Assignment -16

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.**

The scala code to create a **calculator** to work with **rational numbers** is as shown below.

This calculator has the facility to **Add, Subtract, Multiply, Divide**.

build.sbt × CalcObj.scala ×

```scala
object CalcObj {

  def main(args: Array[String]): Unit = {
    val a = new Calc(10,9)
    val b = new Calc(17)

    val c = new Calc(13,26)
    val d = new Calc(11)
    val p = a add 5
    println("Addition : "+p)

    val q = b multiply new Calc(11,9)
    println("Multiply : "+q)

    val r = c subtract new Calc(16,1)
    println("Subtract : "+r)

    val s = d divide 51
    println("Divide : "+s)
  }

}
```

GCD method is created as shown in the below screenshot.

An auxiliary constructor is created using '**this**' keyword.

Every method is overloaded to work with **numbers** and **rational**.

build.sbt × CalcObj.scala ×

```scala
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) // auxiliary constructor

  def add (r:Calc): Calc =
    new Calc(numerator * r.denominator + r.numerator*denominator ,
      denominator*r.denominator)

  def add (i: Int): Calc = // overloaded for add
    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 = // overloaded for subtract
    new Calc(numerator - i * denominator, denominator)

  def multiply (r:Calc) =

    new Calc(numerator*r.numerator,denominator*r.denominator)

  def multiply (i: Int): Calc = // overloaded for multiply
  new Calc(numerator * i , denominator)

  def divide (r:Calc) = new
```

```
def divide (r:Calc) = new
    Calc(numerator*r.denominator,denominator*r.numerator)
def divide (i: Int): Calc = // overloaded for division
  new Calc(numerator , denominator * i)

override def toString : String = numerator + "/" + denominator

}
```

Output for the above calculator code is as shown below.

```
Scala Console   Scala Console   CalcObj
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Addition : 55/9
Multiply : 187/9
Subtract : -31/2
Divide : 11/51

Process finished with exit code 0
```