19AIE105_Object Oriented Programming

PROJECT TOPIC: SCIENTIFIC CALCULATOR

B. Tech (CSE (AI)) (2020 Batch)

Team number: 7

Team members:

AIEA.12026	[DINESH KUMAR M R]
AIEB.29621	[Divi Eswar Chowdary]
AIEB.35658	[Dabbara Harsha]
AIEB.23682	[B.E. Pranav Kumar]

Acknowledgement

We would like to express our special thanks of gratitude to our teacher Dr. Sachin Kumar who gave us the golden opportunity to do this wonderful project on the topic (Scientific Calculator), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.

Table of Contents

List of Figures

Introduction:

Humans naturally use tools and devices to make our work easier, same follows in the field of mathematics. As long back as 2nd century B.C we created basic devices like abacus which made interpretation and memory of intermediate steps in arithmetic calculations easier. With time our understanding of surrounding objects grew and so their creative applications enabled creation of new and better technologies. During the 1960s the first solid-state electronic calculator was created. An electronic calculator is typically a portable electronic device used to perform calculations, ranging from basic arithmetic to complex mathematics. Today, these kinds of calculators are available to the general public from vendors to accountants who perform basic mathematics in their day-to-day life but the scientific and academic communities require a higher degree of mathematical computations to better aid them, to accommodate for their need additional features were added to them and they were termed as scientific calculators. Some of the features distinct to scientific calculators are logarithms, trigonometric functions, exponential functions. Extra features are unique to the need of customers or the company.

Aim:

We have created a digital scientific calculator with a GUI (graphical user interphase) which is coded based on the object-oriented programing concepts in java.

Materials required:

Eclipse (IDE latest version)

Window builder package - we have used this to simplify our code to make it easier to interpret and create the GUI

Approach

The following UML diagram shows the relationship between the classes we have used.

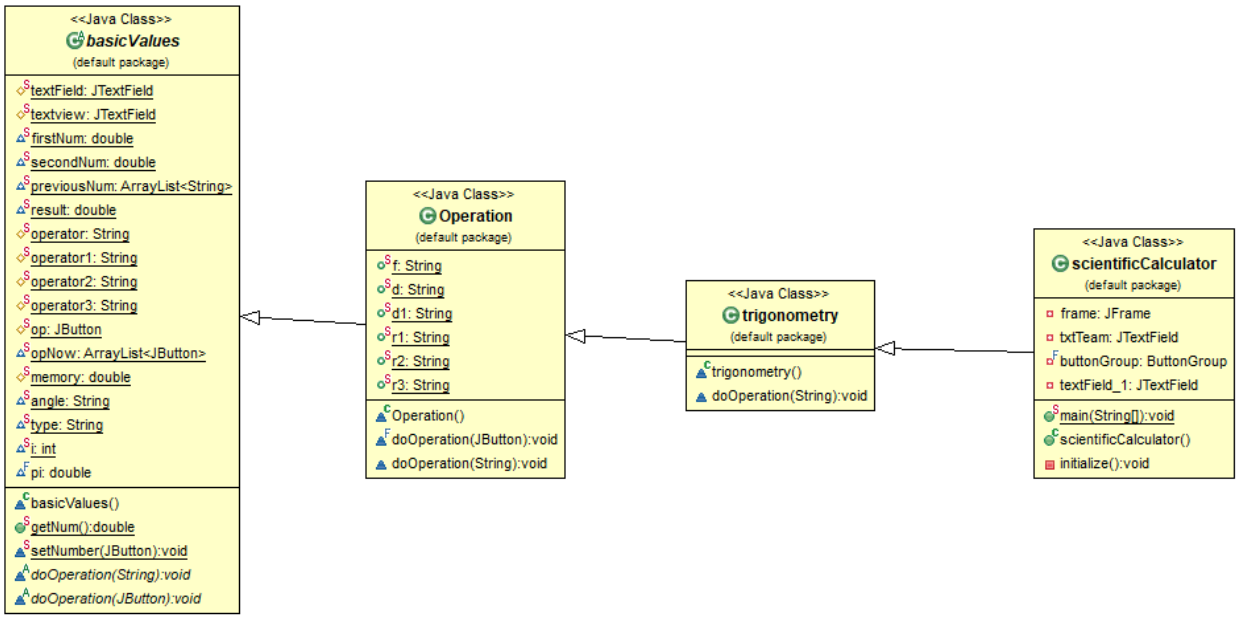

Figure 1- UML DIAGRAM

Clearly, we can see that this is a case of multi-level inheritance.

This type of code promotes reusability and reliability, this can be felt during the process of coding as certain bugs which were common to the entire program were rectified by changing the features of super class.

Algorithm and Code:

Functionalities in our calculator:

1.On/Off: we can control the state of the device with these two buttons.

The calculator cannot exhibit two states at the same time and this functionality is achieve by using a pair of grouped radio buttons. These

kinds of buttons are group complimentary I.e., only one given button of the group can be active at a given time.

Figure 2-Off

We can see from the above image that when the off button is selected the on button is automatically unselected. We can also see that the state of the input buttons (0 to 9) is inactive. Vise-versa when on button is selected.

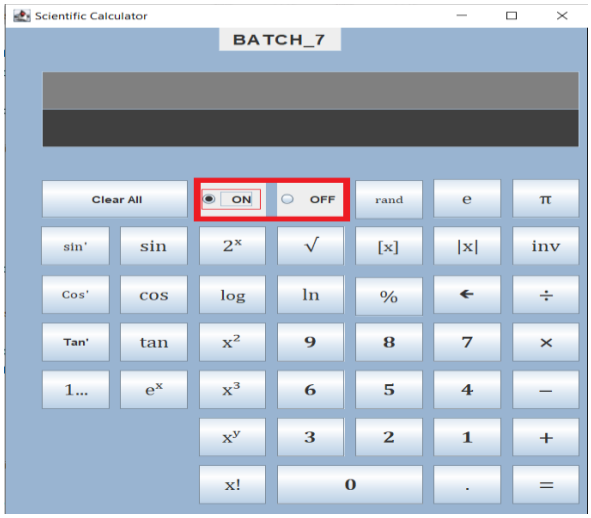

Figure 3-On

Java code:

```
            bt0.setEnabled(true);
            bt1.setEnabled(true);
            bt2.setEnabled(true);
            bt3.setEnabled(true);
            bt4.setEnabled(true);
            bt5.setEnabled(true);
            bt6.setEnabled(true);
            bt7.setEnabled(true);
            bt8.setEnabled(true);
            bt9.setEnabled(true);
            }
});
buttonGroup.add(rdbtnNewRadioButton);
rdbtnNewRadioButton.setBounds(180, 193, 63, 49);
frame.getContentPane().add(rdbtnNewRadioButton);

JRadioButton rdbtnNewRadioButton_1 = new JRadioButton("   OFF");
rdbtnNewRadioButton_1.setSelected(true);
rdbtnNewRadioButton_1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
      textField.setText(null);
      textview.setText(null);

      bt0.setEnabled(false);
      bt1.setEnabled(false);
      bt2.setEnabled(false);
      bt3.setEnabled(false);
      bt4.setEnabled(false);
      bt5.setEnabled(false);
      bt6.setEnabled(false);
      bt7.setEnabled(false);
      bt8.setEnabled(false);
      bt9.setEnabled(false);
}
```

2.Clear all: this button is used to clear the previous input if any was given and if not to ensure that previous commands won't affect the current calculation.

Figure 4-general calculation

Figure 5-Clear all

Java code:

```
textField.setText(null);
textview.setText(null);
firstNum=0;
secondNum=0;
```

3.e and π: these buttons are used for some essential constant values that are most commonly used in calculations. Adding these as buttons allows the user to be faster in calculations.

Figure 6-exponential function

Figure 7-Pi

Java code:

```
operator2=firstNum+" × e";
result=firstNum*(Math.E);
```

```
operator2=firstNum+" × pi";
result=firstNum*(Math.PI);
```

4.rand: this button is used to randomly generate real number values between 0 and 1. This can be used when we required to factor random possibilities during calculations.

Figure 8-Random Number

Java code:

```
operator2="rand ( "+firstNum+" )";
result=Math.random();
```

5.sin, cos and tan: these three buttons are used to perform basic trigonometric functions. The values given for calculating are considered in degrees.

6.sini, Cosi and tani: these three buttons are corresponding inverse functions of the trigonometric functions.

Figure 9-Trignometric Functions

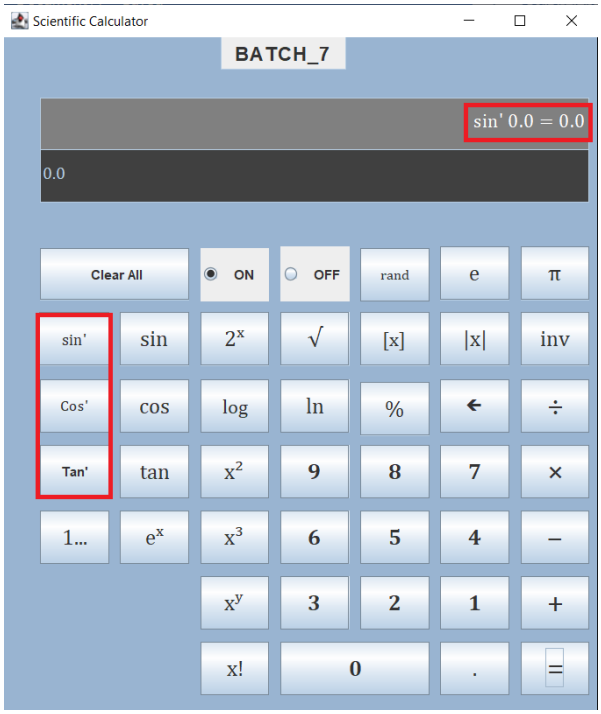

Figure 10-Inverse Trignometric Function

Java code:

```
case "sin":
    operator2="sin "+firstNum;
    result=Math.sin(firstNum*0.0175);
    textview.setText("sin"+"("+firstNum+")");
    break;
case "cos":
    result=Math.cos(firstNum*0.0175);
    operator2="cos "+firstNum;
    textview.setText("Cos"+"("+firstNum+")");
    break;
case "tan":
    result=Math.tan(firstNum*0.0175);
    operator2="tan "+firstNum;
    textview.setText("tan"+"("+firstNum+")");
    break;
case "sini":
    result=57.2958* Math.asin(firstNum);
    operator2="sin' "+firstNum;
    textview.setText("sin'"+"("+firstNum+")");
    break;
case "cosi":
    result=57.2958* Math.acos(firstNum);
    operator2="cos' "+firstNum;
    textview.setText("cos'"+"("+firstNum+")");
    break;
case "tani":
    result=57.2958* Math.tan(firstNum);
    operator2="tan' "+firstNum;
    textview.setText("tan'"+"("+firstNum+")");
}
r3=operator3;
textField.setText(String.valueOf(result));
op = null;
```

7.power functions: square root, x^2 square, x^3cube, x^y power value y, factorial. are certain other common functions.

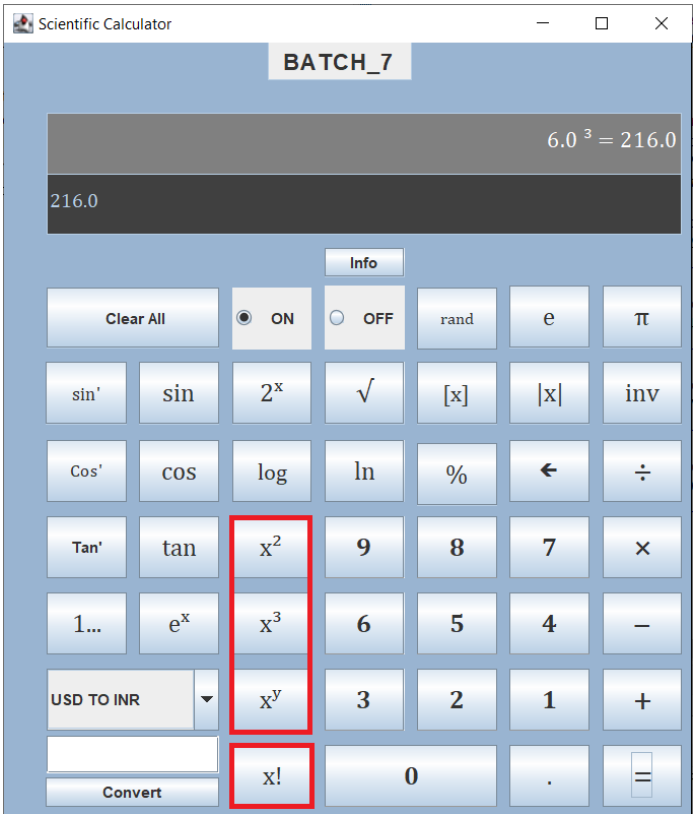

Figure 11-Power functions

Java code:

```
case "x^2":
    operator2=firstNum+" ²";
    result=Math.pow(firstNum,2);

    break;
case "x^3":
    operator2=firstNum +" ³";
    result=Math.pow(firstNum,3);

    break;
case "10^x":
    operator2="10 ^ "+firstNum;
    result=Math.pow(10,firstNum);

    break;
case "2^x":
    operator2="2 ^ "+firstNum;
    result=Math.pow(2,firstNum);

    break;
```

```
case "x\u02B8":                                    //for doing x power y operation
    operator1=firstNum+" ^ "+ secondNum;
    result=Math.pow(firstNum,secondNum);
    break;
```

```
operator2=firstNum+" !";
double fact=1;
for(double i=firstNum; i>0; i--) {
    fact*=i;
}
result=fact;
```

8.exponential functions: this is another common function.

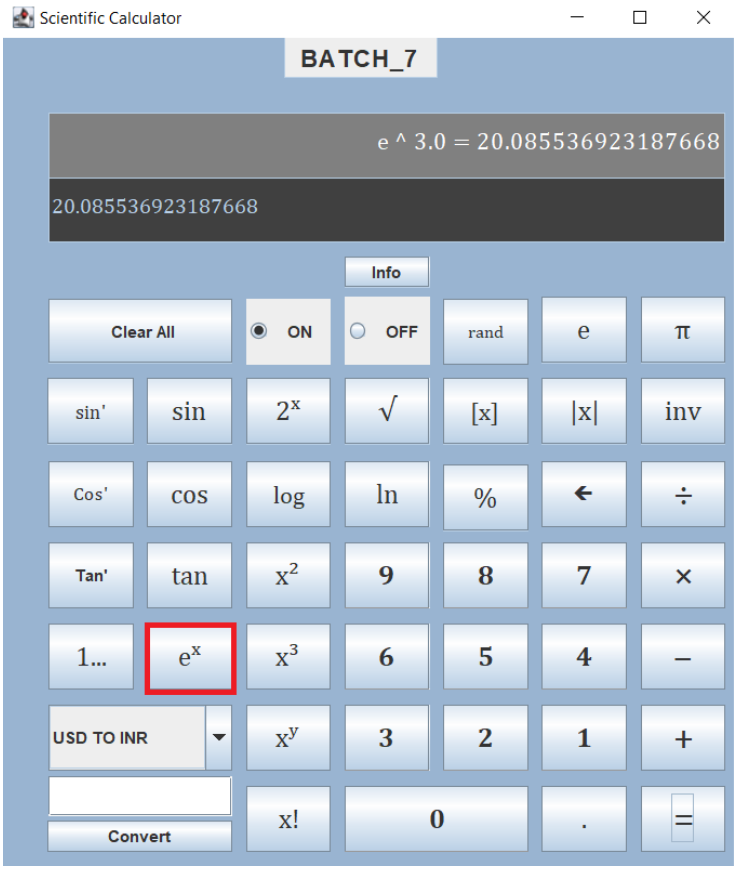

Figure 12-Exponential Function

Java code:

```
operator2="e ^ "+firstNum;
result=Math.exp(firstNum);
```

9. logarithms: two buttons ln and log are logarithmic functions with bases e and 10 respectively.

Figure 13-Logarithm

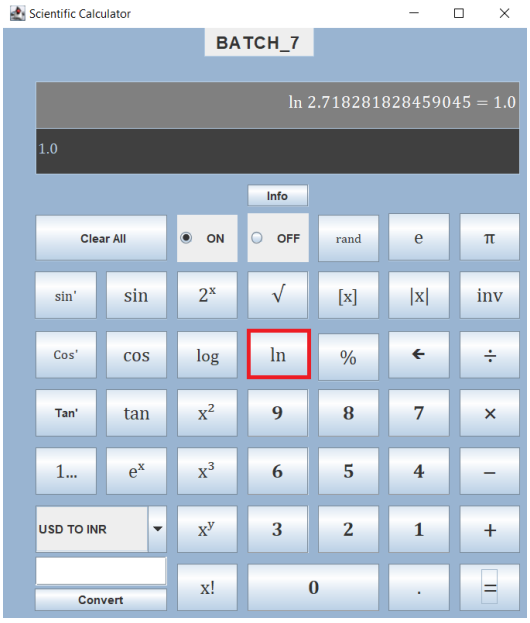

Figure 14-Natural logarithm

Java code:

```
case "log":
    operator2="log "+firstNum;
    result=Math.log10(firstNum);

    break;
case "ln":
    operator2="ln "+firstNum;
    result=Math.log(firstNum);
    break;
```

10.arithmetic functions: these are basic functions that are present in every calculator.

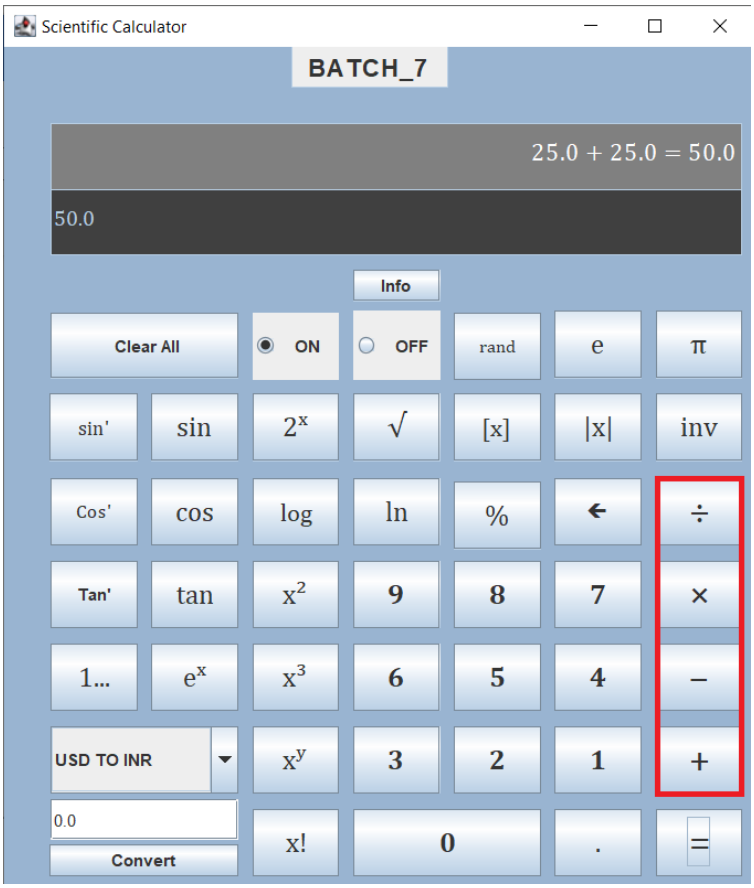

Figure 15-Add , Subtract , Multiply , Divide

Java code:

```
case "+":
    result=firstNum+secondNum;
    operator1=firstNum+" + "+ secondNum;
    break;
case "−":
    operator1=firstNum+" − "+ secondNum;
    result=firstNum-secondNum;
    break;
case "×":
    operator1=firstNum+" × "+ secondNum;
    result=firstNum*secondNum;
    break;
case "÷":
    operator1=firstNum+" ÷ "+ secondNum;
    result=firstNum/secondNum;
    break;
case "%":
    operator1=firstNum+" % "+ secondNum;
    result=firstNum%secondNum;
    break;
case "x\u02B8":
    operator1=firstNum+" ^ "+ secondNum;
    result=Math.pow(firstNum,secondNum);
    break;
case "log\u2090x":
```

11. Backspace: removes the last entered digit.

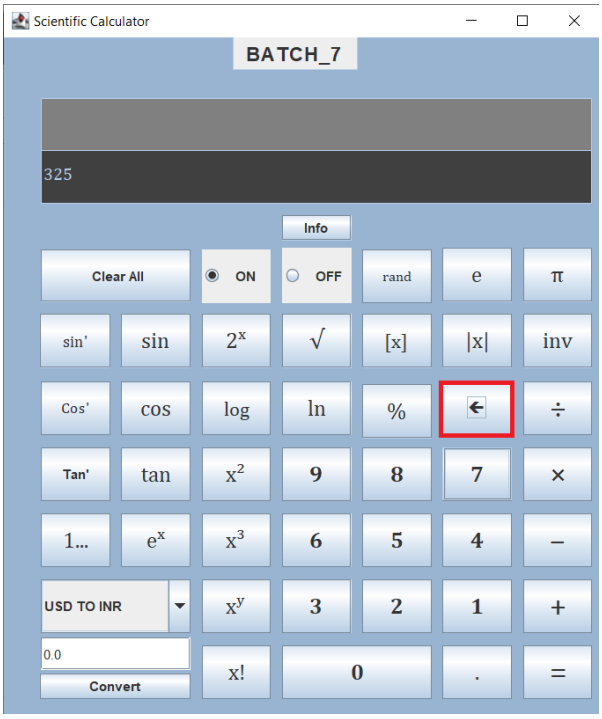

Figure 16-Back space

Java code:

```
if(textField.getText().length()>0) {
    StringBuilder str=new StringBuilder(textField.getText());
    str.deleteCharAt(textField.getText().length()-1);
    backSpace=str.toString();
    textField.setText(backSpace);
```

12. Converter: this section contains a drop box with which we can select some basic conversions.

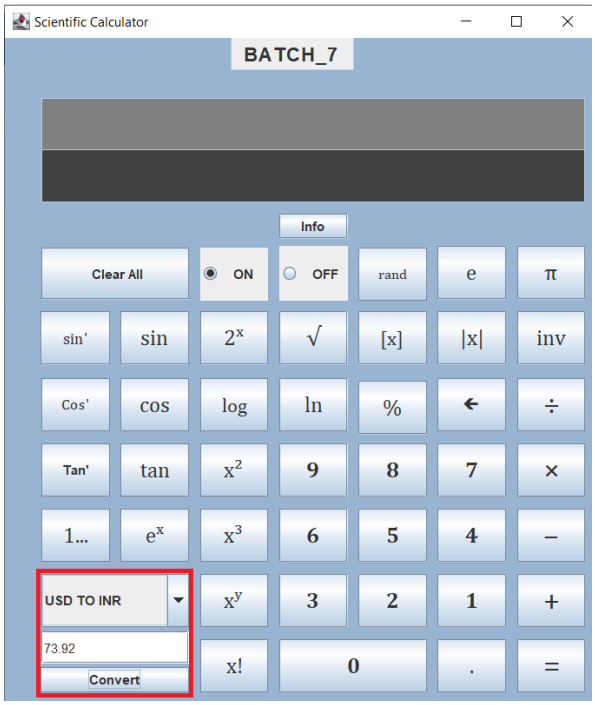

Figure 17-Converter

Java code:

```
firstNum=getNum();
Double value;
if(comboBox.getSelectedItem().toString() == "USD TO INR") {
    value=73.92*firstNum;
    textField_1.setText(String.valueOf(value));
    }
else if(comboBox.getSelectedItem().toString() == "INR TO USD") {
    value=0.014*firstNum;
    textField_1.setText(String.valueOf(value));                    // The conversion of currency as of 28/02/2021 12:03:15
    }
else if(comboBox.getSelectedItem().toString() == "KMPH TO MPS") {
    value=0.27777778*firstNum;
    textField_1.setText(String.valueOf(value));
    }
else if(comboBox.getSelectedItem().toString() == "Rad To Deg") {
    value=57.2957795*firstNum;
    textField_1.setText(String.valueOf(value));
    }
```

13.step, modulus and inverse: predefined functions

Figure 18-Step , module , Inverse

Java code:

```
case "inverse":
    operator2=firstNum+" ⁻¹";
    result=1/firstNum;

    break;
case "modulus":
    operator2="| "+firstNum+" |";
    result=Math.abs(firstNum);
```

Some observations:

- This calculator contains two display outputs rather than one, the first text box is used to show the process undertaken to find the output and the second text box displays the result alone.
- We have divided the calculator into different zones and each zone contains a separate set of related functions. This makes it more user friendly.
- When the calculator is in off state only the input buttons are switched off not all operations.

OOPS concepts used:

1. Inheritance: this can be realized by looking at UML diagram or the division and extension of classes.
2. Polymorphism: we are overloading two methods repeatedly to achieve all functionalities.

Conclusion:

Java offers the real possibility that most programs can be written in a type-safe language. However, for Java to be broadly useful, it needs to have more expressive power than it does at present.

This project addresses one of the areas where more power is needed. It extends Java with a mechanism for parametric polymorphism, which allows the definition and implementation of generic abstractions. This gives a complete design for the extended language. The proposed extension is small and conservative and the project discusses the rationale for many of our decisions. The extension does have some impact on other parts of Java, especially Java arrays, OOPS concepts and the Java class library.

References:

www.google.com

For Project Related Files:

https://amritavishwavidyapeetham-my.sharepoint.com/:f:/g/personal/aieb_29621_cb_students_amrita_edu/EhbpmkwYhPxGigrjqXp-j08BoEX5bmtVkJamb32OUrc_JA?e=rpkWK5