UNIVERSIDAD RAFAEL LANDÍVAR

FACULTAD DE INGENIERÍA

COMPILADORES

ING. DHABY XILOJ

MANUAL PROYECTO COMPILADOR

JORGE STEVENS IZÁS ROJAS – 1508717

MARVIN ADOLFO LÓPEZ LÓPEZ - 1545717

QUETZALTENANGO 18 DE NOVIEMBRE DEL 2019

Introducción

Los compiladores son una parte esencial en la programación aunque también es un programa como los que se realizan diariamente estos sirven para traducir programas a código fuente que la computadora puede entender así como identificar los errores que pueden existir en el código fuente. Un compilador se compone de varias partes como analizador léxico que se encarga de reconocer las palabra del código fuente y pasarlas como tokens, el analizador sintáctico que se encarga de ‘entender’ la estructura de las lineas de código y su relación, el analizador semántico que se encarga de comparar e interpretar el código para verificar que la estructura sea la correcta. La mayoría de compiladores se encuentran implementados en el lenguaje de programación sobre el que se esta trabajando en este caso java.

Por otra parte también existe el uso de assembler desde los inicios de la computadora, se menciona este lenguaje ya que el compilador es un traductor de un código fuente a un código de maquina que pueda ser interpretado por assembler para por ultimo ser implementado en un pic 16f84a.

En este manual se describe no solo el uso para el usuario del compilador si no también la estructura del programa asignado como proyecto desde como se estructura el compilador y como funcionan sus partes por separado y unidas para crear el compilador, se describe las reglas utilizadas en cada parte. Contiguo a la explicación técnica se encuentra el manual para el usuario donde se explica como se utiliza la aplicación correctamente para evitar errores a través de imágenes de la interfaz grafica que ha sido creada para un fácil y entendible uso por parte del usuario final y que también puede investigar más sobre el uso y funcionamiento de los compiladores mediante el manual técnico.

Manual del Compilador

A continuación se explica la creación y las partes que componen al compilador como sus reglas, partes, clases, etc. Así como los componentes utilizados en la aplicación.

SO y Complementos Utilizados

Sistemas operativos sobre los que se desarrollo la aplicación:

- Debian buster 10
- Arch linux

Complementos utilizados para el desarrollo de la aplicación:

- NetBeans IDE 8.2
- OpenJDK
- Libreria jflex
- Librería jcup
- JavaFX Scene Builder 2.

Analizador Léxico

Para la creación del analizador léxico se utilizo la librería jflex, en esta parte se crearon las reglas sobre las cuales se crean los token del archivo previamente cargado a la aplicación. Cualquier identificador no creado en las reglas del analizador léxico se tomaran como errores o no aceptados. En esta parte se guardan los tokens en una tabla de valores sobre la que se explicara su creación y funcionamiento más adelante.

En la siguiente tabla se especifican los token que se definieron en el analizador léxico.

Definición	Token	Expresión Regular
Numero	numero	$[0-9][0-9]^*$
Palabra	letra	$[a-zA-Z]([a-zA-Z]\|[0-9])^*$
Comentario	comentario	"/*" [^*] {palabra} ~"*/" | "/*" "*"+ "/"
Fin de linea	finLinea	\r | \n | \r\n
Espacio en blanco	WhiteSpace	{finLinea} | [\t\f]
Clase	clase	“clase”
propiedades	propiedades	“propiedades”
Metodo	metodo	“metodos”
Propiedades Publicas	publica	“publicas”
Metodos Publicos	publico	“publicos”
Propiedades Privadas	privada	“privadas”
Propiedades Protegidas	protegida	“protegidas”
String	string	“cadena”
Int	int	“entero”

Real	real	“real”
Boolean	boolean	“boleano”
NULL	null	“nulo”
Parentesis Izquierdo	parentL	“(”
Parentesis Derecho	parentR	“)”
Llave Izquierda	llaveL	“{”
Llave Derecha	llaveR	“}”
Corchete Izquierdo	corchL	“[”
Corchete Derecho	corchR	“]”
Igual	igual	“=”
Punto	punto	“.”
Comillas	comilla	‘ “ ‘
Coma	coma	“ , “
Mayor que	mayq	“>”
Menor que	menq	“<”
Igual Comparación	equal	“==”

Diferente de	diferente	“!=”
Suma	suma	“+”
Resta	resta	“-”
Multiplicación	multi	“*”
División	divi	“/”
Modulo	modulo	‘”%”
Exponencial	exp	“^”
OR	or	“OR”
AND	and	“AND”
if	if	“si“
else	else	“entonces“
else if	elif	“sino”
return	return	“devolver”
from	from	“desde”
while	while	“mientras”
incrementar	inc	“incrementar”

decrementar	dec	“decrementar”
do	do	“hacer”
Imprimir	print	“escribir”
Recibir datos	get	“leer”
Instanciar	instanciar	“instanciar”

En la tabla anterior se definen los tokens y palabras reservadas que pueden ser ingresadas al analizador lexico por ejemplo el signo ‘ - ‘ se toma como el token ‘resta’, mientras que ‘escribir’ se toma como una palabra reservada. Al momento de abrir el archivo y presionar el boton compilar el documento se pasa por el analizador léxico para crear una tabla de valores donde se encuentran todos los token aceptados que posteriormente se utilizaran en el analizador sintáctico para crear la estructura del compilador. En este caso se toman también los tabuladores o espacios que se encuentren ya que son los que ayudan al analizador sintáctico a definir si la estructura de un método por ejemplo es correcta y corresponde con los espacios que deben existir antes del método por ejemplo <else, else, 3> donde 3 es la cantidad de tabulaciones antes del token. Todos los token aceptados son almacenados en un archivo donde se escribe el tipo, el token, y el numero de fila donde se encuentra. Los tokens que se encuentran serán enviados dependiendo como estén definidos como estados finales en el analizador sintáctico en este caso el .cup.

```
"=" {return new Symbol(sym.IGUAL,
        new token(yycolumn, yyline, yytext()));}
```

Los token como letra pueden iniciar con una letra y pueden ser seguidas de más numero o letras como declarar una variable, pero no se acepta que inicie con numero o que agregue símbolos a la palabra. Mientras que los comentarios se declaran como los vistos en clase, /* *AQUÍ VA EL COMENTARIO* */. La única regla de los comentario es que inicien y terminen con ‘ /* ‘ pero no son mandados al analizador sintáctico ya que no serán utilizados.

En el caso de los espacios se pueden encontrar espacios como lineas completas en blanco que no serán contadas, existen los fines de linea y los tabuladores que como se explica anteriormente son parte de los tokens para definir la posición en la que se encuentra.

Analizador Sintáctico

Para el analizador sintáctico se utilizo la librería o herramienta jcup con la que se crearon las producciones del lenguaje, las producciones son las que definen si la estructura del código con los tokens definidos anteriormente en el analizador léxico es correcta o la estructura no coincide con lo que se esta definiendo en el archivo. Las producciones que se realizan se van a clasificar por bloques como por ejemplo ‘BLOQUE_CLASE’ donde se definen todas las producciones de la clase empezando por la estructura de la clase como ‘INICIO_CLASE’, ‘CUERPO_CLASE’, ‘FIN_CLASE’. De esta forma se tiene un mejor control de las producciones y al mismo tiempo es más fácil eliminar las ambigüedades al momento de crear la producciones.

```
expr ::=  DECIMAL:d {: RESULT=new Nodo(d.getEntero()); :}
        | ID:id {: RESULT=new Nodo(Nodo.TIPO_IDENTIFICADOR, id.getCadena()); :}
        |expr:l SUMA expr:r {:
            Nodo raiz = new Nodo(Nodo.TIPO_OPERADOR, Nodo.OP_SUMA);
            raiz.agregarHijo(l);
            raiz.agregarHijo(r);
            RESULT=raiz;
        :}
```

En esta parte también se definen los estados terminales donde se encuentran los tokens definidos en el analizador léxico pero usados para las producciones, así como los estados no terminales que se refiere al inicio de las producciones como en la captura anterior se puede observar que ‘DECIMAL’ es un estado terminal mientras que ‘expr’ es un estado no terminal.

Analizador Semántico

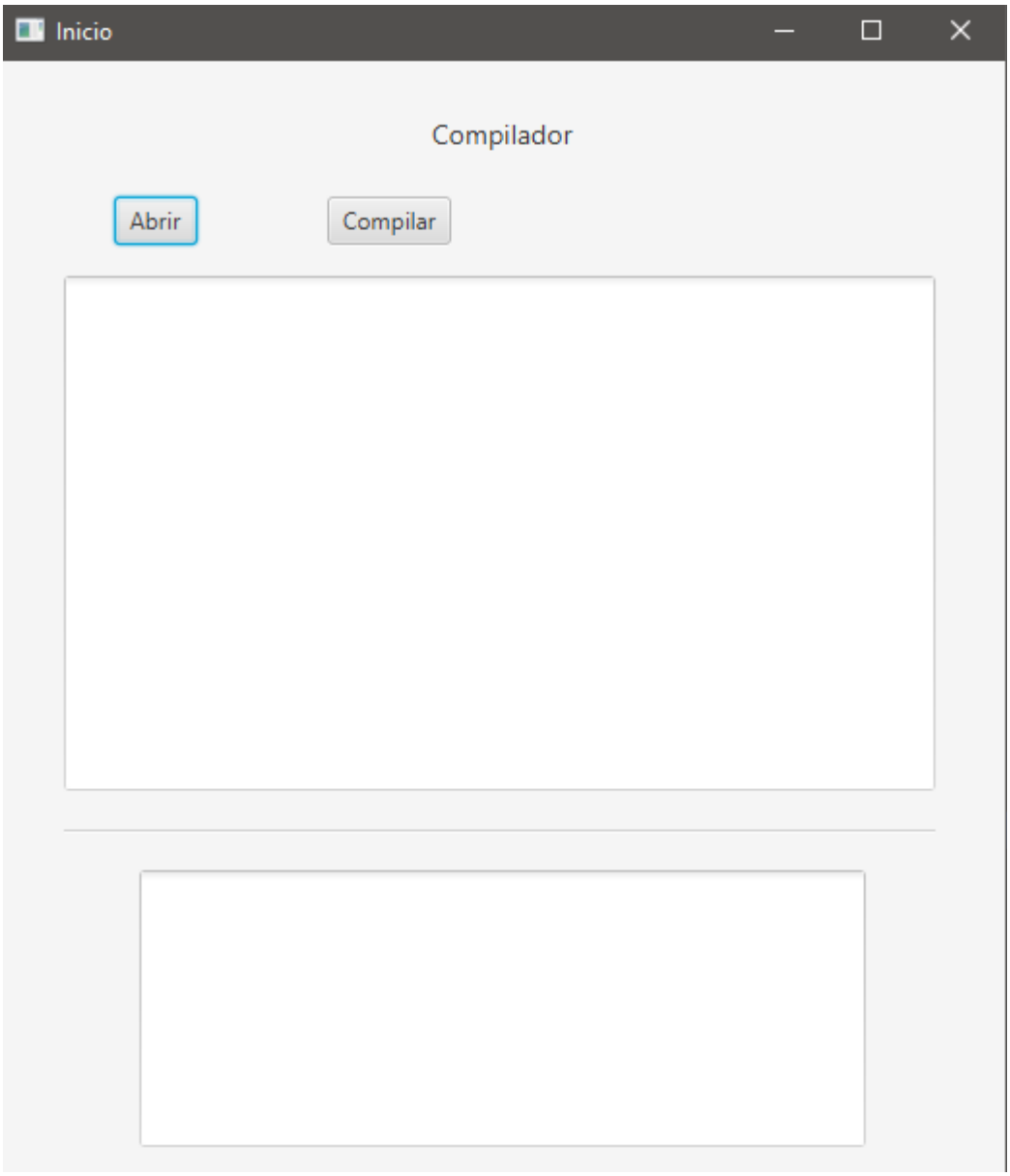
Cuando ya se tienen todas las partes del analizador léxico y el analizador sintáctico se pasa a la parte más difícil del compilador, el analizador semántico en el cual se verifica si los datos ingresados corresponden con la estructura del código, por ejemplo que una variable de tipo ‘int’ solo este formada por números. Por eso es de gran importancia dejar las producciones sin ambigüedades para que los estados a los que se entre sean aceptados y sea más fácil desarrollar el analizador semántico. En este parte es de vital importancia las operaciones como los ‘if’, ‘else’ , el uso de signos como mayor o menor y también la declaración de ciclos ya que todos estos procesos requieren que no solo sean verificados por el analizador sintáctico sino también por el analizador semántico que define si se encuentra correcta la estructura creada anteriormente.

Las clases y los métodos deben ser verificados también en el analizador semántico ya que se debe corroborar que estos cuenten con la estructura correcta como que exista un end, que los parámetros recibidos sean los correctos, que la asignación de datos también sea la correcta, que no se quede la estructura a medias y se empiece una nueva.

Manual de Usuario

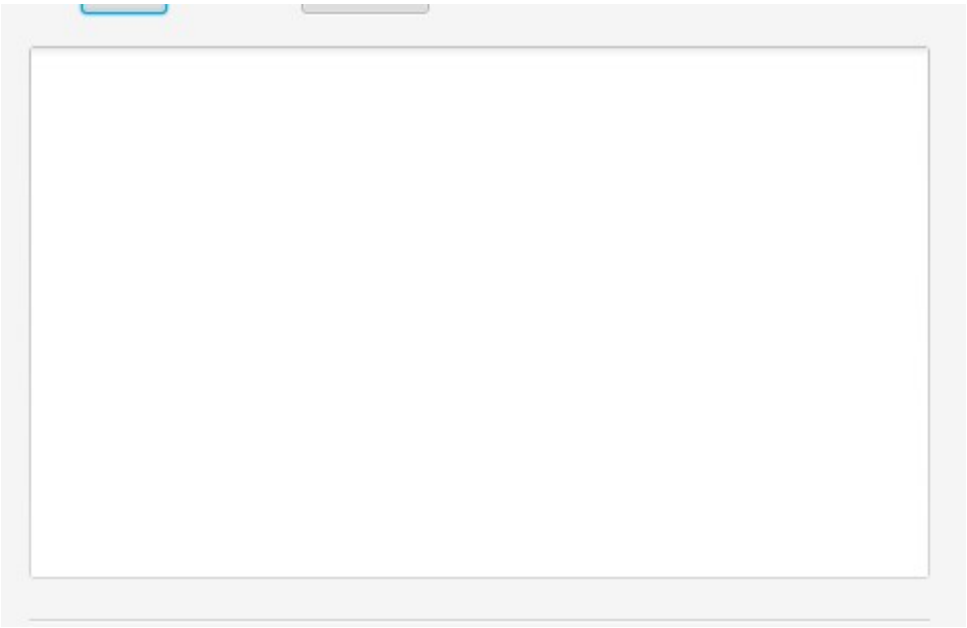
A continuación se demuestra el uso de la aplicación creada en java para un compilador para un uso correcto y que no existan errores al momento del uso.

- Pantalla principal de la aplicación.

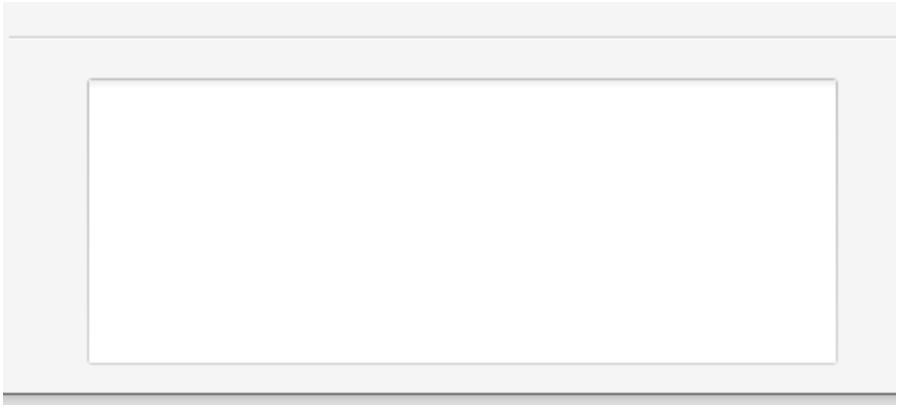


- En la pantalla principal se encuentra el botón ‘abrir’ ubicado en la parte superior izquierda de la aplicación en el cual se podrán abrir los archivos para después ser visualizados en el compilador. Al momento de presionar el botón manda el archivo directamente al analizador léxico

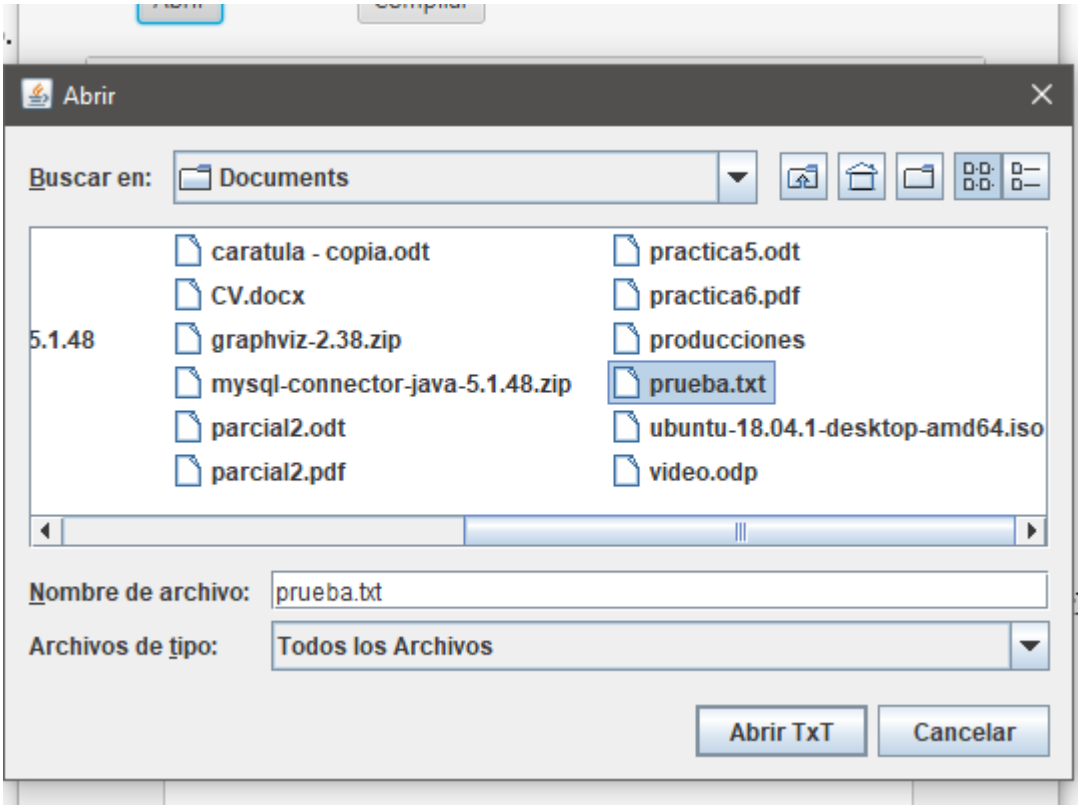
- Se encuentra la pantalla donde se visualizara el código abierto anteriormente, los datos que se encuentran en la pantalla no pueden ser modificados.



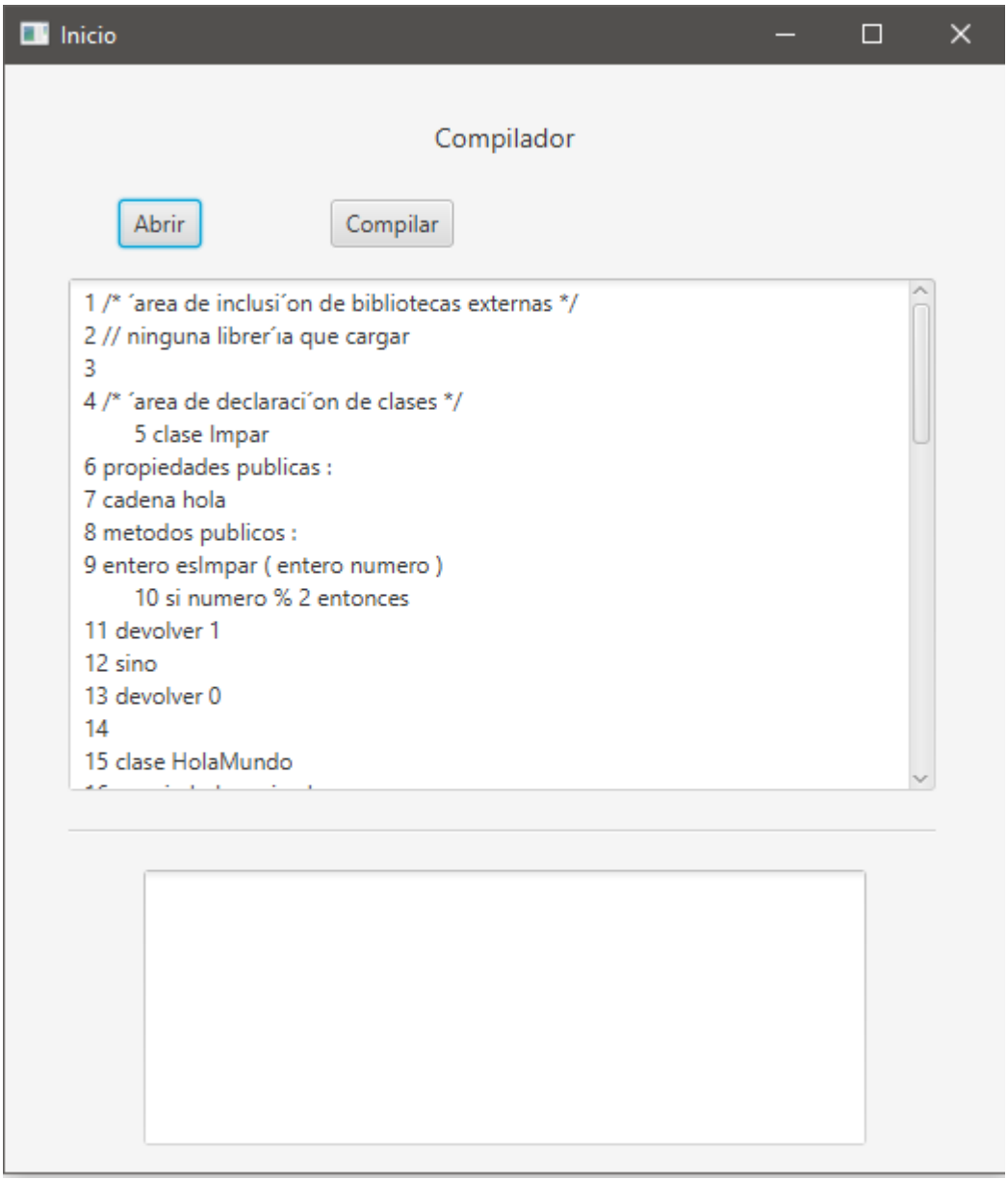
- Por ultimo se encuentra la pantalla donde se muestran las advertencias o los errores que puedan existir en el código.



- Para abrir los documentos se presiona el botón ‘abrir’ en cual se debe seleccionar el archivo, el programa solo aceptara archivos .txt.



- Después de presionar el botón ‘Abrir TxT’ se podrá visualizar el contenido del documento en la primera pantalla, si es más grande que la pantalla se puede bajar con la barra que se encuentra en la parte derecha.



- Para analizar el documento se debe presionar el botón de ‘compilar’ con lo que se llevara a cabo el proceso del compilador.

- Si se quiere abrir un nuevo archivo solo se deben repetir los mismos pasos ya que al abrir un nuevo archivo el programa automáticamente borra el contenido del anterior.

Manual Para la Creación del Lenguaje

Esta es la parte final del manual donde se indica al usuario la creación de un código en el lenguaje de programación definido en este caso ‘LOOP’ para que se tenga conocimiento de los datos con los que se pueden trabajar.

Ejemplo:

```
/* ´area de inclusi´on de bibliotecas externas */
// ninguna librer´ıa que cargar

/* ´area de declaraci´on de clases */
clase Impar
  propiedades publicas :
        cadena hola
  metodos publicos :
        entero esImpar ( entero numero )
          si numero % 2 entonces
                devolver 1
          sino
                devolver 0

clase HolaMundo
  propiedades privadas:
        entero a
        cadena nombre
        Impar i
  metodos publicos :
         constructor ()
                i = nuevo Impar ()
                a = 0
                nombre = ""

        entero inicio ()
```

```
27                escribir " escriba su nombre "
28                // se guarda una cadena de texto en la variable nombre
29                leer nombre ;
30                escribir " escriba un valor entre 1 y 30"
31                // se guarda un numero en la variable a
32                leer a
33                /* escribe en pantalla la constante junto
34                * al valor de la variable nombre
35                */
36                escribir " Hola ", nombre
37                si i. esImpar (a) entonces
38                        escribir "El numero que eligio es impar "
39                sino
40                        escribir "El numero que eligio es par "
41                devolver 0
42
43 /* ´area de declaraci´on de funci´on de carga inicial */
44 entero Principal ()
45        HolaMundo hola = instanciar HolaMundo ()
46        hola . inicio ()
47        devolver 0
```

El programa se puede dividir en clase, método, linea de código.

Clases

- Creación de una clase:

clase ‘NOMBRE DE LA CLASE’

Donde clase es una palabra reservada seguida del nombre que se la va a asignar a la clase.

- Definición de propiedades

propiedades publicas :

Donde propiedades es una palabra reservada y publicas es el tipo de seguridad que se le asigna a las propiedades que también es una palabra reservada, seguido de dos puntos.

- Declaración de propiedades

cadena ‘NOMBRE DE LA PROPIEDAD’

Donde cadena es una palabra reservada para definir el tipo de la propiedad seguido del nombre de la propiedad.

Métodos

- Creación de un método

metodos publicos :

Donde metodos es una palabra reservada que se debe poner antes de escribir cualquier método y publicos es otra palabra reservada que se usa para asignar el nivel de seguridad de los métodos.

- Declaración de un método

entero ‘NOMBRE DEL MÉTODO’ (entero ‘NOMBRE DEL PARÁMETRO’)

Donde entero es el tipo del que va a ser el método, seguido de ‘NOMBRE DEL MÉTODO’ donde se asigna el nombre del método. Se puede crear método sin parámetros pero en caso de que si existan dentro de los paréntesis se coloca primero en tipo del parámetro en este caso entero y seguido del nombre ‘NOMBRE DEL PARÁMETRO’.

- Declaración de variables

‘NOMBRE DE LA VARIABLE’ = “VALOR ASIGNADO”

Donde se define el nombre de la variable seguido del identificador ‘ = ‘ y después el valor que se le asigna a la variable dependiendo su tipo.

Linea de Código

- Creación de un IF

```
si ‘OPERACIÓN’ entonces
        devolver ‘VALOR’
 sino
        devolver ‘VALOR’
```

Donde si es una palabra reservada que debe ser seguida por una operación y por ultimo la palabra reservada entonces, dentro de esto se declara lo que se va a hacer si se cumple la condición, en este caso se utiliza devolver seguido del dato que se va a regresar. Se puede volver a repetir la condición pero con un sino y al igual que anteriormente se manda un devolver seguido del valor que se va a regresar.

- Imprimir

escribir " escriba su nombre "

Donde escribir se utiliza para imprimir el dato mientras que " escriba su nombre " es el dato o valor que se quiere imprimir, si es una palabra o un dato se agrega entre comillas.

- Obtener un dato

leer nombre ;

Donde leer se utiliza para obtener el dato que se quiere leer en este caso 'nombre' solo puede obtener un dato por cada leer.

Tabla de datos

Nombre	Definición
Numero	Se utiliza para definir los números.
Palabra	Se utiliza para definir palabras o variables ya que puede ser una combinación entre números y letras pero siempre empezando por una letra.
Comentario	Se utiliza para definir los comentarios dentro del código ya sea /* */ o también //

Fin de linea	Se utiliza para identificar los fines de linea y saltos de carro
Espacio en blanco	Se utiliza para definir lineas en blanco que pueden existir en código, no se toman en cuenta para no causar problemas con los tokens
clase	Se utiliza para definir el inicio de una clase no se debe confundir con el nombre, la estructura es: **clase 'NOMBRE DE LA CLASE'**
propiedades	Se utiliza para definir el bloque de propiedades de una clase, por lo general se escribe la palabra y seguido el nivel de seguridad por ejemplo: propiedades privadas:
metodo	Se utiliza para definir el bloque de métodos en una clase, no debe confundirse con la creación de un método, la estructura es: **metodo privado:**
publicas	Se utiliza para definir la seguridad y acceso de las propiedades de una clase su estructura es : **propiedades publicas:**
públicos	Se utiliza para definir la seguridad y acceso de las propiedades de una clase su estructura es : **propiedades publicas:**
privadas	Se utiliza para definir la seguridad y acceso de las propiedades de una clase su estructura es : **propiedades privadas:**
protegidas	Se utiliza para definir la seguridad y acceso de las propiedades de una clase su estructura es : **propiedades protegidas:**
string	Palabra reservada que se utiliza para declarar una variable de tipo String: **string nombre = null**

int	Palabra reservada que se utiliza para declarar una variable de tipo entero: **int edad = 19**
real	Palabra reservada que se utiliza para declarar una variable de tipo real: **real medida = 1.89**
boolean	Palabra reservada que se utiliza para declarar variables de tipo boolean: **boolean cont = False;**
NULL	Palabra reservada utilizada para inicializar datos a un valor nulo: **string carrera = null**
(	Símbolo que se utiliza para clases, métodos, instanciar, encerrar datos, etc: **(int valor)**
)	Símbolo que se utiliza para clases, métodos, instanciar, encerrar datos, etc: **(int valor)**
{	Símbolo que se utiliza para clases: **clase persona {** **}**
}	Símbolo que se utiliza para clases: **clase persona {** **}**
[	Símbolo que se utiliza para datos, arrays, variables, etc: **int[8] cant**
]	Símbolo que se utiliza para datos, arrays, variables, etc: **int[8] cant**
=	Símbolo que se utiliza para inicializar variables: **edad = 5**

.	Símbolo que se utiliza para abrir métodos declarados anteriormente, también para la creación de números reales.
“ “	Símbolo que se utiliza para encerrar oraciones para imprimir. **Escribir “hola mundo”**
,	Símbolo que se utiliza para separar parámetros o variables: **edad(int num1, int num2)**
<	Símbolo que se utiliza para comparar datos en un operación if o en un ciclo: **if val1 < val2 entonces**
>	Símbolo que se utiliza para comparar datos en un operación if o en un ciclo: **if val1 > val2 entonces**
==	Símbolo que se utiliza para comparar datos en un operación if o en un ciclo: **if val1 == val2 entonces**
!=	Símbolo que se utiliza para comparar datos en un operación if o en un ciclo: **if val1 != val2 entonces**
+	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1 + dato2**
-	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1 - dato2**
*	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1 * dato2**
/	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1 / dato2**

%	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1 % dato2**
exp	Símbolo que se utiliza para realizar operaciones matemáticas: **dato1exp 8**
OR	Palabra reservada que se utiliza para operaciones lógicas.
AND	Palabra reservada que se utiliza para operaciones lógicas.
si	Palabra reservada que se utiliza para la creación de un if
entonces	Palabra reservada que se utiliza para la creación de un if
sino	Palabra reservada que se utiliza para la creación de un if
regresar	Palabra reservada que se utiliza para la creación de un if o si se cumple una condición se regresa el un valor.
desde	Palabra reservada que se utiliza para la creación de ciclos.
mientras	Palabra reservada que se utiliza para la creación de ciclos.
incrementar	Palabra reservada que se utiliza para incrementar el valor de una variable: **inc edad**
decrementar	Palabra reservada que se utiliza para decrementar el valor de una variable **dec edad**
do	Palabra reservada que se utiliza para la creación de ciclos.
escribir	Palabra reservada que se utiliza para imprimir datos_ **escribir “Hola Mundo”**

leer	Palabra reservada que se utiliza para recibir datos enviados por el usuario: **leer dato1**
Instanciar	Palabra reservada que se utiliza para instaciar métodos: