```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


// package HW2
public class HW2 extends JComponent{

    static int frameWidth;
    static int frameHeight;
    static double Sx, Sy; // Scale factor x and y.
    static double Tx, Ty; // Translation factor x and y.

    static double XV, XW; // Move from Window X to Viewport X
    static double YV, YW; // Move from Window Y to Viewport Y

    static double a, b, c; // varaiable for formula 3

    static double OX, OY; // Start Point of X and Y.

    static double XVmax, XVmin; // Viewport X's max and min
    static double YVmax, YVmin; // Viewport Y's max and min

    static double XWmax, XWmin; // Window X's max and min
    static double YWmax, YWmin; // Window Y's max and min

    static double Temp_YV, Temp_OY;

    static boolean enable_dotted_line;



    public static void main(String[] args) {

        initGraphics();

    }



    public void paintComponent(Graphics g) {

        frameWidth = getWidth();
        frameHeight = getHeight();

        // Horizontal Line of the frame
        g.drawLine(0, frameHeight/2, frameWidth, frameHeight/2);
        // Vertical Line of the frame
        g.drawLine(frameWidth/2, 0, frameWidth/2, frameHeight);

        // Print information in Q4
        PrintInformation(g);
        PrintLable(g);


        // Chnage drawLine brush
        /*
         Graphics2D g2 = (Graphics2D)g;
         float[] dash = {5, 5};
         BasicStroke bs = new
             BasicStroke(1,BasicStroke.CAP_BUTT,BasicStroke.JOIN_MITER,
             10.0f,dash,0.0f);
         g2.setStroke(bs);
         */

        // Graph 1
        setViewport(1);
        setWindow(1);
        windowToViewport(g, 1);

        // Graph 2
        setViewport(2);
        setWindow(2);
        windowToViewport (g, 2);

        // Graph 3
        setViewport(3);
        setWindow(3);
        windowToViewport (g, 3);

        // Graph 4
        setViewport(3);
        setWindow(4);
        windowToViewport (g, 4);

        // Graph 5
        setViewport(3);
        setWindow(5);
        windowToViewport (g, 5);

    }


    // To get Viewport X's, Y's max and min
    // Note that start point always from left down corner
    // parameter: Which quadrant? (1, 2, 3, 4?)
    static public void setViewport(int quadrant) {

        switch (quadrant) {

                // Set location to the first quardant of Viewport
            case 1:
                XVmin = frameWidth /2;
                XVmax = frameWidth;
                YVmin = frameHeight /2;
                YVmax = 0;
                break;

                // second
            case 2:
                XVmin = frameWidth /2;
                XVmax = frameWidth;
                YVmin = frameHeight;
                YVmax = frameHeight / 2;
                break;

                // third
            case 3:
                XVmin = 0;
                XVmax = frameWidth / 2;
                YVmin = frameHeight / 2;
                YVmax = 0;
                break;


        }

    }

    // To get Window X's, Y's max and min, and the graphic's start point
    // parameter: which graph is gonna map?
    static public void setWindow(int graph_number) {

        switch (graph_number) {

            case 1:
                XWmin = 0;
                XWmax = 3 * Math.PI;
                YWmin = -1.8;
                YWmax = 2.6;
                OX = frameWidth / 2;
                OY = frameHeight /2 ;
                break;

            case 2:
                XWmin = -6;
                XWmax = 6;
                YWmin = -27;
                YWmax = 30;
                OX = frameWidth / 2;
                OY = frameHeight;
                break;

            case 3:
                XWmin = -1;
                XWmax = 0.5;
                YWmin = -0.4;
                YWmax = 0.4;
                OX = 0;
                OY = frameHeight / 2;
                a = 0.5;
                b = 0.5;
                c = 1.0;
                break;

            case 4:
                XWmin = -0.5;
                XWmax = 0.5;
                YWmin = -0.4;
                YWmax = 0.4;
                OX = 0;
                OY = frameHeight / 2;
                a = 0.5;
                b = 1.0;
                c = 1.0;
                break;

            case 5:
                XWmin = -0.25;
                XWmax = 0.5;
                YWmin = -0.4;
                YWmax = 0.4;
                OX = 0;
                OY = frameHeight / 2;
                a = 0.5;
                b = 2.0;
                c = 1.0;
                break;

        }
    }

    // Try to move from Window to Viewport
    // and get what we need: Scaled X, Y. And Translation X, Y.
    static public void moveTo2D () {
        Sx = (XVmax - XVmin) / (XWmax - XWmin);
        Sy = (YVmax - YVmin) / (YWmax - YWmin);
        Tx = (XWmax * XVmin - XWmin * XVmax) / (XWmax - XWmin);
        Ty = (YWmax * YVmin - YWmin * YVmax) / (YWmax - YWmin);
    }

    // draw particular graphic on the frame
    static public void drawTo2D (Graphics g, int graphic_number) {


        switch (graphic_number) {

            case 1:
                for (XW = 0; XW <= 3 * Math.PI; XW+=0.01) {
                    YW = 3.0 * (Math.pow(Math.E, -0.33 * XW) * Math.sin(3 * XW));
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    OX = XV;
                    OY = YV;
                }
                break;

            case 2:
                for (XW = -6; XW <= 6; XW+=0.005) {
                    YW = ( (3*XW*XW) - (12*XW) - 15) / ((XW*XW) - 3*XW - 10);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    if (YV > frameHeight/2 && YV < frameHeight)
                        g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    OX = XV;
                    OY = YV;
                }
                break;


            case 3:
                for (XW = -1; XW <= 0.5; XW+=0.01) {
                    YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    Temp_YV = frameHeight / 2 - YV;
                    Temp_OY = frameHeight / 2 - OY;
                    if (Temp_YV < frameHeight / 2)
                        g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)Temp_YV);
                    OX = XV;
                    OY = YV;
                }
                break;

            case 4:
                for (XW = -0.5; XW <= 0.5; XW+=0.01) {
                    YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    Temp_YV = frameHeight / 2 - YV;
                    Temp_OY = frameHeight / 2 - OY;
                    if (Temp_YV < frameHeight / 2)
                        g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)Temp_YV);
                    OX = XV;
                    OY = YV;
                }
                break;

            case 5:
                for (XW = -0.25; XW <= 0.5; XW+=0.01) {
                    YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    Temp_YV = frameHeight / 2 - YV;
                    Temp_OY = frameHeight / 2 - OY;
                    if (Temp_YV < frameHeight / 2)
                        g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)Temp_YV);
                    OX = XV;
                    OY = YV;
                }
                break;
        }

    }
```

```
    // draw particular graphic on the frame
    static public void drawTo2D_Dotted_Line (Graphics g, int graphic_number) {
        int trythis = 0;

        switch (graphic_number) {

            case 1:
                for (XW = 0; XW <= 3 * Math.PI; XW+=0.01) {
                    if (trythis % 3 != 0) {
                        XW += 0.01;
                        YW = 3.0 * (Math.pow(Math.E, -0.33 * XW) * Math.sin(3 *
                            XW));
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        OX = XV;
                        OY = YV;
                    }

                    else {
                        YW = 3.0 * (Math.pow(Math.E, -0.33 * XW) * Math.sin(3 *
                            XW));
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                        OX = XV;
                        OY = YV;
                    }
                    trythis++;

                }
                break;

            case 2:
                for (XW = -6; XW <= 6; XW+=0.005) {
                    if (trythis % 3 != 0) {
                        YW = ( (3*XW*XW) - (12*XW) - 15) / ((XW*XW) - 3*XW - 10);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        OX = XV;
                        OY = YV;
                    }

                    else {
                        YW = ( (3*XW*XW) - (12*XW) - 15) / ((XW*XW) - 3*XW - 10);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;

                        if (YV > frameHeight/2 && YV < frameHeight)
                            g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                        OX = XV;
                        OY = YV;
                    }
                    trythis++;
                }
                break;
            case 3:
                for (XW = -1; XW <= 0.5; XW+=0.01) {
                    if (trythis % 3 != 0) {
                        YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        Temp_YV = frameHeight / 2 - YV;
                        Temp_OY = frameHeight / 2 - OY;
                        OX = XV;
                        OY = YV;
                    }

                    else {
                        YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                        Temp_YV = frameHeight / 2 - YV;
                        Temp_OY = frameHeight / 2 - OY;
                        if (Temp_YV < frameHeight / 2)
                            g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)
                                Temp_YV);
                        OX = XV;
                        OY = YV;
                    }
                    trythis++;
                }
                break;

            case 4:
                for (XW = -0.5; XW <= 0.5; XW+=0.01) {
                    if (trythis % 3 != 0) {
                        YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        Temp_YV = frameHeight / 2 - YV;
                        Temp_OY = frameHeight / 2 - OY;
                        OX = XV;
                        OY = YV;
                    }

                    else {
                        YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                        XV = Sx * XW + Tx;
                        YV = Sy * YW + Ty;
                        g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                        Temp_YV = frameHeight / 2 - YV;
                        Temp_OY = frameHeight / 2 - OY;
                        if (Temp_YV < frameHeight / 2)
                            g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)
                                Temp_YV);
                        OX = XV;
                        OY = YV;
                    }
                    trythis++;
```

```
            }
            break;

        case 5:
            for (XW = -0.25; XW <= 0.5; XW+=0.01) {
                if (trythis % 3 != 0) {
                    YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    Temp_YV = frameHeight / 2 - YV;
                    Temp_OY = frameHeight / 2 - OY;
                    OX = XV;
                    OY = YV;
                }

                else {
                    YW = Math.sqrt(b*c*c*XW*XW*XW + a*c*c*XW*XW);
                    XV = Sx * XW + Tx;
                    YV = Sy * YW + Ty;
                    g.drawLine((int)OX, (int)OY, (int)XV, (int)YV);
                    Temp_YV = frameHeight / 2 - YV;
                    Temp_OY = frameHeight / 2 - OY;
                    if (Temp_YV < frameHeight / 2)
                        g.drawLine((int)OX, (int)Temp_OY, (int)XV, (int)
                            Temp_YV);
                    OX = XV;
                    OY = YV;
                }
                trythis++;
            }
            break;
    }

}

// Call moveTo2D() to get the coordination on the Viewport, then draw it.
static public void windowToViewport (Graphics g, int graphic_number) {
    moveTo2D();


    if (enable_dotted_line == false)
        drawTo2D(g, graphic_number);
    else
        drawTo2D_Dotted_Line(g, graphic_number);
}

// Print out student information at the Q4
public void PrintInformation(Graphics g) {
    int Sx, Sy;
    Sx = 0 + frameWidth/10;
    Sy = frameHeight/2 + frameHeight/10;
    g.drawString("CS_324 Computer Graphic", Sx, Sy);
    g.drawString("Assignment#2", Sx, Sy + Sy/10);
    g.drawString("Chihsiang Wang", Sx, Sy + 2*(Sy/10));
    g.drawString("101-64106", Sx, Sy + 3*(Sy/10));
```

```
        }

        public void PrintLable(Graphics g) {
            g.drawString("y = 3.0e^-0.33x sin(3x)", (int)(frameWidth * 0.67), (int)
                (frameHeight * 0.05) );
            g.drawString("Domain[0, 3PI]", (int)(frameWidth * 0.67), (int)
                (frameHeight * 0.1) );

            g.drawString("y = (3x^2 - 12x - 15) / (x^2 - 3x - 10)", (int)(frameWidth
                * 0.67), (int)(frameHeight * 0.55) );
            g.drawString("Domain[-6, 6]", (int)(frameWidth * 0.67), (int)(frameHeight
                * 0.6) );

            g.drawString("y^2 - bc^2x^4 - ac^2x^2 = 0", (int)(frameWidth * 0.05),
                (int)(frameHeight * 0.05) );
            g.drawString("a= 0.5, c = 1.0", (int)(frameWidth * 0.05), (int)
                (frameHeight * 0.068) );

            g.drawString("b = 0.5", (int)(frameWidth * 0.05), (int)(frameHeight *
                0.12) );
            g.drawString("b = 1.0", (int)(frameWidth * 0.05), (int)(frameHeight *
                0.19) );
            g.drawString("b = 2.0", (int)(frameWidth * 0.05), (int)(frameHeight *
                0.24) );

        }

        // Some initialization
        public static void initGraphics() {

            int frameWidth = 800;
            int frameHeight = 800;
            Sx = 0; Sy = 0;
            Tx = 0; Ty = 0;
            XV = 0; XW = 0;
            YV = 0; YW = 0;
            XVmax = 0; XVmin = 0;
            YVmax = 0; YVmin = 0;
            XWmax = 0; XWmin = 0;
            YWmax = 0; YWmin = 0;

            enable_dotted_line = true;

            JFrame f = new JFrame("HW2");

            // Eixt application when the window is closed.
            f.addWindowListener( new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                    System.exit(0);
                }
            }
                                 );

            f.setSize (frameWidth, frameHeight);
            f.getContentPane().add(new HW2());
            f.setVisible(true);
        }

    }

    /* Programming Log:
     * 1. Checking each graphs' domain and range. Calculate and Save it.
     * 2. Start to code. Set up frame, and use drawline to split frame to 4
         quadrants.
     * 3. Try to draw the first graph. -> It's too small to see anything.
     *    That is why I need to Map?
     * 4. Mapping the first graph to the first quadrant of the Viewport.
     *    Well domain and range. Looks good.
     * 5. second graph done. Try to remove the line between 2 point.
     * 6. Starting to work graph 3, 4, 5.
     * 7. What is the domain and the range?
     * 8. After several tries, now looks good.
     * 9. Try to draw graph with dotted line. Some graphs can't see well.
     * 10. Draw student information at the quadrant 4. Done.
     * 11. Fixed graph#3,4,5.
     * 12. Fixed some code so I can enable dotted line mode easily. Done.
     */
```