08/21

Big Data Engineering with Hadoop & Spark

AcadGild | Big Data Assignment Report |

Session 16: Assignment 16.1

This assignment is aimed at consolidating the concepts that was learnt during the Scala Basics session of the course.



***1***

AcadGild | Big Data Assignment Report |

Problem Statement

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)

**Note:**

− Achieve the above using auxiliary constructors

− Enable method overloading to enable each function to work with numbers and rational

**Solution:**

**Code Explanation:**

The class for the Calculator is made of two integral parts:

− The function to find the GCD of the number

o This is possible as the number is divided into two parts, the numerator and denominator.

*Ex: Rational: 3/7 and Number: 4 or can be written as 4/1*

o Finding the GCD of the numbers and dividing the numbers with their GCD brings them down to their least multiple and makes it easier to perform the arithmetic calculations.

*Ex: 12/4 will become 3/1*

− The Auxiliary constructor and the Overloaded Methods for rationales and intergers

o The Auxiliary constructor helps with constructor overloading and directs the compiler on which method to execute from the list of overloaded methods.

− The Overloaded methods are of two types:

o Methods for the rational operations (Add, Subtract, Multiply and Divide). These methods take an argument as that of the return of the class “***appRationalCalculator***” (i.e. a rational) and performs the necessary calculation



***2***

AcadGild | Big Data Assignment Report |

* 1. Methods for the number operations (Add, Subtract, Multiply and Divide). These methods take an integer value as argument and performs the necessary calculation

− Finally, there is an override for the **“*toString”*** method that returns the result of the numerator and denominator separated by a ***‘/’*** making it a rational.

− To implement the functionalities of the ***appRationalCalc*** class, I have created an Object called ***appRatlCalc***. This object contains the below methods:

* 1. A method that holds the different calculation options for the

Calculator that will be displayed to the user.

1. A method that processes the option chosen by the user and the input provided by the user and uses the match case to call the appropriate

overloaded method from the class above and provide the input from the user to that overloaded method.

1. Finally, we have the driver function that calls the above methods and displays the results of these methods to the user.

**Code for Scala Calculator:**

**class** appRationalCalc (inp1:Int,inp2:Int) {

*require*(inp2 !=0)

**var** *numerator*=0

**var** *denominator*=0

*/\*\*\*Find GCD of input provided.*

*This gives us numerator & denominator.\*\*\*/*

**private def** findGCD(a:Int,b:Int):Int=

**if**(b ==0) a

**else** findGCD(b,a % b)

**if** (inp2 !=0) {

**val** getGCD = findGCD (inp1.abs,inp2.abs)

*numerator* = inp1 / getGCD

*denominator* = inp2 / getGCD

}

*/\*\*\*Auxiliary construct\*\*\*/*

**def this**(n:Int) = **this**(n,1)

*/\*\*\*Addition operations on Rational and Whole Numbers\*\*\*/* **def** +(that: appRationalCalc): appRationalCalc =

**new** appRationalCalc(*numerator*\* that.*denominator*+ that.*numerator*\**denominator*,*denominator* \* that.*denominator*)

*/\*\*\*Method overloading for "+" in order to perform a addition\*\*\*/* **def** +(i:Int): appRationalCalc =

**new** appRationalCalc(*numerator*+ i \**denominator*,*denominator*)

*/\*\*\*Subtraction operations on Rational and Whole Numbers\*\*\*/* **def** -(that: appRationalCalc): appRationalCalc =

**new** appRationalCalc(*numerator*\* that.*denominator*- that.*numerator*\**denominator*,*denominator* \* that.*denominator*)

*/\*\*\*Method overloading for "-" in order to perform a subtraction\*\*\*/*



***3***

AcadGild | Big Data Assignment Report |

**def** -(i:Int): appRationalCalc =

**new** appRationalCalc(*numerator*- i \**denominator*,*denominator*)

*/\*\*\*Multiplication operations on Rational and Whole Numbers\*\*\*/* **def** \*(that: appRationalCalc): appRationalCalc =

**new** appRationalCalc(*numerator*\* that.*numerator*,*denominator*\* that.*denominator*)

*/\*\*\*Method overloading for "\*" in order to perform a multiplication\*\*\*/* **def** \*(i:Int): appRationalCalc =

**new** appRationalCalc(*numerator*\* i,*denominator*)

*/\*\*\*Division operations on Rational and Whole Numbers\*\*\*/* **def** /(that: appRationalCalc): appRationalCalc =

**new** appRationalCalc(*numerator*\* that.*denominator*,*denominator*\* that.*numerator*)

*/\*\*\*Method overloading for "/" in order to perform a division\*\*\*/* **def** /(i:Int): appRationalCalc =

**new** appRationalCalc(*numerator*,*denominator*\* i)

*/\*\*\*To display the output in the format "n/d"*

*we have to override as we are overloading functions using auxiliary constructor\*\*\*/* **override def** toString=*numerator*+"/"+*denominator*

}

**object** appRatlCalc {

*/\*\*\*Display list of operations that can be performed by the user\*\*\*/* **private def** OperationsList() = {

*println*("Rational Calculator")

*println*("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

*println*("Pick an operation to perform")

*println*("1. Addition")

*println*("2. Subtraction")

*println*("3. Multiplication")

*println*("4. Division")

*println*("5. Add a Rational number with an Integer")

*println*("6. Subtract a Rational number with an Integer")

*println*("7. Multiply a Rational number with an Integer")

*println*("8. Divide a Rational number with an Integer")

*println*("9. Exit")

}

*/\*\*\*Get input and call appropriate overloaded method in class based on user's choice\*\*\*/*

**def** Compute(rational: appRationalCalc,number:Int): appRationalCalc = {

number **match** {

**case** 1 =>

**val** n = scala.io.StdIn.readInt()

**val** d = scala.io.StdIn.readInt()

rational.+(**new** appRationalCalc(n, d))

**case** 2 =>

**val** n = scala.io.StdIn.readInt()

**val** d = scala.io.StdIn.readInt()

rational.-(**new** appRationalCalc(n, d))

**case** 3 =>

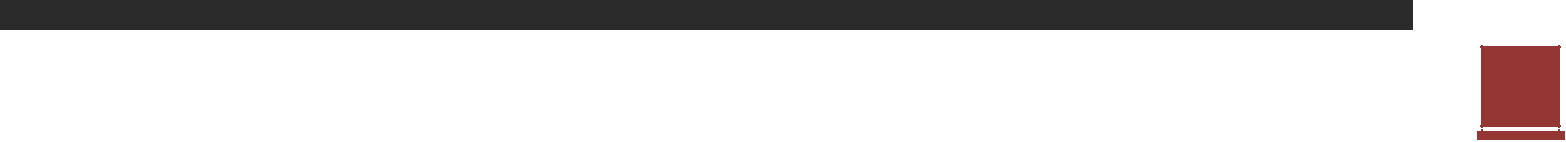
**val** n = scala.io.StdIn.readInt()

**val** d = scala.io.StdIn.readInt()

rational.\*(**new** appRationalCalc(n, d))

**case** 4 =>

**val** n = scala.io.StdIn.readInt()



***4***

AcadGild | Big Data Assignment Report |

**val** d = scala.io.StdIn.readInt()

rational./(**new** appRationalCalc(n, d))

**case** 5 =>

**val** n = scala.io.StdIn.readInt()

rational.+(**new** appRationalCalc(n))

**case** 6 =>

**val** n = scala.io.StdIn.readInt()

rational.-(**new** appRationalCalc(n))

**case** 7 =>

**val** n = scala.io.StdIn.readInt()

rational.\*(**new** appRationalCalc(n))

**case** 8 =>

**val** n = scala.io.StdIn.readInt()

rational./(**new** appRationalCalc(n))

**case** \_ => rational

}

}

*/\*\*\*Driver Method\*\*\*/*

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

*/\*\*\*Create instance of class\*\*\*/*

**var** ratlNumb: appRationalCalc = **new** appRationalCalc(0)

**var** input =0

**do** {

*OperationsList*()

input = scala.io.StdIn.readInt()

ratlNumb = *Compute*(ratlNumb, input)

*println*("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

*println*("OUTPUT: "+ ratlNumb.toString) */\*\*\*Result of the Calculation\*\*\*/ println*("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

}

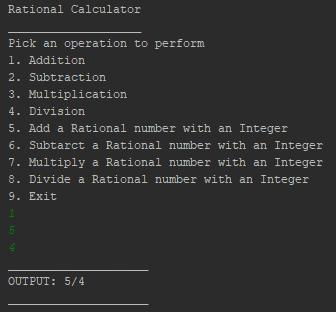
**while** (input !=9)

}

}

**Output:**

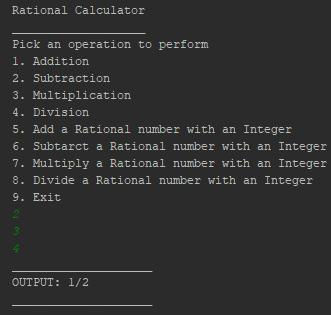
**1. Addition**



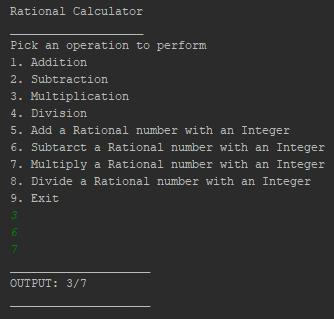
***5***

AcadGild | Big Data Assignment Report |

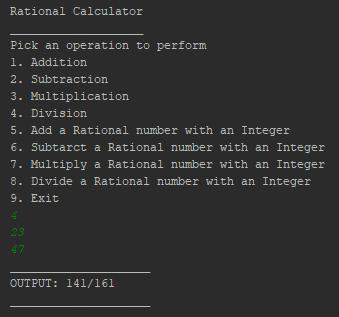
**2. Subtraction**



**3. Multiplication**



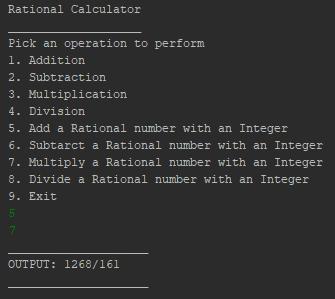
**4. Division**



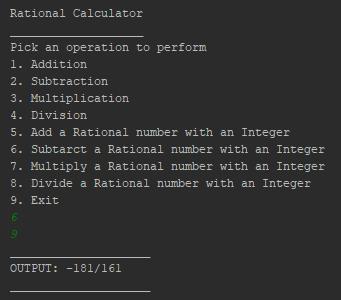
***6***

AcadGild | Big Data Assignment Report |

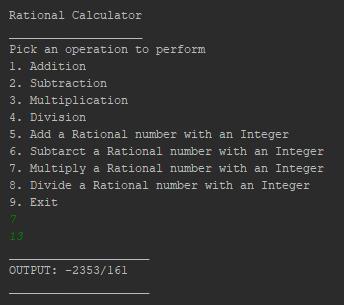
**5. Add a Rational number with an Integer**



**6. Subtract a Rational number with an Integer**



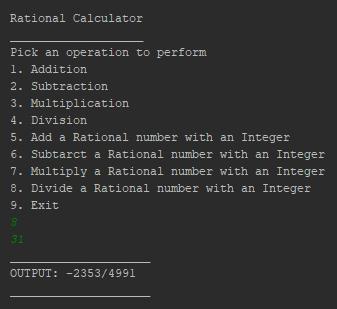
**7. Multiply a Rational number with an Integer**



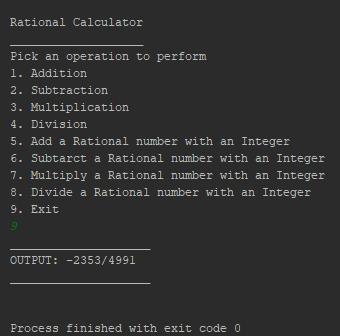
***7***

AcadGild | Big Data Assignment Report |

**8. Divide a Rational number with an Integer**



1. **Exit** (Also, returns final answer)



***Note:***

*Scala code files for each application has been provided separately along with this assignment report.*



***8***