Project #4: Rational Class

In this project, you will implement a class to represent a rational number. Mathematically (and for this assignment), a rational number is defined as a fraction in which the numerator and denominator are integers with the denominator being non-zero.

Requirements

To ensure consistency among solutions each implementation must implement the following requirements.

- Your implementation should reflect the definition of a simplified rational number at all times.
- A `long` (integer) is used to represent both the numerator and denominator.
- The `Rational` class must be able to handle both positive and negative numbers.
- The rational number must be in simplest form after every operation; that is, $8/6$ shall be immediately reduced to $4/3$. In order to simplify a rational number properly, use the the Euclidean Algorithm to determine the greatest common divisor of the two numbers.
- All methods that have an object as parameter must be able to handle an input of `null`.
- By definition, the denominator of a fraction cannot be zero since it leads to an undefined value. In Java, this division by zero results in an `ArithmeticException; the Rational` class must throw an `ArithmeticException when` division by zero is attempted.
- The denominator should be initialized to a sensible and consistent value. The numerator shall be initialized to zero.
- The denominator should always be positive leaving the numerator to indicate the sign of the number (positive or negative).
- The `Rational` class shall reside in the default package.

Methods

There are many methods that one would expect to be supported in a `Rational` class; unless specified, you will have to implement all of the described methods.

Constructors

- `Rational()`
  - Description: constructs a `Rational` initializing the value to 0.
- `Rational(long a)`
  - Parameters: `a` – the default value for the `Rational` number.
  - Description: constructs a `Rational` initializing the value to $a/1$.
- `Rational(long a, long b) throws ArithmeticException`
  - Parameters:
    - `a` – an integer specifying the initial value of the numerator.
    - `b` – an integer specifying the initial value of the denominator.

  - Description: constructs a `Rational` number initializing the object to a/b.

Accessors

- `long getNumerator()`
  - Description: returns the current value of the numerator.
  - Returns: as an accessor method, it only returns the current value of the numerator.
- `long getDenominator()`
  - Description: returns the current value of the denominator.
  - Returns: as an accessor method, it only returns the current value of the denominator.

Mathematical Operations

With each of the following methods, if the input object `r` is `null`, you may treat the input as the value zero (0).

- `Rational add(Rational r)`
  - Parameters: `r` – the rational number to be added to `this` rational.
  - Description: adds `r` and `this`, returning a new object with the reduced sum. A common denominator is required to complete this operation.
  - Returns: returns a new object with the reduced sum.
- `Rational subtract(Rational r)`
  - Parameters: `r` – the rational number to be subtracted from `this` rational.
  - Description: subtracts `r` from `this`, returning a new object with the reduced difference. A common denominator is required to complete this operation.
  - Returns: returns a new object with the reduced difference.
- `Rational multiply(Rational r)`
  - Parameters: `r` – the rational number to be multiplied with `this` rational.
  - Description: multiplies `r` with `this`, returning a new object with the reduced product.
  - Returns: returns a new object with the reduced product.
- `Rational divide(Rational r) throws ArithmeticException`
  - Parameters: `r` – the rational number that is to divide `this` rational.
  - Description: divides `this` by `r`, returning a new object with the reduced quotient.
  - Returns: returns a new object with the reduced quotient.

The Greatest Common Divisor Algorithm

- `private long gcd(long p, long q)`
  - Parameters:
    - `p` – an integer.
    - `q` – an integer.
  - Description: determines the greatest common divisor of the two input integers according to the Euclidean Algorithm; a quick implementation is easily available online.
  - Returns: the (positive) GCD of the two input integers.
  - Notes: It is easier to implement this method if both integers are positive values.

Supplied Methods

Besides the interface specification described above, there are other methods that have been provided; feel free to use them as required in your implementation and testing. ***Do not modify these methods.***

- `String toString()`
  - Description: provides a means of viewing the values contained within the `Rational` object.
  - Returns: a `String` representation of `this` rational value.
- `int compareTo(Object obj)`
  - Parameters: `obj` – an object to compare against `this` rational value.
  - Description: determines whether the content of `obj` is smaller (or larger) than `this` rational value; the result returned is compliant with the `Comparable` interface that the `Rational` class implements.
  - Returns: -1, 0, or 1 when `this` object is less than, equal, or greater than `obj`.
- `boolean equals(Object obj)`
  - Parameters: `obj` – an object to compare against `this` rational value.
  - Description: determines whether the content of `obj` equals `this` rational value.
  - Returns: TRUE or FALSE whether `obj` is an instance of class `Rational` and is equal to the rational value represented in `this` rational value.

Testing

- `public static void main(String args[])`
  - Parameters: `args` – a rational of values that specifies which tests are to be executed.
  - Description: this is a method that will be part of a test class `RationalTester`.
  - Notes: A test class (`RationalTester`) has been provided to expedite development of tests. This method can be modified as desired; you will not have to submit this file.

Submitting

Your program must use the following standard comment at the top of the page as well as a reasonable amount of comments throughout the program. Copy and paste this comment and modify it accordingly.

```
/***************************************************************
<Student Name>
<File Name>
<Assignment>
<Describe, in general, the code contained.>
***************************************************************/
```

Place your Java file for the `Set` class directly into a zip file. The name of the zip file **must be** `proj4.zip`. Make sure your zip file contains the correct files (`Rational.java`). Submit your zip file via Sakai under `Assignments > Project 4`. Be sure to review the university policy on academic dishonesty. This is an individual project.