Written by Caspian Peavyhouse
CS101-02
11/19/2014

(+) Fraction
(-) numerator: int (-) denominator: int (-) undefined: boolean (+) DEFAULT_NUMERATOR = 1: int (+) DEFAULT_DENOMINATOR = 1: int
(+) Fraction(numerator: int, denominator:int) (+) Fraction() (+) add(addend:Fraction): Fraction (+) subtract(subtrahend: Fraction): Fraction (+) multiply(multiplier: Fraction): Fraction (+) divide(divisor:Fraction) : Fraction (+) reciprocal(): Fraction (+) greatestCommonDivisor(inputOne: int, inputTwo:int): int (+) getNumerator(): int (+) getDenominator(): int (+) toString(): String

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

Data Table for Fraction

Variable or Constant	*Description*
numerator	Stores the value of the numerator of the fraction as an int
denominator	Stores the value of the denominator of the fraction as an int
undefined	Stores the boolean value of true if a fraction is undefined
DEFAULT_NUMERATOR	The constant int numerator value of the default fraction
DEFAULT_DENOMINATOR	The constant int denominator value of the default fraction

Algorithms

```
Fraction(numerator, denominator)
        int divisor <-- Fraction.greatestCommonDivisor(numerator, denominator)
        this.numerator <-- (numerator / divisor)
        this.denominator <-- (denominator / divisor)
        if (this.denominator < 0)
                this.denominator *= -1
                this.numerator *= -1
        if (denominator == 0)
                this.undefined <-- true
        else
                this.undefined <-- false
```

```
Fraction()
        numerator <-- DEFAULT_NUMERATOR
        denominator <--DEFAULT_DENOMINATOR

add(addend)
        int newDenominator <-- (this.getDenominator() * addend.getDenominator())
        int convertedNumerator1 <-- (this.getNumerator() * addend.getDenominator())
        int convertedNumerator2 <-- (addend.getNumerator() * this.getDenominator())
        int newNumerator <-- convertedNumerator1 + convertedNumerator2

        Fraction newFration <--new Fraction(newNumerator, newDenominator)
        return newFraction

subtract(subtrahend)
        int newDenominator <-- (this.getDenominator() * subtrahend.getDenominator())
        int convertedNumerator1 <-- (this.getNumerator() * subtrahend.getDenominator())
        int convertedNumerator2 <-- (subtrahend.getNumerator() * this.getDenominator())
        int newNumerator <-- convertedNumerator1 - convertedNumerator2

        Fraction newFration <--new Fraction(newNumerator, newDenominator)
        return newFraction

multiply(multiplier)
        int newDenominator <-- (this.getDenominator() * multiplier.getDenominator())
        int newNumerator <-- (this.getNumerator() * multiplier.getNumerator())

        Fraction newFration <--new Fraction(newNumerator, newDenominator)
        return newFraction

divide(divisor)
        flipDivisor <-- divisor.reciprocal(divisor)
        Fraction newFraction <-- new Fraction()
        newFraction <-- this.multiply(flipDivisor)
        return newFraction

reciprocal()
        int newNumerator <-- this.getDenominator
        int newDenominator <-- this.getNumerator
        Fraction newFraction <-- new Fraction(newNumerator, newDenominator)
        return newFraction

greatestCommonDivisor(int inputOne, int inputTwo)
        inputOne <-- Math.abs(inputOne)
        inputTwo <-- Math.abs(inputTwo)

        int smaller <--  Math.min(inputOne, inputTwo)
        int greatestCommonDivisor <-- 1
```

```
        for (int index = 0; index <= smaller; index++)
                if ((inputOne % index) == 0 && (inputTwo % index) == 0)
                        greatestCommonDivisor <-- index
        return greatestCommonDivisor



getNumerator()
        return this.numerator

getDenominator()
        return this.denominator

toString()
        String output

        if (this.undefined == true)
                output <-- "Undefined"
        else
                output <-- "" + this.getNumerator() + " / " + this.getDenominator()

        return output
```