

Manual Técnico Compiladores 1

10 MARZO

Facultad de Ingeniería USAC
Elder Ariel Lopez Samol



Manual Técnico

Consumo de Recursos	
RAM	72.6 MB
Almacenamiento	2.62 MB

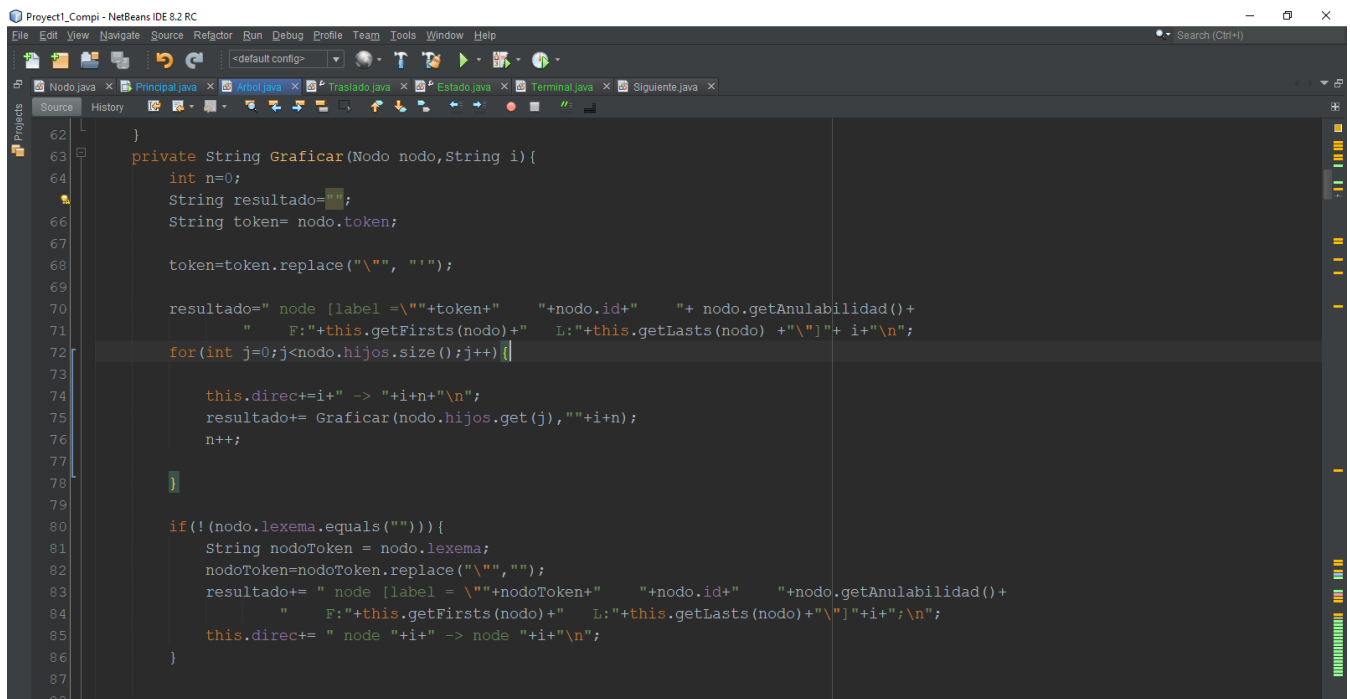
Resumen de métodos.

Nombre de metodo	Descripción
generarReportesHTML()	Con este se genera todo el string, para posteriormente generar le html de errores lexicos y sintacticos.
Graficar()	Con este metodo se genera todo el string para posterior mente genera el arbol binario de la expresion regular.
getFirsts()	Con este metodo se retorna los first de cada nodo, de cada arbol.
getLasts()	Con este metodo se retorn los last de cada nodo, de cada arbol.
GenerarDot()	Ejercuta los comandos de consola para compilar el archivo .dot, para posteriormente generar los grafos.
anul_de_Ramas()	Es el metodo que tiene mas funcionalidades, es un metodo que genera la anulabilidad de cada nodo, calculo los first, last y tambien ayuda a generar la tabla de siguientes.

searchNext()	Es un metodo capaz de identificar los siguientes de cada nodo, para posteriormente generar los follows
generarTablaSiguientes()	Es un metodo que genera el string necesario para poder compilar el .dot para la tabla de siguientes.
generarTabladeTransiciones()	Metodo que se encarga de generar la lista de transiciones, que posteriormente serviran para generar los automatas finitos deterministas.

Graficar()

Con este metodo se genera todo el string para posterior mente genera el arbol binario de la expresion regular.



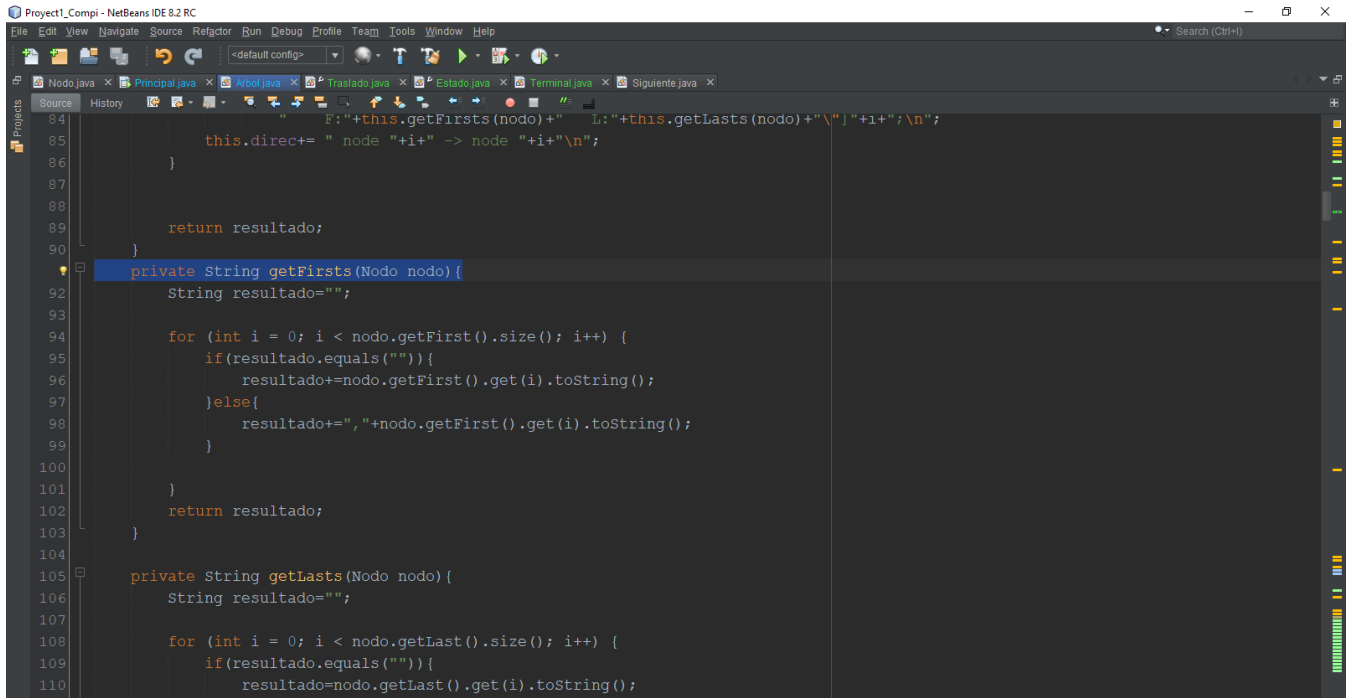
```

62     }
63     private String Graficar(Nodo nodo,String i){
64         int n=0;
65         String resultado="";
66         String token= nodo.token;
67
68         token=token.replace("\\", "");
69
70         resultado=" node [label =\\"+token+" "+nodo.id+" "+nodo.getAnulabilidad()+
71             " F:"+this.getFirsts(nodo)+" L:"+this.getLasts(nodo) +"\"] "+ i+"\n";
72         for(int j=0;j<nodo.hijos.size();j++){
73
74             this.direc+=i+" -> "+i+n+"\n";
75             resultado+= Graficar(nodo.hijos.get(j)," "+i+n);
76             n++;
77         }
78     }
79
80     if(!(nodo.lexema.equals(""))){
81         String nodoToken = nodo.lexema;
82         nodoToken=nodoToken.replace("\\", "");
83         resultado+= " node [label =\\"+nodoToken+" "+nodo.id+" "+nodo.getAnulabilidad()+
84             " F:"+this.getFirsts(nodo)+" L:"+this.getLasts(nodo) +"\"] "+i+"\n";
85         this.direc+= " node "+i+" -> node "+i+"\n";
86     }
87
88

```

getFirsts()

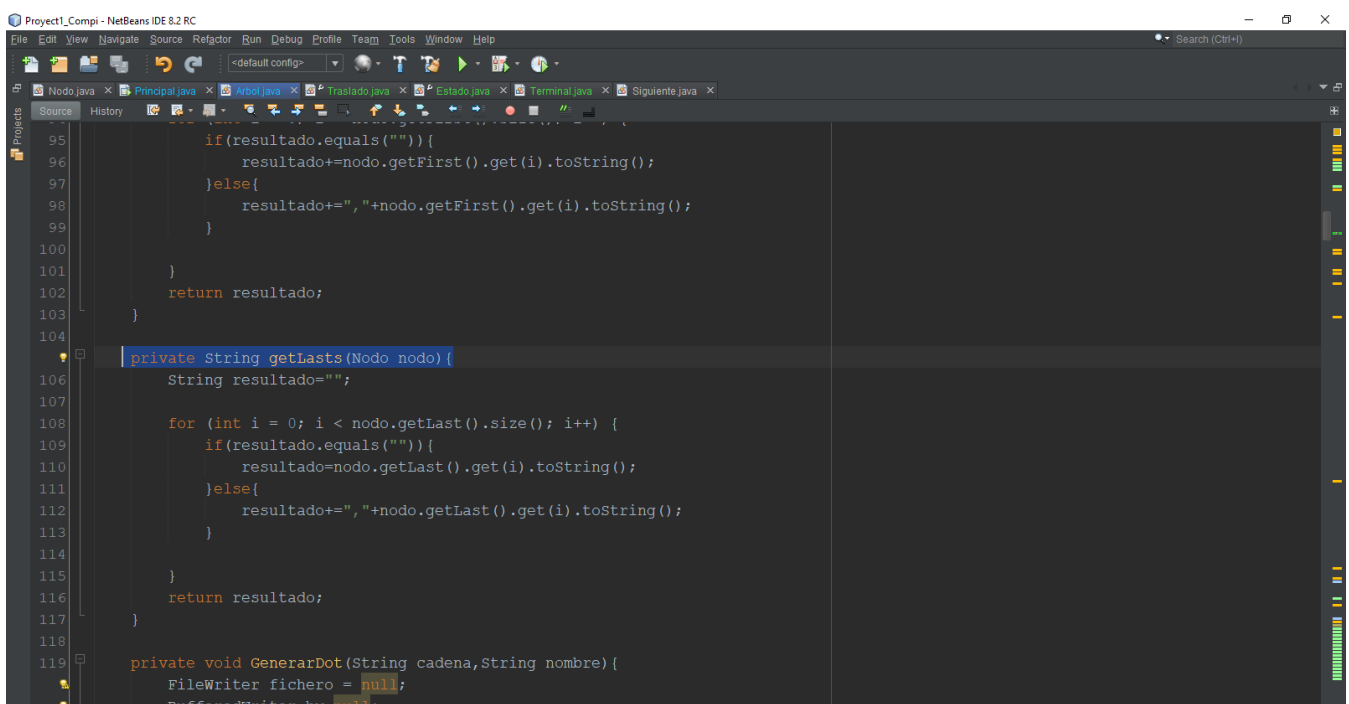
Con este metodo se genera todo el string para posterior mente genera el arbol binario de la expresion regular.



```
84      " F:"+this.getFirsts(nodo)+" L:"+this.getLasts(nodo)+"\n"] "+i+"\n";
85      this.direc+= " node "+i+" -> node "+i+"\n";
86  }
87
88
89  return resultado;
90  }
91
92  private String getFirsts(Nodo nodo){
93      String resultado="";
94
95      for (int i = 0; i < nodo.getFirst().size(); i++) {
96          if(resultado.equals("")){
97              resultado+=nodo.getFirst().get(i).toString();
98          }else{
99              resultado+=","+nodo.getFirst().get(i).toString();
100          }
101      }
102      return resultado;
103  }
104
105  private String getLasts(Nodo nodo){
106      String resultado="";
107
108      for (int i = 0; i < nodo.getLast().size(); i++) {
109          if(resultado.equals("")){
110              resultado+=nodo.getLast().get(i).toString();
111          }else{
112              resultado+=","+nodo.getLast().get(i).toString();
113          }
114      }
115      return resultado;
116  }
117
118  private void GenerarDot(String cadena,String nombre){
119      FileWriter fichero = null;
```

getLasts()

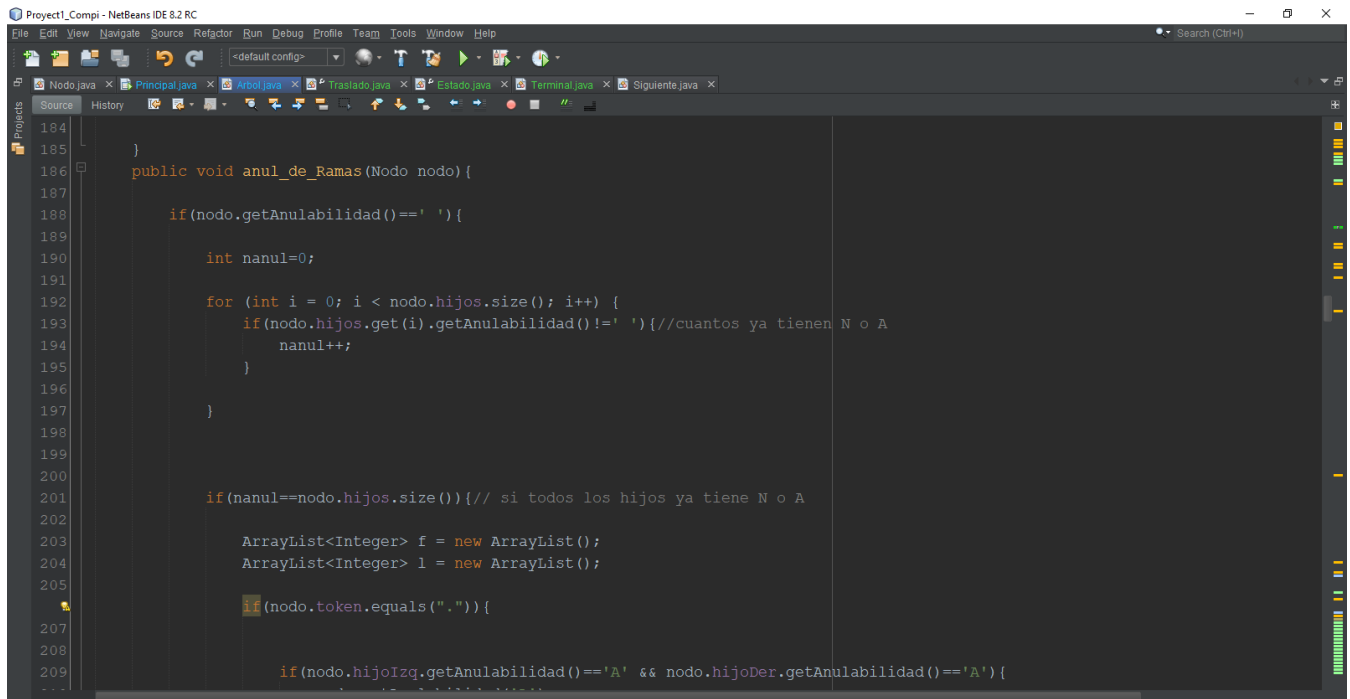
Con este metodo se retorn los last de cada nodo, de cada arbol.



```
95      if(resultado.equals("")){
96          resultado+=nodo.getFirst().get(i).toString();
97      }else{
98          resultado+=","+nodo.getFirst().get(i).toString();
99      }
100  }
101  }
102  return resultado;
103  }
104
105  private String getLasts(Nodo nodo){
106      String resultado="";
107
108      for (int i = 0; i < nodo.getLast().size(); i++) {
109          if(resultado.equals("")){
110              resultado+=nodo.getLast().get(i).toString();
111          }else{
112              resultado+=","+nodo.getLast().get(i).toString();
113          }
114      }
115      return resultado;
116  }
117
118  private void GenerarDot(String cadena,String nombre){
119      FileWriter fichero = null;
```

anul_de_Ramas()

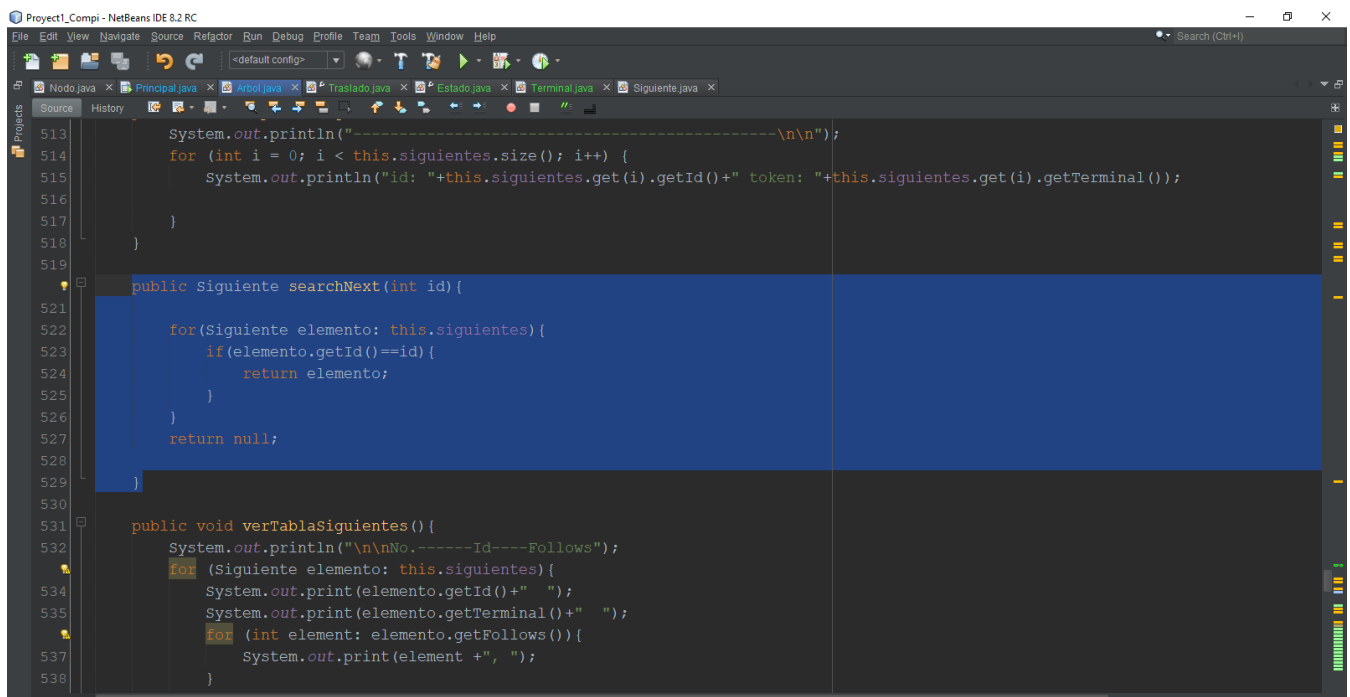
Es el metodo que tiene mas funcionalidades, es un metodo que genera la anulabilidad de cada nodo, calculo los first, last y tambien ayuda a generar la tabla de siguientes.



```
184
185
186 public void anul_de_Ramas(Nodo nodo) {
187
188     if(nodo.getAnulabilidad()==' '){
189
190         int nanul=0;
191
192         for (int i = 0; i < nodo.hijos.size(); i++) {
193             if(nodo.hijos.get(i).getAnulabilidad()!=' '){//cuantos ya tienen N o A
194                 nanul++;
195             }
196         }
197
198
199
200
201         if(nanul==nodo.hijos.size()){// si todos los hijos ya tiene N o A
202
203             ArrayList<Integer> f = new ArrayList();
204             ArrayList<Integer> l = new ArrayList();
205
206             if(nodo.token.equals(".")){
207
208
209                 if(nodo.hijoIzg.getAnulabilidad()=='A' && nodo.hijoDer.getAnulabilidad()=='A'){
```

searchNext()

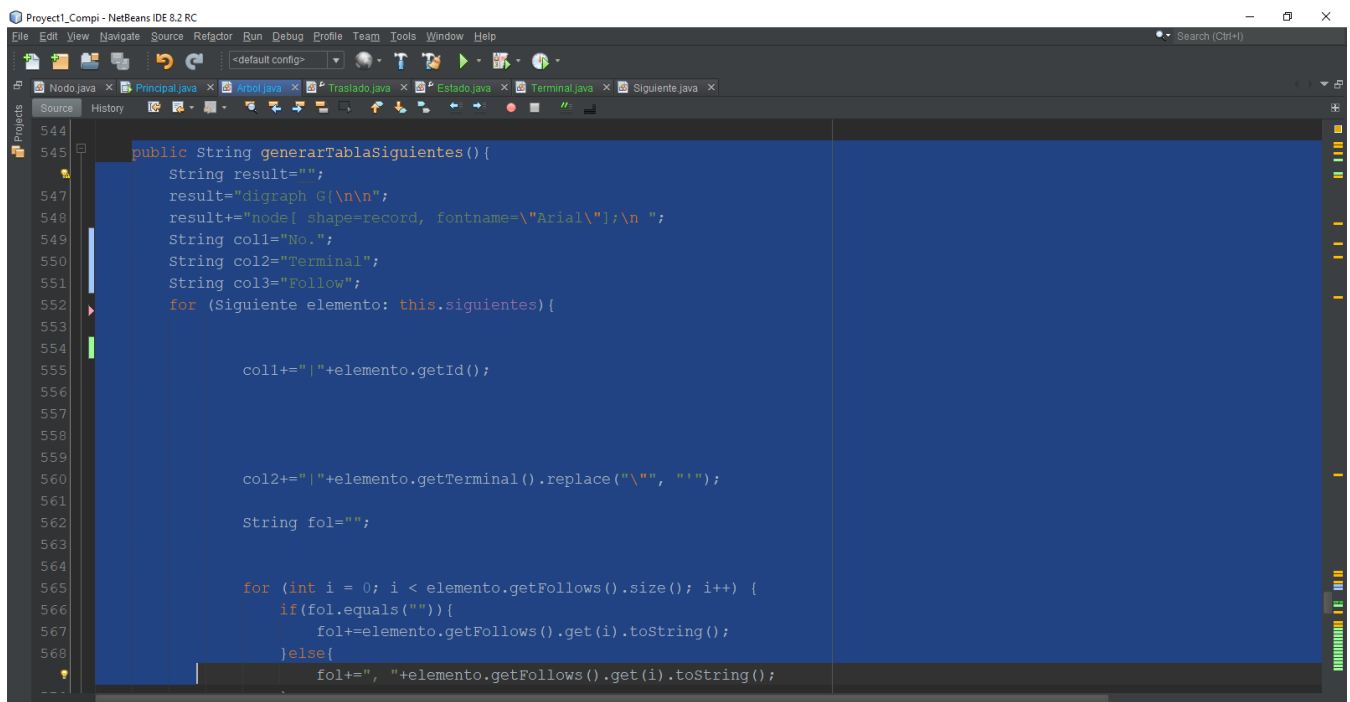
Es un metodo capaz de identificar los siguientes de cada nodo, para posteriormente generar los follows



```
513     System.out.println("-----\n\n");
514     for (int i = 0; i < this.siguietes.size(); i++) {
515         System.out.println("id: "+this.siguietes.get(i).getId()+" token: "+this.siguietes.get(i).getTerminal());
516     }
517
518
519
520
521 public Siguiente searchNext(int id){
522
523     for(Siguiente elemento: this.siguietes){
524         if(elemento.getId()==id){
525             return elemento;
526         }
527     }
528     return null;
529 }
530
531
532 public void verTablaSiguietes(){
533     System.out.println("\n\nNo.-----Id----Follows");
534     for (Siguiente elemento: this.siguietes){
535         System.out.print(elemento.getId()+" ");
536         System.out.print(elemento.getTerminal()+" ");
537         for (int element: elemento.getFollows()){
538             System.out.print(element +", ");
539         }
540     }
541 }
```

generarTablaSiguietes()

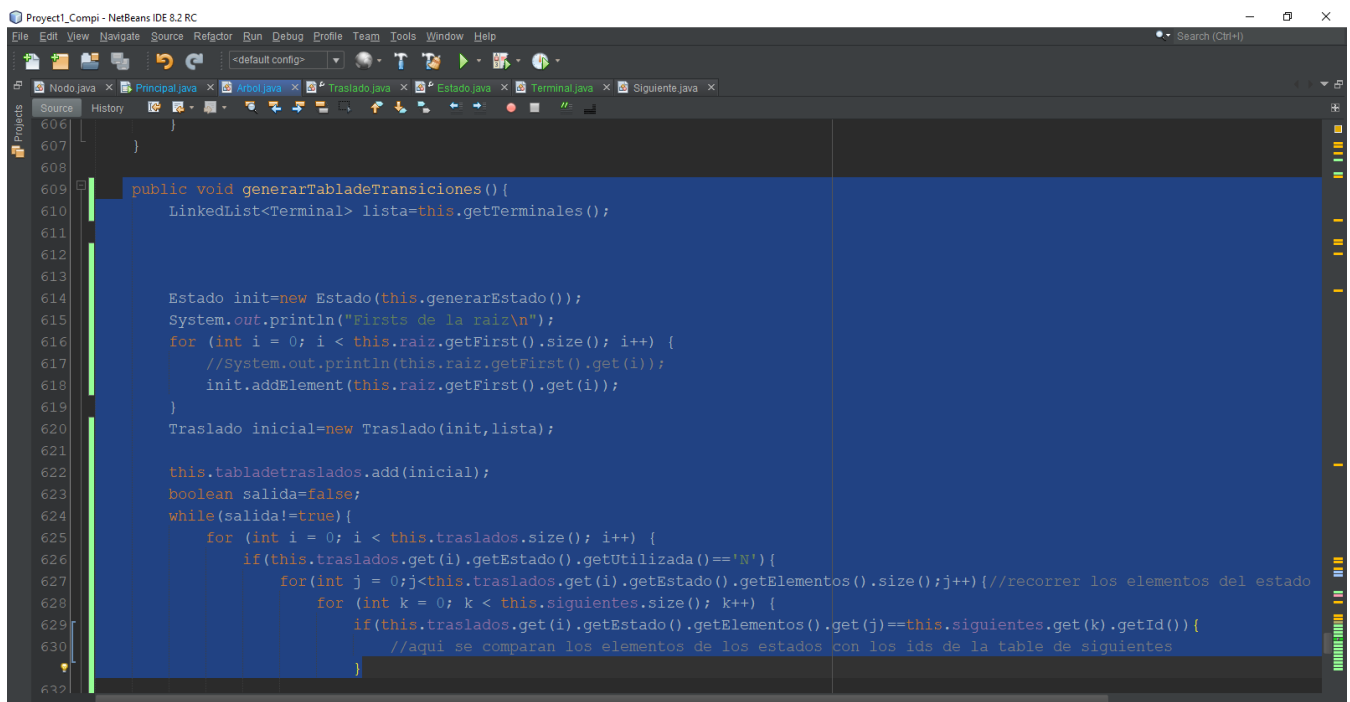
Es un metodo que genera el string necesario para poder compilar el .dot para la tabla de siguientes.



```
544
545 public String generarTablaSiguietes(){
546     String result="";
547     result="digraph G{\n\n";
548     result+="node[ shape=record, fontname=\"Arial\"]; \n ";
549     String col1="No.";
550     String col2="Terminal";
551     String col3="Follow";
552     for (Siguiente elemento: this.siguietes){
553
554
555         coll+="|"+elemento.getId();
556
557
558
559
560         col2+="|"+elemento.getTerminal().replace("\\", "");
561
562         String fol="";
563
564
565         for (int i = 0; i < elemento.getFollows().size(); i++) {
566             if(fol.equals("")){
567                 fol+=elemento.getFollows().get(i).toString();
568             }else{
569                 fol+=", "+elemento.getFollows().get(i).toString();
570             }
571         }
572     }
573 }
```

generarTabladeTransiciones()

Metodo que se encarga de generar la lista de transiciones, que posteriormente serviran para generar los automatas finitos deterministas.



```
606
607 }
608
609 public void generarTabladeTransiciones() {
610     LinkedList<Terminal> lista=this.getTerminales();
611
612
613
614     Estado init=new Estado(this.generarEstado());
615     System.out.println("Firsts de la raiz\n");
616     for (int i = 0; i < this.raiz.getFirst().size(); i++) {
617         //System.out.println(this.raiz.getFirst().get(i));
618         init.addElement(this.raiz.getFirst().get(i));
619     }
620     Traslado inicial=new Traslado(init,lista);
621
622     this.tabladetraslados.add(inicial);
623     boolean salida=false;
624     while(salida!=true){
625         for (int i = 0; i < this.traslados.size(); i++) {
626             if(this.traslados.get(i).getEstado().getUtilizada()=='N'){
627                 for(int j = 0;j<this.traslados.get(i).getEstado().getElementos().size();j++){//recorrer los elementos del estado
628                     for (int k = 0; k < this.siguietes.size(); k++) {
629                         if(this.traslados.get(i).getEstado().getElementos().get(j)==this.siguietes.get(k).getId()){
630                             //aqui se comparan los elementos de los estados con los ids de la tabla de siguientes
631                         }
632                     }
633                 }
634             }
635         }
636     }
637 }
```