Session 15 - Scala Session - II

Assignment 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.

```
package acadgild.calculator

object Rational {
  def main(args: Array[String]) {

    //calling the class's 2 constructors by differentiating the type of arguments passed to it
    var ration = new RationalCalculator1("2/3", "*", "1/4") //for rational numbers
    var ration1 = new RationalCalculator1(2, "*", 3) //for whole numbers
  }
}


    //class definition
class RationalCalculator1() { //primary constructor with 0 args. (because, auxiliary
                              //constructors should call a primary or formerly defined
                                  //constructor)
  def this(op1: String, sign: String, op2: String) { //auxiiliary constructor for rational numbers,
                                                     //accepted as strings
    this()                                                 //calling the primary constructor

    var x = op1
    var y = op2
    var xArray: Array[String] = x.split("/")
```

```
var xnumerator = xArray(0)
var xdenominator = xArray(1)

var yArray: Array[String] = y.split("/")

var ynumerator = yArray(0)                  //processing the string arguements to figure
var ydenominator = yArray(1)                //out the rational numbers

def gcd(a: Int, b: Int): Int = {            //method to find out gcd of the rational
  if (b == 0) a else gcd(b, a % b)          //numbers
}

var g1 = gcd(xnumerator.toInt, xdenominator.toInt);
var xnum = xnumerator.toInt / g1;
var xden = xdenominator.toInt / g1;

var g2 = gcd(ynumerator.toInt, ydenominator.toInt);   //calculating the gcd of both the
var ynum = ynumerator.toInt / g2;                     //operands before performing
var yden = ydenominator.toInt / g2;                   //operations on them

def calFraction(num: Int, den: Int, g: Int): String = {   //calculating the fraction form of
  var resnum = num / g                                    //the result based on the gcd
  var resden = den / g

  return (resnum + "/" + resden)
}

if (sign.equals("+")) {                         //calling approproate method based on
  add(xnum, xden, ynum, yden)                   //the sign specified in the input
} else if (sign.equals("-")) {
  subtract(xnum, xden, ynum, yden)
} else if (sign.equals("*")) {
  multiply(xnum, xden, ynum, yden)
} else if (sign.equals("/")) {
  divide(xnum, xden, ynum, yden)
}
  //mathematical calculations on the input parameters and printing the results
def multiply(xnum: Int, xden: Int, ynum: Int, yden: Int) = {
  var num = xnum * ynum
  var den = xden * yden
  var gcd1 = gcd(num, den)
  var res = calFraction(num, den, gcd1)
  println(xnum + "/" + xden + " * " + ynum + "/" + yden + " = " + res)
}

def add(xnum: Int, xden: Int, ynum: Int, yden: Int) = {
  var num = (xnum * yden) + (xden * ynum)
```

```
    var den = xden * yden
    var gcd1 = gcd(num, den)
    var res = calFraction(num, den, gcd1)
    println(xnum + "/" + xden + " + " + ynum + "/" + yden + " = " + res)
  }

  def subtract(xnum: Int, xden: Int, ynum: Int, yden: Int) = {
    var num = (xnum * yden) - (xden * ynum)
    var den = xden * yden
    var gcd1 = gcd(num, den)
    var res = calFraction(num, den, gcd1)
    println(xnum + "/" + xden + " - " + ynum + "/" + yden + " = " + res)
  }

  def divide(xnum: Int, xden: Int, ynum: Int, yden: Int) = {
    var num = (xnum * yden)
    var den = xden * ynum
    var gcd1 = gcd(num, den)
    var res = calFraction(num, den, gcd1)
    println(xnum + "/" + xden + " / " + ynum + "/" + yden + " = " + res)
  }
}

def this(op1: Int, sign: String, op2: Int) {    //auxiiliary constructor for wholenumbers,
                                                //accepted as Integers
  this()                                        //calling the primary constructor

  if (sign.equals("+")) {                       //calling approproate method based on
    add(op1, op2)                               //the sign specified in the input
  } else if (sign.equals("-")) {
    subtract(op1, op2)
  } else if (sign.equals("*")) {
    multiply(op1, op2)
  } else if (sign.equals("/")) {
    divide(op1, op2)
  }
    //mathematical calculations on the input parameters and printing the results

    //these methods have the same names as those for the rational numbers and both the
    //variants are in the same class.
  def add(op1: Int, op2: Int) = {
    var res = op1 + op2;
    println(res)
  }
  def subtract(op1: Int, op2: Int) = {
    var res = op1 - op2
    println(res)
  }
```

```
    def multiply(op1: Int, op2: Int) = {
      var res = op1 * op2
      println(res)
    }
    def divide(op1: Int, op2: Int) = {
      var res = op1 / op2
      println(res)
    }
  }
}
```

Also included the Eclipse Scala project for this program.