

Assignment -16

Task 1:

Create a calculator to work with rational numbers.

Requirements:

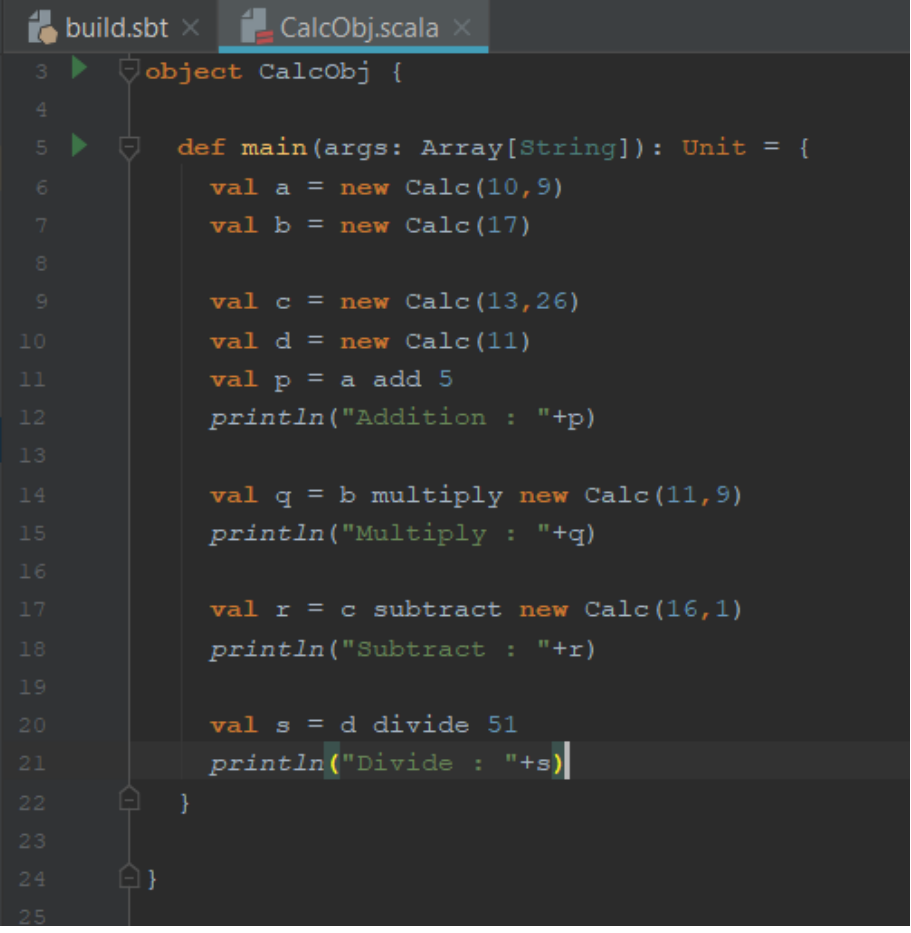
- **It should provide capability to add, subtract, divide and multiply rational Numbers**
- **Create a method to compute GCD (this will come in handy during operations on rational)**

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**.

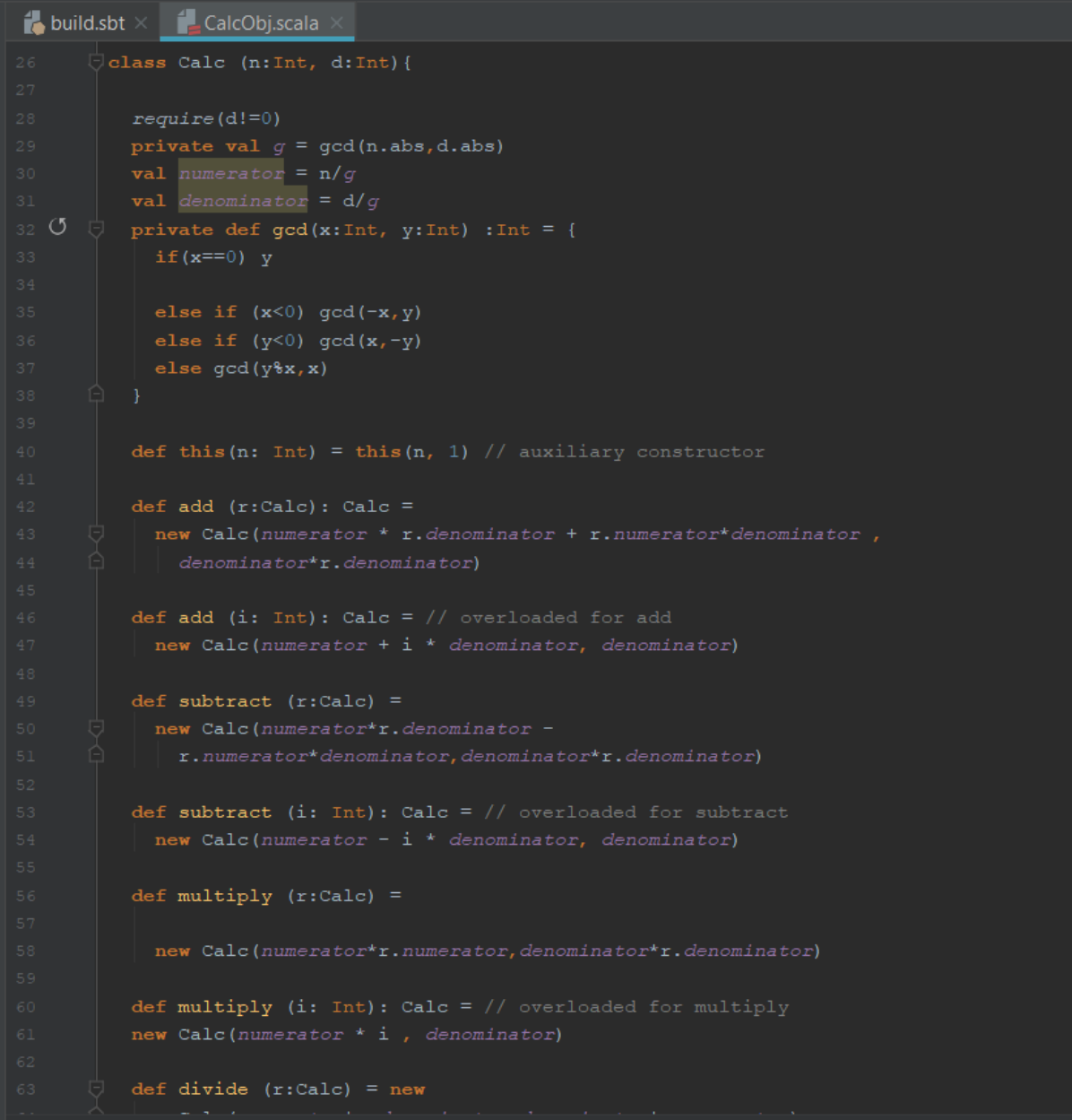


```
build.sbt x  CalcObj.scala x
3  ▶  object CalcObj {
4
5  ▶  def main(args: Array[String]): Unit = {
6      val a = new Calc(10,9)
7      val b = new Calc(17)
8
9      val c = new Calc(13,26)
10     val d = new Calc(11)
11     val p = a add 5
12     println("Addition : "+p)
13
14     val q = b multiply new Calc(11,9)
15     println("Multiply : "+q)
16
17     val r = c subtract new Calc(16,1)
18     println("Subtract : "+r)
19
20     val s = d divide 51
21     println("Divide : "+s)
22 }
23
24 }
25
```

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**.



```
26 class Calc (n:Int, d:Int){
27
28     require(d!=0)
29     private val g = gcd(n.abs,d.abs)
30     val numerator = n/g
31     val denominator = d/g
32     private def gcd(x:Int, y:Int) :Int = {
33         if(x==0) y
34
35         else if (x<0) gcd(-x,y)
36         else if (y<0) gcd(x,-y)
37         else gcd(y*x,x)
38     }
39
40     def this(n: Int) = this(n, 1) // auxiliary constructor
41
42     def add (r:Calc): Calc =
43         new Calc(numerator * r.denominator + r.numerator*denominator ,
44             denominator*r.denominator)
45
46     def add (i: Int): Calc = // overloaded for add
47         new Calc(numerator + i * denominator, denominator)
48
49     def subtract (r:Calc) =
50         new Calc(numerator*r.denominator -
51             r.numerator*denominator,denominator*r.denominator)
52
53     def subtract (i: Int): Calc = // overloaded for subtract
54         new Calc(numerator - i * denominator, denominator)
55
56     def multiply (r:Calc) =
57
58         new Calc(numerator*r.numerator,denominator*r.denominator)
59
60     def multiply (i: Int): Calc = // overloaded for multiply
61         new Calc(numerator * i , denominator)
62
63     def divide (r:Calc) = new
```

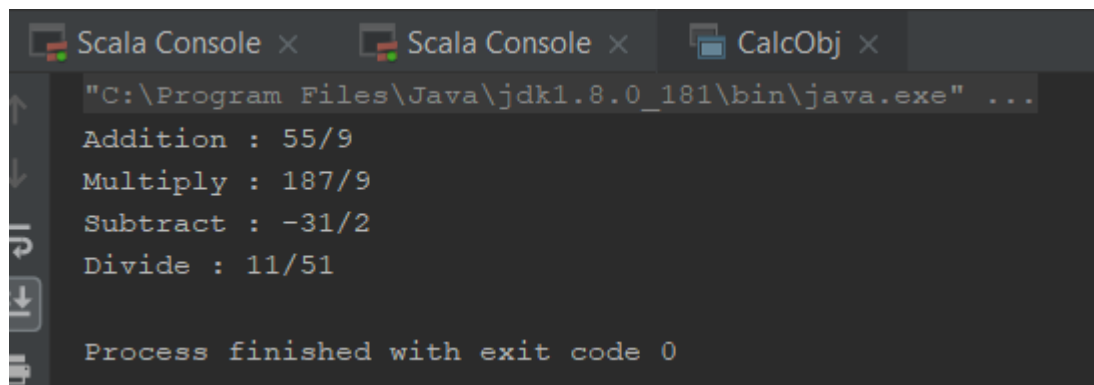
```

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.



The screenshot shows a Scala IDE with three tabs: "Scala Console", "Scala Console", and "CalcObj". The "Scala Console" tab is active, displaying the following output:

```

"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

```