Manual Técnico

Organización de Lenguajes y Compiladores 1 – Proyecto 2

Aldo Saúl Vásquez Moreira
202109754

TYPEWISE

HERRAMIENTAS UTILIZADAS

- **Express**

  Express es un framework de desarrollo web para Node.js. Permite crear aplicaciones web y APIs de manera rápida y sencilla utilizando una variedad de características como el manejo de rutas, el renderizado de vistas, el manejo de solicitudes y respuestas, entre otras. Express es muy popular en el mundo de la programación web debido a su facilidad de uso y su capacidad para escalar aplicaciones web de manera eficiente.

- **Ejs**

  Ejs es un motor de plantillas para JavaScript que se utiliza principalmente en aplicaciones web. Ejs permite a los desarrolladores crear plantillas dinámicas que se pueden utilizar para generar HTML en el lado del servidor. Esto es útil para crear páginas web dinámicas que cambian en función de los datos que se reciben del servidor o de las interacciones del usuario. Ejs es muy popular debido a su sintaxis sencilla y a la facilidad con la que se integra con otros frameworks web como Express.

- **TypeScript**

  TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript. TypeScript añade características adicionales a JavaScript, como el tipado estático, lo que ayuda a los desarrolladores a detectar errores en el código antes de que se ejecute. Además, TypeScript también es compatible con todas las características de JavaScript y se puede utilizar en cualquier lugar donde se pueda utilizar JavaScript. TypeScript es muy popular en el mundo de la programación web debido a su capacidad para ayudar a los desarrolladores a escribir código más seguro y escalable.

- **Nodemon**

  Nodemon es una herramienta de línea de comandos para Node.js que se utiliza para automatizar el proceso de reinicio de una aplicación cuando se realizan cambios en el código. Nodemon monitoriza los archivos de una aplicación Node.js en busca de cambios y, cuando se detecta uno, reinicia automáticamente la aplicación. Esto ahorra tiempo y esfuerzo en el proceso de desarrollo al permitir que los desarrolladores realicen cambios y vean los resultados inmediatamente sin tener que reiniciar manualmente la aplicación cada vez.

- **Jison**

  Jison es un generador de analizadores sintácticos que se utiliza para analizar el lenguaje natural o el código fuente de un programa. Jison toma una gramática definida por el usuario y genera un analizador sintáctico en JavaScript que se puede utilizar para analizar el texto de entrada y generar un árbol de sintaxis abstracta (AST) que representa la estructura del lenguaje. Esto es útil para la construcción de intérpretes y compiladores para lenguajes de programación personalizados. Jison es muy popular en el mundo de la programación de lenguajes y se utiliza en una variedad de proyectos de código abierto y comerciales.

```json
{
    "name": "interpretets",
    "version": "0.0.0",
    "private": true,
    ▷ Depurar
    "scripts": {
        "prepublishOnly": "npm audit --audit-level=moderate"
    },
    "dependencies": {
        "@types/random": "^3.0.1",
        "cookie-parser": "1.4.6",
        "debug": "4.3.4",
        "ejs": "^3.1.9",
        "express": "4.18.2",
        "http-errors": "2.0.0",
        "morgan": "1.10.0"
    },
    "devDependencies": {
        "@types/node": "^18.15.3",
        "tslint": "^5.12.1",
        "typescript": "^3.3.3"
    }
}
```

tsconfig.json > {} compilerOptions

```json
{
  "compilerOptions": {
    "module": "commonjs",
    "esModuleInterop": true,
    "target": "ES2016",
    "moduleResolution": "node",
    "sourceMap": true,
    "outDir": "dist",
  },
  "lib": ["es2016"]
}
```

- **Patrón intérprete:**

  El patrón Intérprete es un patrón de diseño de software que se utiliza para definir un lenguaje y un intérprete que lo analice y ejecute las acciones correspondientes. Este patrón se utiliza para construir un analizador sintáctico personalizado que pueda interpretar el lenguaje que se ha definido.

  El patrón Intérprete consta de tres componentes principales:

  El Contexto: que representa el estado actual del programa y proporciona información sobre el programa que se está interpretando.

  La Expresión: que representa las distintas partes del lenguaje y define cómo se interpretan. Las expresiones se pueden combinar para formar expresiones más complejas y el intérprete las evaluará según corresponda.

  El Intérprete: que analiza el texto de entrada y utiliza las expresiones para interpretar el lenguaje. El intérprete recibe el contexto y una serie de expresiones y las utiliza para evaluar el programa.

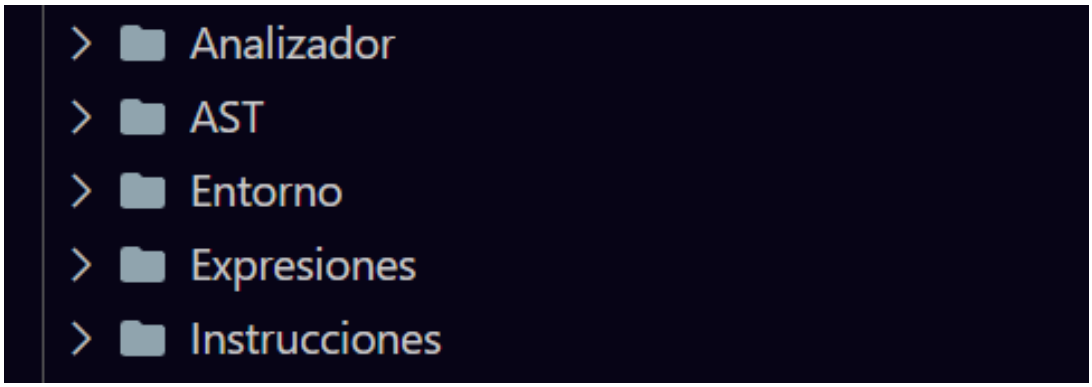
  El patrón Intérprete se utiliza a menudo en la construcción de lenguajes de programación, donde se utiliza para analizar el código fuente y generar un árbol de sintaxis abstracta que luego se utiliza para generar código ejecutable. También se utiliza en la construcción de sistemas de procesamiento de lenguaje natural, donde se utiliza para analizar el lenguaje humano y extraer información relevante.

  Aunque el patrón Intérprete puede ser muy útil en ciertas situaciones, puede ser complejo de implementar y puede tener un impacto en el rendimiento de

una aplicación. Por esta razón, este patrón se utiliza a menudo en situaciones específicas donde la simplicidad y la capacidad de mantenimiento son más importantes que el rendimiento absoluto.

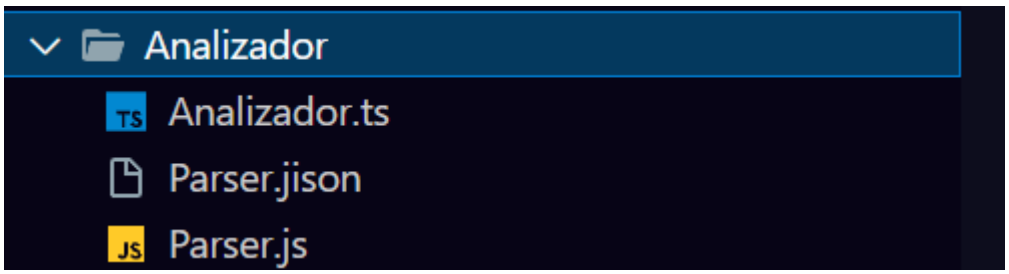
ESTRUCTURA DEL PROGRAMA

Sabiendo lo que es el patrón intérprete, procederemos a visualizar la estructura del proyecto.



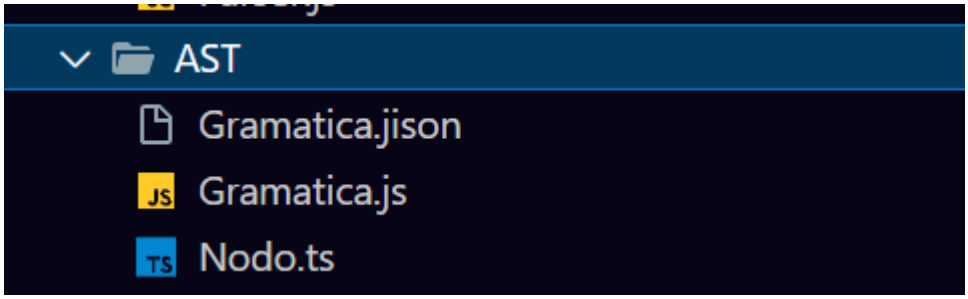
Se encuentra organizado por carpetas, una que contiene las instrucciones y expresiones necesarias para trabajar con dicho patrón. Un entorno o cómo se menciona anteriormente, el contexto. Las carpetas Analizador y AST contienen las gramáticas en los archivos .JISON para generar los parser, tanto para la ejecución del programa como para la generación del Árbol de Análisis Sintáctico o mejor conocido como AST.

CARPETA ANALIZADOR



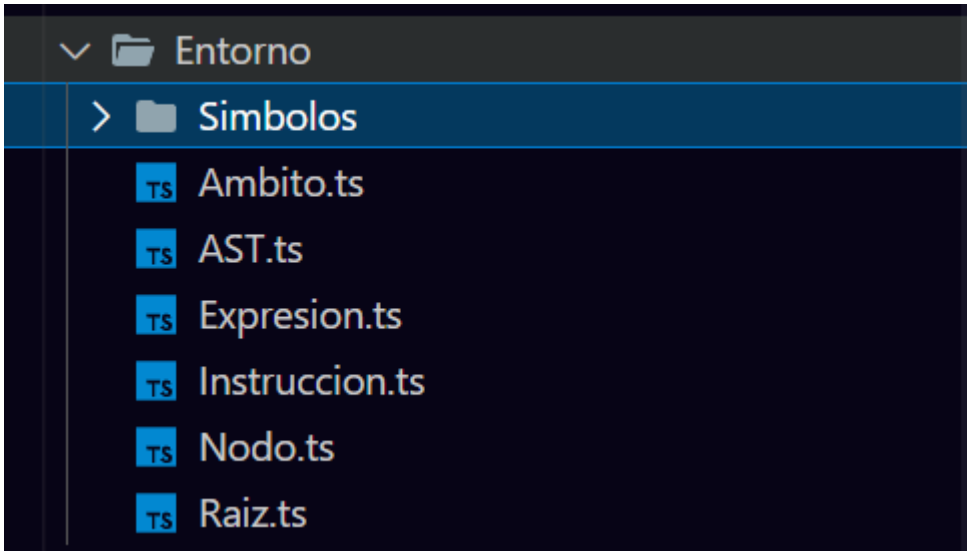
Contiene una clase analizador desarrollado con TS para ejecutar el código compilado del archivo Parser.js.

CARPETA AST



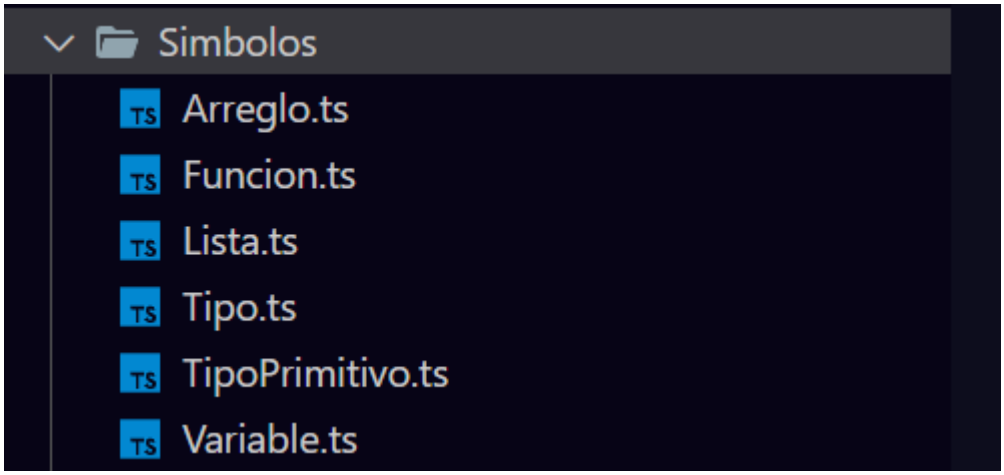
La estructura es similar a la de la carpeta anterior. Difiere en que este parser está destinado a la generación del AST. La clase nodo representa cada uno de los nodos que conformarán el árbol generado.

CARPETA ENTORNO



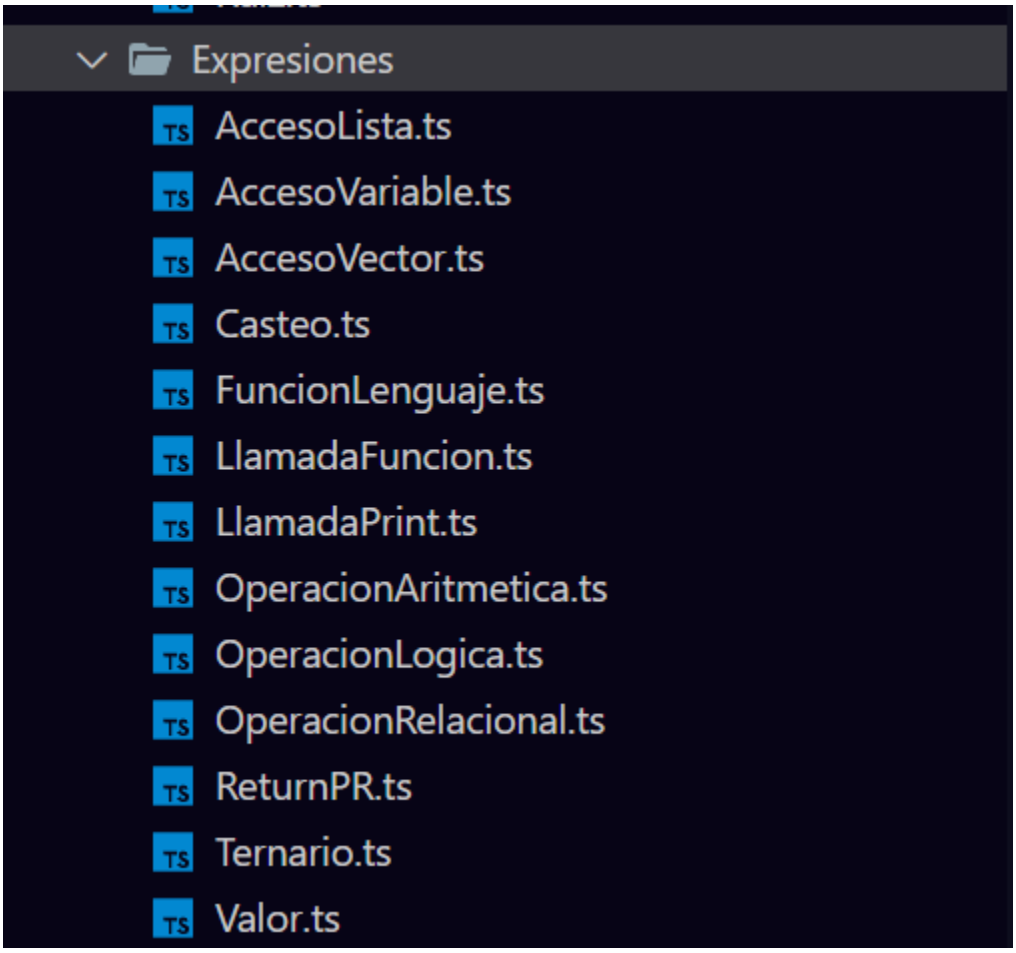
Dicha carpeta es la esencia del patrón intérprete ya que este contiene las clases abstractas que apoyan a las demás clases para determinar si la sentencia procesada se trata de una instrucción o una expresión. Así como la clase ámbito, la cual hace referencia al entorno en el cual se encuentra en ese momento. Es importante recalcar que es importante conservar la esencia de estas clases ya que de no ser así el programa no funcionaría.

CARPETA ENTORNO/SÍMBOLOS



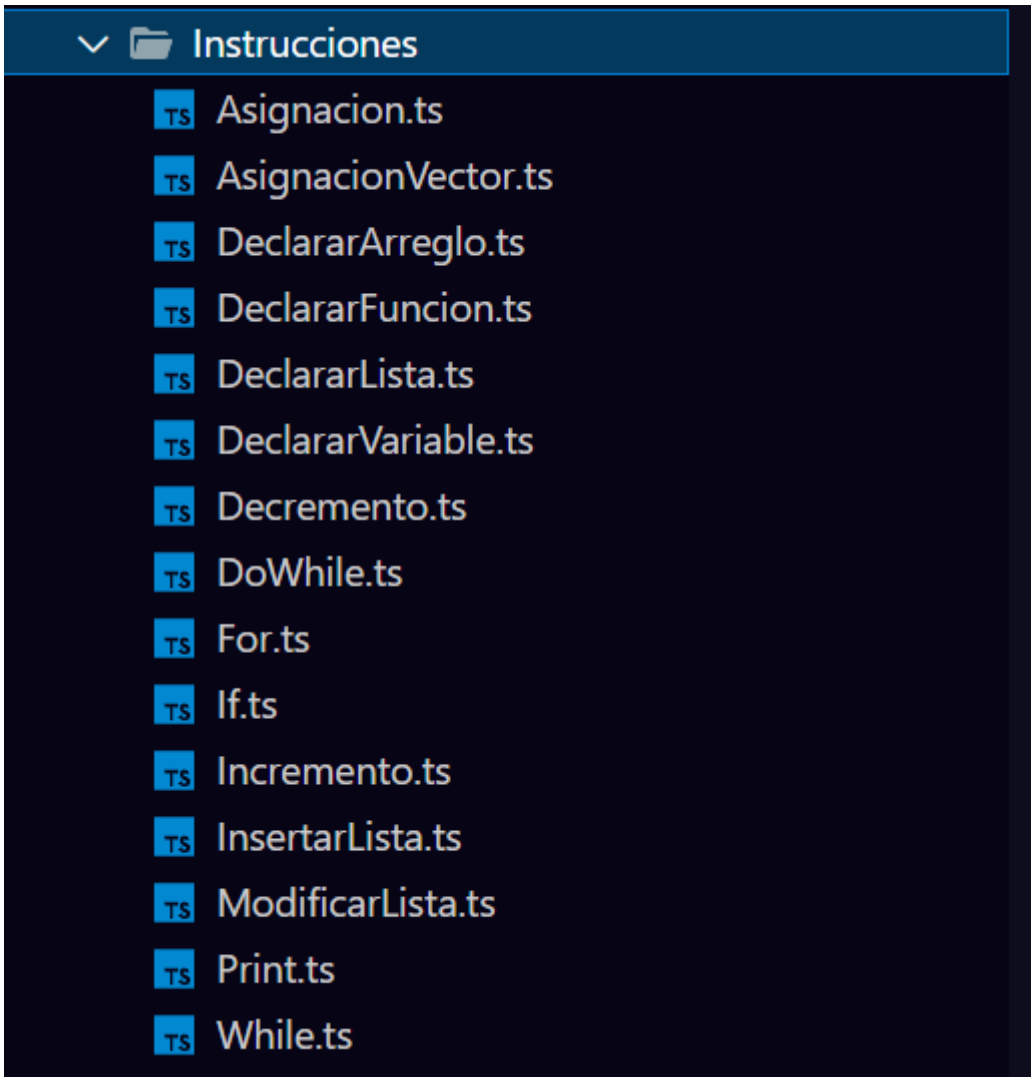
Esta contiene los símbolos que estarán presentes durante la ejecución del programa. Como podemos observar contiene una clase Tipo y TipoPrimitivo para determinar el tipo de las variables declaradas. Además, de las clases que hacer referencias a las estructuras de datos como las listas y vectores y las funciones.

CARPETA EXPRESIONES



Acá se encuentran cada una las clases que retornan un valor. Por ello, todas estas clases heredan de la clase Expresión mencionada anteriormente.

CARPETA INSTRUCCIONES



Acá tenemos a todas las clases que tienen como tarea realizar una acción. Por ello, heredan de la clase instrucción e implementan el método ejecutar.

Básicamente, ese es el funcionamiento del proyecto. Únicamente se deben comprender muy bien el patrón intérprete, luego únicamente es lógica de programación.