

Session 15 - Scala Session - II

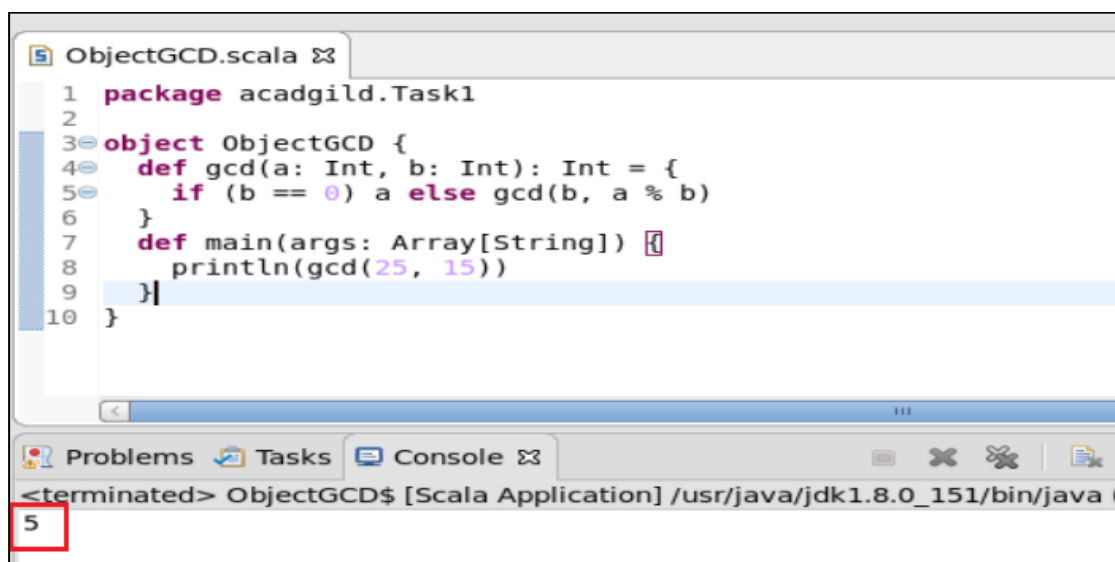
Assignment 1

I created a Scala application(project) in Eclipse and created 3 separate packages for each of the tasks.

Task 1:

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

```
object ObjectGCD {  
  def gcd(a: Int, b: Int): Int = {  
    if (b == 0) a else gcd(b, a % b)  
  }  
  def main(args: Array[String]) {  
    println(gcd(25, 15))  
  }  
}
```



```
ObjectGCD.scala 5  
1 package acadgild.Task1  
2  
3 object ObjectGCD {  
4   def gcd(a: Int, b: Int): Int = {  
5     if (b == 0) a else gcd(b, a % b)  
6   }  
7   def main(args: Array[String]) {  
8     println(gcd(25, 15))  
9   }  
10 }
```

Problems Tasks Console 5

<terminated> ObjectGCD\$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java

The output clearly shows the GCD of 25 and 5 that is 5.

Task 2:

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

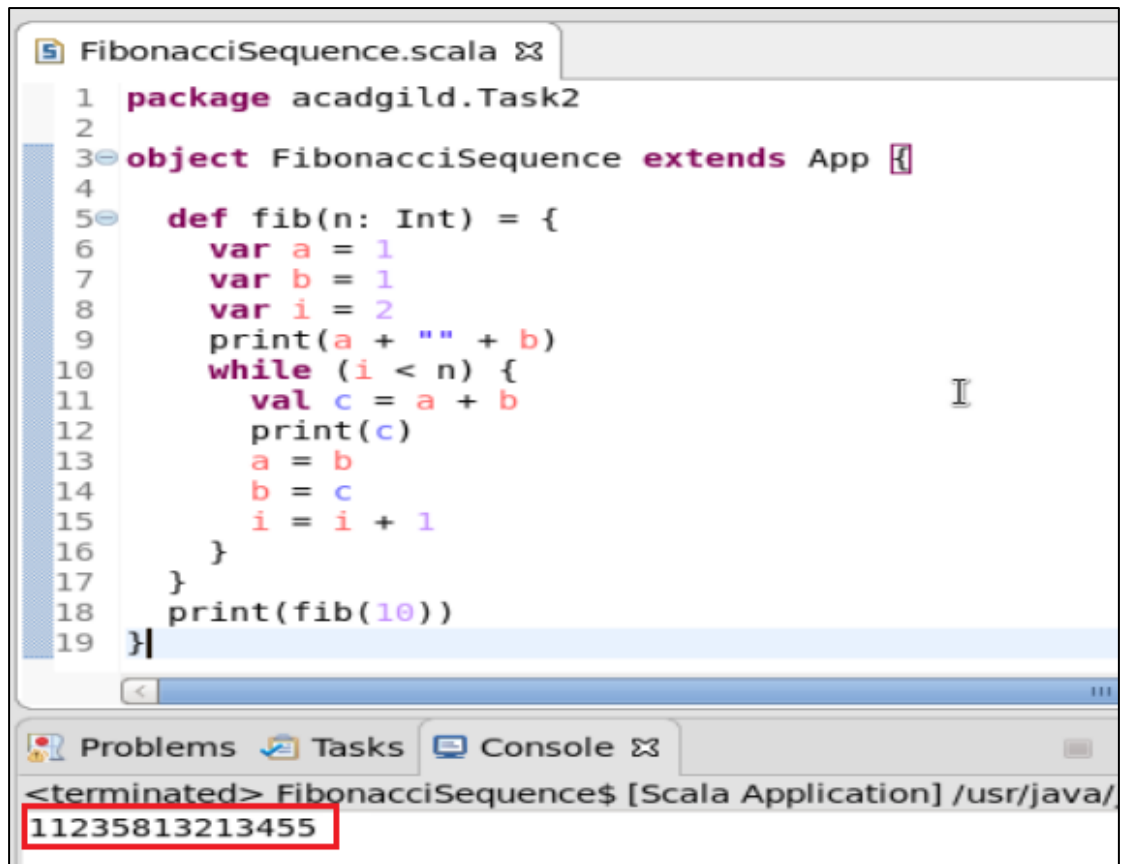
Write a Scala application to find the Nth digit in the sequence.

➤ Write the function using standard for loop

➤ Write the function using recursion

Printing Fibonacci series in orde without spaces

```
object FibonacciSequence extends App {  
  def fib(n: Int) = {  
    var a = 1  
    var b = 1  
    var i = 2  
    print(a + "" + b)  
    while (i < n) {  
      val c = a + b  
      print(c)  
      a = b  
      b = c  
      i = i + 1  
    }  
  }  
  print(fib(10))  
}
```



```
FibonacciSequence.scala
1 package acadgild.Task2
2
3 object FibonacciSequence extends App {
4
5   def fib(n: Int) = {
6     var a = 1
7     var b = 1
8     var i = 2
9     print(a + " " + b)
10    while (i < n) {
11      val c = a + b
12      print(c)
13      a = b
14      b = c
15      i = i + 1
16    }
17  }
18  print(fib(10))
19 }
```

Problems Tasks Console

<terminated> FibonacciSequence\$ [Scala Application] /usr/java/11235813213455

The function takes in an integer n, thus printing the first n numbers from the fibonacci sequence. Here, n =10 and the first 10 values I.e,

1,1,2,3,5,8,13,21,34,55

are printed without any spaces.

Write the function using standard for loop

```
object FibonacciLoop extends App{
  def fib( n : Int ) : Int = {
    var a = 0
    var b = 1
    var i = 0

    while( i < n ) {
      val c = a + b
      a = b
      b = c
      i = i + 1
    }
  }
}
```

```

    }
    return a
  }
  println(fib(10))
}

```

```

1 package acadgild.Task2
2
3 object FibonacciLoop extends App{
4   def fib( n : Int ) : Int = {
5     var a = 0
6     var b = 1
7     var i = 0
8
9     while( i < n ) {
10      val c = a + b
11      a = b
12      b = c
13      i = i + 1
14    }
15    return a
16  }
17  println(fib(10))
18
19 }

```

Problems Tasks Console

<terminated> FibonacciLoop\$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java

55

The function takes in an integer n, thus printing the nth number from the fibonacci sequence. Here, n =10 and the 10th value(starting from 1) i.e,

1,1,2,3,5,8,13,21,34,55

55 is printed.

Write the function using recursion

```

object FibonacciRecursion extends App{
  def fib( n : Int) : Int = n match {
    case 0 | 1 => n
    case _ => fib( n-1 ) + fib( n-2 )
  }
  println(fib(10))
}

```

```
FibonacciRecursion.scala  ✖
1 package acadgild.Task2
2
3 object FibonacciRecursion extends App{
4   def fib( n : Int) : Int = n match {
5     case 0 | 1 => n
6     case _ => fib( n-1 ) + fib( n-2 )
7   }
8   println(fib(10))
9 }

Problems Tasks Console ✖
<terminated> FibonacciRecursion$ [Scala Application] /usr
55
```

The function takes in an integer n , thus printing the n th number from the fibonacci sequence. Here, $n = 10$ and the 10th value(starting from 1) i.e,

1,1,2,3,5,8,13,21,34,55

55 is printed.

The function 'fib' is recursively called by itself multiple times until the value of n is either 0 or 1.

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 $y = n/x$

```
object Babylonian extends App{
  def squareRoot(n:Double):Double = {
```

```

var x = n
var y = 1.000000

var e = 0.000001
while(x-y > e)
{
    x = (x+y)/2
    y=n/x
}
return x
}
println(squareRoot(25))
}

```

The screenshot shows an IDE window titled 'Babyloninan.scala'. The code defines a package 'acadgild.Task3' and an object 'Babyloninan' that extends 'App'. Inside the object, a function 'squareRoot(n:Double):Double' is defined, which implements the Babylonian method using a while loop. The function calculates the square root of 25 and prints the result. The console output at the bottom shows the program has terminated and the result is 5.0000000000053722, which is highlighted with a red box.

```

1 package acadgild.Task3
2
3 object Babyloninan extends App {
4     def squareRoot(n:Double):Double = {
5         var x = n
6         var y = 1.000000
7
8         var e = 0.000001
9         while(x-y > e)
10        {
11            x = (x+y)/2
12            y=n/x
13        }
14        return x
15    }
16    println(squareRoot(25))
17 }

```

<terminated> Babyloninan\$ [Scala Application] /usr/java/jdk1.8.
5.0000000000053722

The function takes in an Double value n whose square root has to be found out. Here n is 25 and the square root of 25 is found out to be 5.0000000000053722 using Babylonian Method to find out square root.