

Assignment17

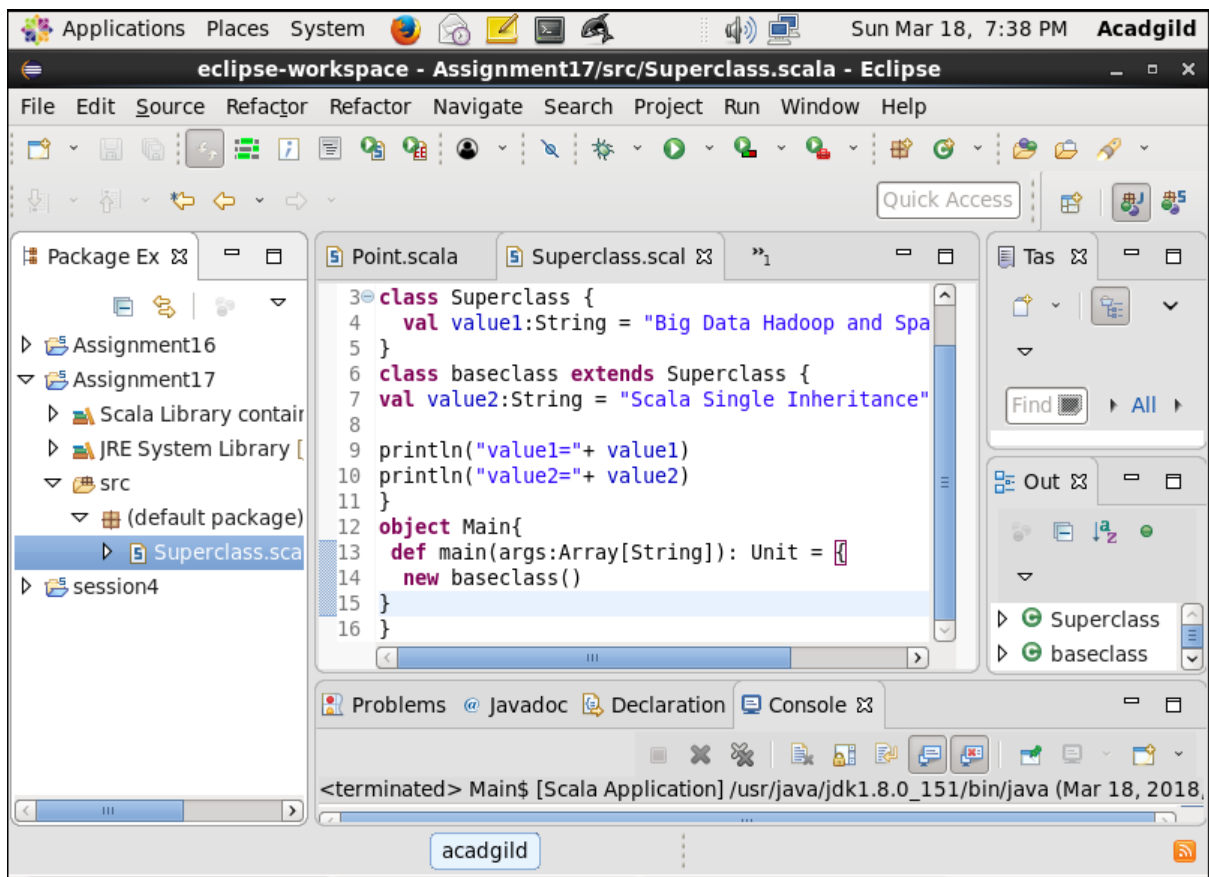
Problem Statement

Task 1

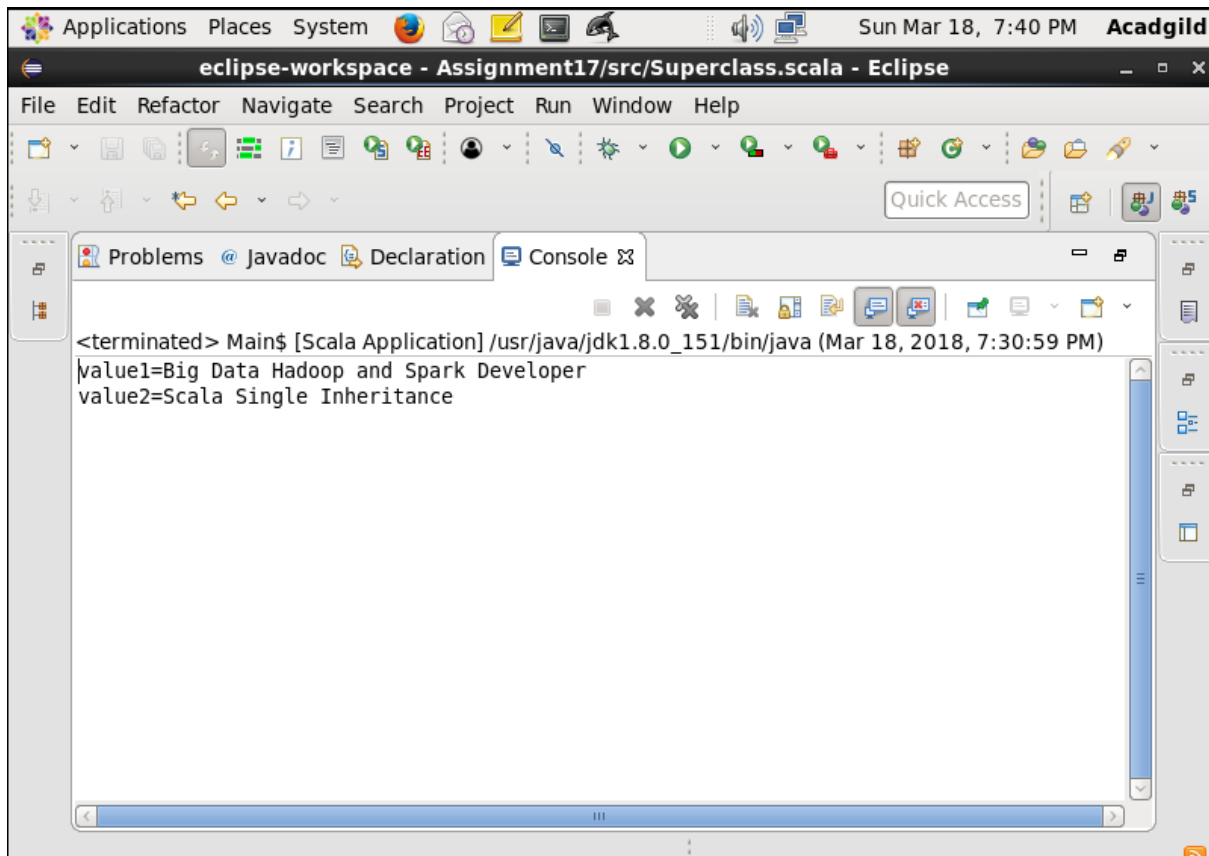
Write a simple program to show inheritance in scala.

Input Scala code:

```
class Superclass
{
val value1:String = "Big Data Hadoop and Spark Developer"
}
class baseclass extends Superclass {
val value2:String = "Scala Single Inheritance"
println("value1="+ value1)
println("value2="+ value2)
}
object Main{
def main(args:Array[String]): Unit = {
new baseclass()
}
}
```



Output:



Task 2

Write a simple program to show multiple inheritance in scala

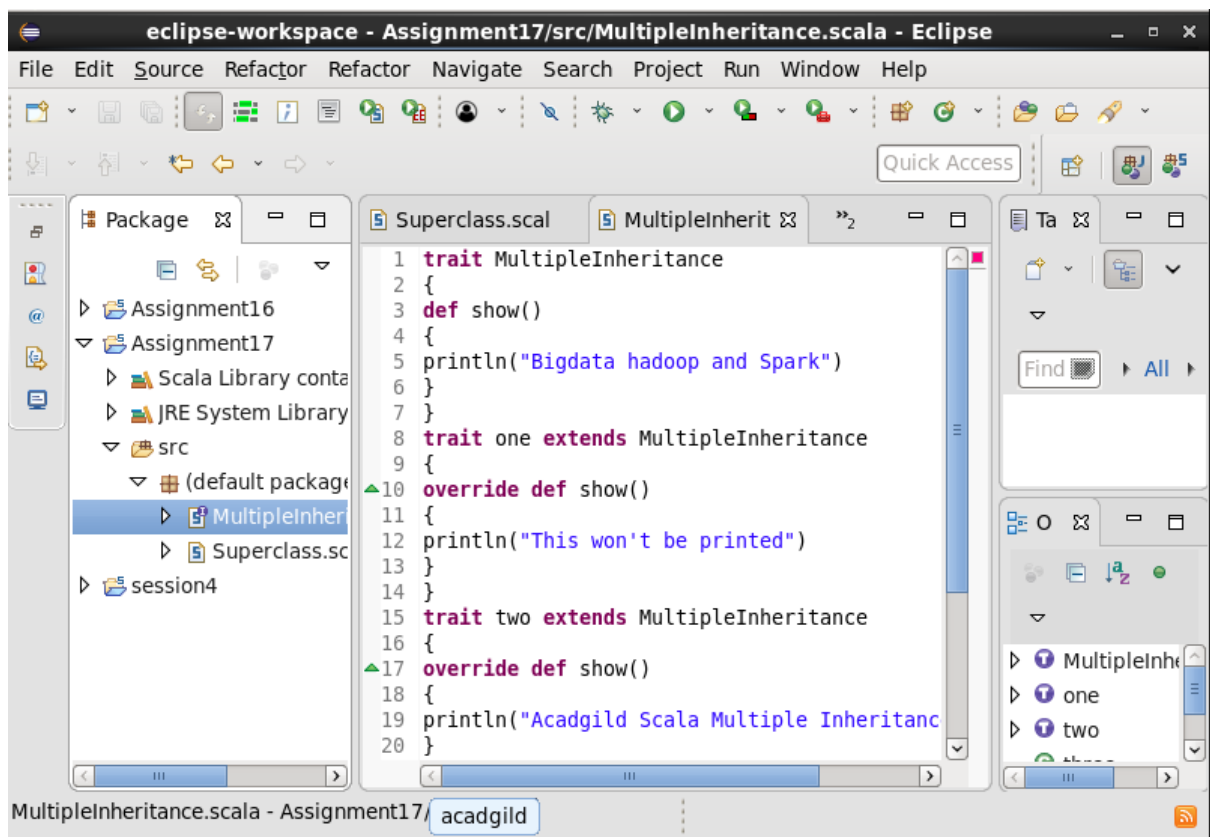
Input Scala Code:

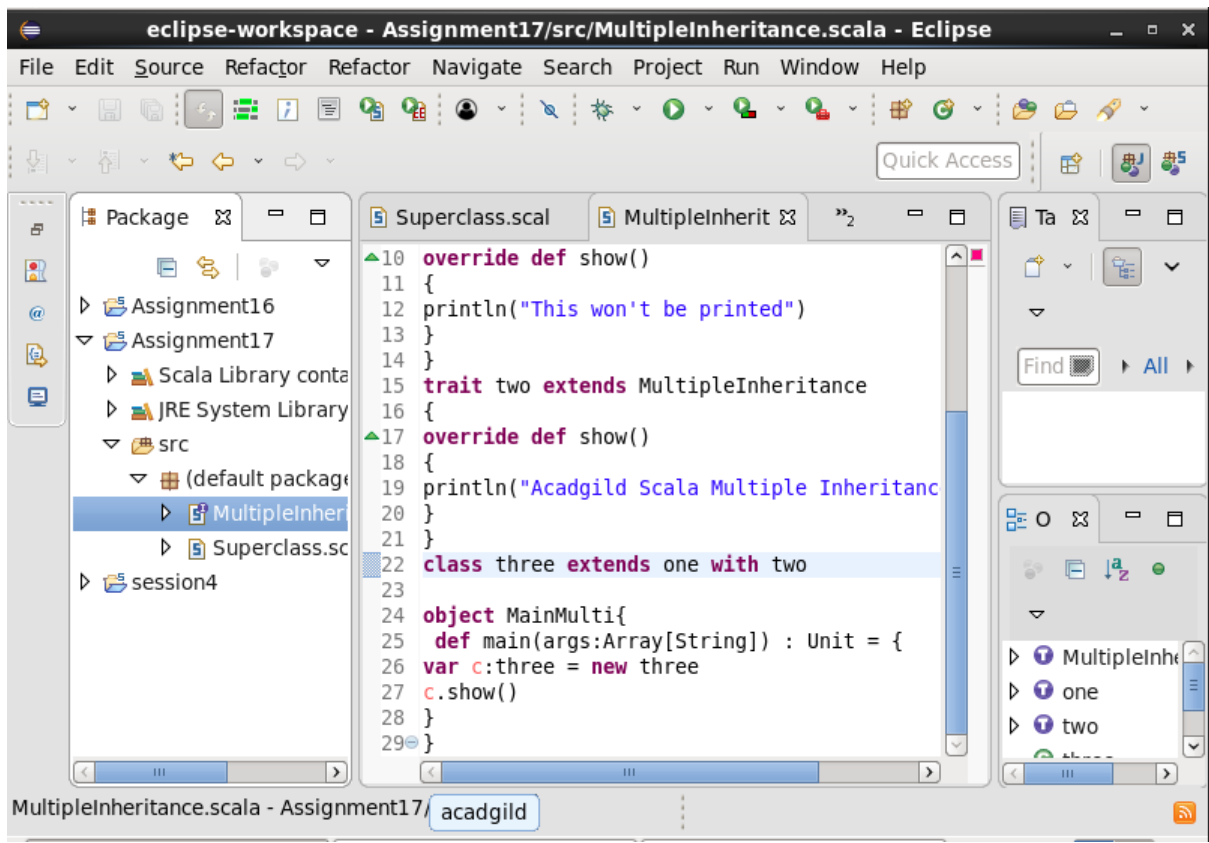
```
trait MultipleInheritance
{
  def show()
  {
    println("Bigdata hadoop and Spark")
  }
}
trait one extends MultipleInheritance
{
  override def show()
  {
    println("This won't be printed")
  }
}
trait two extends MultipleInheritance
{
  override def show()
  {
    println("Acadgild Scala Multiple Inheritance Example")
  }
}
class three extends one with two
show()
```

```

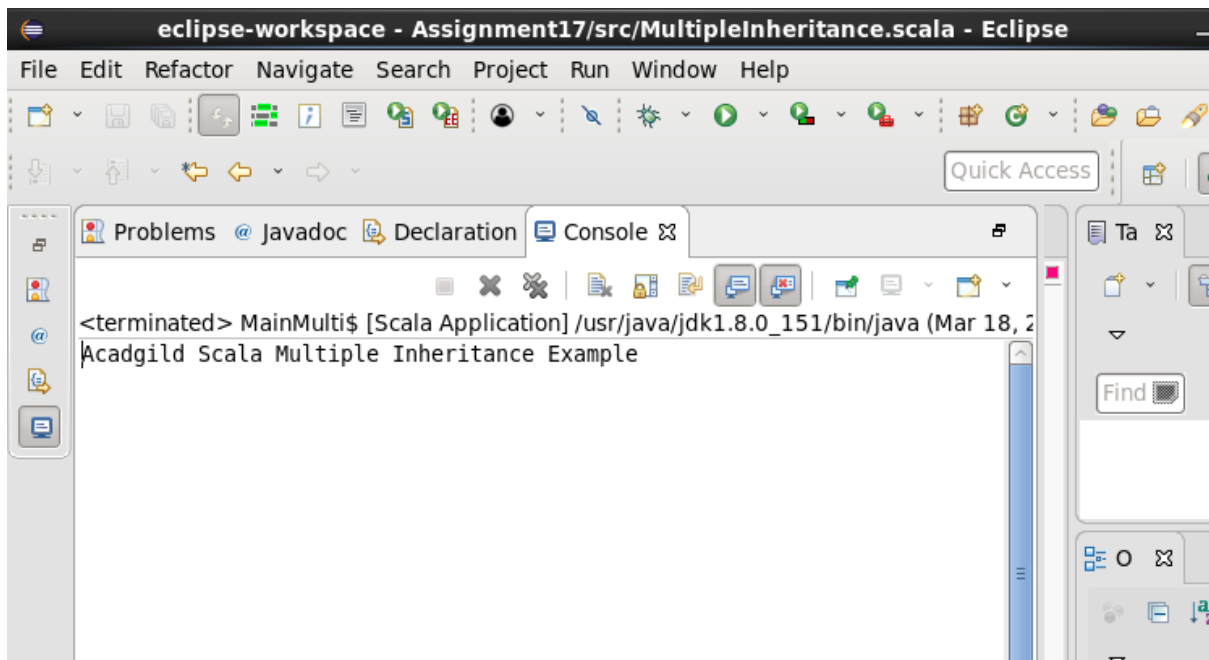
object MainMulti{
  def main(args:Array[String]) : Unit = {
    var c:three = new three
    c.show()
  }
}

```





Output:

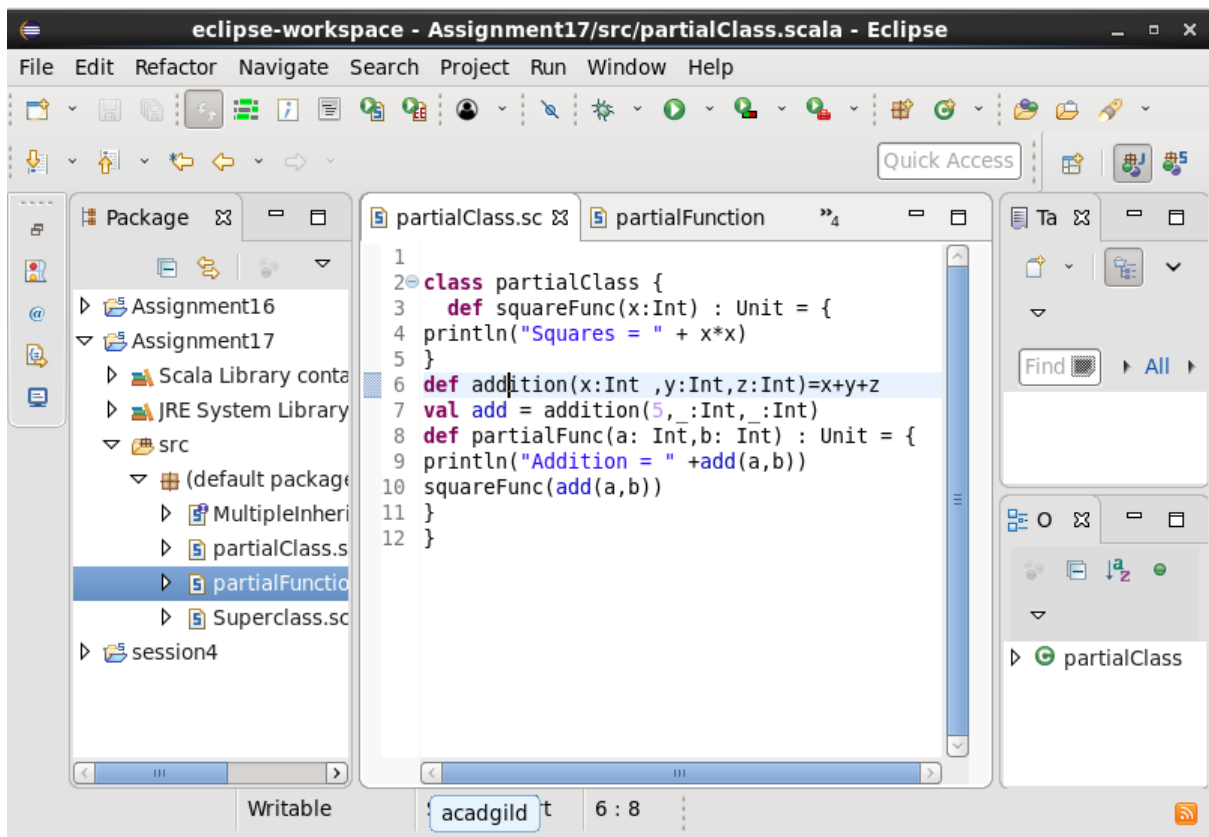


Task 3

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.

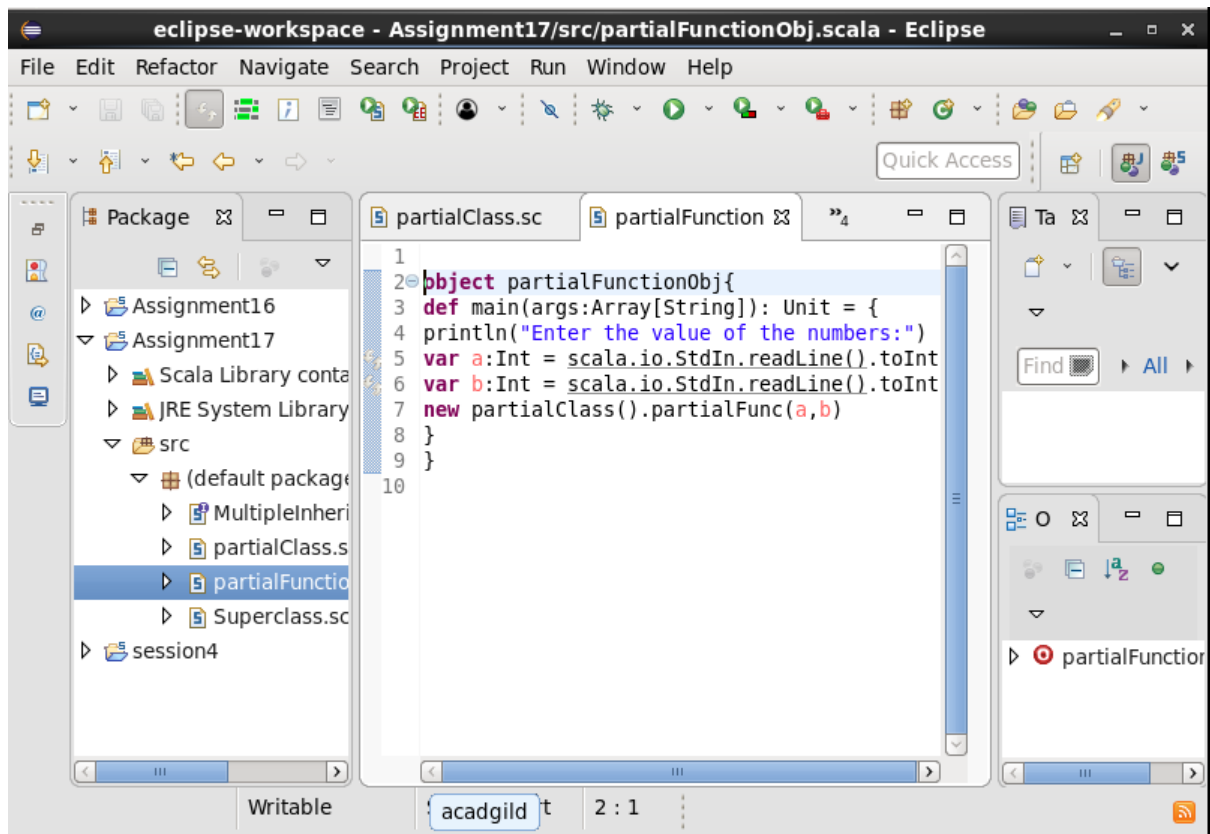
Input Scala Code:

```
class partialClass{  
  def squareFunc(x:Int) : Unit = {  
    println("Squares = " + x*x)  
  }  
  def addition(x:Int ,y:Int,z:Int)=x+y+z  
  val add = addition(5,_,_:Int,_:Int)  
  def partialFunc(a: Int,b: Int) : Unit = {  
    println("Addition = " +d(a,b))  
    squareFunc(add(a,b))  
  }  
}
```

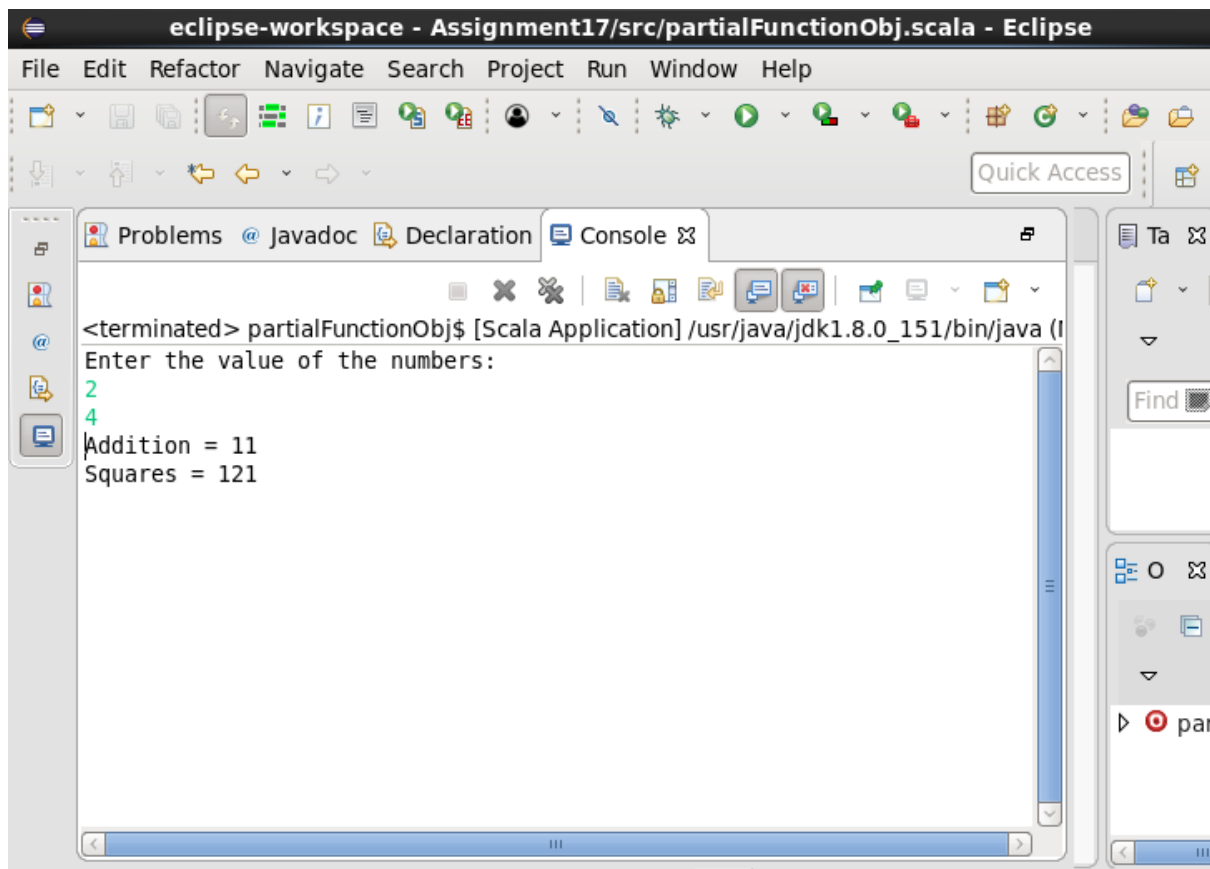


ObjectCode:

```
object partialFunctionObj{  
  def main(args:Array[String]): Unit = {  
    println("Enter the value of the numbers:")  
    var a:Int = scala.io.StdIn.readLine().toInt  
    var b:Int = scala.io.StdIn.readLine().toInt  
    new PartialClass().partialFunc(a,b)  
  }  
}
```



Output:



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

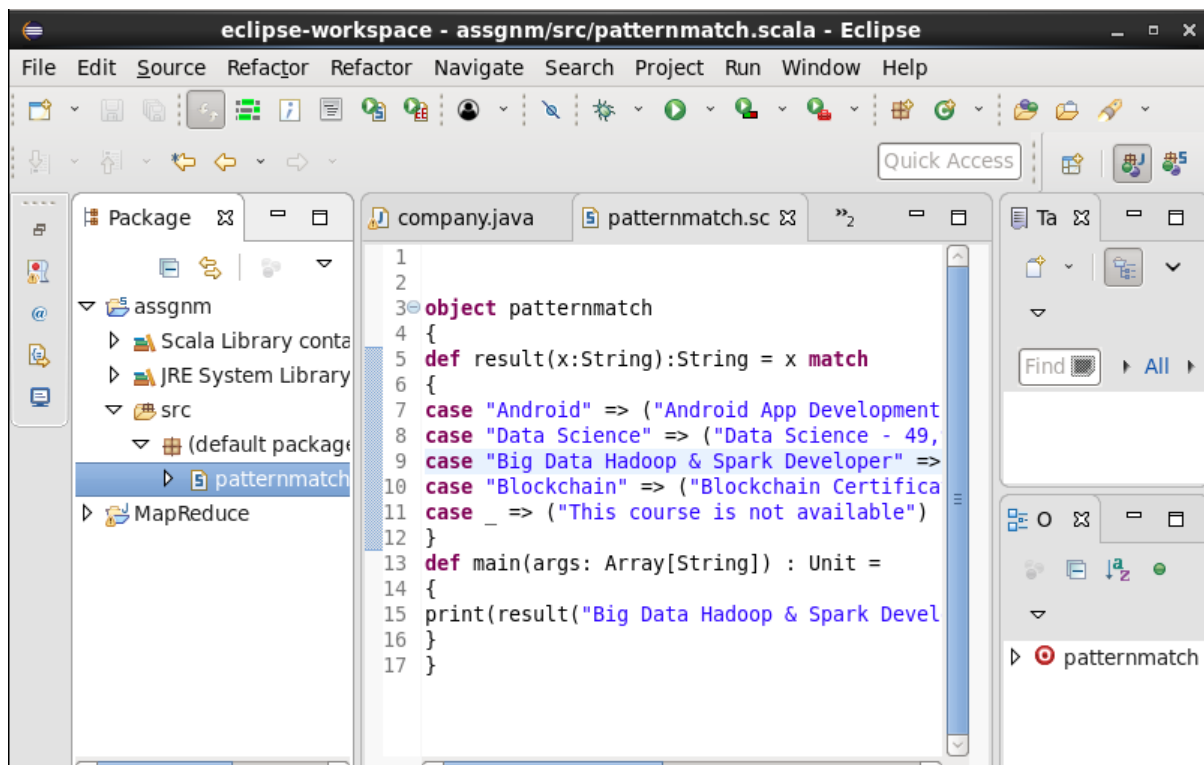
Big Data Hadoop & Spark Developer – 24,999 INR

Blockchain Certification – 49,999 INR

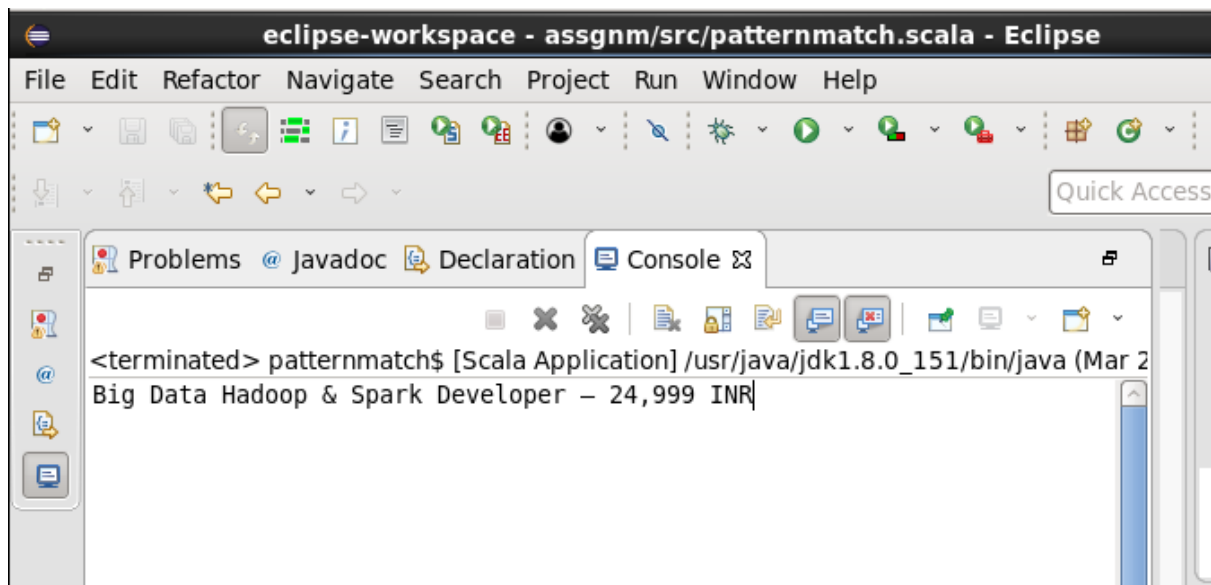
using match and add a default condition if the user enters any other course.

Input Scala Code:

```
object patternmatch
{
  def result(x:String):String = x match
  {
    case "Android" => ("Android App Development -14,999 INR")
    case "Data Science" => ("Data Science - 49,999 INR")
    case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop & Spark
    Developer – 24,999 INR")
    case "Blockchain" => ("Blockchain Certification – 49,999 INR")
    case _ => ("This course is not available")
  }
  def main(args: Array[String]) : Unit =
  {
    print(result("Big Data Hadoop & Spark Developer"))
  }
}
```

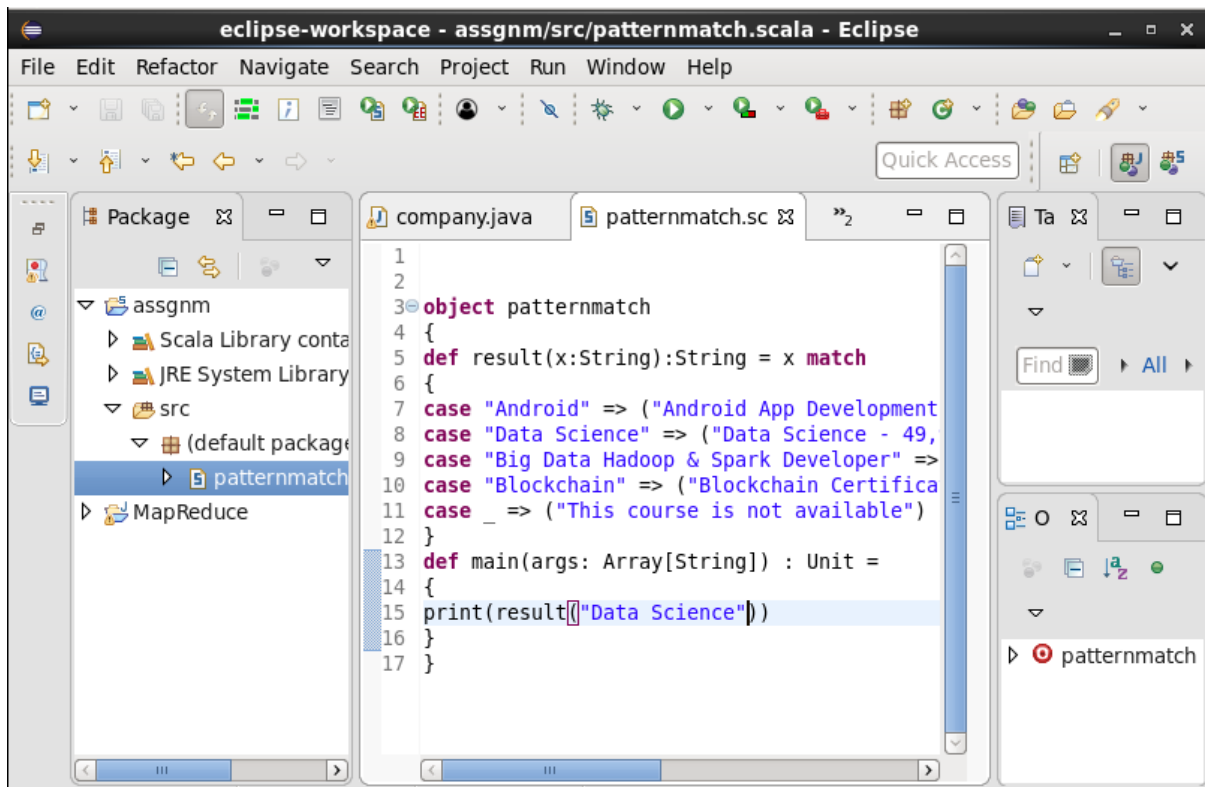


Output:

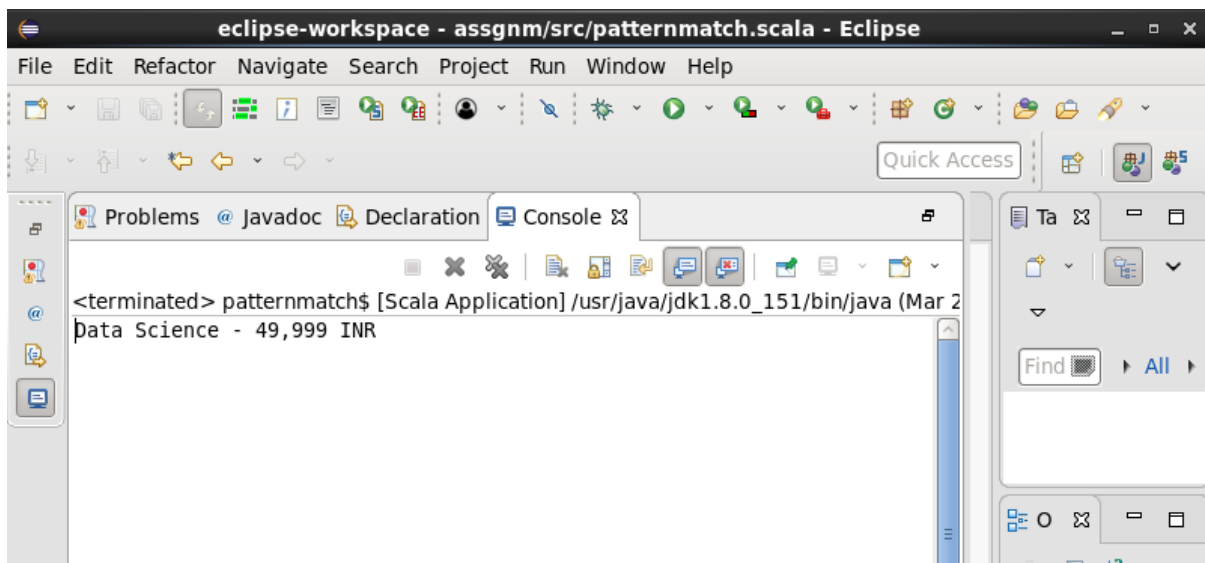


Input Scala Code:

```
object patternmatch
{
def result(x:String):String = x match
{
case "Android" => ("Android App Development -14,999 INR")
case "Data Science" => ("Data Science - 49,999 INR")
case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop & Spark
Developer – 24,999 INR")
case "Blockchain" => ("Blockchain Certification – 49,999 INR")
case _ => ("This course is not available")
}
def main(args: Array[String]) : Unit =
{
print(result ("Data Science"))
}
}
```

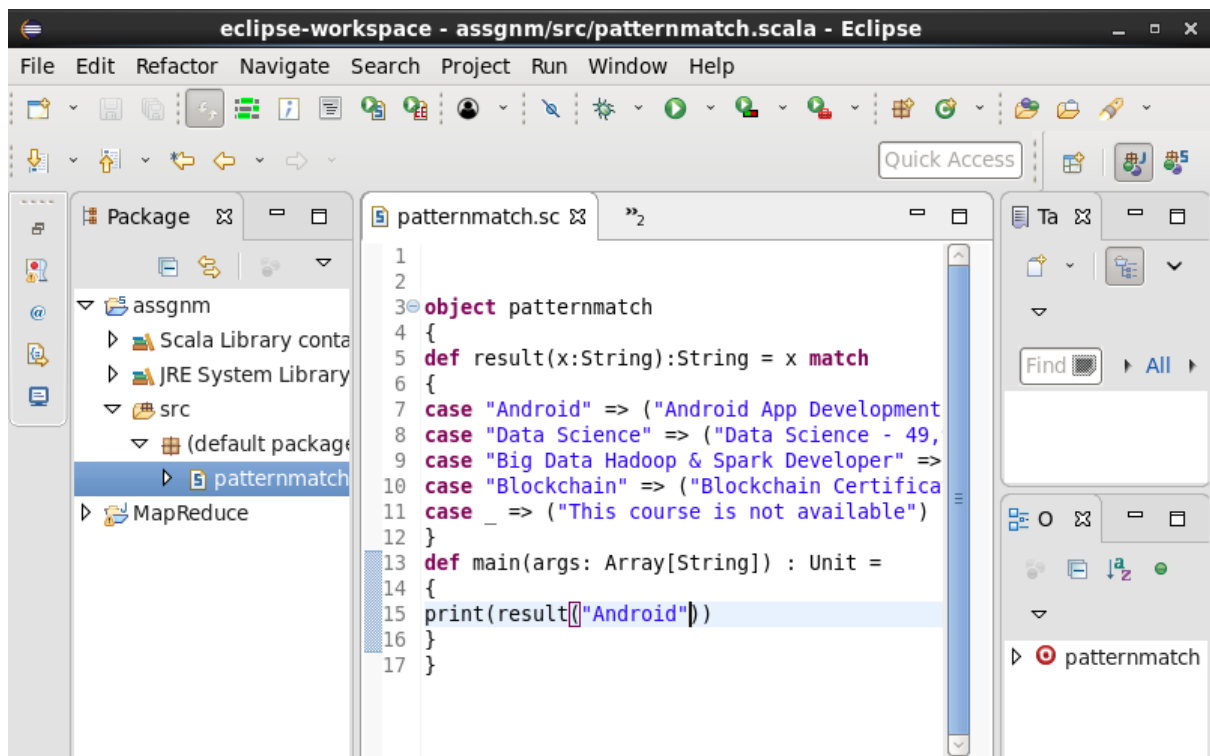


Output:



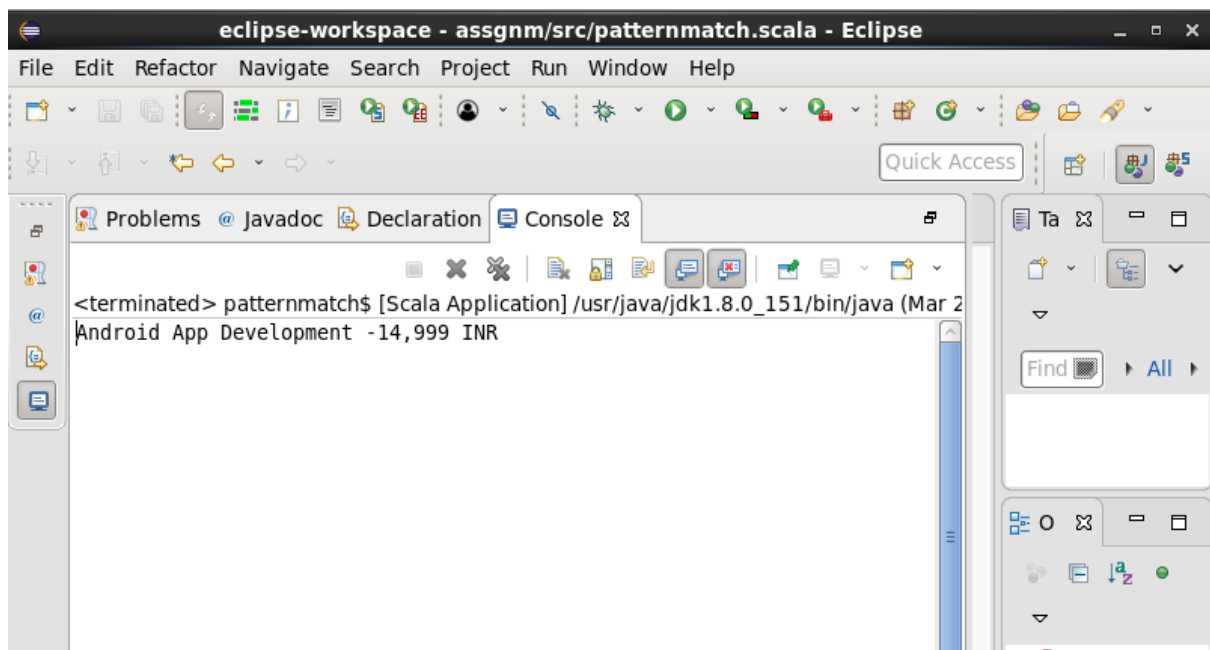
Input Scala Code:

```
object patternmatch
{
def result(x:String):String = x match
{
case "Android" => ("Android App Development -14,999 INR")
case "Data Science" => ("Data Science - 49,999 INR")
case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop & Spark
Developer – 24,999 INR")
case "Blockchain" => ("Blockchain Certification – 49,999 INR")
case _ => ("This course is not available")
}
def main(args: Array[String]) : Unit =
{
print(result ("Android"))
}
}
```



```
1
2
3 object patternmatch
4 {
5   def result(x:String):String = x match
6   {
7     case "Android" => ("Android App Development
8     case "Data Science" => ("Data Science - 49,
9     case "Big Data Hadoop & Spark Developer" =>
10    case "Blockchain" => ("Blockchain Certifica
11    case _ => ("This course is not available")
12  }
13  def main(args: Array[String]) : Unit =
14  {
15    print(result("Android"))
16  }
17 }
```

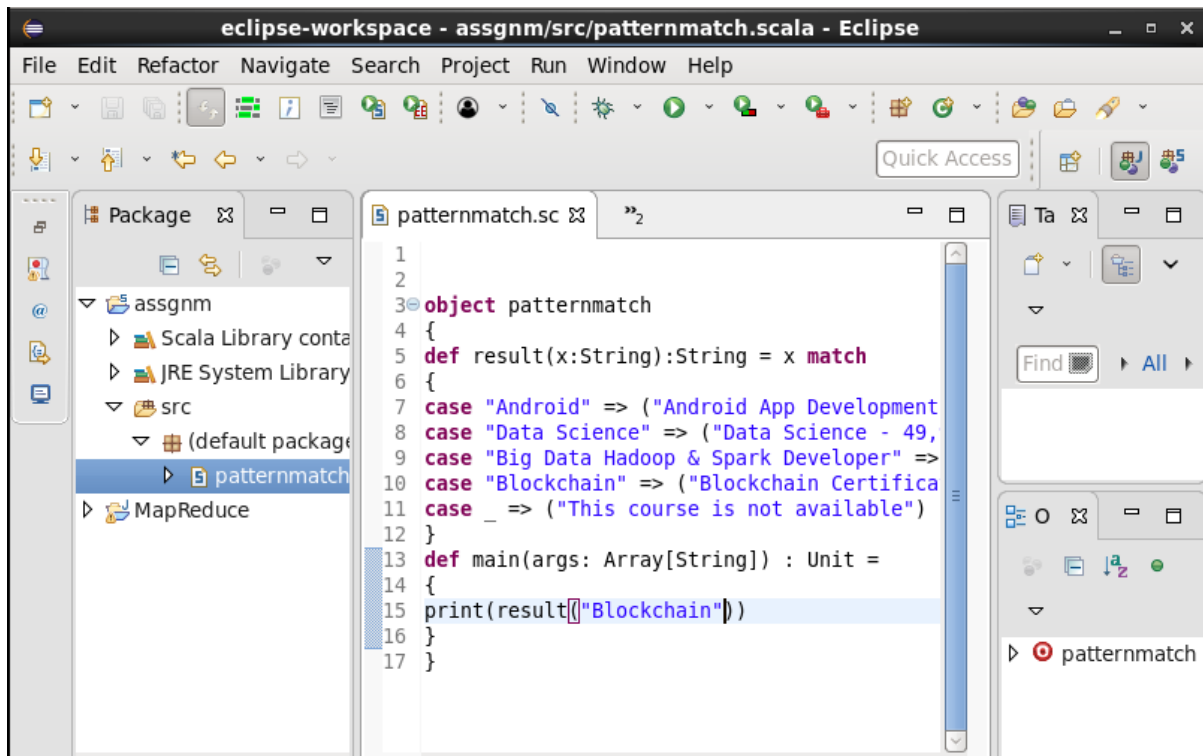
Output:



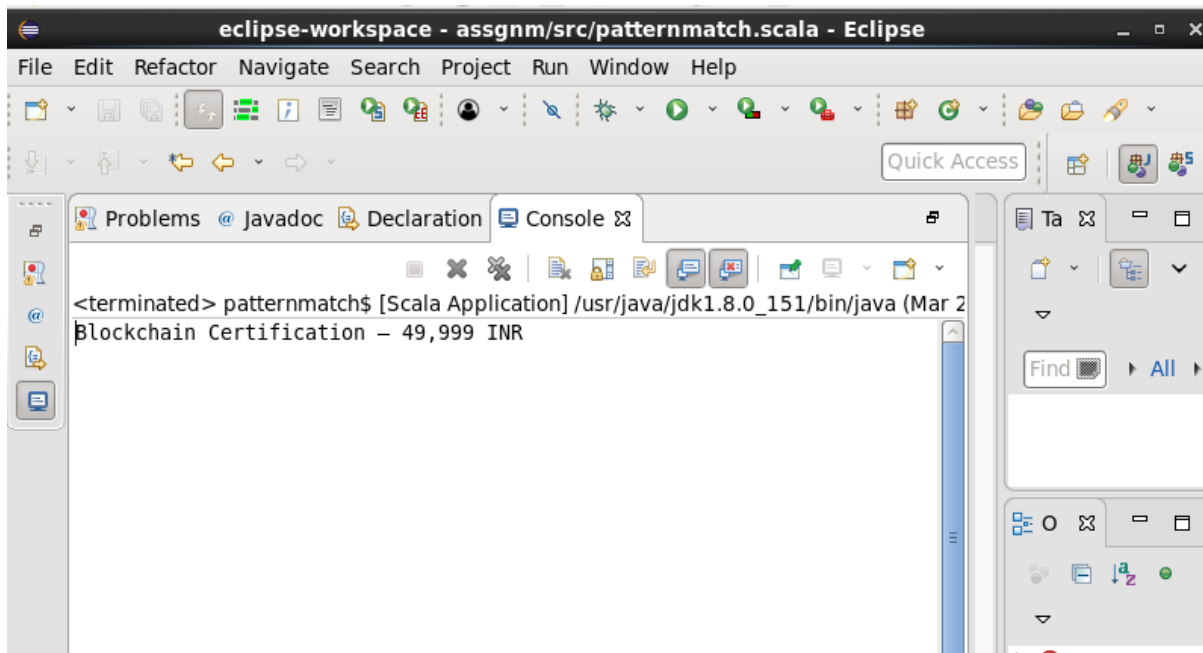
```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 2
Android App Development -14,999 INR
```

Input Scala Code:

```
object patternmatch
{
def result(x:String):String = x match
{
case "Android" => ("Android App Development -14,999 INR")
case "Data Science" => ("Data Science - 49,999 INR")
case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop & Spark
Developer – 24,999 INR")
case "Blockchain" => ("Blockchain Certification – 49,999 INR")
case _ => ("This course is not available")
}
def main(args: Array[String]) : Unit =
{
print(result ("Blockchain"))
}
}
```

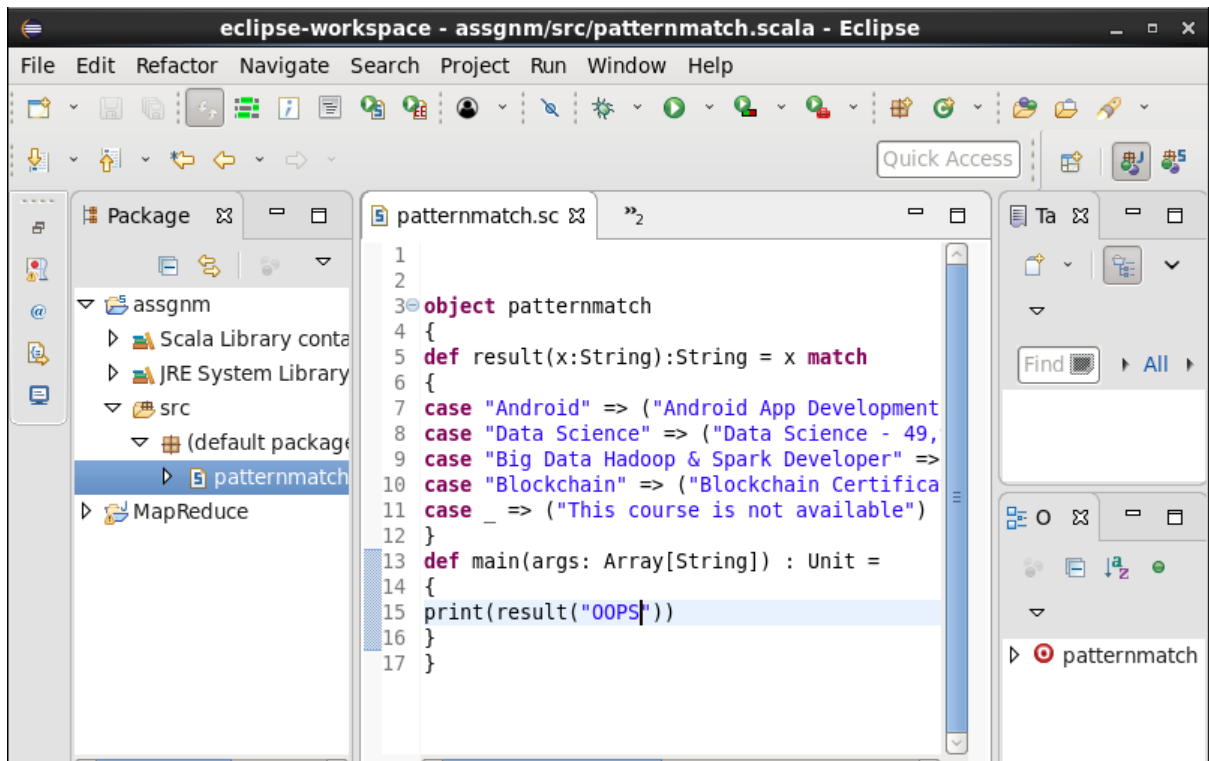


Output:



Input Scala Code:

```
object patternmatch
{
def result(x:String):String = x match
{
case "Android" => ("Android App Development -14,999 INR")
case "Data Science" => ("Data Science - 49,999 INR")
case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop & Spark
Developer – 24,999 INR")
case "Blockchain" => ("Blockchain Certification – 49,999 INR")
case _ => ("This course is not available")
}
def main(args: Array[String]) : Unit =
{
print(result ("OOPS"))
}
}
```



Output:

