|  |  |
| --- | --- |
| **Part 1 – The Main method** | |
| Tell the user to input infix carefully  Prompt the user to type the Infix Scan the infix and strore it to a String array  Check if the infix is valid  IF the infix is valid,  LOOP starts here  Let the user choose an operation to do  CASE: Postfix is chosen  Program will delay  Contents of the array will be shown  Determination of Precedence of each element  Program delays again  Transform the infix to Postfix  CASE: Prefix is chosen   Program will delay  Infix will be reversed  Store the reversed infix to a String array  Contents of the array will be shown  Determination of Precedence of each element  Program delays again  Transform the infix to Prefix  Program delays once again  Prefix string will be reversed and shown  DEFAULT: meaning choice input is wrong   Tell the user to reset the program  WHILE the user wants to continue, Loop the switch case  END LOOP  ELSE: the infix is not valid  Tell user to reset the program  Thank the user and show the developers of the program | Print a message  By using the scanner “in”  CALL the *Array\_InfixS* method  CALL the *Infix\_Checker* method  In this case, *Infix\_Checker* returned proceed as true  Do  Choices here are to convert to Postfix or Prefix  Switch case is used here. For now, choice = 1  CALL *Dots* method  CALL the *display\_Filtered\_Infix* method  CALL *PrecedenceLvlProvider* which returns an int array of plvls  CALL *Dots* method  CALL the *PostFix* and use the infix array and precedence array  Switch case is used here. This time, choice = 2  CALL *Dots* method  CALL *Reverse\_Infix ­*method(from now on, reversed infix is the basis of all)  CALL *Array\_InfixS* ( in this case, the reversed infix is now being used ) CALL *display\_Filtered\_Infix*  CALL *PrecedenceLvlProvider* which returns an int array of plvls  CALL *Dots* method  CALL the *PreFix* and use the infix array and precedence array  CALL *Dots* method  CALL *Reverse\_Infix* that CALLS *Array\_InfixS* as its parameter \*tricky\*  Switch Default if choice is not 1 nor 2  Print a message  While choice is 1, Loop again  End loop if choice is 2  In this case, *Infix\_Checker* returned proceed as false  Print a message  Print a message ( M A M ) |
| **Part 2 – Dots method** | |
| \*This uses the “java.util.concurrent.TimeUnit” package  Simply prints 1 dot for 1 second and is repeated 3x which imitates delay/loading of the program | Code is surrounded by try catch  TimeUnit.SECONDS.sleep(1) for delayinng the program by 1 second before executing next lines |
| **Part 3 – Array\_InfixS method** | |
| Creates a String array  Stores the elements of the string to the String array  Returns the String array | In here, it gets a string  Element is determined when a space is encoutered [.split(“ ”)]  Its purpose is to store the elements of the infix to an array for further evaluation |
| **Part 4 –Infix\_Checker method** | |
| Determine the Precedence lvls for each element  Proceed is initialized as **true**  *For statement 120..136*  Create a limiter based on the number of elements excluding ( )s  Limiter serves as the size of the new filtered plvl array w/o ( )s  Store the plvls of the array Infix to the filtered array  *For statement 138..155*  Proceed will be set as **false** when:  a. if the beginning of the infix is an operator  b. if the element next to the operand is also an operand  c. if the element next to the operator is also an operator  *For statement 157..170*  Proceed will be set as **false** when:  a. if an operator is next to an open parenthesis [ ( \* ]  *element\_kind[count] array[count-1]*  b. if an there is an operator before closed parenthesis [\*)]  *element\_kind[count] array[count-1]*  *For Statement 172..189*  IF the above condition is not ment  Get the number of parentheses  Also get number of open and close parentheses  Store the parenthesis to a new parenthesis array  Proceed will be false if:   1. open parenthesis and close parenthesis is not equal 2. if close parenthesis is at the beginning or open parenthesis is at the end.   Lastly *Statement 217*  Return the proceed | CALL *PrecedenceLvlProvider*  This will soon become false when a conditon is met later on  \*In here we are now talking about the Precedence levels only  Done by using FOR loops and excluding ( )s which is set as a precedence level of 4 afterwards these elements is stored to a new array called “element\_kind\_noparenthesis”  \*This is the purpose why we get the filtered precedence lvls first [w/o ()s]  Array[0] is the begginning  Operands are determined having a precedence lvl = 0  Operators are determind having a precedence lvl 1..3  \*In here, it checks elements next to the parenthesis  The for loop both use the int plvl array and the String array  IF operator [plvl = 1..3] is after an open parenthesis  IF operator [plvl = 1..3] and next to it is close parenthesis  \*This will now focus on the parenthesis  Meaning that proceed is still true and there’s no mistake in operators and operands  Use FOR loops where and when the array[count] is a parenthesis  It now increments the number of parenthesis [index]  As well as the number of open and close parenthesis  This is to know the order of the parenthesis  If open parenthesis != close Parenthesis [2 : 1 ]  If Infix ) A + B (  In this case, if any condition is met, proceed is false making the infix invalid  But if there are no invalidities, proceed will remain true and be returned |
| **Part 5 – PrecedenceLvlProvider method** | |
| Create a new int array with size of the infix array  This checks the precedence level of each element  Precedence level if provided when and stored to the int array:  4 – when the scanned element is a parenthesis  3 – when the scanned element is a “^”  2 – when the scanned element is a “\*” or “/”  1 – when the scanned element is a “+” or “-”  0 – when the scanned element is any character or string  Return the int array | This array serves as the precedence level of each element  FOR loop is used to check each element of the string array  These numbers is stored to the int array  It does it like this:  ( A + B )  4 0 1 0 4 |
| **Part 5 – PrecedenceLvlProvider method** | |
|  |  |