UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

MANUAL TÉCNICO DATAFORGE

PAKAL B'ALAM RODRIGUEZ ESPANTZAY
202201457
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

Un analizador léxico convierte el texto de entrada en una secuencia de token. Los tokens son unidades léxicas como identificadores, palabras clave, números y símbolos. El analizador sintáctico es aquella que se encarga de analizar la estructura gramatical de la secuencia de tokens generada por el analizador léxico. Utiliza una gramática formal para verificar si la secuencia de token sigue las reglas sintácticas del lenguaje de programación.

A continuación, se detallarán algunas herramientas utilizadas para este proyecto:

JFlex: Es una herramienta para generar analizadores léxicos (scanners) en Java. Se utiliza para analizar el texto de entrada y dividirlo en token para el análisis sintáctico posterior.

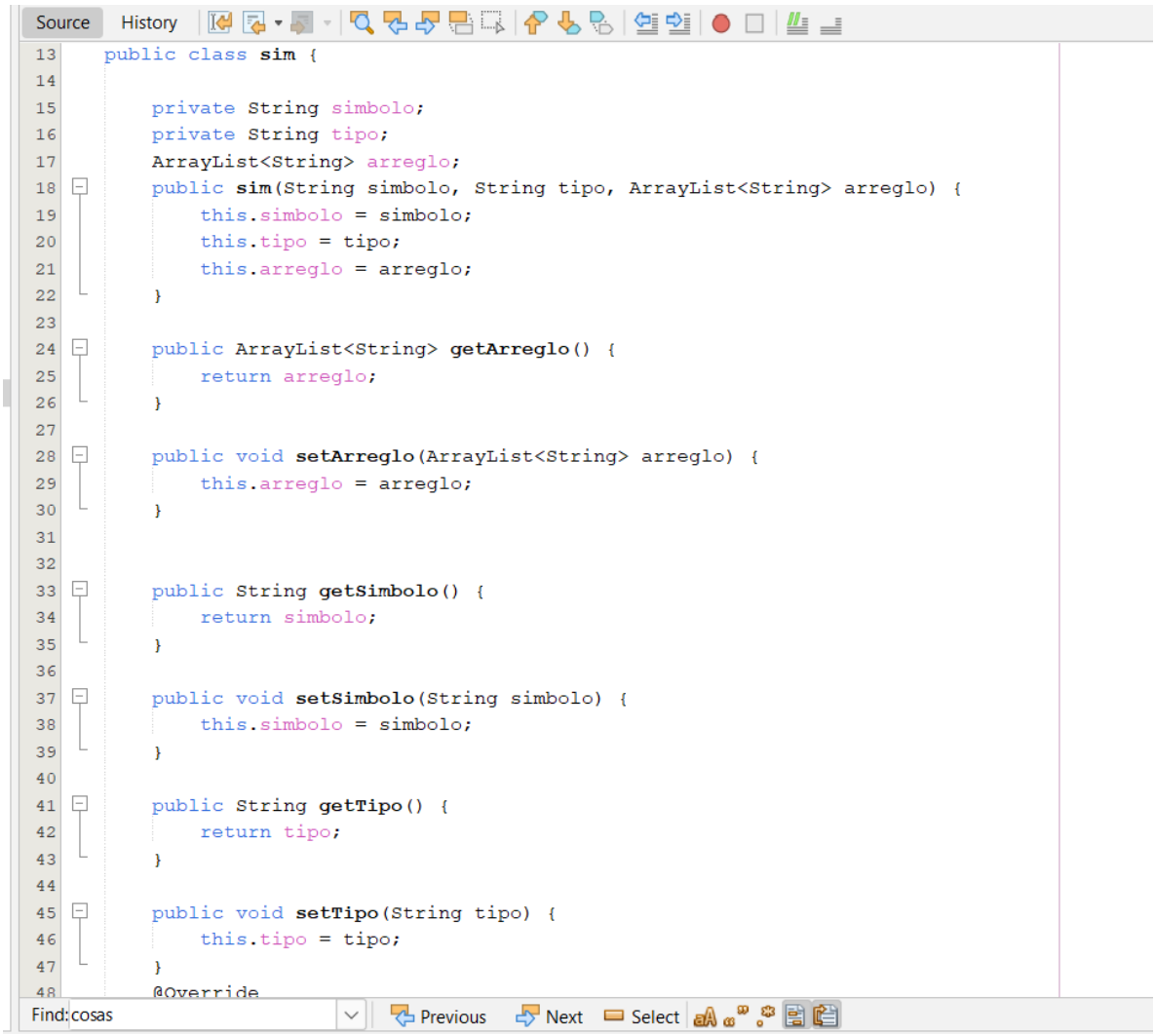
CUP: Es un generador de analizadores sintácticos (parsers) para Java. Se utiliza para definir y generar el analizador sintáctico basado en la gramática especificada.

Estructura del Código

La aplicación está contenida principalmente en 2 paquetes que es donde se trabajó todo, siendo estos los paquetes de main y analizadores, en estos paquetes se desarrollan las distintas clases y formularios necesarios para que se ejecute la aplicación de manera satisfactoria, además la interfaz se realizó con ayuda de un JFrame Form y esta está vinculada a la vez con el jflex y cup.

Clases importantes:

Entre las clases más importantes está el Léxico.jflex, Parser.cup y el archivo main, estos son nuestros archivos principales que luego se complementarán con otros archivos del proyecto. El jflex es el que contiene el analizador léxico, el cup está a cargo del analizador sintáctico. En otro archivo está Reportes que está a cargo de generar nuestros reportes de tokens, errores y símbolos. En reportes se utilizó html para poder ejecutarlo de manera correcta y rápida en el navegador. Además de estos archivos se tienen otros archivos como es el de sim y tokens que nos ayudan a registrar cada símbolo y token.



```java
public class sim {

    private String simbolo;
    private String tipo;
    ArrayList<String> arreglo;
    public sim(String simbolo, String tipo, ArrayList<String> arreglo) {
        this.simbolo = simbolo;
        this.tipo = tipo;
        this.arreglo = arreglo;
    }

    public ArrayList<String> getArreglo() {
        return arreglo;
    }

    public void setArreglo(ArrayList<String> arreglo) {
        this.arreglo = arreglo;
    }


    public String getSimbolo() {
        return simbolo;
    }

    public void setSimbolo(String simbolo) {
        this.simbolo = simbolo;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }
    @Override
```

Find: cosas Previous Next Select

```java
    @Override
    public String toString() {
        if (arreglo != null) {
            return "Isimbolo: " + simbolo + ", Tipo: " + tipo + ", arreglo: " + arreglo;
        } else {
            return "Isimbolo: " + simbolo + ", Tipo: " + tipo;
        }
    }

}
```

Acá tendremos nuestra clase de sim que será corto para símbolos con su respectivo código.

Luego tenemos el archivo instrucción.java:
public class Instruccion {

```
    public void imprimirVariables(ArrayList<String> lista){
            //debe recibir un array list e imprimirlo puede tenner un solo elemento se
separan por ,
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < lista.size(); i++) {
            sb.append(lista.get(i));
            if (i < lista.size() - 1) {
                sb.append(", "); // Agregar coma y espacio entre elementos
            }
        }
        System.out.println(sb.toString());
    }
    public static void imprimirHashMap(HashMap<String, sim> hashMap) {
        System.out.println("Contenido del HashMap:");

        for (Map.Entry<String, sim> entry : hashMap.entrySet()) {
            String clave = entry.getKey();
            sim valor = entry.getValue();
            System.out.println("Clave: " + clave  + valor.toString());
            if (valor.getArreglo() != null) {
                System.out.println("qArreglo: " + valor.getArreglo());
            } else {
                System.out.println(valor.toString());
            }
        }
    }

        public static double calcularMediana(ArrayList<String> arreglo) {
        double[] numeros = convertirAArrayDouble(arreglo);
        java.util.Arrays.sort(numeros);
```

```
    int n = numeros.length;
    if (n % 2 != 0) {
        return numeros[n / 2];
    } else {
        double num1 = numeros[n / 2 - 1];
        double num2 = numeros[n / 2];
        return (num1 + num2) / 2.0;
    }
}

public static double calcularVarianza(ArrayList<String> arreglo) {
    double[] numeros = convertirAArrayDouble(arreglo);
    double media = calcularMedia(arreglo);
    double sumatoria = 0.0;

    for (double num : numeros) {
        double diferencia = num - media;
        sumatoria += Math.pow(diferencia, 2);
    }

    return sumatoria / numeros.length;
}

  public static double calcularMedia(ArrayList<String> arreglo) {
    double sumatoria = 0.0;
    int count = 0;

    for (String str : arreglo) {
        try {
            double num = Double.parseDouble(str);
```

```
                sumatoria += num;
                count++;
            } catch (NumberFormatException e) {
                System.out.println("Error al convertir el elemento " + str + " a double. El elemento será ignorado.");
            }
        }

        if (count == 0) {
            System.out.println("No se encontraron elementos válidos para calcular la media.");
            return 0.0;
        }

        return sumatoria / count;

    }

    public static double calcularMaximo(ArrayList<String> arreglo) {
        double[] numeros = convertirAArrayDouble(arreglo);
        double maximo = numeros[0];

        for (double num : numeros) {
            if (num > maximo) {
                maximo = num;
            }
        }

        return maximo;
    }

    public static double calcularMinimo(ArrayList<String> arreglo) {
```

```
    double[] numeros = convertirAArrayDouble(arreglo);
    double minimo = numeros[0];

    for (double num : numeros) {
        if (num < minimo) {
            minimo = num;
        }
    }

    return minimo;
}

public static double calcularModa(ArrayList<String> arreglo) {
    double[] numeros = convertirAArrayDouble(arreglo);
    Map<Double, Integer> conteo = new HashMap<>();

    for (double num : numeros) {
        if (conteo.containsKey(num)) {
            conteo.put(num, conteo.get(num) + 1);
        } else {
            conteo.put(num, 1);
        }
    }

    int maxFrecuencia = 0;
    double moda = Double.NaN;

    for (Map.Entry<Double, Integer> entry : conteo.entrySet()) {
        if (entry.getValue() > maxFrecuencia) {
            maxFrecuencia = entry.getValue();
```

```
            moda = entry.getKey();
        }
    }

    return moda;
}

private static double[] convertirAArrayDouble(ArrayList<String> arreglo) {
    double[] numeros = new double[arreglo.size()];

    for (int i = 0; i < arreglo.size(); i++) {
        try {
            numeros[i] = Double.parseDouble(arreglo.get(i));
        } catch (NumberFormatException e) {
            System.out.println("Error al convertir el elemento " + arreglo.get(i) + " a double.
Se asignará el valor 0.0");
            numeros[i] = 0.0;
        }
    }

    return numeros;
}

public float media(ArrayList<String> arreglo) {
// Convertir elementos a float
ArrayList<Float> elementos = new ArrayList<>();
for (String elemento : arreglo) {
    try {
        elementos.add(Float.valueOf(elemento));
    } catch (NumberFormatException e) {
        System.out.println("Error al convertir elemento a float: " + e.getMessage());
```

```
        }
    }

    // Calcular media
    float suma = 0;
    for (float numero : elementos) {
        suma += numero;
    }
    return suma / elementos.size();
}

public float moda(ArrayList<String> arreglo) {
    // Convertir elementos a float
    ArrayList<Float> elementos = new ArrayList<>();
    for (String elemento : arreglo) {
        try {
            elementos.add(Float.valueOf(elemento));
        } catch (NumberFormatException e) {
            System.out.println("Error al convertir elemento a float: " + e.getMessage());
        }
    }

    // Calcular moda
    float moda = 0;
    int maximo = 0;
    for (int i = 0; i < elementos.size(); i++) {
        int contador = 0;
        for (int j = 0; j < elementos.size(); j++) {
            if (elementos.get(j) == elementos.get(i)) {
                contador++;
```

```
            }
        }
        if (contador > maximo) {
            moda = elementos.get(i);
            maximo = contador;
        }
    }
    return moda;
}

public float mediana(ArrayList<String> arreglo) {
    // Convertir elementos a float
    ArrayList<Float> elementos = new ArrayList<>();
    for (String elemento : arreglo) {
        try {
            elementos.add(Float.valueOf(elemento));
        } catch (NumberFormatException e) {
            System.out.println("Error al convertir elemento a float: " + e.getMessage());
        }
    }

    // Calcular mediana
    Collections.sort(elementos);
    int n = elementos.size();
    if (n % 2 == 0) {
        return (elementos.get(n / 2) + elementos.get(n / 2 - 1)) / 2;
    } else {
        return elementos.get(n / 2);
    }
}
```

```
public float varianza(ArrayList<String> arreglo) {
    // Convertir elementos a float
    ArrayList<Float> elementos = new ArrayList<>();
    for (String elemento : arreglo) {
        try {
            elementos.add(Float.valueOf(elemento));
        } catch (NumberFormatException e) {
            System.out.println("Error al convertir elemento a float: " + e.getMessage());
        }
    }

    // Calcular varianza
    float media = this.media(arreglo);
    float suma = 0;
    for (float numero : elementos) {
        suma += Math.pow(numero - media, 2);
    }
    return suma / elementos.size();
}



}
```

Acá es el código para poder realizar y devolver las operaciones que nos puedan venir en el archivo.