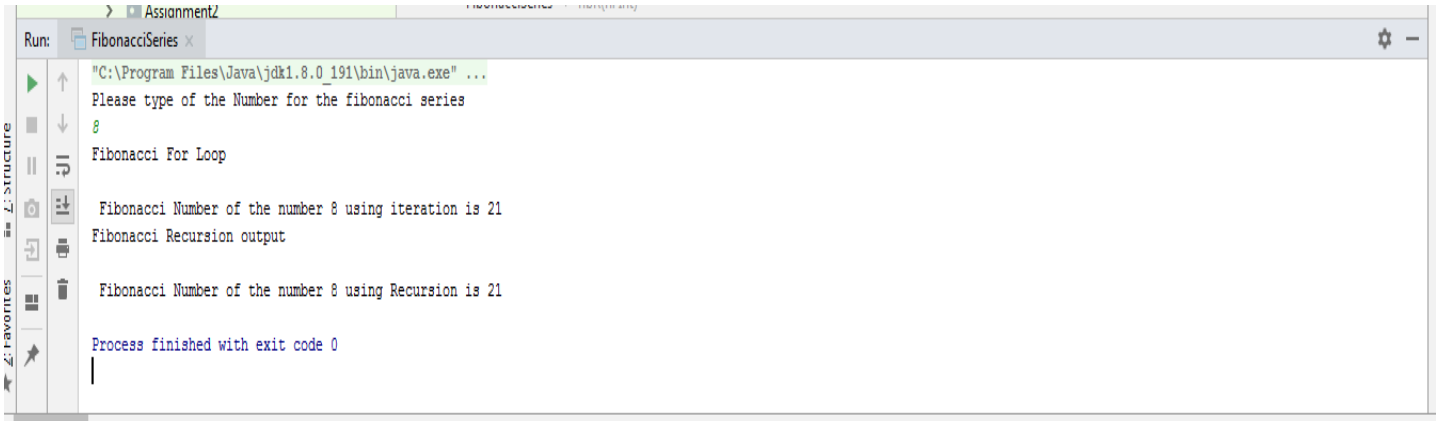
```
/*A Fibonacci series (starting from 1) written in order without any spaces in between,
thus
producing a sequence of digits.
  Write a Scala application to find the Nth digit in the sequence.
 Write the function using standard for loop
 Write the function using recursion
*/

package Assignment2

object FibonacciSeries {

  // Scala function - Fibonacci numbers using For loop
  def fib(n:Int):Int = {
    // first 2 terms
    var a = 1
    var b = 1
    // Sum of n terms
    var c = 0
    //For loop
    for (i <- 3 to n)
    {
      //Add the previous 2 terms in the series and store
      c = a + b
      //Shift the previous terms
      a = b; b = c
    }
    return c
  }


  // Scala function - Fibonacci numbers using Recursion
  def fibR(n:Int):Int = {
    // Declare an array to store Fibonacci numbers
    var f = new Array [Int] (n+2)
    var i = 0
    // First 2 terms
    f(0) = 0
    f(1) = 1
    //Recursion loop
    for (i <- 2 to n)
    {
      //Add the previous 2 terms in the series and store
      f(i) = f(i-1) + f(i-2)
    }
    return f(n)
  }


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

    println("Please type of the Number for the fibonacci series")
    val n=scala.io.StdIn.readInt()
    println("Fibonacci For Loop")
    val fibIteration=fib(n)
    //println(fibIteration)

    println(s"\n Fibonacci Number of the number $n using iteration is $fibIteration")

    println("Fibonacci Recursion output")

    val fibRecurseion=fib(n)
    println(s"\n Fibonacci Number of the number $n using Recursion is $fibRecurseion")
```

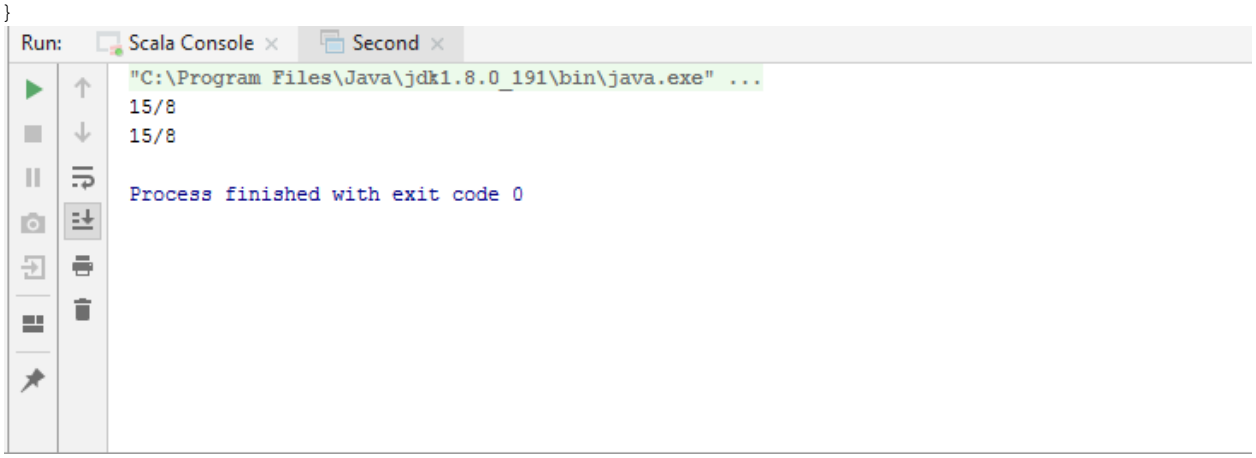
```
    //println(fibRecurseion)

  }
}
```




```
/*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.
*/


package Assignment2

object Second {

  class Rational(n: Int, d: Int) {
    //Scala Class - Rational Numbers

    def this(n: Int) = this(n, 1)

    private def gcd(a: Int, b: Int): Int =
      if (b == 0) a else gcd(b, a % b)
    private val g = gcd(n, d)

    val numer: Int = n / g
    val denom: Int = d / g

    // Add (+) method
    def +(that: Rational): Rational =
      new Rational(numer * that.denom + that.numer * denom,
        denom * that.denom)

    // Subtract (-) method
    def -(that: Rational): Rational =
      new Rational(numer * that.denom - that.numer * denom,
        denom * that.denom)
```

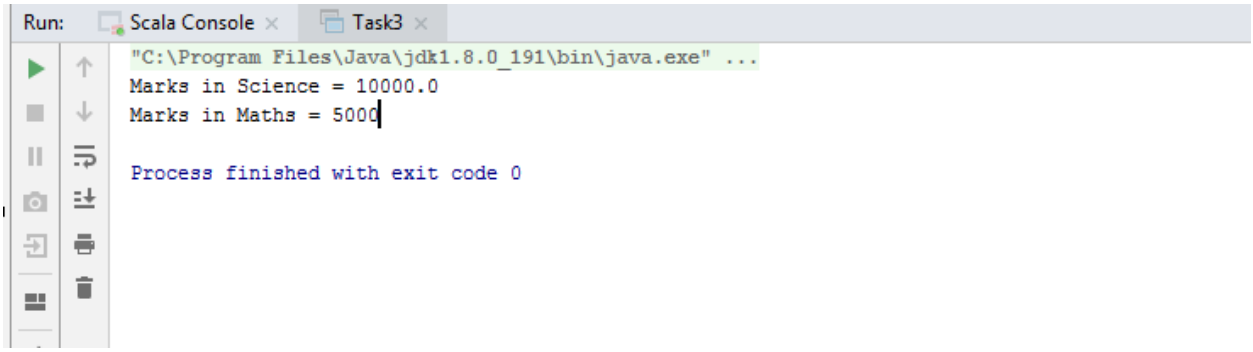
```
    // Multiply (*) method

    def *(that: Rational): Rational =
      new Rational(numer * that.numer, denom * that.denom)

    // Divide (/) method
    def /(that: Rational): Rational =
      new Rational(numer * that.denom, denom * that.numer)
    override def toString() = numer+"/"+denom
  }


  def main(args: Array[String]): Unit = {
    // val s = new Empdetails()

    val x = new Rational(1, 2);
    val y = new Rational(13, 4)
    println( (x + y) * x)
    println( (x.+(y)).*(x)    )


  }
```

}

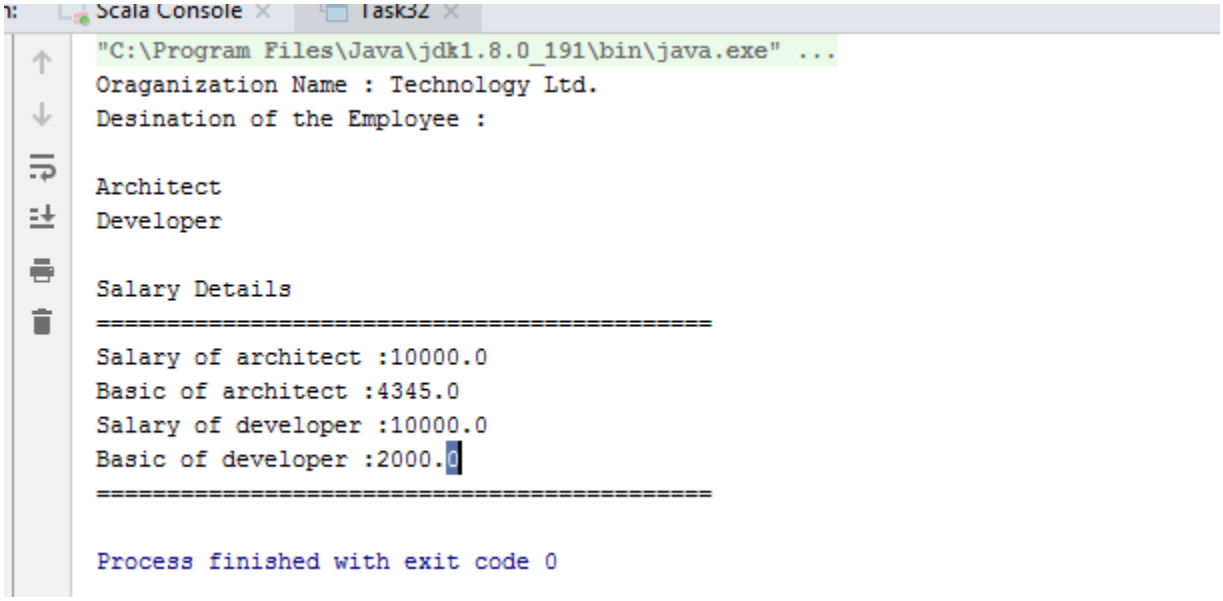



```
/*1.Write a simple program to show inheritance in scala.


package Assignment2

object Task3 {
  def main(args: Array[String]): Unit = {

    //Scala - Inheritance
    //Parent Class
    class Student{
      var ScienceMarks:Float = 10000
    }
    //Child Class
    class Programmer extends Student{
      var MathsMarks:Int = 5000
```

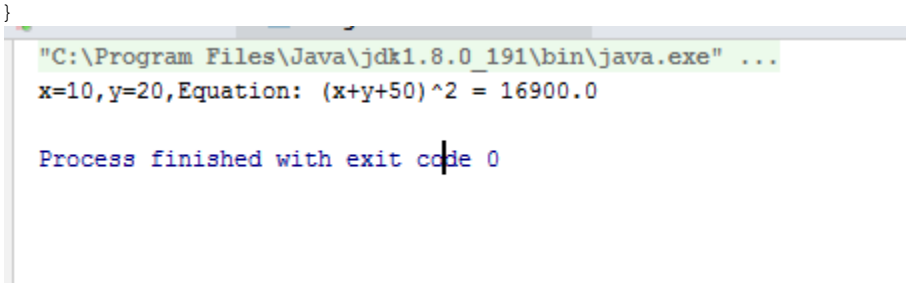
```
      println("Marks in Science = " + ScienceMarks)
      println("Marks in Maths = " + MathsMarks)
    }
    //Create Object
    new Programmer()


  }
}
```




```
/*2.Write a simple program to show multiple inheritance in scala.*/

package Assignment2

object Task32 {class Employee{
  var salary:Float = 10000
}
  class Designation extends Employee{
    var Desi1:String = "Architect"
    var Desi2:String = "Developer"
    var basci_arc:Float = 4345
    var basic_dev:Float = 2000
  }
class Empdetails extends Designation{
  var org:String = "Technology Ltd."
  println("Oraganization Name : "+org)
  println("Desination of the Employee : ")
  print("\n")
  println(Desi1)
  println(Desi2)
  print("\n")
  println("Salary Details ")
  println("=============================================")
  println("Salary of architect :"+salary)
  println("Basic of architect :"+basci_arc)
  println("Salary of developer :"+salary)
  println("Basic of developer :"+basic_dev)
  println("=============================================")
}
  def main(args: Array[String]): Unit = {
    val s = new Empdetails()
  }
}
```




```
/*3.Write a partial function to add three numbers in which one number is constant and two
  numbers can be passed as inputs and define another method which can take the partial
function as input and squares the result.*/

package Assignment2

object Assignment2Problem3 {


  val additon: PartialFunction[(Double,Double),Double] = {
    case(a,b) => a + b + 100
  }
  def squeareRoot(x:Double,y:Double): Double = {

    val sqRoor=Math.pow(additon(x,y),2)
    return  sqRoor
  }

def main(args: Array[String]): Unit = {
  println("x=10,y=20,Equation: (x+y+100)^2 = " + squeareRoot(10,20))
}
}
```




```
/*4.Write a program to print the prices of 4 courses of Acadgild: Android-12999,Big Data
Development-17999,Big Data Development-17999,Spark-19999 using match and add a
default condition if the user enters any other course*/
```

```
package Assignment2

object A2Problem4 {

  def acadgildCourse(): Unit =
     {
        println("Acadgild offers 4 Courses")
        println("Android, Bigdata, Datascience, Spark \n")
       val courseName=scala.io.StdIn.readLine("Type of the coruse do you want to know
the fee")

      val courseFee = courseName match {
      case "Bigdata" => println(s"The price of the $courseName: 17999 \n")
      case "Spark" => println(s"The price of the $courseName: 19999 \n")
      case "Datascience" =>  println(s"The price of the $courseName: 29999 \n")
      case "Android" =>  println(s"The price of the $courseName: 999 \n")
      case _ =>  println(s"PLease enter the corect course Name")
      }
    }

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




  }
}
```