```java
/*  COM1003 Java Programming
    Autumn Semester 2018-9
    Programming Assignment 2
    Written by: Simonas Petkevicius
    Mole Username: acc17sp
    Registration No: 180170308
    Written on: 20/11/18
*/

import sheffield.*;

public class Dragon {

    public static void main(String[] args) {

        // Set scalling variable, window dimensions, and original picture dimensions
        final int SCALLING = 3;
        final int WINDOW_WIDTH = 188;
        final int WINDOW_HEIGHT = 130;
        final int PICTURE_WIDTH = 130;
        final int PICTURE_HEIGHT = 188;

        EasyReader fileInput = new EasyReader("dragon.txt");

        // Creating a scalled graphics window
        EasyGraphics picture = new EasyGraphics(WINDOW_WIDTH * SCALLING, WINDOW_HEIGHT * SCALLING);

        // creating character array for storing the encrypted picture version
        char[][] letter = new char[PICTURE_HEIGHT][PICTURE_WIDTH];

        // Read characters from the file one row at a time
        for (int i = 0; i < letter.length; i++) {
            for (int j = 0; j < letter[i].length; j++) {

                letter[i][j] = fileInput.readChar();
            }
        }

        // Plots the drawing on to the canvas
        for (int i = 0; i < letter.length; i++) {
```

```
41              for (int j = 0; j < letter[i].length; j++) {
42
43                  // if the number is even then it's represented area is coloured green for the dragon
44                  if ((int) letter[i][j] % 2 == 0) {
45
46                      picture.setColor(0, 255, 0);
47                      picture.fillRectangle(i * SCALLING, j * SCALLING, SCALLING, SCALLING);
48                  }
49                  // if the number is odd, then it's represented area is coloured black
50                  else {
51
52                      picture.setColor(0, 0, 0);
53                      picture.fillRectangle(i * SCALLING, j * SCALLING, SCALLING, SCALLING);
54                  }
55              }
56          }
57
58          // minimum and maximum fire burst duration
59          int minFireTime = 1;
60          int maxFireTime = 5;
61
62          // minimum and maximum wait duration between fire bursts
63          int minWaitTime = 1;
64          int maxWaitTime = 3;
65
66          // produces 3 randomly timed fire bursts
67          for (int k = 0; k < 3; k++) {
68
69              // waits between 1 and 3 sec for the fire burst to appear
70              picture.waitSeconds(minWaitTime +
71                                  (int) (Math.random() * ((maxWaitTime - minWaitTime) + 1)));
72
73              // plots the fire burst on the canvas
74              for (int i = 0; i < letter.length; i++) {
75                  for (int j = 0; j < letter[i].length; j++) {
76
77                      /* if the Unicode number is odd and divisible by 3
78                         then number's represented area will be plotted red to make fire appear*/
79                      if (((int) letter[i][j] % 3 == 0) && ((int) letter[i][j] % 2 != 0)) {
80
```

```
                            picture.setColor(255, 0, 0);
                            picture.fillRectangle(i * SCALLING, j * SCALLING, SCALLING, SCALLING);
                        }
                    }
                }

                // waits between 1 and 5 sec for the fire burst to disappear
                picture.waitSeconds(minFireTime +
                                    (int) (Math.random() * ((maxFireTime - minFireTime) + 1)));

                // fire burst is plotted with black and it disappears
                for (int i = 0; i < letter.length; i++) {
                    for (int j = 0; j < letter[i].length; j++) {

                        /* if the Unicode number is odd and divisible by 3
                           then number's represented area will be plotted
                           black to make the fire disappear */
                        if (((int) letter[i][j] % 3 == 0) && ((int) letter[i][j] % 2 != 0)) {

                            picture.setColor(0, 0, 0);
                            picture.fillRectangle(i * SCALLING, j * SCALLING, SCALLING, SCALLING);
                        }
                    }
                }
            }
        }
    }
```