Assignment 16.1
Scala Basics 3

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.**

Here we have created Fraction class and companion object. We have handled both Rationals and whole numbers and also used auxiliary function (def this).

We have also used GCD method to convert given input to the smallest fraction or rationals.

Here following are methods for corresponding operations :
+ → Addition
- → Subtraction
* → Multiplication
/ → Division

Assignment 16.1
Scala Basics 3

SOURCE CODE

```
object Fraction {
  // companion object to Fraction
  def main(args: Array[String]): Unit = {

    println("Addition, Subtraction, Mutiplicatons and Division Operations on
Integers")
    // integer addition
    Fraction(5) + Fraction(6)

    // integer subtraction
    Fraction(11) - Fraction(4)

    // integer multiplication
    Fraction(7) * Fraction(3)

    // integer division
    Fraction(12) / Fraction(4)

    println("Addition, Subtraction, Mutiplicatons and Division Operations on
Rationals")

    // fraction addition
    Fraction(2,4) + Fraction(2,6)

    // fraction subtraction
    Fraction(2,4) - Fraction(2,6)

    // fraction multiplication
    Fraction(2,5) * Fraction(9,6)

    // fraction division
    Fraction(2,5) / Fraction(9,6)


  }

  // helper functions to instantiate Fraction class for both Fraction and integers
  def apply(n1: Int, d1: Int)=
  { new Fraction(n1,d1) }

  def apply(x: Int)={ new Fraction(x) } }


class Fraction private(val n1: Int, val d1: Int)  // primary constructor, private
{
  private def this(x: Int) = // auxiliary constructor, private
  {
    this(x, 1)
  }

  def +(r: Fraction): Unit = {
    // cross multiply the fractions and then add which becomes the result numerator
    // multiply denominators which becomes result denominator
    // get the greatest common divisor of resulting numerator and denominator
    // and divide resultant numerator and denominator with it
    val n = n1 * r.d1 + d1 * r.n1
    val d = d1 * r.d1
    val gcd = GCD(n, d)
    println(frp(n1, d1) + " + " + frp(r.n1, r.d1) + " = " + frp(n / gcd, d / gcd))
  }


  def -(r: Fraction): Unit = {
```

Assignment 16.1
Scala Basics 3

```
    val n = n1 * r.d1 - d1 * r.n1
    val d = d1 * r.d1
    val gcd = GCD(n, d)
    println(frp(n1, d1) + " - " + frp(r.n1, r.d1) + " = " + frp(n / gcd, d / gcd))
  }

  def *(r: Fraction): Unit = {
    // multiply both numerators which becomes the resultant numerator
    // multiply both denominators which becomes the resultant denominator
    // get the GCD of resulting numerator and denominator
    // and divide resultant numerator and denominator with it
    val n = n1 * r.n1
    val d = d1 * r.d1
    val gcd = GCD(n, d)
    println(frp(n1, d1) + " x " + frp(r.n1, r.d1) + " = " + frp(n / gcd, d / gcd))
  }

  def /(r: Fraction): Unit = {
    // logic is same as multiple but invert the second Fraction,
    // basically cross multiply the fractions
    val n = n1 * r.d1
    val d = d1 * r.n1
    val gcd = GCD(n, d)
    println(frp(n1, d1) + " / " + frp(r.n1, r.d1) + " = " + frp(n / gcd, d / gcd))
  }

  // helper function to find greatest common divisor

  def GCD(x: Int, y: Int): Int = {
    if (y == 0) x else GCD(y, x % y)
  }
  // helper method to print fraction or as integer if the denominator is 1 private

  def frp(n: Int, d: Int): String = if (d == 1) n.toString else n + "/" + d

}
```

Output :

Addition, Subtraction, Mutiplicatons and Division Operations on Integers

5 + 6 = 11

11 - 4 = 7

7 x 3 = 21

12 / 4 = 3

Addition, Subtraction, Mutiplicatons and Division Operations on Rationals

2/4 + 2/6 = 5/6

2/4 - 2/6 = 1/6

2/5 x 9/6 = 3/5

2/5 / 9/6 = 4/15

Process finished with exit code 0