```java
package classes;

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 2018-05-19
 */

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);
                break;

        }
    }
```

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

```
90 }
91
```

```java
package classes;

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

/**
 * created by lgcaobianco on 2018-05-19
 */

public class MultilayerPerceptronClassificador {
    private LeitorPontosEntrada leitor;
    private List<Double[]> listaPontosEntrada, classificacaoPontosEntrada, matrizOperacao, classificacaoMatrizOperacao;
    private Double[][] W1;
    private Double[][] W2;
    private Double[][] I1;
    private Double[][] I2;
    private Double[][] W1Inicial;
    private Double[][] W2Inicial;
    private Double[][] Y1;
    private Double[][] Y2;
    private Double[][] Y2Ajustado;
    private Double[][] deltaCamadaEscondida1;
    private Double[][] deltaCamadaSaida;
    private Double taxaAprendizagem = 0.1;

    public List<Double[]> getListaPontosEntrada() {
        return listaPontosEntrada;
    }

    public void setListaPontosEntrada(List<Double[]> listaPontosEntrada) {
        this.listaPontosEntrada = listaPontosEntrada;
    }

    public List<Double[]> getClassificacaoPontosEntrada() {
        return classificacaoPontosEntrada;
    }

    public void setClassificacaoPontosEntrada(List<Double[]> classificacaoPontosEntrada) {
        this.classificacaoPontosEntrada = classificacaoPontosEntrada;
    }

    public List<Double[]> getMatrizOperacao() {
        return matrizOperacao;
    }
```

```
45
46      public void setMatrizOperacao(List<Double[]>
    matrizOperacao) {
47          this.matrizOperacao = matrizOperacao;
48      }
49
50      public List<Double[]> getClassificacaoMatrizOperacao()
     {
51          return classificacaoMatrizOperacao;
52      }
53
54      public void setClassificacaoMatrizOperacao(List<Double
    []> classificacaoMatrizOperacao) {
55          this.classificacaoMatrizOperacao =
    classificacaoMatrizOperacao;
56      }
57
58      public Double[][] getY2() {
59          return Y2;
60      }
61
62      public void setY2(Double[][] y2) {
63          Y2 = y2;
64      }
65
66      public MultilayerPerceptronClassificador() {
67          this.leitor = new LeitorPontosEntrada("/home/
    lgcaobianco/repositorios/epc-rna/epc5/src/base/entrada",
    ".csv");
68          this.listaPontosEntrada = leitor.extrairPontos();
69
70          this.leitor = new LeitorPontosEntrada("/home/
    lgcaobianco/repositorios/epc-rna/epc5/src/base/conjunto-
    operacao",
71                  ".csv");
72          this.matrizOperacao = leitor.extrairPontos();
73
74          this.leitor = new LeitorPontosEntrada("/home/
    lgcaobianco/repositorios/epc-rna/epc5/src/base/
    classificacao-conjunto-operacao",
75                  ".csv");
76          this.classificacaoMatrizOperacao = leitor.
    extrairPontos();
77
78          this.leitor = new LeitorPontosEntrada(
79                  "/home/lgcaobianco/repositorios/epc-rna/
    epc5/src/base/entrada-classificacao", ".csv");
80          this.classificacaoPontosEntrada = leitor.
    extrairPontos();
81          leitor = null;
```

```
 82
 83         this.deltaCamadaEscondida1 = this.I1 = new Double
    [15][1];
 84
 85         for (int i = 0; i < 15; i++) {
 86             I1[i][0] = 0.0;
 87         }
 88
 89         this.W1 = this.W1Inicial = new Double[15][5];
 90         Random random = new Random();
 91
 92         for (int i = 0; i < W1.length; i++) {
 93             for (int j = 0; j < W1[i].length; j++) {
 94                 this.W1[i][j] = this.W1Inicial[i][j] =
    random.nextDouble();
 95             }
 96         }
 97
 98         Y1 = new Double[16][1];
 99         for (int i = 0; i < Y1.length; i++) {
100             Y1[i][0] = 0.0;
101         }
102
103         this.W2 = this.W2Inicial = new Double[3][16];
104
105         this.deltaCamadaSaida = this.Y2Ajustado = this.Y2
     = this.I2 = new Double[3][1];
106         for (int i = 0; i < deltaCamadaSaida.length; i++)
     {
107             deltaCamadaSaida[i][0] = Y2Ajustado[i][0] =
    Y2[i][0] = I2[i][0] = 0.0;
108         }
109
110         for (int i = 0; i < W2.length; i++) {
111             for (int j = 0; j < W2[i].length; j++) {
112                 this.W2[i][j] = this.W2Inicial[i][j] =
    random.nextDouble();
113             }
114         }
115         System.out.println("Fim da construcao. Tamanho do
     cjt de entrada: " + this.listaPontosEntrada.size()
116                 + " linhas e " + this.listaPontosEntrada.
    get(0).length + "  colunas");
117         System.out.println("I1 tem: " + I1.length + "
    linhas e : " + I1[0].length + " colunas");
118         System.out.println("w1 tem: " + W1.length + "
    linhas e : " + W1[0].length + " colunas");
119
120     }
121
```

```
122     public void imprimirMatrizI1() {
123         for (int i = 0; i < I1.length; i++) {
124             for (int j = 0; j < I1[i].length; j++) {
125                 System.out.print(this.I1[i][j] + ",");
126             }
127             System.out.println();
128         }
129     }
130 
131     public void imprimirMatrizW1() {
132         for (int i = 0; i < W1.length; i++) {
133             for (int j = 0; j < W1[i].length; j++) {
134                 System.out.print(this.W1[i][j] + ",");
135             }
136             System.out.println();
137         }
138 
139         System.out.println("\n\n\n");
140     }
141 
142     public void inicializarI1() {
143         for (int i = 0; i < I1.length; i++) {
144             I1[i][0] = 0.0;
145         }
146     }
147 
148     public void inicializarI2() {
149         for (int i = 0; i < I2.length; i++) {
150             I2[i][0] = 0.0;
151         }
152     }
153 
154     public void obterI1(int linhaMatrizEntrada) {
155         inicializarI1();
156         for (int i = 0; i < I1.length; i++) {
157             for (int j = 0; j < W1[i].length; j++) {
158                 this.I1[i][0] += (W1[i][j] *
    listaPontosEntrada.get(linhaMatrizEntrada)[j]);
159 
160             }
161         }
162     }
163 
164     public void obterY1() {
165         Y1[0][0] = -1.0;
166         for (int i = 1; i < Y1.length - 1; i++) {
167             this.Y1[i][0] = 0.5 + 0.5 * Math.tanh((I1[i -
     1][0]) / 2);
168         }
169     }
```

```
170
171     public void obterI2() {
172         inicializarI2();
173         for (int i = 0; i < W2.length; i++) {
174             for (int j = 0; j < W2[i].length; j++) {
175                 this.I2[i][0] += this.W2[i][j] * this.Y1[
    j][0];
176             }
177         }
178     }
179
180     public void obterY2() {
181         for (int i = 0; i < I2.length; i++) {
182             this.Y2[i][0] = 0.5 + 0.5 * Math.tanh((I2[i][
    0]) / 2);
183         }
184     }
185
186     public void obterDeltaCamadaSaida(int
    linhaMatrizEntrada) {
187         for (int i = 0; i < deltaCamadaSaida.length; i++)
     {
188             this.deltaCamadaSaida[i][0] = (this.
    classificacaoPontosEntrada.get(linhaMatrizEntrada)[i] -
    Y2[i][0])
189                     * (this.Y2[i][0] * (1 - this.Y2[i][0]
    ));
190
191         }
192     }
193
194     public void ajustarPesosCamada2() {
195         for (int i = 0; i < W2.length; i++) {
196             for (int j = 0; j < W2[i].length; j++) {
197                 this.W2[i][j] = W2[i][j] + (this.
    taxaAprendizagem * this.deltaCamadaSaida[i][0] * Y1[j][0]
    );
198             }
199         }
200     }
201
202     public void obterDeltaCamadaEscondida() {
203         Double[] aux = new Double[15];
204         for (int i = 0; i < aux.length; i++) {
205             aux[i] = 0.0;
206         }
207         for (int i = 0; i < W2[0].length - 1; i++) {
208             for (int j = 0; j < W2.length; j++) {
209                 aux[i] += deltaCamadaSaida[j][0] * W2[j][
    i];
```

```
210             }
211         }
212         for (int i = 0; i < deltaCamadaEscondida1.length;
     i++) {
213             deltaCamadaEscondida1[i][0] = aux[i] * Y1[i +
     1][0];
214         }
215     }
216 
217     public void ajustarPesosCamada1(int
    linhaMatrizEntrada) {
218         for (int i = 0; i < W1.length; i++) {
219             for (int j = 0; +j < W1[i].length; j++) {
220                 W1[i][j] = W1[i][j] + taxaAprendizagem *
    deltaCamadaEscondida1[i][0]
221                         * listaPontosEntrada.get(
    linhaMatrizEntrada)[j];
222             }
223         }
224     }
225 
226     public double calcularEk(int linhaMatrizEntrada) {
227         double erro = 0.0;
228 
229         for (int i = 0; i < classificacaoPontosEntrada.
    get(0).length; i++) {
230             erro += Math.pow((classificacaoPontosEntrada.
    get(linhaMatrizEntrada)[i] - Y2[i][0]), 2);
231         }
232         return erro / 2;
233     }
234 
235     public double calcularEm() {
236         double erroTotal = 0.0;
237         for (int i = 0; i < classificacaoPontosEntrada.
    size(); i++) {
238             erroTotal += calcularEk(i);
239         }
240         return (erroTotal / classificacaoPontosEntrada.
    size());
241     }
242 
243     public void forwardPropagation(int linhaMatrizEntrada
    ) {
244         this.obterI1(linhaMatrizEntrada);
245         this.obterY1();
246         this.obterI2();
247         this.obterY2();
248     }
249 
```

```
250     public void backwardPropagation(int
    linhaMatrizEntrada) {
251         this.obterDeltaCamadaSaida(linhaMatrizEntrada);
252         this.ajustarPesosCamada2();
253         this.obterDeltaCamadaEscondida();
254         this.ajustarPesosCamada1(linhaMatrizEntrada);
255     }
256 
257 }
```