Assignment 16- Scala 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.

Code:

```
class Rational(n: Int, d: Int) {

require(d != 0)
private val g = gcd(n.abs, d.abs)


val numer = n / g
val denom = d / g
def this(n: Int) = this(n, 1)
def + (that: Rational): Rational =
new Rational(
numer * that.denom + that.numer * denom,
denom * that.denom
)
def + (i: Int): Rational =
new Rational(numer + i * denom, denom)
def - (that: Rational): Rational =
new Rational(
numer * that.denom - that.numer * denom,
denom * that.denom
```

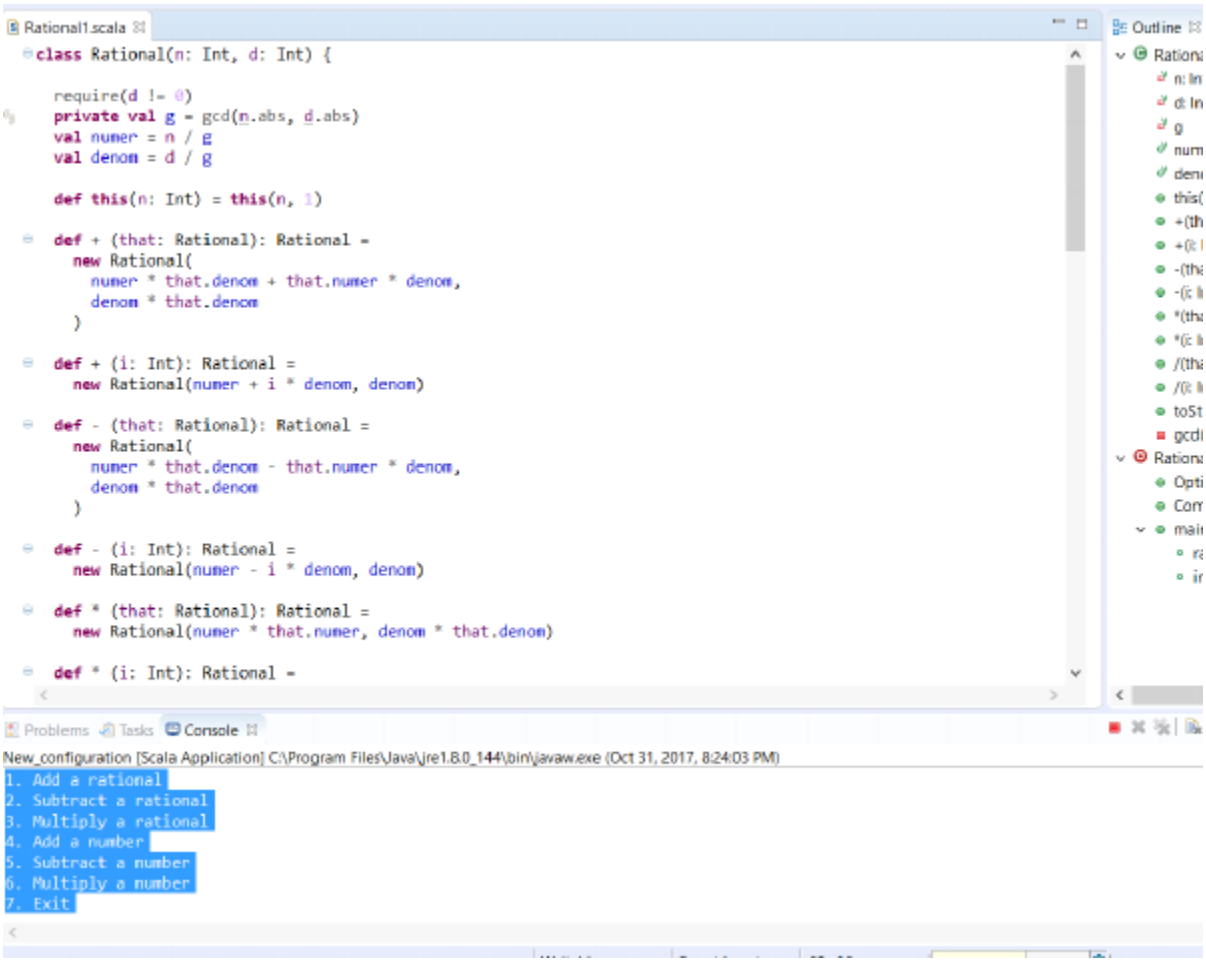
```
)
def - (i: Int): Rational =
new Rational(numer - i * denom, denom)
def * (that: Rational): Rational =
new Rational(numer * that.numer, denom * that.denom)
def * (i: Int): Rational =
new Rational(numer * i, denom)
def / (that: Rational): Rational =
new Rational(numer * that.denom, denom * that.numer)
def / (i: Int): Rational =
new Rational(numer, denom * i)
override def toString = numer +"/"+ denom


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

object RationalMain {
def Options() = {
        println("1. Add a rational")
        println("2. Subtract a rational")
        println("3. Multiply a rational")
        println("4. Add a number")
        println("5. Subtract a number")
        println("6. Multiply a number")
        println("7. Exit")
}


def Compute(rational: Rational, input: Int): Rational = {


input match {
        case 1 =>
                val p = scala.io.StdIn.readInt()
                val q = scala.io.StdIn.readInt()
                rational.+(new Rational(p, q))
```

```
        case 2 =>
                val p = scala.io.StdIn.readInt()
                val q = scala.io.StdIn.readInt()
                rational.-(new Rational(p, q))
        case 3 =>
                val p = scala.io.StdIn.readInt()
                val q = scala.io.StdIn.readInt()
                rational.*(new Rational(p, q))
        case 4 =>
                val p = scala.io.StdIn.readInt()
                rational.+(new Rational(p))
        case 5 =>
                val p = scala.io.StdIn.readInt()
                rational.-(new Rational(p))
        case 6 =>

                val p = scala.io.StdIn.readInt()
                rational.*(new Rational(p))
                case _ =>
        rational
        }
                }

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

        var rationalNumber: Rational = new Rational(0)

var input = 1

do {

        Options()
        input = scala.io.StdIn.readInt()
        rationalNumber = Compute(rationalNumber, input)
        println("Output is : " + rationalNumber.toString)
        } while (input != 7)

}
}
```

Assignment 16- Scala 3

Outcome:

1. Add a rational
2. Subtract a rational
3. Multiply a rational
4. Add a number
5. Subtract a number
6. Multiply a number
7. Exit



Assignment 16- Scala 3