

Programación con B4X

Breve teoría

Version 1.0, marzo 2021

Autor original: [Prokopis Pliroforikos](#)

Traducido al español por [LaMashino](#)



Anywhere Software

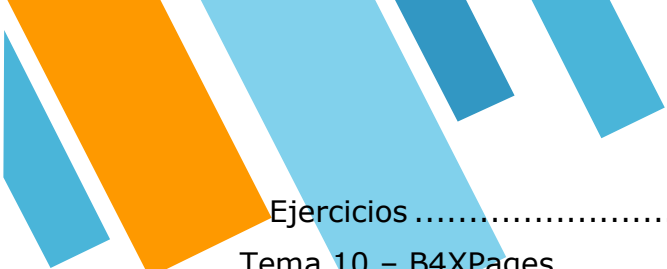
Tabla de Contenidos

Tema 1 – Por qué B4X	5
Instalación de B4X	5
Tema 2 – El significado de Problema	6
El concepto de problema	6
Comprender el problema	6
Buscando una solución o un conjunto de soluciones.	6
Elección de la solución correcta.....	7
Implementación de la solución.....	7
Comprobar si esta solución obtuvo los resultados deseados.....	7
Tema 3 – Mi primer Programa.....	8
Hola Mundo.....	8
Ejercicios	9
Tema 4 - Variables y Rango.....	10
Cómo averiguar cuántas variables necesitas	11
Cómo dar nombre a las Variables.....	12
Declaración de Variables	12
Mi Primera Variable.....	12
Comentarios.....	13
El área de log y la función log	14
Operadores matemáticos.....	15
Cadenas	15
Ejercicios	17
Tema 5 – Diseñador	19
Primeros pasos en el diseño	20
Diseñador visual	21
El Árbol de Vistas	21
Propiedades.....	22
Diseñador Abstracto.....	22
Ejemplo 1	22
Decidir el tamaño de la ventana de tu aplicación.....	22
Establecer una variante adecuada	23
Diseña un esquema de tu pantalla.....	23
Crear las Vistas	23
Ejercicios	26
Tema 6 – Del Diseñador al Código	28



Class_Globals	28
Un estudio más profundo del uso de variables.....	29
Paso de Valores al Código	29
Eventos	30
Escribiendo código en el Evento.....	31
Propiedades.....	32
Ejercicios	33
Tema 7 – Sentencias Condicionales.....	34
Variables Lógicas o Booleanas	34
Operadores relacionales o de comparación	34
Operadores Lógicos	35
Operadores Lógicos en programación.....	36
Ejemplos de evaluación de sentencias lógicas.....	36
Sentencia If	37
If – Else.....	38
If – else - else if.....	38
Algoritmos con If.....	41
Ejercicios	41
Tema 8 – Subrutinas	44
.....	44
Crear una subrutina en B4J.....	44
Ejemplo 1.....	44
La memoria de una subrutina en B4X	46
Devolución de una valor desde una subrutina	47
Ejemplo 2	47
Ejercicios	48
Tema 9 – Clases	49
Clases	49
Ejemplo de clase en B4J	50
Metodología de implementación	50
Insertar un libro	51
Mostrar libro	51
Cambiar libro	51
La rutina “Initialize”	51
Cómo usar la Clase	52
Uso de los métodos.....	52





Ejercicios	53
Tema 10 – B4XPages.....	54
La estructura de las carpetas de una aplicación	54
Iniciar una aplicación con B4XPage	55
Qué es Root.....	55
Crear una nueva B4XPage.....	56
Invocar a una nueva B4XPage	58
Cerrar una B4XPage	59
Transferir información entre páginas	60
La Vida de las B4XPages.....	61
Ejercicios	62

Tema 1 – Por qué B4X

🕒 1h

Muchos programadores nuevos se preguntan en qué lenguaje de programación invertir su tiempo y esfuerzo. Cada lenguaje de programación tiene ventajas pero también desventajas que deben tenerse en cuenta. A menudo, la selección de un idioma también determina la evolución posterior del desarrollador. Más aún, cuando se trata de utilizar el lenguaje con fines educativos, hay elementos individuales que deben tenerse en cuenta.

Así pues, el lenguaje de programación elegido debe ser:

- Moderno y estructurado.
- Ser fácil de aprender para niños y nuevos programadores.
- Proporcionar un entorno de desarrollo integrado sin confusión.
- Permitir al alumno desarrollar aplicaciones en diferentes plataformas como Windows, Android, IOS, Linux, Raspberry Pi, Arduino etc.
- Proporcionar todas las estructuras de datos modernas como listas, mapas, etc.
- Provide all modern data structures as lists, maps etc.
- Mantener el interés de los estudiantes brindándoles la posibilidad de desarrollar diferentes tipos de aplicaciones, por ejemplo, juegos, etc.

El lenguaje de programación **B4X** proporciona todas las características anteriores y también es un lenguaje para desarrollar aplicaciones de alto nivel que pueden ser un trampolín para el desarrollo profesional futuro. Además, cuenta con un público muy entusiasta que con mucho gusto ayuda en cualquier tipo de duda.

Para mas información, se puede visitar el sitio web: <https://www.b4x.com/learn.html>

Instalación de B4X

La información más actualizada sobre cómo instalar B4X está siempre aquí: <https://www.b4x.com/b4j.html>



Tema 2 – El significado de Problema

🕒 1h

Lo que los estudiantes aprenderán

- ¿Qué es un problema?
- ¿Qué son los datos?
- ¿Qué es información?
- Qué pasos deben tomarse para resolver un problema.
- Qué es el algoritmo.

El concepto de problema

Todos los días en nuestras vidas tenemos problemas. Problemas simples de la vida cotidiana y, a menudo, incluso mayores. Como problema solemos definir una situación que estamos viviendo y que nos dificulta afrontarla o solucionarla (Cambridge, 2021).

Cuando pensamos en un problema, estos son los pasos que se vuelven inconscientes en nuestra mente:

- Comprender el problema
- Buscar una solución o un conjunto de soluciones.
- Elegir la solución adecuada.
- Implementar la solución.
- Verificar si esta solución tuvo los resultados deseados.

Comprender el problema

Comprender el problema es el primer paso para diseñar y resolver un problema. Es una etapa particularmente crítica porque esto también afectará el desarrollo de la solución. Incluye los siguientes pasos:

1. Descripción del problema

La descripción del problema generalmente la hacemos nosotros mismos o el que lo tiene. En este paso es importante aclarar todos sus puntos 'oscuros' y no tener dudas sobre su redacción.

2. Encontrar los datos

Encontrar los datos significa en qué se basan estos elementos para resolver un problema. Por ejemplo, en una ecuación $ax + b = 0$, los datos son los factores a y b .

3. Encontrar lo que se pide

La información que necesitamos encontrar para solucionar el problema. Continuando con el ejemplo anterior, la información es la x de la ecuación $ax + b = 0$.

Buscando una solución o un conjunto de soluciones.

Una vez que hemos reconocido los datos y lo que se solicita, se debe buscar una solución para solucionar el problema. A menudo eso no es

fácil. Por lo tanto, es necesario buscar un método o **un conjunto lógico de pasos que conduzcan a la solución**. Estos pasos deben conducir a una solución **siempre** que surja el mismo problema y, además, deben realizarse en **un período relativamente corto** de tiempo.

En matemáticas y ciencias de la computación, un algoritmo es una secuencia finita de instrucciones bien definidas que se pueden implementar por computadora, típicamente para resolver una clase de problemas o realizar un cálculo.

Elección de la solución correcta

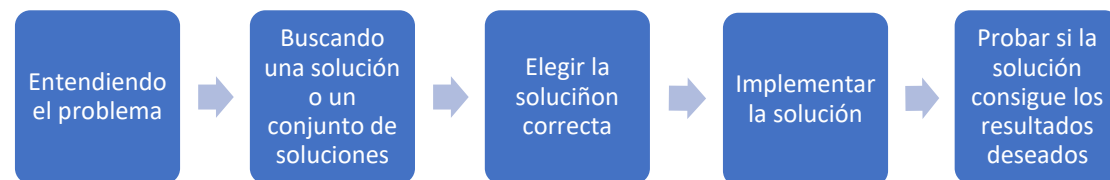
A menudo, un problema puede tener más de una solución o algunas de ellas pueden ser más eficientes que otras. Elegir la solución adecuadamente conduce a un resultado más corto y confiable.

Implementación de la solución

Una vez seleccionado el método de resolución, se deben tomar una serie de acciones para implementarlo. En otras palabras, aplicar **el Algoritmo**.

Comprobar si esta solución obtuvo los resultados deseados

Finalmente, tras aplicar la solución es necesario probar con varios datos, para examinar si conduce a resultados correctos cada vez que se realiza sin situaciones problemáticas.



Dibujo 1. Los pasos

Tema 3 – Mi primer Programa

🕒 2h

Lo que los estudiantes aprenderán

- Cómo iniciar B4J
- Cómo crear y guardar un nuevo proyecto
- Cómo ejecutar un proyecto
- Qué es una pantalla de error
- Cómo usar los comandos Turtle
- Cómo escribir un nuevo proyecto que use Turtle

Hola Mundo



Haz un programa que dibuje una línea recta con la ayuda de la tortuga en la pantalla de la computadora.

Instrucciones recomendadas para el profesor

El objetivo del ejercicio anterior es familiarizar a los estudiantes en la creación de un proyecto con B4J y tener el primer contacto con un entorno de programación. Las siguientes instrucciones no limitan en modo alguno al profesor a adaptar su curso a las condiciones educativas particulares.



Consejo para el profesor

Este es el primer tema, así que ¡haz que sea sencillo!

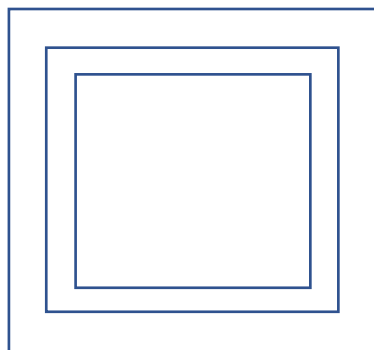
Función	Descripción
1 Cómo crear tu primer proyecto con B4J	<i>Menú Archivo -> Nuevo -> B4Xturtle</i> <i>Carpeta del proyecto</i> <i>Nombre del proyecto</i> Explique a los estudiantes la importancia de los nombres correctos en cada proyecto que crean y el valor del almacenamiento correcto en carpetas de una manera estructurada
2 Ejecutar un Proyecto	<i>Menú Proyecto -> Compilar y Ejecutar</i> Explique qué significa compilar y cómo reconocer errores de sintaxis en el registro. No es necesario que proporcione mucha información sobre la compilación. Solo lo básico para ejecutar un proyecto.
3 #Region Project Attributes	Cambie los valores en <i>MainFormWidth</i> y <i>MainFormHeight</i> para hacer una aplicación de diferente tamaño.



Función	Descripción
4 Sub Turtle_Start	¿Qué significa Sub? Una pequeña cantidad de código que está realizando una determinada actividad.
5 Métodos Turtle	¿Qué es Turtle? ¿Qué hace .MoveForward? ¿Cómo podemos encontrar más comandos? (Dícales a los estudiantes que escriban "Turtle". Para ver la lista de métodos.
6 Errores	Cómo identificar un error en la pantalla de registro (Log)

Ejercicios

1. Usa la tortuga y los métodos **MoveForward**, **TurnLeft**, **TurnRight** para dibujar un cuadrado del tamaño que quieras.
2. Usando los comandos anteriores y **PenUp**, **PenDown** y **Move** dibuja 3 cuadrados como la imagen de abajo.



3. Usando los comandos anteriores, haz el dibujo que quieras. Dale un nombre a tu dibujo y explica cómo lo hiciste.



Tema 4 - Variables y Rango

🕒 3h

Lo que los estudiantes aprenderán

- Explicar cómo un ordenador almacena datos en la RAM
- Explicar qué es una variable
- Cómo nombrar una variable
- Asignar un valor a una variable
- Usar operadores matemáticos
- Usar el comando Log para mostrar una variable

Imagínate que vives en una calle con varios millones de casas seguidas; cada casa tiene su dirección que comienza en el número 1 y termina con el número de la última casa; para poder localizar a un amigo que vive en esa calle es necesario conocer el número de la casa; así que tenemos por un lado un número de casa y por otro el amigo que vive en esa casa.



Figure 1 Computer Memory (<https://www.freepik.com>)

La memoria central de la computadora funciona de la misma manera. Hay muchas casas cada una con su dirección y un "residente" dentro de cada casa. Esta dirección se denomina **dirección de memoria** y al "residente" se le llama contenido. En el ordenador muchas veces el "residente" (que llamaremos **variable**) necesita varias casas para poder tener hueco.

Para que un programador utilice la memoria, debe conocer los datos que necesita y el tipo de datos. Estos pueden ser números enteros o reales, palabras o letras o valores lógicos (verdadero o falso). También necesita un "hogar" en la memoria de la computadora para almacenarlos representados por la dirección.

En B4X los datos se pueden almacenar en diferentes tipos como:

B4X	Tipo de dato
Boolean	Booleano o lógico (verdadero/falso)
Byte	Entero de 8 bits (de 0 a 255)
Short	Entero de 16 bits (0 a 65535)
Int	Entero de 32 bits (0 a 4.294.967.296)
Long	Entero largo de 64 bits (0 a 2^{64})
Float	Número decimal de 32 bits
Double	Número decimal de doble precisión de 64 bits

Char	Carácter ('a', '1', '%', etc.)
String	Cadena de caracteres ("hola", "adiós", "juan",...)

Tabla 1- Tipos básicos de las variables

Cada tipo necesita un espacio diferente en la memoria para almacenar su contenido.

Debido a que es difícil para el desarrollador recordar todas las direcciones de sus datos, a cada dirección se le pone un nombre. Por suerte, esto lo hace el propio lenguaje de programación y todo lo que se necesita es pensar en un buen nombre para tus datos. Por ejemplo, un dato que sea un número entero para la edad podría llamarse "edad". Ahora, hay un "hogar" llamado edad en la memoria de la computadora.



Recuerda

Las variables se utilizan para almacenar información para ser referenciada y manipulada en un programa. También proporcionan una forma de etiquetar los datos con un nombre descriptivo, para que el lector y nosotros mismos podamos entender nuestros programas con mayor claridad. Es útil pensar en las variables como contenedores que contienen información.

Cómo averiguar cuántas variables necesitas

En cualquier problema de programación que encuentre un desarrollador, debería poder ubicar los datos y la información del problema.

En programación debemos dar un nombre a todos aquellos elementos que necesitamos conocer para resolver un problema. Normalmente en un problema de programación los podemos extraer del enunciado del ejercicio con la ayuda de verbos clave como:

- Leer
- Registrar
- Preguntar
- Aceptar
- Teclear

Ejemplo 1: Escribe un programa que convierta los euros que escribimos en dólares.

Ejemplo 2: Haga un programa que acepte un número entero positivo y calcule su cuadrado, cubo y raíz cuadrada.

Información

La Información en un programa es la transformación que tenemos que aplicar a los datos para conseguir un resultado deseado. Normalmente la manera de obtener la información la podemos obtener del enunciado del problema fijándonos en palabras clave como:

- Calcular
- Mostrar



- Escribir
- Contar
- Convertir

¿Qué debemos calcular en los ejemplos anteriores?

Cómo dar nombre a las Variables

Los nombres de las variables en B4X deben seguir estas reglas:

- Deben comenzar con mayúscula o minúscula.
- Después, pueden tener dígitos o el carácter de subrayado (_).
- B4X no distingue entre mayúsculas y minúsculas.

Además, es una buena práctica poner 3 letras delante del nombre de la variable indicando el tipo de variable que es y continuar con 1 letra mayúscula y una palabra significativa. Por ejemplo:

- Dim **intEdad** as Int
- Dim **fltCantidad** as Float
- Dim **strNombre** as String

Esto te ayudará mucho cuando encuentre una variable en el código pudiendo así reconocer el tipo y el valor que almacena.

Declaración de Variables

Mi Primera Variable



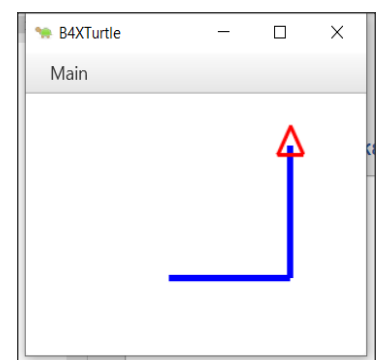
Escribe un programa que asigne un valor a un número entero y luego dibuje con la ayuda de la tortuga una línea de longitud tan larga como el valor de la variable.

En B4X, para usar una variable, primero debemos informar al lenguaje de su existencia para que le asigne espacio en la memoria del ordenador para almacenar su valor.

Por ejemplo, en el siguiente código, la declaración es la siguiente:

```
'Programa: Mi Primera Variable
Sub Turtle_Start
  Private intDistancia As Int
  Public intGiro As Int
  intDistancia = 100
  intGiro = 90

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistancia)
  Turtle.TurnLeft(intGiro)
  Turtle.MoveForward(intDistancia)
End Sub
```



La declaración de variables comienza con la palabra clave **Private** o **Public**.

Private significa que la variable se conoce solo en el espacio específico declarado y ningún otro programa o subprograma conoce su existencia y, por lo tanto, el valor que contiene.

En cambio, una declaración de variable que comienza con la palabra clave **Public** puede ser conocida por otros programas, subprogramas o clases, etc.

Después de la palabra clave **Private** o **Public** sigue el nombre de la variable. Aquí es donde se aplican las reglas discutidas anteriormente. Finalmente, sigue el tipo de variable. Para variables simples, los tipos son todos los descritos en el *Tabla 1- Tipos básicos de las variables*.



Consejo para el profesor

No es necesario que explique todas las variables ni su uso. Para que sus estudiantes comiencen a programar, los conceptos básicos de integer, float, string son suficientes. A medida que avanza en los cursos, puede incluir otros tipos según sus necesidades.

Comentarios

En programación un comentario es una explicación o anotación que se escribe en el código fuente de un programa. Se agregan con el propósito de hacer que el código fuente sea más fácil de entender para los humanos y generalmente son ignorados por compiladores e intérpretes. La sintaxis de los comentarios en varios lenguajes de programación varía considerablemente. (Wikipedia, 2021)

En B4X, los comentarios se insertan poniendo el carácter ' (comilla simple) antes del comentario. Ese carácter hace que se ignore todo lo que viene a continuación. Generalmente, los comentarios se suelen poner en lugares donde es importante recordar lo que se está haciendo. Los comentarios se distinguen fácilmente en el código por el color verde que les da el entorno de programación (IDE).

Ejemplo

```
'Programa: Mi Primera Variable
'Este programa dibuja un ángulo recto cuyos lados son tan largos
'como el valor de la variable intDistancia
Sub Turtle_Start
  Private intDistancia As Int
  Public intGiro As Int
  intDistancia = 100      'Longitud de los lados del ángulo recto
  intGiro = 90            '90 grados del ángulo recto

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistancia)
  Turtle.TurnLeft(intGiro)
  Turtle.MoveForward(intDistancia)
```

El área de log y la función log

Cuando se programa se suelen producir errores. Generalmente, los errores en la programación se dividen en dos categorías: **sintácticos** y **lógicos**. Por ahora trataremos los errores de sintaxis que son reconocidos por el lenguaje de programación y los indicaremos en la pantalla de logs. Para acceder a la pantalla de Logs, debemos hacer clic en la **pestaña de Log** en la parte inferior derecha.

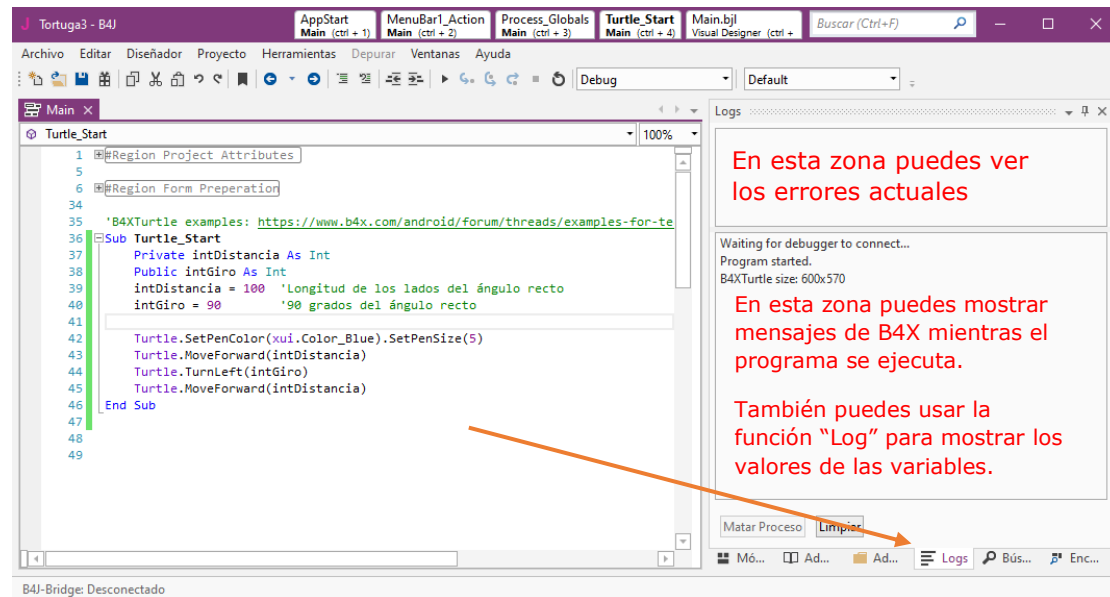


Figura 2. Pantalla de Log

La pantalla de Logs está dividida en dos zonas; en la superior se muestran los errores y en la inferior los mensajes generados por el propio lenguaje B4J o bien aquella información que nosotros hemos dicho que queremos mostrar con la función **Log()**. El uso de la función **Log ()** te ayuda a mostrar mensajes mientras se ejecuta un programa, así como valores de variables para ayudar a controlar el correcto funcionamiento del programa.

Para mostrar cualquier información en la pantalla es suficiente usar la función **log ()** como en la siguiente imagen.

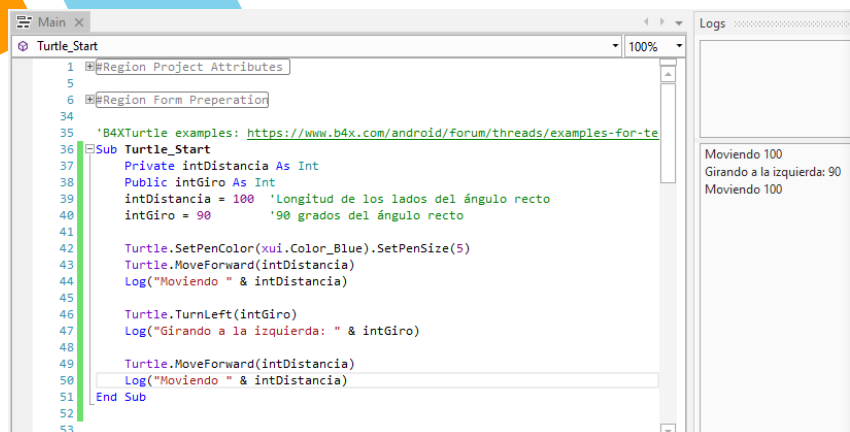


Figura 3. Uso de la función Log

Operadores matemáticos

B4X admite todas las **operaciones** matemáticas más comunes:

Operador	Ejemplo	Operación
+	$x + y$	Suma
-	$x - y$	Resta
*	$x * y$	Multipliación
/	x / y	División
Mod	$x \text{ Mod } y$	Resto de la división
Power	$\text{Power}(x,y) \quad x^y$	Potencia

Ejemplos:

```

Private intA, intB, intC, intS As Int
Private fltD, fltM As Float
intA = 40
intB = 20
intC = 30

intS = intA + intB + intC
Log(intS)           'Muestra 90

fltD = intS / 3
Log(fltD)           'Muestra 30

intA = intA + 1
Log(intA)           'Incrementar intA en 1
                    'Muestra 41

intS = Power(intA - 11, 2)
Log(intS)           ' 302
                    'Muestra 900 (30*30)

fltM = intA mod 2
Log(fltM)           'Resto de dividir 41 entre 2
                    'Muestra 1

```

Cadenas

En programación de ordenadores, una cadena es tradicionalmente una secuencia de caracteres, ya sea una constante literal o una variable. En este último caso se puede cambiar su contenido y también su longitud, o puede ser de longitud fija (después de crearla) (Wikipedia, Wikipedia - Strings, 2021).

Una cadena se declara como las otras variables usando la declaración **String**

```
Private strNombre as String
```

La asignación de valor a una cadena se puede hacer con el símbolo = o leyendo un valor que escribe el usuario (algo que veremos más adelante).

```
Private strNombre, strApellidos as String  
strNombre = "Juan"  
strApellidos = "García Gómez"
```

También podemos unir dos cadena usando el carácter &

```
Private strNombre, strApellidos as String  
strNombre = "Juan"  
strApellidos = "García Gómez"  
Private strPersona as String  
  
strPersona = strNombre & " " & strApellidos  
log(strPersona) ` Muestra Juan García Gómez en la pantalla de Log  
  
Private strNombre2 as String  
strNombre2 = "Antonio"  
strNombre2 = strNombre2 & " López"
```

También hay muchas funciones para trabajar con cadenas que son muy útiles:

CharAt (Índice)	Devuelve el carácter en el índice dado.
CompareTo (Otra)	Compara lexicográficamente la cadena con la Otra cadena.
Contains (SearchFor)	Comprueba si la cadena contiene la cadena SearchFor dada.
EndsWith (Sufijo)	Devuelve True si la cadena termina con la subcadena de sufijo dada.
EqualsIgnoreCase (Otra)	Devuelve True si ambas cadenas son iguales ignorando si están en mayúsculas o minúsculas.
Length	Devuelve la longitud, el número de caracteres, de la cadena.
Replace (Destino, Reemplazo)	Devuelve una nueva cadena resultante de la sustitución de todas las apariciones de Destino con Reemplazo.
StartsWith (Prefijo)	Devuelve True si esta cadena comienza con el prefijo dado.
ToLowerCase	Devuelve una nueva cadena que es el resultado de ponerla en minúsculas.
ToUpperCase	Devuelve una nueva cadena que es el resultado de ponerla en mayúsculas.
Trim	Devuelve una copia de la cadena original sin espacios en blanco iniciales o finales.

Tabla 2. Funciones de cadena (<https://www.b4x.com/android/documentation.html>)



Consejo para el profesor

Puedes encontrar más información sobre manipulación de cadenas en los folletos sobre el lenguaje en:

<https://www.b4x.com/android/documentation.html>

Ejercicios

1. En los siguientes ejercicios, identifica las variables que necesita declarar. Para cada uno de ellas, escriba la declaración y asígnele un nombre apropiado.
 - Calcula el volumen de un cilindro con un radio de un metro y una altura de dos metros.
 - Haga un programa que acepte un número entero positivo y calcule su cuadrado, cubo y raíz cuadrada.
 - Haga un programa que lea una suma de dinero en € y calcule y muestre la cantidad correspondiente en \$.
 - Escriba un programa que lea la longitud de los lados de un rectángulo desde el teclado y calcule y muestre su área.
 - La resistencia total R de dos resistencias $R1$ y $R2$ conectadas en serie es $R1 + R2$ y paralelo $(R1 * R2) / (R1 + R2)$ respectivamente. Crea un programa que lea dos valores de resistencia $R1$ y $R2$ y calcule la resistencia total en serie y en paralelo y la muestre con la función "Log".

2. En los siguientes nombres de variables, seleccione cuáles son correctas y cuáles no:

Nombre	Correcto	Incorrecto
int Edad	<input type="checkbox"/>	<input type="checkbox"/>
_fltCantidad	<input type="checkbox"/>	<input type="checkbox"/>
strNombre	<input type="checkbox"/>	<input type="checkbox"/>
1miEdad	<input type="checkbox"/>	<input type="checkbox"/>
int_valor	<input type="checkbox"/>	<input type="checkbox"/>

3. Es el final del trimestre y obtuviste tus calificaciones de tres clases: Geometría, Álgebra y Física. Crea un programa que:
 - Guarde en 3 variables las calificaciones de estas 3 clases (las calificaciones van de 0 a 10)
 - Calcule la nota media de sus calificaciones y la muestre con la función "Log".



4. ¡Has comprado un Bitcoin y ahora está subiendo! Crea un programa que:

- Asigne el valor del bitcoin en el momento de la compra.
- Asigne el porcentaje de aumento (o disminución)
- Emplee la función "Log" para ver el valor total de tu bitcoin.
- Emplee la función "Log" para ver el valor de aumento o disminución.

5. Tienes una casa y deseas calcular su área total. Crea un programa que:

1. Lea el ancho y el alto en dos variables.
2. Calcule y muestre con la función "Log" el área total.

6. Quieres comprar un nuevo portátil. Miras el precio y ves que el precio es de 300 euros sin incluir el IVA del 21%. Crea un programa que:

1. Asigne el precio del portátil en una variable.
2. Asigne el porcentaje de impuestos en una segunda variable.
3. Calcule y muestre con la función "Log" el precio final con IVA.

7. En una empresa, el salario mensual de un empleado se calcula partiendo del salario mínimo de 400 € al mes, más 20€ multiplicado por el número de años empleados, más 30€ por cada hijo que tenga. Cree un programa que:

1. Asigne el número de años del empleado en una variable.
2. Asigne el número de hijos que tiene el empleado en la segunda variable.
3. Calcule y muestre con la función "Log" el salario del empleado.

8. Crea un programa que use la función "Log" para mostrar el último dígito de un número entero dado.

9. Crea dos variables 'a' y 'b' y ponles un valor inicial diferente a cada una. Escribe un programa que intercambie ambos valores.

Ejemplo: a = 10, b = 20

Salida: a = 20, b = 10

10. Crea dos variables 'a' y 'b' y ponles un valor inicial diferente a cada una. Escribe un programa que doble el valor de la variable 'a' e incremente el valor de la variable 'b' en 1.

Ejemplo: a = 10, b = 20

Salida: a = 20, b = 21



Lo que los estudiantes aprenderán

- Hablar sobre el diseñador
- Diseñar la primera pantalla.
- Insertar y personalizar vistas: etiquetas, campos de texto, botones, paneles
- Guardar formularios
- Diseñar tu propia pantalla principal utilizando esquemas

Hasta ahora, hemos usado la tortuga para moverla por la pantalla y la función “Log” para mostrar información en la pantalla de Log de B4X. ¿Qué pasa si le pide al usuario que introduzca un valor? ¿O qué sucede cuando se desea mostrar información al usuario? B4X tiene un entorno de diseño de interfaces de usuario especial. A través de él puedes diseñar el aspecto de las pantallas y comunicarte con los usuarios de tu aplicación.

Cada vez que debas diseñar una aplicación, debes tener en cuenta que el aspecto visual de tu aplicación es lo que atraerá a los usuarios. Es decir, no basta con que haga lo que tiene que hacer, sino que tiene que ser fácil de usar y ofrecer información de forma organizada sin confundir.

Antes de diseñar una aplicación, ten en cuenta estos principios de diseño (usability.org, 2021):

Manten la interfaz simple. Las mejores interfaces son casi invisibles para el usuario. Evitan elementos innecesarios y son claros en el lenguaje que utilizan en las etiquetas y en los mensajes.

Sea coherente y utilice elementos concidos en la interfaz de usuario. Al utilizar elementos comunes en su interfaz de usuario, los usuarios se sienten más cómodos y pueden hacer las cosas más rápidamente.

Usa color y textura correctamente. Puedes dirigir o desviar la atención sobre los elementos usando el color, la luz, el contraste y la textura.

Utilice la tipografía para crear jerarquía y claridad. Considere cuidadosamente cómo usa la tipografía. Diferentes tamaños, fuentes y disposición del texto para ayudar a aumentar la escalabilidad y la legibilidad.

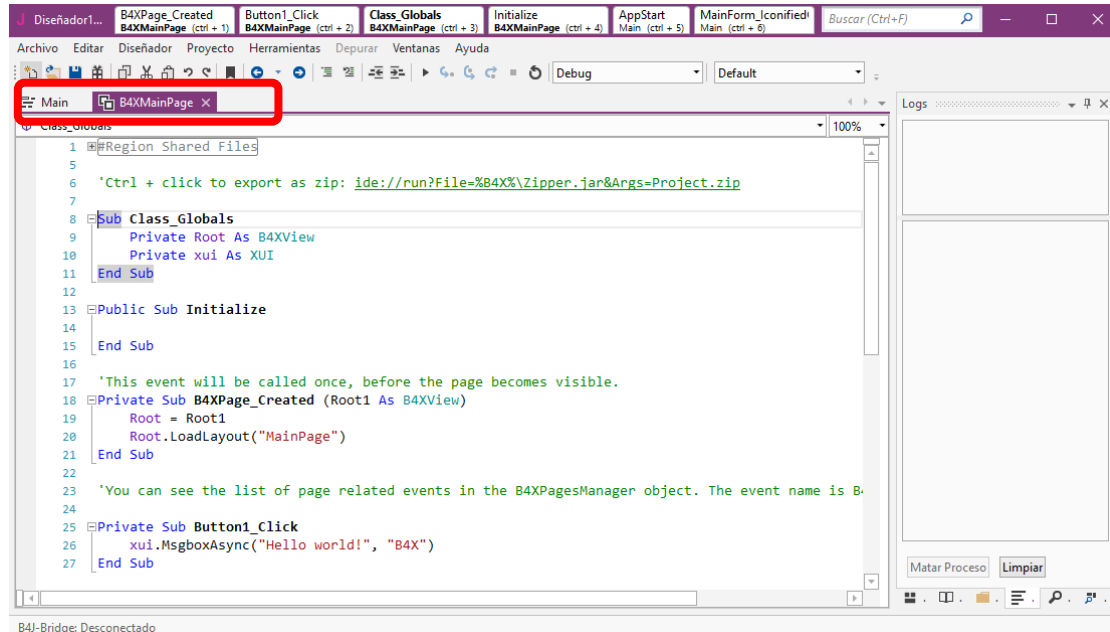
Asegúrese de que el programa comunique lo que está sucediendo. Informe siempre a sus usuarios sobre la ubicación, las acciones, los cambios de estado o los errores.

Piensa en los valores por defecto. Al pensar detenidamente y anticiparse a lo que hacen tus usuarios, puedes crear valores por defecto representativos. Esto es muy importante cuando se trata del diseño de formularios, ya que los campos pueden estar rellenos previamente.



Primeros pasos en el diseño

En primer lugar, arranca B4J y desde el menú de **Archivo**, elija **Nuevo** y **B4XPages**. Elija un directorio y escriba un nombre para su proyecto. Verá el código a continuación. Verás dos pestañas de código, la primera llamada **Main** y la segunda **B4XMainPage**.



No te preocupes por estas pestañas ahora. Hablaremos más adelante sobre ellas. ¡Ahora todo lo que necesitas saber es que dentro de B4XMainPage suceden todas las cosas interesantes de nuestro código!

Ahora desde el menú **Diseñador** elige **Abrir Diseñador Interno**.

Aquí comienza el proceso de diseño. Se abrirán dos ventanas, la primera es el diseñador y la segunda es la vista previa de la pantalla que se está diseñando.

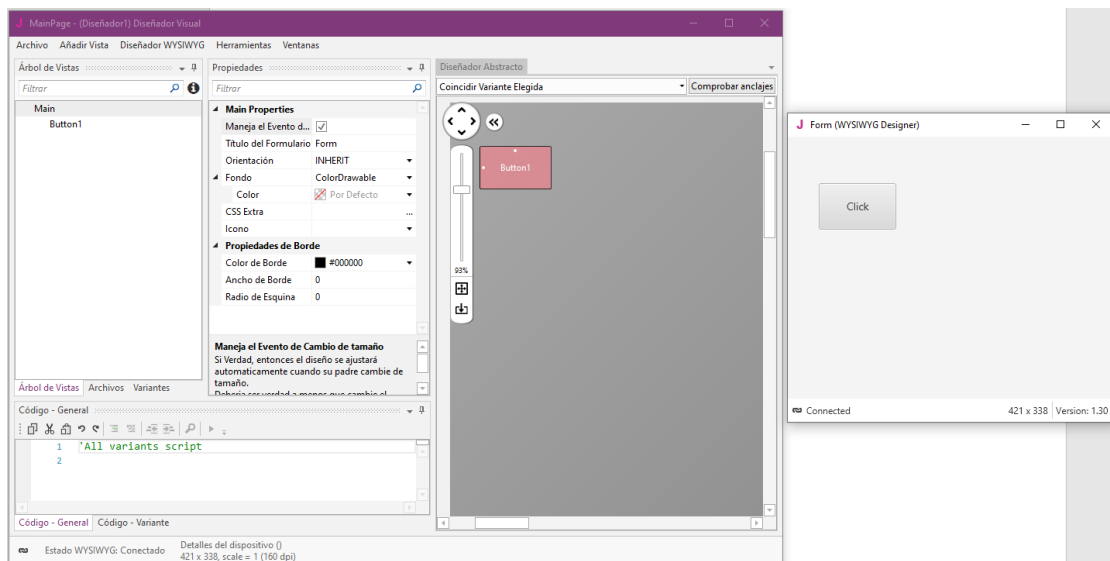


Figura 2. La pantalla del Diseñador

Diseñador visual

El menú **Añadir Vista** incluye todos los objetos necesarios para crear nuestra pantalla.

Elige la opción **Label** desde el menú **Añadir Vista** y muévelo dentro de la ventana **Diseñador Abstracto** donde quieras:



Recuerda

Puedes mover todos los objetos seleccionándolos y arrastrándolos con el ratón.

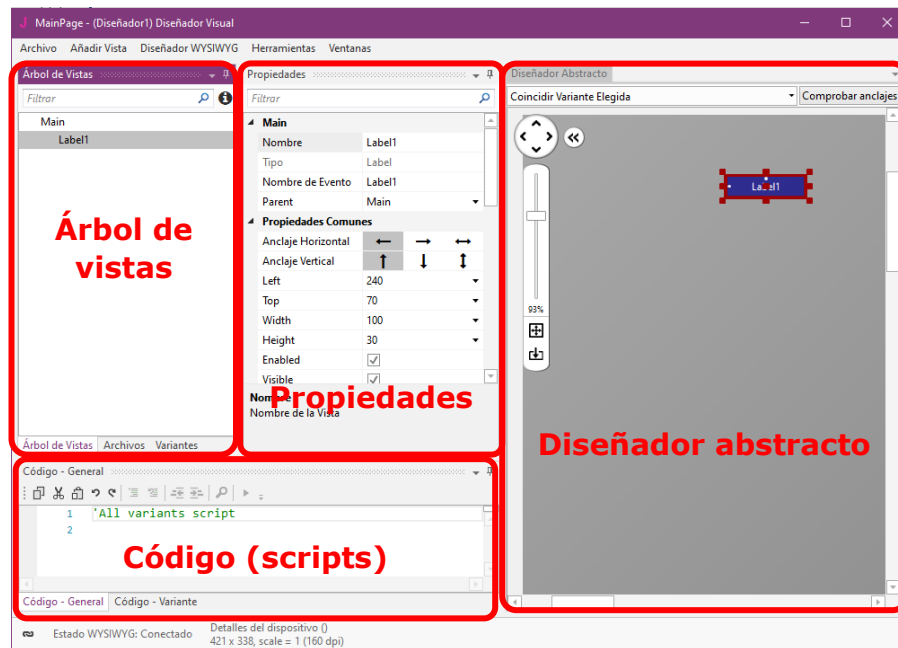


Figura 3. Zonas del Diseñador

El Árbol de Vistas

Aquí puedes ver todos los objetos de tu diseño. Ten en cuenta que los objetos se dibujan en pantalla empezando por el primero, con lo que los que aparecen en primer lugar quedan ocultos por los de más abajo:

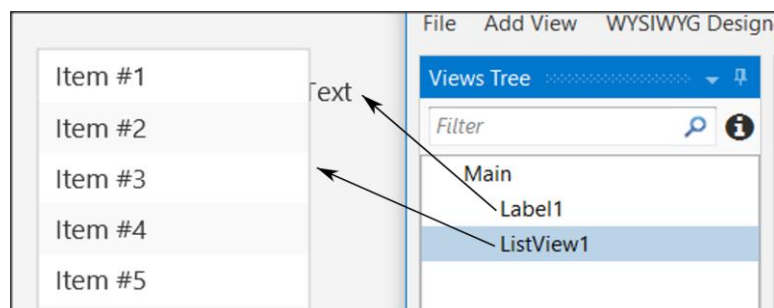


Figura 4. Árbol de vistas. Fíjate que "Label1" queda por debajo de "ListView1"

Propiedades

Cada objeto tiene sus propiedades como tamaño, posición en pantalla, colores, tipo de letra, etc. Cada propiedad se puede cambiar dentro de la ventana de propiedades o posteriormente a través del código del programa.

Una de las propiedades más importantes es el **nombre del objeto**. Al igual que con las variables, debes seguir unas reglas para indicar su tipo. Por ejemplo, en la *Tabla 3. Nombrar objetos* mostramos algunos casos:

Tipo	Prefijo	Ejemplo
Label	lbl	lblNombre
Button	btn	btnGuardar
TextField	txt	txtEdad
Spinner	spn	spnAños
Pane	pn	pnLínea1

Tabla 3. Nombrar objetos

Diseñador Abstracto

El Diseñador Abstracto permite seleccionar la posición y cambiar el tamaño de las Views (vistas). Es una función muy útil para colocar rápidamente objetos en la posición correcta (sin embargo, para una ubicación más precisa es mejor usar la pestaña Propiedades).

Ejemplo 1



Imagina que quieres hacer un programa que lea en pantalla dos enteros y que calcule y muestre su suma.

Decidir el tamaño de la ventana de tu aplicación

Dependerá de la cantidad de información que debemos mostrar, así como de los elementos individuales como menús, gráficos, etc.

El tamaño de la aplicación debes cambiarlo **antes de iniciar el Diseñador**. Para hacerlo, vete a la pestaña **Main** y cambia las primeras líneas del código Width y Height:

```
#Region Project Attributes
#MainFormWidth: 600
#MainFormHeight: 400
#End Region
```

Guarda tu proyecto y abre el **Diseñador Interno**.

Establecer una variante adecuada

Una variante es una versión de tu aplicación con una resolución diferente a la que tienes por defecto.

Por lo general, debes establecer las variantes con los parámetros **MainFormWidth** y **MainFormHeight**. Esto te ayudará a diseñar sin arriesgarte a salirte de la pantalla.

Para crear una variante, elige la pestaña **Variantes** y luego **Nueva variante** y escribe el ancho y el alto.

Puedes tener tantas variantes como quieras para diferentes tamaños de pantalla, pero por ahora, sólo usaremos una. Puedes eliminar cualquier variante seleccionándola y eligiendo **Eliminar Seleccionado**.

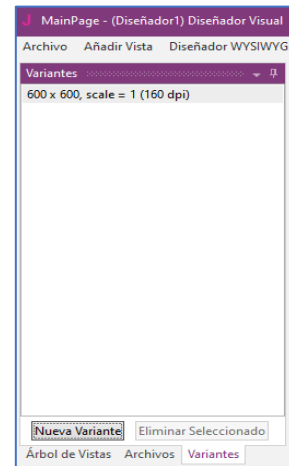


Figura 5. Pantalla de Variantes

Diseña un esquema de tu pantalla

Para aplicaciones pequeñas, este paso es opcional, pero es una buena costumbre haber decidido desde el principio dónde quieres mostrar tus datos. Puede utilizar una simple hoja de papel o varios programas para ayudar a crear esquemas.

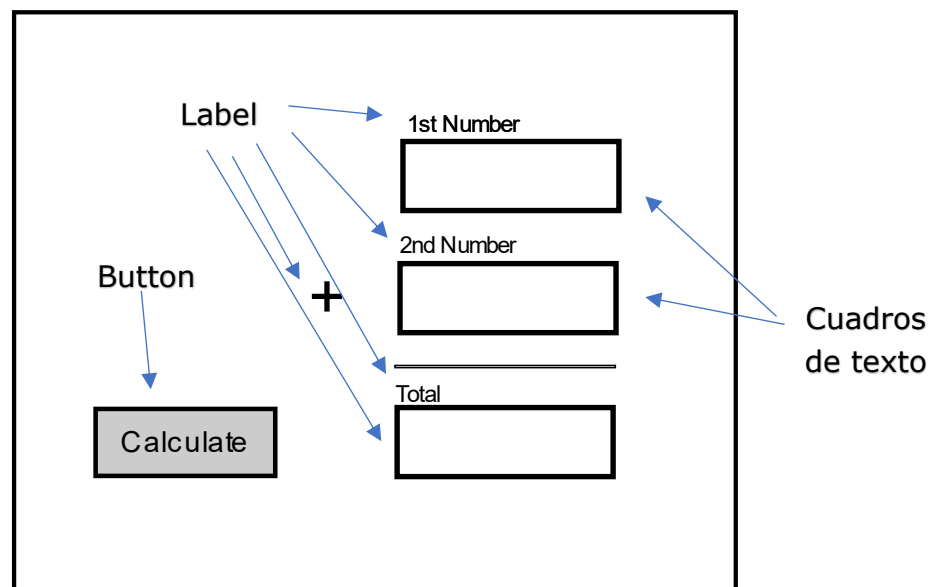


Figura 6. Esquema

Crear las Vistas

Ahora que ya sabes lo que necesitas y dónde colocarlo, vamos a usar las herramientas del Diseñador para completar el diseño de la interfaz.

Insertar una etiqueta (label)

En el menú **Añadir Vista**, seleccionamos **Label** y aparecerá un objeto de etiqueta en tu **Árbol de vistas** y en el **Diseñador Abstracto**. Muévelo en el lugar que quieras según tu esquema de la pantalla y elige un nombre adecuado dentro de Propiedades.

Ahora desplázate hacia abajo en Propiedades y fija estos valores:

- **Width:** 180
- **Height:** 30
- **Text:** Primer Número
- **Alineación:** CENTER_LEFT
- **Font:** SansSerif
- **Tamaño:** 13

Experimenta las propiedades y observa cómo queda en el panel de previsualización.

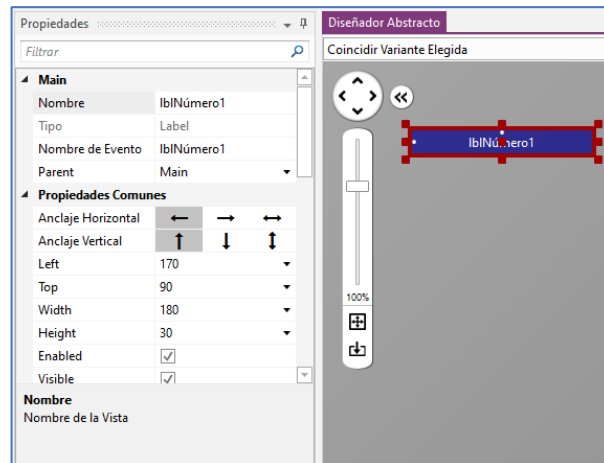


Figura 7. Etiquetas

Inserta una segunda etiqueta o duplica la primera. Selecciónala y presiona Ctrl-D. El segundo método genera una segunda etiqueta exactamente igual a la primera, excepto en la propiedad de nombre. Fija el nombre en "lblNúmero2" y "Segundo número" como **Text**. Crea una tercera etiqueta con el nombre "lblTotal" y **Text**: "Total".

Insertar un campo de texto (TextField)

Los campos de texto se utilizan para introducir datos al programa. No hay restricciones sobre el tipo de datos que puede leer. Desde el menú **Añadir Vista**, elige **TextField** y configúralo así:

- **Nombre:** txtNúmero1
- **Width:** 180
- **Height:** 40
- **Font:** SansSerif
- **Bold:** marcado

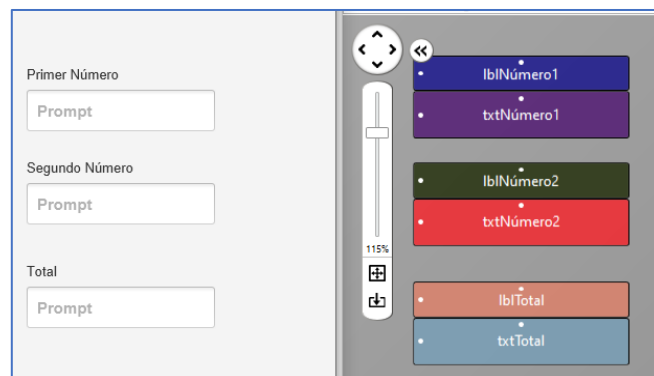


Figura 8. Campos de texto

Coloca el TextField debajo de la etiqueta "Primer número". Crea otro TextField con el nombre "txtNúmero2" y colócalo debajo de la etiqueta "Segundo número". Al final, crea un tercer TextField con el nombre "txtTotal" y colócalo debajo de la etiqueta "Total". Debe quedarte also similar a la **Figura 8. Campos de texto**

Insertar un botón (button)

Los botones se utilizan para realizar acciones. El programa detecta el clic y luego ejecuta los comandos apropiados según el botón presionado.

Para cada botón, puedes configurar diferentes propiedades como tamaño, color, forma, etc. para que se destaque en pantalla y los usuarios lo vean fácilmente.

Para añadir un botón, haz clic en **Añadir Vista**, elige **Button** y déjalo así:

- **Nombre:** btnCalcular
- **Width:** 150
- **Height:** 40
- **Color de borde:** #3C0000
- **Ancho de borde:** 2
- **Radio de esquina:** 20
- **Text:** Calcular
- **Color de texto:** #FF3C0000
- **Font:** SansSerif
- **Tamaño:** 15
- **Bold:** marcado

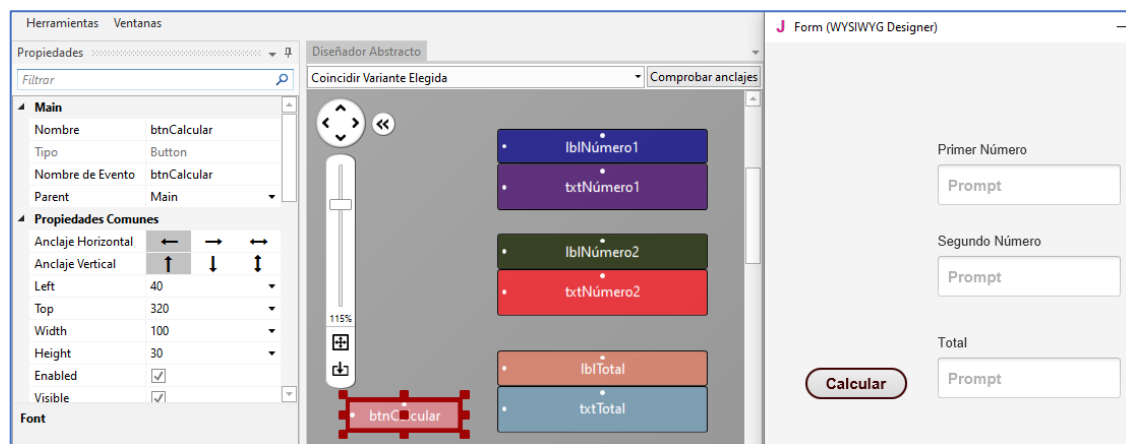


Figura 9. Botones



Recuerda

Archivo -> Salvar (o Ctrl – S) cada vez que hagas algo importante.

Insertar un panel

Puedes usar un panel para agrupar visualmente objetos en la pantalla. El panel está rodeado de un marco y tiene propiedades como el color, borde, relleno, etc. También puedes ponerle una altura muy pequeña (1 o 2) para que sea una línea en pantalla.

En este ejemplo usaremos un panel para dibujar una línea antes del total. Haz clic en **Añadir Vista**, elige **Pane** y configúralo así:

- **Name:** pnLínea
- **Width:** 180
- **Height:** 1
- **Color de borde:** #000000
- **Ancho de borde:** 2

Ahora, elige la opción **Archivo→Salvar** para guardar el formulario. Normalmente se le asigna el nombre **"MainPage"** indicando que es la "página principal" de tu programa. No hace falta que le pongas otro nombre.

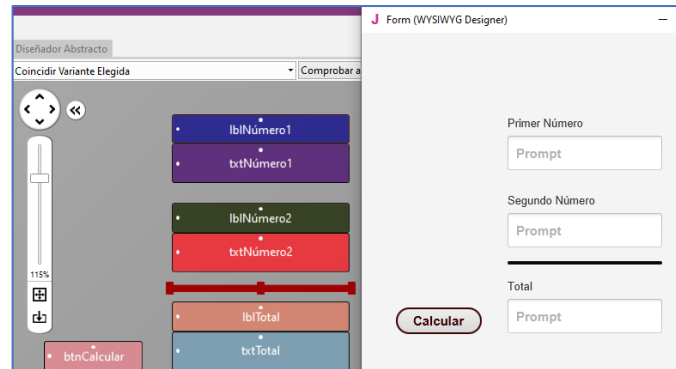


Figura 10. Panel

Ejercicios


1. Usa el diseñador para crear los siguientes esquemas.



Consejo para el profesor

Esta es la parte divertida. Deja que los pupilos experimenten libremente con las vistas.

 The diagram shows a form layout within a rounded rectangle. It has four input fields arranged in a 2x2 grid: 'Nombre' (top-left), 'Clase' (top-right), 'Apellidos' (bottom-left), and 'Edad' (bottom-right). Below these fields are three buttons: 'Salvar' (left), 'Borrar' (middle), and 'Cancelar' (right).



Calculadora de nota media			
Matemáticas	<input type="text"/>	Música	<input type="text"/>
Física y Quím.	<input type="text"/>	Informática	<input type="text"/>
Lengua	<input type="text"/>	Ed. Física	<input type="text"/>
Inglés	<input type="text"/>	Geo. e Historia	<input type="text"/>
Nota media		<input type="text"/>	<input type="button" value="Calcular"/>

11. Piensa y diseña tu “Aplicación Soñada”. Dale un nombre, crea un esquema de su interfaz de usuario en tu ordenador y crea la Vista de Diseño.

Tema 6 – Del Diseñador al Código

🕒 2h

Lo que los estudiantes aprenderán

- Class_Globals
- Variables y subrutinas (Subs)
- Paso de Valores al Código
- Eventos
- Atributos

Diseñar la apariencia de la aplicación en el Diseñador es el primer paso al crear una aplicación. A menudo, los nuevos programadores recurren a él para rediseñar, corregir o agregar información individual.

Cuando la pantalla está lista, el desarrollador pasa a la siguiente etapa que consiste en programar las funciones. Es decir, hay que hacer que todos los elementos que se incluyeron en el diseño realicen sus funciones correctamente. Por ejemplo, los cuadros de texto deben poder leer datos, los botones deben lanzar funciones, las listas deben mostrar datos, etc.

Class_Globals

Al comienzo del código en la pestaña **B4XMainPage** hay un conjunto de declaraciones de variables entre **Sub Class_Globals** y **End Sub**.

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11 End Sub
```

Imagen 2. Sub Class_Globals

Como se vio en el tema 3, una Sub (subrutina) es un trozo de código que realiza una operación específica. Dentro de **Class_Globals** se recogen las declaraciones de variables que queremos que se

conozcan a lo largo del código de la pestaña **B4XMainPage**, es decir, en cada subrutina.

Además, si una sentencia de declaración de variables empieza con la palabra **Public**, entonces estará disponible para cualquier otra "pestaña".



Un estudio más profundo del uso de variables
En el siguiente código puedes ver 3 subrutinas:

B4XMainPage

```
6  @Sub Class Globals
7      Private Root As B4XView
8      Private xui As XUI
9      Private intNumber As Int = 32
10     Private intNewTotal As Int
11 End Sub
12
13 @Public Sub Initialize
14 End Sub
15
16 @Private Sub B4XPage_Created (Root1 As B4XView)
17     Root = Root1
18     Root.LoadLayout("MainPage")
19     Private intN1, intN2, intTotal As Int
20     intN1 = 45
21     intN2 = 32
22     intTotal = intN1 + intN2
23     Log("Total = " & intTotal)
24
25     intNewTotal = intTotal + intNumber
26     Log("New Total = " & intNewTotal)
27 End Sub
28
29 @Sub Button1_Click
30     xui.MsgboxAsync("Hello world!", "B4X")
31     Private intN1 As Int = 5
32     intNewTotal = intN1 + intNumber
33     Log("New Total = " & intNewTotal)
34 End Sub
```

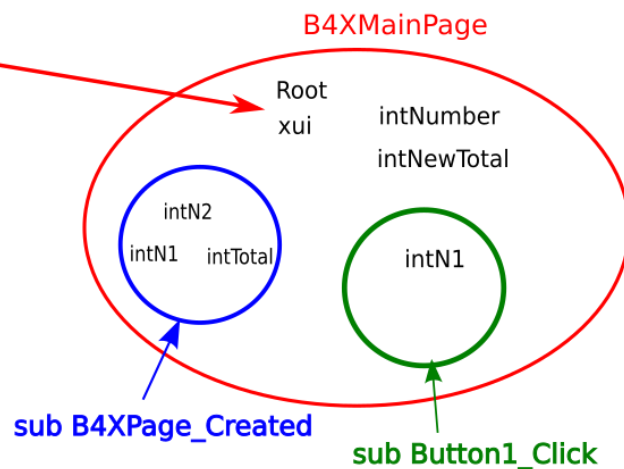


Imagen 3. Variables y ámbitos

Las variables **intNumber**, **intNewTotal**, **Root** y **xui** declaradas dentro de **Class_Globals** "viven" dentro del módulo **B4XMainPage**. Se trata de variables de ámbito global que pueden ser accedidas desde cualquier subrutina dentro de la pestaña **B4XmainPage**.

Por el contrario, las variables **intN1**, **intN2**, **intTotal** (color azul) "viven" dentro de la subrutina **B4XPage_Created** y ninguna otra subrutina podrá usarlas. Al mismo tiempo, la variable **intN1** declarada dentro del botón **Button1_Click** **no es la misma** que la que hay dentro de **B4XPage_Created**. Ambas tienen su propia zona de memoria y pueden tener el mismo nombre por estar en subrutinas distintas.



Consejo para el profesor

El uso y el ámbito de las variables son conceptos que confunden a los nuevos programadores. En función de tu alumnado, sería útil usar ejemplos que les resulten familiares.

Paso de Valores al Código

En la pantalla que has diseñado en el tema 5 hay objetos (etiquetas, textos, botones, etc.) que hay que declarar dentro de **Class_Globals** antes de usarlos en tu programa.



Algunos de los objetos de la pantalla no hay que incluirlos en el código porque no se van a usar. Por ejemplo, ninguna de las “etiquetas” va a ser gestionada en el código. Sin embargo, los botones y los campos de texto (TextFields) sí que van a ser modificados en el código.

Hay dos formas de insertar objetos en el código. La primera se hace fácilmente a través del **Diseñador** siguiendo estos pasos:

- Vamos a **Diseñador** → **Abrir Diseñador interno**
- Elegimos dentro del menú **Herramientas** la opción **Generar Miembros**.

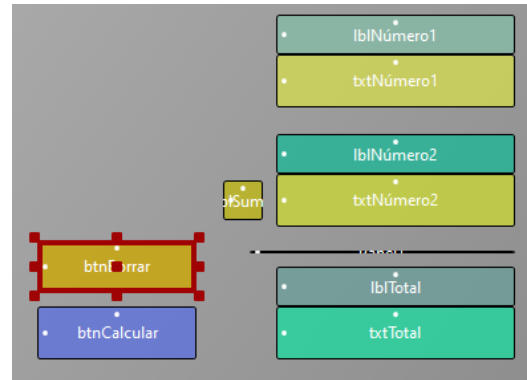


Imagen 4. Ejemplo 1. Diseñador Abstracto

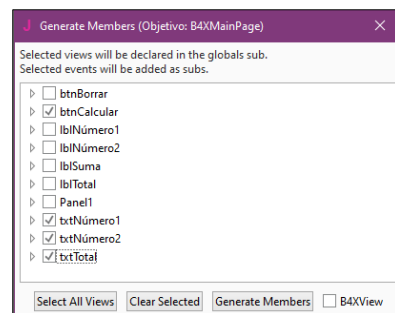


Imagen 5. Generate Members

Dentro de la pantalla de **Generate Members** hacemos clic en los objetos:

- btnCalcular
- txtNúmero1
- txtNúmero2
- txtTotal

y pulsamos en **Generate Members**.

Tu código en la subrutina **Class_Globals** se actualizará automáticamente con las nuevas variables.

```

8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11    Private txtTotal As TextField
12    Private btnCalcular As Button
13    Private txtNúmero1 As TextField
14    Private txtNúmero2 As TextField
15 End Sub

```

Imagen 6. Class_Globals



Recuerda

Cada objeto que importamos es de un tipo concreto que coincide con el tipo de las variables que se han creado

La segunda opción consiste en escribir las variables uno mismo, prestando atención a que los nombres de los objetos de la pantalla sean los mismos que los que escribes como variables (y también su tipo: Button, TextField, etc).

Eventos

Tras declarar las variables para los objetos, el último paso es activar las funciones del formulario.



Esto dependerá de lo que haga tu aplicación. En el ejemplo que estamos viendo hay un botón llamado **Calcular** que lo que hará es sumar los dos números de la pantalla y mostrará el resultado en pantalla.

La operación de suma es disparada por un proceso llamado **Evento**. El programador debe detectar el evento de pulsar en el botón **Calcular** para realizar las operaciones necesarias para mostrar el resultado.

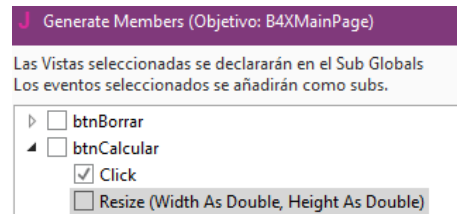


Recuerda

Hay miles de eventos que pueden dispararse en una aplicación; es tu responsabilidad como programador decidir cómo responder a ellos.

La detección de un evento es fácil y sólo hay que crear una nueva subrutina con el nombre del evento. Esto se puede hacer al mismo tiempo que se realizó la declaración de las variables a través del **Diseñador**.

- Vamos a **Diseñador** → **Abrir Diseñador interno**
- Elegimos dentro del menú **Herramientas** la opción **Generar Miembros**.
- Dentro de la pantalla de **Generate Members** desplegamos la lista que hay en el botón **btnCalcular**.
- Marcamos el evento **Click** y de nuevo pulsamos en **Generate Members**.



Ahora aparecerá una **nueva subrutina** para el evento **btnCalcular_Click** donde podrás escribir el código necesario:

```
29 Private Sub btnCalcular_Click
30
31 End Sub
```

Imagen 7. El evento Click

Escribiendo código en el Evento

Dentro de la subrutina de un Evento se escriben todas las acciones que deben lanzarse cuando se produce ese evento.

```
29 Private Sub btnCalcular_Click
30     txtTotal.Text = txtNúmero1.Text + txtNúmero2.Text
31 End Sub
```

Imagen 8. Realizar cálculos con los TextFields

En la Imagen 9 estamos haciendo lo siguiente:

El contenido del TextField llamado txtTotal será igual a la suma de los contenidos de los TextFields txtNúmero1 y txtNúmero2



Propiedades

Cada objeto que insertar en tu código tiene varias propiedades. Por ejemplo, puede tener un color, un tamaño, una posición en pantalla, un contenido, etc. Estas propiedades pueden leerse o modificarse dentro de tu código. Por ejemplo, la propiedad **txtNúmero1.Text** nos dice qué contiene el campo de texto **txtNúmero1**; pero también nos permite modificar su contenido dependiendo de cómo la usemos.

Las propiedades de un objeto podemos averiguarlas simplemente escribiendo el nombre del objeto y poniendo un punto (.) a continuación del nombre. Entonces, nos aparecerá una lista de todas las propiedades (ver Imagen 10).

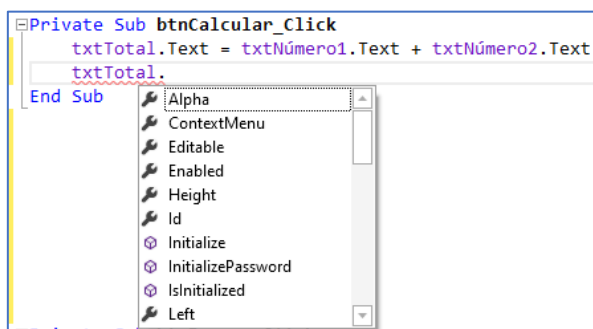


Imagen 9. Lista de propiedades de un objeto

Imaginemos que queremos crear una nueva función en la calculadora que borre el formulario para escribir nuevos números. Las acciones que tienes que hacer son las siguientes:

- Abrir el **Diseñador** y añadir un nuevo botón llamado **btnBorrar**.
- Crear una variable dentro de **Class_Global** para acceder a ella.
- Crear el evento **btnBorrar_Click**.



Recuerda

Puedes hacer operaciones matemáticas con cadenas de texto **sólo si contienen números**.

- Dentro del evento, fija la propiedad **Text** de **txtNúmero1**, **txtNúmero2** y **txtTotal** al valor "" (cadena vacía).

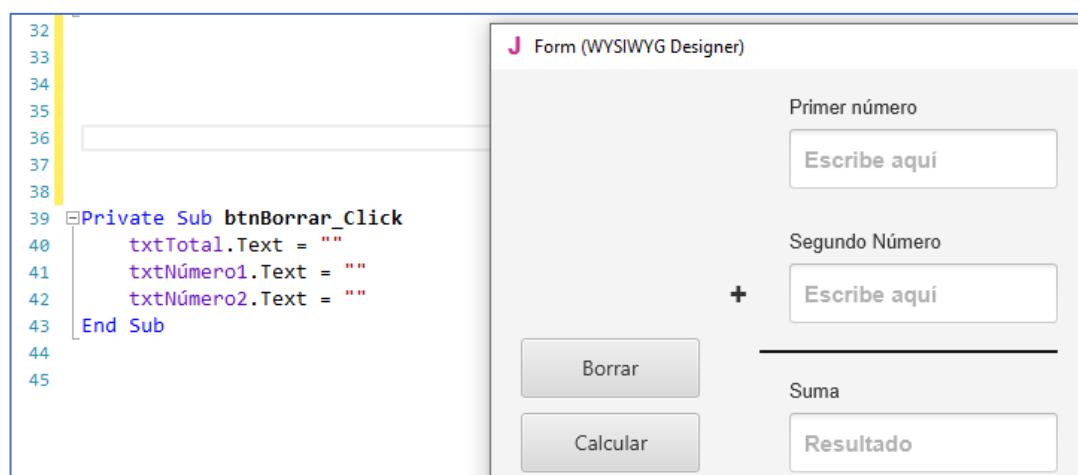


Imagen 10. Botón de Borrar y código asociado

Cada vez que pulses el botón **Borrar**, se borrará el formulario.



Ejercicios

1. Amplía el ejemplo visto en clase para que realice las 4 operaciones: suma, resta, multiplicación y división cuando se pulse el botón adecuado. Añade los objetos necesarios en el diseñador y completa el código.
2. En el ejercicio 2 del tema anterior, continúa y completa la aplicación de calcular notas medias para que el botón "Calcular" calcule la nota media de todas las materias.

Calculadora de nota media			
Matemáticas	<input type="text"/>	Música	<input type="text"/>
Física y Quím.	<input type="text"/>	Informática	<input type="text"/>
Lengua	<input type="text"/>	Ed. Física	<input type="text"/>
Inglés	<input type="text"/>	Geo. e Historia	<input type="text"/>
Nota media		<input type="text"/>	<input type="button" value="Calcular"/>

Debes comprobar que las notas de cada materia están entre 0 y 10. Si no, deberías mostrar un mensaje de error.

Tema 7 – Sentencias Condicionales

🕒 4h

Lo que los estudiantes aprenderán

- Variables Booleanas (o lógicas)
- Operadores relacionales
- Operadores lógicos
- Sentencia If
- Sentencia If-Else
- Sentencia If-Else-Else If
- Algoritmos para calcular el Máximo

Variables Lógicas o Booleanas

Anteriormente hemos visto tres tipos de variables: entero, decimal y cadena. Las variables lógicas son un tipo de dato muy sencillo, ya que sólo admiten dos valores posibles: True o False (Verdadero o Falso). Internamente el ordenador las representa como un 1 o un 0 (si hay corriente eléctrica o no).

La declaración de una variable lógica (Booleana) se hace igual que con los otros tipos de datos ya vistos:

```
Private intDistancia = 100, intTotalViaje = 0 As Int
Private blnIndicador As Boolean = False
Private blnHecho As Boolean
```

Operadores relacionales o de comparación

Los operadores relacionales se usan para realizar comparaciones entre valores. Son los mismos que en matemáticas, aunque en informática se escriben de una forma un poco diferente:

Símbolo matemático	B4X	Significado
=	=	Igual a
≤	<=	Menor o igual a
≥	>=	Mayor o igual a
≠	<>	Distinto
<	<	Menor que
>	>	Mayor que

En general, para hacer una comparación debes **comparar variables del mismo tipo**. Por ejemplo, enteros con enteros, decimales con decimales, cadenas con cadenas, etc. Sin embargo, en B4X puedes comparar también enteros con decimales y cadenas con números porque internamente se realiza la conversión de cadenas a números.

```

38 Private intDistancia = 100 As Int ' Fijate en la diferente forma de declarar
39 Private intTotalViaje As Int = 0 ' dos enteros
40 Private fltD As Float = 100.45
41 Private strN As String = "100"
42 Private s As String = "Colegio"
43
44 Log( fltD > intDistancia) 'Muestra True
45 Log( strN = intDistancia) 'Muestra True
46 Log( strN = intTotalViaje) 'Muestra False
47 Log( s = intTotalViaje) 'Muestra False
48 Log( intTotalViaje <> strN ) 'Muestra True

```

Imagen 11. Comparación de Variables



Recuerda

El resultado de una comparación es siempre un valor lógico (True o False)

Operadores Lógicos

Piensa en una afirmación como "Voy al colegio ahora". ¿Es cierta o falsa?

El matemático George Boole creó un álgebra basada en sentencias lógicas.

En el álgebra booleana los valores de las variables son verdadero o falso. Normalmente se representan por un 1 o un 0, respectivamente. Al contrario que con el álgebra a la que estás acostumbrado, en el álgebra booleana hay 3 operaciones posibles: Y, O y Negación (en inglés, **AND**, **OR** y **NOT**).



Recuerda

AND (conjunción), indicado como **x AND y**. El resultado será 1 si **x** e **y** valen los dos 1. En otro caso, el resultado será 0.

OR (disyunción), indicado como **x OR y**. El resultado será 0 si **x** e **y** valen 0. En cualquier otro caso, el resultado es 1.

NOT (negación), indicado como **NOT x**. El resultado será 0 si **x** valía 1, o bien 1 si **x** valía 0.

Ejemplo:

1ª Afirmación: *Está diluviando,*

2ª Afirmación: *Voy al colegio ahora*

Está diluviando (P1)	Voy al colegio ahora (P2)	P1 AND P2	P1 OR P2	NOT P1
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True



En la tabla podemos ver:

- Dos afirmaciones que al unirse con el operador **AND** dan como resultado un valor cierto sólo si las dos eran ciertas.
- Dos afirmaciones que al unirse con el operador **OR** dan como resultado un valor cierto si alguna de las dos eran ciertas.
- El operador NOT que invierte la veracidad o falsedad de una afirmación.

Operadores Lógicos en programación

Los operadores lógicos se usan en programación para crear expresiones de comparación complejas. Esto permite al programador optimizar su código con menos líneas y con un código más sencillo.

En B4X las variables lógicas se usan así:

```
50 'Operadores lógicos
51 Private blnL1, blnL2 As Boolean
52 blnL1 = True
53 blnL2 = False
54
55 Log (blnL1 And blnL2) ' Muestra False
56 Log (blnL1 Or blnL2) ' Muestra True
57 Log (Not(bl nL1)) ' Muestra False
58 Log (Not(bl nL2)) ' Muestra True
```

Imagen 12. Ejemplos de uso de variables lógicas

Fíjate que los operadores lógicos emplean variables lógicas o expresiones lógicas como veremos más adelante.

Ejemplos de evaluación de sentencias lógicas

Supongamos que las siguientes variables tienen estos valores:

```
Private intA = 10, intB = 20, intC = 30 As int
Private strNombre1 = "Jorge", strNombre2 = "Málaga" As String
Private blnA = True, blnB = False, blnC = False As Boolean
```

Calcula el valor de las siguientes expresiones lógicas:

1.

blnA	AND	blnB
True		False
False		

2.

Inta	>	intC	AND	blnA
10		30		True
False				
False				

3.

intA + intB	>=	intC	AND	(bInA	OR	bInB)
10 + 20		30		True		False
	True		True		True	

4.

intA + intB	>=	intC	OR	(bInA	AND	bInB)
10 + 20		30		True		False
	True		True		False	

5.

strNombre1	=	"Jorge"	OR	strName2	=	"Juan"
Jorge		Jorge		Málaga		Juan
	True		True		False	

Sentencia If

Al igual que la vida real nos hacemos preguntas, en programación también hace falta comprar valores o cambiar el orden de ejecución del programa para que haga diferentes cosas.

La **sentencia If** se usa para realizar esas preguntas.

Su forma básica es al siguiente:

```
If ( condición ) Then
    Instrucciones
End If
```

Donde **condición** debe contener una comparación o una expresión lógica de las vistas antes.

El significado es: **SI** la condición es **TRUE**, ejecuta las instrucciones que hay entre **Then** y **End If**.

Ejemplos:

```
Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " es un número positivo")
End If

If intA > 10 Or intB > 10 Then
    Log("Uno o los dos números son mayores que 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " es un número par ")
End If
```



If – Else

El comando **Else** permite ejecutar instrucciones cuando la condición del **If** es falsa:

Su forma básica es la siguiente:

```
If ( condición ) Then
    Instrucciones_Verdadero
Else
    Instrucciones_Falso
End If
```

El significado es: Si la **condición** es **True**, entonces ejecutar las **Instrucciones_Verdadero**. Si la condición es **False**, ejecutar las **Instrucciones_Falso**.

Ejemplos:

```
Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " es un número positivo ")
Else
    Log(intA & " no es un número positivo ")
End If

If intA > 10 Or intB > 10 Then
    Log("Uno o los dos números son mayores que 10")
Else
    Log("Ninguno de los números es mayor que 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " es un número par")
Else
    Log(intA & " es un número impar")
End If
```

If – else - else if

Podemos usar varios **If** con varias condiciones para extender la funcionalidad de un único comando **if** (se llama "anidar" comandos **If**).

Cómo lo construimos:

```
If ( condición1 ) Then
    Instrucciones1
Else If ( condición2 ) Then
    Instrucciones2
Else If ( condición3 ) Then
    Instrucciones3
Else If ( condición4 ) Then
    Instrucciones4
...
Else
    Instrucciones_Todo_Falso
End If
```

El funcionamiento de los múltiples **If** anidados es el siguiente:

1. Se comprueba la primera condición (**condición1**). Si es **True**, ejecutamos el código **Instrucciones1** y el **If** finaliza (no se ejecuta nada más).
2. Si la primera condición del primer **If** (**condición1**) es **False**, entonces se comprueba la condición del segundo **If** (**condición2**) y, si es cierta, se ejecutan las **Instrucciones2** (y no se ejecuta nada más).
3. Para el resto de **If**, se comprueban sus condiciones si las anteriores son falsas.
4. Si **ninguna** de las condiciones de los **If** es cierta, se ejecutan las **Instrucciones_Todo_Falso**. El **Else** no es obligatorio usarlo.

Ejemplo 3

Un restaurante de comida rápida ofrece estas comidas:

Comida	Precio
Hamburguesa	5 €
Pizza	3 €
Salchicha	1,5 €

Crea un programa que:

Lea el tipo de comida que el cliente quiere. Imprimir el coste de la comida.
Por ejemplo: Entrada: "Salchicha". Salida: "Salchicha 1,50 €"

Solución

Paso 1

Crea un nuevo proyecto de tamaño 300x300.

Paso 2

En el diseñador, crea la pantalla de la aplicación

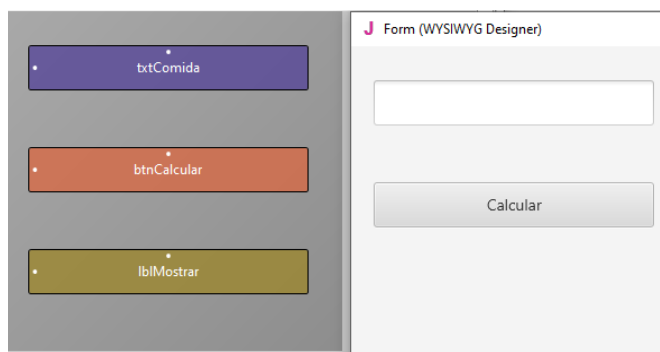
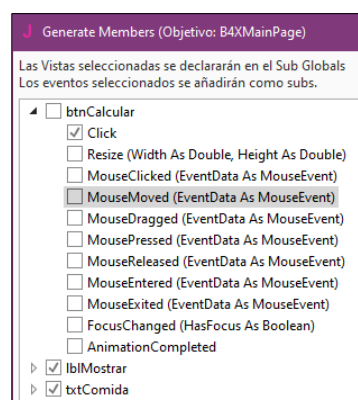


Imagen 13. Pantalla de la aplicación

Paso 3

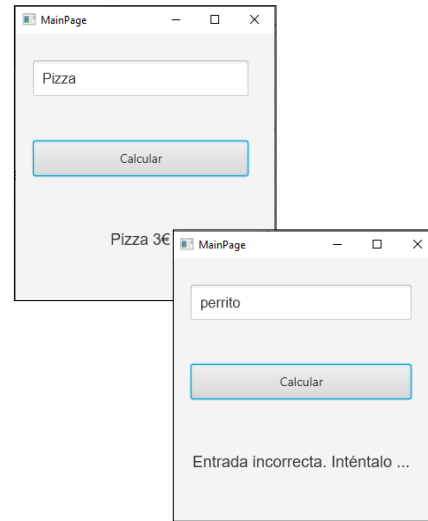
Genera miembros para **txtComida**, **btnCalcular**, **lblMostrar** y **btnCalcular_Click**.



Paso 4

El código que tienes que escribir en el **btnCalcular_Click** es:

```
7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private btnCalcular As Button
11    Private lblMostrar As Label
12    Private txtComida As TextField
13 End Sub
14
15 Public Sub Initialize
16
17 End Sub
18
19 'This event will be called once, before the page becomes visible.
20 Private Sub B4XPage_Created (Root1 As B4XView)
21     Root = Root1
22     Root.LoadLayout("MainPage")
23 End Sub
24
25 Private Sub btnCalcular_Click
26     If txtComida.Text = "Hamburguesa" Then
27         lblMostrar.Text = "Hamburguesa 5€"
28     else if txtComida.Text = "Pizza" Then
29         lblMostrar.Text = "Pizza 3€"
30     else if txtComida.Text = "Salchicha" Then
31         lblMostrar.Text = "Salchicha 1.5€"
32     Else
33         lblMostrar.Text = "Entrada incorrecta. Inténtalo de nuevo"
34     End If
35 End Sub
```



Fíjate que el texto que introducimos distingue entre mayúsculas y minúsculas, con lo que si escribimos "perrito" con la letra "p" en minúscula, no lo reconoce como válido porque en el código hemos dicho que tiene ponerse "Perrito", con la "P" en mayúscula.

Algoritmos con If

Vamos a ver cómo encontrar el máximo de un grupo de números.

Supongamos que leemos 3 números enteros y queremos encontrar el mayor de los 3. Veamos 3 formas diferentes de hacerlo:

Método 1 – Sentencia If simple

```
If Inta > intB AND Inta >
intC then
    Log(Inta)
End If
If intB > Inta AND intB >
intC then
    Log(intB)
End If
If intC > Inta AND IntC >
```

Método 2 –If anidado

```
If Inta > intB then
    If Inta > intC then
        Log(Inta)
    End If
End If
If IntB > IntA then
    If IntB > intC then
        Log(IntB)
    End If
End
If IntC > IntA then
    If IntC > IntA then
        Log(IntC)
    End If
End If
```

Método 3 – Algoritmo Máx

```
Máx = inta
If intB > Máx then
    Máx = intB
End If
If intC > Máx then
    Máx = intC
End If
Log(Máx)
```

Ejercicios

1. Escribe las siguientes condiciones como expresiones lógicas:

- i. **A** pertenece al intervalo [-5, 6)
- ii. **A** es menor que 3 o mayor que 15
- iii. **A** es igual a **B** y **C**
- iv. **A** no tiene el valor de 3
- v. **A** es menor que 2 o **B** es mayor de 78
- vi. **A** y **B** son verdad y **C** es falsa
- vii. **A** es verdad y o bien **B** o **C** son verdad

2. Calcula el resultado (True o False) de las siguientes expresiones lógicas suponiendo los siguientes valores de las variables:

A = 10, B = 2, C = -4, D = 9 and E = 1

- i. (A>B) or (D=10)
- ii. (D >= B) and (E <> C)
- iii. not (E<=C) or (D<=C)
- iv. not ((B<=C) and (D<2))
- v. not (not (B<=E) or not (C<=B))
- vi. ((E<=A) and (E>=C)) and not (C>=A)
- vii. not (not (A >= 2) and (C <>9))

3. Un restaurante de comida rápida ofrece estas comidas:

Comida	Precio
Hamburguesa	5 €
Pizza	3 €
Salchicha	1,5 €

Crea un programa que:

Primero lea la comida que el cliente quiere y después cuántos productos quiere. Muestre el coste total de la comida.

Ejemplo de entrada: "Salchicha", 2

Salida: " 2 x Salchicha 3 €"

4. Has gastado una cantidad de X megabits en la Wikipedia y una cantidad Y de megabits en memes. El coste de visitar la Wikipedia es de 0'1€ por megabit y el de ver memes es de 0'05 € por megabit. Si el consumo total es mayor de 100 €, debes mostrar el mensaje "Consumo excesivo". Si gastas más en ver memes que en visitar la Wikipedia, debes imprimir el mensaje "Demasiados memes". Crea un programa que:
- Lea el valor de X (consumo de megabits de Wikipedia) y el de Y (consumo de megabits por ver memes).
 - Calcule el consumo total.
 - Si el consumo total es mayor de 100€, debe imprimir el mensaje adecuado. Si el consumo en ver memes es mayor que el de visitar la Wikipedia, se imprime el mensaje correcto.
5. Un cibercafé tiene 2 formas de cobrar. Si el usuario es miembro, pagaría 2€ por hora, si no, paga 5€ por hora. Averigua si alguien es miembro o no y calcula el precio que pagaría en función de las horas que emplee. Si el usuario es miembro, el impuesto es del 10%, si no, es del 20%. Crea un programa que:
- Lea cuántas horas ha usado el cliente.
 - Pregunte si es miembro o no.
 - Añada el impuesto adecuado.
 - Imprima lo que tiene que pagar el usuario mostrando el texto: "El usuario es miembro y ha estado 2 horas por 2€/hora más el 10% de impuesto, que hace un total de 4'4€".
6. Quiere comprar algo en Amazon. El vendedor cobra diferentes gastos de envío según tu ubicación. Para EEUU son 5€, para Europa 7€, por Canadá 3€ y para el resto 9€. Crea un programa que:
- Lea el coste del producto.
 - Lea tu ubicación.
 - Imprima los gastos de envío.
 - La salida debe ser: "Tienes que pagar 23€, 20€ por el producto y 3€ por los gastos de envío".

7. Una empresa vende un producto por 0'70€ la unidad si se piden hasta 200 unidades, o bien 0'50€ si se piden más de 200 unidades. Lee el número de piezas que se piden y calcula su valor.

8. Una empresa de telefonía tiene las siguientes tarifas:

Coste fijo 25€	
Duración de la llamada (en segundos)	Coste (€/por segundo)
1-500	0,01
501-800	0,008
801+	0,005

Crea un programa que:

- Lea el número de segundos que todas las llamadas.
 - Calcule la factura mensual para el cliente.
 - Imprima la cantidad total.
 - Salida: "Importe total: 48€".
 - Fíjate que el cargo de los primeros 500 segundos es 0'01€ por segundo, para los segundos 501 a 800 es de 0'008€ y a partir de ahí 0'005€.
9. En la fase de clasificación en salto de longitud de los juegos olímpicos, un atleta realiza 3 intentos iniciales y, si logra una distancia mayor a 7'50 metros, entonces pasa de ronda y puede realizar otros 3 intentos más. Escribe un programa que lea los primeros 3 intentos de un atleta e imprima un mensaje diciendo si puede pasar de ronda o no y que también muestre la mayor distancia que ha saltado.

Puedes usar el método Visible = True or False para ocultar o mostrar Labels, TextFields y Buttons.

10. En una ciudad hay aparcamientos de pago. El coste de aparcar se calcula de la siguiente forma:

Duración aparcamiento	Coste por hora
Hasta 1 hora	3.50 €
Las siguientes 2 horas	8.00 €
Las siguientes 2 horas	12.00€
Más de 5 horas	15.00 €

Escribe un programa que lea el tiempo que alguien dejó su coche en el aparcamiento y que calcule el coste total.

Tema 8 – Subrutinas

🕒 3h

Lo que los estudiantes aprenderán

- Qué es una subrutina (Sub)
- Declaración de una Subrutina
- Paso de Valores
- Devolución de Valores de una Subrutina

En programación, una subrutina es una secuencia de instrucciones que realizar una tarea concreta y que forman una unidad. Esta unidad puede usarse después en los programas dondequiera que esa tarea sea realizada.

Las subrutinas pueden definirse dentro de nuestro programa o dentro de bibliotecas que agrupan múltiples subrutinas para que puedan ser usadas por otros programas. Según el lenguaje de programación, a las subrutinas se les puede llamar funciones, subprogramas, métodos o procedimiento. Para crear una subrutina el programador debe tener en cuenta esto:

- Debe realiza una única tarea.
- Debe ser relativamente pequeña e, idealmente, no mayor que lo que cabría en una pantalla para que pueda leerse fácilmente.
- Debe tener un nombre que haga referencia a lo que hace.

Crear una subrutina en B4J

Ya hemos visto la subrutina **Button1_Click** en B4J cuando se crea un nuevo programa. Fíjate que los eventos de "Click" se gestionan mediante subrutinas.

Ejemplo 1

Imagínate una función que deba sumar dos números introducidos por el usuario. Se trata de un ejemplo muy sencillo, pero lo usaremos para entender el concepto de subrutina y ver cómo funciona.

Los dos enteros intA e intB se declaran en el programa, se les asigna un valor y después invocamos la subrutina MostrarSuma1.

```
8 Sub class_globals
9 Private Root As B4XView
10 Private xui As XUI
11 End Sub
12
13 Public Sub Initialize
14
15 End Sub
16
17 'This event will be called once, before the page becomes visible.
18 Private Sub B4XPage_Created (Root1 As B4XView)
19 Root = Root1
20 Root.LoadLayout("MainPage")
21 End Sub
22
23 'You can see the list of page related events in the B4XPagesManager d
24
25 Sub Button1_Click
26 xui.MsgboxAsync("¡Hola mundo!", "B4X")
27 End Sub
```

Imagen 14. Subrutinas



Invocar a una subrutina se hace escribiendo su nombre (el que queramos) y entre paréntesis ponemos las variables cuyos valores debe conocer para funcionar.

La subrutina se escribe antes o después del programa actual:

```
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     Private intA, intB As Int
22     intA = 10
23     intB = 30
24
25     MuestraSuma1(intA, intB)
26     Log(intSum)
27
28     Log(Suma(intA, intB))
29 End Sub
30
31 Private Sub MuestraSuma1(a As Int, b As Int)
32     intSum = a + b
33     Log(intSum)
34 End Sub
```

Imagen 15. Invocando una subrutina

Si delante ponemos la palabra reservada **Private**, significará que esta subrutina sólo se conocerá dentro del código B4XmainPage; si ponemos **Public**, la subrutina se podrá usar en otras partes de nuestra aplicación.

La palabra "Sub" es una abreviatura de subrutina. Entre paréntesis escribimos el nombre de las variables cuyos valores se enviarán a la función.

Fíjate en la **Imagen 2** en que los datos se envían a la función en el orden en que están escritos al invocar a la subrutina. Por ej. el valor de "intA" se introduce en la variable "a" y el valor de "intB" en la variable "b".



Recuerda

Las variables usadas en las subrutinas cuando son invocadas se llaman **parámetros**.

La subrutina funcionará con los datos contenidos en los parámetros "a" y "b" y NO con las variables "intA" e "intB".



La memoria de una subrutina en B4X

Cada subrutina tiene su propio espacio de memoria donde almacenar variables. La única excepción es la subrutina **Class_Globals** cuyas variables pueden ser consultados por todas las rutinas de B4XMainPage por sus nombres.

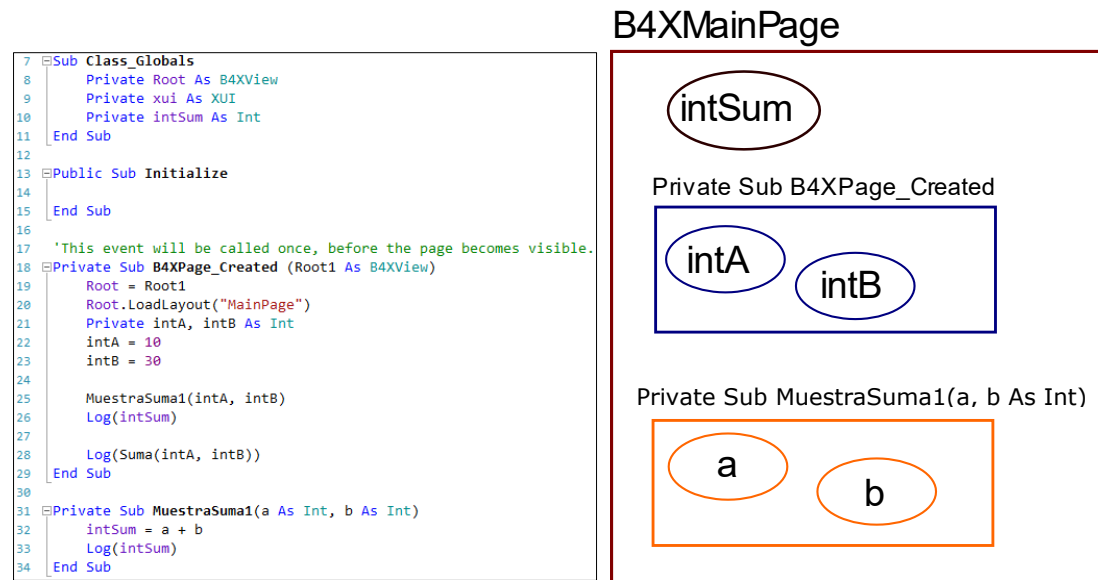


Imagen 16. La memoria

En la **Imagen 3** puedes ver que la variable **intSum** es accesible desde cualquier subrutina y se puede consultar escribiendo su nombre. En B4X estas variables se representan en un color diferente para que el programador pueda distinguirlas fácilmente. Por el contrario, las variables declaradas dentro de una subrutina sólo son accesible desde esa subrutina.

Recuerda



Las variables declaradas dentro de **Class_Globals** se pueden ver desde cualquier subrutina y se llaman **Globales**.

Las variables declaradas dentro de una subrutina sólo se pueden ver dentro de esa subrutina y se llaman **Locales**.

Devolución de un valor desde una subrutina

Una subrutina puede devolver un valor al código que la invocó. Esto se hace de la siguiente forma:

```
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     Private intA, intB As Int
22     intA = 10
23     intB = 30
24
25     MuestraSuma1(intA, intB)
26     Log(intSum)
27
28     Log(Suma(intA, intB))
29 End Sub
30
31 Private Sub MuestraSuma1(a As Int, b As Int)
32     intSum = a + b
33     Log(intSum)
34 End Sub
35
36 Private Sub Suma(a As Int, b As Int) As Int
37     Return(a+b)
38 End Sub
```

Imagen 17. Devolver valores al programa

1. En la declaración de la subrutina tienes que incluir el tipo de variable que devolverá (**as Int** en la imagen).
2. Debes usar la sentencia **Return** dentro de la subrutina para devolver el valor que quieras.
3. El código que ha invocado la subrutina usará el valor devuelto por la subrutina como si fuera una variable.



Recuerda

Normalmente, a las subrutinas que devuelven valores al código se les llama **Funciones**.

Ejemplo 2

Escribe un programa que lea 3 enteros y devuelva el mayor.

```
5 Sub Class_Globals
6     Private Root As B4XView
7     Private xui As XUI
8 End Sub
9
10 Public Sub Initialize
11 End Sub
12
13 Private Sub B4XPage_Created (Root1 As B4XView)
14     Root = Root1
15     Root.LoadLayout("MainPage")
16     Private intA, intB, intC As Int
17     intA = 20
18     intB = 10
19     intC = 300
20
21     Log(CalcularMax(intA, intB, intC)) ' Muestra 300
22 End Sub
23
24 Private Sub CalcularMax(a As Int, b As Int, c As Int) As Int
25     Private intM As Int
26     intM = a
27     If b > intM Then
28         intM = b
29     End If
30     If c > intM Then
31         intM = c
32     End If
33
34     Return(intM)
35 End Sub
```

Imagen 18. Ejemplo 2

1. Declaramos 3 variables enteras (Inta, intB, intC) en la subrutina **B4XPage_Created**.
2. Asignamos valores a las 3 variables.
3. Invocamos a la subrutina CalcularMax con las 3 variables como parámetros.
4. La subrutina aplica el algoritmo Máximo a las variables **a, b, c** y devuelve el resultado intM calculado
5. El valor mayor es mostrado en la ventana de Log.



Recuerda

Nos encantan las subrutinas porque:

- Es aburrido escribir siempre el mismo código.
- Es más rápido usarlas que escribir de nuevo el código.
- Es útil no repetir los mismos fallos. ¡Ya los has corregido una vez!
- Los buenos programadores usan subrutinas.



Ejercicios



Consejo para el profesor

Animar a los estudiantes a solucionar y discutir sus soluciones en clase.
Dedicar al menos una hora a explicarlos.

1. Escribe un programa que calcule el área de un círculo. El usuario debe introducir el radio del círculo en un `textField` y después usa una subrutina para calcular y devolver el área.
2. Escribe un programa que calcule la solución a la ecuación de segundo grado $ax^2 + bx + c = 0$.
 - a. El usuario introducirá los coeficientes **a**, **b** y **c** en `TextFields`.
 - b. El resultado aparecerá en uno o dos `textField` según el valor del discriminante.
 - c. El discriminante debe calcularse con una subrutina que devolverá el valor.
 - d. No se permitirá calcular el discriminante hasta que todos los coeficientes a, b y c se introduzcan.

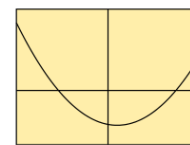
Para mostrar mensajes de error puedes usar `xui.MsgboxAsync("Message","Title")`

a δ
b x1
c x2

Imagen 20. Esquema

Ecuación cuadrática

$$ax^2 + bx + c = 0$$



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Imagen 19. Origen:
Wikipedia.org

3. Construye un programa que use la tortuga y que dibuje cuadrados de lado igual a un valor introducido por el usuario. Crea una subrutina que reciba la longitud del lado y que dibuje el cuadrado empezando por el lugar donde esté actualmente la tortuga y que se mueva en el sentido de las agujas del reloj.
4. Un teatro tiene 3 tipos de entradas: anfiteatro, galería y proscenio. Cada entrada vale 20, 30 y 40€ respectivamente. Construye un programa que:
 - a. Lea los números 1, 2 o 3 que representan las 3 categorías (1: anfiteatro; 2: galería y 3: proscenio).
 - b. Le el número de entradas que se quieren.
 - c. Calcule el valor total de las entradas con una subrutina que devuelva la cantidad y que se muestre el resultado en un `textField`.



Tema 9 – Clases

🕒 3h

Lo que los estudiantes aprenderán

- ¿Qué es una clase?
- ¿Qué es un objeto?
- ¿Qué son los atributos y los métodos?
- Crear y usar una clase sencilla en B4J.

A menudo en programación necesitamos referirnos a objetos similares de una única forma. Por ejemplo, los estudiantes en el colegio. Cada estudiante posee un nombre, una dirección, unas clases a las que asistir, títulos, etc. Para gestionar esta información un programador debe organizar estos datos sistemáticamente.



Consejo para el profesor

Las Clases son un tema complejo en programación. Evita entrar en detalles profundos como herencia, encapsulación, etc.

Clases

En el ejemplo de los estudiantes podríamos decir que cada estudiantes posee la siguiente información:

Estudiante	
1	Identificador Escolar
2	Nombre
3	Apellidos
4	Dirección
5	Teléfono
6	Correo electrónico

También necesitamos las siguientes funciones:

- Registrar un nuevo estudiante
- Cambiar la información del estudiante
- Traspasar un estudiante
- Mostrar la información de un estudiante
- Borrar un estudiante

Así pues, para 3 estudiantes, tendríamos los siguientes elementos:

Estudiante 1	
1	125310
2	Augusta
3	Ada Byron
4	Londres
5	37535795
6	ada@lon.uk

Estudiante 2	
1	125311
2	Maria
3	Curie
4	Varsovia
5	678433
6	maria@var.pol

Estudiante 3	
1	125312
2	Muhammad
3	al-Khwarizmi
4	Jiva
5	646456456
6	algor@jiva.uz



Del ejemplo anterior, podemos concluir que los estudiantes poseen datos y les podemos aplicar funciones similares. El hecho de agrupar los datos de los estudiantes y las funciones que podemos hacer con ellos en un único trozo de código se llama "**clase**". Cada estudiante es un **Objeto** o **Instancia** de la clase. Las variables que caracterizan a un estudiante (nombre, teléfono, dirección, etc.) se llaman **propiedades** y las funciones que podemos aplicarles se llaman **Métodos**.



Recuerda

Llamamos **clase** al agrupamiento de datos y de funciones en un único trozo de código independiente.

Objeto o **Instancia** de la clase son todos los elementos independientes que resultan de usar la clase.

Las variables de un objeto se llaman **propiedades**.

Las funciones que aplicamos a un objeto se llaman **Métodos**.

Las ventajas de usar clases son la flexibilidad en el uso del código, mayor velocidad y facilidad de desarrollo de aplicaciones y la posibilidad de reutilizar el código en otros programas.

Ejemplo de clase en B4J

Una biblioteca posee un conjunto de libros que presta a los lectores registrados. Cada libro tiene propiedades como el título, el autor, la editorial o el año de publicación. Los libros se puede insertar, mostrar o cambiar.

Crea un aplicación en B4J que implemente la clase Libro con las propiedades y métodos descritos.

Metodología de implementación

1. Crea un aplicación **B4XPages** y ponle de nombre "biblioteca".
2. Elige la opción **Proyecto – Añadir nuevo módulo – Módulo de Clase – Standard Class**.
3. En la ventana, ponle de nombre **clsLibro** (significa: clase Libro)
4. Aparecerá una nueva pestaña llamada **clsLibro**.

```
1 Sub Class_Globals
2   Private fx As JFX
3   Public strEscritor, strT
4 End Sub
5
6 'Inicializa el objeto. Puede
7 Public Sub Initialize
8
9 End Sub
```

5. Dentro de la rutina **Class_Globals**, añade las variables que representarán las propiedades de la clase:
 - a. Título del libro
 - b. Nombre del autor
 - c. Editorial
 - d. Año de publicación



```

Sub Class_Globals
    Private fx As JFX
    Public strEscriitor, strTítulo, strAño, strEditorial As String
End Sub

```

6. Finalmente, implementa las subrutinas que ejecutarán los métodos:
- Insertar un libro
 - Mostrar un libro
 - Cambiar un libro



Consejo para el profesor

Ten cuidado de explicar que todas las variables y métodos que hemos creado son públicos, con lo que los objetos podrán usarlos.

Insertar un libro

Esta subrutina aceptará 4 cadenas de texto como parámetros y las asignará en el orden en que aparecen a las variables **strTítulo**, **strEscriitor**, **strAño** y **strEditorial**.

```

14 Public Sub insertarLibro(str1, str2, str3, str4 As String)
15     strTítulo = str1
16     strEscriitor = str2
17     strAño = str3
18     strEditorial = str4
19 End Sub

```

Mostrar libro

Esta subrutina mostrará con el comando "Log" las propiedades del libro indicado.

```

21 Public Sub mostrarLibro
22     Log("Título : " & strTítulo)
23     Log("Escriitor : " & strEscriitor)
24     Log("Año : " & strAño)
25     Log("Editorial : " & strEditorial)
26 End Sub

```

Cambiar libro

Este método aceptará 4 parámetros para cambiar las propiedades del libro en el mismo orden que en el método "insertarLibro":

```

28 Public Sub cambiarLibro(str1, str2, str3, str4 As String)
29     strTítulo = str1
30     strEscriitor = str2
31     strAño = str3
32     strEditorial = str4
33 End Sub

```

La rutina "Initialize"

Esta rutina se usa para dar valores iniciales a las variables o bien para realizar alguna otra acción al crear un objeto de la clase:



```

7 Public Sub Initialize
8     strTitulo = ""
9     strEscriitor = ""
10    strAño = ""
11    strEditorial = ""
12 End Sub

```

Cómo usar la Clase

Volvemos a la pestaña **B4XMainPage** para usar la clase **clsLibro**.

1. Primero creamos los objetos de la clase clsLibro.

```

7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private libro1 As clsLibro
11    Private libro2 As clsLibro
12 End Sub

```

donde libro1 y libro2 son dos objetos de la clase clsLibro con todas las propiedades y métodos discutidos anteriormente.

Uso de los métodos

```

19 Private Sub B4XPage_Created (Root1 As B4XView)
20     Root = Root1
21     Root.LoadLayout("MainPage")
22
23     libro1.Initialize
24     libro2.Initialize
25
26     libro1.insertarLibro("Neuromante", "William Gibson", "1984", "Ace")
27     libro2.insertarLibro("2001: Una odisea del espacio", "Arthur C. Clarke", "1968", "Ace")
28
29     libro1.mostrarLibro
30     libro2.mostrarLibro
31 End Sub

```

El primer método que se usará en los objetos es el método **Initialize**.



Recuerda

Initialize **no es un método opcional**. Es el primer método que hay que invocar antes de usar un objeto.

El método **insertarLibro** introduce los valores de los dos libros en las propiedades de los objetos.

El método **mostrarLibro** muestra las propiedades de cada libro.

```

Titulo : Neuromante
Escriitor : William Gibson
Año : 1984
Editorial : Ace
Titulo : 2001: Una odisea del espacio
Escriitor : Arthur C. Clarke
Año : 1968
Editorial : Ace

```

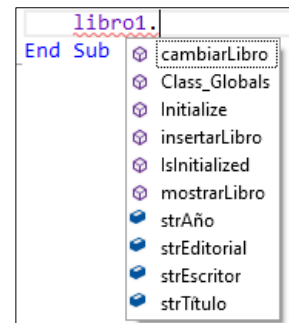


Recuerda

Cada propiedad se puede consultar escribiendo el nombre del objeto, seguido de un punto y después el nombre de la propiedad o método. Algunos métodos necesitan parámetros y otros no.



Cada vez que escribas el nombre de un objeto y pulses la tecla con el punto, B4X mostrará una ventana con todas las propiedades y métodos de la clase que hay disponibles. El método **IsInitialized** comprueba si el objeto está inicializado y existe en todas las clase que has creado.



Ejercicios

1. Siguiendo el primer ejemplo del tema, implementa la clase estudiante en B4J con las siguientes propiedades:

- Identificador escolar
- Nombre
- Apellidos
- Clase
- Teléfono
- Correo electrónico

Y los métodos:

- Nuevo estudiante
- Mostrar estudiante
- Cambiar estudiante
- Cambiar teléfono

2. Supongamos que un profesor sólo imparte una materia en un colegio. Implementa la clase Curso con las siguientes propiedades:

- a. Tema
- b. Clase
- c. Horas
- d. Profesor

Y los métodos:

- a. Nuevo Curso
- b. Cambiar Horas
- c. Cambiar Profesor
- d. Mostrar Curso

3. Una tienda vende ordenadores. De cada uno guarda lo siguiente:

- a. El tipo (sobremesa, portátil)
- b. El modelo
- c. El precio
- d. Su CPU (i3, i5, i7, i9)

Crea una clase que implemente un ordenador con las propiedades anteriores y los métodos nuevoOrdenador, mostrarOrdenador, cambiarCPU y cambiarPrecio. Comprueba que los valores introducidos en CPU y en Tipo son los que aparecen entre paréntesis.

Tema 10 – B4XPages

🕒 3h

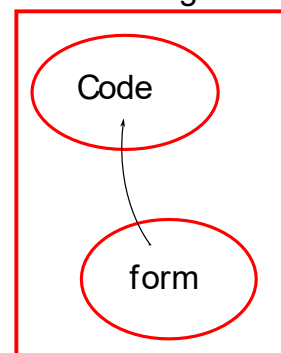
Lo que los estudiantes aprenderán

- Qué es una B4XPage
- Cómo crear y borrar una B4XPage
- Paso de valores entre páginas

B4XPages es una biblioteca de software. Incluye clases y métodos para crear múltiples formularios de comunicación con el usuario. Además, ayuda a portar aplicaciones a diferentes plataformas usando las herramientas de B4A, B4i y B4J.

Cada aplicación que has creado con B4J ya incluye una B4XPage. Se trata de la B4XMainPage que siempre es el primer formulario que se muestra al usuario. De forma más general, podemos decir que cada B4XPage gestiona todo el código necesario para que la interfaz de usuario (GUI) funcione.

B4XMainPage



La estructura de las carpetas de una aplicación

Cuando se crea un nuevo programa con B4XPage, se crea la siguiente estructura de carpetas:

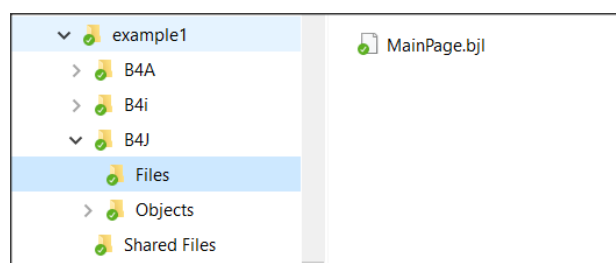


Imagen 1. Carpetas del Ejemplo 1

Cada una de las carpetas B4A, B4i y B4J incluye el código necesario para crear aplicaciones para Android, iOS y PCs (Windows, Linux, etc.) respectivamente.

En concreto, en la carpeta **B4J** está la carpeta **Files** que

contiene todos los ficheros creados con el **Diseñador** y otros ficheros que se usan al ejecutar el programa como, por ejemplo, las imágenes. El fichero **MainPage.bjl** se crea automáticamente al crear la aplicación y es la pantalla de inicio del programa. La carpeta **Shared Files** incluye los ficheros que los 3 tipos de aplicaciones pueden compartir si el programador **desea** crear una aplicación para Android iOS y PC.

La carpeta raíz de la aplicación contiene todos los ficheros que crean las diferentes B4XPages de nuestra aplicación:

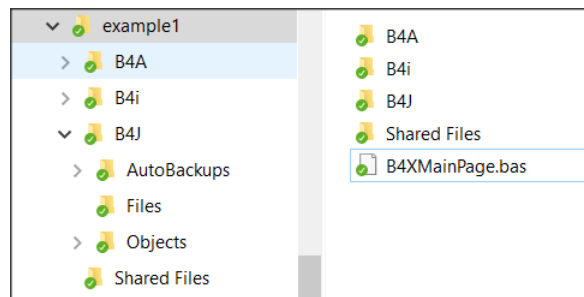


Imagen 2. Ficheros de la B4XPage

La primera página que se crea debe tener el nombre **"B4XMainPage.bas"** y no puede cambiarse; a todas las demás páginas se les puede cambiar el nombre.

Iniciar una aplicación con B4XPage

Al crear una nueva aplicación con B4Xpage, se crea automáticamente la primera página cuyo nombre es **B4XMainPage.bas**. Además, se crea un formulario (o pantalla de interfaz de usuario) para comunicarse con el usuario (que se llama **MainPage.bjl**). Para gestionar las páginas se crea también un mecanismo llamado B4XPagesManager.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
End Sub

Public Sub Initialize

End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub
```

Qué es Root

La variable **Root** es un objeto de la clase **B4XView**. Se encarga de gestionar la pantalla en los diferentes formularios que crea el programador (también está relacionado con la compartición de código en B4J, B4A, B4i). Así, con la sentencia **Root.LoadLayout("MainPage")**, el objeto Root carga en pantalla el formulario **MainPage**.

Crear una nueva B4XPage

Paso 1.

Creemos un formulario desde el menú **Proyecto – Añadir nuevo módulo – Módulo de Clase – B4XPage** y lo nombramos **B4XPage1**.

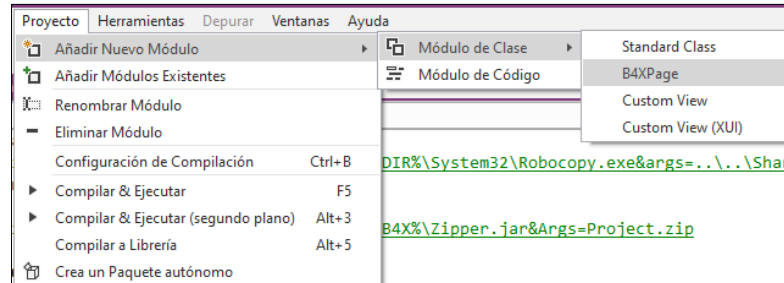


Imagen 21. Crear una B4XPage

Se creará una clase con el nombre "B4XPage1" más algún código básico para iniciarla. La interfaz de usuario (GUI) no se ha creado aún. Ello se hará después mediante el Diseñador.

```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
End Sub

'You can add more parameters here.
Public Sub Initialize As Object
    Return Me
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
End Sub
```

Imagen 22. La nueva B4XPage

Paso 2.

Abre el **Diseñador** y elige en el menú **Archivo** la opción **Nuevo**.

En la pestaña **Variantes** indica las dimensiones del formulario que quieras diseñar y añade una etiqueta y un botón al formulario como se ve en la Imagen 3.

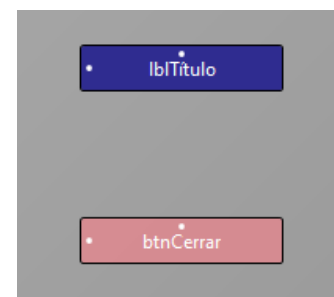


Imagen 23. Formulario

Paso 3.

Elige la opción **Herramientas – Generar** Miembros para insertar los dos objetos en tu código junto con el evento Clic del botón. Recuerda que esta acción debe hacerse cuando estés en el código de la **B4XPage1**.

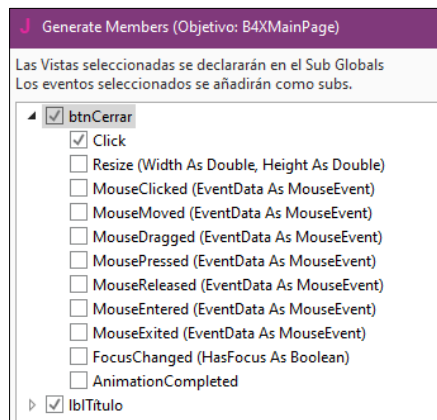


Imagen 24. Generar Miembros

Desde el menú **Archivo – Salvar** guarda el formulario con el nombre **frmPage1** (puedes elegir un nombre más representativo en otra aplicación).

Se creará el siguiente código (Imagen 5) y el fichero **frmPage1.bjl** aparecerá en la carpeta **"Files"**.

```
1 Sub Class_Globals
2     Private Root As B4XView 'ignore
3     Private xui As XUI 'ignore
4     Private btnCerrar As Button
5     Private lblTitulo As Label
6 End Sub
7
8 'You can add more parameters here.
9 Public Sub Initialize As Object
10     Return Me
11 End Sub
12
13 Private Sub B4XPage_Created (Root1 As B4XView)
14     Root = Root1
15     'load the layout to Root
16 End Sub
17
18
19
20 Private Sub btnCerrar_Click
21
22 End Sub
```

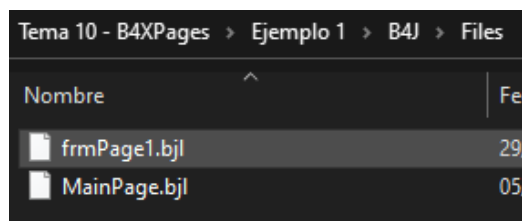


Imagen 25. frmPage1



Paso 4.

Para enlazar el formulario frmPage1 con B4XPage1 hay que invocar al método LoadLayout con la sentencia Root.LoadLayout("frmPage1") dentro del evento B4XPage_Created:

```
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
    Root.LoadLayout("frmPage1")
End Sub
```

Los siguientes pasos consistirán en programar el resto de botones y acciones en función de lo que el programa quiera hacer.

En nuestro ejemplo, hemos hecho que se cambie a la pantalla B4XPage1 al hacer clic en un botón de la B4XMainPage y que se vuelva a la B4XMainPage al hacer clic en el botón que hemos creado en la B4XPage1.

Invocar a una nueva B4XPage

Cada B4XPage que creas es una clase. Así, antes de usarla hay que crear un objeto basado en ella. Esto se suele hacer en la B4XPage que invocará a la nueva página. Por lo tanto, en el ejemplo anterior escribimos lo siguiente en B4XMainPage (**iError! No se encuentra el origen de la referencia.**):

```
7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private página1 As B4XPage1
11    Private Boton1 As Button
12 End Sub
13
14 Public Sub Initialize
15
16 End Sub
17
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     página1.Initialize
22     B4XPAGES.AddPage("Mi primera página", página1)
23 End Sub
24
25
26 Private Sub Boton1_Click
27
28     B4XPAGES.ShowPage("Mi primera página")
29
30     B4XPAGES.ShowPageAndRemovePreviousPages("mi primera página")
31 End Sub
```

1. Creamos un objeto de la clase B4XPage1.

2. Invocamos el método **Initialize** para iniciar el objeto que acabamos de crear.

3. Creamos un identificador para la página (en el ejemplo se llama "Mi primera página").

4. Mostramos la nueva página mientras la principal sigue abierta.

5. Entre comentarios, indicamos cómo mostrar la nueva página cerrando la ventana anterior (puedes ver el resultado en la Imagen 7).

Imagen 26. Crear una B4XPage



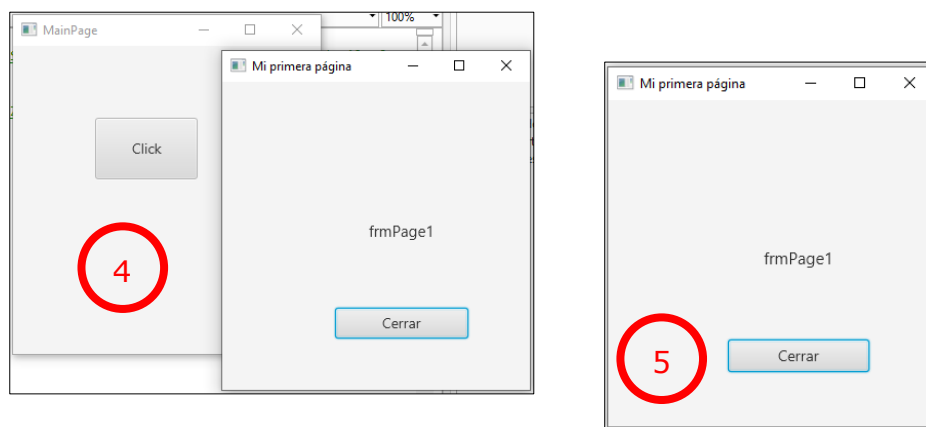


Imagen 27. Las dos formas de abrir una B4Xpage

Cerrar una B4XPage

Cuando se invoca a una B4XPage, el control del programa se pasa a esa página. Así, para cerrar una página tiene que ocurrir algún evento como, por ejemplo, pulsar un botón o pulsar el botón de cerrar ventana en la esquina superior derecha de la ventana. Normalmente, dependerá de cómo se haya abierto la página:

Cuando se abre con:	Normalmente se cierra con:
<code>B4XPages.ShowPage("Mi primera Página")</code>	<code>B4XPages.ClosePage(Me)</code>
<code>B4XPages.ShowPageAndRemovePreviousPages("Mi primera página")</code>	<code>B4XPages.ShowPageAndRemovePreviousPages("MainPage")</code>

Con la primera forma se cierra la ventana actual, mientras que con la segunda se abre la página principal (MainPage) cuando se cierra la ventana actual.

Transferir información entre páginas

Para que una página acceda a los datos de otra, deben tener sus variables declaradas con la palabra reservada **Public**. Las propias variables que contienen las páginas deben ser también públicas, con independencia de si se declaran en el **MainPage** o en otro lugar.



Ejemplo 2

Para este ejemplo usaremos la aplicación del ejemplo 2 que incluye 3 páginas: MainPage, B4XPage1 y B4XPage2. En el diseñador se han creado formularios también. Abre el ejemplo 2, ejecútalo y observa su comportamiento.

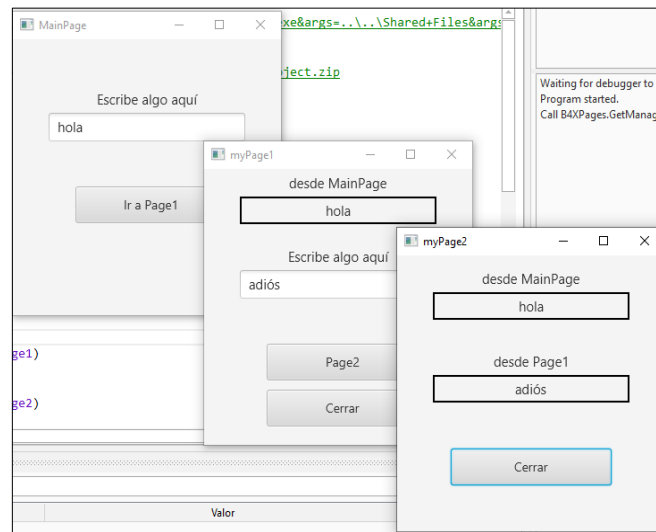


Imagen 28. Ejemplo 2

Al ejecutar la aplicación **Ejemplo 2**, fíjate que el texto de los TextFields se transfiere a las demás páginas. Esto es así porque la **page1**, la **page2** y el **TextFiled** se declararon como públicos.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Public page1 As B4XPage1
    Public page2 As B4XPage2
    Public txtGlobal As TextField
End Sub
```

```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
    Private lblGlobal1 As Label
    Private lblGlobal2 As Label
    Public txtGlobal2 As TextField
End Sub
```

Imagen 29. Declaraciones Public en MainPage y en B4XPage1

Para que **page1** tenga acceso a la variable **txtGlobal1**, debe indicarse el nombre de la página donde fue creada:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
```

donde **lblGlobal1** es una etiqueta que muestra el contenido leído en la pantalla de la **page1**.

Del mismo modo, **Page2** tiene acceso a la variable **txtGlobal1** de la **MainPage** y a la variable **txtGlobal2** de la **Page1** haciendo lo siguiente:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
lblGlobal2.Text = B4XPages.MainPage.page1.txtGlobal2.Text
```

donde **lblGlobal1** y **lblGlobal2** son dos etiquetas que muestran los contenidos de dos variables públicas en la pantalla de la **page2**.



La Vida de las B4XPages

En el anterior ejemplo, prueba a cerrar todas las ventanas excepto **MainPage**, escribe un nuevo texto y pulsa el botón **Ir a Page1**. Verás que el valor mostrado no es el nuevo, sino el primero que escribiste.

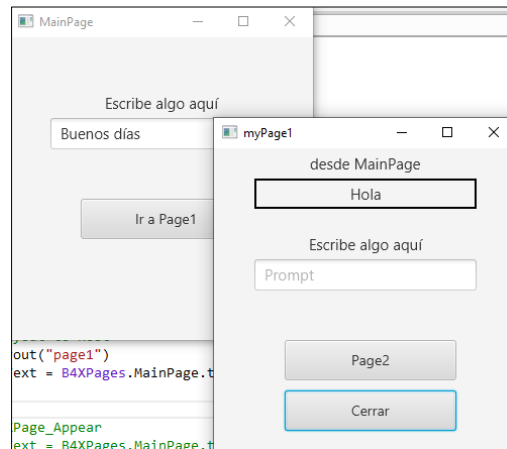


Imagen 30. Vida de una B4XPage

Esto sucede porque las B4XPages permanecen en la memoria del ordenador, así que el evento **B4XPage_Created** no se ejecuta de nuevo cuando volvemos a abrir la página. Para evitar esto, podemos usar el evento **B4XPage_Appear** para volver a leer las variables desde MainPage:

```
Private Sub B4XPage_Appear
    lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
End Sub
```

Al contrario que el evento **B4XPage_Created** que se ejecuta una única vez al crear por primera vez la página, el evento **B4XPage_Appear** se ejecuta cada vez que la página aparece en primer plano, con lo que puedes usarlo para transferir variables de unos formularios a otros.

Ejercicios

1. La pequeña enciclopedia de los perros. Crea una aplicación donde 3 razas de perros diferentes se muestren en una pantalla de inicio y, tras hacer clic en el nombre correspondiente, se muestre información acerca de la raza junto con dos fotos.

Puedes usar un TextArea en el diseñador para hacer más grandes los TextAreas con una barra de desplazamiento.

2. Construye una aplicación que resuelva:
 - a. La ecuación de primer grado $ax+b=0$,
 - b. La ecuación de segundo grado $ax^2+bx+c=0$
 - c. Calcule la hipotenusa de un triángulo dadas las longitudes de los dos lados verticales.

La raíz cuadrada de un número x se calcula con $\text{sqrt}(x)$.

3. Construye una aplicación que cree una tienda virtual como la siguiente: la pantalla de inicio mostrará 4 imágenes de diferentes objetos (por ejemplo, portátiles) y un TextField por cada elemento donde el cliente escribe la cantidad. Después, pulsando el botón **Comprar** el programa mostrará en una nueva página el Valor Total y la cantidad de elementos elegidos. *Sin contar con MainPage, sólo debes usar una única página más.*

