Yo Treenawat Manomairat

Document analysis of Project 5

Professor Rafal Jabrzemski

CS 2334 Programming Structures and Abstractions

2 May 2019

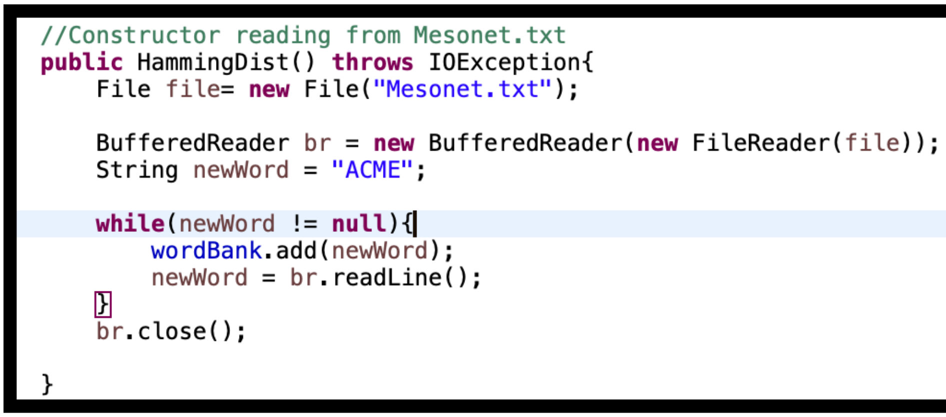
Analysis of Project 5

Similar to first three project, this project focused on Hamming Distance by reading from Mesonet file. I think this project is separate into 2 part, the first part is about calculating while the second part is GUI. The different thing about this project compare to first 3 is that this project not only do we involved GUI into the code, but we also have a special array where new station can be added.

Calculating part (HammingDistant.java)

Looking into calculating part, The thing that need to be done are Reading file from meson for comparison. The calculating how much the distance is from this station to another. This including Distance from 1-4 and array that contain all the different station from 1-4 as well. Last is the arrayList that will add more station if the user input is four characters.

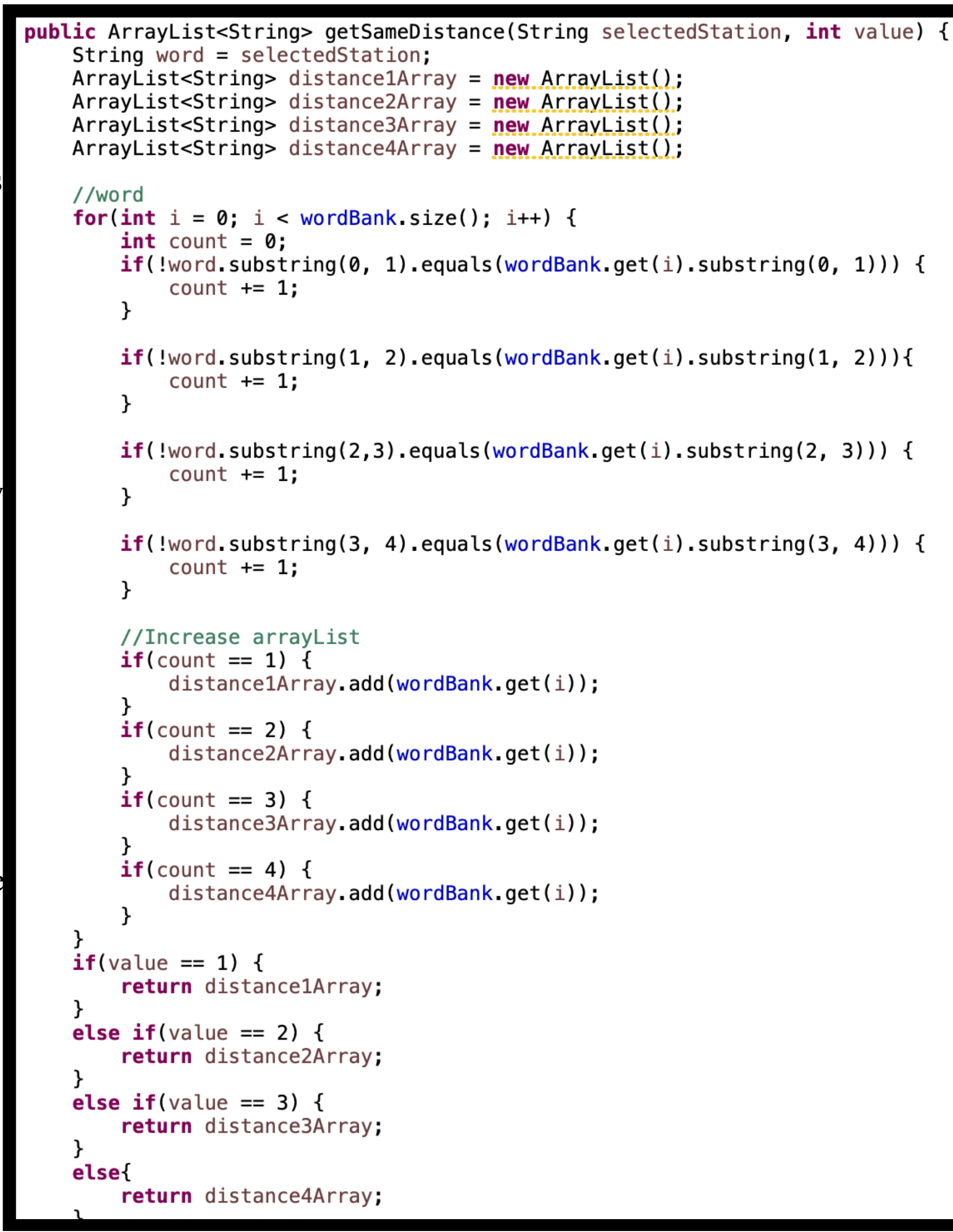
Use buffer reading to read from Mesonet.txt. For this, I create buffer reading that read from Menonet.txt and add them to wordBank to compare them with the station use choose.



```
//Constructor reading from Mesonet.txt
public HammingDist() throws IOException{
    File file= new File("Mesonet.txt");

    BufferedReader br = new BufferedReader(new FileReader(file));
    String newWord = "ACME";

    while(newWord != null){
        wordBank.add(newWord);
        newWord = br.readLine();
    }
    br.close();

}
```

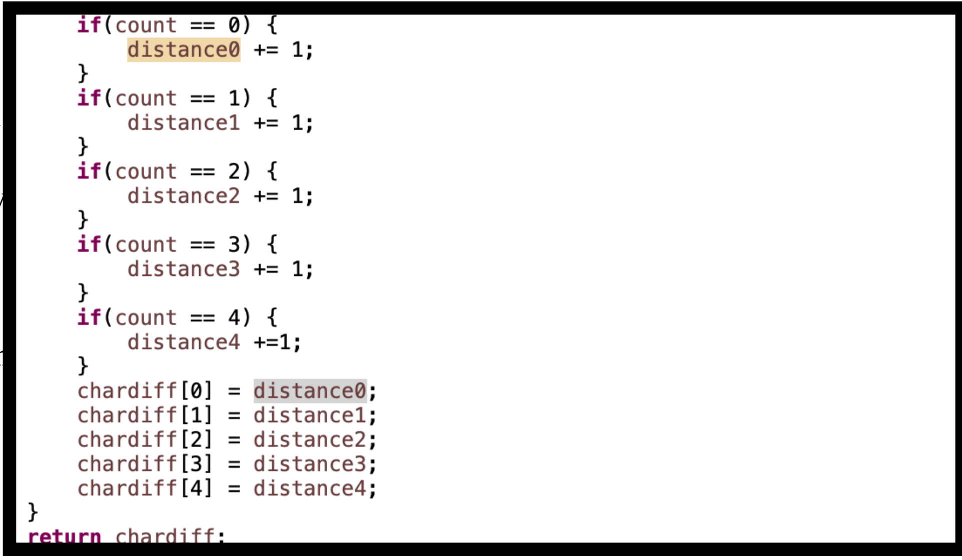
Calculating the difference in Distant.

For this, I use the code similar to the first project, but what I do different is creating an array, that each would contain the stations that have same distance. For this, I go through each character and compare. If there is any difference then, count would **+1,** in the end, the amount of count would determine how which array the station would be added to. After that return the array that is the same to the value that are being used on the header.



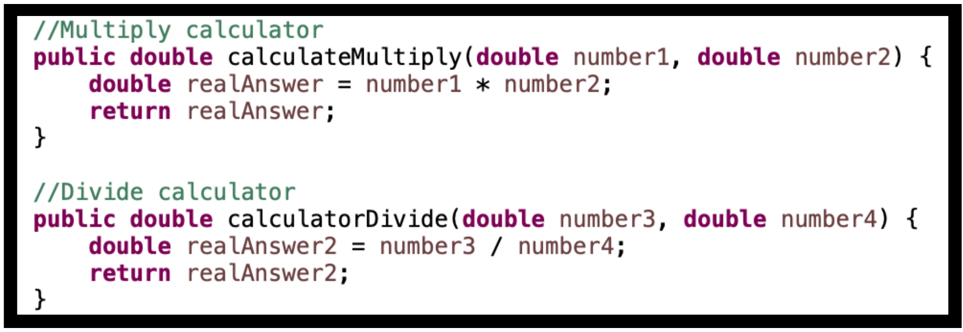
```
public ArrayList<String> getSameDistance(String selectedStation, int value) {
    String word = selectedStation;
    ArrayList<String> distance1Array = new ArrayList();
    ArrayList<String> distance2Array = new ArrayList();
    ArrayList<String> distance3Array = new ArrayList();
    ArrayList<String> distance4Array = new ArrayList();

    //word
    for(int i = 0; i < wordBank.size(); i++) {
        int count = 0;
        if(!word.substring(0, 1).equals(wordBank.get(i).substring(0, 1))) {
            count += 1;
        }

        if(!word.substring(1, 2).equals(wordBank.get(i).substring(1, 2))){
            count += 1;
        }

        if(!word.substring(2,3).equals(wordBank.get(i).substring(2, 3))) {
            count += 1;
        }

        if(!word.substring(3, 4).equals(wordBank.get(i).substring(3, 4))) {
            count += 1;
        }

        //Increase arrayList
        if(count == 1) {
            distance1Array.add(wordBank.get(i));
        }
        if(count == 2) {
            distance2Array.add(wordBank.get(i));
        }
        if(count == 3) {
            distance3Array.add(wordBank.get(i));
        }
        if(count == 4) {
            distance4Array.add(wordBank.get(i));
        }
    }
    if(value == 1) {
        return distance1Array;
    }
    else if(value == 2) {
        return distance2Array;
    }
    else if(value == 3) {
        return distance3Array;
    }
    else{
        return distance4Array;
    }
```

Another calculation is to find the difference in each character as you can see in the second picture that I separate each character into an array and each array tell how many character difference there are between this word and other in meson.txt



```
    if(count == 0) {
        distance0 += 1;
    }
    if(count == 1) {
        distance1 += 1;
    }
    if(count == 2) {
        distance2 += 1;
    }
    if(count == 3) {
        distance3 += 1;
    }
    if(count == 4) {
        distance4 +=1;
    }
    chardiff[0] = distance0;
    chardiff[1] = distance1;
    chardiff[2] = distance2;
    chardiff[3] = distance3;
    chardiff[4] = distance4;
}
return chardiff;
```

Last one is an extra for right side of the GUI, as I choose to create two calculator, first for Multiply and second for divide.



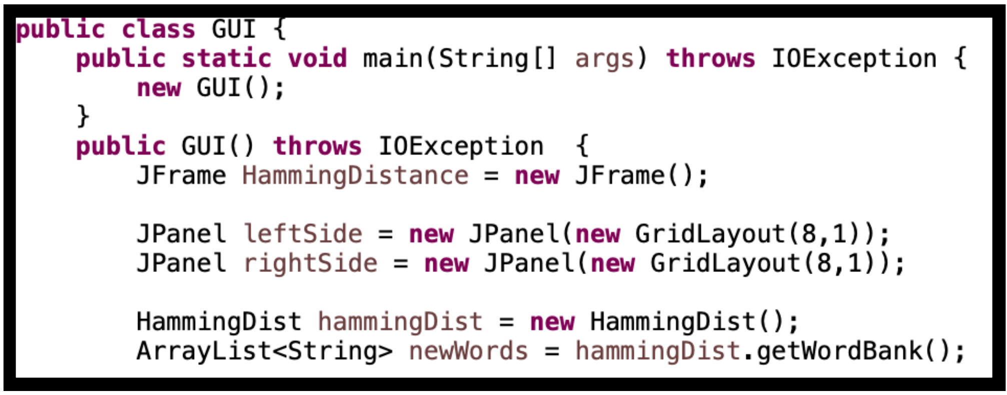
```java
//Multiply calculator
public double calculateMultiply(double number1, double number2) {
    double realAnswer = number1 * number2;
    return realAnswer;
}

//Divide calculator
public double calculatorDivide(double number3, double number4) {
    double realAnswer2 = number3 / number4;
    return realAnswer2;
}
```

The GUI Panel(GUI.java)

For this class, I create a GUI, in which is the copy of what told to be done. The left side and the right side. The left is what we suppose to do, while the right is the creative part, in which as seen above, I choose to create multiply and divide calculator.

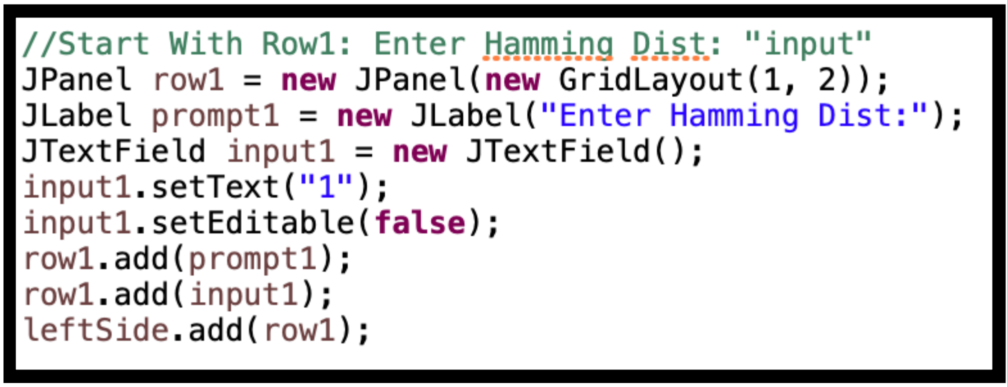
First thing, I create class with main and GUI constructor

The first thing I done is create the JFrame, and the separate it into left and right side. then create an array newWords that contain the second version of Masonet.txt, so that when comparing we won't compare the new word we include in ourself.



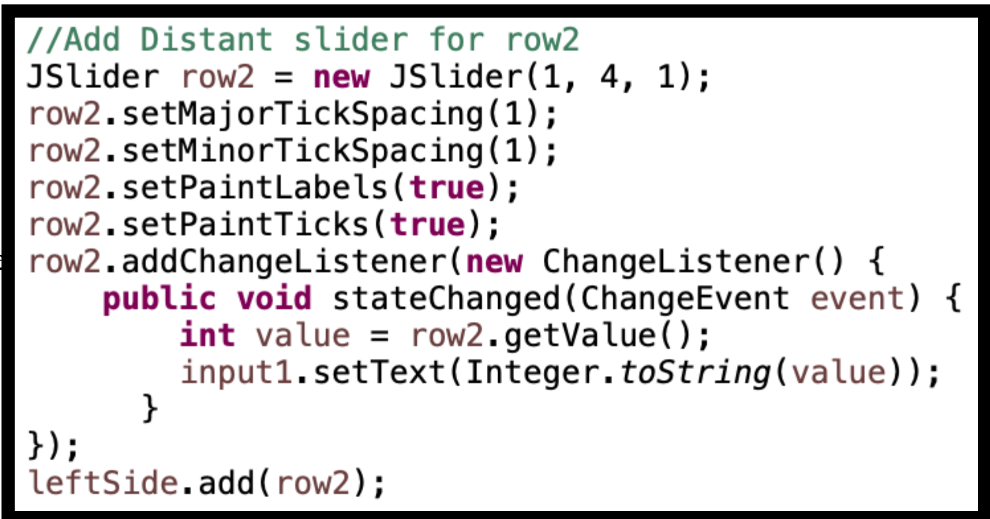
```java
public class GUI {
    public static void main(String[] args) throws IOException {
        new GUI();
    }
    public GUI() throws IOException  {
        JFrame HammingDistance = new JFrame();

        JPanel leftSide = new JPanel(new GridLayout(8,1));
        JPanel rightSide = new JPanel(new GridLayout(8,1));

        HammingDist hammingDist = new HammingDist();
        ArrayList<String> newWords = hammingDist.getWordBank();
```

Start off with row1, it is separate into 2, the left is prompt saying "Enter Hamming Dist:" and the right is JTextField. I mark this as uneducable and then add it.



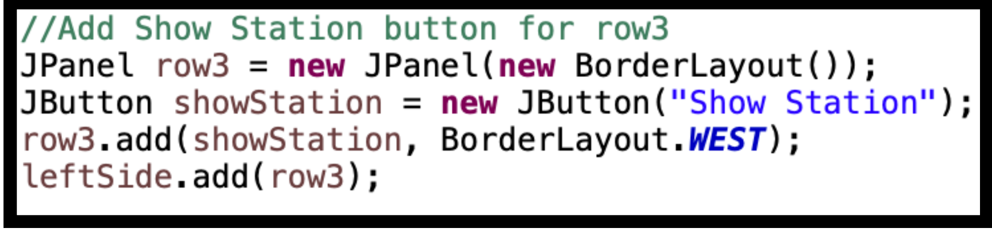
```java
//Start With Row1: Enter Hamming Dist: "input"
JPanel row1 = new JPanel(new GridLayout(1, 2));
JLabel prompt1 = new JLabel("Enter Hamming Dist:");
JTextField input1 = new JTextField();
input1.setText("1");
input1.setEditable(false);
row1.add(prompt1);
row1.add(input1);
leftSide.add(row1);
```

This one is different, as I change from the JPanel to JSlider, and this is the first one that needed ChangeListener that track down the value.



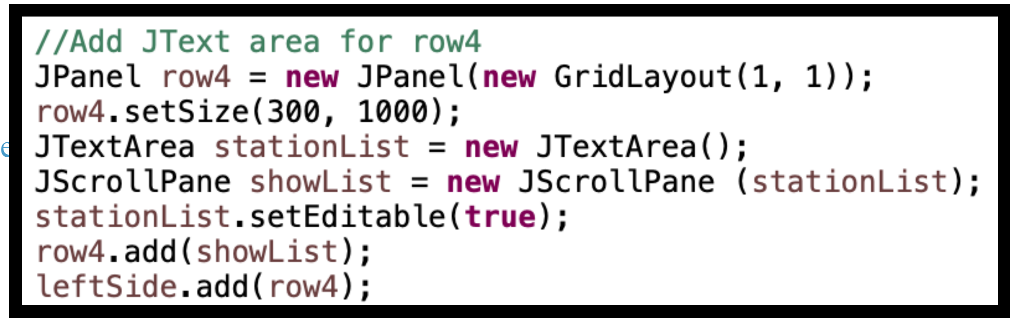
```java
//Add Distant slider for row2
JSlider row2 = new JSlider(1, 4, 1);
row2.setMajorTickSpacing(1);
row2.setMinorTickSpacing(1);
row2.setPaintLabels(true);
row2.setPaintTicks(true);
row2.addChangeListener(new ChangeListener() {
    public void stateChanged(ChangeEvent event) {
        int value = row2.getValue();
        input1.setText(Integer.toString(value));
      }
});
leftSide.add(row2);
```

This one is quite different, as we only need to create a button saying “Show Station”. The add it on the left side. West make it stick to left.



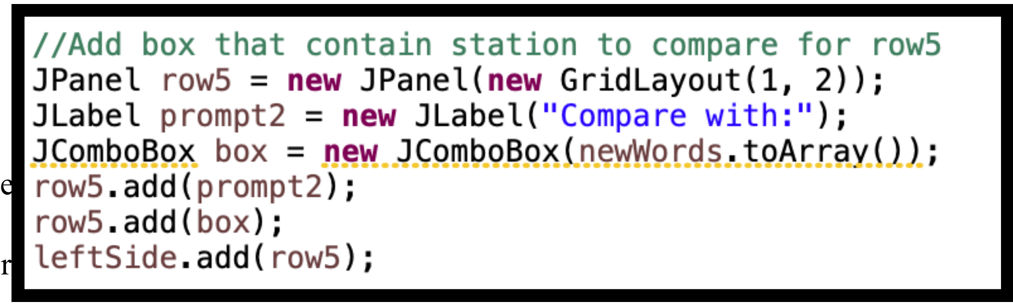
```java
//Add Show Station button for row3
JPanel row3 = new JPanel(new BorderLayout());
JButton showStation = new JButton("Show Station");
row3.add(showStation, BorderLayout.WEST);
leftSide.add(row3);
```

As for row4, I need to create textArea where the result calculate from row 3 is shown.



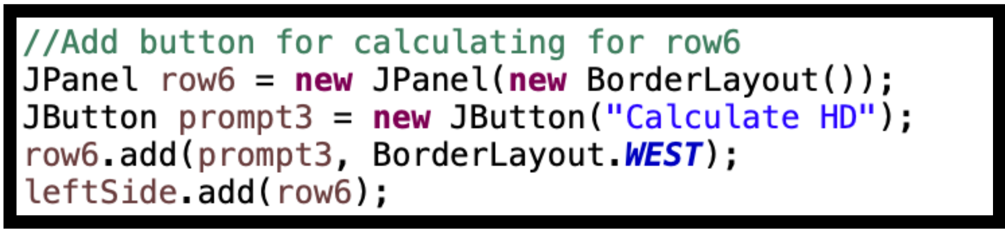
```java
//Add JText area for row4
JPanel row4 = new JPanel(new GridLayout(1, 1));
row4.setSize(300, 1000);
JTextArea stationList = new JTextArea();
JScrollPane showList = new JScrollPane (stationList);
stationList.setEditable(true);
row4.add(showList);
leftSide.add(row4);
```

Row 5 is separate into 2 side, the first showing prompt and the second is a box that we choose the word we want to compare to other



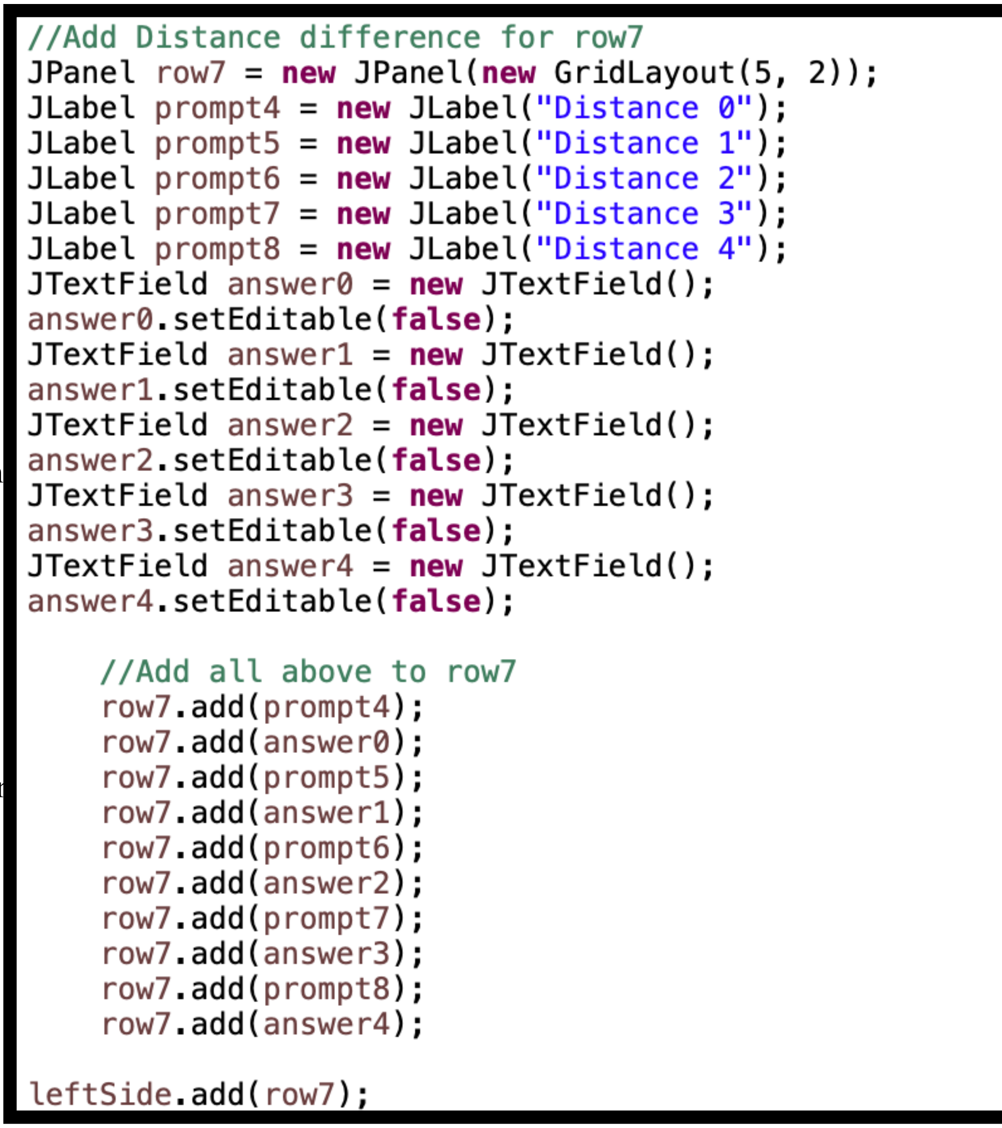
```java
//Add box that contain station to compare for row5
JPanel row5 = new JPanel(new GridLayout(1, 2));
JLabel prompt2 = new JLabel("Compare with:");
JComboBox box = new JComboBox(newWords.toArray());
row5.add(prompt2);
row5.add(box);
leftSide.add(row5);
```

Row6 is the button that when press will calculate the different



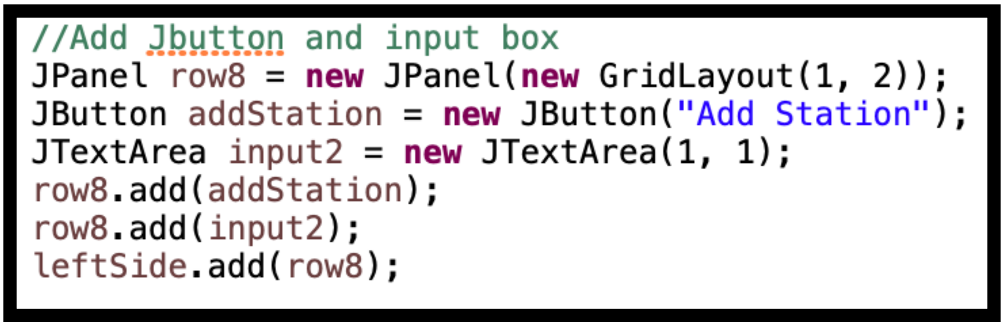
```java
//Add button for calculating for row6
JPanel row6 = new JPanel(new BorderLayout());
JButton prompt3 = new JButton("Calculate HD");
row6.add(prompt3, BorderLayout.WEST);
leftSide.add(row6);
```

Row7 is more complicate because row, we include all distance from 0-4, so we need to create 10 in in total. The first five being prompt and the last five is the answers. This tell the difference in distance as having 1 character different mean distance1 and 4 mean 4, this mean that 0 is the same station that exist within the Mesonet. Each answer is set to be uneditable same as the example then I add all of them to the leftside.



```
//Add Distance difference for row7
JPanel row7 = new JPanel(new GridLayout(5, 2));
JLabel prompt4 = new JLabel("Distance 0");
JLabel prompt5 = new JLabel("Distance 1");
JLabel prompt6 = new JLabel("Distance 2");
JLabel prompt7 = new JLabel("Distance 3");
JLabel prompt8 = new JLabel("Distance 4");
JTextField answer0 = new JTextField();
answer0.setEditable(false);
JTextField answer1 = new JTextField();
answer1.setEditable(false);
JTextField answer2 = new JTextField();
answer2.setEditable(false);
JTextField answer3 = new JTextField();
answer3.setEditable(false);
JTextField answer4 = new JTextField();
answer4.setEditable(false);

    //Add all above to row7
    row7.add(prompt4);
    row7.add(answer0);
    row7.add(prompt5);
    row7.add(answer1);
    row7.add(prompt6);
    row7.add(answer2);
    row7.add(prompt7);
    row7.add(answer3);
    row7.add(prompt8);
    row7.add(answer4);

leftSide.add(row7);
```

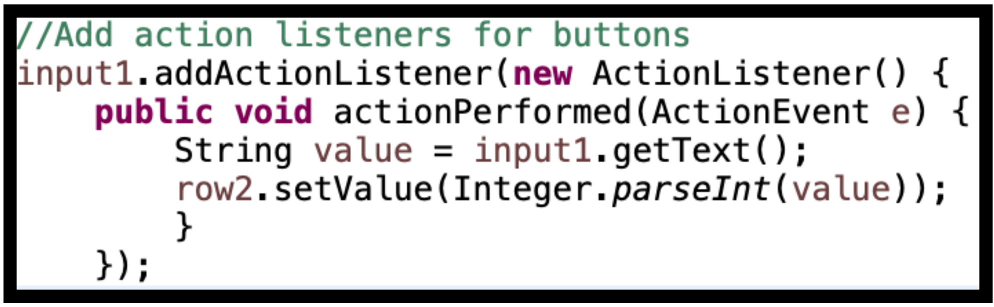
The last row, is separate into two part, the first is the activate button and another is JTextArea.



```
//Add Jbutton and input box
JPanel row8 = new JPanel(new GridLayout(1, 2));
JButton addStation = new JButton("Add Station");
JTextArea input2 = new JTextArea(1, 1);
row8.add(addStation);
row8.add(input2);
leftSide.add(row8);
```

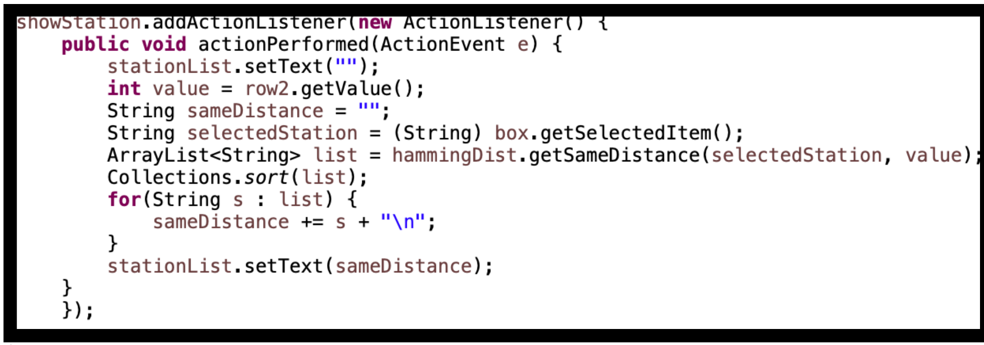
After this, I will explain the activation part of each button.

The first one is connect to the slider in, which we getting the value from the text box and set it as value.



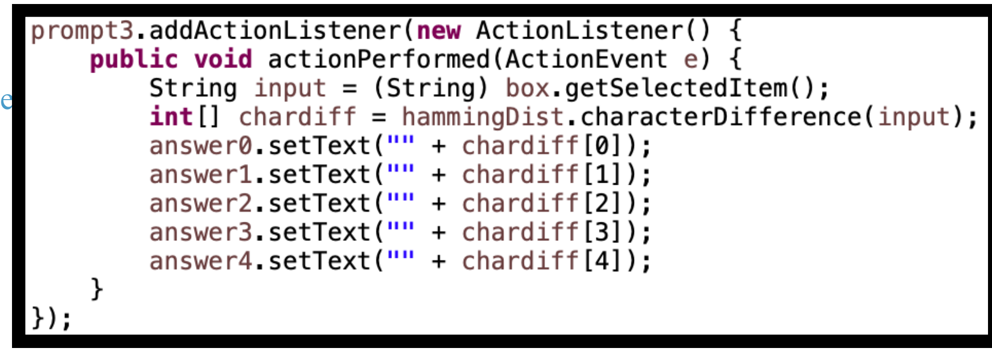
```
//Add action listeners for buttons
input1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String value = input1.getText();
        row2.setValue(Integer.parseInt(value));
        }
    });
```

After that, it is showStation, which is the button that when click will bring the answer to array in row3 this is calculate in the calculation part, in which separate the difference into an array. For this part, we took an array that have same amount of different value to the value on the slider and present it in row 3



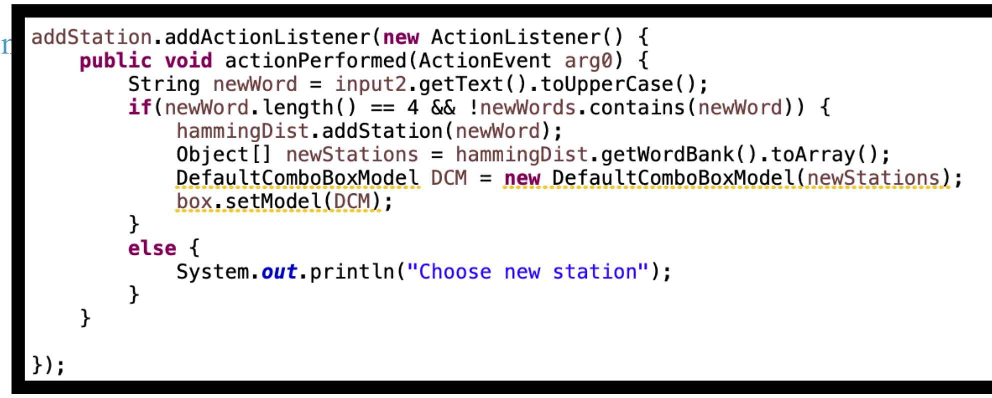
```
showStation.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        stationList.setText("");
        int value = row2.getValue();
        String sameDistance = "";
        String selectedStation = (String) box.getSelectedItem();
        ArrayList<String> list = hammingDist.getSameDistance(selectedStation, value);
        Collections.sort(list);
        for(String s : list) {
            sameDistance += s + "\n";
        }
        stationList.setText(sameDistance);
    }
    });
```

Next is the part that tell how many difference are there in each characte start from 0 difference character to 4 difference character.



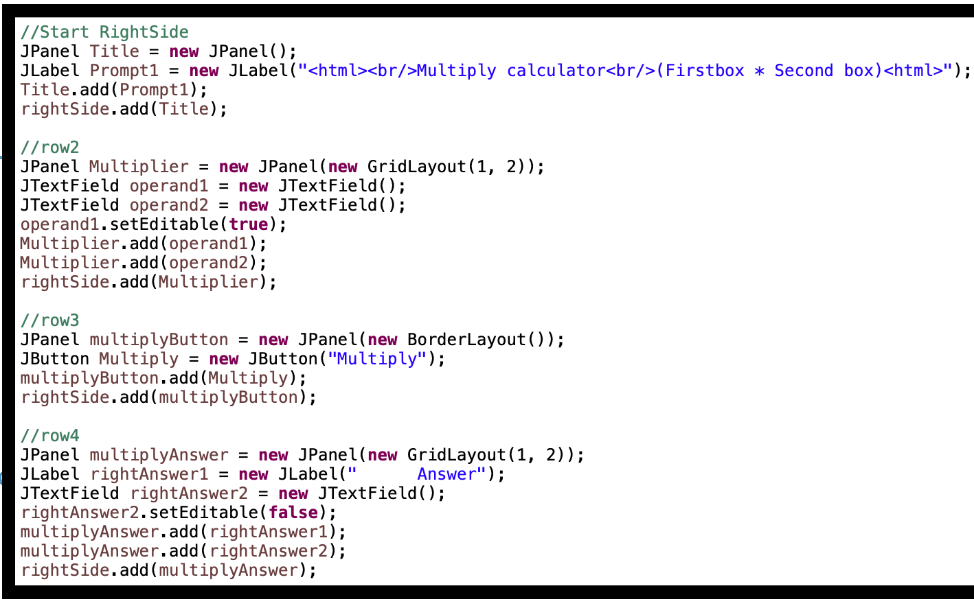
```
prompt3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String input = (String) box.getSelectedItem();
        int[] chardiff = hammingDist.characterDifference(input);
        answer0.setText("" + chardiff[0]);
        answer1.setText("" + chardiff[1]);
        answer2.setText("" + chardiff[2]);
        answer3.setText("" + chardiff[3]);
        answer4.setText("" + chardiff[4]);
    }
});
```

Last is the part where we addStation , in which this word is added to array that are not masonet and it is not a duplicated. For this, it is



```
addStation.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        String newWord = input2.getText().toUpperCase();
        if(newWord.length() == 4 && !newWords.contains(newWord)) {
            hammingDist.addStation(newWord);
            Object[] newStations = hammingDist.getWordBank().toArray();
            DefaultComboBoxModel DCM = new DefaultComboBoxModel(newStations);
            box.setModel(DCM);
        }
        else {
            System.out.println("Choose new station");
        }
    }

});
```

already done so with in another class, in which I add new word into the array that are the copy of masonet, so that when compare there won't be the same station name as the one we add in the array we use. For this, I use if statement that force the station to be 4 character and that it not a duplicate word.



```
//Start RightSide
JPanel Title = new JPanel();
JLabel Prompt1 = new JLabel("<html><br/>Multiply calculator<br/>(Firstbox * Second box)<html>");
Title.add(Prompt1);
rightSide.add(Title);

//row2
JPanel Multiplier = new JPanel(new GridLayout(1, 2));
JTextField operand1 = new JTextField();
JTextField operand2 = new JTextField();
operand1.setEditable(true);
Multiplier.add(operand1);
Multiplier.add(operand2);
rightSide.add(Multiplier);

//row3
JPanel multiplyButton = new JPanel(new BorderLayout());
JButton Multiply = new JButton("Multiply");
multiplyButton.add(Multiply);
rightSide.add(multiplyButton);

//row4
JPanel multiplyAnswer = new JPanel(new GridLayout(1, 2));
JLabel rightAnswer1 = new JLabel("      Answer");
JTextField rightAnswer2 = new JTextField();
rightAnswer2.setEditable(false);
multiplyAnswer.add(rightAnswer1);
multiplyAnswer.add(rightAnswer2);
rightSide.add(multiplyAnswer);
```

After finish with the left side,

I moved on to the right using similar

step to what I did with the left.

I create four row, the first being the

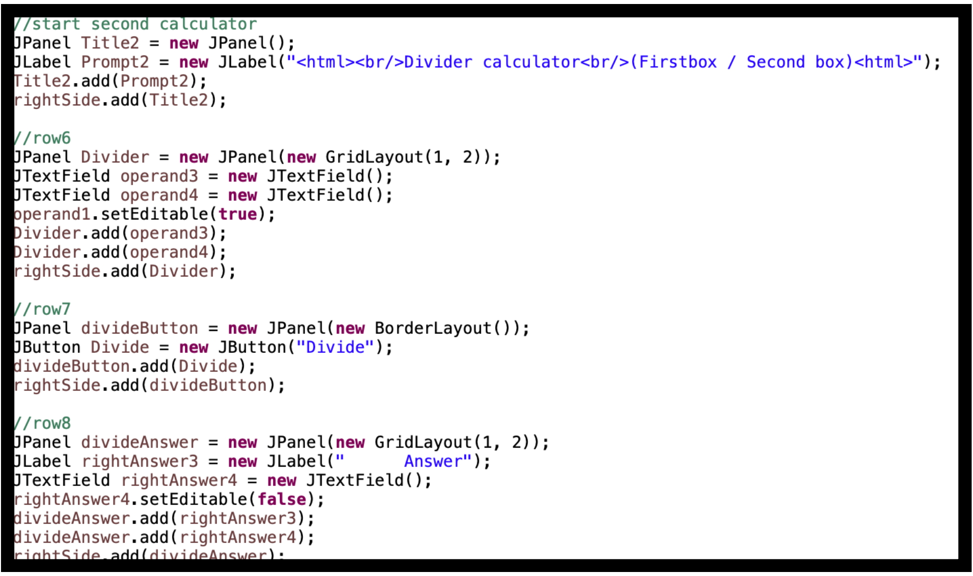
title

Then I create second row for operan

the first box + force operator + last

operand.

Then I crate Third row for button that are activate when you click to calculate the number you input as a operand.



```
//start second calculator
JPanel Title2 = new JPanel();
JLabel Prompt2 = new JLabel("<html><br/>Divider calculator<br/>(Firstbox / Second box)<html>");
Title2.add(Prompt2);
rightSide.add(Title2);

//row6
JPanel Divider = new JPanel(new GridLayout(1, 2));
JTextField operand3 = new JTextField();
JTextField operand4 = new JTextField();
operand1.setEditable(true);
Divider.add(operand3);
Divider.add(operand4);
rightSide.add(Divider);

//row7
JPanel divideButton = new JPanel(new BorderLayout());
JButton Divide = new JButton("Divide");
divideButton.add(Divide);
rightSide.add(divideButton);

//row8
JPanel divideAnswer = new JPanel(new GridLayout(1, 2));
JLabel rightAnswer3 = new JLabel("      Answer");
JTextField rightAnswer4 = new JTextField();
rightAnswer4.setEditable(false);
divideAnswer.add(rightAnswer3);
divideAnswer.add(rightAnswer4);
rightSide.add(divideAnswer);
```

Last row I separate into two part

the first being label saying answer

and the second is the textfield that

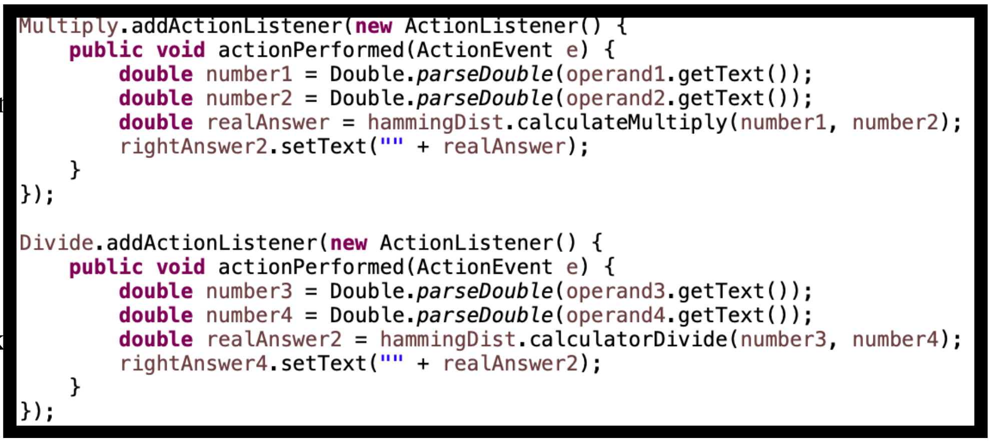
is uneditable that contain the answer

to what you want.

I repeat this step for the divide

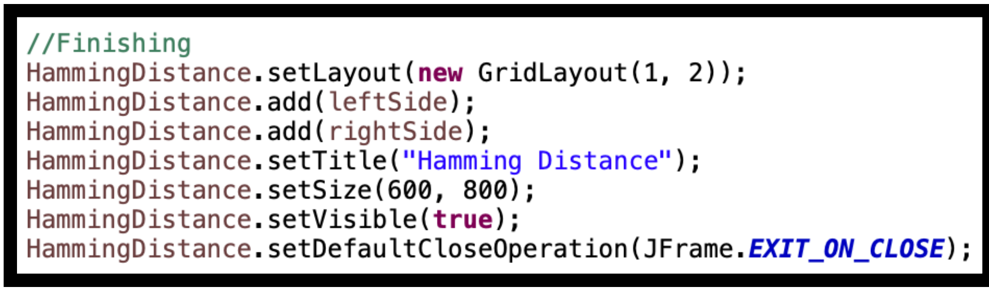
calculator.

After finishing with setting up for the calculator, I create the activatio part. For this, I get the first and second operand and calculate them in another class and pass them back into this class.



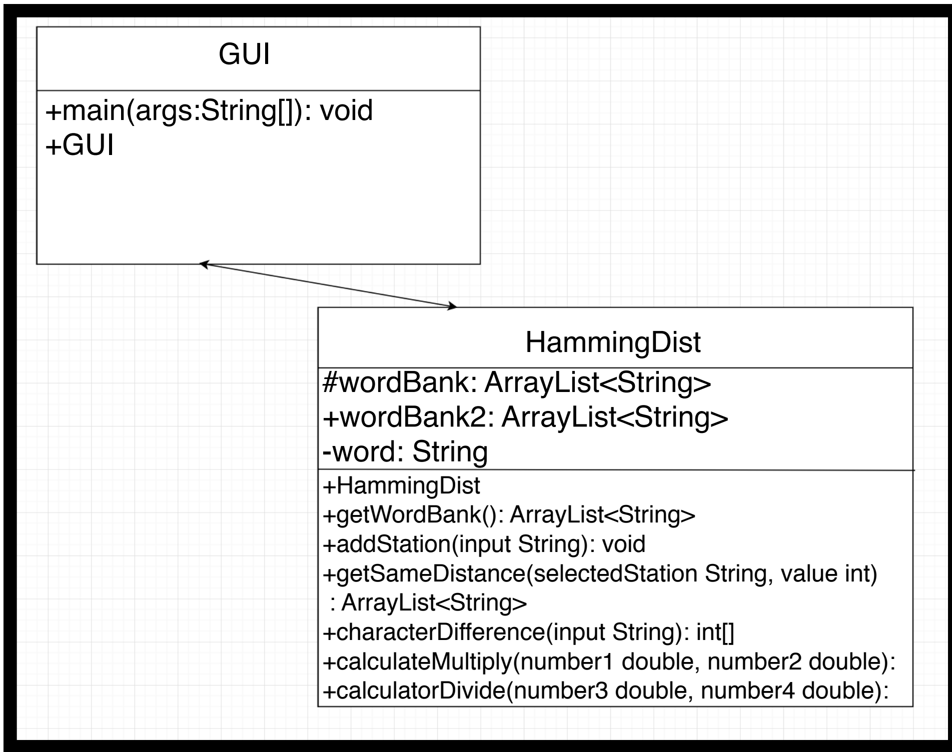
```
Multiply.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        double number1 = Double.parseDouble(operand1.getText());
        double number2 = Double.parseDouble(operand2.getText());
        double realAnswer = hammingDist.calculateMultiply(number1, number2);
        rightAnswer2.setText("" + realAnswer);
    }
});

Divide.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        double number3 = Double.parseDouble(operand3.getText());
        double number4 = Double.parseDouble(operand4.getText());
        double realAnswer2 = hammingDist.calculatorDivide(number3, number4);
        rightAnswer4.setText("" + realAnswer2);
    }
});
```

After finish with everything, I finalize the GUI by setting the layout, adding left and right side, the title, the size, set it to visible and close it when exit.



```
//Finishing
HammingDistance.setLayout(new GridLayout(1, 2));
HammingDistance.add(leftSide);
HammingDistance.add(rightSide);
HammingDistance.setTitle("Hamming Distance");
HammingDistance.setSize(600, 800);
HammingDistance.setVisible(true);
HammingDistance.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Overall, this is a very hard project, that although took the part of calculating from former project, it still hard because of creating the GUI. For this, I did learn a lot on how GUI work, how it operate into different panel, and how to activate each of the component I added in order to make stuff work. In conclusion, this is a very challenging project that teach me a lot on the GUI component.



GUI

+main(args:String[]): void
+GUI

HammingDist

#wordBank: ArrayList<String>
+wordBank2: ArrayList<String>
-word: String

+HammingDist
+getWordBank(): ArrayList<String>
+addStation(input String): void
+getSameDistance(selectedStation String, value int)
 : ArrayList<String>
+characterDifference(input String): int[]
+calculateMultiply(number1 double, number2 double):
+calculatorDivide(number3 double, number4 double):