

B.Eng. (Software Engineering)/BSc (Computing)
SWE 4207-Fundamentals of Software Engineering
2022/2023(Semester 1)

CALCULATOR PROJECT REPORT

By – James Cedric Baisus

Group members:

Les Paul Ranalan

Mark Andrew

Anvin Tom

Tutor name: Ibtisam Mogul

Module guide- Computer Science Fundamentals

Contents

Project aim	3
Functional Requirements	4
Design the application.....	6
What is Use Case Analysis?	6
Implementation of Case Analysis in the Assignment	6
Implementation of Flow chart in the Assignment.....	7
Application development	13
Test Application.....	27
Conclusion and Further development.....	30
References.....	30

Project aim

The aim for this project is to use the skills learned in the Basic Computer science fundamentals of logical thinking to formulate programs that are mathematical related to basic arithmetic, equations, and numerical conversions.

Our team James Cedric Baisus, Paul Ranalan, Mark Andrew, Anvin Tom have come together to work towards a common goal which is to implement a modified calculator that would be satisfy all ages of students in the campus and in this project, we are able to share and combine all our ideas and opinionated queries to form our project to life. The team members will have to work together, using their diverse skills and knowledge to overcome challenges and obstacle, to collaborate, and find creative solutions

Our software implementation of choice is Java programming language this is because our goal is to make a fully functional calculator application therefore a GUI would be very beneficial for this since Java has a built-in extension for that.

Functional Requirements

Modes	Requirements
Basic Calculator	<ul style="list-style-type: none">- JFrame, for the window- JPanel, where all the other components will be- JButton, for the number and operation options- JTextField for displaying results- Addition Algorithm, for ADD operation- Subtraction Algorithm, for SUBB operation- Multiplication Algorithm, for MUL operation- Division Algorithm, for DIV operation
Odd or even	<ul style="list-style-type: none">- JFrame, for the window- JPanel, where all the other components will be- JButton, for the number and operation options- JTextField for displaying results- Odd or even Algorithm mainly using “If else”, for distinguishing which one is odd, and which is even
Factorial	<ul style="list-style-type: none">- JFrame, for the window- JPanel, where all the other components will be- JButton, for the number and operation options- JTextField for displaying results- Factorial Algorithm mainly using Long and String methods to print large value of numbers in the text field- Using for loop until the iteration is over

Average	<ul style="list-style-type: none"> - JFrame, for the window - JPanel, where all the other components will be - JButton, for the number and operation options - JTextField for displaying results - Average Algorithm utilizing addition and division operation
Binary, Octal, and Hex Conversion	<ul style="list-style-type: none"> - JFrame, for the window - JPanel, where all the other components will be - JButton, for the number and operation options - JTextField for displaying results
Area Finder	<ul style="list-style-type: none"> - JFrame, for the window - JPanel, where all the other components will be - JButton, for the number and operation options - JTextField for displaying results - Square area formula - Rectangle area formula - Triangle area formula - Circle area formula
Fibonacci	<ul style="list-style-type: none"> - JFrame, for the window - JPanel, where all the other components will be - JButton, for the number and operation options - JTextField for displaying results - Fibonacci Algorithm implementing a Fibonacci sequence to find the result
Min and Max	<ul style="list-style-type: none"> - JFrame, for the window - JPanel, where all the other components will be - JButton, for the number and operation options - JTextField for displaying results - An Algorithm mainly using "If else", for distinguishing which one is min, and which is max

Table 1 (Requirements)

Design the application

What is Use Case Analysis?

As explained previously, use case diagrams are employed to ascertain a system's usage requirements. You can use the information in a variety of ways depending on your needs. Here are a few uses for them.

- To determine roles and their interactions with tasks - The use case diagrams' main goal.
- When presenting to managers or other users, this is very helpful for getting a high-level view of the system. Without delving too deeply into the system's inner workings, you might emphasize the roles that interact with it and the functionality it offers.
- To distinguish between internal and exterior factors — This may seem basic, but in large, intricate projects, a system may be identified as playing an external function in another use case.

Implementation of Case Analysis in the Assignment

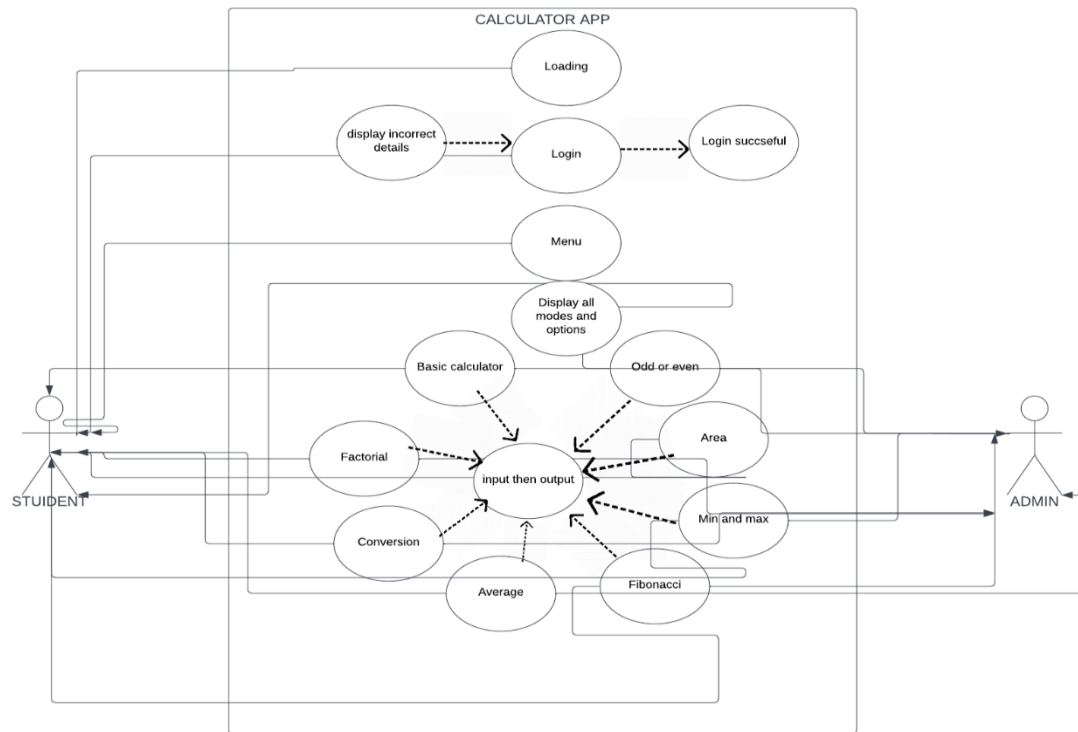


Figure 1 (Use Case diagram)

Students access the loading screen and login system if the user and password are valid then include case will bring the user to the menu if such conditions aren't meant then it extends to an error display case, Students can also access all of the components of the application including all the Loading screen, login, menu, and calculator modes while the Admin will continue to relay and extend result(output) if the user inputs a valid value.

Implementation of Flow chart in the Assignment

Flowchart for Basic Calculator

The user interface of a basic calculator typically enables the user to enter numbers and perform mathematical operations such addition, subtraction, multiplication, and division. Using the calculator's keyboard or buttons, the user would input the numbers and operations. The calculator would then carry out the required computations and provide the user the outcome.

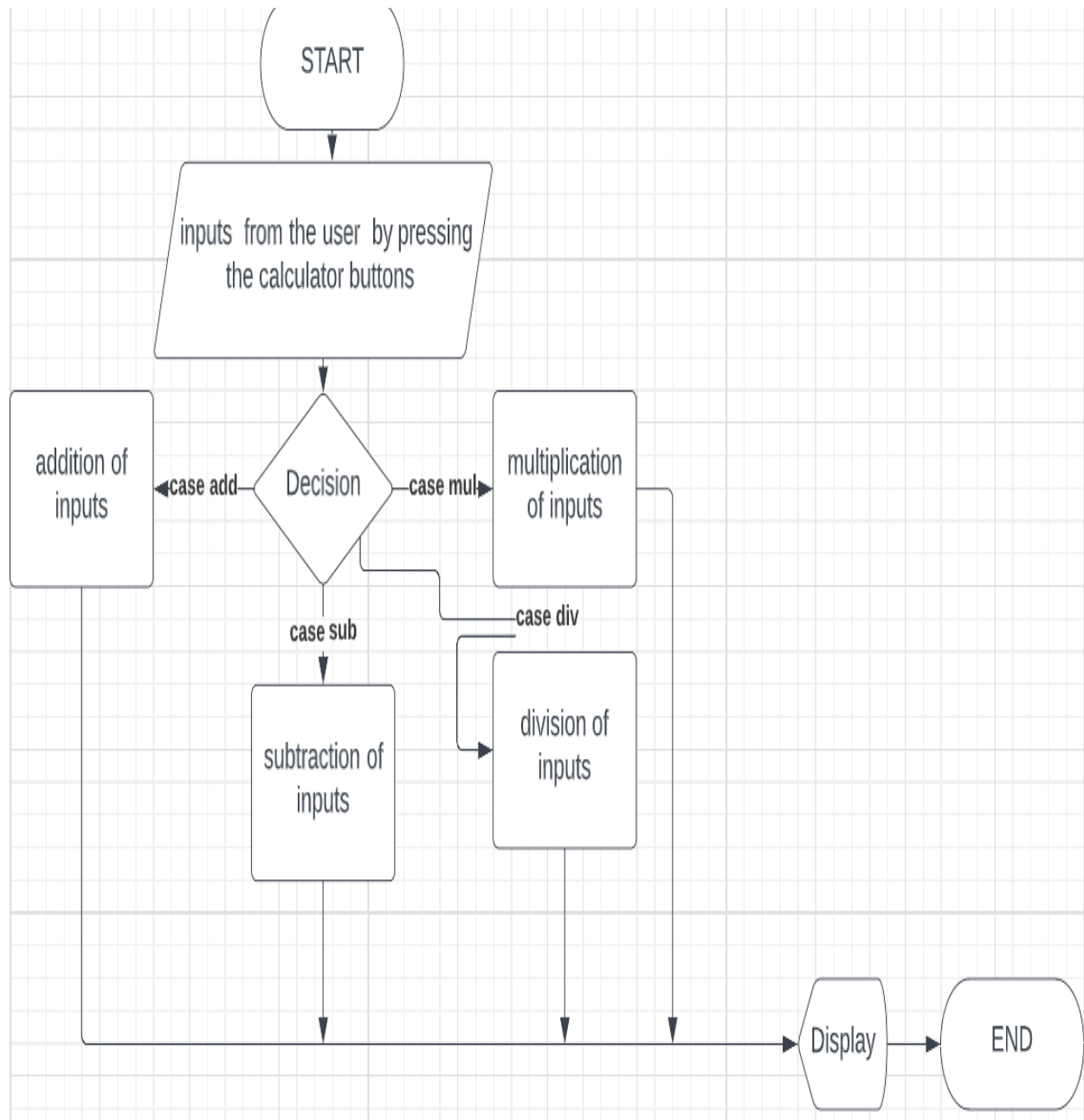


Figure 2 (Flowchart for basic calculator)

Flowchart for odd or even

A program would need to carry out a straightforward mathematical operation that verifies the integer's remainder after being divided by two to determine whether a number is odd or even. The number is even if the remainder is zero, and it is odd if the remaining is one.

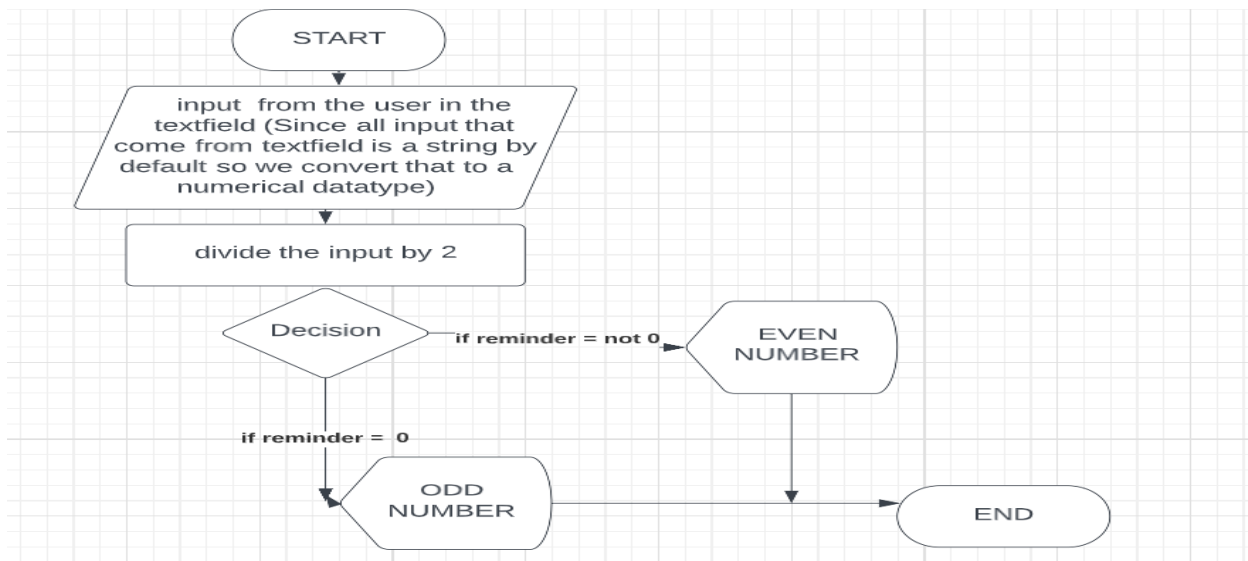


Figure 3 (Flow chart for odd or even)

Flow chart for min and max

The code will utilize “if else” conditions in this program to determine which is the biggest or the smallest number available in the input list that came from the user

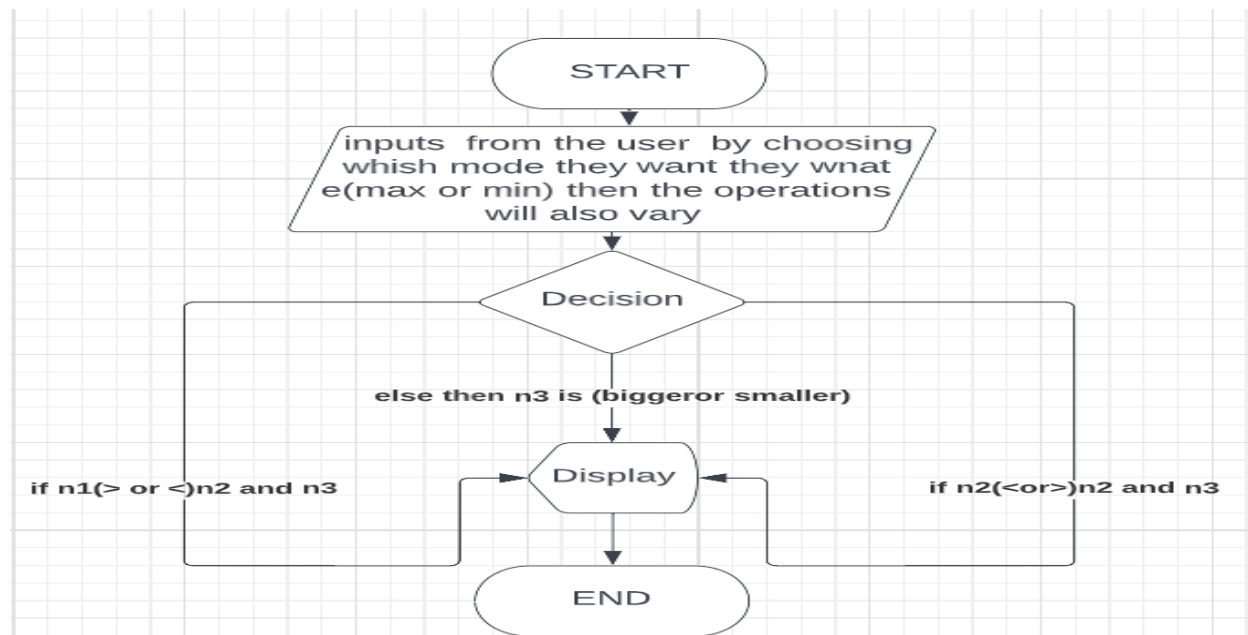


Figure 4 (Flowchart for min and max)

Flowchart for Factorial

A mathematical operation called a factorial involves multiplying a number by every other number between that amount and one. The factorial of 5, for instance, is $5 \times 4 \times 3 \times 2 \times 1 = 120$.

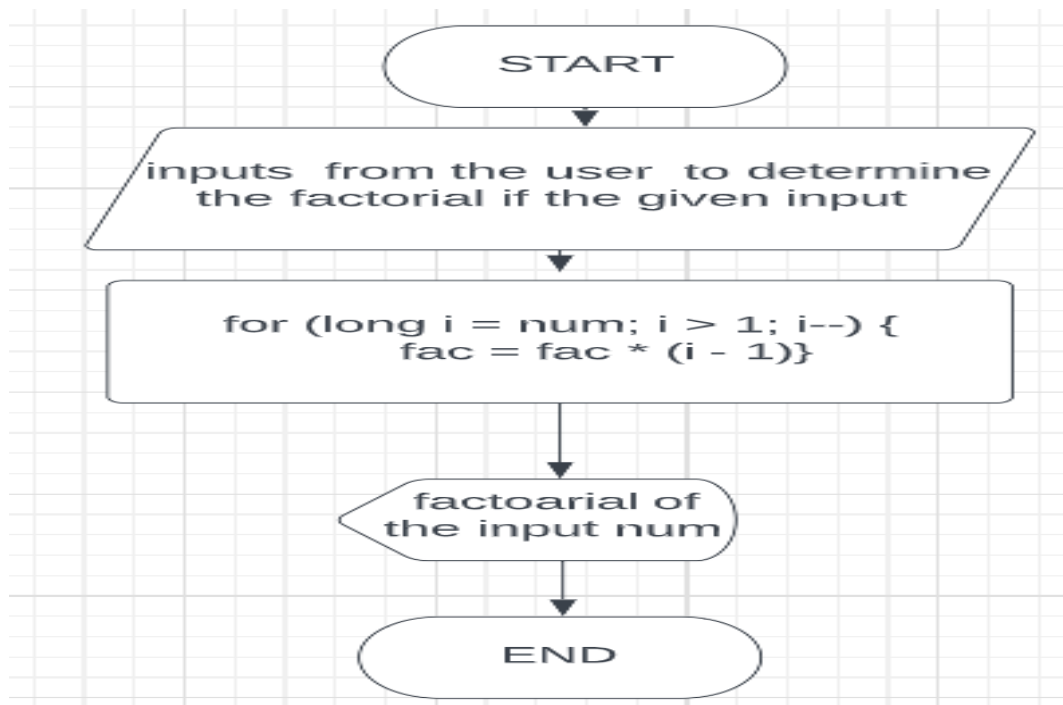


Figure 5 (Flowchart for factorial)

Flowchart for Area of shapes

Java program will allow the user a freedom to find the area, volume, and perimeter of a shape so for each corresponding shape there is a built-in formula for their area, volume, and perimeter.

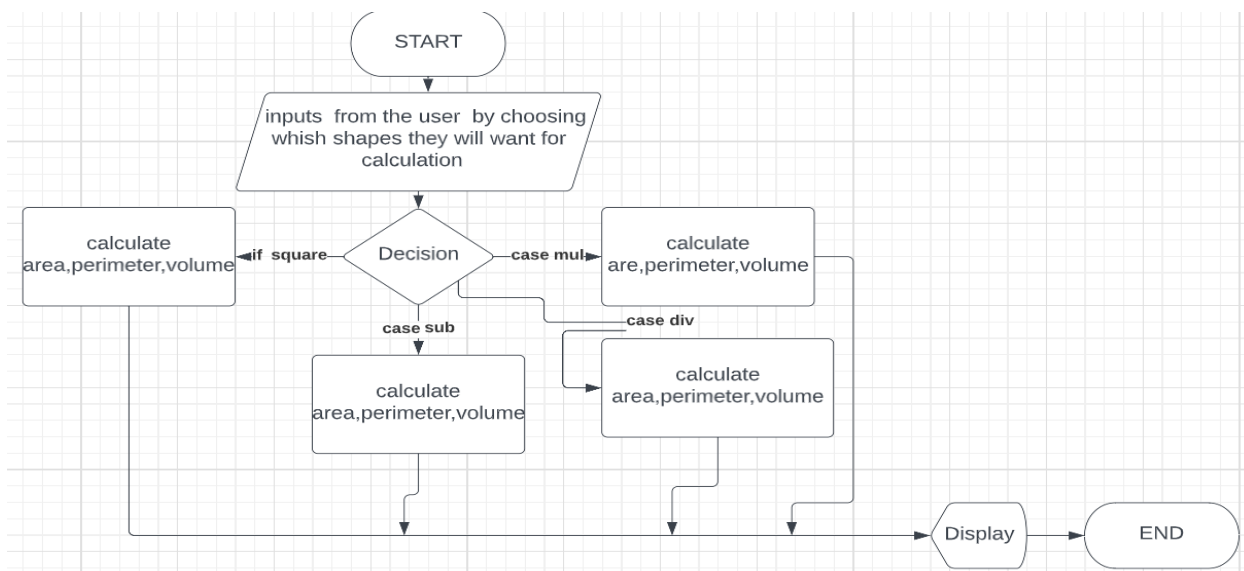


Figure 6 (Flowchart for Area finder)

Flowchart for Number system conversion

Java program will allow the user to choose which section of number system he wants to go then to which number system he/she will convert it to

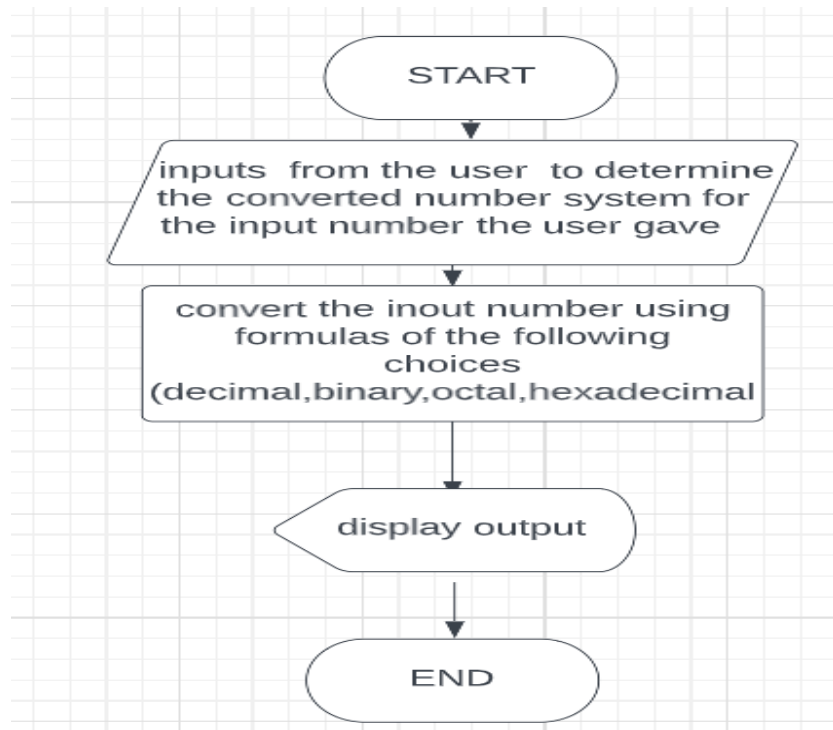


Figure 7 (Flowchart for conversion of number system)

Flowchart for Fibonacci sequence

Java program will allow the user to input a number then from a loop will be utilized to calculate the 2 Fibonacci value before that and add that to get the output

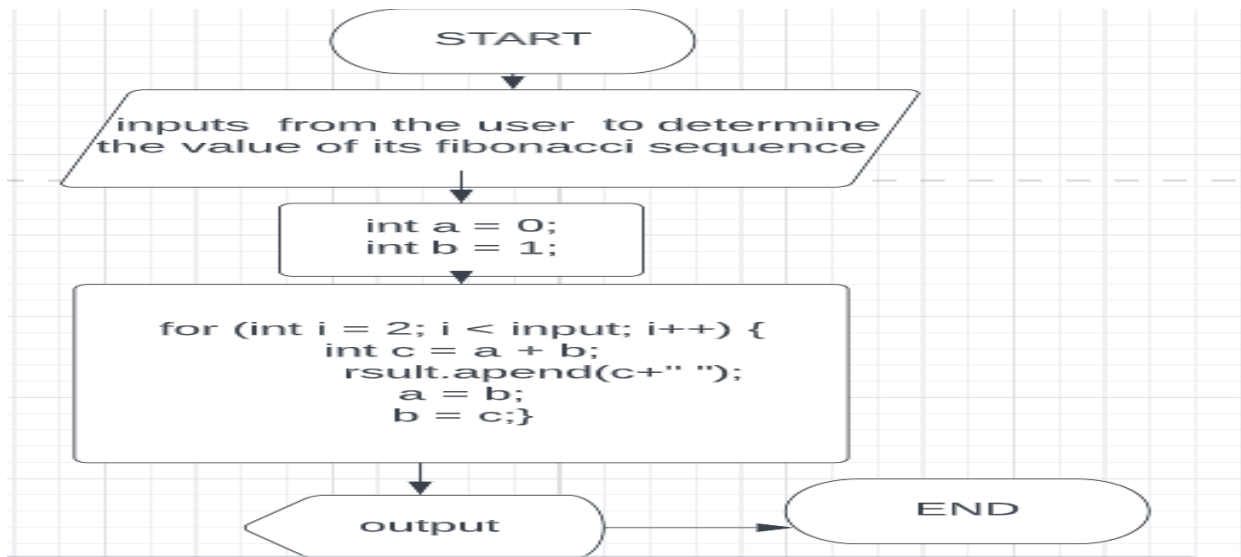


Figure 8 (Flowchart for Fibonacci sequence)

Flowchart for average

Java program will allow the user to input a list of number and from their all of the numbers in the list will be added altogether and be divided by their own quantity to the get average

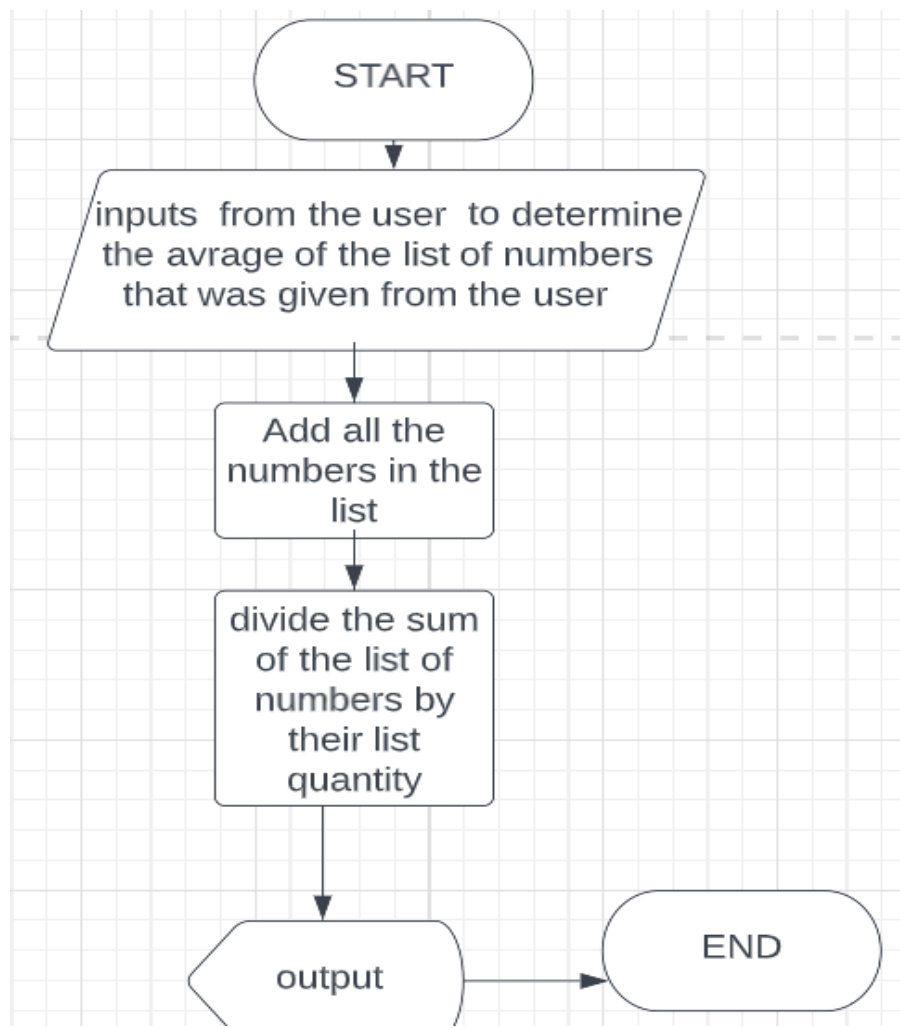


Figure 9 (Flowchart for average)

Application development

Firstly, when the student accesses the application, it will bring them a sprite screen where the JFrame are consisting of the following attributes of Java swing:

- Background
- Buttons to start the application and one for exit
- GIF
- Title

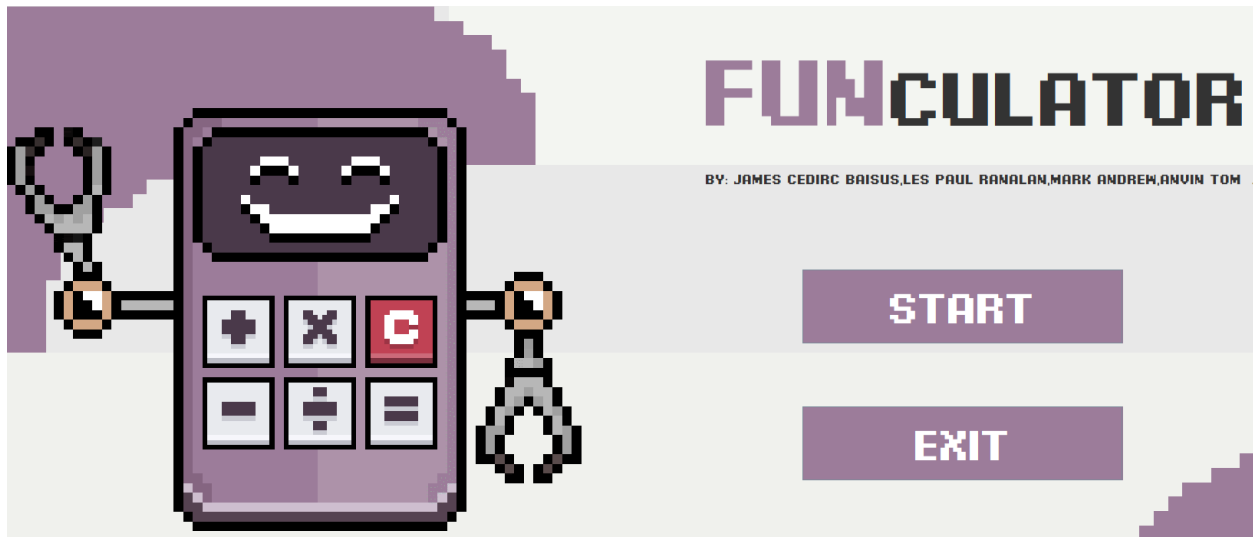


Figure 10 (Loading screen)

Code Implementation (Java)

```
private void initialize() {
    frame = new JFrame();
    frame.setBounds(100, 100, 1920, 1080);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null);
    frame.getContentPane().setLayout(null);

    JLabel lblNewLabel = new JLabel("New label");
    lblNewLabel.setIcon(new ImageIcon(Loadimg.class.getResource("/calculator/Cal(GIF).gif")));
    lblNewLabel.setBounds(63, 160, 603, 487);
    frame.getContentPane().add(lblNewLabel);

    JLabel lblNewLabel_2 = new JLabel("FUN");
    lblNewLabel_2.setForeground(new Color(156, 124, 154));
    lblNewLabel_2.setFont(new Font("Pixel Digivolve", Font.PLAIN, 99));
    lblNewLabel_2.setBounds(793, 119, 194, 86);
    frame.getContentPane().add(lblNewLabel_2);

    JButton btnNewButton = new JButton("START");
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Login.main(null);
            frame.dispose();
        }
    });
    btnNewButton.setBackground(new Color(156, 124, 154));
    btnNewButton.setForeground(Color.WHITE);
    btnNewButton.setFont(new Font("Pixel Digivolve", Font.PLAIN, 45));
    btnNewButton.setBounds(894, 351, 334, 77);
    frame.getContentPane().add(btnNewButton);

    JButton btnExit = new JButton("EXIT");
    btnExit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            frame = new JFrame("EXIT");
            if (JOptionPane.showConfirmDialog(frame, "Confirm if you want to exit", "Login", JOptionPane.YES_NO_OPTION) == JOptionPane.YES_NO_OPTION) {
                System.exit(0);
            }
        }
    });
    btnExit.setForeground(Color.WHITE);
    btnExit.setBackground(new Color(156, 124, 154));
    btnExit.setFont(new Font("Pixel Digivolve", Font.PLAIN, 45));
    btnExit.setBounds(894, 493, 334, 77);
    frame.getContentPane().add(btnExit);
}
```

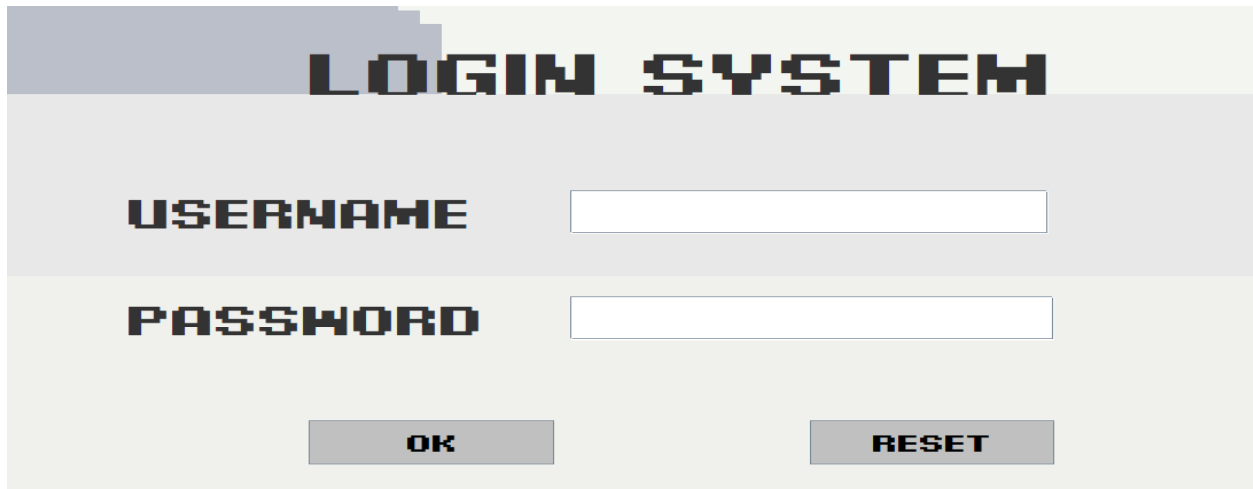
This just the main part of functionality in the program where the buttons are programmed to transfer user to the login system or to exit out the program (Note that this isn't the whole program as it is too big to be transferred here)

JFrame (Login system) are consisting of the following attributes of Java swing:

- Background

- Buttons to start the application and one for exit
- Text field and password field
- Title

If start button is then pressed these transfers the user to the login system, this allows user to close and save its progress when he/she exits the application



The image shows a graphical user interface for a login system. At the top, the title 'LOGIN SYSTEM' is displayed in a large, bold, black, monospace-style font. Below the title, there are two rows of labels and input fields. The first row has the label 'USERNAME' in a bold, black, monospace-style font, followed by a white rectangular text input field with a thin black border. The second row has the label 'PASSWORD' in the same font style, followed by a white rectangular password input field with a thin black border. At the bottom of the form, there are two buttons. The button on the left is labeled 'OK' in a bold, black, monospace-style font. The button on the right is labeled 'RESET' in the same font style. Both buttons have a light gray background and a thin black border.

Figure 11 (Login system)

Code Implementation (Java)

```

textField = new JTextField();
textField.setBounds(705, 345, 334, 47);
frame.getContentPane().add(textField);
textField.setColumns(10);

passwordField = new JPasswordField();
passwordField.setBounds(705, 459, 338, 47);
frame.getContentPane().add(passwordField);

JButton btnNewButton = new JButton("OK");
btnNewButton.setBackground(Color.LIGHT_GRAY);
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        String username = textField.getText();
        @SuppressWarnings("deprecation")
        String password = passwordField.getText();

        if (username.equals("admin") && password.equals("123")) {
            textField.setText(null);
            passwordField.setText(null);
            Menu.main(null);
            frame.dispose();
        }
        else {
            JOptionPane.showMessageDialog(null, "Invalid username and password");
        }
    }
});

```

As u can see an if else statement is used to validate if the authentication details are right before going to the menu page if the authentication catches an exception, then else code will trigger in here, we used an Joption where a mini pop up window will appear and will tell the user that his/her account is invalid.

The following JFrame (Menu) are consisting of the following attributes of Java swing:

- Background
- Buttons to go to every mode
- Title

After getting the correct authentication details the application will bring the user to the menu section where all the calculator mods are presented



Figure 12 (Menu)

Code implementation (Java)


```

private void initialize() {
    frame = new JFrame();
    frame.setBounds(100, 100, 1920, 1080);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);
    frame.setLocationRelativeTo(null);

    JLabel lblNewLabel = new JLabel("MENU");
    lblNewLabel.setBackground(Color.LIGHT_GRAY);
    lblNewLabel.setFont(new Font("Pixel Digivolve", Font.PLAIN, 90));
    lblNewLabel.setBounds(650, 0, 272, 110);
    frame.getContentPane().add(lblNewLabel);

    JButton btnNewButton = new JButton("BASIC CALCULATOR");
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            BasicCal.main(null);
            frame.dispose();
        }
    });
    btnNewButton.setFont(new Font("Pixel Digivolve", Font.PLAIN, 40));
    btnNewButton.setBackground(new Color(168, 208, 243));
    btnNewButton.setBounds(550, 120, 461, 61);
    frame.getContentPane().add(btnNewButton);

    JButton btnOddOrEven = new JButton("ODD OR EVEN");
    btnOddOrEven.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            OddorEven.main(null);
            frame.dispose();
        }
    });
    btnOddOrEven.setFont(new Font("Pixel Digivolve", Font.PLAIN, 40));
    btnOddOrEven.setBackground(new Color(164, 236, 212));
    btnOddOrEven.setBounds(550, 191, 461, 61);
    frame.getContentPane().add(btnOddOrEven);

    JButton btnNewButton_1_1 = new JButton("AVERAGE");
    btnNewButton_1_1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Avg.main(null);
            frame.dispose();
        }
    });
}

```

As u can see that under every Button they all have a designated application window to go to because it will take the user to multiple calculator modes that are available in the application this window.it also has a designated log out button if the user tries to the log in system to sign in a diff account.

The following JFrame (Basic calculator) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Basic calculator window should look like this



Figure 13 (Basic Calculator)

Code implementation (Java)

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == b1)
        t.setText(t.getText().concat("1"));
    if (e.getSource() == b2)
        t.setText(t.getText().concat("2"));
    if (e.getSource() == b3)
        t.setText(t.getText().concat("3"));
    if (e.getSource() == b4)
        t.setText(t.getText().concat("4"));
    if (e.getSource() == b5)
        t.setText(t.getText().concat("5"));
    if (e.getSource() == b6)
        t.setText(t.getText().concat("6"));
    if (e.getSource() == b7)
        t.setText(t.getText().concat("7"));
    if (e.getSource() == b8)
        t.setText(t.getText().concat("8"));
    if (e.getSource() == b9)
        t.setText(t.getText().concat("9"));
    if (e.getSource() == b0)
        t.setText(t.getText().concat("0"));
    if (e.getSource() == bdec)
        t.setText(t.getText().concat("."));
    if (e.getSource() == badd) {
        a = Double.parseDouble(t.getText());
        operator = 1;
        t.setText("");
    }
    if (e.getSource() == bsub) {
        a = Double.parseDouble(t.getText());
        operator = 2;
        t.setText("");
    }
    if (e.getSource() == bmul) {
        a = Double.parseDouble(t.getText());
        operator = 3;
        t.setText("");
    }
    if (e.getSource() == bdiv) {
        a = Double.parseDouble(t.getText());
        operator = 4;
        t.setText("");
    }
}
```

Every time a button is pressed by the user it displays this in the text field, but this datatype is in a String format and it should be an integer for calculations that's why as you can tell there is another set of if else below the group of the if the buttons are pressed because that if else will be responsible for converting the String into numerical datatype.

The following JFrame (Odd or Even) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output

- Title
- Text field

If the user decides to go to the Basic calculator window should look like this

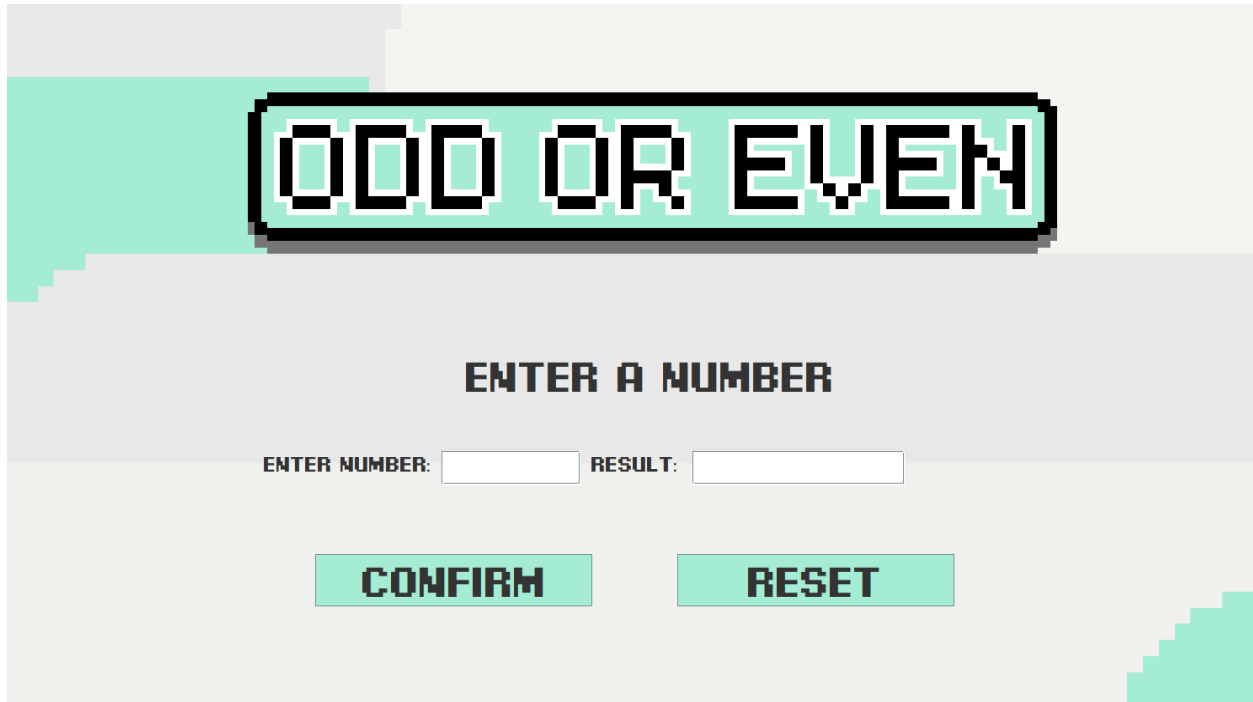


Figure 14 (Odd or even)

Code implementation (Java)

```
String num1 = "EVEN NUMBER";
String num2 = "ODD NUMBER";
int num = Integer.parseInt(txtnumber.getText());
if (num%2==0) {
    textField.setText(num1);
}
if (num%2!=0) {
    textField.setText(num2);
}
```

The 2 String data type are declared to be odd or even as this will the output the text number is the text field where we get the input from the user, we then convert this to a numerical data type to perform mathematical equations to find if the input is Odd or Even.

The following JFrame (Factorial) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output

- Title
- Text field

If the user decides to go to the Factorial window should look like this



Figure 15 (Factorial)

Code implementation (Java)

```

JButton btnNewButton_factorial_1 = new JButton("CONFIRM");
btnNewButton_factorial_1.setBackground(new Color(255, 182, 191));
btnNewButton_factorial_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        String number = txtnumber.getText();
        long num = Long.parseLong(number);
        long fac = num;
        for (long i = num; i > 1; i--) {
            fac = fac * (i - 1);
        }
        textField.setText(Long.toString(fac));
    }
});
btnNewButton_factorial_1.setFont(new Font("Pixel Digivolve", Font.PLAIN, 40));
btnNewButton_factorial_1.setBounds(458, 528, 263, 49);
frame.getContentPane().add(btnNewButton_factorial_1);

```

The content is initially pulled from the text field text number by the code, which then saves it as a String named number. The Long.parseLong() method is then used to turn this String into a long object called num. The value of fac is then changed by the code to num.

The code then begins a for loop, iterating through the range from $i = \text{num}$ to $i = 1$. The loop calculates $\text{fac} = \text{fac} * (i - 1)$ for each iteration, effectively multiplying fac by all the values from num down to 1 to determine the factorial of num.

Finally, using the Long.toString() function, the code updates the text of a text field called textField to the value of fac transformed to a String. This successfully informs the user of the factorial calculation's outcome.

The following JFrame (Fibonacci) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Factorial window should look like this

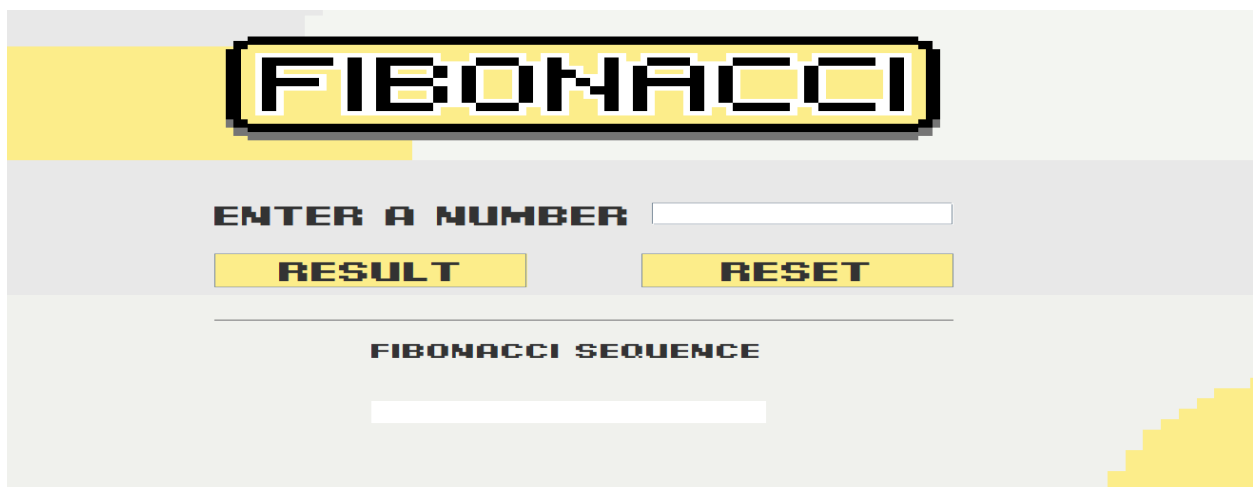


Figure 15 (Fibonacci)

Code implementation (Java)

```
public void actionPerformed(ActionEvent e) {  
    try {  
        int input = Integer.parseInt(txtnumber.getText());  
        int a = 0;  
        int b = 1;  
        result.setText(a+" "+b+" ");  
  
        for (int i = 2; i < input; i++) {  
            int c = a + b;  
            result.append(c+" ");  
            a = b;  
            b = c;  
        }  
    }  
    catch (Exception ex) {  
        JOptionPane.showMessageDialog(null, "invalid input.");  
    }  
};
```

The text is initially parsed into an int named input by the code before being read from a text field called txtnumber. Two int variables, a and b, are then initialized to 0 and 1, respectively. These two numbers are the first two in the Fibonacci series. The text of a text field named result is then set by the code to the values of a and b, separated by a space.

The program then starts a for loop, iterating i from 2 to input $- 1$. By combining a and b and preserving the result as c , the loop calculates the following number in the Fibonacci sequence for each iteration. After that, it appends the value of c and a space to the text in the result text box. The user is effectively shown the Fibonacci sequence up to the entered integer.

In order to calculate the following number in the sequence during the subsequent iteration of the loop, the code updates the values of a and b by setting a to the value of b and b to the value of c .

The following JFrame (Average) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Factorial window should look like this

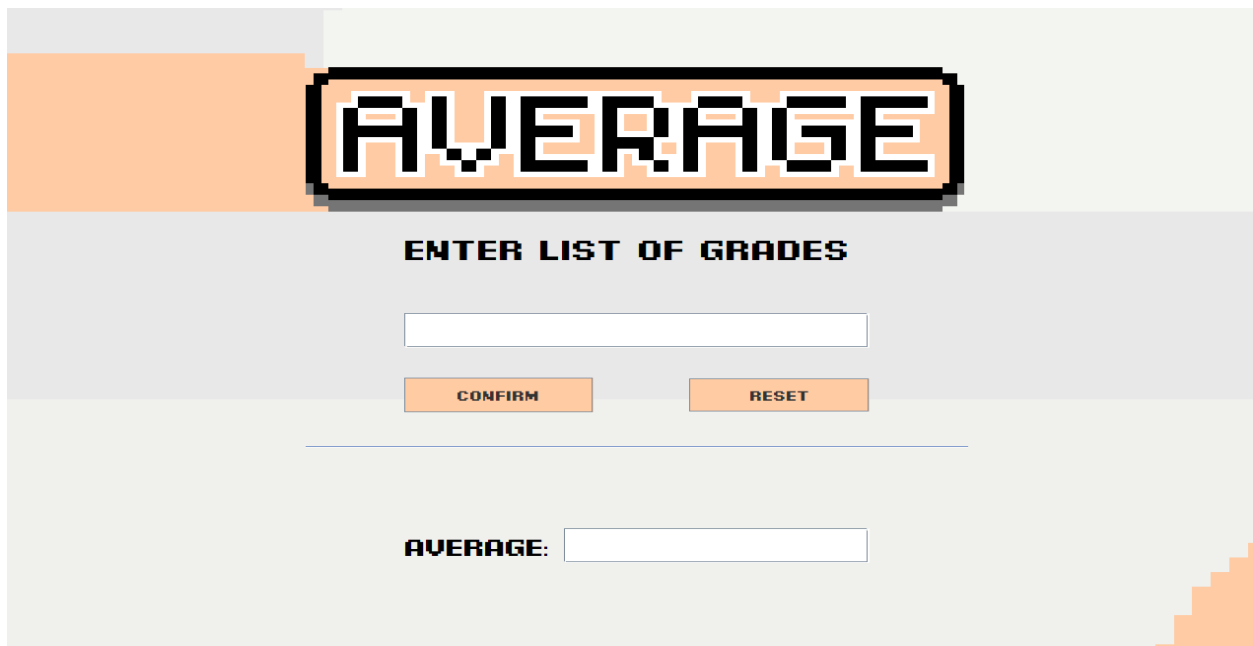


Figure 16 (Average)

Code implementation (Java)

```

    public void actionPerformed(ActionEvent e) {
        try {
            String numlist = avg.getText();

            String[] numlistArray = numlist.split(" ");

            double sum = 0;
            for (String numString : numlistArray) {
                double num = Double.parseDouble(numString);
                sum = sum + num;
            }

            double avg = sum / numlistArray.length;
            avgres.setText(""+avg);
        }
        catch (Exception ex) {
            JOptionPane.showMessageDialog(null,"invalid input.");
        }
    }
});

```

The code retrieves the text from the avg text field and saves it as numlist, a String. It is presumed that this string contains a list of numbers, each one separated by a space.

The split() method is then used to divide the numlist String into an array of Strings called numlistArray. By splitting the String at each space, this function creates elements for the numlistArray array.

The next step is to set the initial value of the double variable sum to 0. The for loop that follows iterates through each numString entry in the numlistArray array. The numString is converted to a double called num and added to the value of sum in each iteration of the loop using the Double.parseDouble() method. With this, the total of all the numbers in the numlistArray array is effectively calculated.

By dividing the sum value by the length of the numlistArray array and saving the result as a double called avg, the code then calculates the average of the integers.

By dividing the sum value by the length of the numlistArray array and saving the result as a double called avg, the code then calculates the average of the integers.

The following JFrame (Area) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Factorial window should look like this

AREA

SQUARE

SIDE:

HEL...

AREA:

PERI...

VOLUME:

RECTANGLE

LENGTH:

WIDTH:

HEIGH...

AREA:

PERI...

VOLUME:

CIRCLE

RADIUS:

HEIG...

AREA:

PERI...

VOL...

TRIANGLE

HEIGHT:

LENG...

BASE:

AREA:

PERI...

VOL...

Figure 17 (Area calculator)

Code implementation (Java)

In this code it asks for the user for an input depending on what shape they want to determine the area, perimeter, volume when user inputs those inputs will then be transferred to the respective formulas to get the area, perimeter, volume.

The following JFrame (Conversion) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Factorial window should look like this

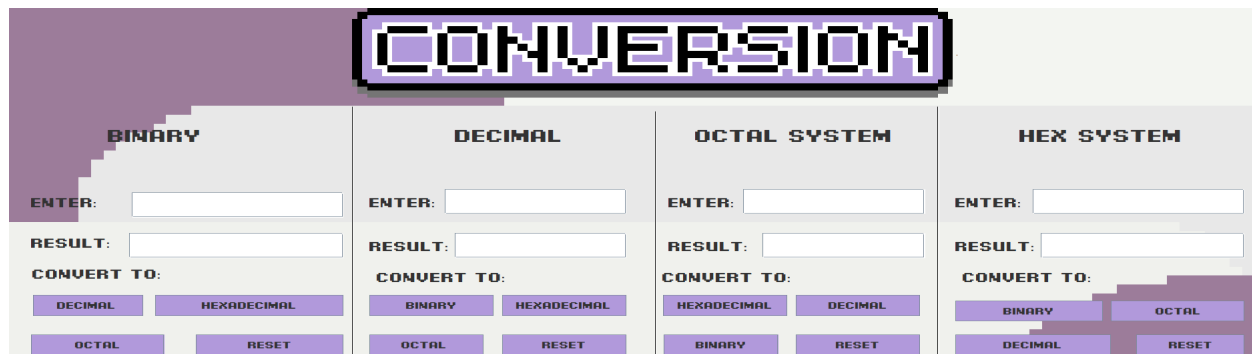


Figure 18 (Number system conversion)

Code implementation (Java)

```
public void actionPerformed(ActionEvent e) {
    try {
        String s = num2.getText();
        int i = Integer.parseInt(s);
        String s1 = Integer.toBinaryString(i);
        textField.setText(s1);
    } catch (Exception e1) {
        JOptionPane.showMessageDialog(null, "input is not a decimal number.");
    }
}
});
```

All of the number system have the same format of formula but the only key diff is the string method conversion the following methods were used for number conversion

- toBinary
- toOctal
- toHex

The following JFrame (Max and Min) are consisting of the following attributes of Java swing:

- Background
- Buttons for input and output
- Title
- Text field

If the user decides to go to the Factorial window should look like this

Figure 18 (MIN AND MAX)

Code implementation (Java)

```
String number = n1min.getText();
String number2 = n2min.getText();
String number3 = n3min.getText();
long num1 = Long.parseLong(number);
long num2 = Long.parseLong(number2);
long num3 = Long.parseLong(number3);

if(num1<num2&&num1<num3) {
    minresult.setText("the Minimum number is "+num1);
}
else if(num2<num1&&num2<num3) {
    minresult.setText("the Minimum number is "+num2);
}
else {
    minresult.setText("the Minimum number is "+num3);
}

});
```

To save the text as Strings with the names `number`, `number2`, and `number3`, respectively, the code first retrieves the text from three text fields with the names `n1min`, `n2min`, and `n3min`. It then employs the `Long.parseLong()` function to transform these Strings into longs with the names `num1`, `num2`, and `num3`.

The code then compares the values of `num1`, `num2`, and `num3` to see which one the minimum is using an if-else expression. The code changes the text of a text field called `minresult` to a String that says "the

Minimum number is [value of num1]" if num1 is smaller than both num2 and num3. The code changes the text of minresult to a String that says "the Minimum number is [value of num2]" if num2 is smaller than both num1 and num3. If not, the code inserts a String into the text of the minresult variable that reads, "The Minimum number is [value of num3]." This effectively shows the user the lowest of the three figures.

Test Application

For the test plan we targeted the programs that are very exception sensitive wherein invalid amounts that could come from the user will terminate and crash the whole application. This includes the 2 modes that had a lot of issue

- Odd or even
- Factorial

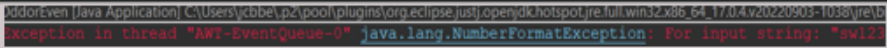

module no	TC-OddOrEven-1
test name	invalid input
test scenario / steps	<p>user inputs non-integer value.</p> <ol style="list-style-type: none"> 1. open program 2. enter value 3. click result button
test input	sw123
expected result	program will not terminate
actual result	program exited to errors, "for input string: "sw123"
pass / fail	fail
screenshot	 <pre> OddOrEven [Java Application] C:\Users\ycbbe\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\java.exe Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "sw123" </pre>
solution	<p>instantiate the result button action variable to an integer data type, that way even if a different data type is entered, it will not terminate the program.</p>
solution output	 <p>The screenshot shows a Java Swing window titled "ENTER A NUMBER". It contains a text input field with the value "sw123". Below the input field are two buttons: "RESULT" and "RESET". The "RESULT" button is highlighted in green.</p>

Figure 18 (Test case 1)

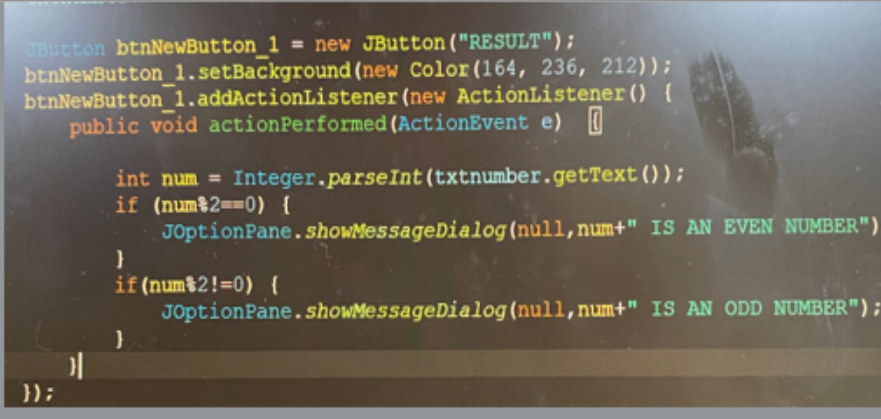
test id	TC02
module no	TC-OddOrEven-2
test name	invalid input
test scenario / steps	user inputs non-integer value 1. open program 2. enter value 3. click result button
test input	sw123
expected result	program will not terminate
actual result	program did not terminate
pass / fail	pass
screenshot	 <pre> JButton btnNewButton_1 = new JButton("RESULT"); btnNewButton_1.setBackground(new Color(164, 236, 212)); btnNewButton_1.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) { int num = Integer.parseInt(txtnumber.getText()); if (num%2==0) { JOptionPane.showMessageDialog(null,num+" IS AN EVEN NUMBER") } if(num%2!=0) { JOptionPane.showMessageDialog(null,num+" IS AN ODD NUMBER"); } } }); </pre>

Figure 19 (Test case 2)

Analysis of test result

Main problem:

- We realized that in every formula we must catch the numerical exception since all of the modes are interacted by the user, we were able to apply exception handling for the most part and it did fix the problem

Problem 1 and 2:

- These problems occurred when our project was at its first state of development wherein we will get conversion errors whenever we type an invalid datatype, further research and found out that text fields (input) are by default String this proved to be the main root cause of the problem

And was eventually solved

Conclusion and Further development

In conclusion project was partially difficult in terms of the technicality of the project and since it is also a team-based project contribution of ideas played a huge role in completing the assigned task .in the end We really made sure to use GUI because this helps the aspects of being a user-friendly application towards students of all ages.

Technically speaking, the project was somewhat challenging, particularly when a GUI was involved. But that's the appeal of programming—finding new, creative ways to address various issues or challenges. Since the project was a team effort, each team member developed an algorithm specifically for the operation they were given to work on using their own methods and peculiarities.

Overall, the task was successful in terms of design aesthetic and Java code. Yes, there may be a few minor glitches here and there. The GUI is a little strange for macOS computers. However, it functions flawlessly on Windows systems with 1080p panels. But despite all that, I believe the application is already adequate for version 1 and only requires a small amount of tweaking or code cleaning to make it operate better on platforms and generally.

Implementation of database would also be useful for the future as this will also the user to save and recognize different authorization details.

References

Nishadha (2022) *Use case diagram tutorial (guide with examples)*, Creately Blog. Available at: <https://creately.com/blog/diagrams/use-case-diagram-tutorial/?fbclid=IwAR3BNPnUGTbvISxdzq5lPcwJ1TGILzpt-5OttDcfLAJqEWNgYhL1rUTm6RU#:~:text=Use%20case%20diagram%20is%20a,roles%20interact%20with%20the%20system> (Accessed: December 13, 2022).

What is a flowchart, and why is it important? (No date) *Workflow Management Software* by Integrify. Available at: <https://www.integrify.com/what-is-a-flowchart/#:~:text=Flowcharts%20are%20visual%20diagrams%20that,rely%20on%20for%20various%20purposes>. (Accessed: December 13, 2022).

