

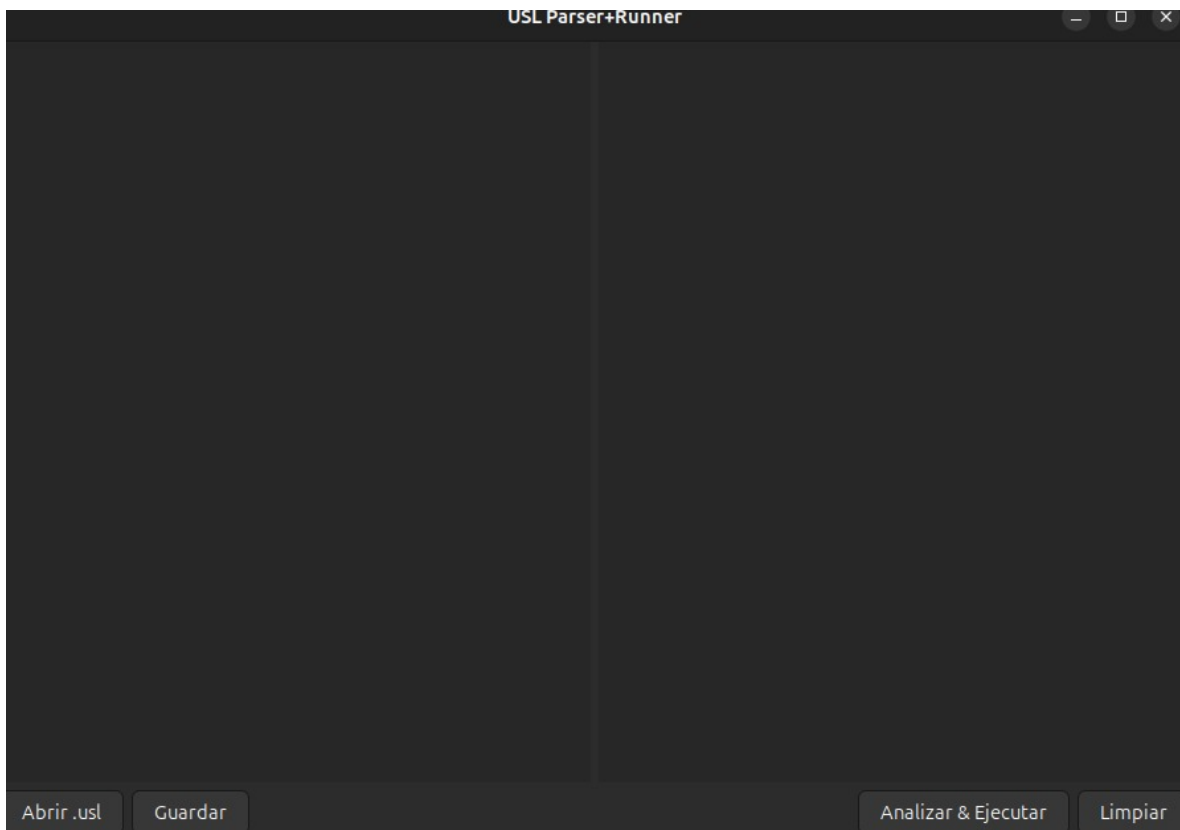
- **Manual de Usuario JavaLang**

Se debe abrir el programa Gui.

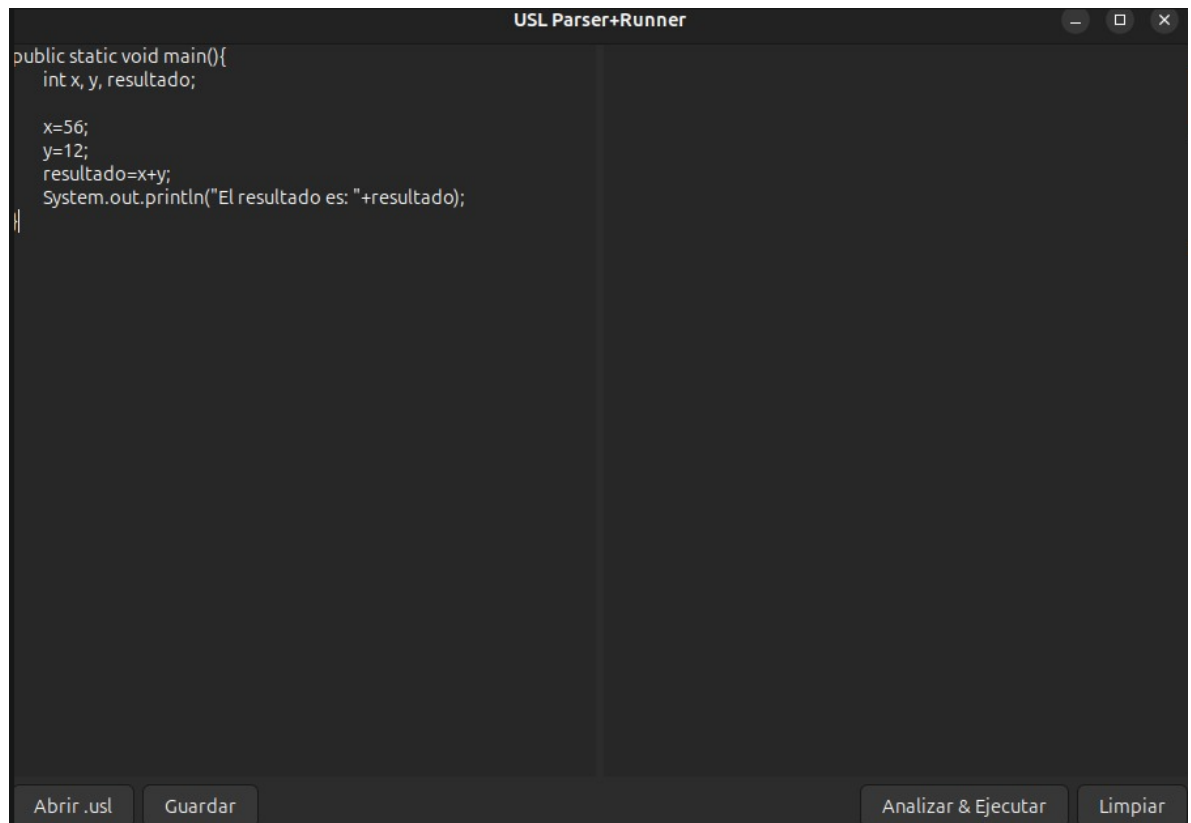


- **Interfaz gráfica Principal**

La cual permite editor código en su edito de lado izquierdo y guardarlo, asi mismo permite abrir código en archivos con extensión.usl



- Si escribiremos un código lo haremos en el panel de lado izquierdo.



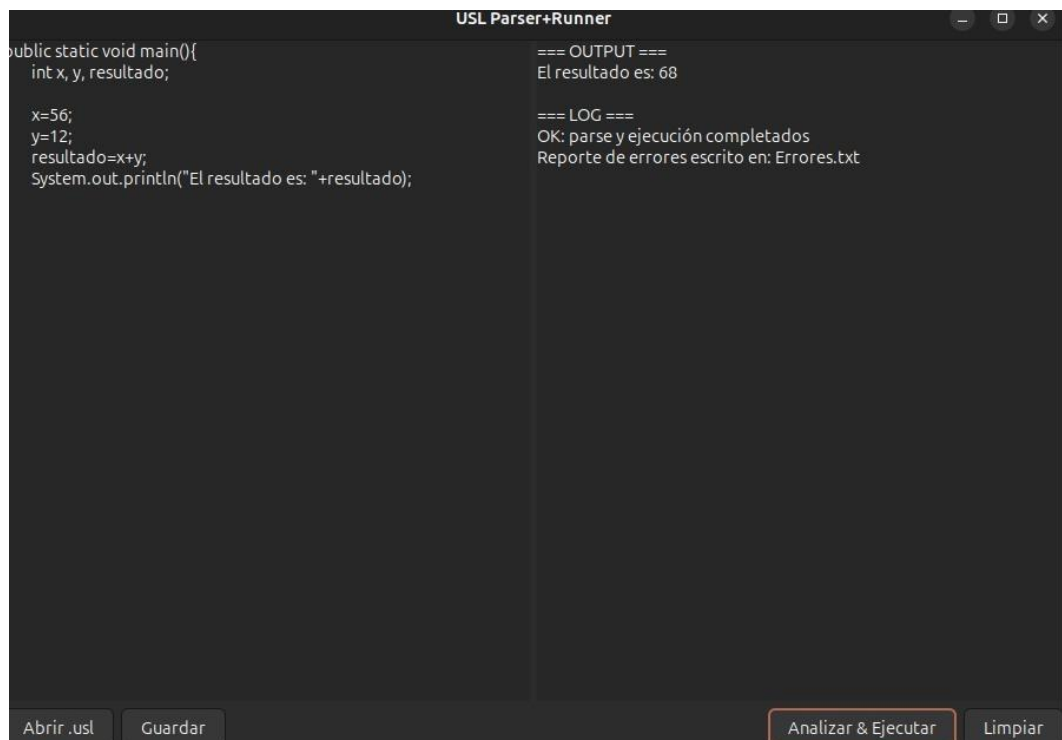
The screenshot shows the 'USL Parser+Runner' application window. The left panel contains the following code:

```
public static void main(){
    int x, y, resultado;

    x=56;
    y=12;
    resultado=x+y;
    System.out.println("El resultado es: "+resultado);
}
```

The right panel is empty. At the bottom, there are four buttons: 'Abrir .usl', 'Guardar', 'Analizar & Ejecutar', and 'Limpiar'.

- Para poder realizar el análisis y la ejecución del programa presionamos el botón analizar y ejecutar.



The screenshot shows the 'USL Parser+Runner' application window after execution. The left panel contains the same code as before. The right panel now displays the output and log:

```
=== OUTPUT ===
El resultado es: 68

=== LOG ===
OK: parse y ejecución completados
Reporte de errores escrito en: Errores.txt
```

At the bottom, the 'Analizar & Ejecutar' button is highlighted with a red border.

- Si ni hay errores en el código se ejecutarán las acciones, mostrara las salidas y se generara dos archivos txt el cual es el árbol ast y tabla de símbolos.



- Archivo de AST.txt

```
Árbol de Sintaxis Abstracta (AST)
=====

- Función main : void
  - Cuerpo
    - Block
      - Seq
        - Decl x : int
        - Decl y : int
        - Decl resultado : int
      - Assign x =
        - Int(56)
      - Assign y =
        - Int(12)
      - Assign resultado =
        - BinOp(+)
          - Id(x)
          - Id(y)
      - println
        - BinOp(+)
          - String("El resultado es: ")
          - Id(resultado)
```

- Archivo de tabla de símbolos

| ID | Tipo símbolo | Tipo dato | Ámbito | Línea | Columna |
|-----------|--------------|-----------|--------|-------|---------|
| main | Función void | Global | 1 | 20 | |
| x | Variable | int | main | 2 | 6 |
| y | Variable | int | main | 2 | 9 |
| resultado | Variable | int | main | 2 | 12 |

- Si existieran errores en el código se genera un archivo txt de errores y la GUI mostrara en la salida los errores.

The screenshot shows the 'USL Parser+Runner' application window. The left pane contains the following Java code:

```
public static void main(){
    int x, y, resultado;

    x=56;
    y=12;
    resultado=x+y;
    System.out.println("El resultado es: "+resultado);
    multiplicacion=x*y;
}
```

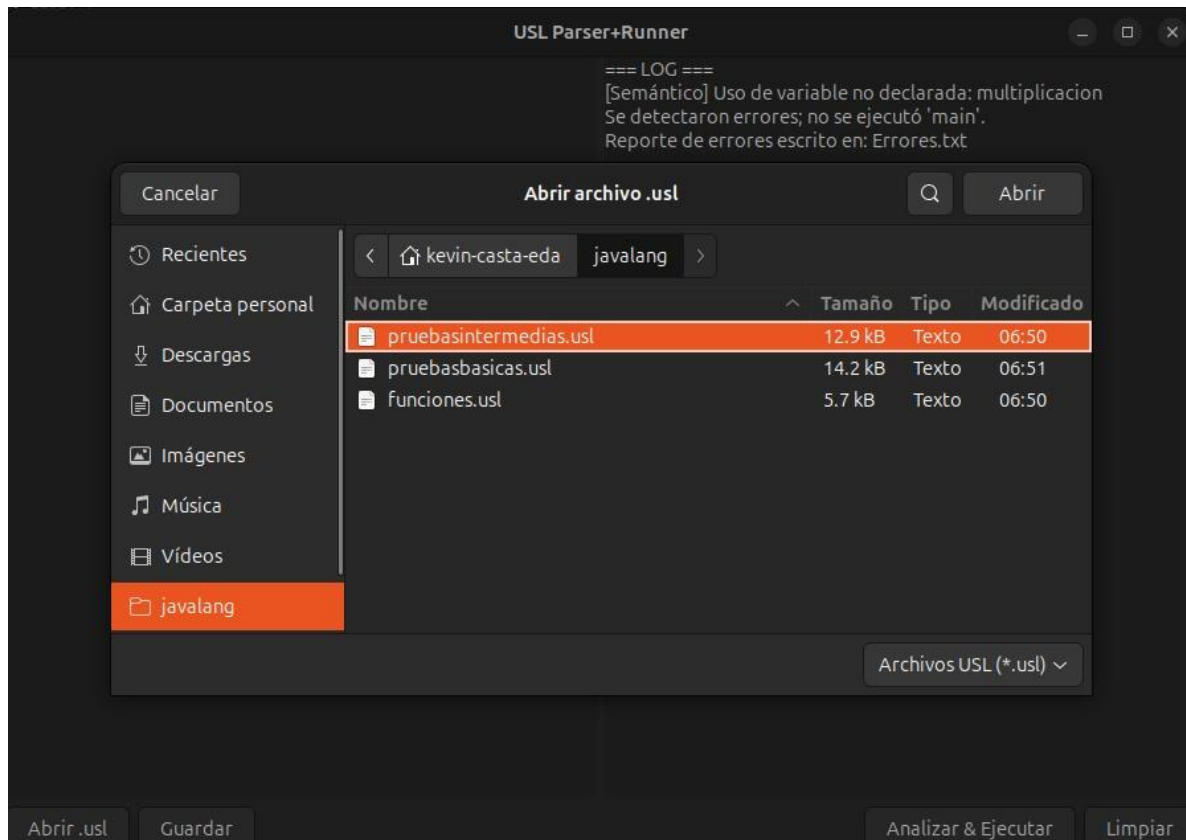
The right pane shows the output log:

```
=== LOG ===
[Semántico] Uso de variable no declarada: multiplicacion
Se detectaron errores; no se ejecutó 'main'.
Reporte de errores escrito en: Errores.txt
```

At the bottom of the window, there are four buttons: 'Abrir .usl', 'Guardar', 'Analizar & Ejecutar', and 'Limpiar'.

| No. | Descripción | Ámbito | Línea | Columna |
|-----|---|-------------|-------|---------|
| 1 | [Semántico] Uso de variable no declarada: <u>multiplicacion</u> | <u>main</u> | 8 | 20 |

- Para abrir un archivo ya existente con extensión.usl presionamos el botón de abrir .usl y seleccionar el archivo que deseamos abrir.



- Al abrir el archivo se muestra el contenido, y podemos realizar la ejecución y análisis de dicho archivo así como guardarlo.

The screenshot shows a window titled "USL Parser+Runner — funciones.usl". The window is split into two panes. The left pane contains the source code of the USL file, and the right pane shows the output of the parser and runner.

```
// ===== Funciones del enunciado =====
// 1.3.1 / 1.3.2
public static void saludar() {
    System.out.println("¡Hola, mundo!");
}
public static int obtenerNumero() { return 42; }
public static void saludarPersona(String nombre) {
    System.out.println("¡Hola, " + nombre + "!");
}
public static int sumar(int a, int b) { return a + b; }

// 1.3.3 Recursivas
public static int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}
public static int fibonacci(int n) {
    if (n <= 1) return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
public static void hanoi(int n, String origen, String auxiliar, String destino) {
    if (n == 1) {
        System.out.println("Mover disco 1 de " + origen + " a " + destino);
        return;
    }
    hanoi(n - 1, origen, destino, auxiliar);
    System.out.println("Mover disco " + n + " de " + origen + " a " + destino);
    hanoi(n - 1, auxiliar, origen, destino);
}

=== OUTPUT ===
=== Archivo de prueba de funciones (rubrica nueva) ===
=== 1.3.1 Funciones no recursivas sin parámetros (obligatorio) ===
###Validacion Manual
¡Hola, mundo!
obtenerNumero() -> 42
OK 1.3.1
\n==== 1.3.2 Funciones no recursivas con parámetros (obligatorio)
===
###Validacion Manual
¡Hola, Juan!
sumar(10,20) -> 30
OK 1.3.2
\n==== 1.3.3 Funciones recursivas ====
Factorial(5) -> 120
fibonacci(10) -> 55
###Validacion Manual (Hanoi)
Mover disco 1 de A a C
Mover disco 2 de A a B
Mover disco 1 de C a B
Mover disco 3 de A a C
Mover disco 1 de B a A
Mover disco 2 de B a C
Mover disco 1 de A a C
\n==== 1.3.4 Parseo de enteros ====
\n"123"\n -> 123
OK 1.3.4
\n==== 1.3.5 Parseo de flotantes ====
\n"123.45"\n -> 123.45
\n"123"\n -> 123
```

At the bottom of the window, there are four buttons: "Abrir .usl", "Guardar", "Analizar & Ejecutar", and "Limpiar".

- El botón limpiar borra todo de la parte visual de la GUI.