ANEXO 4

CODIGO DEL ACTOR CUANTIFICADOR LOGARÍTMICO

A continuación se presenta el código generado en la construcción del actor cuantificador logarítmico y que es parte de la aplicación propuesta.

```
/* Este actor implementa un cuantificador logaritmico

*/

package ptolemy.actor.lib; // Paquete en el se incluye el nuevo
                              actor, el cual es polimorfico.

import ptolemy.actor.*;
import ptolemy.kernel.CompositeEntity;
import ptolemy.kernel.util.*;
import ptolemy.data.*;
import ptolemy.data.type.ArrayType;
import ptolemy.data.type.BaseType;
import ptolemy.data.expr.Parameter;


public class CuantificadorLogaritmico extends Transformer {

/* Hereda la clase Transformer, la cual es una clase básica de
Ptolemy II que transforma un flujo de datos de entrada en un flujo
de datos de salida.
Proporciona un puerto de entrada y un puerto de salida que son
instancias de la clase TypedIOPort.
*/
```

```
public CuantificadorLogaritmico(CompositeEntity container,
        String name) throws NameDuplicationException,
        IllegalActionException  {

    super(container, name);

    niveles = new Parameter(this, "niveles");
    niveles.setExpression("{128.0, 129.0, 130.0, 131.0,
    132.0, 133.0, 134.0, 135.0, 136.0, 137.0, 138.0, 139.0,
    140.0, 141.0, 142.0, 143.0, 144.0, 145.0, 146.0, 147.0,
    148.0, 149.0, 150.0, 151.0, 152.0, 153.0, 154.0, 155.0,
    156.0, 157.0, 158.0, 160.0, 161.0, 162.0, 163.0, 164.0,
    165.0, 166.0, 167.0, 168.0, 169.0, 170.0, 171.0, 172.0,
    173.0, 174.0, 175.0, 176.0, 177.0, 178.0, 179.0, 180.0,
    181.0, 182.0, 183.0, 184.0, 185.0, 186.0, 187.0, 188.0,
    189.0, 190.0, 191.0, 192.0, 193.0, 194.0, 195.0, 196.0,
    197.0, 198.0, 199.0, 200.0, 201.0, 202.0, 203.0, 204.0,
    205.0, 206.0, 207.0, 208.0, 209.0, 210.0, 211.0, 212.0,
    213.0, 214.0, 215.0, 216.0, 217.0, 218.0, 219.0, 220.0,
    221.0, 222.0, 223.0, 224.0, 225.0, 226.0, 227.0, 228.0,
    229.0, 230.0, 231.0, 232.0, 233.0, 234.0, 235.0, 236.0,
    237.0, 238.0, 239.0, 240.0, 241.0, 242.0, 243.0, 244.0,
    245.0, 246.0, 247.0, 248.0, 249.0, 250.0, 251.0, 252.0,
    253.0, 254.0, 255.0}");

    niveles.setTypeEquals(new ArrayType(BaseType.DOUBLE));

    umbrales = new Parameter(this, "umbrales");
    umbrales.setExpression("{0.5, 2.5, 4.5, 6.5, 8.5, 10.5,
    12.5, 14.5, 16.5, 18.5, 20.5, 22.5, 24.5, 26.5, 28.5,
    30.5, 34.5, 38.5, 42.5, 46.5, 50.5, 54.5, 58.5, 62.5,
    66.5, 70.5, 74.5, 78.5, 82.5, 86.5, 90.5, 94.5, 102.5,
    110.5, 118.5, 126.5, 134.5, 142.5, 150.5, 158.5, 166.5,
    174.5, 182.5, 190.5, 198.5, 206.5, 214.5, 222.5, 238.5,
    254.5, 270.5, 286.5, 302.5, 318.5, 334.5, 350.5, 366.5,
    382.5, 398.5, 414.5, 430.5, 446.5, 462.5, 478.5, 510.5,
    542.5, 574.5, 606.5, 638.5, 670.5, 702.5, 734.5, 766.5,
    798.5, 830.5, 862.5, 894.5, 926.5, 958.5, 990.5, 1054.5,
    1118.5, 1182.5, 1246.5, 1310.5, 1374.5, 1438.5, 1502.5,
    1566.5, 1630.5, 1694.5, 1758.5, 1822.5, 1886.5, 1950.5,
```

```
            2014.5, 2142.5, 2270.5, 2398.5, 2526.5, 2654.5, 2782.5,
            2910.5, 3038.5, 3166.5, 3294.5, 3422.5, 3550.5, 3678.5,
            3806.5, 3934.5, 4062.5, 4318.5, 4574.5, 4830.5, 5086.5,
            5342.5, 5598.5, 5854.5, 6110.5, 6366.5, 6622.5, 6878.5,
            7134.5, 7390.5, 7646.5, 7902.5, 8158.5}");

            umbrales.setTypeEquals(new ArrayType(BaseType.DOUBLE));

            input.setTypeEquals(BaseType.DOUBLE);
            output.setTypeEquals(BaseType.DOUBLE);
      }

////////////////////////////////////////////////////////////////
////                     puertos y parametros                ////

      public Parameter niveles;
      public Parameter umbrales;

      /* la clase Parameter de la cual la variable niveles es una
      instancia  permite que las variables que sean instancias de
      ella sean visibles al usuario, para que se puedan modificar.
      */

////////////////////////////////////////////////////////////////
///////                          metodos públicos            /////

     /** Saca el nivel de cuantización de acuerdo a los umbrales.
      Si no hay una entrada, entonces no hay salida. La excepción
      IllegalActionException se lanza si no hay director.
      */

     public void fire() throws IllegalActionException {

      if (input.hasToken(0)) {
             double in = ((DoubleToken)input.get(0)).doubleValue();
             int indice = _getIndiceCuantizacion(in);
            output.send(0,
           ((ArrayToken)niveles.getToken()).getElement(indice));
         }
     }
```

```
/////////////////////////////////////////////////////////////////
///////                          métodos privados            /////


    /* Calcula el segmento de salida de acuerdo a la entrada.
     */


     private int _getIndiceCuantizacion(double in)throws
               IllegalActionException  {

          ArrayToken valorUmbrales =
               (ArrayToken)umbrales.getToken();
          double[] _umbrales = new double[valorUmbrales.length()];
          for (int j = 0; j < valorUmbrales.length(); j++){
          _umbrales[j] =
               ((DoubleToken)valorUmbrales.getElement(j))
               .doubleValue();
          }
          int indice = _umbrales.length;
          for (int i = 0; i < _umbrales.length; i++) {
               if (in < _umbrales[i] && in >= _umbrales[0]) {
                    indice = i - 1;
                    break;
               } else if (in == _umbrales[i]) {
                    indice = i;
                    break;
               } else if (in <= _umbrales[0]) {
                    indice = 0;
                    break;
               }
          }
          return indice;
     }


}
```
