



Introducción al Lenguaje de Programación JAVA



ESTRUCTURA DE CONTENIDOS

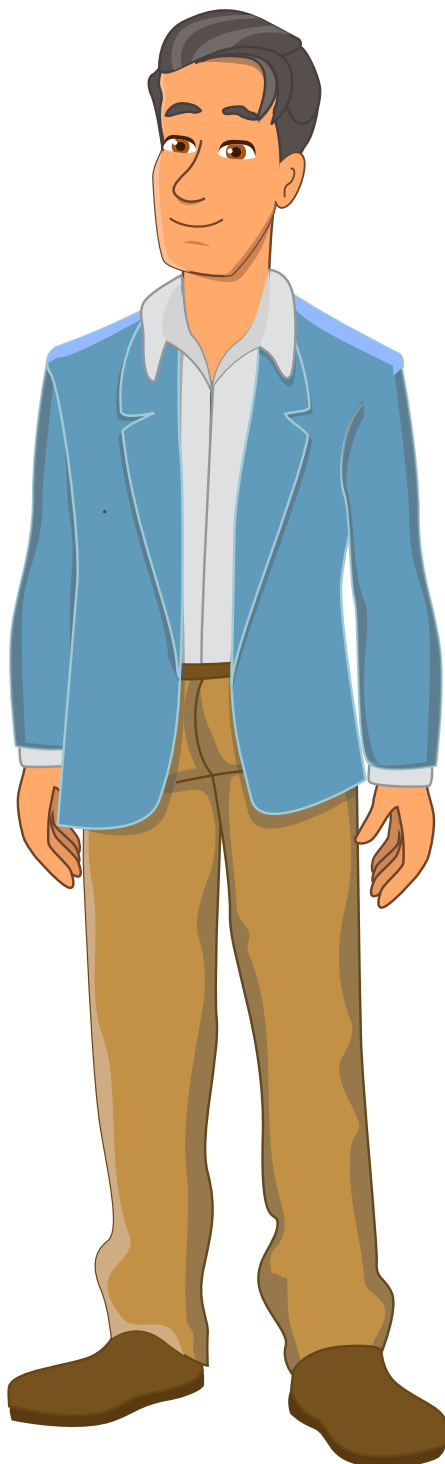
	Pág.
Introducción	3
Mapa de contenido	4
1. JAVA en el tiempo.....	5
2. Elementos de la plataforma JAVA.	6
3. Estructura de un programa en JAVA	8
3.1. Las clases.....	8
3.2. Los métodos.....	8
3.3. Normas básicas.....	9
4. Tipos de datos en JAVA.....	10
5. Simil entre un diagrama de flujo y JAVA.....	11
6. Programar en JAVA desde NetBeans 7.X.	11
7. Transformación de diagrama de flujo a programación en JAVA.....	15
8. Transformación de un algoritmo desarrollado en DFD a una interfaz de desarrollo con lenguaje JAVA.....	18
9. Transformación de un algoritmo desarrollado en LPP a una Interfaz de desarrollo con lenguaje JAVA.....	21
Glosario	26
Bibliografía.....	27
Control del documento	28

INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA

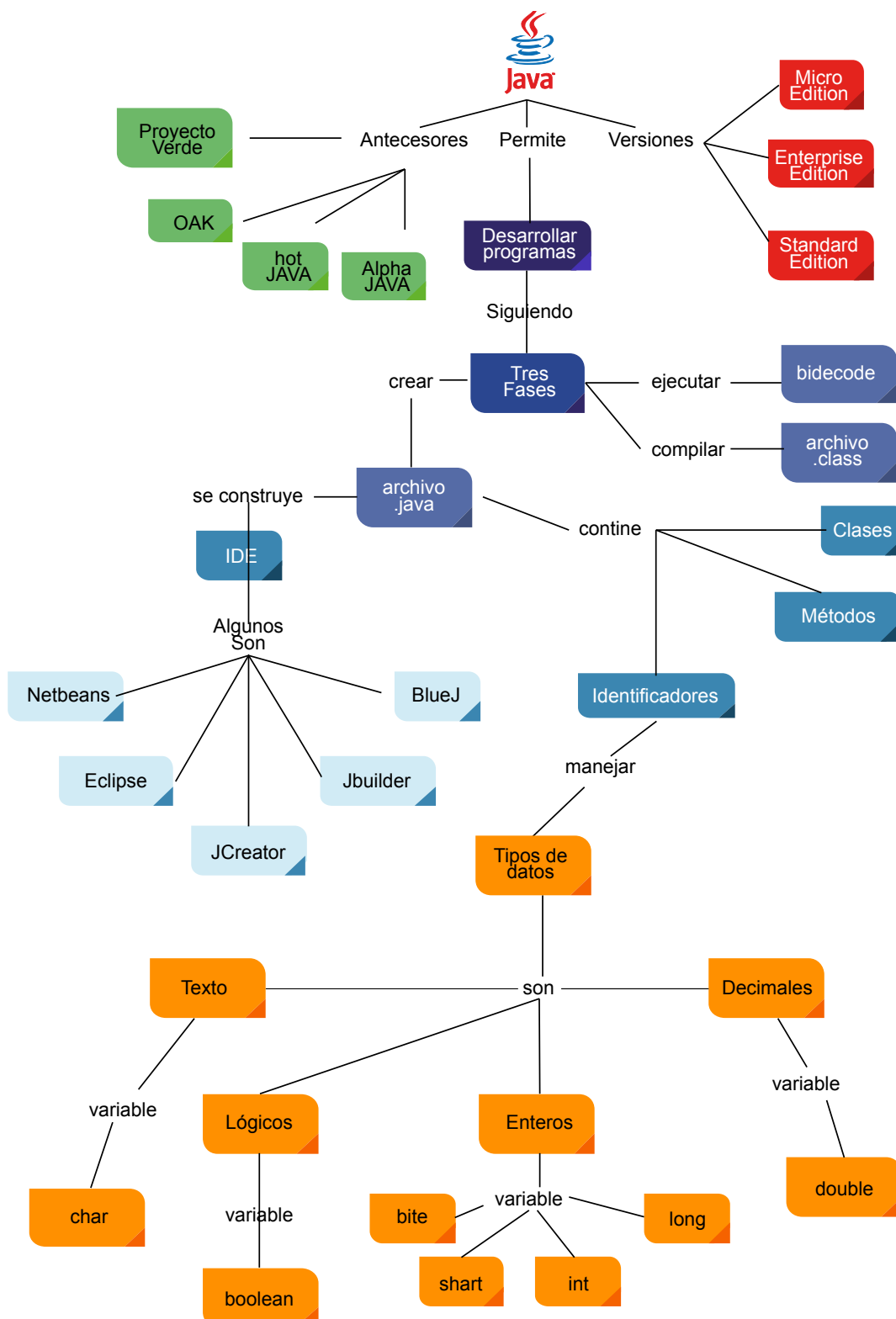
INTRODUCCIÓN

Un lenguaje de programación es un conjunto de expresiones, símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de esos elementos, permitiendo definir procesos que son ejecutados por un computador para la realización de algún objetivo específico.

Java es un lenguaje de programación orientado a objetos y desarrollado por Sun Microsystems, posteriormente fue Adquirido por Oracle y en la actualidad es uno de los más populares con aproximadamente 9 millones de usuarios alrededor del mundo.



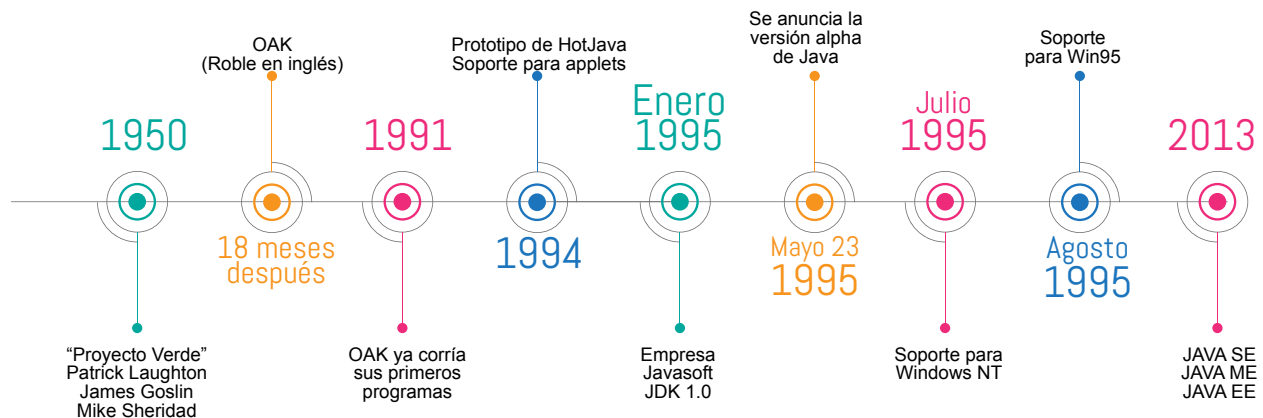
MAPA DE CONTENIDO



DESARROLLO DE CONTENIDOS

1. JAVA en el tiempo.

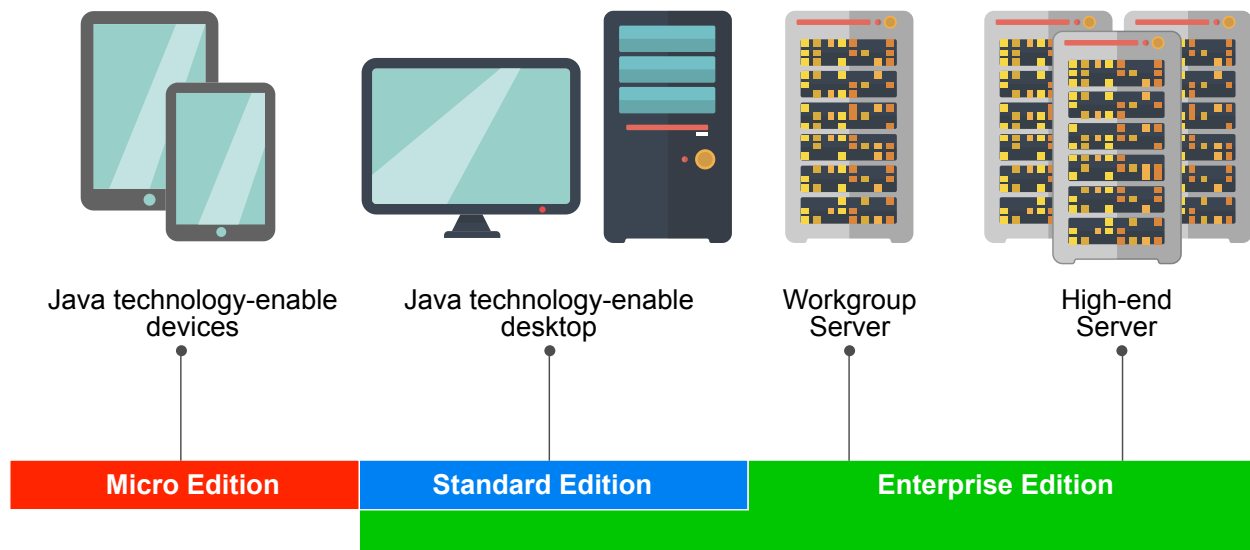
La línea de tiempo que se presenta a continuación, representa la evolución histórica de JAVA.



Como se puede observar este lenguaje de programación está en desarrollo desde 1950, cuando tres ingenieros, uno de ellos de la empresa "Sun Microsystems" iniciaron con el proyecto denominado "Proyecto Verde".

Después de muchos años de trabajo y por cuestiones de propiedad intelectual, se asigna el nombre de JAVA.

Sun Services, empresa creadora de esta herramienta, representa su avance respecto a las plataformas tecnológicas que maneja, de la siguiente manera:



2. Elementos de la plataforma JAVA.

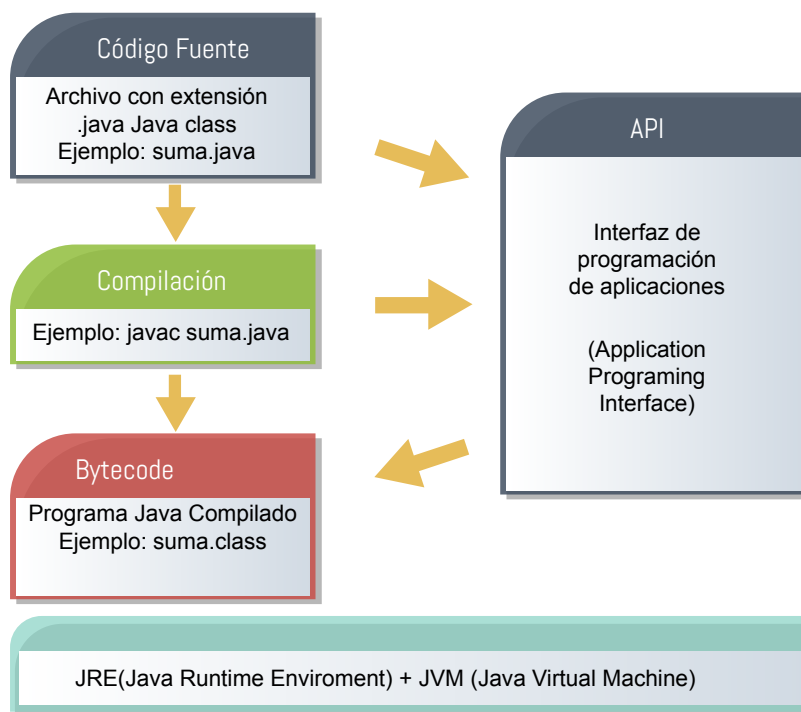
En el proceso de desarrollo en esta plataforma se incluyen términos como: código fuente, compilador, Bytecode, API, JRE (Entorno de Ejecución) y Máquina Virtual (JVM).

El proceso de codificación en JAVA, básicamente consiste en lo siguiente:

- Crear un archivo .java con las líneas de programación necesarias para dar solución al problema planteado. Por ejemplo, para sumar dos números se crearía un java class con el nombre suma.java. Es importante aclarar que las instrucciones utilizadas deben ser reconocidas por el API de java.
- Compilar la clase creada en el paso anterior. Compilar es, que el computador convierta las instrucciones digitadas en el lenguaje (código fuente) a código de máquina. La instrucción utilizada para este proceso es `javac suma.java`. Esta instrucción (`javac` y el nombre de la clase) genera un archivo adicional con extensión .class.
- El archivo .class se conoce como Bytecode y es interpretado por la máquina virtual (JVM), esto ocurre en el entorno de ejecución (JRE).

El JRE y la Máquina Virtual son los encargados de que los programas compilados en JAVA corran en cualquier sistema operativo.

La figura a continuación representa el proceso de codificación en JAVA:



Para poder escribir programas en esta plataforma, se necesita un entorno de desarrollo JAVA, el cual puede ser descargado desde la página:



<http://www.oracle.com/technetwork>

Las opciones de descarga pueden ser:

- a. Descargar únicamente el JDK: Esta primera opción solo le permitirá correr las aplicaciones desarrolladas en JAVA.
- b. b. Descargar el JDK + Netbeans: Netbeans es uno de los Entornos Integrados de Desarrollo (IDE), estos IDE permiten desarrollar aplicativos de manera más rápida.

Existen varios IDE, para trabajar en JAVA, algunos de ellos son:

IDE	LOGOSÍMBOLO	PÁGINA DE DESCARGA
Netbeans	 NetBeans	https://netbeans.org/downloads/
Eclipse	 eclipse	https://eclipse.org/downloads/
JCreator	 JCreator XINOX software	http://www.jcreator.org/download.htm
JBuilder	 JBuilder	https://jbuilder-2007-enterprise-for-window.waxoo.com/
Jblue	 BlueJ	http://bluej.org/download/download.html

NOTA: La funcionalidad de los vínculos depende de los proveedores de esos productos. Su uso en este documento es únicamente de carácter informativo.

Cuando ya se tiene instalado el JDK, requisito indispensable para correr los archivos de java, en el computador se genera la siguiente estructura, se explicarán a continuación algunos de estos componentes.

3. Estructura de un programa en JAVA.

En un programa desarrollado en JAVA se pueden identificar elementos como comentarios, definición de clases, definición de métodos y sentencias.

Los Comentarios: un programa en JAVA generalmente inicia con un comentario, el delimitador para los comentarios al inicio del programa es `/*` y el de final de comentario `*/`.

Se recomienda, que el comentario inicial del programa sea una breve descripción de lo que hace el mismo. Estos comentarios son ignorados por el compilador, pero son muy útiles para el programador.

Un comentario puede ocupar una o varias líneas, para el caso de varias líneas el delimitador es `/*` espacio para los comentarios `*/`. Para el caso de comentarios de una sola línea el delimitador es `//`.

3.1. Las clases.

Cuando se crea una clase utilizando cualquiera de los IDE para trabajar JAVA (netbeans, jCreator, etc) inmediatamente después de los comentarios iniciales, se visualiza la clase.

Una clase podría asimilarse como un archivo donde se incluyen todas las sentencias necesarias para dar solución a un problema.

El identificador de una clase sería `public class nombre {` y termina con el carácter `}`.

3.2. Los métodos.

Los métodos se pueden asimilar como una secuencia de instrucciones o líneas de código, que permiten desarrollar un proceso. Todo método tiene un nombre, seguido de paréntesis `()` e inicia con el carácter `{` Y termina con el carácter `}`.

Existen métodos vacíos y métodos con argumentos, si un método tiene argumentos, éstos se ubicarían dentro de los paréntesis, un ejemplo de la declaración de métodos sería, para el caso de un método vacío: `public void calcular(){` , y para un método con argumentos: `public void suma(int num1, int num2).`

Todas las clases que van a presentar algún resultado al usuario final deben contener el método `main`, y en este método se deben incluir las sentencias a ejecutar cuando se corre el programa.

3.3. Normas Básicas.

Antes de realizar ejercicios básicos en JAVA, se deben manejar algunas tips, en pro de la calidad en el desarrollo de software:

- Todo se maneja por clases: se recomienda que el nombre de la clase inicie con Mayúscula. Ej Suma.java.
- Las clases inician con {, y terminan en,}.
- Documentar el código: para comentarios de una sola línea, este va precedido de // . Ej. //Ejercicio para sumar dos números.

Para comentarios de múltiples líneas /* comentario
*de múltiples líneas
*/

- Las instrucciones terminan en punto y coma.
- Los métodos tienen un inicio que se representan con el carácter {y un fin, representado con el carácter}.

Las preguntas y los ciclos inician con el carácter {y terminan con el carácter}.

Para el nombre de los identificadores o variables:

- Todo identificador debe empezar con una letra y estar seguida de más letras o números.
- El nombre de la variable debe iniciar con minúscula, en el caso que el nombre de la variable sea de más de una palabra la inicial de las siguientes palabras debe ser en mayúscula.

Ejemplo: valor, valorCompra, colorProducto.

- Es conveniente utilizar nombres apropiados para las variables, buscando la legibilidad del programa, con solo leer el nombre de la variable se debe dar la idea de la función que ésta tiene.
- Cada variable tiene un tipo, estos tipos de datos se relacionan en la tabla a continuación.

4. Tipos de datos en JAVA.

Tipo	Tipo de Variable	Descripción	Bytes ocupados en memoria	Rango	Ejemplo
Lógicos	Boolean	Para variables que tendrán la opción true o False	1 byte		boolean esColombiano=true;
Texto	char	Para variables que almacenan un solo carácter (letra, signo, ?)	2 bytes		char sexo="m"
Enteros	byte	Para variables con valores enteros menores o iguales a 127	1 byte	-128 y 127	byte edad=50
	short	Para variables con valores enteros menores o iguales a 32767	2 bytes	-32768 y 32767	short kilometrosdia= 1200
	int	Para variables con valores enteros menores o iguales a 2.147.483.647 Una tardej de identidad no estaría en este rango	4 bytes	-2.147.483.648 y 2.147.483.647	int valorProducto= 500000
	long	Para variables con valores enteros menores o iguales a 223.372.036.854.775.807	8 bytes	-9.223.372.036.854.775.808 y 9.223.372.036.854.775.807	long gananciaAnual= 1147483648
Decimales	double	Números con unas 15 cifras decimales	8 bytes	De - 1, 79769313486232E308 a - 4,9405645841247E324E-324 y de 4,9405645841247E324E-324E a 1, 79769313486232E308	double definitiva=4.5134

5. Simil entre un diagrama de flujo y JAVA.

DIAGRAMA DE FLUJO	DESCRIPCIÓN		EJEMPLO CON SINTAXIS EN JAVA
	Declaración de inicio de la clase	1	<pre>public class ClsEjemplo {</pre>
Entrada de datos	Leer dato	1 2	<pre>nombre=JOptionPane.showInputDialog ("Digite su nombre");</pre>
Proceso	variable1 = variable2;	1	<pre>res = nombre;</pre>
Impresión	Imprimir variables res	1	<pre>System.out.println("res");</pre>
	Fin de la clase	1	<pre>}</pre>
	Estructura Condicional doble	1 2 3 4 5	<pre>if (a>b){ System.out.println("Estoy imprimiendo el valor de "+a); } else{ System.out.println("Estoy imprimiendo el valor de "+b); }</pre>
	Estructura con número Finito de repeticiones	1 2	<pre>for (int i = 1; i <=3; i++) { }</pre>

En el próximo tema se va a trabajar ejercicios básicos de cada uno de los conceptos de algoritmos, se iniciará con algoritmos básicos, luego se pasará a estructuras condicionales sencillas y compuestas y se terminará en estructuras repetitivas.

6. Programar en JAVA desde NetBeans 7.X.

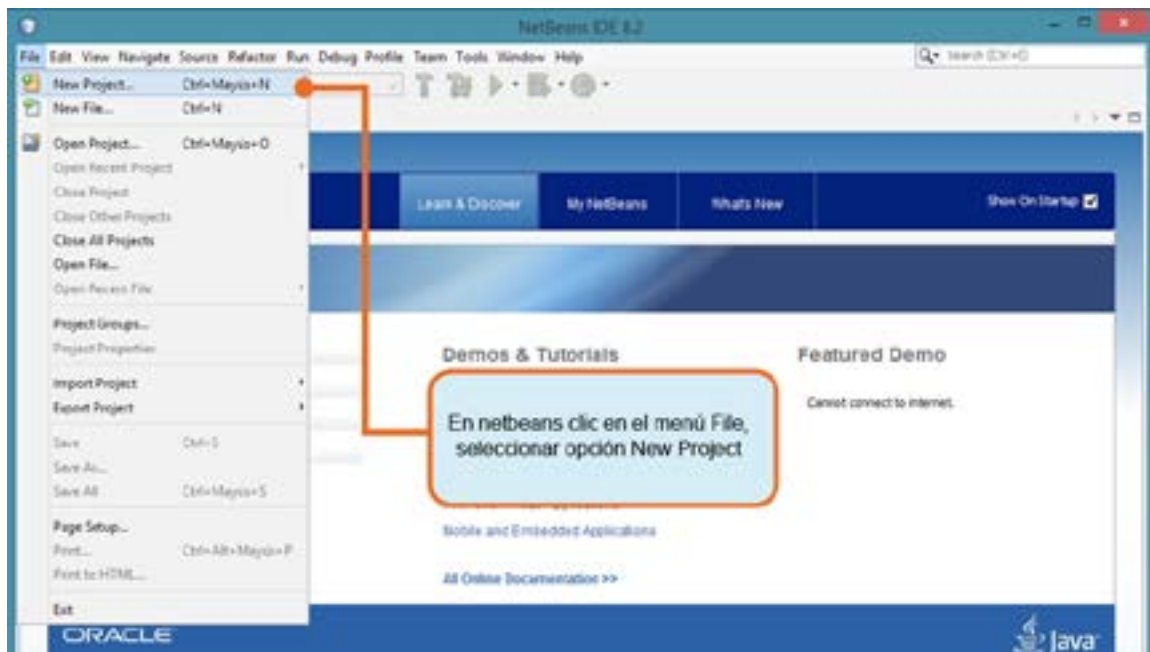
Una vez se tiene instalado el IDE de java, para nuestro caso Netbeans 7.x, cuando se habla de 7.x quiere decir que puede ser alguna de las versiones 7.1, 7.2. 7.3 dependiendo de la versión que tenga instalado el computador.

Estas versiones funcionan de manera muy similar, a continuación, se va a crear el primer programa en java.

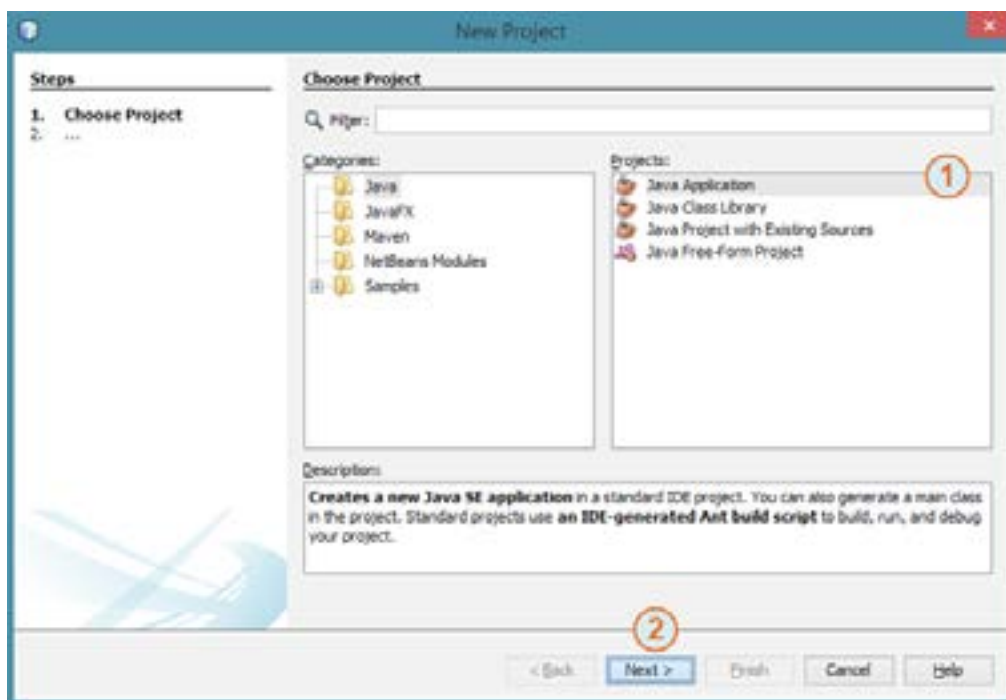


Doble clic en el icono de Netbeans 7.x NetBeans IDE 7.1.2

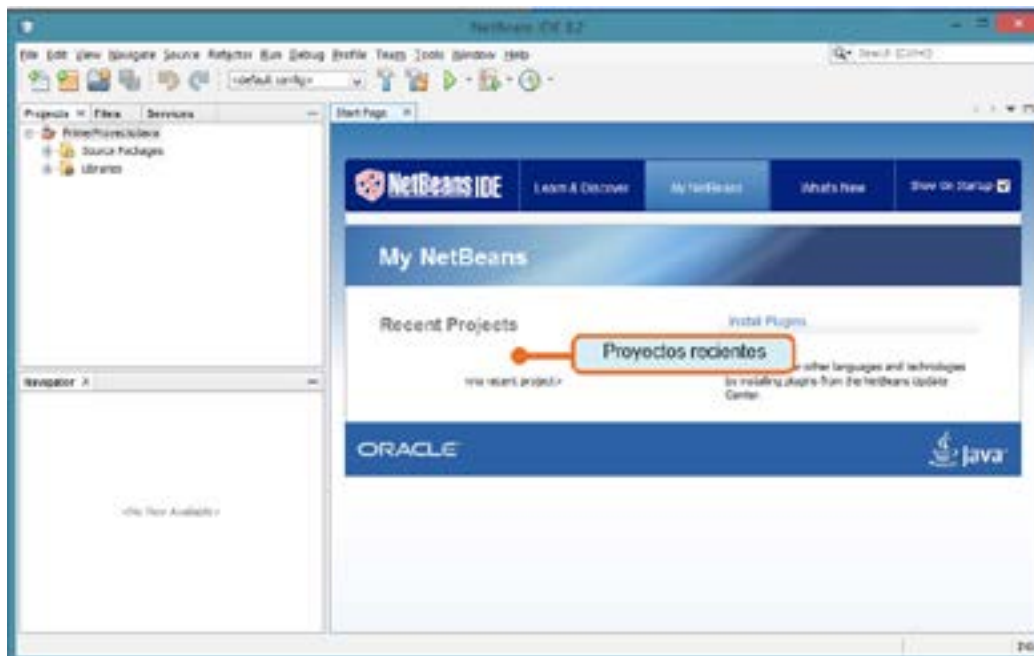
En netbeans clic en el menú File, seleccionar opción New Project:



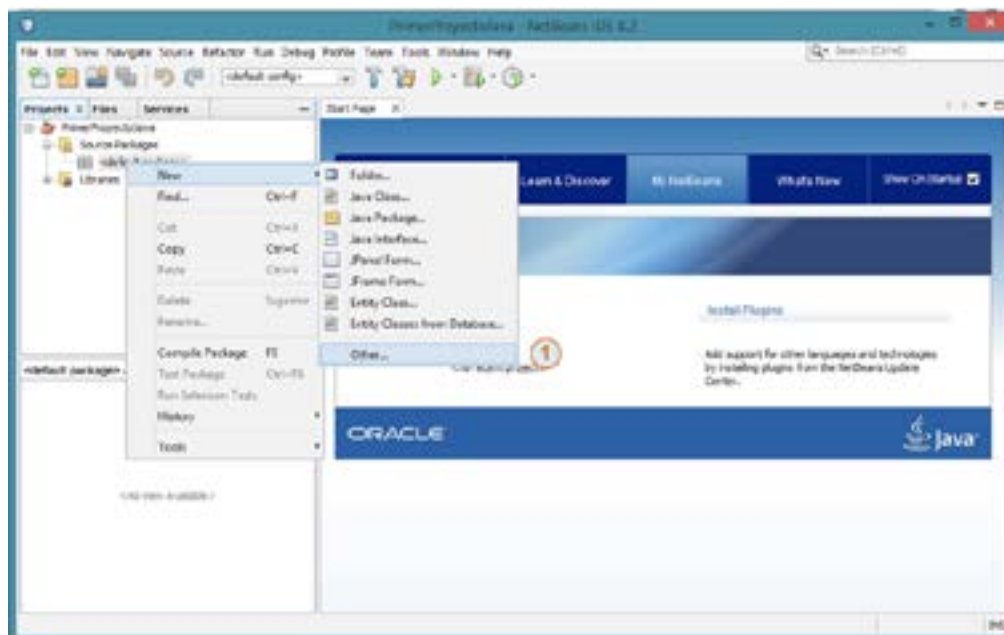
Automáticamente aparecerá la siguiente presentación:



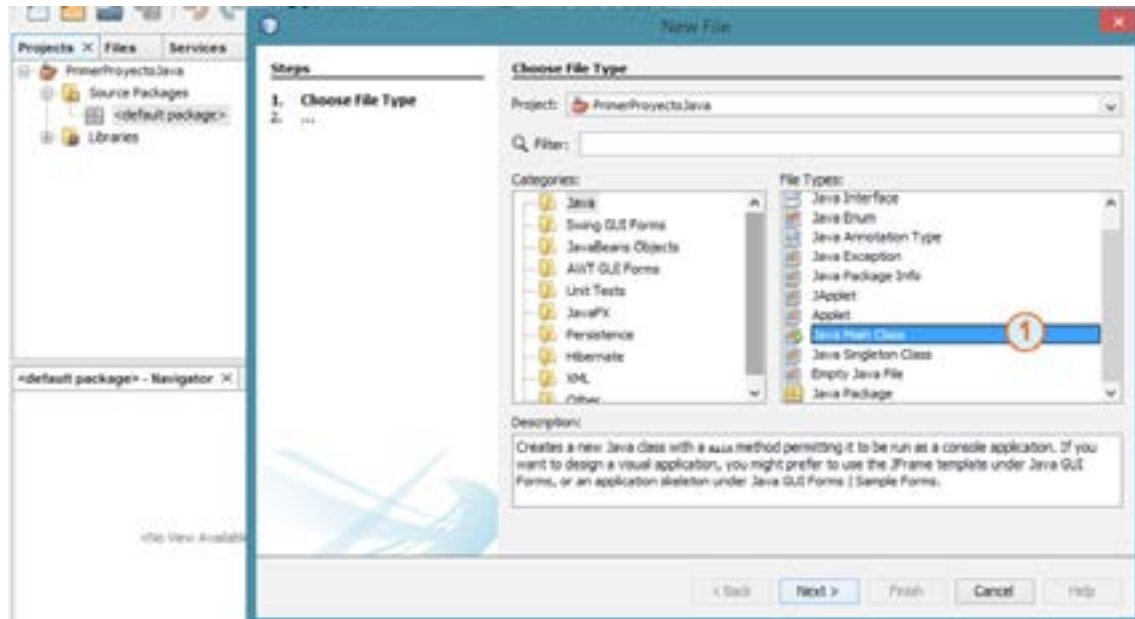
Al finalizar la creación del primer proyecto aparecerá la siguiente estructura:



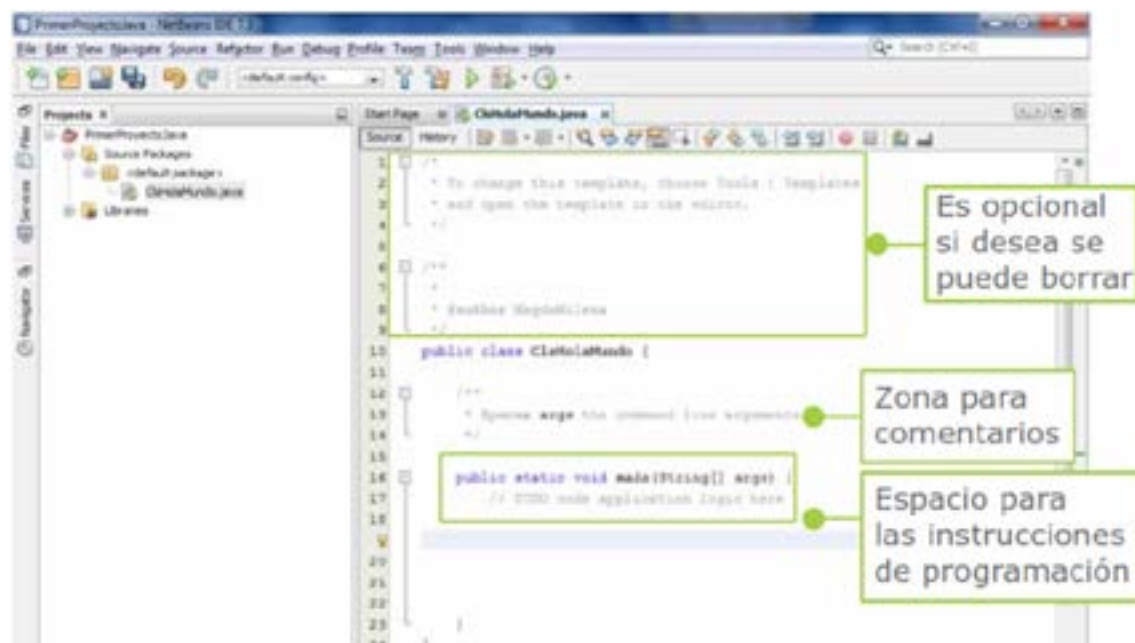
Ahora a crear una clase, dando clic derecho sobre Source Packages se selecciona la opción New Java Main Class:



Aparece la siguiente presentación para que se asigne el nombre de la clase y click en finish:



Aparecerá la siguiente estructura:

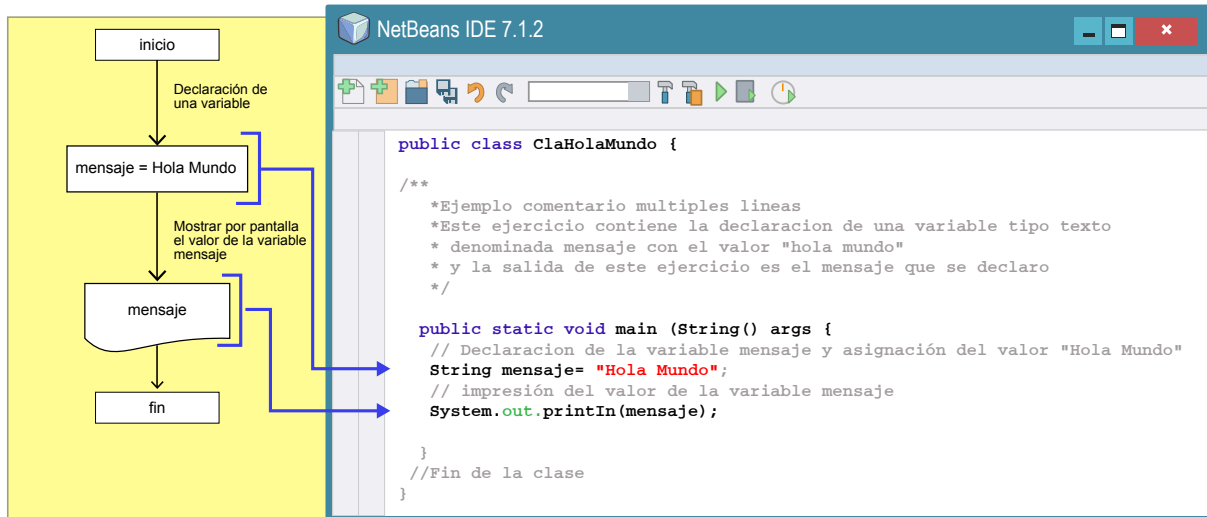


Ya con la clase creada se procede a codificar y una vez se tiene el código construido se ejecuta la clase, con la combinación de teclas shift + F6. En la página siguiente se presenta un ejemplo básico para iniciar la programación en JAVA.

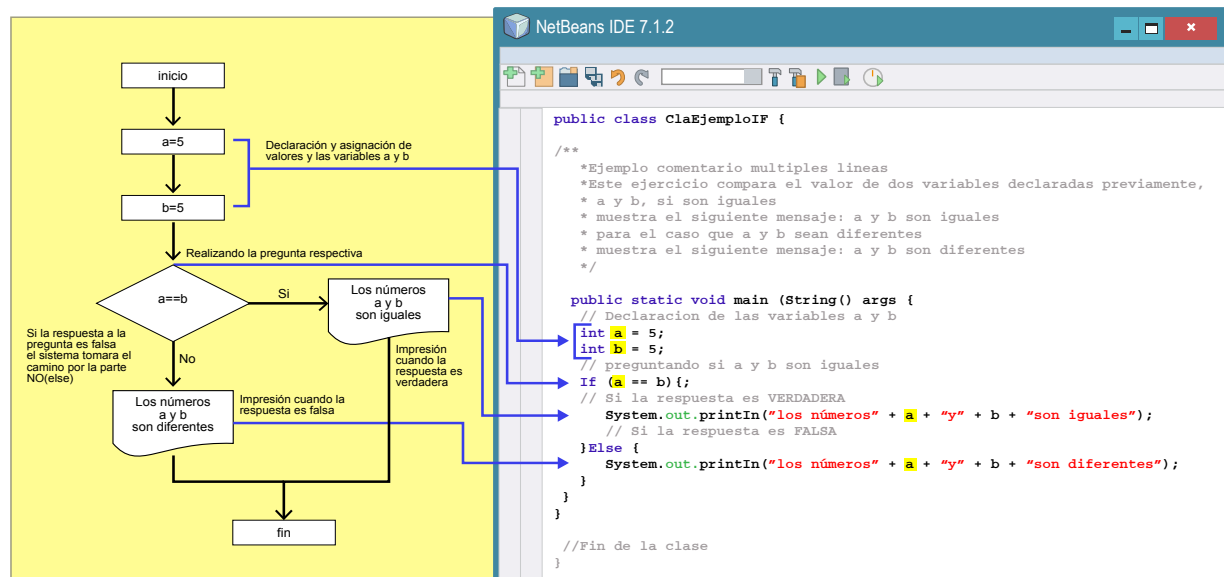
7. Transformación de diagrama de flujo a programación en JAVA.

Ejemplo, diagrama de flujo vs JAVA, estructura cíclica “hacer mientras”:

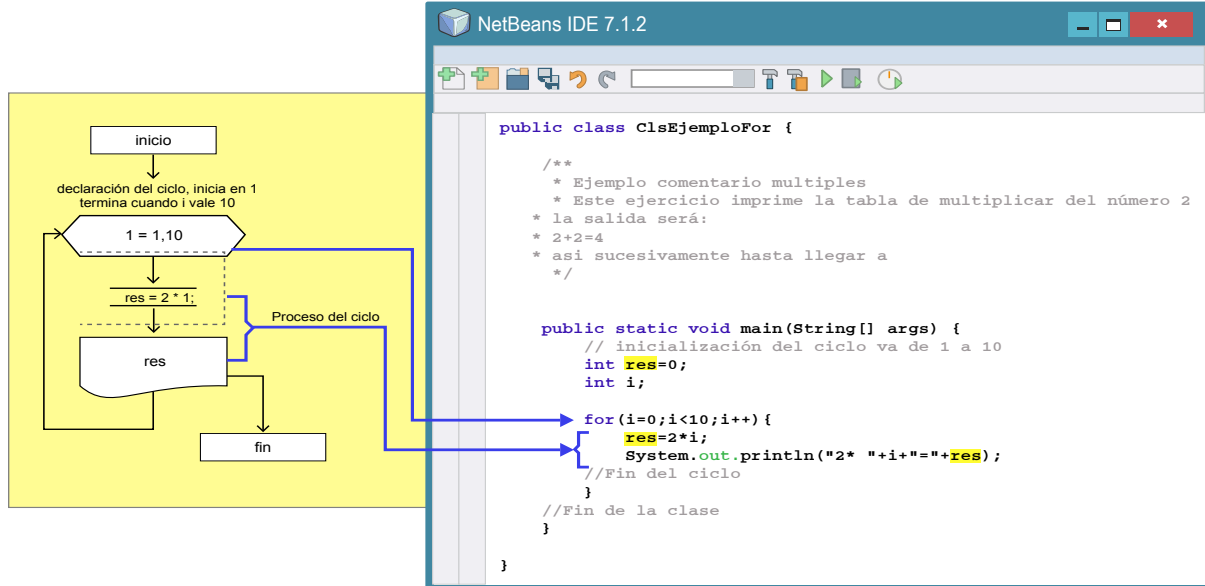
Recordar que, en esta estructura, el número de iteraciones del ciclo depende de la condición que se plantee al comienzo del proceso.



Ejemplo, diagrama de flujo vs JAVA, aplicando condicionales sencillos:

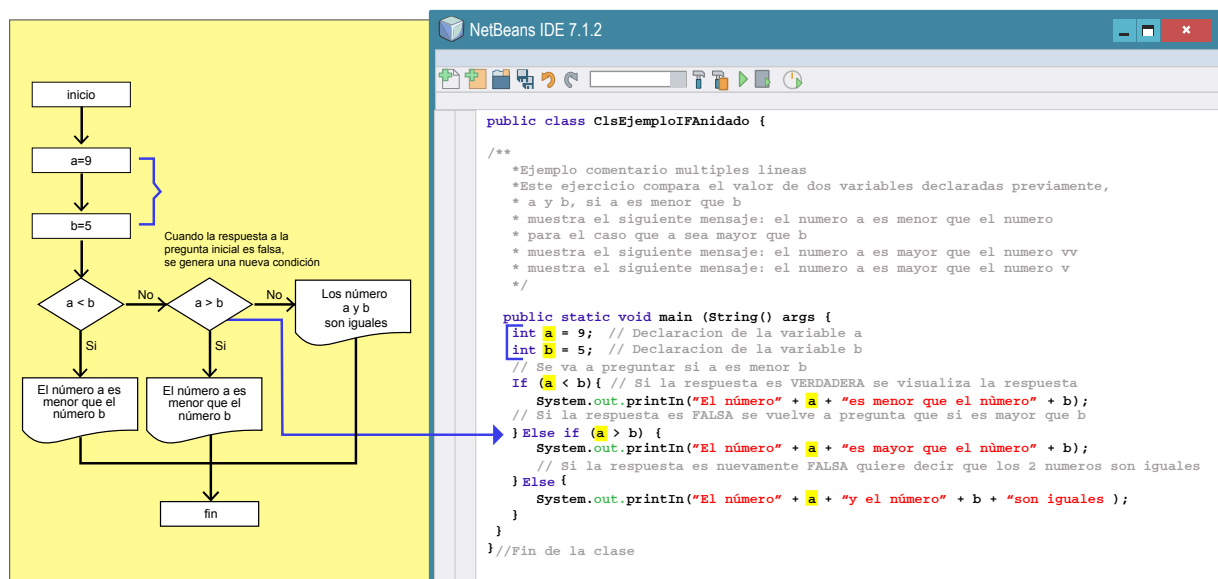


Ejemplo, diagrama de flujo vs JAVA, aplicando condicionales compuestos:



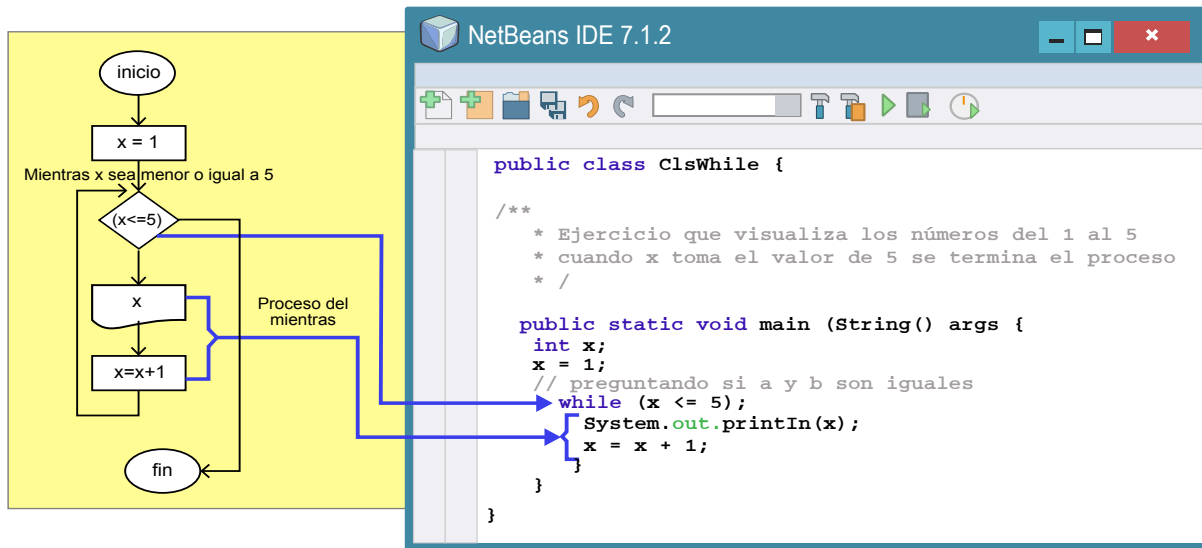
Es muy importante tener claro que cuando se utilizan condicionales compuestos se generan if anidados, por eso notará que por la parte correspondiente al else se maneja una nueva pregunta con la instrucción elseif. La sintaxis tanto de los condicionales como en los condicionales compuestos es la misma, pregunta, respuesta para la parte verdadera y respuesta para la parte falsa.

Ejemplo, diagrama de flujo vs JAVA, estructura cíclica hacer-para:



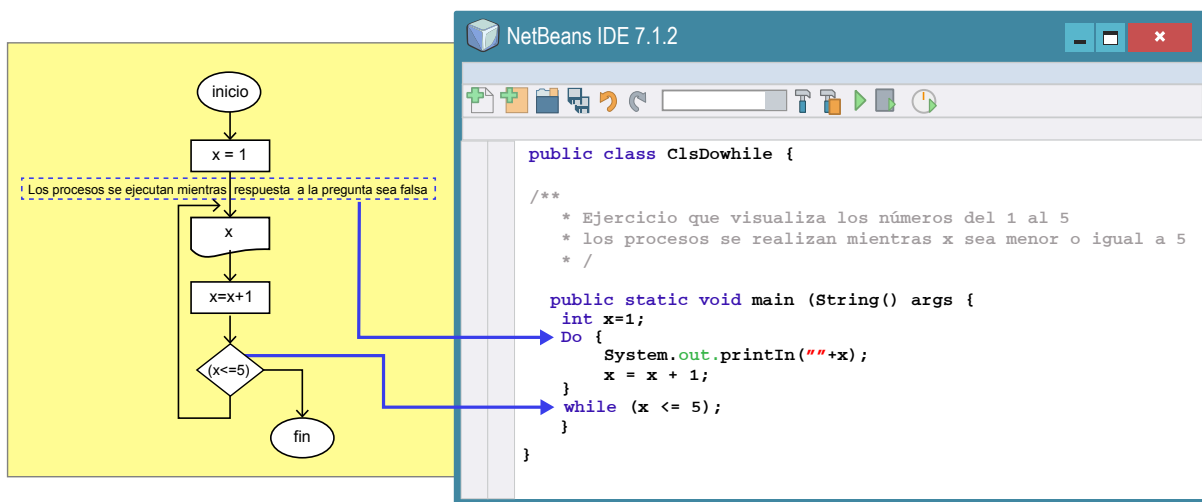
Recordar que en esta estructura el número de iteraciones se conoce antes de ejecutarse el ciclo.

Ejemplo, diagrama de flujo vs JAVA, estructura cíclica hacer-mientras:



Recordar que, en esta estructura, el número de iteraciones del ciclo depende de la condición que se plantee al comienzo del proceso.

Ejemplo, diagrama de flujo vs JAVA, estructura cíclica DO-While:

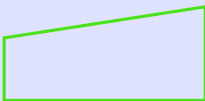
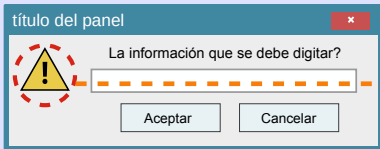
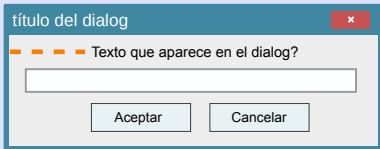
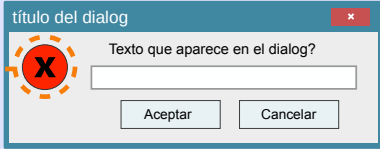
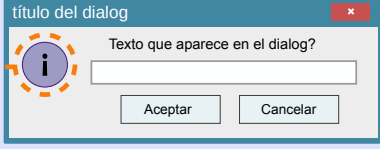
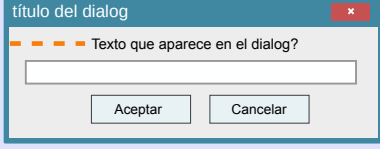


Recordar que, en esta estructura, el número de iteraciones del ciclo depende de la condición que se plantee al final del proceso.

8. Transformación de un algoritmo desarrollado en DFD a una interfaz de desarrollo con lenguaje JAVA.

DFD es una herramienta que nos permite editar e interpretar de manera muy sencilla un diagrama de Flujo, el siguiente paso después trabajar en DFD, es pasar esos diagramas a un lenguaje de programación para este caso es JAVA.

Antes de iniciar con este paso de DFD a JAVA, ver algunas herramientas y términos que facilitaran el trabajo:

DFD	JAVA
<p>Entrada</p> 	<p>a) JOptionPane.showMessageDialog</p> <p>Sintaxis <code>JOptionPane.showMessageDialog(null, "Texto que aparece en el dialog", "titulo del dialog" 2)</code></p>  <p>Los valores para cambiar el ícono podrían variar:</p> <p>Sin ícono (PLAIN_MESSAGE)</p> <p>-1</p>  <p>Error (ERROR_MESSAGE)</p> <p>0</p>  <p>Información (INFORMATION_MESSAGE)</p> <p>1</p>  <p>Cuidado (WARNING_MESSAGE)</p> <p>2</p> 

Pregunta (QUESTION_MESSAGE)



Como normalmente los datos de entrada se convierten en variables o identificadores, y en este caso es necesario realizar conversiones de tipos, esto se conoce técnicamente con casting.

Las conversiones más utilizadas son:

De Texto a entero, de texto a Double, de texto a Byte, de Texto a Float, la sintaxis. Para el caso de las entradas desde un JOptionPane, estas conversiones quedarían de la siguiente manera:

Byte.parseByte (JOptionPane.showInputDialog(null,"texto que aparece en el dialog?","título del dialog",1));

la instrucción que interpreta el sistema "la información que se recibe del dialog conviértala a variable tipo byte.

para el caso del casting de texto a entero la instrucción sería:

Integer.parseInt (JOptionPane.showInputDialog(null, Texto que aparece en el dialog?","título del dialog",1));

Para los otros casos sería:

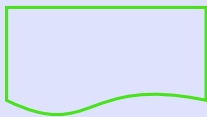
Convertir a double: Double.parseDouble(valor a convertir);

Convertir a short: Short.parseShort(valor a convertir);

Convertir a long: Long.parseLong(valor a convertir);

DFD

Salida



JAVA

a) JOptionPane.showMessageDialog

Sintaxis

JOptionPane.showMessageDialog(null,"mensaje a visualizar", "Título del diálogo", 1);

Los Números para los íconos son iguales a los del dialog de entrada de datos.

- 1 Sin ícono (PLAIN_MESSAGE)
- 0 Error (ERROR_MESSAGE)
- 1 Información (INFORMATION_MESSAGE)
- 2 Cuidado (WARNING_MESSAGE)
- 3 Pregunta (QUESTION_MESSAGE)

b) System.out

Esta instrucción va acompañada de la palabra referenciada print o println, se utiliza para visualizar los resultados por consola y estos resultados pueden verse en una sola línea (print) o con un salto de línea.

La sintaxis para las salidas en una sola línea, sería la siguiente:

```
System.out.print ("Mensaje");
System.out.print ("Mensaje ");
System.out.print ("de una sola ");
System.out.print ("línea");
```

Salida por consola

```
run
mensaje de una sola línea BUID SUCCESSFUL (total time: 5 seconds)
```

Note que aunque se arlizaron 3 líneas de código, el mensaje se visualiza solo en un línea; esto porque el print todo lo maneja en UNA SOLA LÍNEA.

La sintaxis para las salidas en varias sola línea, sería la siguiente:

```
System.out.println("mensaje ");
System.out.println("de una sola ");
System.out.println("línea");
```

Salida por consola

```
run
Mensaje
de una sola
línea
BUID SUCCEFUL (total time: 5 seconds)
```

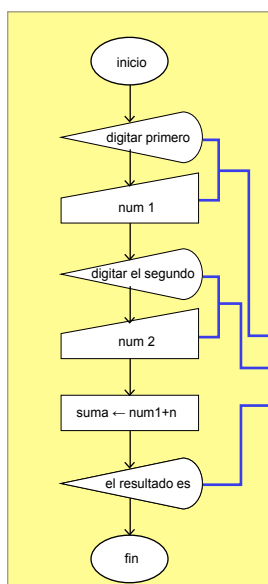
Note que aunque dice mensaje de una sola línea, por el hecho de manejarse la palabra reservada println, la respuesta del sistema fue en varias líneas.

Para el caso de concatenar un texto estático con una variable se utilizaría el operador más(+), la sintaxis sería la siguiente:

```
System.out.println("mensaje " + nombredelavariabale);
```

Ejemplo. DFD vs JAVA, sumando dos números:

DFD



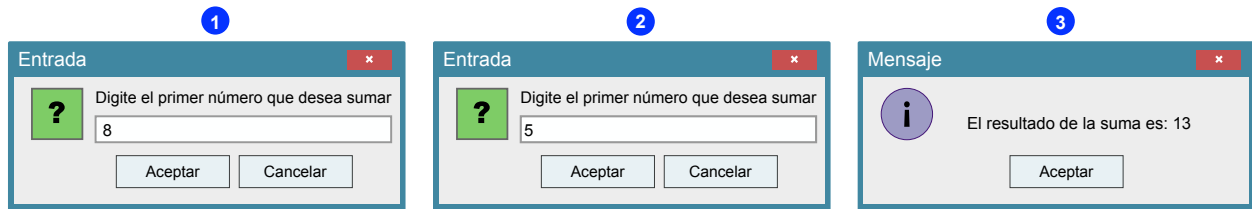
Codificación en JAVA

```

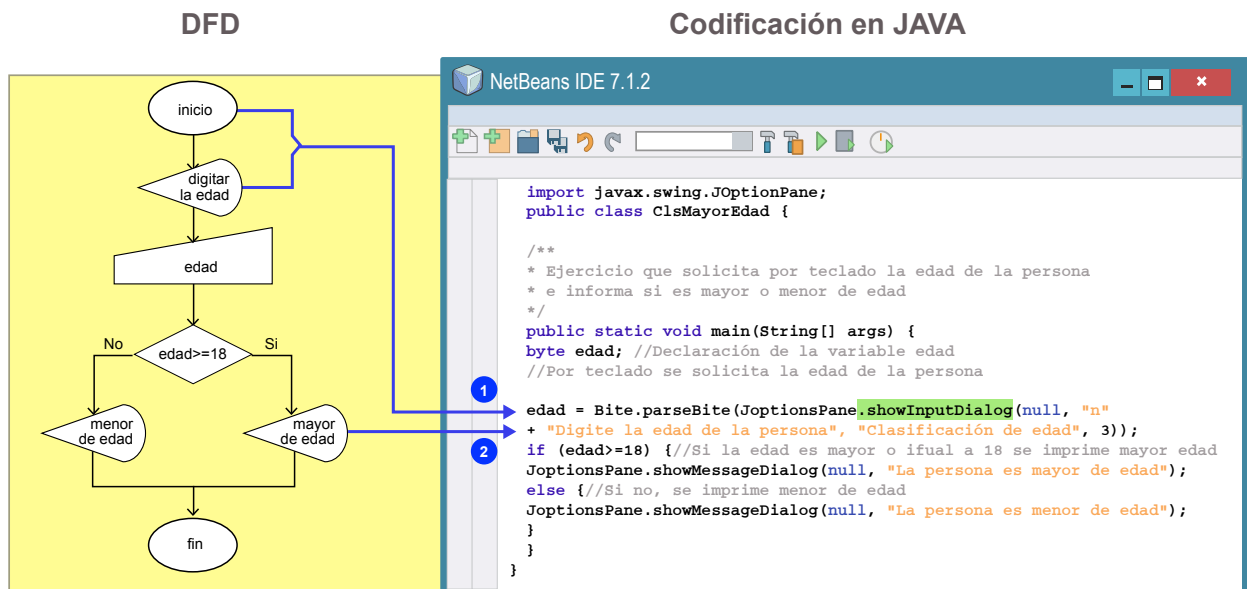
import javax.swing.JOptionPane; //Esta clase se importa para poder visualizar la caja de dialogo
public class Clssumar {

    /**
     * Ejercicio que ssolicita dos datos por teclado
     * los suma y visualiza el resultado de su suma
     */
    public static void main(String[] args) {
        int num1;
        int num2;
        int suma;
        //Se solicita el valor del número uno y se le asigna la variable num1
        num1 = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el primer número que desea sumar"));
        //Se solicita el valor del número dos y se le asigna la variable num2
        num2 = Integer.parseInt(JOptionPane.showInputDialog(null, "Digite el segundo número que desea sumar"));
        //Se realiza la suma de num1+num2 y se le asigna la variable suma
        suma = num1 + num2;
        //En una caja de diálogo se presenta el resultado
        JOptionPane.showInputDialog(null, "El resultado de la suma es: " + suma);
    }
}
  
```

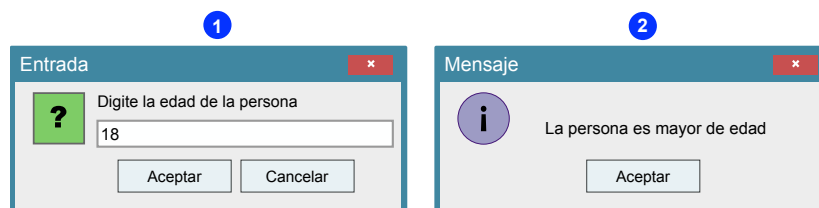
Salidas en el sistema:



Ejemplo. DFD vs JAVA, determinar si una persona es mayor o menor de edad.



Salidas en el sistema:



9. Transformación de un algoritmo desarrollado en LPP a una Interfaz de desarrollo con lenguaje JAVA.

En la secuencia para adquirir sus fundamentos de programación, ha vivido los siguientes momentos, primero trabajó Algoritmos Básicos probablemente en papel, luego pasó a trabajar algoritmos con el uso de la herramienta DFD, después de esto inició con un trabajo

en LPP y ahora con JAVA. Los ejercicios que se presentan a continuación buscan reforzar los fundamentos de JAVA, tomando como referente algunos ejercicios desarrollados en la herramienta LPP.

Código LPP

```
cadena [20] nombre 1
2 inicio
3 escriba "Por favor ingrese su nombre:"
4 lea nombre
5 llamar nueva_linea
6 escriba "Bienvenido a LPP", nombre
fin
```

Codificación en JAVA

```
NetBeans IDE 7.1.2

import javax.swing.JOptionPane;
//Ejercicio que le da la bienvenida a JAVA
public class LppLeernombre{
    public static void main (String [] args {
        //Declaración de la variable nombre de tipo de texto.
        String nombre;
        //Presenta por pantalla una caja de diálogo para que se capture
        //el nombre por parte del usuario
        nombre = JOptionPane.showInputDialog(null "\n"
        + "Por favor ingrese su nombre", "Bienvenida", 3);
        JOptionPane.showInputDialog(null "\n"
        + "Bienvenido a JAVA", "Bienvenida", 1);
    }
}
```

Simil entre las dos herramientas

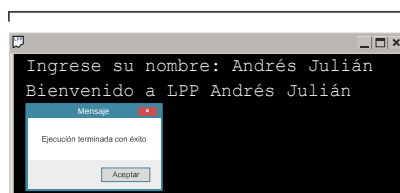
LPP	JAVA
Línea 1	Línea 6
Líneas 3 y 4	Líneas 9 y 10
Líneas 5 y 6	Líneas 11 y 12

Notar que cada una de las líneas de código que implementó en la herramienta LPP se convirtieron a JAVA y se conserva la estructura de un programa.

Declaración de variables, lectura de datos y salidas en el sistema.

Presentaciones en el sistema.

LPP



JAVA



En LPP se desarrolló ejercicios con cálculos matemáticos, en el ejemplo que se presenta a continuación se realiza el simil entre JAVA y LPP para el manejo de operaciones

matemáticas. Tomar como referencia el cálculo de la definitiva de una materia que tiene dos notas. Para dar solución a este enunciado se necesita declarar tres variables, nota1, nota2, y la variable para realizar el promedio. Las dos notas serán variables de entrada y la de promedio será una variable de salida.

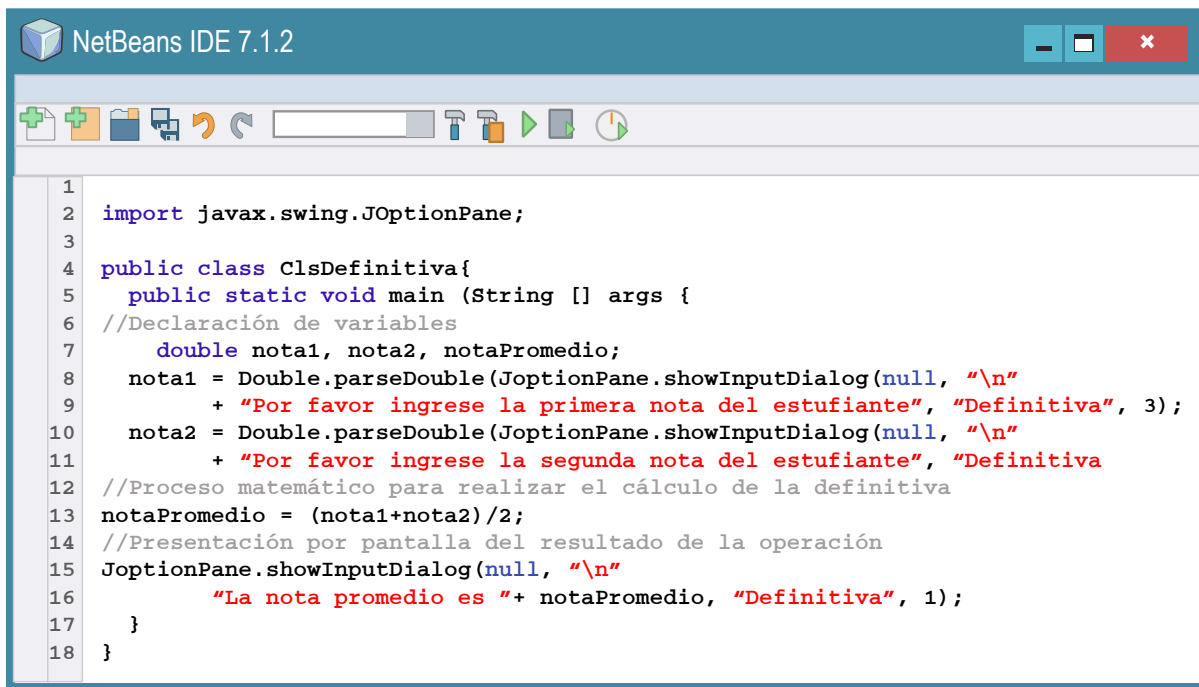
El código en las dos herramientas se presenta a continuación:

Código LPP

```

Real nota1, nota2, notaPromedio 1
inicio
2  escriba "Ingrese la primera nota del
3  estudiante:"
  lea nombre
  escriba "Ingrese la segunda nota del
  estudiante:"
  lea nombre
4  notaPromedio <- (nota1 + nota2) / 2
  escriba "La nota promedio es", Nota Promedio
fin
  
```

Codificación en JAVA



```

1  import javax.swing.JOptionPane;
2
3
4  public class ClsDefinitiva{
5      public static void main (String [] args {
6          //Declaración de variables
7          double nota1, nota2, notaPromedio;
8          nota1 = Double.parseDouble(JOptionPane.showInputDialog(null, "\n"
9              + "Por favor ingrese la primera nota del estudiante", "Definitiva", 3);
10         nota2 = Double.parseDouble(JOptionPane.showInputDialog(null, "\n"
11             + "Por favor ingrese la segunda nota del estudiante", "Definitiva
12         //Proceso matemático para realizar el cálculo de la definitiva
13         notaPromedio = (nota1+nota2)/2;
14         //Presentación por pantalla del resultado de la operación
15         JOptionPane.showInputDialog(null, "\n"
16             + "La nota promedio es " + notaPromedio, "Definitiva", 1);
17     }
18 }
  
```

Explicación Líneas de Código en JAVA	
Línea	Explicación.
8	Esta línea se asemeja a la línea 1 en LPP, se cambia el tipo de dato porque en JAVA no existe el tipo Real, en JAVA el manejo de datos con números que pueden tener decimales se declara con el tipo "double"
9 y 10	Estas líneas corresponden a la entrada de datos(línea 2 en LPP) , para este caso la variable nota1, el dialog, como se trabajó anteriormente a esta ventana se le debe realizar una conversión porque el dato que se va recibir y a procesar es de tipo double, por ese motivo se utiliza la instrucción "Double.parseDouble".
11 y 12	Estas líneas corresponden a la entrada de datos(línea 3 en LPP) , para este caso la variable nota1, el dialog, como se trabajó anteriormente a esta ventana se le debe realizar una conversión porque el dato que se va recibir y a procesar es de tipo double, por ese motivo se utiliza la instrucción "Double.parseDouble".
14	Esta línea se asemeja a la línea 4 en LPP, corresponde al proceso de calcular la notapromedio, notar que en JAVA, el símbolo <- se cambia por el símbolo igual.
19	Esta línea se asemeja a la línea 4 en LPP, corresponde al proceso de calcular la notapromedio, notar que en JAVA, el símbolo <- se cambia por el símbolo igual.

En LPP como en cualquier herramienta de programación, se trabajan estructuras condicionales; se presenta a continuación un ejemplo con condicionales. Notar que la lógica sigue siendo la misma, lo que cambia son algunos detalles en la sintaxis.

Código LPP

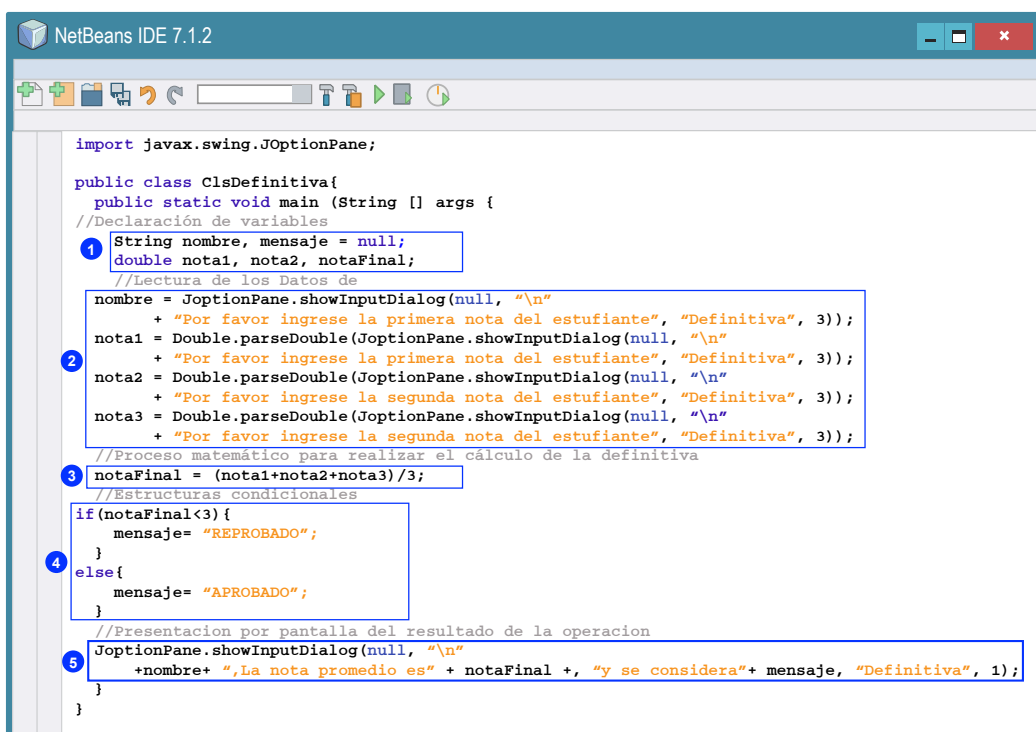
```

//Declaracion de Variables
1 Cadena [25] nombre
Real nota1, nota2, nota3, notaFinal

Inicio
//Lectura de los datos de entrada
escriba "Ingrese el nombre del estudiante:"
lea nombre
escriba "Ingrese el valor de la primera nota:"
2 lea nota1
escriba "Ingrese el valor de la segunda nota:"
lea nota2
escriba "Ingrese el valor de la tercera nota:"
lea nota3
//Cálculo de la nota final
3 notaFinal <- (nota1 + nota2 + nota3) / 3
//Ecritura de la salida
llamar nueva_linea
llamar nueva_linea
escriba " INFORMACIÓN DEL ESTUDIANTE"
4 llamar nueva_linea
escriba "NOMBRE ----->", nombre
llamar nueva_linea
escriba "NOTA FINAL -->", notaFinal
//Estructura Condicional Doble
Si notaFinal < 3 Entonces
escriba "REPROBADO"
5 Sino
escriba "APROBADO"
Fin Si
Fin

```


Codificación en JAVA



```

import javax.swing.JOptionPane;

public class ClsDefinitiva{
    public static void main (String [] args {
        //Declaración de variables
        1 String nombre, mensaje = null;
        double nota1, nota2, notaFinal;

        //Lectura de los Datos de
        nombre = JOptionPane.showInputDialog(null, "\n"
            + "Por favor ingrese la primera nota del estudiante", "Definitiva", 3));
        2 nota1 = Double.parseDouble(JOptionPane.showInputDialog(null, "\n"
            + "Por favor ingrese la primera nota del estudiante", "Definitiva", 3));
        nota2 = Double.parseDouble(JOptionPane.showInputDialog(null, "\n"
            + "Por favor ingrese la segunda nota del estudiante", "Definitiva", 3));
        nota3 = Double.parseDouble(JOptionPane.showInputDialog(null, "\n"
            + "Por favor ingrese la segunda nota del estudiante", "Definitiva", 3));

        //Proceso matemático para realizar el cálculo de la definitiva
        3 notaFinal = (nota1+nota2+nota3)/3;

        //Estructuras condicionales
        4 if(notaFinal<3){
            mensaje= "REPROBADO";
        }
        else{
            mensaje= "APROBADO";
        }

        //Presentacion por pantalla del resultado de la operacion
        5 JOptionPane.showInputDialog(null, "\n"
            +nombre+ ",La nota promedio es" + notaFinal +, "y se considera"+ mensaje, "Definitiva", 1);
    }
}
    
```

Explicación Líneas de Código en JAVA

Fragmento	Explicación.
1	Como en LPP, e cualquier herramienta se deben declarar las variables, para nuestro ejemplo se tomaron variables de tipo String (char en LPP), para manejo de caracteres y double (real en LPP) para el manejo de números con posiciones decimales.
2	Corresponde al proceso de inclusión de datos por teclado por medio de una ventana de dialogo, para esto se utiliza como en la mayoría de los ejercicios que se han desarrollado el JOptionPane.showInputDialog.
3	Se realiza el cálculo matemático, para este caso la notaFinal.
4	Manejo de los condicionales, como se ha trabajado desde el inicio de algoritmia se utiliza la palabra reservada "IF" con su respectiva pregunta y las dos opciones de respuesta para la pate verdadera y para la parte falsa.
5	Finalmente se presenta la salida del sistema, para esto, se utilizó el JOptionPane.showMessageDialog.

GLOSARIO

DFD: Diagrama de Flujo de Datos.

IDE: Entorno Integrado de Desarrollo.

Out: salida.

Print: impresión de los datos sin salto de línea.

Println: impresión de los datos seguido de un salto de línea.

JOptionPane: ventana emergente utilizada para visualizar o incluir datos por pantalla.

showMessage.Dialog(): ventana emergente que solo muestra un aviso al usuario.

showInputDialog(): ventana emergente utilizada para que el usuario incluya datos al sistema.

JDK: código fuente, compilador, Bytecode, API, JRE (Entorno de Ejecución) y Máquina Virtual(JVM).

BIBLIOGRAFÍA

Ceballos, J. (2006). *Java 2: lenguaje y aplicaciones*. RA-MA Editorial.

Prieto, N. (2016). *Empezar a programar usando Java*. (3ª. ed.). España: Editorial de la Universidad Politécnica de Valencia.

Sánchez, J. (2009). *Programación en JAVA*. España: McGraw-Hill.

CONTROL DEL DOCUMENTO

CONSTRUCCIÓN OBJETO DE APRENDIZAJE



INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA

Centro Industrial de Mantenimiento Integral - CIMI
Regional Santander

Líder línea de producción: Santiago Lozada Garcés.

Asesores pedagógicos: Rosa Elvia Quintero Guasca.
Claudia Milena Hernández Naranjo.

Líder expertos temáticos: Rita Rubiela Rincón Badillo.
Expertos temáticos: Magda Liliana García Gamboa (V1).
Edgar Eduardo Vega Arango (V2).

Diseño multimedia: Tirso Fernán Tabares Carreño.

Programador: Francisco José Lizcano Reyes.

Producción de audio: Víctor Hugo Tabares Carreño.

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.



**creative
commons**

BY NC SA

JAVA

© 1996, 1996, Sun Microsystems, Inc. All rights reserved.

Netbeans

Copyright © 1997 - 2009, Sun Microsystems, Inc. All rights reserved.



Registered trademark