FractionMath Version 4 UML
Author: Caspian Peavyhouse
CS101-02

Legend

(+) public
(-) private
( ) package
(#) protected

(+) FractionMath
(+) main() (-)readLine(currentLine:String, writerOutput:FileWriter) (-) readAdd(writerOutput:FileWriter, currentFraction:Fraction, nextFraction:Fraction) (-) readSubtract(writerOutput:FileWriter, currentFraction:Fraction, nextFraction:Fraction) (-) readMultiply(writerOutput:FileWriter, currentFraction:Fraction, nextFraction:Fraction) (-) readDivide(writerOutput:FileWriter, currentFraction:Fraction, nextFraction:Fraction)

Data Table for main

Algorithms:

```
main()
        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())
        writerOutput.close()

readLine(String currentLine, FileWriter writerOutput)
        writerOutput write(currentLine + \n)

        currentLine = currentLine.replace('/', ' ')
```

```
Scanner stringScan <-- new Scanner(currentLine)
Scanner numberScan <-- new Scanner(currentLine)
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()
```

Algorithm for readAdd

```
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
```

Algorithm for readSubtract

```
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
```

Algorithm for readMultiply

```
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
```

Algorithm for readDivide

```
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
```