Diagram Notes:

Stack Class:

- A stack class is created to contain its own methods in order to manipulate its own data using basic stack operations rather than treating it as a means to just store data

Cannot exist without Expression class (composition)

Expression Class:

- Expression class will create instances of operandStack and operatorStack as to manage that specific expression

- Assume that expression has the default methods **setExp(exp) and getExp()** so that we can hypothetically receive a command line argument and set the value of expression which can later be retrieved through the getter

- Assume private level protection which requires getter and setter.

Calculator Class:

- Reduced original python logic into more digestible operations such as add, sub, div, and mult which are methods used by evaluateExpression()

- The method judgePriority(char) was deemed necessary in our view of an OO design as it would be useful in the main method evaluateExpression()

- This method should effectively use the Expression class and the operand/ operator stacks stored within it to parse the expression and then use its own methods to perform the actual calculation.

- In the event there is no expression "attached" to our Calculator we would hypothetically have an "if null" condition which returns some default value

- You can imagine the other methods still offer individual functionality if a user wants to manually perform an operation rather than letting it be parsed via command line.

Has a relationship with Expression: If you "throw away" expression, calculator can be connected to a new one if desired; else it will evaluate a base case expression and return 0 (Aggregation)

Unary/BinaryCalc Classes:

- In the context of a calculator we can also create the classes BinaryCalc and UnaryCalc to process simple expressions such as "1 + 5" or "-(7)" etc.

- This has no purpose in relation to calculator's operation but rather is an optional inheritance of the Calculator class if a user would like to only input a specific kind of expression

- In practice, these should have a smaller runtime and might be preferred if the user needs to process a large amount of calculations at a time

This is an example of inheritance where these subclasses have access to all the methods of Calculator but are more specialized and offer better functionality depending on the user's needs