener un tamaño fijo.
- Carece de un formato de recursos. No se puede separar el código de lo que es propiamente interfaz. No hay ningún diseñador de interfaz toda vía.

Interfaces Clases:

- ActiveEvent AlphaComposite
- Adjustable AWTEvent

→ **Java.applet**

El paquete java.applet permite la creación de applets a través de la clase Applet, proporciona interfaces para conectar un applet a un documento web y para audición de audio.

Interfaces Clases:

- AppletContext Applet
- AppletStub
- AudiClip

→ **Java.math**

Esta es la clase que representa la librería matemática de Java. Las funciones que contiene son las de todos los lenguajes, parece que se han metido en una clase solamente a propósito de agrupación, por eso se encapsulan en Math, y lo mismo sucede con las demás clases que corresponde a objetos que tiene un tipo equivalente (carácter, Float, etc.)

La clase Math es public para que se pueda llamar desde cualquier sitio y static para que no haya que iniciarla.

Proporciona cálculos en entero grande y real grande.

Clases:

- Bigdecimal
- Biginteger
- Además de la clase Math.

→ **Java.rml**

Este paquete hace posible que un objeto se ejecute en una máquina virtual Java invoque métodos de otro objeto que se ejecuta en la máquina virtual distinta; dicha máquina virtual puede encontrarse en ordenadores diferentes conectados a través de una red TCP/IP.

- Interfaces Clases
- Rmote MarshalledObject
- Naming
- RMISecurityManager

→ **Java.text**

Contiene clase que permiten dar formato especializado a fechas, números y mensajes.

Interfaces Clases:

- AttributedChacterIterator Annotation
- CharacterIterator AttibutedCharacterIterator
- ChoceFormat
- DateFormat
- Format
- MessageFormat
- NumberFormat
- ParsePosition

→ **Java.sound.midi**

Paquete con clase e interfaces que permitan la captura, procesamiento y reproducción de música MIDI.

Interfaces Clases:

- ControllerEventListener Instrument
- MataEventListener MeteMessage
- MidiChannel MidiDevice.info
- MidiDevice MidiEvent
- Receiver MidiFileFormat
- Sequecer Midemessage

→ **Java.SQL**

Junto con el paquete javax.sql, incluido en java 2 SDK Edición para la empresa, forma parte del API de java 2.0 (conexión Java a Base de Datos), y permite la

conexión de base de datos, él envió de sentencias SQL y la interpretación de los resultados de las consultas.

Intefaces Clases:

- Array Date
- Blob DriverManager
- CallabeStatement DriverPropertyInfo
- Clob SQLPermission
- Connecction Timer
- DatabaseMetaDate Timestamp
- Driver Type
- Ref
- SQLData
- SQLInput
- SQLOutput
- Struct

→ **JAVA.SWING**

Paquete que mejora e AWT, proporcionando un conjunto de componentes que se ejecutan de manera uniforme en todas las plataformas.

Interfaces Clases:

- Action AbstractAction
- ComboBoxEditor ActonMap
- Icon Box.Filler
- ListModel CellRendererPane
- MenuElement DebugGraphics
- WindowsConstants DefaulListSelectionModel
- JApplet
- Jbutton
- JCheckBox
- JFrame JMenu
- JLabel
- JPanel
- JTextField
- JTree
- JWindows
- Temer
- UIManager, etc.

2.3 Creación de componentes visuales y no visuales definidos por el usuario

Los componentes visuales se refieren a objetos gráficos que aparecen en la interfaz de usuario de una aplicación, como etiquetas, campos de texto, botones personalizados, gráficos, tablas, barras de progreso, menús desplegables, controles de entrada de datos personalizados, ventanas emergentes y elementos de diseño personalizados.

En cuanto a los componentes no visuales definidos por el usuario, son elementos que no se ven directamente en la interfaz de usuario, pero que realizan una función detrás de escena, algunos ejemplos podrían ser clases y objetos personalizados para interactuar con bases de datos, servicios web personalizados, herramientas de procesamiento de datos personalizadas, y todo tipo de lógica de negocio personalizada.

Dado que un componente es un objeto como otro cualquiera, podremos aplicar en todas las técnicas de la orientación a objetos: encapsulación, herencia y polimorfismo.

Para crear componentes visuales y no visuales definidos por el usuario, generalmente se siguen los siguientes pasos:

- Definir la clase base: La clase base es la plantilla para el componente que se va a crear. Se deben definir las propiedades, métodos y eventos que serán comunes a todos los componentes que se basen en esta clase.
- Heredar la clase base: Se hereda la clase base en una nueva clase que represente al componente específico que se va a crear. Esta nueva clase tendrá todas las propiedades, métodos y eventos de la clase base.
- Agregar propiedades, métodos y eventos específicos del componente: Se agregan las propiedades, métodos y eventos específicos del componente que se va a crear. Por ejemplo, si se está creando un botón, se agregarían propiedades como Texto y Imagen, métodos como Click y eventos como MouseEnter.

- Implementar el componente: Se implementa el componente en la aplicación o programa. Dependiendo del lenguaje de programación utilizado, esto puede implicar agregar el componente a un formulario, ventana o página web.

Al crear componentes visuales y no visuales definidos por el usuario, es importante asegurarse de que estén diseñados de manera eficiente y que sean fáciles de usar para otros desarrolladores. También es importante documentar adecuadamente el componente para que otros puedan entender cómo funciona y cómo se utiliza.

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

En Java, los paquetes (packages) se utilizan para agrupar de manera lógica los componentes relacionados de una aplicación. Los paquetes pueden contener una variedad de elementos, como clases, interfaces, archivos de texto y más. Al utilizar paquetes, podemos organizar y modularizar nuestra aplicación de una manera efectiva, lo que ayuda a mantener una estructura clara y organizada. Los paquetes en Java nos permiten categorizar diferentes estructuras que componen nuestro software, lo que facilita el mantenimiento y la escalabilidad del proyecto en el futuro.

Pero los paquetes no son las únicas herramientas que tenemos para el desarrollo de software, de igual forma contamos con las librerías en Java y otros lenguajes de programación, con las que es posible utilizar los métodos, clases y atributos que componen la librería, en lugar de tener que implementar esas funcionalidades nosotros mismos.

Para crear una biblioteca/paquetes definidos por el usuario, generalmente se siguen los siguientes pasos:

1. Definir las funcionalidades: Identificar las funcionalidades que se desean incluir en la biblioteca/paquete.
2. Escribir el código: Escribir el código necesario para implementar las funcionalidades identificadas en el paso anterior.

3. Empaquetar el código: Empaquetar el código en un archivo ZIP o TGZ que contenga todos los archivos necesarios para utilizar la biblioteca/paquete.
4. Publicar la biblioteca/paquete: Publicar la biblioteca/paquete en un repositorio en línea o servidor web, para que otros desarrolladores puedan acceder y utilizar la biblioteca/paquete.

Para utilizar una biblioteca/paquetes definidos por el usuario, se deben seguir los siguientes pasos:

1. Instalar la biblioteca/paquete: Descargar y configurar la biblioteca/paquete para su uso en el proyecto. Esto puede implicar la instalación de dependencias adicionales y la configuración de rutas de acceso.
    - Para importar librerías en Java se usa la palabra reservada **import** seguido de la "ruta" del paquete o clase que deseamos agregar al proyecto. Cabe resaltar que el **import** permite agregar a nuestro proyecto una o varias clases (paquete) según lo necesitemos.
2. Importar la biblioteca/paquete: Importar la biblioteca/paquete en el proyecto y usar las funcionalidades proporcionadas por la biblioteca/paquete.

Bibliografía

Blogspot. (Marzo de 2012). *Tópicos Avanzados de Programación - ITCA*. Obtenido de http://progitca.blogspot.com/2012/03/35-creacion-y-uso-de-paqueteslibrerias.html

González, J. D. (2022). *Importar librerías estándar de Java y librerías propias.* Obtenido de https://www.programarya.com/Cursos/Java/Librerias

González, J. D. (2022). *Uso de los paquetes en Java*. Obtenido de https://www.programarya.com/Cursos/Java/Paquetes

Haverbeke, M. (2018). *Eloquent JavaScript.*

Yañez, C. (2018). *Desarrollo de interfaces: cómo crear componentes visuales*. Obtenido de https://www.ceac.es/blog/desarrollo-de-interfaces-como-crear-componentes-visuales