ASSIGNMENT 13.1

Problem Statement:

Create a Scala application to find the GCD of two numbers.

Solution:

Greatest Common Divisor (GCD) of two numbers:

GCD of two numbers is the largest positive number which can divide both numbers without any remainder.

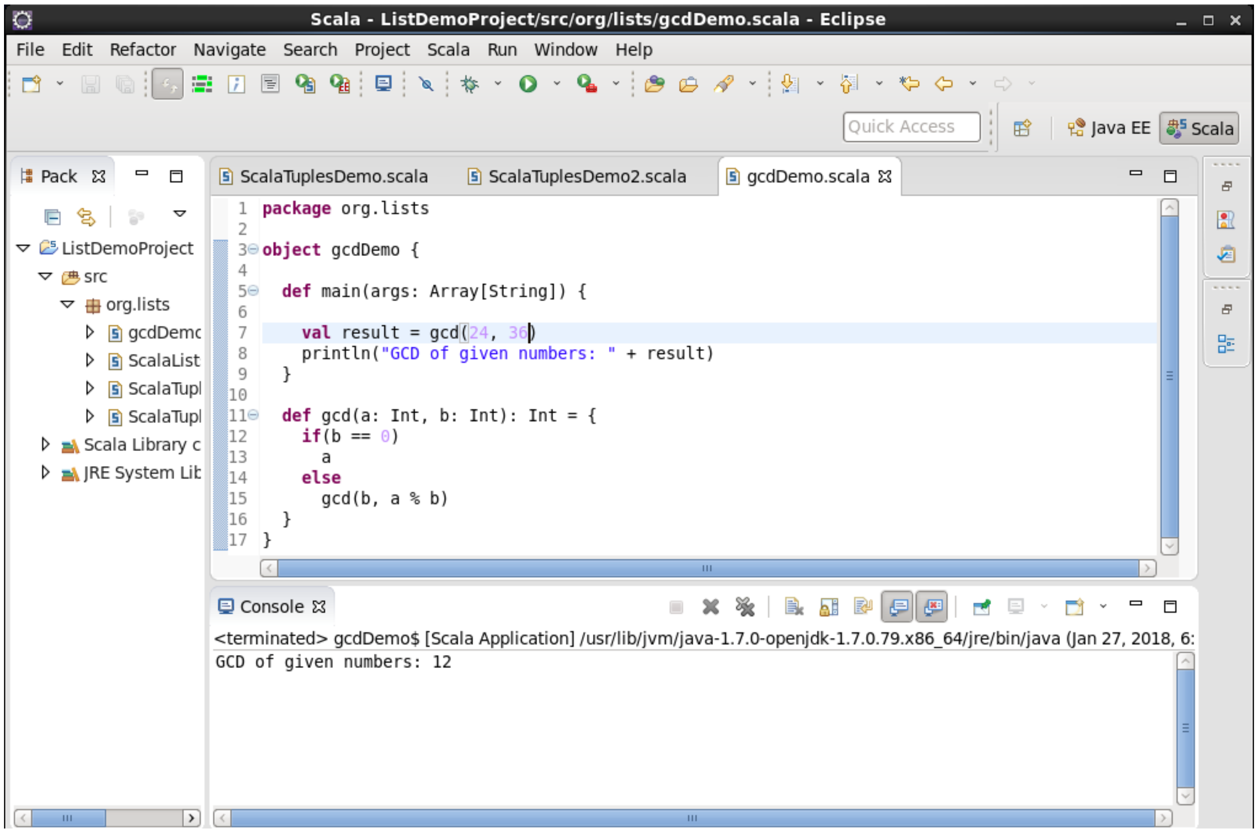
For example, if we take two numbers say 24 and 36, these two numbers are divisible by 1, 2, 3, 4, 6, and 12. Thereby 12 is the largest number that can divide both 24 and 36 leaving no remainder.

Let's do it programmatically using Scala. Scala provides a functionality called **'tail recursion'** which can be used to solve this of problem. A tail recursion is a function where the last action will call the same function itself again until the result reaches a specific value. Let's see the implementation and then figure out how it works with the help of an example.

```
object gcdDemo {
    def main(args: Array[String]) {
        val result = gcd (24, 36)                         // gcd (num1, num2) function invocation
        println("GCD of given numbers: " + result)  // print result of gcd computation
    }
    def gcd(a: Int, b: Int): Int = {
    if ( b == 0 )                  // base condition: if second number is zero,
        a                          // return first number as result
    else
        gcd ( b, a % b )           // otherwise, take second number in place of first number,
     }                             // calculate num1 % num2 and call gcd () recursively
}
```

Output:

```
GCD of given numbers: 12
```



```
package org.lists

object gcdDemo {

  def main(args: Array[String]) {

    val result = gcd(24, 36)
    println("GCD of given numbers: " + result)
  }

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

```
<terminated> gcdDemo$ [Scala Application] /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.79.x86_64/jre/bin/java (Jan 27, 2018, 6:
GCD of given numbers: 12
```

Let’s try to understand the above piece of code by taking the same numbers as in output:

a = 24, b = 36

1st iteration - gcd (24, 36):

36 != 0

gcd (36, 24 % 36) => gcd (36, 24)

2nd iteration - gcd (36, 24):

24 != 0

gcd (24, 36 % 24) => gcd (24, 12)

3rd iteration - gcd (24, 12):

12 != 0

gcd (12, 24 % 12) => gcd (12, 0)

4th iteration - gcd (12, 0):

0 == 0

Return 12 as GCD of 24 and 36.