BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	Qalbiah	Anisa Dzikra	2502043116

Course Code : COMP6699001 **Course Name** : Object Oriented Programming

Class : L2AC **Name of Lecturer(s)** : Jude Joseph Lamug Martinez

Major : Computer Science

Title of Assignment : Fruit Memory Game

Type of Assignment : Final Project

Submission Pattern

Due Date : June 10 **Submission Date** : June 10

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

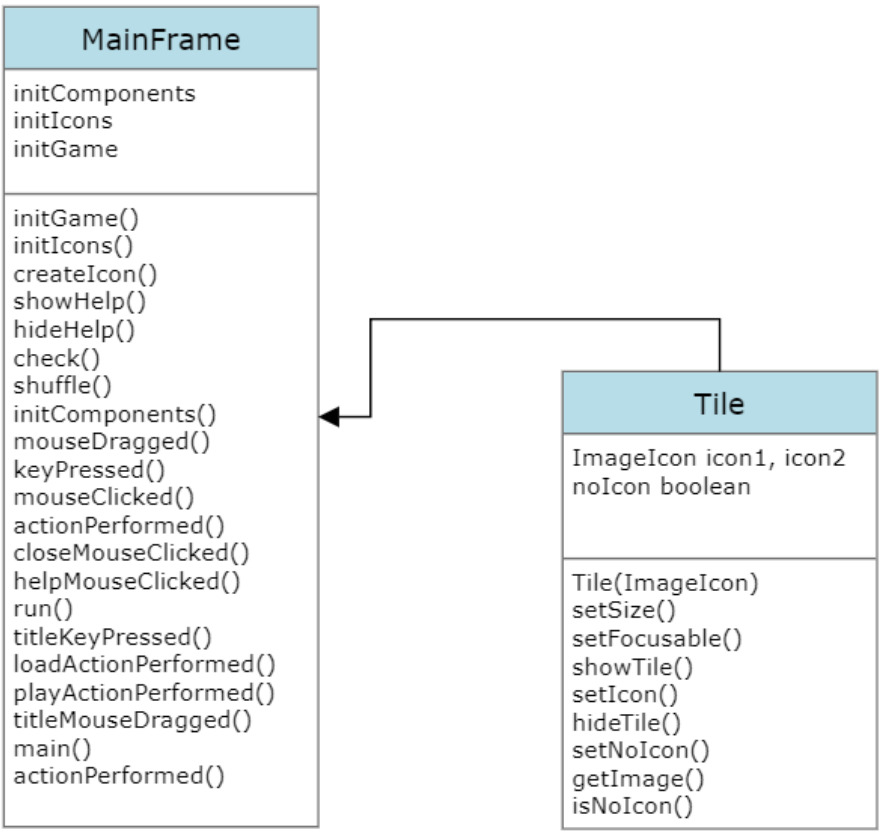
Anisa Dzikra Qalbiah

Program Description

Fruit Memory Game is a game with an end goal to clear all the pictures available by matching the same pictures to one another. The pictures are randomized, so the player must guess where the other picture is. Every time the player matches the wrong card the game will reduce the score, but if the player matches the correct picture, the game will give the player some score.

As a gamer who loves to play a lot of online games, sometimes when there is no connection available it is hard to entertain ourselves, so we move onto the offline games that usually is available on our device. As this game can be runned offline, it is hoped that it will cure some boredom.

Class Diagram



Design

Main Window

This window is the main window where the game is running. The '?' button when pressed will show the whole picture for a bit so if the player feel stuck, they can cheat a little bit using the button. The 'score' is where the game will be keeping track of the score. The 'PLAY' button is if you want a new set of picture to play, or in a simple way 'new game'. The 'LOAD' button is if you want to change the picture of the tiles.

Implementation and Explanation of The Code

This project consists of 3 files. The MainFrame file, the Sound file, and the Tile file. This project uses mainly Javax Swing and other utilities.

MainFrame.java

```
public class MainFrame extends javax.swing.JFrame implements ActionListener {
    public MainFrame() {
        initComponents();
        initIcons();
        initGame();
    }

    private void initGame() {
        score = 0;
        int x = 0;
        for (int i = 0; i < tiles.length; i++) {
            tiles[i] = new Tile(icons[x], new ImageIcon(getClass().getResource("/images/logo.png")));
            tiles[i].addActionListener(this);
            gamePanel.add(tiles[i]);
            if ((i + 1) % 2 == 0) {
                x++;
            }
        }
        title.setText("Score: " + score);
        shuffle();

    }

    private void initIcons() {
        Image img;
        for (int i = 0; i < icons.length; i++) {
            img = new ImageIcon(getClass().getResource("/images/card" + i + ".png")).getImage();
            icons[i] = createIcon(img);
        }

    }
```

This is where the initialization for the main frame happen. The MainFrame will initialize the components, icons, and the game itself.

initGame initializes the top of the window frame where the scoring will count, also where each of the back tiles will be putted on the logo.

initIcons initializes the icons or picture that will be shown on the tile. With the preferred length of the icon, is this case 18. The getResource method will collect the picture from the files, then it will proceed to create the icon.

```java
//creating the icon from the image that has been taken
private ImageIcon createIcon(Image img) {
    BufferedImage bi = new BufferedImage(img.getWidth(null), img.getHeight(null),
    bi.createGraphics().drawImage(img, 0, 0, null);
    img = bi.getScaledInstance(80, 80, 1);
    return new ImageIcon(img);
}

private void showHelp() {
    //show help or hint, so everytime you look at hint the score -50
    //if tiles[0] would be null than all the tiles would be null here
    if (tiles[0] != null) {
        for (int i = 0; i < tiles.length; i++) {
            if (!tiles[i].isNoIcon()) {
                tiles[i].showTile();
                tiles[i].removeActionListener(this);
            }
        }
        score -= 50;
        title.setText("Score: " + score);
    }
}
```

createIcon will create the dimension of the picture/image that have been taken from the file. It will only get the width and height hence (0, 0, null). Then it will get scaled into (80, 80, 1) as a new ImageIcon.

showHelp will happens when you want to look at the hints, so when you clicked on the hint all of the tiles will unfold and the answer can be visible, hence the isNoIcon will return noIcon, then it will show the tile for about 10 seconds. Every time the hint is pressed the player will lose 50 points of their score. The player can't also click on the tiles to prevent getting the answer while the answer is revealed, hence the removeActionListener.

```
//hiding(folding) the help/hint on its own
private void hideHelp() {

    for (int i = 0; i < tiles.length; i++) {
        if (!tiles[i].isNoIcon()) {
            tiles[i].hideTile();
            tiles[i].addActionListener(this);
        }
    }

}
```

hideHelp is when the program will hide the tile after showing the hint to the player hence hideTile, then player can now click on the tiles to continue the game hence the addActionListener.

```
//check whether the user win or not
private void check() {

    if (predict1 != predict2 && predict1.getImage() == predict2.getImage()) {
        new Thread() {
            @Override
            public void run() {
                for (int i = 0; i < 3; i++) {
                    try {
                        predict1.hideTile();
                        predict2.hideTile();
                        Thread.sleep(100);
                        predict1.showTile();
                        predict2.showTile();
                        Thread.sleep(100);
                    } catch (InterruptedException ex) {
                        System.out.println(ex);
                    }
                }
                predict1.setNoIcon();
                predict2.setNoIcon();
                for (int i = 0; i < tiles.length; i++) {
                    if (!tiles[i].isNoIcon()) {
                        won = false;
                        break;
                    }

                    else {
                        won = true;
                    }
                }
```

check is when the program will check whether the user have won the game or lose the game. If there is no more icon(tile) in the window, it means that the player have cleared the game, hence the isNoIcon.

```
                if (won) {
                    if (score > 0) {
                        JOptionPane.showMessageDialog(gamePanel, "You Won! Your S
                    }

                    else {
                        JOptionPane.showMessageDialog(gamePanel, "You Loose! Your
                    }
                    initGame();
                }
            }
        }.start();//animation if the user is correct
        predict1.removeActionListener(this);
        predict2.removeActionListener(this);
        score += 100;
        title.setText("Score: " + score);

    } else {//if user is wrong
        predict1.hideTile();
        predict2.hideTile();
        score -= 10;
        title.setText("Score: " + score);
    }
}
```

If the player end score is more than 0, then the program will show a message dialog that said the player have won with the score printed after. But, if the player score is below 0, then the program will show a message dialog that said the player have lose with the score printed after (ex/ -70).

If the player match the tile with one another, +100 will be added to score and the tile will be removed from the window, but if the user mismatched the tile, 10 points will be reduced from the score and the tile will be folded once again.

```
//shuffling/randomizing the tiles
private void shuffle() {
    gamePanel.removeAll();
    ArrayList<Integer> al = new ArrayList<~>();
    for (int i = 0; i < 36;) {
        int x = (int) (Math.random() * 36);
        if (!al.contains(x)) {
            al.add(x);
            i++;
        }
    }
    for (int i = 0; i < 36; i++) {
        gamePanel.add(tiles[al.get(i)]);
        tiles[al.get(i)].hideTile();
    }
}
```

shuffle is when the game will shuffle or randomize the tiles every time the player open the game or pressed the PLAY button. It will first remove the previous tiles from the window, then it will randomized it in an new arrayList using also the Math.random. When the tiles are already randomized, the hideTile() will runs.

```
//initializing the components of the frame
private void initComponents() {

    titlePanel = new javax.swing.JPanel();
    title = new javax.swing.JTextField();
    close = new javax.swing.JLabel();
    help = new javax.swing.JLabel();
    gamePanel = new javax.swing.JPanel();
    controlPanel = new javax.swing.JPanel();
    play = new javax.swing.JButton();
    load = new javax.swing.JButton();

    //getting the name for the main frame, colors, and the logo
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Matching Game");
    setBackground(new java.awt.Color(255, 255, 255));
    setIconImage(new ImageIcon(getClass().getResource("/images/logo.png")).getIm
    setLocationByPlatform(true);
    setName("MainFrame");
    setUndecorated(true);

    //background and dimension for the title bar/panel
    titlePanel.setBackground(new java.awt.Color(0, 0, 0));
    titlePanel.setPreferredSize(new java.awt.Dimension(300, 25));
    titlePanel.setLayout(new java.awt.BorderLayout());
```

initComponents is where the program initializes the components that will be put onto the MainFrame. It includes the titlePanel, title, close, help, gamePanel, controlPanel, play, and load.

setDefaultCloseOperation is when the user clicks the ‘X’ button on the top of the window, the program will close.

setTitle is the title that will appear on top of the window frame.

setBackground is the background of the window.

setIconImage is the image icon of the program. getResource will get the logo from the file.

setLocationByPlatform sets whether the program will appear at the default location for the native windowing system.

setName is used to change the name of the thread into MainFrame.

setUndercorated is used to enables decorations for this frame.

setPreferredSize is used to set the preferred size of this component to a constant value.

setLayout to sets the layout manager for this container.

```
//setting the title bar/panel (font and color, placing)
title.setEditable(false);
title.setBackground(new java.awt.Color(0, 0, 0));
title.setFont(new java.awt.Font("Tahoma", 1, 11));
title.setForeground(new java.awt.Color(255, 255, 255));
title.setHorizontalAlignment(javax.swing.JTextField.CENTER);
title.setText("Score: ");
title.setToolTipText("After Clicking mouse here use arrow keys to move");
title.setBorder(null);
title.setCursor(new java.awt.Cursor(java.awt.Cursor.MOVE_CURSOR));
title.setSelectionColor(new java.awt.Color(0, 0, 0));
title.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
    public void mouseDragged(java.awt.event.MouseEvent evt) {
        titleMouseDragged(evt);
    }
});
title.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        titleKeyPressed(evt);
    }
});
titlePanel.add(title, java.awt.BorderLayout.CENTER);
```

setEditable is used so the user cannot type into the field.

setFont is used to set the font of the text.

setHorizontalAlignment is used to sets the horizontal position of the text, in this case CENTER.

setToolTipText is when the method will register the component as having a tool tip.

setBorder is set to null as the border will not be shown on the panel.

setCursor is the move cursor type.

setSelectionColor is to set color of selected features.

The user can only dragged and pressed the title panel.

The title is also placed on the CENTER of the border layout.

```
//the X button to close bar
close.setBackground(new java.awt.Color(0, 0, 0));
close.setFont(new java.awt.Font("Tahoma", 1, 14));
close.setForeground(new java.awt.Color(255, 255, 255));
close.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
close.setText("X");
close.setToolTipText("Close");
close.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
close.setPreferredSize(new java.awt.Dimension(25, 25));
close.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        closeMouseClicked(evt);
    }
});
titlePanel.add(close, java.awt.BorderLayout.LINE_END);
```

The setCursor is set into HAND CURSOR, as when the user hover over the ‘X’ button the cursor will turn into hand.

The user can clicked on the ‘X’ button then the program will close.

The ‘X’ is placed on the LINE END of the border layout.

```
//the help option at the bar
help.setFont(new java.awt.Font("Tahoma", 1, 18));
help.setForeground(new java.awt.Color(255, 255, 255));
help.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
help.setText("?");
help.setToolTipText("Right click to hide controls and Left click to see
help.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
help.setPreferredSize(new java.awt.Dimension(25, 25));
help.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        helpMouseClicked(evt);
    }
});
titlePanel.add(help, java.awt.BorderLayout.LINE_START);

getContentPane().add(titlePanel, java.awt.BorderLayout.NORTH);
```

The '?' is placed on the LINE START of the border layout.

getContentPane returns the contentPane object for the frame, so the whole titlePanel will be put NORTH of the border layout.

```
//color for the background of the game panel
gamePanel.setBackground(new java.awt.Color(0, 0, 0));
gamePanel.setPreferredSize(new java.awt.Dimension(630, 630));
gamePanel.setLayout(new java.awt.GridLayout(6, 6, 5, 5));
getContentPane().add(gamePanel, java.awt.BorderLayout.CENTER);

//color for control panel
controlPanel.setBackground(new java.awt.Color(0, 0, 0));
controlPanel.setPreferredSize(new java.awt.Dimension(300, 40));
controlPanel.setLayout(new java.awt.GridLayout(1, 2));

//color and font for the play button
play.setBackground(new java.awt.Color(255, 255, 255));
play.setFont(new java.awt.Font("Tahoma", 0, 18));
play.setForeground(new java.awt.Color(153, 153, 255));
play.setText("PLAY");
play.setToolTipText("Play new Game");
play.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
play.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        playActionPerformed(evt);
    }
});
controlPanel.add(play);
```

The gamePanel is where the tiles are placed, also where the game will be played.

getContentPane returns the contentPane object for the frame, so the whole gamePanel will be put CENTER of the border layout.

The controlPanel is where the PLAY button and the LOAD button will be placed.

The play option is where the new game will be play.

The PLAY button is placed into the controlPanel.

```
    //color and font for the load button
    load.setBackground(new java.awt.Color(255, 255, 255));
    load.setFont(new java.awt.Font("Tahoma", 0, 18));
    load.setForeground(new java.awt.Color(153, 153, 255));
    load.setText("LOAD");
    load.setToolTipText("Load your favourite images");
    load.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    load.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            loadActionPerformed(evt);
        }
    });
    controlPanel.add(load);

    getContentPane().add(controlPanel, java.awt.BorderLayout.SOUTH);

    pack();
}
```

The load option is where the player can change the image of the tiles.

The LOAD button is placed into the controlPanel.

getContentPane returns the contentPane object for the frame, so the whole controlPanel will be put SOUTH of the border layout.

pack() is used to sizes the frame so all of the contents are at the preferred size.

```
//if mouse click close then it will close
private void closeMouseClicked(java.awt.event.MouseEvent evt) {
    if (evt.getButton() == MouseEvent.BUTTON1) {
        this.dispose();
    }
}
```

If the mouse click the ‘X’ button the program will “dispose”.

```java
//if mouse click help it will show help
private void helpMouseClicked(java.awt.event.MouseEvent evt) {
    if (evt.getButton() == MouseEvent.BUTTON1) {
        if (!helping) {
            new Thread() {
                @Override
                public void run() {
                    try {
                        helping = true;
                        showHelp();
                        Thread.sleep(10000); //time the hint will show
                        hideHelp();
                        helping = false;
                    } catch (InterruptedException ex) {
                        System.out.println(ex);
                    }
                }
            }.start();
        }
    }
    if (evt.getButton() == MouseEvent.BUTTON3) {
        if (controlPanel.isVisible()) {
            setSize(600, 625);
            controlPanel.setVisible(false);
        } else {
            setSize(600, 665);
            controlPanel.setVisible(true);
        }
    }
}
```

If the '?' button is clicked the images of the tiles will be visible for about 10 seconds then it will close again.

If the user right click the '?' button then the control panel will be not be visible (hidden). But if the user right click '?' once more the control panel will be visible again.

```
//what would happen if the title is pressed
private void titleKeyPressed(java.awt.event.KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_LEFT) {
        setLocation(getX() - 5, getY());
    }
    if (evt.getKeyCode() == KeyEvent.VK_RIGHT) {
        setLocation(getX() + 5, getY());
    }
    if (evt.getKeyCode() == KeyEvent.VK_UP) {
        setLocation(getX(), getY() - 5);
    }
    if (evt.getKeyCode() == KeyEvent.VK_DOWN) {
        setLocation(getX(), getY() + 5);
    }
}
```

This shows what will happen if the arrow keys on the keyboard are pressed. If the LEFT arrow is pressed the window will move left about 5 pixel. If the RIGHT arrow is pressed the window will move right about 5 pixel. If the UP arrow is pressed the window will move up about 5 pixel. If the DOWN arrow is pressed the window will move down about 5 pixel.

```
//what would happen if the LOAD button is clicked
private void loadActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser chooser = new JFileChooser();
    chooser.setMultiSelectionEnabled(true);
    int response = chooser.showOpenDialog(predict1);
    if (response == JFileChooser.APPROVE_OPTION) {
        File[] file = chooser.getSelectedFiles();
        if (file.length >= 18) {
            for (int i = 0; i < 18; i++) {
                icons[i] = createIcon(new ImageIcon(file[i].toString()).ge
            }
            initGame();
        }
        else {
            JOptionPane.showMessageDialog(gamePanel, "Please select 18 Fi
        }
    }
}
```

The loadActionPerformed is when the LOAD button is pressed then the program will pop open the window where the user can choose the 18 images for the new tiles. Then, the program will create a new ImageIcon for the new tiles. If the player did not choose at least 18 images, the message dialog will pop up for the user to select 18 files.

```
//what would happen if the PLAY button is clicked
private void playActionPerformed(java.awt.event.ActionEvent evt) {
    initGame();
}

//what would happen if the title gets dragged
private void titleMouseDragged(java.awt.event.MouseEvent evt) {
    setLocation(evt.getXOnScreen() - 300, evt.getYOnScreen());
}

//calling for the app
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            new MainFrame().setVisible(true);
        }
    });
}
```

playActionPerformed is when the game will initialized when the PLAY button is pressed.

titleMouseDragged is when the title panel will get dragged and moved on the setLocation.

main is where the program will run at first and the MainFrame will be visible.

```
Tile[] tiles = new Tile[36]; //how many tiles in a game
ImageIcon[] icons = new ImageIcon[18]; //how many image needed
int status, score;
Tile predict1, predict2;
private boolean won, helping;

//declaring the variable on main frame
private javax.swing.JLabel close;
private javax.swing.JPanel controlPanel;
private javax.swing.JPanel gamePanel;
private javax.swing.JLabel help;
private javax.swing.JButton load;
private javax.swing.JButton play;
private javax.swing.JTextField title;
private javax.swing.JPanel titlePanel;
```

This is where the program declares all the variable on the MainFrame.

```java
@Override
//if a tile gets clicked, what would happen
public void actionPerformed(ActionEvent e) {
    if (status == 0) {
        predict1 = (Tile) e.getSource();
        predict1.showTile();
        status++;
    }
    else if (status == 1) {
        status++;
        predict2 = (Tile) e.getSource();
        new Thread() {
            @Override
            public void run() {
                try {
                    predict2.showTile();
                    Thread.sleep(500);
                    check();
                    Thread.sleep(600);
                    status = 0;
                }
                catch (Exception e) {
                    System.out.println(e);
                }
            }
        }.start();

    }
}
```

actionPerformed is when the user clicks on a tile, the tile will stays up hence the showTile. But if the user matches the tile, the “animation” will starts.

Title.java

```
class Tile extends JButton {

    ImageIcon icon1;
    ImageIcon icon2;
    private boolean hidden, noIcon;

    //setting the tile
    public Tile(ImageIcon icon1, ImageIcon icon2) {
        this.icon1 = icon1;
        this.icon2 = icon2;
        setSize(100, 100);
        setFocusable(false);
    }

    //show tile
    public synchronized void showTile() {
        setIcon(icon1);
        hidden = false;
    }

    //hiding the tile
    public synchronized void hideTile() {
        setIcon(icon2);
        hidden = true;
    }
```

icon1 is the front of the tile or the picture of the tile, while icon2 is is the logo or the back of the tile.

setFocusable is the focusable state of the component to a specific value.

showTile is when the picture of the tile will be shown, hence the hidden is false because the picture is not hidden.

hideTile is when the back of the tile will be shown, hence the hidden is true because the picture is hidden.

```
//if setting no icon
public synchronized void setNoIcon() {
    setIcon(null);
    noIcon = true;
}

//getting image
public ImageIcon getImage() {
    return icon1;
}

//if there is no icon
public synchronized boolean isNoIcon() {
    return noIcon;
}
```

setNoIcon is when the setIcon is null then it means noIcon will be shown, hence noIcon true.

getImage is when the image or picture of the tile has been chosen, hence it returns icon1(the picture of the tile).

isNoIcon is when there is not tiles on the main window, hence it returns noIcon.

Video Demo Links

https://drive.google.com/drive/folders/1ZKUb1r_P_DxApD89mbDqduRv3MrVXcDl?usp=sharing

References and Helpful Links

https://www.tutorialspoint.com/swing/swing_mouse_listener.htm#:~:text=The%20class%20which%20processes%20the,using%20the%20addMouseListener()%20method.

https://www.javatpoint.com/java-thread-setname-method

https://www.javatpoint.com/java-class-getresource-method

https://mkyong.com/java/how-to-resize-an-image-in-java/

https://docs.oracle.com/javase/7/docs/api/java/awt/Frame.html

https://docs.oracle.com/javase/7/docs/api/java/awt/Container.html#setLayout(java.awt.LayoutManager)

https://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html

https://docs.oracle.com/javase/tutorial/uiswing/components/tooltip.html

https://docs.oracle.com/javase/tutorial/uiswing/components/border.html

https://docs.oracle.com/javase/7/docs/api/java/awt/Cursor.html

https://docs.oracle.com/javase/6/docs/api/java/awt/Window.html#setLocationByPlatform%28boolean%29

https://www.tutorialspoint.com/when-can-we-use-the-pack-method-in-java

https://www.thoughtco.com/create-a-simple-window-using-jframe-2034069

https://www.youtube.com/watch?v=HgkBvwgciB4