COSC 2P13 – Assignment 1

Matt Laidman

5199807 (ml12ef)

February 13, 2015

Table of Contents

[Introduction/Execution: 2](#_Toc411516809)

[Design Choices: 4](#_Toc411516810)

[Source Code 6](#_Toc411516811)

# Introduction/Execution:

The RationalCalc.jar program has been written for Assignment 1 of COSC 2P13. The program will accept input in the form of any arithmetic on rational numbers with integer numerator and denominator. The program also supports the storing of rational numbers as variables which can later be used in the arithmetic.

The program was compiled using Java 1.8, and can be executed vie the command line with the command:

java –jar RationalCalc.jar

An example execution of the program is as follows:

> 7/10 + 18/5

43/10

> 7/10 \* 3

21/10

> -7/10 - -11/6

17/15

> -7/10 / 1/15

-21/2

> 1/2 < 1/3

False

> 1/2 <= 1/3

False

> 1/2 <= 1/2

True

> -1/2 > -1/3

False

> -1/3 < 1/3

True

> 1/2 == 1/2

True

> 1/3 != 1/2

True

> 1/2 == -1/2

False

> f := 1/4

> g := 1/2

> f \* g

> f + 1/2

> f

5/8

> decimal f

0.625

> f <= g

False

> \0

Implemented operations are; add, subtract, multiply, divide, equal, not equal, less than, greater than, less than or equal to, and greater than or equal to. Each operation has a symbol associated with it; +, -. \*, /, ==, !=, <, >, <=, and >= respectively. There is also one special operator, := which will assign a fraction to an alphabetical character (a-z, A-Z) which can then be used as if it were a rational.

The command “decimal” followed by a rational or variable will divide the rational and print the decimal representation of it to the screen.

# 

# Design Choices:

The calculator uses the RationalNum data type written specifically for this purpose. All source code can be found within the jar. It was also submitted alongside the jar, and has been included in the back of this document. Certain snippets of the source code will be shown below.

Listing of .java files:

* Package:Rationals
  + Rational.java
    - Interface for the RationalNum class, as per the specifications included in the rationals.doc file found on the assignment information web page.
  + RationalNum.java
    - Implementation of the Rational.java interface. Further discussed below.
  + RationalException.java
    - Exception to be thrown by RationalNum.java
* Package: RationalCalc
  + RationalCalc.java
    - Command line interpreter for the RationalNum class, translates traditional arithmetic operators into appropriate function calls to RationalNum.
* Package: List
  + List.java
    - Basic Linked-List implemented for the storing of variables.
  + Node.java
    - Node class to store variables in the list.

The RationalNum class is where all rational arithmetic is handled. The RationalCalc class simply translates the traditional arithmetic symbols into function calls to the RationalNum class.

Algorithms used are as computationally “cheap” as possible. For example, all comparison operations are done using the same method below. The less than function has been used as an example:

public boolean lt (Rational r) {

return num\*r.den() < r.num()\*den;

}

This is a simple operation that is easily computed by the processor.

Addition and subtraction are done in an equally cheap manner, the add function has been included below as an example:

public void add (Rational r) {

num = num\*r.den() + r.num()\*den;

den = den\*r.den();

if (num > MAXIMUM || den > MAXIMUM ||

num < MAXIMUM || den < MAXIMUM) {

normal();

}

}

The MAXIMUM variable has been set to the square root of the maximum integer. If either the numerator of the denominator of any rational exceeds this number, the fraction will be reduced if possible. This is done to help avoid the possibility of overflow.

Whenever the rational number is printed to the screen, it is normalized if possible as well.

The normalize function simply ensures that the negative sign is in the numerator, and that if both the numerator and denominator are negative then the negative signs are dropped and the rational is made positive. It then calls the private getGCD function to divide the numerator and denominator by their greatest common divisor.

The getGCD function is also as computationally “cheap” as possible as it uses a single modulo function. This can be seen below:

private int getGCD (int n, int d) {

int t;

if (n < 0) {

n = 0 - n;

}

if (d < 0) {

d = 0 - d;

}

while (d != 0) {

t = d;

d = n % d;

n = t;

}

return n;

}

# Source Code

The source files have been included in the following order:

1. RationalCalc.java
2. Rational.java
3. RationalNum.java
4. RationalException.java
5. List.java
6. Node.java

A sample execution has also been included at the back.