Assignment-2

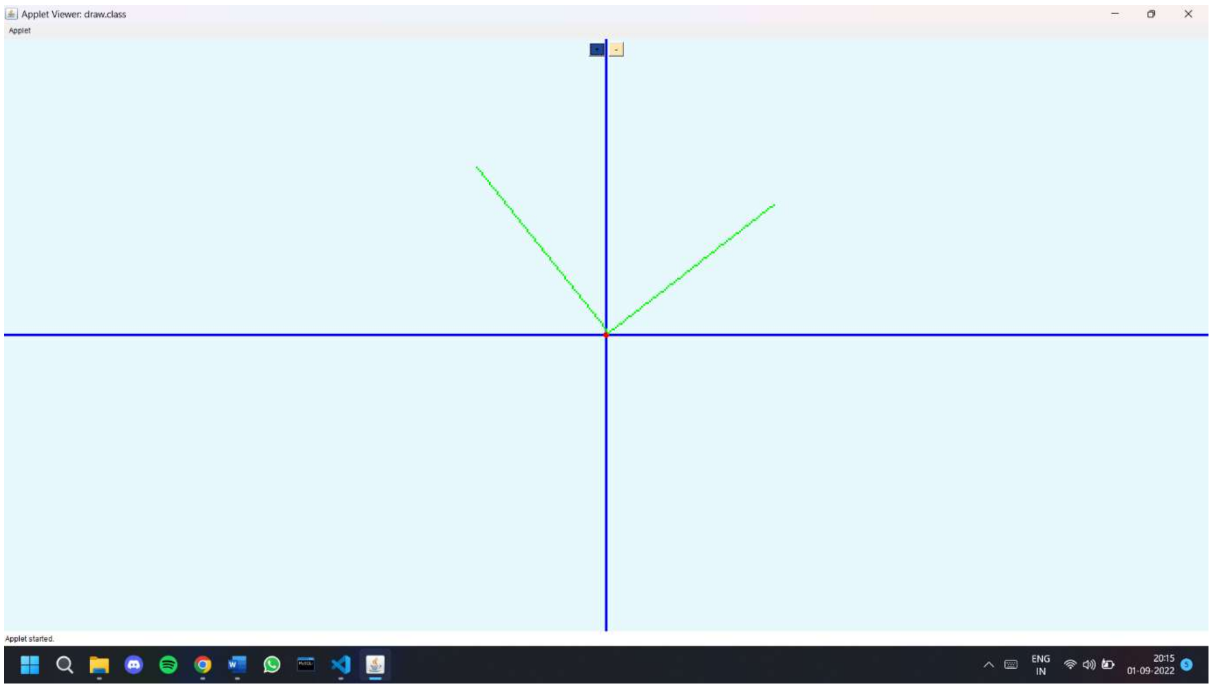
COMPUTER GRAPHICS LAB

Subhajit Samanta
2020CSB046

- Draw straight line using the following line drawing methods keeping the same grid structure in order to view resolution for each case.

  i) DDA

```
 import java.applet.*;

import java.awt.*;
import java.awt.event.*;
public class draw extends Applet implements ActionListener, MouseWheelListener
{

  int originX, originY;
  int height, width;
  int gap = 20;
  Button b1 = new Button(" + ");
  Button b2 = new Button(" - ");

  public void init() {
    setBackground(new Color(232, 249, 253));
    b1.setBackground(new Color(31, 70, 144));
    b2.setBackground(new Color(255, 229, 180));
    add(b1);
    add(b2);
    addMouseWheelListener(this);
    b1.addActionListener(this);
    b2.addActionListener(this);
  }

  public void paint(Graphics g) {
    g.setColor(Color.BLACK);
    height = getHeight();
    width = getWidth();
    originX = (getX() + width) / 2;
    originY = (getY() + height) / 2;


    //drawGrid(g);
    drawXaxis(g);
    drawYaxis(g);
    drawOriginCircle(g);
    DDALine(g, 1, 1, -70, 90);
    DDALine(g, 1, 1, 90, 70);

  }

  //Function to draw origin
  public void drawOriginCircle(Graphics g) {
    g.setColor(Color.RED);
    g.fillOval(originX - 5, originY - 5, 10, 10);
  }

  //Function for plotting points
  public void plotPoint(Graphics g, int x, int y, Color c) {
    g.setColor(c);
    g.fillRect(
      originX + (x * gap) - gap / 2,
      originY - (y * gap) - gap / 2,
      gap,
      gap
    );
  }

  //Function to draw X-axis
  public void drawXaxis(Graphics g) {
    g.setColor(Color.BLUE);
    g.fillRect(0, originY - 2, width, 4);
  }

  //Function to draw Y-axis
  public void drawYaxis(Graphics g) {
    g.setColor(Color.BLUE);
    g.fillRect(originX - 2, 0, 4, height);
  }

  // Function to draw the Grid
  public void drawGrid(Graphics g) {
    drawHorizontalLines(g);
    drawVeritcalLines(g);
  }

  //Function to draw the horizontal lines of the grid
  public void drawHorizontalLines(Graphics g) {
    g.setColor(Color.YELLOW);
    for (int i = originX; i <= width; i += gap) {
      g.drawLine(i, 0, i, height);
    }
    for (int i = originX; i >= 0; i -= gap) {
      g.drawLine(i, 0, i, height);
    }
  }

  //Function to draw the vertical lines of the grid
  public void drawVeritcalLines(Graphics g) {
    g.setColor(Color.YELLOW);
    for (int i = originY; i <= height; i += gap) {
      g.drawLine(0, i, width, i);
      // add coordinate text
    }
    for (int i = originY; i >= 0; i -= gap) {
      g.drawLine(0, i, width, i);
    }
  }

  //Function for the buttons
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == b1) zoom(10);
    if (e.getSource() == b2) zoom(-10);
  }

  //Function for the mousewheel
  public void mouseWheelMoved(MouseWheelEvent e) {
    int z = e.getWheelRotation();
    zoom(z);
  }

  //Function for the zoom in feature
  public void zoom(int i) {
    if (gap + i >= 1 && gap + i <= 300) {
      gap += i;
      repaint();
    }
  }

  int round(float n) {
    if (n - (int) n < 0.5) return (int) n;
    return (int) (n + 1);
  }

  void DDALine(Graphics g, int x0, int y0, int x1, int y1) {
    int dx = (x1 - x0);
    int dy = (y1 - y0);

    int step;
    if (Math.abs(dx) > Math.abs(dy)) {
      step = Math.abs(dx);
    } else {
      step = Math.abs(dy);
    }

    float x_incr = (float) dx / step;
    float y_incr = (float) dy / step;
    float x = (float)x0;
    float y = (float)y0;

    for (int i = 0; i < step; i++) {
      plotPoint(g, round(x), round(y), Color.green);
      x += x_incr;
      y += y_incr;
    }
  }
}
```



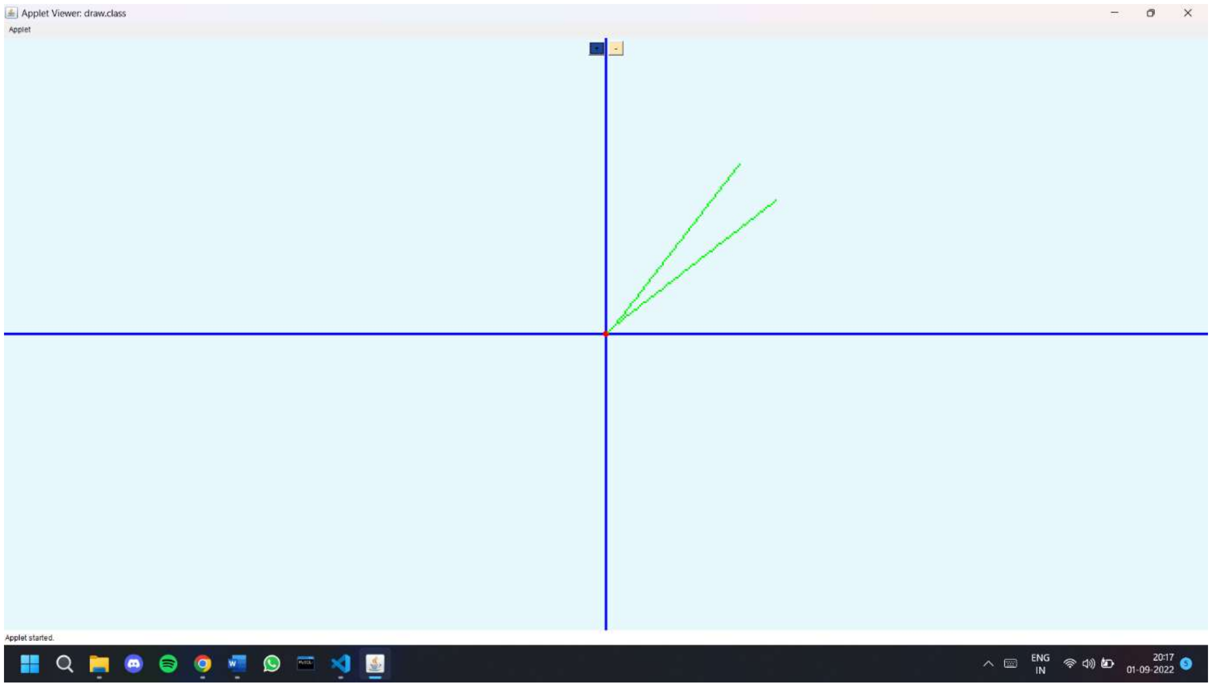
ii) Bresenham’s

```
 import java.applet.*;

import java.awt.*;
import java.awt.event.*;

public class draw extends Applet implements ActionListener, MouseWheelListener
{

  int originX, originY;
  int height, width;
  int gap = 20;
  Button b1 = new Button(" + ");
  Button b2 = new Button(" - ");

  public void init() {
    setBackground(new Color(232, 249, 253));
    b1.setBackground(new Color(31, 70, 144));
    b2.setBackground(new Color(255, 229, 180));
    add(b1);
    add(b2);
    addMouseWheelListener(this);
    b1.addActionListener(this);
    b2.addActionListener(this);
  }

  public void paint(Graphics g) {
    g.setColor(Color.BLACK);
    height = getHeight();
    width = getWidth();
    originX = (getX() + width) / 2;
    originY = (getY() + height) / 2;
    //drawGrid(g);
    drawXaxis(g);
    drawYaxis(g);
    drawOriginCircle(g);

    bresenham(g, 1, 1, 90, 70);
    bresenham(g, 1, 1, 70, 90);
    //bresenham(g,-100,-100,10,0);
  }

  //Function to draw origin
  public void drawOriginCircle(Graphics g) {
    g.setColor(Color.RED);
    g.fillOval(originX - 5, originY - 5, 10, 10);
  }
```

```
//Function for plotting points
public void plotPoint(Graphics g, int x, int y, Color c) {
  g.setColor(c);
  g.fillRect(
    originX + (x * gap) - gap / 2,
    originY - (y * gap) - gap / 2,
    gap,
    gap
  );
}

//Function to draw X-axis
public void drawXaxis(Graphics g) {
  g.setColor(Color.BLUE);
  g.fillRect(0, originY - 2, width, 4);
}

//Function to draw Y-axis
public void drawYaxis(Graphics g) {
  g.setColor(Color.BLUE);
  g.fillRect(originX - 2, 0, 4, height);
}

// Function to draw the Grid
public void drawGrid(Graphics g) {
  drawHorizontalLines(g);
  drawVeritcalLines(g);
}

//Function to draw the horizontal lines of the grid
public void drawHorizontalLines(Graphics g) {
  g.setColor(Color.YELLOW);
  for (int i = originX; i <= width; i += gap) {
    g.drawLine(i, 0, i, height);
  }
  for (int i = originX; i >= 0; i -= gap) {
    g.drawLine(i, 0, i, height);
  }
}

//Function to draw the vertical lines of the grid
public void drawVeritcalLines(Graphics g) {
  g.setColor(Color.YELLOW);
  for (int i = originY; i <= height; i += gap) {
    g.drawLine(0, i, width, i);
    // add coordinate text
  }
```

```
    for (int i = originY; i >= 0; i -= gap) {
      g.drawLine(0, i, width, i);
    }
  }

  //Function for the buttons
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == b1) zoom(10);
    if (e.getSource() == b2) zoom(-10);
  }

  //Function for the mousewheel
  public void mouseWheelMoved(MouseWheelEvent e) {
    int z = e.getWheelRotation();
    zoom(z);
  }

  //Function for the zoom in feature
  public void zoom(int i) {
    if (gap + i >= 1 && gap + i <= 300) {
      gap += i;
      repaint();
    }
  }


  public void bresenham(Graphics g, int x1, int y1, int x2, int y2) {
    int dy = Math.abs(y2 - y1);
    int dx = Math.abs(x2 - x1);


    if (dy <= dx) {
      int p = 2 * dy - dx;
      for (int x = x1, y = y1; x <= x2; x++) {
        plotPoint(g, x, y, Color.green);
        p += 2 * dy;
        if (p >= 0) {
          y++;
          p -= 2 * dx;
        }
      }
    }
    else{
      int p = 2 * dx - dy;
      for (int x = x1, y = y1; y <= y2; y++) {
        plotPoint(g, x, y, Color.green);
        p += 2 * dx;
        if (p >= 0) {
          x++;
```

```
                p -= 2 * dy;
            }
        }
    }
  }
}
```

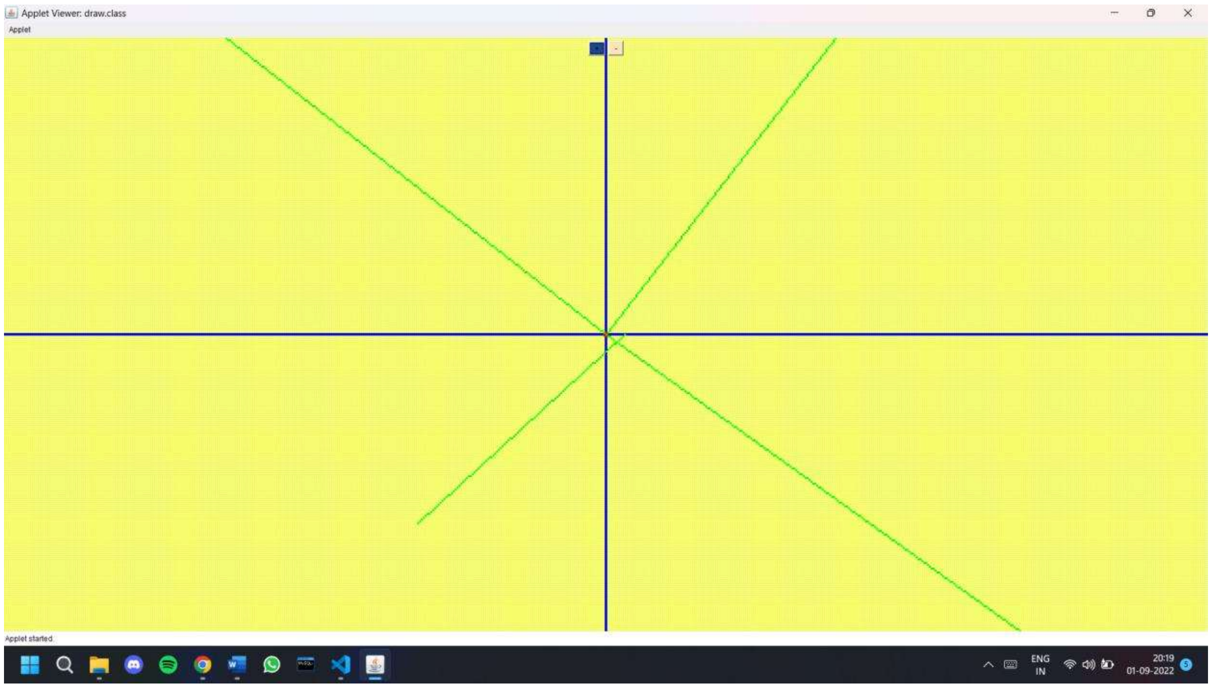

Applet Viewer: draw.class
Applet
+
-
Applet started.
ENG
IN
20:17
01-09-2022

iii) Midpoint

```
 import java.applet.*;

import java.awt.*;
import java.awt.event.*;

public class draw extends Applet implements ActionListener, MouseWheelListener
{

  int originX, originY;
  int height, width;
  int gap = 20;
  Button b1 = new Button(" + ");
  Button b2 = new Button(" - ");

  public void init() {
    setBackground(new Color(232, 249, 253));
    b1.setBackground(new Color(31, 70, 144));
    b2.setBackground(new Color(255, 229, 180));
    add(b1);
    add(b2);
    addMouseWheelListener(this);
    b1.addActionListener(this);
    b2.addActionListener(this);
  }

  public void paint(Graphics g) {
    g.setColor(Color.BLACK);
    height = getHeight();
    width = getWidth();
    originX = (getX() + width) / 2;
    originY = (getY() + height) / 2;
    drawGrid(g);
    drawXaxis(g);
    drawYaxis(g);
    drawOriginCircle(g);

    drawLine(g, 0, 0, 700, 900);
    drawLine(g, 0,0,-900,700);
    drawLine(g, 700,-500,0,0);
    drawLine(g,-100,-100,10,0);
  }

  //Function to draw origin
  public void drawOriginCircle(Graphics g) {
    g.setColor(Color.RED);
    g.fillOval(originX - 5, originY - 5, 10, 10);
  }

  //Function for plotting points
  public void plotPoint(Graphics g, int x, int y, Color c) {
    g.setColor(c);
    g.fillRect(
      originX + (x * gap) - gap / 2,
      originY - (y * gap) - gap / 2,
      gap,
      gap
    );
  }

  //Function to draw X-axis
  public void drawXaxis(Graphics g) {
    g.setColor(Color.BLUE);
    g.fillRect(0, originY - 2, width, 4);
  }

  //Function to draw Y-axis
  public void drawYaxis(Graphics g) {
    g.setColor(Color.BLUE);
    g.fillRect(originX - 2, 0, 4, height);
  }

  // Function to draw the Grid
  public void drawGrid(Graphics g) {
    drawHorizontalLines(g);
    drawVeritcalLines(g);
  }

  //Function to draw the horizontal lines of the grid
  public void drawHorizontalLines(Graphics g) {
    g.setColor(Color.YELLOW);
    for (int i = originX; i <= width; i += gap) {
      g.drawLine(i, 0, i, height);
    }
    for (int i = originX; i >= 0; i -= gap) {
      g.drawLine(i, 0, i, height);
    }
  }

  //Function to draw the vertical lines of the grid
  public void drawVeritcalLines(Graphics g) {
    g.setColor(Color.YELLOW);
    for (int i = originY; i <= height; i += gap) {
      g.drawLine(0, i, width, i);
      // add coordinate text
```

```java
    }
    for (int i = originY; i >= 0; i -= gap) {
      g.drawLine(0, i, width, i);
    }
  }

  //Function for the buttons
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == b1) zoom(10);
    if (e.getSource() == b2) zoom(-10);
  }

  //Function for the mousewheel
  public void mouseWheelMoved(MouseWheelEvent e) {
    int z = e.getWheelRotation();
    zoom(z);
  }

  //Function for the zoom in feature
  public void zoom(int i) {
    if (gap + i >= 1 && gap + i <= 300) {
      gap += i;
      repaint();
    }
  }


public void drawLine(Graphics g, int x1, int y1, int x2, int y2) {
  int x = x1;
  int y = y1;
  double m = (double)(y2 - y1) / (x2 - x1);
if(m>=0){
  if (m <= 1) {
    double p = ((double)1/2) - m;
    plotPoint(g,x,y,Color.green);
    while (x<x2) {
      x++;
      if (p < 0) {


        y = y + 1;
        p = p + 1 - m;
        plotPoint(g, x, y, Color.green);
      } else{

        p = p - m;
        plotPoint(g, x, y, Color.green);
      }
```

```
        }
      }
      else{
        double p=1-((double)m/2);
        plotPoint(g,x,y,Color.green);
        while (x<x2){
          y++;
          if (p < 0) {
            p = p + 1;
            plotPoint(g, x, y, Color.green);
          } else{
            x++;
            p = p - m+1;
            plotPoint(g, x, y, Color.green);
          }
        }
      }
    }
    else{
      if (Math.abs(m) <= 1) {
        double p = ((double)1/2) + m;
        plotPoint(g,x,y,Color.green);
        while (x>x2) {
          x--;
          if (p < 0) {


            y = y + 1;
            p = p + 1 + m;
            plotPoint(g, x, y, Color.green);
          } else{

            p = p + m;
            plotPoint(g, x, y, Color.green);
          }
        }
      }
      else{
        double p=1+((double)m/2);
        plotPoint(g,x,y,Color.green);
        while (x<x2){
          y--;
          if (p < 0) {
            p = p + 1;
            plotPoint(g, x, y, Color.green);
          } else{
            x++;
            p = p + m+1;
        plotPoint(g, x, y, Color.green);
      }
    }
  }
}
}

}
```



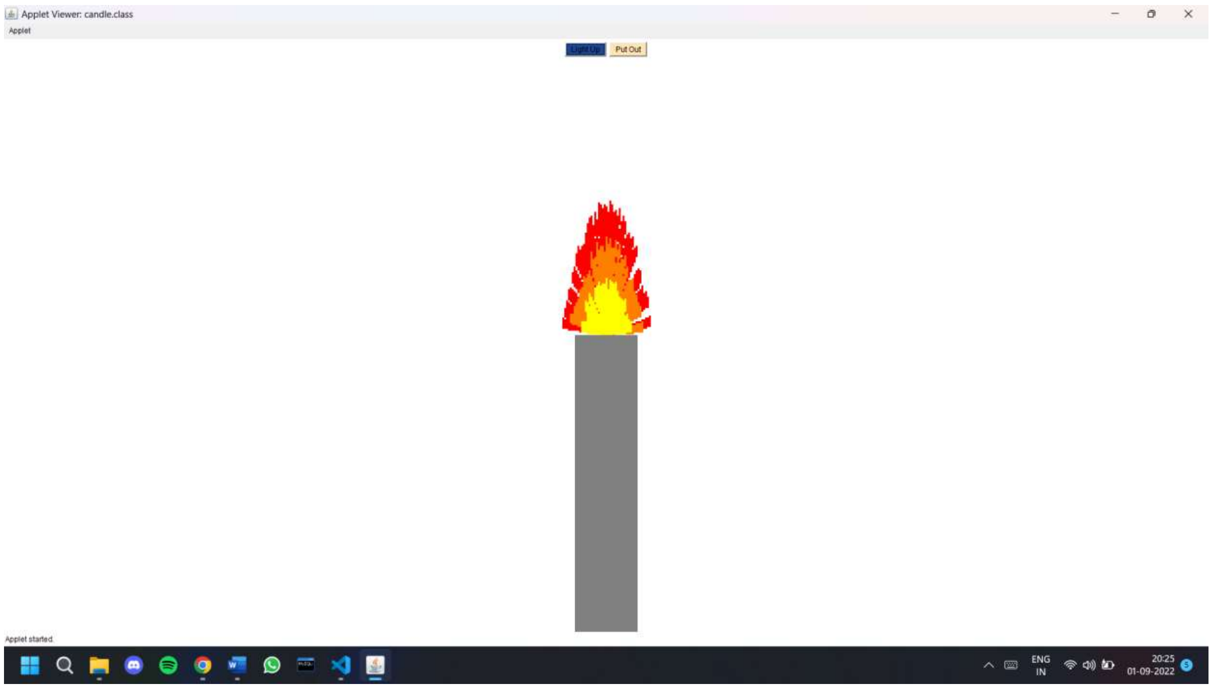
➢

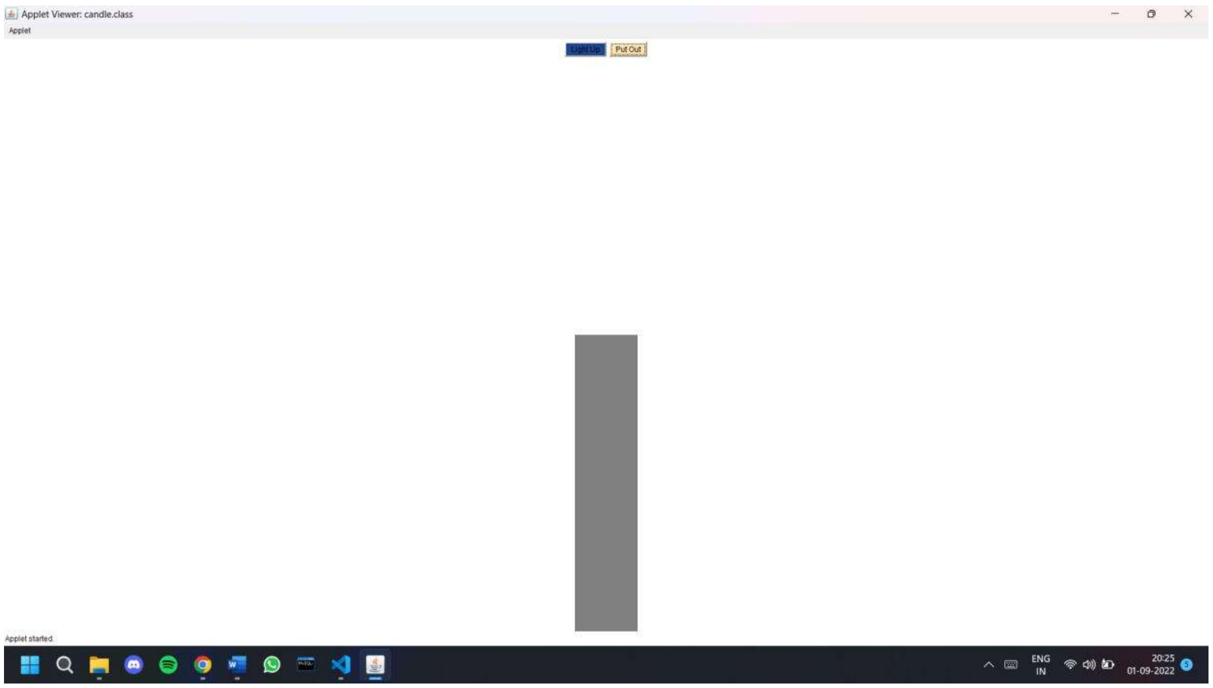
Prepare a class ‘Fire’ following instructions below.

i. Fire (Fig. 2) is created by collection of straight lines which are very closed together.
ii. Use any line drawing algorithm that is implemented in Part-I, Assignment 2.
iii. Height of the straight lines change over time by changing endpoints away from the source of fire
iv. Colour of fire may vary as the flame is away from the source.

Hence create a class ‘Candle’ (Fig. 3) having at least two methods light_candle () put_out_candle()

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class candle
  extends Applet
  implements ActionListener, MouseWheelListener {

  int originX, originY;
  int height, width;
  int gap = 20;
  int temp = 1;
  Button b3 = new Button(" Light Up ");
Button b4 = new Button(" Put Out ");

public void init() {
  setBackground(new Color(255, 255, 255));
  b3.setBackground(new Color(31, 70, 144));
  b4.setBackground(new Color(255, 229, 180));
  add(b3);
  add(b4);
  addMouseWheelListener(this);
  b3.addActionListener(this);
  b4.addActionListener(this);
}

public void paint(Graphics g) {
  g.setColor(Color.BLACK);
  height = getHeight();
  width = getWidth();
  originX = (getX() + width) / 2;
  originY = (getY() + height) / 2;
  Candle f = new Candle();
  f.drawCandle(g);
}

public void plotPoint(Graphics g, int x, int y, Color c) {
  g.setColor(c);
  g.fillRect(
    originX + (x * gap) - gap / 2,
    originY - (y * gap) - gap / 2,
    gap,
    gap
  );
}

public void actionPerformed(ActionEvent e) {
  Candle c = new Candle();
  if (e.getSource() == b3) c.light_candle();
  if (e.getSource() == b4) c.put_out_candle();
}

public void mouseWheelMoved(MouseWheelEvent e) {
  int z = e.getWheelRotation();
  zoom(z);
}

public void zoom(int i) {
  if (gap + i >= 1 && gap + i <= 300) {
    gap += i;
    repaint();
    }
  }

  int round(float n) {
    if (n - (int) n < 0.5) return (int) n;
    return (int) (n + 1);
  }

  void DDALine(Graphics g, int x0, int y0, int x1, int y1, Color c) {
    int dx = (x1 - x0);
    int dy = (y1 - y0);

    int step;
    if (Math.abs(dx) > Math.abs(dy)) {
      step = Math.abs(dx);
    } else {
      step = Math.abs(dy);
    }

    float x_incr = (float) dx / step;
    float y_incr = (float) dy / step;
    float x = (float) x0;
    float y = (float) y0;

    for (int i = 0; i < step; i++) {
      plotPoint(g, round(x), round(y), c);
      x += x_incr;
      y += y_incr;
    }
  }

  class Fire {

    int x1;
    int x2;
    int a;

    Fire() {
      x1 = -400;
      x2 = 400;
      a = 600;
    }

    public void paint(Graphics g) {
      drawFire(g);
    }

    public void drawFire(Graphics g) {
      while (x1 != x2) {
        if (a - (x1 * x1) >= 0) {
          int r = (int) (Math.random() * 10);
          Color c1 = new Color(255, 0, 0);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r + 1, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r + 2, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r + 3, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r + 4, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1)) / 10 + r + 5, c1);
        }
        x1++;
      }
      x1 = -400;
      x2 = 400;
      while (x1 != x2) {
        if (a - (x1 * x1) - 200 >= 0) {
          int r = (int) (Math.random() * 10);
          Color c1 = new Color(255, 128, 0);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r + 1, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r + 2, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r + 3, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r + 4, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 200) / 10 + r + 5, c1);
        }
        x1++;
      }
      x1 = -400;
      x2 = 400;
      while (x1 != x2) {
        if (a - (x1 * x1) - 400 >= 0) {
          int r = (int) (Math.random() * 10);
          Color c1 = new Color(255, 255, 0);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r + 1, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r + 2, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r + 3, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r + 4, c1);
          DDALine(g, 0, 0, x1, (a - (x1 * x1) - 400) / 10 + r + 5, c1);
        }
        x1++;
      }
    }
  }

  class Candle {
    public void paint(Graphics g) {
      drawCandle(g);
    }

    public void drawCandle(Graphics g) {
      if (temp == 1) {
        Fire f = new Fire();
        f.paint(g);
      }
      drawBase(g, new Color(128, 128, 128));
    }

    public void light_candle() {
      temp = 1;
      repaint();
    }

    public void put_out_candle() {
      temp = 0;
      repaint();
    }

    public void drawBase(Graphics g, Color c) {
      g.setColor(c);
      g.fillRect((originX - 50), originY, 100, 600);
    }
  }
}
```


Applet Viewer: candle.class
Applet
Light Up
Put Out
Applet started.
ENG
IN
20:25
01-09-2022


Applet Viewer: candle.class
Applet
Light Up
Put Out
Applet started.
ENG
IN
20:25
01-09-2022