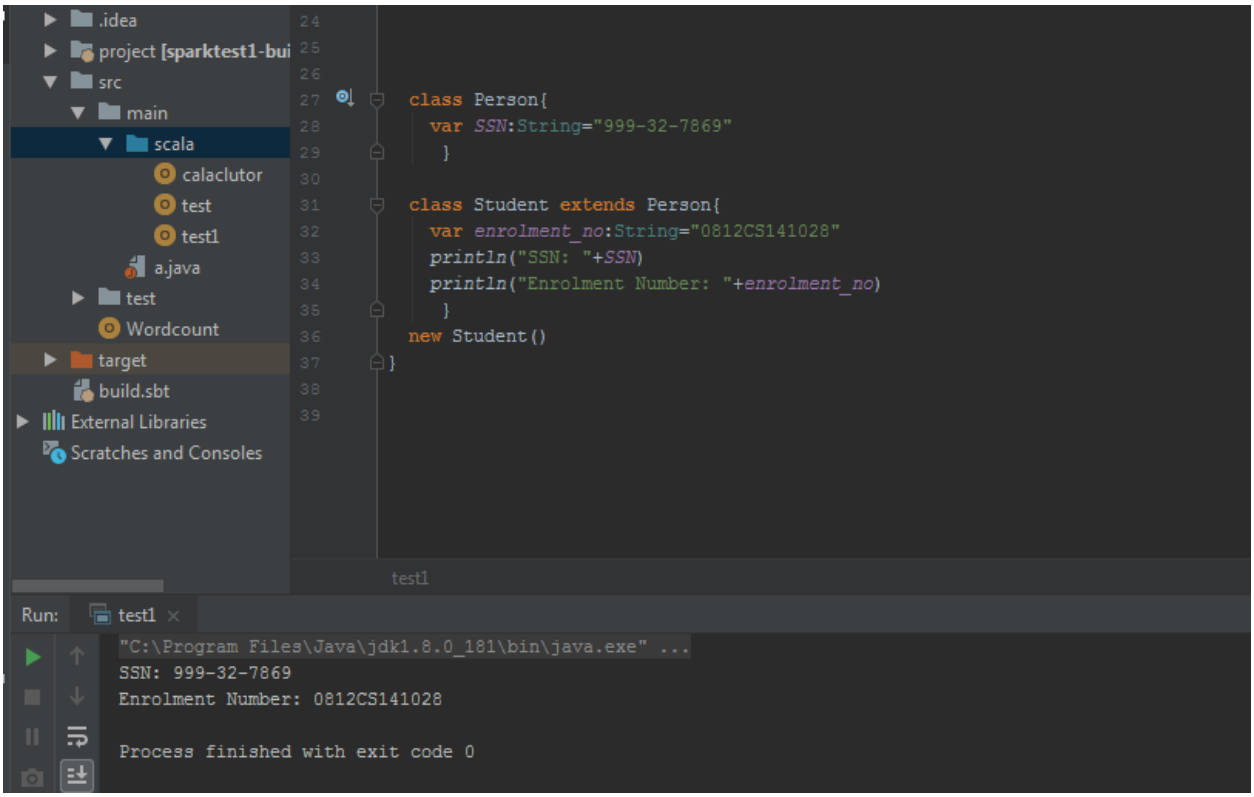
Session 17: SCALA BASICS 4: Assignment 1

Task 1

Write a simple program to show inheritance in scala.

Code:

```
class Person{
   var SSN:String="999-32-7869"
    }

  class Student extends Person{
   var enrolment_no:String="0812CS141028"
   println("SSN: "+SSN)
   println("Enrolment Number: "+enrolment_no)
    }
  new Student()
}
```



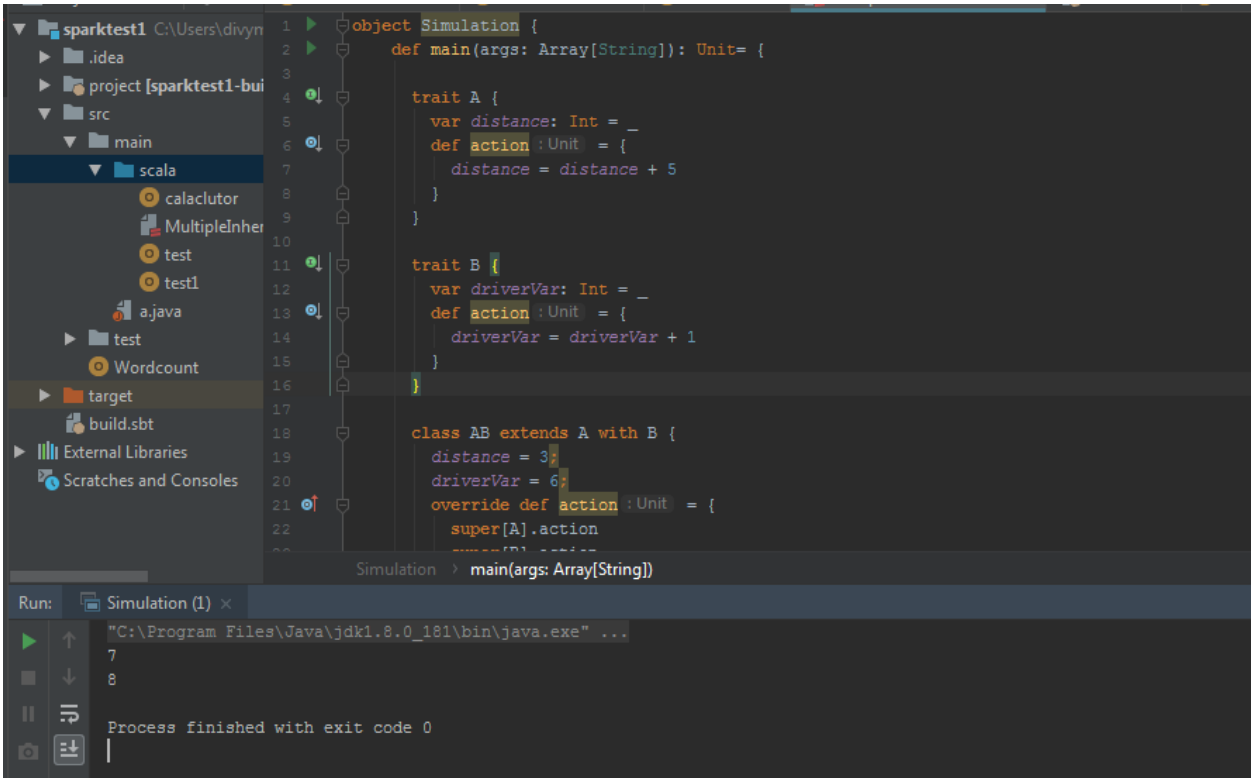
Task 2

Write a simple program to show multiple inheritance in Scala

Scala doesn't allow for multiple inheritance *per se*, but allows us to extend multiple traits.

CODE:

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

     trait A {
       var distance: Int = _
       def action = {
         distance = distance + 5
       }
     }

     trait B {
       var driverVar: Int = _
       def action = {
         driverVar = driverVar + 1
       }
     }

     class AB extends A with B {
       distance = 3;
       driverVar = 6;
       override def action = {
         super[A].action
         super[B].action
       }
     }

     var ab = new AB
     ab.action
     println(ab.driverVar)
     println(ab.distance)
   }


}
```

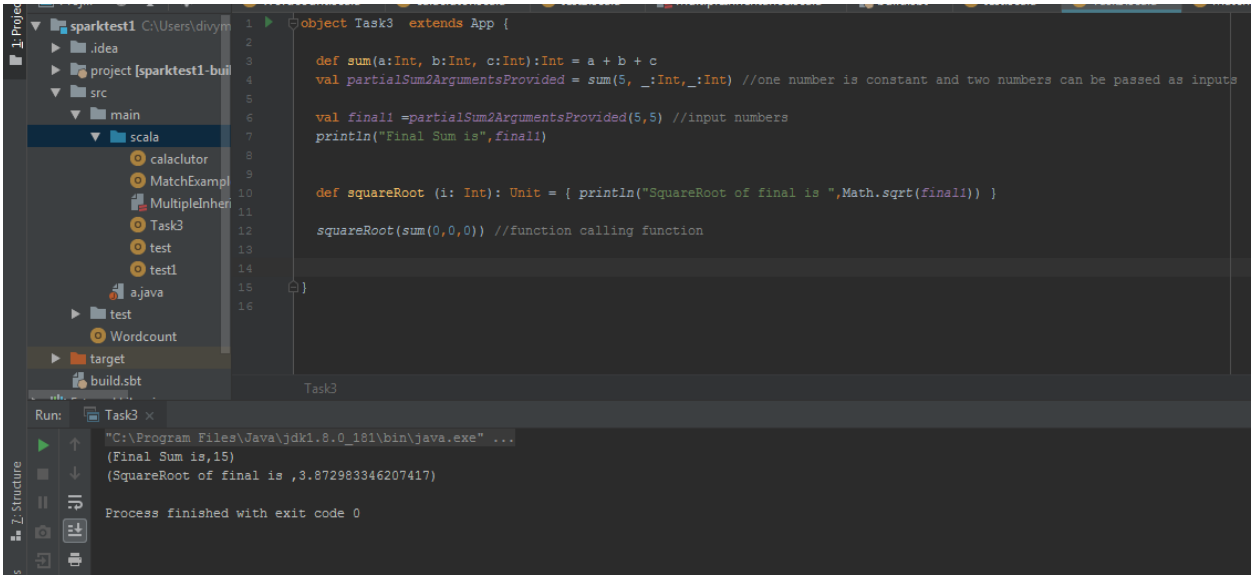


Task 3

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.

Code:

```
object Task3  extends App {

  def sum(a:Int, b:Int, c:Int):Int = a + b + c
  val partialSum2ArgumentsProvided = sum(5, _:Int,_:Int) //one number is constant and
two numbers can be passed as inputs

  val final1 =partialSum2ArgumentsProvided(5,5) //input numbers
  println("Final Sum is",final1)


  def squareRoot (i: Int): Unit = { println("SquareRoot of final is ",Math.sqrt(final1)) }

  squareRoot(sum(0,0,0)) //function calling function


}
```



Task 4

Write a program to print the prices of 4 courses of Acadgild:

Android App Development -14,999 INR

Data Science - 49,999 INR

Big Data Hadoop & Spark Developer – 24,999 INR

Blockchain Certification – 49,999 INR

using match and add a default condition if the user enters any other course.

```
object MatchExample {
  def main(args: Array[String]) {
    println(matchTest(1))
    println(matchTest(6))
  }

  def matchTest(x: Int): String = x match {
    case 1 => "Android App Development -14,999 INR"
    case 2 => "Data Science - 49,999 INR"
    case 3 => "Big Data Hadoop & Spark Developer – 24,999 INR"
    case 4 => "Blockchain Certification – 49,999 INR"
    case _ => "This Course is not Available "//default condition if the user enters any other
course.
  }

}
```

Code with output

```scala
object MatchExample {
  def main(args: Array[String]) : Unit  {
    println(matchTest(1))
    println(matchTest(6))
  }

  def matchTest(x: Int): String = x match {
    case 1 => "Android App Development -14,999 INR"
    case 2 => "Data Science - 49,999 INR"
    case 3 => "Big Data Hadoop & Spark Developer - 24,999 INR"
    case 4 => "Blockchain Certification - 49,999 INR"
    case _ => "This Course is not Available "//default condition if the user enters any other course.
  }

}
```

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Android App Development -14,999 INR
This Course is not Available

Process finished with exit code 0
```