

## Session 15: SCALA BASICS 2 : Assignment 1

### Task 1

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

The screenshot shows a virtual machine environment with the following components:

- Host OS:** Windows 10 taskbar at the bottom with search, task view, and application icons.
- Virtual Machine:** Oracle VM VirtualBox running Hadoop 2.6.1.1.1.
- Guest OS:** Linux environment with user 'acadgild'.
- Terminal Window:** Shows the execution of Scala commands:

```
[acadgild@localhost ~]$ scalac gcd.scala
[acadgild@localhost ~]$ scala gcd.scala
5
[acadgild@localhost ~]$
```
- Editor Window:** gedit editing 'gcd.scala' with the following code:

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

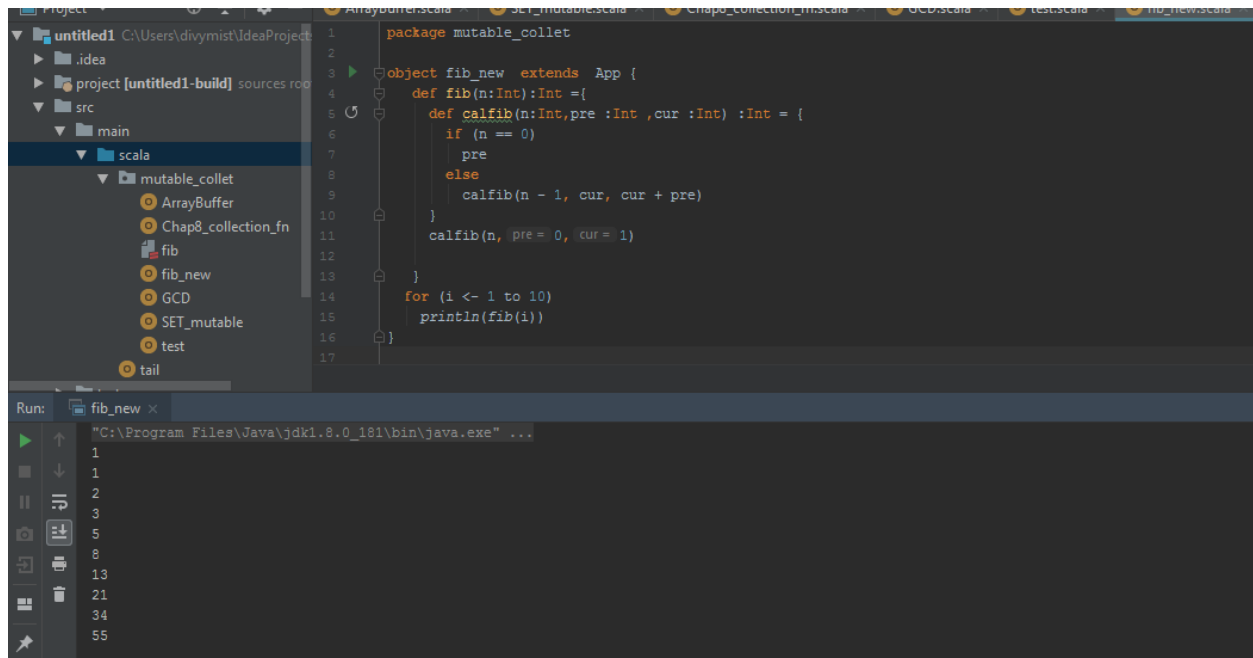
  def main(args: Array[String]) {
    println(gcd(25, 15))
  }
}
```

## 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



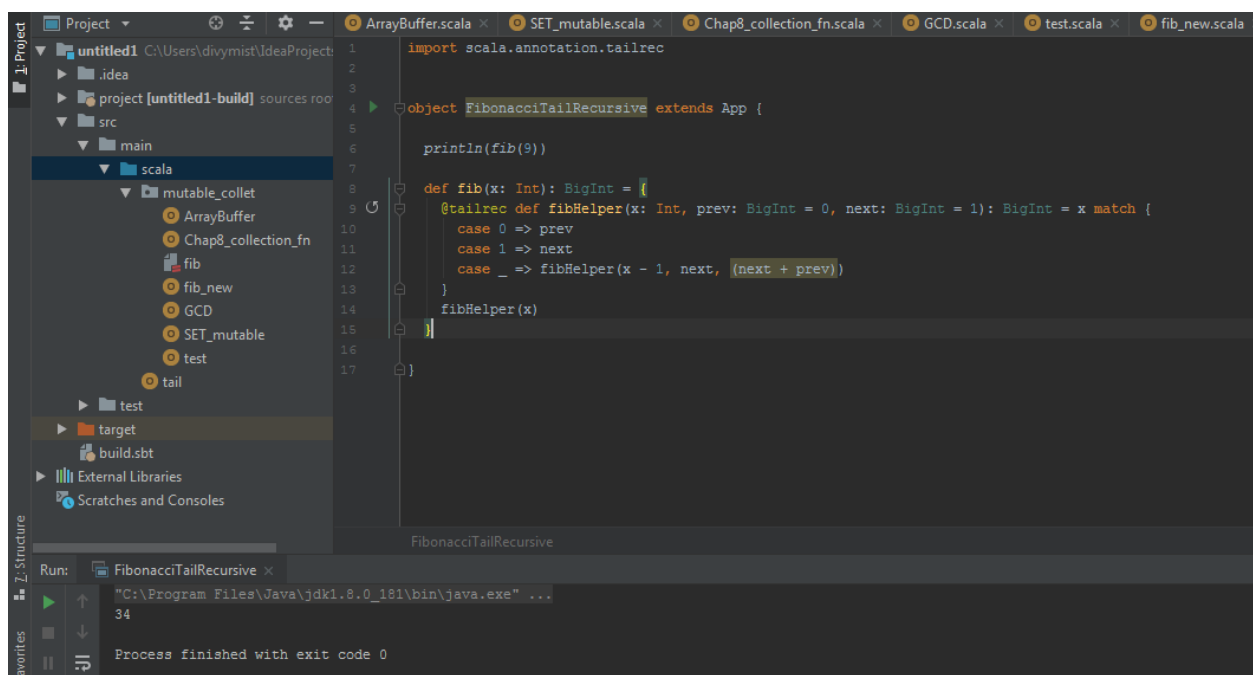
The screenshot shows an IDE with a project named 'untitled1'. The file explorer on the left shows a directory structure with 'src/main/scala' containing files like 'mutable\_collet', 'ArrayBuffer', 'Chap8\_collection\_fn', 'fib', 'fib\_new', 'GCD', 'SET\_mutable', 'test', and 'tail'. The main editor displays the code for 'fib\_new.scala'.

```
1 package mutable_collet
2
3 object fib_new extends App {
4   def fib(n: Int): Int = {
5     def calfib(n: Int, pre: Int, cur: Int): Int = {
6       if (n == 0)
7         pre
8       else
9         calfib(n - 1, cur, cur + pre)
10    }
11    calfib(n, pre = 0, cur = 1)
12  }
13
14  for (i <- 1 to 10)
15    println(fib(i))
16}
```

The Run console at the bottom shows the execution of 'fib\_new' with the following output:

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

➤ Write the function using recursion



The screenshot shows the same IDE with the 'fib\_new.scala' file replaced by 'FibonacciTailRecursive.scala'. The file explorer on the left shows the same directory structure. The main editor displays the code for 'FibonacciTailRecursive.scala'.

```
1 import scala.annotation.tailrec
2
3 object FibonacciTailRecursive extends App {
4   println(fib(9))
5
6   def fib(x: Int): BigInt = {
7     @tailrec def fibHelper(x: Int, prev: BigInt = 0, next: BigInt = 1): BigInt = x match {
8       case 0 => prev
9       case 1 => next
10      case _ => fibHelper(x - 1, next, (next + prev))
11    }
12    fibHelper(x)
13  }
14}
```

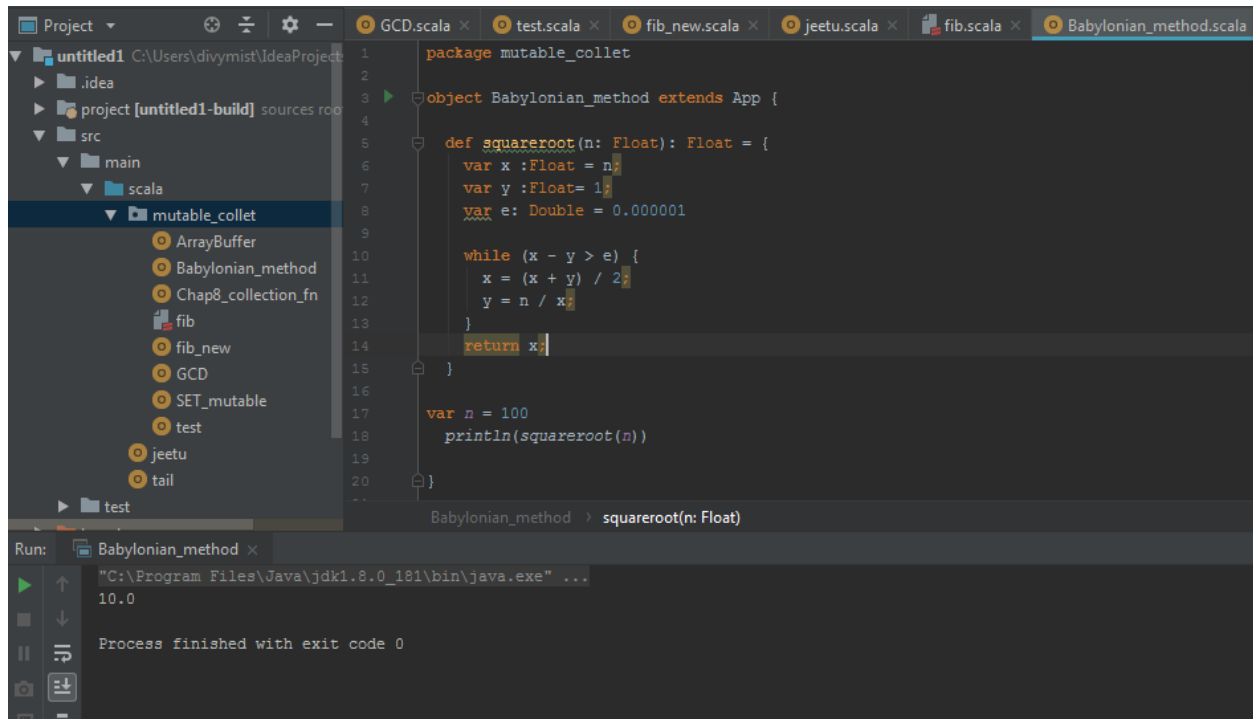
The Run console at the bottom shows the execution of 'FibonacciTailRecursive' with the following output:

```
34
Process finished with exit code 0
```

### 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  $y = n/x$



The screenshot shows an IDE with a project named 'untitled1'. The file explorer on the left shows a directory structure with 'src/main/scala/mutable\_collet' containing several files, including 'Babylonian\_method'. The main editor displays the code for 'Babylonian\_method.scala'. The code defines a package 'mutable\_collet' and an object 'Babylonian\_method' extending 'App'. It implements a 'squareroot' function that takes a 'Float'  $n$  and returns a 'Float'. The function initializes  $x$  as  $n$ ,  $y$  as 1, and a tolerance  $e$  as 0.000001. It then enters a 'while' loop where  $x$  and  $y$  are updated until their difference is less than  $e$ . Finally, it prints the result of 'squareroot(100)'.

```
1 package mutable_collet
2
3 object Babylonian_method extends App {
4
5   def squareroot(n: Float): Float = {
6     var x :Float = n
7     var y :Float= 1
8     var e: Double = 0.000001
9
10    while (x - y > e) {
11      x = (x + y) / 2
12      y = n / x
13    }
14    return x
15  }
16
17  var n = 100
18  println(squareroot(n))
19
20 }
```

The Run console at the bottom shows the output '10.0' and 'Process finished with exit code 0'.