RATIONAL CLASS AND RATIONALTEST

`Rational` class

A `Rational` Number is a value that can be written as a ratio of two integers (a fraction). You will be writing a `Rational` class and a test program (`RationalTest`) for this class.

- A `Rational` has 2 instance fields (numerator, denominator)
- A `Rational` has the following methods:

1. one constructor with two parameters, one for the value of the numerator and one for the value of the denominator. This constructor will ensure that the denominator is unsigned (positive) and non zero and will reduce the `Rational` by calling the `private` method `reduce.` Examples (found in the test program):

   `Rational r1 = new Rational (-2,-4);`
   after the constructor code has executed,
   `r1.numerator = 1` and `r1.denominator = 2`

   `Rational r1 = new Rational (2,-4);`
   after the constructor code has executed,
   `r1.numerator = -1` and `r1.denominator = 2`

   `Rational r1 = new Rational (2,0);`
   after the constructor code has executed,
   `r1.numerator = 2` and `r1.denominator = 1`

   `Rational r1 = new Rational (-2,-4);`
   after the constructor code has executed,
   `r1.numerator = 1` and `r1.denominator = 2`

   `Rational r1 = new Rational (-2, 4);`
   after the constructor code has executed,
   `r1.numerator = -1` and `r1.denominator = 2`

2. `public` method `getNumerator` that returns the integer value of the numerator
3. `public` method `getDenominaotr` that returns the integer value of the denominator

4. `public` method `getReciprocal` that returns a Rational that is the reciprocal of this `Rational.`
5. `public` method `add` that has one `Rational` parameter and returns a `Rational` that is the sum of `this Rational` and `add'`s `Rational` parameter.
6. `public` method `subtract` that has one `Rational` parameter and returns a `Rational` that is the difference of `this Rational` and `subtract'`s `Rational` parameter.
7. `public` method `multiply` that has one `Rational` parameter and returns a `Rational` that is the product of `this Rational` and `multiply'`s `Rational` parameter.
8. `public` method `divide` that has one `Rational` parameter and returns a `Rational` that is the quotient of `this Rational` and `divide'`s `Rational` parameter.
9. `private` method `reduce` that will reduce the numerator and denominator by their largest common factor.
10. `public` method `equals` that has one `Object` parameter. Compares `this Rational` to the parameter. The result is `true` if and only if the two have `numerator`s values that are equal and `denominator` values that are equal.
11. `public` method `compareTo` that has one `Object` parameter. Compares `this Rational` to the parameter. Returns the value `0` if this `Rational` is equal to the parameter; a value less than `0` if this `Rational` is numerically less than the parameter; and a value greater than `0` if this `Rational` is numerically greater than the parameter (signed comparison).
12. `public` method `toString` that returns a `String` representation of `this Rational.` Example : if `numerator = 5` and `denominator = 7,` `toString` returns "5/7".

RationalTest

Your `RationalTest` contains the `main` method.

Get a values for the numerator and denominator of `r1` (`n` and `d`) from the user with the Scanner Class. These value should be integers between -20 and 20 inclusive.

```
Rational r1 = new Rational(n,d);
```

Get a values for the numerator and denominator of `r2` (`n2` and `d2`)by randomly generating them using the `java.util.Random` class. These value should be integers between -20 and 20 inclusive.

```
Rational r2 = new Rational(n2,d2);
```

Include the following statements in your test program:

```
Rational rSum, rDiff, rProd, rQuot, rRecip;
System.out.println ("First rational number: " + r1);
System.out.println ("Second rational number: " + r2);

if (r1.equals(r2))
    System.out.println ("r1 and r2 are equal.");
else
    System.out.println ("r1 and r2 are NOT equal.");

if (r1.compareTo(r2) < 0)
    System.out.println ("r1 is less than r2.");
else if (r1.compareTo(r2) > 0)
    System.out.println ("r1 is greater than r2.");
else
    System.out.println ("r1 and r2 are equal.");

rRecip = r1.reciprocal();
System.out.println ("The reciprocal of r1 is: " + r3);

rSum = r1.add(r2);
rDiff = r1.subtract(r2);
rProd = r1.multiply(r2);
rQuot = r1.divide(r2);

System.out.println ("r1 + r2: " + rSum);
System.out.println ("r1 - r2: " + rDiff);
System.out.println ("r1 * r2: " + rProd);
System.out.println ("r1 / r2: " + rQuot);
```

- If you find it useful to add other "utility methods" to your `Rational` class, you may do so.
- Remember to choose appropriate test cases.