

Assignment 14

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.

Scala Program Code:

```
class Calc (n:Int,d:Int) {
  require(d!=0)
  private val g = gcd(n.abs, d.abs)
  val numerator = n/g
  val denominator = d/g

  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)
  }
  def this(n:Int) = this(n,1)
  def add (r:Calc):Calc =
    new Calc(numerator * r.denominator + r.numerator*denominator,
denominator*r.denominator)
  def add (i:Int):Calc = new Calc(numerator + i * denominator, denominator)
  def subtract ( r:Calc) = new Calc (numerator*r.denominator -
r.numerator*denominator,denominator*r.denominator)
  def subtract (i: Int): Calc = new Calc(numerator - i * denominator, denominator)
  def multiply (r:Calc) = new Calc(numerator*r.numerator,denominator*r.denominator)
  def multiply (i: Int): Calc = new Calc(numerator * i , denominator)
  def divide (r:Calc) = new Calc(numerator*r.denominator,denominator*r.numerator)
  def divide (i: Int): Calc = new Calc(numerator , denominator * i)
  override def toString = numerator + "/" + denominator
}
```

```

1 class Calc (n:Int,d:Int) {
2     require(d!=0)
3     private val g = gcd(n.abs, d.abs)
4     val numerator = n/g
5     val denominator = d/g
6
7     private def gcd(x:Int, y:Int):Int = {
8         if(x==0) y
9         else if (x<0) gcd(-x,y)
10        else if (y<0) gcd(x,-y)
11        else gcd(y%x,x)
12    }
13    def this(n:Int) = this(n,1)
14    def add (r:Calc):Calc =
15        new Calc(numerator * r.denominator + r.numerator*denominator, denominator*r.denominator)
16    def add (i:Int):Calc = new Calc(numerator + i * denominator, denominator)
17    def subtract ( r:Calc) = new Calc (numerator*r.denominator - r.numerator*denominator,denominator*r.denominator)
18    def subtract (i: Int): Calc = new Calc(numerator - i * denominator, denominator)
19    def multiply (r:Calc) = new Calc(numerator*r.numerator,denominator*r.denominator)
20    def multiply (i: Int): Calc = new Calc(numerator * i , denominator)
21    def divide (r:Calc) = new Calc(numerator*r.denominator,denominator*r.numerator)
22    def divide (i: Int): Calc = new Calc(numerator , denominator * i)
23    override def toString = numerator + "/" + denominator
24 }
25

```

The statement: `def this(n:Int) = this(n,1)` works as auxiliary constructor

Below is the object class to run the above program code.

This statement enables us to work with whole numbers which are also rational numbers i.e.(n/1)

each function- add, subtract, multiply, divide has been defined in such a manner via method overloading that it allows the user to work with numbers and rational.

```

object CalcObj {

    def main(args: Array[String]): Unit = {
        val a = new Calc(29,10)
        val b = new Calc(17)
        val c = new Calc(18,2)
        val d = new Calc(11)
        val p = a add 5
        println(p)
        val q = b multiply new Calc(11,9)
        println(q)
        val r = c subtract new Calc(16,1)
        println(r)
        val s = d divide 51
        println(s)
    }
}

```

```

26
27 ▶ object CalcObj {
28
29 ▶ def main(args: Array[String]): Unit = {
30     val a = new Calc(29,10)
31     val b = new Calc(17)
32     val c = new Calc(18,2)
33     val d = new Calc(11)
34     val p = a add 5
35     println(p)
36     val q = b multiply new Calc(11,9)
37     println(q)
38     val r = c subtract new Calc(16,1)
39     println(r)
40     val s = d divide 51
41     println(s)
42 }
43 }

```

Output:

Run CalcObj (1)

```

"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
79/10
187/9
-7/1
11/51
Process finished with exit code 0

```