Scala Assignment 14

For this assignment, I've used Intellij Idea.

Create a calculator to work with rational numbers.

1) It should provide capability to add, subtract, divide and multiply rational numbers
2) Create a method to compute GCD (this will come in handy during operations on rational)
3) 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.

Steps followed:

A rational number is any number that can be expressed as the quotient or fraction p/q of two integers.

1) Created a class which takes 2variables (n-> numerator, d-> demoninator) as below.The denominator shouldn't be 0 so specified here.
2) The program should also support whole numbers, so specifying auxilary constructor to call the 2 variable constructor, as below.

```
//creating a class
class Rational(n: Int, d: Int) {
  require(d != 0)

//Defining constructor
// auxiliary constructor to support whole number a rational number
  def this(n: Int) = this(n, 1)

  // auxiliary constructor to support no arg
  def this() = this(1, 1) }
```

3) Created a method to compute GCD the method used is as below:

```
//Method to compute gcd:
  private def gcd(a: Int, b: Int): Int =
```

```
if (b == 0) a else gcd(b, a % b)
```

4) When a pair of rational number is passed as argument, gcd of argument n & d. Then set the numer value as n/gcd(n,d) and denom as d/gcd(n,d)

```
//Creating a numerator and denomenator for a rational number
private val g = gcd(n.abs, d.abs)
 val numer = n / g
 val denom = d / g
```

5) Created a method "+" by passing a rational number as argument, and add rational number as below.

```
//Add for Rational numbers
 def + (that: Rational): Rational =  new Rational(
    numer * that.denom + that.numer * denom,
    denom * that.denom
  )


 //Overloaded Add for whole numbers
 def + (i: Int): Rational =  new Rational(numer + i * denom, denom)
```

6) Created a method method "-" by passing a rational number as argument, and subtract rational number as below.

```
//Subract for Rational numbers
 def - (that: Rational): Rational =  new Rational(
    numer * that.denom - that.numer * denom,
    denom * that.denom
  )

 //Overloaded Subract for whole numbers
 def - (i: Int): Rational =   new Rational(numer - i * denom, denom)
```

7) Created a method "*" by passing a rational number as argument, and multiply the rational number as below..

```
//Multiply for Rational numbers
 def * (that: Rational): Rational =
  new Rational(numer - i * denom, denom)

 //Overloaded Multiply for whole numbers
 def * (i: Int): Rational =
  new Rational(numer * i, denom)
```

8) Created a method "/" by passing a rational number as argument, and divide the rational number as below.

```
//Div for Rational numbers
 def / (that: Rational): Rational =
  new Rational(numer * that.denom, denom * that.numer)

 //Overloaded Div for whole numbers
 def / (i: Int): Rational =
  new Rational(numer, denom * i)
```

9) Finally writtern a toString method, to print the spring representation of the object as below

```
//overridden To String method, to print meaningful output
 override def toString = numer + "/" + denom
```

10) Tested the calculator methods by passing the input.

```
object assignment14 extends App {
  //rational number input
  val r1 = new Rational(3, 5)                          // r1 = 3/5
  val r2 = new Rational(2, 7)                          // r2 = 2/7
  val r3 = new Rational(5, 9)                          // r3 = 5/9

  println(r1 +" + "+ r2 +" = "+  (r1 + r2));           // 3/5+2/7 = 31/35
  println(r1 +" - "+ r2 +" = "+  (r1 - r2));           // 3/5-2/7 = 11/35
  println(r1 +" * "+ r2 +" = "+  (r1 * r2));           // 3/5*2/7 = 6/35
  println(r1 +" / "+ r2 +" = "+  (r1 / r2));           // 3/5/2/7 = 21/10
  println(r1+" == "+r2+" : " +(r1 == r2))              // false

  val x = new Rational(7, 3)                           // x = 1/3
  val y = new Rational(5, 7)                           // y = 5/7
  val z = new Rational(3, 2)                           // z = 3/2

  println(x+" - "+y+" - "+z+" = " +(x - y - z))        // 7/3-5/7-3/2 = 5/42

  //passing whole number as rational number
  val a = new Rational(5)                              // a = 5/1
  val b = new Rational                                 // b = 1/1
  println(a+" + "+b+" = "+(a+b));                      // 5/1+1/1 = 6/1
}
```

Output printed in the console is:

assignment14

```
"C:\Program Files\Java\jdk1.8.0_121\bin\java" ...
3/5 + 2/7 = 31/35
3/5 - 2/7 = 11/35
3/5 * 2/7 = 6/35
3/5 / 2/7 = 21/10
3/5 == 2/7 : false
7/3 - 5/7 - 3/2 = 5/42
5/1 + 1/1 = 6/1
```

Screenshot of IntelliJ source code:

Rational | +(that: Rational) | <anonymous>

```
/* Calculator program 14.1 */
class Rational(n: Int, d: Int) {
  require(d != 0)

  // auxiliary constructor to support whole number a rational number
  def this(n: Int) = this(n, 1)

  // auxiliary constructor to support no arg
  def this() = this(1, 1)

  private val g = gcd(n.abs, d.abs)
  val numer = n / g
  val denom = d / g

  //Add for Rational numbers
  def + (that: Rational): Rational =
    new Rational(
      numer * that.denom + that.numer * denom,
      denom * that.denom
    )

  //Overloaded Add for whole numbers
  def + (i: Int): Rational =
    new Rational(numer + i * denom, denom)
```

ssignment14

```
"C:\Program Files\Java\jdk1.8.0_121\bin\java" ...
3/5 + 2/7 = 31/35
3/5 - 2/7 = 11/35
3/5 * 2/7 = 6/35
3/5 / 2/7 = 21/10
3/5 == 2/7 : false
7/3 - 5/7 - 3/2 = 5/42
5/1 + 1/1 = 6/1
```

Source code:

```
/*Calculator program 14.1 */
class Rational(n: Int, d: Int) {
  require(d != 0)

  //Defining constructor
  def this(n: Int) = this(n, 1) // auxiliary constructor to support single arg
  def this() = this(1, 1) // auxiliary constructor to support no arg

  private val g = gcd(n.abs, d.abs)
  val numer = n / g
  val denom = d / g

  //Add for Rational numbers
  def + (that: Rational): Rational =
    new Rational(
      numer * that.denom + that.numer * denom,
      denom * that.denom
    )

  //Overloaded Add for whole numbers
  def + (i: Int): Rational =
    new Rational(numer + i * denom, denom)

  //Subract for Rational numbers
  def - (that: Rational): Rational =
    new Rational(
      numer * that.denom - that.numer * denom,
      denom * that.denom
    )

  //Overloaded Subract for whole numbers
  def - (i: Int): Rational =
    new Rational(numer - i * denom, denom)

  //Multiply for Rational numbers
  def * (that: Rational): Rational =
    new Rational(numer * that.numer, denom * that.denom)

  //Overloaded Multiply for whole numbers
  def * (i: Int): Rational =
    new Rational(numer * i, denom)

  //Div for Rational numbers
  def / (that: Rational): Rational =
    new Rational(numer * that.denom, denom * that.numer)

  //Overloaded Div for whole numbers
  def / (i: Int): Rational =
    new Rational(numer, denom * i)

  //GCD
  private def gcd(a: Int, b: Int): Int =
    if (b == 0) a else gcd(b, a % b)

  //overridden To String method, to print meaningful output
  override def toString = numer + "/" + denom
}

object assignment14 extends App {

 //rational number input
```

```
  val r1 = new Rational(3, 5)                          // r1 = 3/5
  val r2 = new Rational(2, 7)                          // r2 = 2/7
  val r3 = new Rational(5, 9)                          // r3 = 5/9

  println(r1 +" + "+ r2 +" = "+   (r1 + r2));          // 3/5+2/7 = 31/35
  println(r1 +" - "+ r2 +" = "+   (r1 - r2));          // 3/5-2/7 = 11/35
  println(r1 +" * "+ r2 +" = "+   (r1 * r2));          // 3/5*2/7 = 6/35
  println(r1 +" / "+ r2 +" = "+   (r1 / r2));          // 3/5/2/7 = 21/10
  println(r1+" == "+r2+" : " +(r1 == r2))              // false

  val x = new Rational(7, 3)                           // x = 1/3
  val y = new Rational(5, 7)                           // y = 5/7
  val z = new Rational(3, 2)                           // z = 3/2

  println(x+" - "+y+" - "+z+" = " +(x - y - z))        // 7/3-5/7-3/2 = 5/42

  //passing whole number as rational number
  val a = new Rational(5)                              // a = 5/1
  val b = new Rational                                 // b = 1/1
  println(a+" + "+b+" = "+(a+b));                      // 5/1+1/1 = 6/1
}
```