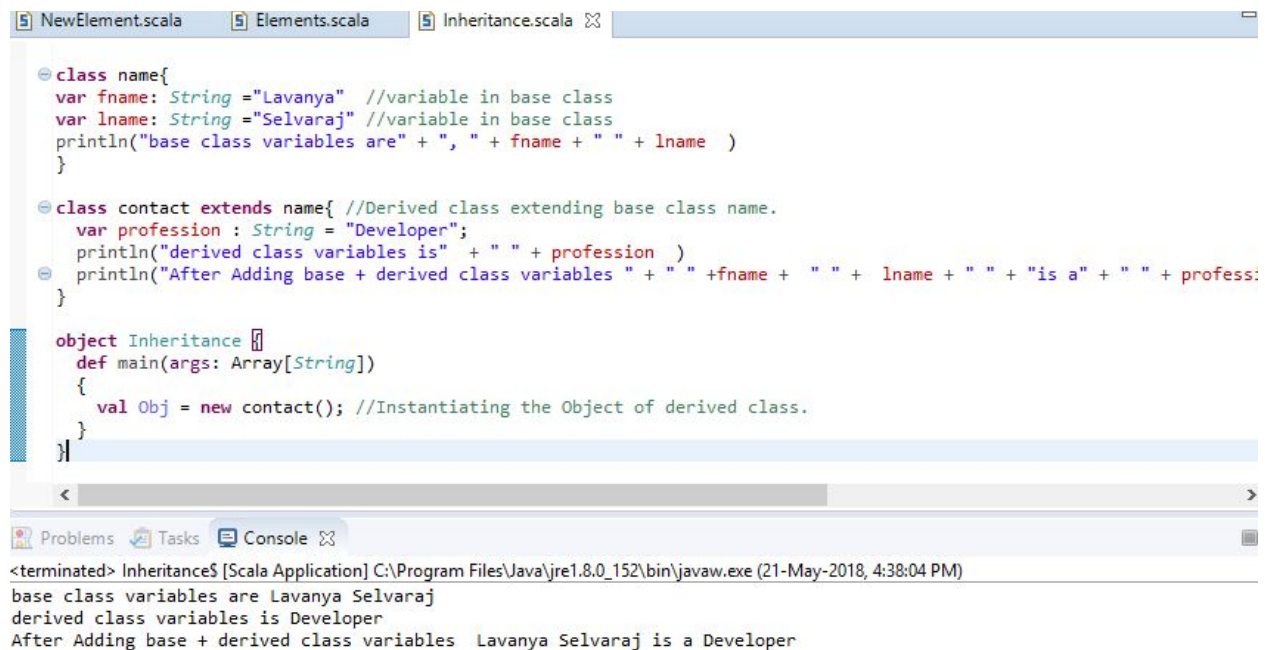


Assignment17(Scala 4):

All the files for the below tasks are uploaded separately with explanation.

Task 1: Write a simple program to show inheritance in scala.

Output:



```
class name{
  var fname: String = "Lavanya" //variable in base class
  var lname: String = "Selvaraj" //variable in base class
  println("base class variables are" + ", " + fname + " " + lname )
}

class contact extends name{ //Derived class extending base class name.
  var profession : String = "Developer";
  println("derived class variables is" + " " + profession )
  println("After Adding base + derived class variables " + " " + fname + " " + lname + " " + "is a" + " " + profession)
}

object Inheritance {
  def main(args: Array[String]) {
    val Obj = new contact(); //Instantiating the Object of derived class.
  }
}
```

<terminated> Inheritance\$ [Scala Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (21-May-2018, 4:38:04 PM)
base class variables are Lavanya Selvaraj
derived class variables is Developer
After Adding base + derived class variables Lavanya Selvaraj is a Developer

Task 2 Write a simple program to show multiple inheritance in scala

Multiple inheritance cannot be implemented using normal classes like c++.

Class base

```
{ }
```

Class deriv :extends base()

```
{def: show() }
```

Class deriv2 : extends base()

```
{def :show }
```

Class deriv3 : extends deriv, deriv2()

```
{ }
```

In the above example class deriv3 inherits from both deriv and deriv2 which gets inherited from base class already which will cause a diamond problem. In this problem the main function gets confused on to call which function/variable that was inherited (ie deriv:show() or deriv2:show. This can be fixed using traits as shown in the snapshot example below.

The below statement throws an error in scala,

Class deriv3 : extends deriv, deriv2()

Class deriv3 : extend public deriv with public deriv2()

Instead it can be as below:

Abstract Class base

```
{ def: show = println("This is base class")}
```

Trait deriv :extends base()

```
{override def: show = println("This is base class")}
```

Trait deriv2 : extends base()

```
{override def: show = println("This is base class")}
```

Class deriv3 : extends deriv, deriv2()

```
{}
```

Output:

```
NewElement.scala  Elements.scala  Inheritance.scala  MultipleInheritance.scala  partialFunc.scala

}

trait Engineer extends profession{
  override def workat = "Engineer works at Office"
  def workswith = "Engineer works with Computer"
}

class Farmers extends doctor with Engineer {

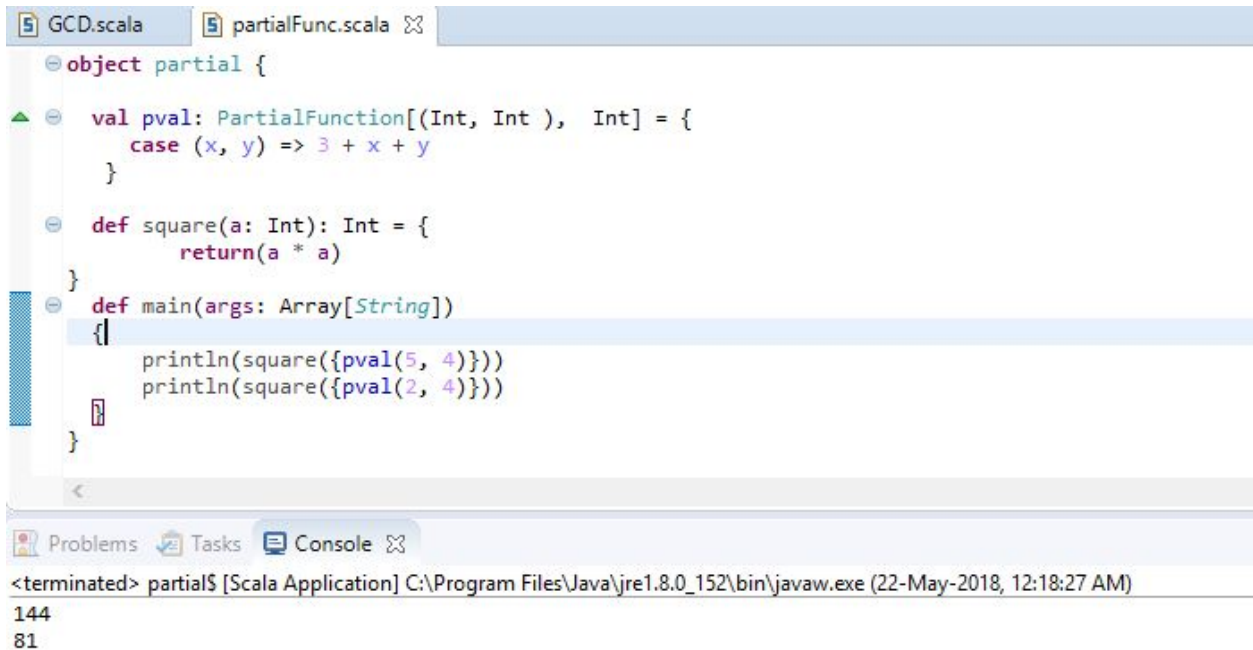
  println(super[doctor].workat)
}

object MultipleInheritance {
  def main(args: Array[String]) {
    val Obj = new Farmers()
    println(Obj.workat)
    println(Obj.workswith)
    println(Obj.wears)
  }
}

<
Problems  Tasks  Console  ⌵
<terminated> MultipleInheritance$ [Scala Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (21-May-2018, 6:05:03 PM)
Doctor works at hospital
Engineer works at Office
Engineer works with Computer
Doctor wears white court
```

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.

Output:



The screenshot shows an IDE with two tabs: 'GCD.scala' and 'partialFunc.scala'. The 'partialFunc.scala' tab is active, displaying the following Scala code:

```
object partial {  
  val pval: PartialFunction[(Int, Int), Int] = {  
    case (x, y) => 3 + x + y  
  }  
  
  def square(a: Int): Int = {  
    return(a * a)  
  }  
  
  def main(args: Array[String])  
  {  
    println(square({pval(5, 4)}))  
    println(square({pval(2, 4)}))  
  }  
}
```

Below the code editor, there is a 'Console' tab showing the output of the program:

```
<terminated> partial$ [Scala Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (22-May-2018, 12:18:27 AM)  
144  
81
```

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

Output:

```
LearningScala2.sc  Matchcase.scala  GCD.scala

object Matchcase {

  def printCourse(course : Int) = {

    // val course1 = 10                                //> number : Int = 10
    course match {
      case 1 => println(f"Android App Development - 14,999 INR")
      case 2 => println("Data Science - 49,999 INR")
      case 3 => println("Data Hadoop & Spark Developer - 24,999 INR")
      case 4 => println("Blockchain Certification - 49,999 INR")
      case _ =>println("Different course chosen")
    }
  }

  def main(args: Array[String]){
    for(x <- 1 to 5)
    {
      printCourse(x)
    }
  }
}

Problems  Tasks  Console
<terminated> Matchcase$ [Scala Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (22-May-2018, 12:13:04 AM)
Android App Development - 14,999 INR
Data Science - 49,999 INR
Data Hadoop & Spark Developer - 24,999 INR
Blockchain Certification - 49,999 INR
Different course chosen
```