Assignment 16.1

Task 1

Create a calculator to work with rational numbers.

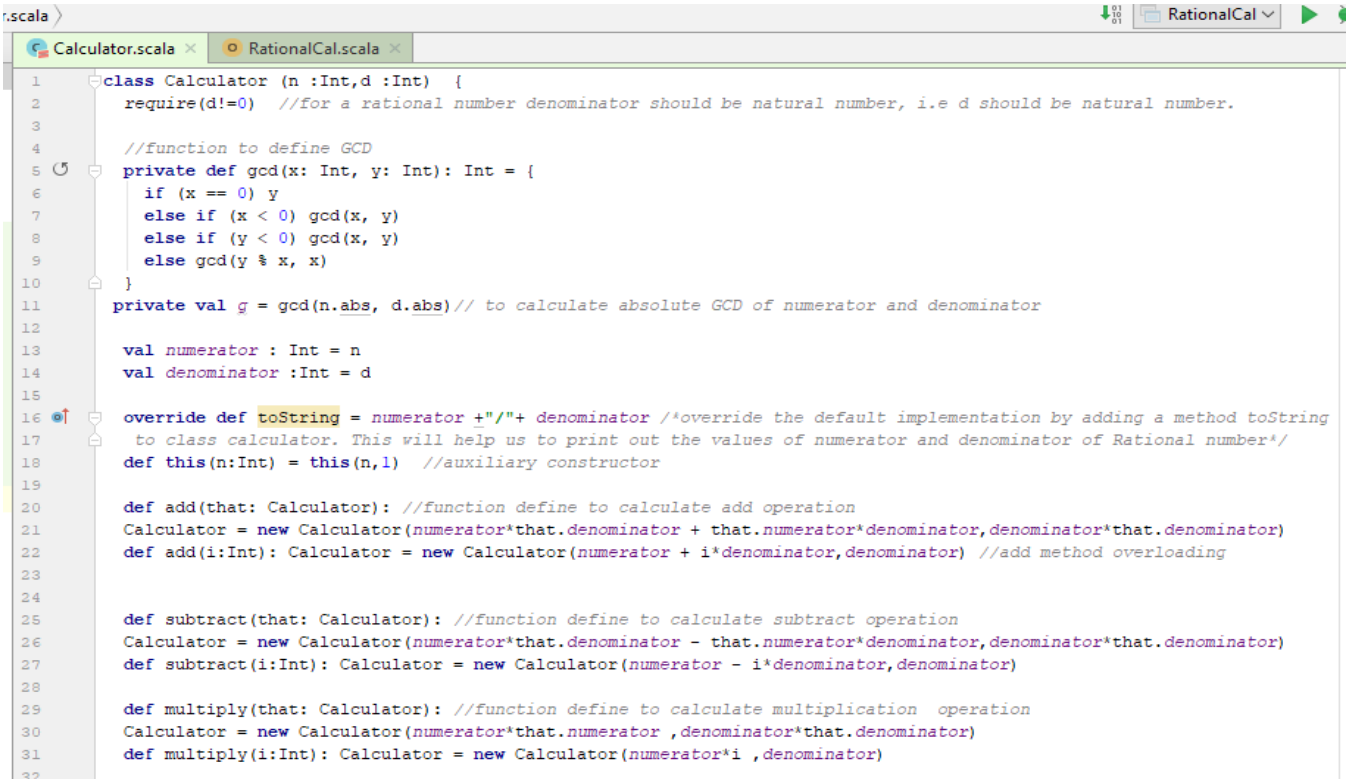
Requirements:

- **It should provide capability to add, subtract, divide and multiply rational Numbers.**
- **Create a method to compute GCD (this will come in handy during operations on Rational).**

Add option to work with whole numbers, which are also rational numbers i.e. (n/1)

- **Achieve the above using auxiliary constructors**
- **Enable method overloading to enable each function to work with numbers and rational.**

Screen shot show the program for calculator to work with rational number:




```
class Calculator (n :Int,d :Int)  {
  require(d!=0)  //for a rational number denominator should be natural number, i.e d should be natural number.

  //function to define GCD
  private def gcd(x: Int, y: Int): Int = {
    if (x == 0) y
    else if (x < 0) gcd(x, y)
    else if (y < 0) gcd(x, y)
    else gcd(y % x, x)
  }
 private val g = gcd(n.abs, d.abs)// to calculate absolute GCD of numerator and denominator

  val numerator : Int = n
  val denominator :Int = d

  override def toString = numerator +"/"+ denominator /*override the default implementation by adding a method toString
   to class calculator. This will help us to print out the values of numerator and denominator of Rational number*/
  def this(n:Int) = this(n,1)  //auxiliary constructor

  def add(that: Calculator): //function define to calculate add operation
  Calculator = new Calculator(numerator*that.denominator + that.numerator*denominator,denominator*that.denominator)
  def add(i:Int): Calculator = new Calculator(numerator + i*denominator,denominator) //add method overloading


  def subtract(that: Calculator): //function define to calculate subtract operation
  Calculator = new Calculator(numerator*that.denominator - that.numerator*denominator,denominator*that.denominator)
  def subtract(i:Int): Calculator = new Calculator(numerator - i*denominator,denominator)

  def multiply(that: Calculator): //function define to calculate multiplication  operation
  Calculator = new Calculator(numerator*that.numerator ,denominator*that.denominator)
  def multiply(i:Int): Calculator = new Calculator(numerator*i ,denominator)
```

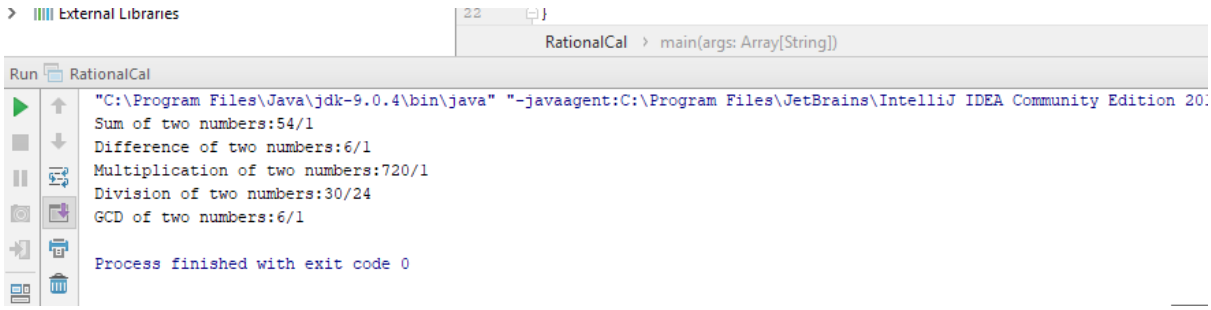
```
32
33
34      def division(that: Calculator): //function define to calculate division  operation
35      Calculator = new Calculator(numerator*that.denominator ,denominator*that.numerator)
36      def division(i:Int): Calculator = new Calculator(numerator ,denominator*i)
37
38
39      def gcd_Rational(that: Calculator): //function define to calculate GCD operation for two rational numbers
40      Calculator = new Calculator((gcd(numerator*that.denominator,denominator*that.numerator))/(denominator*that.denominator))
41      def gcd_rational(i:Int) : Calculator = new Calculator(gcd(numerator/denominator,i))
42  }
43
```

Screen shot below shows the object created for add, subtract, multiplication, division and gcd on rational numbers.

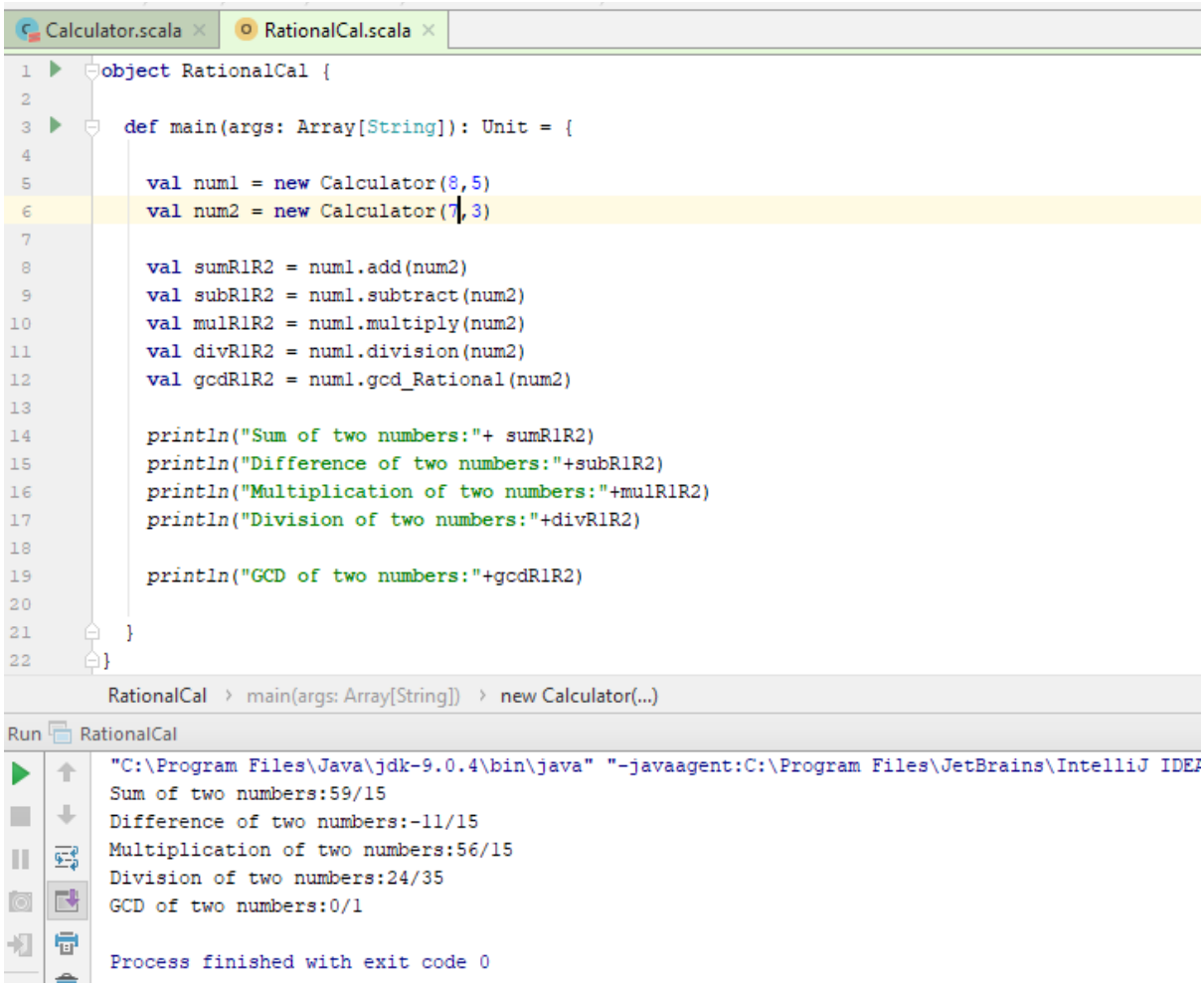
Calculator.scala | RationalCal.scala

```
1   object RationalCal {
2
3     def main(args: Array[String]): Unit = {
4
5       val num1 = new Calculator(30)
6       val num2 = new Calculator(24)
7
8       val sumR1R2 = num1.add(num2)
9       val subR1R2 = num1.subtract(num2)
10      val mulR1R2 = num1.multiply(num2)
11      val divR1R2 = num1.division(num2)
12      val gcdR1R2 = num1.gcd_Rational(num2)
13
14      println("Sum of two numbers:"+ sumR1R2)
15      println("Difference of two numbers:"+subR1R2)
16      println("Multiplication of two numbers:"+mulR1R2)
17      println("Division of two numbers:"+divR1R2)
18
19      println("GCD of two numbers:"+gcdR1R2)
20
21    }
22  }
23
```

Sample output of above object:



Output with rational numbers, num1=8/5 and num2=7/3



Output with rational numbers, num1=27 and num2= 18