BIG DATA HADOOP & SPARK TRAINING

Assignment on Scala IV

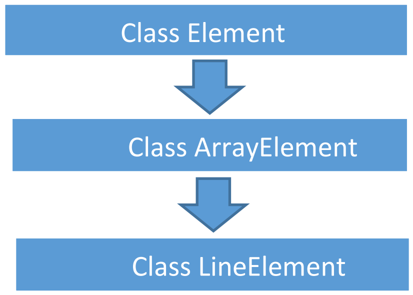
Rashmi Krishna

Task 1

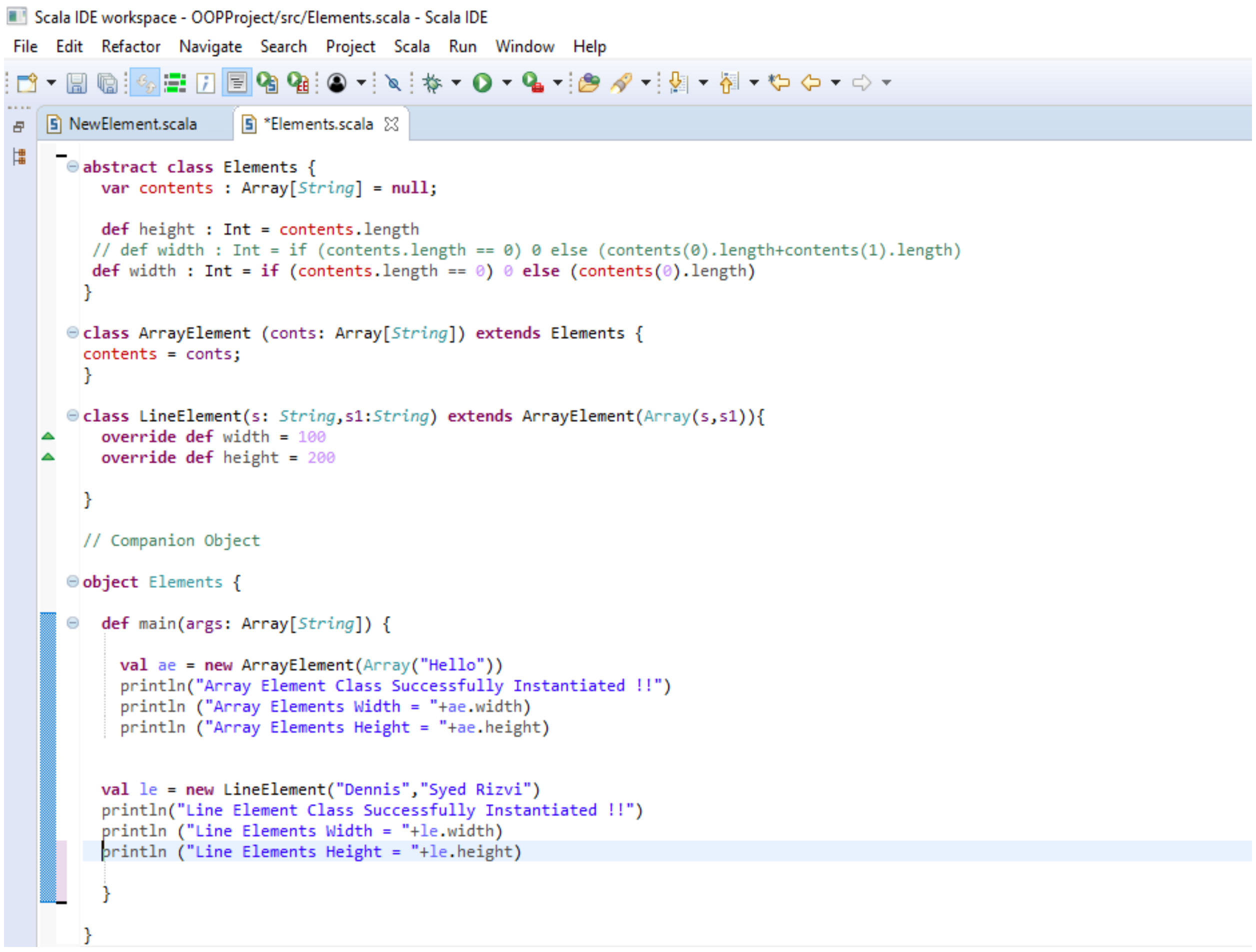
Write a simple program to show inheritance in scala.

- Here we have created an abstract class “Elements” which has two abstract methods namely “height & width”
  - Height is the number of lines in a file or string
  - Width is the length of string or number of words in a file or string
- We create another class “ArrayElement” which extends the abstract class Elements.
  - Here Elements is the superclass for ArrayElements
  - ArrayElements is subclass of Elements
  - We pass an array of string to superclass Elements from the ArrayElements subclass
  - Methods of Elements class can be utilized in ArrayElements class as, ArrayElements class has inherited non-private attributes of class Elements
- We create another class “LineElements” which further extends ArrayElements.
  - LineElements is subclass of ArrayElements and Elements
  - ArrayElements is the superclass of LineElements, where ArrayElements is subclass of Elements, hence Elements is superclass of both ArrayElements and LineElements.
  - Methods of Elements class can be utilized in LineElements class, as LineElements class has inherited all properties or attributes from ArrayElements class, and ArrayElements class has inherited non-private attributes of class Elements.
  - So, we can pass two strings “S & S1” which have the height and width constants to superclass Elements.
- We create a companion object “Elements”
  - It has the same name that of the abstract class, that’s why this singleton object is called as companion object here
  - This has a main method, where the application starts to run the program
  - Here we pass arguments to ArrayElements and LineElements class

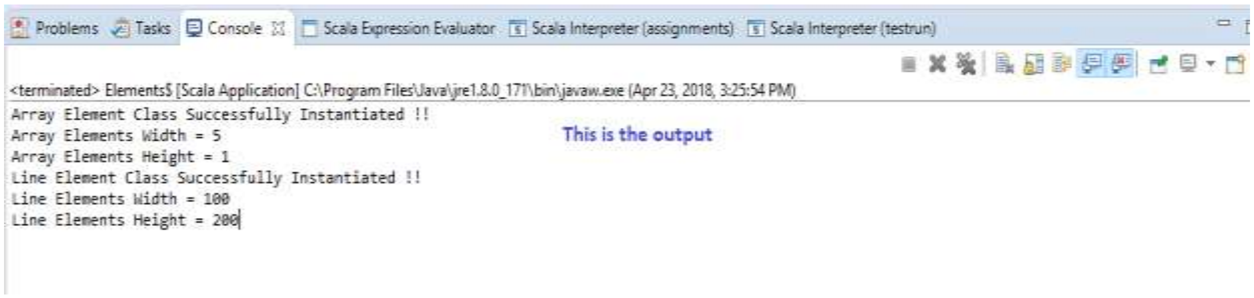
- We can see the outputs:
  - Firstly the ArrayElements class is instantiated for the word “Hello”
    - Width :5 (word length)
    - Height:1(number of words)
  - Secondly, the LineElements class is instantiated
    - This prints the constant values for height and width that is given in the LineElements class
    - Height:200, width:100



This diagram, pictorially represents that all non-private attribute of Elements class is inherited by ArrayElement class and LineElement Class. As Element class is super class for both ArrayElement and LineElement


Scala IDE workspace - OOPProject/src/Elements.scala - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
NewElement.scala
*Elements.scala

```
abstract class Elements {
  var contents : Array[String] = null;

  def height : Int = contents.length
 // def width : Int = if (contents.length == 0) 0 else (contents(0).length+contents(1).length)
 def width : Int = if (contents.length == 0) 0 else (contents(0).length)
}

class ArrayElement (conts: Array[String]) extends Elements {
contents = conts;
}

class LineElement(s: String,s1:String) extends ArrayElement(Array(s,s1)){
  override def width = 100
  override def height = 200

}

// Companion Object

object Elements {

  def main(args: Array[String]) {

    val ae = new ArrayElement(Array("Hello"))
    println("Array Element Class Successfully Instantiated !!")
    println ("Array Elements Width = "+ae.width)
    println ("Array Elements Height = "+ae.height)


  val le = new LineElement("Dennis","Syed Rizvi")
  println("Line Element Class Successfully Instantiated !!")
  println ("Line Elements Width = "+le.width)
  println ("Line Elements Height = "+le.height)

  }

}
```


Problems
Tasks
Console
Scala Expression Evaluator
Scala Interpreter (assignments)
Scala Interpreter (testrun)
This is the output

```
<terminated> Elements$ [Scala Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Apr 23, 2018, 3:25:54 PM)
Array Element Class Successfully Instantiated !!
Array Elements Width = 5
Array Elements Height = 1
Line Element Class Successfully Instantiated !!
Line Elements Width = 100
Line Elements Height = 200
```

Task 2

Write a simple program to show multiple inheritance in Scala

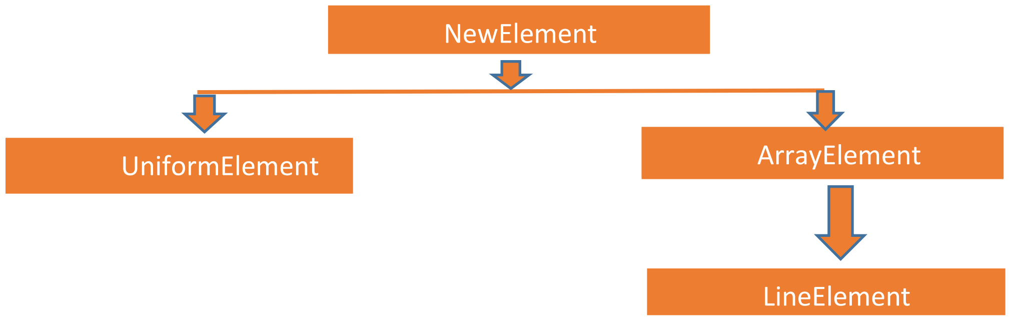
- Here we have created an abstract class "NewElement" which has a abstract method namely "demo"
    - It just prints "NewElement implementation invoked"
- We create another class "ArrayElement" which extends the abstract class NewElement.
    - Here NewElement is the superclass for ArrayElements
    - ArrayElements is subclass of NewElement
    - Methods of NewElement class can be utilized in ArrayElements class as, ArrayElements class has inherited non-private attributes of class NewElement
    - Here we override the function "demo" of NewElement class and print "ArrayElements implementation is invoked"

- We create another class "LineElements" which further extends ArrayElements.
    - LineElements is subclass of ArrayElements and NewElement
    - ArrayElements is the superclass of LineElements, where ArrayElements is subclass of NewElement, hence NewElement is superclass of both ArrayElements and LineElements.
    - Methods of NewElement class can be utilized in LineElements class, as LineElements class has inherited all properties or attributes from ArrayElements class, and ArrayElements class has inherited non-private attributes of class NewElement.
    - So, we again override the "demo" function to print "LineElements implementation is invoked".

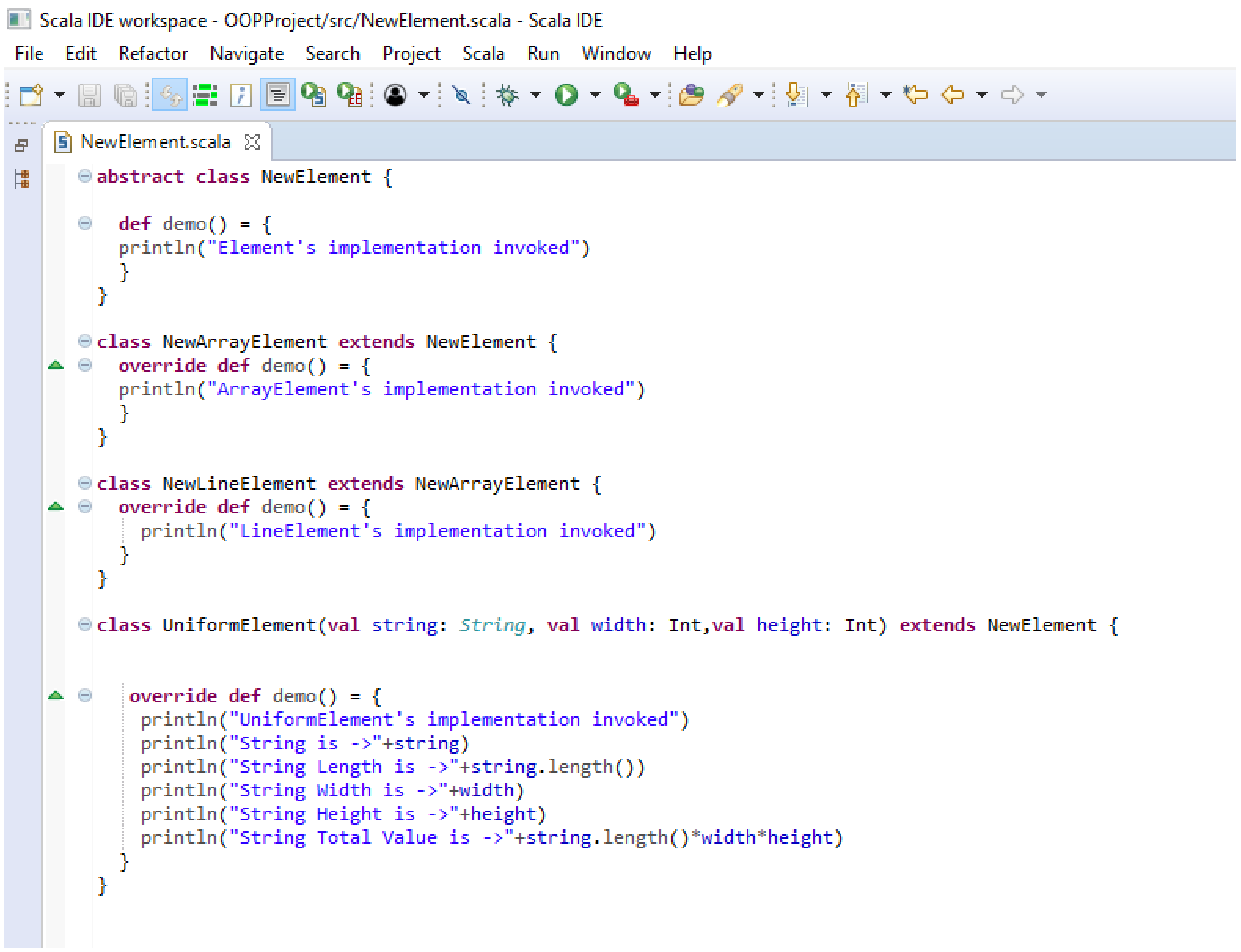
- We create another class "UniformElement" which directly extends "NewElement" and it has 3 parameters "String, height and width". This is called "Parameterized constructors"
    - This class again overrides the NewElement "demo" method to print the string and height and width of the string

- We create a companion object “NewElement”
  - It has the same name that of the abstract class, that’s why this singleton object is called as companion object here
  - Here we invoke the NewElement superclass.
  - This has a main method, where the application starts to run the program
  - Here we instantiate to ArrayElements ,LineElements and uniformElements class and also pass required arguments to UniformElements class i.e string-Rashmi, string length-6, width -2 and height-3
  - Print the lines of all the classes

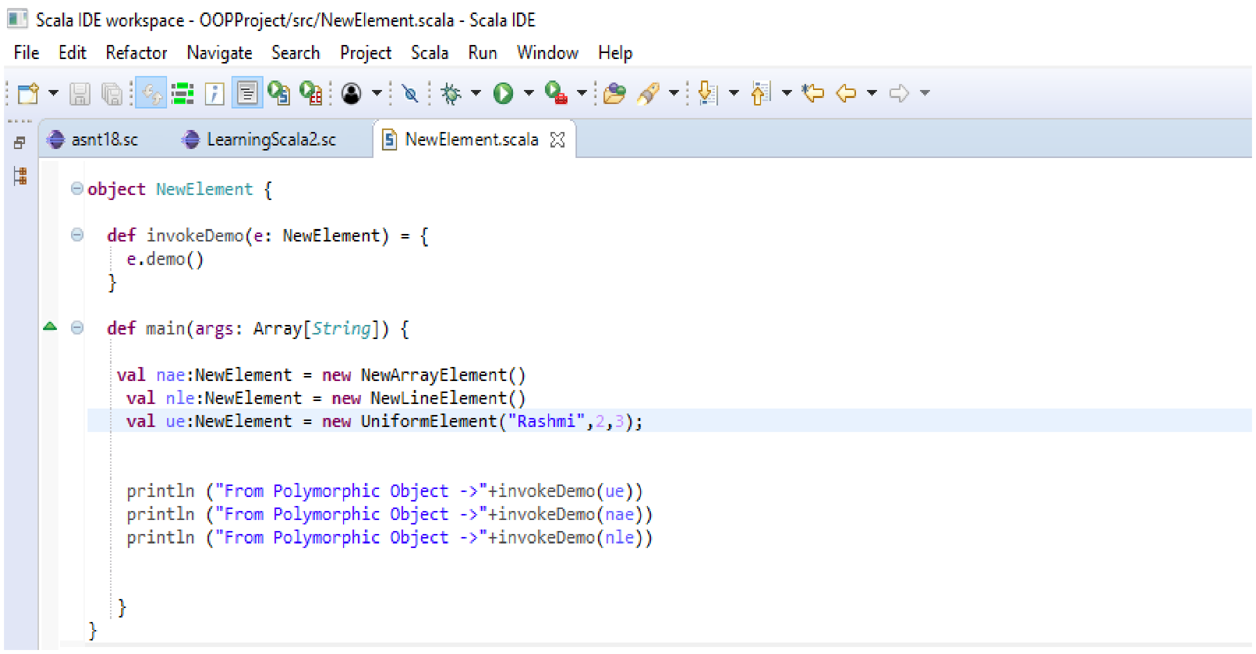
- We can see that first UniformElement is activated which prints the passed arguments and then ArrayElement and LineElement



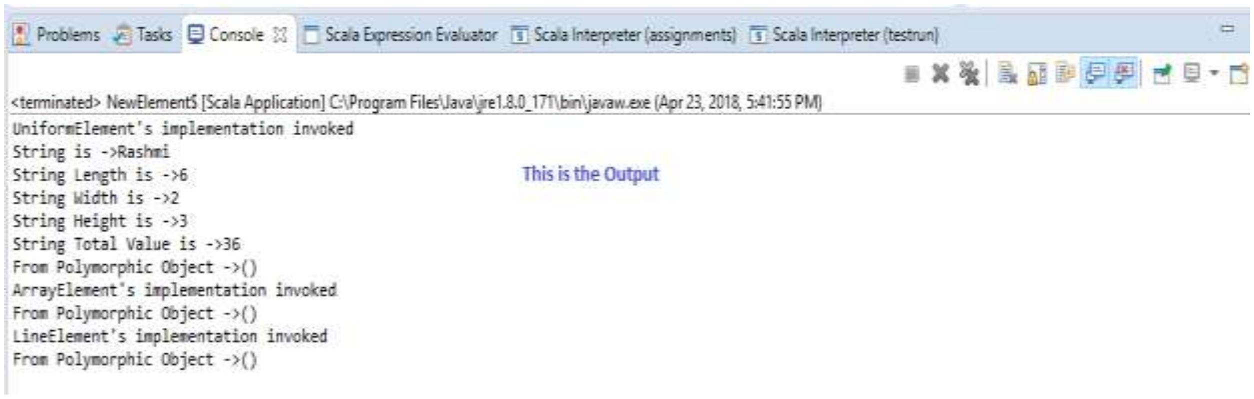
This diagram, pictorially represents that all non-private attribute of Elements class is inherited by ArrayElement class and LineElement Class inherits from ArrayElements class. As Element class is super class for both ArrayElement and LineElement. UniformElement is directly the subclass of NewElement. Thereby, NewElement is subclass of all classes


Scala IDE workspace - OOPProject/src/NewElement.scala - Scala IDE

```
abstract class NewElement {

  def demo() = {
  println("Element's implementation invoked")
  }
}

class NewArrayElement extends NewElement {
  override def demo() = {
  println("ArrayElement's implementation invoked")
  }
}

class NewLineElement extends NewArrayElement {
  override def demo() = {
    println("LineElement's implementation invoked")
  }
}

class UniformElement(val string: String, val width: Int,val height: Int) extends NewElement {


   override def demo() = {
    println("UniformElement's implementation invoked")
    println("String is ->"+string)
    println("String Length is ->"+string.length())
    println("String Width is ->"+width)
    println("String Height is ->"+height)
    println("String Total Value is ->"+string.length()*width*height)
  }
}
```


Scala IDE workspace - OOPProject/src/NewElement.scala - Scala IDE

```
object NewElement {

  def invokeDemo(e: NewElement) = {
    e.demo()
  }

  def main(args: Array[String]) {

   val nae:NewElement = new NewArrayElement()
    val nle:NewElement = new NewLineElement()
    val ue:NewElement = new UniformElement("Rashmi",2,3);


    println ("From Polymorphic Object ->"+invokeDemo(ue))
    println ("From Polymorphic Object ->"+invokeDemo(nae))
    println ("From Polymorphic Object ->"+invokeDemo(nle))


   }
}
```


Problems
Tasks
Console
Scala Expression Evaluator
Scala Interpreter (assignments)
Scala Interpreter (testrun)
This is the Output

```
<terminated> NewElement$ [Scala Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Apr 23, 2018, 5:41:55 PM)
UniformElement's implementation invoked
String is ->Rashmi
String Length is ->6
String Width is ->2
String Height is ->3
String Total Value is ->36
From Polymorphic Object ->()
ArrayElement's implementation invoked
From Polymorphic Object ->()
LineElement's implementation invoked
From Polymorphic Object ->()
```

Task 3

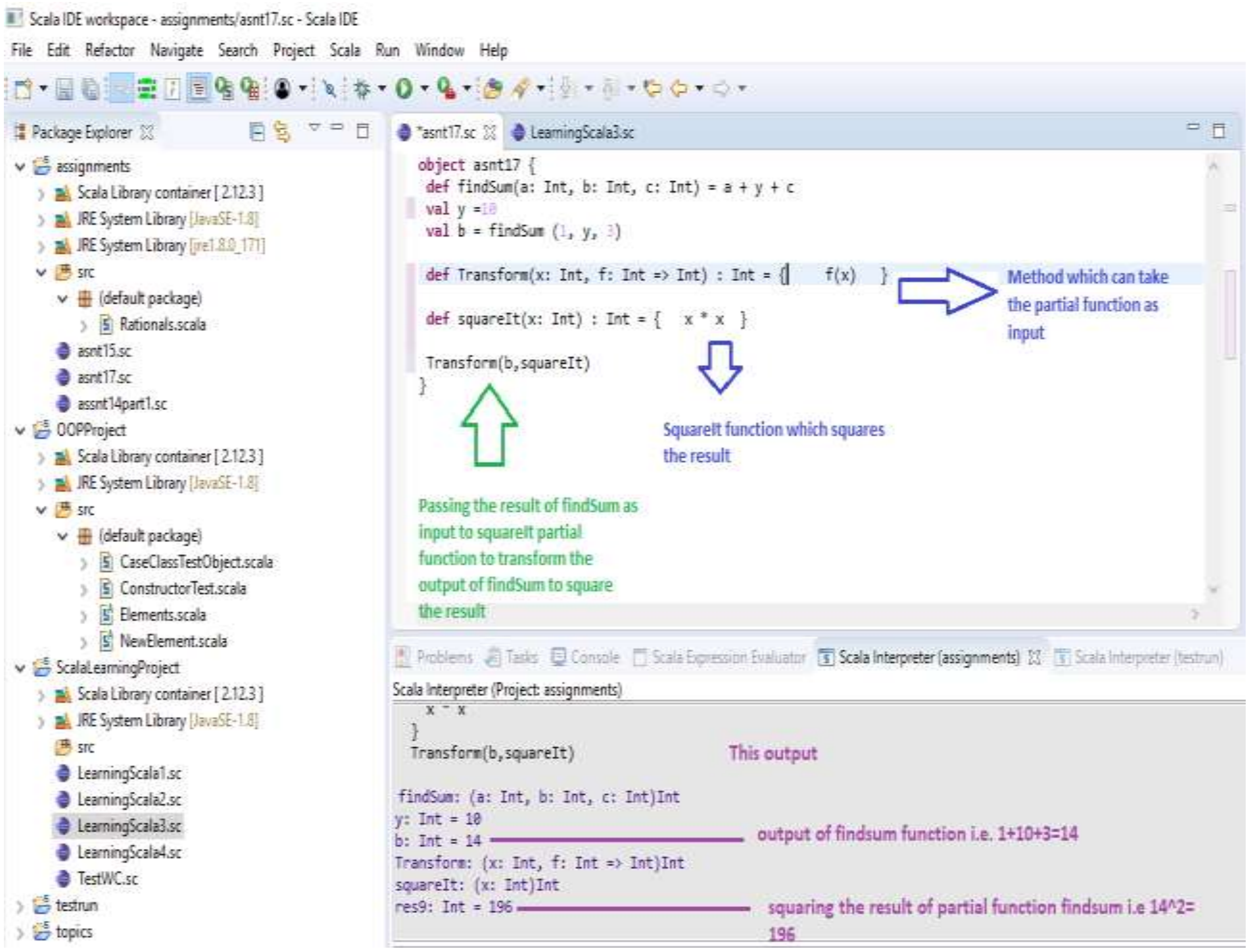
Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result.

- The below function finds sum of 3 variables, where b is constant which is inialized as y which is 10 and variables “a” and “c” are passed as inputs to the function.

```
def findSum(a: Int, b: Int, c: Int) = a + y + c
val y =10
val b = findSum (1, y, 3)
```

- The below function will calculate the square of the output of above function

```
def squareIt(x: Int) : Int = {   x * x   }

def Transform(x: Int, f: Int => Int) : Int = {f(x) }
Transform(b,squareIt)
```


Scala IDE workspace - assignments/asnt17.sc - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
Package Explorer
assignments
Scala Library container [2.12.3]
JRE System Library [JavaSE-1.8]
JRE System Library [jre1.8.0_171]
src
(default package)
Rationals.scala
asnt15.sc
asnt17.sc
assnt14part1.sc
OOPProject
Scala Library container [2.12.3]
JRE System Library [JavaSE-1.8]
src
(default package)
CaseClassTestObject.scala
ConstructorTest.scala
Elements.scala
NewElement.scala
ScalaLearningProject
Scala Library container [2.12.3]
JRE System Library [JavaSE-1.8]
src
LearningScala1.sc
LearningScala2.sc
LearningScala3.sc
LearningScala4.sc
TestWC.sc
testrun
topics
*asnt17.sc
LearningScala3.sc
Problems
Tasks
Console
Scala Expression Evaluator
Scala Interpreter (assignments)
Scala Interpreter (testrun)
Scala Interpreter (Project: assignments)

```
object asnt17 {
 def findSum(a: Int, b: Int, c: Int) = a + y + c
 val y =10
 val b = findSum (1, y, 3)

 def Transform(x: Int, f: Int => Int) : Int = {     f(x)   }

 def squareIt(x: Int) : Int = {   x * x  }

 Transform(b,squareIt)
}
```

Method which can take the partial function as input

SquareIt function which squares the result

Passing the result of findSum as input to squareIt partial function to transform the output of findSum to square the result

```
   x * x
 }
 Transform(b,squareIt)

findSum: (a: Int, b: Int, c: Int)Int
y: Int = 10
b: Int = 14
Transform: (x: Int, f: Int => Int)Int
squareIt: (x: Int)Int
res9: Int = 196
```

This output

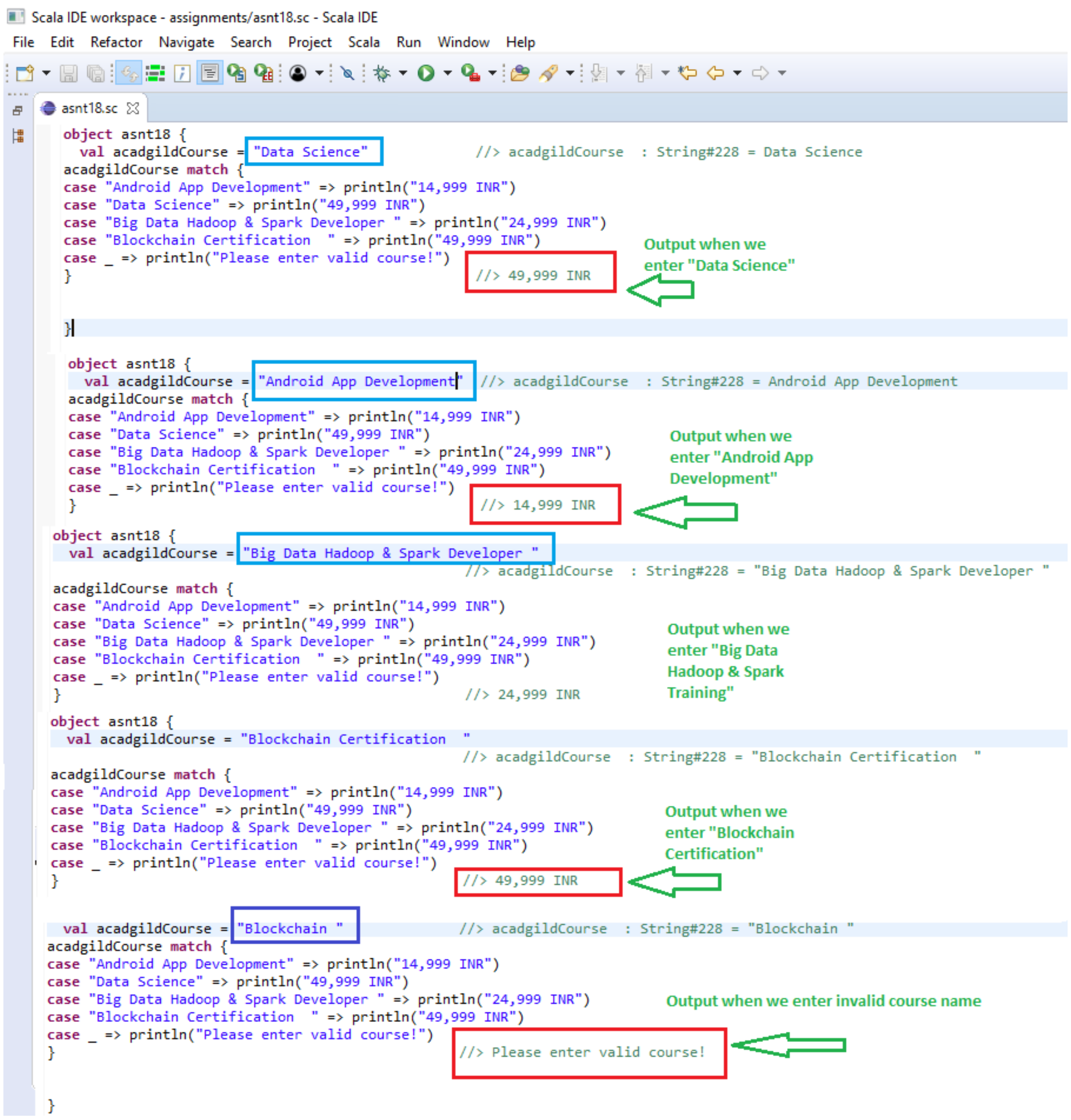
output of findsum function i.e. 1+10+3=14

squaring the result of partial function findsum i.e 14^2= 196

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**
- **Block chain Certification – 49,999 INR**
- **using match and add a default condition if the user enters any other course.**


Scala IDE workspace - assignments/asnt18.sc - Scala IDE
File Edit Refactor Navigate Search Project Scala Run Window Help
asnt18.sc

```
object asnt18 {
  val acadgildCourse = "Data Science"               //> acadgildCourse  : String#228 = Data Science
acadgildCourse match {
case "Android App Development" => println("14,999 INR")
case "Data Science" => println("49,999 INR")
case "Big Data Hadoop & Spark Developer " => println("24,999 INR")
case "Blockchain Certification  " => println("49,999 INR")
case _ => println("Please enter valid course!")
}                                                  //> 49,999 INR


}
```

Output when we enter "Data Science"

```
object asnt18 {
  val acadgildCourse = "Android App Development" //> acadgildCourse  : String#228 = Android App Development
acadgildCourse match {
case "Android App Development" => println("14,999 INR")
case "Data Science" => println("49,999 INR")
case "Big Data Hadoop & Spark Developer " => println("24,999 INR")
case "Blockchain Certification  " => println("49,999 INR")
case _ => println("Please enter valid course!")
}                                                  //> 14,999 INR
```

Output when we enter "Android App Development"

```
object asnt18 {
  val acadgildCourse = "Big Data Hadoop & Spark Developer "
                                                  //> acadgildCourse  : String#228 = "Big Data Hadoop & Spark Developer "
acadgildCourse match {
case "Android App Development" => println("14,999 INR")
case "Data Science" => println("49,999 INR")
case "Big Data Hadoop & Spark Developer " => println("24,999 INR")
case "Blockchain Certification  " => println("49,999 INR")
case _ => println("Please enter valid course!")
}                                                 //> 24,999 INR
```

Output when we enter "Big Data Hadoop & Spark Training"

```
object asnt18 {
  val acadgildCourse = "Blockchain Certification  "
                                                  //> acadgildCourse  : String#228 = "Blockchain Certification  "
acadgildCourse match {
case "Android App Development" => println("14,999 INR")
case "Data Science" => println("49,999 INR")
case "Big Data Hadoop & Spark Developer " => println("24,999 INR")
case "Blockchain Certification  " => println("49,999 INR")
case _ => println("Please enter valid course!")
}                                                 //> 49,999 INR
```

Output when we enter "Blockchain Certification"

```
  val acadgildCourse = "Blockchain "              //> acadgildCourse  : String#228 = "Blockchain "
acadgildCourse match {
case "Android App Development" => println("14,999 INR")
case "Data Science" => println("49,999 INR")
case "Big Data Hadoop & Spark Developer " => println("24,999 INR")
case "Blockchain Certification  " => println("49,999 INR")
case _ => println("Please enter valid course!")
}                                                 //> Please enter valid course!


}
```

Output when we enter invalid course name