**Assignment 14.1**

**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:**

In mathematics, a rational number is any number that can be expressed as the quotient or fraction p/q of two integers, a numerator p and a non-zero denominator q

Case 1

if there are two rational number are a/b and c/d then

Addition of rational number is : a/b+c/d=(ad+bc)/bd

two rational are 13/4 and 4/5

13(a) 4(b) 4(c) 5(d)

13\*5+4\*5=65+20=85

4\*5=20

result is :85/25=17/5

Subtraction of two rational numbers : a/b-c/d=(ad-bc)/bd

two rational numbers 13/4 and 4/5

13(a) 4(b) 4(c) 5(d)

13\*5 - 4\*4=65 -16=49

4\*5=20

result is :85/25=49/20

Multiplication of rational number is : a/b\*c/d=(ac)/bd

two rational numbers 13/4 and 4/5

13(a) 4(b) 4(c) 5(d)

13\*4=52

4\*5=20

result is : 52/20=13/5

Division of rational number is : a/b/c/d=(ad)/(bc) here c!=0

two rational numbers 13/4 and 4/5

13(a) 4(b) 4(c) 5(d)

13\*5=65

4\*4=16

result is : 52/20=65/16

Case 2

If there are one rational number a/b and other number is c then

Addition of number is : a/b+c/1=(a+bc)/b

13(a)

5b)

4(c)

1(d)

13\*+5\*4=13+20=33

5=5

result is :33/5

Subtraction of number is : a/b-c/1=(a-bc)/b

13(a)

5b)

4(c)

1(d)

13\*-5\*4=13-20=-7

5=5

result is :-7/5

Multiplication of number is : a/b\*c/1=(ac)/b

13(a)

5b)

4(c)

1(d)

13\*4=52

5=5

result is :52/5

Division of number is : a/b/c/1=(a)/bc

13(a)

5b)

4(c)

1(d)

13=13

5\*4=20

result is :13/20

**package** com.calculator

//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.

**class** RationalCalculator(numerator: Int, denominator: Int) {

require(denominator != 0)

**private** **val** g = gcd(numerator.abs, denominator.abs)

**val** numer = numerator / g

**val** denom = denominator / g

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

//addition with Rational Number

**def** +(that: RationalCalculator): RationalCalculator =

**new** RationalCalculator(

numer \* that.denom + that.numer \* denom,

denom \* that.denom)

//addition with number

**def** +(number: Int): RationalCalculator =

**new** RationalCalculator(numer + number \* denom, denom)

//subtraction with Rational Number

**def** -(that: RationalCalculator): RationalCalculator =

**new** RationalCalculator(

numer \* that.denom - that.numer \* denom,

denom \* that.denom)

//subtraction with Number

**def** -(number: Int): RationalCalculator =

**new** RationalCalculator(numer - number \* denom, denom)

//Multiplication with Rational Number

**def** \*(that: RationalCalculator): RationalCalculator =

**new** RationalCalculator(numer \* that.numer, denom \* that.denom)

//Multiplication with Number

**def** \*(number: Int): RationalCalculator =

**new** RationalCalculator(numer \* number, denom)

//Division with Rational Number

**def** /(that: RationalCalculator): RationalCalculator =

**new** RationalCalculator(numer \* that.denom, denom \* that.numer)

//Division with Number

**def** /(number: Int): RationalCalculator =

**new** RationalCalculator(numer, denom \* number)

**override** **def** toString = numer + "/" + denom

**private** **def** gcd(firstNumber: Int, secondNumber: Int): Int =

**if** (secondNumber == 0) firstNumber **else** gcd(secondNumber, firstNumber % secondNumber)

}

**object** RationalCalculatorMain {

**def** Options() = {

println("1. Add two rational numbers")

println("2. Subtract two rational numbers")

println("3. Multiply two rational numbers")

println("4. Divide two rational numbers")

println("5. Add a rational number with number")

println("6. Subtract a rational number with number")

println("7. Multiply a rational number with number")

println("8. Divide a rational number with number")

println("9. Exit")

}

**def** Compute(rational: RationalCalculator, choice: Int): RationalCalculator = {

choice **match** {

**case** 1 => //Add two rational numbers

println("Enter first rational number : ")

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

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

println("Enter second rational number : ")

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

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Addition ")

firstRational.+(**new** RationalCalculator(secondRationalNumerator, secondRationalDenominator))

**case** 2 => //Subtract two rational numbers

println("Enter first rational number : ")

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

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

println("Enter second rational number : ")

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

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Subtraction ")

firstRational.-(**new** RationalCalculator(secondRationalNumerator, secondRationalDenominator))

**case** 3 => //Multiply two rational numbers

println("Enter first rational number : ")

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

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

println("Enter second rational number : ")

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

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Multiplication ")

firstRational.\*(**new** RationalCalculator(secondRationalNumerator, secondRationalDenominator))

**case** 4 => //Divide two rational numbers

println("Enter first rational number : ")

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

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

println("Enter second rational number : ")

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

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Division ")

firstRational./(**new** RationalCalculator(secondRationalNumerator, secondRationalDenominator))

**case** 5 => //Add a rational number with number

// val rationalNum = scala.io.StdIn.readInt()

// rational.+(new RationalCalculator(rationalNum))

println("Enter rational number is : ")

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

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

println("Enter number : ")

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Addition ")

firstRational.+(number)

**case** 6 => //Subtract a rational number with number

println("Enter rational number is : ")

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

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

println("Enter number : ")

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Subtraction ")

firstRational.-(number)

**case** 7 => // Multiply a rational number with number

println("Enter rational number is : ")

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

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

println("Enter number : ")

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Multiplication ")

firstRational.\*(number)

**case** 8 => //Divide a rational number with number

println("Enter rational number is : ")

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

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

println("Enter number : ")

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

**val** firstRational = **new** RationalCalculator(firstRationalNumerator, firstRationalDenominator);

println("After Division ")

firstRational./(number)

**case** \_ =>

rational

}

}

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

**var** rationalNumber: RationalCalculator = **new** RationalCalculator(0)

**var** input = 0

**do** {

Options()

input = scala.io.StdIn.readInt()

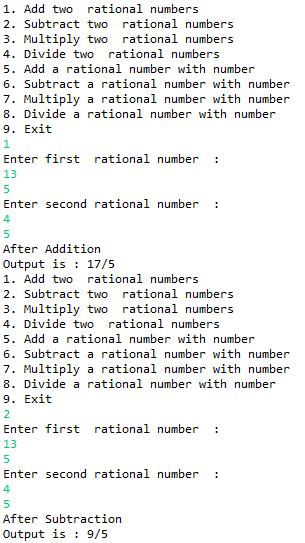
rationalNumber = Compute(rationalNumber, input)

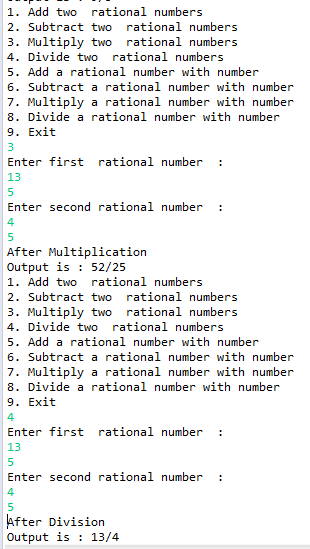
println("Output is : " + rationalNumber.toString)

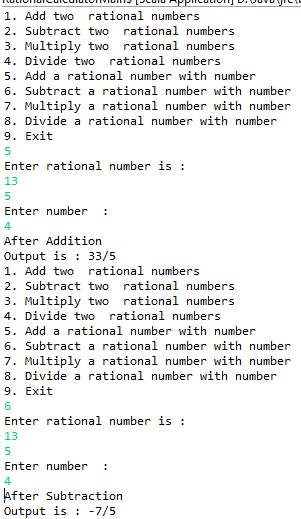
} **while** (input != 7)

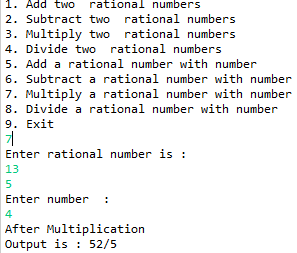
}

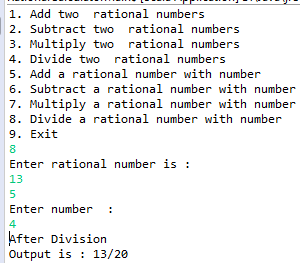
}

****

****

****

****

****