```java
package classes;

/**
 * created by lgcaobianco on 2018-04-29
 */

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * created by lgcaobianco on 2018-04-14
 */

public class LeitorPontosEntrada {

    private String nomeArquivo;
    private String formato;
    private String separadorValor;

    private String getNomeArquivo() {
        return nomeArquivo;
    }

    private String getFormato() {
        return formato;
    }

    private String getSeparadorValor() {
        return separadorValor;
    }

    public  LeitorPontosEntrada(String nomeArquivo, String formato) {
        this.nomeArquivo = nomeArquivo;
        this.formato = formato;
        switch (formato) {
            case ".csv":
                this.separadorValor = ",";
                break;
            case ".txt":
                this.separadorValor = " ";
                break;
            default:
                System.out.println("Formato ainda não suportado");
                System.exit(1);
```

```
49                  break;
50
51              }
52          }
53
54
55
56          public List<Double[]> extrairPontos() {
57              List<Double[]> matrizPontos = new ArrayList<Double
    []>();
58              String linhaLida = "";
59              BufferedReader stream = null;
60              try {
61                  stream = new BufferedReader(new FileReader(
    getNomeArquivo() + getFormato()));
62                  while ((linhaLida = stream.readLine()) != null
    ) {
63                      String[] temporario = linhaLida.split(
    getSeparadorValor());
64                      Double[] numerosSeparados = new Double[
    temporario.length];
65                      for (int i = 0; i < temporario.length; i++
    ) {
66                          numerosSeparados[i] = Double.
    parseDouble(temporario[i]);
67                      }
68                      matrizPontos.add(numerosSeparados);
69
70                  }
71
72              } catch (IOException e) {
73                  e.printStackTrace();
74                  System.exit(1);
75              } finally {
76                  if (stream != null) {
77                      try {
78                          stream.close();
79                      } catch (IOException e) {
80                          e.printStackTrace();
81                          System.exit(1);
82                      }
83                  }
84              }
85              return matrizPontos;
86          }
87
88          public int lerInputTerminal() {
89              Scanner reader = new Scanner(System.in);  //
    Reading from System.in
90              return reader.nextInt(); // Scans the next token
```

```
 90 of the input as an int.
 91
 92     }
 93
 94 }
 95
```

```java
package classes;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

/**
 * created by lgcaobianco on 2018-04-29
 */

public class MultiLayerPerceptron {

    private Double[][] W1, W1Inicial, W2, W2Inicial, deltaCamada1;
    private List<Double[]> matrizInputs, matrizOperacao;
    private Double[][] I1;
    private Double I2 = 0.0, Y2;
    private Double deltaCamada2 = 0.0;
    private double taxaAprendizagem = 0.1;
    private Double[][] Y1;

    // O construtor irá efetuar as operações essenciais para o funcionamento dos
    // métodos.
    public MultiLayerPerceptron() {
        LeitorPontosEntrada leitor = new LeitorPontosEntrada(
                "/home/lgcaobianco/repositorios/epc-rna/epc4/src/base/conjunto-treinamento", ".csv");
        this.matrizInputs = leitor.extrairPontos();

        leitor = new LeitorPontosEntrada("/home/lgcaobianco/repositorios/epc-rna/epc4/src/base/conjunto-operacao",
                ".csv");
        this.matrizOperacao = leitor.extrairPontos();
        leitor = null; // para leitor se tornar candidato ao garbage collector

        Random random = new Random();

        W2Inicial = W2 = new Double[10][1];
        deltaCamada1 = I1 = new Double[10][1];
        Y1 = new Double[11][1];
        W1Inicial = W1 = new Double[10][4];

        for (int i = 0; i < I1.length; i++) {
            I1[i][0] = 0.0;
            deltaCamada1[i][0] = 0.0;
        }
```

```
44
45          // sorteia W1
46          for (int i = 0; i < W1.length; i++) {
47              for (int j = 0; j < W1[i].length; j++) {
48                  W1Inicial[i][j] = W1[i][j] = random.
    nextDouble();
49              }
50          }
51
52          // sorteia W2
53          for (int i = 0; i < W2.length; i++) {
54              W2[i][0] = W2Inicial[i][0] = random.nextDouble
    ();
55          }
56      }
57
58      public List<Double[]> getMatrizInputs() {
59          return matrizInputs;
60      }
61
62      public void setMatrizInputs(List<Double[]>
    matrizInputs) {
63          this.matrizInputs = matrizInputs;
64      }
65
66      public List<Double[]> getMatrizOperacao() {
67          return matrizOperacao;
68      }
69
70      public void setMatrizOperacao(List<Double[]>
    matrizOperacao) {
71          this.matrizOperacao = matrizOperacao;
72      }
73
74
75      public void imprimirY2() {
76          System.out.println(Y2);
77      }
78
79      public void obterI1(int linhaMatrizInput) {
80          for (int i = 0; i < W1.length; i++) {
81              for (int j = 0; j < W1[i].length; j++) {
82                  I1[i][0] += (matrizInputs.get(
    linhaMatrizInput)[j] * W1[i][j]);
83              }
84          }
85      }
86
87      public void obterY1() {
88          Y1[0][0] = -1.0;
```

```
 89          for (int i = 0; i < I1.length; i++) {
 90              Y1[i + 1][0] = 0.5 + 0.5 * Math.tanh((I1[i][0
    ]) / 2);
 91          }
 92      }
 93
 94      public void obterI2() {
 95          I2 = 0.0;
 96          for (int i = 0; i < W2.length; i++) {
 97              I2 += Y1[i][0] * W2[i][0];
 98          }
 99      }
100
101      public void obterY2() {
102          Y2 = 0.0;
103          Y2 = 0.5 + 0.5 * Math.tanh(I2 / 2);
104      }
105
106      public void obterDeltaCamada2(int linhaMatrizInput) {
107          // A derivada de g(Ij) pode ser expressa como f(x
    ) * (1 - f(x))!
108          this.deltaCamada2 = (matrizInputs.get(
    linhaMatrizInput)[4] - Y2) * (Y2 * (1 - Y2));
109      }
110
111      public void ajustarPesosCamada2() {
112          for (int i = 0; i < W2.length; i++) {
113              this.W2[i][0] = this.W2[i][0] + (this.
    taxaAprendizagem * this.deltaCamada2 * Y1[i][0]);
114          }
115      }
116
117      public void obterDeltaCamada1() {
118          for (int i = 0; i < deltaCamada1.length; i++) {
119              deltaCamada1[i][0] = deltaCamada2 * W2[i][0]
    * (Y1[1][0] * (1 - Y1[1][0]));
120          }
121      }
122
123      public void ajustarPesosCamada1(int linhaInputMatriz)
     {
124          for (int i = 0; i < W1.length; i++) {
125              for (int j = 0; j < W1[i].length; j++) {
126                  W1[i][j] = W1[i][j] + taxaAprendizagem *
    deltaCamada1[i][0] * matrizInputs.get(linhaInputMatriz)[j
    ];
127              }
128          }
129      }
130
```

```
131     public double calcularErroRelativo(int
    linhaInputMatriz) {
132         double erro = 0.0;
133         erro = (Math.abs((matrizInputs.get(
    linhaInputMatriz)[4] - Y2)) / ((matrizInputs.get(
    linhaInputMatriz)[4]))
134                 * 100);
135         return erro;
136     }
137 
138     public double calcularErroTotal() {
139         double erroTotal = 0;
140         for (int i = 0; i < matrizInputs.size(); i++) {
141             erroTotal += (Math.pow((matrizInputs.get(i)[4
    ] - Y2), 2) / 2);
142         }
143         return (erroTotal / matrizInputs.size());
144     }
145 
146     public void forwardPropagation(int linhaMatrizInput)
    {
147         this.obterI1(linhaMatrizInput);
148         this.obterY1();
149         this.obterI2();
150         this.obterY2();
151     }
152 
153     public void backwardPropagation(int linhaMatrizInput)
     {
154         this.obterDeltaCamada2(linhaMatrizInput);
155         this.ajustarPesosCamada2();
156         this.obterDeltaCamada1();
157         this.ajustarPesosCamada1(linhaMatrizInput);
158 
159     }
160 }
161 
```