BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	**Surname**	**Given Names**	**Student ID Number**
1.	Claudio	Alexandro Joe	2501963160

Course Code : COMP6510 **Course Name** : Object Oriented Programming

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

Major : Computer Science

Title of Assignment : NOMUK

Type of Assignment : Final Project

Submission Pattern

Due Date : **Submission Date** :

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)

Title of Content

Demo Video Link :

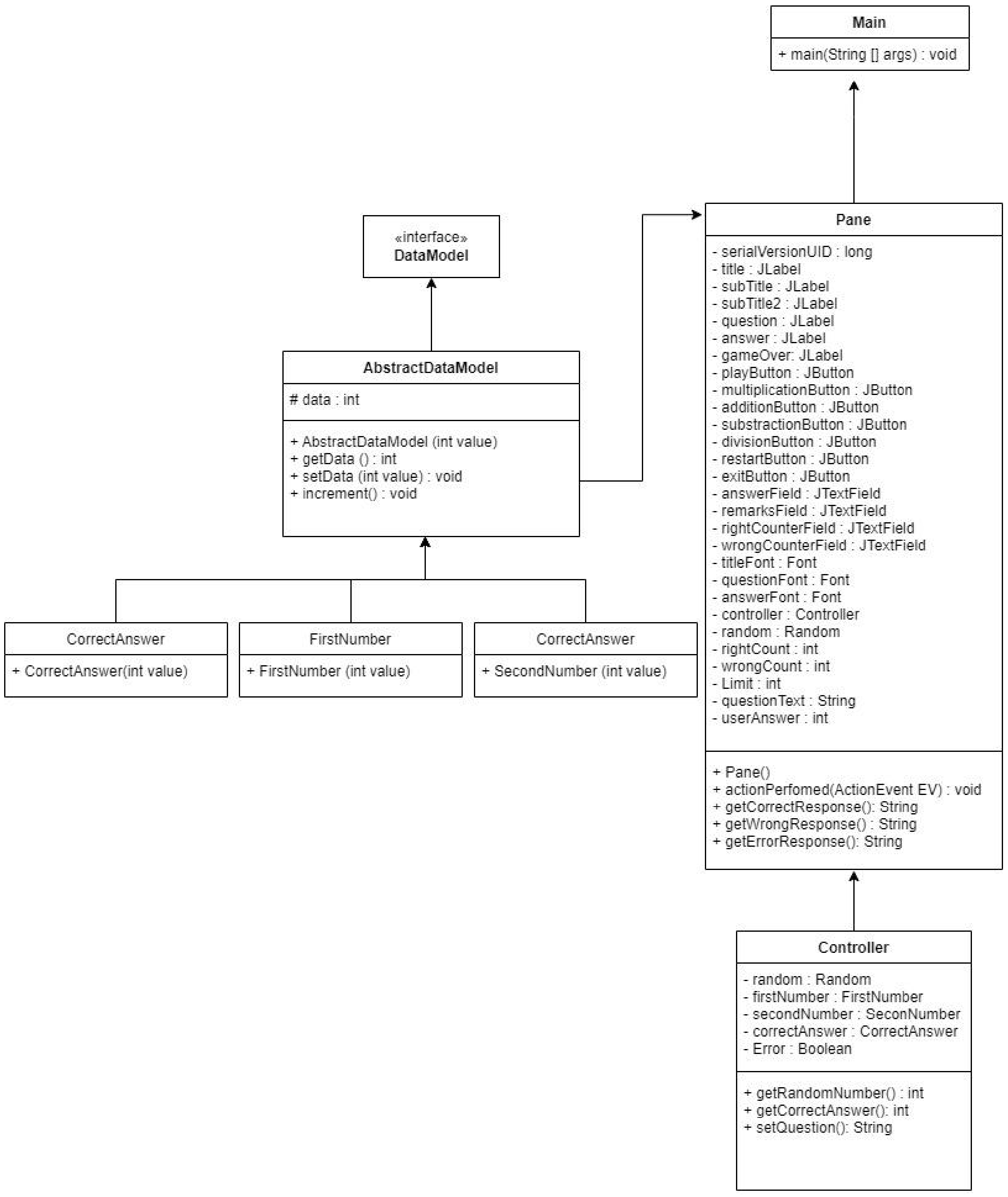
https://drive.google.com/file/d/1-8R086ttZx9cYH8OSewajKIOMTrYgygR/view?usp=sharing

I. Program Description

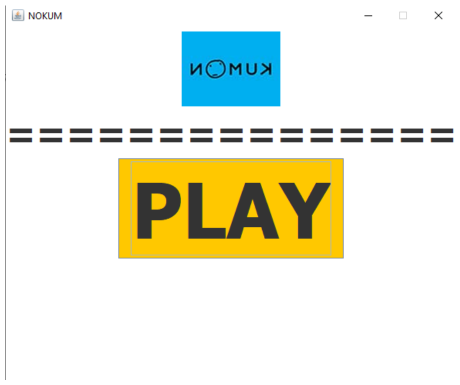
When I read the final project description, I was confused about what I am going to create. At first, I thought that I will make a car rental management. But then I realized, it will be easier if I make something from a simple daily problem. At that moment I still don’t get what I am about to make. Until one day, I had a small family gathering and I saw my little cousin doing Kumon, then I have the idea to make a Kumon program.

So I called it Nomuk, the backward spelling of Kumon, the program has 4 operators, multiplication, addition, subtraction, and division. The numbers will be randomized. The goal of the program is to make math exercises for the kids.

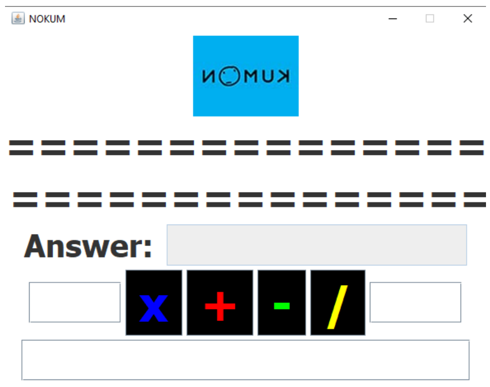
II. Class Diagram


Main
+ main(String [] args) : void
«interface»
DataModel
AbstractDataModel
# data : int
+ AbstractDataModel (int value)
+ getData () : int
+ setData (int value) : void
+ increment() : void
CorrectAnswer
+ CorrectAnswer(int value)
FirstNumber
+ FirstNumber (int value)
CorrectAnswer
+ SecondNumber (int value)
Pane
- serialVersionUID : long
- title : JLabel
- subTitle : JLabel
- subTitle2 : JLabel
- question : JLabel
- answer : JLabel
- gameOver: JLabel
- playButton : JButton
- multiplicationButton : JButton
- additionButton : JButton
- substractionButton : JButton
- divisionButton : JButton
- restartButton : JButton
- exitButton : JButton
- answerField : JTextField
- remarksField : JTextField
- rightCounterField : JTextField
- wrongCounterField : JTextField
- titleFont : Font
- questionFont : Font
- answerFont : Font
- controller : Controller
- random : Random
- rightCount : int
- wrongCount : int
- Limit : int
- questionText : String
- userAnswer : int
+ Pane()
+ actionPerfomed(ActionEvent EV) : void
+ getCorrectResponse(): String
+ getWrongResponse() : String
+ getErrorResponse(): String
Controller
- random : Random
- firstNumber : FirstNumber
- secondNumber : SeconNumber
- correctAnswer : CorrectAnswer
- Error : Boolean
+ getRandomNumber() : int
+ getCorrectAnswer(): int
+ setQuestion(): String

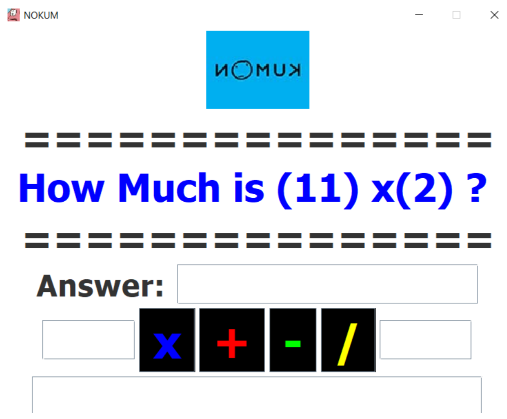
III. Application Flow



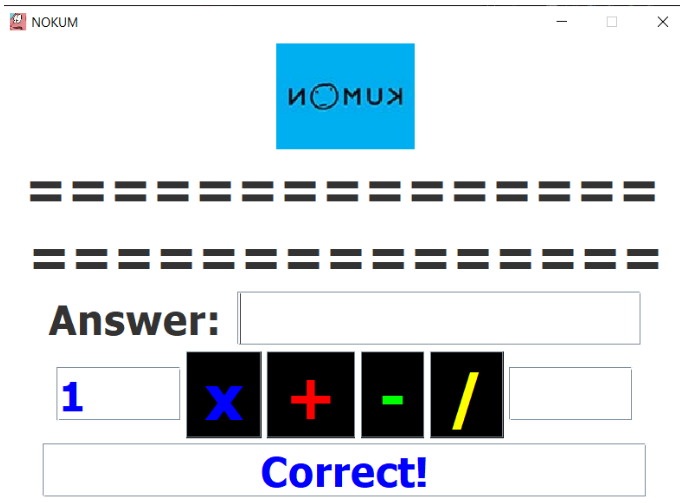
1. This is the starting screen when the user hit the run button



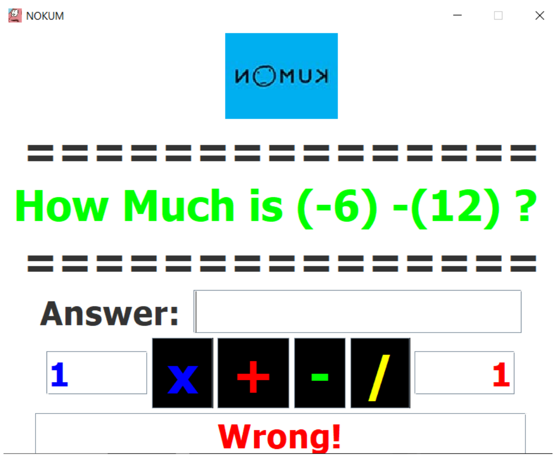
2. This is the screen after the user presses the play button, still empty and the question is not yet generated.



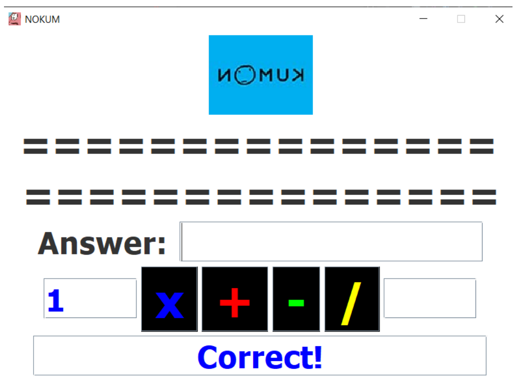
3. This is the screen after I press on the operators, I can change which kind of question I want by clicking on the operator symbol. The user can change the question without answering it first and is not limited to 1 operator.



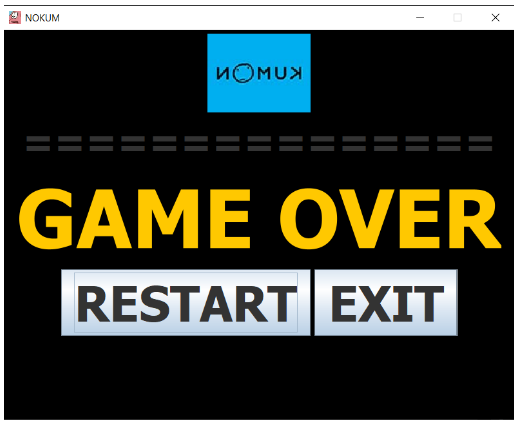
4. This is the screen after the user enters a correct answer, the question area will be empty so that the user can generate again by clicking on the operator they want. The right counter on the left side will increment by 1.



→



5. This is the screen after the user enters a wrong answer. After the user enters a wrong number the wrong counter will increment by 1. The question will stay in the question area so that the user can read again and re-answer it. After the user enters a wrong number the wrong counter will increment by 1. After entering the right answer, the screen will be the same as the fourth step.

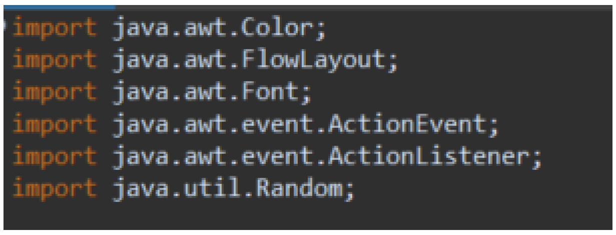


6. This is the final screen, I set the limit of the wrong answer to 10. After 10 wrong attempts this screen will appear, then there will be 2 options. If the user chooses to restart it will go back to the image on number 1. If the user chooses to exit it will close.

IV. Code Explanation

4.1 Libraries

To start the code explanation I am going to list down the libraries that I imported. For this project, I used some Java external libraries to help me.

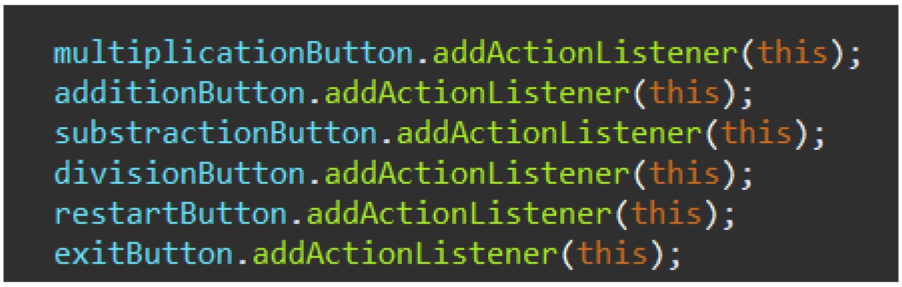


```
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;
```

#Pane.java

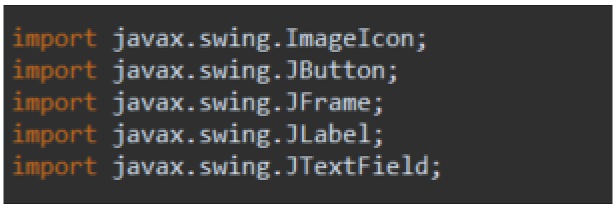
On this program I use java.util.random in some cases. The first one is to generate responses to the answer. The other thing is to generate random responses to the user's answer. I also use the font to set up fonts in different elements.

I used java.awt.Color to create colors on the elements that I have. I used java.awt.FlowLayout to arrange the elements on the frame. Action event and action listener is the important thing because I have some buttons so when the user does an action, the listener will receive it and I set action listener to some buttons.



```
multiplicationButton.addActionListener(this);
additionButton.addActionListener(this);
substractionButton.addActionListener(this);
divisionButton.addActionListener(this);
restartButton.addActionListener(this);
exitButton.addActionListener(this);
```

#Pane.java

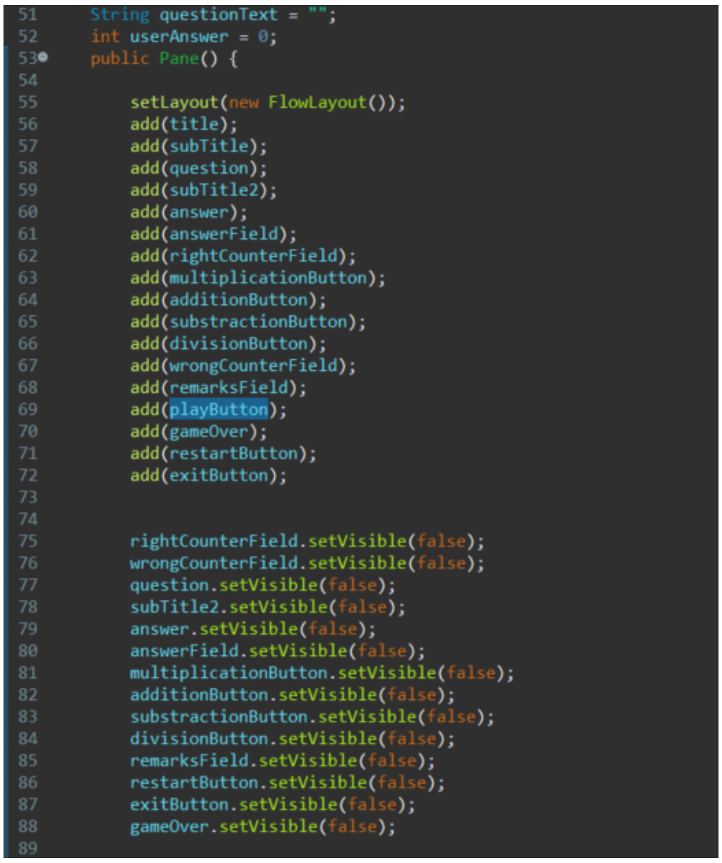


```
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
```

#Pane.java

For the GUI, I choose Swing, so then I import them. Image Icon is to make icons painted from URL or local file that I have on our computer. JButton is to make a button so that later on I can implement some events on the button. JTextField is where I put some information on the screen, in this program I print the question set, right counter, wrong counter, and responses to the question. JFrame is basically a container to provide a window on the screen, where I can place things like buttons, labels, text fields, etc.

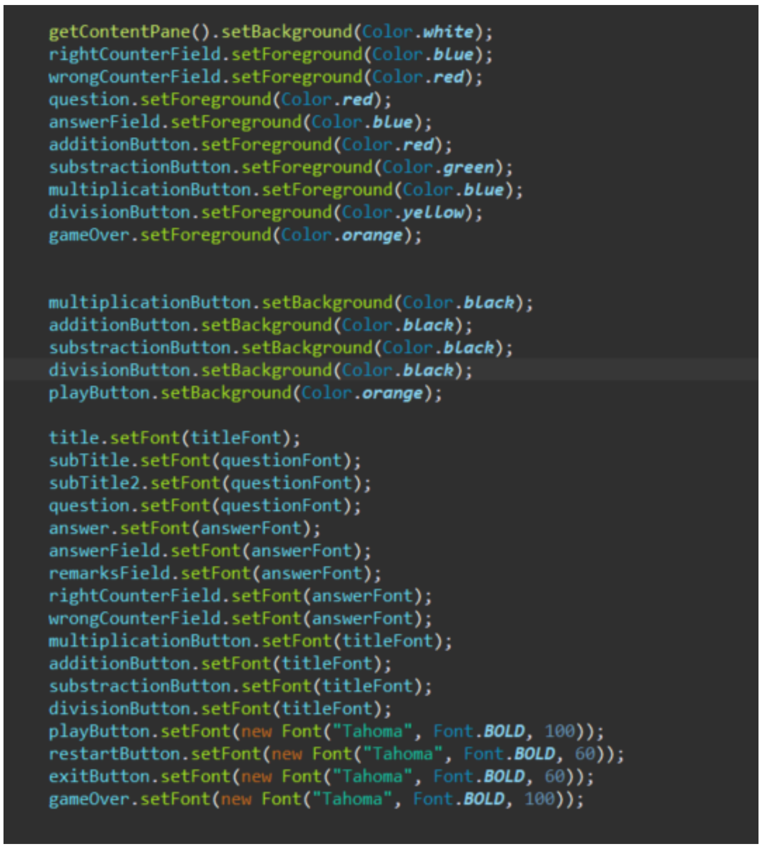
4.2 Start of the program



```java
    String questionText = "";
    int userAnswer = 0;
    public Pane() {

        setLayout(new FlowLayout());
        add(title);
        add(subTitle);
        add(question);
        add(subTitle2);
        add(answer);
        add(answerField);
        add(rightCounterField);
        add(multiplicationButton);
        add(additionButton);
        add(substractionButton);
        add(divisionButton);
        add(wrongCounterField);
        add(remarksField);
        add(playButton);
        add(gameOver);
        add(restartButton);
        add(exitButton);


        rightCounterField.setVisible(false);
        wrongCounterField.setVisible(false);
        question.setVisible(false);
        subTitle2.setVisible(false);
        answer.setVisible(false);
        answerField.setVisible(false);
        multiplicationButton.setVisible(false);
        additionButton.setVisible(false);
        substractionButton.setVisible(false);
        divisionButton.setVisible(false);
        remarksField.setVisible(false);
        restartButton.setVisible(false);
        exitButton.setVisible(false);
        gameOver.setVisible(false);

```

#Pane.java

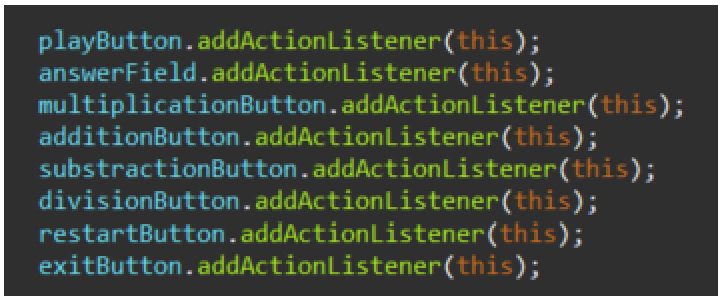
At the start of the program, as seen on the application flow, there is a screen with the title image and the play button. In this image, I add all of the elements using the flow layout. To set the other elements and make the play button and the title visible, I set all to false except the title and play button.



```
getContentPane().setBackground(Color.white);
rightCounterField.setForeground(Color.blue);
wrongCounterField.setForeground(Color.red);
question.setForeground(Color.red);
answerField.setForeground(Color.blue);
additionButton.setForeground(Color.red);
substractionButton.setForeground(Color.green);
multiplicationButton.setForeground(Color.blue);
divisionButton.setForeground(Color.yellow);
gameOver.setForeground(Color.orange);


multiplicationButton.setBackground(Color.black);
additionButton.setBackground(Color.black);
substractionButton.setBackground(Color.black);
divisionButton.setBackground(Color.black);
playButton.setBackground(Color.orange);

title.setFont(titleFont);
subTitle.setFont(questionFont);
subTitle2.setFont(questionFont);
question.setFont(questionFont);
answer.setFont(answerFont);
answerField.setFont(answerFont);
remarksField.setFont(answerFont);
rightCounterField.setFont(answerFont);
wrongCounterField.setFont(answerFont);
multiplicationButton.setFont(titleFont);
additionButton.setFont(titleFont);
substractionButton.setFont(titleFont);
divisionButton.setFont(titleFont);
playButton.setFont(new Font("Tahoma", Font.BOLD, 100));
restartButton.setFont(new Font("Tahoma", Font.BOLD, 60));
exitButton.setFont(new Font("Tahoma", Font.BOLD, 60));
gameOver.setFont(new Font("Tahoma", Font.BOLD, 100));
```

#Pane.java

This part of the code is to set the foreground and the background color which I can use by importing the java.awt.color. Also, I set the font to different elements.

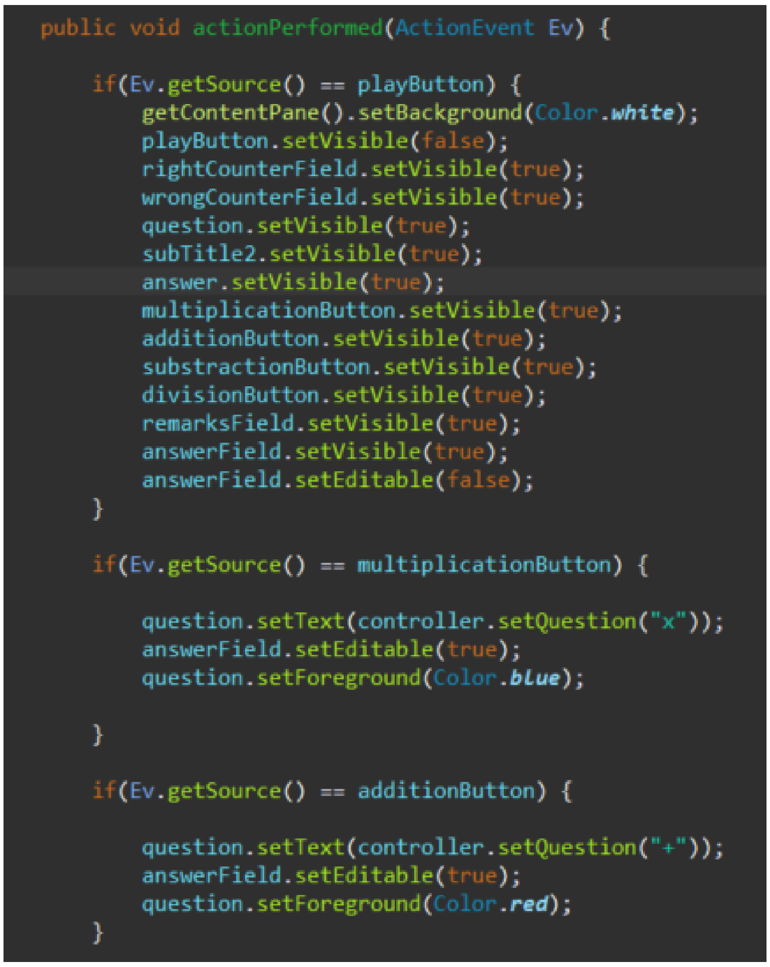


```
playButton.addActionListener(this);
answerField.addActionListener(this);
multiplicationButton.addActionListener(this);
additionButton.addActionListener(this);
substractionButton.addActionListener(this);
divisionButton.addActionListener(this);
restartButton.addActionListener(this);
exitButton.addActionListener(this);
```

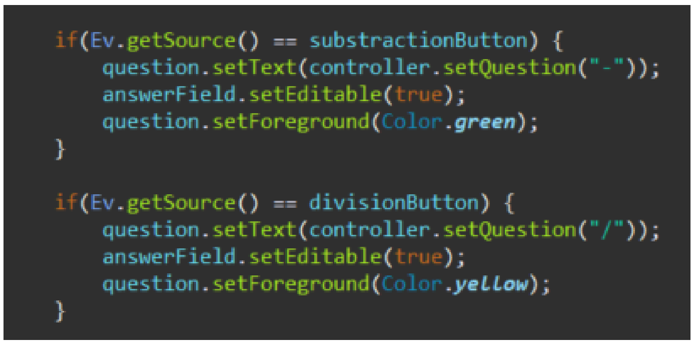
#Pane.java

This part of the code is where I add action listeners to the buttons so that the button can handle events from the user input.

4.3 Buttons



```
public void actionPerformed(ActionEvent Ev) {

    if(Ev.getSource() == playButton) {
        getContentPane().setBackground(Color.white);
        playButton.setVisible(false);
        rightCounterField.setVisible(true);
        wrongCounterField.setVisible(true);
        question.setVisible(true);
        subTitle2.setVisible(true);
        answer.setVisible(true);
        multiplicationButton.setVisible(true);
        additionButton.setVisible(true);
        substractionButton.setVisible(true);
        divisionButton.setVisible(true);
        remarksField.setVisible(true);
        answerField.setVisible(true);
        answerField.setEditable(false);
    }

    if(Ev.getSource() == multiplicationButton) {

        question.setText(controller.setQuestion("x"));
        answerField.setEditable(true);
        question.setForeground(Color.blue);

    }

    if(Ev.getSource() == additionButton) {

        question.setText(controller.setQuestion("+"));
        answerField.setEditable(true);
        question.setForeground(Color.red);
    }
```

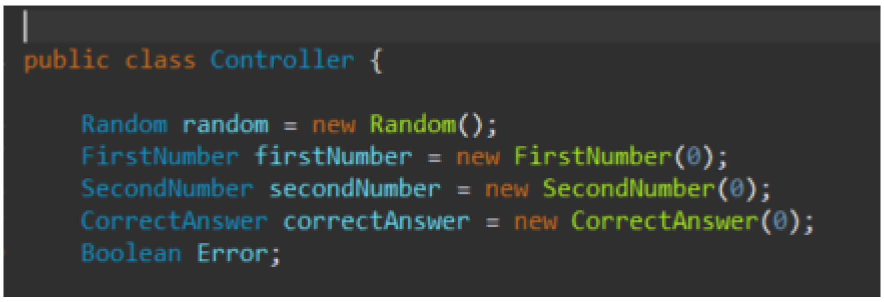


```
if(Ev.getSource() == substractionButton) {
    question.setText(controller.setQuestion("-"));
    answerField.setEditable(true);
    question.setForeground(Color.green);
}

if(Ev.getSource() == divisionButton) {
    question.setText(controller.setQuestion("/"));
    answerField.setEditable(true);
    question.setForeground(Color.yellow);
}
```

#Pane.java

Those 2 pictures of the code are under a function called actionPerfomed that accepts the action from the button that I already add actionListener to it. The first button which is the play button is quite different because when the user presses the button it will activate most of the elements. When the user presses the play button, the play button will disappear. I set the answerField is not editable because at this moment the question is still not generated, so I don’t want to create any error. For the multiplication, addition, subtraction, and division button, each of them has a different color that I already set. Also on those buttons, there is 1 function that I call which is question.setText(controller.setQuestion(operator). This is the part where the program prints randomized numbers to be the question and also set the correct answer. I set the setQuestion function on a different file (Controller.java).

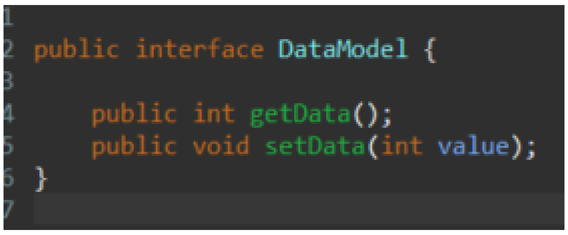
4.3 Set Question and Correct Answer



```java
public class Controller {

    Random random = new Random();
    FirstNumber firstNumber = new FirstNumber(0);
    SecondNumber secondNumber = new SecondNumber(0);
    CorrectAnswer correctAnswer = new CorrectAnswer(0);
    Boolean Error;
```

#Controller.java

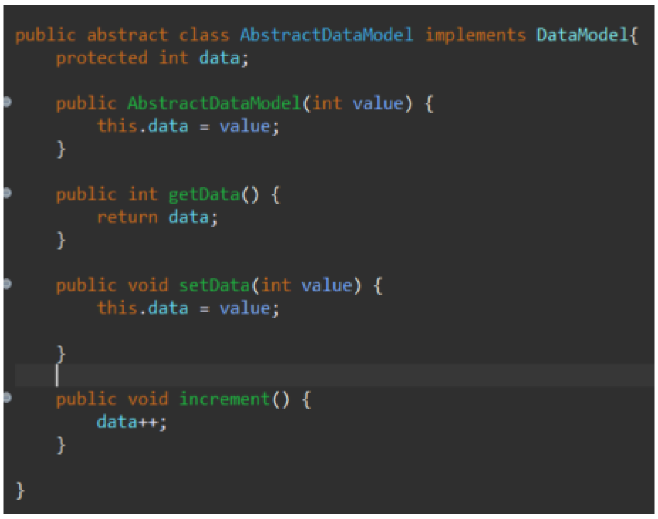
Before I explain about generating the question, I want to explain about the firstNumber, secondNumber, and CorrectAnswer. When I designed this, I feel like three of them have some similarities, which we have to set, then we can get the number, and increment.



```java
public interface DataModel {

    public int getData();
    public void setData(int value);
}
```

#DataModel.java

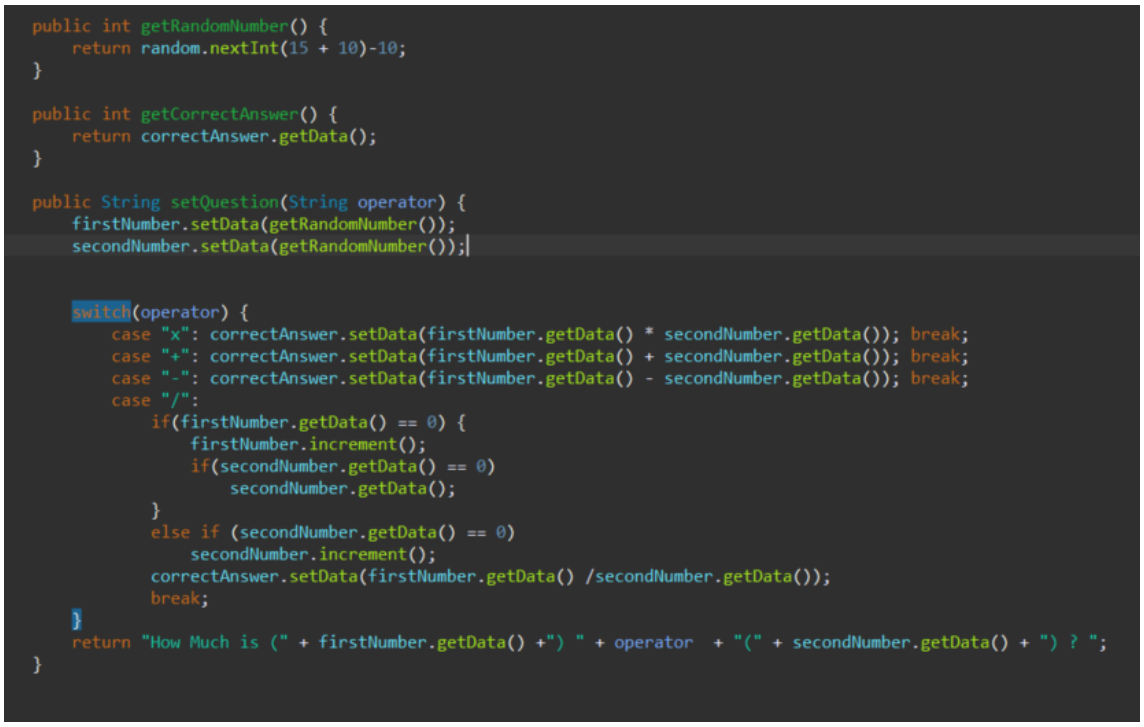
So then I make an interface class on java, which has 2 functions (getData and setData).



```java
public abstract class AbstractDataModel implements DataModel{
    protected int data;

    public AbstractDataModel(int value) {
        this.data = value;
    }

    public int getData() {
        return data;
    }

    public void setData(int value) {
        this.data = value;

    }

    public void increment() {
        data++;
    }

}
```

#AbstractDataModel.java

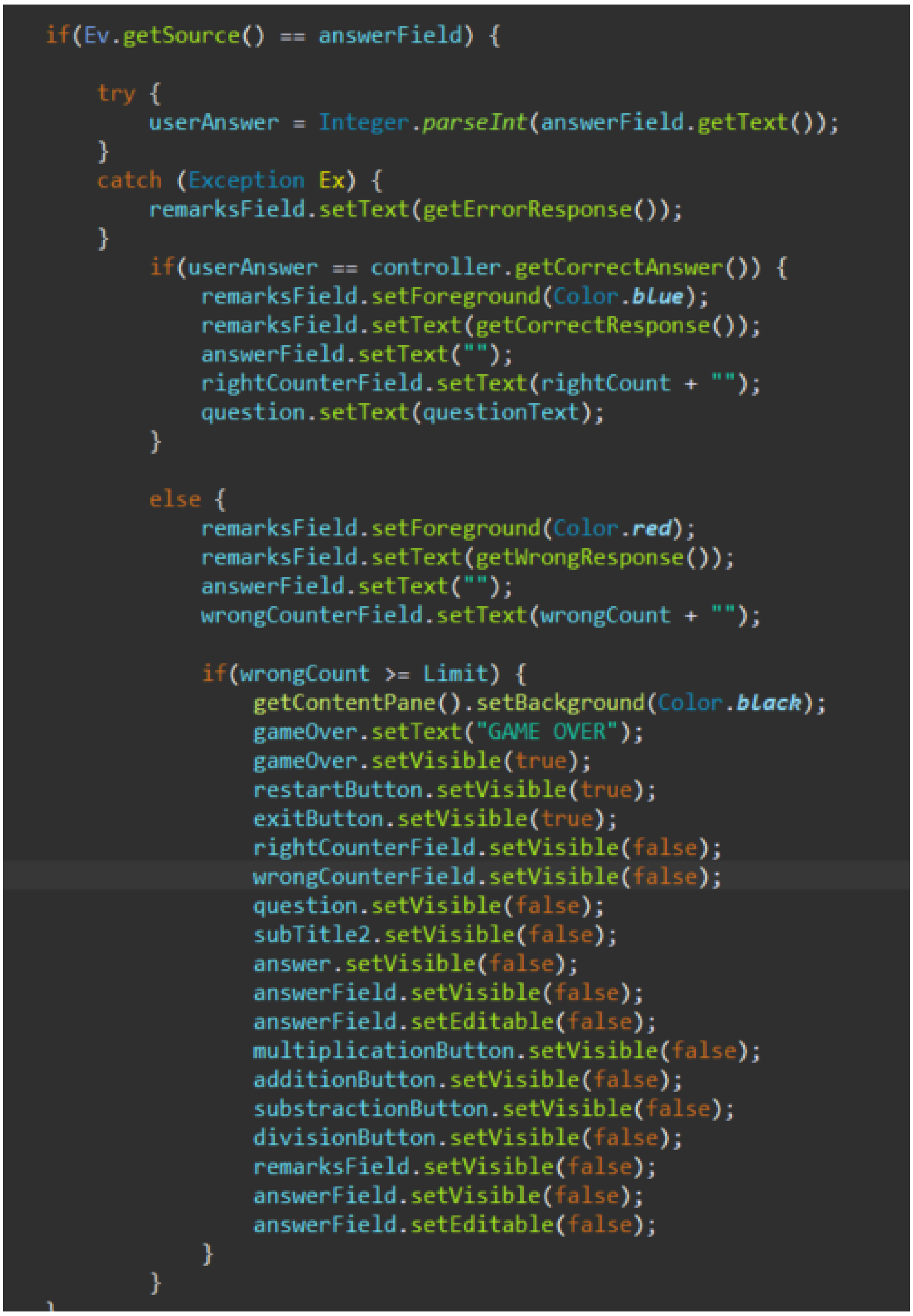
I also make the abstract class where it specifies the function of the interface class. There is getData where it returns data, setData so that I can set the value that is on the int type and increment. I use increment later on the setting up the question. After that, I make FirstNumber class, SecondNumber class, and CorrectAnswer class which extends the AbstractDataModel.



```java
public int getRandomNumber() {
    return random.nextInt(15 + 10)-10;
}

public int getCorrectAnswer() {
    return correctAnswer.getData();
}

public String setQuestion(String operator) {
    firstNumber.setData(getRandomNumber());
    secondNumber.setData(getRandomNumber());


    switch(operator) {
        case "x": correctAnswer.setData(firstNumber.getData() * secondNumber.getData()); break;
        case "+": correctAnswer.setData(firstNumber.getData() + secondNumber.getData()); break;
        case "-": correctAnswer.setData(firstNumber.getData() - secondNumber.getData()); break;
        case "/":
            if(firstNumber.getData() == 0) {
                firstNumber.increment();
                if(secondNumber.getData() == 0)
                    secondNumber.getData();
            }
            else if (secondNumber.getData() == 0)
                secondNumber.increment();
            correctAnswer.setData(firstNumber.getData() /secondNumber.getData());
            break;
    }
    return "How Much is (" + firstNumber.getData() +") " + operator  + "(" + secondNumber.getData() + ") ? ";
}
```

#Controller.java
At the button part (4.3 Button), I set each of the operators a parameter string on the controller.setQuestion that contain “x”,”+”, “-”, and “/”, this is where it works. For example, if the user presses the multiplication button, the multiplication button has the parameter “x”. Then it goes to the setQuestion function on the image above. I use the switch to make it easier, so when the user clicks the multiplication button it sends to the setQuestion which operator this function has to generate. It’s also returning a string which is the question
For the firstNumber, I set a random number to it by using the setData function. It is also the same as the second number. I also limit the random number that will be generated between 15 to -10.
Earlier I said that I will need increment, so I use it on the division. I want to avoid generating a question that which a number is divided by 0 or it’s dividing 0. So I use if-else, where every time the randomizer picks 0, it will increment by 1.
At the correctAnswer, I use the setData function to set the right answer that later on I can use to check the answer. I pass the correct answer to the getCorrectAnswer function. Later on, I will use this function to check the correct answer and combine it with the user input.

4.4 User Input

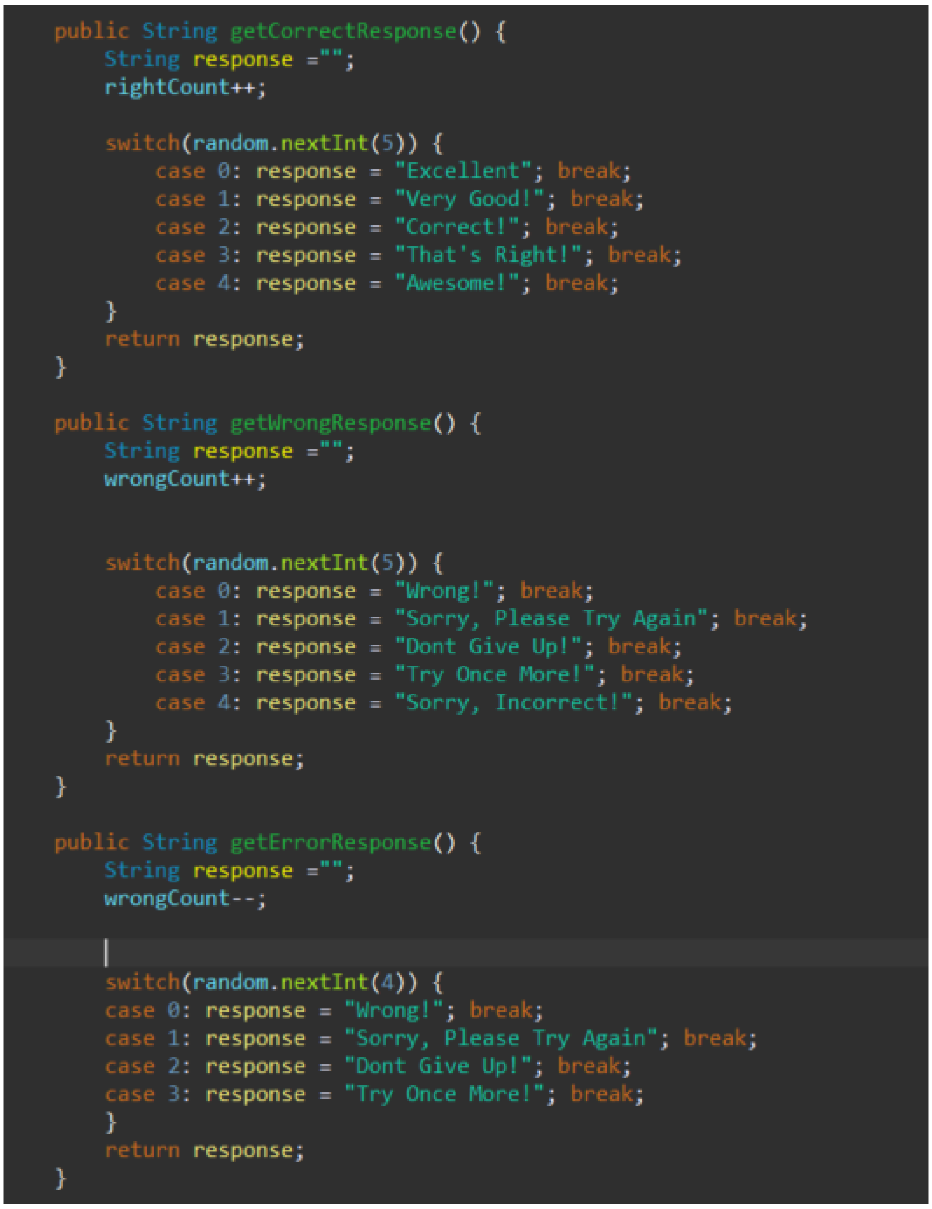


```
if(Ev.getSource() == answerField) {

    try {
        userAnswer = Integer.parseInt(answerField.getText());
    }
    catch (Exception Ex) {
        remarksField.setText(getErrorResponse());
    }
        if(userAnswer == controller.getCorrectAnswer()) {
            remarksField.setForeground(Color.blue);
            remarksField.setText(getCorrectResponse());
            answerField.setText("");
            rightCounterField.setText(rightCount + "");
            question.setText(questionText);
        }

        else {
            remarksField.setForeground(Color.red);
            remarksField.setText(getWrongResponse());
            answerField.setText("");
            wrongCounterField.setText(wrongCount + "");

            if(wrongCount >= Limit) {
                getContentPane().setBackground(Color.black);
                gameOver.setText("GAME OVER");
                gameOver.setVisible(true);
                restartButton.setVisible(true);
                exitButton.setVisible(true);
                rightCounterField.setVisible(false);
                wrongCounterField.setVisible(false);
                question.setVisible(false);
                subTitle2.setVisible(false);
                answer.setVisible(false);
                answerField.setVisible(false);
                answerField.setEditable(false);
                multiplicationButton.setVisible(false);
                additionButton.setVisible(false);
                substractionButton.setVisible(false);
                divisionButton.setVisible(false);
                remarksField.setVisible(false);
                answerField.setVisible(false);
                answerField.setEditable(false);
            }
        }
```

#Pane.java

In this part of the code, it will check the user input and the correct answer. I applied the parseInt to parse the string to an integer so that it will match the real answer that I set on the controller.java file. In this part, I use the try-catch, so that when the user inputs string data type it won't show an error code on the console, but it will appear on the remarks field, the same field where I put the response on the correct or wrong answer. I added if statement to limit the wrong answer to 10. After 10 wrong attempts, it will go to another screen where the user can choose between restarting and exiting the game.

4.5 Generating Responses



```java
public String getCorrectResponse() {
    String response ="";
    rightCount++;

    switch(random.nextInt(5)) {
        case 0: response = "Excellent"; break;
        case 1: response = "Very Good!"; break;
        case 2: response = "Correct!"; break;
        case 3: response = "That's Right!"; break;
        case 4: response = "Awesome!"; break;
    }
    return response;
}

public String getWrongResponse() {
    String response ="";
    wrongCount++;


    switch(random.nextInt(5)) {
        case 0: response = "Wrong!"; break;
        case 1: response = "Sorry, Please Try Again"; break;
        case 2: response = "Dont Give Up!"; break;
        case 3: response = "Try Once More!"; break;
        case 4: response = "Sorry, Incorrect!"; break;
    }
    return response;
}

public String getErrorResponse() {
    String response ="";
    wrongCount--;

    
    switch(random.nextInt(4)) {
    case 0: response = "Wrong!"; break;
    case 1: response = "Sorry, Please Try Again"; break;
    case 2: response = "Dont Give Up!"; break;
    case 3: response = "Try Once More!"; break;
    }
    return response;
}
```

These are the functions to generate responses that will be shown on the remarksField. I use the random function again to randomize which statement will be shown on the remarksField. getErrorResponse is also the same function as the getWrongResponse and getCorrectResponse. The difference is, that I called this function when the user enters unexpected input such as a string. It also has the decrement function so that when the user enters unexpected input, it won't increment the wrong count.

IV. References

https://www.javatpoint.com/java-jframe
https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm
https://stackoverflow.com/questions/5887709/getting-random-numbers-in-java