```

//Purpose: take input from a file and output the fraction value of the operation
//Caspian Peavyhouse
//CS101-02
//11/23/14

import java.util.*;
import java.io.*;

public class FractionMath
{
/*
     Algorithm for main(args)
main(args)
   File inputFile <-- new File(args[0])
   Scanner input <-- new Scanner(inputFile)

   File outputFile <-- new File(args[1])
   FileWriter writerOutput <-- new FileWriter(outputFile)
   String currentLine <-- new String(input.nextLine())

   writerOutput.write("Fraction Math Version 4")
   writerOutput.write("Written by Caspian Peavyhouse")
   writerOutput.write("CS101-02")

   do
       currentLine <-- currentLine.toLowerCase()
       if currentLine contains("quit")
          break
       else
          readLine(currentLine, writerOutput)
       currentLine <-- input.nextLine()

   while (input.hasNextLine())



*/

/*
     Data Table for main

Variable or Constant     Purpose
_____________________    ___________________
args                     parameter for main
*/

   public static void main(String [] args)throws Exception
   {
      File inputFile = new File(args[0]);
      Scanner input = new Scanner(inputFile);

      File outputFile = new File(args[1]);
      FileWriter writerOutput = new FileWriter(outputFile);
      String currentLine = new String(input.nextLine());

      writerOutput.write("FractionMath Version 4\n");
      writerOutput.write("Written by Caspian Peavyhouse\n");
      writerOutput.write("CS101-02\n\n");

      do
      {
         currentLine = currentLine.toLowerCase();
         if (currentLine.contains("quit"))
         {
            break;
         }

```

```
         else
         {
            readLine(currentLine, writerOutput);
         }
         currentLine = input.nextLine();

      } while (input.hasNextLine());

      writerOutput.close();

   }//main(args)


/*
readLine(String currentLine, FileWriter writerOutput)

   writerOutput write(currentLine + \n)

   currentLine = currentLine.replace('/', ' ')
      Scanner stringScan <-- new Scanner(currentLine)
      Scanner numberScan <-- new Scanner(currentLine)
      if (!((currentLine.charAt(0)>= '0' && currentLine.charAt(0) <= '9')
            || currentLine.charAt(0) == '-'))
         numberScan.next()

      int firstNum <-- numberScan.nextInt()
      int secondNum <-- numberScan.nextInt()

      String currentString <-- new String(stringScan.next())

      Fraction currentFraction <-- new Fraction(firstNum, secondNum)
      Fraction nextFraction <-- new Fraction()

      while (stringScan.hasNext())

         if (currentString.equals("add"))
            numberScan.next()
            Fraction nextFraction <-- new Fraction(numberScan.nextInt(),
               numberScan.nextInt())
            currentFraction <-- readAdd(writerOutput, currentFraction, nextFraction)
         else if (currentString.equals("sub"))
            numberScan.next()
            Fraction nextFraction <-- new Fraction(numberScan.nextInt(),
               numberScan.nextInt())
            currentFraction <-- readSubtract(writerOutput,
               currentFraction, nextFraction)
         else if (currentString.equals("mul"))
            numberScan.next()
            Fraction nextFraction <-- new Fraction(numberScan.nextInt(),
               numberScan.nextInt())
            currentFraction <-- readMultiply(writerOutput, currentFraction,
               nextFraction)
         else if (currentString.equals("div"))
            numberScan.next()
            Fraction nextFraction <-- new Fraction(numberScan.nextInt(),
               numberScan.nextInt())
            currentFraction <--readDivide(writerOutput,currentFraction,nextFraction)
         else if (currentString.equals("rec"))
            writerOutput.write("\tthe reciprocal of " + currentFraction.toString()
               + " is " + currentFraction.reciprocal() + "\n")
            currentFraction <-- currentFraction.reciprocal()
         currentString <-- stringScan.next()


*/
/*
     Data Table for readLine

```

```
Variable or Constant     Purpose
_____________________    ___________________
args                     parameter for readLine
firstNum                 storest the value for the firs numerator
secondNum                stores the value for the first denominator
*/
   private static void readLine(String currentLine, FileWriter writerOutput)
      throws Exception
   {
      writerOutput.write(currentLine + "\n");
      currentLine = currentLine.replace('/', ' ');
      Scanner stringScan = new Scanner(currentLine);
      Scanner numberScan = new Scanner(currentLine);
      if (!((currentLine.charAt(0)>= '0' && currentLine.charAt(0) <= '9')
            || currentLine.charAt(0) == '-'))
      {
         numberScan.next();
      }
      int firstNum = numberScan.nextInt();
      int secondNum = numberScan.nextInt();

      String currentString = new String(stringScan.next());

      Fraction currentFraction = new Fraction(firstNum, secondNum);




      while (stringScan.hasNext())
      {

         if (currentString.equals("add"))
         {
            numberScan.next();
            Fraction nextFraction = new Fraction(numberScan.nextInt(), numberScan.nextInt()
);
            currentFraction = readAdd(writerOutput, currentFraction, nextFraction);
         }
         else if (currentString.equals("sub"))
         {
            numberScan.next();
            Fraction nextFraction = new Fraction(numberScan.nextInt(), numberScan.nextInt()
);
            currentFraction = readSubtract(writerOutput, currentFraction, nextFraction);
         }
         else if (currentString.equals("mul"))
         {
            numberScan.next();
            Fraction nextFraction = new Fraction(numberScan.nextInt(), numberScan.nextInt()
);
            currentFraction = readMultiply(writerOutput, currentFraction, nextFraction);
         }
         else if (currentString.equals("div"))
         {
            numberScan.next();
            Fraction nextFraction = new Fraction(numberScan.nextInt(), numberScan.nextInt()
);
            currentFraction = readDivide(writerOutput, currentFraction, nextFraction);
         }
         else if (currentString.equals("rec"))
         {
            writerOutput.write("\tthe reciprocal of " + currentFraction.toString()
               + " is " + currentFraction.reciprocal() + "\n");
            currentFraction = currentFraction.reciprocal();
         }
         currentString = stringScan.next();
      }

```

```
   }//readLine


/*
readAdd(FileWriter writerOutput, Fraction currentFraction, Fraction nextFraction)
   writerOutput.write("\t" + currentFraction.toString() + "\n")
   writerOutput.write("\tadd " + nextFraction.toString() + " equals "
      + currentFraction.add(nextFraction) + "\n")
   currentFraction = currentFraction.add(nextFraction)
   return currentFraction
*/
/*
     Data Table for readAdd

Variable or Constant     Purpose
____________________     ___________________
args                     parameter for readLine

*/
   private static Fraction readAdd(FileWriter writerOutput,
      Fraction currentFraction, Fraction nextFraction) throws Exception
   {
      writerOutput.write("\t" + currentFraction.toString() + "\n");
      writerOutput.write("\tadd " + nextFraction.toString() + " equals "
         + currentFraction.add(nextFraction) + "\n");
      currentFraction = currentFraction.add(nextFraction);
      return currentFraction;
   }//readAdd


/*
readSubtract(FileWriter writerOutput, Fraction currentFraction, Fraction nextFraction)
   writerOutput.write("\t" + currentFraction.toString() + "\n")
   writerOutput.write("\tsubtract " + nextFraction.toString() + " equals "
      + currentFraction.subtract(nextFraction) + "\n")
   currentFraction = currentFraction.subtract(nextFraction)
   return currentFraction
*/
/*
     Data Table for readSubtract

Variable or Constant     Purpose
____________________     ___________________
args                     parameter for readLine

*/
   private static Fraction readSubtract(FileWriter writerOutput,
      Fraction currentFraction, Fraction nextFraction) throws Exception
   {
      writerOutput.write("\t" + currentFraction.toString() + "\n");
      writerOutput.write("\tsubtract " + nextFraction.toString() + " equals "
         + currentFraction.subtract(nextFraction) + "\n");
      currentFraction = currentFraction.subtract(nextFraction);
      return currentFraction;
   }//readSubtract


/*
readMultiply(FileWriter writerOutput, Fraction currentFraction, Fraction nextFraction)
   writerOutput.write("\t" + currentFraction.toString() + "\n")
   writerOutput.write("\tmultiply by " + nextFraction.toString() + " equals "
      + currentFraction.multiply(nextFraction) + "\n")
   currentFraction = currentFraction.multiply(nextFraction)
   return currentFraction
*/
/*
     Data Table for readMultiply

```

```
Variable or Constant     Purpose
_____________________    ___________________
args                     parameter for readLine

*/
   private static Fraction readMultiply(FileWriter writerOutput,
      Fraction currentFraction, Fraction nextFraction) throws Exception
   {
      writerOutput.write("\t" + currentFraction.toString() + "\n");
      writerOutput.write("\tmultiply by " + nextFraction.toString() + " equals "
         + currentFraction.multiply(nextFraction) + "\n");
      currentFraction = currentFraction.multiply(nextFraction);
      return currentFraction;
   }//readMultiply


/*
readDivide(FileWriter writerOutput, Fraction currentFraction, Fraction nextFraction)
   writerOutput.write("\t" + currentFraction.toString() + "\n")
   writerOutput.write("\tdivide by " + nextFraction.toString() + " equals "
      + currentFraction.divide(nextFraction) + "\n")
   currentFraction = currentFraction.divide(nextFraction)
   return currentFraction
*/
/*
     Data Table for readDivide

Variable or Constant     Purpose
_____________________    ___________________
args                     parameter for readLine

*/
   private static Fraction readDivide(FileWriter writerOutput,
      Fraction currentFraction, Fraction nextFraction) throws Exception
   {
   writerOutput.write("\t" + currentFraction.toString() + "\n");
   writerOutput.write("\tdivide by " + nextFraction.toString() + " equals "
      + currentFraction.divide(nextFraction) + "\n");
   currentFraction = currentFraction.divide(nextFraction);
   return currentFraction;
   }//readDivide


}//class FractionMath
```