Assignment project 'RANDOM ART'

CSC319 Object-Oriented Software Development

Submitted to:
Asst.Prof.Dr. Chonlameth Arpnikanondt

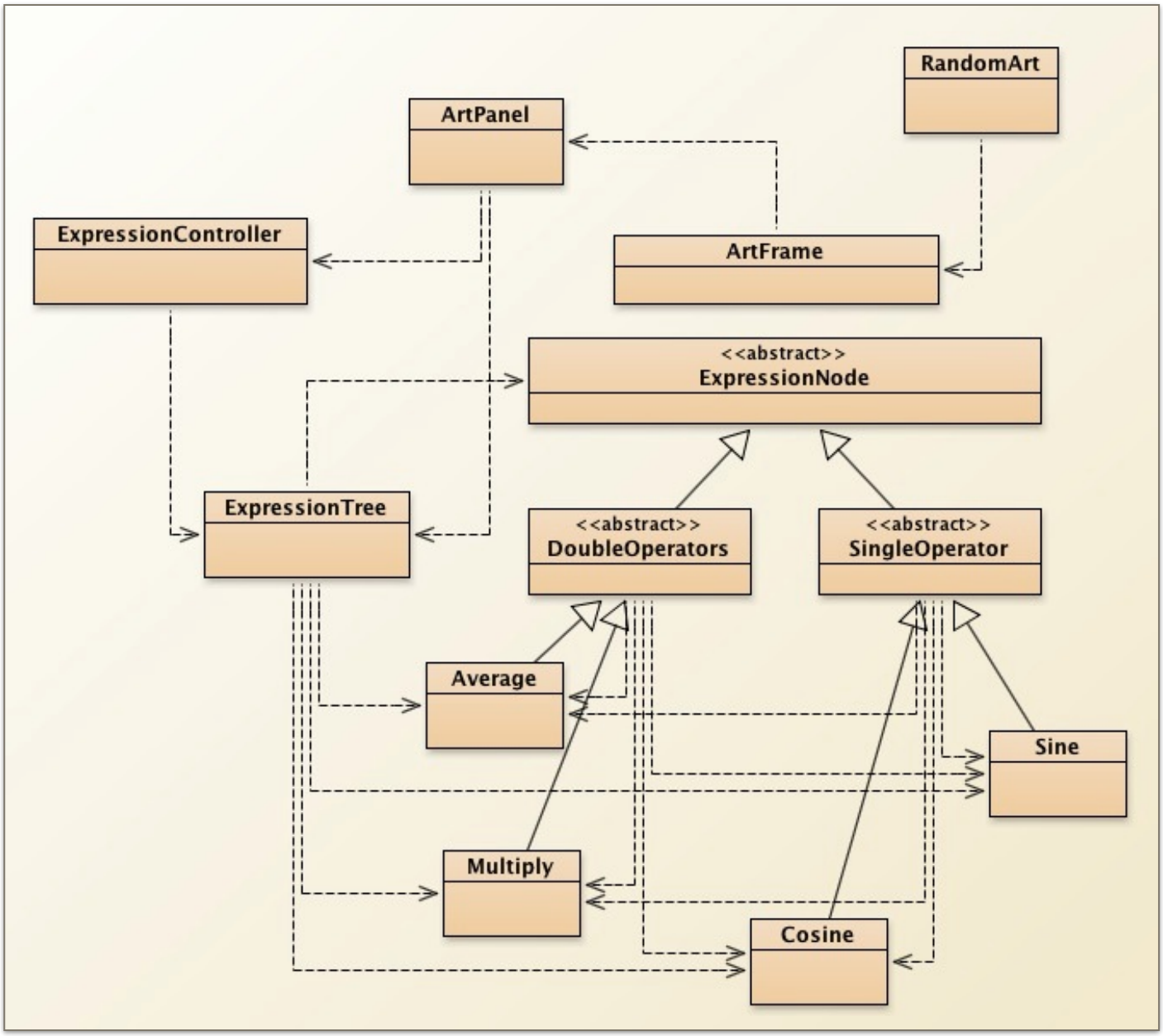
By:
Group No. 17
55130500205 Khemmachart Chutapetch
55130500239 Nontachai Booontavornsakun

Github:
https://github.com/hachiban-ramen

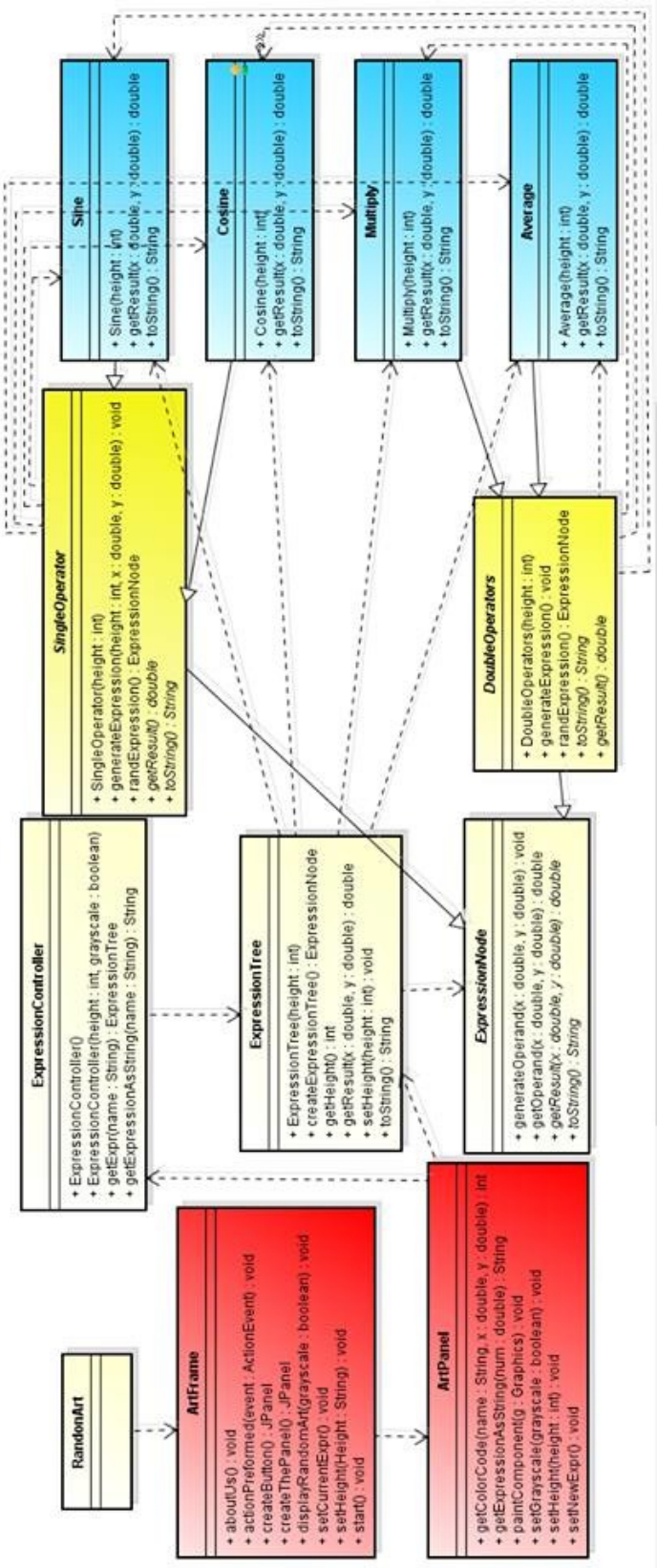
B.Sc.(Computer Science)
School of Information Technology
King Mongkut's University of Technology Thonburi

Date of submission: 22 November, 2013!

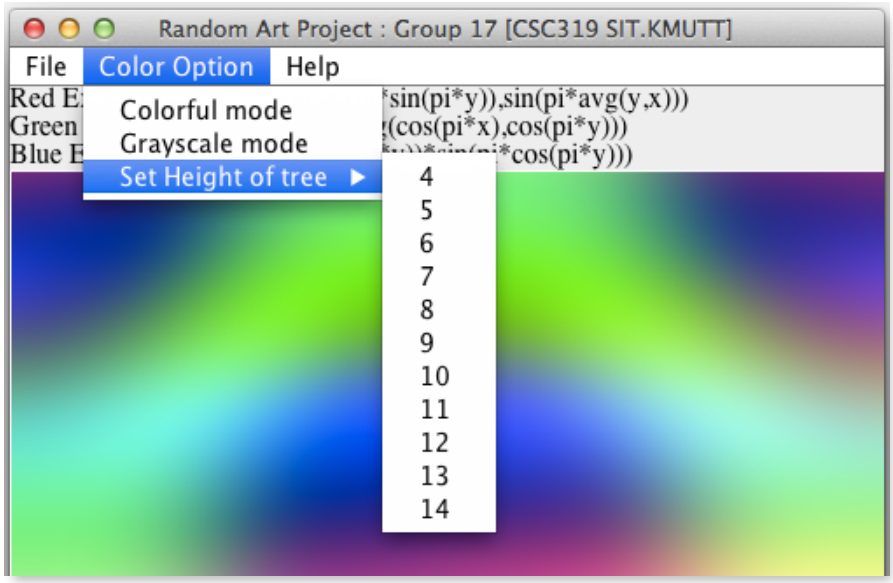
UML Diagrams

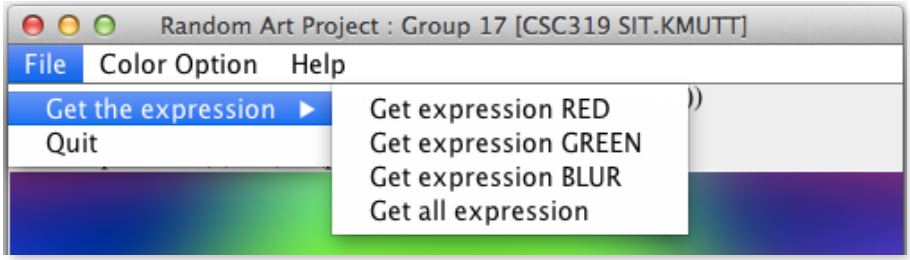


UML Diagrams

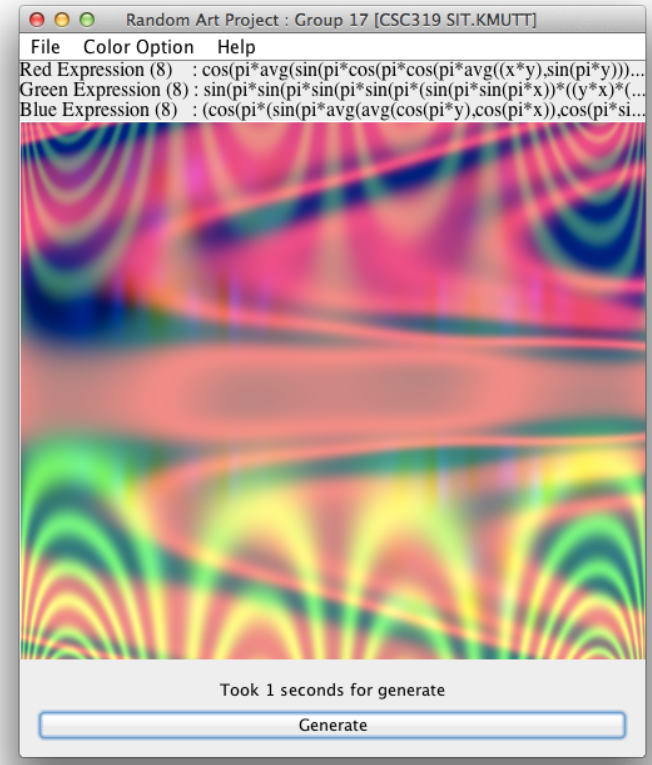


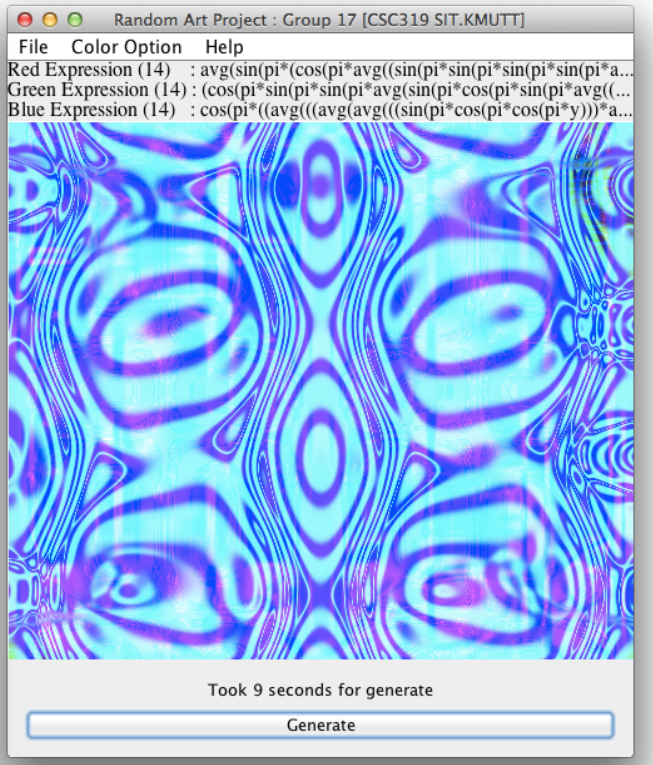
Example of Graphic User Interface

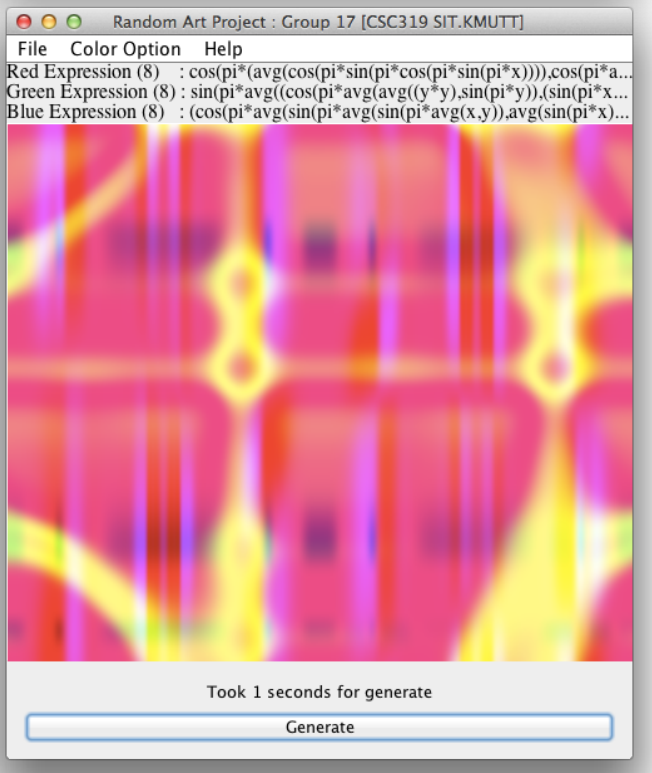




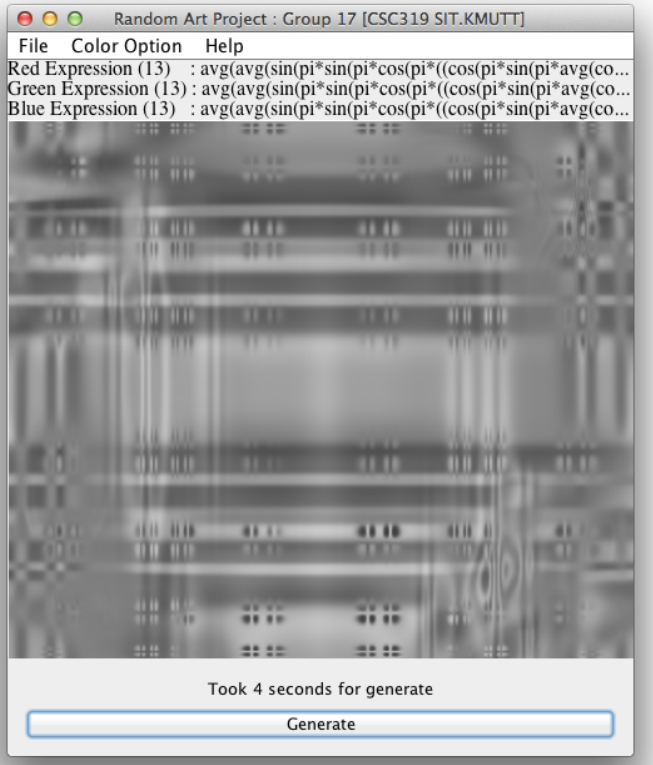
Example of Random Art







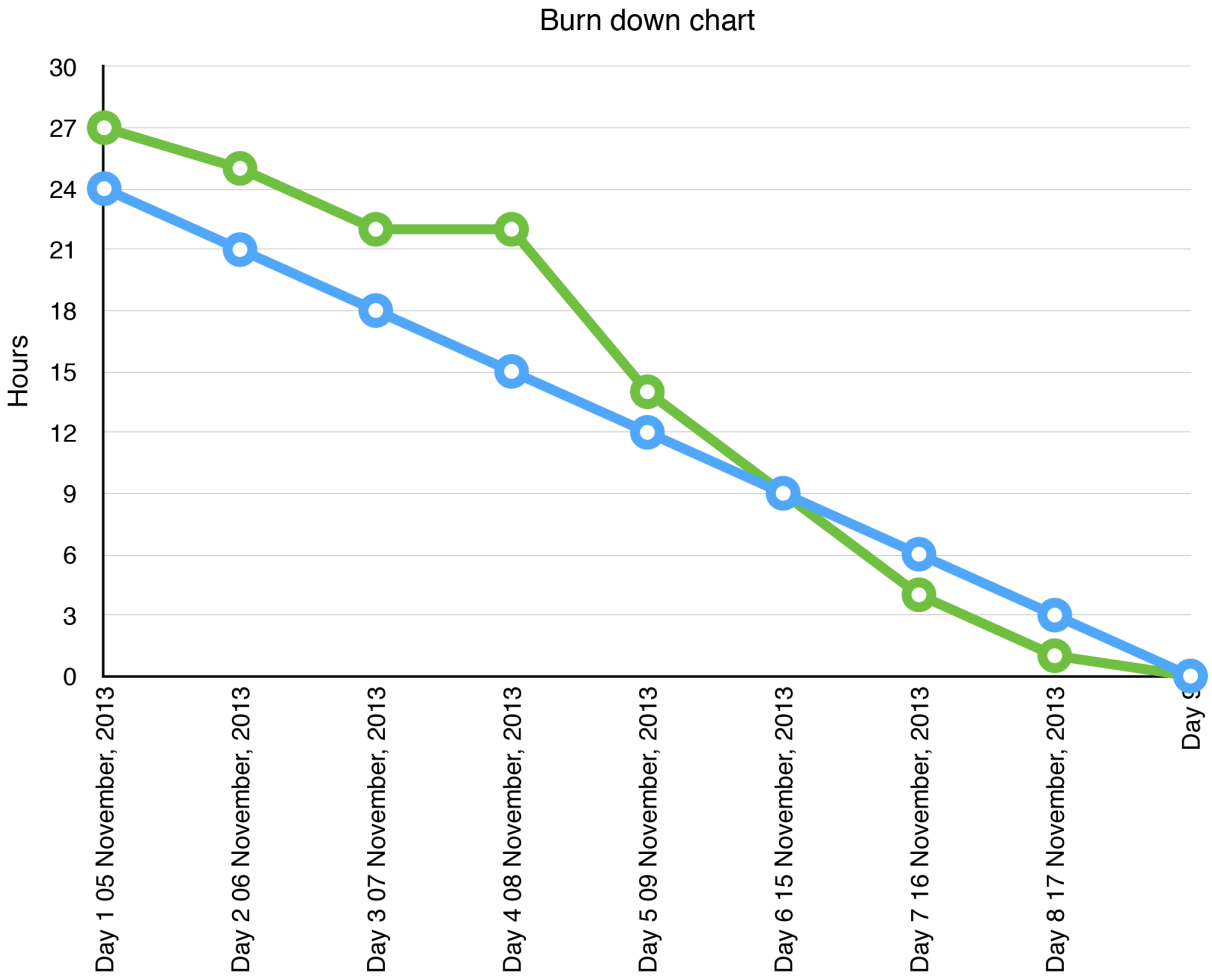
In Grayscale mode



Group No. 17

55130500205 Khemmachart Chutapetch
55130500239 Nontachai Booontavornsakun

User Story	Function List		Hours / Days	Day 1 05 November, 2013	Day 2 06 November, 2013	Day 3 07 November, 2013	Day 4 08 November, 2013	Day 5 09 November, 2013	Day 6 15 November, 2013	Day 7 16 November, 2013	Day 8 17 November, 2013	Day 9
As a member, I can read profiles of other members so that I can find someone to date.	Random Expression											
	- Random four operators (*, avg, sin, cos)		5		2	2		1				
	- Make an expression tree		4			1		3				
	- Function for input height of expression tree		2							2		
	Random Art Panel											
	- Function for make a random-art in rectangle		4					4				
	- Function for random-art using expression		4						4			
	- Function for grayscale and colorful option		1						1			
	Graphic User Interface											
	- Design a graphic user interface		2							1	1	
	- Put random art panel		2							1	1	
	- Complete UI		3							1	1	1
		Ideal	27	24	21	18	15	12	9	6	3	0
		Actual	27	27	25	22	22	14	9	4	1	0


Burn down chart
Hours
30
27
24
21
18
15
12
9
6
3
0
Day 1 05 November, 2013
Day 2 06 November, 2013
Day 3 07 November, 2013
Day 4 08 November, 2013
Day 5 09 November, 2013
Day 6 15 November, 2013
Day 7 16 November, 2013
Day 8 17 November, 2013

Discussion and Conclusion

Our discussion:

- What is SCRUMS
- What is the random art
- How to implement the random art
- How each function work
- How random art make itself difference
- How to implement random art in Object Oriented Programming
- How to use design pattern to implement our project
- How to make GUI for our project
- How to use Github

Out benefits of this assignment:

- Got the SCRUMS or how to works with partner.
- Learned how to plan a work, such as, function list, burn down chart.
- We can adopt the OO principles to solve the problem.
- Learned design pattern and know how it help when we have a lot of code
- Knows how to use Github

SourceFile

```
/**
 * This class represents frame for random art project.
 * The frame include GUI panel and random-art panel
 * that GUI panel has menu to control expression and the expression will exffect to random-art
panel
 */
public class ArtFrame extends JFrame implements ActionListener {

  private ArtPanel thePanel;

  private JLabel theCurrentExpression_RED;          // Represent the expression of red color
  private JLabel theCurrentExpression_GREEN;        // Represent the expression of green color
  private JLabel theCurrentExpression_BLUE;         // Represent the expression of blue color
  private JLabel timeForGenerate;                   // This label will show time that use for
                                                       generate the random art panel
  private JButton buttonGenerate;                   // The generate button will repaint the
                                                       random art panel

  private static final int EQUATION_FONT_SIZE = 16;       // The defualt size of front
  private ArrayList<JMenuItem> itemHeight;                // Represent the all of menu height,
                                                             for input of user

  public ArtFrame(){
    // Set the common properties of the frame
    setTitle("Random Art Project : Group 17 [CSC319 SIT.KMUTT]");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(500,600);

    // Set the window display on the centre on the screen
    Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
    int x = (int) (dimension.getWidth()/2 - getWidth()/2);
    int y = (int) (dimension.getHeight()/2 - getHeight()/2);
    setLocation(x, y);

    // create the panel to represent random art panel
    thePanel = new ArtPanel();

    // add the menu for Colored Art options
    setJMenuBar(createMenu());
    // add components to frame
    add( createExpressionAsString(), BorderLayout.NORTH );          // panel for show string of
                                                                       expression
    add( createThePanel(), BorderLayout.CENTER );    // panel for show random art
    add( createButton(), BorderLayout.SOUTH);         // create all button
```

```
  /**
   * Return the panel that represent random art
   */
  public JPanel createThePanel () {
      return thePanel;
  }

  /**
   * Create button and panel in south panel of the frame
   */
  public JPanel createButton () {
    // South panel include button for generate and stirng for time that took for generate
    JPanel southPanel = new JPanel();
    southPanel.setLayout(new GridLayout(2,0));
    southPanel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));

    timeForGenerate = new JLabel("",JLabel.CENTER);     // represent time that took for generate
    buttonGenerate = new JButton("Generate");           // generate button
    buttonGenerate.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent event) {
        long startTime = System.currentTimeMillis();    // first time
        thePanel.setNewExpr();                          // set new expression
        setCurrentExpr();                               // get string of expression
        repaint();                                      // repaint random art
        long endTime = System.currentTimeMillis();      // second time
        timeForGenerate.setText("Took " +(endTime-startTime)+ " seconds for generate");     //
second time minus first time
      }
    });

    // add thw panels into the south panel
    southPanel.add(timeForGenerate);
    southPanel.add(buttonGenerate);

    return southPanel;
  }

  /**
   * Get the string of expreestion for each color (red, green, and blue) then show these
expression in the panel
   */
  public JPanel createExpressionAsString () {
    JPanel exprString = new JPanel(new GridLayout(3,0));

    theCurrentExpression_RED = new JLabel( thePanel.getExpressionAsString("RED") );
    theCurrentExpression_RED.setFont(new Font("Serif", Font.PLAIN,
EQUATION_FONT_SIZE));
    theCurrentExpression_GREEN = new JLabel( thePanel.getExpressionAsString("GREEN") );
```

```
    theCurrentExpression_GREEN.setFont(new Font("Serif", Font.PLAIN,
EQUATION_FONT_SIZE));
    theCurrentExpression_BLUE = new JLabel( thePanel.getExpressionAsString("BLUE") );
    theCurrentExpression_BLUE.setFont(new Font("Serif", Font.PLAIN,
EQUATION_FONT_SIZE));

    exprString.add(theCurrentExpression_RED);
    exprString.add(theCurrentExpression_GREEN);
    exprString.add(theCurrentExpression_BLUE);

    return exprString;
  }

  /**
   * Set the current expression that use to represent the random art
   */
  public void setCurrentExpr () {
    theCurrentExpression_RED.setText( thePanel.getExpressionAsString("RED") );
    theCurrentExpression_GREEN.setText( thePanel.getExpressionAsString("GREEN") );
    theCurrentExpression_BLUE.setText( thePanel.getExpressionAsString("BLUE") );
  }

  /**
  * Create menu bar and all menu need
  */
  private JMenuBar createMenu() {
    JMenuBar menu = new JMenuBar();

    JMenu menuFile = new JMenu("File");
    JMenu menuColorOptions = new JMenu("Color Option");
    JMenu menuSetHeight = new JMenu("Set Height of tree");
    JMenu menuGetString = new JMenu("Get the expression");
    JMenu menuHelp = new JMenu("Help");

    JMenuItem itemQuit = new JMenuItem("Quit");
    JMenuItem itemColorful = new JMenuItem("Colorful mode");
    JMenuItem itemGrayscale = new JMenuItem("Grayscale mode");
    JMenuItem itemAbout = new JMenuItem("About us");
    JMenuItem itemString_R = new JMenuItem("Get expression RED");
    JMenuItem itemString_G = new JMenuItem("Get expression GREEN");
    JMenuItem itemString_B = new JMenuItem("Get expression BLUR");
    JMenuItem itemString_ALL = new JMenuItem("Get all expression");


    itemHeight = new ArrayList<JMenuItem>();
    for (int i=4 ; i<15 ; i++) {
        JMenuItem buttonHeight = new JMenuItem(""+i);
        buttonHeight.addActionListener( new ActionListener() {
            public void actionPerformed (ActionEvent e) {
```

```
                setHeight (e.getActionCommand());
            }
        });
        itemHeight.add(buttonHeight);
    }

    itemQuit.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            System.exit(0);
        }
    });

    itemColorful.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            displayRandomArt(false);
        }
    });

    itemGrayscale.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            displayRandomArt(true);
        }
    });


    itemAbout.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            aboutUs();
        }
    });

    itemString_R.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            String text = "";
            String temp = thePanel.getExpressionAsString ("RED");
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }
            JOptionPane.showMessageDialog(null, text, "Expression for Red",
JOptionPane.INFORMATION_MESSAGE);
        }
    });

    itemString_G.addActionListener( new ActionListener() {
```

```
        public void actionPerformed (ActionEvent e) {
            String text = "";
            String temp = thePanel.getExpressionAsString ("GREEN");
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }
            JOptionPane.showMessageDialog(null, text, "Expression for Green",
JOptionPane.INFORMATION_MESSAGE);
        }
    });

    itemString_B.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            String text = "";
            String temp = thePanel.getExpressionAsString ("BLUE");
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }
            JOptionPane.showMessageDialog(null, text, "Expression for Blue",
JOptionPane.INFORMATION_MESSAGE);
        }
    });

    itemString_ALL.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            String text = "";
            String temp = thePanel.getExpressionAsString ("RED");
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }

            text = text + "\n\n";

            temp = thePanel.getExpressionAsString ("GREEN");
```

```
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }


            text = text + "\n\n";
            temp = thePanel.getExpressionAsString ("BLUE");
            for (int i=0 ; i<temp.length() ; i++) {
                if (i>1 && i%100==0) {
                    text = text + "\n";
                }
                else {
                    text = text + temp.charAt(i);
                }
            }
            JOptionPane.showMessageDialog(null, text, "Expression for Red, Green, and Blue",
JOptionPane.INFORMATION_MESSAGE);
        }
    });

    menu.add(menuFile);
    menu.add(menuColorOptions);
    menu.add(menuHelp);

    menuFile.add(menuGetString);
    menuFile.add(itemQuit);
    menuColorOptions.add(itemColorful);
    menuColorOptions.add(itemGrayscale);
    menuColorOptions.add(menuSetHeight);
    menuGetString.add(itemString_R);
    menuGetString.add(itemString_G);
    menuGetString.add(itemString_B);
    menuGetString.add(itemString_ALL);

    Iterator<JMenuItem> it = itemHeight.iterator();
    while (it.hasNext()) {
        menuSetHeight.add(it.next());
    }
    menuHelp.add(itemAbout);

    return menu;
  }

  public void aboutUs () {
```

```
    String text = "RANDOM ART PROJECT\n" +
                "CSC319 Object-Oriented Software Development\n" +
                "School of Information Technology\n" +
                "King Mongkut's University of Technology Thonburi\n\n" +

                "Group No. 17 \n" +
                "55130500205 Khemmachart Chutapetch \n" +
                "55130500239 Nontachai Booontavornsakun\n";
    JOptionPane.showMessageDialog(null, text, "About us",
JOptionPane.INFORMATION_MESSAGE);
  }

  /**
   * repaint the random art, if grayscale is true then the random art will use the same expression
   * that's mean the art will be grayscale
   */
  public void displayRandomArt (boolean grayscale) {
     thePanel.setGrayscale(grayscale);
     repaint();
     setCurrentExpr();
  }

  /**
   * the default action performed method
   */
  public void actionPerformed(ActionEvent event) {
    System.out.println("This is "+ event.getActionCommand() +" function call from
actionPerformed method!");
  }

  /**
   * method set height of tree will call the method set height of tree from the random art panel
   */
  public void setHeight (String s) {
     int height = Integer.parseInt(s);
     thePanel.setHeight(height);
    }

  /**
   * Starting method
   */
  public void start(){
    setVisible(true);
  }
}

/**
 * This lass contain the main panel that represent the random art
 * and the common methods that need to generate random art
```

```
 */
public class ArtPanel extends JPanel   {

  // CS324e students.
  // Add class constants and instance variables here

  public static final int SIZE = 400;
  public static final int NUM_COLOR_OPTIONS = 2;
  public static final double pi = Math.PI;

  private ExpressionController exprController;
  private Color color;
  private int height;
  private boolean grayscale;

  /**
   * The default constructor will initial dafual value need
   * the default height is 4, and grayscale mode is off
   */
  public ArtPanel(){
    this.height = 4;
    this.grayscale = false;
    setPreferredSize(new Dimension(SIZE, SIZE));
    exprController = new ExpressionController(height, grayscale);
  }

  /**
   * This method will be called when user want to create new random art
   */
  public void setNewExpr() {
    exprController = new ExpressionController(height, grayscale);
  }

  /**
   * Get the string of expression of random art
   */
  public String getExpressionAsString (String name) {
      return exprController.getExpressionAsString( name.toUpperCase() );
    }

  /**
   * Calculate and paint the conponent
   */
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;

    // CS324e students: add and change as necessary
    g2.setColor(Color.WHITE);
```

```
    g2.fillRect(0, 0, getWidth() - 1, getHeight() - 1);

    for(int i=0 ; i<getWidth() ; i++) {
      for(int j=0 ; j<getHeight() ; j++) {

        // x and y are always between -1.0 and 1.0, inclusive.
        // x and y represent the coordinates of a pixel in the panel scaled from -1.0 to 1.0.
        // So for example if we had a panel that was 200 pixels wide by 400 pixels tall, the upper
left corner pixel with coordinates (0,0)
        // would be scaled to (-1.0, -1.0). This assumes we have not translated or scaled the
graphics objects.
        // The pixel at the lower right corner would have coordinates(199, 399) and this would scale
to (1.0, 1.0).
        // Pixels in between are scaled based on the size of the panel.

        double x = ((((double) i) / getWidth()) * 2.0) - 1.0;
        double y = ((((double) j) / getHeight()) * 2.0) - 1.0;

        // For a given pixel the random function returns a value that is also between -1.0 and 1.0.
        // This value is then scaled to a grayscale value. -1.0 is black, Color (0,0,0) and 1.0 is
white, Color(255, 255, 255).
        // The graphics object is set to that color and a rectangle of size 1 is filled at the
coordinates for that pixel.
        // (You may find using fillRect instead of fill on a Rectangle object improves the speed of
your program.)

        int red = getColorCode( "RED", x, y );
        int green = getColorCode( "GREEN", x, y );
        int blue = getColorCode( "BLUE", x, y );

        color = new Color(red, green, blue);
        g2.setColor(color);
        g2.fillRect(i,j,i,j);
      }
    }
  }

  /**
   * method for set the grayscale mode, if grayscale mode is true that's mean the tree color will
   * use the same expression and the art will be grayscale
   */
  public void setGrayscale (boolean grayscale) {
      this.grayscale = grayscale;
      this.setNewExpr();
  }

  /**
   * Case the double value of expression [-1,1] to the int value of color between [0,255]
   */
```

```
  public int getColorCode (String name, double x, double y) {
    return (int)(( (exprController.getExpr( name.toUpperCase() )).getResult( x , y ) + 1 ) * 255/2 );
  }

  /**
   * THe code that pass the expression will pass into this method and will generate to value
between (0-255)
   */
  public int getColorCode2 (Double num) {
    return (int)(( num + 1 ) * 255/2 );
  }

  /**
   * Set height of the expression
   */
  public void setHeight (int height) {
      this.height = height;
  }
}
```

```
/**
 * This class is expression controller, when people want to generate the new expression
 * user can call method in this class then three expression will executes in one times
 *
 * that's mean user need not to call three method for generate red, green, and blue.
 * but they can control those three method by using one method in this class
 */

public class ExpressionController {
  public static final int DEFAULT_MINIMUM_HEIGHT = 4;

  // Three expression for each color
  private ExpressionTree exprRed;
  private ExpressionTree exprGreen;
  private ExpressionTree exprBlue;

  private int currentHeight;

  /**
   * Default constructure will call another constructure by using default input value
   */
  public ExpressionController () {
    this(DEFAULT_MINIMUM_HEIGHT, false);
    currentHeight = DEFAULT_MINIMUM_HEIGHT;
  }

  /**
   * Main constructure
   */
  public ExpressionController (int height, boolean grayscale) {
    if (!grayscale) {
        exprRed   = new ExpressionTree(height);
        exprGreen = new ExpressionTree(height);
        exprBlue  = new ExpressionTree(height);
    }
    else {
        exprRed   = new ExpressionTree(height);
        exprGreen = exprBlue = exprRed;
    }
    currentHeight = height;
  }

  /**
   * Get each expression by verify that name of expression that pass by parameter
   */
  public ExpressionTree getExpr (String name) {
    if ( (name.toUpperCase()).equals("RED") ) {
        return exprRed;
    }
```

```
    else if ( (name.toUpperCase()).equals("GREEN") ) {
        return exprGreen;
    }
    else if ( (name.toUpperCase()).equals("BLUE") ) {
        return exprBlue;
    }
    else {
        return null;
    }
  }

  /**
   * Get string of each expression by verify that name of expression that pass by parameter
   */
  public String getExpressionAsString (String name) {
    if ( (name.toUpperCase()).equals("RED") ) {
        return "Red Expression (" +exprRed.getHeight()+ ")    : " +exprRed.toString();
    }
    else if ( (name.toUpperCase()).equals("GREEN") ) {
        return "Green Expression (" +exprGreen.getHeight()+ ") : " +exprGreen.toString();
    }
    else if ( (name.toUpperCase()).equals("BLUE") ) {
        return "Blue Expression (" +exprBlue.getHeight()+ ")   : " +exprBlue.toString();
    }
    else if ( (name.toUpperCase()).equals("GREY") ) {
        return "Grey Expression (" +exprRed.getHeight()+ ")   : " +exprRed.toString();
    }
    else {
        return "Sorry, not found your choice.";
    }
  }
}
```

```
/**
 * First of all, the expression tree will create one kind of Expression Node, such as, Average, Multoply, Sine, and Cosine
 * to be the root of the tree, then the root will create another subtree autometically until height of that node is one, it's mean the left node
 *
 * If the height of tree that input from user is less than DEFAULT_MINIMUM_HEIGHT, its will be equal minimun height
 *
 */

public class ExpressionTree {

  public static final int DEFAULT_MINIMUM_HEIGHT = 4;              // Minimum height of tree that require
  private ExpressionNode root;                                     // Root node, can be any posiible node
  private int height;                                              // Height of tree, need to create each node
  private boolean grayscale;
  private double x;                                                // x value's
  private double y;                                                // y value's

  /**
   * This method will initial the height of tree
   */
  public ExpressionTree (int height) {
    // If height of tree that less than minimum height requirement, it will be equal minimum height
    if ( height < DEFAULT_MINIMUM_HEIGHT ) {
      this.height = DEFAULT_MINIMUM_HEIGHT;
    }
    else {
      this.height = height;
    }

    // Create an expresstion tree
    this.root = createExpressionTree();
  }

  /**
   * The createExpression method will create one kind of poosible node (average, multiply, sine, and cosine) to be the root
   * then the root will create its subtree autometically
   */
  public ExpressionNode createExpressionTree () {
    int randNum = (int)(Math.random() * 4);
    if (randNum == 0) {
      return (new Multiply(this.height));
    }
    else if (randNum == 1) {
      return (new Average(this.height));
```

```
    }
    else if (randNum == 2) {
      return (new Sine(this.height));
    }
    else {
      return (new Cosine(this.height));
    }
  }

  public void setHeight (int height) {
     this.height = height;
  }

  public int getHeight () {
     return height;
  }

  // toString() will return all of expreesion in expression tree
  public String toString () {
    return root.toString();
  }

  // Get result from expression tree
  public double getResult (double x, double y) {
     return root.getResult(x, y);
  }
}
```

```
/**
 * ExpressionNode is the super class of all node, contains attributes and methods that every
nodes need
 */
public abstract class ExpressionNode {

  protected String operand;     // Keep an string of each expression
  protected int height;         // Height of each nodes in expression tree
  protected double x;           // Keep the value that input from user
  protected double y;           // Keep the value that input from user


  /**
   * When node is created, its will initial the height, and values of x and y into attribute (instance
variable)
   */
  public ExpressionNode (int height) {
    this.height = height;
  }

  public void generateOperand (double x, double y) {
    int randOperand = (int)(Math.random() * 2);
    if (randOperand == 0) {
      operand = "x";
    }
    else {
      operand = "y";
    }
  }

  /**
   * get the current value of operand
   */
  public double getOperand (double x, double y) {
        if ( operand.equals("x") ) {
            return x;
        }
        else {
            return y;
        }
  }

  public abstract double getResult (double x, double y);     // Get result of each node, it need to
execute all of expression before return.
  public abstract String toString();                         // toString() will return all of expression as
String
}
```

```
public abstract class DoubleOperators extends ExpressionNode {

  // The name the two varible to left and right
  protected ExpressionNode left;
  protected ExpressionNode right;

  public DoubleOperators (int height) {
    super(height);
    generateExpression();
  }

  /**
   * If node is leaf node (height is equal one) its will return one of x and y
   * otherwise, if it's not a leaf node, its will create another subtree in it children
   */
  public void generateExpression () {
    if (height != 1) {
      left = randExpression();
      right = randExpression();
    }
    else {
      generateOperand(x, y);
    }
  }

  /**
   * Random one of poosible expression node to be child
   */
  public ExpressionNode randExpression () {
    int randNum = (int)(Math.random() * 4);
    if (randNum == 0) {
      return (new Multiply(this.height -1));
    }
    else if (randNum == 1) {
      return (new Average(this.height -1));
    }
    else if (randNum == 2) {
      return (new Sine(this.height -1));
    }
    else {
      return (new Cosine(this.height -1));
    }
  }

  public abstract double getResult (double x, double y);     // Get result of each node, it need to
execute all of expression before return.
  public abstract String toString();                         // toString() will return all of expression as
String
}
```

```
public abstract class SingleOperator extends ExpressionNode {
  protected ExpressionNode nextNode;
  public SingleOperator (int height) {
    super(height);
    generateExpression(height, x, y);
  }

  public void generateExpression (int height, double x, double y) {
    if (height != 1) {
      nextNode = randExpression();
    }
    else if (height == 1) {
      generateOperand(x, y);
    }
  }

  public ExpressionNode randExpression () {
    int randNum = (int)(Math.random() * 4);
    if (randNum == 0) {
      return (new Multiply(this.height -1));
    }
    else if (randNum == 1) {
      return (new Average(this.height -1));
    }
    else if (randNum == 2) {
      return (new Sine(this.height -1));
    }
    else {
      return (new Cosine(this.height -1));
    }
  }

  public abstract double getResult (double x, double y);     // Get result of each node, it need to
execute all of expression before return.
  public abstract String toString();                         // toString() will return all of expression as
String}

}
```

```
public class Average extends DoubleOperators {

  public Average (int height) {
    super(height);
  }

  /**
   * If it's leaf node, it will return one of x and y value
   * otherwise, will create abother subtree and use (x+y)/2 equalation
   */
  public double getResult (double x, double y) {
    if (height != 1) {
      return ( left.getResult(x, y) + right.getResult(x, y) ) / 2;
    }
    else {
      return getOperand(x, y);
    }
  }

  public String toString () {
    if(height == 1) {
      return operand;
    }
    else {
      return "avg(" +left.toString()+ "," +right.toString()+ ")";
    }
  }
}
```

```
public class Multiply extends DoubleOperators {

  public Multiply (int height) {
    super(height);
  }

  public double getResult (double x, double y) {
    if (height != 1) {
      return left.getResult(x, y) * right.getResult(x, y);
    }
    else {
      return getOperand(x, y);
    }
  }

  public String toString () {
    if(height == 1) {
      return operand;
    }
    else {
      return "(" +left.toString()+ "*" +right.toString()+ ")";
    }
  }
}
```

```
public class Cosine extends SingleOperator {

  public Cosine (int height) {
    super(height);
  }

  public double getResult (double x, double y) {
    if (height != 1) {
      return Math.cos( Math.PI * nextNode.getResult(x, y) );
    }
    else {
      return getOperand(x, y);
    }
  }

  public String toString () {
    if(height == 1) {
      return operand;
    }
    else {
      return "cos(pi*" +nextNode.toString()+ ")";
    }
  }

}
```

```
public class Sine extends SingleOperator {

  public Sine (int height) {
    super(height);
  }

  public double getResult (double x, double y) {
    if (height != 1) {
      return Math.sin( Math.PI * nextNode.getResult(x, y) );
    }
    else {
      return getOperand(x,y);
    }
  }

  public String toString () {
    if(height == 1) {
      return operand;
    }
    else {
      return "sin(pi*" +nextNode.toString()+ ")";
    }
  }

}
```