Manual técnico

El interprete TypeWise se programó en el lenguaje de programación TypeScript que es un superconjunto de JavaScript.

Así mismo, también se utilizó la herramienta Jison para generar el análisis léxico y sintáctico.

Se utilizaron 2 clases abstractas llamadas Expresión e Instrucción. Estas clases abstractas son el molde para las demás clases.

```
export abstract class Expresion {

    public linea: number;
    public columna: number;

    constructor(linea: number, columna: number) {
        this.linea = linea;
        this.columna = columna;
    }

    public abstract ejecutar(entorno:Ambiente,consola:Consola):any;

    public OperacionD(tipoA: Tipo,tipoB:Tipo):Tipo{
        return Tabla[tipoA][tipoB];
    }
}
```

La clase expresión recibe 2 parámetros, tales como línea y columna y un método abstracto llamado ejecutar. Esta clase por lo general se encargará de retornar objetos o valores que necesite el intérprete.

Clase abstracta Instrucción:

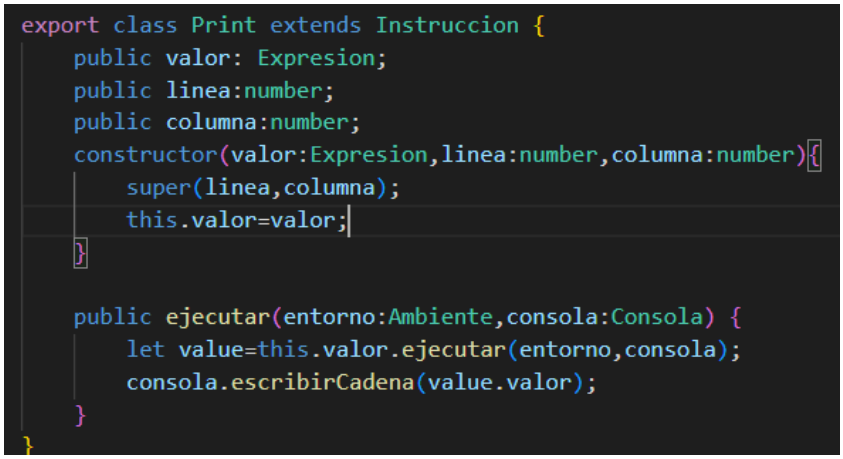
Esta clase se encarga de ejecutar las distintas instrucciones que requiera el interprete por ejemplo ejecutar la instrucción print. Por lo tanto, también implementa el método abstracto ejecutar.

```
export abstract class Instruccion{
    public linea:number;
    public columna:number;
    constructor(linea:number,columna:number){
        this.linea=linea;
        this.columna=columna;
    }

    public abstract ejecutar(entorno:Ambiente,consola:Consola):any;
```

Función print:

Esta función se ejecuta al encontrar un objeto instanciado de la clase Print. Así mismo, dentro de la función print puede venir una expresión, ya se una operación aritmética, un retorno de una llamada a función, etc.



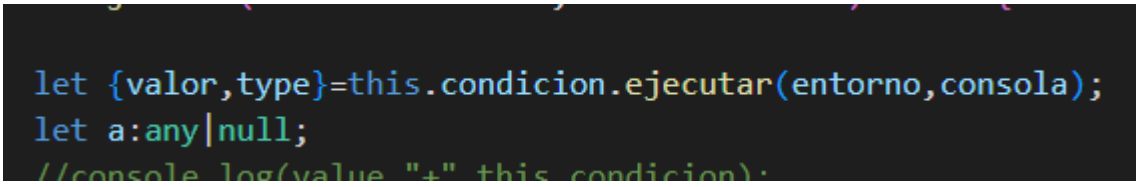
```
export class Print extends Instruccion {
    public valor: Expresion;
    public linea:number;
    public columna:number;
    constructor(valor:Expresion,linea:number,columna:number){
        super(linea,columna);
        this.valor=valor;
    }

    public ejecutar(entorno:Ambiente,consola:Consola) {
        let value=this.valor.ejecutar(entorno,consola);
        consola.escribirCadena(value.valor);
    }
}
```

La función print antes de escribir en consola debe ejecutar la expresión que viene dentro para posteriormente escribir en la consola de salida.

Clase IF:

Esta clase ejecuta primeramente la condición



```
let {valor,type}=this.condicion.ejecutar(entorno,consola);
let a:any|null;
```

Esta ejecución devuelve la comparación de dicha condición. Seguidamente procede a verificar dicha comparación

```
if(valor){
    a=this.list.ejecutar(entorno,consola);
    if(a!=null) return a;

}else{
    if(this.elsf!=null){
        this.elsf.ejecutar(entorno,consola);
    }
}
```

Si el valor de retorno es un true entonces se ejecuta el bloque de instrucciones dentro de la instrucción if. De lo contrario se ejecutan las instrucciones else si las hubiese.

Entornos:

Cada vez que se encuentra con la sintaxis { }, se procede a crear un nuevo entorno. En dicho entorno se pueden crear nuevas variables, así como nuevas funciones. Una variable puede estar declarada con el mismo nombre si se encuentran en distinto entorno.

```
public ejecutar(entorno: Ambiente,consola:Consola){
    let nuevo=new Ambiente(entorno);
    let a:any;
```

En este caso se crea un nuevo entorno pues se encuentra con una instrucción { }. Con esto ya se pueden guardar variables en este nuevo entorno, así como funciones.

```
constructor(anterior: Ambiente | null) {
    this.anterior = anterior
    this.variables = new Map();
    this.metodos = new Map();
}
```

Al declarar un entorno nuevo este tiene una referencia al entorno anterior ya que la variable utilizada puede encontrarse en otro entorno por lo que buscara la variable o función en cada uno de los entornos.