Universidad de San Carlos de Guatemala

Lenguajes Formales de Programación

Ing. Zulma Aguirre

Aux. Luis Yela

Manual Técnico

Elmer Gustavo Sánchez García

201801351

Fecha de entrega 31/03/2020

Menu Principal

```python
def menuMain():
    while True:
        print("------------------Menu Principal----------------------")
        print("|                                                    |")
        print("| 1.Crear AFD                                        |")
        print("| 2.Crear Gramatica                                  |")
        print("| 3.Evauluar Cadena                                  |")
        print("| 4.Reportes                                         |")
        print("| 5.Cargar Archivo de Entrada                        |")
        print("| 6.Guardar                                          |")
        print("|   SALIR                                            |")
        print("|                                                    |")
        print("------------------------------------------------------")
        print("")
        print(">> ",end="")
        opcion = input()
        opcion = opcion.strip()
        if opcion == "1":
            os.system("cls")
            menuAFD()
        elif opcion == "2":
            os.system("cls")
            menuGramatica()
        elif opcion =="3":
            os.system("cls")
            menuEvaluarCadena()
```

La función principal de este método es dar la opciones a escoger determinada función del programa como crear afd,crear gramática evaluar cadenas,generar reportes cargar archivos de entrada y guardar algún afd o gramática.

Menú ADF

```
def menuAFD():
    nombre = setNombreAFD()
    if nombre != "update":
        if ManejadorAFD.newAFD(nombre) == True:
            while True:
                print("----------------------Menu AFD-----------------------")
                print("|                                                   |")
                print("| 1.Ingresar Estados                                |")
                print("| 2.Ingresar Alfabeto                               |")
                print("| 3.Estado Inicial                                  |")
                print("| 4.Estado Aceptacion                               |")
                print("| 5.Transiciones                                    |")
                print("| 6.Ayuda                                           |")
                print("|   SALIR                                           |")
                print("|                                                   |")
                print("-----------------------------------------------------")
                print("")
                print(">> ",end="")
                opcion = input()
                opcion = opcion.strip()
                if opcion == "1":
                    os.system("cls")
                    ManejadorAFD.setEstadosUpdate(nombre)
                elif opcion == "2":
                    os.system("cls")
                    ManejadorAFD.setAlfabeto(nombre)
                elif opcion =="3":
                    os.system("cls")
                    ManejadorAFD.setEstadoInicialAFD(nombre)
                elif opcion == "4":
```

Este menú es el encargado de realizar todo lo debido al AFD, como ingresar estados, ingresar alfabetos, el estado inicial, estados de aceptación y transiciones.

Menú Gramática

```
def menuGramatica():
    nombre = setNombreGramatica()
    if nombre != "update":
        if ManejadorGramatica.newGramatica(nombre) == True:
            while True:
                print("---------------------Menu Gramatica--------------------")
                print("|                                                      |")
                print("| 1.Ingresar NT                                        |")
                print("| 2.Ingresar Terminales                                |")
                print("| 3.NT Inicial                                         |")
                print("| 4.Producciones                                       |")
                print("| 5.Mostar gramatica y Transformada                    |")
                print("| 6.Ayuda                                              |")
                print("|   SALIR                                              |")
                print("|                                                      |")
                print("--------------------------------------------------------")
                print("")
                print(">> ",end="")
                opcion = input()
                opcion = opcion.strip()
                if opcion == "1":
                    os.system("cls")
                    ManejadorGramatica.setNewNoTerminal(nombre)
                elif opcion == "2":
                    os.system("cls")
                    ManejadorGramatica.setNewTerminal(nombre)
                elif opcion == "3":
                    os.system("cls")
```

Este menu es el encargado de toda la gramtica como ingresar no terminales,ingresar Terminales, no terminal inicial ,produccion y mostrarar gramatica y transformada.

Alerta

```python
def alertaError(mensaje):
    os.system("cls")
    print("------------------------ALERTA-------------------------")
    print("|                                                     |")
    print("| Error:                                              |")
    print("| ->>No Se Realizo La Operacion                       |")
    print(f"| -->>{mensaje}                 ")
    print("|                                                     |")
    print("-------------------------------------------------------")

def alertaExito(mensaje):
    os.system("cls")
    print("------------------------ALERTA-------------------------")
    print("|                                                     |")
    print("| Operacion:                                          |")
    print(f"| ->>{mensaje}")
    print("|                                                      ")
    print("|                                                     |")
    print("-------------------------------------------------------")
```

Este son notificaciones por alguno posible error presentado en la ejecución del programa o cuando se quiere notificar alguna operación realizada.

Obtener Ruta

```python
def getRutaArchivo(nombre):
    tipo,objeto = getObjet(nombre.strip())
    if objeto != None and tipo != None:
        if tipo == "automata":
            root = Tk()
            root.filename =  filedialog.asksaveasfilename(initialdir = "/",title = "Select file",filetypes = (("AFD files","*.afd"),("all files",
            ruta = root.filename
            if is_empty(ruta.strip()) == False:
                ruta = f"{ruta}.afd"
                root.destroy()
                return (ruta.strip())
            else:
                alertaError("No escribio ningun nombre")
                root.destroy()
                return None
        elif tipo == "grammar":
            root = Tk()
            root.filename =  filedialog.asksaveasfilename(initialdir = "/",title = "Select file",filetypes = (("Grammar files","*.grm"),("all fil
            ruta = root.filename
            if is_empty(ruta.strip()) == False:
                ruta = f"{ruta}.grm"
                root.destroy()
                return (ruta.strip())
            else:
                alertaError("No escribio ningun nombre")
                root.destroy()
                return None
```

Este método se encarga de obtener la ruta dependiendo si es para una gramática o un afd, se encarga de forma mas visual, y mas sencilla de obtener la ruta.

Preparar para Escribir

```python
def transformarArchivo(nombre):
    nombre = nombre.strip()
    ruta = getRutaArchivo(nombre)
    tipo,objeto = getObjet(nombre.strip())
    if ruta != None:
        if tipo == "automata":
            texto = ""
            diccionario = objeto.getTrancisiones()
            listEstadoAceptacion = objeto.getEstadosDeAceptacion()
            for key,value in diccionario.items():
                for listCadena in value:
                    cadena = listCadena.split(" ")
                    alfabeto = cadena[0]
                    estado = cadena[1]
                    if searchInListAceptacion(key,listEstadoAceptacion) == False and searchInListAceptacion(estado,listEstadoAceptacion) == False
                        texto = f"{key},{estado},{alfabeto};false,false"
                        writeArchivo(texto,ruta)
                    elif searchInListAceptacion(key,listEstadoAceptacion) == False and searchInListAceptacion(estado,listEstadoAceptacion) == Tru
                        texto = f"{key},{estado},{alfabeto};false,true"
                        writeArchivo(texto,ruta)
                    elif searchInListAceptacion(key,listEstadoAceptacion) == True and searchInListAceptacion(estado,listEstadoAceptacion) == Fals
                        texto = f"{key},{estado},{alfabeto};true,false"
                        writeArchivo(texto,ruta)
                    elif searchInListAceptacion(key,listEstadoAceptacion) == True and searchInListAceptacion(estado,listEstadoAceptacion) == True
                        texto = f"{key},{estado},{alfabeto};true,true"
                        writeArchivo(texto,ruta)
```

Este método se encargara de prepara el texto, llevarlo a la forma pedida para luego escribirlo.

Escribir Archivo

```python
def writeArchivo(texto,ruta):
    archivo  = open(f"{ruta}","a")
    texto = f"{texto}\n"
    archivo.writelines(texto)
    archivo.close()
```

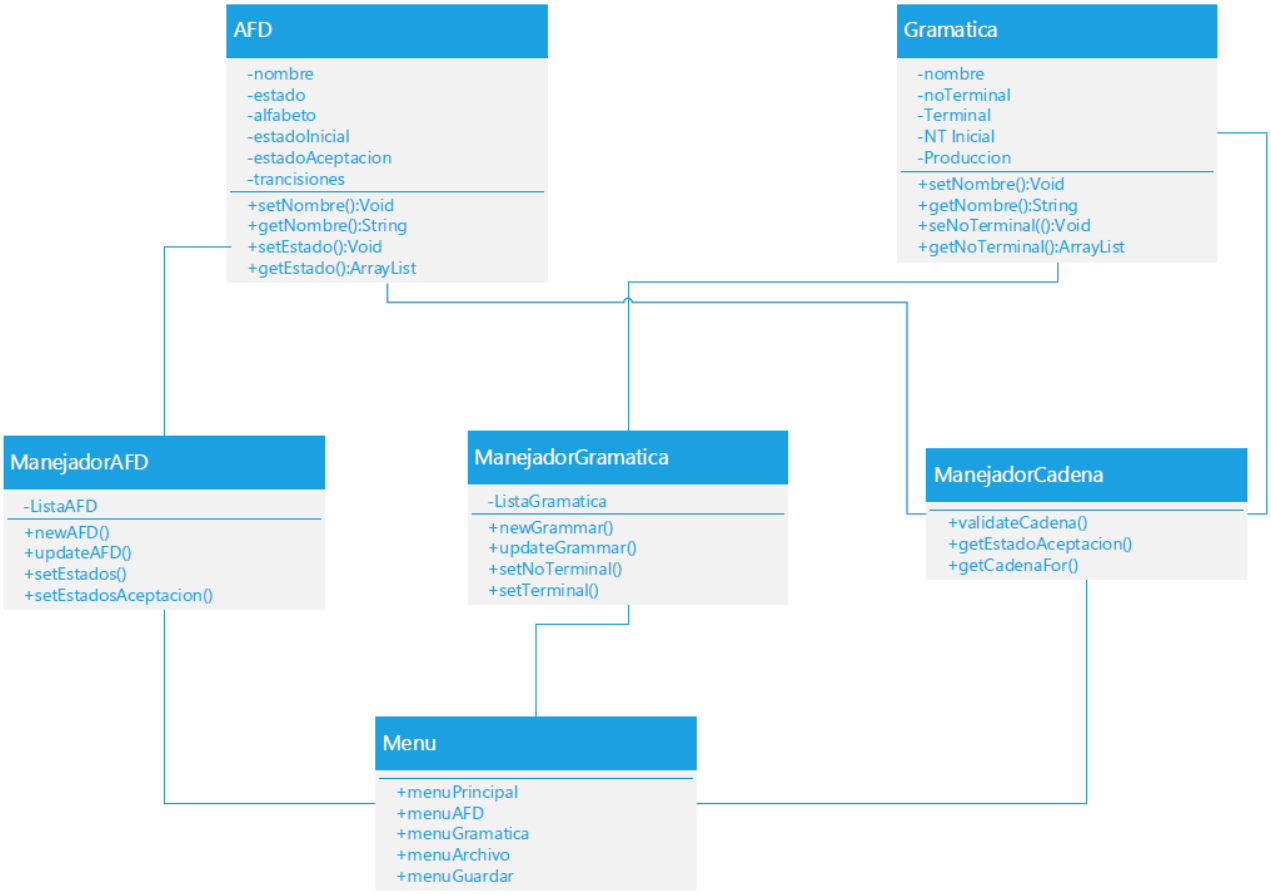
Este método se encarga de escribir archivos en la ruta indicada, recibe dos parámetros, el texto a escribir y la ruta.

Validar Cadena

```
def validateCadena(name,soloValidar,forma):
    try:
        cadena = getCadena()
        cade = cadena
        name = name.strip()
        text = ""
        condicion = ""
        tipo,objeto = getObjet(name)
        if tipo != None and objeto != None:
            if tipo == "automata" and forma == "automata":
                estadoInicial = objeto.getEstadoInicial()
                estadoInicial = estadoInicial.strip()
                estadosAceptacion = objeto.getEstadosDeAceptacion()
                listaIncial = ManejadorAFD.getListDiccionario(objeto,estadoInicial)
                texto = estadoInicial
                listaTrancion = objeto.getTrancisiones()
                textoPrimero = ""
                listaCadena = objeto.getCadena()
                listCadenaValida = objeto.getCadenaValida()
                listCadenaNoValida = objeto.getCadenaNoValida()
                if estadoInicial != None and estadosAceptacion != None and listaIncial != None:
                    #print(listaTrancion)
                    #print("Estado de aceptacion:",estadosAceptacion)
                    #print("--------------------------------------")
                    for i in range(len(cadena)):
                        valor = ""
                        if i == 0:
                            if i != len(cadena) - 1:
                                valor = getEstadoCadena(cadena[i],listaIncial)
                                if valor != None:
                                    destino = valor.split(" ")
                                    con = destino[0]
```

Este método es el encargado de interpreta las cadenas a validar , ya sea por el afd o la gramática mostrara valida o no valida dependiendo la situación también es capaz de expandir por gramática o mostrar ruta por afd

UML


AFD
-nombre
-estado
-alfabeto
-estadoInicial
-estadoAceptacion
-trancisiones
+setNombre():Void
+getNombre():String
+setEstado():Void
+getEstado():ArrayList
Gramatica
-nombre
-noTerminal
-Terminal
-NT Inicial
-Produccion
+setNombre():Void
+getNombre():String
+seNoTerminal(():Void
+getNoTerminal():ArrayList
ManejadorAFD
-ListaAFD
+newAFD()
+updateAFD()
+setEstados()
+setEstadosAceptacion()
ManejadorGramatica
-ListaGramatica
+newGrammar()
+updateGrammar()
+setNoTerminal()
+setTerminal()
ManejadorCadena
+validateCadena()
+getEstadoAceptacion()
+getCadenaFor()
Menu
+menuPrincipal
+menuAFD
+menuGramatica
+menuArchivo
+menuGuardar