Lab02: Exception Handling

CSCI: Object-Oriented Programming

October 10, 2022

1 Introduction

In this lab, you will implement a simple calculator program. The program would take an expression (upto to two operands, and one operator) from the command prompt and returns the result of the evaluated expression. In case runtime error occurs, your program will catch the exception error, and provide feedback message to the user.

2 Topics Covered

- Handle Exceptions using try/catch/finally block
- Implement Custom Exceptions

3 Client Input

The user will be asked to enter an expression. Each line that is typed by the user is interpreted as a potential expression. Valid expressions would constitute one, two or three tokens (each token is separated from the preceding token by **one spaces**), and may take one of the following forms:

- Takes 1 token: If the token entered is **quit**, the program responds by exiting
- Takes 2 token: If the token entered contains first token as ”+” and second token as an integer than the progam responds by displaying the positive number of the given integer. For eg:

  ```
  Enter an expression: + 4
  The result is: +4
  ```

- Takes 3 tokens: If the first and third tokens are integer numbers, and the middle token is an operator (“*” or “/”) only, the program responds by displaying the result of the expression.

Any illegal integer operation or not following one of above format, would result in the display of a specific error message.

is the sample output:

```
4 * 2
The result is: 8
Input was: 4 * 2
42*7
Illegal input: Illegal Argument
Input was: 42*7
4 / 2
The result is: 2
Input was: 4 / 2
foo * 2
Illegal input: Illegal Argument
Input was: foo * 2
42 ^ 3
Illegal input: Illegal Operator
Input was: 42 ^ 3

Illegal input: Illegal Argument
Input was:
32 ^ baz
Illegal input: Illegal Argument
Input was: 32 ^ baz
foobar ^ 3
Illegal input: Illegal Argument
Input was: foobar ^ 3
4 / 0
Tried to divide by zero
Input was: 4 / 0
+4
Illegal input: Illegal Argument
Input was: +4
+ 4
The result is: +4
Input was: + 4
1 * 2 * 3
Illegal input: Illegal Token Length
Input was: 1 * 2 * 3
45* 2
Illegal input: Illegal Operator
Input was: 45* 2
quit
Quitting
Input was: quit
```

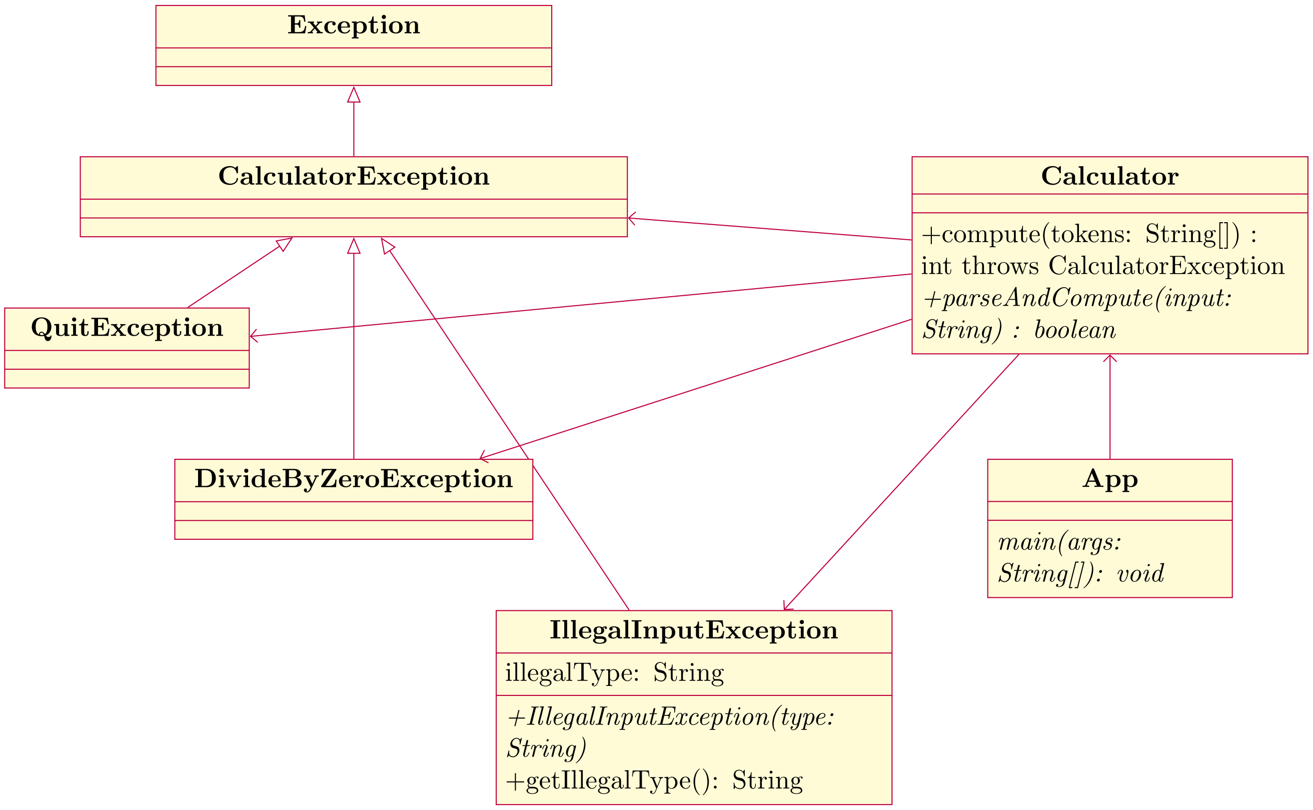
4 Design

Below is the UML diagram for the set of classes that you will implement for this lab. The *Exception* class is provided by the Java API. You are required to write the **CalculatorException** class and all of its subclasses. By having a custom Exception type, we enable the program to pass more detailed information about what errors have occurred, just by having a more specific class. Below is the description of what kind of exceptions the following subclasses would handle:

- QuitException: thorwn when the user inputs "quit" to end the program. (case insensitive).
- DivideByZeroException: thrown when the program attempts to divide by 0.
- IllegalInputException: thrown when the input does not match with a format that we expect. This exception also has a private String variable *illegalType* and a getter method (*getIllegalType()*) for the variable. *illegalType* is set through IllegalInputException constructor and gives some more detail about what kind of input error occurred. Various values that *illegalType* can take are:
  - "Illegal Token Length": when the number of tokens is neither 1, 2, or 3
  - "Illegal Token Length": when a token does not match the type of token expected in it's position. For example: in the input "3 * foo", "foo" is an illegal argument, as it is not an int
  - "Illegal Operator": when a token in an operator position is not supported by the program. The program only accepted "*" and "/" operators for binary operations. So, " 2 - 1" would give error.

The *Calculator* class also has two methods:

- *parseAndCompute(String input)*
  1. This method splits the string into array of tokens (separated by space)
  2. Calls *compute* method to evaluate the expression
  3. Prints out one line with the result or an error message. What error message is printed is dependent on what exception type is caught. The messages will be the following:
     - If no message is caught, then the message will be **"The result is" + result**
     - If QuitException is caught, then the message will be **"Quitting"**
     - If IllegalInputException is caught, then the message will be **"Illegal input: " + e.getIllegalType()**
     - If CalculatorException is caught, then the message will be **"Tried to divide by zero"**, since *DivideByZeroException* is the only one left, and since it's supertype is CalculatorException, it can be caught with *CalculatorException* type.
  4. Prints out a second line with what the input was, even when the method is returning. (hint: use finally block)
  5. Returns a boolean indicate whether the program should terminate.


Exception
CalculatorException
Calculator
+compute(tokens: String[]) : int throws CalculatorException
+parseAndCompute(input: String) : boolean
QuitException
DivideByZeroException
App
main(args: String[]): void
IllegalInputException
illegalType: String
+IllegalInputException(type: String)
+getIllegalType(): String

The *compute()* method is responsible for evaluating the expression and if it's can't then throw a **CalculatorException** exception (any one of *QuitException*, *DivideByZeroException*, *IllegalInputException*) to it's caller method. (Hint: Remember that the *compute()* method throws checked exception, therefore, you should declare it properly). When there is exactly one token that is equal to the String "quit" (case insensitive), a *QuitException* is thrown.

5 Grading

- QuitException: 10 points
- DivideByZeroException: 10 points
- IllegalInputException: 10 points
- CalculatorException: 10 points
- parseAndCompute method: 25 points
- calculate method: 25 points
- design and style: 10 points

6 Instructions

- Please download the skeleton code for this lab from D2L.
- Please upload the lab in your github repo and add me as collaborator.
- There should be multiple catch blocks to handle each catch, except DivideByZeroException which will be handled by CalculatorException itself.