Classes and Objects 1

Consider the Fraction that we have created in class.

1. Complete the definitions of the following instance methods for the Fraction class.
   1. public Fraction times (Fraction other)

| //This method creates a new object to return  public Fraction times(Fraction other)  {  Fraction output = new Fraction((other.numerator\*this.numerator),(other.denomimator\*this.denomimator));  return output;  }  //This method changes the implicit Fraction object  //Print Fraction, for output reference  public static String outputFraction(Fraction input)  {  return (input.numerator+"/"+input.denomimator);  } |
| --- |
| Input: Fraction f1 = new Fraction(5,4); Fraction f2 = new Fraction(2,3);  Product: |

This method returns a Fraction object that is the product of the implicit object parameter and the explicit parameter other. This method does NOT need to simplify the returned Fraction yet -- we will do that in part 3c.

* 1. public Fraction plus (Fraction other)

This method returns a Fraction object whose value is the sum of the implicit object parameter and the explicit parameter other. The method should leave both its explicit and implicit parameters unchanged. Similar to 1a, this method does NOT need to simplify the returned Fraction yet -- we will do that in part 3c.

| public Fraction plus(Fraction other)  {  Fraction output = new Fraction(( (this.numerator\*other.denomimator)+(this.denomimator\*other.numerator)),this.denomimator\*other.denomimator);  return output;  } |
| --- |
| (Same object values) |

1. Assume the larger method is declared as followed:

public Fraction larger (Fraction other) {

if (this.size() >= other.size() ) {

return this;

} else {

return other;

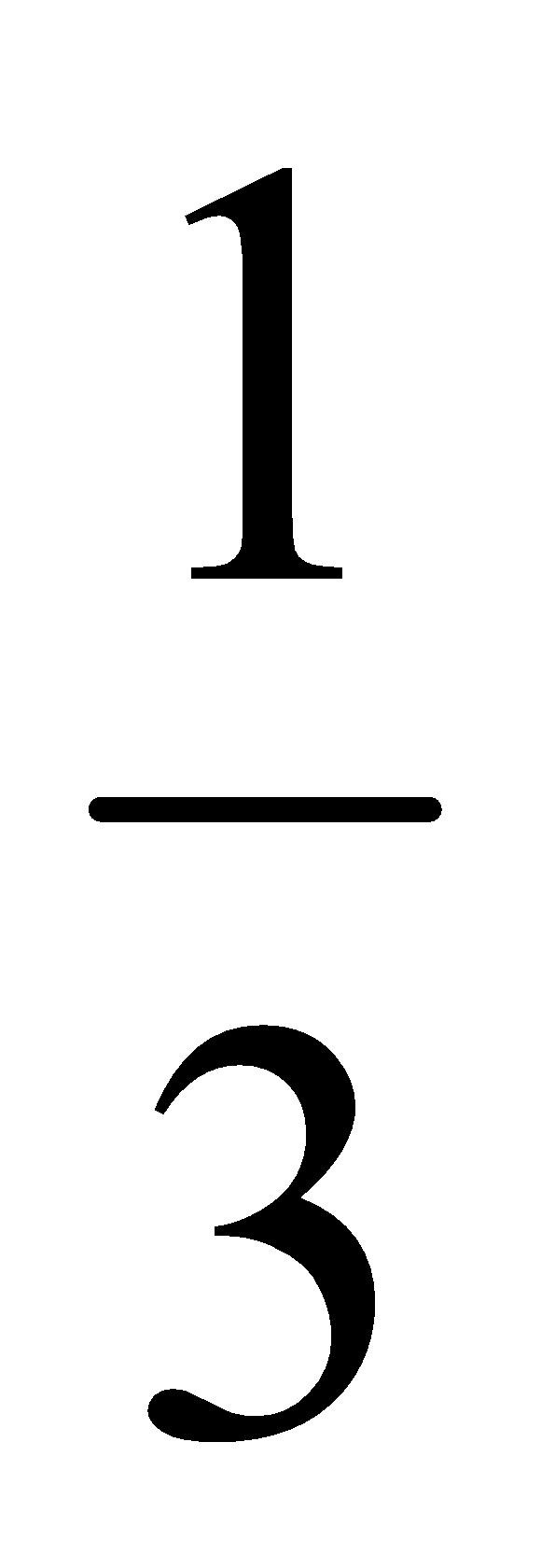
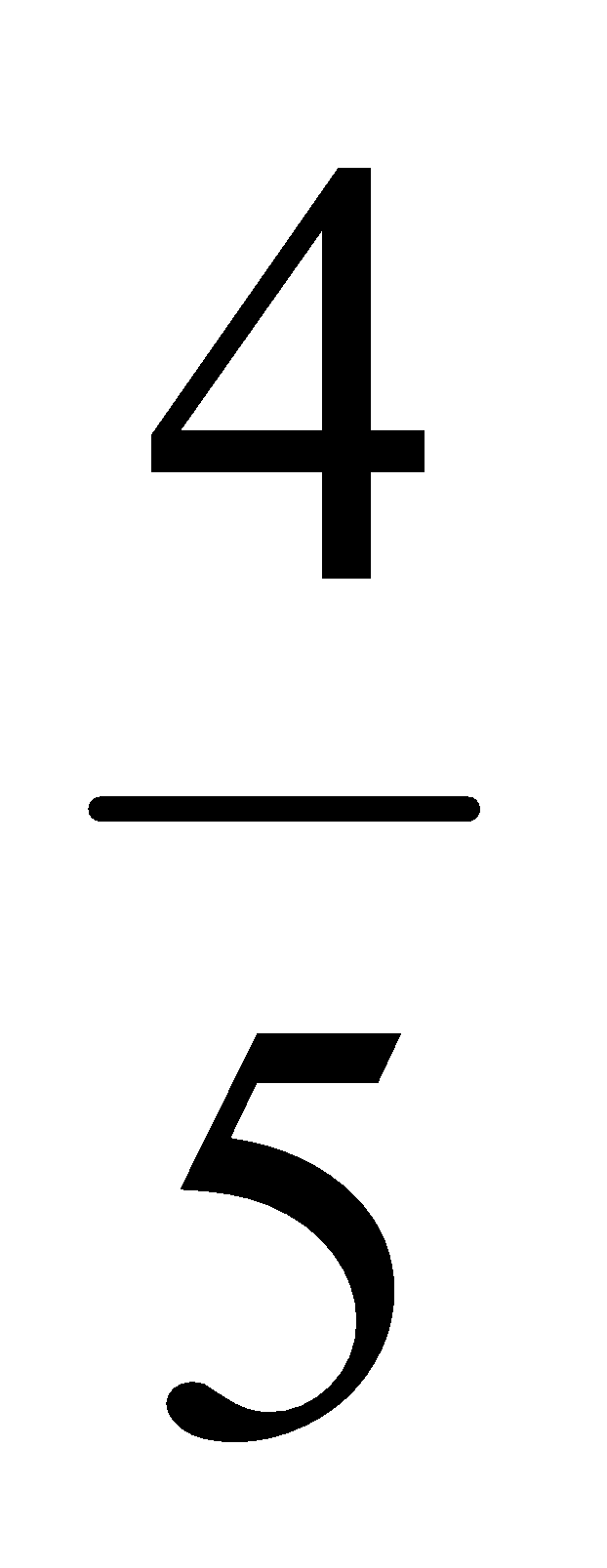
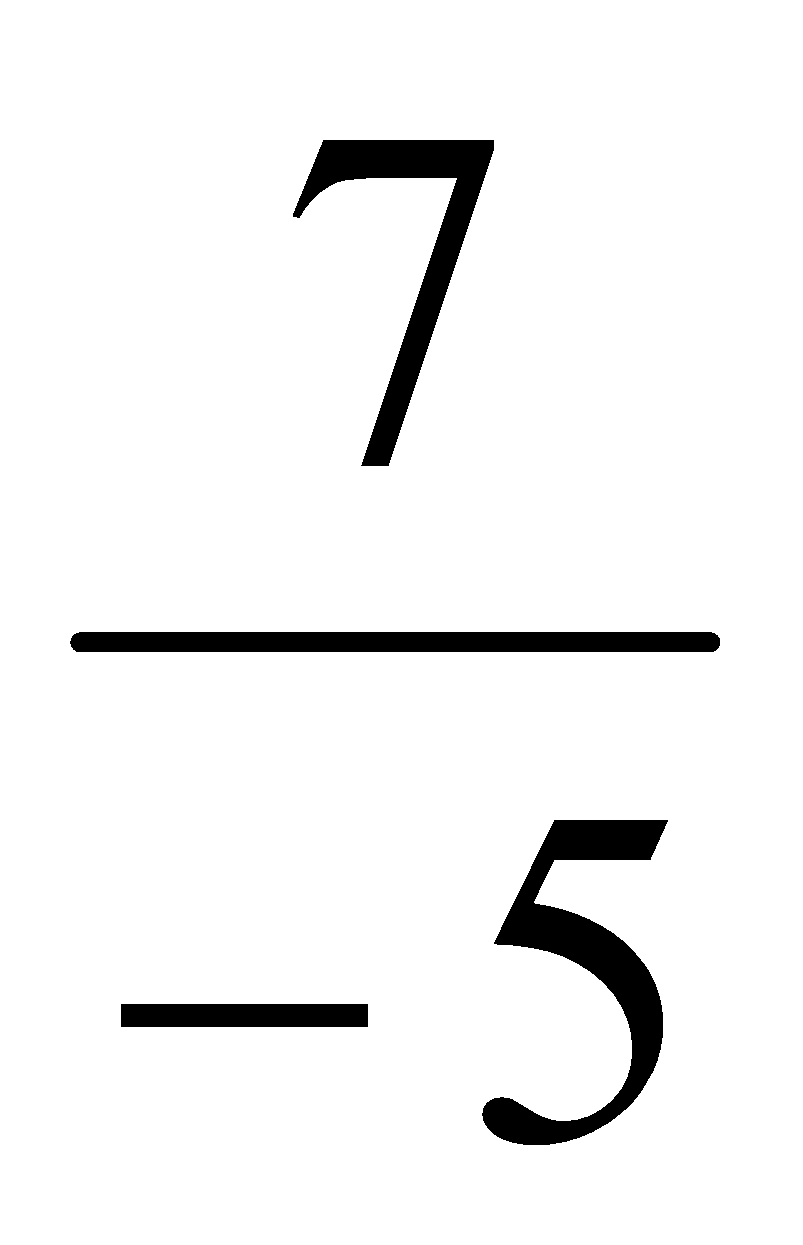
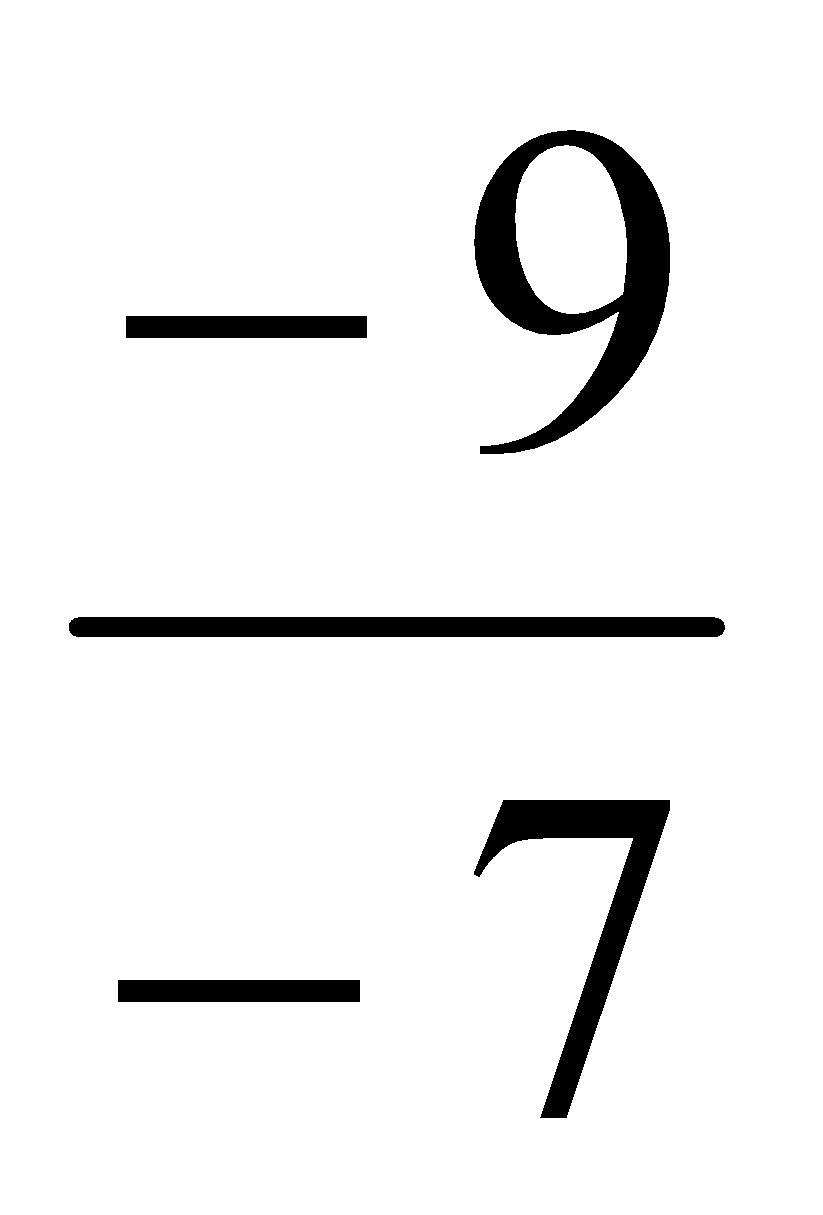
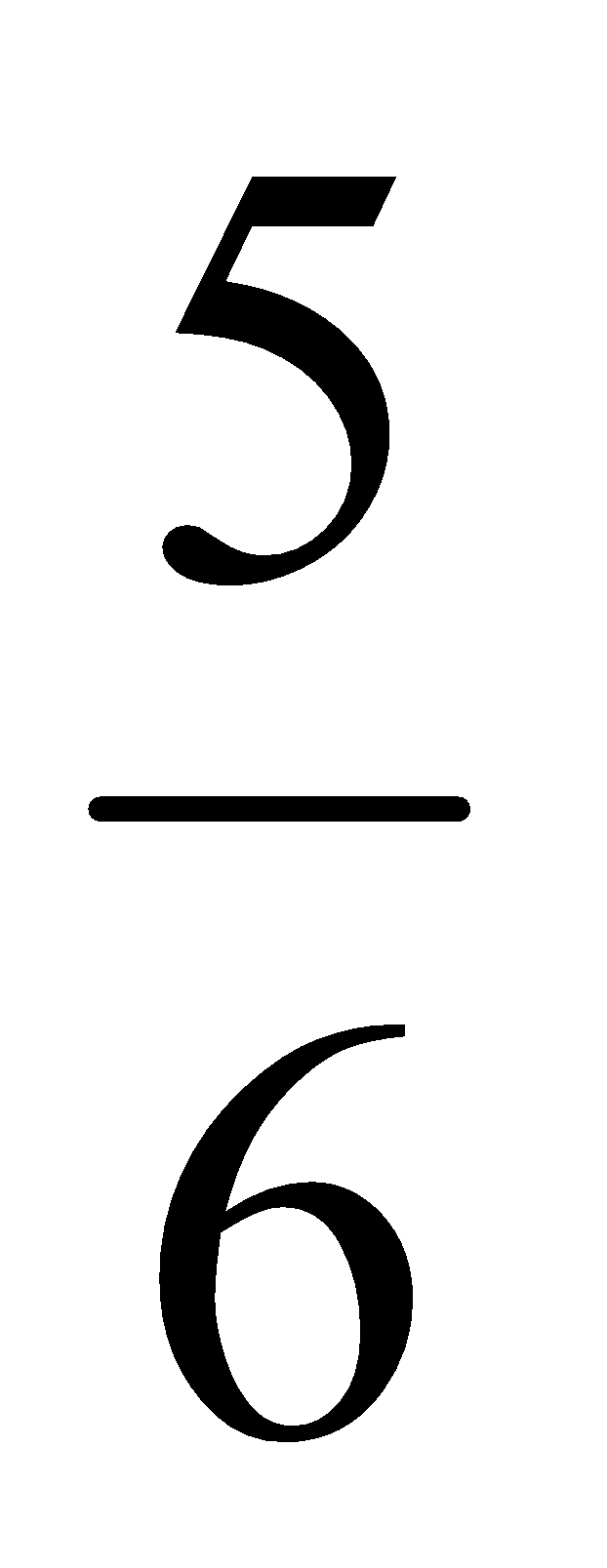
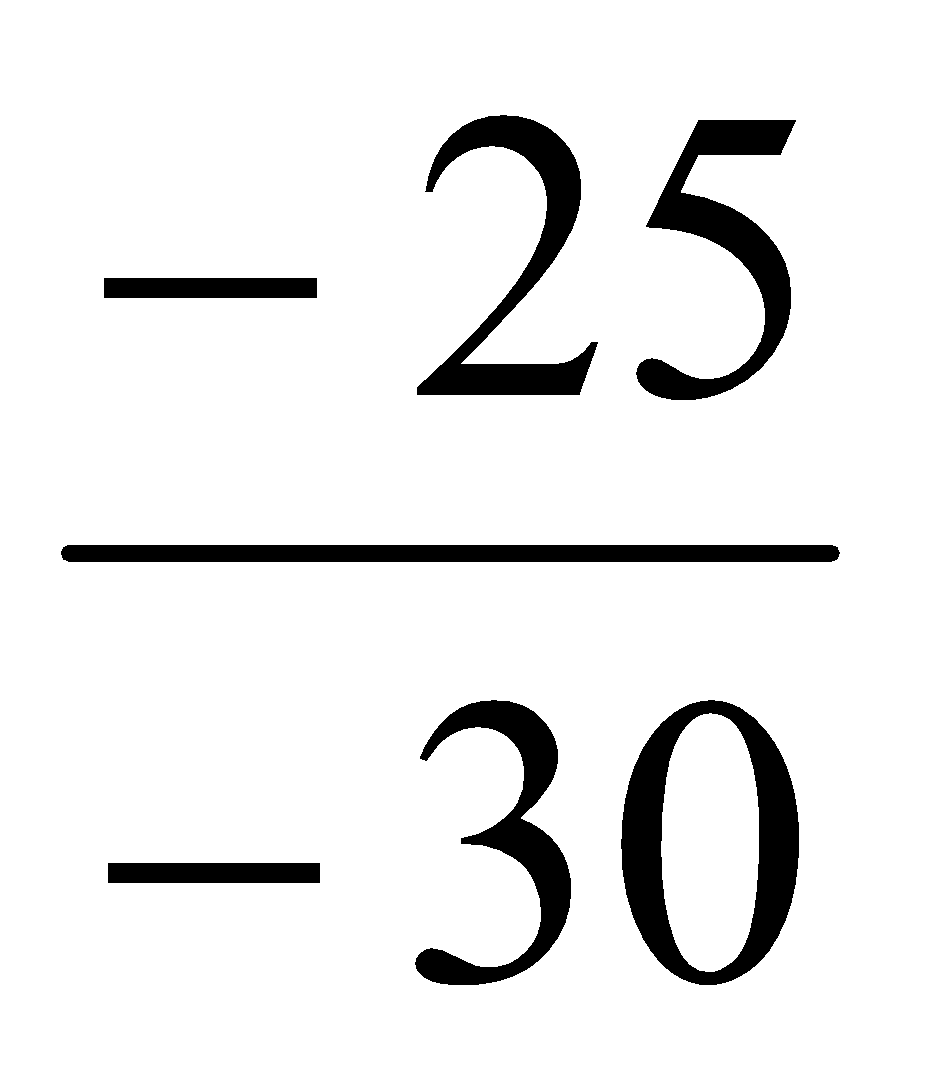
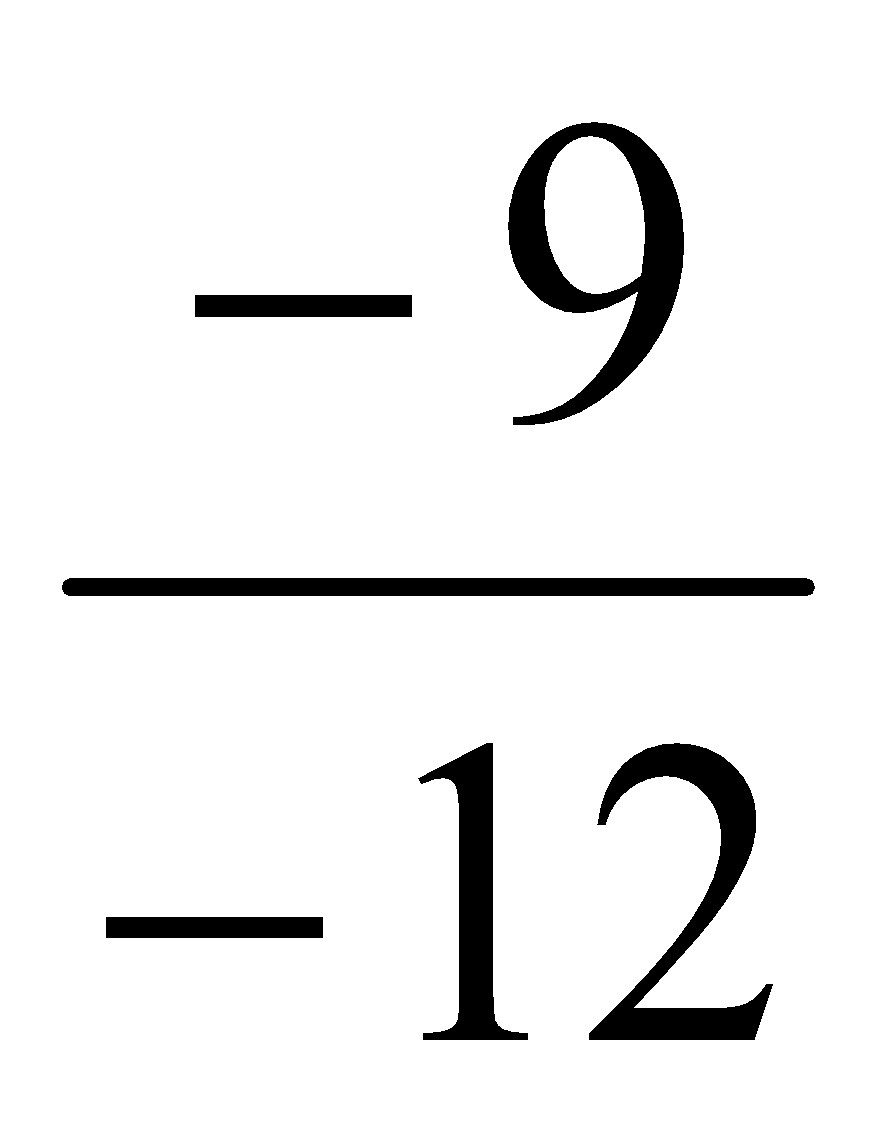
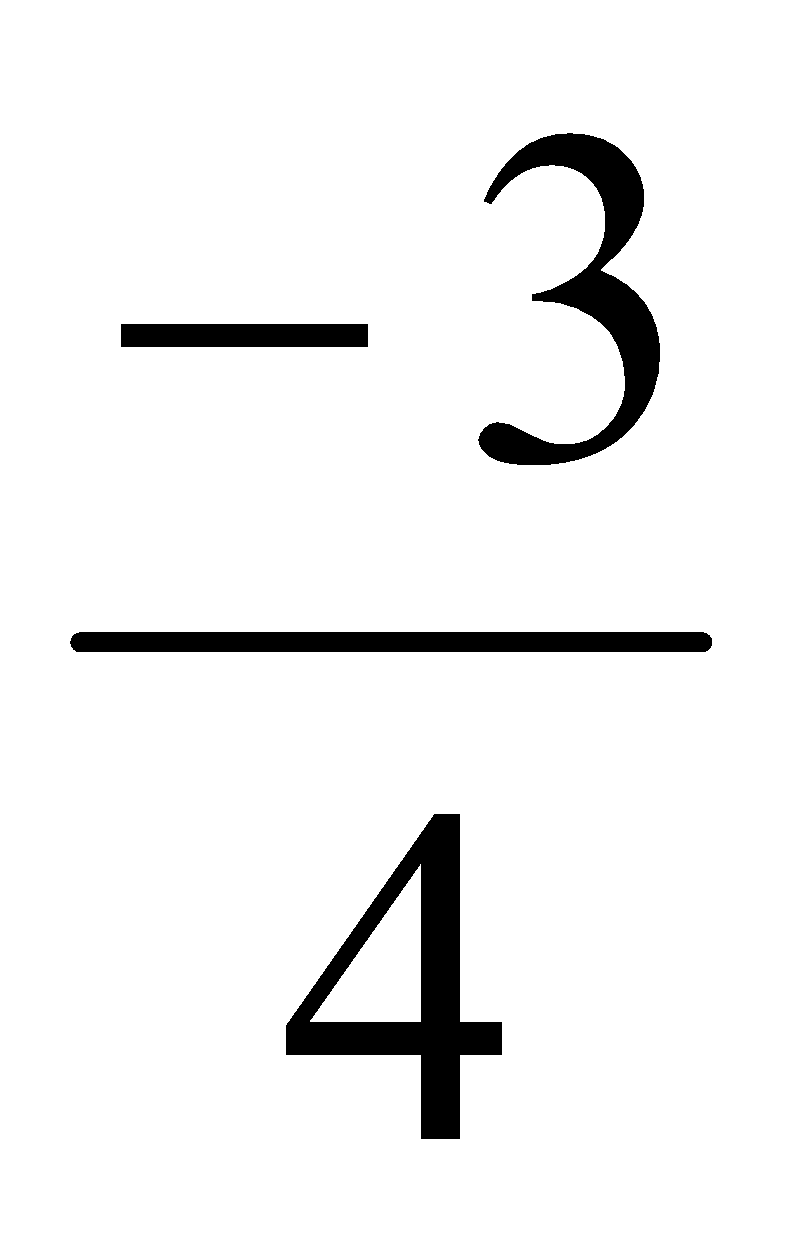
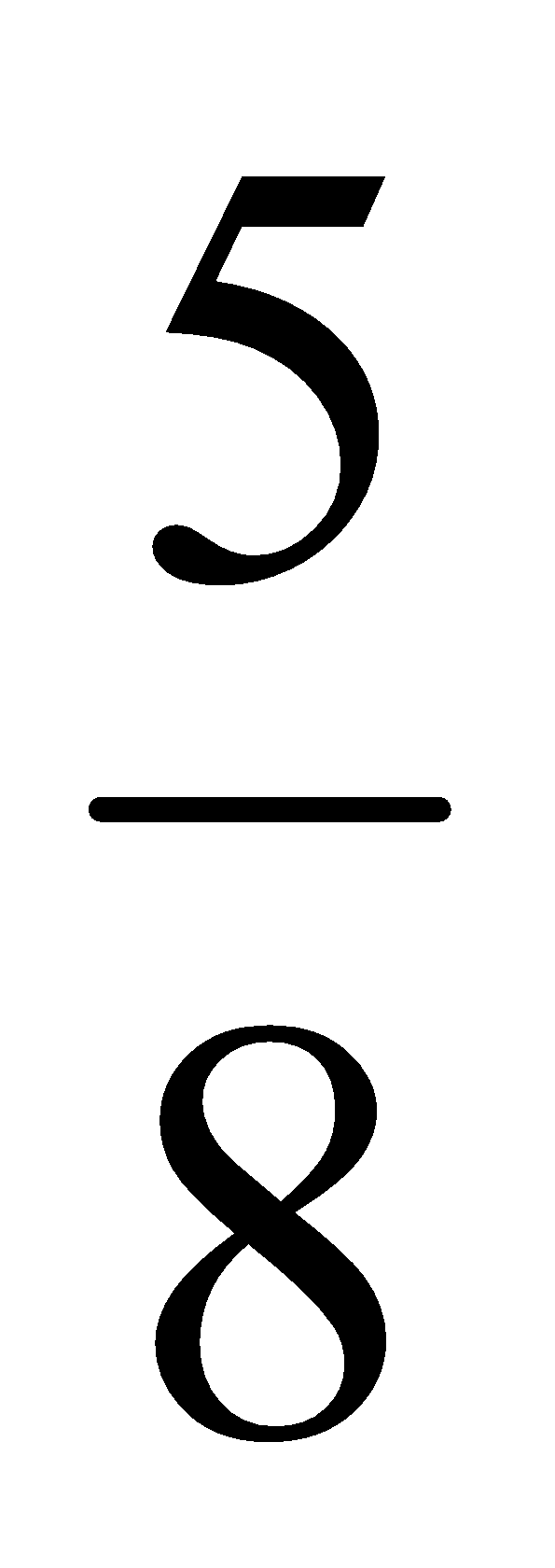
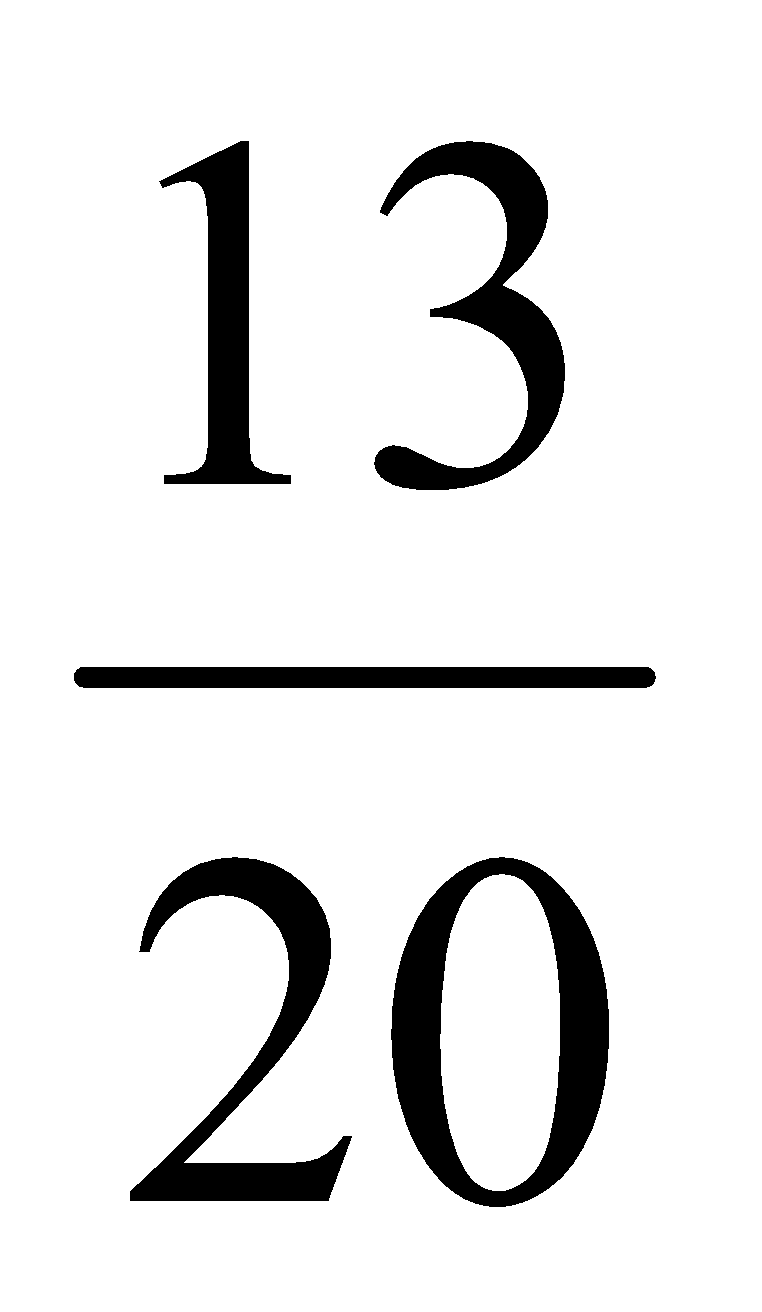
}

}

Suppose that p, q, and r are all objects of type Fraction. What fraction would r represent after the statement

r = p.larger(q);

is executed?

* 1. p represents and q represents  r = q
  2. p represents and q represents  r = p
  3. p represents and q represents  r = p
  4. p represents and q represents r = p
  5. p represents and q represents r = q

1. Complete the definitions of the following instance methods for the Fraction class.
   1. public void timesEquals(Fraction p)

This method should have the same effect (for Fraction objects) that the \*= operator has for primitive numeric types. Thus, if called by the statement

p.timesEquals(q);

(where p and q are objects of type Fraction), the method would make p represent the product of the fractions currently represented by p and q while the value of q would be left unchanged.

|  |
| --- |
|  |

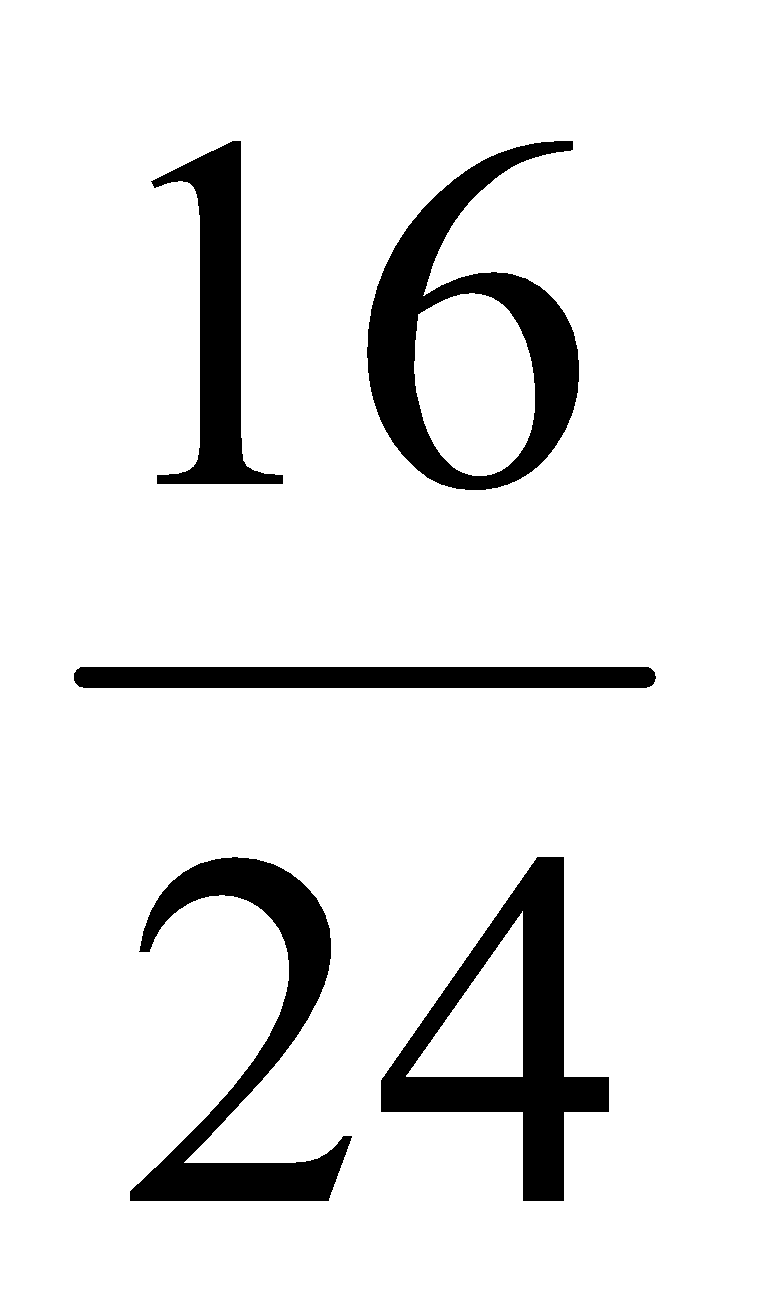
* 1. public Fraction plus (Fraction f)

The method should return a Fraction object whose value is the sum of the implicit object parameter (this) and the explicit parameter (f). The method should leave both its explicit and implicit parameters unchanged.

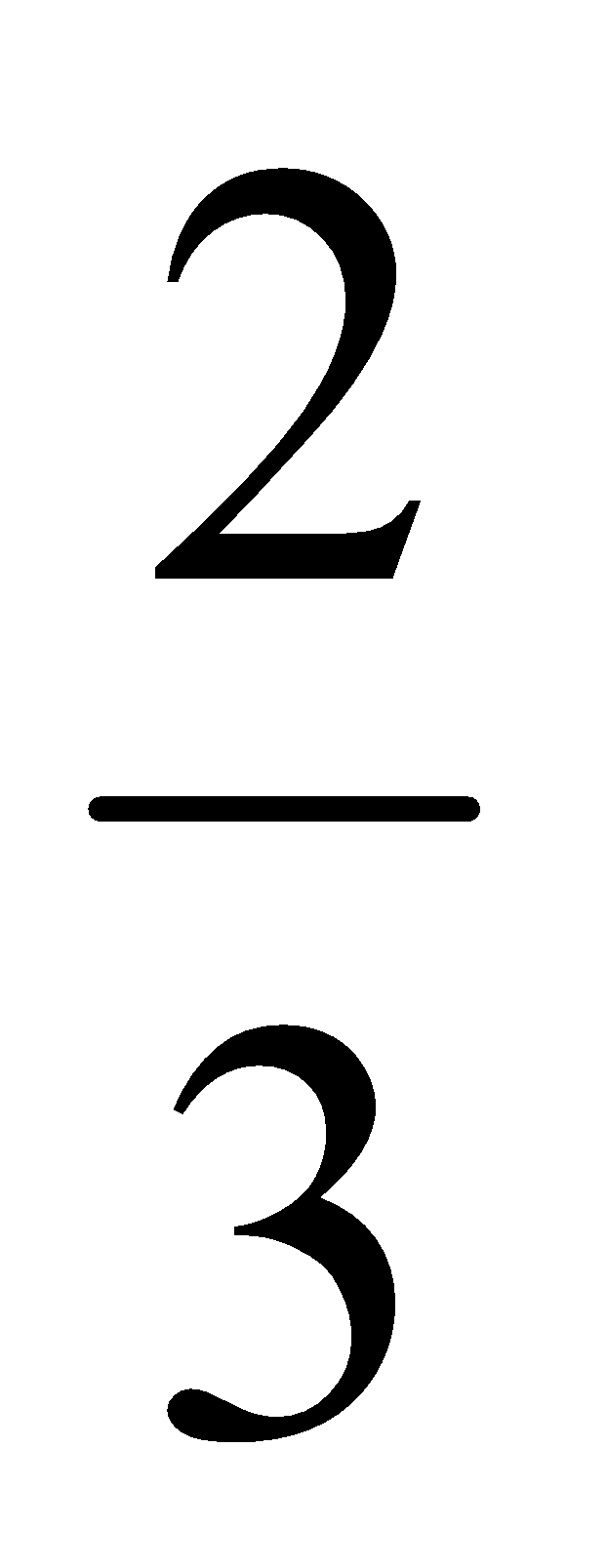
**\*Note:** you already created this method in exercise 1b. At this stage, ensure that your method is making use of the keyword this to help differentiate between the Fraction to which the method belongs (this), and the Fraction being passed as a parameter (other).

| public Fraction plus(Fraction other)  {  Fraction output = new Fraction(( (this.numerator\*other.denomimator)+(this.denomimator\*other.numerator)),this.denomimator\*other.denomimator);  output.reduce();  return output;  } |
| --- |
| Same as before: |

* 1. public void reduce()

The method should reduce its implicit Fraction parameter to lowest terms. For example, if f represents the fraction, the statement

f.reduce();

should change f so that it represents the fraction.  
If they do not do so already, your times(), timesEquals() and plus() methods should use your reduce() method to ensure their answers are always reduced to lowest terms.

|  |
| --- |
| Input (999/111) = |