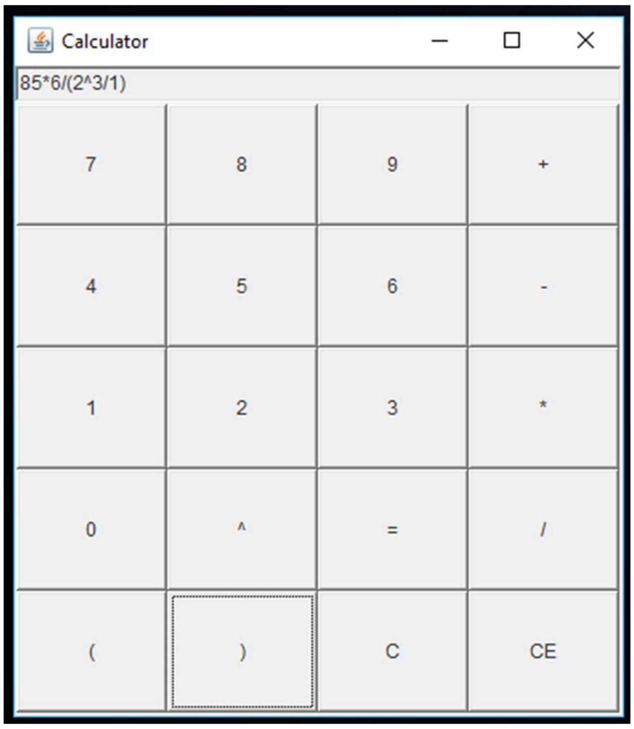
Assignment 1 - Expression Evaluator and Calculator GUI

By Mitul Savani

Product - Calculator



Summary

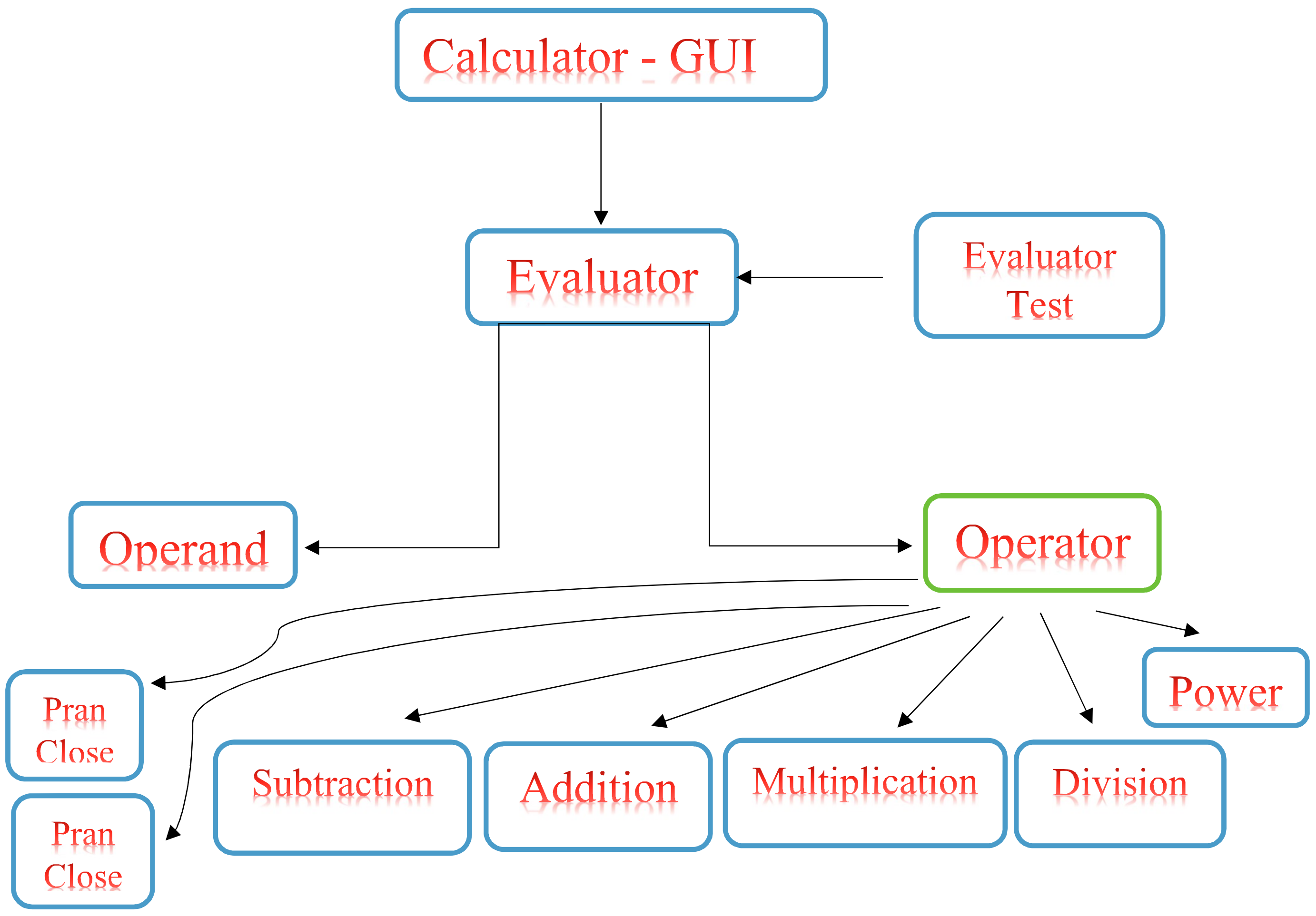
In this assignment the task was to make a Non-Scientific Calculator, which can take any length of expression with only constraint of being an Integer type. So trying to figure out how this product can be built was the hardest part, so I took couple of days to go over the skeleton code was given to me. After a while I started making very small intuitive flow chart in my mind which later helped me to design the code structure of our calculator. My main goal was to make design and architecture of this product very simple and handy such that if I want to add any other function to my Calculator later then it should give me accessibility to add functions quickly. So I used HashMap, abstract class and making different files for every different functions.

Execution

- I used "NETBEANS" as an IDE.
- In order to run program, just hit Run in *EvaluatorUI* file.

  Some test script are:
  - "2+3-5*((2-3)*2-5*2+3*(2-3-5-5*6)+4/2)*2-9"
  - (((5+5)*(5+5)) +(2+2)*(5*2))
  - (5+2)+(4*9)*(2+2)^2
  - 2+3*((5+2*2^2) +6-3*(6*2-5))*2-9

Architecture



BLUE – Classes
GREEN – Abstract class

I made this design, and declared data types *private, protected, and public* where ever it was required. I did not want to give my functions implementation file to user so I decided to make **Operator** as an abstract class.

I first started with class Operator and gave all the function a priority of execution which are as given below;

Priority Chart

Functions	Priority
PranOpen	1
PranClose	1
Addition	2
Subtraction	2
Multiplication	3
Division	3
Power	4

In operator classes:

- I implemented functions which can return the priority number to any caller class which is given below;

```
protected int priority()
{
    return (Appropriate Priority);
}
```

- Now the time was to make a function which take two number which we call as *operand* and perform appropriate execution. So, I am implemented a function;

```
protected Operand execute(Operand op1, Operand op2)
{
   return new Operand(op1.getValue() / op2.getValue());
}  //This is for division operator
```

- For Parentheses open and close, I return *null*(which means return nothing)

In operand class

- The main function in operand class was *check* function which can legit check if the input is and number inclusive from ‘0-9’. Functions is given below,

```
public static boolean check( String token )
{
   try
   {
          Integer.parseInt(token);
          return true;
   }
   catch(NumberFormatException e)
   {
          return false;
   }
}
```

Functions takes *token* as a *string* and return true/false.

In Evaluator Class

Evaluator class is a class which take a Mathematical expression and return the answer. Since skeleton code was already given so that make it little easy for us to understand how to approach this class and implement code in it. Let me start by telling you what are the things that I used to structure this class,

OperandStack -> which can store number from 0-9.
OperatorStack -> which can store operator like (,),-,+,/,*,^

Think of it as bucket where operand/operator can be lined up on each other.

Algorithm:

Lets us take an example to discuss the algorithm I came up with,
Eg: (2+3)

- Expression will be broken down like (,2,+,)
- I first check whether the token is operand or operator, then the task was to push the token into appropriate stacks. Since ‘(‘ is an operator so we push into the operatorStack().
- Now it was 2, which is interprets as an operand, so push it to operandStack().
- The token is now ‘+’, so we compare it with the priority of the previous operator. Logic was if the previous operator’s priority is greater than the current than we perform execution by poping(taking out) two operand from operandStack() and calling respective operation class to get an answer.
- After getting the answer we push the answer back into the operandStack().
- When code catches a token ‘)’ then it will call an other function which will pop operand and operator until ‘)’ is found and will perform execution on the operands.
- This loop will stop when there is no token found.
- Now, it was time to perform operation on whatever operand and operators are left in the stacks.

Some of the important loops in *Evaluator* class are shown below:

Open Parentheses token:

```
else if("(".equals(token))
{
   operatorStack.push(new PranOpen());
}
```

Discussion : I atomatically added ‘(‘ into operatorStack() when I detected that operator.

Close Parentheses token:

```
private void PerformParenthesesOperation()
{
      while((operatorStack.peek().priority()!=1))
      {
          Operator lastOperator = operatorStack.pop();
          Operand op2 = operandStack.pop();
          Operand op1 = operandStack.pop();

       operandStack.push(lastOperator.execute(op1op2));
      }
      operatorStack.pop();
}
```

Discussion : Logic in this loop is I Performed operation by poping out items from stacks until I see operator ‘)’.

In EvaluatorUI function(GUI):

This class is to connect the working good with a display of calculator. Skeleton code was given to us where we were asked to asked to implement *actionPerformed()* function.

- The way I approached this class was I first looked at the given code and tried to grasp the tools that was given to us (i.e) buttons and design of calculator.

- I checked which command is being pressed, code is shown below;

```
String command = arg0.getActionCommand();
```

- Then I made a switch case in *actionPerformed(),* which checks which buttons is being pressed.

  For an example of ‘+’ button;

```
case "+":
  txField.setText(txField.getText()+command);
  break;
```

Discussion: I basically got what is in the text display and added an ‘+’ command to it.

Difficulties I faced:

- There were tons of difficulties I faced while implementing all the classes I discussed above,
- At first my algorithm was not correct.
- I got EmptyStackException error.
- I was not able to figure out how parentheses should be handles.

After getting numbers of errors, I started debugging my code and tried to see where the actual problem was. I always started printing out what operation are being executed along with its answer on console. Moreover, I sometimes tried to find the reason behind my problems on Stack Overflow

I didn’t used # sign in any of my implementation file.

References

Stack Overflow
Link: https://stackoverflow.com/