Luis Ricardo Reyes Villar 21070343

```
1   import java.awt.Color;
2   import java.util.ArrayList;
3   import java.util.HashMap;
4   import java.util.LinkedHashSet;
5   import java.util.logging.Level;
6   import java.util.logging.Logger;
7   public class FrameEvaluacion extends javax.swing.JFrame {
8       public FrameEvaluacion() {
9           initComponents();
10      }
11      @SuppressWarnings("unchecked")
12      Generated Code
        private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
88          // TODO add your handling code here:
89          txtaGramatica.setText(t: "");
90          txtaResultados.setText(t: "");
91      }
        private void btnEvaluarActionPerformed(java.awt.event.ActionEvent evt) {
93          try {
94              txtaResultados.setForeground(fg: Color.black);
95              String inicial = "";
96              String gramatica = txtaGramatica.getText();
97              String resultados = "";
98              if (!gramatica.equals(anObject: "")) {
99                  char c = gramatica.charAt(index: 0);
100                 if (c < 65 || c > 90) {
101                     txtaResultados.setForeground(fg: Color.red);
102                     txtaResultados.setText(t: "Introduzca una grámatica válida.");
103                 } else {
104                     LinkedHashSet<String> v = new LinkedHashSet<>();
105                     int cont1 = -1;
106                     int cont2 = 0;
107                     String s = "";
108                     for (int i = 0; i < gramatica.length(); i++) {
109                         c = gramatica.charAt(index: i);
110                         s += c;
111                         int res = cont2 - cont1;
112                         if (c == 8594 && res == 1) {
113                             s = s.substring(beginIndex: 0, s.length() - 1);
114                             s = s.replace(target: " ", replacement:"");
115                             v.add(e: s);
116                             cont1 = cont2;
117                         } else if (c == 10) {
118                             s = "";
119                             cont2++;
120                         } else if (c == 8594 && res != 1) {
121                             txtaResultados.setForeground(fg: Color.red);
122                             txtaResultados.setText(t: "Introduzca una grámatica válida.");
123                             return;
```

```
                    }
                }
                Object[] vTemp = v.toArray();
                inicial = (String) vTemp[0];
                s = "";
                HashMap<String, ArrayList<String>>[] cadenas = new HashMap[v.size()];
                for (int i = 0; i < cadenas.length; i++) {
                    cadenas[i] = new HashMap<>();
                }
                ArrayList<String> temp = new ArrayList<>();
                for (int i = 0, cont = 0; i < gramatica.length(); i++) {
                    c = gramatica.charAt(index: i);
                    s += c;
                    if (c == 8594) {
                        s = "";
                    } else if (c == '|') {
                        s = s.substring(beginIndex: 0, s.length() - 1).replace(target: " ", replacement:"");
                        temp.add(e: s);
                        s = "";
                    } else if (c == 10) {
                        s = s.substring(beginIndex: 0, s.length() - 1).replace(target: " ", replacement:"");
                        temp.add(e: s);
                        cadenas[cont].put((String) vTemp[cont], value: temp);
                        temp = new ArrayList<>();
                        cont++;
                    } else if (i == gramatica.length() - 1) {
                        s = s.replace(target: " ", replacement:"");
                        temp.add(e: s);
                        cadenas[cont].put((String) vTemp[cont], value: temp);
                        temp = new ArrayList<>();
                        cont++;
                    }
                }
                LinkedHashSet<String> t = new LinkedHashSet<>();
                s = "";
                for (int i = 0; i < vTemp.length; i++) {
                    HashMap<String, ArrayList<String>> hashTemporal = cadenas[i];
                    ArrayList<String> aux = hashTemporal.get((String) vTemp[i]);
                    boolean band = false;
                    for (int j = 0; j < aux.size(); j++) {
                        s = aux.get(index: j);
                        if (s.equals(anObject: s.toLowerCase())) {
                            if (s.length() == 1) {
                                t.add(e: s);
                            } else {
                                for (int l = 0; l < s.length(); l++) {
                                    boolean boo = t.add(e: String.valueOf(c: s.charAt(index: l)));
                                    if (boo) {
```

```
                                t.remove(o: String.valueOf(c: s.charAt(index: l)));
                            } else {
                                band = true;
                                break;
                            }
                        }
                        if (!band) {
                            t.add(e: s);
                        }
                    }
                } else {
                    char caracter;
                    for (int k = 0; k < s.length(); k++) {
                        caracter = s.charAt(index: k);
                        if (caracter < 65 || caracter > 90) {
                            band = false;
                            for (int l = 0; l < vTemp.length; l++) {
                                String variableTemporal = (String) vTemp[l];
                                if (variableTemporal.length() > l) {
                                    for (int m = 0; m < variableTemporal.length(); m++) {
                                        if (caracter == variableTemporal.charAt(index: m)) {
                                            band = true;
                                            break;
                                        }
                                    }
                                }
                                if (band) {
                                    break;
                                }
                            }
                            if (!band) {
                                t.add(e: String.valueOf(c: caracter));
                            }
                        } else {
                            v.add(e: String.valueOf(c: caracter));
                        }
                    }
                }
            }
        }
        LinkedHashSet<String> nuevo = new LinkedHashSet<>();
        LinkedHashSet<String> anterior = new LinkedHashSet<>();
        LinkedHashSet<String> union = new LinkedHashSet<>();
        String nuevoC = null;
        String anteriorC = nuevoC;
        String unionC = getConjuntoUnion(t, anteriorC);
        if (nuevoC == null && anteriorC == null) {
            nuevoC = anteriorC = String.valueOf(obj:"{}");
```

```
                }
                nuevo.add(e: nuevoC);
                anterior.add(e: anteriorC);
                union.add(e: unionC);
                nuevoC = getVariables(v: vTemp, cadenas, union: unionC);
                while (!nuevoC.equals(anObject: anteriorC)) {
                    anteriorC = nuevoC;
                    unionC = getConjuntoUnion(t, anteriorC);
                    nuevo.add(e: nuevoC);
                    anterior.add(e: anteriorC);
                    union.add(e: unionC);
                    nuevoC = getVariables(v: vTemp, cadenas, union: unionC);
                }
                Object[] nuevoV = nuevo.toArray();
                Object[] anteriorV = anterior.toArray();
                Object[] unionSV = union.toArray();
                resultados += "V = " + v.toString() + "\n";
                resultados += "T = " + t.toString() + "\n";
                resultados += "S = " + inicial + "\n\n";
                resultados += String.format(format: "%25s", args: "Nuevo");
                resultados += String.format(format: "%25s", args: "Anterior");
                resultados += String.format(format: "%25s", args: "T U Anterior") + "\n\n";
                for (int i = 0; i < nuevo.size(); i++) {
                    resultados += String.format(format: "%25s", nuevoV[i]);
                    resultados += String.format(format: "%25s", anteriorV[i]);
                    resultados += String.format(format: "%25s", unionSV[i]) + "\n";
                }
                ArrayList<String> simbolos = new ArrayList<>();
                for (int i = 0; i < nuevoC.length(); i++) {
                    if (nuevoC.charAt(index: i) != ' '
                            && nuevoC.charAt(index: i) != ','
                            && nuevoC.charAt(index: i) != '['
                            && nuevoC.charAt(index: i) != ']') {
                        simbolos.add(nuevoC.charAt(index: i) + "");
                    }
                }
                boolean band = false;
                for (int i = 0; i < simbolos.size(); i++) {
                    if (inicial.equals(anObject: simbolos.get(index: i))) {
                        band = true;
                        break;
                    }
                }
                if (band) {
                    resultados += "L(G) <> {}";
                } else {
                    resultados += "L(G) = {}";
                }
```

```
                txtaResultados.setText(t: resultados);
            }
        } else {
            txtaResultados.setForeground(fg: Color.red);
            txtaResultados.setText(t: "Introduzca una grámatica válida.");
        }
    } catch (Exception e) {
        txtaResultados.setForeground(fg: Color.red);
        txtaResultados.setText(t: "Introduzca una grámatica válida.");
        Logger.getLogger(name: FrameEvaluacion.class.getName()).log(level: Level.SEVERE, msg:null, thrown: e
    }
}
private String getConjuntoUnion(LinkedHashSet<String> t, String anteriorC) {
    LinkedHashSet<String> res = new LinkedHashSet<>();
    Object[] temp = t.toArray();
    for (int i = 0; i < temp.length; i++) {
        res.add((String) temp[i]);
    }
    if (anteriorC != null) {
        String s = "";
        for (int i = 0; i < anteriorC.length(); i++) {
            char c = anteriorC.charAt(index: i);
            s += c;
            if (c == ',' || c == ']') {
                s = s.substring(beginIndex: 1, s.length() - 1);
                res.add(e: s);
                s = "";
            }
        }
    }
    return res.toString();
}
private String getVariables(Object[] v, HashMap<String, ArrayList<String>>[] cadenas, String union) {
    LinkedHashSet<String> temp = new LinkedHashSet<>();
    ArrayList<String> simbolos = new ArrayList<>();
    for (int i = 0; i < union.length(); i++) {
        if (union.charAt(index: i) != ' '
                && union.charAt(index: i) != ','
                && union.charAt(index: i) != '['
                && union.charAt(index: i) != ']') {
            simbolos.add(union.charAt(index: i) + "");
        }
    }
    for (int i = 0; i < v.length; i++) {
        HashMap<String, ArrayList<String>> temporal = cadenas[i];
        ArrayList<String> cad = temporal.get((String) v[i]);
        for (int j = 0; j < cad.size(); j++) {
            String s = cad.get(index: j);
            boolean[] band = new boolean[s.length()];
```

```
                for (int k = 0; k < band.length; k++) {
                    band[k] = false;
                }
                for (int k = 0; k < s.length(); k++) {
                    for (int l = 0; l < simbolos.size(); l++) {
                        if (String.valueOf(c: s.charAt(index: k)).equals(anObject: simbolos.get(index: l))) {
                            band[k] = true;
                            break;
                        }
                    }
                }
                boolean boo = band[0];
                for (int k = 0; k < s.length(); k++) {
                    boo = boo && band[k];
                }
                if (boo) {
                    temp.add((String) v[i]);
                }
            }
        }
        return temp.toString();
    }
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FrameEvaluacion().setVisible(b: true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton btnEvaluar;
    private javax.swing.JButton btnLimpiar;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea txtaGramatica;
    private javax.swing.JTextArea txtaResultados;
    // End of variables declaration
}
```