Group 1

Comprehensive Programming
Section - 2

Group Members

- Imraul Emmaka (Leader)
- Mustansir Nawaz aka Mukhi
- SM Shamim Al Mamun aka Sam
- Didarul Islam
- Mohammad Kemal
- Seif Mugisha

Task:

- Write a program that evaluates fully parenthesized arithmetic expressions using the standard operators for addition, subtraction, multiplication, and division, and a special operator for exponentiation ("^").
- (18.4 - ((2.3 * 8.5) / (19.5 + (2.7 ^ 4.9))))

Steps:

- Tokenize the input/expression – StringSplitter Class
- Evaluation: Shunting-Yard Algorithm

Procedure in Brief:

- Take input using Scanner and put it into a String variable
- Take an instance of StringSplitter to tokenize the input
- NumberStack = new Stack(), symbolStack = new Stack();
- Evaluate the expression
- If no error encountered, print result
- Else print Illegal Expression

StringSplitter Class:

- Fields:
  - Private Queue<Character> characters;
  - Private String token;
  - Constant String SPECIAL_CHARS=”()+-*/^”;
- Constructor:
  - Parameter: String line
  - Initialize character
  - Put each char of line into characters queue
- Methods:
  - HasNext()
    - if there’s no more token returns false, otherwise true
  - Next()
    - Returns the next token
  - findNextToken()

FindNextToken() or Tokenization:

- Few Ground Rules:
  - No white space
  - If it belongs to any of our special characters, it’s a token
  - Numbers together – one token
- Peek if the first elem of the queue is a whitespace
  - If so - remove()

//the queue could be empty at this point

- If the queue is empty : token = null

FindNextToken()

- Else
  - Ch = first element of the queue

  //we’ll set token as ch, but before that we need check if ch is legal

  //according to our problem statement
  - If ch is a digit or a dot or a SPECIAL_CHARACTER
    - Initialize token as ch;
  - Else print(ch is not legal!!) and throw IllegalArgumentException
  - If (token is not a SPECIAL_CHAR)//then it’s a num
    - Boolean done = false; //we’re done when we took all the contiguous digits as a token
    - while(queue is not empty and not done)
      - Ch2 = queue.peek();
      - If ch2 is a whitespace or special_char: done = true
      - Else if ch2 is a digit or a dot: token = token+queue.remove()
      - Else print(ch2 is not legal!!) and throw IllegalArgumentException

      return token;

Evaluation() from client class

- Does two things:
  - Evaluates the expression and returns false if no error encountered
  - Returns true if any error encountered
- Formal parameters:
  - StringSplitter splitter, Stack numStack, Stack symbolStack
- Returns boolean err_flag

Evaluation() from client class

- Boolean err_flag = false;
- While no error and there’s more token:
  - Str = next token
    - If str isNumereic: s=numStack.push(str);
  - Else
    - If str is “(” or an operator: symbolStack.push(str)
    - Else if str is “)”: //go on and evaluate the sub-expr
      - Num2=numStack.pop(), num1 =numStack.pop();
      - Operator = symbolStack,pop();
      - If operator is not any of “+-*/^”:
        - println(operator is not legal!!), println(Error n parenthesis!!)
        - Return true
      - Result = calculate(num1, num2, operator)
      - NumStack.push(result)
      - If symbolStack not empty: symbolStack.pop()
    - Else err_flag = true;
- Return err_flag

Other Methods in Client Class

- IsNumeric(String str):
  - Return str.matches("-?\\d+(\\.\\d+)?");
- Calculate(double num1, double num2, String operator, boolean err_flag):
  - Result = num1 operator num2;

Thanks Everyone for Listening!!