

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE

TEORÍA COMPUTACIONAL

TAREA

REPORTES DE PRÁCTICAS

ALUMNO

BARRERA ESTRELLA EMANUEL

PROFESOR

JUÁREZ MARTÍNEZ GENARO

GRUPO

2CM1

1. Expresión Regular

Este programa que se presenta genera automáticamente cadenas de la expresión regular $(0 + 10)^*(\varepsilon + 1)$, se codificó en lenguaje Java.

```
1 import java.io.*;
2 import java.util.Random;
3 import java.util.logging.Level;
4 import java.util.logging.Logger;
5
6 public class ExpresionRegular {
7     public static void main(String [] args) {
8         String cadena;
9         int primer_union, veces, segunda_union;
10        String union1, union2;
11        Random r= new Random();
12        comand("rm expresion_regular.txt");
13        comand("echo >> expresion_regular.txt");
14        PrintWriter writer;
15        try {
16            writer = new PrintWriter("
17                expresion_regular.txt", "UTF-8");
18            System.out.println("Expresion_regular_
19                (0+10)*(E+1)");
20            System.out.println("Generando_6_
21                expresiones_regulares_en_el_archivo_
22                TXT...");
23            for (int i=0; i<6; i++){
24                cadena="";
25                union1="";
26                union2="";
27                primer_union = r.nextInt(2);
28                veces=r.nextInt(100);
29                switch(primer_union){
30                    case 0:
31                        union1="10";
32                        break;
33                    case 1:
34                        union1="0";
35                        break;
36                }
37                writer.println("De_la_union_de_0+10_
38                    se_ha_usado_"+union1);
39                writer.println("Se_repite_"+veces+"_
40                    veces");
41                for (int j=0; j<veces; j++){
42                    cadena+=union1;
```

```

37         }
38         segunda_union= r.nextInt(2);
39         switch(segunda_union){
40             case 0:
41                 union2="E";
42                 break;
43             case 1:
44                 union2="1";
45                 break;
46         }
47         writer.println("De la segunda union
48         de E+1 se ha usado "+union2);
49         cadena+=union2;
50         writer.println("La cadena "+(i+1)+"
51         es: "+cadena);
52         writer.println("
53         *****
54         ");
55     }
56     writer.close();
57 } catch (FileNotFoundException ex) {
58     Logger.getLogger(ExpresionRegular.class.
59     getName()).log(Level.SEVERE, null, ex);
60 } catch (UnsupportedEncodingException ex) {
61     Logger.getLogger(ExpresionRegular.class.
62     getName()).log(Level.SEVERE, null, ex);
63 }
64 }
65
66 public static char random_01() {
67     char letra = 0;
68     Random rand = new Random();
69     if (rand.nextInt(2) == 0) {
70         letra = '0';
71     } else {
72         letra = '1';
73     }
74     return letra;
75 }
76
77 public static void comand(String cmd) {
78     try {
79         Process p = Runtime.getRuntime().exec(cmd);
80         BufferedReader stdInput

```

```

77         = new BufferedReader(new
              InputStreamReader(p.getInputStream
              ()));
78     } catch (IOException ex) {
79         ex.printStackTrace();
80     }
81 }
82
83 public static String scan() {
84     String scan = "";
85     char c = 0;
86     BufferedReader br
87         = new BufferedReader(new
              InputStreamReader(System.in));
88     while (c != '\n') {
89         try {
90             c = (char) br.read();
91             if (c != '\n') {
92                 scan += c;
93             }
94         } catch (IOException ex) {
95             ex.printStackTrace();
96         }
97     }
98     return scan;
99 }
100 }

```

Su ejecución es:

```

emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$ javac ExpresionRegular.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$ java ExpresionRegular
Expresion regular (0+10)*(E+1)
Generando 6 expresiones regulares en el archivo TXT...
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$ █

```

Su salida es:

```

1 De la union de 0+10 se ha usado 0
2 Se repite 23 veces
3 De la segunda union de E+1 se ha usado 1
4 La cadena 1 es: 000000000000000000000001
5 *****
6 De la union de 0+10 se ha usado 10
7 Se repite 64 veces
8 De la segunda union de E+1 se ha usado 1

```



```

10     public static final String ANSI_YELLOW = "\u001B[33m"
11         ;
12     public static final String ANSI_BLUE = "\u001B[34m";
13     public static final String ANSI_PURPLE = "\u001B[35m"
14         ;
15     public static final String ANSI_CYAN = "\u001B[36m";
16     public static final String ANSI_WHITE = "\u001B[37m";
17
18     public static void main(String [] args) {
19         String cadena = "";
20         PDA pda = new PDA();
21         String s_opc;
22         int opc = 0;
23         do {
24             System.out.println("PDA");
25             System.out.println("1) Generar_cadena_de_
26                 manera_manual");
27             System.out.println("2) Generar_cadena_de_
28                 manera_aleatoria");
29             System.out.println("3) Salir");
30             System.out.println("Ingrese_una_opcion");
31             s_opc = scan();
32             try {
33                 opc = Integer.parseInt(s_opc);
34             } catch (NumberFormatException ex) {
35                 opc = 0;
36             }
37         } while (!(opc >= 1 && opc <= 3));
38         if (opc >= 1 && opc <= 2) {
39             comand("rm_estados.txt");
40             comand("echo_>>_estados.txt");
41             try {
42                 PrintWriter writer
43                     = new PrintWriter("estados.txt",
44                         "UTF-8");
45                 writer.print("{i,}");
46
47                 if (opc == 1) {
48                     char c = 0;
49                     BufferedReader br
50                         = new BufferedReader(new
51                             InputStreamReader(System.
52                                 in));
53                     while (c != '\n') {
54                         try {
55                             c = (char) br.read();

```

```

49         if (c != '\n') {
50             cadena += c;
51             pda.analizar(c);
52             if (pda.getEstado() == 'f
53                 ') {
54                 writer.print("p,␣");
55             } else {
56                 writer.print(" " + pda
57                     .getEstado() + ",␣
58                     ");
59             }
60         }
61     } catch (java.io.IOException ex)
62     {
63         ex.printStackTrace();
64     }
65 }
66
67 if (opc == 2) {
68     char c = 0;
69     Random rand = new Random();
70     int random = rand.nextInt(4);
71     if (random == 0) {
72         random = rand.nextInt(100);
73         for (int i = 0; i < random; i++)
74         {
75             c = random_01();
76             cadena += c;
77             pda.analizar(c);
78             if (pda.getEstado() == 'f') {
79                 writer.print("p,␣");
80             } else {
81                 writer.print(" " + pda.
82                     getEstado() + ",␣");
83             }
84             System.out.print(c);
85         }
86         System.out.println();
87     } else {
88         random = rand.nextInt(50);
89         for (int i = 0; i < random; i++)
90         {
91             c = '0';
92             cadena += c;
93             pda.analizar(c);
94             if (pda.getEstado() == 'f') {

```

```

88         writer.print("p,␣");
89     } else {
90         writer.print(" " + pda.
            getEstado() + ",␣");
91     }
92     System.out.print(c);
93 }
94 for (int i = 0; i < random; i++)
95 {
96     c = '1';
97     cadena += c;
98     pda.analizar(c);
99     if (pda.getEstado() == 'f') {
100         writer.print("p,␣");
101     } else {
102         writer.print(" " + pda.
103             getEstado() + ",␣");
104     }
105     System.out.print(c);
106 }
107 System.out.println();
108 }
109 writer.println("f␣");
110 writer.close();
111 if (pda.getEstado() == 'f') {
112     System.out.println(ANSI_GREEN + "La␣
113         cadena␣pertenece␣al␣lenguaje" +
114         ANSI_GREEN);
115
116     AnimacionPda animacion = new
117         AnimacionPda();
118     animacion.animar(cadena, 0);
119     animacion.setVisible(true);
120     esperar(1f);
121     int i = 0;
122     do {
123         animacion.animar(cadena.substring
124             (i + 1), (i + 1));
125         animacion.setVisible(true);
126         esperar(1f);
127         i++;
128     } while (cadena.charAt(i) == '0');
129     cadena = cadena.substring(i);
130     for (int j = 0; j < cadena.length();
131         j++) {

```



```

126         i--;
127         animacion.animar(cadena.substring
            (j + 1), i);
128         animacion.setVisible(true);
129         esperar(1f);
130     }
131 } else {
132     System.out.println(ANSI_RED + "La
        cadena no pertenece al lenguaje" +
        ANSI_RED);
133 }
134 } catch (IOException ex) {
135     ex.printStackTrace();
136 }
137 }
138 }
139
140 public static char random_01() {
141     Random rand = new Random();
142     return ((char) (rand.nextInt(2) + 48));
143 }
144
145 public static void comand(String cmd) {
146     try {
147         Process p = Runtime.getRuntime().exec(cmd);
148         BufferedReader stdInput
149             = new BufferedReader(new
                InputStreamReader(p.getInputStream()
                ));
150     } catch (IOException ex) {
151         ex.printStackTrace();
152     }
153 }
154
155 public static String scan() {
156     String scan = "";
157     char c = 0;
158     BufferedReader br
159         = new BufferedReader(new
            InputStreamReader(System.in));
160     while (c != '\n') {
161         try {
162             c = (char) br.read();
163             if (c != '\n') {
164                 scan += c;
165             }

```

```

166         } catch (IOException ex) {
167             ex.printStackTrace();
168         }
169     }
170     return scan;
171 }
172
173 public static void esperar(float s) {
174     try {
175         Thread.sleep((int) (s * 1000));
176     } catch (InterruptedException ex) {
177         ex.printStackTrace();
178     }
179 }
180 }

```

Para la animación se usó:

```

1 import java.awt.*;
2 import javax.swing.*;
3
4 public class PumpingLemma extends JPanel {
5
6     private int n;
7     private String cadena;
8     private int x_rect;
9     private int y_rect;
10    private int y_space;
11    private int x_space;
12
13    public PumpingLemma(String s,int i) {
14        cadena=s;
15        n = i;
16        x_rect = 40;
17        y_rect = 30;
18        y_space = 80;
19        x_space = 120;
20    }
21
22    public void setCadena(String cadena) {
23        this.cadena = cadena;
24    }
25
26
27    @Override
28    protected void paintComponent(Graphics g) {
29        super.paintComponent(g);

```

```

30 Graphics2D g2 = (Graphics2D) g;
31 g2.setColor( Color.BLACK);
32 g2.drawString(cadena, x_space + 70, y_space);
33 g2.drawLine(x_space+72, y_space+2, x_space+72,
34             y_space+20);
35 int [] vx2 = {x_space+72, x_space+76, x_space+68};
36 int [] vy2 = {y_space+30, y_space+20, y_space+20};
37 g2.fillPolygon(vx2, vy2, 3);
38
39 for (int i = 0; i < n; i++) {
40     //Contorno del espacio de la pila
41     g2.setColor( Color.BLACK);
42     g2.drawRect(x_space + 50, (i + 1) * y_rect +
43                y_space, x_rect, y_rect);
44     g2.setColor( Color.GREEN);
45     g2.fillRect(x_space + 51, (i + 1) * y_rect +
46                1 + y_space, x_rect - 1, y_rect - 1);
47     g2.setColor( Color.BLACK);
48     g2.drawString("X",x_space + 50+17, (i + 1) *
49                y_rect + y_space+20);
50 }
51 g2.setColor( Color.BLACK);
52 g2.drawRect(x_space + 50, (n + 1) * y_rect +
53            y_space, x_rect, y_rect);
54 g2.setColor( Color.ORANGE);
55 g2.fillRect(x_space + 51, (n + 1) * y_rect + 1 +
56            y_space, x_rect - 1, y_rect - 1);
57 g2.setColor( Color.BLACK);
58 g2.drawString("Z0",x_space + 50+13, (n+ 1) *
59            y_rect + y_space+20);
60 }
61 }

1 import java.awt.*;
2 import java.awt.event.ActionEvent;
3 import java.awt.event.ActionListener;
4 import javax.swing.*;
5
6 public class AnimacionPda extends JFrame implements
7     ActionListener {
8     private JScrollPane scroll;
9     private JPanel panel;
10

```

```

11     public AnimacionPda() {
12         setTitle(" Animacion");
13         setLocation(200, 50);
14         setSize(400, 500);
15         setDefaultCloseOperation(WindowConstants.
16             EXIT_ON_CLOSE);
17         setLayout(null);
18         scroll = new JScrollPane();
19         scroll.setBounds(0, 0, getWidth(), getHeight());
20         scroll.setHorizontalScrollBarPolicy(JScrollPane.
21             HORIZONTAL_SCROLLBAR_NEVER);
22         scroll.setVerticalScrollBar().
23             addAdjustmentListener(this);
24         add(scroll);
25     }
26
27     public void animar(String s, int n) {
28         panel = new PumpingLemma(s,n);
29         panel.setPreferredSize(new Dimension(400, (n+2)
30             *40));
31         scroll.setViewportView(panel);
32     }
33
34     @Override
35     public void adjustmentValueChanged(AdjustmentEvent ae
36         ) {
37         //animar();
38         setVisible(true);
39     }
40 }

```

El autómata se codificó de la siguiente manera:

```

1  public class PDA {
2
3      private Pila pila;
4      private char estado;
5
6      public PDA() {
7          pila = new Pila();
8          estado = 'q';
9          pila.push("Z0");
10     }
11
12     public char getEstado() {
13         return estado;
14     }

```

```

15     public void analizar(char input) {
16         if (estado != 'e') {
17             switch (input) {
18                 case '0':
19                     switch (estado) {
20                         case 'q':
21                             pila.push("X");
22                             break;
23                         default:
24                             estado = 'e';
25                             break;
26                     }
27                     break;
28                 case '1':
29                     switch (estado) {
30                         case 'q':
31                             estado = 'p';
32                             pila.pop();
33                             break;
34                         case 'p':
35                             pila.pop();
36                             break;
37                         default:
38                             estado = 'e';
39                             break;
40                     }
41                     break;
42                 default:
43                     estado = 'e';
44             }
45             if (pila.top().equals("Z0") && estado != 'e') {
46                 estado = 'f';
47             }
48         }
49     }
50 }

```

Para el TAD pila se usó:

```

1  import java.util.ArrayList;
2
3  public class Pila {
4      private ArrayList<String> pila;
5      private int tope;
6      public Pila() {
7          pila = new ArrayList<>();
8          tope = -1;

```

```

9      }
10     public void push(String n){
11         pila.add(++tope, n);
12     }
13     public String pop(){
14         String n=" ";
15         if(!estaVacia()){
16             n=pila.get(tope);
17             pila.remove(tope--);
18         }
19         return n;
20     }
21     public String top(){
22         String n=" ";
23         if(!estaVacia()){
24             n=pila.get(tope);
25         }
26         return n;
27     }
28     public boolean estaVacia(){
29         return tope== -1;
30     }
31 }

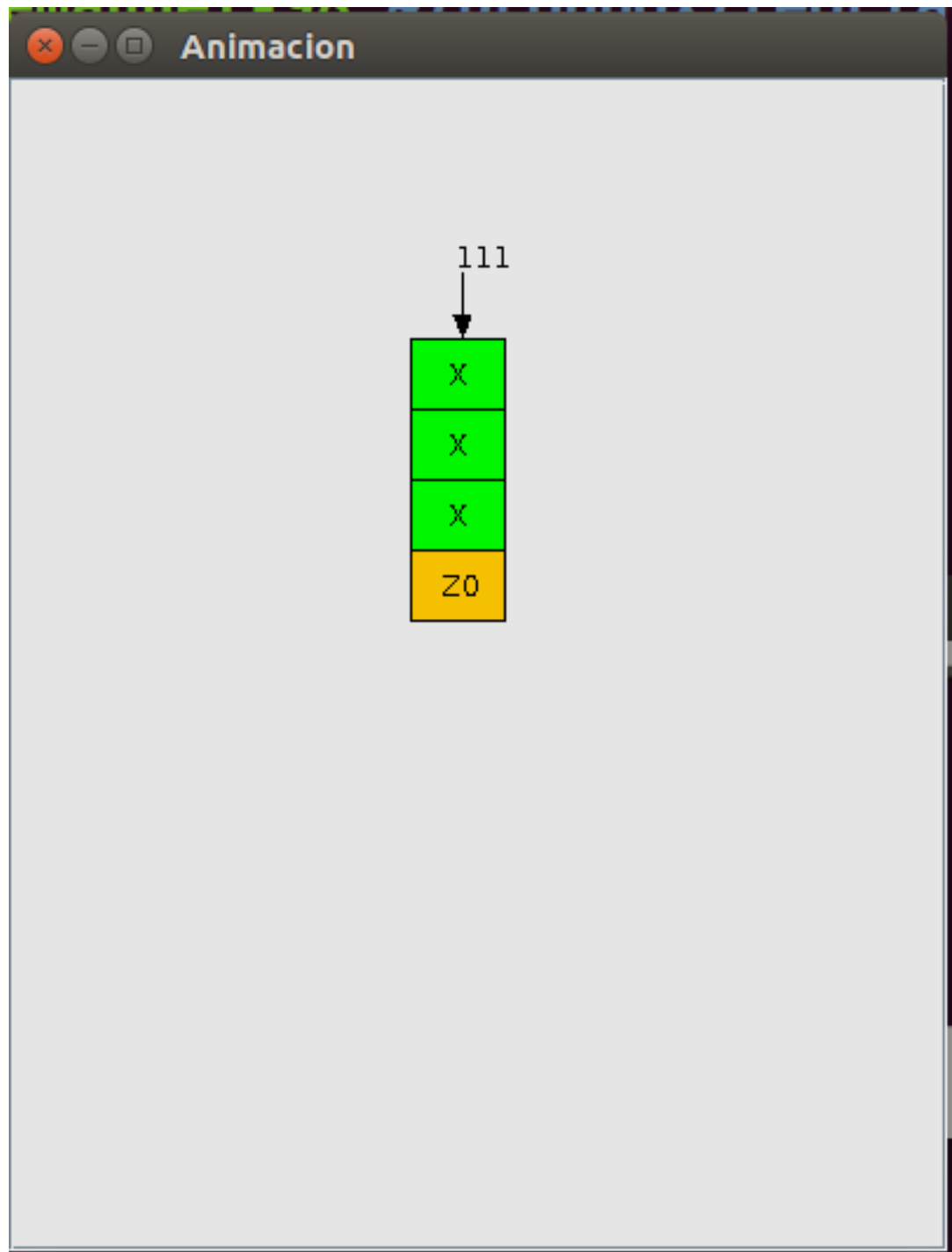
```

Su ejecución es:

```

emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p2$ javac *.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p2$ java PushdownAutomata
PDA
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
000111
La cadena pertenece al lenguaje

```



Su salida es:

```
1 { i, q, q, q, p, p, p, f }
```

3. Palíndromo

Este programa simula una gramática libre de contexto que genera una cadena que es palíndromo. La gramática se define como:

$$P \rightarrow aPa | b | \varepsilon$$

```
1 import java.io.*;
2 import java.util.Random;
3
4 public class Palindromo {
5
6     public static final String ANSI_RESET = "\u001B[0m";
7     public static final String ANSI_BLACK = "\u001B[30m";
8     public static final String ANSI_RED = "\u001B[31m";
9     public static final String ANSI_GREEN = "\u001B[32m";
10    public static final String ANSI_YELLOW = "\u001B[33m";
11
12    ;
13    public static final String ANSI_BLUE = "\u001B[34m";
14    public static final String ANSI_PURPLE = "\u001B[35m";
15
16    ;
17    public static final String ANSI_CYAN = "\u001B[36m";
18    public static final String ANSI_WHITE = "\u001B[37m";
19
20    public static void main(String[] args) {
21        char c;
22        String cadena = "";
23        String s_opc;
24        int opc = 0;
25        do {
26            System.out.println("Palindromo");
27            System.out.println("P->aPa|b|E");
28            System.out.println("1) Verificar si una cadena
29                cualquiera es palindromo");
30            System.out.println("2) Generar palindromo de
31                manera manual");
32            System.out.println("3) Generar palindromo de
33                manera cualquiera");
34            System.out.println("4) Salir");
35            System.out.println("Ingrese una opcion");
36            s_opc = scan();
37            try {
38                opc = Integer.parseInt(s_opc);
39            } catch (NumberFormatException ex) {
```



```

33         opc = 0;
34     }
35 } while (!(opc >= 1 && opc <= 4));
36 if (opc >= 1 && opc <= 3) {
37     comand("rm_palindromo.txt");
38     comand("echo_>>_palindromo.txt");
39     comand("rm_derivaciones.txt");
40     comand("echo_>>_derivaciones.txt");
41     try {
42         PrintWriter writer_palindromo
43             = new PrintWriter("palindromo.txt", "UTF-8");
44         PrintWriter writer_derivaciones
45             = new PrintWriter("derivaciones.txt", "UTF-8");
46         if (opc == 1) {
47             c = 0;
48             int length = 0;
49             boolean esPalindromo = true;
50             boolean epsilon;
51             BufferedReader br
52                 = new BufferedReader(new
53                     InputStreamReader(System.
54                         in));
55             while (c != '\n') {
56                 try {
57                     c = (char) br.read();
58                     if (c != '\n') {
59                         cadena += c;
60                         length++;
61                     }
62                 } catch (IOException ex) {
63                     ex.printStackTrace();
64                 }
65             }
66             for (int i = 0, l = length - 1; i <
67                 length; i++, l--) {
68                 if (cadena.charAt(i) != cadena.
69                     charAt(l)) {
70                     esPalindromo = false;
71                 }
72             }
73             if (esPalindromo) {
74                 if (length % 2 == 0) {
75                     epsilon = true;
76                 } else {

```

```

73         epsilon = false;
74     }
75     cadena = cadena.substring((length
76         + 1) / 2);
77     System.out.println(ANSI_GREEN + "
78     Es palindromo" + ANSI_GREEN);
79     String palindromo = "";
80     if (epsilon) {
81         palindromo += 'E' + "";
82         writer_palindromo.println("E"
83             );
84     }
85     for (int i = 0; i < cadena.length
86         (); i++) {
87         c = cadena.charAt(i);
88         switch (palindromo) {
89             case "":
90                 if (c != 'E') {
91                     writer_derivaciones
92                         .println("P_>
93                             _b");
94                 }
95                 palindromo = c + "";
96                 break;
97             case "E":
98                 writer_derivaciones.
99                     println("P_>_E");
100                 palindromo = c + " +
101                     c;
102                 writer_derivaciones.
103                     println("P_>_aPa"
104                         );
105                 break;
106             default:
107                 writer_derivaciones.
108                     println("P_>_aPa"
109                         );
110                 palindromo = c +
111                     palindromo + c;
112                 break;
113         }
114         writer_palindromo.println(
115             palindromo);
116     }
117 } else {
118     cadena = "";

```

```

105         System.out.println(ANSI_RED + "No
           es palindromo" + ANSI_RED);
106     }
107 }
108 if (opc == 2) {
109     c = 0;
110     BufferedReader br
111         = new BufferedReader(new
           InputStreamReader(System.
           in));
112     while (c != '\n') {
113         try {
114             c = (char) br.read();
115             if (c != '\n') {
116                 switch (cadena) {
117                     case "":
118                         if (c != 'E') {
119                             writer_derivaciones
                               .println("
                               P->b");
120                         }
121                         cadena = c + "";
122                         break;
123                     case "E":
124                         writer_derivaciones
                               .println("P->
                               aPa");
125                         cadena = c + " +
                               c;
126                         writer_derivaciones
                               .println("P->
                               aPa");
127                         break;
128                     default:
129                         writer_derivaciones
                               .println("P->
                               aPa");
130                         cadena = c +
                               cadena + c;
131                         break;
132                 }
133                 writer_palindromo.println
                     (cadena);
134             }
135         } catch (IOException ex) {
136             ex.printStackTrace();

```

```

137         }
138     }
139 }
140 if (opc == 3) {
141     Random rand = new Random();
142     int random = rand.nextInt(10);
143     for (int i = 0; i < random; i++) {
144         c = random_alphabet();
145         switch (cadena) {
146             case "":
147                 if (c != 'E') {
148                     writer_derivaciones.
149                         println("P_>b");
150                 }
151                 cadena = c + "";
152                 break;
153             case "E":
154                 writer_derivaciones.
155                     println("P_>E");
156                 cadena = c + " " + c;
157                 writer_derivaciones.
158                     println("P_>aPa");
159                 break;
160             default:
161                 writer_derivaciones.
162                     println("P_>aPa");
163                 cadena = c + cadena + c;
164                 break;
165         }
166         writer_palindromo.println(cadena)
167         ;
168         System.out.print(c);
169     }
170     System.out.println();
171 }
172 }
173 }
174
175 public static char random_alphabet() {
176     Random rand = new Random();
177     return ((char) (rand.nextInt(26) + 97));

```

```

178     }
179
180     public static void comand(String cmd) {
181         try {
182             Process p = Runtime.getRuntime().exec(cmd);
183             BufferedReader stdInput
184                 = new BufferedReader(new
185                     InputStreamReader(p.getInputStream()
186                         ));
187         } catch (IOException ex) {
188             ex.printStackTrace();
189         }
190     }
191
192     public static String scan() {
193         String scan = "";
194         char c = 0;
195         BufferedReader br
196             = new BufferedReader(new
197                 InputStreamReader(System.in));
198         while (c != '\n') {
199             try {
200                 c = (char) br.read();
201                 if (c != '\n') {
202                     scan += c;
203                 }
204             } catch (IOException ex) {
205                 ex.printStackTrace();
206             }
207         }
208         return scan;
209     }
210 }

```

Su ejecución es:

```

emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p3$ java Palindromo
Palindromo
P-> aPa | b | E
1)Verificar si una cadena cualquiera es palindromo
2)Generar palindromo de manera manual
3)Generar palindromo de manera cualquiera
4)Salir
Ingrese una opcion
1
osobaboso
Es palindromo
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p3$

```

Su primer salida despliega el número de derivaciones y cómo se derivó la gramática para hacer el palíndromo:

```
1 P -> b
2 P -> aPa
3 P -> aPa
4 P -> aPa
5 P -> aPa
```

Su segunda salida despliega cómo esta generándose la cadena recursivamente de acuerdo a la gramática:

```
1 a
2 bab
3 obabo
4 sobabos
5 osobaboso
```

4. Gramática libre de contexto para balancear parentésis

Lo siguiente es un programa que verifica si una expresión está balanceada de paréntesis, la gramática libre de contexto que verifica esta expresión está dada por:

$$B \rightarrow (RB | \varepsilon$$

$$R \rightarrow) | (RR$$

```
1 import java.io.*;
2 import java.util.Random;
3
4 public class Gramatica {
5
6     public static final String ANSI_RESET = "\u001B[0m";
7     public static final String ANSI_BLACK = "\u001B[30m";
8     public static final String ANSI_RED = "\u001B[31m";
9     public static final String ANSI_GREEN = "\u001B[32m";
10    public static final String ANSI_YELLOW = "\u001B[33m"
11    ;
12    public static final String ANSI_BLUE = "\u001B[34m";
13    public static final String ANSI_PURPLE = "\u001B[35m"
14    ;
15    public static final String ANSI_CYAN = "\u001B[36m";
16    public static final String ANSI_WHITE = "\u001B[37m";
17
18    public static void main(String[] args) {
19        Analizador analizador;
20        String s_opc;
21        char c;
```

```

20     int opc = 0;
21     do {
22         System.out.println("Gramatica de balanceo de
                parenthesis");
23         System.out.println("B->(RB|E");
24         System.out.println("R->)|_(RR");
25         System.out.println("1) Generar cadena de
                manera manual");
26         System.out.println("2) Generar cadena de
                manera aleatoria");
27         System.out.println("3) Salir");
28         System.out.println("Ingrese una opcion");
29         s_opc = scan();
30         try {
31             opc = Integer.parseInt(s_opc);
32         } catch (NumberFormatException ex) {
33             opc = 0;
34         }
35     } while (!(opc >= 1 && opc <= 3));
36     if (opc >= 1 && opc <= 2) {
37         analizador = new Analizador();
38         c = 0;
39         comand("rm gramatica.txt");
40         comand("echo >> gramatica.txt");
41         try {
42             PrintWriter writer
43                 = new PrintWriter("gramatica.txt"
44                                     , "UTF-8");
45             writer.println("B");
46             if (opc == 1) {
47                 BufferedReader br
48                     = new BufferedReader(new
49                                         InputStreamReader(System.
50                                             in));
51
52                 do {
53                     try {
54                         c = (char) br.read();
55                         analizador.analizar(c);
56                         writer.println(analizador.
57                                     getDerivacion());
58                     } catch (IOException ex) {
59                         ex.printStackTrace();
60                     }
61                 } while (c != '\n');
62             }
63         }
64         if (opc == 2) {

```

```

59         while (c != '\n') {
60             c = random_parenthesis();
61             analizador.analizar(c);
62             System.out.print(c);
63             writer.println(analizador.
                           getDerivacion());
64         }
65         System.out.println();
66     }
67     if (analizador.isBalanceado()) {
68         System.out.println(ANSI_GREEN + "Esta
69             balanceado" + ANSI_GREEN);
70         writer.println("Esta balanceado");
71     } else {
72         System.out.println(ANSI_RED + "No
73             esta balanceado" + ANSI_RED);
74         writer.println("No esta balanceado");
75     }
76     writer.println("");
77     writer.close();
78 } catch (IOException ex) {
79     ex.printStackTrace();
80 }
81
82 public static char random_parenthesis() {
83     Random rand = new Random();
84     char character = 0;
85     int random = rand.nextInt(9);
86     if (random == 5) {
87         character = '\n';
88     } else {
89         character = (char) (rand.nextInt(2) + 40);
90     }
91     return character;
92 }
93
94 public static void comand(String cmd) {
95     try {
96         Process p = Runtime.getRuntime().exec(cmd);
97         BufferedReader stdInput
98             = new BufferedReader(new
99                 InputStreamReader(p.getInputStream()
100 ));
101     } catch (IOException ex) {

```



```

100         ex.printStackTrace();
101     }
102 }
103
104 public static String scan() {
105     String scan = "";
106     char c = 0;
107     BufferedReader br
108         = new BufferedReader(new
109             InputStreamReader(System.in));
110     while (c != '\n') {
111         try {
112             c = (char) br.read();
113             if (c != '\n') {
114                 scan += c;
115             }
116         } catch (IOException ex) {
117             ex.printStackTrace();
118         }
119     }
120     return scan;
121 }

```

```

1 public class Analizador {
2     private boolean balanceado;
3     private String derivacion;
4     private int index;
5
6     public Analizador(){
7         balanceado=true;
8         derivacion="B";
9         index=0;
10    }
11
12    public boolean isBalanceado() {
13        return balanceado;
14    }
15
16    public String getDerivacion() {
17        return derivacion;
18    }
19
20
21    public void analizar(char c) {
22        try{

```

```

23     String derivada="";
24     switch(derivacion.charAt(index)){
25         case 'B':
26             switch(c){
27                 case '(':
28                     derivada="(RB";
29                     balanceado=false;
30                     break;
31                 case '\\n':
32                     derivada="";
33                     balanceado=true;
34                     break;
35                 default:
36                     balanceado=false;
37                     break;
38             }
39             break;
40         case 'R':
41             switch(c){
42                 case '(':
43                     derivada="(RR";
44                     balanceado=false;
45                     break;
46                 case ')':
47                     derivada=")";
48                     balanceado=true;
49                     break;
50                 default:
51                     balanceado=false;
52                     break;
53             }
54             break;
55         default:
56             balanceado=false;
57             break;
58     }
59     derivacion=derivacion.substring(0, index)+
60     derivada+derivacion.substring(index
        +1);
61     index++;
62 } catch (Exception e){
63 }
64 }
65 }

```

Su ejecución es:

```

emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$ javac *.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$ java Gramatica
Gramatica de balanceo de parentesis
B-> (RB | E
R-> ) | (RR
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
((()())
Esta balanceado
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$

```

Su salida muestra línea por línea las derivaciones que se hicieron para analizar el balanceo de paréntesis:

```

1 B
2 (RB
3 ((RRB
4 (()RB
5 (() )B
6 (() ) (RB
7 (() ) ()B
8 (() ) ()
9 Esta balanceado

```

5. Máquina de Turing

Este programa está hecho en base a la regla 110, considerando la máquina $M = \{Q_M, \sum_M, \Gamma, \delta, q_0, B\}$ capaz de emular el comportamiento del autómata celular elemental conocido como Regla 110. $Q_M = \{S_{x0}, S_{01}, S_{11}, S_B\}$ es el conjunto de estados de la máquina, $\sum_M = \{0, 1\}$ es el alfabeto de entrada, $\Gamma = \{0, 1, B\}$ es el alfabeto de la cinta, δ es la función de transición mostrada a continuación:

State	0	1	B
S_{x0}	$S_{x0}, 0, R$	$S_{01}, 1, R$	$S_B, 0, L$
S_{01}	$S_{x0}, 1, R$	$S_{11}, 1, R$	
S_{11}	$S_{x0}, 1, R$	$S_{11}, 0, R$	
S_B	$S_B, 0, L$	$S_B, 1, L$	$S_{x0}, 0, R$

Donde $q_0 = S_{x0}$ es el estado inicial de la máquina y B es el símbolo en blanco. La Máquina de Turing está codificada como:

```

1 import java.awt.*;
2 import javax.swing.*;
3
4 public class Turingmachine extends JPanel {
5
6     private int n;
7     private int index;

```

```

8      private String cadena;
9      private int x_rect;
10     private int y_rect;
11     private int y_space;
12     private int x_space;
13
14     public Turingmachine(String s, int i) {
15         cadena =s;
16         n =cadena.length();
17         index=i;
18         x_rect = 40;
19         y_rect = 30;
20         y_space = 80;
21         x_space = 120;
22     }
23
24     public void setCadena(String cadena) {
25         this.cadena = cadena;
26     }
27
28
29     @Override
30     protected void paintComponent(Graphics g) {
31         super.paintComponent(g);
32         Graphics2D g2 = (Graphics2D) g;
33         g2.setColor( Color.BLACK);
34         g2.drawRect(x_space + 50+40*(index), y_rect +
35             y_space, x_rect, y_rect);
36         g2.setColor( Color.ORANGE);
37         g2.fillRect(x_space + 51+40*(index), y_rect + 1
38             + y_space, x_rect - 1, y_rect - 1);
39         g2.setColor( Color.BLACK);
40         g2.drawString("q",x_space + 50+17+40*(index),
41             y_rect + y_space+20);
42         g2.drawLine(x_space+72+40*(index), y_space+60,
43             x_space+72+40*(index), y_space+80);
44         g2.fillPolygon(new int [] { x_space+72+40*(index),
45             x_space+76+40*(index), x_space+68+40*(index) },
46             new int [] { y_space+90, y_space+80,
47                 y_space+80}, 3);
48         g2.drawLine(x_space, y_space+90, x_space+n
49             *40+180, y_space+90);
50         g2.drawLine(x_space, y_space+120, x_space+n
51             *40+180, y_space+120);
52         g2.drawString(".",x_space+40+17, y_rect + y_space
53             +80);

```

```

45         g2.drawString(".", x_space+50+17, y_rect + y_space
46             +80);
47         g2.drawString(".", x_space+60+17, y_rect + y_space
48             +80);
49         g2.drawLine(x_space+40+50, y_space+90, x_space
50             +40+50, y_space+120);
51         for (int i = 0; i < n; i++) {
52             //Contorno del espacio de la pila
53             g2.setColor(Color.BLACK);
54             g2.drawString(cadena.charAt(i)+",", x_space
55                 +40*(i+1)+50+17, y_rect + y_space+80);
56             g2.drawLine(x_space+40*(i+2)+50, y_space+90,
57                 x_space+40*(i+2)+50, y_space+120);
58         }
59         g2.drawString(".", x_space+40*(n+2)+17, y_rect +
60             y_space+80);
61         g2.drawString(".", x_space+40*(n+2)+27, y_rect +
62             y_space+80);
63         g2.drawString(".", x_space+40*(n+2)+37, y_rect +
64             y_space+80);
65     }
66 }

1  import java.awt.*;
2  import java.awt.event.ActionEvent;
3  import java.awt.event.ActionListener;
4  import javax.swing.*;
5
6  public class AnimacionTM extends JFrame implements
7      ActionListener {
8
9      private JScrollPane scroll;
10     private JPanel panel;
11
12     public AnimacionTM() {
13         setTitle("Animacion");
14         setLocation(200, 50);
15         setSize(800, 500);
16         setDefaultCloseOperation(WindowConstants.
17             EXIT_ON_CLOSE);
18         setLayout(null);
19         scroll = new JScrollPane();
20         scroll.setBounds(0, 0, getWidth(), getHeight());
21         scroll.setVerticalScrollBarPolicy(JScrollPane.

```

```

                VERTICAL_SCROLLBAR_NEVER);
20         scroll.setVerticalScrollBar().
            addAdjustmentListener(this);
21         add(scroll);
22     }
23
24     public void animar(String s, int i) {
25         panel = new Turingmachine(s,i);
26         panel.setPreferredSize(new Dimension(s.length() *
            50+200, 500));
27         scroll.setViewportViewView(panel);
28     }
29
30     @Override
31     public void adjustmentValueChanged(AdjustmentEvent ae
        ) {
32         //animar();
33         setVisible(true);
34     }
35 }

```

```

1  public class MaquinaTuring {
2
3      private String estado;
4      private char character;
5      private char movimiento;
6      public MaquinaTuring(){
7          estado="Sx0";
8      }
9
10     public String getEstado() {
11         return estado;
12     }
13
14     public char getCaracter() {
15         return character;
16     }
17
18     public char getMovimiento() {
19         return movimiento;
20     }
21
22     public void analizar(char entrada) {
23         switch (entrada) {
24             case '0':
25                 switch (estado) {

```

```

26         case "Sx0":
27             estado = "Sx0";
28             caracter='0';
29             movimiento='R';
30             break;
31         case "S01":
32             estado = "Sx0";
33             caracter='1';
34             movimiento='R';
35             break;
36         case "S11":
37             estado = "Sx0";
38             caracter='1';
39             movimiento='R';
40             break;
41         case "SB":
42             estado = "SB";
43             caracter='0';
44             movimiento='L';
45             break;
46     }
47     break;
48 case '1':
49     switch (estado) {
50         case "Sx0":
51             estado = "S01";
52             caracter='1';
53             movimiento='R';
54             break;
55         case "S01":
56             estado = "S11";
57             caracter='1';
58             movimiento='R';
59             break;
60         case "S11":
61             estado = "S11";
62             caracter='0';
63             movimiento='R';
64             break;
65         case "SB":
66             estado = "SB";
67             caracter='1';
68             movimiento='L';
69             break;
70     }
71     break;

```

```

72         case 'B':
73             switch (estado) {
74                 case "Sx0":
75                     estado = "SB";
76                     caracter='0';
77                     movimiento='L';
78                     break;
79                 case "SB":
80                     estado = "Sx0";
81                     caracter='0';
82                     movimiento='R';
83                     break;
84             }
85             break;
86     }
87 }
88
89 }

1 import java.io.*;
2 import java.util.Random;
3
4 public class MainTuring {
5     public static void main(String [] args) {
6         AnimacionTM animacion;
7         MaquinaTuring maquina;
8         String s_opc;
9         String cadena;
10        String cadena_turing;
11        char c;
12        int index;
13        int opc;
14        do {
15            System.out.println("Gramatica de balanceo de
16                               parentesis");
17            System.out.println("1) Generar cadena de
18                               manera manual");
19            System.out.println("2) Generar cadena de
20                               manera aleatoria");
21            System.out.println("3) Salir");
22            System.out.println("Ingrese una opcion");
23            s_opc = scan();
24            try {
                opc = Integer.parseInt(s_opc);
            } catch (NumberFormatException ex) {
                opc = 0;
            }
        } while (opc != 3);
    }
}

```



```

25     }
26 } while (!(opc >= 1 && opc <= 3));
27 if (opc >= 1 && opc <= 2) {
28     cadena_turing="";
29     cadena="";
30     animacion = new AnimacionTM();
31     index=1;
32     animacion.animar("B",index);
33     animacion.setVisible(true);
34     maquina = new MaquinaTuring();
35     c = 0;
36     comand("rm_turing.txt");
37     comand("echo >>_turing.txt");
38     try {
39         PrintWriter writer
40             = new PrintWriter("turing.txt", "
41             UTF-8");
42         writer.print("{_Sx0_");
43         if (opc == 1) {
44             BufferedReader br
45                 = new BufferedReader(new
46                     InputStreamReader(System.
47                         in));
48             while (c != '\n') {
49                 try {
50                     c = (char) br.read();
51                     cadena+=c+"";
52                     maquina.analizar(c);
53                     cadena_turing+=maquina.
54                         getCaracter()+"";
55                     if(maquina.getMovimiento()=='
56                         R'){
57                         index++;
58                     }else if(maquina.
59                         getMovimiento()=='L'){
60                         index--;
61                     }
62                     writer.print(maquina.
63                         getEstado()+"_");
64                     animacion.animar(cadena,index
65                         -1);
66                     animacion.setVisible(true);
67                     esperar(1f);
68                 } catch (IOException ex) {
69                     ex.printStackTrace();
70                 }

```

```

63         }
64         animacion.setVisible(false);
65     }
66     if (opc == 2) {
67         while (c != '\n') {
68             c = random_01B();
69             System.out.print(c);
70             cadena+=c+" ";
71             maquina.analizar(c);
72             cadena_turing+=maquina.
                getCaracter()+" ";
73             if(maquina.getMovimiento()=='R'){
74                 index++;
75             } else if (maquina.getMovimiento()
                =='L'){
76                 index--;
77             }
78             writer.print(maquina.getEstado()+
                "□");
79             animacion.animar(cadena,index-1);
80             animacion.setVisible(true);
81             esperar(1f);
82         }
83         System.out.println();
84         animacion.setVisible(false);
85     }
86     writer.println("{}");
87     writer.close();
88 } catch (IOException ex) {
89     ex.printStackTrace();
90 }
91 }
92 System.exit(0);
93 }
94
95 public static char random_01B() {
96     Random rand = new Random();
97     char character;
98     int random=rand.nextInt(10);
99     if(random==9){
100         character='\n';
101     } else if (random<5){
102         character='B';
103     } else{
104         character=(char) (rand.nextInt(2) + 48);
105     }

```

```

106         return character;
107     }
108
109     public static void comand(String cmd) {
110         try {
111             Process p = Runtime.getRuntime().exec(cmd);
112             BufferedReader stdInput
113                 = new BufferedReader(new
114                     InputStreamReader(p.getInputStream
115                         ()));
116         } catch (java.io.IOException ex) {
117             ex.printStackTrace();
118         }
119     }
120
121     public static String scan() {
122         String scan = "";
123         char c = 0;
124         BufferedReader br
125             = new BufferedReader(new
126                 InputStreamReader(System.in));
127         while (c != '\n') {
128             try {
129                 c = (char) br.read();
130                 if (c != '\n') {
131                     scan += c;
132                 }
133             } catch (IOException ex) {
134                 ex.printStackTrace();
135             }
136         }
137         return scan;
138     }
139
140     public static void esperar(float s) {
141         try {
142             Thread.sleep((int) (s * 1000));
143         } catch (InterruptedException ex) {
144             ex.printStackTrace();
145         }
146     }

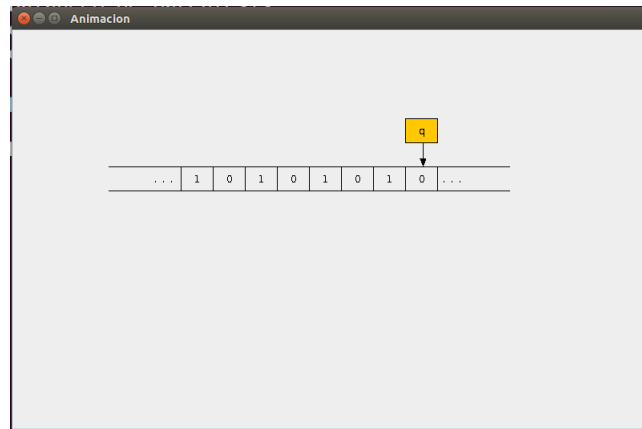
```

Su ejecución es:

```

emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p5$ java MainTuring
Gramatica de balanceo de parentesis
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
10101010010B0B010B

```



Su salida es:

1 { Sx0 SB SB SB }