I defined a class Rational and two private attributes numerator and denominator which are of type BigInt. I have chosen BigInt so that same program can be used for very big numbers. I defined one auxiliary constructor so that whole number can be used. Defined the following method:

sum: There are two overloaded methods, first one take Rational as a parameter and returns sum as a rational number. Second one takes BigInt as parameter and returns sum as a Rational number. Sum is computed by:

(Numerator of Object \* Denominator of parameter + Denominator of Object \* Numerator of parameter ) / (Denominator of Object \* Denominator of Parameter)

subtract: There are two overloaded methods, first one take Rational as a parameter and returns subtracts from the object and return as a rational number. Second one takes BigInt as parameter and subtracts the number from the object and returns as a Rational number . Subtract is computed by:

(Numerator of Object \* Denominator of parameter - Denominator of Object \* Numerator of parameter ) / (Denominator of Object \* Denominator of Parameter)

multiply: There are two overloaded methods, first one take Rational as a parameter and returns multiplies with the object and return as a rational number. Second one takes BigInt as parameter and multiply with the number from the object and returns as a Rational number. Muliply is computed by:

(Numerator of Object \* Numerator of Parameter) / (Denominator of Object \* Denominator of Parameter)

divide: There are two overloaded methods, first one take Rational as a parameter and divides from the object and return as a rational number. Second one takes BigInt as parameter and divides from the object and returns as a Rational number :

(Numerator of Object \* Denominator of Parameter) / (Denominator of Object \* Numerator of Parameter)

gcd: There are two overloaded methods, first one take Rational as a parameter and finds gcd with the object and return as a rational number. Second one takes BigInt as parameter and finds gcd with the object and returns as a Rational number. Gcd of Rational Number is calculated by:

GCD of numerator of Object and Parameter/ LCM of denominator of Object and Parameter

LCM is computed using a method compute\_lcm which takes two BigInt numbers and computes the LCM, by repeatedly summing till both the numbers are equal

GCD is computed by a method compute\_gcd, This method first first finds greater of two numbers and assign to first\_number and second\_number respectively. Remainder is calculated by using modulus operator (%) between first\_number and second\_number. This steps is done repeated in while loop till remainder is 0. When remainder becomes 0, first\_number is returned as the gcd.

I have also defined a method printObject which will print the numerator and denominator value of Rational object

I have written one singleton Object RationalMain having main method. Here I have taken two sets of Rational Number. In one set I have taken wo Rational Number 15/12 and 6/8 and calculated sum, subtract, multiply, divide, gcd

In another set I have taken two whole Numbers 15 and 6 and computed sum, subtract, multiply, divide, gcd

I have created a file RationalMain.scala and its content is as below:

class Rational (x:BigInt, y:BigInt) {

private val numerator:BigInt = x

private val denominator:BigInt = y

def this(a:BigInt) = this(a, 1)

def sum(b: Rational):Rational = {

return new Rational(numerator \* b.denominator + denominator \* b.numerator, denominator \* b.denominator)

}

def sum(b: BigInt):Rational = {

return new Rational(numerator + b, 1)

}

def subtract(b: Rational):Rational = {

return new Rational(numerator \* b.denominator - denominator \* b.numerator, denominator \* b.denominator)

}

def subtract(b: BigInt):Rational = {

return new Rational(numerator - b, 1)

}

def multiply(b: Rational):Rational = {

return new Rational(numerator \* b.numerator, denominator \* b.denominator)

}

def multiply(b: BigInt):Rational = {

return new Rational(numerator \* b, 1)

}

def divide(b: Rational):Rational = {

return new Rational(numerator \* b.denominator, denominator \* b.numerator)

}

def devide(b: BigInt):Rational = {

return new Rational(numerator / b, 1)

}

def compute\_lcm(m: BigInt, n:BigInt):BigInt = {

var a = m

var b = n

while ( a != b) {

if (a<b ) a = a + m

else b = b + n

}

return a

}

def compute\_gcd(a:BigInt, b:BigInt) : BigInt = {

var first\_number:BigInt = 0

var second\_number:BigInt = 0

if (a>b) {

first\_number = a

second\_number = b

} else {

first\_number = b

second\_number = a

}

var remainder:BigInt = 1

while (remainder != 0) {

remainder = first\_number % second\_number

first\_number = second\_number

second\_number = remainder

}

return first\_number

}

def gcd(b:Rational) : Rational = {

val x:BigInt = compute\_gcd(numerator, b.numerator)

val y:BigInt = compute\_lcm(denominator, b.denominator)

return new Rational(x, y)

}

def gcd(b:BigInt) : Rational = {

val z:Rational = new Rational(b, 1)

return gcd(y)

}

def printObject = println("numerator =" + numerator + " denominator=" + denominator)

}

object RationalMain {

def main(args: Array[String]):Unit = {

val r1 = new Rational(15,12)

println("For RationaL Number r1")

r1.printObject

val r2 = new Rational(6,8)

println("For RationaL Number r2")

r2.printObject

val sum\_r1\_r2 = r1.sum(r2)

println(" For r1 + r2: ")

sum\_r1\_r2.printObject

val subtract\_r1\_r2 = r1.subtract(r2)

println(" For r1 - r2: ")

sum\_r1\_r2.printObject

subtract\_r1\_r2.printObject

val multiply\_r1\_r2 = r1.multiply(r2)

println(" For r1 \* r2: ")

multiply\_r1\_r2.printObject

val divide\_r1\_r2 = r1.divide(r2)

println( " For r1 / r2: ")

divide\_r1\_r2.printObject

val gcd\_r1\_r2 = r1.gcd(r2)

println (" gcd r1 and r2: ")

gcd\_r1\_r2.printObject

val r3 = new Rational(15)

println("For RationaL Number r3")

r3.printObject

val r4 = new Rational(6)

println("For RationaL Number r4")

r4.printObject

val sum\_r3\_r4 = r3.sum(r4)

println(" For r3 + r4: ")

sum\_r3\_r4.printObject

val subtract\_r3\_r4 = r3.subtract(r4)

println(" For r3 - r4: ")

sum\_r3\_r4.printObject

subtract\_r3\_r4.printObject

val multiply\_r3\_r4 = r3.multiply(r4)

println(" For r3 \* r4: ")

multiply\_r3\_r4.printObject

val divide\_r3\_r4 = r3.divide(r4)

println(" For r3 / r4: ")

divide\_r3\_r4.printObject

val gcd\_r3\_r4 = r3.gcd(r4)

println(" gcd r3 and r4: ")

gcd\_r3\_r4.printObject

}

}

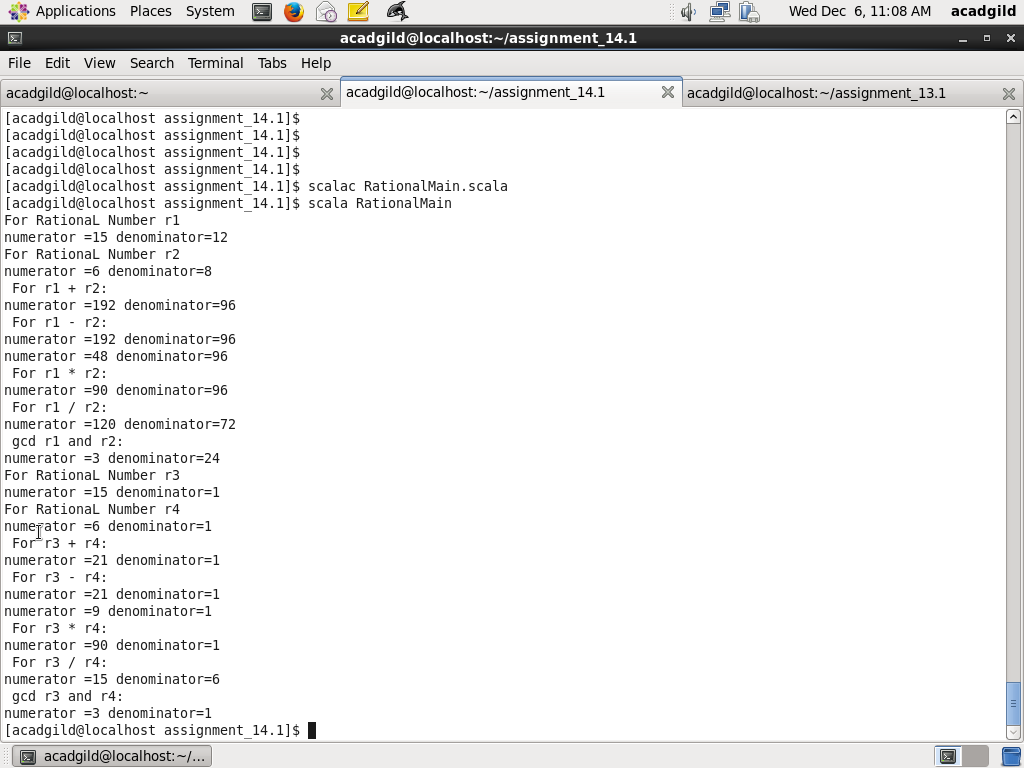
I have compiled the code using

scalac RationalMain.scala

And Run using:

scala RationalMain

The screenshot of compilation and output is as below:



Screenshots of Source Code is as below:

