SCALA - SESSION III

Assignment

Student Name: Subham Vishal

Course: Big Data Hadoop & Spark Training

Assignment

Create a calculator to work with rational numbers using Scala.

Contents

Introduction

In this assignment, we are going to write a SCALA code to create a Calculator to work with rational numbers,

Problem Statement

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.

Task 1 – Create a Scala Class *“Calc”*

Scala Code

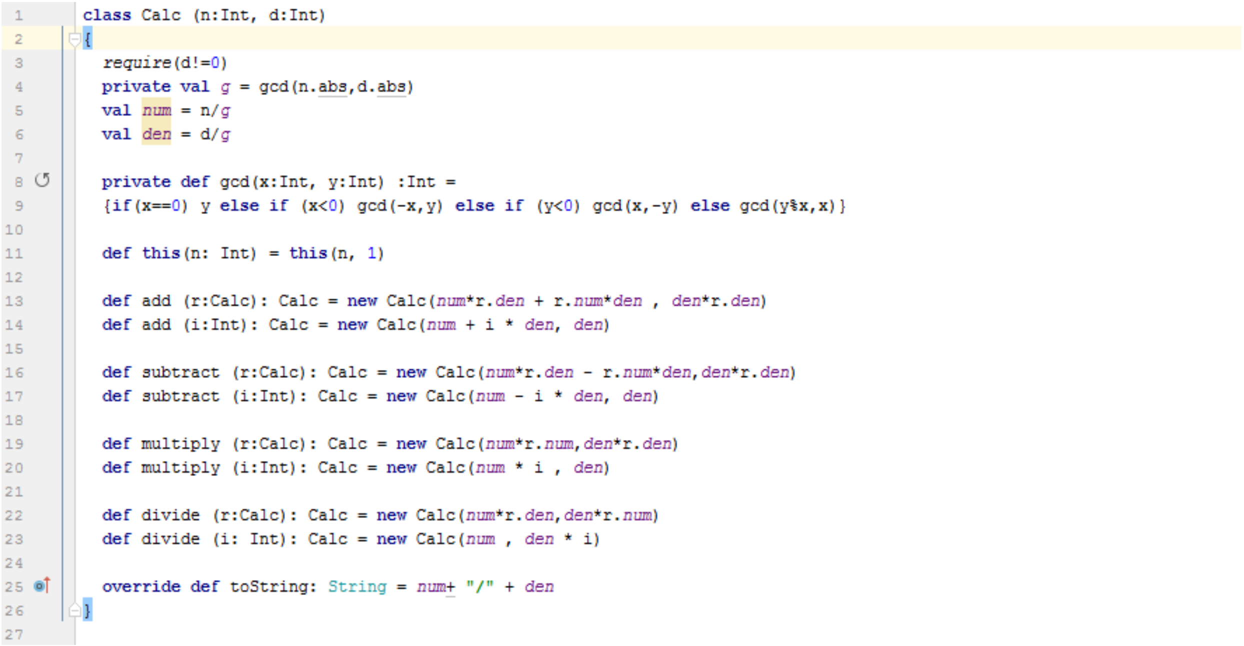
```
class Calc (n:Int, d:Int)
{
  require(d!=0)
  private val g = gcd(n.abs,d.abs)
  val num = n/g
  val den = d/g

  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(num*r.den + r.num*den , den*r.den)
  def add (i:Int): Calc = new Calc(num + i * den, den) //method overloading for add

  def subtract (r:Calc): Calc = new Calc(num*r.den - r.num*den,den*r.den)
  def subtract (i:Int): Calc = new Calc(num - i * den, den)//method overloading for
subtract

  def multiply (r:Calc): Calc = new Calc(num*r.num,den*r.den)
  def multiply (i:Int): Calc = new Calc(num * i , den)//method overloading for
multiplication

  def divide (r:Calc): Calc = new Calc(num*r.den,den*r.num)
  def divide (i: Int): Calc = new Calc(num , den * i)//method overloading for division

  override def toString: String = num+ "/" + den
```

The statement, ***“def this(n: Int) = this(n, 1) “*** is an auxiliary constructor, we have created an Object **“CalcObj”** to perform the above functions.

We have Enabled method **overloading** to enable each function (add, sub, multiplication and division) to work with numbers and rational.

We have written the code in such a way that it works with whole numbers as well as with rational numbers (n/1).

IntelliJ console,



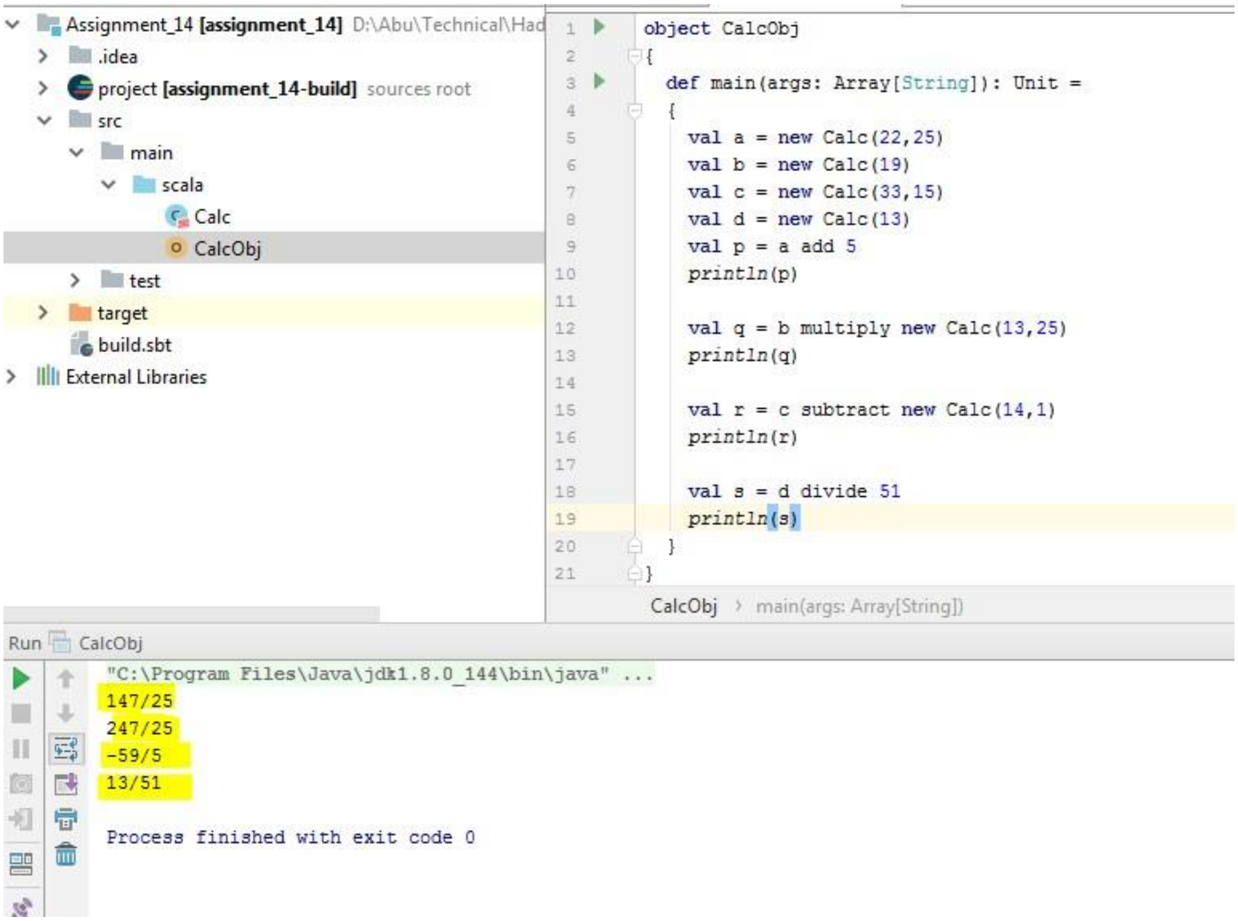
```scala
class Calc (n:Int, d:Int)
{
  require(d!=0)
  private val g = gcd(n.abs,d.abs)
  val num = n/g
  val den = d/g

  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(num*r.den + r.num*den , den*r.den)
  def add (i:Int): Calc = new Calc(num + i * den, den)

  def subtract (r:Calc): Calc = new Calc(num*r.den - r.num*den,den*r.den)
  def subtract (i:Int): Calc = new Calc(num - i * den, den)

  def multiply (r:Calc): Calc = new Calc(num*r.num,den*r.den)
  def multiply (i:Int): Calc = new Calc(num * i , den)

  def divide (r:Calc): Calc = new Calc(num*r.den,den*r.num)
  def divide (i: Int): Calc = new Calc(num , den * i)

  override def toString: String = num+ "/" + den
}
```

Task 2 – Create a Scala Object “CalObj”

```scala
object CalcObj
{
  def main(args: Array[String]): Unit =
  {
    val a = new Calc(22,25)
    val b = new Calc(19)
    val c = new Calc(33,15)
    val d = new Calc(13)

    val p = a add 5
    println(p)

    val q = b multiply new Calc(13,25)
    println(q)

    val r = c subtract new Calc(14,1)
    println(r)

    val s = d divide 51
    println(s)
  }

}
```

Expected Output

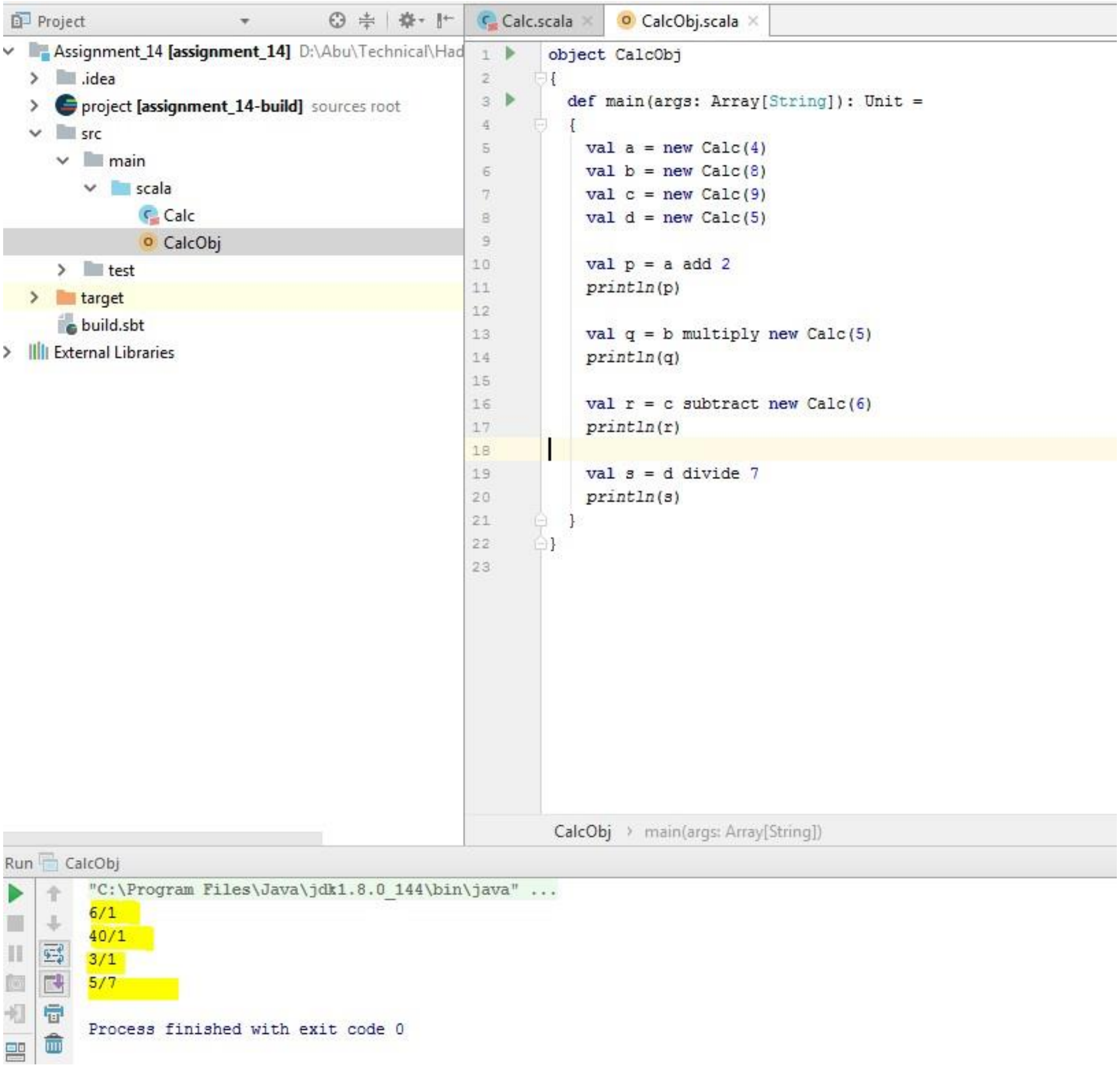
1. Example 1,


Assignment_14 [assignment_14] D:\Abu\Technical\Had
.idea
project [assignment_14-build] sources root
src
main
scala
Calc
CalcObj
test
target
build.sbt
External Libraries
CalcObj › main(args: Array[String])
Run CalcObj

```
object CalcObj
{
  def main(args: Array[String]): Unit =
  {
    val a = new Calc(22,25)
    val b = new Calc(19)
    val c = new Calc(33,15)
    val d = new Calc(13)
    val p = a add 5
    println(p)

    val q = b multiply new Calc(13,25)
    println(q)

    val r = c subtract new Calc(14,1)
    println(r)

    val s = d divide 51
    println(s)
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
147/25
247/25
-59/5
13/51

Process finished with exit code 0
```

2. Example 2,


Project
Assignment_14 [assignment_14] D:\Abu\Technical\Had
.idea
project [assignment_14-build] sources root
src
main
scala
Calc
CalcObj
test
target
build.sbt
External Libraries
Calc.scala
CalcObj.scala
CalcObj › main(args: Array[String])
Run CalcObj

```
object CalcObj
{
  def main(args: Array[String]): Unit =
  {
    val a = new Calc(4)
    val b = new Calc(8)
    val c = new Calc(9)
    val d = new Calc(5)

    val p = a add 2
    println(p)

    val q = b multiply new Calc(5)
    println(q)

    val r = c subtract new Calc(6)
    println(r)

    val s = d divide 7
    println(s)
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
6/1
40/1
3/1
5/7

Process finished with exit code 0
```