

HISTORIA DE LA PROGRAMACIÓN

Jaime Espinosa
Jhon Granados
Francisco Lenin Morán

**Jaime Espinosa
Jhon Granados
Francisco Lenin Morán**

**HISTORIA DE LA
PROGRAMACIÓN**

Título original: HISTORIA DE LA PROGRAMACIÓN

Primera edición: enero 2020

© 2020, Jaime Espinosa, Jhon Granados
Francisco Lenin Morán

Publicado por acuerdo con los autores.

© 2020, Editorial Grupo Compás
Guayaquil-Ecuador

Grupo Compás apoya la protección del copyright, cada uno de sus textos han sido sometido a un proceso de evaluación por pares externos con base en la normativa del editorial.

El copyright estimula la creatividad, defiende la diversidad en el ámbito de las ideas y el conocimiento, promueve la libre expresión y favorece una cultura viva. Quedan rigurosamente prohibidas, bajo las sanciones en las leyes, la producción o almacenamiento total o parcial de la presente publicación, incluyendo el diseño de la portada, así como la transmisión de la misma por cualquiera de sus medios, tanto si es electrónico, como químico, mecánico, óptico, de grabación o bien de fotocopia, sin la autorización de los titulares del copyright.

Editado en Guayaquil - Ecuador

ISBN: 978-9942-33-159-5

Cita.

J. Espinosa, J. Granados, F. Moran (2020) Historia de la programación, Editorial Grupo Compás, Guayaquil Ecuador, 67 pag

Historia de la programación

Gottfried Wilheml Von Leibniz (1646-1716), quien aprendió matemáticas de forma autodidacta (método no aconsejable en programación) construyó una máquina similar a la de Pascal, aunque algo más compleja, podía dividir, multiplicar y resolver raíces cuadradas.

Pero quien realmente influyó en el diseño de los primeros computadores fue **Charles Babbage** (1793-1871). Con la colaboración de la hija de Lord Byron, **Lady Ada Countess of Lovelace** (1815-1852), a la que debe su nombre el lenguaje ADA creado por el DoD (Departamento de defensa de Estados Unidos) en los años 70. Babbage diseñó y construyó la "máquina diferencial" para el cálculo de polinomios.

Más tarde diseñó la "máquina analítica" de propósito general, capaz de resolver cualquier operación matemática. Murió sin poder terminarla, debido al escepticismo de sus patrocinadores y a que la tecnología de la época no era lo suficientemente avanzada. Un equipo del Museo de las Ciencias de Londres, en 1991, consiguió construir la máquina analítica de Babbage, totalmente funcional, siguiendo sus dibujos y especificaciones.

Un hito importante en la historia de la informática fueron las tarjetas perforadas como medio para "alimentar" los computadores. Lady Ada Lovelace propuso la utilización de las tarjetas perforadas en la máquina de Babbage.

ÍNDICE

Historia de la programación.....	1
Unidad 1	4
Generalidades	4
Historia de Java.....	5
Librería, Biblioteca o Paquete	10
Trabajar con JOptionPane	10
Método showMessageDialog	11
Manipulador de Formato.....	12
Operadores Aritméticos	13
Reglas de Prioridad.....	14
Método showInputDialog().....	15
UNIDAD 2.....	20
Estructuras Selectivas, Definición	21
Instrucción if.....	22
Estructura Selectiva Simple.....	22
Estructuras Selectivas Dobles.....	27
UNIDAD 3.....	32
PROGRAMACIÓN ORIENTA A OBJETO	33
JAVA (FORMULARIOS)	33
Pasos para crear un proyecto.....	33
Public	39
Hace referencia al tipo de clase, es decir, publica.....	39
Void.....	39
Instrucción setText	39
Instrucción requestFocus.....	40
Botón LIMPIAR:.....	43

-

UNIDAD 4.....	47
Estructuras Selectivas, Definición	48
Tipos de Estructuras Selectivas son:	48
Estructura Selectiva Simple.....	49
UNIDAD 6.....	61
Introducción a la Programación HTML.....	62
Crear una página HTML	62
Introducción a Javascript	67



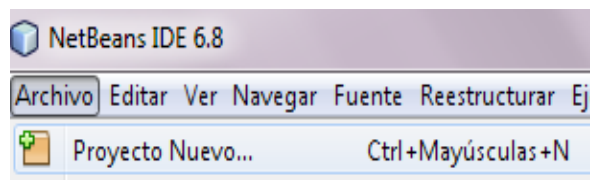
Historia de Java

Tal vez la contribución más importante a la fecha, por parte de la revolución del microprocesador, es que hizo posible el desarrollo de las computadoras personales, que ahora suman cientos de millones a nivel mundial.

Muchas personas creen que la siguiente área importante en la que los microprocesadores tendrán un profundo impacto, es en los dispositivos electrónicos para uso doméstico. Al reconocer esto, **Sun Microsystems** patrocinó en 1991 un proyecto interno de investigación denominado **Green**. El proyecto desembocó en el desarrollo de un lenguaje basado en **C++** al que su creador, **James Gosling**, llamó **Oak** debido a un árbol de roble que tenía a la vista desde su ventana en las oficinas de **Sun**.

Posteriormente se descubrió que ya existía un lenguaje de programación con el mismo nombre. Fue entonces cuando un grupo de gente de **Sun** visitó una cafetería local y surgió el nombre **Java** (una variedad de café) y así quedó, como lo vemos en su icono. Como lenguaje de programación para computadores, **Java** se introdujo a finales de 1995.

Pasos para crear un proyecto:
siga los siguientes pasos:
Archivo → Proyecto Nuevo



Se presenta: **Proyecto Nuevo**,
Seleccionar proyecto:
Categorías: La carpeta **Java** y en
Proyectos: **Aplicación Java**.

Visualizamos la **Nueva Aplicación de Java**, que sirve para almacenar los datos.

Nombre del proyecto: aquí va el nombre de nuestro (**Ejercicio1**).

Ubicación del Proyecto: es la ruta (**drive**) donde se guardará la clase.

Carpeta proyecto: que generalmente es el nombre de la carpeta donde se almacenará nuestra clase o ejercicio y clic en **Terminar**.

Nombre y ubicación

Nombre proyecto:

Ubicación del proyecto:

Carpeta proyecto:

Usar una carpeta dedicada para almacenar las bibliotecas

Carpeta de Bibliotecas:

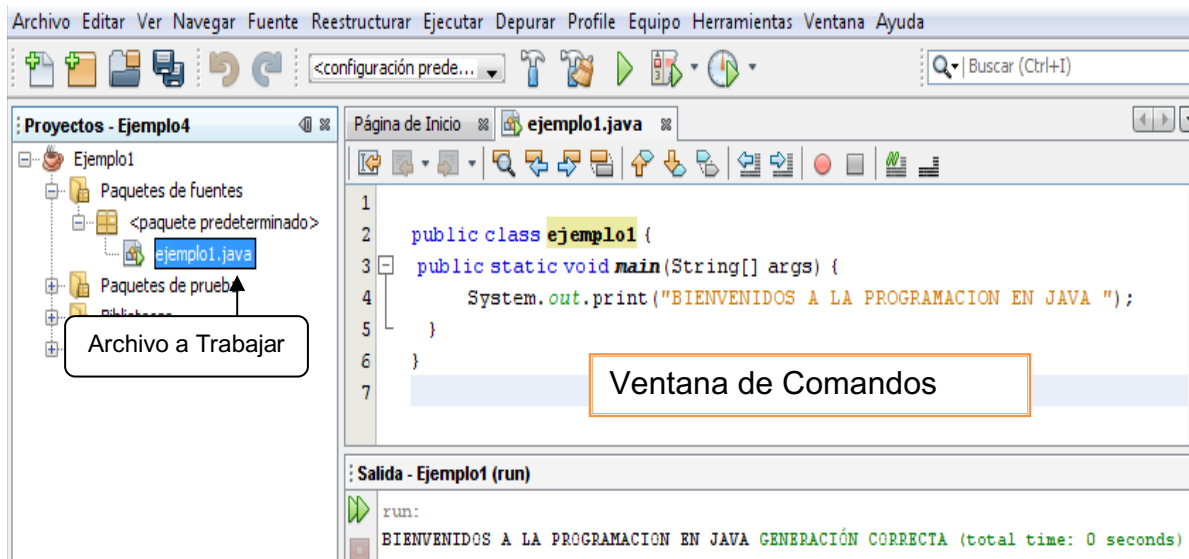
Usuarios y proyectos diferentes pueden compartir las mismas librerías de compilación (ver la Ayuda para más detalles).

Crear clase principal

Configurar como proyecto principal

Debes borrar el **punto main**, en cada ejercicio que realices.

Al finalizar el editor (**Asistente**) genera un archivo, donde vamos a comenzar a escribir la codificación de nuestro proyecto:



Luego procedemos a escribir en la **Ventana de Comandos** la siguiente codificación:

System.out.print("BIENVENIDOS A LA PROGRAMACIÓN EN JAVA");

NOTA: Todo esto debe hacerlo entre las llaves.

Indica a la computadora que realice una acción, es decir, que presente, que muestre o visualice la **cadena** de caracteres contenida entre los caracteres de comillas dobles.

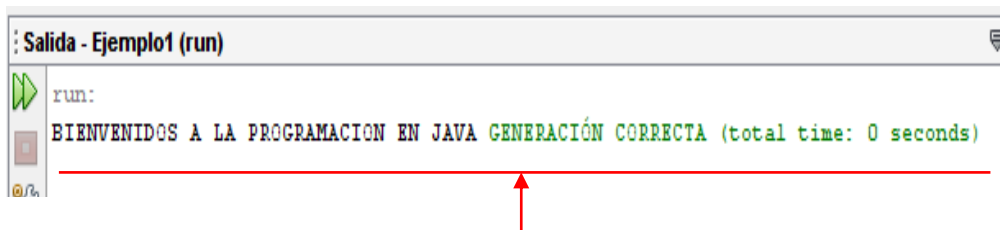
F6 es para ejecutar o el siguiente botón:



System.out. Se conoce como objeto de salida estándar y permite a las aplicaciones en Java mostrar conjuntos de caracteres en la ventana de comandos, desde la cual se ejecuta la aplicación en Java.

El método **System.out.print** muestra, presenta, visualiza o imprime una línea de texto en la ventana de comandos

El mismo que presenta un mensaje por **consola**, que es lo que está entre comillas, recuerda que todo **Mensaje** debe ir entre comillas dobles.



Como se dará cuenta el **print**, se **concatena** con otro texto, para que esto no ocurra trabaje con la instrucción **println**.

Print. Esta función presenta, visualiza, muestra o imprime una expresión. Una expresión puede ser un mensaje, una variable, una función o cualquier combinación de ellas. Si la expresión es un **mensaje** debe ir entre comillas, ejemplo:

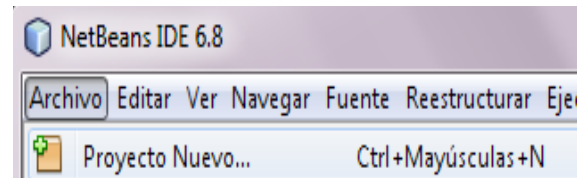
("BIENVENIDOS A LA PROGRAMACIÓN EN JAVA");

Punto y coma. La utilidad de la instrucción punto y coma (;) al final de cada una de estas líneas es la de separar dos instrucciones consecutivas; si usted no la incluye al momento de ejecutar el programa presentará error.

Ejercicio de Aplicación 1

A continuación, un proyecto que presenta varios mensajes las líneas distintas.

Archivo → Proyecto Nuevo:

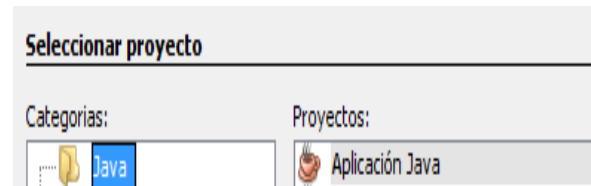


Se presenta el **Proyecto Nuevo**,

Seleccionar proyecto:

Categorías: Carpeta **Java** y en

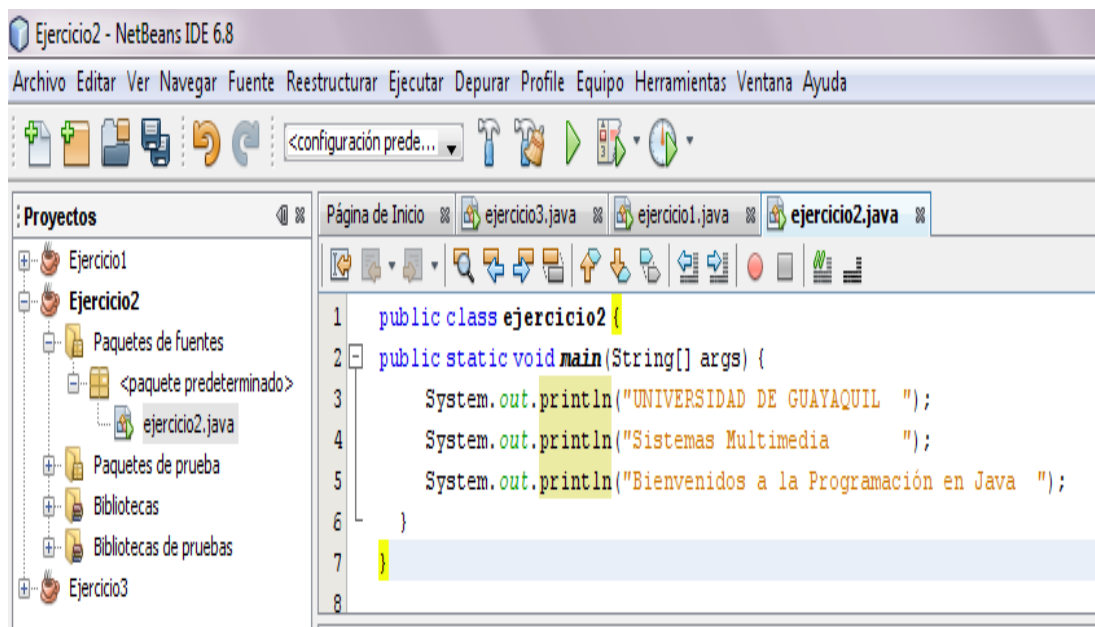
Proyectos: **Aplicación Java**



- Crear clase principal `ejercicio2.Main`
- Configurar como proyecto principal

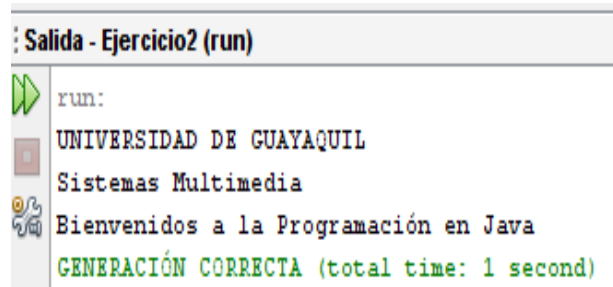
Nota: En cada ejercicio debes borrar el punto y el main

Luego escriba las siguientes líneas:



-

F6 → para ejecutar el proyecto, y se visualizara por consola los siguientes mensajes:



```
Salida - Ejercicio2 (run)
run:
UNIVERSIDAD DE GUAYAQUIL
Sistemas Multimedia
Bienvenidos a la Programación en Java
GENERACIÓN CORRECTA (total time: 1 second)
```

Println

Parecida a la instrucción **print**, pero con la diferencia de que, por cada **println**, los mensajes se presentarán en una línea distinta, es como que usted utilice la tecla **enter**.

Ventana de Comandos

Programa usado para escribir cualquier tipo de texto (sin formato, ej: bloc de notas), en este caso, una codificación en **“Java”**.

Compiladores

Un grupo de programas que se encargan de revisar los errores de una codificación (escritura de un programa, en nuestro caso una clase).

Programa Fuente

Es la codificación original de un programa, en cualquier lenguaje de Programación, en nuestro caso, en **“Java”**, el programa es una clase, y cada clase en un archivo.

-

Programa Objeto

Es el programa en "**Java**", listo para ejecutar libre de errores, es decir, es el resultado de la compilación de un programa Fuente.

Librería, Biblioteca o Paquete

Es un conjunto de archivos que contienen una serie de funciones que serán usadas en el presente programa, que generalmente deben ser importadas dentro del compilador. Una librería, biblioteca o paquete es una colección de clases de nombres, esto se lo conoce como **bibliotecas de clase java** o la **Interfaz de Programación de Aplicaciones de Java (API)** por sus siglas en inglés).

Las librerías o paquetes del **API**, se dividen en básicos y opcionales. Los nombres de la mayoría de las librerías del **API** de Java comienzan, ya sea con **"java"** (paquetes básicos) o **"javax"** (paquetes opcionales), muchos de ellos se incluyen como parte del Kit de desarrollo del software para Java.

Uso del paquete javax.swing.*;

Este paquete, contiene dos de los métodos más importantes, en cuanto a lo que a entorno gráfico se refiere. Por ejemplo, si queremos mandar imprimir en pantalla algún mensaje, por medio de una ventana, la sintaxis es la siguiente:

JOptionPane.showMessageDialog(null, "Mensaje");

En donde:

Null, es argumento que, SIMPRE lo pondremos en el método **MessageDialog**
"Mensaje", es la cadena de caracteres que queremos presentar o visualizar.

Trabajar con JOptionPane

Es habitual que el usuario tenga que ingresar datos, confirmar una acción o pedirle algún dato sencillo, darle a elegir entre varias acciones o simplemente mostrarle un aviso. Por lo tanto, es necesario abrir una ventana secundaria, donde el usuario deba realizar alguna acción que se requiera y cerrarla.

Método JOptionPane

Proporciona cuadros de diálogos previamente empaquetados, los cuales permiten a los programadores mostrar ventanas que contengan mensajes para los usuarios.

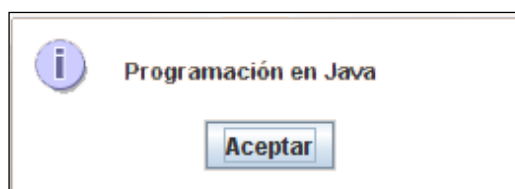
Comentarios //

Los comentarios no afectan la ejecución del programa. Son opcionales. Todo comentario comienza con dos barras (**//**), y no es necesario finalizarlas. Cualquier carácter entre estas marcas es ignorado por el compilador de **"Java"**.

Ejercicio de Aplicación 2 2

A continuación, un proyecto que presenta un mensaje en un cuadro de dialogo.

```
1 //Llamamos a los Paquetes de java
2 import javax.swing.JOptionPane; //Vamos a trabajar con JOptionPane
3 public class ejercicio3 {
4     public static void main(String[] args) {
5         JOptionPane.showMessageDialog(null, "Programación en Java ");
6     }
```



Línea 1, → Es un comentario.

Línea 2, → **import javax.swing. JOptionPane;**

Indica que el programa importara la librería **javax.swing** que utilizaremos en la clase **JOptionPane**.

Línea 3, → Comienza la declaración de la **clase ejercicio3**; el nombre de archivo para esta clase **public** debe ser **ejercicio3**

Línea 4, → Es el punto de toda aplicación en Java. Los paréntesis después de **main** indican que este es un bloque de construcción del programa, al cual se llama *método*. La palabra clave **void** indica que es un método y que realizará una tarea (mostrará un texto).

(String args[]) es una parte requerida de la declaración del método **main**.

Línea 5, → Llama al método **showMessageDialog** de la clase **JOptionPane** para mostrar un cuadro de dialogo que contiene un mensaje y requiere de dos argumentos, el primero siempre será la palabra clave **null** que debe ir separado por una coma (,) y el segundo es el **mensaje** a mostrar en el cuadro de dialogo.

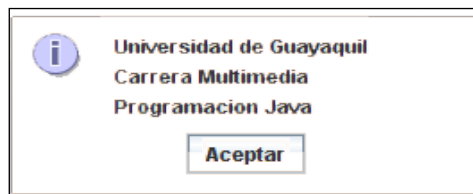
Método showMessageDialog

Presenta una ventana de aviso, la misma que se cierra hasta que el usuario realice una acción en la misma.

Ejercicio de Aplicación 3

El siguiente proyecto, visualiza un cuadro de dialogo con un mensaje en varias líneas.

```
1 import javax.swing.JOptionPane;
2 public class ejercicio5 {
3     public static void main(String[] args) {
4         JOptionPane.showMessageDialog(null,"Universidad de Guayaquil \nCarrera Multimedia \nProgramacion Java");
5     }
```



Manipulador de Formato

Existe una serie de caracteres especiales que no podríamos representar con letras o números, ejemplo: un salto de línea, o un tabulador. Esto es lo que se conoce como **Manipulador de Formato**:

<code>\n</code>	salto de línea
<code>\r</code>	tabulador horizontal

Palabras Reservadas

Existe una serie de **palabras reservadas** las cuales tienen un significado especial para **Java** y por lo tanto no se pueden utilizar como nombres de variables. Dichas palabras son:

abstract	boolean	Break	byte	case	catch
char	const	Class	continue	default	do
double	else	Extends	final	finally	float
for	goto	If	implements	import	instanceof
int	interface	Long	native	new	null
throws	transient	Try	void	volatile	while

Variable

Una variable es un **casillero de memoria** en el cual podemos almacenar datos; estos datos pueden variar dependiendo de la ejecución del programa. Toda variable debe tener un nombre. Los nombres de las variables deben cumplir con las siguientes reglas:

- Deben de empezar siempre con una **letra**, (minúscula) y pueden ser una **combinación** de **letras** y **números**. Por ejemplo, nombres correctos de variables: **suma**, **xx**, **m1**, **m2**, **xyz123**, **sueldo**, etc....
- **No** pueden tener **espacios en blanco** ni símbolos especiales intermedios. Para nombres de variables largos, use abreviaturas (sin puntos). Por ejemplo: incorrecto es usar **sueldo por hora**, correcto sería **suelxhor**.

- Deben ser **nombres cortos** y significativos. Evite usar nombres largos y difíciles. Por ejemplo, en lugar de usar la variable **promedio** use **prom**.

Cada nombre de variable debe ser **único**, es decir, no puede existir otra variable con el mismo nombre, en la codificación del programa.

Tipos de Variables

Todas las variables en el lenguaje **Java** deben tener un tipo de dato.

El tipo de la variable determina los valores que la variable puede contener y las operaciones que se pueden realizar con ella.

TIPO	RANGO
String	Cadena de Caracteres
Int	Número real, cantidades pequeñas
Float	-32768.....32767 números enteros
Double	Flotante doble

Ejemplo, para el uso de los tipos de variables:

String	Nombres, Apellidos, Dirección, Estado Civil, Cargo, es decir, toda una cadena de caracteres.
Int	Edad, Cantidad, Días, Meses, Años, Número de Hijos y cantidades pequeñas, pero enteras
Float	Estatuta, Iva, Subtotal, Resultado de una división, cantidades que contengan números con decimales o números enteros grandes
Double	Para cantidades de más de 10 cifras, ejemplo: Número de población, extensión de un territorio, etc.

Operadores Aritméticos

La mayoría de los programas realizan cálculos aritméticos que permiten efectuar **procesos** o cálculos **matemáticos elementales**. Los operadores aritméticos son:

Operador	Operación	Ejemplo
*	MULTIPLICACIÓN	15 * 2 = 30
/	DIVISIÓN	15 / 2 = 7.5
%	RESIDUO DE DIVISIÓN	15 % 2 = 1
+	SUMA	15 + 2 = 17
-	RESTA	15 - 2 = 13

En el caso de los operadores “/” y “%”, son exclusivamente para la división.

El operador “/” es usado para la división decimal, es decir, la división común y corriente. Por ejemplo:

$$6 / 3 = 2$$

$$6 / 4 = 1.5$$

$$7 / 2 = 3.5$$

$$4 / 3 = 1.33333$$

-

El operador % obtiene el residuo, es decir, lo que sobra de una división.

Por ejemplo:

- (9 % 2); → 1 (sobra 1 al dividir 9 para 2)
- (6 % 4); → 2 (divida 6 para 4 y le sobran 2)
- (14 % 3); → 2 (porque 14 / 3 es 4 sobrando 2)
- (10 % 5); → 0 (divida 10 para 5 y no le sobra nada)

Reglas de Prioridad

Dentro de una expresión matemática compleja siempre se efectúan, **primero** los **paréntesis** más internos, luego las divisiones y las multiplicaciones y al final se realizan las sumas y restas.

Por ejemplo: $5 + 4 / 2 - 1$

Primero se efectúa $4 / 2$, o sea 2 y luego $5 + 2 - 1$. El resultado es 6.

Reglas de Prioridad

()	Prioridad Máxima
/, %, *	Prioridad Intermedia
+, -	Prioridad Mínima

Por ejemplo:

$$5 + 2 * 4 - 3$$

Es 10; primero se efectúa $2*4$ y luego se suma y se resta. Sin embargo, los **paréntesis** siempre se evalúan **primeros**.

Por ejemplo:

$$(5 + 2) * 4 - 3$$

Es 25; primero se hace el paréntesis $(5+2)$ y luego se multiplica por 4 y se resta 3.

Veamos ahora los siguientes ejemplos:

A) $2 + 1 * 5 + 2$
Es 9, porque primero es $1 * 5 = 5$; luego $2 + 5 + 2 = 9$

B) $(5 \% 2) + 2 / 2$
Primero es $\% (5 \% 2) \rightarrow 1$ y $2 / 2 \rightarrow 1$; entonces $1 + 1 = 2$

C) $5 + 2+3 / 2$
Primero $3 / 2 = 1.5$; más $5 + 2 + 1.5 = 8.5$

Fórmulas

Una fórmula es una expresión que se utiliza para **obtener** un resultado. Por ejemplo:

$$su = pn + st$$

-
Es la fórmula que representa la suma de dos valores, donde la variable “pn” representa el primer valor, la variable “st” el segundo valor y la “su” la suma.

Las fórmulas siempre se evalúan de **derecha** a **izquierda**, es decir, colocando la expresión a calcular al lado derecho **del igual** y la variable que guarda el resultado del **lado izquierdo**.

La siguiente fórmula representa el promedio de tres calificaciones de un estudiante:


Expresión a calcular

Resultado a Obtener → $pro = (pn + sn + tn) / 3$

Método showInputDialog()

Permite crear un campo de texto en la ventana para que el usuario pueda ingresar de datos libremente, y estos son almacenados en una variable de memoria.

Ejercicio de Aplicación 4

El siguiente proyecto solicita el ingreso de su nombre y edad, y luego presenta los datos ingresados, en la misma línea.

```
1 //Llamamos a los paquetes de java
2 import javax.swing.JOptionPane;
3 public class ejemplo4 {
4 public static void main(String[] args) {
5 //Declaramos a las variables que almacenaran lo que vamos a ingresar
6 String ed,nom;
7 //nom = almacena los datos a ingresar y showInputDialog, crea un campo de texto
8 nom=JOptionPane.showInputDialog(" Ingrese su nombre -->>");
9 //ed = almacena los datos ingresados y showInputDialog, crea un campo de texto
10 ed=JOptionPane.showInputDialog(" Ingrese su edad -->>");
11 //showMessageDialog presenta un mensaje y el contenido de las variables nom y ed
12 JOptionPane.showMessageDialog(null, "Tu nombre es "+nom+" y tienes "+ed+" años ");
13 }
```

Línea 2,

import javax.swing.JOptionPane;

Indica que el programa importara la librería de **javax.swing** que vamos a trabajar, en este caso a **JOptionPane**, que sirve para los cuadros de diálogos.

Línea 6,

String ed, nom; declaramos las variables de tipo cadena de caracteres, que sirven para almacenar los datos ingresados.

Línea 8 y 10,

Las variables **ed y nom** almacenaran los datos ingresados y **showInputDialog** crea un campo de texto, el mismo que utilizaremos para ingresar los datos.

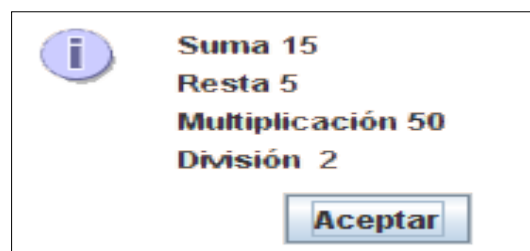
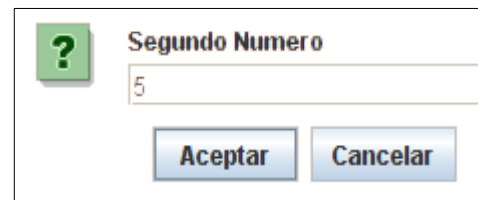
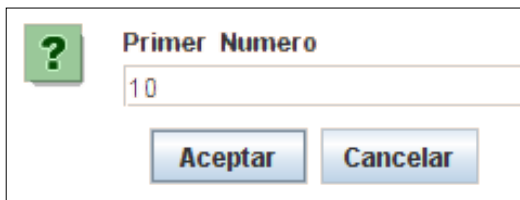
Línea 12,

JOptionPane.showMessageDialog, permite presentar un cuadro de dialogo, acompañado de un mensaje y lo almacenado en sus variables respectivas.

Ejercicio de Aplicación 5

Realice un proyecto que ingrese dos números y obtenga las cuatro operaciones matemáticas.

```
1 import javax.swing.JOptionPane;
2 public class ejercicio1 {
3     public static void main(String[] args) {
4         int pn,sn, su, re, mu, di;
5         pn=Integer.parseInt(JOptionPane.showInputDialog("Primer Numero "));
6         sn=Integer.parseInt(JOptionPane.showInputDialog("Segundo Numero "));
7         su=pn+sn;
8         re=pn-sn;
9         mu=pn*sn;
10        di=pn/sn;
11
12        JOptionPane.showMessageDialog(null,"    La Suma es    " + su +
13                                     "\n La Resta es    " + re +
14                                     "\n La Multiplicacion " + mu+
15                                     "\n La Division    " + di);
```



Primero debemos importar la librería para trabajar con los cuadros de diálogos.

A continuación, declaramos las variables que vamos a ingresar y las que van a contener un proceso matemático de tipo entero (**pn, sn, su, re, mu, di**).

Luego ingresamos los datos que se almacenaran en las variables **pn, sn**; esto es gracias a las siguientes líneas:

```
pn=Integer.parseInt(showInputDialog ("Primer Número "));  
sn=Integer.parseInt(showInputDialog ("Segundo Número "));
```

Recuerda: al final de cada línea el **punto y coma**.

Procedemos a realizar varios procesos matemáticos como:

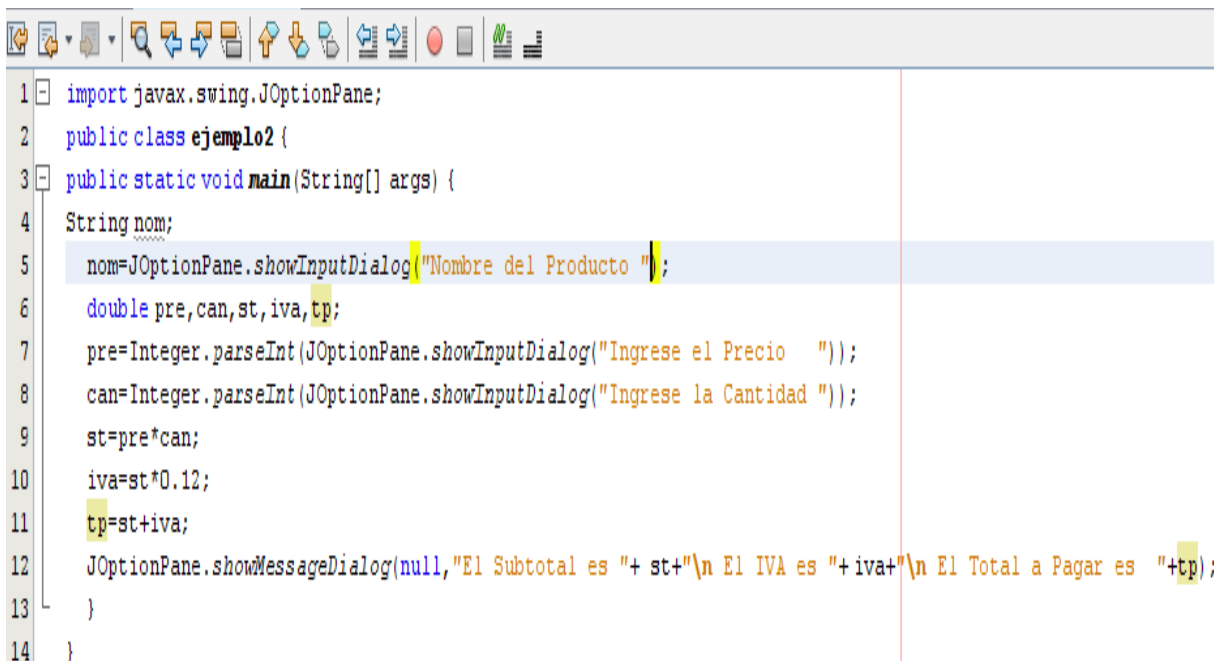
Suma	su = pn + sn;
Resta	re = pn - sn;
Multiplicación	mu = pn * sn;
División	di = pn / sn;

Finalmente presentamos los resultados en un cuadro de dialogo, pero en diferentes líneas:

```
JOptionPane.showMessageDialog(null,  
"Suma " +su+ "\n Resta " +re+ "\n Multiplicación " +mu+ "\n División"  
+di)
```

Ejercicio de Aplicación 6

Ingrese el nombre, precio y cantidad de un producto a comprar, calcule y visualice el subtotal, el 12% del Iva y el total a pagar.



```
1 import javax.swing.JOptionPane;  
2 public class ejemplo2 {  
3     public static void main(String[] args) {  
4         String nom;  
5         nom=JOptionPane.showInputDialog("Nombre del Producto ");  
6         double pre,can,st,iva,tp;  
7         pre=Integer.parseInt(JOptionPane.showInputDialog("Ingrese el Precio "));  
8         can=Integer.parseInt(JOptionPane.showInputDialog("Ingrese la Cantidad "));  
9         st=pre*can;  
10        iva=st*0.12;  
11        tp=st+iva;  
12        JOptionPane.showMessageDialog(null,"El Subtotal es "+ st+"\n El IVA es "+ iva+"\n El Total a Pagar es "+tp);  
13    }  
14 }
```

-
Recuerda siempre importar la librería para trabajar con los cuadros de diálogos. Inicializamos la variable **nom** de tipo **String** para la cadena de caracteres:

String nom, que es donde se almacenara los datos a ingresar.

Declaramos las variables que vamos ingresar como el Precio (**pre**) y la Cantidad (**can**), así como también las que van a contener el resultado **st**, **iva**, **tp**. Luego realizamos el cálculo matemático:

SUBTOTAL	st = pre * can;
IVA	iva = st * 0.12;
TOTAL a PAGAR	tp = st + iva;

Ejercicio de Aplicación 7

Un empleado gana \$3.00 la hora, lea la cantidad de horas trabajadas y luego visualice su salario neto, el 9.35% de descuento al IESS y el salario a recibir.

```
1 import javax.swing.*;
2 public class ejercicio7 {
3     public static void main(String[] args) {
4         double ht, sn, des, sr;
5         ht=Double.parseDouble(JOptionPane.showInputDialog("Horas Trabajadas "));
6         sn=ht*3;
7         des=sn*9.35/100;
8         sr=sn-des;
9         JOptionPane.showMessageDialog(null, "\nSalario Neto " + sn + "\n Descuento " + des + "\n Salario a recibir " + sr)
10    }
```

Declaramos las Horas Trabajadas, Salario Neto, Descuento y Salario a Recibir. En este ejercicio solo ingresamos la Horas trabajadas, ya que el sueldo es de \$3.00 por cada hora trabajada y hacemos los siguientes cálculos:

Salario Neto	sn = ht * 3;	Descuento	des = sn * 9.35/100;
Salario a Recibir	sr = sn - des;	Una vez hecho los respectivos cálculos matemáticos, procedemos a presentar los resultados en varias líneas.	

Ejercicio de Aplicación 8

Tres personas deciden invertir en un negocio, cada uno aporta una cantidad distinta, ingresar el aporte de cada socio y determinar en porcentaje, cuál es la participación de % en el negocio.

```
1
2 import javax.swing.JOptionPane;
3 public class ejemplo4 {
4 public static void main(String[] args) {
5     double i1,i2,i3,ti,p1,p2,p3;
6     i1=Double.parseDouble(JOptionPane.showInputDialog("Inversionista UNO "));
7     i2=Double.parseDouble(JOptionPane.showInputDialog("Inversionista DOS "));
8     i3=Double.parseDouble(JOptionPane.showInputDialog("Inversionista TRES "));
9     ti=i1+i2+i3;
10    p1=(i1*100)/ti;
11    p2=(i2*100)/ti;
12    p3=(i3*100)/ti;
13    JOptionPane.showMessageDialog(null, "Total de Inversión "+ti+"\n Porcentaje Uno "+p1+
14                                   "\nPorcentaje Dos   "+p2+"\nPorcentaje Tres "+p3 );
15 }
```



Estructuras Selectivas, Definición

Una Estructura Selectiva es una pregunta con tan sólo dos respuestas posibles: **Verdadero** o **Falso**. Las Estructuras Selectivas se construyen a partir de las condiciones. Una **condición** es una expresión lógica de la siguiente forma:

Variable **operador relacional** Variable/valor

Operadores relacionales son:

Operador Relacional	Significado	Ejemplo
>	Mayor que	10 > 4
<	Menor que	30 < 60
==	Igual	1 == 1
>=	Mayor o igual que	40 >= 30
<=	Menor o igual que	50 <= 100
!=	Distinto a	20 < > 70

De esta manera, utilizamos los operadores lógicos y podemos construir las siguientes condiciones, suponiendo que:

A = 1; B = 2 y C = 0;

- a) **A > B**
- b) **C < A**
- c) **C == 0**
- d) **A >= 1**
- e) **B <= 0**
- f) **A != C**

El **literal A**) es falso, porque "A" que vale 1 no es mayor que "B" (que vale 2).

El **Literal B**) es verdadero porque "C" vale 0 y A vale 1.

El **Literal C**) es verdadero porque "C" = 0.

El **Literal D**) es verdadero porque "A" que vale 1 si es mayor o igual que 1.

El **Literal E**) es falso porque "B" no es menor o igual que 0.

El **Literal F**) es verdadero porque "A" si es distinto que "C".

En un programa las **condiciones** se representan mediante el símbolo de la Estructura Selectiva, indicando claramente cuáles serán las dos alternativas: **Verdadero** o **Falso**.

Tipos de Estructuras Selectivas son:

Estructura Selectiva Simple: (sintaxis)

if (**condición**)
 Acción por Verdadero;

Estructura Selectiva Doble: (sintaxis)

if (**condición**)
 Acción por Verdadero;
else
 Acción por Falso;

Estructura Selectiva Múltiples: (sintaxis)

```
if (condición1)
    Acción por Verdadero;
else if (condición2)
    Acción por Verdadero;
...
else
    Acción por Falso n;
```

Instrucción if

Utilizada para evaluar una condición. **if** evalúa el resultado de una **condición** y dependiendo de su resultado (verdadero o falso) **ejecuta** un enunciado simple o compuesto.

Enunciado Simple, utilizado para una sola instrucción:

```
if condición
    (Acción por Verdadero)
else
    (Acción por Falso)
```

Si la condición es verdadera se ejecutarán las instrucciones que se encuentren a continuación; pero si la condición es falsa, se ejecutan todas las instrucciones después de **else**.

Enunciados Múltiples, utilizados para dos o procesos matemáticos y se deben encerrar entre llaves:

if condición

```
{
    (Acciones por Verdadero)
    (Acciones por Verdadero)
}
else
{
    (Acciones por Falso)
    (Acciones por Falso)
}
```

Observe: La diferencia entre enunciados **Simple** y **Múltiples**.

La **Simple** sólo le permite realizar una acción tanto por verdadero como por falso. En las **Múltiples** usted puede realizar más de dos acciones ya sea por **verdadero** o por **falso**.

Estructura Selectiva Simple

Toda Estructura Selectiva tiene dos alternativas: **verdadero** y **falso**. Sin embargo, las Estructuras Selectivas Simple sólo realizan acciones cuando la **Condición es Verdadera**, cuando la **Condición es Falsa** no se realiza ninguna acción, es decir, el programa continúa su flujo normal.

-

Las Estructuras Selectivas **Simples** son de la siguiente forma:

if (condición)

Acción por Verdadero

Observe que por **Verdadero** realiza una acción, mientras que por **Falso** no hace nada.

Ejercicio de aplicación 9

En nuestro país toda persona que tenga más de 64 años de edad tiene un descuento del 50% en los valores de servicios públicos y privados. Elabore un programa que lea el número de meses que adeuda una persona en impuestos municipales y el valor a pagar. Además, lea su edad. Visualice el total a pagar y el total del descuento (si es necesario calcularlo)

```
1 import javax.swing.JOptionPane;
2 public class ejercicio9 {
3     public static void main(String[] args) {
4         int ed, nm, vm, tp;
5         ed=Integer.parseInt(JOptionPane.showInputDialog("Ingrese la Edad --->"));
6         nm=Integer.parseInt(JOptionPane.showInputDialog("Meses que Adeuda --->"));
7         vm=Integer.parseInt(JOptionPane.showInputDialog("Valor por Mes --->"));
8         tp = nm * vm;
9         if (ed>=65)
10            tp=tp/2;
11         JOptionPane.showMessageDialog(null,"El total a Pagar es " + tp);
12     }
13 }
```

Declaramos las variables: edad, número de meses, valor por mes y el total a pagar. Ingresamos la edad, número de meses que adeuda, valor mensual. Realizamos un proceso:

Total a Pagar → $tp = nm * vm$;

Realizamos la **bifurcación**, preguntando si es una persona de la 3ra edad, **if (ed >=65)** (años) por verdadero realizamos una fórmula, **tp=tp/2**, es decir, paga la mitad.

Si es falsa la bifurcación entonces paga completo, es decir, no hay descuento alguno.

Finalmente visualizamos el **Total a Pagar**

Ejercicio de aplicación 10

A un trabajador se le aplica un aumento del 15 % en su salario si este es menor a \$400. Realice un programa presente el sueldo del trabajador.

```
1 import javax.swing.JOptionPane;
2 public class ejercicio10 {
3     public static void main(String[] args) {
4         int su;
5         String nom;
6         nom=JOptionPane.showInputDialog("Ingrese su Nombre: ");
7         su=Integer.parseInt(JOptionPane.showInputDialog("Ingrese su Sueldo --->"));
8         if (su<400)
9             su=su+(su*15)/100;
10        JOptionPane.showMessageDialog(null,nom+ " Usted recibi " + su);
11    }
12 }
```

Ingresamos el sueldo de un empleado (**su**) y el nombre (**nom**); preguntamos si **su<400**, por **verdadero** realizamos el proceso, al sueldo le sumamos el porcentaje del 15%, para luego presentar el **nombre** ingresado más un mensaje y el valor almacenado en la variable de la **suma**.

Ejercicio de aplicación 11

Sara Figueroa va de compras al centro comercial Aventura Plaza y en el local de Aromas y Recuerdos al momento de pagar, el cajero le dice que es el cliente número 1000; por tal motivo recibe un 20% de descuento, siempre y cuando sus compras superen los \$200. Caso contrario no tiene descuento. Determinar cuánto pagará el cliente. Ingresamos el valor de la compra (com) y el nombre (nom); preguntamos si **com>200** por verdadero realizamos el proceso, al valor de la compra le restamos el 20% de descuento. Luego presentamos el nombre **ingresado más un mensaje y el valor de compra a pagar**:

```

1 import javax.swing.JOptionPane;
2 public class ejercicio11 {
3     public static void main(String[] args) {
4         int com;
5         String nom;
6         JOptionPane.showMessageDialog(null,"CENTRO COMERCIAL \n AVENTURA PLAZA \n AROMAS Y RECUERDOS");
7         nom=JOptionPane.showInputDialog("CENTRO COMERCIAL AVENTURA \n"+"Ingrese su Nombre: ");
8         com=Integer.parseInt(JOptionPane.showInputDialog("Valor de su compra --->"));
9         if (com>200)
10            com=com-(com*20)/100;
11         JOptionPane.showMessageDialog(null,nom+" Usted tiene que Pagar " + com);

```

Ejercicio de aplicación # 12

Realizar un programa que calcule el total a pagar por la compra de camisas. Si se compran más de 5 camisas aplican un descuento de un tercio del total de la compra.

```

1 import javax.swing.JOptionPane;
2 public class ejercicio12 {
3     public static void main(String[] args) {
4         int pre,can,tp;
5         String nom;
6         JOptionPane.showMessageDialog(null,"CENTRO COMERCIAL \n MULTIMEDIA");
7         nom=JOptionPane.showInputDialog( "Ingrese su Nombre: ");
8         can=Integer.parseInt(JOptionPane.showInputDialog("Numero de Camisas --->"));
9         pre=Integer.parseInt(JOptionPane.showInputDialog("Precio de la Camisa --->"));
10        tp=pre*can;
11        if (can>5)
12            tp=tp-(tp*33)/100;
13        JOptionPane.showMessageDialog(null,nom+" Usted tiene que Pagar " + tp);
14    } }

```

-

Declaramos el Precio(**pre**), Cantidad(**can**) y el Total a Pagar (**tp**), ingresamos el nombre, el precio y la cantidad de camisas a comprar. Realizamos un proceso para calcular el **total a pagar multiplicando** el **precio** por la **cantidad**.

Luego preguntamos si cantidad es mayor a 5, es decir que cuando se compre 6 o más camisas tendrá el respectivo descuento. Caso contrario no tiene descuento y finalmente presentamos el **Total a Pagar**.

Estructuras Selectivas Dobles

Son aquellas en donde necesitamos usar la cláusula **else**, debido a que debemos de efectuar una **acción por verdadero** y otra **acción por falso**. Su sintaxis es:

```
If (condición1)
{
  Acciones por verdadero;
}
else
{
  Acciones por falso;
}
```

Tanto por **Verdadero** como por **Falso** deben ir entre llaves, ya que realizan operaciones matemáticas.

Ejercicio de aplicación 13

Dado como dato el sueldo de un trabajador, considere un aumento del 15% si su sueldo es inferior a 1000 y de un 12% en caso contrario. Imprima el sueldo con el aumento incorporado.

```
1 import javax.swing.JOptionPane;
2 public class ejercicio13{
3     public static void main(String[] args) {
4         int sue, aum, tr;
5         String nom, car;
6         JOptionPane.showMessageDialog(null, "CALCULO DE SUELDO ");
7         nom=JOptionPane.showInputDialog( "Ingrese su Nombre: ");
8         car=JOptionPane.showInputDialog( "Ingrese su Cargo : ");
9         sue=Integer.parseInt(JOptionPane.showInputDialog("Ingrese su sueldo "));
10        if (sue<1000)
11        {
12            aum=sue*15/100;
13            tr=sue+aum;
14        }
15        else
16        {
17            aum=sue*12/100;
18            tr=sue+aum;
19        }
20        JOptionPane.showMessageDialog(null, nom+ " Tu sueldo es      " + sue +
21            "\n Tienes un aumento de      " + aum +
22            " \n El Total a Recibir es " + tr);
23    }}
24
```

Declaramos las variables para el **sueldo**, **aumento** y **total a recibir**, también para el ingreso del nombre y cargo.

Preguntamos si el sueldo es menor a 1000, por verdadero realizamos un proceso:

Aumento **aum = sue * 15 / 100;**

Total a Recibir **tr = sue + aum;**

Recuerda: que estos procesos deben ir encerrados entre llaves.

Entonces por verdadero vamos aplicar un aumento del 15%.

-
En caso de que los ingresos sean superiores a 1000, el aumento será del 12% y realizamos el mismo proceso, pero con la diferencia del porcentaje. Estos procesos deben ir entre llaves

Finalmente presentamos el nombre ingresado, su sueldo y el aumento.

Ejercicio de aplicación 14

Calcular el mayor de dos números leídos del teclado y visualizarlo en pantalla

```
1 import javax.swing.JOptionPane;
2 public class ejercicio14 {
3     public static void main(String[] args) {
4         int pn, sn;
5         pn=Integer.parseInt(JOptionPane.showInputDialog("Primer Numero "));
6         sn=Integer.parseInt(JOptionPane.showInputDialog("Segundo Numero "));
7         if(pn>sn)
8             JOptionPane.showMessageDialog(null, "El numero " + pn + " es mayor que " + sn);
9         else
10            JOptionPane.showMessageDialog(null, " El numero " + sn + " es mayor que "+pn);
11    }
```

Declaramos e ingresamos dos números enteros **pn**, **sn**. Después preguntamos si **pn>sn**, por verdadero presentamos un mensaje con el número mayor acompañado del número menor, lo mismo ocurre por falso.

Ejercicio de aplicación 15

Ingrese las notas de los 3 trimestres de un estudiante, el nombre y el paralelo. Visualice su suma. Si dicha suma es mayor o igual a 40 visualice el mensaje "Aprobado". Si la suma es menor a 40, visualice el mensaje "Reprobado".

```
1 import javax.swing.JOptionPane;
2 public class ejercicio15 {
3     public static void main(String[] args) {
4         int pt, st, tt, su;
5         pt=Integer.parseInt(JOptionPane.showInputDialog(" Primer Trimestre "));
6         st=Integer.parseInt(JOptionPane.showInputDialog(" Segundo Trimestre "));
7         tt=Integer.parseInt(JOptionPane.showInputDialog(" Tercer Trimestre "));
8         su=pt+st+st;
9         if(su>=40)
10            JOptionPane.showMessageDialog(null, "Felicitaciones Aprobo con " + su);
11        else
12            JOptionPane.showMessageDialog(null, " Ponga de su parte y Reprobo con "+su);
13    }}
```

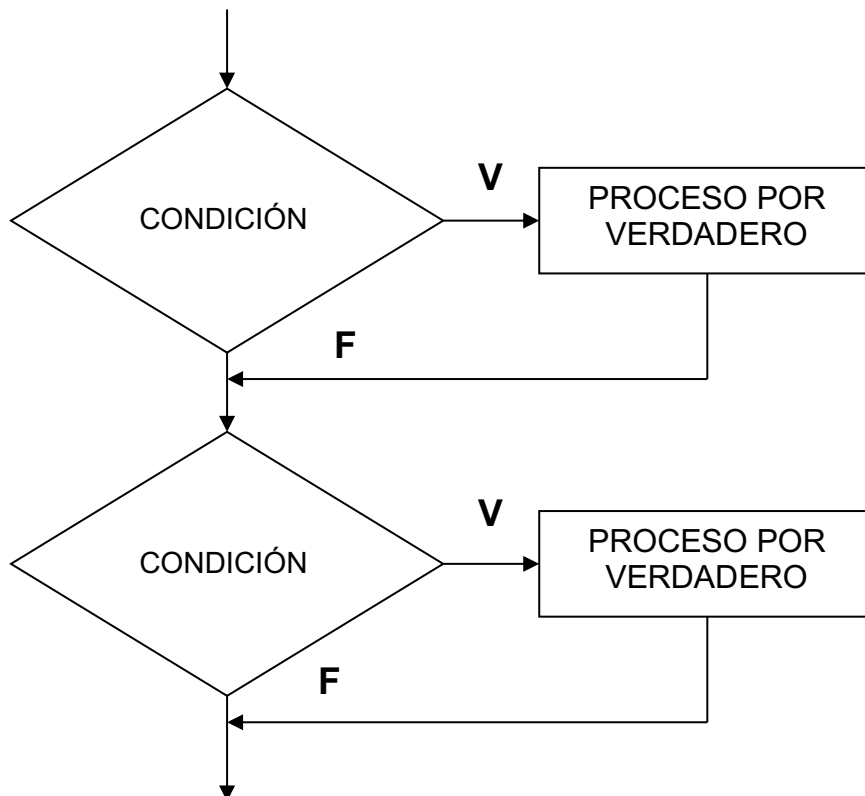
-
Declaramos las notas de los tres trimestres y la suma. Luego ingresamos las **3** notas para sumarlás y almacenarla en la variable **su**.

Preguntamos y sólo cuando la suma sea **mayor** o **igual** a **40**, por verdadero se mostrará el mensaje **Felicitaciones Aprobó con**, caso contrario **Ponga de su parte y Reprobó con**

Estructuras Selectivas Múltiples

Se dice que una **Estructuras Selectivas** es **Múltiple** cuando colocamos varias **Estructuras**, una a continuación de otra.

Las **Bifurcaciones Múltiples** son de la siguiente forma:



Donde podemos colocar una determinada cantidad de **Estructuras Selectivas**, siempre una a continuación de otra. Su sintaxis es.

```
if(condicion1)
    Acción 1;
elseif(condicion2)
    Acción 2;
elseif(condicion3)
    Acción 3;
...
else
    Acción n;
```

Ejercicio de aplicación 16

Realice una página que lea el marcador de un clásico del astillero, es decir, cuántos goles marcó el equipo de Barcelona y cuántos anotó el equipo del Emelec. Visualice el mensaje con el nombre del equipo ganador o si es que hubo empate.

```
1 import javax.swing.JOptionPane;
2 public class ejejercicio15 {
3     public static void main(String[] args) {
4         int gb, ge;
5         gb=Integer.parseInt(JOptionPane.showInputDialog("Goles de Barcelona.... "));
6         ge=Integer.parseInt(JOptionPane.showInputDialog("Goles de Emelec ..... "));
7         if (gb>ge)
8             JOptionPane.showMessageDialog(null, "Gano Barcelona con: " + gb + ", Emelec perdió con: " + ge);
9         else if (ge>gb)
10            JOptionPane.showMessageDialog(null, "Gano Emelec con: " + ge + ", Barcelona perdió con: " + gb);
11        else
12            JOptionPane.showMessageDialog(null, " Hubo empate, Barcelona: " + gb + " Emelec: " + ge);
13    }
}
```

Para este programa ingresamos los goles tanto de Barcelona como de Emelec. Si la condición **gb > ge** es verdadera, significa que Barcelona consiguió un mayor número de goles que Emelec. Entonces **gana** Barcelona. Si la condición es falsa, preguntamos si **ge > gb**. Si es verdadera gana Emelec. Sin embargo, si esta nueva condición también es falsa, entonces significa que hubo un empate.

Analice. Si **gb > ge** es falsa y **ge > gb** también es falso, entonces **BAR** y **EME** son iguales.

Ejercicio de aplicación 17

Realice un programa que lea tres números. Visualice al mayor de ellos.

```
1 import javax.swing.JOptionPane;
2 public class ejercicio16 {
3     public static void main(String[] args) {
4         int pn, sn, tn;
5         pn=Integer.parseInt(JOptionPane.showInputDialog("Primer Numero "));
6         sn=Integer.parseInt(JOptionPane.showInputDialog("Segundo Numero "));
7         tn=Integer.parseInt(JOptionPane.showInputDialog("Tercer Numero "));
8         if (pn>sn)
9             if (pn>tn)
10                JOptionPane.showMessageDialog(null, "El Primer numero es el mayor: " + pn);
11            else
12                JOptionPane.showMessageDialog(null, "El Tercero numero es el mayor: " + tn);
13        else if (sn>tn)
14            JOptionPane.showMessageDialog(null, "El Segundo numero es el mayor: " + sn);
15        else
16            JOptionPane.showMessageDialog(null, " El Tercer numero es el mayor:: " + tn);
17    }
}
```

Ingresamos los 3 números y preguntamos si el **pn>sn** por verdadero preguntamos si **pn>tn**, por verdadero el número mayor es el primero, por falso es el tercer número.



-

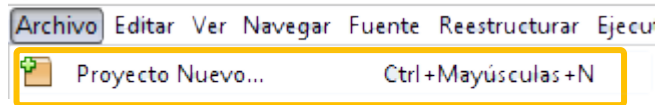
Pero si el primer número no es mayor que el segundo número por falso preguntamos si **sn>tn**, por verdadero el segundo número es el mayor, caso contrario el número mayor sería el tercer número.



PROGRAMACIÓN ORIENTA A OBJETO JAVA (FORMULARIOS)

Pasos para crear un proyecto  NetBeans IDE 6.8

Archivo → Proyecto Nuevo:



A continuación, se presenta el cuadro de diálogo del Proyecto Nuevo, en la sección Categorías, seleccione la carpeta Java y en Proyectos: Aplicación de escritorio Java y finalmente botón Siguiente.



Luego visualizamos el cuadro de diálogo Nueva Aplicación de escritorio, el mismo que sirve para colocar los datos de almacenamiento.

Nombre del proyecto: que generalmente es el nombre de la carpeta donde se almacenará nuestro proyecto.

Ubicación del Proyecto: la ruta (drive) donde se guardará el proyecto.

Escoja el intérprete de órdenes de la aplicación (tipo de aplicación de escritorio), para nosotros será una Aplicación básica, y por último clic en **Terminar**



-

Al finalizar el editor (**Asistente**) genera 3 archivos:

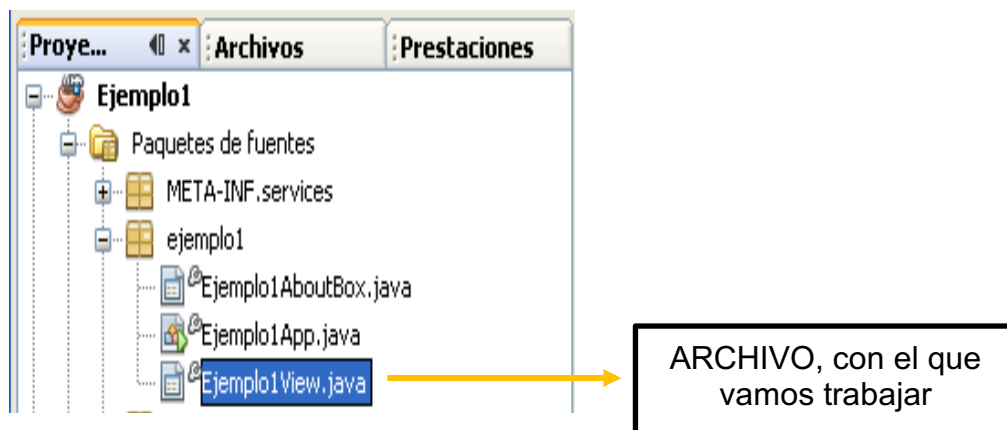
- 1) El archivo de código de la clase base o principal
- 2) El formulario de la ayuda acerca de..., y
- 3) La clase interfaz, donde se diseña el formulario, con que vamos a trabajar.



Los 3 archivos generados comienzan con el Nombre del proyecto o la carpeta en la que se guardó los archivos (Ejemplo1).

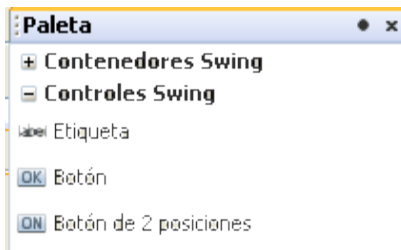
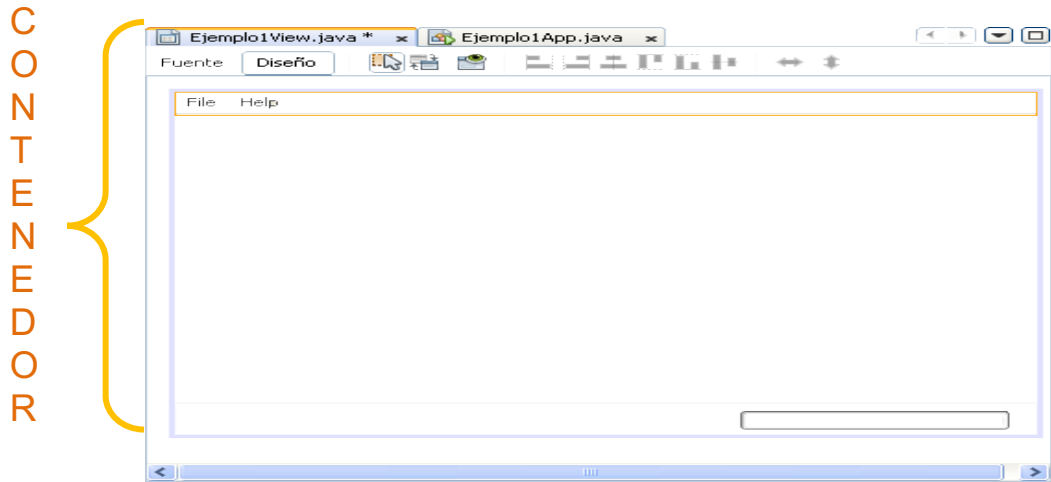
El terminado en el archivo **AboutBox.Java**, es el cuadro de diálogo que presenta la ayuda acerca de.... ;

App.Java, es el archivo de la clase principal y finalmente el View.Java que contiene la codificación necesaria para ejecutar al formulario.



Diseño del Formulario

La barra de herramientas, como se le llama en otros editores de lenguajes visuales - aquí la llamamos Paleta, contiene los íconos a partir de donde se dibujan los controles visuales en los formularios.



Controles Swing

Son clases de objetos o controles parecidos a los de otros lenguajes visuales, que sirven para ingresar datos como: Etiqueta, Botón, Casilla de Activación, Botón de opción, Grupo de botones, Lista desplegable, Lista, Campo de texto, entre otros.

Consideraciones en el diseño de formas

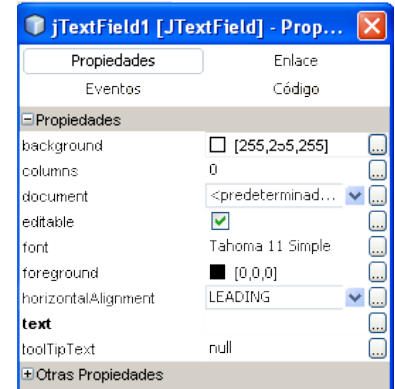


Para dibujar los objetos visuales sobre el formulario que tiene el contenedor para ello damos clic sobre el formulario menú emergente (clic secundario sobre el contenedor) y escogemos la opción que más se acomode a nuestras necesidades. La más recomendada será la de Diseño Absoluto ya que este me permite colocar los objetos donde queramos y con mayor facilidad.

Ventana de Propiedades

Permite modificar las propiedades de los objetos que figuran en un formulario. Una propiedad es una característica de un objeto o control, como: su tamaño, título, color, entre otros.

Una lista diferente aparecerá en la ventana de Propiedades cada vez que seleccione un control o al formulario mismo.



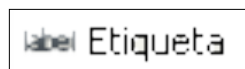
Para visualizar esta ventana debe seleccionar un objeto o control:
clic secundario → Propiedades.

Agregando objetos al contenedor.

Para agregar controles al contenedor, active la Paleta luego el Control Swing, haga clic en el botón correspondiente al control que desee, sitúe el puntero del mouse sobre el formulario y haga clic o arrástrelo para ajustar su tamaño.

Controles más usados

Control Etiqueta jLabel



Este control se utiliza para colocar texto no modificable por el usuario; como títulos, mensajes, junto a los cuadros de texto, entre otros.

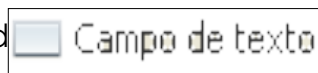
Título (text). Es el texto que aparecerá en el formulario.

Para cambiar el valor a cualquier variable, es suficiente dar clic sobre el valor actual (**jLabel1**) y escribir el nuevo valor.

El resto de propiedades como alineación del texto, fuente del texto, color del texto, se debe dar clic sobre el botón de tres puntos (...) luego escoger los valores que deseamos, y en su caso desplegar la lista y escoger el valor.



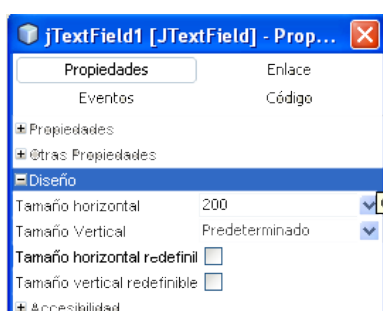
Control Campo de Texto jTextField



Permite al usuario el ingreso de datos y su almacenamiento en variables de memoria o campos; también permite mostrar información que el usuario puede modificar.

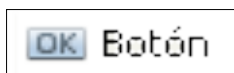


Para asignar nombres a las variables, seleccionamos el campo de texto y recurrimos al menú emergente (clic secundario) y escogemos la opción de Cambiar Nombre de la Variable, y aparece la ventana de diálogo donde digitamos el nuevo nombre de la variable, que será por medio del cual será reconocido en el código o programación.



Tamaño del campo de texto. Podemos definir el tamaño de un campo desde la propiedad columns, que determinan el tamaño por el número de caracteres que se pueden ver; y por la propiedad Tamaño horizontal, donde el tamaño se define en pixeles.

Control Botón Comando JButton



Este control se emplea para iniciar acciones tales como aceptar los valores ingresados en un formulario, cerrar o mostrar una ventana, o cada una de las opciones de una barra de tarea.

Propiedad text que es el título que aparecerá para que el usuario sepa cómo identificar el botón a nivel de vista, mientras el nombre (al igual que las cajas de texto) es la propiedad que ayuda a identificar al objeto al momento de programar.

Editor de Texto

Programa usado para escribir cualquier tipo de texto (sin formato, ej: bloc de notas), en este caso, una codificación en "Java".

Compiladores

Un grupo de programas que se encargan de revisar los errores de una codificación (escritura de un programa, en nuestro caso una clase).

Programa Fuente

Es la codificación original de un programa, en cualquier lenguaje de Programación, en nuestro caso, en “**Java**”, el programa es una clase, y cada clase en un archivo.

Programa Objeto

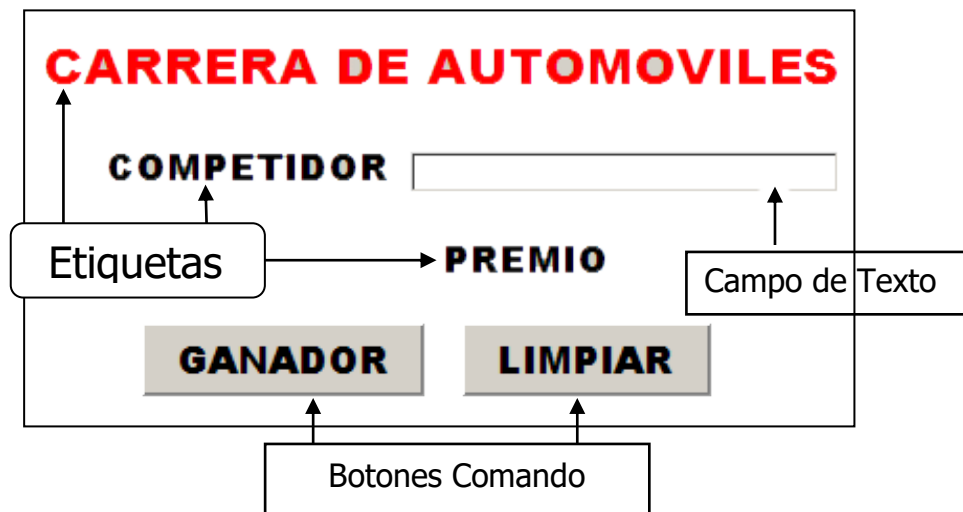
Es el programa en “**Java**”, listo para ejecutar libre de errores, es decir, es el archivo (con extensión **.class**) resultado de la compilación de un programa Fuente.

Procesos Simples I con Formularios

A continuación, un programa que visualiza por pantalla un mensaje.

Empezamos diseñando nuestro formulario, con los objetos: 3 Etiquetas, 1 Campo de Texto y 2 Botones Comandos.

Asignamos las variables com, para el campo de Texto y pre para la Etiqueta (PREMIO).

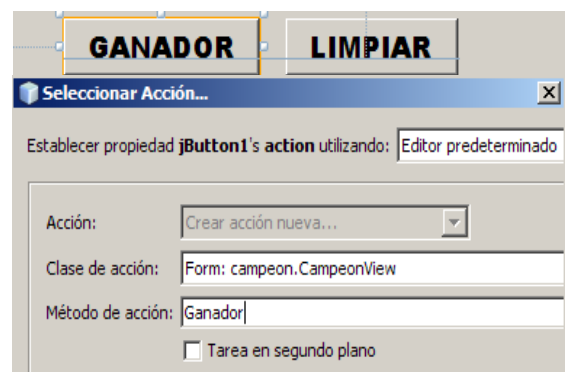


Para agregar la programación a los botones para que se ejecute al momento de darse el evento primordial del objeto (clic), damos doble clic o clic secundario sobre el objeto y escogemos **Seleccionar Acción....**, y aparecerá el siguiente cuadro de dialogo para especificar el método que se ejecutará en su momento, tal como se visualiza en la siguiente ilustración:

Escogemos en Acción:

Crear acción nueva...., que generalmente lleva el nombre del botón que vamos a programar.

En Clase de acción debe estar el nombre del paquete que generalmente es el nombre de la carpeta en minúsculas seguido con punto (.) y el nombre de la clase del formulario (en nuestro caso termina en View.Java).



Método de acción: Aquí va generalmente va el nombre del botón que vamos a programar, es decir, el botón: Ganador. Posterior a ello damos clic en el botón Aceptar y nos aparecerá la ventana para programar el método, es decir, darle las instrucciones que se van a ejecutar.

No olvide que las llaves indican el inicio y el fin del grupo de instrucciones del bloque, en este caso del método Ganador. Debemos escribir por lo tanto las instrucciones entre las llaves { }

```
@Action
public void Ganador() {
    pre.setText("La Moto se la gano:. " + com.getText());
}
```

Recuerda: Las instrucciones deben estar escritas en letra minúscula.

Public

Hace referencia al tipo de clase, es decir, publica.

Void

Orden que llama a la función principal (Ganador)

Llave abierta {

La llave abierta, es el inicio del cuerpo de la declaración de una clase.

Llave cerrada }

La llave cerrada, es el fin del cuerpo de la declaración de una clase.

Punto y Coma (;)

La utilidad de la instrucción punto y coma (;) al final de cada una de estas líneas es la de separar dos instrucciones consecutivas; si usted no la incluye al momento de compilar el programa presentará error.

```
pre.setText ("La Moto se la gano " + com.getText ( ) );
```

pre es el nombre la variable que se le asignó a la etiqueta (PREMIO) dentro del diseñador del formulario, y es donde se va visualizar el mensaje "La Moto se la gano"

Instrucción setText

Esta instrucción se la utiliza para Presentar una expresión, que puede ser un mensaje, una variable o la combinación de las dos.

```
(" MENSAJE " + com.getText ( ) );
```

com, es el nombre de la variable para el campo de texto.

getText () Ingresa el texto al campo de texto.

com.getText (), es decir, lo que ingresamos en la variable **com** del formulario, es extraído y presentado gracias a **setText ()**

Luego damos doble clic al botón LIMPIAR, y realizamos los pasos anteriores (Clic secundario → Seleccionar Acción → Acción: Crear Acción nueva...) y vamos al Editor de Texto y procedemos a codificar lo siguiente:

-

```
@Action  
public void Limpiar() {  
    this.com.setText("");  
    this.pre.setText("");  
    this.com.requestFocus();  
}
```

com.setText (" ");
com, nombre de la variable del campo de texto del formulario.

setText (" ") Presenta un valor de la caja de texto, en este caso lo que está dentro de los paréntesis, es decir, un valor en blanco.

Lo mismo ocurre para la siguiente línea:
pre.setText("");

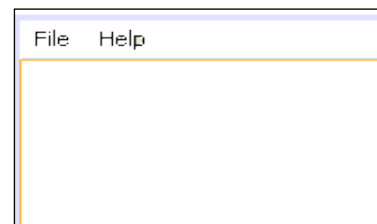
Instrucción requestFocus

Sirve para dar el enfoque a un campo de texto, en este caso a la variable com.

Diseñador de formulario

Procesos Simples II (FORMULARIOS)

Es aquí donde, podrá agregar los objetos o controles al formulario, personalizarlo, ajustando y alineando los controles, entre otros.



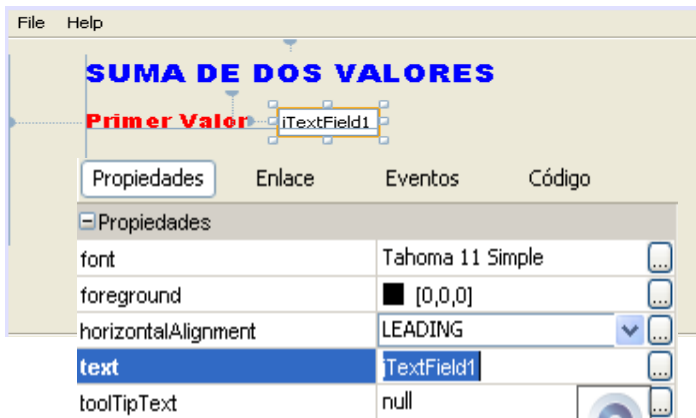
Dentro del diseñador de formulario van los Objetos Controles visuales, más usados como son:

- | | | |
|-----------------------|----------|------------------|
| Etiqueta | o | jLabel |
| Campo de Texto | o | textField |
| Botón comando | o | Button |

Ejercicio de aplicación Diseñemos un formulario.

Necesitamos resolver un gran problema como "Sumar dos valores reales". Lo primero que debemos hacer es diseñar la apariencia del formulario que servirá como interface para interactuar entre el usuario y el sistema. A continuación, debemos darle la funcionalidad a cada uno de los objetos programando los diferentes eventos que se den sobre los objetos como control de ingreso de datos como también de presentación de resultados.

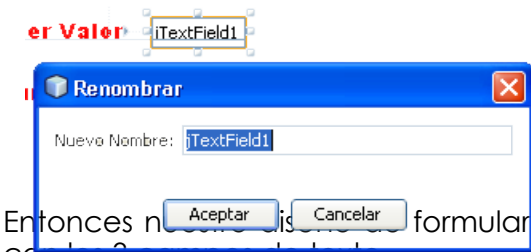
Empezamos diseñando dentro de nuestro Diseñador de Formulario, esto lo hacemos con los diferentes Objetos o Controles Swing, a los mismos que le aplicamos varios formatos. (Tamaño, color, size, entre otros).



Luego seleccionamos el Objeto o Control campo de texto y la ubicamos en el formulario y nos vamos a la propiedad text y borramos jTextField1 para que cuando se ejecute el formulario esta caja de texto se visualice vacía (sin texto)



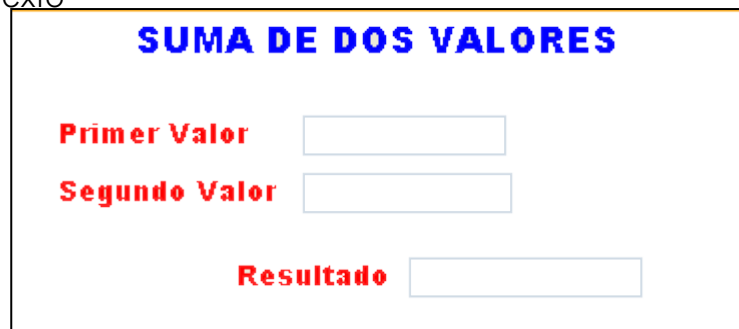
A continuación, seleccionamos el primer campo de texto y clic secundario, opción **Cambiar Nombre de la Variable:**



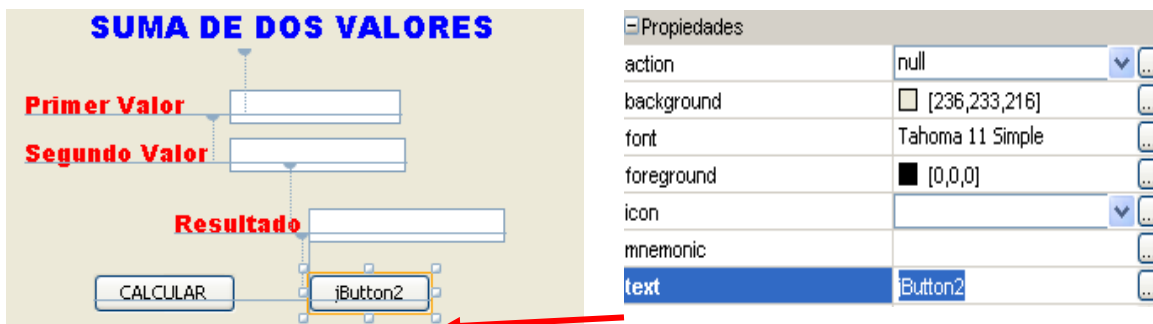
Visualizamos en recuadro Renombrar, en Nuevo Nombre, asignamos el nombre de la variable que vamos a identificar al campo de texto, en este caso pv. Al segundo campo de texto le asignamos sv y al resultado la variable su

Entonces nos vamos al formulario con los 3 campos de texto

ite

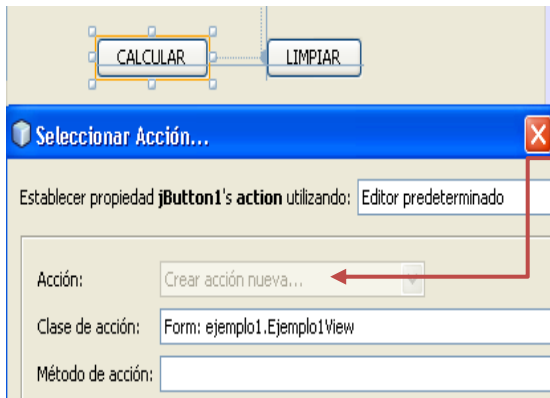


Luego agregamos 2 botones a nuestro formulario, como lo vemos en la siguiente ilustración:



En la Paleta de **Propiedades**, opción **text** ubicamos la leyenda que se visualizará en el **Botón**.

Para agregar la programación a los botones para que se ejecute al momento de darse el evento primordial del objeto (clic), damos doble clic o clic secundario sobre el objeto y escogemos Selección Acción, y aparecerá el siguiente formulario para especificar el método que se ejecutará en su momento, tal como se visualiza en la siguiente ilustración:



En Acción, escoger Crear acción nueva....

En Clase de acción debe estar el nombre del paquete que generalmente es el nombre de la carpeta en minúsculas seguido con punto (.) y el nombre de la clase del formulario (en nuestro caso termina en View.Java).

Método de acción: Calcular

Posterior a ello damos clic en el botón Aceptar y nos parecerá la ventana para programar el método, es decir, darle las instrucciones que se van a ejecutar.

No olvide que las llaves indican el inicio y el fin del grupo de instrucciones del bloque, en este caso del método calcular. Debemos escribir por lo tanto las instrucciones entre las llaves { }

```
@Action
public void CALCULAR () {
|
}
}
```

A continuación, les presento la codificación del método como solución al problema. En nuestro caso como es muy sencillo, no vamos a implementar la clase como solución al problema, sino que lo vamos a hacer directamente en la función asociada con el evento clic del botón CALCULAR.

```
@Action
public void Calcular() {
1 int pv,sv,su;
2 pv=Integer.parseInt(this.pv.getText());
3 sv=Integer.parseInt(this.sv.getText());
  su=pv+sv;
3 this.su.setText(su+"");
}
}
```

Para una mejor explicación del ejercicio, vamos a identificar cada línea de la programación con un número.

La línea 1:

Declaramos las variables que vamos a utilizar en la codificación, dentro de editor de texto. Las mismas que son las que se asignaron a los campos de texto.

La línea 2:

```
pv=Integer.parseInt(this.pv.getText());
```

En el campo de texto de la variable pv, ingresamos texto, entonces la función, **getText** me devuelve el texto del objeto; como es texto debemos convertir dicho valor a Integer con la función **parseInt**.

La línea 3:

Ahora hacemos lo de la línea 2 pero con la variable **sv**.

La línea 4:

Realizamos una suma común y corriente **su = pv + sv**

La línea 5:

```
this.su.setText(su+" ");
```

this hace referencia al objeto actual, en este caso al campo de texto de la variable **su**, y **setText** presenta el proceso de la suma almacenada en la variable su.

Palabra this

Todo objeto o control puede hacer referencia a sí mismo, mediante la palabra clave **this** es decir, hacer referencia al objeto o control actual, es decir, al que está acompañando. Ejemplo:

```
this.su.setText(su+" ");
```

Nos indica que vamos a trabajar con la variable **su**.

Función parseInt

Convierte un texto cuyo contenido sea dígitos, a su valor numérico.

Función getText

Get(Ingresa); Devuelve el texto del objeto o control, en este caso al campo de texto (**pv.getText()**)

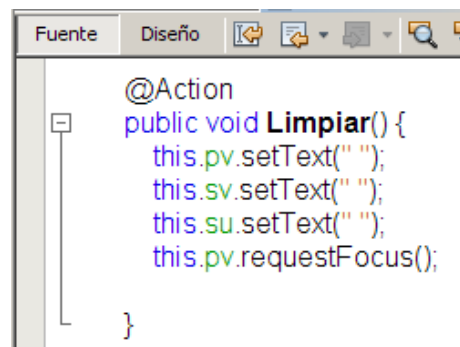
Función setText

Set(Presenta), Establece el contenido del objeto o control en este caso al campo de texto (**su.setText(su+"")**)

Botón LIMPIAR:

Lo que vamos hacer es dejar en blanco a los controles, en este caso los 3 objetos de mi diseño de formulario. Para luego poder ingresar nuevos datos, esto lo hacemos con la función **setText**.

setText, presenta los campos de textos de las variables pv, sv, su, en blanco.



```
@Action
public void Limpiar() {
    this.pv.setText("");
    this.sv.setText("");
    this.su.setText("");
    this.pv.requestFocus();
}
```

Función requestFocus ()

Esta función es la que se encarga de ubicar el curso en un control, para el ingreso de datos, para lo cual debe de anteponerse la variable.

Ejercicio de aplicación # 2

Codifique un formulario que ingrese el nombre, apellido, asignatura, y las 3 notas parciales de un estudiante. Visualice su suma y promedio trimestral.

Recuerda: antes de ingresar los objetos, debes dar clic secundario sobre el formulario y seleccionar Diseño Absoluto.

PROMEDIO DE NOTAS

Nombres MIA VALERIA

Apellidos ESPINOSA FIGUEROA

Asignatura PROGRAMACIÓN 1

Primer Parcial 20

Segundo Parcial 20

Tercer Parcial 20

La Suma es 60.0

Promedio es 20.0

CALCULAR **LIMPIAR**

Identificación de variables en los campos de texto:

Nombres → nom
Apellidos → ape
Asignatura → asi
Primer Parcial → pp
Segundo Parcial → sp
Tercer Parcial → tp
La Suma es → su
Promedio es → pro

Nota: Cuando asignamos las variables estas deben ser las mismas que vamos a utilizar en el diseño del formulario y en la codificación de la fuente, es decir, en el editor de texto.

CALCULAR

Damos doble clic a este botón y se nos presenta la siguiente ventana:

Seleccionar Acción...

Establecer propiedad jButton1's action utilizando: Editor predeterminado

Acción: <ninguna>

Clase de acción:

Método de acción:

Tarea en segundo plano

Acción: Escoger **Crear acción nueva...**

Clase de acción: Debe estar el nombre de la carpeta y el nombre de la clase del formulario.

Método de acción:
Procesar.

Damos clic en **ACEPTAR** y se presenta la siguiente ventana:
Que es donde codificamos el programa fuente.

```
@Action
public void Procesar() {
    int pp, sp, tp, su, pro;
    pp=Integer.parseInt(this.pp.getText());
    sp=Integer.parseInt(this.sp.getText());
    tp=Integer.parseInt(this.tp.getText());
    su = pp + sp + tp;
    pro = su / 3;
    this.su.setText(su+"");
    this.pro.setText(pro+"");
}
```

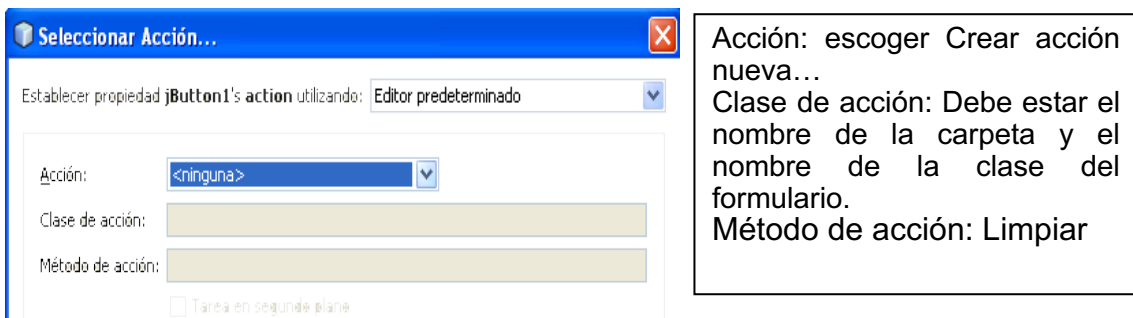
Primero declaramos las variables que vamos a ingresar y las que van a contener el proceso de la suma (su) y promedio (pro).

Las variables pp, sp, tp; son las que vamos a utilizar para ingresar los datos, las mismas que es necesario convertirlas → Integer a parseInt, luego, hacemos referencia al control con la palabra (this) acompañado de la variable, que se ingresaron los datos gracias a la función getText()

Luego realizamos los procesos para la **SUMA** y **PROMEDIO**, los mismos que son almacenados en las variables **su**, **pro** y finalmente visualizamos sus resultados con la palabra clave **this** acompañados de sus variables.



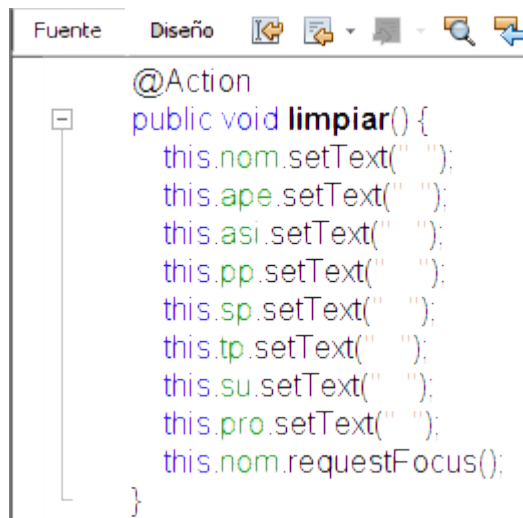
Damos doble clic a este botón y se nos presenta la siguiente ventana:



Que es donde codificamos el programa fuente.

```
Página de Inicio x Ejemplo1App.java x Ejemplo
Fuente Diseño
188
189 @Action
190 public void Limpiar () {
191
192
193 }
```

La codificación del botón **Limpiar** es la siguiente:



```
Fuente  Diseño  [Iconos de herramientas]
@Action
public void limpiar() {
    this.nom.setText(" ");
    this.ape.setText(" ");
    this.asi.setText(" ");
    this.pp.setText(" ");
    this.sp.setText(" ");
    this.tp.setText(" ");
    this.su.setText(" ");
    this.pro.setText(" ");
    this.nom.requestFocus();
}
```

Para hacer referencia a la variable de campo de texto necesitamos a la palabra clave **this**.

A cada campo de texto del formulario es necesario limpiarlo, es decir, darles un valor en blanco y lo hacemos, con la función **setText(" ")**,

nom.requestFocus, es para que el curso se ubique en el campo de texto de la variable nom.



Estructuras Selectivas, Definición

Una Estructura Selectiva es una pregunta con tan sólo dos respuestas posibles: **Verdadero** o **Falso**. Las Estructuras Selectivas se construyen a partir de las condiciones. Una **condición** es una expresión lógica de la siguiente forma:

Variable **operador relacional** Variable/valor

Operadores relacionales son:

Operador Relacional	Significado	Ejemplo
>	Mayor que	10 > 4
<	Menor que	30 < 60
==	Igual	1 == 1
>=	Mayor o igual que	40 >= 30
<=	Menor o igual que	50 <= 100
!=	Distinto a	20 < > 70

En un programa las **condiciones** se representan mediante el símbolo de la Estructura Selectiva, indicando claramente cuáles serán las dos alternativas: **Verdadero** o **Falso**.

Tipos de Estructuras Selectivas son:

```
Estructura Selectiva Simple: (sintaxis)  
if (condición)  
  Acción por Verdadero;
```

```
Estructura Selectiva Dobles: (sintaxis)  
if (condición)  
  Acción por Verdadero;  
else  
  Acción por Falso;
```

```
Estructura Selectiva Múltiples: (sintaxis)  
if (condición1)  
  Acción por Verdadero;  
else if (condición2)  
  Acción por Verdadero;  
...  
else  
  Acción por Falso n;
```

Instrucción if

Utilizada para evaluar una condición. **if** evalúa el resultado de una **condición** y dependiendo de su resultado (verdadero o falso) **ejecuta** un enunciado simple o compuesto.

Enunciado Simple, utilizado para una sola instrucción:

```

if condición
    (Acción por Verdadero)
else
    (Acción por Falso)

```

Si la condición es verdadera se ejecutarán las instrucciones que se encuentren a continuación; pero si la condición es falsa, se ejecutan todas las instrucciones después de **else**.

Enunciados Múltiples, utilizados para dos o procesos matemáticos y se deben encerrar entre llaves:

if condición

```

{
    (Acciones por Verdadero)
    (Acciones por Verdadero)
}
else
{
    (Acciones por Falso)
    (Acciones por Falso)
}

```

Observe: La diferencia entre enunciados Simples y Múltiples. La Simple sólo le permite realizar una acción tanto por verdadero como por falso. En las Múltiples usted puede realizar más de dos acciones ya sea por verdadero o por falso.

Estructura Selectiva Simples.

Toda Estructura Selectiva tiene dos alternativas: **verdadero** y **falso**. Sin embargo, las Estructuras Selectivas Simples sólo realizan acciones cuando la **Condición es Verdadera**, cuando la **Condición es Falsa** no se realiza ninguna acción, es decir, el programa continúa su flujo normal.

Las Estructuras Selectivas **Simples** son de la siguiente forma:

```

if (condición)
    Acción por Verdadero

```

Observe que por **Verdadero** realiza una acción, mientras que por **Falso** no hace nada.

Ejercicio de aplicación 3

Elabore un programa que lea el nombre, el precio individual y el número de entradas al cine a comprar. Adicionalmente ingrese la edad de la persona. Calcule y visualice el total a pagar. Considere que si la persona es de la 3ra. Edad (65 años o más) debe pagar sólo la mitad de todo.

```

@Action
public void Calcular ()
{
    //Declaramos las variables
    int eda, can;
    double pre, des, tot;
    //Conversion de caracteres a numeros
    can=Integer.parseInt(this.can.getText());
    eda=Integer.parseInt(this.eda.getText());
    pre=Double.parseDouble(this.pre.getText());
    tot=can*pre;
    //Preguntamos
    if (eda>=65)
        tot=tot/2;

    this.tot.setText(tot+"");
}

```

Se ingresa el **precio** y el número de entradas a comprar. También se la **edad**. Se calcula el **total a pagar** multiplicando el precio por la cantidad (**tot = can * pre**)

A continuación, realizamos la **Estructura Selectiva**, preguntando si es una persona de la 3ª edad, **if ed >= 65** (años) por verdadero realizamos un proceso.

A continuación, tenemos la codificación del **Botón LIMPIAR**:

```
@Action
public void Limpiar ()
{
    this.pre.setText ("");
    this.can.setText ("");
    this.eda.setText ("");
    this.tot.setText ("");
    this.pre.requestFocus ();
}
```

requestFocus, Sirve para dar el enfoque a un **campo de texto**, en este caso al PRECIO de la variable **pre**.

Hemos incluido una leyenda para **Salir de la Ejecución** del Formulario, el mismo que utiliza la combinación de teclas **CRTL+ Q**

El diseño para este formulario sería el siguiente, con los respectivos nombres de las **variables** de los **Campos de Entrada**.

Dicho de otra manera, cuando presione el Botón **LIMPIAR**, los Campos de Texto de las variables: **pre, can, eda, des, tot**; toman valores en blanco, como esta en la codificación:

```
this.pre.setText (" ");
```

Ejercicio de aplicación 19

Ingrese el nombre, el cargo, el sueldo por hora y las horas trabajadas por un empleado. Sólo si el salario neto a recibir por el empleado es menor a \$200.00 páguesele por concepto de transporte \$50.00 adicional. Visualice el salario neto a recibir.

CALCULO DE SUELDO	
Nombre	<input type="text" value="nom"/>
Cargo	<input type="text" value="car"/>
Horas Trabajadas	<input type="text" value="hr"/>
Sueldo * Horas	<input type="text" value="sh"/>
Salario Neto	<input type="text" value="sn"/>
Bono de Transporte	<input type="text" value="bt"/>
Salario a Recibir	<input type="text" value="sr"/>
<input type="button" value="PROCESAR"/> <input type="button" value="OTRO DATOS"/>	

Luego de ingresar el nombre, el cargo, el sueldo por hora y las horas trabajadas, se efectúa el cálculo del salario neto ($sn = ht * sh$). Si dicho salario es ($sn < 200$) se le suma al salario \$ 50.

Ejemplo: $sh * ht$ es **1.5** y sus horas trabajadas **ht** son **40**, el salario neto a recibir es ($1.5 * 40$) => **60**; entonces la condición es verdadera. A los 60 le sumamos los **50** del bono de transporte y nos da → 110.

```
@Action
public void Procesar ()
{
    double ht, sh, sn, sr, bt=50;
    ht=Double.parseDouble(this.ht.getText());
    sh=Double.parseDouble(this.sh.getText());
    sn=ht*sh;
    if (sn<200)
        bt=50;

    sr=sn+bt;
    this.sn.setText(sn+"");
    this.bt.setText(bt+"");
    this.sr.setText(sr+"");
}
```

```
@Action
public void Limpiar() {
    this.car.setText("");
    this.nom.setText("");
    this.ht.setText("");
    this.sh.setText("");
    this.sn.setText("");
    this.bt.setText("");
    this.sr.setText("");
    this.ht.requestFocus();
}
```

Botón Limpiar:
requestFocus, sirve para dar enfoque

Estructuras Selectivas Dobles

Son aquellas en donde necesitamos usar la cláusula **else**, debido a que debemos de efectuar una **acción por verdadero** y otra **acción por falso**. Su sintaxis es:

```
If (condición1)
{
  Acciones por verdadero;
}
else
{
  Acciones por falso;
}
```

Tanto por **Verdadero** como por **Falso** deben ir entre llaves, ya que realizan operaciones matemáticas.

Ejercicio de aplicación 5

Ejecute un programa que ingrese dos números determinados y visualice el mayor de ellos.

VISUALIZA EL NUMERO MAYOR

Primer Numero
pn

Segundo Numero
sn

El Numero Mayor es..
men1

CALCULARLIMPIAR

Declaramos las variables
int pn, sn;

Es necesario inicializar las variables a ingresar:

```
pn = Integer.parseInt(this.pn.getText( ));  
sn = Integer.parseInt(this.sn.getText( ));
```

Luego realizar la pregunta **if(pn > sn)** y dependiendo de los números ingresados se mostrará el mayor de ellos. En este caso vamos mostrar un mensaje **"El Primer Numero es el Mayor "** + el número almacenado en la variable correspondiente (**pn**) o (**sn**)

```
@Action
public void calcular() {
  int pn, sn;
  pn= Integer.parseInt(this.pn.getText());
  sn = Integer.parseInt(this.sn.getText());

  if (pn > sn)
    this.men1.setText("El Primer Numero es el Mayor " + pn );
  else
    this.men1.setText("El Segundo Numero es el Mayor " + sn);
}
```

Ejercicio de aplicación 6

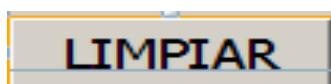
Ingrese las notas de los 3 trimestres de un estudiante, el nombre y el paralelo. Visualice su suma. Si dicha suma es mayor o igual a 40 visualice el mensaje "Aprobado". Si la suma es menor a 40, visualice el mensaje "Reprobado".

Para presentar el mensaje hemos utilizado un control **Label**, para variar

```
@Action
public void calcular() {
    int pt, st, tt, su ;
    pt = Integer.parseInt(this.pt.getText());
    st = Integer.parseInt(this.st.getText());
    tt = Integer.parseInt(this.tt.getText());
    su=pt + st + tt;

    if(su >= 40)
        this.men.setText("Esta Aprobado, Felicitaciones: " + su );
    else
        this.men.setText("Esta Reprobado, lo sentimos : " + su );
}
```

Sumamos los **tres** trimestres. Sólo cuando la suma sea **mayor o igual a 40**, por verdadero se mostrará el mensaje **"Esta Aprobado, Felicitaciones: " + (la suma)**; caso contrario **"Esta Reprobado:" + (la suma)**;



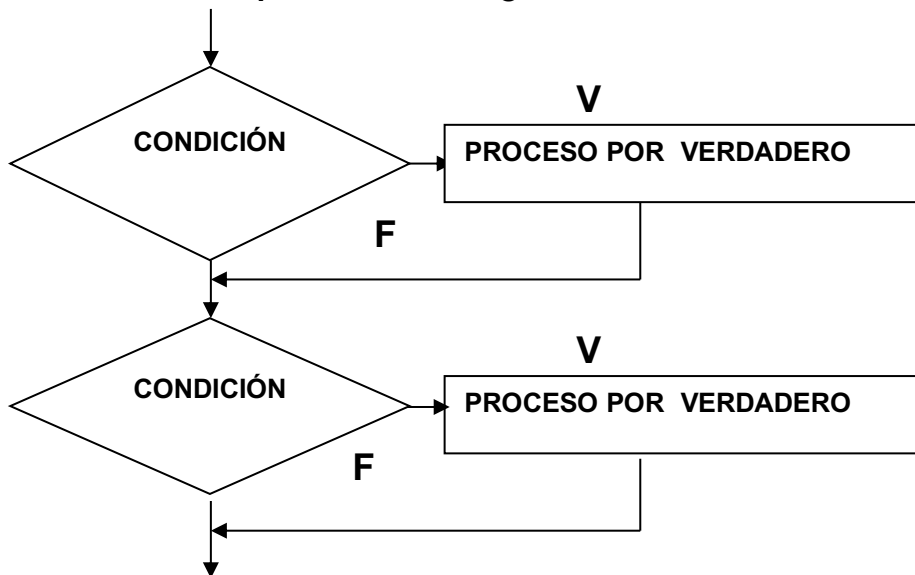
Codificación del **Botón Limpiar**, y al final de la línea damos enfoque a la variable **NOM**

```
@Action
public void Limpiar() {
    this.nom.setText("");
    this.par.setText("");
    this.pt.setText("");
    this.st.setText("");
    this.tt.setText("");
    this.nom.requestFocus();
}
```


Estructuras Selectivas Múltiples

Se dice que una **Estructuras Selectivas** es **Múltiple** cuando colocamos varias **Estructuras**, una a continuación de otra.

Las **Bifurcaciones Múltiples** son de la siguiente forma:



```
if(condicion1)
  Acción 1;
elseif(condicion2)
  Acción 2;
elseif(condicion3)
  Acción 3;
...
else
  Acción n;
```

Donde podemos colocar una determinada cantidad de **Estructuras Selectivas**, siempre una a continuación de otra. Su sintaxis es:

Ejercicio de aplicación 20

Elabore un programa que lea el nombre, paralelo, jornada y las notas del primer y segundo parcial de un estudiante de la Universidad. Calcule y visualice su suma. Si dicha suma es mayor o igual a 14, visualice el mensaje "APROBADO"; si la suma está entre 10 y 13 visualice el mensaje "Suspenso"; y si la suma es menor o igual a 9 el mensaje "Reprobado".

UNIVERSIDAD DE GUAYAQUIL
SISTEMAS MULTIMEDIA

Primer Parcial

Segundo Parcial

La Suma es..

Mensaje

CALCULAR **LIMPIAR**

```

@Action
public void calcular() {
    int pp, sp, su;
    pp = Integer.parseInt(this.pp.getText());
    sp = Integer.parseInt(this.sp.getText());
    su = pp + sp;

    if (su >=14)
        this.men.setText( "Esta Aprobado " + su + " Felicitaciones ");
    else if (su >=10)
        this.men.setText( "Suspenso " + su + " Ponga de su parte ");
    else
        this.men.setText( "Pierde Año " + su + " Lo Sentimos ");

    this.su.setText(su + "");
}

```

Ingresamos, primer parcial **pp** y segundo parcial **sp**. Calculamos su suma. El enunciado indica que si la suma es \geq que 14 se debe visualizar el mensaje **“Esta Aprobado”**.

Por eso la primera condición **suma \geq 14**. Luego dice “...si la suma está entre 10 y 13...”. Si la primera condición **suma \geq 14** es falsa entonces por simple lógica se presupone que la suma será por ley menor o igual que 13.

Insistimos, si la suma no es mayor o igual que 14, entonces es menor o igual que 13.

Sin embargo, si la suma es mayor o igual que 10, entonces automáticamente sabemos que dicha suma está entre 10 y 13.

-

De esta manera la segunda condición **suma >= 10**, por verdadero presenta "**Suspenso**". Finalmente, si la suma no es mayor o igual que 14, ni mayor o igual que 10 indudablemente es menor o igual que 9.

Entonces se visualiza "**Pierde Año**"

```
@Action
public void limpiar() {
    this.pp.setText("");
    this.sp.setText("");
    this.su.setText("");
    this.men.setText("");
    this.pp.requestFocus();
}
```

-
Estructura switch.- Cuando una variable es sometida a varias condiciones, es recomendable usar la instrucción **case**.

La orden **case** permite evaluar varias condiciones y ejecuta la primera que resulte verdadera. Su sintaxis es:

```
switch (condición)
{
    case 1:
        instrucciones 1;
        break;

    case 2:
        instrucciones 2;
        break;

    case 3:
        instrucciones 3;
        break;

    case n:
        instrucciones n;
        break
    default:
}
}
```

En donde, la **condición** es la que, al evaluarla, nos indicará que camino debemos seguir. Además, que, la **condición** constante que acompaña a la palabra reservada **case** debe ser del mismo tipo que **condición**. La cláusula **default** es opcional y puede omitirse en los programas que desarrollemos.

Se puede especificar cualquier número de **case** para las condiciones que se requieran.

Instrucción break. Termina abruptamente la ejecución del ciclo, ignorando todas las instrucciones después del mismo y finaliza el ciclo.

Ejercicio de aplicación 21

Una exportadora tiene los siguientes tipos de camarón:

CODIGO	TIPO DE CAMARÓN	PRECIO x LIBRA
1	EXTRA	\$5.00
2	PREMIUM	\$8.50
3	SUPER PREMIUM	\$12.50

Elabore un programa que ingrese el código del camarón y las libras a exportar. Se debe de visualizar el tipo de camarón, el precio por libras y el total a pagar.

Exportadora de Camarón "KAMARON MULTI"

CODIGO	TIPO DE CAMARON	PRECIO POR LIBRA
1	EXTRA	\$ 5.00
2	PREMIUM	\$8.50
3	SUPER PREMIUM	\$12.50

Codigo de Camarón Libras a Exportar

El Tipo de Camaron es:
 El Precio por Libra es:
 El Total a Pagar es :

```

@Action
public void Comprar() {
    int cod;
    double lib, tp = 0;

    cod=Integer.parseInt(this.cod.getText());
    lib=Double.parseDouble(this.lib.getText());
    switch(cod)
    {
        case 1:
            this.men1.setText("EXTRA");
            this.men2.setText("$5,00");
            tp=5*lib;
            break;
        case 2:
            this.men1.setText("PREMIUM");
            this.men2.setText("$8,50 ");
            tp=8.5*lib;
            break;
        case 3:
            this.men1.setText("SUPER PREMIUM ");
            this.men2.setText("$12,50 ");
            tp=12.5*lib;
            break;
    }
    this.tp.setText("El Total a Pagar es.." +tp );
}

```

```

@Action
public void Limpiar() {
    this.cod.setText("");
    this.lib.setText("");
    this.men1.setText("");
    this.men2.setText("");
    this.tp.setText("");
    this.cod.requestFocus();
}

```

-

Instrucción **switch**, la utilizamos porque la variable (**codcam**) es sometida a varias condiciones y por lo tanto es recomendable usar la instrucción **case**.

Con un **case** preguntamos por el código ingresado. Dependiendo del caso, se asigna el tipo de camarón, el precio. En cada **case** se realiza el cálculo del **Total a Pagar**, multiplicando la cantidad por el precio para cada tipo de camarón.

Al final de cada **case** utilizamos la instrucción **break** que sirve para finalizar el ciclo.



Introducción a la Programación HTML

Programación HTML

Programación HTML, consiste en la programación de páginas de Internet por códigos **HTML**:

Hyper

Text

Markup

Language

Lenguaje Combinado de Hyper Texto

Crear una página HTML

Para crear una página WEB, existen varias formas. Pero generalmente contamos con dos maneras comunes:

1) Es con el **Editor de Texto (Bloc de Notas)** que viene incluido en el Windows y no se necesita instalar ningún otro software, ya que utilizamos los comandos e instrucciones del lenguaje HTML, DHTML o JAVASCRIPT.

2) Es instalando un software específico diseñado para la edición y/o elaboración de páginas WEB, ejemplo FrontPage 2003.

Nosotros utilizaremos el **bloc de notas** que viene incluido en el Windows, el mismo que no formatea el texto, ya que el lenguaje **HTML** está basado en el código ASCII. Claro está que usted puede trabajar con otros programas como el Word, Wordperfect, etc.

También necesitamos de un navegador WEB como el Internet Explorer, Netscape, etc, los mismos que son los encargados de descifrar el código **HTML** de nuestro documento y mostrarlo en la pantalla.

Pasos para crear un documento HTML:

1.- Botón Inicio

2.- **Bloc de Notas**, es aquí, donde se escribe la codificación del documento **HTML** y al momento de guardarlo se antepone la extensión **.HTML** (punto HTML).

- Es preferible no poner acentos, "ñ" ni símbolos extraños.
- No ponerle espacios en blanco al nombre del fichero.
- Y guardar como documento de TEXTO.

Partes de un documento HTML:

Una vez abierto el programa **editor de texto** escriba los códigos o "tags". Las instrucciones, códigos o "tags" de **HTML** van entre dos signos: **<(menor)** y **>(mayor)** y tenemos que poner un código de inicio y uno de cierre.

Todas las páginas **HTML** comienzan con el código de inicio **<HTML>** y terminan con el código **</HTML>**, ejemplo:

<HTML>

.....

.....

.....

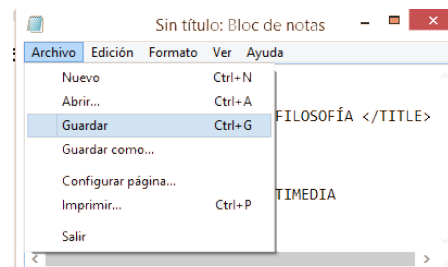
</HTML>

La primera parte de un documento **HTML**, casi siempre empieza con la etiqueta **<HEAD>** y termina con la etiqueta **</HEAD>**. Dentro de estos **tags** se incluye la etiqueta **<TITLE>** que especifica el título de la página, es decir este nombre se visualizará en la barra de título de nuestra página y termina con la etiqueta **</TITLE>**

A continuación, nuestro primer ejercicio realizado en el bloc de notas

<BODY> (cuerpo). Contiene **el inicio y fin del cuerpo de nuestra página** y empieza con la etiqueta **<BODY>** y termina con la etiqueta **</BODY>**, y es donde se coloca el contenido de nuestro documento como son: texto, imágenes, videos, sonidos, enlaces, scripts, etc. Ejemplo:

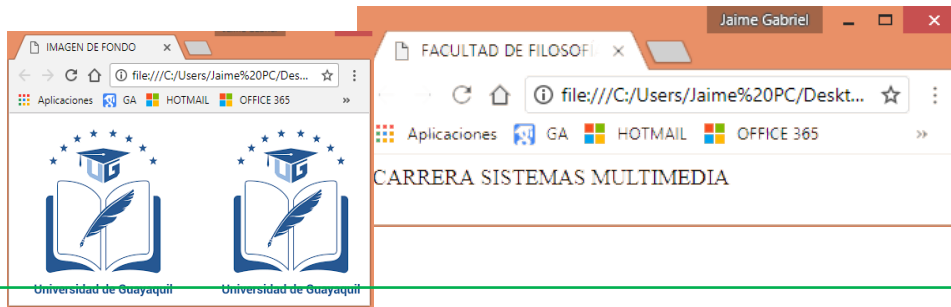
Luego procedemos a guardar nuestro documento:



Seleccionamos el lugar donde guardar nuestro documento, en este caso en el Escritorio. Recuerda al momento de escribir el nombre de nuestro documento debemos digitar la extensión **(punto) .HTML** y damos clic en el botón guardar.

En nuestro escritorio se visualizará el siguiente icono de nuestra Primera Página Web

Procedemos a dar doble clic a nuestro icono e inmediatamente visualizaremos nuestra página WEB, y en la barra de título vemos el nombre de nuestra página (**FACULTAD DE FILOSOFIA**) y el contenido de nuestro texto **"CARRERA SISTEMAS MULTIMEDIA"**



Características de la etiqueta **<BODY> </BODY>**

Dentro de estas etiquetas es donde se escribe el contenido de nuestro documento, es decir, el texto, y donde podremos definir el fondo, el tipo de fuente, tamaño, color y la alineación de nuestra página HTML.

Instrucción **BACKGROUND="RUTA"**

BACKGROUND= Estable el fondo de nuestra página; la **"RUTA"** es la dirección donde usted tiene almacenada su imagen, la misma que es necesario definir su ubicación. Se puede dar el caso que su imagen no ocupe toda nuestra página, cuando esto ocurra, se presentara en estilo mosaico.

Su sintaxis es:

<BODY BACKGROUND="\IMAGEN \ UG-NUEVO.PNG">

BACKGROUND → Define el fondo de mi página.

"\IMAGEN → Es el nombre de la carpeta que contiene mi archivo de tipo PNG, que esta almacenada en nuestro Disco Duro.

\UG-NUEVO.PNG"> → Es el nombre de la imagen (**UG-NUEVO**) con su extensión. Cada vez que quiera ubicar una imagen de fondo es necesario especificar su respectiva ubicación y extensión. Ejemplo:

```

Ejemplo: Bloc de notas
Archivo Edición Formato Ver Ayuda
<HTML>
  <HEAD>
    <TITLE> IMAGEN DE FONDO </TITLE>
  </HEAD>
  <BODY BACKGROUND="\IMAGEN\UG - NUEVO.PNG" > </BODY>
</HTML>

```

Se presenta nuestro fondo, en este caso la Minerva y se la visualiza en forma de mosaico, ya que la imagen no ocupa toda la pantalla.

Dar Formato a las Letras:

Para cambiar los atributos de las letras utilizamos el comando ** ... ** (fuente o tipo de letra). El tamaño o el color son atributos del elemento tipo de letra.

Tipo de Fuente

Para especificar el tipo de fuente necesitamos el comando **FACE**. Su sintaxis es:

```
<FONT FACE = "TIPO DE FUENTE"> AQUÍ VA EL TEXTO </FONT>
```

Ejemplo:

```
<FONT FACE = "ARIAL BLACK"> PROGRAMACIÓN EN HTML </FONT>
```

Se visualizará el texto PROGRAMACIÓN EN HTML con el tipo de fuente Arial Black

Pasar a la siguiente línea:

Cuando queremos pasar el texto a otra línea podemos utilizar el "tab" **
** que no tiene "tab" de cierre.

Veamos como separamos el siguiente texto:

Carrera Sistemas Multimedia

DIGITAMOS así:

```
Carrera <BR>
Sistemas <BR>
Multimedia <BR>
```

Nuestra página **HTML** se presentará:

```
Carrera
Sistemas
Multimedia
```

8.9 Poner Titulares.

Los titulares o encabezamientos de los textos son otra forma de modificar el tamaño del texto.

La orden para encabezamiento es **"H1"** para un titular muy grande, **"H2"** para uno grande **"H3"**, **"H4"**, **"H5"** y para letra pequeña **"H6"**. Ejemplo:

-

<H1>Encabezamiento muy grande</H1>

Encabezamiento muy grande

<H2>Encabezamiento grande</H2>

Encabezamiento grande

<H6>Encabezamiento muy pequeño</H6>

Encabezamiento muy pequeño

Introducción a Javascript

JavaScript.

Es un lenguaje de programación considerado y elaborado únicamente para el uso de Internet. Se lo escribe dentro del código **HTML** de las páginas Web; y es interpretado y ejecutado por el navegador del usuario.

Para poder programar en este lenguaje (**SCRIPTS**), es necesario tener algunas nociones del lenguaje **HTML**.

Usted no necesita ningún tipo de programa especial para programar en **JavaScript**, solo es necesario un editor de texto, el mismo que se utiliza para programar en **HTML**.

Windows tiene todo lo necesario para poder programar en **JavaScript**. Aquellos que tienen otro sistema operativo, pueden encontrar en Internet fácilmente las herramientas necesarias.

Introducir Script. La programación de **JavaScript** se realiza adentro del editor de texto utilizando los propios códigos de **HTML**, entre de las siguientes etiquetas:

<SCRIPT >

.....

.....

</SCRIPT>

Los códigos y las funciones de **JavaScript** deben ir entre estas dos etiquetas. Los **SCRIPTS** se deben escribir dentro del cuerpo de página, es decir entre:
<BODY> AQUÍ SE ESCRIBEN LOS SCRIPTS </BOBY>

La etiqueta **<SCRIPT>** tiene un atributo **Language**, el mismo que nos indica el lenguaje que estamos utilizando.

Formularios. Los formularios permiten solicitar información al visitante desde adentro de una presentación Web. Estos formularios estarán compuestos por varios campos como informaciones que deseamos obtener.

Para crear formularios necesitamos de las etiquetas **<FORM> ... </FORM>** y dentro de estas van todas las variables de entrada. Dentro de un formulario podemos incluir: cuadros de textos, botones de selección, menús de opciones, etc.

Entrada de Datos. Para especificar los elementos de entrada de datos se utiliza la etiqueta **<INPUT>**, con el siguiente formato:

-
<INPUT TYPE="button" NAME="boton" VALUE="haga clic aquí"

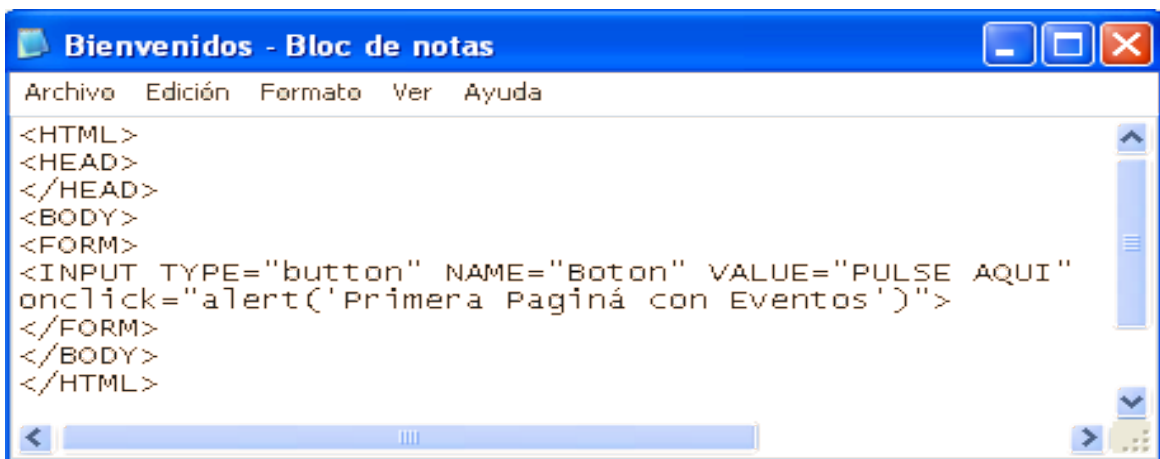
TYPE.- Especifica el tipo de campo de entrada que vamos a utilizar. Ejemplo: Button, Password, Checkbox, Radio, una variable, etc.

NAME.- Es el nombre de la variable introducida en el campo, es decir, el nombre de los ejemplos anteriores.

VALUE.- Valor que se visualizará en el tipo de campo ingresado.

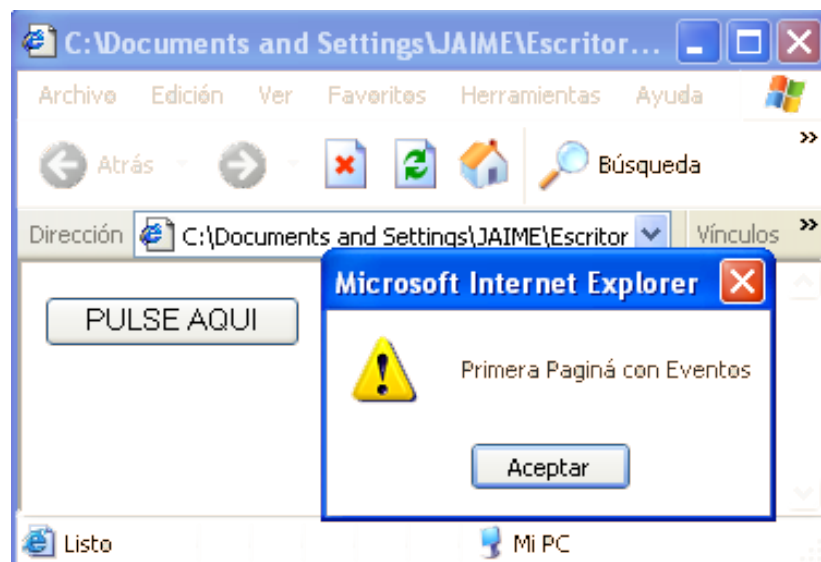
Eventos. Son acciones que realiza el usuario, existen varios tipos de eventos: la pulsación de un botón, el movimiento del mouse o la selección de texto de página.

OnClick. Es un controlador de eventos, es decir, cuando el usuario acciona el botón "PULSE AQUÍ" el evento click se dispara y ejecuta el código que tenga entre comillas. El método **alert** es el que se encarga de mostrar el mensaje en la pantalla ("Primera Página con Eventos")



```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" NAME="Boton" VALUE="PULSE AQUÍ"
onclick="alert('Primera Pagin  con Eventos')">
</FORM>
</BODY>
</HTML>
```

Ejemplo de Eventos:



Descubre tu próxima lectura

Si quieres formar parte de nuestra comunidad,
regístrate en <https://www.grupocompas.org/suscribirse>
y recibirás recomendaciones y capacitación



   @grupocompas.ec
compasacademico@icloud.com

compAs
Grupo de capacitación e investigación pedagógica



@grupocompas.ec
compasacademico@icloud.com


```
<div class="bigPhotoDiv">
  <table cellpadding="0" cellspacing="0" >
    <tr>
      <td valign="middle">
        
        
      </td>
    </tr>
  </table>
</div>
```

ISBN: 978-9942-33-159-5



@grupocompas.ec
compasacademico@icloud.com

compas
Grupo de capacitación e investigación pedagógica

Los Comentarios.- <!-- //-->

Son de mucha utilidad y generalmente se especifican como comentarios el nombre del programa, el del programador, fecha de elaboración, o la descripción de algún proceso. Los comentarios no afectan la ejecución del programa y son opcionales. Están comprendidos entre:

<!-- Aquí se escriben los comentarios //-->

Instrucción eval

Extrae el valor numérico de una expresión.

Instrucción function

Es la respuesta a un evento.

Instrucción reset

Devuelve los valores de un formulario a su estado inicial, es decir, si usted le asigna un valor de **0 (value=0)** al inicio del formulario y cuando pulse el botón **LIMPIAR** la instrucción **reset** se dispara y visualiza el valor inicial asignado (**0**)

Ejercicios de Aplicación 21

Elabore una página que lea dos valores. Calcule y visualice su suma.

```
Archivo Edición Formato Ver Ayuda
<!-- Elab. Sistemas Multimedia -->
<html>
  <title>EJEMPLO # 1</title>
<body>
  <h1>SUMA DE DOS NUMEROS </h1>
  <! ++++++ INGRESO DE DATOS ++++++ !>
  <form name = "suma" >
    Ingrese Primer Numero <input name = pn value = 0 <br><br>
    Ingrese Segundo Numero <input name = sn value = 0 <br><br>

    <! ++++++ APLICAR FORMULA ++++++ !>
  <script>
    function proceso ( )
    {
      document.suma.su.value = eval(document.suma.pn.value) + eval(document.suma.sn.value)
    }
  </script>

  <! ++++++ DESARROLLO DEL BOTON CALCULAR ++++++ !>
  <input type=button name=boton value = CALCULAR onclick="proceso ( )" <br><br>

  <! ++++++ VISUALIZAR EL RESULTADO ++++++ !>
  La Suma es <input name = su value = 0 <br><br>

  <! ++++++ BOTON LIMPIAR ++++++ !>
  <input type = reset value = LIMPIAR <br><br>

</body>
</html>
```

-

Empezamos con un comentario:
<!--Elab por Jaime Espinosa -->

Incluimos un título:
<h1> Suma de Dos Números</h1>

Creamos un formulario con el nombre de suma:
<form name =" suma">

y procedemos a ingresar los datos
**Ingrese Primer Número: <input name=pn value=0

**
**Ingrese Segundo Número: <input name=sn value=0

**
<input es para crear un cuadro de texto.
name=pn, es el nombre de la variable
value=0, es el valor que tendrá mi variable
**

**, es para el salto de línea
<script>
function proceso ()
function es la respuesta a un evento, en este caso **proceso**
{
Llave abierta, inicio del proceso
document.suma.su.value=
eval(document.suma.pn.value) + eval(document.suma.sn.value)
document, nombre del documento actual.
suma, nombre del formulario
su, nombre de la variable que contendrá la fórmula
value, valor del cuadro de texto.
eval extrae una cadena de caracteres
}
Llave cerrada, fin del proceso
</Script>

**<input type=Button name=Boton value=Calcular Onclick=" proceso ()"

**
Esta instrucción me permite diseñar un botón con el nombre **Calcular**.

Onclick=" Proceso ()", es el controlador del evento, es decir, cuando el usuario accione el botón **Calcular** el evento click se dispara y ejecuta el código que tenga entre comillas" **Proceso ()**"

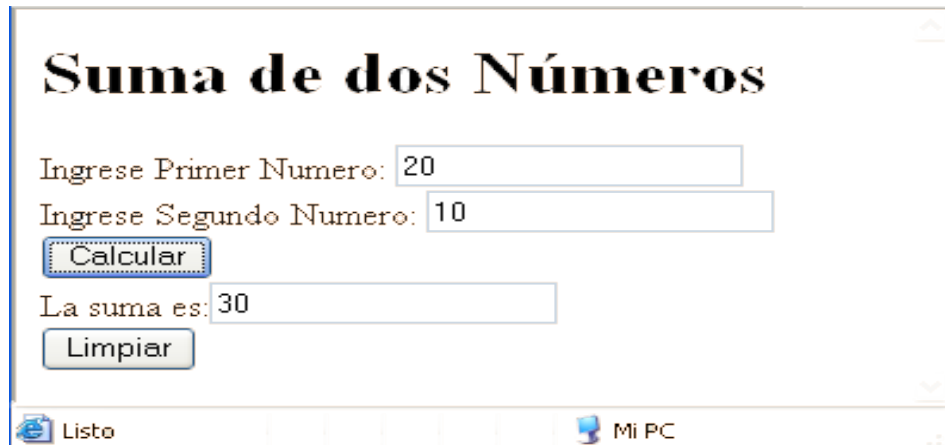
**La suma es: <input name=su value=0

**

Visualiza el resultado de la suma en un cuadro de texto.

<input type=reset value=Limpiar>

Botón que limpia los cuadros de texto, y asigna un valor de **cer**



Ejercicio de Aplicación 22

Ejecute una página que ingrese dos números. Calcule y presente sus cuatro operaciones fundamentales

```
operaciones - Bloc de notas
Archivo Edición Formato Ver Ayuda
<!-- Elab. por Jaime Espinosa-->
<html>
<title>Ejemplo#2</title>
<body>
<h1>OPERACIONES BÁSICAS</h1>

<form name="basi">
Ingrese Primer Numero: <input name=pn value=0 <br> <br>
Ingrese Segundo Numero: <input name=sn value=0 <br> <br>

<script>
function proceso()
{
document.basi.su.value = eval(document.basi.pn.value)+eval(document.basi.sn.value)
document.basi.re.value = eval(document.basi.pn.value)-eval(document.basi.sn.value)
document.basi.mu.value = eval(document.basi.pn.value)*eval(document.basi.sn.value)
document.basi.di.value = eval(document.basi.pn.value)/eval(document.basi.sn.value)
}
</script>

<input type=Button name=boton value=Calcular onclick="proceso()" <br> <br>

La Suma es :<input name=su value=0 <br> <br>
La Resta es :<input name=re value=0 <br> <br>
La Multiplicación es:<input name=mu value=0 <br> <br>
La División es :<input name=di value=0 <br> <br>

<input type=reset value=Limpiar <br> <br>

</form>
</body>
</html>
```

Empezamos con un comentario.

Luego agregamos un encabezado a nuestra página.

-

Para el ingreso de datos, es necesario crear un formulario, con la instrucción:

<form name="basi" >

Ingresamos los datos a nuestras variables **pn, sn**.

La instrucción **input** me permite diseñar un **CUADRO DE TEXTO**.

Dentro del **script** trabajamos con una **función** que la he llamado **proceso()**, y realizamos las fórmulas correspondientes, para la suma, resta, multiplicación y división:

$$su = pn + sn$$

$$re = pn - sn$$

$$mu = pn * sn$$

$$di = pn / sn$$

Para diseñar un botón necesito la instrucción:

<input type=button, que permite diseñar un botón Comando.

value=Calcular, es lo que se visualizará dentro del botón Comando.

OnClick ejecuta lo que está entre comillas, es decir, "**proceso()**", que es donde se realizan los procesos de sus 4 operaciones fundamentales.

Finalmente presentamos los cuadros de textos con los valores de sus cuatro operaciones básicas.

Y diseñamos el botón **Limpiar**, que sirve para darles el valor de **cero** a los cuadros de texto.

