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.

Solution:

Calculator.Scala

```
package acadgild_14
import java.util.Scanner;
class Calculator(num1 : Int, num2 :Int)
{
  var numerator = 0
  var denominator = 0

  //Finding the GCD of the inputs provided.

  private def findGCD(a : Int, b : Int) : Int = if (b==0) a else
findGCD(b,a%b)

  if (num2 != 0)
  {
    val GetGCD = findGCD(num1.abs,num2.abs)
    numerator = num1 /GetGCD
    denominator = num2 /GetGCD
  }

  //Auxiliary Constructor
  def this(n : Int) = this(n,1)

  //Addition Operations on Rationals and Whole Numbers
  def +(that : Calculator) : Calculator =
    new Calculator(numerator * that.denominator + that.numerator *
denominator, denominator * that.denominator)

  def +(i:Int):Calculator =
    new Calculator(numerator + i * denominator,denominator)

  //Subtraction Operations on Rationals and Whole Numbers
  def -(that : Calculator) : Calculator =
    new Calculator(numerator * that.denominator - that.numerator *
denominator, denominator * that.denominator)

  def -(i:Int):Calculator =
    new Calculator(numerator - i * denominator,denominator)

  //Multiplication Operations on Rationals and Whole Numbers
  def *(that : Calculator) : Calculator =
    new Calculator(numerator * that.numerator ,denominator *
that.denominator)
```

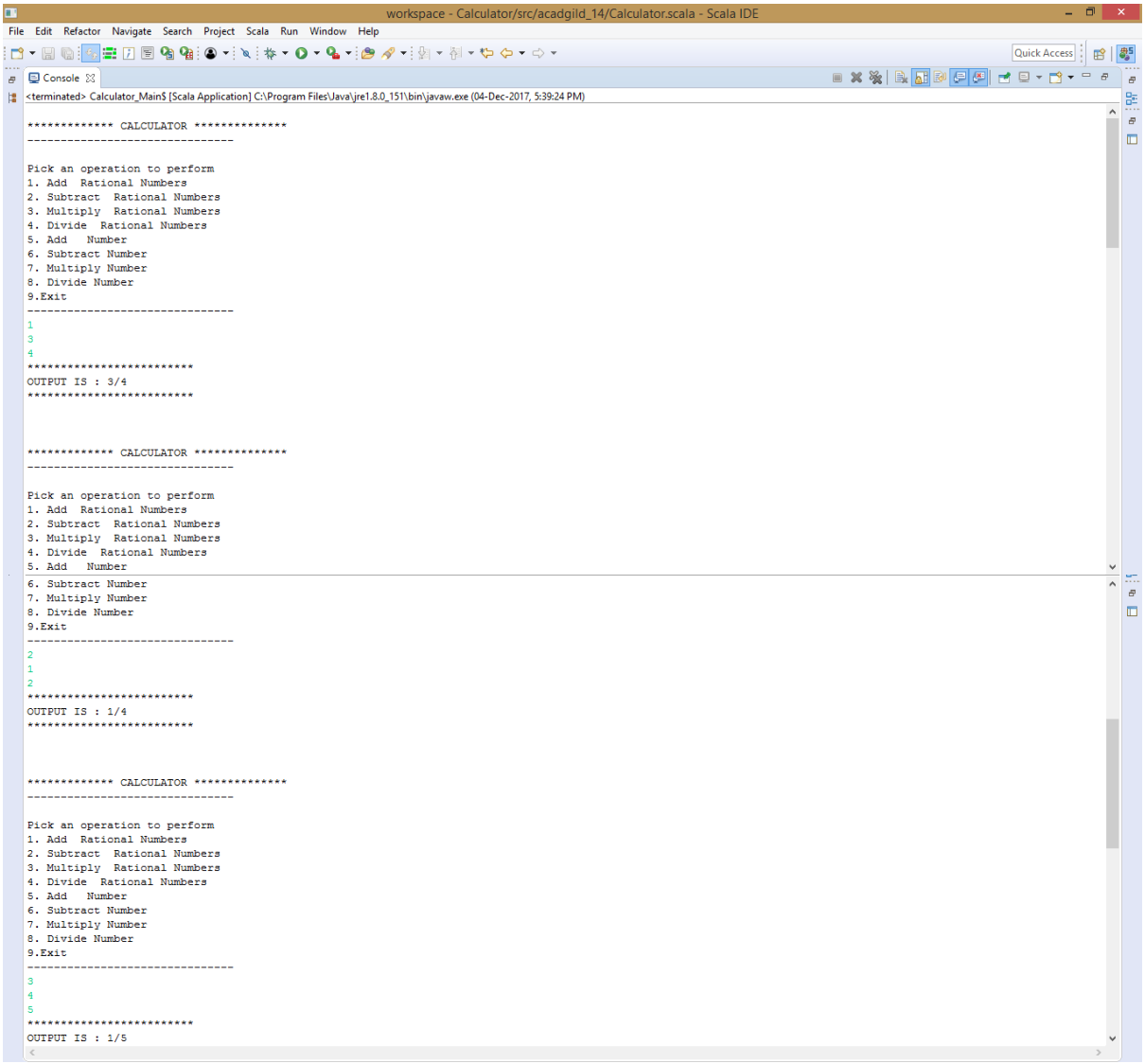
```
  def *(i:Int):Calculator =
    new Calculator(numerator * i, denominator)

  //Division Operations on Rationals and Whole Numbers
  def /(that : Calculator) : Calculator =
    new Calculator(numerator * that.denominator ,denominator *
that.numerator)

  def /(i:Int):Calculator =
    new Calculator(numerator , denominator *i)

  override def toString = numerator + "/" + denominator

}
```

Calculator Main.Scala

```
package acadgild_14

import java.util.Scanner;
object Calculator_Main
{
  //Display options available in the Calculator to the User

  private def OperationsList() =
  {
    println("\n************* CALCULATOR **************")
    println("-------------------------------\n")
    println("Pick an operation to perform")
    println("1. Add  Rational Numbers")
    println("2. Subtract  Rational Numbers")
    println("3. Multiply  Rational Numbers")
    println("4. Divide  Rational Numbers")
    println("5. Add   Number")
    println("6. Subtract Number")
    println("7. Multiply Number")
    println("8. Divide Number")
    println("9.Exit")
    println("-------------------------------")
  }

  //Get input and call appropriate overloaded method in class based on user
choice
  def Compute(rational : Calculator, number:Int):Calculator =
  {
    number match
    {
      case 1 =>
        val n = scala.io.StdIn.readInt()
        val d = scala.io.StdIn.readInt()
        rational.+(new Calculator(n,d))

        case 2 =>
        val n = scala.io.StdIn.readInt()
        val d = scala.io.StdIn.readInt()
        rational.-(new Calculator(n,d))

        case 3 =>
        val n = scala.io.StdIn.readInt()
        val d = scala.io.StdIn.readInt()
```

```
        rational.*(new Calculator(n,d))

        case 4 =>
        val n = scala.io.StdIn.readInt()
        val d = scala.io.StdIn.readInt()
        rational./(new Calculator(n,d))

        case 5 =>
        val n = scala.io.StdIn.readInt()
        rational.+(new Calculator(n))

        case 6 =>
        val n = scala.io.StdIn.readInt()
        rational.-(new Calculator(n))

        case 7 =>
        val n = scala.io.StdIn.readInt()
        rational.*(new Calculator(n))

        case 8 =>
        val n = scala.io.StdIn.readInt()
        rational./(new Calculator(n))

        case _ => rational

    }

  }

  //Main Method
  def main (args : Array[String]): Unit =
  {
    //Create instance of class
    var rationalNumber :Calculator =new Calculator(0)
    var input = 0

    do
    {
      OperationsList()
      input= scala.io.StdIn.readInt()
      rationalNumber = Compute(rationalNumber,input)

      println("*************************")
      //Result of calculation based on operations choosen
      println("OUTPUT IS : " + rationalNumber.toString)
      println("*************************\n\n")
    }while(input != 9)
  }
}
```

Output ScreenShot:



```
*************************



************* CALCULATOR **************
--------------------------------

Pick an operation to perform
1. Add  Rational Numbers
2. Subtract  Rational Numbers
3. Multiply  Rational Numbers
4. Divide  Rational Numbers
5. Add   Number
6. Subtract Number
7. Multiply Number
8. Divide Number
9.Exit
--------------------------------
5
5
*************************
OUTPUT IS : 26/5
*************************



************* CALCULATOR **************
--------------------------------

Pick an operation to perform
1. Add  Rational Numbers
2. Subtract  Rational Numbers
3. Multiply  Rational Numbers
4. Divide  Rational Numbers
5. Add   Number
6. Subtract Number
7. Multiply Number
8. Divide Number
9.Exit
--------------------------------
9
*************************
OUTPUT IS : 26/5
*************************
```