Universidad de San Carlos de Guatemala.

Facultad de ingeniería.

Ingeniería en Ciencias y Sistemas.

Laboratorio Organización de Lenguajes y Compiladores 1.

Laboratorista: Ing. Kevin Adiel Lajpop Ajpacaja

Proyecto 1 – SubSetify (Manual Técnico)

Estudiante: Evelio Marcos Josué Cruz Soliz.

Carnet: 202010040

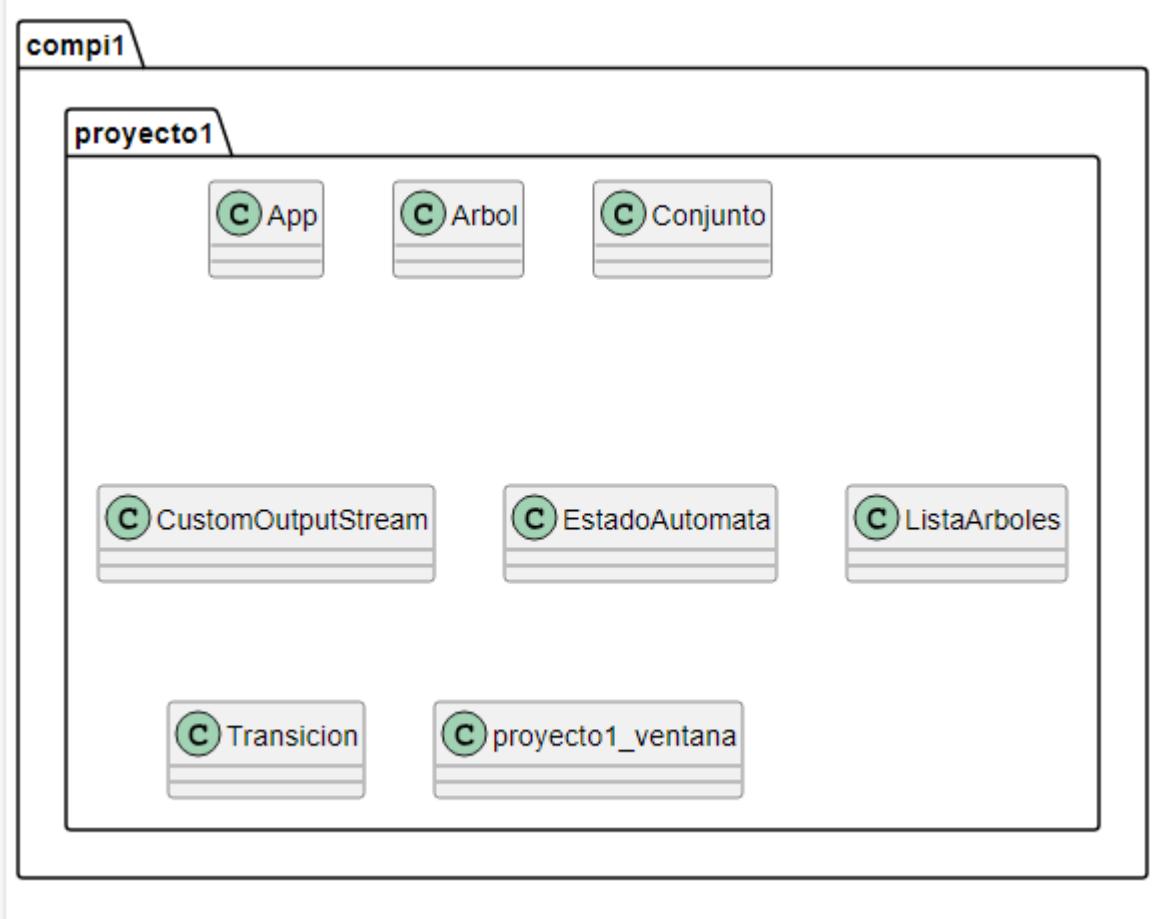
Guatemala, 17 de diciembre. de 2023

Entorno donde se realizó la solución:

Para la realización de este proyecto se utilizó el IDE NetBeans el cual es un entorno de desarrollo integrado (IDE) muy versátil que ofrece una amplia gama de herramientas para programadores, soporta múltiples lenguajes de programación, incluyendo Java, C/C++, PHP, HTML, y más. Las razones por las cuales se eligió este IDE son las siguientes:

- **Compatibilidad con CUP**. En eclipse suele ocurrir una incompatibilidad con la librería de cup utilizada, lo que genera errores en el IDE que suelen ser tediosos de corregir.
- **Compatibilidad con proyecto de Maven**. Al momento de ejecutar una aplicación, esta genera automáticamente los archivos resultantes de las librerías de Maven, es decir no se debe realizar la generación manual con cmd y mover las clases resultantes a la carpeta donde se encuentra el App.java.
- **Facilidad de creación de interfaces gráficas**. NetBeans ofrece una buena mantenibilidad de aplicaciones graficas gracias a su diseño intuitivo y a su capacidad “Drag and Drop” nativa.

Diagrama de clases.



Funcionamiento del método de Thomson.

Para poder realizar el método de Thompson se utilizaron estructuras de datos las cuales son las siguientes.

- **Transición**. Es la estructura más básica y solamente apunta hacia un nodo siguiente.
- **Nodo**. Es un estado que puede tener el autómata, este posee múltiples transiciones. Es aquí donde se realizan las distintas operaciones de loa autómatas.
- **Conjunto**. Un conjunto es una clase que simplificará la lógica de los nodos y transiciones, el conjunto más sencillo será:

S0 ---------- a ----------> S1

Una vez entendidos estos conceptos, se explicarán los métodos principales utilizados.

Concatenación.

La concatenación es el apuntado que toma una transición ya predefinida y la cambia. Los pasos para realizarla son los siguientes:

```
------------> Metodo para concatenar INICIO
La concatenacion es el apuntado que toma una transicion ya predefinida y la cambia
(Se asume que S0 es inicial y S2 es final

TR 1.        (S0)               ---------a--------->      (S1)
TR 2.    (Sx)               --------b--------->      (S2)

Paso 1. Toma el puntero o los punteros de Sx y se lo asigna a S1

(S0) ------ a ------> (S1) ------- b -------> (S2)

*/
Asume this = S0
No importa que objeto concatene, lo que importan son los argumentos
```

OR. Las transiciones para los OR se pueden resumir en concatenaciones con EPSILON más un reajuste de los nodos. Los pasos para realizar esta operación son los siguientes:

```
S0.add(Nueva_trancision_1)
S0.add(Nueva_trancision_2)

                             -----> (S1) ---a-----> (S2)
                            /e
Inicial (SO)
                            \e
                             ----> (S3) ----b-----> (S4)

5. Se crea una instancia, cuyo numero es uno mayor que la instancia posterior de b
(S5)

6. se crean trancisiones hacia ese nodo y se apuntan
Nueva_trancision_3 = e-------> (S5)
final_a.add(Nueva_trancision_3)
final_b.add(Nueva_trancision_3)
                      -----> (S1) ---a-----> (S2) \
                             /e                    \
Inicial (SO)                                        (S5)
                             \e                    /
                              ----> (S3) ----b-----> (S4) /


7. Se retorna el nodo final
*/
Se asume que el estado actual es S0, es decir que this = a.inicial
```

```
Por ejemplo se tienen
TR 1.        (S0)              --------a--------->      (S1)
TR 2.        (S2)              --------b--------->      (S3)

Donde S0, S1 son los estados iniciales y finales del conjunto a
    y S2, S3 son los estados iniciales y finales del conjunto b
El procedimiento (a|b) para el OR seria el siguiente:

1. Se crea una copia de S0 que sera el predecesor de a y S0 pasara a ser el nodo inicial y se limpia S0

2. Se toma el numero del predecesor de a y se le suma 1,
tambien se le suma 1 al estado final
(S1) ------ a --------- > (S2)

3. Se repite el paso 2 con b
(S3) ----b-----> (S4)

4. S0 se conecta con el predecesor de a y con el predecesor de b, mediante epsilon
Nueva_trancision_1 = e ----> (S1)
Nueva_trancision_2 = e ----> (S3)

S0.add(Nueva_trancision_1)
S0.add(Nueva_trancision_2)
```

Kleene.

```
Suponer que a es un conjunto de la manera
(S0) --------- a -----------> (S1)
prev_a         a              final_a

Paso 1.
Crear una copia de prev_a y prev_a se convierte en el nodo inicial

(S0) (S0_copia) --------- a -----------> (S1)

Paso 2.
Sumarle 1 al los estados de a

(S0) (S1) --------- a -----------> (S2)

Paso 3.
Crear un nodo final con el estado final_a + 1
nodo_final = S3

(S0) (S1) --------- a -----------> (S2) (S3)

Paso 4.
Conectar final_a ---Epsilon --> copia
```

```
(S0) (S1) --------- a -----------> (S2) (S3)
      ^                            |
       ----------------------------

Paso 5.
Conectar S0 con S3
 (S0) (S1) --------- a -----------> (S2) (S3)
   |    ^                          |     ^
   |    ----------------------------      |
   ----------------------------------------

Paso 6. Conectar (S0) con (S1) y (S2) con (S3)

 (S0)->(S1) --------- a -----------> (S2)->(S3)
   |    ^                          |     ^
   |    ----------------------------      |
   ----------------------------------------

Paso 7. Setea el nodo actual al nodo inicial, se guarda un nodo temporal y devielve el nodo final
```

Cerradura positiva. Se utiliza el concepto del libro: a+ = aa*