

Reporte

Nombre: Christian Arturo Morales Garcia

U. A. Teoria Computacional

Grupo: 2CV1

Prof. Juarez Martinez Genaro

Problema de los Planetas

El problema nos habla de un numero determinado de 3 especies y un planeta, los cuales siguen unas sencillas reglas:

1. Se inicia con un numero total de individuos este total es formado por el numero de individuos de cada especie.
2. Si muere un individuo de 2 especies diferentes, nacen 2 de la especie que no murio.

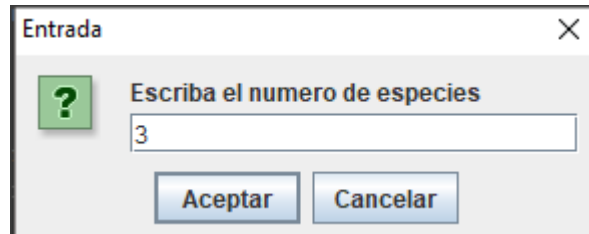
Con estas reglas es posible representar el comportamiento de las especies dentro del planeta. Creando automatas los cuales tienen estados siendo estos estados el numero de cada uno de las especies dentro del planeta, este comportamiento se obtiene obedeciendo a las reglas antes mencionadas. los automatas generados tienen 2 posibilidades, llegar a un punto donde ya no es posible continuar aplicando las reglas o seguir de manera indefinida. Con el desarrollo de un programa se pretende llegar a un patron que nos indique cuando un numero total de individuos llegara a un estado donde ya no se puedan aplicar las reglas.

Solucion propuesta

El problema se soluciono obteniendo todas las combinaciones posibles entre las 3 especies, teniendo como valor minimo el 0 y valor maximo el $n-1$ siendo n el numero total de individuos iniciales. Una vez teniendo todas las combinaciones posibles lo que quedaba por hacer era relacionar cada uno de los estados con los datos con los que tenia una transicion, la transicion de cada uno de los estados se obtuvo aplicando las reglas del problema y conforme iba apareciendo cada una de las transiciones se iba agrupando cada conjunto para formar un automata el cual podia tener o no tener dentro de el un estado final el cual se reconocia por tener a n como numero total de una de las especies. Finalmente se encontro que cuando n era un numero multiplo de 3 sus automatas fallaban.

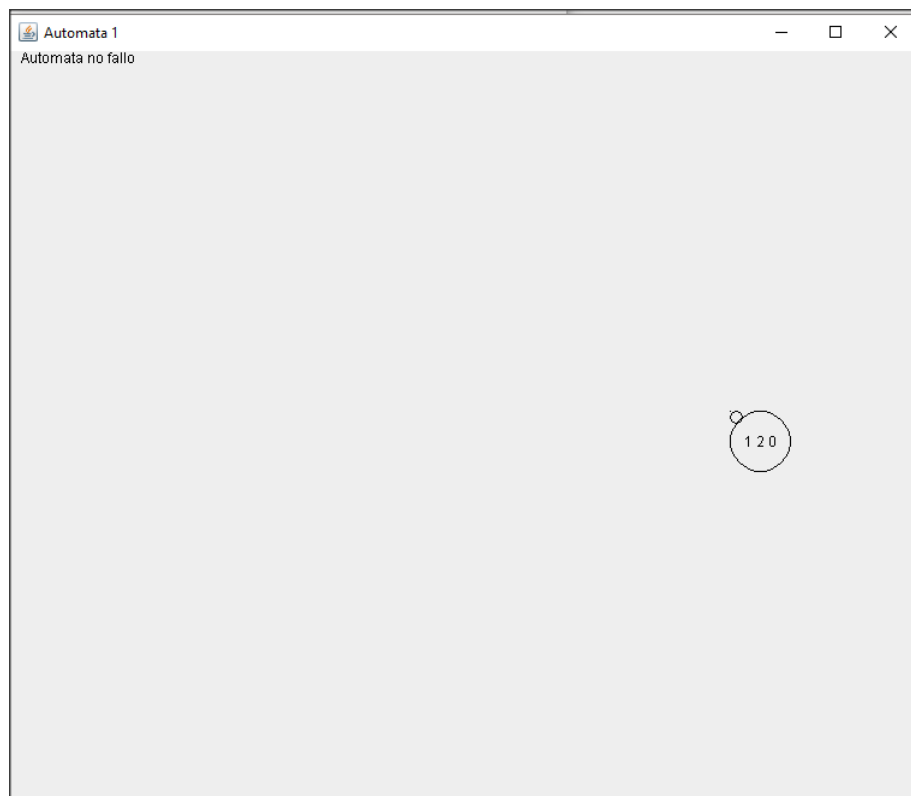
Pruebas del programa

Primero comenzamos con un numero n de 3 especies, al ser n un multiplo de 3 el resultado sera un automata que falle y otro que no.

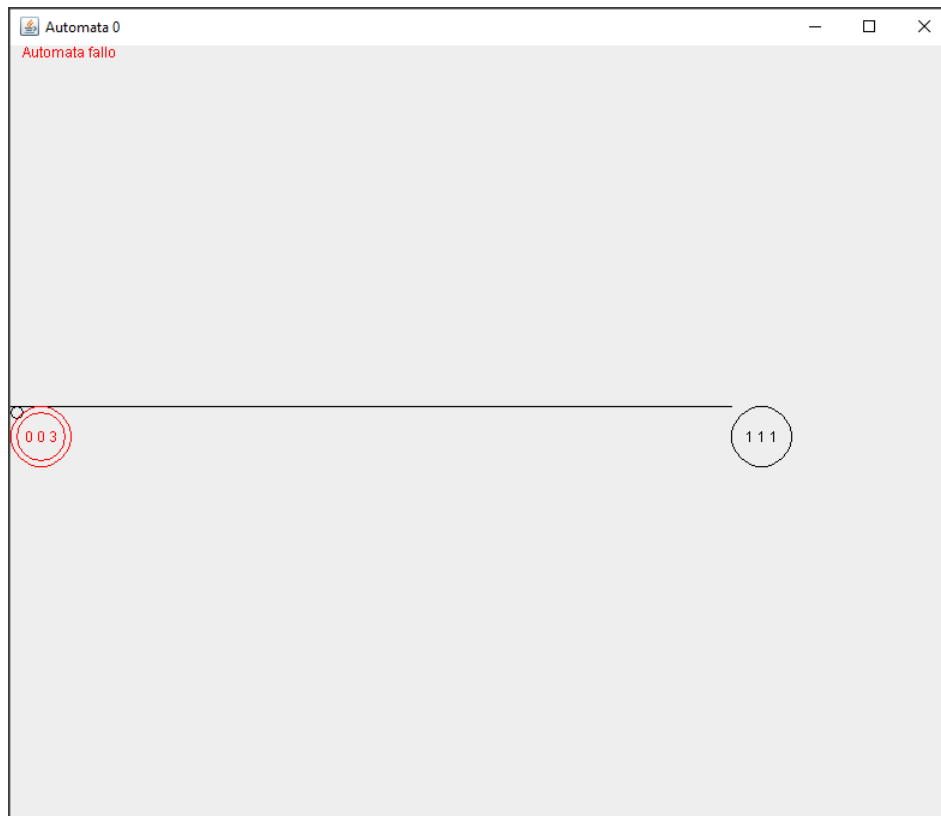


A screenshot of a Windows-style dialog box titled "Entrada". It contains a green square icon with a white question mark. To the right of the icon is the text "Escriba el numero de especies". Below this text is a text input field containing the number "3". At the bottom of the dialog are two buttons: "Aceptar" and "Cancelar".

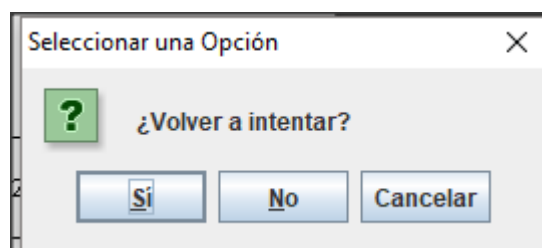
Primero nos pregunta el numero n .



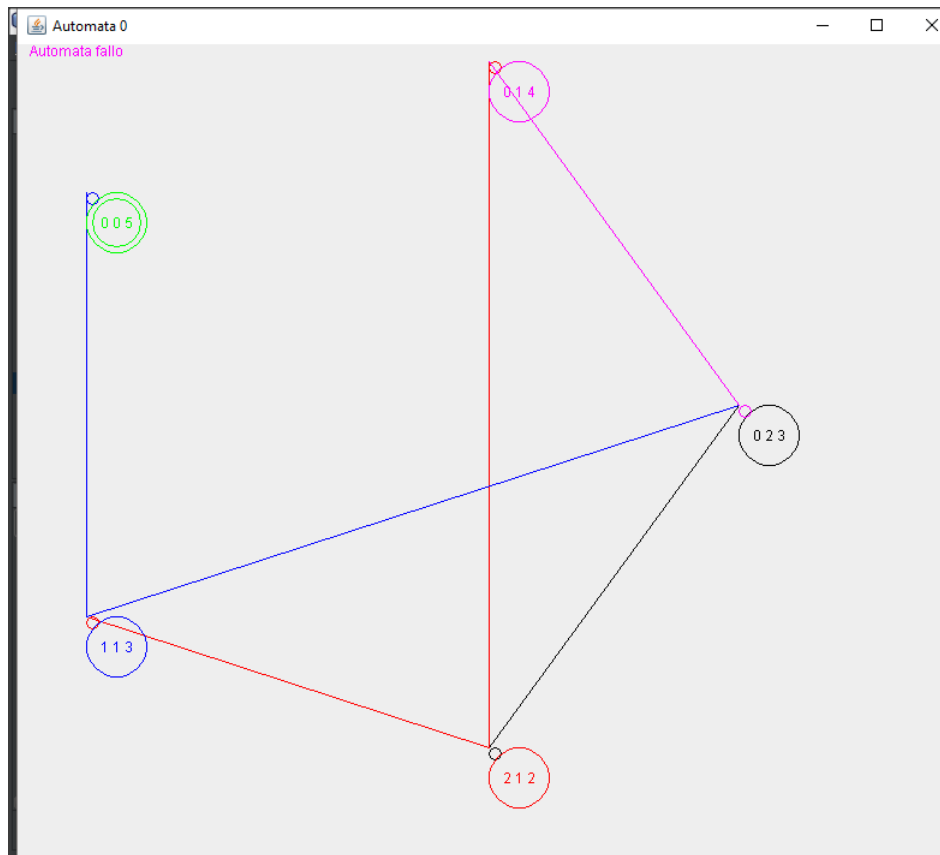
Se observa que el autoamta no falla y que esta ciclado.



Se observa que el autoamta falla y llega a un estado final.

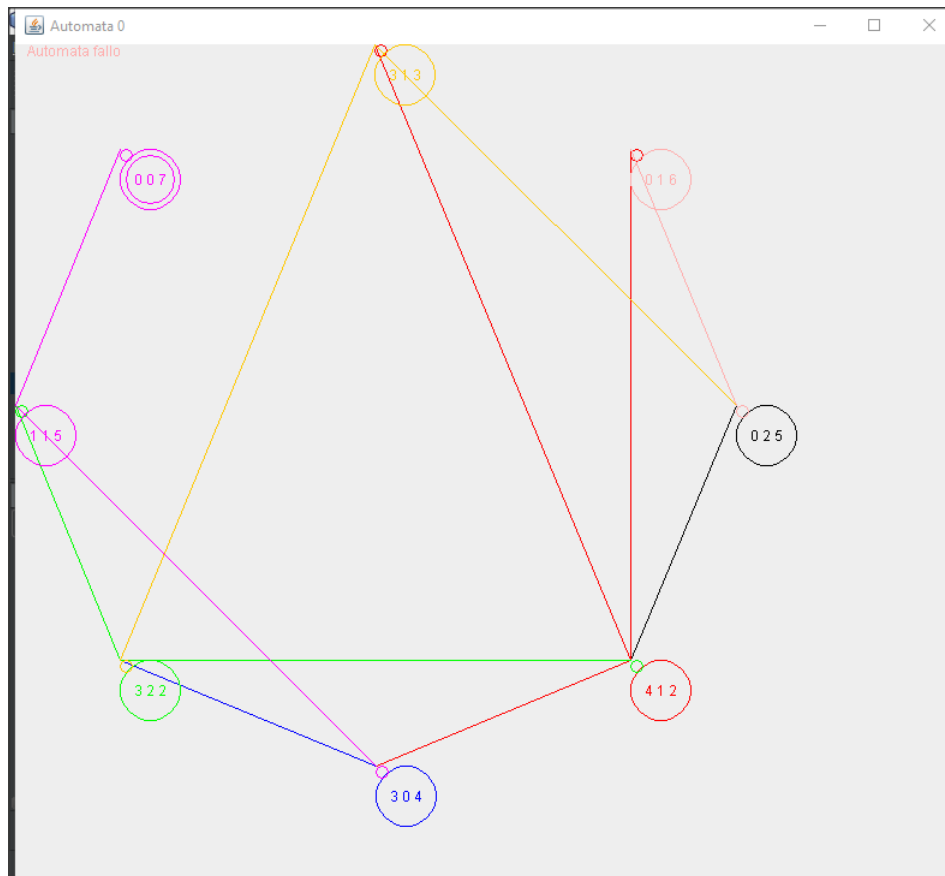


Nos pregunta si queremos volver a intentar.



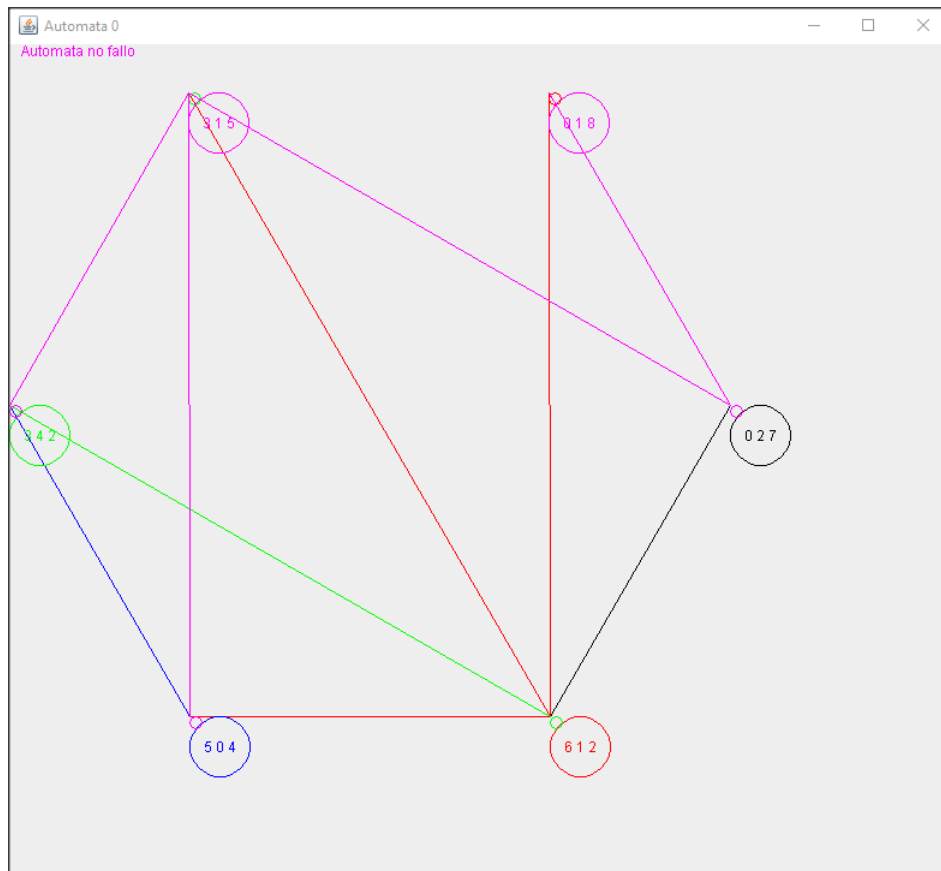
Observamos que solo nos da un automata el cual falla.

Ahora introducimos el numero 7 y observamos.

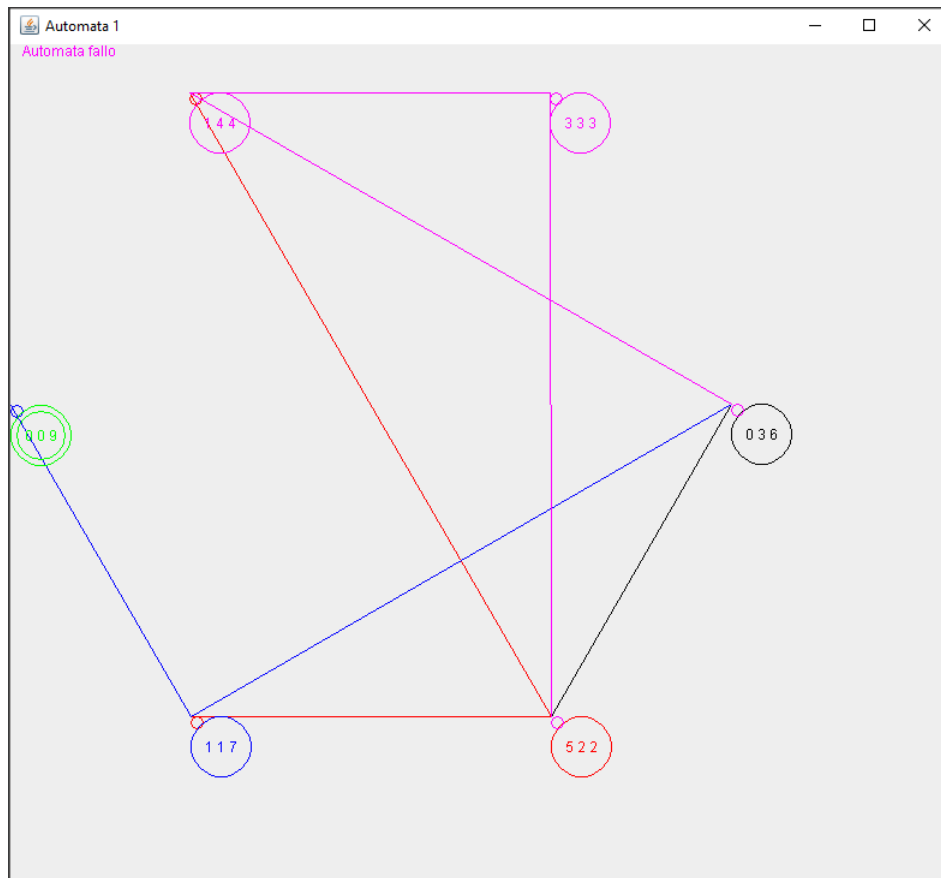


Observamos que solo nos da un automata el cual falla.

Ahora introducimos el numero 9 como vemos es multiplo de 3 por lo tanto nos dara 2 automatas uno que falla y otro que no. Veamos.

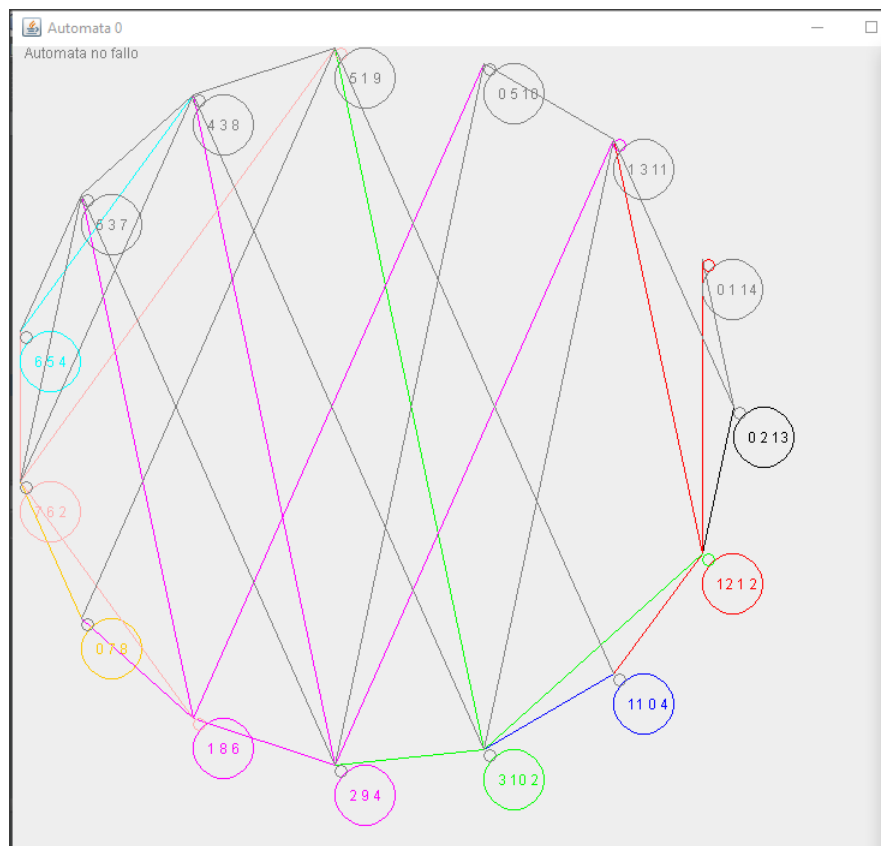


Observamos que nos da un automata el cual no falla.

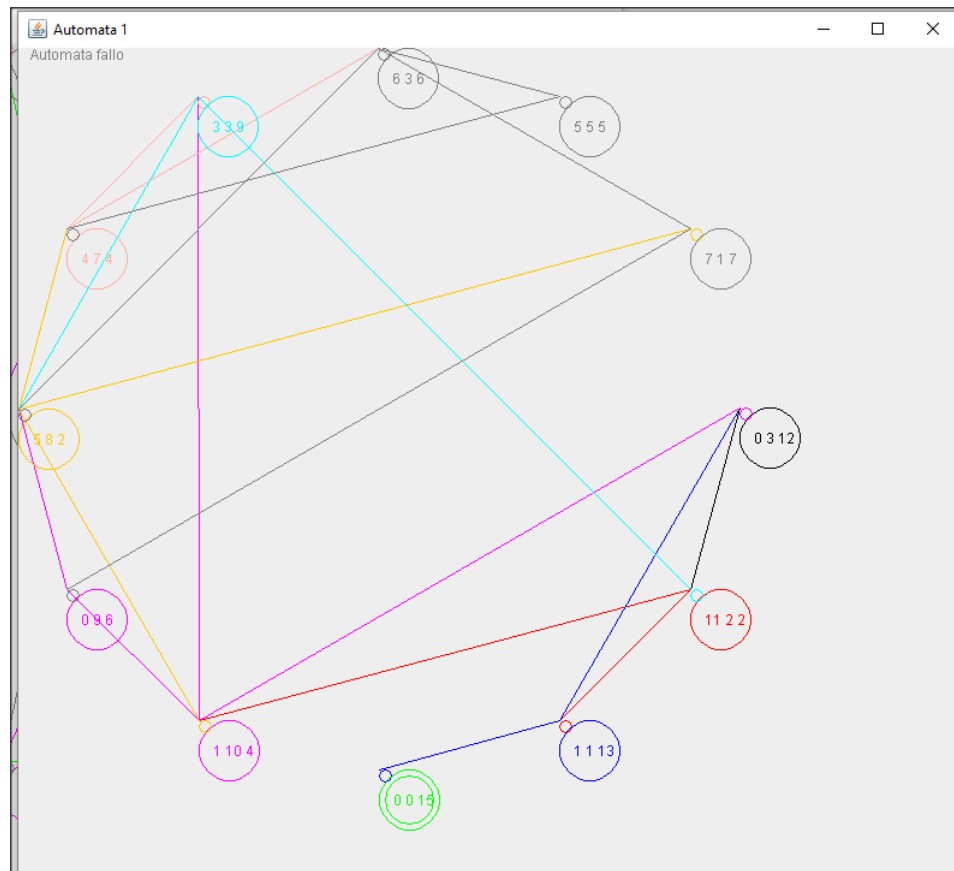


Observamos que nos da un automata el cual falla.

Ahora introducimos el numero 15 como vemos es multiplo de 3 por lo tanto nos dara 2 automatas uno que falla y otro que no. Veamos.



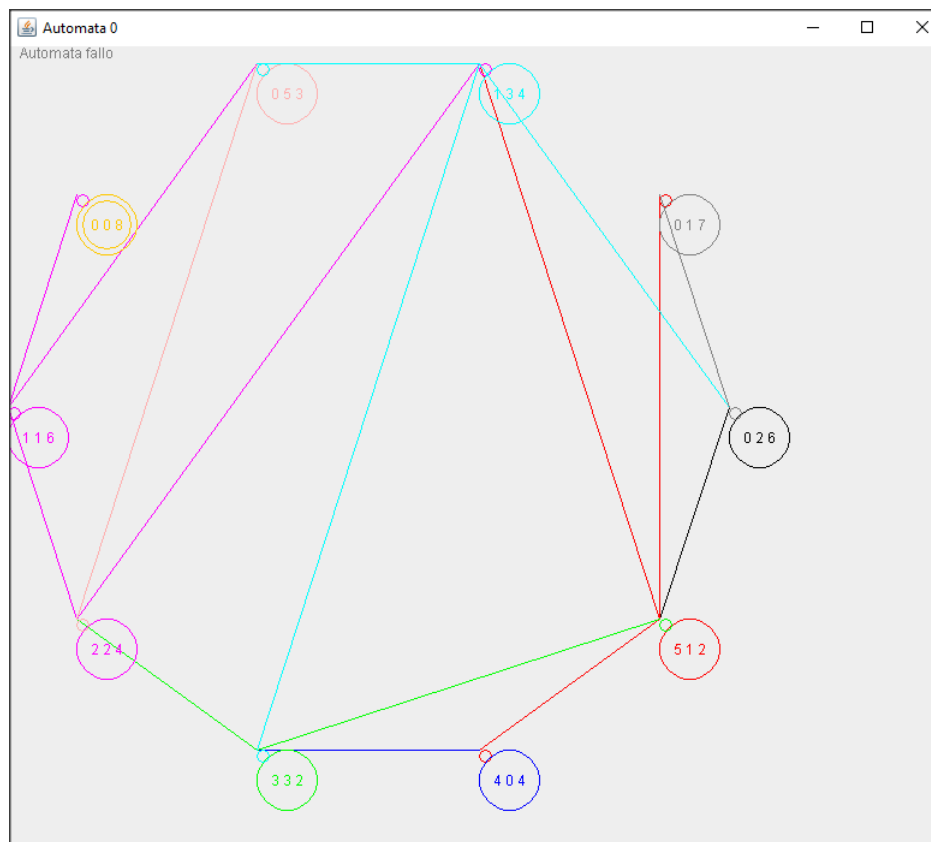
Observamos que nos da un automata el cual no falla.



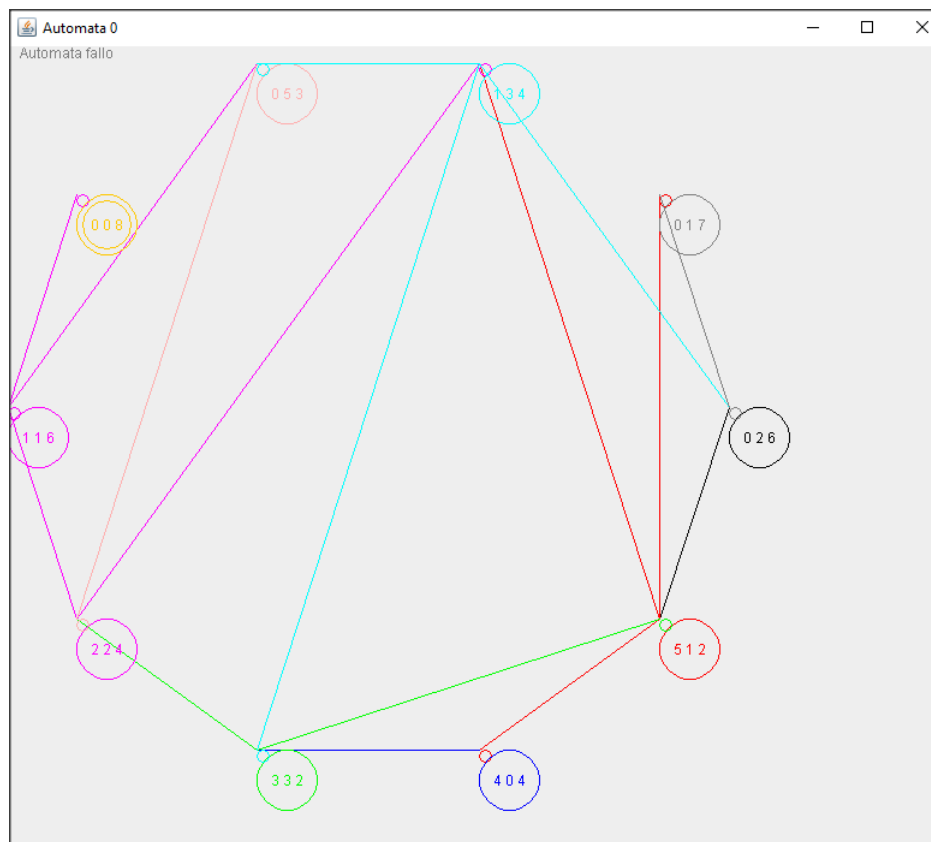
Observamos que nos da un automata el cual falla.

Como vemos despues de unas cuentas pruebas somos capaces de ver el patron, por lo que se concluye que cuando se introduce un numero total de especies que sea multiplo del 3 se obtienen 2 automatas uno que falla y otro que no.

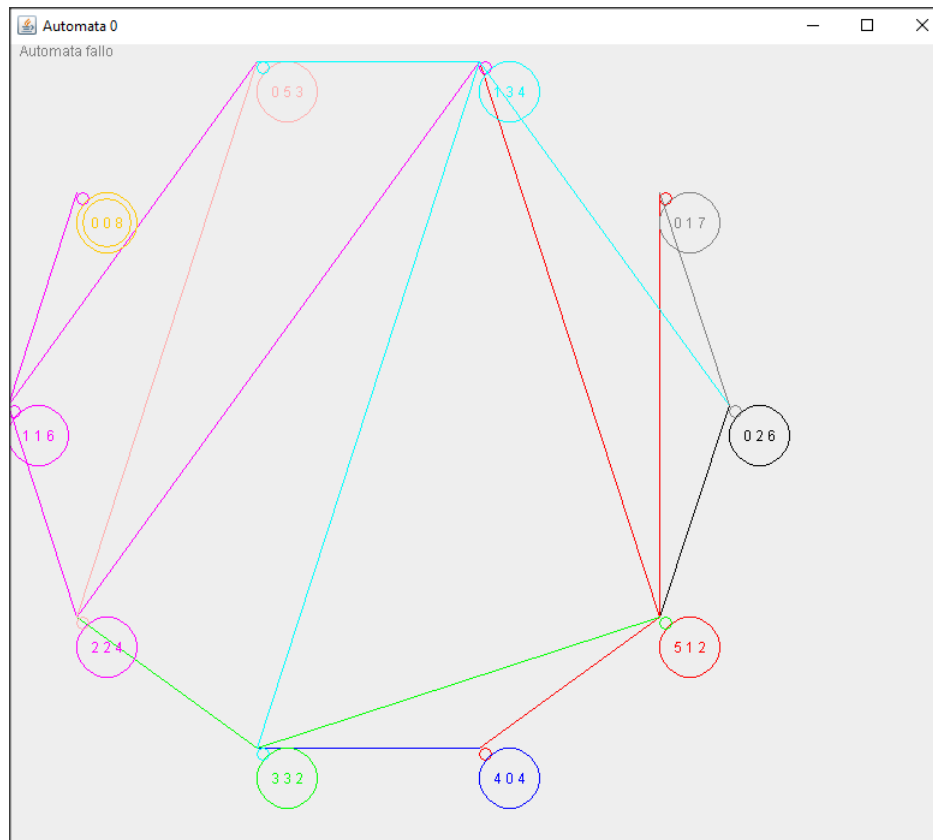
Hacemos unas ultimas pruebas con numeros que no sean multiplos de 3



Observamos que nos da un automata el cual falla donde n es igual a 8.



Observamos que nos da un automata el cual falla donde n es igual a 10.



Observamos que nos da un automata el cual falla donde n es igual a 13.

Clase principal Planeta

La clase principal Planeta es la encargada de hacer todo el analisis del las combinaciones.

Codigo fuente del programa

Clase Principal

```
package planeta;

import java.util.ArrayList;
import java.util.Scanner;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class Planeta {
    public static void main(String [] args) {
        int op = 0;
        while (op == 0){
            String aux;
            //Scanner rd = new Scanner (System.in);
            //System.out.println("dato:");
            //caso = rd.nextInt();
            aux = JOptionPane.showInputDialog("Escriba el numero de especies");
            int caso =Integer.parseInt(aux);
            Planeta p1 = new Planeta (caso);
            p1.getpermutaciones();
            especies = p1.getCombinaciones(especies);
            p1.crearAutomatas();
            p1.t();
            p1.generarDatosDibujo(p1.automatas);
            p1.dibujar(p1.automatas);
            especies.clear();
            op = JOptionPane.showConfirmDialog(null, " Volver a intentar?");
        }
    }

    static ArrayList <String []> especies = new ArrayList <String []>();
    ArrayList<String []> aaa = new ArrayList<String []>();
    ArrayList <Nodo> nodos = new ArrayList <Nodo>();
    ArrayList <Automata> automatas = new ArrayList <Automata>();
    int n = 0;
    String [] cadena = new String [3];

    public Planeta(int n){
        this.n = n;
    }

    public void getpermutaciones(){
        int a = 0;
        for (int i = 0; i<n; i++){
            for (int j = 0; j<n; j++){
                for (int z = 0; z<n; z++){
```

```

        if ((i+j+z) == n){
            especies.add(new String[3]);
            (especies.get(a))[0] = Integer.toString(i)
            ;
            (especies.get(a))[1] = Integer.toString(j)
            ;
            (especies.get(a))[2] = Integer.toString(z)
            ;
            a++;
        }
    }
}

public ArrayList <String[]> getCombinaciones(ArrayList <String
[]> especi){
    for (int i = 0; i<especi.size(); i++ ) {
        for (int j = 0; j<especi.size(); j++){
            if (i != j) {
                if (comparador (especi.get(i), especi.get(j))
                    == true){
                    especi.remove(j);
                    j = 0;
                    i = 0;
                    break;
                }
            }
        }
    }
    return especi;
}

public boolean comparador(String[] a , String[] b){
    ArrayList<String> aa = new ArrayList<String>();
    ArrayList<String> bb = new ArrayList<String>();

    for (int x = 0; x <3; x++) {
        aa.add(a[x]);
    }
    for (int y = 0; y <3; y++) {
        bb.add(b[y]);
    }
    for (int i = 0; i < aa.size(); i++) {
        for (int j = 0; j < bb.size(); j++) {
            if (aa.get(i).equals(bb.get(j)) == true) {
                bb.remove(j);
            }
        }
    }
    if (bb.isEmpty()) {
        return true;
    }
}

```

```

        return false;
    }

    public String [] apareo (String [] estado){
        ArrayList <String> cc = new ArrayList<String>();
        ArrayList <String> dd = new ArrayList<String>();

        for (int j=0; j<3; j++){
            cc.add(estado[j]);
        }
        for (int i = 0; i<cc.size(); i++) {
            if(cc.size() != 1){
                if (!cc.get(i).equals("0")){
                    dd.add(Integer.toString((Integer.parseInt(cc.get(i))-1)));
                    cc.remove(i);
                    i=-1;
                }
            }else{
                dd.add(Integer.toString((Integer.parseInt(cc.get(i))+2)));
                cc.remove(i);
                i=-1;
            }
        }
        String nuevo [] = new String [3];
        nuevo [0] = dd.get(0);
        nuevo [1] = dd.get(1);
        nuevo [2] = dd.get(2);
        return nuevo;
    }

    public boolean buscarTope(String [] ca){
        for (int i = 0; i<3; i++) {
            if (ca[i].equals(Integer.toString(n))) {
                return true;
            }
        }
        return false;
    }

    public ArrayList<String []> dameTodosApareos (String []
    valorNodo){
        ArrayList<String []> permutaciones = new ArrayList<String
        []> (permutar (stringArrayList(valorNodo)));
        ArrayList<String []> apareos = new ArrayList<String []> ();
        for (int i = 0; i<permutaciones.size(); i++) {
            apareos.add(apareo(permutaciones.get(i)));
        }
        getCombinaciones(apareos);
        return apareos;
    }
}

```

```

public ArrayList<String []> permutar( ArrayList<String []>
conjunto) {
    ArrayList<String []> permutas = new ArrayList<String []>();
    int per=0;
    for (int i= 0;i<3 ;i++ ) {
        for (int j = 0;j<3 ;j++ ) {
            for (int z=0;z<3 ;z++ ) {
                if (i!=j && j!=z && i!=z){
                    permutas.add(new String [3]);
                    permutas.get(per)[0] = conjunto.get(0)[i];
                    permutas.get(per)[1] = conjunto.get(0)[j];
                    permutas.get(per)[2] = conjunto.get(0)[z];
                    per++;
                }
            }
        }
    }
    return permutas;
}

public ArrayList<String []> stringArrayList(String [] cadena){

    ArrayList <String []> a = new ArrayList <String []>();
    a.add(cadena);
    return a;
}

public void falla (String [] raiz){
    Nodo ra = new Nodo(raiz);
    ArrayList <String []> apareos = new ArrayList <String []>();
    apareos = dameTodosApareos(ra.valorNodo);
    nodos.add(new Nodo(raiz));
    for (int i = 0; i<apareos.size(); i++){
        int inter = 0;
        if (buscarTope(apareos.get(i)) == true){
            //System.out.println("Automata falla ");

            nodos.add(new Nodo(apareos.get(i)));
        }
        for (int j = 0; j<nodos.size(); j++){
            if (comparador (nodos.get(j).valorNodo, apareos.get(i)
) == true){
                inter = 1;
            }
        }
        if (inter == 0 && buscarTope(apareos.get(i)) == false){
            falla (apareos.get(i));
        }
    }
}
}

```



```

public void crearAutomatas () {
    int inter=0;
    for (int i = 0; i<especies.size(); i++){
        if (i==0){
            System.out.println("Automata_" + i + "_raiz_" +
                especies.get(i)[0]+ especies.get(i)[1]+ especies.
                get(i)[2] );
            falla(especies.get(i));
            ArrayList <Nodo> n = new ArrayList <Nodo>(nodos);
            automatas.add(new Automata (i, n));
            automatas.get(i).verAutomata();
            nodos.clear();
        } else {
            for (int a=0; a<automatas.size(); a++){
                for (int b= 0; b<automatas.get(a).nodos.size();
                    b++) {
                    if (comparador (automatas.get(a).nodos.get
                        (b).valorNodo, especies.get(i)) ==true)
                    {
                        inter = 1;
                    }
                }
            }
            if(inter == 0){
                System.out.println("Automata_" + i + "_raiz_" +
                    especies.get(i)[0]+ especies.get(i)[1]+
                    especies.get(i)[2] );
                falla(especies.get(i));
                ArrayList <Nodo> n2 = new ArrayList <Nodo>(
                    nodos);
                automatas.add(new Automata (i, n2));
                automatas.get(i).verAutomata();
                nodos.clear();
            }
        }
    }
}

public void generarDatosDibujo (ArrayList<Automata> automatas) {
    int radio = 300;
    int x =2*radio;
    int y = radio;
    for (int i = 0; i < automatas.size(); i++) {
        if(automatas.get(i).nodos.size() != 0) {
            double separacion = Math.toRadians(360/automatas.
                get(i).nodos.size());
            double au = separacion;
            for (int j = 0; j < automatas.get(i).nodos.size();
                j++) {
                automatas.get(i).nodos.get(j).radio = 50;
                automatas.get(i).nodos.get(j).x = x;
                automatas.get(i).nodos.get(j).y = y;
                if (j != 0 ) {

```

```

        separacion += au;
    }
    x = (int)((radio*Math.cos(separacion)) + radio);
    y = (int)((radio*Math.sin(separacion)) + radio);
}
}
}

for (int i = 0; i<automatas.size(); i++){
    for (int j = 0; j < automatas.get(i).nodos.size(); j++)
    {
        for (int z = 0; z < automatas.get(i).nodos.get(j).
            transiciones.size(); z++){

            for (int a = 0; a < automatas.get(i).nodos.size
                (); a++) {

                if (comparador (automatas.get(i).nodos.get(
                    a).valorNodo, automatas.get(i).nodos.get
                    (j).transiciones.get(z))) {
                    automatas.get(i).nodos.get(j).
                        dibujoTransicion.add(new Transicion(
                            automatas.get(i).nodos.get(j).x
                                ,automatas.get(i).nodos.get(
                                    j).y,
                            automatas.get(i).nodos.get(a).x
                                ,automatas.get(i).nodos.get(
                                    a).y
                                ));
                }
            }
        }
    }
}

public void dibujar( ArrayList<Automata> a){
    ArrayList<JFrame> ventanas = new ArrayList<JFrame>();
    ArrayList<Dibujo> dibujos = new ArrayList<Dibujo>();

    int v = 0;
    int v2= 0;
    int aasd=0;
    for (int i = 0; i< automatas.size(); i++){
        //Dibujo dibujo = new Dibujo (automatas, i);
        //JFrame frame = new JFrame();

        dibujos.add(new Dibujo(automatas, i));
        ventanas.add(new JFrame());
        ventanas.get(v).setTitle("Automata_" + i);
        ventanas.get(v).setSize(800, 800);
    }
}

```

```

        ventanas.get(v).setDefaultCloseOperation(JFrame.
            EXIT_ON_CLOSE);
        ventanas.get(v).getContentPane().add(dibujos.get(v2));
        ventanas.get(v).setVisible(true);
        dibujos.get(v2).repaint();
        v++;
        v2++;
    }
}

public void t () {
    for (int i = 0; i < automatas.size(); i++) {
        for (int j = 0; j < automatas.get(i).nodos.size(); j++)
        {
            if (!buscarTope(automatas.get(i).nodos.get(j).
                valorNodo)) {
                automatas.get(i).nodos.get(j).datosT(
                    dameTodosApareos(automatas.get(i).nodos.get
                        (j).valorNodo));
            } else {
                automatas.get(i).nodos.get(j).tipo=1;
                automatas.get(i).tipo = 1;
            }
        }
    }
}
}
}
}

```

Clase Nodo

La clase nodo es la representacion de los estados de cada uno de los automatas que se forma contiene a una de las diferentes combinaciones de un Automata.

```
package planeta;

import java.util.ArrayList;

public class Nodo{
    String valorNodo [] = new String [3];
    ArrayList<String[]> transiciones = new ArrayList<String[]> ();
    ArrayList<Transicion> dibujoTransicion = new ArrayList<
        Transicion> ();
    int tipo=0;

    int x,y,radio;

    public Nodo (String [] nodo){
        for (int i = 0;i<3 ;i++ ) {
            valorNodo[i] = nodo[i];
        }
    }

    public String getValorNodo(){
        String cadena="";
        for (int i=0; i<3; i++){
            if (i!=2){
                cadena = cadena + valorNodo[i] + "_";
            }else{
                cadena = cadena + valorNodo[i];
            }
        }
        return cadena;
    }
    public void datosT(ArrayList<String[]> transiciones){
        this.transiciones = transiciones;
    }
}
```

Clase Automata

La clase Automata es la representacion de un automata el cual contiene datos como lo son los nodos y las transiciones de cada uno de ellos ademas de si tiene o no tiene estado final.

```
package planeta;

import java.util.ArrayList;

public class Automata{
    int idAutomata;
    int conodo = -1;
    ArrayList <Nodo> nodos = new ArrayList <Nodo>();
    int tipo = 0;

    public Automata (int idAutomata, ArrayList <Nodo> nodos){
        this.idAutomata = idAutomata;
        this.nodos = nodos;
    }

    public void insertarNodo(Nodo nodo){
        nodos.add(nodo);
        conodo++;
    }

    public void verAutomata(){
        for (int i= 0; i<nodos.size(); i++){
            System.out.println (nodos.get(i).valorNodo[0] + nodos.
                get(i).valorNodo[1] +nodos.get(i).valorNodo[2]);
            //System.out.println ("transicion tamaño " + nodos.
                get(i).transiciones.size());
            /*for (int j = 0; j<nodos.get(i).transiciones.size();
                j++){
                System.out.println ("transicion " + j + " " +
                    nodos.get(i).transiciones.get(j)[0] +
                    nodos.get(i).transiciones.get(j)[1] +
                    nodos.get(i).transiciones.get(j)[2]);
            }*/
        }
    }
}
```

Clase Dibujo

La clase Dibujo es la encargada de dibujar en una ventana el automata, por lo que necesita los datos de un automata con todos sus datos como lo son sus nodos sus transiciones y si tiene o no estado final.

```
package planeta;

import java.awt.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class Dibujo extends Canvas{
    ArrayList<Automata> automatas = new ArrayList <Automata>();
    int i;

    public Dibujo (ArrayList <Automata> automatas , int i){
        this.automatas = automatas;
        this.i = i;
    }

    public void paint (Graphics g){
        //System.out.println ("Automata " + i + " a dibujar");
        automatas.get(i).verAutomata();

        for (int j = 0; j < automatas.get(i).nodos.size(); j++){
            int x = automatas.get(i).nodos.get(j).x;
            int y = automatas.get(i).nodos.get(j).y;
            int r = automatas.get(i).nodos.get(j).radio;
            //System.out.println ("x " + x + " y " + y);
            if (j == 0){g.setColor(Color.black);}
            if (j == 1){g.setColor(Color.red);}
            if (j == 2){g.setColor(Color.blue);}
            if (j == 3){g.setColor(Color.GREEN);}
            if (j == 4){g.setColor(Color.MAGENTA);}
            if (j == 6){g.setColor(Color.ORANGE);}
            if (j == 7){g.setColor(Color.pink);}
            if (j == 8){g.setColor(Color.cyan);}
            if (j == 9){g.setColor(Color.gray);}

            g.drawOval(x,y,r,r);
            g.drawString(automatas.get(i).nodos.get(j).
                getValorNodo() , (x)+r/4, y+30);

            if ((automatas.get(i).tipo == 1)){
                g.drawString("Automata_fallo",10, 10);
            }
            if (automatas.get(i).nodos.get(j).tipo==1){
                g.drawOval(x+5,y+5,r-10,r-10);
            }
        }
    }
}
```

```

    }
    }else{
    g.drawString("Automata_no_fallo", 10, 10);
    }

    for (int k = 0; k < automatas.get(i).nodos.get(j).
    dibujoTransicion.size(); k++) {
    g.drawLine (automatas.get(i).nodos.get(j).
    dibujoTransicion.get(k).i1 ,
    automatas.get(i).nodos.get(j).dibujoTransicion.get
    (k).f1 ,
    automatas.get(i).nodos.get(j).dibujoTransicion.get
    (k).i2 ,
    automatas.get(i).nodos.get(j).dibujoTransicion.get
    (k).f2);

    g.drawOval(automatas.get(i).nodos.get(j).
    dibujoTransicion.get(k).i2 ,
    automatas.get(i).nodos.get(j).dibujoTransicion.get
    (k).f2 ,
    10, 10);
    }
    }
}

```

Clase Transicion

La clase transicion es la encargada de almacenar los datos de un dibujo de una transicion, como la transicion es una linea la transicion tendra un punto final, uno inicial y el direccion hacia donde estara apuntando.

```
package planeta;

public class Transicion {
    int i1 , f1 ;
    int i2 , f2 ;

    public Transicion(int i1 , int f1 , int i2 , int f2){
        this.i1 = i1 ;
        this.f1 = f1 ;
        this.i2 = i2 ;
        this.f2 = f2 ;
    }
}
```


Programa de una expresion regular

El siguiente problema crea cadenas que genera la siguiente expresion regular

$O(11)^*((00+01)^*(3+00+01)^*)^*+00$

Codigo fuente

```
package expresionregular;
import java.util.Random;

public class ExpresionRegular {
    public static void main(String [] args) {

        ExpresionRegular er = new ExpresionRegular();
        String cadena;

        if (er.lanzarRandom(1)==0){
            int vciclo = er.lanzarRandom(0);

            cadena = "0";
            for (int i = 0; i<vciclo; i++){
                cadena = cadena + "11";
            }

            vciclo = er.lanzarRandom(0);

            for (int i = 0; i<vciclo; i++){
                if (er.lanzarRandom(1) == 0){

                    int v2ciclo = er.lanzarRandom(0);

                    for (int j = 0; j< v2ciclo; j++){
                        if (er.lanzarRandom(1)==0){
                            cadena = cadena + "00";
                        }else {
                            cadena = cadena + "01";
                        }
                    }
                }else {
                    int v3ciclo = er.lanzarRandom(0);

                    for (int j = 0; j< v3ciclo; j++){
                        if (er.lanzarRandom(1)==0){
                            cadena = cadena + "00";
                        }else {
                            cadena = cadena + "01";
                        }
                    }
                }
            }
        }
    }
}
```

```

        }

    }else {
        cadena = "00";
    }

    System.out.println(cadena);

}
public int lanzarRandom(int op){

    Random ra = new Random();

    if (op == 1 ){
        return (int)(Math.random()*2);
    }
    else {
        return (int)(Math.random()*10);
    }
}
}

```

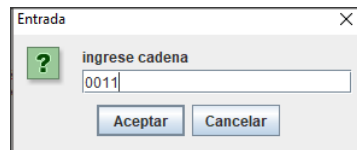
Programa de Automata de Pila

El siguiente programa es un Automata de Pila que esta diseñado para aceptar

$$\{0^n + 1^1 | n \geq 1\}$$

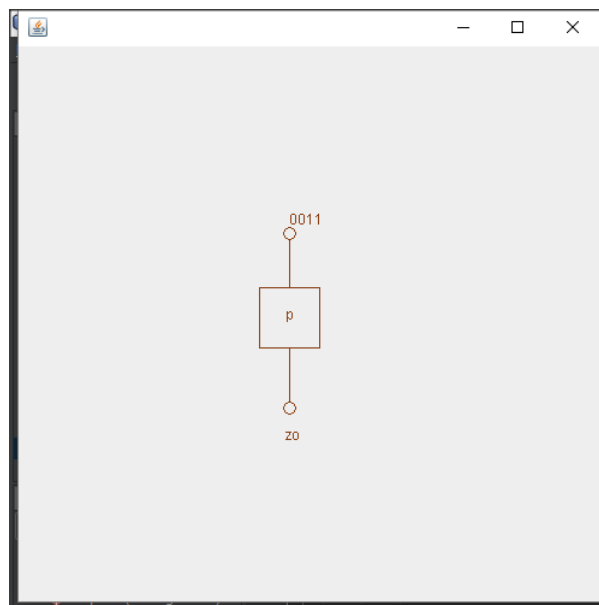
Pruebas del programa

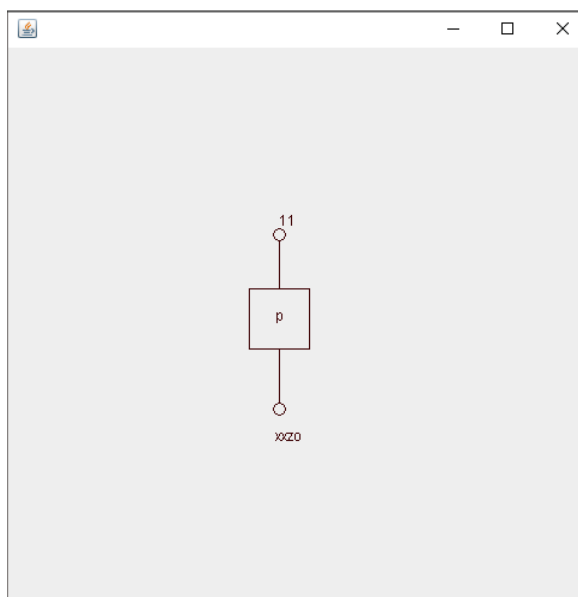
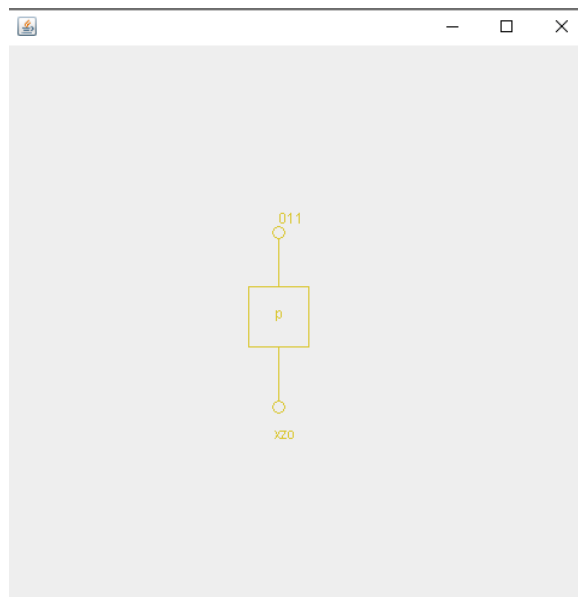
Introducimos la cadena 0011 la cual debe ser aceptada por el programa.

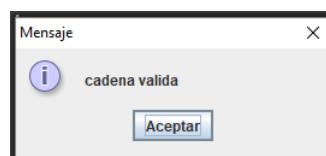
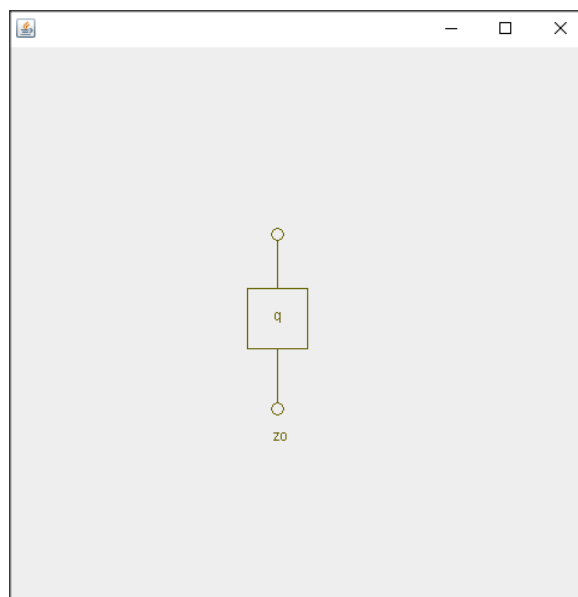
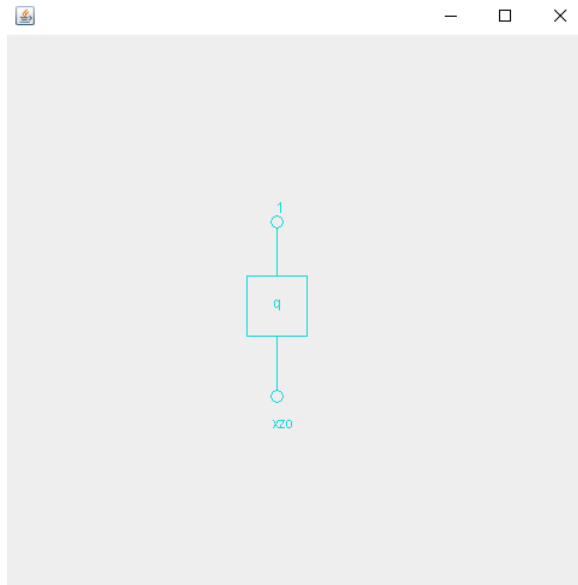


Primero nos pregunta la cadena que queremos evaluar.

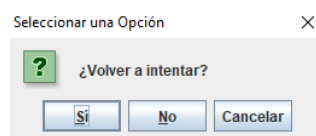
Y comienza la evaluacion.





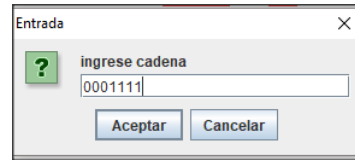


Nos indica que la cadena es valida para el lenguaje del Automata



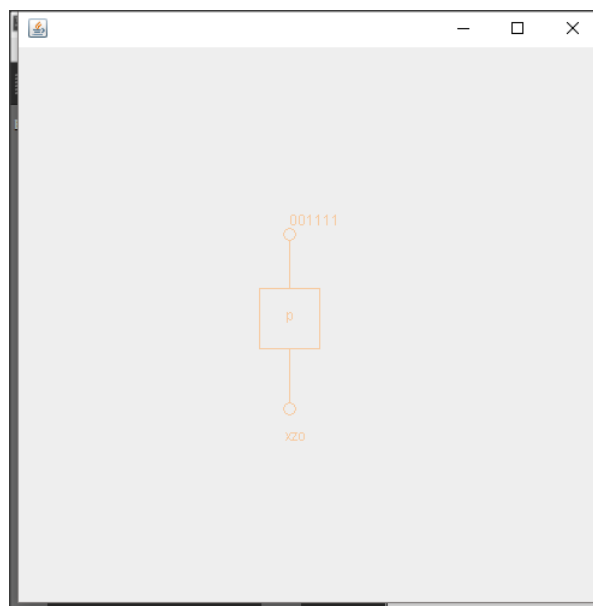
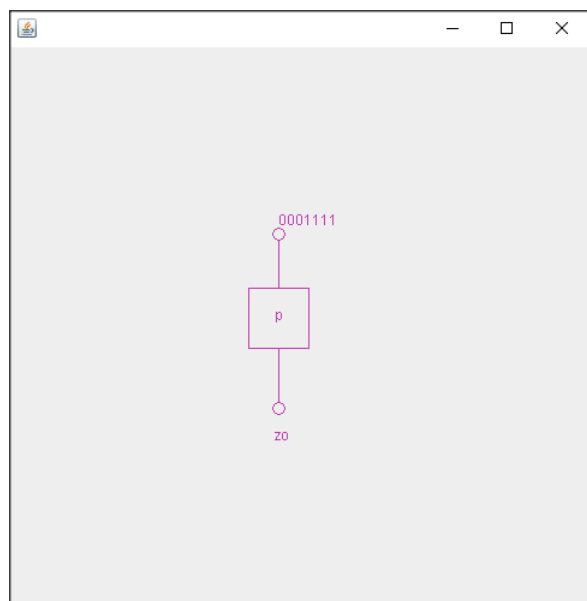
Le decimos que si e introducimos una nueva cadena.

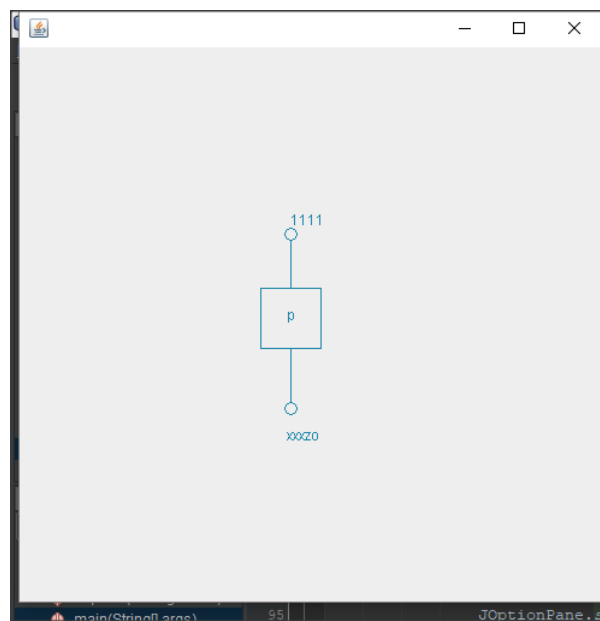
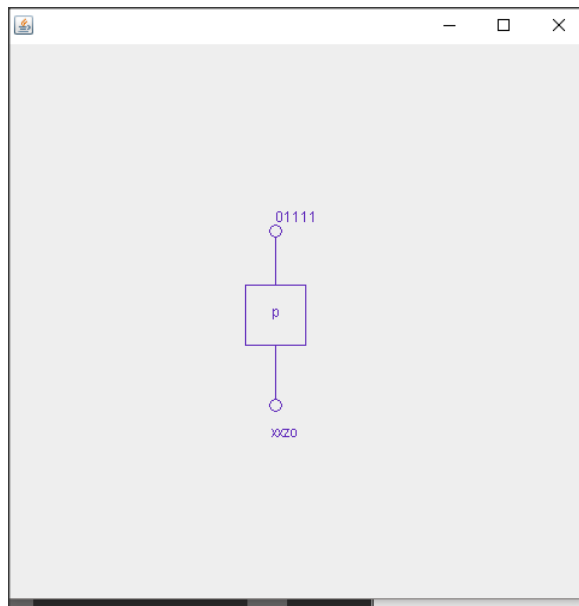
Ahora haremos una prueba con una cadena no perteneciente al lenguaje "0001111".

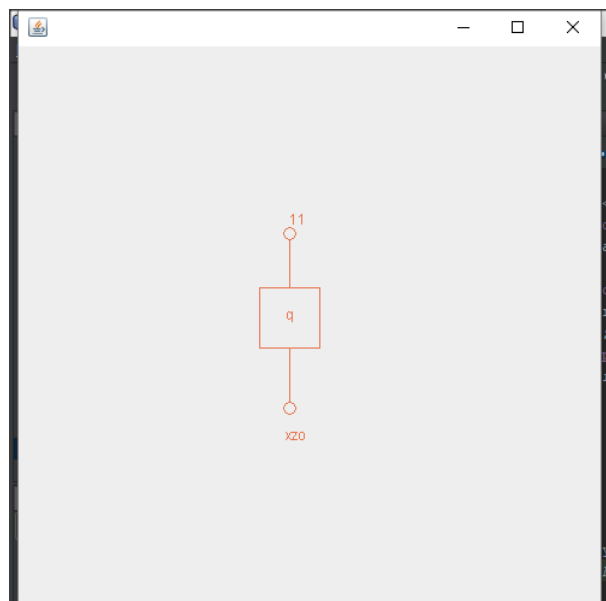
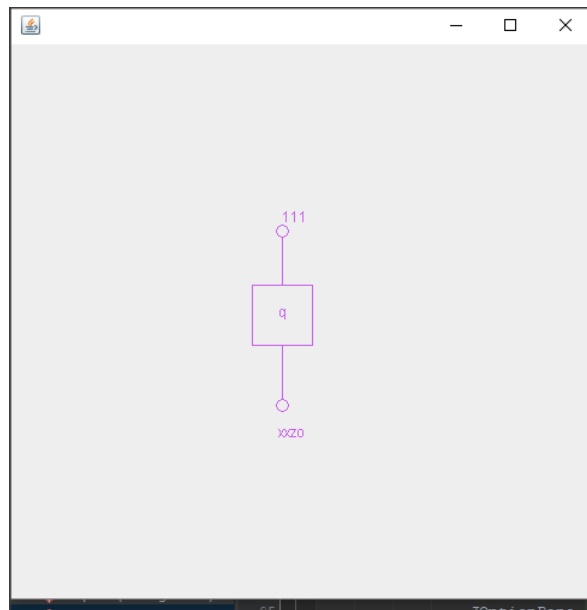


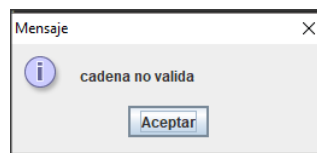
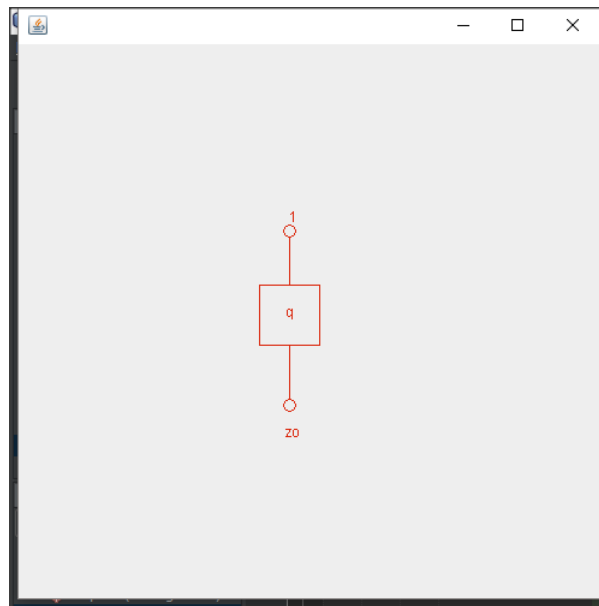
Introducimos la cadena 0001111

Y observamos el proceso de la evaluacion.









Como observamos la cadena no es valida. porque la pila esta vacia antes de leer todos los datos.

Codigo fuente

Clase Principal

Es la encargada de la logica del programa, los datos generados por esta clases se le pasan a la clase dibujoPila. La clase principal es la encargada de hacer la animacion del analisis de las cadenas que se le introduzcan.

```
package pila;  
  
import java.util.ArrayList;  
import java.util.Timer;  
import java.util.TimerTask;  
import javax.swing.JFrame;  
import javax.swing.JOptionPane;  
  
public class Pila {  
  
    static ArrayList<String> pila = new ArrayList <String>();  
    DibujoPila dibujo = new DibujoPila();  
  
    public boolean meter (String v){  
        pila.add(v);  
    }  
}
```

```

        return true;
    }

    public boolean sacar () {
        if (!pila.isEmpty()) {
            pila.remove(pila.size()-1);
            return true;
        }
        return false;
    }

    public String mostrar () {
        return pila.get(pila.size()-1);
    }

    public static void esperar (int segundos) {
        try {
            Thread.sleep(segundos*1000);
        } catch (InterruptedException e) {

        }
    }

    public void dibujar(String cadena2) {
        dibujo.cadena = cadena2;
        JFrame venta = new JFrame ();
        venta.setSize(500, 500);
        venta.add(dibujo);
        venta.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        venta.setVisible(true);
    }

    public void repintar() {
        dibujo.repaint();
    }

    public static void main(String [] args) {
        int op=0;
        while (op==0){
            Pila p = new Pila();
            String cadena = JOptionPane.showInputDialog("ingrese _
            cadena");
            String cadena2 = cadena;
            int i =0;

            p.dibujar(cadena);

            Pila.esperar(7);

            for (int j = 0; j<cadena.length();j++){
                p.dibujo.estado = "p";
                if (cadena.charAt(j) == '0') {
                    cadena2 = cadena2.substring(1,cadena2.length()
                    );
                    p.dibujo.cadena =cadena2;
                }
            }
        }
    }

```

```

        p.repaint();
        p.meter("x");
        p.dibujo.pila = p.pila;
        Pila.esperar(10);
    }else{
        i=j;
        break;
    }
}
for (int a = i; a<cadena.length(); a++){
    p.dibujo.estado = "q";
    if (cadena.charAt(a)=='1' && !pila.isEmpty()){
        cadena2 = cadena.substring(1,cadena.length()
        );
        p.dibujo.cadena = cadena2;
        p.repaint();
        p.sacar();
        p.dibujo.pila = p.pila;
        Pila.esperar(10);
    }else{
        i=a;
        break;
    }
    i=a;
}

if (p.pila.isEmpty() && cadena2.length() == 0){
    JOptionPane.showMessageDialog(null, "cadena_valida
    ");
}else{
    JOptionPane.showMessageDialog(null, "cadena_no_
    valida");
}
Pila.esperar(7);
op = JOptionPane.showConfirmDialog(null, " Volver  a
    intentar?");
}

}
}

```

Clase DibujoPila

Esta clase recibe todos los datos generados por la clase principal y dibuja los datos en un Canvas.

```

package pila;

import java.awt.Canvas;
import java.awt.Color;

```

```

import java.awt.Graphics;
import java.util.ArrayList;

public class DibujoPila extends Canvas{
    String cadena;
    String estado = "p";
    ArrayList <String> pila = new ArrayList <String>();
    String pila2 = "zo";
    public void paint (Graphics g){
        int random1 =(int)(Math.random()*254)+1;
        int random2 =(int)(Math.random()*254)+1;
        int random3 =(int)(Math.random()*254)+1;

        g.setColor(new Color (random1,random2,random3));
        g.drawLine(225, 160, 225, 200);
        g.drawLine(225, 250, 225,295);
        g.drawString(cadena, 225, 148);
        g.drawRect(200, 200, 50, 50);
        g.drawString(estado, 222, 227);
        g.drawOval(220, 150, 10, 10);
        g.drawOval(220, 295, 10, 10);

        pila2="zo";
        for(int i =0; i<pila.size(); i++){
            pila2 = pila.get(i)+ pila2;
        }

        g.drawString(pila2, 222, 327);
    }
}

```