# Lab Report

EXERCISE 7: FUN WITH CALCULATORS 2

| | Name | Titel des Kurses | Datum |
|---|---|---|---|
| Juri Wiechmann<br>Fatima Rindert | 571085<br>571628 | Prof. Dr. Weber-Wulff<br>Info 2<br>Group 2 | 9.12.2019 |

## Index

Introduction

Assignments

1. Make a copy of one of your Calculator. Make sure that it works before you…
2. Rework it to accept a long String of single digits separated by operators that…
3. Once you get a String input, add in calls to convert this expression to postfix…
4. What will be a good test cases for the precendence of * over +? Find 15 good…
5. Check that this still works with hexadecimal numbers.
6. Now include the exponentiation operator ^ (2^4 = 16). It has a higer priority…
7. Extend your integer calculate to do multidigit  and doubles.

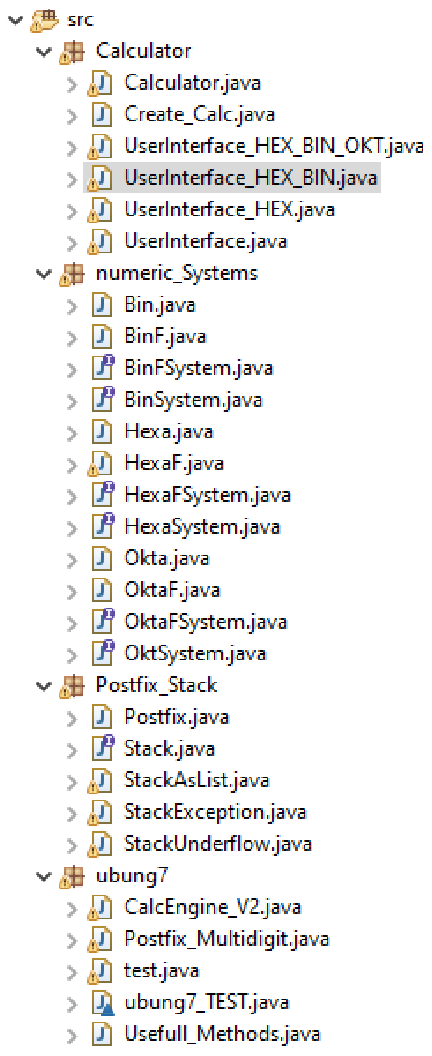
Reflections

Juri

Fatima

## Introduction

In this week's lab we worked with the Calculator from lab 5 and with the Reverse Polish Notation classes from the last lab. Because we had a lot extra things in the Calculator like extra Textfields and the other numeric system it will be a tough challenge. Also we wanted to create the floating points on each numeric system, multi digits and negative numbers. Unknowing of the hardships we would face on our way to the goal we had set ourselves. In this report we will focus more on the final result and explaining then the way to the final result concomitant by some mention of issues that we had. On the right you can see our final result. The only thing we didn't use was the Postfix-class. Because last we hadn't multidigits and Hexadecimal-numbers we needed a new one. We also created for each numeric System a floating-point version which is marked by the F. But this time the static methods which we need to convert are written in the interfaces.



```
src
  Calculator
    Calculator.java
    Create_Calc.java
    UserInterface_HEX_BIN_OKT.java
    UserInterface_HEX_BIN.java
    UserInterface_HEX.java
    UserInterface.java
  numeric_Systems
    Bin.java
    BinF.java
    BinFSystem.java
    BinSystem.java
    Hexa.java
    HexaF.java
    HexaFSystem.java
    HexaSystem.java
    Okta.java
    OktaF.java
    OktaFSystem.java
    OktSystem.java
  Postfix_Stack
    Postfix.java
    Stack.java
    StackAsList.java
    StackException.java
    StackUnderflow.java
  ubung7
    CalcEngine_V2.java
    Postfix_Multidigit.java
    test.java
    ubung7_TEST.java
    Usefull_Methods.java
```

## Assignments

1. **Make a copy of one of your Calculators. Make sure that it works before you begin! If neither of you got a Calculator to work, ask colleagues for permission to use theirs. Give them credit in your report!**

From lab 5 we used the Juri`s Calculator which had every nouns worth numeric system included.

2. **Rework it to accept a long String of single digits separated by operators that have precedence. The bored may use multi-digit numbers and floating-point or scientific notation, if they please.**

Because we had a clean Calculator and a postfix Method. We replaced the Calcengine with our Calcengine_V2 which is extended by our Postfix-class. Now we needed to make some changed to in the ActionPerformed-methods that we put every number and operator in the DisplayValue and by pressing "=" we give this DisplayValue to the Postfix-method. It worked only with Decimal-numbers.

We extended it by 2 seperated Text fields at the top. One that gives us the way how we calculate something and one for our Input or our result which works as Input for the next calculation. To every time we pressing a Number it Adds it to the Input. Every time we press an Operator we add our Input to the DisplayValue and every time we press “=” we send the whole DisplayValue to our Postfix-methods. Later in 7 we will add some else if Statements and variables to get it work.  We had a lot trouble here. Because we want to use our result for more calculations we needed a specific order of updating and setting the Textfields.

```java
public void actionPerformed (ActionEvent event)
{
    String command = event.getActionCommand();
    //Operator!
    if(Usefull_Methods.CharisOperator(command.charAt(command.length()-1))) {
         calc.ADDtoDisplayValue(calc.getinput());
         // for "="
         if(command.contentEquals("=")) {
                try {
                       calc.calcResult();
                       } catch (StackUnderflow | StackException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                       }
                calc.setinput(Double.valueOf(calc.result));
                redisplay();
                calc.setDisplayValue(calc.result);
                reIndex_Display();
                calc.setinput(0);
         }
         //Everything else
         else {
                calc.ADDtoDisplayValue(command);
                calc.setinput(0);
                redisplay();
         }
    }
  //Number
    else if(Usefull_Methods.StringisNumber(command)){
         AddNumber(command);
    else if(command.equals("Clear")) {
        calc.clear();
        redisplay();

    }
    else if(command.equals("?")) {
        showInfo();
    }
    if(!command.contentEquals("=")) {
         reIndex_Display();
    }
}
```

3. **Once you get a String input, add in calls to convert this expression to postfix and evaluate the postfix when = is pressed. Presto, your calculator now takes care of operator priorities, like magic! Just a little bit of mathematical thought, and you can introduce new functionality!**

In our Engine we called now the "calcResult()"-method which just returns the result:

```
public void calcResult() throws StackUnderflow, StackException {
	result = Double.toString(evaluate(infixToPostfix(getDisplayValue())));
	}
```

4. **What will be good test cases for the precedence of * over +? Find 15 good test cases and try them out, documenting the results.**

We wrote a Method with 7 char-parameter which we call 15 times with different char values which can contain operator and number:

```
private Calculator
MULToverADD(Calculator Test,char num1, char op1, char num2, char op2, char
num3, char op3, char num4){
	event = new ActionEvent(Test.gui, 0, "Clear");
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+num1);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0, ""+op1);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+num2);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+op2);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+num3);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+op3);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0,""+num4);
		Test.gui.actionPerformed(event);
	event = new ActionEvent(Test.gui, 0, "=");
		Test.gui.actionPerformed(event);
		return Test;

	}

	//for 4 15 Test cases:
	Test = MULToverADD(Test, '1', '+', '2', 'x', '3', '+', '4');
	assertEquals("11.0",Test.engine.result);
	Test = MULToverADD(Test, '2', '+', '1', 'x', '4', '+', '3');
	assertEquals("9.0",Test.engine.result);
	Test = MULToverADD(Test, '1', '+', '2', 'x', '3', '+', '4');
	assertEquals("11.0",Test.engine.result);
	Test = MULToverADD(Test, '2', '+', '5', 'x', '5', '+', '2');
	assertEquals("29.0",Test.engine.result);
	Test = MULToverADD(Test, '1', 'x', '2', '+', '3', 'x', '4');
	assertEquals("14.0",Test.engine.result);
	Test = MULToverADD(Test, '2', 'x', '1', '+', '4', 'x', '3');
	assertEquals("14.0",Test.engine.result);
	Test = MULToverADD(Test, '2', 'x', '5', '+', '5', 'x', '2');
	assertEquals("20.0",Test.engine.result);
	Test = MULToverADD(Test, '1', 'x', '2', 'x', '3', '+', '5');
```

```
        assertEquals("11.0",Test.engine.result);
        Test = MULToverADD(Test, '2', 'x', '1', 'x', '4', '+', '1');
        assertEquals("9.0",Test.engine.result);
        Test = MULToverADD(Test, '2', 'x', '5', 'x', '5', '+', '2');
        assertEquals("52.0",Test.engine.result);
        Test = MULToverADD(Test, '1', '+', '2', 'x', '3', 'x', '4');
        assertEquals("25.0",Test.engine.result);
        Test = MULToverADD(Test, '2', '+', '1', 'x', '4', 'x', '3');
        assertEquals("14.0",Test.engine.result);
        Test = MULToverADD(Test, '2', '+', '5', 'x', '5', 'x', '2');
        assertEquals("52.0",Test.engine.result);
        Test = MULToverADD(Test, '1', '+', '2', 'x', '1', '0', '0');
        assertEquals("201.0",Test.engine.result);
        Test = MULToverADD(Test, '4', '+', '3', 'x', '2', '/', '6');
        assertEquals("5.0",Test.engine.result);
```

Everything was right.

## 5. Check that this still works with hexadecimal numbers.

Well no. It doesn't we discusses what we can do about it. We could add Hexadecimal-Numbers so our Number count and deal with them this way but then its really hard to work with other numeric system. We also wanted only to use the Decimal-system in our Calculator and Postfix-class. Because of this we decided to add multi digits here. Last Lab I already wanted to do this but we found out, that you need spaces between each token to make it clearly how the code have to deal with the String if numbers got more than one digit. To do so we want to make clear, that there aren't any spaces first. So we wrote a code that delete all spaces in our String:

```
        public static String deleteSpaces(String s) {
                return s.replace("\\s+", "");
        }
```

Not that much of a deal but now we need a method that can separate between multidigit numbers and operators. We also wanted negative numbers to this task is going to be hard. Our Method: `setSpacesEachToken(String S)` will first delete all spaces. Then we create an char Array with S. Later we will deal with floating points numbers so we need to count "." as number too. In order of avoid more dots in a row we first searching for dots right behind each other. If there is one we return null:

```
        public static String setSpacesEachToken(String S) {
                S = deleteSpaces(S);
                char [] S_Array = S.toCharArray();

                //If the input String has 2 dots in a row its wrong Tested here.
                int count = 0;
                for(char E : S_Array) {
                        if(E == '.')
                                count++;
                        else
                                count = 0;
                        if(count >1)
                                return null;
                }
```

Now we create a String “token”. And set S to =”” to return it later with tokens that we add in our for loop in specific rules:

```
String token = "";
S = "";
```

The main idea is that we iterate every char and add it to the token in a inner loop. We got 3 If-statements to ask to which kind of token our char belongs. A negative, a positive number or an operator. The operator once was the easiest one because we have all time only one char that we need to add. So simply if it’s an operator we add them to the Token and at the Token then to the String:

```
                //operatoren
                if( i > 0  && CharisOperator(S_Array[i])) {
                        token += S_Array[i];
                        S += token + " ";
                        continue;
                }
```

Let’s continue with the positive numbers. If our char is an Number we add them to the token and then we call a loop which continue as long the next char is an number too. This way we still got a O(N) complexity because it continues the outer loop:

```
        //positive Zahlen
        else if(CharisNumberHEX(S_Array[i])){
                token = Character.toString(S_Array[i]);
                try {
                        while(CharisNumberHEX(S_Array[i+1]) || S_Array[i+1] == '.') {
                                i++;
                                token += S_Array[i];
                        }
                }
                catch(Exception e) {}
                S += token + " ";
                continue;
        }
```

…

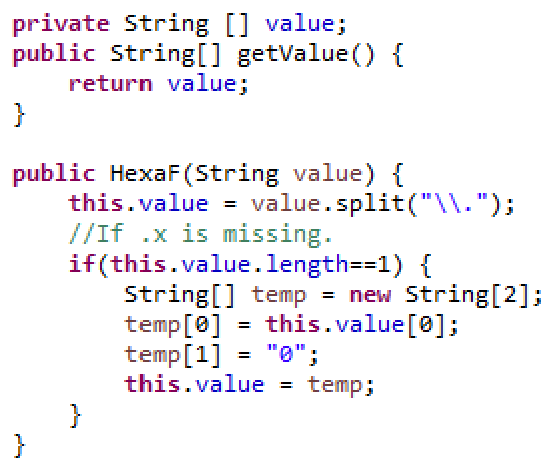
```
        public static boolean CharisNumberHEX(char number) {
                if( number > 47 && number < 58 ||
                        number > 64 && number < 71)
                        return true;
                else
                        return false;
        }
```

Now we get to the hardest condition, negative numbers. We go in here if we get a -. We add them to the Token. Then we have to decide if its an – followed by a number or followed by an “(“. If its followed by a number we also need to take a look if it’s the first digit like: -1+4 or if in front of our “-“ is another operator:

```
//negativ numbers
if(     S_Array[i] == '-') {
        token += S_Array[i];

        try {
                //if its an neg followed by a number
                if(CharisNumberHEX(S_Array[i+1])){
                        //If - is the first digit:
                        if(i==0) {
                              while(CharisNumberHEX(S_Array[i+1])||S_Array[i+1]=='.') {
                                        i++;
                                        token += S_Array[i];
                                }
                        }
                        else {
                                //if the - is not at the first digit
                                if(CharisOperator(S_Array[i-1])) {
                                        while(CharisNumberHEX(S_Array[i+1]) || S_Array[i+1] == '.') {
                                                i++;
                                                token += S_Array[i];
                                        }
                                }
                        }
                }
                //if the neg is followed by an (
                else if(S_Array[i+1] == '(') {
                        i++;
                        token += S_Array[i];
                }
        }
        catch(Exception e) {}
        S += token + " ";
        continue;
}
```

Now we have spaces between each token. We need to change some things in the Postfix-class. We change here also our Token to a String and changed everything that it worked the same. Code only be showed in the appendix because no functionality changed.

Now we need to convert a String like: A+B*5+(A-4). We also wanted to have floating point Number in every numeric system so we created 3 new Interfaces and 3 new Classes of HEX,BIN and OKT which can deal with floating point and gives us an method to convert whole Strings with operations in it. I will only explain it in HEX because the other ones are really similar. Only the Base change and the methods names that we call. Our Interface Contains 3 static methods with Body and 6 which are abstract. The abstact once are only Calculation Methods that I never used. The idea is that we safe our Before and After points values in a 2 container long String Array.



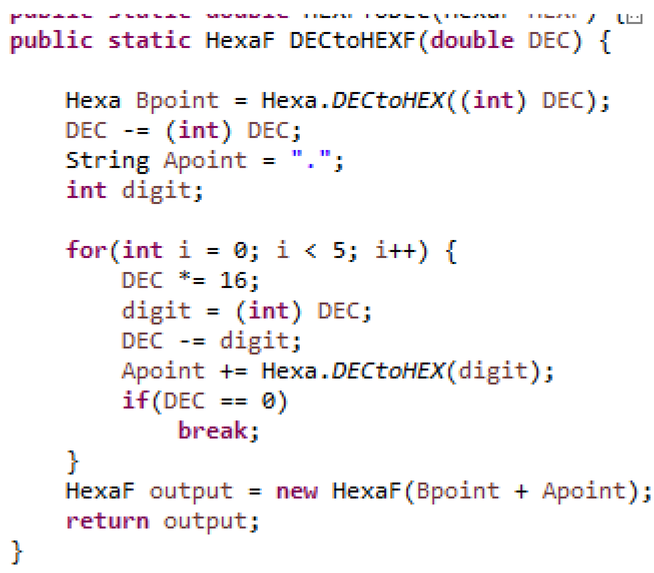
```
private String [] value;
public String[] getValue() {
    return value;
}

public HexaF(String value) {
    this.value = value.split("\\.");
    //If .x is missing.
    if(this.value.length==1) {
        String[] temp = new String[2];
        temp[0] = this.value[0];
        temp[1] = "0";
        this.value = temp;
    }
}
```

Let’s go through each static Method:

```
public static double HEXFToDEC(HexaF HEXF) {
        int Bpoint = Hexa.HEXToDEC(new Hexa(HEXF.getValue()[0]));
        char[] c_Apoint = HEXF.getValue()[1].toCharArray();
        double Apoint = 0;

        for(int i = 0 ; i< c_Apoint.length;i++) {
                int hex_Value = Hexa.HEXToDEC(new Hexa (Character.toString(c_Apoint[i])));
                Apoint +=  hex_Value*Math.pow(16, -(i+1));
        }
        return Bpoint + Apoint;
}
```

I work 2 variables. Bpoint which contains everything before the point. We simply get this value by asking for the first String of our HEXF. Then we safe every digit after the point in an char array. Then we Add every iteration of the array our hex_Value(A=10,B=11…) in the right decimal place. At the end we just need to Add A to our Bpoint.

The second method converts from HEXFtoDEC

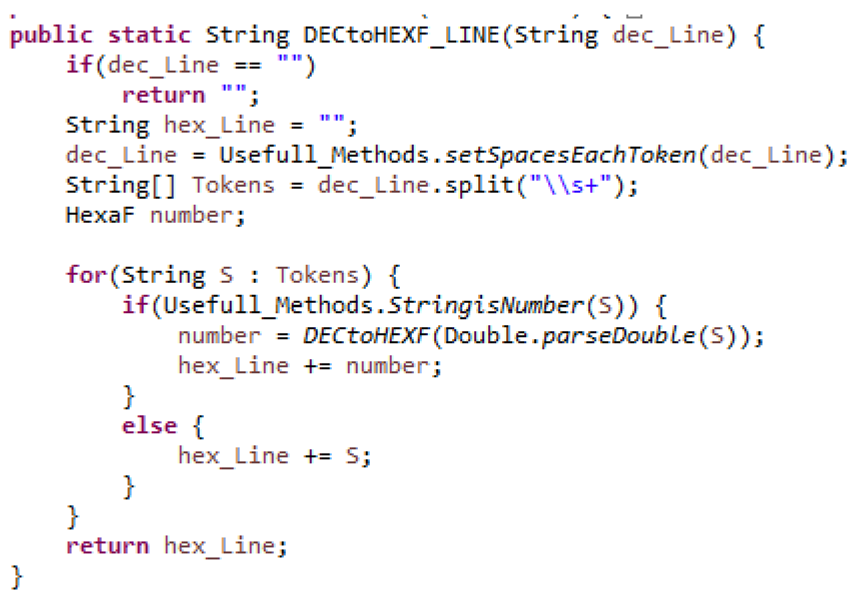


```
public static HexaF DECtoHEXF(double DEC) {

    Hexa Bpoint = Hexa.DECtoHEX((int) DEC);
    DEC -= (int) DEC;
    String Apoint = ".";
    int digit;

    for(int i = 0; i < 5; i++) {
        DEC *= 16;
        digit = (int) DEC;
        DEC -= digit;
        Apoint += Hexa.DECtoHEX(digit);
        if(DEC == 0)
            break;
    }
    HexaF output = new HexaF(Bpoint + Apoint);
    return output;
}
```

Here we do the same with the Bpoint and use the work from 2 weeks ago. For the Apoints digits we round to a maximum of 6 digits(i<5).

We use the algorithm that we multiplay every time with the Base(S1)

Now cames the most complicated one. A whole Line that we can to convert:

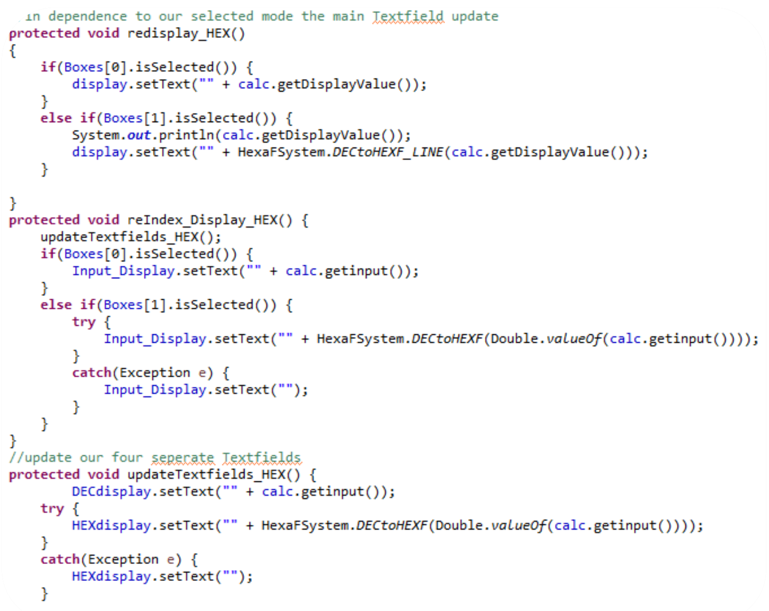


```java
public static String DECtoHEXF_LINE(String dec_Line) {
    if(dec_Line == "")
        return "";
    String hex_Line = "";
    dec_Line = Usefull_Methods.setSpacesEachToken(dec_Line);
    String[] Tokens = dec_Line.split("\\s+");
    HexaF number;

    for(String S : Tokens) {
        if(Usefull_Methods.StringisNumber(S)) {
            number = DECtoHEXF(Double.parseDouble(S));
            hex_Line += number;
        }
        else {
            hex_Line += S;
        }
    }
    return hex_Line;
}
```

Here we used our `setSpacesEachToken(String S)`method and then we split if to get a String Array in each container a Token. Now if the Token is an Number we convert it to HEXF and add it to the new Line. If its not a number we just add the operator to the Line.

Now it finally works with hexadecimal numbers. Even for multidigit numbers and floating point number all at the same moment Now we realised we did the 7th assignment. But in 7 we will Test all this methods together. But we are not finished yet. We need to adjust our Interfaces, more specific our `actionPerformed(ActionEvent event)` method. We tried to make it as good as we can to have a nice overview. At the start it was a mess and it costs more than pure 30 hours and debugging. To get everything to work as we wanted. We also realised that we now can calculate floating point number but actually not putting them in. In 7 we explained how we managed this.

To update every Text field we used the methods from our numeric interfaces:



```java
 in dependence to our selected mode the main Textfield update
protected void redisplay_HEX()
{
    if(Boxes[0].isSelected()) {
        display.setText("" + calc.getDisplayValue());
    }
    else if(Boxes[1].isSelected()) {
        System.out.println(calc.getDisplayValue());
        display.setText("" + HexaFSystem.DECtoHEXF_LINE(calc.getDisplayValue()));
    }

}
protected void reIndex_Display_HEX() {
    updateTextfields_HEX();
    if(Boxes[0].isSelected()) {
        Input_Display.setText("" + calc.getinput());
    }
    else if(Boxes[1].isSelected()) {
        try {
            Input_Display.setText("" + HexaFSystem.DECtoHEXF(Double.valueOf(calc.getinput())));
        }
        catch(Exception e) {
            Input_Display.setText("");
        }
    }
}
//update our four seperate Textfields
protected void updateTextfields_HEX() {
        DECdisplay.setText("" + calc.getinput());
    try {
        HEXdisplay.setText("" + HexaFSystem.DECtoHEXF(Double.valueOf(calc.getinput())));
    }
    catch(Exception e) {
        HEXdisplay.setText("");
    }
```

6. **Now include the exponentiation operator ^ (2^4 = 16). It has a higher priority than either multiplication or division.**

Well time for something not that stressful. We added a button for our “^” and handled it like an operator. It worked perfect because our clean prework. In 7 we prove that its working in the Okta-numeric-system.

7. **Extend your integer calculator to do multidigit and doubles.**

So multi digits already worked to we can chop off this. But we still need to input somehow a point which act like a number and don’t trigger our `ADDtoDisplayValue(calc.getinput());` because our input isn’t finished now. So in order to so see we created a field that we call exp. Exp is safing the information if we are in the Apoint-area and how far we are in. With negative numbers we can represent each digit even in each numeric system. So after pre pressed we went down from 0 to -1 and from here on we counting every time down if we press another number. After we put in our Input to our DisplayValue we reset it so 0. To do so we need to change our `AddNumber(String command)` method that it now acts normal if we are before or after the point. If we are after the point we need to get the whole input. Then convert it to an double and add the pressed number at the right position by multiply it with the exp power of our Base:

```
    protected void AddNumber(String command) {
      if(exp == 0) {
             calc.ADDtoinput((Hexa.HEXToDEC(new Hexa(command))), Base);
      }
      else {
             int digit = Hexa.HEXToDEC(new Hexa(command));
             double input = Double.valueOf(calc.getinput());
             double ADDNumber = digit;
             ADDNumber = ADDNumber*Math.pow(Base, exp);
             calc.setinput(input + ADDNumber);
      }
}
```

Now we only need to handle our exp right in the `actionPerformed (ActionEvent event)` method. We needed to change a few things here and there.

```
    public void actionPerformed(ActionEvent event){

       String command = event.getActionCommand();
       boolean BaseChange = false;

       for(JCheckBox E : Boxes)
               if(E.getText().equals(command))
                       BaseChange = true;
       if(BaseChange) {
               setBase_HEX();
               GreyButtons_HEX();
              redisplay_HEX();
       }
         //Operator!
       else if(Usefull_Methods.CharisOperator(command.charAt(command.length()-1)))
{
               exp = 0;
               calc.ADDtoDisplayValue(calc.getinput());
               // for "="
               if(command.contentEquals("=")) {
                       try {
                                       calc.calcResult();
                               } catch (StackUnderflow | StackException e) {
                                       // TODO Auto-generated catch block
                                       e.printStackTrace();
                               }
                       calc.setinput(Double.valueOf(calc.result));
                       redisplay_HEX();
                       calc.setDisplayValue(calc.result);
                       reIndex_Display_HEX();
                       calc.setinput(0);
               }
               //Everything else
               else {
                       calc.ADDtoDisplayValue(command);
                       calc.setinput(0);
                       redisplay_HEX();
               }
         }
       //Number
         else if(Usefull_Methods.StringisNumber(command)){
               AddNumber(command);
               if(exp<0)
                       exp--;
         }
       else if(command.contentEquals(".")) {
               exp = -1;
       }
         else if(command.equals("Clear")) {
               exp = 0;
              calc.clear();
              redisplay_HEX();
         }
         else if(command.equals("?")) {
              showInfo();
         }
         if(!command.contentEquals("=")) {
               reIndex_Display_HEX();
         }
    }
```

The `actionPerformed(ActionEvent event)` methods in the higher UserInterfaces looks exactly the same. Only that we use _BIN or _OKT update methods in which the methods from the subclasses with the same name gets called. We only add there the extension that is needed to deal with the type of this class to. Which look like this:

```
protected void reIndex_Display_OKT() {
        updateTextfields_OKT();
        reIndex_Display_BIN();

        if(Boxes[3].isSelected()) {
                try {
                        Input_Display.setText("" +
                        OktaFSystem.DECtoOKTF(Double.valueOf(calc.getinput())));
                }
                catch(Exception e) {
                Input_Display.setText("");
                }
        }
}
```

Now we tested everything. Yeah…everything. To do so we got a nearly 250 rows long Test-class. But the most spaces got lost to the simulation of pressing a button. That's why we exclude them into methods that we can better see it something doesn't work:

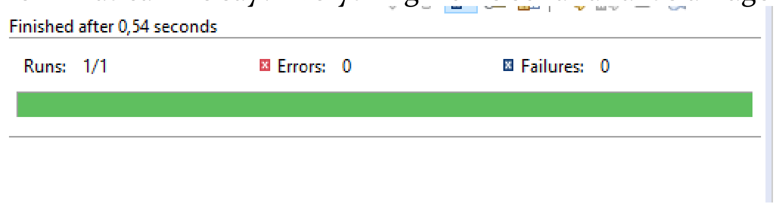
First the simplest one:

```
Test = new Calculator(4);
        //Button

        //Numbers from 1-9
        event = new ActionEvent(Test.gui, 0, "1");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "2");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "3");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "4");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "5");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "6");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "7");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "8");
                Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "9");
                Test.gui.actionPerformed(event);

        //Now all of them should be in the engine.input field.
        assertEquals(123456789, Double.valueOf(Test.engine.getinput()));

        //Now if we press "=" they should be in DisplayCalue before its
        pressed Display is empty
        assertEquals("", Test.engine.getDisplayValue());
        event = new ActionEvent(Test.gui, 0, "=");
        Test.gui.actionPerformed(event);
        assertEquals(123456789,
                Double.valueOf(Test.engine.getDisplayValue()));
        //Now we clear it
        event = new ActionEvent(Test.gui, 0, "Clear");
        Test.gui.actionPerformed(event);
        assertEquals("", Test.engine.getDisplayValue());
```

Time to get to the hard real Tests:

```
//lets now make a really complicated calculation like: -4.2*-1*(5.35+6.789) = 50.9838.
              //Picture
              Test = DECComplicatedCalc(Test);
              assertEquals(Test.engine.result, "50.9838");

              //now lets see if we can work with this result: i want -10 so we need -
60.9838.
              Test = DEC_negativ(Test);
              //u know Java stupid round things -9.999999999999993 PICTURE
              double result =
Math.round(Double.valueOf(Test.engine.result)*1000000000)/1000000000;
              assertEquals(-10, result);

              //Lets do some simple HEXF BINF and OKTF calculations
              event = new ActionEvent(Test.gui, 0, "Clear");
              Test.gui.actionPerformed(event);

              //HEX -A.A*-1*(B.CD+E.FAB) = 11C.89AE
              //First lets simulate a BOX selection
              Test.gui.Boxes[1].setSelected(true);
              event = new ActionEvent(Test.gui, 0, "HEX");
              Test.gui.actionPerformed(event);
              assertEquals(16, Test.gui.Base);
              //now the Calculation
              Test = HEXComplicatedCalc(Test);
              HexaF number_HEX = HexaFSystem.DECtoHEXF(Double.valueOf(Test.engine.result));
              assertEquals("11C.89AE", number_HEX.toString());

              event = new ActionEvent(Test.gui, 0, "Clear");
              Test.gui.actionPerformed(event);

              //At least lets Test BIN cos we cant really Test OKT
              //-4.5*-1*(5.75+6.25) = 54
              //-100.1*-1*(101.11+110.01) = 110110.0
              Test.gui.Boxes[2].setSelected(true);
              event = new ActionEvent(Test.gui, 0, "BIN");
              Test.gui.actionPerformed(event);
              assertEquals(2, Test.gui.Base);
              //now the Calculation
              Test = BINComplicatedCalc(Test);
              BinF number_BIN = BinFSystem.DECtoBINF(Double.valueOf(Test.engine.result));
              System.out.println(number_BIN);
              assertEquals("110110.0", number_BIN.toString());

              event = new ActionEvent(Test.gui, 0, "Clear");
              Test.gui.actionPerformed(event);

              //For Assignment 6 and with Okta:
              //1.5*2^2 = 6
              Test.gui.Boxes[3].setSelected(true);
              event = new ActionEvent(Test.gui, 0, "OKT");
              Test.gui.actionPerformed(event);
              assertEquals(8, Test.gui.Base);
              //now the Calculation
              Test = OKTComplicatedCalc(Test);
              OktaF number_OKT = OktaFSystem.DECtoOKTF(Double.valueOf(Test.engine.result));
              System.out.println(number_OKT);
              assertEquals("6", number_OKT.toString());
```

Well what can we say? Everything works so far and it’s a huge relief.



Finished after 0,54 seconds

Runs: 1/1    Errors: 0    Failures: 0

## Reflections

### FATIMA

This week's lab, was interesting because we could use methods and other stuff we had programmed in the last weeks, so it was like code-recycling. I really liked the whole Calculator-thing. During the Lab arisen some questions for me, which Juri had fortunately some answers for.  So I could learn/repeat some stuff that was a bit unclear for me. We finished the tasks 1-3 in the Lab, the rest Juri did at home.

### JURI

Like we said at the beginning this week was a really stressful one. Not only I pushed the limit of what we do in this time, I also had to deal with some private stuff which costs a lot of quality. Normally I would have tried to implement other stuff like. If we put an –(Stuff) it automatically buts an 1 between it for the Postfix class. Also I m still not satisfied with the negative number. In the numeric systems they looks really weird like 11111111101. I know why but it would be much nicer if we just would put an – in front of the String. Also I still don't know how we can combine Button input with the input directly in the Text fields. Because of Fields like the exp we can't just write into it.

Source:

S1: https://www.youtube.com/watch?v=KNqB4-gRbho

Appendix:

```
Calculator creation/handeling and Interfaces:
        Calculator
        Create_Calc
        UserInterface
        UserInterface_HEX
        UserInterface_HEX_BIN
        UserInterface_HEX_BIN_OKT
Numeric systems
        Bin
        BinF
        BinFSystem
        BinSystem
        Hexa
        HexaF
        HexaFSystem
        HexaSystem
        Okta
        OktaF
        OktaFSystem
        OktaSystem
Postfix-Stack
        Postfix
        Stack
        StackAsList
        StackException
        StackUnderflow
Ubung7
        CalcEngine_V2
        Postfix_Multidigit
        Ubung7_TEST
        Usefull_Methods
```

```java
package Calculator;


public class Create_Calc {
	public static Calculator Test;
	public static void main(String[] args) {
		//Test = new Calculator(1);
		//Test = new Calculator(2);
		//Test = new Calculator(3);
		Test = new Calculator(4);
	}
}
package Calculator;

import Postfix_Stack.Postfix;
import ubung7.CalcEngine_V2;

/**
 * The main class of a simple calculator. Create one of these and
you'll
 * get the calculator on screen.
 *
 * @author David J. Barnes and Michael Kolling
 * @version 2008.03.30
 */
public class Calculator
{
    public CalcEngine_V2 engine;
    public UserInterface gui;

    /**
     * Create a new calculator and show it.
     * @param Set the Numeric System
     * mode == 1: Basic Calc
     * mode == 2: HexCalc
     * mode == 3: BIN_HEX Calc
     * mode == 4: OKT_BIN_HEX Calc
     */
    public Calculator(int mode)
    {
        engine = new CalcEngine_V2();
        if(mode == 1) gui = new UserInterface(engine);
        if(mode == 2) gui = new UserInterface_HEX(engine, mode);
        if(mode == 3) gui = new UserInterface_HEX_BIN(engine,
mode);
        if(mode == 4) gui = new UserInterface_HEX_BIN_OKT(engine,
mode);
```

```
    }

    /**
     * In case the window was closed, show it again.
     */
    public void show()
    {
        gui.setVisible(true);
    }
    public void setEngine(CalcEngine_V2 engine) {
      this.engine = engine;
    }
}

package Calculator;


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

import Postfix_Stack.Postfix;
import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.Hexa;
import ubung7.CalcEngine_V2;
import ubung7.Usefull_Methods;

/**
 * A graphical user interface for the calculator. No calculation is
being
 * done here. This class is responsible just for putting up the
display on
 * screen. It then refers to the "CalcEngine" to do all the real
work.
 *
 * @author David J. Barnes and Michael Kolling
 * @version 2008.03.30
 */
public class UserInterface
    implements ActionListener
{
      public JCheckBox[] Boxes;
    protected CalcEngine_V2 calc;
    protected boolean showingAuthor;
    protected JFrame frame;
    protected JTextField display;
    protected JTextField Input_Display;
    protected JLabel status;

    protected JPanel buttonPanel;
    protected JPanel Northside;
    protected JPanel contentPane;
    protected int exp = 0;
    public int Base = 10;
    /**
     * Create a user interface.
     * @param engine The calculator engine.
     */
    public UserInterface(CalcEngine_V2 engine)
    {
        calc = engine;
        showingAuthor = true;
        makeFrame();
        frame.setVisible(true);
    }

    /**
     * Set the visibility of the interface.
     * @param visible true if the interface is to be made visible,
false otherwise.
     */
    public void setVisible(boolean visible)
    {
        frame.setVisible(visible);
    }

    /**
     * Make the frame for the user interface.
     */
    private void makeFrame()
    {
        frame = new JFrame(calc.getTitle());

        contentPane = (JPanel)frame.getContentPane();
        contentPane.setLayout(new BorderLayout(8, 8));
        contentPane.setBorder(new EmptyBorder( 10, 10, 10, 10));

        display = new JTextField("0");
        Input_Display = new JTextField("0");
        Northside = new JPanel(new GridLayout(3, 1));

        Northside.add(display);
        Northside.add(Input_Display);

        contentPane.add(Northside, BorderLayout.NORTH);

        buttonPanel = new JPanel(new GridLayout(6, 4));

            addButton(buttonPanel, "(");
            addButton(buttonPanel, "Clear");
            addButton(buttonPanel, ")");
            addButton(buttonPanel, "/");


            addButton(buttonPanel, "7");
            addButton(buttonPanel, "8");
            addButton(buttonPanel, "9");
            addButton(buttonPanel, "x");


            addButton(buttonPanel, "4");
            addButton(buttonPanel, "5");
            addButton(buttonPanel, "6");
            addButton(buttonPanel, "-");


            addButton(buttonPanel, "1");
            addButton(buttonPanel, "2");
            addButton(buttonPanel, "3");
            addButton(buttonPanel, "+");

            addButton(buttonPanel, ".");
            addButton(buttonPanel, "0");
            addButton(buttonPanel, "?");
            addButton(buttonPanel, "=");
            addButton(buttonPanel, "^");


        contentPane.add(buttonPanel, BorderLayout.CENTER);

        status = new JLabel(calc.getAuthor());
        contentPane.add(status, BorderLayout.SOUTH);

        frame.pack();
    }
```

```java
    /**
     * Add a button to the button panel.
     * @param panel The panel to receive the button.
     * @param buttonText The text for the button.
     */
    protected void addButton(Container panel, String buttonText)
    {
        JButton button = new JButton(buttonText);
        button.addActionListener(this);
        panel.add(button);
    }
    protected void addCheckbox(Container panel, JCheckBox JCB) {
      panel.add(JCB);
    }

    /**
     * An interface action has been performed.
     * Find out what it was and handle it.
     * @param event The event that has occured.
     */
    public void actionPerformed (ActionEvent event)
    {
        String command = event.getActionCommand();
        //Operator!
        if(Usefull_Methods.CharisOperator(command.charAt(command.le
ngth()-1))) {
            exp = 0;
            calc.ADDtoDisplayValue(calc.getinput());
            // for "="
            if(command.contentEquals("=")) {
                  try {
                              calc.calcResult();
                        } catch (StackUnderflow | StackException e)
{
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  calc.setinput(Double.valueOf(calc.result));
                  redisplay();
                  calc.setDisplayValue(calc.result);
                  reIndex_Display();
                  calc.setinput(0);
            }
            //Everything else
            else {
                  calc.ADDtoDisplayValue(command);
                  calc.setinput(0);
```

```
                redisplay();
            }

        }
      //Number
        else if(Usefull_Methods.StringisNumber(command)){
            AddNumber(command);
            if(exp<0)
                exp--;
        }
      else if(command.contentEquals(".")) {
            exp = -1;
      }
        else if(command.equals("Clear")) {
            exp = 0;
            calc.clear();
            redisplay();

        }
        else if(command.equals("?")) {
            showInfo();
        }
        if(!command.contentEquals("=")) {
            reIndex_Display();
        }
    }
    /**
     * Update the interface display to show the current value of
the
     * calculator.
     */
    protected void redisplay()
    {
        display.setText("" + calc.getDisplayValue());
    }
    protected void reIndex_Display() {
        Input_Display.setText("" + calc.getinput());

    }

    /**
     * Toggle the info display in the calculator's status area
between the
     * author and version information.
     */
    protected void showInfo()
    {
        if(showingAuthor)
            status.setText(calc.getVersion());
        else
            status.setText(calc.getAuthor());

        showingAuthor = !showingAuthor;
    }
    protected void AddNumber(String command) {
      if(exp == 0) {
            calc.ADDtoinput((Hexa.HEXToDEC(new Hexa(command))),
Base);
      }
      else {
            int digit = Hexa.HEXToDEC(new Hexa(command));
            double input = Double.valueOf(calc.getinput());
            double ADDNumber = digit;
            ADDNumber = ADDNumber*Math.pow(Base, exp);
            calc.setinput(input + ADDNumber);
      }
      }
}
package Calculator;


import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Enumeration;

import javax.swing.AbstractButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.Hexa;
import numeric_Systems.HexaF;
import numeric_Systems.HexaFSystem;
import ubung7.CalcEngine_V2;
import ubung7.Usefull_Methods;
```

```java
/**
 * A graphical user interface for the calculator. No calculation is
being
 * done here. This class is responsible just for putting up the
display on
 * screen. It then refers to the "CalcEngine" to do all the real
work.
 *
 * @author Juri Wiechmann
 * @version 2019.11.25
 */
public class UserInterface_HEX extends UserInterface
    implements ActionListener
{
    protected ButtonGroup CheckBoxGrp = new ButtonGroup();

    private JPanel HEX_BOX;

    JPanel Checkboxes = new JPanel(new GridLayout(4, 2));

    JTextField DECdisplay;
    JTextField HEXdisplay;

    protected ButtonGroup DECButtons = new ButtonGroup();
    protected ButtonGroup HEXButtons = new ButtonGroup();


    Component[] comp;

      public UserInterface_HEX(CalcEngine_V2 engine, int
Amount_Checkboxes) {
            super(engine);
            this.Boxes = new JCheckBox[Amount_Checkboxes];
            addLayout_HEX();
      }
      public void addLayout_HEX() {

        Boxes[0] = new JCheckBox("DEC",true);
        Boxes[1] = new JCheckBox("HEX");

        Boxes[0].addActionListener(this);
        Boxes[1].addActionListener(this);

        DECdisplay = new JTextField("0");
        HEXdisplay = new JTextField("0");
        addCheckbox(Checkboxes, Boxes[0]);
        Checkboxes.add(DECdisplay);
        addCheckbox(Checkboxes, Boxes[1]);
        Checkboxes.add(HEXdisplay);

        CheckBoxGrp.add(Boxes[0]);
        CheckBoxGrp.add(Boxes[1]);

        Northside.add(Checkboxes);

        Font font1 = new Font("SansSerif", Font.BOLD, 20);
        Input_Display.setFont(font1);


        contentPane.add(Northside, BorderLayout.NORTH);

        HEX_BOX = new JPanel(new GridLayout(5,2));
            buttonPanel.add(HEX_BOX);

            HEX_BOX.add(new JLabel(" "));
            HEX_BOX.add(new JLabel(" "));
            addButton(HEX_BOX, "A");
        addButton(HEX_BOX, "B");
        addButton(HEX_BOX, "C");
        addButton(HEX_BOX, "D");
        addButton(HEX_BOX, "E");
        addButton(HEX_BOX, "F");
        HEX_BOX.add(new JLabel(" "));
        HEX_BOX.add(new JLabel(" "));

        contentPane.add(HEX_BOX, BorderLayout.WEST);
        frame.pack();

        assignButtonGrps_HEX();
        GreyButtons_HEX();
      }
    public void actionPerformed(ActionEvent event){

      String command = event.getActionCommand();
      boolean BaseChange = false;

      for(JCheckBox E : Boxes)
            if(E.getText().equals(command))
                  BaseChange = true;

      if(BaseChange) {
            setBase_HEX();
```

```
			GreyButtons_HEX();
			redisplay_HEX();
		}
		  //Operator!
		else
if(Usefull_Methods.CharisOperator(command.charAt(command.length()-
1))) {
			exp = 0;
			calc.ADDtoDisplayValue(calc.getinput());
			// for "="
			if(command.contentEquals("=")) {
				try {
						calc.calcResult();
					} catch (StackUnderflow | StackException e)
{
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
				calc.setinput(Double.valueOf(calc.result));
				redisplay_HEX();
				calc.setDisplayValue(calc.result);
				reIndex_Display_HEX();
				calc.setinput(0);
			}
			//Everything else
			else {
				calc.ADDtoDisplayValue(command);
				calc.setinput(0);
				redisplay_HEX();
			}

		  }
		//Number
		  else if(Usefull_Methods.StringisNumber(command)){
			AddNumber(command);
			if(exp<0)
				exp--;
		  }
		else if(command.contentEquals(".")) {
			exp = -1;
		}
		  else if(command.equals("Clear")) {
			exp = 0;
			calc.clear();
			redisplay_HEX();

		  }
```

```
        else if(command.equals("?")) {
            showInfo();
        }
        if(!command.contentEquals("=")) {
            reIndex_Display_HEX();
        }
    }
    //in dependence to our selected mode the main Textfield update
      protected void redisplay_HEX()
    {
      if(Boxes[0].isSelected()) {
            display.setText("" + calc.getDisplayValue());
      }
      else if(Boxes[1].isSelected()) {
            System.out.println(calc.getDisplayValue());
            display.setText("" +
HexaFSystem.DECtoHEXF_LINE(calc.getDisplayValue()));
      }

    }
    protected void reIndex_Display_HEX() {
      updateTextfields_HEX();
      if(Boxes[0].isSelected()) {
            Input_Display.setText("" + calc.getinput());
      }
      else if(Boxes[1].isSelected()) {
            try {
                  Input_Display.setText("" +
HexaFSystem.DECtoHEXF(Double.valueOf(calc.getinput())));
            }
            catch(Exception e) {
                  Input_Display.setText("");
            }
      }
    }
      //update our four seperate Textfields
    protected void updateTextfields_HEX() {
            DECdisplay.setText("" + calc.getinput());
        try {
            HEXdisplay.setText("" +
HexaFSystem.DECtoHEXF(Double.valueOf(calc.getinput())));
      }
      catch(Exception e) {
            HEXdisplay.setText("");
      }
    }
    //Set the Base based on the selected mode
    protected void setBase_HEX() {

      if(Boxes[0].isSelected()) {
            System.out.println("DEC");
            Base = 10;
      }
      else if(Boxes[1].isSelected()) {
            System.out.println("HEX");
            Base = 16;
      }
      redisplay_HEX();
    }
    /*Disable all Buttons and then enable only these which are in
the
      right ButtonGroup that corresponds to the selection mode
    */
    protected void GreyButtons_HEX() {
      //https://coderanch.com/t/336958/java/disable-buttonGroup
      //https://stackoverflow.com/questions/7160568/iterating-
through-enumeration-of-hastable-keys-throws-nosuchelementexception-
err
      //https://stackoverflow.com/questions/1625855/how-to-disable-
javax-swing-jbutton-in-java

      //Make all Grey
      Enumeration<AbstractButton> HEX = HEXButtons.getElements();
      while(HEX.hasMoreElements()) {
            JButton Button = (JButton) HEX.nextElement();
            Button.setEnabled(false);
      }
      Enumeration<AbstractButton> DEC_theRest =
DECButtons.getElements();
      while(DEC_theRest.hasMoreElements()) {
            JButton Button = (JButton) DEC_theRest.nextElement();
            Button.setEnabled(false);
      }
      //DEC-Mode
      if(Boxes[0].isSelected()) {
            Enumeration<AbstractButton> DEC =
DECButtons.getElements();
            while(DEC.hasMoreElements()) {
                  JButton Button = (JButton) DEC.nextElement();
                  Button.setEnabled(true);
            }
      }
      //HEX-Mode
      else if(Boxes[1].isSelected()) {
            HEX = HEXButtons.getElements();
            while(HEX.hasMoreElements()) {
                  JButton Button = (JButton) HEX.nextElement();
                  Button.setEnabled(true);
            }
            Enumeration<AbstractButton> DEC =
DECButtons.getElements();
            while(DEC.hasMoreElements()) {
                  JButton Button = (JButton) DEC.nextElement();
                  Button.setEnabled(true);
            }
      }
      }
    //assign all the Buttons that are relevant to their ButtonGroup
    protected void assignButtonGrps_HEX() {

      comp = buttonPanel.getComponents();
      //DEC
      for(Component E : comp) {
            if(E instanceof JButton)
                  if(isNumber(((JButton)E).getText()))

      if(isBetween(Integer.parseInt(((JButton)E).getText()), 0, 9))
                        DECButtons.add((JButton)E);
      }
      //HEX
      comp = HEX_BOX.getComponents();
      for(Component E : comp) {
            if(E instanceof JButton) {
                  HEXButtons.add((JButton)E);
            }
      }
    }
    //return true if min <= value <= max
    protected boolean isBetween(int value, int min, int max) {
      if(value >= min && value <= max)
            return true;
      else
            return false;
    }
    //Checks if the String is a single digit Number
    protected boolean isNumber(String value) {
      if( value.equals("0") ||
            value.equals("1") ||
            value.equals("2") ||
            value.equals("3") ||
            value.equals("4") ||
```

```java
            value.equals("5") ||
            value.equals("6") ||
            value.equals("7") ||
            value.equals("8") ||
            value.equals("9") ||
            value.equals("A") ||
            value.equals("B") ||
            value.equals("C") ||
            value.equals("D") ||
            value.equals("E") ||
            value.equals("F"))
                  return true;
            else
                  return false;
    }

}
package Calculator;


import java.awt.Component;
import java.awt.event.ActionEvent;
import java.util.Enumeration;

import javax.swing.AbstractButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTextField;

import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.Bin;
import numeric_Systems.BinFSystem;
import numeric_Systems.Hexa;
import numeric_Systems.HexaFSystem;
import numeric_Systems.OktaFSystem;
import ubung7.CalcEngine_V2;
import ubung7.Usefull_Methods;

public class UserInterface_HEX_BIN extends UserInterface_HEX {

      protected ButtonGroup BINButtons = new ButtonGroup();

      JTextField BINdisplay;

      public UserInterface_HEX_BIN(CalcEngine_V2 engine, int
Amount_Checkboxes) {
            super(engine, Amount_Checkboxes);
            addLayout_BIN();
      }
      private void addLayout_BIN() {

            Boxes[2] = new JCheckBox("BIN");

            Boxes[2].addActionListener(this);

            BINdisplay = new JTextField("0");

            CheckBoxGrp.add(Boxes[2]);

        addCheckbox(Checkboxes, Boxes[2]);
        Checkboxes.add(BINdisplay);

        frame.pack();

        assignButtonGrps_BIN();
        GreyButtons_BIN();
      }
      protected void assignButtonGrps_BIN() {

            assignButtonGrps_HEX();

            comp = buttonPanel.getComponents();

      for(Component E : comp) {
            if(E instanceof JButton)
                  if(isNumber(((JButton)E).getText()))

      if(isBetween(Integer.parseInt(((JButton)E).getText()), 0, 1))
                        BINButtons.add((JButton)E);
      }
      }
      protected void GreyButtons_BIN() {

            GreyButtons_HEX();

            if(Boxes[2].isSelected()) {
            Enumeration<AbstractButton> BIN =
BINButtons.getElements();
            while(BIN.hasMoreElements()) {
                  JButton Button = (JButton) BIN.nextElement();
                  Button.setEnabled(true);
            }
      }
      }
    public void actionPerformed(ActionEvent event){

      String command = event.getActionCommand();
      boolean BaseChange = false;

      for(JCheckBox E : Boxes)
            if(E.getText().equals(command))
                  BaseChange = true;

      if(BaseChange) {
            setBase_BIN();
            GreyButtons_BIN();
            redisplay_BIN();
      }
        //Operator!
      else 
if(Usefull_Methods.CharisOperator(command.charAt(command.length()-
1))) {
            exp = 0;
            calc.ADDtoDisplayValue(calc.getinput());
            // for "="
            if(command.contentEquals("=")) {
                  try {
                              calc.calcResult();
                        } catch (StackUnderflow | StackException e)
{
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  calc.setinput(Double.valueOf(calc.result));
                  redisplay_BIN();
                  calc.setDisplayValue(calc.result);
                  reIndex_Display_BIN();
                  calc.setinput(0);
            }
            //Everything else
            else {
                  calc.ADDtoDisplayValue(command);
                  calc.setinput(0);
                  redisplay_BIN();
            }

        }
      //Number
        else if(Usefull_Methods.StringisNumber(command)){
            AddNumber(command);
            if(exp<0)
                  exp--;
        }
      else if(command.contentEquals(".")) {
            exp = -1;
      }
        else if(command.equals("Clear")) {
            exp = 0;
            calc.clear();
            redisplay_BIN();
        }
        else if(command.equals("?")) {
            showInfo();
        }
        if(!command.contentEquals("=")) {
            reIndex_Display_BIN();
        }
    }
      protected void reIndex_Display_BIN() {
            updateTextfields_BIN();
            reIndex_Display_HEX();

            if(Boxes[2].isSelected()) {
            try {
                  Input_Display.setText("" +
BinFSystem.DECtoBINF(Double.valueOf(calc.getinput())));
            }
            catch(Exception e) {
                  Input_Display.setText("");
            }
            }
      }
      protected void updateTextfields_BIN() {
            updateTextfields_HEX();
            try {
                  BINdisplay.setText("" +
BinFSystem.DECtoBINF(Double.valueOf(calc.getinput())));
            }
            catch(Exception e) {
                  BINdisplay.setText("");
            }
      }
      protected void redisplay_BIN() {
            redisplay_HEX();
            if(Boxes[2].isSelected()) {
            display.setText("" +
BinFSystem.DECtoBINF_LINE(calc.getDisplayValue()));
      }
      }
    protected void setBase_BIN() {
      setBase_HEX();

      if(Boxes[2].isSelected()) {
            System.out.println("BIN");
            Base = 2;
      }

      redisplay_BIN();
    }
}
package Calculator;


import java.awt.Component;
import java.awt.event.ActionEvent;
import java.util.Enumeration;

import javax.swing.AbstractButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTextField;

import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.BinFSystem;
import numeric_Systems.Hexa;
import numeric_Systems.Okta;
import numeric_Systems.OktaFSystem;
import ubung7.CalcEngine_V2;
import ubung7.Usefull_Methods;

public class UserInterface_HEX_BIN_OKT extends
UserInterface_HEX_BIN {

      protected ButtonGroup OKTButtons = new ButtonGroup();

      JTextField OKTdisplay;

      public UserInterface_HEX_BIN_OKT(CalcEngine_V2 engine, int
Amount_Checkboxes) {
            super(engine, Amount_Checkboxes);
            addLayout_OKT();
      }
      private void addLayout_OKT() {

            Boxes[3] = new JCheckBox("OKT");

            Boxes[3].addActionListener(this);

            OKTdisplay = new JTextField("0");

            CheckBoxGrp.add(Boxes[3]);

        addCheckbox(Checkboxes, Boxes[3]);
        Checkboxes.add(OKTdisplay);

        frame.pack();

        assignButtonGrps_OKT();
        GreyButtons_OKT();
      }
      protected void assignButtonGrps_OKT() {

            assignButtonGrps_BIN();

            comp = buttonPanel.getComponents();

      for(Component E : comp) {
            if(E instanceof JButton)
                  if(isNumber(((JButton)E).getText()))

      if(isBetween(Integer.parseInt(((JButton)E).getText()), 0, 7))
                        OKTButtons.add((JButton)E);
      }
      }
      protected void GreyButtons_OKT() {

            GreyButtons_BIN();

            if(Boxes[3].isSelected()) {
            Enumeration<AbstractButton> OKT =
OKTButtons.getElements();
            while(OKT.hasMoreElements()) {
                  JButton Button = (JButton) OKT.nextElement();
                  Button.setEnabled(true);
            }
      }
      }
    public void actionPerformed(ActionEvent event){
      String command = event.getActionCommand();
      System.out.println(command);
      System.out.println(calc.getDisplayValue());

      boolean BaseChange = false;

      for(JCheckBox E : Boxes)
            if(E.getText().equals(command))
                  BaseChange = true;

      if(BaseChange) {
            setBase_OKT();
            GreyButtons_OKT();
            redisplay_OKT();
      }
        //Operator!
      else
if(Usefull_Methods.CharisOperator(command.charAt(command.length()-
1))) {
            exp = 0;
            calc.ADDtoDisplayValue(calc.getinput());
            // for "="
            if(command.contentEquals("=")) {
                  try {
                              calc.calcResult();
                        } catch (StackUnderflow | StackException e)
{
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                  calc.setinput(Double.valueOf(calc.result));
                  redisplay_OKT();
                  calc.setDisplayValue(calc.result);
                  reIndex_Display_OKT();
                  calc.setinput(0);
            }
            //Everything else
            else {
                  calc.ADDtoDisplayValue(command);
                  calc.setinput(0);
                  redisplay_OKT();
            }

        }
      //Number
        else if(Usefull_Methods.StringisNumber(command)){
            AddNumber(command);
            if(exp<0)
                  exp--;
        }
      else if(command.contentEquals(".")) {
            exp = -1;
      }
        else if(command.equals("Clear")) {
            exp = 0;
            calc.clear();
            redisplay_OKT();
        }
        else if(command.equals("?")) {
            showInfo();
        }
        if(!command.contentEquals("=")) {
            reIndex_Display_OKT();
        }
    }
      private void setBase_OKT() {
            setBase_BIN();

      if(Boxes[3].isSelected()) {
            System.out.println("OKT");
            Base = 8;
      }
      redisplay_OKT();
      }
      private void updateTextfields_OKT() {
            updateTextfields_BIN();

            try {
                  OKTdisplay.setText("" +
OktaFSystem.DECtoOKTF(Double.valueOf(calc.getinput())));
            }
            catch(Exception e) {
                  OKTdisplay.setText("");
            }     }
      private void redisplay_OKT() {
            redisplay_BIN();
            if(Boxes[3].isSelected()) {
            display.setText("" +
OktaFSystem.DECtoOKTF_LINE(calc.getDisplayValue()));
      }
      }
      protected void reIndex_Display_OKT() {
            updateTextfields_OKT();
            reIndex_Display_BIN();

            if(Boxes[3].isSelected()) {
            try {
                  Input_Display.setText("" +
OktaFSystem.DECtoOKTF(Double.valueOf(calc.getinput())));
            }
            catch(Exception e) {
                  Input_Display.setText("");
            }
            }
      }
}


 package numeric_Systems;

 public class Bin implements BinSystem {

      String value;

      public Bin(String value) {
            setValue(value);
      }
      @Override
      public Bin Add(Bin bin) {
            int a = Bin.BINToDEC(this);
            int b = Bin.BINToDEC(bin);

            return Bin.DECtoBIN(a+b);
      }
      @Override
      public Bin Subtract(Bin bin) throws Exception {
            int a = Bin.BINToDEC(this);
            int b = Bin.BINToDEC(bin);
            if(a<b) throw new Exception("B is bigger then A");

            return Bin.DECtoBIN(a-b);
      }
      @Override
      public Bin Multiplycation(Bin bin) {
            int a = Bin.BINToDEC(this);
            int b = Bin.BINToDEC(bin);

            return Bin.DECtoBIN(a*b);
      }
	@Override
	public Bin Division(Bin bin) {
		int a = Bin.BINToDEC(this);
		int b = Bin.BINToDEC(bin);

		return Bin.DECtoBIN(a/b);
	}
	public static int BINToDEC(Bin bin) {
		return Integer.parseInt(bin.getValue(), 2);
	}
	public static Bin DECtoBIN(int DEC) {
		return new Bin(Integer.toBinaryString(DEC));
	}
	public String toString() {
		return value;
	}
	public String getValue() {
		return value;
	}
	public void setValue(String value) {
		this.value = value;
	}
}
package numeric_Systems;

public class BinF implements BinFSystem {
	private String [] value;
	public String[] getValue() {
		return value;
	}

	public BinF(String value) {
		this.value = value.split("\\.");
		//If .x is missing.
		if(this.value.length==1) {
			String[] temp = new String[2];
			temp[0] = this.value[0];
			temp[1] = "0";
			this.value = temp;
		}
	}
	@Override
	public BinF Add(BinF BINF) {

		double a = BinFSystem.BINFToDEC(this);
		double b = BinFSystem.BINFToDEC(BINF);
```

```
			return BinFSystem.DECtoBINF(a+b);
		}
		@Override
		public BinF Subtract(BinF BINF) {

			double a = BinFSystem.BINFToDEC(this);
			double b = BinFSystem.BINFToDEC(BINF);

			return BinFSystem.DECtoBINF(a-b);
		}
		@Override
		public BinF Multiplication(BinF BINF) {

			double a = BinFSystem.BINFToDEC(this);
			double b = BinFSystem.BINFToDEC(BINF);

			return BinFSystem.DECtoBINF(a*b);
		}
		@Override
		public BinF Division(BinF BINF) {

			double a = BinFSystem.BINFToDEC(this);
			double b = BinFSystem.BINFToDEC(BINF);

			return BinFSystem.DECtoBINF(a/b);
		}
		public BinF Pow(BinF BINF) {

			double a = BinFSystem.BINFToDEC(this);
			double b = BinFSystem.BINFToDEC(BINF);

			return BinFSystem.DECtoBINF(Math.pow(a, b));
		}
		@Override
		public String toString() {
			return value[0] + "." + value[1];
		}
	}
	package numeric_Systems;

	import ubung7.Usefull_Methods;

	public interface BinFSystem {
		public static double BINFToDEC(BinF BINF) {
```

```
            int Bpoint = Bin.BINToDEC(new 
Bin(BINF.getValue()[0]));


            char[] c_Apoint = BINF.getValue()[1].toCharArray();
            double Apoint = 0;

            for(int i = 0 ; i< c_Apoint.length;i++) {
                  int bin_Value = Bin.BINToDEC(new Bin
(Character.toString(c_Apoint[i])));
                  Apoint +=  bin_Value*Math.pow(2, -(i+1));
            }
            return Bpoint + Apoint;
      }
      public static BinF DECtoBINF(double DEC) {

            Bin Bpoint = Bin.DECtoBIN((int) DEC);
            DEC -= (int) DEC;
            String Apoint = ".";
            int digit;

            for(int i = 0; i < 5; i++) {
                  DEC *= 2;
                  digit = (int) DEC;
                  DEC -= digit;
                  Apoint += Bin.DECtoBIN(digit);
                  if(DEC == 0)
                        break;
            }
            BinF output = new BinF(Bpoint + Apoint);
            return output;
      }
      public static String DECtoBINF_LINE(String dec_Line) {
            if(dec_Line == "")
                  return "";
            String bin_Line = "";
            dec_Line = 
Usefull_Methods.setSpacesEachToken(dec_Line);
            String[] Tokens = dec_Line.split("\\s+");
            BinF number;

            for(String S : Tokens) {
                  if(Usefull_Methods.StringisNumber(S)) {
                        number = 
DECtoBINF(Double.parseDouble(S));
                        bin_Line += number;
                  }
                  else {
                        bin_Line += S;
                  }
            }
            return bin_Line;
      }
      public BinF Add(BinF BINF);
      public BinF Subtract(BinF BINF);
      public BinF Multiplication(BinF BINF);
      public BinF Division(BinF BINF);
      public BinF Pow(BinF BINF);
      public String toString();
}
package numeric_Systems;

public interface BinSystem {

      public Bin Add(Bin bin);
      public Bin Subtract(Bin bin) throws Exception;
      public Bin Multiplycation(Bin bin);
      public Bin Division(Bin bin);
      public String toString();
}
package numeric_Systems;


public class Hexa implements HexaSystem {
      private String value;

      public Hexa(String value) {
            setValue(value);
      }
      public String getValue() {
            return value;
      }
      public void setValue (String value) {

            char[] Cvalue = value.toCharArray();
            for(char E : Cvalue) {
                  if((E < 48 && E > 57) || (E < 65 && E > 70)) {
                        throw new IllegalArgumentException("only
Chars from 1-9 and A-F");
                  }
            }
            this.value = value;
```

```
	}
	@Override
	public Hexa Add(Hexa HEX) {

		int a = Hexa.HEXToDEC(this);
		int b = Hexa.HEXToDEC(HEX);

		return Hexa.DECtoHEX(a+b);
	}

	@Override
	public Hexa Subtract(Hexa HEX) {
		int a = Hexa.HEXToDEC(this);
		int b = Hexa.HEXToDEC(HEX);

		return Hexa.DECtoHEX(a-b);
	}

	@Override
	public Hexa Multiplycation(Hexa HEX) {
		int a = Hexa.HEXToDEC(this);
		int b = Hexa.HEXToDEC(HEX);

		return Hexa.DECtoHEX(a*b);
	}

	@Override
	public Hexa Division(Hexa HEX) {
		int a = Hexa.HEXToDEC(this);
		int b = Hexa.HEXToDEC(HEX);

		return Hexa.DECtoHEX(a/b);
	}
	public static int HEXToDEC(Hexa HEX) {
		return Integer.parseInt(HEX.getValue(), 16);
	}
	public static Hexa DECtoHEX(int DEC) {
		return new
Hexa(Integer.toHexString(DEC).toUpperCase());
	}
	public String toString() {
		return value;
	}
}
package numeric_Systems;
```

```
import ubung7.Usefull_Methods;

public class HexaF implements HexaFSystem{

	private String [] value;
	public String[] getValue() {
		return value;
	}

	public HexaF(String value) {
		this.value = value.split("\\.");
		//If .x is missing.
		if(this.value.length==1) {
			String[] temp = new String[2];
			temp[0] = this.value[0];
			temp[1] = "0";
			this.value = temp;
		}
	}
	public HexaF Add(HexaF HEXF) {

		double a = HexaFSystem.HEXFToDEC(this);
		double b = HexaFSystem.HEXFToDEC(HEXF);

		return HexaFSystem.DECtoHEXF(a+b);
	}
	public HexaF Subtract(HexaF HEXF) {

		double a = HexaFSystem.HEXFToDEC(this);
		double b = HexaFSystem.HEXFToDEC(HEXF);

		return HexaFSystem.DECtoHEXF(a-b);
	}
	public HexaF Multiplication(HexaF HEXF) {

		double a = HexaFSystem.HEXFToDEC(this);
		double b = HexaFSystem.HEXFToDEC(HEXF);

		return HexaFSystem.DECtoHEXF(a*b);
	}
	public HexaF Division(HexaF HEXF) {

		double a = HexaFSystem.HEXFToDEC(this);
		double b = HexaFSystem.HEXFToDEC(HEXF);

		return HexaFSystem.DECtoHEXF(a/b);
```

```
        }
        public HexaF Pow(HexaF HEXF) {

                double a = HexaFSystem.HEXFToDEC(this);
                double b = HexaFSystem.HEXFToDEC(HEXF);

                return HexaFSystem.DECtoHEXF(Math.pow(a, b));
        }
        public String toString() {
                return value[0] + "." + value[1];
        }
}
package numeric_Systems;

import ubung7.Usefull_Methods;

public interface HexaFSystem {

        public static double HEXFToDEC(HexaF HEXF) {

                int Bpoint = Hexa.HEXToDEC(new
Hexa(HEXF.getValue()[0]));


                char[] c_Apoint = HEXF.getValue()[1].toCharArray();
                double Apoint = 0;

                for(int i = 0 ; i< c_Apoint.length;i++) {
                        int hex_Value = Hexa.HEXToDEC(new Hexa
(Character.toString(c_Apoint[i])));
                        Apoint +=  hex_Value*Math.pow(16, -(i+1));
                }
                return Bpoint + Apoint;
        }
        public static HexaF DECtoHEXF(double DEC) {

                Hexa Bpoint = Hexa.DECtoHEX((int) DEC);
                DEC -= (int) DEC;
                String Apoint = ".";
                int digit;

                for(int i = 0; i < 5; i++) {
                        DEC *= 16;
                        digit = (int) DEC;
                        DEC -= digit;
                        Apoint += Hexa.DECtoHEX(digit);
                        if(DEC == 0)
                              break;
                }
                HexaF output = new HexaF(Bpoint + Apoint);
                return output;
        }
        public static String DECtoHEXF_LINE(String dec_Line) {
                if(dec_Line == "")
                        return "";
                String hex_Line = "";
                dec_Line =
Usefull_Methods.setSpacesEachToken(dec_Line);
                String[] Tokens = dec_Line.split("\\s+");
                HexaF number;

                for(String S : Tokens) {
                        if(Usefull_Methods.StringisNumber(S)) {
                              number =
DECtoHEXF(Double.parseDouble(S));
                              hex_Line += number;
                        }
                        else {
                              hex_Line += S;
                        }
                }
                return hex_Line;
        }
        public HexaF Add(HexaF HEXF);
        public HexaF Subtract(HexaF HEXF);
        public HexaF Multiplication(HexaF HEXF);
        public HexaF Division(HexaF HEXF);
        public HexaF Pow(HexaF HEXF);
        public String toString();
}
package numeric_Systems;

public interface HexaSystem {

        public Hexa Add(Hexa HEX);
        public Hexa Subtract(Hexa HEX);
        public Hexa Multiplycation(Hexa HEX);
        public Hexa Division(Hexa HEX);
        public String toString();
}
package numeric_Systems;
```

```
public class Okta implements OktSystem{

        String value;



        public Okta(String value) {
                setValue(value);
        }
        @Override
        public Okta Add(Okta OKT) {
                int a = Okta.OKTToDEC(this);
                int b = Okta.OKTToDEC(OKT);

                return Okta.DECtoOKT(a+b);
        }
        @Override
        public Okta Subtract(Okta OKT) throws Exception {
                int a = Okta.OKTToDEC(this);
                int b = Okta.OKTToDEC(OKT);

                return Okta.DECtoOKT(a-b);
        }
        @Override
        public Okta Multiplycation(Okta OKT) {
                int a = Okta.OKTToDEC(this);
                int b = Okta.OKTToDEC(OKT);

                return Okta.DECtoOKT(a*b);
        }
        @Override
        public Okta Division(Okta OKT) {
                int a = Okta.OKTToDEC(this);
                int b = Okta.OKTToDEC(OKT);

                return Okta.DECtoOKT(a/b);
        }
        public static int OKTToDEC(Okta OKT) {
                return Integer.parseInt(OKT.getValue(), 8);
        }
        public static Okta DECtoOKT(int DEC) {
                return new Okta(Integer.toOctalString(DEC));
        }
        public String getValue() {
                return value;
        }
        public void setValue(String value) {

                char[] Cvalue = value.toCharArray();
                for(char E : Cvalue) {
                        if(E < 48 && E > 55) {
                                throw new IllegalArgumentException("only
Chars from 1-7");
                        }
                }
                this.value = value;
        }
        public String toString() {
                return value;
        }

}
package numeric_Systems;

public class OktaF implements OktaFSystem {
        private String [] value;
        public String[] getValue() {
                return value;
        }

        public OktaF(String value) {
                this.value = value.split("\\.");
                //If .x is missing.
                if(this.value.length==1) {
                        String[] temp = new String[2];
                        temp[0] = this.value[0];
                        temp[1] = "0";
                        this.value = temp;
                }
        }

        @Override
        public String toString() {
                return value[0] + "." + value[1];
        }

        @Override
        public OktaF Add(OktaF OKTF) {
                double a = OktaFSystem.OKTFToDEC(this);
                double b = OktaFSystem.OKTFToDEC(OKTF);
```

```
		return OktaFSystem.DECtoOKTF(a+b);
	}

	@Override
	public OktaF Subtract(OktaF OKTF) {
		double a = OktaFSystem.OKTFToDEC(this);
		double b = OktaFSystem.OKTFToDEC(OKTF);

		return OktaFSystem.DECtoOKTF(a-b);
	}

	@Override
	public OktaF Multiplication(OktaF OKTF) {
		double a = OktaFSystem.OKTFToDEC(this);
		double b = OktaFSystem.OKTFToDEC(OKTF);

		return OktaFSystem.DECtoOKTF(a*b);
	}

	@Override
	public OktaF Division(OktaF OKTF) {
		double a = OktaFSystem.OKTFToDEC(this);
		double b = OktaFSystem.OKTFToDEC(OKTF);

		return OktaFSystem.DECtoOKTF(a/b);
	}

	@Override
	public OktaF Pow(OktaF OKTF) {
		double a = OktaFSystem.OKTFToDEC(this);
		double b = OktaFSystem.OKTFToDEC(OKTF);

		return OktaFSystem.DECtoOKTF(Math.pow(a, b));
	}
}
package numeric_Systems;

import ubung7.Usefull_Methods;

public interface OktaFSystem {

	public static double OKTFToDEC(OktaF OKTF) {

		int Bpoint = Okta.OKTToDEC(new
Okta(OKTF.getValue()[0]));
```

```
            char[] c_Apoint = OKTF.getValue()[1].toCharArray();
            double Apoint = 0;

            for(int i = 0 ; i< c_Apoint.length;i++) {
                  int hex_Value = Okta.OKTToDEC(new
Okta(Character.toString(c_Apoint[i])));
                  Apoint +=  hex_Value*Math.pow(8, -(i+1));
            }
            return Bpoint + Apoint;
      }
      public static OktaF DECtoOKTF(double DEC) {

            Okta Bpoint = Okta.DECtoOKT((int) DEC);
            DEC -= (int) DEC;
            String Apoint = ".";
            int digit;

            for(int i = 0; i < 5; i++) {
                  DEC *= 8;
                  digit = (int) DEC;
                  DEC -= digit;
                  Apoint += Okta.DECtoOKT(digit);
                  if(DEC == 0)
                        break;
            }
            OktaF output = new OktaF(Bpoint + Apoint);
            return output;
      }
      public static String DECtoOKTF_LINE(String dec_Line) {
            if(dec_Line == "")
                  return "";
            String okt_Line = "";
            dec_Line =
Usefull_Methods.setSpacesEachToken(dec_Line);
            String[] Tokens = dec_Line.split("\\s+");
            OktaF number;

            for(String S : Tokens) {
                  if(Usefull_Methods.StringisNumber(S)) {
                        number =
DECtoOKTF(Double.parseDouble(S));
                        okt_Line += number;
                  }
                  else {
                        okt_Line += S;
```

```
                }
            }
            return okt_Line;
        }
        public OktaF Add(OktaF OKTF);
        public OktaF Subtract(OktaF OKTF);
        public OktaF Multiplication(OktaF OKTF);
        public OktaF Division(OktaF OKTF);
        public OktaF Pow(OktaF OKTF);
        public String toString();
}
package numeric_Systems;

public interface OktSystem {
        public Okta Add(Okta OKT);
        public Okta Subtract(Okta OKT) throws Exception;
        public Okta Multiplycation(Okta OKT);
        public Okta Division(Okta OKT);
        public String toString();
}
```

```
package Postfix_Stack;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import ubung7.Usefull_Methods;

public class Postfix {

        char token;

        public String infixToPostfix (String infix) throws
StackUnderflow, StackException
        {
                String result ="";
                StackAsList stack = new StackAsList();

                infix = Usefull_Methods.deleteSpaces(infix);

                for (int i = 0; i < infix.length(); i++)
                {
                        token = infix.charAt(i);
                        if (token > 47 && token < 58)
                            {
                                result += token;
                            }

                        else if (token == '(')
                        {
                            stack.push(token);
                        }
                        else if (token == ')')
                        {
                            while (stack.top() != (Character)'(')
                            {
                                result += stack.top();
                                stack.pop();
                            }
                            stack.pop();
                        }
                        else if (isOperator(token))

                        {
                            while (!stack.isEmpty()
&&(!((precedence((char) stack.top()) < precedence(token) )||
                                        (token == '^' &&
precedence((char) stack.top()) == precedence(token) ))))
                            {
                                result += stack.top();
                                stack.pop();
                            }

                            stack.push(token);
                        }
                        else {
                            throw new StackException("Wrong
Input");
                        }

                }

                while (!stack.isEmpty())
                {
                        result += stack.top();
                        stack.pop();
                }
```

```
		return result;
	}

	public boolean isOperator (char token) {

		if (token == '+' ||
			token == '-' ||
			token == 'x' ||
			token == '/' ||
			token == '^')
			return true;

		else
			return false;
	}


	public int precedence (char token)
	{
		switch(token) {

		case '+':
		case '-':
			return 0;
		case 'x':
		case '/':
			return 1;
		case '^':
			return 2;
		default:
			return -1;

		}
	}

	public double evaluate (String pfx) throws
StackUnderflow, StackException {


		double rhs= 0;
		double lhs = 0;
		double tokenInt = 0;

		double result = 0;
```

```
            StackAsList stack = new StackAsList();

            pfx = Usefull_Methods.deleteSpaces(pfx);

            for (int i = 0; i <pfx.length(); i++)
            {
                  token = pfx.charAt(i);

                  if (token > 47 && token < 58)
                  {
                        tokenInt =
Character.getNumericValue(token);
                        stack.push(tokenInt);
                  }

                  else if (isOperator(token))
                  {

                        rhs = (double) stack.top();
                        stack.pop();
                        lhs =  (double) stack.top();
                        stack.pop();

                        if (token == '+')
                              {result = lhs + rhs;
                              stack.push(result);
                              }
                        else if (token == '-')
                              {result = lhs - rhs;
                              stack.push(result);
                              }
                        else if (token == 'x')
                              {result = lhs * rhs;
                              stack.push(result);
                              }

                        else if (token == '/')
                              {result =  lhs /rhs;
                              stack.push(result);
                              }
                        else if (token == '^')
                              {result = (double) Math.pow(lhs,
rhs);

                              stack.push(result);
                              }
                  }
                  else {
                        throw new StackException("Wrong
Input");
                  }
            }

            return (double) stack.top();

      }

      public void readInfix() throws StackUnderflow,
IOException, StackException
      {
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            String input = br.readLine();

      System.out.println(evaluate(infixToPostfix(input)));

      }

}
package Postfix_Stack;


public interface Stack<E> {
       public void push (E item);
       public void pop () throws StackUnderflow;
       public Object top () throws StackUnderflow;
       public boolean isEmpty ();
       public void Empty ();
       public void print();
       public String toString();
      }
package Postfix_Stack;


public class StackAsList implements Stack {
      private Node myStack;
      private class Node {

            public Node(Object data, Node next) {
                  this.data = data;
                  this.next = next;
            }
            public Node() {
```

```
			data = null;
			next = null;
		}
		Object data;
		Node next;
		public String toString() {
			return data.toString();
		}
	}



	public StackAsList() {
		myStack=null;
	}

	@Override
	public void push(Object item) {

		Node temp = new Node(item, myStack);
		myStack = temp;
	}

	@Override
	public void pop() throws StackUnderflow {
		if(!isEmpty())
		myStack = myStack.next;
	}

	@Override
	public Object top() throws StackUnderflow {
		// TODO Auto-generated method stub
		return myStack.data;
	}

	@Override
	public boolean isEmpty() {
		if(myStack == null) {
			return true;
		}
		else
			return false;
	}

	@Override
	public void Empty() {
```

```
            myStack = null;
      }

      @Override
      public void print() {
            System.out.println(this.toString());
      }
      @Override
      public String toString() {
            String S_myString = "";
            Node temp = new Node();
            temp = myStack;
            while (isEmpty()) {
                  if(temp.next == null) {
                        break;
                  }
                  temp = temp.next;
                  S_myString += temp.data + "\r\n";
            }
            return S_myString;
      }
}
package Postfix_Stack;

public class StackException extends Exception
{
    public StackException( String message )
    {
       super( message );
    }
 }
package Postfix_Stack;


public class StackUnderflow extends Exception {

}


 package ubung7;

 import Postfix_Stack.Postfix;
 import Postfix_Stack.StackException;
 import Postfix_Stack.StackUnderflow;

 public class CalcEngine_V2 extends Postfix_Multidigit {
```

```
private String displayValue;
public String getDisplayValue() {
	return displayValue;
}
public void ADDtoDisplayValue(String AddString) {
	displayValue += AddString;
}
public void setDisplayValue(String displayValue) {
	this.displayValue = displayValue;
}

private double input;
public String getinput() {
	if(input == 0)
		return "";
	else
		return Double.toString(input);
}
public void ADDtoinput(int AddNumber, int Base){
	if(!(input == 0 && AddNumber == 0)) {
		input = input*Base + AddNumber;
	}
}
public void setinput(double input) {
	this.input = input;
}

public String result;
public CalcEngine_V2() {
	super();
	this.displayValue = "";
}
public String getTitle()
{
	return "Cool Java Calculator";
}
/**
* @return The author of this engine.
*/
public String getAuthor()
{
	return "David J. Barnes and Michael Kolling";
}
/**
* @return The version number of this engine.
*/
public String getVersion()
{
		return "Version 1.0";
	}
	public void clear() {
		displayValue = "";
		input = 0;
	}
	public void calcResult() throws StackUnderflow,
StackException {
		result =
Double.toString(evaluate(infixToPostfix(getDisplayValue())));
	}
	public String getResult() {
		return result;
	}

}
package ubung7;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import Postfix_Stack.StackAsList;
import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.Hexa;

public class Postfix_Multidigit {

	String token;

	public String infixToPostfix (String infix) throws
StackUnderflow, StackException
	{
		String result ="";
		StackAsList stack = new StackAsList();
		infix = Usefull_Methods.setSpacesEachToken(infix);

		//https://stackoverflow.com/questions/7899525/how-
to-split-a-string-by-space/7899558
		String [] infix_splitted = infix.split("\\s+");


		for (int i = 0; i < infix_splitted.length; i++)
		{
			token = infix_splitted[i];
			System.out.println(token);

			if (Usefull_Methods.StringisNumber(token))
```

```
                        {
                              result += token + " ";
                        }

                  else if (token.equals("("))
                  {
                        stack.push(token);
                  }
                  else if (token.equals(")"))
                  {
                        while (!stack.top().equals("("))
                        {
                              result += stack.top() + " ";
                              stack.pop();
                        }
                        stack.pop();
                  }
                  else if
(isOperator(token.charAt(token.length()-1)))

                  {
                        while (!stack.isEmpty()
&&(!((precedence(stack.top().toString()) < precedence(token))
)||
                                    (token.equals("^") &&
precedence(stack.top().toString()) == precedence(token))))
                        {
                              result += stack.top() + " ";
                              stack.pop();
                        }

                        stack.push(token);
                  }
                  else {
                        System.out.println(token);
                        throw new StackException("Wrong Input
1");
                  }

            }

            while (!stack.isEmpty())
            {
                  result += stack.top() + " ";
                  stack.pop();
            }

            return result.stripTrailing();
      }
```

```
	public boolean isOperator (char token) {

		if (token == '+' ||
			token == '-' ||
			token == 'x' ||
			token == '/' ||
			token == '^')
			return true;

		else
			return false;
	}
	public int precedence (String token)
	{
		switch(token) {

		case "+":
		case "-":
			return 0;
		case "x":
		case "/":
			return 1;
		case "^":
			return 2;
		default:
			return -1;

		}
	}
	public double evaluate (String pfx) throws
StackUnderflow, StackException {


		double rhs= 0;
		double lhs = 0;
		double tokenInt = 0;

		double result = 0;

		StackAsList stack = new StackAsList();

		String [] pfx_splitted = pfx.split("\\s+");

		for (int i = 0; i <pfx_splitted.length; i++)
		{
			token = pfx_splitted[i];

			if (Usefull_Methods.StringisNumber(token))
			{
```

```
                        tokenInt = Double.parseDouble(token);
                        stack.push(tokenInt);
                }

                else if
(isOperator(token.charAt(token.length()-1)))
                {

                        rhs = (double) stack.top();
                        stack.pop();
                        lhs =  (double) stack.top();
                        stack.pop();

                        if (token.equals("+")) {
                              result = lhs + rhs;
                              stack.push(result);
                              }
                        else if (token.equals("-")) {
                              result = lhs - rhs;
                              stack.push(result);
                              }
                        else if (token.equals("x")) {
                              result = lhs * rhs;
                              stack.push(result);
                              }

                        else if (token.equals("/")) {
                              result =  lhs /rhs;
                              stack.push(result);
                              }
                        else if (token.equals("^")) {
                              result = (double) Math.pow(lhs,
rhs);

                              stack.push(result);
                              }
                }
                else {
                        throw new StackException("Wrong
Input");
                }
          }
          return (double) stack.top();
      }
}
package ubung7;

import static org.junit.jupiter.api.Assertions.*;

import java.awt.event.ActionEvent;
import org.junit.jupiter.api.Test;

import Calculator.Calculator;
import numeric_Systems.BinF;
import numeric_Systems.BinFSystem;
import numeric_Systems.HexaF;
import numeric_Systems.HexaFSystem;
import numeric_Systems.OktaF;
import numeric_Systems.OktaFSystem;

class ubung7_TEST {

      Calculator Test;
      ActionEvent event;

      @Test
      void testActionPerformed() {
            Test = new Calculator(4);
            //Button

            //Numbers from 1-9
            event = new ActionEvent(Test.gui, 0, "1");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "2");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "3");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "4");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "5");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "6");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "7");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "8");
                  Test.gui.actionPerformed(event);
            event = new ActionEvent(Test.gui, 0, "9");
                  Test.gui.actionPerformed(event);

            //Now all of them should be in the engine.input
field.
            assertEquals(123456789,
Double.valueOf(Test.engine.getinput()));

            //Now if we press "=" they should be in
DisplayCalue before its pressed Display is empty
            assertEquals("", Test.engine.getDisplayValue());
            event = new ActionEvent(Test.gui, 0, "=");
            Test.gui.actionPerformed(event);
            assertEquals(123456789,
Double.valueOf(Test.engine.getDisplayValue()));
            //Now we clear it
            event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);
            assertEquals("", Test.engine.getDisplayValue());

            //for 4 15 Test cases:
            Test = MULToverADD(Test, '1', '+', '2', 'x', '3',
'+', '4');
            assertEquals("11.0",Test.engine.result);
            Test = MULToverADD(Test, '2', '+', '1', 'x', '4',
'+', '3');
            assertEquals("9.0",Test.engine.result);
            Test = MULToverADD(Test, '1', '+', '2', 'x', '3',
'+', '4');
            assertEquals("11.0",Test.engine.result);
            Test = MULToverADD(Test, '2', '+', '5', 'x', '5',
'+', '2');
            assertEquals("29.0",Test.engine.result);
            Test = MULToverADD(Test, '1', 'x', '2', '+', '3',
'x', '4');
            assertEquals("14.0",Test.engine.result);
            Test = MULToverADD(Test, '2', 'x', '1', '+', '4',
'x', '3');
            assertEquals("14.0",Test.engine.result);
            Test = MULToverADD(Test, '2', 'x', '5', '+', '5',
'x', '2');
            assertEquals("20.0",Test.engine.result);
            Test = MULToverADD(Test, '1', 'x', '2', 'x', '3',
'+', '5');
            assertEquals("11.0",Test.engine.result);
            Test = MULToverADD(Test, '2', 'x', '1', 'x', '4',
'+', '1');
            assertEquals("9.0",Test.engine.result);
            Test = MULToverADD(Test, '2', 'x', '5', 'x', '5',
'+', '2');
            assertEquals("52.0",Test.engine.result);
            Test = MULToverADD(Test, '1', '+', '2', 'x', '3',
'x', '4');
            assertEquals("25.0",Test.engine.result);
            Test = MULToverADD(Test, '2', '+', '1', 'x', '4',
'x', '3');
            assertEquals("14.0",Test.engine.result);
            Test = MULToverADD(Test, '2', '+', '5', 'x', '5',
'x', '2');
            assertEquals("52.0",Test.engine.result);
            Test = MULToverADD(Test, '1', '+', '2', 'x', '1',
'0', '0');
            assertEquals("201.0",Test.engine.result);
            Test = MULToverADD(Test, '4', '+', '3', 'x', '2',
'/', '6');
            assertEquals("5.0",Test.engine.result);

            event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);

            //lets now make a really complicated calculation
like: -4.2*-1*(5.35+6.789) = 50.9838.
            //Picture
            Test = DECComplicatedCalc(Test);
            assertEquals(Test.engine.result, "50.9838");

            //now lets see if we can work with this result: i
want -10 so we need -60.9838.
            Test = DEC_negativ(Test);
            //u know Java stupid round things -
9.999999999999993 PICTURE
            double result =
Math.round(Double.valueOf(Test.engine.result)*1000000000)/1000
000000;
            assertEquals(-10, result);

            //Lets do some simple HEXF BINF and OKTF
calculations
            event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);

            //HEX -A.A*-1*(B.CD+E.FAB) = 11C.89AE
            //First lets simulate a BOX selection
            Test.gui.Boxes[1].setSelected(true);
            event = new ActionEvent(Test.gui, 0, "HEX");
            Test.gui.actionPerformed(event);
            assertEquals(16, Test.gui.Base);
            //now the Calculation
            Test = HEXComplicatedCalc(Test);
            HexaF number_HEX =
HexaFSystem.DECtoHEXF(Double.valueOf(Test.engine.result));
            assertEquals("11C.89AE", number_HEX.toString());

            event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);

            //At least lets Test BIN cos we cant really Test
OKT
            //-4.5*-1*(5.75+6.25) = 54
            //-100.1*-1*(101.11+110.01) = 110110.0
            Test.gui.Boxes[2].setSelected(true);
            event = new ActionEvent(Test.gui, 0, "BIN");
            Test.gui.actionPerformed(event);
            assertEquals(2, Test.gui.Base);
            //now the Calculation
            Test = BINComplicatedCalc(Test);
            BinF number_BIN =
BinFSystem.DECtoBINF(Double.valueOf(Test.engine.result));
            System.out.println(number_BIN);
            assertEquals("110110.0", number_BIN.toString());

            event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);

            //For Assignment 6 and with Okta:
            //1.5*2^2 = 6
            Test.gui.Boxes[3].setSelected(true);
            event = new ActionEvent(Test.gui, 0, "OKT");
            Test.gui.actionPerformed(event);
            assertEquals(8, Test.gui.Base);
            //now the Calculation
            Test = OKTComplicatedCalc(Test);
            OktaF number_OKT =
OktaFSystem.DECtoOKTF(Double.valueOf(Test.engine.result));
            System.out.println(number_OKT);
            assertEquals("6", number_OKT.toString());

      }
      private Calculator MULToverADD(Calculator Test,char
num1, char op1, char num2, char op2, char num3, char op3, char
num4){
      event = new ActionEvent(Test.gui, 0, "Clear");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+num1);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ""+op1);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+num2);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+op2);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+num3);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+op3);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0,""+num4);
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "=");
      Test.gui.actionPerformed(event);
      return Test;
}
private Calculator OKTComplicatedCalc(Calculator Test) {

event = new ActionEvent(Test.gui, 0, "1");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, ".");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "5");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "x");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "2");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "^");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "2");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "=");
      Test.gui.actionPerformed(event);
      return Test;
}
private Calculator DECComplicatedCalc(Calculator Test) {
      event = new ActionEvent(Test.gui, 0, "-");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "4");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ".");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "2");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "x");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "-");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "1");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "x");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "(");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "5");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ".");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "3");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "5");
```

```
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "+");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "6");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ".");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "7");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "8");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "9");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ")");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "=");
            Test.gui.actionPerformed(event);
      return Test;
}
private Calculator DEC_negativ(Calculator Test) {
event = new ActionEvent(Test.gui, 0, "-");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "6");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "0");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, ".");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "9");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "8");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "3");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "8");
      Test.gui.actionPerformed(event);
event = new ActionEvent(Test.gui, 0, "=");
      Test.gui.actionPerformed(event);
      return Test;
}
private Calculator HEXComplicatedCalc(Calculator Test) {
      event = new ActionEvent(Test.gui, 0, "-");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "A");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, ".");
            Test.gui.actionPerformed(event);
      event = new ActionEvent(Test.gui, 0, "A");
            Test.gui.actionPerformed(event);
```

```
        event = new ActionEvent(Test.gui, 0, "x");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "-");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "1");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "x");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "(");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "B");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, ".");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "C");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "D");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "+");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "E");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, ".");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "F");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "A");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "B");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, ")");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "=");
              Test.gui.actionPerformed(event);
        return Test;
}
private Calculator BINComplicatedCalc(Calculator Test) {
        event = new ActionEvent(Test.gui, 0, "-");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "1");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "0");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "0");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, ".");
              Test.gui.actionPerformed(event);
        event = new ActionEvent(Test.gui, 0, "1");
              Test.gui.actionPerformed(event);
```

```java
		event = new ActionEvent(Test.gui, 0, "x");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "-");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "x");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "(");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "0");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, ".");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "+");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "0");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, ".");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "0");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "1");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, ")");
		      Test.gui.actionPerformed(event);
		event = new ActionEvent(Test.gui, 0, "=");
		      Test.gui.actionPerformed(event);
		return Test;
	}
}
package ubung7;


public class Usefull_Methods {

	public static String deleteSpaces(String s) {
			return s.replace("\\s+", "");
		}

		public static boolean StringisNumber(String number) {

			if(number.startsWith("."))
				return false;

			char[] cNumber = number.toCharArray();

			int i = 0;
			try {
				if(cNumber[0] == '-' &&
CharisNumberHEX(cNumber[1]))
					i = 1;
			}
			catch (Exception e) {}

			for (; i < cNumber.length ; i++) {
				if(!(CharisNumberHEX(cNumber[i]) ||
cNumber[i] == '.'))
					return false;
			}
			return true;
		}
		public static boolean CharisNumberHEX(char number) {
			if( number > 47 && number < 58 ||
				number > 64 && number < 71)
				return true;
			else
				return false;
		}
		public static String setSpacesEachToken(String S) {
			S = deleteSpaces(S);
			char [] S_Array = S.toCharArray();

			//If the input String has 2 dots in a row its
wrong Tested here.
			int count = 0;
			for(char E : S_Array) {
				if(E == '.')
					count++;
				else
					count = 0;
				if(count >1)
					return null;
			}
```

```
                String token = "";
                S = "";
                for(int i = 0; i< S_Array.length;i++) {
                      token = "";

                      //negativ numbers
                      if(   S_Array[i] == '-') {
                            token += S_Array[i];

                            try {
                                  //if its an neg followed by a
number

      if(CharisNumberHEX(S_Array[i+1])){
                                        //If - is the first digit:
                                        if(i==0) {

      while(CharisNumberHEX(S_Array[i+1]) || S_Array[i+1] ==
'.') {
                                                    i++;
                                                    token +=
S_Array[i];
                                              }
                                        }
                                        else {
                                              //if the - is not at
the first digit

      if(CharisOperator(S_Array[i-1])) {

      while(CharisNumberHEX(S_Array[i+1]) || S_Array[i+1] ==
'.') {
                                                          i++;
                                                          token +=
S_Array[i];
                                                    }
                                              }
                                        }
                                  }
                                  //if the neg is followed by an (
                                  else if(S_Array[i+1] == '(') {
                                        i++;
                                        token += S_Array[i];
                                  }
                            }
                            catch(Exception e) {}
                            S += token + " ";
                            continue;
                      }
```

```
            //operatoren
            if( i > 0  && CharisOperator(S_Array[i])) {
                token += S_Array[i];
                S += token + " ";
                continue;
            }
            //positive Zahlen
            else if(CharisNumberHEX(S_Array[i])){
                token =
Character.toString(S_Array[i]);
                try {

    while(CharisNumberHEX(S_Array[i+1]) || S_Array[i+1] ==
'.') {
                        i++;
                        token += S_Array[i];
                    }
                }
                catch(Exception e) {}
                S += token + " ";
                continue;
            }
            else {
                //If input is wrong.
                return null;
            }
        }


        return S.stripTrailing();
    }
    public static boolean CharisToken(char c) {
        if(CharisNumberHEX(c) || CharisOperator(c)) {
            return true;
        }
        else
            return false;
    }
    public static boolean CharisOperator(char c) {
        if( c == '+' ||
            c == '-' ||
            c == 'x' ||
            c == '*' ||
            c == '/' ||
            c == '^' ||
            c == '(' ||
            c == ')' ||
            c == '=')
            return true;
        else
            return false;
    }
}
```