Exception Handling -Assignments

You are required to compute the power of a number by implementing a calculator. Create a class MyCalculator which consists of a single method long power(int, int). This method takes two integers, n and p, as parameters and finds np. If either n or p is negative, then the method must throw an exception which says "n or p should not be zero". Also, if both and are zero, then the method must throw an exception which says "n and p should not be zero"  
For example, -4 and -5 would result in. java.lang.Exception: n or p should not be negative.  
Complete the function power in class MyCalculator and return the appropriate result after the power operation or an appropriate exception as detailed above.  
Input Format  
Each line of the input contains two integers, and . The locked stub code in the editor reads the input and sends the values to the method as parameters.  
Constraints  
• -10 <=n<=10  
• -10 <=p<=10  
Output Format  
Each line of the output contains the result np, if both n and p are positive. If either n or p is negative, the output contains "n and p should be non-negative". If both n and p are zero, the output contains "n and p should not be zero.".  
Sample Input   
3 5  
2 4  
0 0  
-1 -2  
-1 3  
Sample Output  
243  
16  
java.lang.Exception: n and p should not be zero.  
java.lang.Exception: n or p should not be negative.  
java.lang.Exception: n or p should not be negative.  
Explanation   
• In the first two cases, both and are postive. So, the power function returns the answer correctly.  
• In the third case, both and are zero. So, the exception, "n and p should not be zero.", is printed.  
• In the last two cases, at least one out of and is negative. So, the exception, "n or p should not be negative.", is printed for these two cases.

2. Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception “AgeNotWithinRangeException”. If name contains numbers or special symbols raise exception “NameNotValidException”. Define the two exception classes.

3. Define class MyDate with members day, month, year. Define default and parameterized constructors. Accept 7 values from the command line and create a date object. Throw user defined exceptions – “InvalidDayException” or “InvalidMonthException” if the day and month are invalid. If the date is valid, display message “Valid date”.

4. write a class Factorial which has service method int getFactorial(int num) which returns the factorial of a number num.  
  
if the number given by the user is less than 2 then throw InvalidInputException.  
  
find what is the highest value of int and if the factorial answer exceeds the highest value then throw FactorialException to the caller.

5.

|  |
| --- |
| **Hands-on Exercise** |
| After completing the hands-on exercises, you will be able to:   * Create Custom Exceptions and handle it using Try/Catch/Finally blocks. |

**Problem Statement 1:**

Develop a simple application to calculate the tax for a particular employee based on his salary.

Develop a class named “**TaxCalculator“** with a method named **calculateTax** with the following method parameters,

|  |  |
| --- | --- |
| Variable Name | Data Type |
| empName | String |
| isIndian | boolean |
| empSal | double |

This method should return a double taxAmount.

The business logic for calculating the tax is as follows, this has to be implemented inside the method,

If the employee is not a Indian

The calculator should throw a ***CountryNotValidException***

If the employee name is null or empty

The calculator should throw a ***EmployeeNameInvalidException***

If empSal is greater than one lakh and isIndian true

taxAmount =empSal \*8/100

Otherwise If empSal is between 50K and 1lakh and isIndian true

taxAmount =empSal \*6/100

Otherwise If empSal is between 30 and 50 Thousand and isIndian true

taxAmount =empSal \*5/100

Otherwise If empSal is between 10 and 30 Thousand and isIndian true

taxAmount =empSal \*4/100

Otherwise

The calculator should throw a ***TaxNotEligibleException.***

Develop a main class ***CalculatorSimulator***  , implement the following logic in main method

1. Execute the calculateTax Method and print the tax amount as **“Tax amount is “<Tax Amount>**
2. In case the calculateTaxMethod throws exceptions, this method needs to catch the appropriate exception print the stack trace and display the following messages,
   1. **Country not valid:**  “**The employee should be an Indian citizen for calculating tax**”
   2. **Employee name not valid:**  “**The employee name cannot be empty**”
   3. **Not eligible for Tax calculation:**  “**The employee does not need to pay tax**”

The following test cases to be executed, change the data in main method and run it and verify the output messages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **Employee Name** | **Employee Salary** | **Is Indian** | **Message Expected** |
| Test Case 1 | Ron | 34000 | **False** | **The employee should be an Indian citizen for calculating tax.** |
| Test Case 2 | Tim | 1000 | **True** | **The employee does not need to pay tax** |
| Test Case 3 | Jack | 55000 | **True** | **Tax amount is 3300** |
| Test Case 4 |  | 30000 | **True** | **The employee name cannot be empty** |

6. Design and implement a program that creates an exception class called StringTooLongException, designed to be thrown when a string is discovered that has too many characters in it. In the main driver of the program, read strings from the user until the user enters “DONE”. If a string is entered that has too many characters (say 20), throw the exception. Allow the thrown exception to terminate the program.

7. Write a program that asks the user to input a set of floating-point values. When the user enters a value that is not a number, give the user a second chance to enter the value. After two chances, quit reading input. Add all correctly specified values and print the sum when the user is done entering data. Use exception handling to detect improper inputs.