Partición

Clase para almacenar subconjuntos de estados que se pueden asociar por no ser distinguibles si se ve el autómata como una caja negra.

```
package funmatnb;

import java.util.ArrayList;

public class Particion {
    public Subconjunto[] subconjuntos; //Como Particion, yo tengo subconjuntos. Cada
subconjunto sabe cual es su id.
    public String idParticion;
    private int cantidadDeSubconjuntosActual;

    public boolean edoAceptacion;
    public ArrayList<String> x0destino;
    public ArrayList<String> x1destino;
    //Las particiones pueden nombrar a sus subconjuntos, esto facilita la identificac
ion de un estado dentro de aquellos
    //Y la creacion de nuevas Particiones.

    public Particion(String idParticion, int estados){
        //Como dicho en el escrito, el maximo numero de particiones de hecho es la ca
ntidad de estados que tiene este
        subconjuntos = new Subconjunto[estados];

       // Voy a poder nombrar el subconjunto con el array ya que el subconjunto 2 cor
responde al subconjunto de  subconjuntos[2]
        this.idParticion = idParticion; //Es publico por el uso que le implementamos
luego, para optimizar lo volvemos privado.
        cantidadDeSubconjuntosActual = 1;
        // Le creo un subconjunto vacio?

    }

    public boolean creaSubconjunto(String idSub){
        Subconjunto s = new Subconjunto(idSub);
        boolean res = contieneSubconjunto(s);
        if(!res){
            subconjuntos[cantidadDeSubconjuntosActual] = s;
```

```
            cantidadDeSubconjuntosActual ++;

        }
        return res;
    }


    public boolean contieneSubconjunto (Subconjunto a){
        boolean res = false;
        int i =0;
        while (i< cantidadDeSubconjuntosActual && !res){
            if (!subconjuntos[i].equals(a)){
                res = true;
            }else{
            i++;
            }
          }

        return res; //Regresa true si la particion ya contiene a ese subconjunto.
      }
}
```

Subconjunto

Clase para agrupar estados de acuerdo a los subconjuntos de la partición anterior a los que llevan sus transiciones, estos subconjuntos determinan el id de cada subconjunto en la siguiente partición.

```
package funmatnb;

import java.util.ArrayList;

public class Subconjunto {
    //Creamos una clase de subconjuntos que sabe a cuales estados pertenecen. Estos s
ubconjuntos
    //Son parte de las particiones.
    public String id;
    public ArrayList<String> estados;

    public Subconjunto(String id){
        this.id = id;
        estados = new ArrayList();

    }
    public void agregaEstadoAlSubconjunto(Estado est){
        if(!contieneEstado(est)){
            estados.add(est.id);
        }
    }
    public boolean contieneEstado(Estado a){
        /*Revisa en su grupo de estados si esta el actual, para no repetir. */
        boolean res = true;
        if (!estados.contains(a.id)){
            res = false;
        }
        return res;
    }
    public boolean equals (Subconjunto a){
        return (this.id == a.id);
    }
}
```

Estado

Clase para caracterizar estados, sus transiciones y si es o no estado de aceptación.

```
package funmatnb;

import java.util.ArrayList;

public class Estado{
```

```
public String id;
public boolean edoAceptacion;
public ArrayList<String> x0destino;
public ArrayList<String> x1destino;

//Constructor mas simple (ideologia de ir armando el estado poco a poco)
public Estado(String id){
  this.id = id;
  edoAceptacion = false;
  x0destino = new ArrayList();
  x1destino = new ArrayList();
}

//Constructor para utilizar cuando estamos armando estados deterministas (i.e. destin
os de 1 solo estado)
public Estado(String id, String x0, String x1){
  this.id = id;
  edoAceptacion = false;
  x0destino = new ArrayList();
  x1destino = new ArrayList();

  x0destino.add(x0);
  x1destino.add(x1);
}


//chance no se acaba usando este constructor. Lo dejo porque pueque
public Estado(String id,boolean edoAceptacion, ArrayList<String> x0destino, ArrayList
<String> x1destino){
  this.id = id;
  this.edoAceptacion = edoAceptacion;
  this.x0destino = x0destino;
  this.x1destino = x1destino;
}

public String toString(){
  return "id: " + id + ", edoAceptacion: " + edoAceptacion + ", x0destino: " + x0dest
ino + ", x1destino: " + x1destino;
}

//Metodo para agregar solo un estado a los arrayLists de estados a los que puede hace
r transicion
public void setEstadosDet(String x0, String x1){
```

```
  //borron de los estados anteriores, solamente queremos los agregados
  x0destino = new ArrayList();
  x1destino = new ArrayList();
  x0destino.add(x0);
  x1destino.add(x1);
}


}
```