```java
/* Class DotPlot works out the technical aspects of the graph
 * by computing the thresholds, window size and where sequences match.
 * It is also responsible for drawing the points of the Dot-plot.
 */

import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

class DotPlot
{
    // Required attibutes including the Image to be displayed,seq_1, seq_2
    // window and threshold vlaues
    Image img;
    String seq1 = null;
    String seq2 = null;
    int windowSize = 0;
    int threshold = 0;
    Vector vecPoints = new Vector();
    int index = 0;
    int noOfCoors = 0;
    int imageWidth = 1000;
    int imageHeight = 1000;
    int scale = 1000;

    // Initialise the classes attributes
    public void readInGenome(String seq1, String seq2, int windowSize, int threshold)
    {
        this.windowSize = windowSize;
        this.threshold = threshold;
        this.seq1 = seq1;
        this.seq2 = seq2;
    }

    /* Sets up the dimensions for the to be
    *  displayed image depending on whether the sequence
    *  is longer, shoter or equla to the default image dimensions
    */
    public void setDimensions()
    {
        if(seq1.length() == seq2.length())//Both seqs same length
        {
            if(seq1.length() > scale)//Both greater than scale
            {
                imageWidth = scale;
                imageHeight = scale;
            }
            else//Both less than scale
            {
                imageWidth = seq1.length();
                imageHeight = seq2.length();
            }
        }
        else//Different sized seqs
        {
            if(seq1.length() < scale && seq2.length() < scale)//seq_1 = seq_2
            {
                imageWidth = seq1.length();
                imageHeight = seq2.length();
            }
            else if(seq1.length() > seq2.length())//seq_1 > seq_2
            {
                imageWidth = scale;
                float fHeight = (float) ( (float)seq2.length()/(float)seq1.length() ) * (flo
                imageHeight = (int) fHeight;
            }
            else if(seq2.length() > seq1.length())//seq_2 > seq_1
            {
                imageHeight = scale;
                float fWidth = (float) ( (float)seq1.length()/(float)seq2.length() ) * (floa
                imageWidth = (int) fWidth;
            }
        }
    }

```

```
77      /* Calculates if the sequential seegment starting
78      *  at 'a' and ending at 'b' meeets the current
79      *  threshold
80      */
81      public boolean satisfiesThreshold(int a, int b)
82      {
83          int outCount = 0;
84          int matchCount = 0;
85
86          for(int d=1; d<=windowSize; d++)
87          {
88              if(inBounds(a-d,b-d))
89              {
90                  if(isMatch(a-d,b-d))
91                      matchCount++;
92              }
93              else
94                  outCount++;
95
96              if(inBounds(a+d,b+d))
97              {
98                  if(isMatch(a+d,b+d))
99                      matchCount++;
100             }
101             else
102                 outCount++;
103         }
104
105         if(matchCount >= threshold)
106             return true;
107         else
108             return false;
109     }
110
111     /* Check if the base indexes match
112     *  True or False is returned.
113     */
114     public boolean isMatch(int a, int b)
115     {
116         String base1 = seq1.substring(a,a+1);
117         String base2 = seq2.substring(b,b+1);
118         if(base1.equals(base2))
119             return true;
120         else return false;
121     }
122
123     /* This makes sure we don't run over bounds with our
124     *  windowing.
125     *  True or False is returned.
126     */
127     public boolean inBounds(int a, int b)
128     {
129         if(a < 0 || b < 0 || a > seq1.length()-1 || b > seq2.length()-1)
130             return false;
131         else
132             return true;
133     }
134
135     /* Uses java.Grphics to draw the points
136     *  tothe screen using the fore-mentioned
137     *  integers a and b
138     */
139     public void drawPoint(int a, int b, Graphics g)
140     {
141         int xcoor = 0;
142         int ycoor = 0;
143
144         if(seq1.length() > imageWidth)
145         {
146             float xx = (float) ( (float)a/(float)seq1.length() * (float)imageWidth );
147             xcoor = (int) xx;
148         }
149         else
150             xcoor = a;
151
152         if(seq2.length() > imageHeight)
```

```
153         {
154             float yy = (float) ( (float)b/(float)seq2.length() * (float)imageHeight );
155             ycoor = (int)yy;
156         }
157         else
158             ycoor = b;
159 
160         g.drawLine(xcoor,ycoor,xcoor,ycoor);
161     }
162     /* Uses java.Graphics to set the points
163     *  for the Gragh. this is somewhat of a culmination
164     *  of methods.
165     */
166     public void setUpPoints(Graphics g)
167     {
168         String base1 = null;
169         String base2 = null;
170 
171         for(int i=0; i<seq1.length(); i++)
172         {
173             for(int j=0; j<seq2.length(); j++)
174             {
175                 base1 = seq1.substring(i,i+1);
176                 base2 = seq2.substring(j,j+1);
177                 if(base1.equals(base2))
178                 {
179                     if(satisfiesThreshold(i,j))
180                         drawPoint(i,j,g);
181                 }
182             }
183         }
184     }
185 
186     public void draw(Component c)
187     {
188         img = c.createImage(imageWidth,imageHeight);
189         Graphics g = img.getGraphics();
190         g.setColor(new Color(38,0,133));
191         setUpPoints(g);
192     }
193 }
```