DELPHI

c04

"Objetos en Delphi"

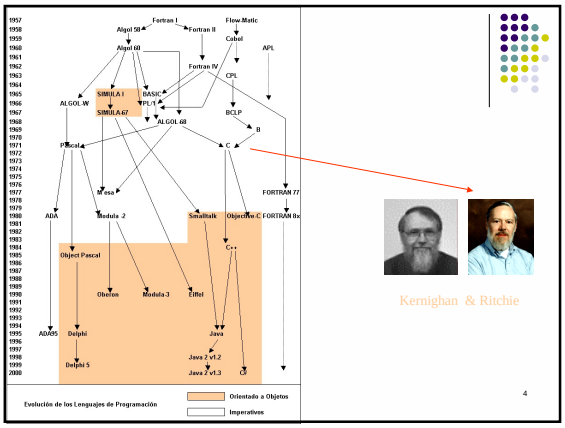
Pablo Roncagliolo

DELPHI

AÑO	LENGUAJE	INVENTOR	DESCRIPCION
1900s	BINARIO	Bool	primer lenguaje
1950	ASM (ensamblador)		lenguaje ensamblador
1951	A-0	Grace Hopper	fue el primer compilador
1952	AUTOCODE	Alick E. Glennie	compilador muy rudimentario
1956	FORTRAN	IBM	sistema de TRAducción de FORmulas matemáticas
1956	COBOL		Compilador
1958	ALGOL 58		
1960	LISP		Interprete orientado a la Inteligencia Artificial
1961	FORTRAN IV	IBM	sistema de TRAducción de FORmulas matemáticas
1961	COBOL 61 Extendido		
1960	ALGOL 60 Revisado		
1964	PASCAL	Niklaus Wirth	programacion estructurada
1964	BASIC	Universidad de Dartmouth (california)	Beginners All Purpose Symbolic Instruction Code
1966	FORTRAN 66	IBM	sistema de TRAducción de FORmulas matemáticas
1967	SIMULA 67		
1970s	GW-BASIC		antiguo y clasico BASIC
1972	C	Laboratorios Bell	lenguaje con tipos
1980s	SMALLTALK/V	Digitalk	pequeño y rapido
1980	C con clases	Laboratorios Bell	lenguaje con clases

DELPHI

AÑO	LENGUAJE	INVENTOR	DESCRIPCION
1981	PROLOG	Ministerio Japonés de Comercio Internacion	Lenguaje estandar para la Inteligencia Artificial
1982	ADA	Ministerio de Defensa de los EE.UU	lenguaje muy seguro
1984	C++	AT&T Bell Laboratories (Bjarne Stroustrup)	compilador
1985	CLIPPER		compilador para bases de datos
1985	QuickBASIC	Microsoft®	compilador de BASIC
1989	ASIC v5.0		interprete tipo QBASIC shareware
1990s	VISUAL C++		
1990s	VISUAL BASICScript	Microsoft®	lenguaje de script
1990	HTML	Tim Berners-Lee	para internet
1993	XML	C. M. Sperberg-McQueen	para internet
1993	SGML	Charles F. Goldfarb	para internet
1990s	WML		para internet
1990s	ASP	Microsoft®	para internet
1990s	PHP		para internet
1991	VISUAL BASIC	Microsoft®	
1995	JAVA	Sun Microsystems	para internet y proposito general
1995	CLIPPER 5.01		compilador para bases de datos
1995	GNAT ADA95	Ministerio de Defensa de los EE.UU	lenguaje muy seguro
1995	FORTRAN 95	IBM	sistema de TRAducción de FORmulas matemáticas
1995	DELPHI		
2001	VISUAL BASIC .NET	Microsoft®	La evolución de Visual Basic



DELPHI

- ***RAD:*** *Desarrollo Rápido de Aplicaciones.*
- ***Delphi:*** *Utiliza el lenguaje Object Pascal.*
- ***POO:*** *Programación Orientada a Objetos (OOP en Inglés)*

P.O.O.

Problemas que resuelve:

- Portabilidad
- Reusabilidad
- Modificación de Código (=>herencias)
- Tiempos de Programación
- Codificación Intuitiva

Actualmente una de las áreas más candentes en la industria y en el ámbito académico es la orientación a objetos. La orientación a objetos promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

DELPHI

Característica de un Lenguaje P.O.O:

- Basado en Objetos
- Utiliza Clases
- Herencia

Un lenguaje orientado a objetos ataca estos problemas. Tiene tres características básicas: debe estar basado en objetos, basado en clases y capaz de tener herencia de clases. Muchos lenguajes cumplen uno o dos de estos puntos; muchos menos cumplen los tres. La barrera más difícil de sortear es usualmente la herencia.

El concepto de programación orientada a objetos (OOP) no es nuevo, lenguajes clásicos como SmallTalk se basan en ella. Dado que la OOP. se basa en la idea natural de la existencia de un mundo lleno de objetos y que la resolución del problema se realiza en términos de objetos, un lenguaje se dice que está basado en objetos si soporta objetos como una característica fundamental del mismo.

P.O.O.

OBJETO:

- **un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización**

Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

P.O.O.

OBJETO:

- **Relaciones**
- **Propiedades**
- **Métodos**

Cada uno de estos componentes desempeña un papel totalmente independiente:

- Las relaciones permiten que el objeto se inserte en la organización y están formadas esencialmente por punteros a otros objetos.
- Las propiedades distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.
- Los métodos son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

P.O.O.

OBJETO: silla

- **Relaciones**
- **Propiedades**
- **Métodos**

Relaciones:
- Sobre el "Piso"
- En la habitación

Propiedades:
- Nº de patas = 4
- Material = madera
- Color = café
- Estilo = rústico

Métodos:
- Sentarse en la silla
- Pararse de la silla
- Mover la silla

P.O.O.

OBJETO:

- **Encapsulamiento**
- **Ocultación**

Como hemos visto, cada objeto es una estructura compleja en cuyo interior hay datos y programas, todos ellos relacionados entre sí, como si estuvieran encerrados conjuntamente en una cápsula. Esta propiedad (encapsulamiento), es una de las características fundamentales en la OOP.

Los objetos son inaccesibles, e impiden que otros objetos, los usuarios, o incluso los programadores conozcan cómo está distribuída la información o qué información hay disponible. Esta propiedad de los objetos se denomina ocultación de la información.

Esto no quiere decir, sin embargo, que sea imposible conocer lo necesario respecto a un objeto y a lo que contiene. Si así fuera no se podría hacer gran cosa con él. Lo que sucede es que las peticiones de información a un objeto. deben realizarse a través de mensajes dirigidos a él, con la orden de realizar la operación pertinente. La respuesta a estas ordenes será la información requerida, siempre que el objeto considere que quien envía el mensaje está autorizado para obtenerla.

El hecho de que cada objeto sea una cápsula facilita enormemente que un objeto determinado pueda ser transportado a otro punto de la organización, o incluso a otra organización totalmente diferente que precise de él.

P.O.O.

Estructura de un OBJETO:

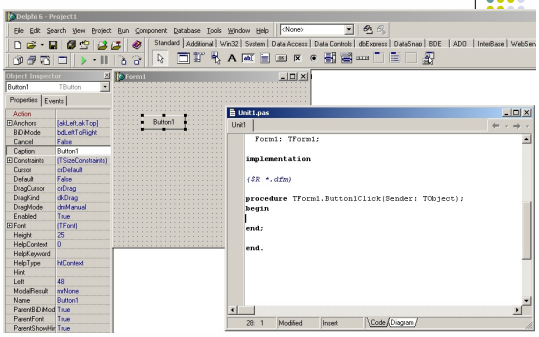
- **Relaciones**
  - **Jerárquicas (Obj. Padre-Hijos)**
  - **Semánticas (Clasificación de objetos)**
- **Propiedades**
  - **Propias**
  - **Heredadas (de Obj. Padres)**
- **Métodos**
  - **Propios**
  - **Heredados**

P.O.O.

Ej. de OBJETOS en Delphi:

- **Button**
  - Relación
    - ClassInfo (pointer)
  - Propiedad
    - Caption
    - Font
  - Método
    - OnClick
- **Edit**
  - Relación
    - ClassInfo (pointer)
  - Propiedad
    - Text
  - Método
    - OnChange

DELPHI: "visual pascal"

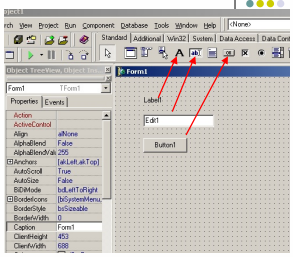


Objetos Básicos

Label
Label.caption
Label.font

Edit
Edit.text
Edit.font

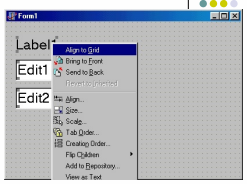
Button
Button.caption



Ejemplos en Delphi

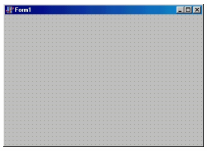
FORM

El objeto TForm



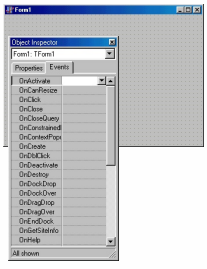
- La Grilla
- Seleccionar Objetos
- PopupMenu
  - "Alinear"
  - "Bring to Front" y "Send to Back"
- "Tiempo de Diseño"
- "Tiempo de Ejecución"
- El ícono (Project | Options)

FORM



Propiedad	Descripción	Valores
Caption	Título	String...
Name	Nombre	Form1
BorderStyle	Borde	Single, Sizeable, Tool
Color	Color	clRed, clBlue,clGreen ...
WindowState	Estado	Normal, maximizado, minimizado

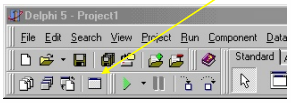
FORM



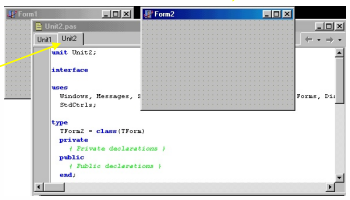
Eventos	Descripción
OnActivete	Cada vez que se selecciona el form
OnCreate	Sólo cuando comenzó el programa el form
OnClose	Cuando se cierra el form
OnKeyPress	Cuando se presiona una tecla
OnShow	Cuando se muestra el form
OnClick	Cuando se presiona el botón del mouse sobre el form

MULTIPLES FORM

Para crear un nuevo Form

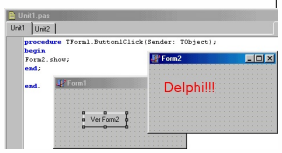


Se crear un nuevo Unit



MULTIPLES FORM

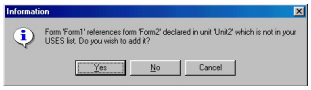
Para activar un Form en "tiempo de ejecución":



xxx.Show

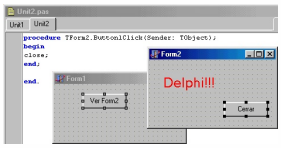
Donde xxx es el nombre del Form

Refernecias "cruzadas"



MULTIPLES FORM

Para cerrar un Form en "tiempo de ejecución":



Close;

Mediante refernecias "cruzadas":

xxx.close;

Donde xxx es el nombre del Form que se desea cerrar

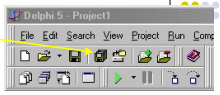
EJERCICIO PROPUESTO

Muchas aplicaciones mantienen la tradición de utilizar una ventana denominada "acerca de" (about), para incluir el nombre de los autores, empresa desarrolladora, versión del software, logo, fecha de creación etc.

Realice un programa básico que posea una ventana "acerca de" que pueda ser controlada desde un menú.

GUARDAR

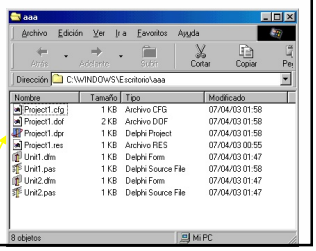
Guardar todo "save all"



- No utilizar nombre extraños
- Crear un directorio para cada proyecto
- Utilizar siempre el botón "save all"

Tipos de archivos...

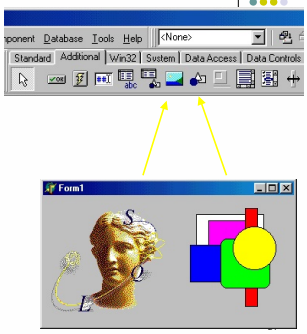
Abrir siempre el .dpr



OBJETOS ESPECIALES

IMAGE

- **Picture**: carga imágenes BMP, JPG y otras
- **Autosize**: ajusta el tamaño del objeto al de la imagen.
- **Stretch**: permite ajustar el tamaño de la imagen.
- **Transparent**: permite transparentar las partes no coloreadas de la imagen.



SHAPE

- **Brush | Color**: define el color de relleno
- **PEN | Color**: define color del borde
- **Shape**: tipo de figura (círculo, rectángulo, etc.)

OBJETOS ESPECIALES

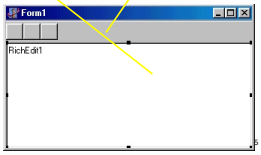
RichEdit

Editor de Texto

- **Font**
- **Align**: permite autoajustar el objeto según el tamaño del Form (client, top, left, none)

Toolbar

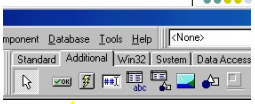
Permite crear una barra de herramientas (botones)



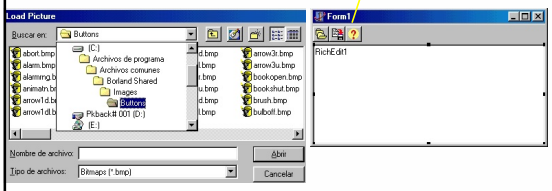
OBJETOS ESPECIALES

SpeedButton

- **Glyph**: permite asignar una imagen al botón.

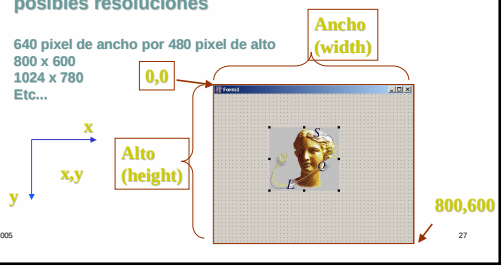


Imágenes....



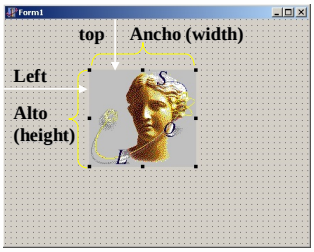
TIPs

- El tamaño de los objetos y los forms se mide en pixels (puntos)
- La pantalla de un computador posee varias posibles resoluciones
  - 640 pixel de ancho por 480 pixel de alto
  - 800 x 600
  - 1024 x 780
  - Etc...



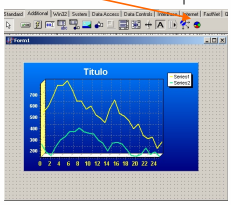
TIPs

- Cada objeto en el form se posiciona según las propiedades Left, Top.



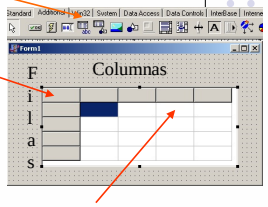
Gráficos (Chart)

- Tipo: línea, barras, pie
- Axis:
  - Left
  - Bottom
- Title
- 3D o 2D
- Gradiente



- Series...
  - Series1.Add(y,'',clred)
  - Series1.AddXY(x,y,'',clteecolor)
  - Series1.AddBar(valor,'',clteecolor)
  - Series1.clear
  - Series1.count

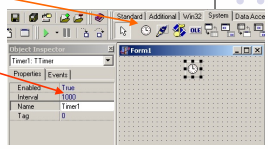
Grilla (StringGrid)



- Grid1.Cells[col,fil]
- Grid1.colcount
- Grid1.rowcount
- Grid1.fixedCols
- Grid1.fixedRows
- Grid1.Options.goEditing:=true

Reloj (Timer)

Intervalo en milisegundos



- Timer1.enabled:= true o false
- Timer1.interval:= 200;