**Lab 9** (25 points)

There are several learning objectives to this assignment

* Creating Interface
* Creating Class that implements Interface
* Creating an exception
* Using exceptions to handle user input errors and divide by zero errors
* Creating a driver class to test your class

Using object oriented design:

1. Create both RationalInterface and Rational class that implements RationalInterface with the following instance params (int numerator, int denom, double result) and public methods
   1. doRational()–primary method in Rational class. Does not return anything (void):
      1. Parameters – none
      2. Throws InputMismatchException and DivideByZeroException (passes the buck to the calling method)
      3. Flow of method - Calls setUserInput() individually to first set inst var numerator and then second set instance var denom. After setting numerator and denom, doRational() calls calcRational(). Instance var result is assigned the output of calcRational() and the following is written to output - “With numerator <numerator> and denominator <denom>, the result is <result>”. doRational() should then ask is the user wants to calculate another number (“enter y for yes or n for no”). The coder should ensure that y, Y, n, or N are valid responses. HINT: most of the code of doRational will need to run in a do-while loop
   2. setUserInput() – Returns an integer that assigns the result of setUserInput() to either numerator or denom instance vars. NOTE: This is public for the purpose of this lab to be included in the interface, but technically could be a private helper method
      1. Parameters – none
      2. Throws InputMismatchException (passes the buck to the calling method)
      3. Flow of method – asks user for integer input and returns an integer to the calling method.
   3. handleInputMismatchException() – allows the user to have another try to run doRational(). Does not return anything.
      1. Parameters - InputMismatchException parameter.
      2. Runs doRational() in a protected block (ie try/catch)
      3. Flow of Method – Output a message to the user indicating that “You entered a non-integer input”. Then call doRational() in a protected block. If either an InputMismatchException or DivideByZeroException throw occurs, you should print a message indicating the issue based on the exception thrown, tell the user the program is ending, and then exit the program.
   4. calcRational() – calculates and returns a double based on numerator / denominator. If the denominator is 0, calcRational will throw a DivideByZeroException. NOTE: This is public for the purpose of this lab to be included in the interface, but technically could be a private helper method
      1. Parameters – none
      2. Throws DivideByZeroException (passes the buck to the calling method)
      3. Flow of Method – test to see if denominator is 0, if so throw DivideByZeroException. Otherwise, calculate and return a double to the calling method
2. Create an exception called DivideByZeroException that extends Exception. **Do not** **pass** a “message” during the DivideByZeroException throw, DivideByZeroException should set a default message “Divide by zero error”.
3. Create a driver class called RationalDemo. RationalDemo does not have any instance paramaters or named constants. RationalDemo should:
   1. Create a Rational object called rat1.
   2. Invoke doRational() based on rat1 created above in a protected block.
   3. If an InputMismatchException is caught, invoke handleInputMismatchException with the appropriate arguments.
   4. If a DivideByZeroException is caught, write to output the exception object’s message (ie e.getMessage())
   5. In order to let the user know that the program has completed running, write an output such as “Thanks for playing” before exiting main().

Run the following use cases:

Use Case 1

1. Numerator 1, Denom 3, y
2. Enter a non-integer input to create a InputMismatchException throw
3. Numerator 2, Denom 7, Y
4. Numerator 7, Denom 0 to create a DivideByZeroException throw

Use Case 2

1. Enter a non-integer input to create a InputMismatchException throw
2. Enter a non-integer input to create a InputMismatchException throw from handleInputMismatchException()

Use Case 3

1. Run again Numerator 1. Denom 3, n or N.

**Submitting your work**

For all labs you will need to provide a copy of all .java files. **DO NOT PROVIDE .class files. I cannot grade, what I cannot read.** In addition to your .java files, you will need to produce a pix of the screen output in .png or .jpg format for each project that covers all use cases in the lab. For persons using Windows 7 and above OS, you can use the built in snipping tool. Mac OS users, you can see how to take screenshots using the following url - <http://www.wikihow.com/Take-a-Screenshot-in-Mac-OS-X>

You will need to zip your files into a single container. **DO NOT USE .rar for Mac OS.** Submit appropriate .java files and screenshots to show input and output in either .png or .jpg format.