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)

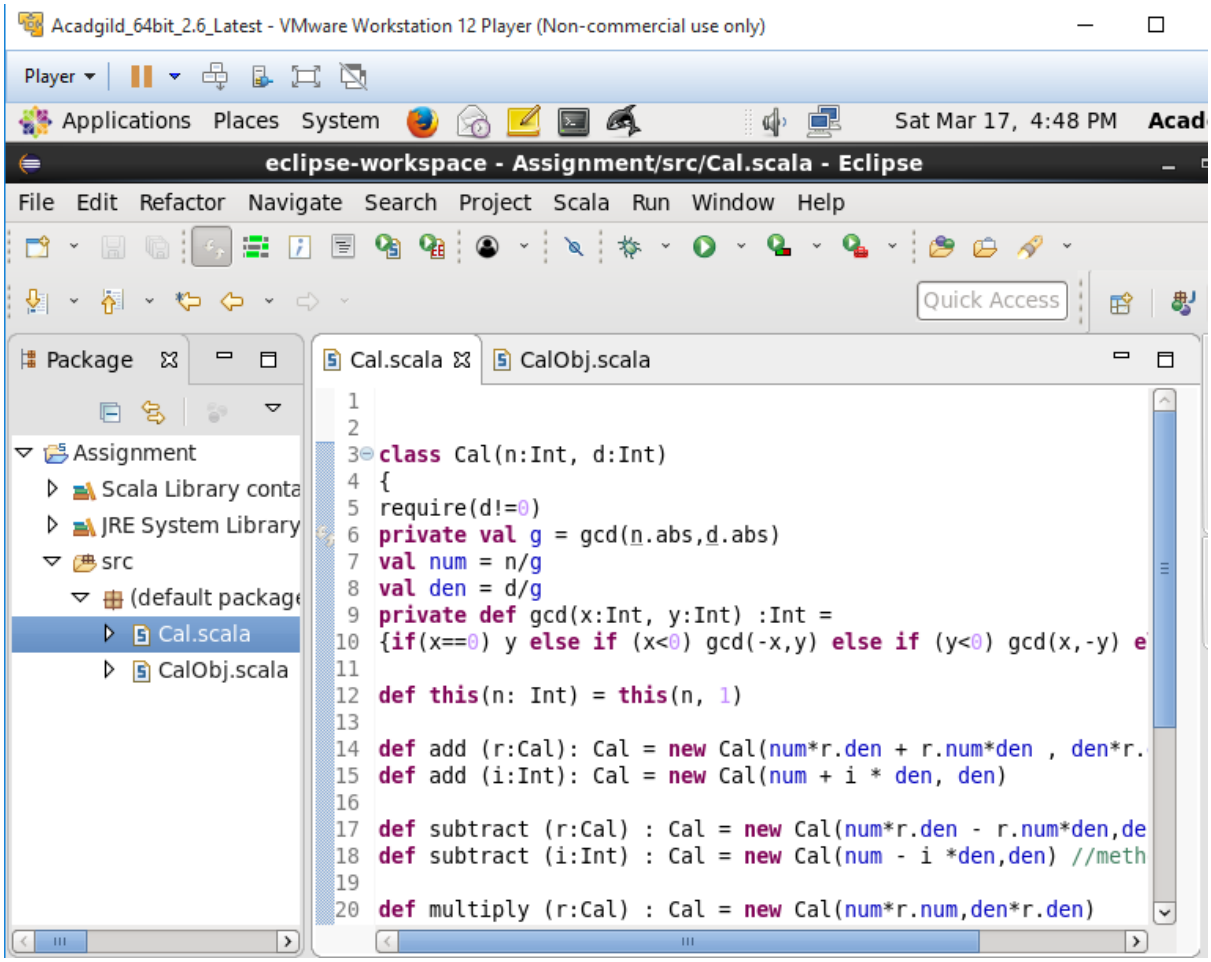
Add option to work with whole numbers which are also rational numbers i.e. (n/1)

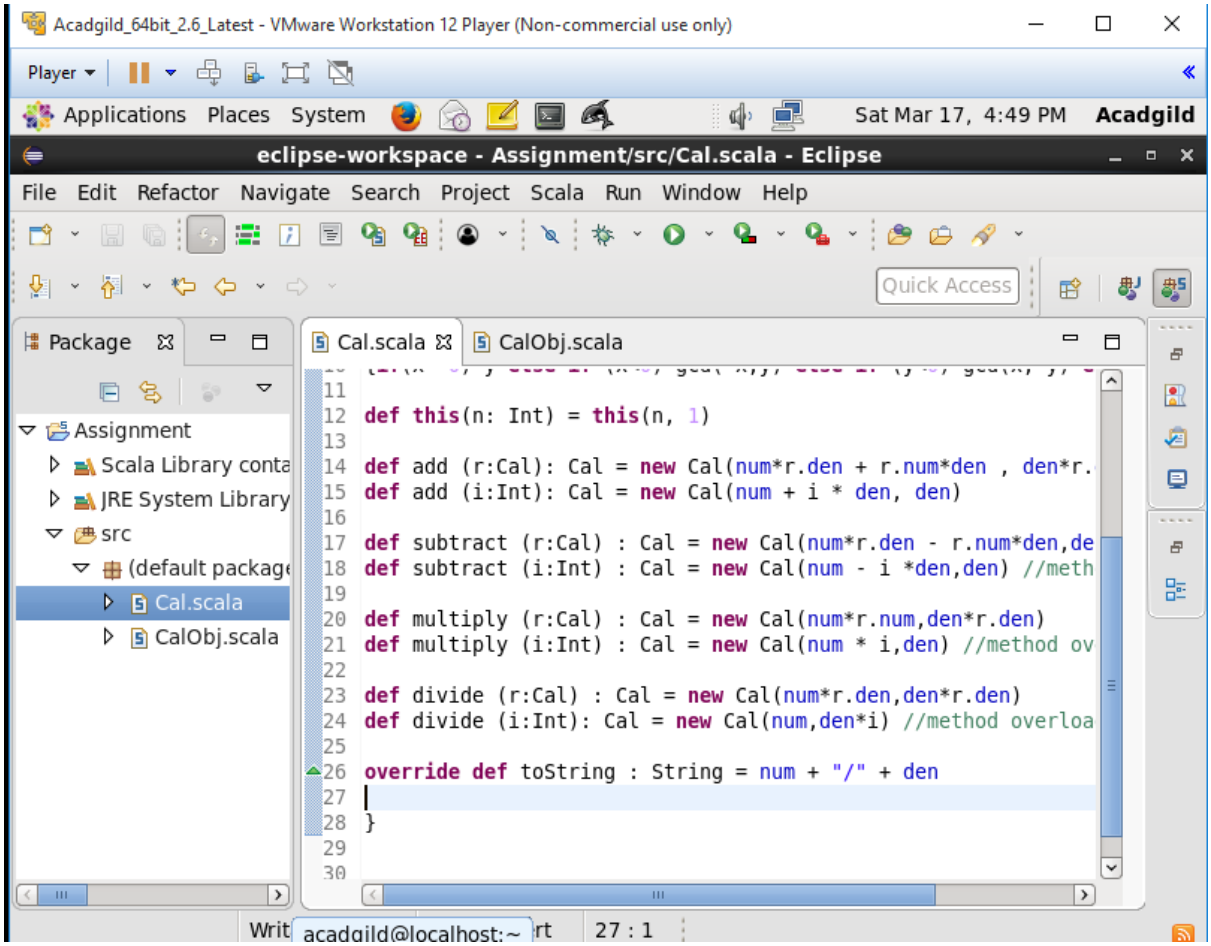
➢ achieve the above using auxiliary constructors

➢ enable method overloading to enable each function to work with numbers and rational

Create a Scala class “Calc”

```
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) //method overloading
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
def divide (r:Calc) : Calc = new Calc(num*r.den,den*r.den)
def divide (i:Int): Calc = new Calc(num,den*i) //method overloading
override def toString : String = num + "/" + den
}
```

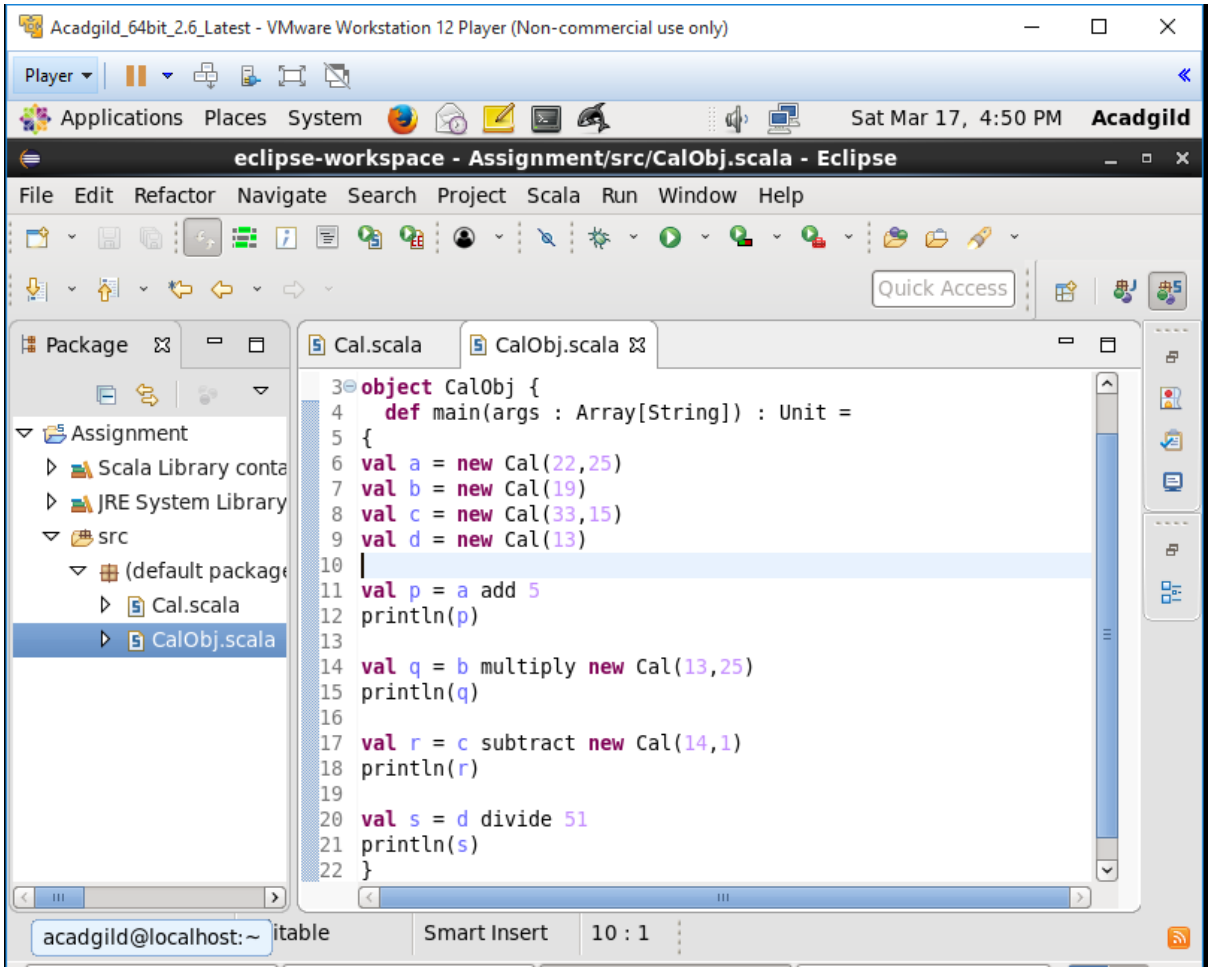

Acadgild_64bit_2.6_Latest - VMware Workstation 12 Player (Non-commercial use only)
Player
Applications Places System
Sat Mar 17, 4:48 PM
eclipse-workspace - Assignment/src/Cal.scala - Eclipse
File Edit Refactor Navigate Search Project Scala Run Window Help
Quick Access
Package
Assignment
Scala Library conta
JRE System Library
src
(default package
Cal.scala
CalObj.scala
Cal.scala
CalObj.scala
1
2
3 class Cal(n:Int, d:Int)
4 {
5 require(d!=0)
6 private val g = gcd(n.abs,d.abs)
7 val num = n/g
8 val den = d/g
9 private def gcd(x:Int, y:Int) :Int =
10 {if(x==0) y else if (x<0) gcd(-x,y) else if (y<0) gcd(x,-y) e
11
12 def this(n: Int) = this(n, 1)
13
14 def add (r:Cal): Cal = new Cal(num*r.den + r.num*den , den*r.
15 def add (i:Int): Cal = new Cal(num + i * den, den)
16
17 def subtract (r:Cal) : Cal = new Cal(num*r.den - r.num*den,de
18 def subtract (i:Int) : Cal = new Cal(num - i *den,den) //meth
19
20 def multiply (r:Cal) : Cal = new Cal(num*r.num,den*r.den)



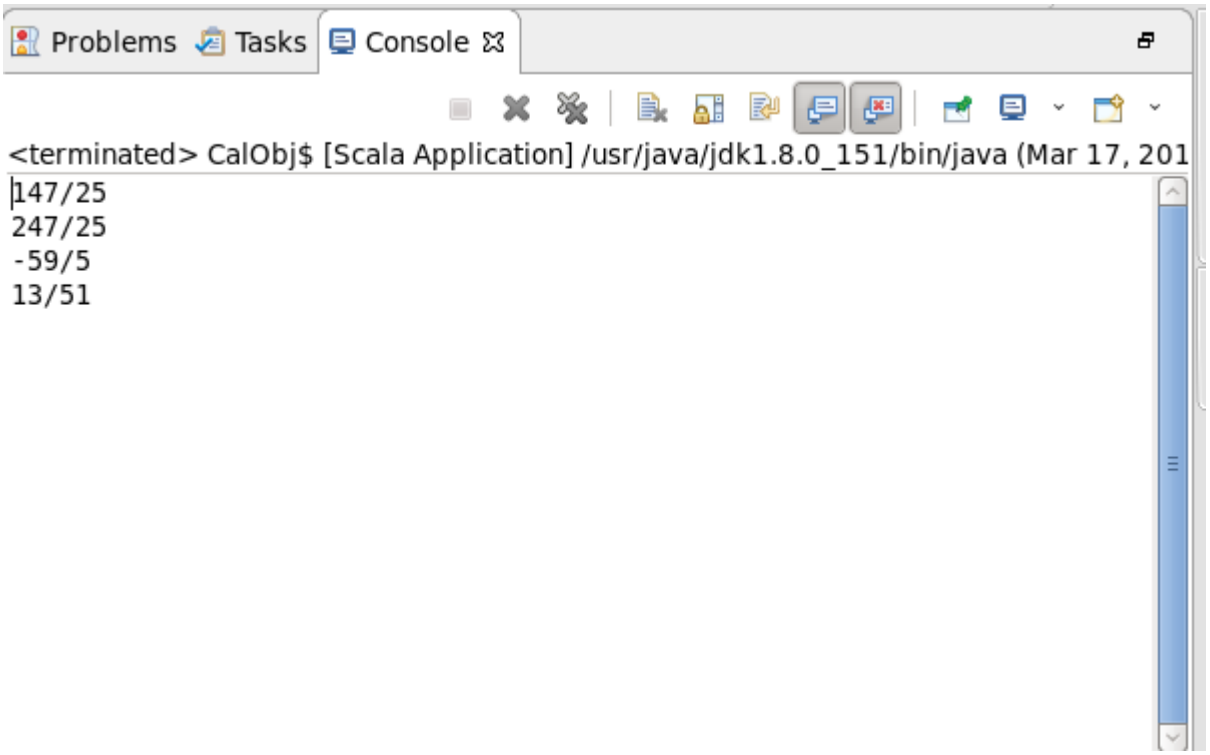
Example 1: Create Object "CalcObj" with different parameters

```
object CalObj

{

def main(args : Array[String]) : Unit =

{

val a = new Cal(22,25)

val b = new Cal(19)

val c = new Cal(33,15)

val d = new Cal(13)

val p = a add 5

println(p)

val q = b multiply new Cal(13,25)

println(q)
```

```
val r = c subtract new Cal(14,1)
println(r)
val s = d divide 51
println(s)
}
```



Output



Example 2 :Create Object "CalcObj" with different parameters

```
{
def main(args : Array[String]) : Unit =
{
val a = new Cal(4)
val b = new Cal(8)
val c = new Cal(9)
val d = new Cal(5)
val p = a add 2
println(p)
val q = b multiply new Cal(5)
println(q)
val r = c subtract new Cal(6)
println(r)
val s = d divide 7
println(s)
```

```
}
```

Cal.scala | CalObj.scala

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

val p = a add 2
println(p)

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

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

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

Output

Problems | Tasks | Console

```
<terminated> CalObj$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 17, 20
6/1
40/1
3/1
5/7
```