BINUS UNIVERSITY
BINUS INTERNATIONAL

Student information :	Surname:	Given Names	Student ID Number
1	Widjaja	Tiffany	2502055596

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

Class : L2CC **Name of Lecturer(s) :** Jude Joseph Lamug Martinez

Major : Computer Science

Title of Assignment : PrimoClicker

Type of Assignment : Final Project

Submission Pattern

Due Date: 6/13/2022 **Submission Date:** 6/12/2022

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's 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)
Tiffany Widjaja

Table of Content

Chp. 1 – Project Specification:

A. Program Description:

- The program I've decided to create for my final OOP project is a game called PrimoClicker. The concept of the game is inspired by an incremental web browser game called Cookie Clicker where the goal of the game is to produce as many cookies as the user can. In this case, I have decided to create a Genshin-themed version of it and so instead of clicking on cookies, the user will be clicking on the giant Primogem in order to generate one of the most wanted in-game currencies in Genshin called Primogems. The game also offers upgrades that the user can purchase with the acquired Primogems. These upgrades are in the form of Genshin Characters as they will help you generate X amount of Primogems per second independently, thus helping the user to acquire Primogems faster.

B. Program Flow Summary:w

- Upon running the program, the first thing that will pop up on the user's screen is a window that shows the title of the game "PrimoClicker" and a red button that says "CLICK!" with a chibi Paimon on top of it. The user will then be expected to click on the red button as it is designed to stand out with the selected quirky font.
- Upon pressing the red button, the main game will appear where the Primogem count and the number of Primogems produced per second is still 0 since the user has not clicked on the giant Primogem. Once the user has clicked on the giant Primogem, the amount of Primogems you have will start showing according to how many times you have clicked on it. The Primogems produced per second will still show nothing.
- The user will then notice on the right of the giant Primogem, there is a list of upgrades with the name called "Amber" and the rest with just a "?". Upon hovering over the name, the user will see a description of how much Primogems the upgrade gives and its cost. Hovering over the "?" will only give the user a pop-up message that says "This item is currently locked".
- Once the user has obtained the amount of desired Primogems from clicking, the user will purchase one of the upgrades and so on which will increase the amount of Primogems

produced per second. After each purchase, the price of the upgrade will go up by X amount, with the latter upgrades being more costly. Eventually, the user will have all upgrades available and shown after the amount of Primogems owned satisfies the cost of the upgrade.

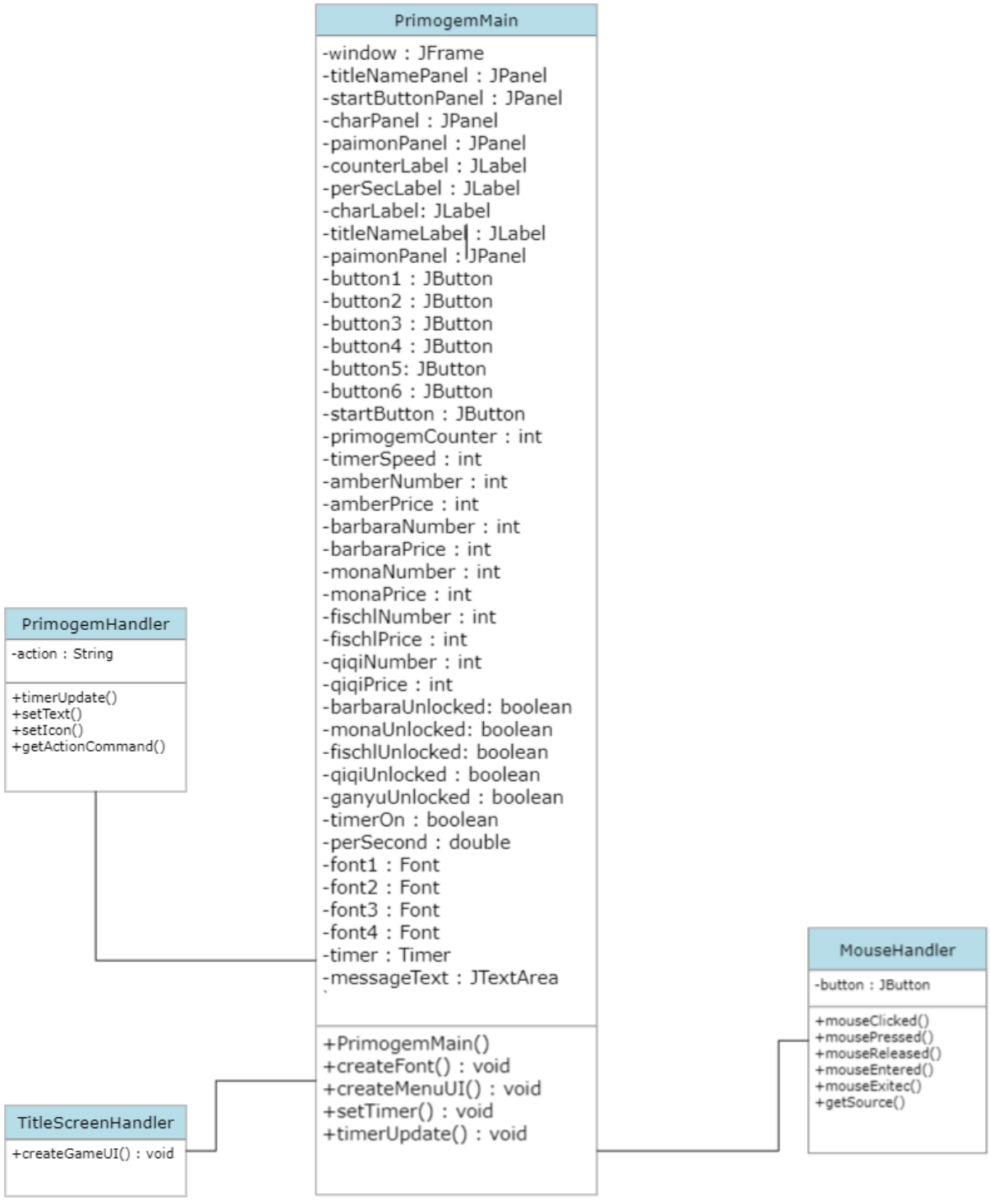
- The characters available for purchase are Amber, Barbara, Mona, Fischl, Qiqi, and Ganyu. Any recent character upgrade purchased will be shown with the chibi that appears on the right side of the window. The user can easily obtain Primogems while being idle after getting the upgrades and have all the Primogems they want.

C. IDE & Imported Libraries:

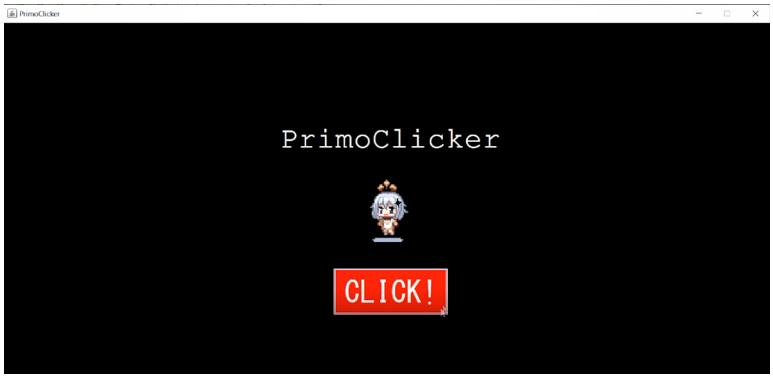
- Eclipse IDE for Java Developers (2022) is the IDE used to write the code for this program as well as to run it. The reason why I chose this IDE was that I wasn't able to load images into the window using Visual Studio Code as the resources folder was missing. It was much easier as well for me to use the Eclipse and adapt to it compared to when I tried VS Code for this project and so I decided to switch.
- Libraries:
    - javax.swing.JFrame
    - javax.swing.ImageIcon
    - javax.swing.JLabel
    - javax.swing.JButton
    - javax.swing.JPanel
    - javax.swing.Timer
    - javax.swing.JTextArea
    - java.awt.Color
    - java.awt.Font;
    - java.awt.GridLayout;
    - java.awt.event.ActionEvent;
    - java.awt.event.ActionListener;
    - java.awt.event.MouseEvent;
    - java.awt.event.MouseListener;

Chp. 2 - Solution Design:

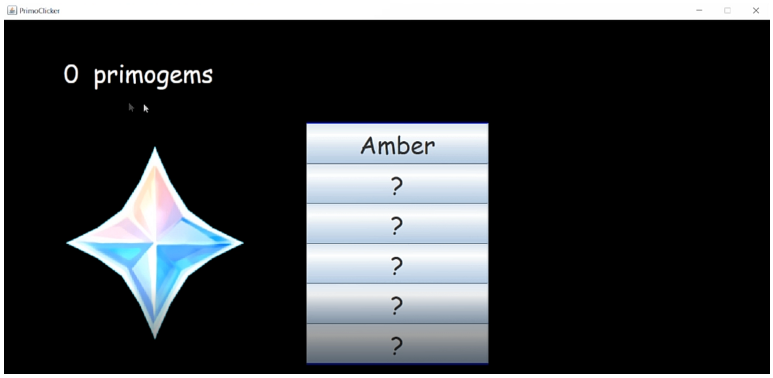
A. Class Diagram:



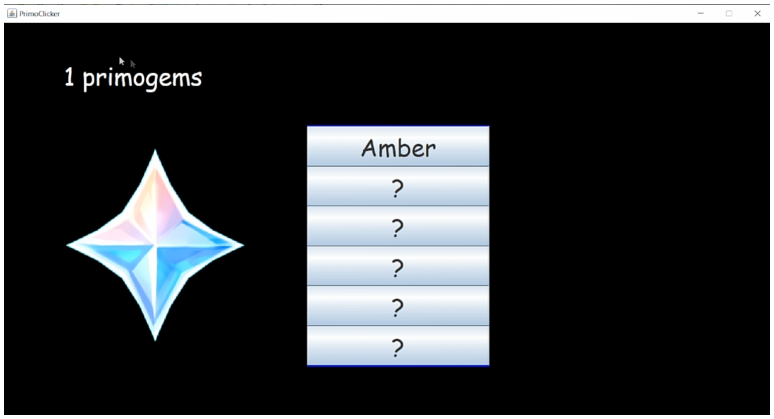
B. Program Flow:



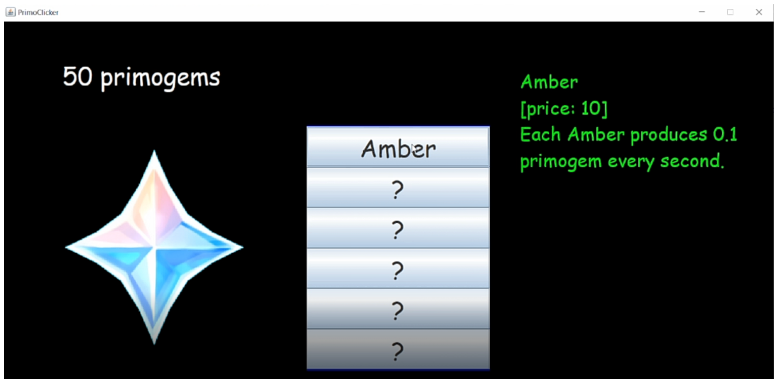
Pressing the red "CLICK!" Button will lead to the createGameUI()

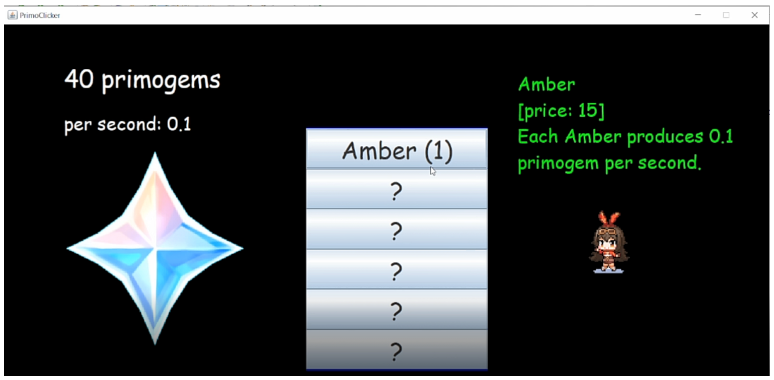


After clicking on the giant Primogem one time.

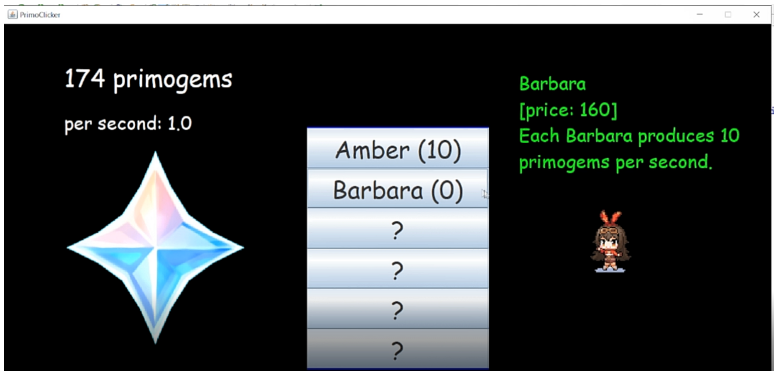


Hovering over the "Amber" Button that displays the cost and its Primogems production rate.

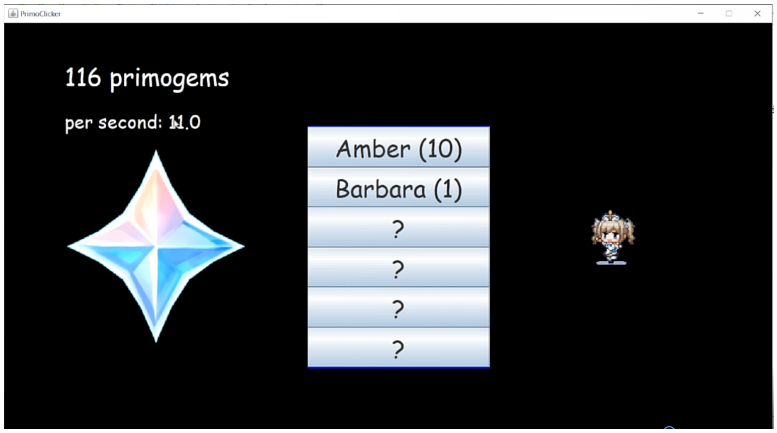




After purchase, most recent bought character appears.



Reaches a certain amount of owned primogems and thus unlocked Barbara.



One Barbara is then purchased.

Chp. 3 - Code Explanation:

A. PrimogemMain.Java:

```
public class PrimogemMain { //class for main program
        JFrame window = new JFrame(); //using JFrame to set up window
        JPanel titleNamePanel, startButtonPanel, charPanel, paimonPanel; //Jpanels
        JLabel counterLabel, perSecLabel, charLabel, titleNameLabel, paimonLabel; //JLabels
        JButton button1, button2, button3, button4, button5, button6, startButton; //JButtons
        int primogemCounter, timerSpeed, amberNumber, amberPrice, barbaraNumber, barbaraPrice, monaNumber, monaPrice, fischlNumber, fischlPrice, qiqiNumber, qiqiPrice, ganyuNu
        double perSecond; //for the primogem persecond
        boolean timerOn, barbaraUnlocked, monaUnlocked, fischlUnlocked, qiqiUnlocked, ganyuUnlocked; //boolean type
        Font font1, font2, font3, font4; //fonts
        PrimogemHandler pHandler = new PrimogemHandler();
        Timer timer; //Timer from javax.swing.Timer called timer
        JTextArea messageText; //JtextArea
        MouseHandler mHandler = new MouseHandler();
        TitleScreenHandler TsHandler = new TitleScreenHandler();

    public static void main(String[] args) {
        new PrimogemMain();
    }
```

After importing the libraries needed to create this program which is shown before this, I decided to name my main class PrimogemMain and start off creating a window using JFrame. I also initialized panels, buttons and labels using JPanel, JLabel and JButton. Also Initialized primogemCounter, timerSpeed, xPrice, xNumber as an int. Initialized perSecond as a double since a double allows the number to be with a decimal point unlike int. timerOn, xUnlocked are initialized as boolean types. font 1, 2 and so on using Font. Created pHandler, mHandler and TsHandler for the classes later. messageText is initialized using JTextArea. The PrimogemMain() is then called in the public static void main.

```
public PrimogemMain(){ //initializing variables
    timerOn = false;
    perSecond = 0;
    primogemCounter = 0;
    //setting the characters initial values of (
    amberNumber = 0;
    amberPrice = 10;
    barbaraNumber = 0;
    barbaraPrice = 160;
    barbaraUnlocked = false;
    monaNumber = 0;
    monaPrice = 800;
    monaUnlocked = false;
    fischlNumber = 0;
    fischlPrice = 1600;
    fischlUnlocked = false;
    qiqiNumber = 0;
    qiqiPrice = 12000;
    qiqiUnlocked = false;
    ganyuNumber = 0;
    ganyuUnlocked = false;
    ganyuPrice = 28800;
    //calling the methods
    createFont();
    createMenuUI();
}
```

Here in PrimogemMain(), I set all the variables into what I want it to initially hold a value of. For instance, the timerOn is set to false and primogemCounter as 0 since that is how we want it to be at the beginning. There I have set the different prices for each character and making sure that they are not unlocked yet by setting it to false first. I also called a createFont() and createMenuUI(). Inside createFont(), it is just 4 different types of fonts that are available.

```
public void createFont() { //function to create different fonts
    font1 = new Font("Comic Sans MS", Font.PLAIN, 40); //(font name, font style, font size)
    font2 = new Font("Comic Sans MS", Font.PLAIN, 30);
    font3 = new Font("DialogInput", Font.PLAIN, 50);
    font4 = new Font("Monospaced", Font.PLAIN, 50);
}
```

Next, I created a method called createMenuUI() which is to display the starting screen that the user sees when they first launch the program. That's why window.setSize is used, which is to set up the window's resolution. window.setDefaultCloseOperation is just to make sure that it closes the window properly and to prevent it from getting any errors. To set the window's background color as black, I had to use window.getContentPane().setBackground(Color.Black). As to set the window title as PrimoClicker, I used window.setTitle() and make sure that the user cannot resize the window by using window.setResizable(false). Subsequently, to decorate the menu, I created 3 panels using JPanel() for the title, start button, and the Paimon image. I made sure to set the background and the bounds for each panel using .setBounds() and .setBackground(). For the titleNamePanel, I needed to create a label using JLabel() so that it can display the title of the game. With setFont() and setSize(), it allowed me to choose my own desired font and size to set the label. I did the same for the paimonPanel since I needed to display the image on the panel by creating a label first and then adding that label into the panel. It's a bit different since I had to create a new ImageIcon called paimonpixel and use setIcon(paimonpixel) to attach it to the label. Lastly using window.add(paimonpanel) to display the whole panel containing the label and the image icon on the window screen.

```
public void createMenuUI() { //method to create menu user interface

    window.setSize(1280, 720);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //to close the window prope
    window.getContentPane().setBackground(Color.black); //to set the window's backgrou
    window.setTitle("PrimoClicker"); //sets the window title
    window.setLayout(null); //to disable the default layout
    window.setResizable(false); //make sure window cannot be resized

    titleNamePanel = new JPanel(); //Panel for the game title
    titleNamePanel.setBackground(Color.black);
    titleNamePanel.setBounds(340, 150, 600, 300); //setting x,y,width,height of the pa
    titleNameLabel = new JLabel("PrimoClicker"); // label for the game title
    titleNameLabel.setFont(font3); //applying the font already existing
    titleNameLabel.setSize(700,700); //to set size of the label
    titleNameLabel.setForeground(Color.white); //to set the foreground color of the la
    titleNamePanel.add(titleNameLabel); //adding the title to the label

    startButtonPanel = new JPanel(); //panel for the start button
    startButtonPanel.setBounds(240, 400, 800, 200); //setting x,y,width,height of the
    startButtonPanel.setBackground(Color.black); //setting background color of panel a

    paimonPanel = new JPanel(); //create panel for Paimon
    paimonPanel.setBounds(535, 250, 200, 150); //setting x,y,width,height of the panel
    paimonPanel.setBackground(Color.black); //set background color for paimon panel
    paimonLabel = new JLabel(); //create label for paimon
    paimonPanel.add(paimonLabel); //adding the label to the panel
    window.add(paimonPanel); //adding the panel to the window
    ImageIcon paimonpixel = new ImageIcon(getClass().getResource("paimonpixel.png"));
    paimonLabel.setIcon(paimonpixel); //setting icon of paimon into the Label
```

```
startButton = new JButton("CLICK!"); //the button to start the game that will direct to the cookie
startButton.setSize(900, 500); //setting width height size of button
startButton.setBackground(Color.red); //set button background color
startButton.setForeground(Color.WHITE); //set foreground button color
startButton.setFont(font4); //applying the font for the text in start button
startButton.addActionListener(TsHandler);
startButton.setFocusPainted(false); //setting it false so that there is no blue focus ring around
startButtonPanel.add(startButton); //adding the start button into the start button panel
window.add(startButtonPanel); //adding the panels into the window
window.add(titleNamePanel);

window.setVisible(true); //making the window appeal on the screen
```

For the start button, I used JButton() and also added an ActionListener() called TsHandler so that when the user clicks on the

start button, it will fire an action event. Lastly, to make sure that everything will appear on the window, I had to set the window.setVisible() as true.

```
//function to create the primoclicker UI
public void createGameUI(){
    titleNamePanel.setVisible(false); //making panels from the menu screen invisible
    startButtonPanel.setVisible(false);
    paimonPanel.setVisible(false);
    window.setSize(1280, 720); //Setting window size
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    window.getContentPane().setBackground(Color.black);
    window.setTitle("PrimoClicker"); //setting window title
    window.setLayout(null);
    window.setResizable(false); //not allowing user to fullscreen the window

    JPanel primoPanel = new JPanel();
    primoPanel.setBounds(100, 200, 300, 600);
    primoPanel.setBackground(Color.black);
    window.add(primoPanel);

    JPanel charPanel = new JPanel();
    charPanel.setBounds(900, 300, 200, 150);
    charPanel.setBackground(Color.black);
    window.add(charPanel);

    ImageIcon primogem = new ImageIcon(getClass().getResource("primogemimg.png")); //to get pict

    JButton primogemButton = new JButton();
    primogemButton.setBackground(Color.black);
    primogemButton.setFocusPainted(false); //disable the focus line around the button
    primogemButton.setBorder(null); //disable the border around the button
    primogemButton.setIcon(primogem); //set the primogem image as an icon on the button
    primogemButton.addActionListener(pHandler); //phandler gets called when button gets clicked
    primogemButton.setActionCommand("primogem");
    primogemButton.setContentAreaFilled(false);
    primoPanel.add(primogemButton);
```

The next step was to create the main game and so I created a createGameUI() method. Inside the method, I made sure to set all the panels from createMenuUI() method to disappear by setting the visibility to false. This time I added more panels for the giant primogem and the characters which are then added to the window. Also loaded the image of the giant primogem the same way I loaded the paimonpixel image earlier. Also created a button using JButton for primogemButton so that the user can click on the giant primogem. I set the FocusPainted as false for the primogem button so that there will be no focus line around it when the user clicks on the Primogem. I also made sure to disable the border around the button by setting it to null. Also added in actionlistener for the primogem so when it gets clicked, it calls pHandler. I made setActionCommand as “primogem” for the switch case later. Lately added the button once again into the panel.

```
JPanel counterPanel = new JPanel();
counterPanel.setBounds(100,50,400,150);
counterPanel.setBackground(Color.black);
counterPanel.setLayout(new GridLayout(2,1)); //(vertical column, horizontal column)
window.add(counterPanel); //panel for the primogem counter added

charLabel = new JLabel();
charPanel.add(charLabel);

counterLabel = new JLabel(primogemCounter + "  primogems");
counterLabel.setForeground(Color.white);
counterLabel.setFont(font1);
counterPanel.add(counterLabel);

perSecLabel = new JLabel();
perSecLabel.setForeground(Color.white);
perSecLabel.setFont(font2);
counterPanel.add(perSecLabel);

JPanel itemPanel = new JPanel();
itemPanel.setBounds(500, 170, 300, 400);
itemPanel.setBackground(Color.blue);
itemPanel.setLayout(new GridLayout(6,1));
window.add(itemPanel);

button1 = new JButton("Amber");
button1.setFont(font1);
button1.setFocusPainted(false);
button1.addActionListener(pHandler);
button1.setActionCommand("amber");
button1.addMouseListener(mHandler);
itemPanel.add(button1);
```

```
button2 = new JButton("?");
button2.setFont(font1);
button2.setFocusPainted(false);
button2.addActionListener(pHandler);
button2.setActionCommand("barbara");
button2.addMouseListener(mHandler);
itemPanel.add(button2);

button3 = new JButton("?");
button3.setFont(font1);
button3.setFocusPainted(false);
button3.addActionListener(pHandler);
button3.setActionCommand("mona");
button3.addMouseListener(mHandler);
itemPanel.add(button3);

button4 = new JButton("?");
button4.setFont(font1);
button4.setFocusPainted(false);
button4.addActionListener(pHandler);
button4.setActionCommand("fischl");
button4.addMouseListener(mHandler);
itemPanel.add(button4);

button5 = new JButton("?");
button5.setFont(font1);
button5.setFocusPainted(false);
button5.addActionListener(pHandler);
button5.setActionCommand("qiqi");
button5.addMouseListener(mHandler);
itemPanel.add(button5);
```

To keep track of how many Primogems the user has, I created a counterPanel and a counterLabel. Also created an itemPanel which will store a button that displays the purchasable character names such as “Amber”, “Barbara”, “Qiqi” and so on. Of course, I have created more buttons for the characters that can be bought while displaying the rest as “?” at first other than “Amber”. perSecLabel to keep track of the Primogems produced per second. I also created another panel that says messagePanel which will store a JTextArea called messageText. In this part, I have not set any text message yet and only added it to the panel first.

```
JPanel messagePanel = new JPanel();
messagePanel.setBounds(850, 70, 500, 200);
messagePanel.setBackground(Color.black);
window.add(messagePanel);

messageText = new JTextArea();
messageText.setBounds(700, 70, 500, 150);
messageText.setForeground(Color.green);
messageText.setBackground(Color.black);
messageText.setFont(font2);
messageText.setLineWrap(true);
messageText.setWrapStyleWord(true);
messageText.setEditable(false);
messagePanel.add(messageText);

window.setVisible(true);
```

```
public void setTimer(){ //function to set timer for when to show names of the characters available to be purchased + how much of it has been bought
        timer = new Timer(timerSpeed, new ActionListener(){ //parameters consist of timerspeed
                @Override
                public void actionPerformed(ActionEvent e){ //when timer activates, everything here runs and repeats
                        primogemCounter++; //increase primo number by 1
                        counterLabel.setText(primogemCounter + " primogems");

                        if(barbaraUnlocked==false){
                                if(primogemCounter>=160){
                                        barbaraUnlocked=true;
                                        button2.setText("Barbara " + "(" + barbaraNumber + ")");
                                }
                        }
```

Then I created a function called setTimer() which is to start the timer the moment the giant Primogem gets clicked which will be checked by the actionPerformed. Here if the primogem gets clicked, the primogemCounter will increase by 1 and it will then be displayed in the counterLabel using .setText(). It will also keep track of the characters where if the primogem counter reaches a certain amount, the characterUnlocked will be changed from false to true which will then display the character's name Instead of a "?".

```
public void timerUpdate(){

        if(timerOn==false){
                timerOn=true;
        }
        else if(timerOn==true){
                timer.stop();
        }
        double speed = 1/perSecond*1000;
        timerSpeed = (int)Math.round(speed); //how often it increases cookie number

        String s = String.format("%.1f", perSecond);
        perSecLabel.setText("per second: " + s); //to display how many primogems generated per second

        setTimer();
        timer.start();
    }
```

Here I had to create a function called timerUpdate() which updates the timer for each time a new character is hired/bought to increase the passive Primogem income. The speed is set as a double since if I were to use an int, it will not work since the start of producing a Primogem is 0.1 for Amber. When the timerOn is false or in other words the timer is not on, then we will start the timer. If it is already on, then the timer gets stopped. This is where I set the text for the

persecLabel which is to display how many primogems is generated every second. The setTimer() is then called inside and also starts it.

```
public class TitleScreenHandler implements ActionListener{
    public void actionPerformed(ActionEvent event) {
        createGameUI();
    }
}
```

TitleScreenHandler implements ActionListener() was for when you click on the start button in the starting menu, it will call the createGameUI() method.

```
public class PrimogemHandler implements ActionListener {
    public void actionPerformed(ActionEvent event) {

            String action = event.getActionCommand();

            switch(action){ //using switch case
            case "primogem":
                    primogemCounter++; //++ = +1 when primogem button gets click, the counter increase +1
                    counterLabel.setText(primogemCounter + " primogems"); //setting text on the label for the primogem counter
                    break;

            case "amber":
                    if(primogemCounter>=amberPrice) { //when primogems obtained is = amber's price or even more
                            primogemCounter = primogemCounter - amberPrice; //when amber is bought, primogem counter will be updated
                            amberPrice= amberPrice + 5; //price of amber increases everytime its bought
                            counterLabel.setText(primogemCounter + " primogems"); //set text for the counter label of how much primogems owned (updated)

                            amberNumber++; //+1
                                    button1.setText("Amber " + "(" + amberNumber + ")"); //number of ambers purchased gets updated in the button
                                    messageText.setText("Amber\n[price: " + amberPrice + "]\nEach Amber produces 0.1 primogem per second."); //set text when
                                    ImageIcon amberpixel = new ImageIcon(getClass().getResource("amberpixel.png")); //load amber pixel image as imageicon
                                    charLabel.setIcon(amberpixel); //set Icon of amber pixel into a label

                                    perSecond = perSecond + 0.1; //auto clicks every 10 second
                                    timerUpdate(); //call timer update function
                    }
                    else {
                            messageText.setText("\n You need more primogems!"); //if primogems is not enough, it will display this text instead
                    }
                    break;
```

Here in the PrimogemHandler class which implements the interface ActionListener is where all the events take place. If Primogem gets clicked then it will increase the Primogem counter. For the case “amber” and other characters, the case only works once the PrimogemCounter reaches the number of Primogems needed to buy the character. primogemCounter will be updated and deducted by the price of the character. Once the user clicks on the button, not only does the number of characters you have increase by one each click but the price as well. For instance, each amber price goes up by 5 Primogems for each time it is bought. Of course the PrimogemCounter would also need to be updated and by that we have to setText for the Label once more. Here I wanted to added in an image pixel to show the recent characters bought so each time a button of one of them is clicked, it will display an image of the character. timerUpdate function is also called to update the time.

```
public class MouseHandler implements MouseListener{ //class of mousehandler

            @Override // helps prevent the case when you write a function that you think overrides another one
            public void mouseClicked(MouseEvent e){

            }
            @Override
            public void mousePressed(MouseEvent e){

            }
            @Override
            public void mouseReleased(MouseEvent e){

            }
            @Override
            public void mouseEntered(MouseEvent e){ //when mouse is hovering some object
                    JButton button = (JButton)e.getSource(); //determines which button you press your mouse on

                    if(button == button1){
                            messageText.setText("Amber\n[price: " + amberPrice + "]\nEach Amber produces 0.1 primogem every second.");
                    }

                    else if(button == button2){
                            if(barbaraUnlocked==false){ //if it is still locked, it will display a text
                                    messageText.setText("\n\nThis item is\ncurrently locked!");
                            }
                            else{ //if it is not locked, it will display the description of the character and upgrade
                                    messageText.setText("Barbara\n[price: " + barbaraPrice + "]\nEach Barbara produces 10 primogems per second.");
                            }
                    }
```

Here by using mousehandler and mouselistener, if the mouse hovers an object like a button for instance, it will display a text message like hovering a button that displays “?”. It will state that it is currently locked still when characterUnlocked is still false. If it is not, it will display the description of the setText. The way it detects this is by using the (JButton)e.getSource().

```
    }
    @Override
    public void mouseExited(MouseEvent e){ //function when mouse is not hovering oveer the button, it will run this

            JButton button = (JButton)e.getSource();

            if(button == button1){
                    messageText.setText(null); //when it detects that pointer is not hovering over the button, the message text will disappear
            }
            else if(button == button2){
                    messageText.setText(null);
            }
            else if(button == button3){
                    messageText.setText(null);
            }
            else if(button == button4){
                    messageText.setText(null);
            }
            else if(button == button5){
                    messageText.setText(null);
            }
            else if(button == button6){
                    messageText.setText(null);
            }
    }
```

Here mouseExited function is used to see if the mouse is not hovering over something by our pointer. Lets say it is not hovering over the button, the text message that was supposed to be visible if we hover it will disappear if we leave the button and by that we set it to null.

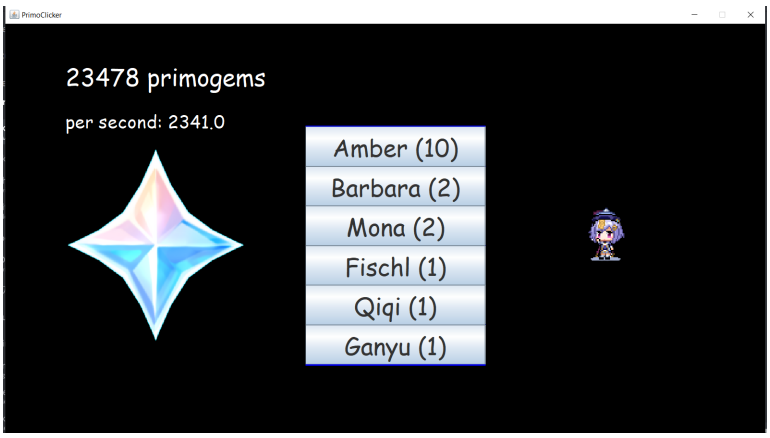
Chp. 4 - Proof of Working Program:

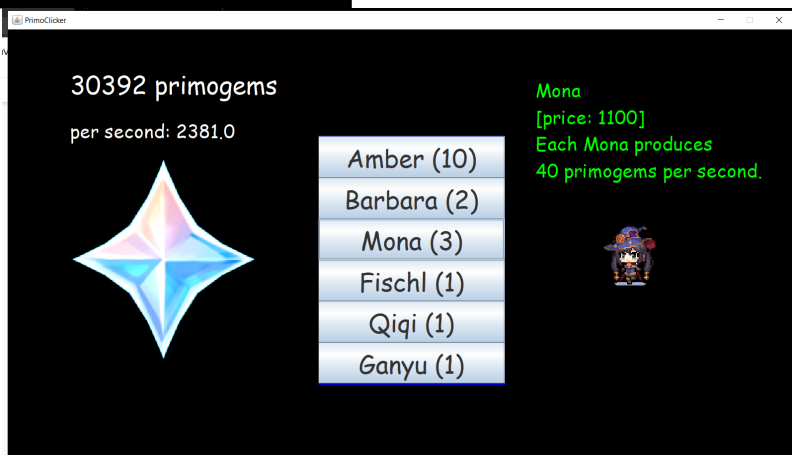
A. Program Files:

- Github Repository:

https://github.com/Pandalmation/PrimoClicker

B. Screenshots:





Resources:

https://www.youtube.com/watch?v=5AEIgLzueHk&list=PL_QPQmz5C6WXSKlDFM_cXEhA4X85Wh-X8&index=1&ab_channel=RyiSnow

- **Stackoverflow**
- **Nicholas' method of how to make the elements disappear**