```java
package classes;

import java.util.List;

/**
 * created by lgcaobianco on 2018-06-16
 */

public class Printer {
    public static void imprimirQualquer(double[][] object) {
        for (int i = 0; i < object.length; i++) {
            for (int j = 0; j < object[i].length; j++) {
                System.out.print(object[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println("\n\n");
    }

    public static void imprimirQualquer(double[] object) {
        for (int i = 0; i < object.length; i++) {
            System.out.print(object[i] + " ");
        }
        System.out.println("\n\n");
    }

    public static void imprimirQualquer(List<Double[]> object) {
        for (int i = 0; i < object.size(); i++) {
            for (int j = 0; j < object.get(i).length; j++) {
                System.out.print(object.get(i)[j] + " ");
            }
            System.out.println();
        }
        System.out.println("\n\n");
    }

}

```

```java
package classes;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.List;

/**
 * created by lgcaobianco on 2018-06-16
 */

public class Hopfield {
    private double[][] vetoresZ;
    private double[][] w;
    private double[][] sinalRuidoso;
    private double[] vetorSaida;
    private double[] vetorSaidaAnterior;
    private List<Double[]> valoresEntrada;
    private double[][] matrizIdentidade;
    private int p = 4;
    private int n = 45;
    private double eta = 0.01;

    public Hopfield() {
        vetoresZ = new double[p][n];
        matrizIdentidade = new double[n][n];
        w = new double[n][n];
        vetorSaida = new double[n];
        vetorSaidaAnterior = new double[n];

        // obter a matriz de vetores Z
        LeitorPontosEntrada leitor = new
LeitorPontosEntrada(
                "/home/lgcaobianco/repositorios/epc-rna/
epc9/src/base/padrao1", ".csv");
        valoresEntrada = leitor.extrairPontos();
        Printer.imprimirQualquer(valoresEntrada);
        for (int i = 0; i < p; i++) {
            for (int j = 0; j < n; j++) {
                vetoresZ[i][j] = valoresEntrada.get(i)[j];
                vetorSaidaAnterior[j] = 0;
            }
        }

        // obter o sinal ruidoso
        leitor = new LeitorPontosEntrada("/home/
lgcaobianco/repositorios/epc-rna/epc9/src/base/
sinalRuidoso", ".csv");
        valoresEntrada = leitor.extrairPontos();
        sinalRuidoso = new double[valoresEntrada.size()][
```

```
46 valoresEntrada.get(0).length];
47         for (int i = 0; i < valoresEntrada.size(); i++) {
48             for (int j = 0; j < valoresEntrada.get(i).
   length; j++) {
49                 sinalRuidoso[i][j] = valoresEntrada.get(i)
   [j];
50             }
51         }
52 
53         Printer.imprimirQualquer(sinalRuidoso);
54 
55         // obter a matriz identidade
56         for (int i = 0; i < matrizIdentidade.length; i++)
   {
57             for (int j = 0; j < matrizIdentidade[i].length
   ; j++) {
58                 if (i == j) {
59                     matrizIdentidade[i][j] = 1;
60                     continue;
61                 }
62                 matrizIdentidade[i][j] = 0;
63             }
64         }
65         // obter a matriz W
66         double somatorio = 0;
67         for (int i = 0; i < w.length; i++) {
68             for (int j = 0; j < w[i].length; j++) {
69                 for (int k = 0; k < p; k++) {
70                     somatorio += (vetoresZ[k][i] *
   vetoresZ[k][j]);
71                 }
72                 w[i][j] = (somatorio / n) - ((p / n) *
   matrizIdentidade[i][j]);
73                 somatorio = 0;
74             }
75         }
76 
77         Printer.imprimirQualquer(w);
78 
79         Printer.imprimirQualquer(matrizIdentidade);
80     }
81 
82     public void obterVetorSaida() {
83         double somatorio = 0;
84         for (int i = 0; i < w.length; i++) {
85             for (int j = 0; j < w[i].length; j++) {
86                 somatorio += (w[i][j] * sinalRuidoso[0][j]
   );
87             }
88             vetorSaida[i] = funcaoG(somatorio);
```

```
 89              somatorio = 0;
 90          }
 91          Printer.imprimirQualquer(vetorSaida);
 92      }
 93
 94      public double funcaoG(double saidaRede) {
 95          if (saidaRede > 0) {
 96              return 1;
 97          }
 98          return -1;
 99      }
100
101      public void obterVetorSaidaAnterior() {
102          for (int i = 0; i < sinalRuidoso.length; i++) {
103              vetorSaidaAnterior[i] = sinalRuidoso[0][i];
104              sinalRuidoso[0][i] = vetorSaida[i];
105          }
106      }
107
108      public double obterErro() {
109          double erro = 0;
110          for (int k = 0; k < p; k++) {
111              for (int i = 0; i < vetorSaida.length; i++) {
112                  erro += Math.pow((vetoresZ[k][i] -
    vetorSaida[i]), 2);
113              }
114          }
115
116          return erro;
117      }
118
119      public void obterRespostaRede() {
120          obterVetorSaida();
121          double erro = obterErro();
122          int contadorRepeticoes = 0;
123          while (Math.abs(erro) > 0.1 && contadorRepeticoes
     < 1000) {
124              System.out.println();
125              obterVetorSaidaAnterior();
126              obterVetorSaida();
127              erro = obterErro();
128              contadorRepeticoes++;
129          }
130
131          try {
132              imprimirResultadoEmFormatoTabela();
133          } catch (Exception e) {
134              e.printStackTrace();
135          }
136      }
```

```
137
138     private void imprimirResultadoEmFormatoTabela()
    throws FileNotFoundException,
    UnsupportedEncodingException {
139         PrintWriter writer = new PrintWriter(
140                 "/home/lgcaobianco/repositorios/epc-rna/
    epc9/src/base/imagemRecuperada.csv", "UTF-8");
141         for (int i = 0; i < n; i++) {
142             if(i%5==0) {
143                 writer.append("\n");
144             }
145             writer.append(vetorSaida[i] + ",");
146
147         }
148
149         writer.close();
150     }
151
152 }
153
```

```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 2018-06-16
 */
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;
        }
    }

    public List<Double[]> extrairPontos() {
        List<Double[]> matrizPontos = new ArrayList<Double
```

```
48 []>();
49         String linhaLida = "";
50         BufferedReader stream = null;
51         try {
52             stream = new BufferedReader(new FileReader(
   getNomeArquivo() + getFormato()));
53             while ((linhaLida = stream.readLine()) != null
   ) {
54                 String[] temporario = linhaLida.split(
   getSeparadorValor());
55                 Double[] numerosSeparados = new Double[
   temporario.length];
56                 for (int i = 0; i < temporario.length; i++
   ) {
57                     numerosSeparados[i] = Double.
   parseDouble(temporario[i]);
58                 }
59                 matrizPontos.add(numerosSeparados);
60             }
61 
62         } catch (IOException e) {
63             e.printStackTrace();
64             System.exit(1);
65         } finally {
66             if (stream != null) {
67                 try {
68                     stream.close();
69                 } catch (IOException e) {
70                     e.printStackTrace();
71                     System.exit(1);
72                 }
73             }
74         }
75         return matrizPontos;
76     }
77 
78     public int lerInputTerminal() {
79         Scanner reader = new Scanner(System.in); //
   Reading from System.in
80         return reader.nextInt(); // Scans the next token
   of the input as an int.
81     }
82 }
83 
```