Unidad III: Componentes y librerías

3.1. Definición conceptual de componentes, paquetes/librerías

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

LIBRERÍAS

La utilización de objetos dinámicos supone dejar pendiente en el montaje de la aplicación el enlace de dichos objetos. Cuando la aplicación está en ejecución, y sólo entonces, se produce el enlace (dinámico) con los objetos contenidos en la librería.

La creación de librerías dinámicas corre a cargo del enlazador o montador (en nuestro caso el ld) aunque también es posible indicar al compilador las opciones necesarias para el montaje y de ese modo, será él quien se encargue de pasárselas al montador.

Cuando se crea un objeto dinámico es necesario que dicho código objeto sea independiente de la posición, para conseguir este tipo de código debe especificarse al compilador la opción -fPIC (Position IndependentCode). Dicho flag debe indicarse tanto en la compilación como en el montaje de la librería.

Para montar los objetos es necesario además indicar la opción -shared para que el resultado sea un fichero objeto ‘compartible’.

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

Un componente desde el punto de vista de programación esta compuesto por varias secciones y el conjunto de todas ellas dan lugar a la creación de dichocomponente.

Por tanto, primero para no perder el norte, vamos a empezar definiendo elconvenio que es utilizado para organizar dichas secciones por TinyOs. En

general, uncomponente posee tres grandes secciones que son: Configuration, Implementation,Modul e. Estas tres secciones han de estar obligatoriamente presentes en cualquiercomponente aunque puedan estar vacías.

El estándar de T i n y O s determina, que las secciones de Configuration eImplementatión han de ir en un fichero que recibirá el nombre del componente con laextensión .nc y la tercera sección de Module deberá de ir en otro fichero aparte querecibirá el nombre del componente concatenado con un M mayúscula (la M da elsignificado al fichero, es el significado de Module) , este último fichero también poseeráa extensión .nc.

Otra buena costumbre consiste en crear un fichero de header o cabecera con extensión .h que contenga todas las enumeraciones, registros o tipos de datos creadospor el usuario de los que hace uso la aplicación, y cuando se realiza esto la forma deligar dicho fichero con los otros dos es utilizando al principio de los otros fichero ladirectiva includesheader; aunque como mención especial decir que si nos fijamosmejor en este directiva se puede ver que no se incorpora la extensión .h en la misma.

Ahora que ya sabes cuales son las secciones que va a contener cada ficherovamos a empezar a explicar cada una de ellas.

ImplementatiónEsta sección se va a encargar de definir las conexiones que hay entre losdiferentes componentes que utiliza la aplicación, esto es debido a que si recordamos unpoco, se ha comentado que la programación de un componente (que se llevará a cabo en la sección de module) se hace utilizando interfaces y dichas interfaces para poderutilizarlas las ha de proporcionar un componente, entonces básicamente es esta secciónse definen cuales son los componentes que proporcionan las interfaces a nuestraaplicación (por lo general serán componentes primitivos).

Una vez que conocemos la finalidad de esta sección y llegados a este punto, vamos a insertar un concepto nuevo que es la diferencia que existe entre una aplicaciónque esta ya disponible para ser ejecutada en un sensor y un componente cualquiera. Ladiferencia es muy poca, y consiste en que una aplicación es un componente comocualquier cosa en este lenguaje que en su

sección de implementación hace uso de uncomponente especial denominado Main.

3.3. 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 mas 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
VoidString, etc...

Java.io
Archivos de stream y acceso aleatorio. Librería estándar de entrada y salida.
Interfaces Clases
DataInputBufferedInputStream
DataOutputBufferedOutputStream
ExternalizableBufferedReader
FilefilterBufferedwrite

FilenameFilterByteArrayInputStream
OdjectInputByteArrayOutputStream
SerializableDataOutputStream
File
InputStream reader
Writer,etc..

Java.net
Librería que apoya interfaces con telnet y URL.

Interfaces Clases
ContentHandlerFactory Authenticator
DatagramSocketImplFactoryContentHandler
FileNameMapDatagramPacket
SocketOptionsDatagramSocketImpl
URLStreamHanlerFactoryHttpURKConnection 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
EventListenerAdstrectSecquentialList
InteratorAdstractSet

List ArreyList
Observer Collection
SortedSetEventObject
Random Stack
Timer
Vector
Date,etc.

Java.Awt

AbstractWindowingToolkit 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 controládores básicos.

No se usan posiciones fijas de los componentes, si no estan situados a traves de una disposición controlado (layouts)

El común denominador de mas 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

ActiveEventAlphaComposite

AdjustableAWTEvent

Java.applet

El paquete java.applet permite la creación de appletsatraves 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

Proporciona cálculos en entero grande y real grande.

Clases

Bigdecimal

Biginteger

Además de la clase 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.

Java.rmi

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

Interfaces Clases

RmoteMarshalledObject

Naming

RMISecurityManager

Java.text

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

Interfaces Clases

AttributedChacterIterator Annotation

CharacterIteratorAttibutedCharacterIterator

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

MataEventListenerMeteMessage

MidiChannel MidiDevice.info

MidiDeviceMidiEvent

Receiver MidiFileFormat

SequecerMidemessage

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, el envió de sentencias SQL y la interpretación de los resultados de las consultas.

IntefacesClases

Array Date

Blob DriverManager

CallabeStatementDriverPropertyInfo

ClobSQLPermission

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
ComboBoxEditorActonMap
Icon Box.Filler
ListModelCellRendererPane
MenuElementDebugGraphics
WindowsConstantsDefaulListSelectionModel
JApplet
Jbutton
JCheckBox
JFrameJMenu
JLabel
JPanel
JTextField
JTree
JWindows
Temer
UIManager

3.4. Creación de componentes (visuales y no visuales) definidos por el usuario 3.5. Creación y uso de paquetes/librerías definidas por el usuario

Se puede establecer muchas clasificaciones para los componentes. Una de ellas es la de visuales o controles, frente a no visuales.

Un componente es visual cuando tiene una representación gráfica en tiempo de diseño y ejecución (botones, barras de scroll, cuadros de edición, etc.), y se dice no visual en caso contrario (temporizadores, cuadros de diálogo-no visibles en la fase de diseño, etc). Por lo demás no existen más diferencias entre ellos, excepto, claro está, las derivadas de la visualización del componente.

Los componentes no visuales se pueden colocar en los formularios de la misma manera que los controles, aunque en este caso su posición es irrelevante.

Para empezar, los componentes visuales podemos dividirlos a su vez en dos tipos:

-Componentes interactivos: permiten que el usuario final los manipule, ya sea introduciendo datos, seleccionado elementos, etc. De forma que estos componentes pueden recibir el foco (con SetFocus) así como los eventos propios del teclado y del ratón. Normalmente, el propio sistema operativo es el encargado de dibujar el aspecto del componente, haciendo el componente las llamadas correspondientes para que este aspecto cambie.

-Componente gráficos: el propio componente es el encargado de dibujar en la pantalla lo que crea oportuno, bien a través de las funciones básicas del API de Windows (con el objeto TCanvas) o bien a través de otras librerías gráficas, como OpenGL, DirectX, etc. Estos componentes, no suelen recibir eventos del usuario final, aunque si eventos del propio programador, ya que su cometido no suele ir más allá de mostrar ciertos gráficos o imágenes en la pantalla.

Si tuviéramos que crear un componente interactivo desde el principio, sería demasiado complejo, ya que tendríamos que luchar encontrar el propio API del sistema operativo, gestionando sus mensajes, las llamadas las funciones a bajo nivel, etc. Sin embargo, podemos aprovechar la mayoría del trabajo hecho por

Borland en la VCL, y crear componentes interactivos a partir de otros ya existentes, aplicado la técnica de herencia.