CSC 230: Elementary Data Structures and Algorithms

Fall 2022

Exam 2

- General programming guidelines:
  - Create a folder with the name "Exam-2" and save your Netbeans project in there.
  - Do not forget necessary javadoc comments before classes and methods.
  - You should use *single line* or *multiline* comments, **if it is required**. Do not put unnecessary comments (Do not state the obvious!!!).
  - Use meaningful identifier.
  - Don't forget to check the parameters of your methods and throw appropriate exceptions as necessary.
  - **Zipping your final exam folder, and testing it before uploading are your responsibilities. If any of these steps fails, you will receive a grade of zero.**
  - If you have questions or need clarifications, just ask.
- Academic integrity policy
  - You are **not** allowed to use any online resources EXCEPT Revel, class lecture notes and Java documentation.
  - You are not allowed to discuss questions with anybody. That include seeking help in online forums.
  - All programs/ code must be your own work.
  - You should be able to clearly explain every line of your code, if instructor requests you to do so.
  - Any violation of these policies will be considered as plagiarism and dealt accordingly.

Task: You got an internship in *Acme.inc* and your project manager asked you to read the given code and add more functionality to it **without completely re-coding**. This will test your ability to:

- Read/ understand code with minimum documentation written by others,
- Read/ understand/ use Javadoc to utilize existing API,
- and *Minimally* modify existing code to add functionality.

Carefully study the provided `BigRational` class (Unzip the given file and you will find a Netbeans project). Some of the tasks given below require you to read Java API documentation of classes/ interfaces (e.g. `BigInteger` and `Collections`).
Do the following modifications to the `BigRational` class. Make sure to throw appropriate exceptions, when necessary.

Note: Test cases have been provided with the expected output in the `main` method. As you start coding, you should uncomment the corresponding test cases and test your code. Passing these test cases are necessary but may not be sufficient to show that your logic is indeed correct as *Acme.inc* cannot provide unlimited test cases. DO NOT change any code provided in the `main` method except removing multiline comments of each question when you are ready to test your method.

Question 1 (10 Points) Read and understand provided code for the `BigRational` class. Make sure to understand each of the existing methods and check the accuracy of them. Add necessary comments and Javadocs comments for each method.

Question 2 (20 Points) Write code for `BigRational pow(int exp)`. It returns a `BigRational` whose value is $(\text{this}^{\text{exponent}})$.

Throw an `ArithmeticException` with "Exponent is negative" as the string parameter, if `exp < 0`. Note that `exponent` is an integer rather than a `BigRational`.

Question 3 (10 Points) Write code for `BigRational reciprocal()`. It returns the reciprocal of a `BigRational` number.

Question 4 (10 Points) Write code for `BigInteger toBigInteger()`. It returns the `BigInteger` version of `BigRational` if the denominator is 1. If the denominator is not 1, throw an `ArithmeticException` with "Not an integer value" as the string parameter.

Question 5 (10 Points) Write code for `int toInteger( )`. It returns the `Integer` version of `BigRational` if the denominator is 1. If the denominator is not 1, throw an `ArithmeticException` with "Not an integer value" as the string parameter.

Question 6 (20 Points) Modify the `BigRational` class so that 0/0 is legal and as interpreted as "indeterminate" by `toString`

Question 7 (20 Points) Implement `Comparable` interface for `BigRational` class. Note that you cannot compare an indeterminate with anything. In your comparison, if you have an indeterminate operand, any attempt to compare should throw `ArithmeticException` with "One of them is indeterminate" as the string parameter. Once you correctly write code, you should be able to sort a list of BigRationals as long as the list doesn't have any indeterminate.