```
67      /**
68       * boolean isUnaryMinus(char curr, char prev, int i):
69       * @param curr, the current character
70       * @param prev, the previous character
71       * @param i, whether the character is the first character in a string
72       * @return a boolean, whether a minus is a binary unus
73       */
74      boolean isUnaryMinus(char curr, char prev, int i){
75          //For unary minus either the preceding character is an operator, or open paranthesis or it is the 1st character in the string
76          return curr == '-' && (i==0 || getPrecedence(prev) > 0 || prev == '(');
77      }
78
79      /**
80       * private void createPostfixListfromInfix(String s):
81       * The algorithm populates the private ArrayList of strings in PostFix from from input Infix String as follows:
82       * If the character is operand, append character to the current element of string in ArrayList
83       * If the character is operator:
84       * 1. If the stack is empty push this operator into the stack.
85       * 2. Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the current operator.
86       *    Then push the current operator to the stack.
87       * 3. If character is openings paranthesis '(', push it into the stack.
88       * 4. If character is closing paranthesis ')', pop all elements from the stack until matching '(' is reached while adding each character
89       *    into the ArrayList as a separate element.
90       * @param: s, the string we want to create a Postfix list out of
91       */
92  @  private void createPostfixListfromInfix(String s) {
93          MyStack<Character> mystack = new MyStack<>();
94          char prevChar = '\0';
95          boolean inNumSeqFlag = false;
96          //loop through the string s
97          for (int i = 0; i < s.length(); i++) {
```

Run: ArithmeticExpressionEvaluator

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home/bin/java ...
********************************************************************************************************
********************** ARITHMETIC EXPRESSION CALCULATOR ************************************************
********************************************************************************************************
Please enter an Infix expression to convert to Postfix and evaluate at the prompt with a RETURN or
ENTER key at the end of the expression. For example (41-3*(85/4))
You can use any valid positive or negative decimal numbers for the operands. The basic arithmetic
operations addition(+), subtraction(-), multiplication(*), division(/), or
exponent(^) are allowed. You can also use any whitespace characters space, tab etc.
********************************************************************************************************
Enter Infix Expression> $123.45*UUI/34567(
Invalid input. Please enter a valid infix expression which contains only valid decimal numbers (0-9) and valid arithmetic operators (+, -, *, / , ^)  and parenthesis ()
Enter Infix Expression> (53.45 + (23.4*9))(
Invalid input. Please enter a valid infix expression which contains balanced parenthesis
Enter Infix Expression> (10.25  ^4 - 23+4.9*(53/2500^0.5))
You Entered> (10.25  ^4 - 23+4.9*(53/2500^0.5))
Postfix Expression>  10.25 4 ^ 23 - 4.9 53 2500 0.5 ^ / * +
Calculator Answer> 11020.323
********************************************************************************************************
Do more calculations? (Enter Y for yes and any other key to stop)> Y
Enter Infix Expression> (-5^20  *-45*23+(-51*2.78^4/-53))
You Entered> (-5^20     *-45*23+(-51*2.78^4/-53))
Postfix Expression>  -5 20 ^ -45 * 23 * -51 2.78 4 ^ * -53 / +
Calculator Answer> -98705291748046816
********************************************************************************************************
Do more calculations? (Enter Y for yes and any other key to stop)> n

Process finished with exit code 0
```