Fraction.java

/* Name: Richard Eisenberg
 * File: Fraction.java
 * Desc: A simple Fraction class
 */

class Fraction implements Printable
{
    // INVARIANT: The numerator and denominator have no common divisors.
    // INVARIANT: The denominator is always a positive number.
    private int numerator;
    private int denominator;

    /** Create a fraction representing an integer
     * @param n The integer to represent
     */
    public Fraction(int n)
    {
        numerator = n;
        denominator = 1;
    }

    /** Create a fraction representing the quotient of two integers
     * @param n the numerator
     * @param d the denominator
     */
    public Fraction(int n, int d)
    {
        numerator = n;
        denominator = d;
        normalize();
    }

    /** @return the numerator of this fraction, reduced so that the numerator and denominator have no common divisor
     */
    public int getNumerator()
    {
        return numerator;
    }

    /** @return the denominator of this fraction, reduced so that the numerator and denominator have
     *        no common divisor
     */
    public int getDenominator()
    {
        return denominator;
    }

    /** Adds this fraction with another. This method returns the sum; it does not modify the current Fraction object.
     * @param other The other fraction
     * @return The sum fraction
     */
    public Fraction add(Fraction other)
    {
        return new Fraction(numerator * other.getDenominator() +
                             other.getNumerator() * denominator,
                             denominator * other.getDenominator());
    }

    /** Multiplies this fraction against another. This method returns the product; it does not modify the current
     * Fraction object.
     * @param other The other fraction
     * @return The product fraction
     */
    public Fraction multiply(Fraction other)
Fraction.java

    {   
      return new Fraction(numerator * other.getNumerator(),
                          denominator * other.getDenominator());
    }

    /** @return a new fraction that is the negation of the 
     * current one */
    public Fraction negate()
    {   
      return new Fraction(numerator * -1, denominator);
    }

    /** @return a new fraction that is the reciprocal of the 
     * current one */
    public Fraction reciprocal()
    {   
      return new Fraction(denominator, numerator);
    }

    /** @return a string representation of this Fraction */
    @Override
    public String toString()
    {   
      return numerator + "/" + denominator;
    }

    /** Normalizes the current Fraction, by reducing the 
     * numerator and denominator until they share no common 
     * factor.  
     * NB: This is a private method, used internally only. */
    private void normalize()
    {   
      // fix the signs first
      if(denominator < 0)
      {   
        numerator *= -1;
        denominator *= -1;
      }

      int posNumerator = Math.abs(numerator);
      int gcd = gcd(posNumerator, denominator);
      numerator /= gcd;
      denominator /= gcd;
    }

    /** @return the greatest common divisor of the two arguments */
    private static int gcd(int a, int b)
    {   
      if(a == 0)
      {   
        return b;
      }

      while(b != 0)
      {   
        if(a > b)
        {   
            a = a - b;
        }
        else
        {   
            b = b - a;
        }
      }

      return a;
```java
/** Returns whether or not this Fraction equals another
 * @return true iff the other fraction equals this one */

public boolean equals(Object other) {
    if (other instanceof Fraction) {
        Fraction f = (Fraction)other;
        return numerator == f.getNumerator() &&
               denominator == f.getDenominator();
    } else {
        return false;
    }
}

// Prints out the contents of this object
public void print() {
    System.out.println(numerator + "/" + denominator);
}
```