**Regarding coding**

1. **I have used Parallel stream for multitasking, if user add nth number of exp strings to evaluate so on click on calculate button it will use all cores/processor to achieve result which is much faster than manual.**
2. **I have implemented two approach to calculate**

A>

**public** **void** bodmasEvalByManual(ExpEntity ent) {

**try** {

**char**[] tokens = ent.getExpString().toCharArray();

// Stack for numbers: 'values'

Stack<Integer> values = **new** Stack<Integer>();

// Stack for Operators: 'ops'

Stack<Character> ops = **new** Stack<Character>();

**for** (**int** i = 0; i < tokens.length; i++)

{

// Current token is a whitespace, skip it

**if** (tokens[i] == ' ')

**continue**;

// Current token is a number, push it to stack for numbers

**if** (tokens[i] >= '0' && tokens[i] <= '9')

{

StringBuffer sbuf = **new** StringBuffer();

// There may be more than one digits in number

**while** (i < tokens.length && tokens[i] >= '0' && tokens[i] <= '9')

sbuf.append(tokens[i++]);

values.push(Integer.*parseInt*(sbuf.toString()));

}

// Current token is an opening brace, push it to 'ops'

**else** **if** (tokens[i] == '(')

ops.push(tokens[i]);

// Closing brace encountered, solve entire brace

**else** **if** (tokens[i] == ')')

{

**while** (ops.peek() != '(')

values.push(*applyOp*(ops.pop(), values.pop(), values.pop()));

ops.pop();

}

// Current token is an operator.

**else** **if** (tokens[i] == '+' || tokens[i] == '-' ||

tokens[i] == '\*' || tokens[i] == '/')

{

// While top of 'ops' has same or greater precedence to current

// token, which is an operator. Apply operator on top of 'ops'

// to top two elements in values stack

**while** (!ops.empty() && *hasPrecedence*(tokens[i], ops.peek()))

values.push(*applyOp*(ops.pop(), values.pop(), values.pop()));

// Push current token to 'ops'.

ops.push(tokens[i]);

}

}

// Entire expression has been parsed at this point, apply remaining

// ops to remaining values

**while** (!ops.empty())

values.push(*applyOp*(ops.pop(), values.pop(), values.pop()));

// Top of 'values' contains result, return it

ent.setOutput(values.pop()+"");

**this**.createOrUpdate(ent);

} **catch** (Exception e) {

e.printStackTrace();

}

}

// Returns true if 'op2' has higher or same precedence as 'op1',

// otherwise returns false.

**public** **static** **boolean** hasPrecedence(**char** op1, **char** op2)

{

**if** (op2 == '(' || op2 == ')')

**return** **false**;

**if** ((op1 == '\*' || op1 == '/') && (op2 == '+' || op2 == '-'))

**return** **false**;

**else**

**return** **true**;

}

// A utility method to apply an operator 'op' on operands 'a'

// and 'b'. Return the result.

**public** **static** **int** applyOp(**char** op, **int** b, **int** a)

{

**switch** (op)

{

**case** '+':

**return** a + b;

**case** '-':

**return** a - b;

**case** '\*':

**return** a \* b;

**case** '/':

**if** (b == 0)

**throw** **new**

UnsupportedOperationException("Cannot divide by zero");

**return** a / b;

}

**return** 0;

}

B>

ScriptEngineManager manager = **new** ScriptEngineManager();

ScriptEngine engine = manager.getEngineByName("js");

System.***out***.println(engine.eval(ent.getExpString()));

ent.setOutput(engine.eval(ent.getExpString())+"");

**this**.createOrUpdate(ent);

1. **All said points have been covered**
2. Use java 8 or above
3. Please demonstrate the use of collections, streams, and lambdas
4. Please include JUnit test case(s)
5. Once you complete the java solution, please convert it into a web application (war file) that accepts input using HTML form and prints results
6. For converting the solution **must include** use of following technologies

 -Spring MVC or Spring Boot

 -JSP and JSP tags …

 -spring-tags (form:input, form:hidden, form:checkbox etc.)

 -Demonstrate use of Javascript/JQuery and JUnit

 -Demonstrate the use of CSS3 and HTML5 …

 -Spring annotations - @Service, @Repository, @Entity etc. …

 -JPA/Hibernate with h2 dB v …

 -Maven POM for building …

1. Please provide a README on how to build and run it
2. Please upload your solution to your **GitHub account and provide the link.**