Resumen del código por funcionalidad

Leer un archivo con las especificaciones de un autómata

```
//ArrayList de arreglo de strings, para ir guardando lo que leemos, pues tendremos qu
e releerlo.
ArrayList<Object[]> alestados = new ArrayList();

//HashMap (diccionario, hashtable, whatever...) de String a objeto Estad, para poder
//Acceder directo al objeto estado correspondiente a la letra de su id.s
HashMap<String, Estado> mapestados = new HashMap();

System.out.println("Leyendo datos:");

try{

    File file = new File ("ruta/de/archivo/automata.txt");

  Scanner fileScanner = new Scanner(file); //Scanner para el archivo
  Scanner scanLine; //Scanner para las lineas individuales
  Scanner scanStates; //Scanner para los estados

  //mientras haya lineas en el archivo
  while(fileScanner.hasNext()){
    scanLine = new Scanner(fileScanner.nextLine());
    scanLine.useDelimiter("\t");

    scanStates = new Scanner(scanLine.next());
    scanStates.useDelimiter(",");



    ArrayList<String> estados0 = new ArrayList(); //Son los múltiples estados que pue
de tener (A,B) 1 a n
    ArrayList<String> estados1 = new ArrayList();

    //leer estados0
    while(scanStates.hasNext()){
      estados0.add(scanStates.next());
    }
```

```
    //leer el tipo de estados0
    String transicion0 = scanLine.next();

    scanStates = new Scanner(scanLine.next()); //Reinicializo en ScanStates para leer
 los siguientes estados que igual pueden ser multiples.
    scanStates.useDelimiter(",");

    //leer estados1
    while(scanStates.hasNext()){
      estados1.add(scanStates.next());
    }

    String transicion1 = scanLine.next();

    System.out.println(estados0 + " " + transicion0 + " " + estados1 + " " + transici
on1);

    //Arreglo de tipo Object para poder pasarle objetos de diferente tipo, hay que de
sempaquetarlo bien, con cuidado.
    Object[] arrEstado = {estados0, transicion0, estados1, transicion1};
    alestados.add(arrEstado);

  }

}
catch(Exception e){
  //Rete mal manejo de excepcion pero que flojera hacerlo bien
  System.out.println("Algun tipo de exception estilo ioException sucedio...: " + e.ge
tMessage());

}

//imprimpiendo los estados para checar
for (int i = 0; i < alestados.size() ; i++) {
  for (int j = 0; j < 4 ; j++) {
    System.out.print(alestados.get(i)[j]);
  }
  System.out.println();
}

//Ir creando objetos estado con los estados a los que pueden transicionar.
for (int i = 0;i < alestados.size() ;i++ ) {
  Estado e = new Estado(alfabeto[i]);
  e.x0destino = (ArrayList)alestados.get(i)[0];
```

```
  e.x1destino = (ArrayList)alestados.get(i)[2];

  mapestados.put(e.id,e); //Agregar el estado al HashMap
}

/*Ir checando los estados que son de aceptacion.
  En caso de que uno sea, con el HashMap lo encontramos
  y vamos y le decimos que SI es de aceptacion
*/
for (int i = 0;i < alestados.size();i++) {
  if(alestados.get(i)[1].equals("1")){
    //Iterar para cada estado que encuentre en cada lista
    for (int j = 0; j < ((ArrayList)alestados.get(i)[0]).size(); j++) {
      mapestados.get(((ArrayList)alestados.get(i)[0]).get(j)).edoAceptacion = true;
    }
  }

  if(alestados.get(i)[3].equals("1")){
    //Iterar para cada estado que encuentre en cada lista
    for (int j = 0; j < ((ArrayList)alestados.get(i)[2]).size(); j++) {
      mapestados.get(((ArrayList)alestados.get(i)[2]).get(j)).edoAceptacion = true;
    }
  }

}

//checando que todo bien....
for (int i = 0; i < alestados.size() ;i++ ) {
  System.out.println(mapestados.get(alfabeto[i]).toString());
}

//Crear objeto autómata
Automata a = new Automata(mapestados);
```

Determinar si el autómata es no determinista

```
public boolean esDeterminista(){
  boolean res = true;

  Estado[] edos = estados.values().toArray(new Estado[estados.size()]);

  for (int i = 0; i< edos.length ; i++) {
    if(edos[i].x0destino.size() > 1 || edos[i].x0destino.get(0).equals("-")  || edos[
i].x1destino.size() > 1  || edos[i].x1destino.get(0).equals("-")){
      return false;
    }
  }
  return res;
}
```

Convertir a autómata determinista

Creación de estados compuestos (metaestados)

En esta sección se identifican los estados que adicionales para poder volver el autómata determinista. Si es necesario, se agrega el estado vacío.

```
//volver al automata determinista si es que no lo es
public HashMap<String, Estado> detMetaestados(){


  Estado[] edos = estados.values().toArray(new Estado[estados.size()]);

  HashMap<String, Estado> metaEstadosHM = new HashMap();

  //Agregamos el primer estado, pues siempre es el de entrada. Agregamos si es o no d
e aceptacion (en este proyecto no se puede)
  metaEstadosHM.put(edos[0].id,edos[0]);

  boolean flagVacio = false; //Flag que se debe poner en true para saber si se debe a
gregar el estado vacio.

  //vemos los metaestados de transicion de cada uno de los estados
  for (int i = 0;i < edos.length; i++) {

    //variables para los metaestados
```

```
    String x0 = "";
    String x0Aceptacion = "0";
    String x1 = "";
    String x1Aceptacion = "0";

    for (int j = 0; j < edos[i].x0destino.size() ; j++) {
      x0 += edos[i].x0destino.get(j);

      //Si alguno de los estados agregados es de aceptacion, todo el metaestado es de
 aceptacion tambien
      if(estados.get(edos[i].x0destino.get(j)).edoAceptacion){
        x0Aceptacion = "1";
      }

    }

    for (int j = 0; j < edos[i].x1destino.size() ; j++) {
      x1 += edos[i].x1destino.get(j);

      if(estados.get(edos[i].x1destino.get(j)).edoAceptacion){
        x1Aceptacion = "1";
      }
    }

    //Si alguna transicion estaba vacia, activar la flag del estado vacio
    if(x0.isEmpty() || x1.isEmpty()){
      flagVacio = true;
    }

    //Hacer un ordenamiento lexicografico del metaestado (e.g. ACB -> ABC) para asegu
rar no repeticiones
    x0 = ordenarLex(x0);
    x1 = ordenarLex(x1);


    //creo que no se necesita esto
    boolean repetidos = false;
    if(x0.equals(x1)){
      repetidos = true;
    }

    //Para el primer estado, settear sus metaestados de transicion
    if(i == 0){
      metaEstadosHM.get(edos[i].id).setEstadosDet(x0,x1);
    }
```

```
    if(!metaEstadosHM.containsValue(x0) && !x0.isEmpty()){
      Estado e = new Estado(x0);
      if(x0Aceptacion.equals("1")){
        e.edoAceptacion = true;
      }
      metaEstadosHM.put(x0, e);
    }

    if(!metaEstadosHM.containsValue(x1) && !x1.isEmpty()){
      Estado e = new Estado(x1);
      if(x1Aceptacion.equals("1")){
        e.edoAceptacion = true;
      }
      metaEstadosHM.put(x1, e);
    }


  }//end for

  //Agregar el estado vacio solamente si es necesario
  if(flagVacio){
    if(!metaEstadosHM.containsValue("-")){
      Estado e = new Estado("-");
      metaEstadosHM.put("-", e);
    }
  }

  return metaEstadosHM;

}
```