

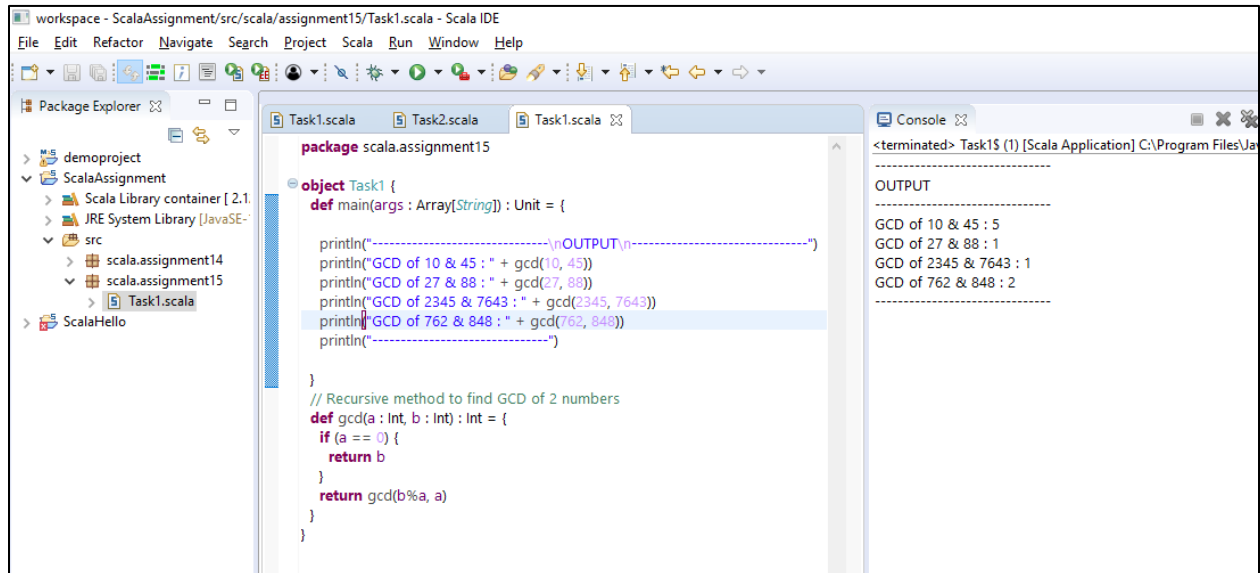
Session 15:
SCALA BASICS 2
Assignment 1

TASK 1

Create a Scala application to find the GCD of two numbers

EXECUTION:

Created a new scala object "Task1.scala". I have added a recursive method to calculate GCD of 2 numbers. Below is the screenshot of the program and the output of GCD for different set of numbers.



The screenshot shows the Scala IDE interface. The Package Explorer on the left shows a project named 'demoproject' with a sub-project 'ScalaAssignment'. Inside 'ScalaAssignment', there are two sub-projects: 'scala.assignment14' and 'scala.assignment15'. Under 'scala.assignment15', there is a file named 'Task1.scala'. The main editor displays the code for 'Task1.scala'. The code defines a package 'scala.assignment15' and an object 'Task1'. The 'main' method calls 'gcd' for four pairs of numbers: (10, 45), (27, 88), (2345, 7643), and (762, 848). A recursive method 'gcd' is defined to calculate the GCD of two numbers. The Console on the right shows the output of the program, which is the GCD for each pair of numbers.

```
package scala.assignment15

object Task1 {
  def main(args : Array[String]) : Unit = {

    println("-----\nOUTPUT\n-----")
    println("GCD of 10 & 45 : " + gcd(10, 45))
    println("GCD of 27 & 88 : " + gcd(27, 88))
    println("GCD of 2345 & 7643 : " + gcd(2345, 7643))
    println("GCD of 762 & 848 : " + gcd(762, 848))
    println("-----")

  }

  // Recursive method to find GCD of 2 numbers
  def gcd(a : Int, b : Int) : Int = {
    if (a == 0) {
      return b
    }
    return gcd(b%a, a)
  }
}
```

Console Output:

```
<terminated> Task1$ (1) [Scala Application] C:\Program Files\Ja
OUTPUT
GCD of 10 & 45 : 5
GCD of 27 & 88 : 1
GCD of 2345 & 7643 : 1
GCD of 762 & 848 : 2
```

TASK 2

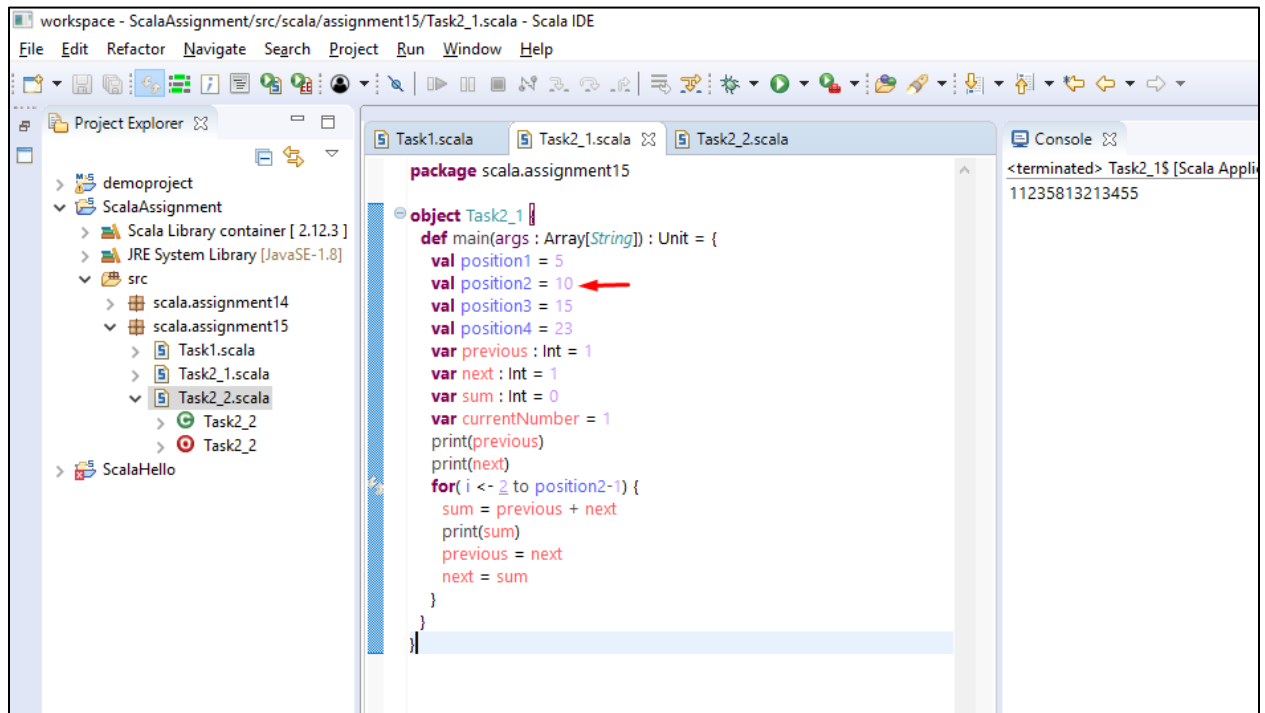
Fibonacci series (starting from 1) written in order without any spaces in between, thus producing a sequence of digits.

Write a Scala application to find the Nth digit in the sequence.

- Write the function using standard for loop
- Write the function using recursion

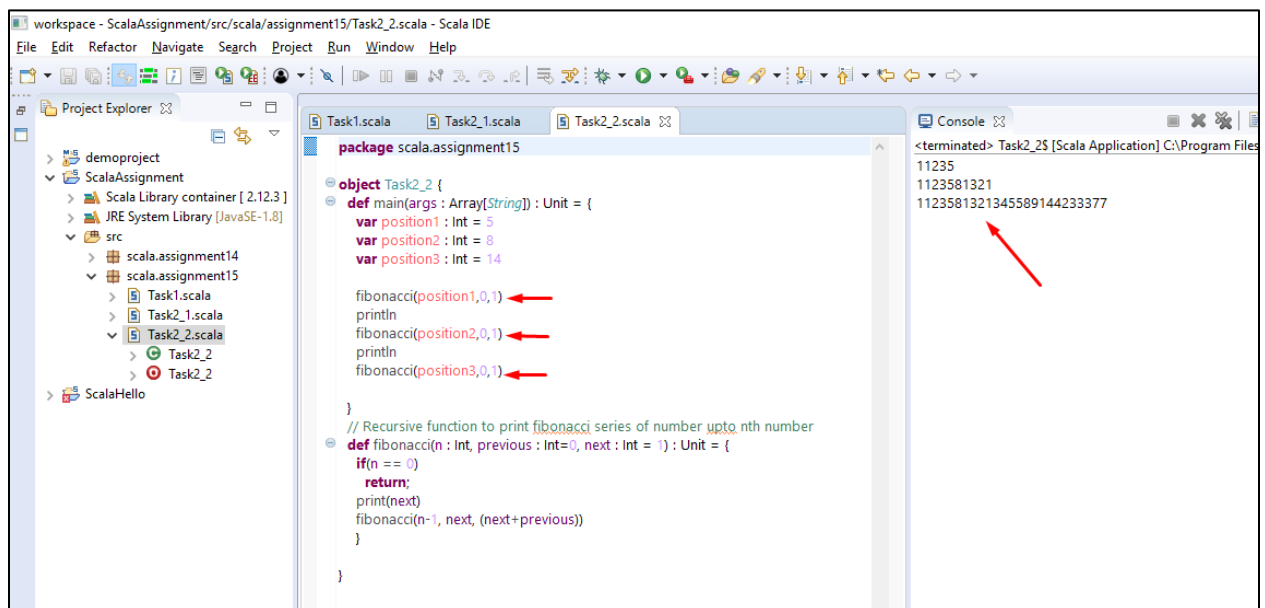
EXECUTION:

Write the function using standard for loop



Write the function using recursion

Created a scala program that generates Fibonacci series of numbers upto Nth digit using recursive function



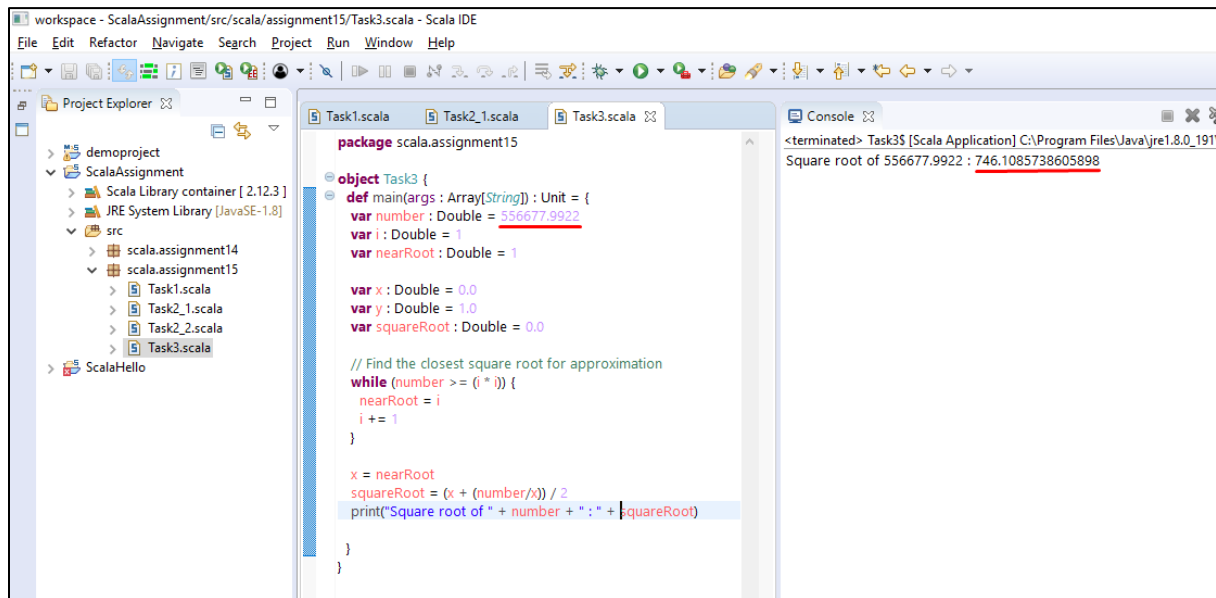
TASK 3

Find square root of number using Babylonian method.

1. Start with an arbitrary positive start value x (the closer to the root, the better).
2. Initialize $y = 1$.
3. Do following until desired approximation is achieved.
 - a) Get the next approximation for root using average of x and y
 - b) Set $y = n/x$

EXECUTION

Created a scala program to calculate the square root of a number using Babylonian method.



```
workspace - ScalaAssignment/src/scala/assignment15/Task3.scala - Scala IDE
File Edit Refactor Navigate Search Project Run Window Help

Project Explorer
  demoproject
  ScalaAssignment
    Scala Library container [ 2.12.3 ]
    JRE System Library [JavaSE-1.8]
    src
      scala.assignment14
      scala.assignment15
        Task1.scala
        Task2_1.scala
        Task2_2.scala
        Task3.scala
      ScalaHello

Task1.scala Task2_1.scala Task3.scala
package scala.assignment15

object Task3 {
  def main(args: Array[String]): Unit = {
    var number: Double = 556677.9922
    var i: Double = 1
    var nearRoot: Double = 1

    var x: Double = 0.0
    var y: Double = 1.0
    var squareRoot: Double = 0.0

    // Find the closest square root for approximation
    while (number >= (i * i)) {
      nearRoot = i
      i += 1
    }

    x = nearRoot
    squareRoot = (x + (number/x)) / 2
    print("Square root of " + number + " : " + squareRoot)
  }
}

Console
<terminated> Task3$ [Scala Application] C:\Program Files\Java\jre1.8.0_191\
Square root of 556677.9922 : 746.1085738605898
```

