

Lab Report

EXERCISE 8: FUN WITH CALCULATORS 3

	Name	Titel des Kurses	Datum
Juri Wiechmann	571085	Prof. Dr.	16.12.2019
Bartholomäus Berresheim	568624	Weber-Wulff	
		Info 2 Group 2	

Index

Introduction

Pre-lab

1. a. Either polish your extended JulianDate class or see if one of the bored has made ...
2. How do you get your calculator to accept input of your chosen data type and to ...

Assignments

1. Make **another** new copy of the Calculator (don't wreck your previous versions!)
2. If you are doing the *date calculator*, implement the following functions:
 1. input a date
 2. get the calculator to display a String for the weekday in the window (hint: you will need a button to push)
 3. add a number of days to a date, displaying the new date
 4. subtract a number of days from a date, displaying the new date
 5. subtract two dates, giving the number of days between the dates

Reflections

Juri
Bartholomäus

Introduction

This week's lab concludes our work with our calculators. All in all it combines the work of five different labs. For this lab, we chose to make a copy of Juri's previous calculator, since it had a couple more functions than the one from Bartholomäus. We won't explain how the previous Calculator works and how the different UnserInterface-classes work together. Methods that update the Text Fields aren't a relevant part of this Lab. Otherwise, if we were to do this, the report would be way too big.

PRE-LAB

1. We are now going to make a calculator for a *different* data type that just integers. Of course, the bored are welcome to implement any fascinating new data types they can think up for using a calculator with! Choose one of the following data types:
 - a. Either polish your extended JulianDate class or see if one of the bored has made something useful. Make sure that you can read and write dates in a specific format. Add methods such as public void addDays (int days); if necessary. You will be producing a date calculator. Decide how you are going to map functions such as adding or subtracting a number of days to a date, determining the number of days between a date, and determining the weekday to the buttons +, -, *, and /.

Our previous JD-classes didn't have a field to save the value of a JulianDay. So we did some minor changes like creating said field and making all non static methods use the field instead of getting an int as parameter. We also put all static methods like: get_FirstDay(); JulianToGregor(int JD) and GregorToJulian(GregorDate Day) into the interface, since it's how Java want us to handle static methods now. We also added the method subtractDays(int Days) so that we have 4 methods that we can map our operators to:

addDays(int Days)	= "+",
subtractDays(int Days)	= "-",
daysBetween(GregorDate Day)	= "x",
get_Weekday()	= "/".

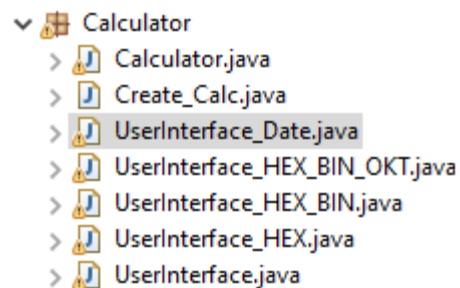
2. How do you get your calculator to accept input of your chosen data type and to display output of the chosen type?

Our Calculator is working in different modi which can be selected by the checkboxes. The Engine only works so far with decimal numbers. We only convert them for the displays. Now we can simply add one Checkbox and say to our actionPerformed(ActionEvent event) method that we want to differentiate between our previous Calculator with the numeric systems, and the Date mode. If it's in the Date mode we just use the JulianDate methods that we mapped on the buttons

Assignments

1. Make another new copy of the Calculator (don't wreck your previous versions!) before you start. Or you can extend your Calculator to be able to do decimal, hex, and your chosen data type!

We made a copy of Juri's Calculator and thanks to its Interface-layer-system we can simply extend it by another Interface, in which we can focus on dealing with dates.



2. If you are doing the *date calculator*, implement the following functions:
 1. Input a date

We thought about how to differentiate between the different modes that we can have now. Everytime we selected a different Checkbox, we change the Base which is needed to have some calculation behind the scenes for the numeric systems. We can simply say, if we select the Date Checkbox then our Base becomes 0. Now we check the Base, if it is higher than 0, we go for the numeric calculation setup. If it is equal to 0 we use the Dates setup. With this we have a simple way of adding other systems later on, if needed. We can deal with them by mapping their Base to -1, -2 and so on.

```
//numeric systems
else if(Base > 0) {
    numericCalc(command);
}
//Dates
else if(Base == 0) {
    dateCalc(command);
}
```

Now we can focus on the Date things in the dateCalc(String command) method without worrying about crashing our working numeric systems. Our Main idea is the following: Everytime we add a number or an "." we simple add them to the input String. We call them "Date-parts". If we press an operator we add the input to the DisplayValue. Then we need to discern if its an "=" or something else. If it's something else we also need to check if its an "/" or something else, since the operator "/" doesn't need another input. Its working similar to "=". For the other ones that are mapped, we simply add them to our DisplayValue and that's it. You also can see that we don't use "." but instead "sep". We wanted to be free to change the separator at any time and maybe later extend it to work with other formats.

```
//Date-parts
else if(Usefull_Methods.StringIsNumber(command) || command == sep){
    calc.ADDtoinput(command);
}
```

At this point we also extended our GregorDate-class to be able to handle a lot of different separators:

```

public GregorDate(String Date) throws Exception {
    //can handle .;/-;_
    String sep = "\\\" + Usefull_Methods.getSep(Date);

    String[] Date_ = Date.split(sep);

    if(Date_.length != 3) {
        throw new Exception("Wrong Date input");
    }
    else {
        setYear(Integer.valueOf(Date_[0]));
        setMonth(Integer.valueOf(Date_[1]));
        setDay(Integer.valueOf(Date_[2]));
    }
}
}

```

Here we can extend the Character that count as separator:

```

public static String getSep(String input) {
    //if you want too really searching for an sepperator u need a more complicated way, this is the bugged version
    if(input.contains("/"))
        return "/";
    else if(input.contains("-"))
        return "-";
    else if(input.contains("."))
        return ".";
    else if(input.contains("_"))
        return "_";
    return null;
}

```

2. Get the calculator to display a String for the weekday in the window (hint: you will need a button to push)

Ok lets now focus on the Weekday-Display. As we mentioned before, we handled this one in another way because it does not need a second input after the mapped operator is pressed. First we create a MyJulianDate with the DisplayValue as String. We simply set our Input to the Weekday which we calculate using our get_Weekday() method, updated our Input and DisplayValue-Textfield. Than we set our Input to "o" so that we can work with it again without pressing the "clear" Button.

```

else {
    myJD_1 = new MyJulianDate(new GregorDate(calc.getinput()));
    // / dont need = to work thats why we put it in here.
    if(command.equals("/")) {
        calc.setinput(myJD_1.get_Weekday());
        redisplay_Date();
        reIndex_Display_Date();
        calc.setinput(0);
    }
    else if(command.equals("(")||
            command.equals(")")||
            command.equals("^")){
        System.out.println("NOT MAPPED COMMAND");
    }
    else {
        calc.ADDtoDisplayValue(command);
        calc.setinput(0);
        redisplay_Date();
    }
}
}

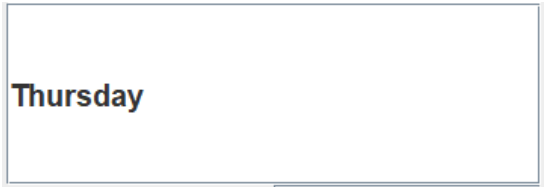
```

As an example, if we input todays date into the calculator:



02019.12.12

and we then proceed by pushing the “/” button, we get :



Thursday

By quickly checking the calendar, we can confirm that our calculator is working as intended.

3. Add a number of days to a date, displaying the new date

Now we are talking about the “+” operator. As we mentioned before, we simply add the operator to the DisplayValue String, that contains a gregorian date. Then we input the second value, that is needed, in this case an int with the amount of days we want to add. Afterwards we press the “=” button. To decide what to do, we search now in our DisplayValue String for one of those mapped operators using the contains() method, and do the calculations using the methods that we declared in the MyJulianDate-class:

```
if(command.equals("=")) {  
    //if + is in String  
    if(calc.getDisplayValue().contains("+")) {  
        calc.setInput(myJD_1.addDays(Integer.valueOf(calc.getInput())).toString(sep));  
    }  
}
```

We do something similar to what we did in the “/”-if statement. We put the result in our input, update it, put it into the DisplayValue, update it to finally set it to “o” again.

We also added a modified toString(String sep) to our GregorDate-class. It now puts the selected separator between each number:

```
public String toString(String sep) {  
    String FirstLetterDay = ((this.day<10) ? "0" : "");  
    String FirstLetterMonth = ((this.month<10) ? "0" : "");  
    return Integer.toString(year) +  
        sep +  
        FirstLetterMonth +  
        Integer.toString(month) +  
        sep + FirstLetterDay +  
        Integer.toString(day);  
}
```

Our addDays(int Days)- method looks like this:

```

public GregorDate addDays(int Days) throws Exception {
    //no negativ JulianDates allowed
    if(JulianDate.GregorToJulian(this.Day)+Days<0)
        throw new Exception("negativ JulianDate not exsistent");
    else
        return JulianDate.JulianToGregor(JulianDate.GregorToJulian(this.Day)+Days);
}

```

After having finished the code in a way that the calculation should have worked, we tried to make a simple calculation of adding a couple days to the current date but it didn't work. For some reason the first inputted gregorian date had a "o" at the start of it and the number and the second input which signified the number of days that we wanted to add, was preceded by a "o.o", which resulted in the inability of the calculator to complete the calculation after having pushed the equals button.

02019.12.12+
0.03

We then added some System.out.println() to make sure that the input was properly read and that the if-statement for the plus-operator was entered.

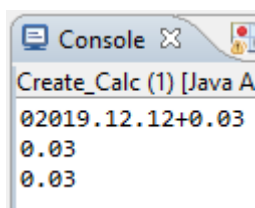
```

if(command.equals("=")) {
    //if + is in String
    System.out.println(calc.getDisplayValue());
    System.out.println(calc.getInput());

    if(calc.getDisplayValue().contains("+")) {
        System.out.println(calc.getInput());
        calc.setInput(myJD_1.addDays(Integer.valueOf(calc.getInput())).toString(sep));
    }
}

```

Which turned out to be the case.

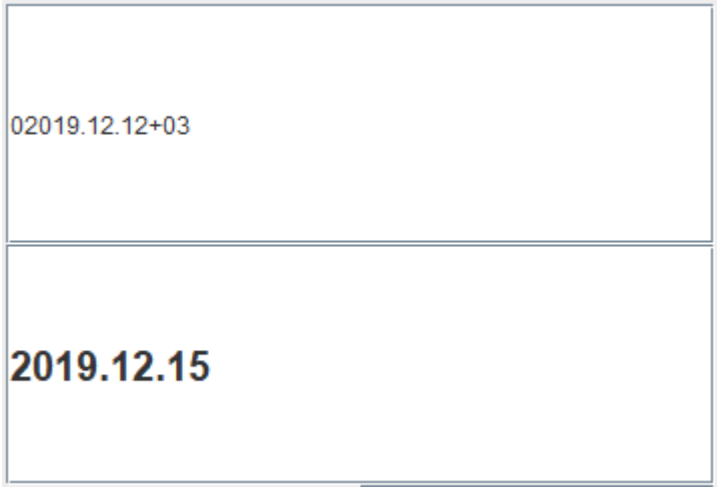


After some short deliberation, we came to the conclusion that the origin of the problem probably lies with the "o.o" in front of the second input.

Although we couldn't find the origin of that problem, we found a way of removing the "o.o" and later the "o", so that the calculator would work properly. We managed this by adding an if-else statement in the `getinput()` method from the `CalcEngine_V2` class, that would check if the input String is equal to "o" OR if it's first three chars from the input are equal to "o.o". If either are true, the input would be set to "o" and returned. If not, then input would be returned

```
public String getinput() {  
    if (input == "0" || (input.charAt(0) == '0' && input.charAt(1) == '.' && input.charAt(2) == '0'))  
        return input = "0";  
    else return input;  
}
```

Now if we try our previous calculation again, we get the following result:



The image shows a calculator interface with two input fields. The top field contains the text "02019.12.12+03". The bottom field contains the text "2019.12.15".

As we can see, the calculation works now, but we still have that pesky "o" preceding both inputs. We managed to remove it, by adding an if-else statement to the `ADDtoinput()` method from the `CalcEngine_V2` class. In it, we checked if `this.input` is equal to "o", if true it gets overwritten by the input used to call the `ADDtoinput()` method. When false, the input gets added to `this.input`.

```
public void ADDtoinput(String input){  
    if (this.input == "0")  
        this.input = input;  
    else this.input += input;  
}
```

Now when we try our previous calculation again, we get :

2019.12.12+3
2019.12.15

As we can see, the calculator is working perfectly fine, and we lost the “o” preceding the inputs.

4. Subtract a number of days from a date, displaying the new date.

This assignment was rather short, since its working very similarly to the one about the “+” operator. After pressing the “=” button, we search for the “-”, then we call the `subtractDays()` method with out JulianDate:

```
//if - is in String
else if(calc.getDisplayValue().contains("-")) {
    calc.setInput(myJD_1.subtractDays(Integer.valueOf(calc.getInput())).toString(sep));
}
```

The method looks like this:

```
public GregorDate subtractDays(int Days) throws Exception {
    //no negativ JulianDates allowed
    if(JulianDate.GregorToJulian(this.Day)-Days<0)
        throw new Exception("negativ JulianDate not exsistent");
    else
        return JulianDate.JulianToGregor(JulianDate.GregorToJulian(this.Day)-Days);
}
```

As an example, let’s try to subtract 5 days from todays date:

2019.12.13-
5

2019.12.13-5
2019.12.08

As we can see, our way of subtracting days from a gregorian date works as intended.

5. “Subtract two dates, giving the number of days between the dates.

Same here. We search for “x”. If we find it, we first initialize our second MyJulianDate to save the second input. Then we calculate the result and set it as our input:

```
//if x is in String
else if(calc.getDisplayValue().contains("x")) {
    myJD_2 = new MyJulianDate(new GregorDate(calc.getinput()));
    calc.setinput(myJD_1.daysBetween(myJD_2.getDay()));
}
```

Like this we can simply extend not only our whole interface to something like sets just by creating a new Interface, we also can extend our Date-Calcengine by more functions that we can map to unmapped Buttons like ^,(,).

For a final example, let’s try and calculate the number of days between today and the first of december this year:

2019.12.13x2019.12.01
12.0

As we can see, this part of our calculator also works as intended.

Reflections

BARTHOLOMÄUS

I can't say if I'm happy or not that we closed the curtains on our calculator with this lab. On one hand it was indeed fun to work on it over multiple labs instead of switching to a new topic every week, it was also very gratifying to see that the time we spent in our previous labs was worth it, as we combined our previous works with our current one. Except for the JulianDate one, which we had to rework a lot, compared to the Reverse Polish Notation one. On the other hand, I'm glad that we will finally move on to a different subject, since working on calculators for so long can get quite tedious over time.

JURI

This lab was third time that I've worked together with Bartholomäus. The first time was during the first Lab (nr°0), the second time was for the lab where we were allowed to choose our partners and now we got together due to the God of randomness. We worked more fluently and we minimised the time that we would normally need for a lab, by a lot. To be honest, for this week it was kinda needed because at the moment we got a lot of other stuff to do and we want to get finished with all of them. For me personally i think it's good to work in groups but maybe not changing them every week. In a company normally you also don't change your Team every week. Its normal that you need some time to get warm up with your partner and to get a nice workflow. I think is better to get the experience to get to the point where you have this and not starting every week again at o. Maybe a monthly change of partners would be a good balance between working with different people and getting the experience of creating a good workflow.

You can use the following link if you want to test the code:

<https://github.com/LuffyGM/Calculator2.git>

Appendix:

Calculator creation/handling and Interfaces:

ADT

GregorDate
JulianDate
MyJulianDate

Calculator

Calculator
Create_Calc
UserInterface
UserInterface_HEX
UserInterface_HEX_BIN
UserInterface_HEX_BIN_OKT
UserInterface_Date

Numeric systems

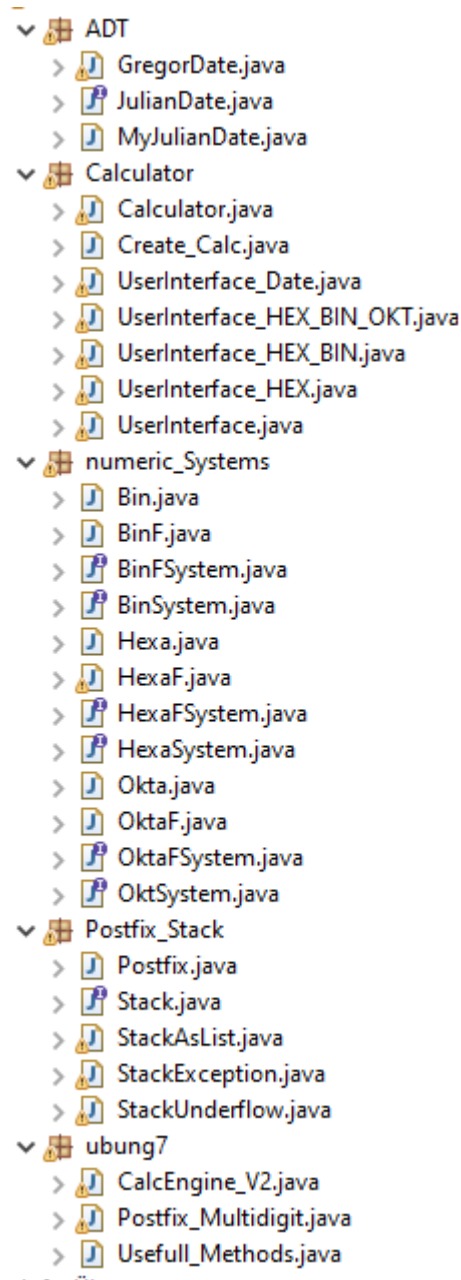
Bin
BinF
BinFSystem
BinSystem
Hexa HexaF
HexaFSystem
HexaSystem
Okta
OktaF
OktaFSystem
OktaSystem

Postfix-Stack

Postfix
Stack
StackAsList
StackException
StackUnderflow

Ubung7

CalcEngine_V2
Postfix_Multidigit
Usefull_Methods



```

package ADT;

import java.util.Calendar;
import java.util.GregorianCalendar;

import ubung7.Usefull_Methods;

public class GregorDate {
    private int day;
    private int month;
    private int year;
    public Boolean AC;

    //month start with 0
    private static Calendar cal;

    public GregorDate(int year, int month, int day, Boolean AC) {

        this.cal = Calendar.getInstance();
        this.AC = AC;

        setYear(year);
        setMonth(month);
        setDay(day);
    }
    public GregorDate(String Date) throws Exception {
        //can handle .;/;-;_
        String sep = "\\." + Usefull_Methods.getSep(Date);

        String[] Date_ = Date.split(sep);

        if(Date_.length != 3) {
            throw new Exception("Wrong Date input");
        }
        else {
            setYear(Integer.valueOf(Date_[0]));
            setMonth(Integer.valueOf(Date_[1]));
            setDay(Integer.valueOf(Date_[2]));
        }
    }

    public GregorDate() {

    }
    public String toString(String sep) {

```

```

String FirstLetterDay = ((this.day<10) ? "0" : "");
String FirstLetterMonth = ((this.month<10) ? "0" : "");
return Integer.toString(year)      +
        sep                        +
        FirstLetterMonth          +
        Integer.toString(month)    +
        sep + FirstLetterDay       +
        Integer.toString(day);
}
public static int MonthsDays(int month, int year) {

    if(!(0 < month && month < 13))
        throw new IllegalArgumentException("months out of
range");

    switch(month) {
    case 4:
        return 30;
    case 6:
        return 30;
    case 9:
        return 30;
    case 11:
        return 30;

    case 2:
        return ((isLeapYear(year)) ? 29 : 28);
    }
    return 31;
}
public GregorianCalendar toGregorianCalendar() {
    return new
GregorianCalendar(this.year,this.month,this.day);
}
//First Day is day 1
public int getDayOfTheYear() {

    int dayOfTheYear = 0;

    for(int i = 1; i<month;i++) {
        dayOfTheYear += MonthsDays(i, this.year);
    }
    return dayOfTheYear + day;
}
public static GregorDate get_current_date() {

```

```

        //Here we want month + 1 cuz we our Date start with
month 1
        GregorDate outcome = new
GregorDate(cal.get(Calendar.YEAR),cal.get(Calendar.MONTH) +
1,cal.get(Calendar.DAY_OF_MONTH),true);
        return outcome;
    }
    //Gives us the Month in which the Day of the year is.
    public static int getMonthOfDayOfTheYear(int day, int year) {
        int month = 1;
        for(; MonthsDays(month, year) < day; month++) {
            day -= MonthsDays(month, year);
        }

        return month;
    }
    //Gives us the Day of any month by input the Day of the Year
    public static int DayOfYearToDayOfMonth(int day, int year) {
        int month = 1;
        for(; MonthsDays(month, year) < day; month++) {
            day -= MonthsDays(month, year);
        }

        return day;
    }
    public static boolean isLeapYear(int year) {
        return ((year % 4 == 0 && year % 100 != 0) || (year %
400 == 0)) ? true : false);
    }
    //compares month and day of this Day and the param Day
    public boolean equalsMonth_Day(GregorDate Date2) {

        if(this.month == Date2.getMonth() && this.day ==
Date2.getDay())
            return true;
        else
            return false;
    }

    public int getDay() {

```

```

        return day;
    }
    public int getMonth() {
        return month;
    }
    public int getYear() {
        return year;
    }
    public void setDay(int day) {
        if(0 < day && day <= MonthsDays(month, year))
            this.day = day;
        else
            throw new IllegalArgumentException("days out of
range");
    }
    public void setMonth(int month) {
        if(0 < month && month < 13)
            this.month = month;
        else
            throw new IllegalArgumentException("months out of
range");
    }
    public void setYear(int year) {
        this.year = year;
    }
}
package ADT;

public interface JulianDate {

    static GregorDate FirstDay = new GregorDate(4712,1,1, false);

    public String get_Weekday();
    int daysBetween(GregorDate Day);
    GregorDate addDays(int Days) throws Exception;
    GregorDate subtractDays(int Days) throws Exception;

    public static GregorDate get_FirstDay() {
        return FirstDay;
    }
    public static GregorDate JulianToGregor(int JD) {

        GregorDate outcome = new GregorDate();

```

```

        int JD0 = 1721426;
        int N400 = (JD-JD0)/146097;
        int R400 = (JD-JD0)%146097;
        int N100 = R400/36524;
        int R100 = R400%36524;
        int N4 = R100/1461;
        int R4 = R100%1461;
        int N1 = R4/365;
        int LT = R4%365;

        int LJ = 400*N400 + 100*N100 + 4*N4 + N1;

        outcome.setYear(LJ+1);
        outcome.setMonth((LT+1)/30 + 1);
        outcome.setDay(GregorDate.DayOfYearToDayOfMonth(LT + 1,
outcome.getYear()));

        return outcome;
    }
    //https://de.wikipedia.org/wiki/Umrechnung_zwischen_julianisc
hem_Datum_und_gregorianischem_Kalender
    public static int GregorToJulian(GregorDate Day) {
        int JD;
        int JD0 = 1721426;
        int LJ = Day.getYear()-1;
        int N400 = LJ/400;
        int R400 = LJ%400;
        int N100 = R400/100;
        int R100 = R400%100;
        int N4 = R100/4;
        int N1 = R100%4;
        int LT = Day.getDayOfTheYear()-1;

        // -1 weil getDayOfTheYear bei 1 anfängt und nicht wie
in der Formel bei 0
        JD = JD0 + N400*146097 + N100*36524 + N4*1461 + N1*365
+ LT;

        return JD;
    }
}
package ADT;

```



```

public class MyJulianDate implements JulianDate {

    private GregorDate Day;
    public GregorDate getDay() {
        return Day;
    }
    public void setDay(GregorDate day) {
        Day = day;
    }

    public MyJulianDate(GregorDate Day) {
        this.Day = Day;
    }
    public MyJulianDate(int Day) {
        this.Day = JulianDate.JulianToGregor(Day);
    }

    @Override
    public int daysBetween(GregorDate Day) {

        int JDay1 = JulianDate.GregorToJulian(this.Day);
        int JDay2 = JulianDate.GregorToJulian(Day);
        return Math.abs(JDay2-JDay1);
    }
    @Override
    public String get_Weekday() {

        int Weekday = (JulianDate.GregorToJulian(Day)+1)%7;

        switch(Weekday) {
            case 0:
                return "Sunday";
            case 1:
                return "Monday";
            case 2:
                return "Tuesday";
            case 3:
                return "Wednesday";
            case 4:
                return "Thursday";
            case 5:
                return "Friday";
            case 6:
                return "Saturday";
        }
    }
}

```

```

        }
        return null;
    }

    @Override
    public GregorDate addDays(int Days) throws Exception {
        //no negativ JulianDates allowed
        if(JulianDate.GregorToJulian(this.Day)+Days<0)
            throw new Exception("negativ JulianDate not
existent");
        else
            return
JulianDate.JulianToGregor(JulianDate.GregorToJulian(this.Day)+Days)
;
    }
    @Override
    public GregorDate subtractDays(int Days) throws Exception {
        //no negativ JulianDates allowed
        if(JulianDate.GregorToJulian(this.Day)-Days<0)
            throw new Exception("negativ JulianDate not
existent");
        else
            return
JulianDate.JulianToGregor(JulianDate.GregorToJulian(this.Day)-
Days);
    }
}

```

```

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);
            if(mode == 5) gui = new UserInterface_Date(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;

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);
        Test = new Calculator(5);
    }
}

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();
}

/**
 * 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 occurred.
     */
    public void actionPerformed (ActionEvent event)
    {
        String command = event.getActionCommand();
        //Operator!

        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();
            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;

/**
 * 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 SideDisplays = new JPanel(new GridLayout(1, 2));
    JPanel Checkboxes = new JPanel(new GridLayout(5, 1));
    JPanel Displays = new JPanel(new GridLayout(5, 1));

    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]);
addCheckbox(Checkboxes, Boxes[1]);

Displays.add(DECdisplay);
Displays.add(HEXdisplay);

SideDisplays.add(Checkboxes);
SideDisplays.add(Displays);

Northside.add(SideDisplays);

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

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") ||
        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.BinFSsystem;
import numeric_Systems.Hexa;
import numeric_Systems.HexaFSsystem;
import numeric_Systems.OktaFSsystem;
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]);
        Displays.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("" +
BinFSsystem.DECtoBINF(Double.valueOf(calc.getinput())));
        }
        catch(Exception e) {
            Input_Display.setText("");
        }
    }

    protected void updateTextfields_BIN() {
        updateTextfields_HEX();
        try {
            BINdisplay.setText("" +
BinFSsystem.DECtoBINF(Double.valueOf(calc.getinput())));
        }
        catch(Exception e) {
            BINdisplay.setText("");
        }
    }

    protected void redisplay_BIN() {
        redisplay_HEX();
        if(Boxes[2].isSelected()) {
            display.setText("" +
BinFSsystem.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.Enumeraation;

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.BinFSsystem;
import numeric_Systems.Hexa;
import numeric_Systems.Okta;
import numeric_Systems.OktaFSsystem;
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]);
    }
}

```

```

        Displays.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();
    }
}

protected void setBase_OKT() {
    setBase_BIN();

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

protected void updateTextfields_OKT() {
    updateTextfields_BIN();

    try {
        OKTdisplay.setText("" +
OktaFSystem.DECtoOKTF(Double.valueOf(calc.getinput())));
    }
    catch(Exception e) {
        OKTdisplay.setText("");
    }
}

protected 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 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 ADT.GregorDate;
import ADT.JulianDate;
import ADT.MyJulianDate;
import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;
import numeric_Systems.BinFSsystem;
import numeric_Systems.Hexa;
import numeric_Systems.Okta;
import numeric_Systems.OktaFSsystem;
import ubung7.CalcEngine_V2;
import ubung7.Usefull_Methods;

public class UserInterface_Date extends UserInterface_HEX_BIN_OKT {

    protected ButtonGroup DateButtons = new ButtonGroup();

    MyJulianDate myJD_1;
    MyJulianDate myJD_2;
    String sep = ".";
    JTextField Datedisplay;

    public UserInterface_Date(CalcEngine_V2 engine, int
Amount_Checkboxes) {
        super(engine, Amount_Checkboxes);
        addLayout_Date();
    }

```

```

}
private void addLayout_Date() {

    Boxes[4] = new JCheckBox("DATE");

    Boxes[4].addActionListener(this);

    Datedisplay = new JTextField("0");

    Displays.add(Datedisplay);

    CheckBoxGrp.add(Boxes[4]);

    addCheckbox(Checkboxes, Boxes[4]);

    frame.pack();

    assignButtonGrps_Date();
    GreyButtons_Date();
}
protected void assignButtonGrps_Date() {

    assignButtonGrps_OKT();

    comp = buttonPanel.getComponents();

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

    if(isBetween(Integer.parseInt(((JButton)E).getText()), 0, 9))
        DateButtons.add((JButton)E);
    }
}
protected void GreyButtons_Date() {

    GreyButtons_OKT();

    if(Boxes[4].isSelected()) {
        Enumeration<AbstractButton> Date =
DateButtons.getElements();
        while(Date.hasMoreElements()) {
            JButton Button = (JButton) Date.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_Date();
            if(Base == 0)
            {
                calc.clear();
                redisplay_Date();
                reIndex_Display_Date();
            }
            GreyButtons_Date();
            redisplay_Date();
        }
        //numeric systems
        else if(Base > 0) {
            nummericCalc(command);
        }
        //Dates
        else if(Base == 0) {
            dateCalc(command);
        }
    }
    private void setBase_Date() {
        setBase_OKT();

        if(Boxes[4].isSelected()) {
            System.out.println("Date");
            Base = 0;
        }
        redisplay_Date();
    }
    private void updateTextfields_Date() {

    }
    private void redisplay_Date() {

```

```

        redisplay_OKT();
        if(Boxes[4].isSelected()) {
            display.setText(calc.getDisplayValue());
        }
    }
    protected void reIndex_Display_Date() {
        updateTextfields_Date();
        reIndex_Display_OKT();
        if(Boxes[4].isSelected()) {
            Input_Display.setText(calc.getinput());
            Datedisplay.setText(calc.getinput());
        }
    }
    private void numericCalc(String command) {
        //Operator!

        if(Usefull_Methods.CharisOperator(command.charAt(command.length()-1))) {
            exp = 0;
            calc.ADDtoDisplayValue(calc.getinput());
            // for "="
            if(command.equals("=")) {
                try {
                    calc.calcResult();
                } catch (StackUnderflow | StackException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                calc.setInput(Double.valueOf(calc.result));
                redisplay_Date();
                calc.setDisplayValue(calc.result);
                reIndex_Display_Date();
                calc.setInput(0);
            }
            //Everything else
            else {
                calc.ADDtoDisplayValue(command);
                calc.setInput(0);
                redisplay_Date();
            }
        }
        //Number
        else if(Usefull_Methods.StringisNumber(command)){

```

```

        AddNumber(command);
        if(exp<0)
            exp--;
    }
    else if(command.equals(".")) {
        exp = -1;
    }
    else if(command.equals("Clear")) {
        exp = 0;
        calc.clear();
        redisplay_Date();
    }
    else if(command.equals("?")) {
        showInfo();
    }
    if(!command.equals("=")) {
        reIndex_Display_Date();
    }
}

private void dateCalc(String command) {
    if(command.equals("Clear")) {
        calc.clear();
        redisplay_OKT();
    }
    else if(command.equals("?")) {
        showInfo();
    }
    //Operator!
    else
if(Usefull_Methods.CharisOperator(command.charAt(command.length()-
1))) {

        try {
            calc.ADDtoDisplayValue(calc.getinput());
            // for "="
            if(command.equals("=")) {

                //if + is in String
                if(calc.getDisplayValue().contains("+")) {

                    calc.setinput(myJD_1.addDays(Integer.valueOf(calc.getinput())
).toString(sep));

                }
                //if - is in String
                else if(calc.getDisplayValue().contains("-

```

```

")) {

    calc.setInput(myJD_1.subtractDays(Integer.valueOf(calc.getinput()))
        .toString(sep));
    }
    //if x is in String
    else
    if(calc.getDisplayValue().contains("x")) {
        myJD_2 = new MyJulianDate(new
        GregorDate(calc.getinput()));

        calc.setInput(myJD_1.daysBetween(myJD_2.getDay()));
        }
        redisplay_Date();
        calc.setDisplayValue(calc.getinput());
        reIndex_Display_Date();
        calc.setInput(0);
    }
    //Everything else
    else {
        myJD_1 = new MyJulianDate(new
        GregorDate(calc.getinput()));
        // / dont need = to work thats why we put
        it in here.

        if(command.equals("/")) {
            calc.setInput(myJD_1.get_Weekday());
            redisplay_Date();
            reIndex_Display_Date();
            calc.setInput(0);
        }
        else if(command.equals("(")||
            command.equals(")")||
            command.equals("^")){
            System.out.println("NOT MAPPED
COMMAND");
        }
        else {
            calc.ADDtoDisplayValue(command);
            calc.setInput(0);
            redisplay_Date();
        }
    }
} catch (Exception e) {}
}
//Date-parts

```

```
        else if(Usefull_Methods.StringisNumber(command) || command
== sep){
            calc.ADDtoinput(command);
        }
        if(!command.equals("=") && !command.equals("/")) {
            reIndex_Display_Date();
        }
    }
}
```

```

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 Multiplication(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 Multiplication(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 Multiplication(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 Multiplication(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 Multiplication(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 OktaSystem {
    public Okta Add(Okta OKT);
    public Okta Subtract(Okta OKT) throws Exception;
    public Okta Multiplication(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 Calculator.UserInterface_Date;
import Postfix_Stack.StackException;
import Postfix_Stack.StackUnderflow;

public class CalcEngine_V2 extends Postfix_Multidigit {

    private UserInterface_Date UIDate;
```

```

private String displayValue;
public String getDisplayValue() {
    return displayValue;
}
public void ADDtoDisplayValue(String AddString) {
    if(!AddString.equals("0.0"))
        displayValue += AddString;
    else
        System.out.println("OP after =");
}
public void setDisplayValue(String displayValue) {
    this.displayValue = displayValue;
}
private String input = "0";

public String getinput() {
    if (input == "0" || (input != "" && input.charAt(0) ==
'0' && input.charAt(1) == '.' && input.charAt(2) == '0'))
        return input = "0";
    else
        return input;
}
public void ADDtoinput(int AddNumber, int Base){
    if(!(input.equals("0.0") && AddNumber == 0)) {
        System.out.println(input);
        input =
Double.toString((Double.valueOf(input))*Base + AddNumber);
    }
}
public void ADDtoinput(String input){
    if (this.input == "0")
        this.input = input;
    else this.input += input;
}
public void setinput(double input) {
    this.input = Double.toString(input);
}
public void setinput(String 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,
StackOverflowException {
        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.StackOverflowException;
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 [] infixSplitted = infix.split("\\s+");

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

```

```

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;
                                }

                                public static String getSep(String input) {
                                    //if you want too really seaching for an sepperator u
                                    need a more complicated way, this is the bugded version
                                    if(input.contains("/"))
                                        return "/";
                                }

```

```
        else if(input.contains("-"))
            return "-";
        else if(input.contains("."))
            return ".";
        else if(input.contains("_"))
            return "_";
        return null;
    }
}
```