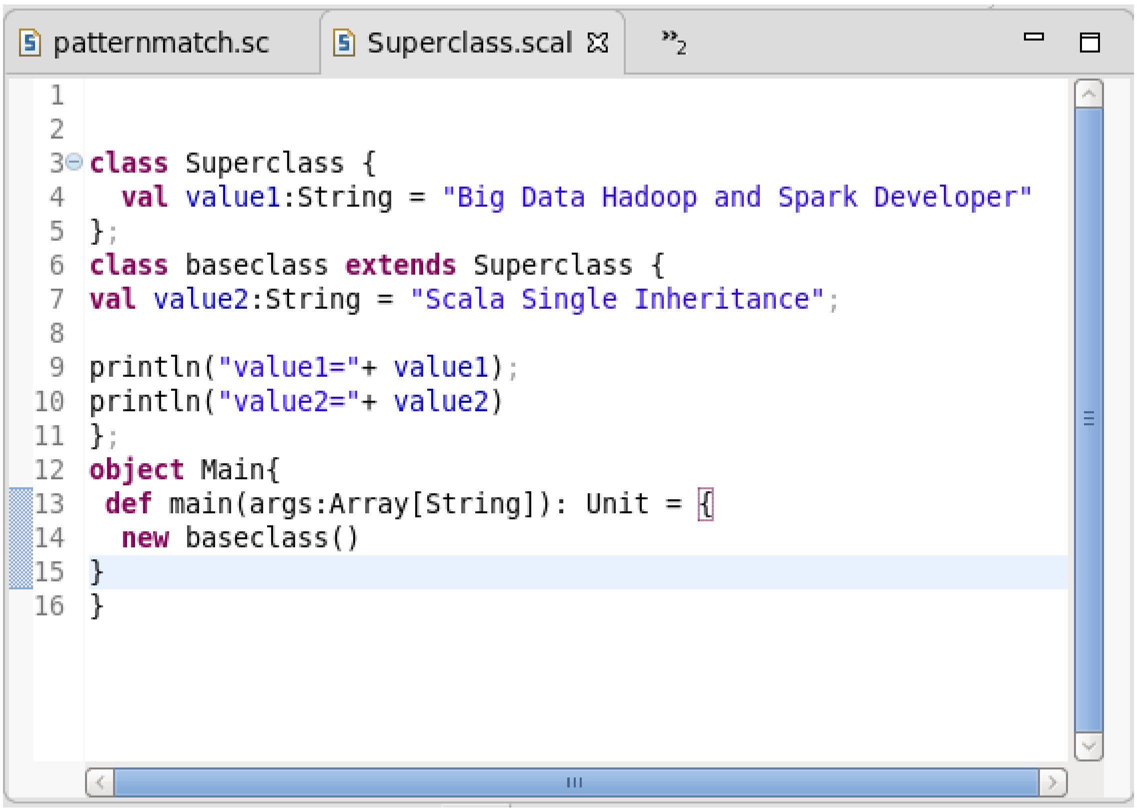
Assignment 17

Task 1

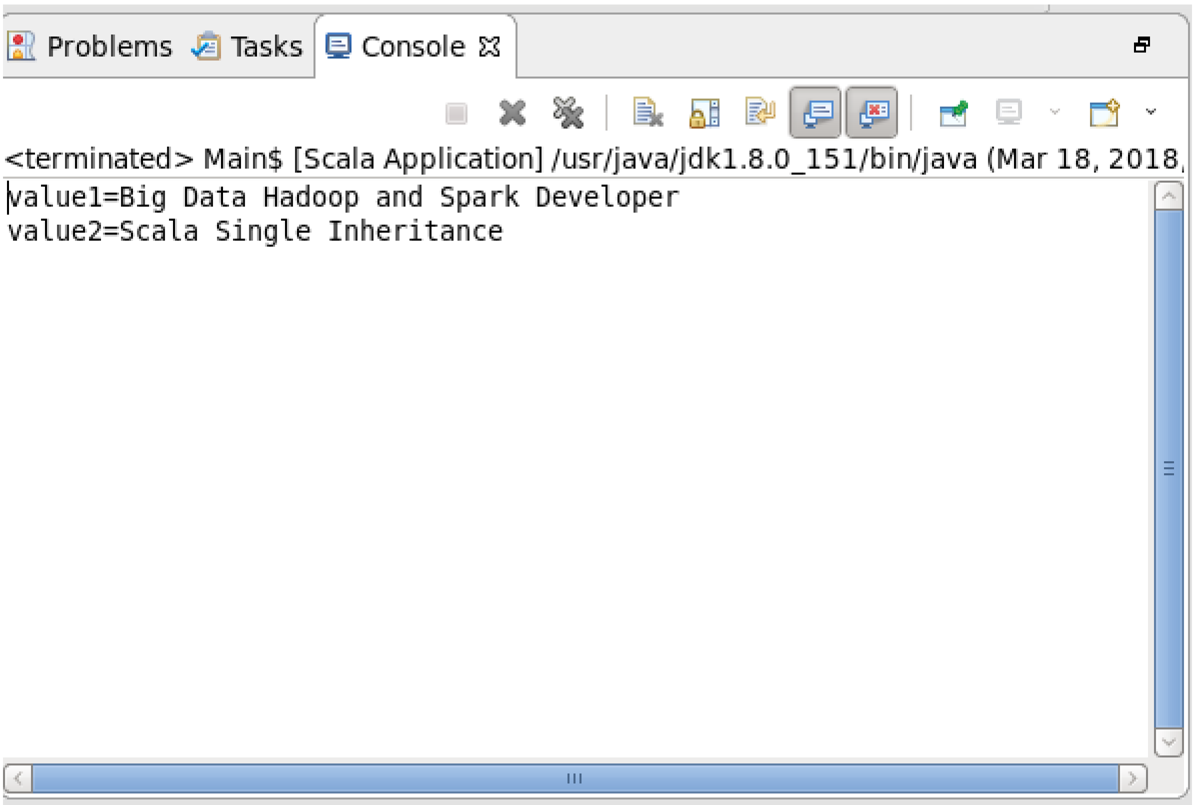
Write a simple program to show inheritance in scala.

```
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()
}
}
```



```
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



```
<terminated> Main$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 18, 2018,
value1=Big Data Hadoop and Spark Developer
value2=Scala Single Inheritance
```

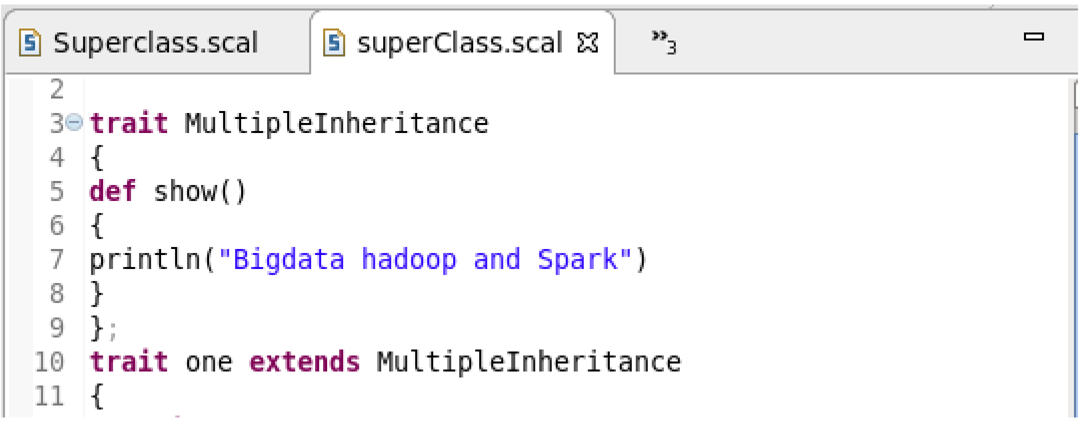
Task 2

Write a simple program to show multiple inheritance in scala

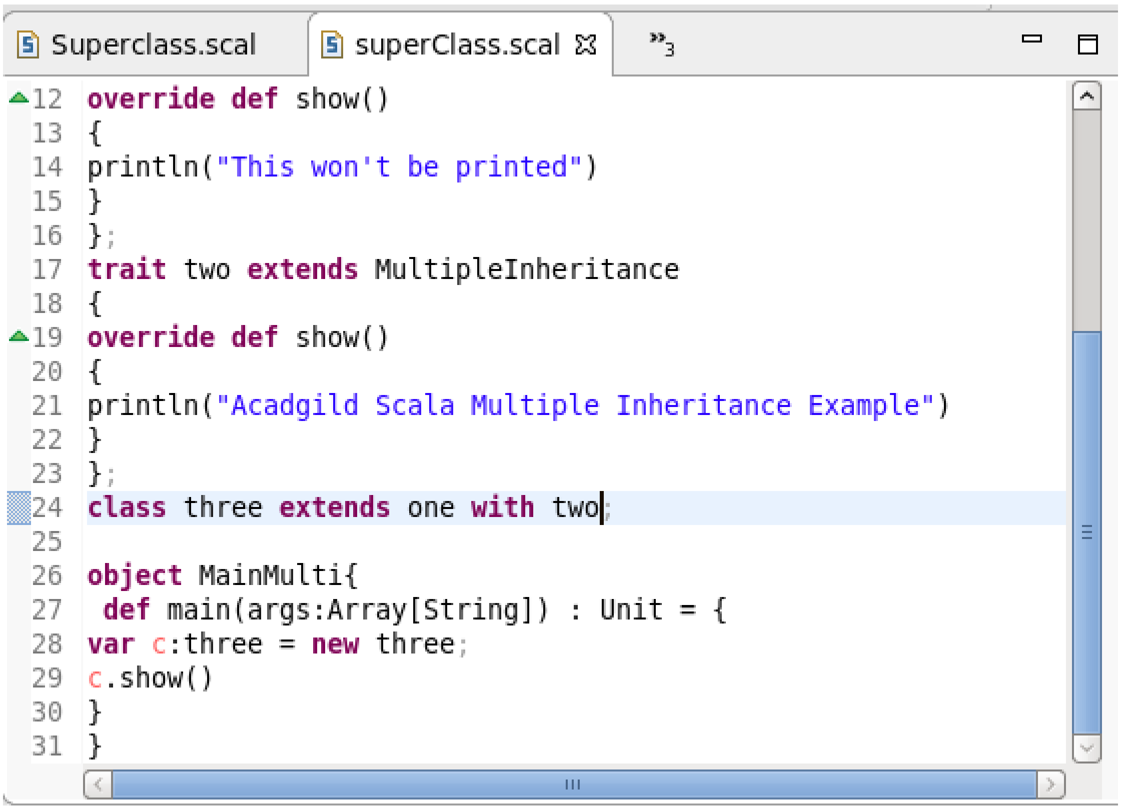
- **Multiple Inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and feature from more than one parent object or parent class**
- **Scala supports various types of inheritance including single,multilevel,multiple and hybrid.**
- **Multiple and hybrid can only be achieved by using traits**
- **Scala doesn't allow for multiple inheritance, but allows to extend multiple traits.**
- **A trait is like an interface with a partial implementation.**
- **In scala, trait is a collection of abstract and non-abstract methods.**
- **You can create trait that can have all abstract methods or some abstract and some non-abstract methods.**

```
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()
}
}
```

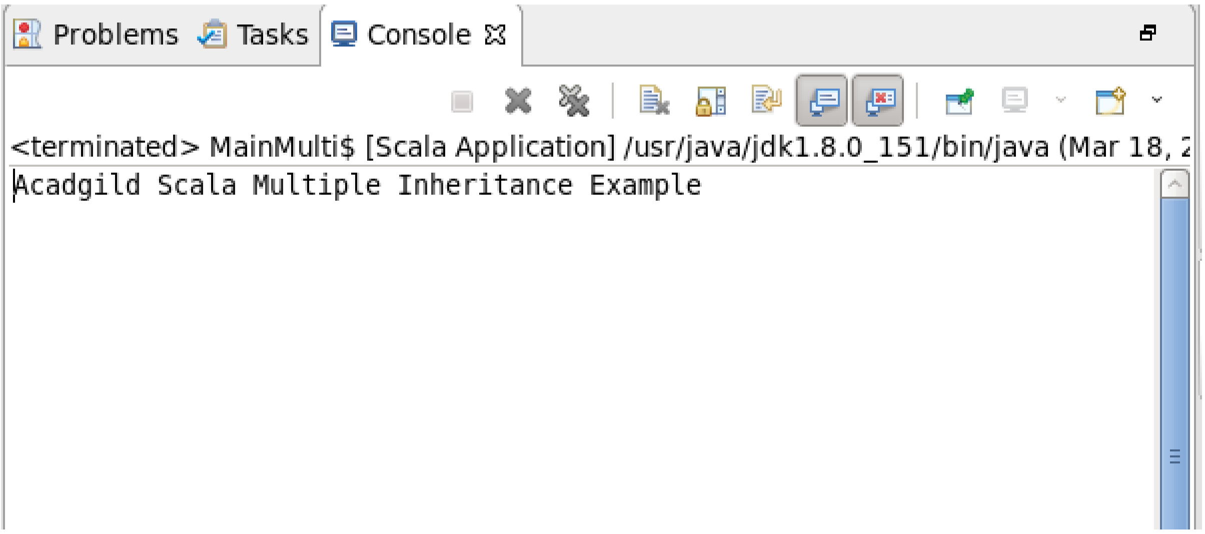
Example 1



```

trait MultipleInheritance
{
def show()
{
println("Bigdata hadoop and Spark")
}
};
trait one extends MultipleInheritance
{
```

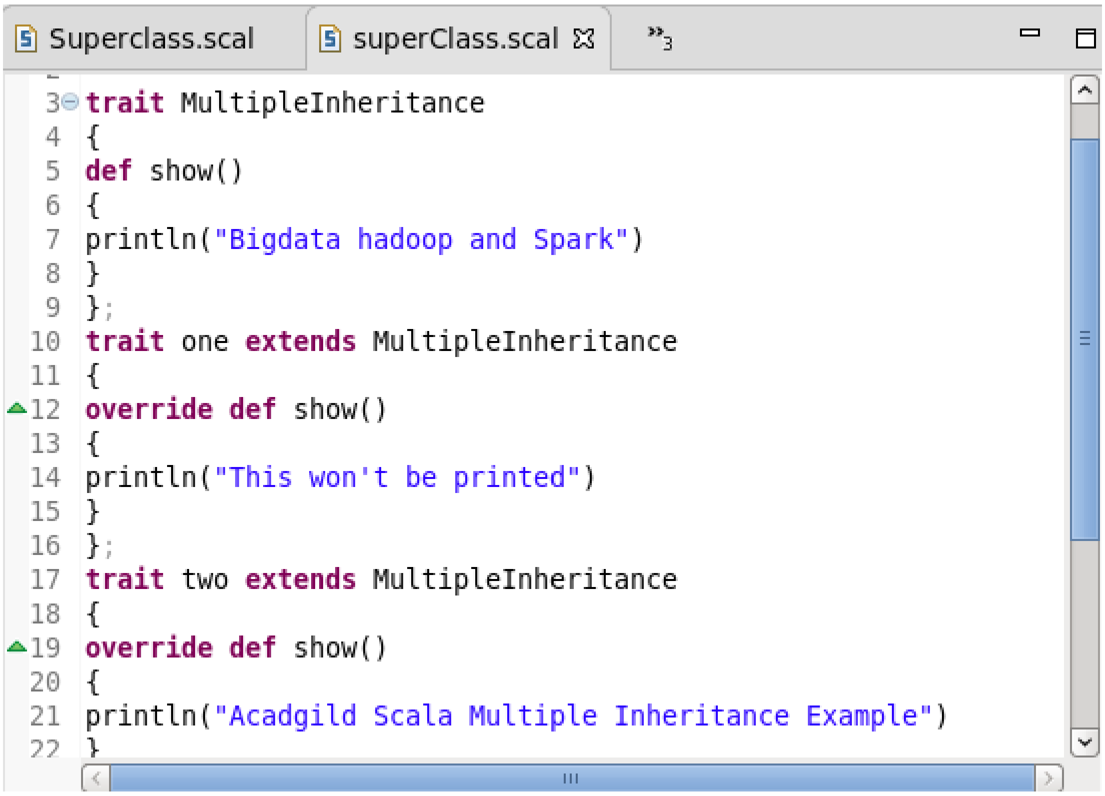


```
override def show()
{
println("This won't be printed")
}
};
trait two extends MultipleInheritance
{
override def show()
{
println("Acadgild Scala Multiple Inheritance Example")
}
};
class three extends one with two;

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

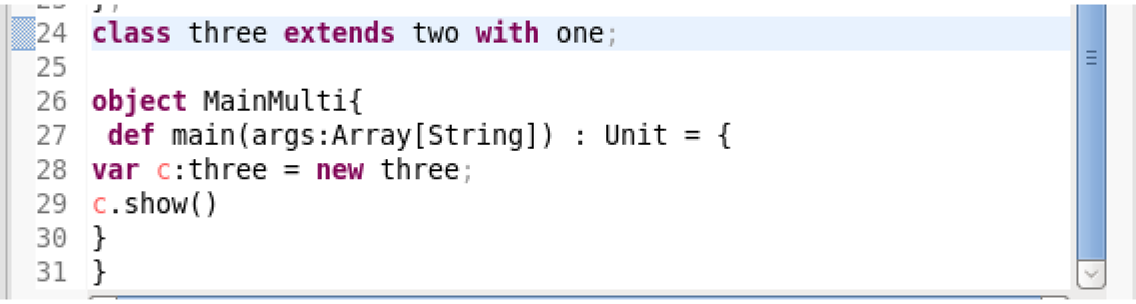


```
<terminated> MainMulti$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 18, 2
Acadgild Scala Multiple Inheritance Example
```

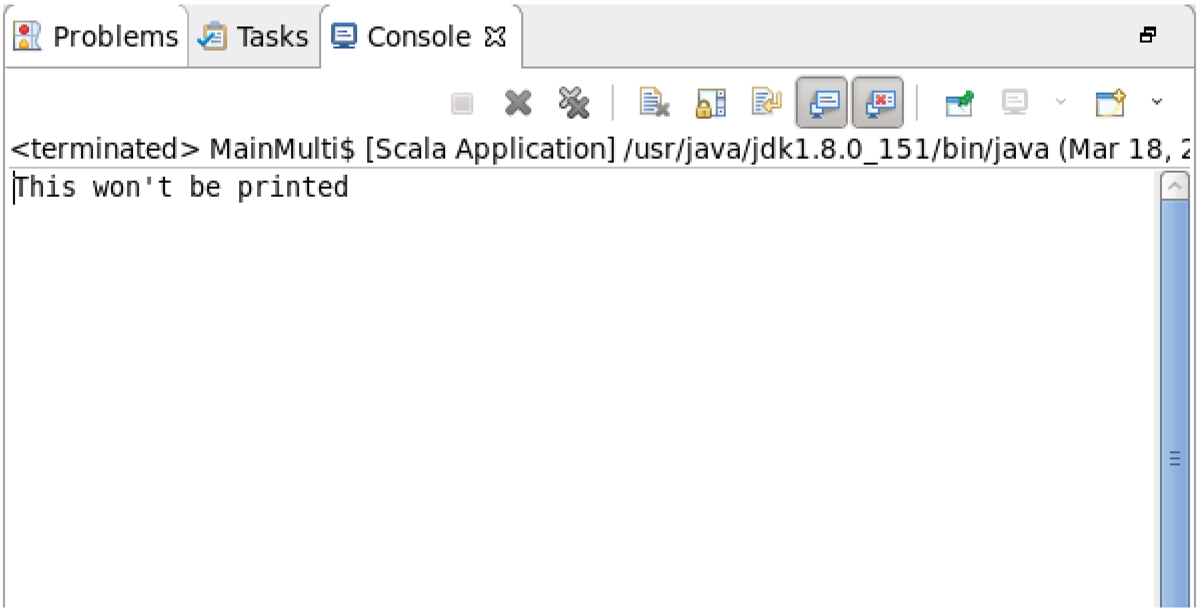
Example 2



```
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 two with one;

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

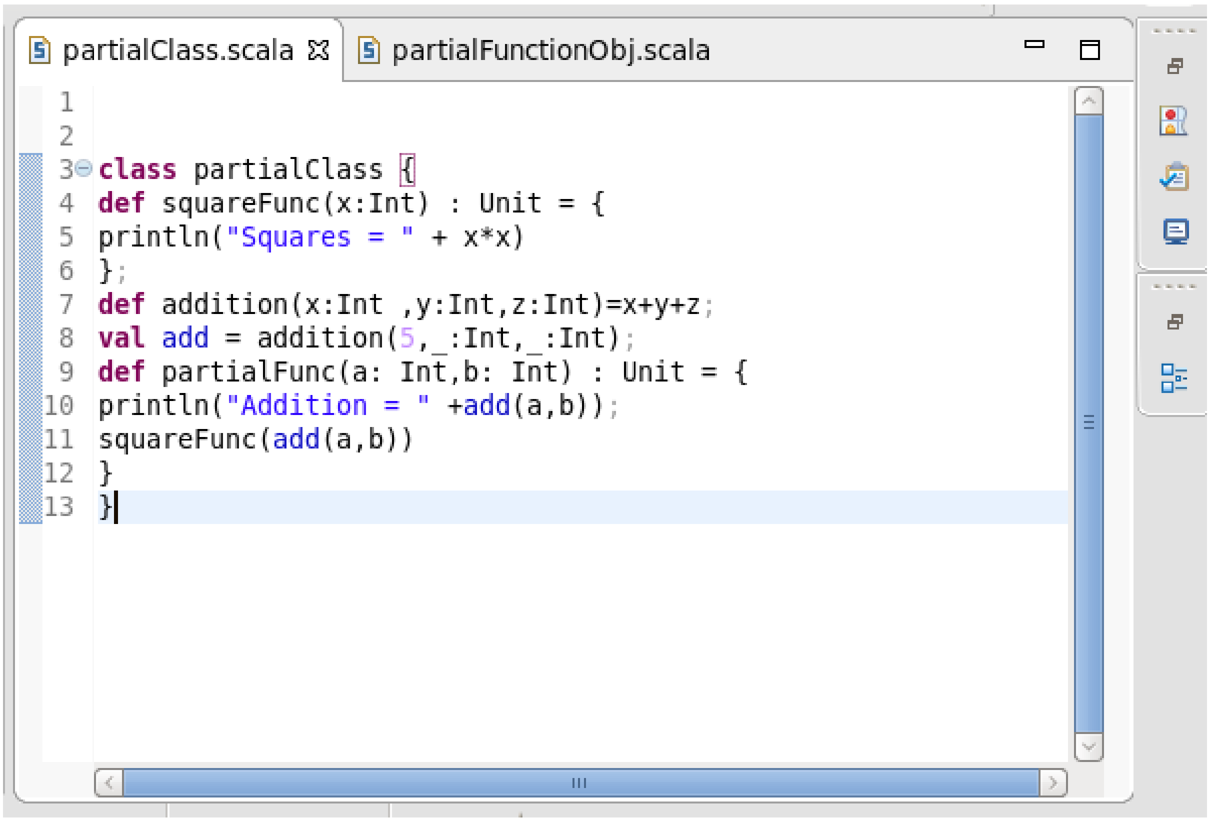


```
<terminated> MainMulti$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 18, 2
This won't be printed
```

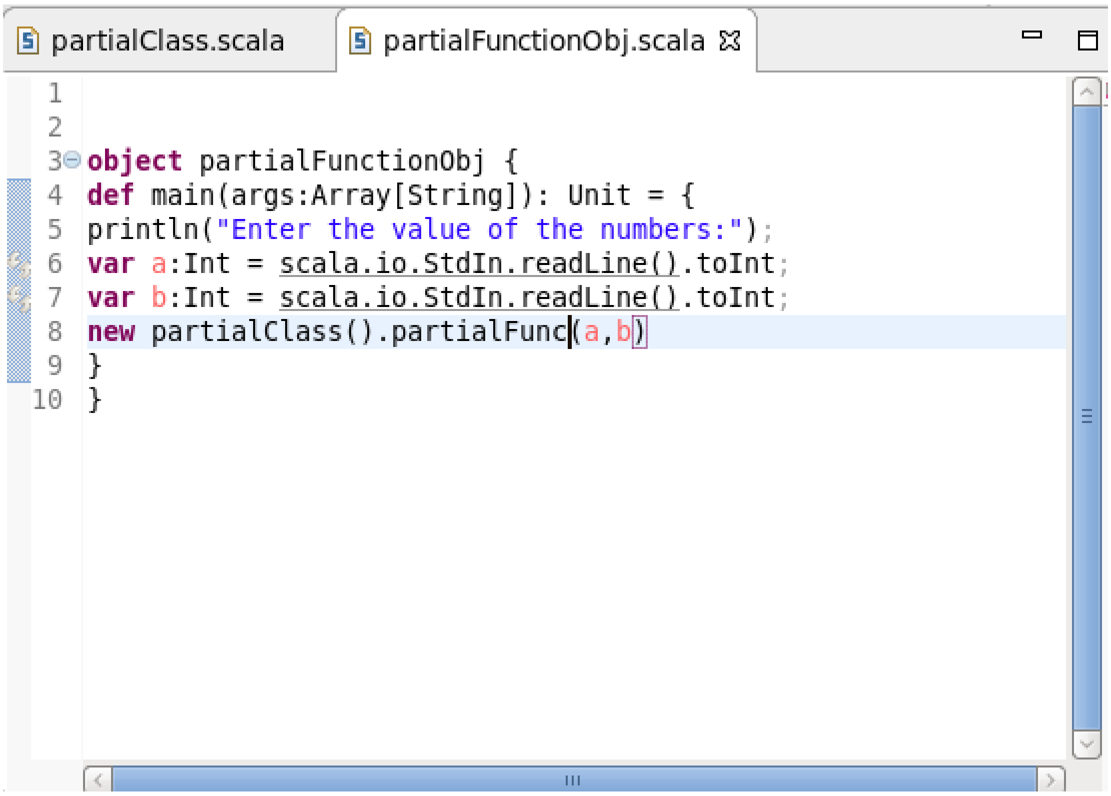
Task 3

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

```
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))
}
}
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)
}
}
```

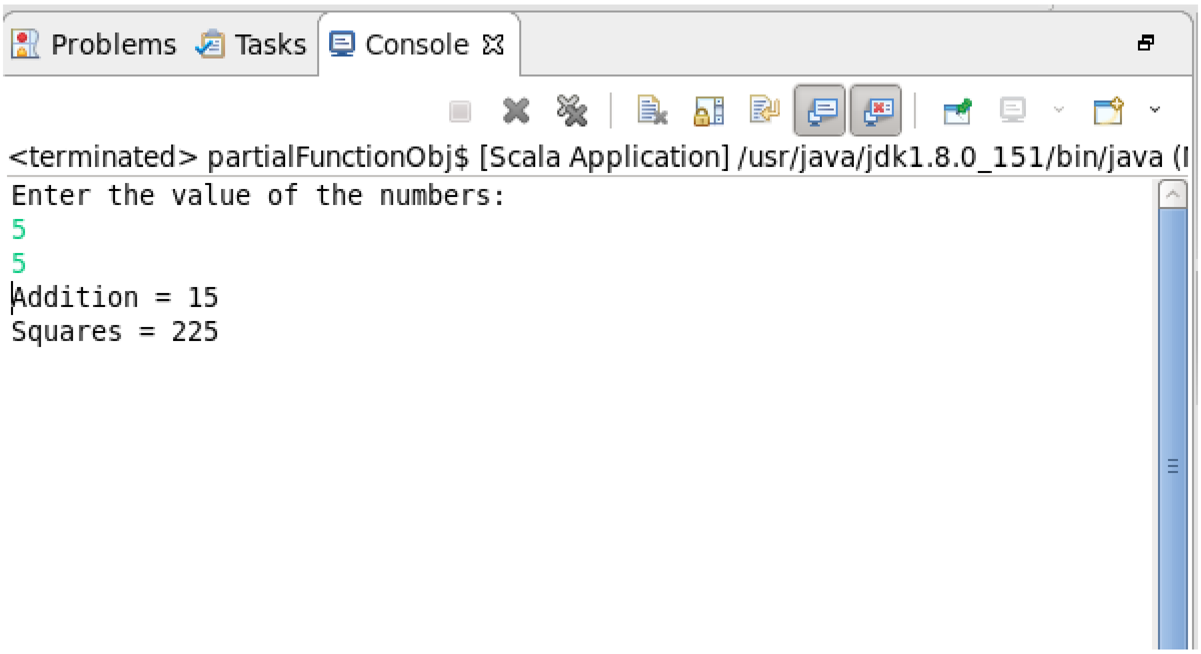


```
class partialClass {
def squareFunc(x:Int) : Unit = {
println("Squares = " + x*x)
};
def addition(x:Int ,y:Int,z:Int)=x+y+z;
val add = addition(5,_:Int,_:Int);
def partialFunc(a: Int,b: Int) : Unit = {
println("Addition = " +add(a,b));
squareFunc(add(a,b))
}
}
```



```
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



```
<terminated> partialFunctionObj$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (I
Enter the value of the numbers:
5
5
Addition = 15
Squares = 225
```

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

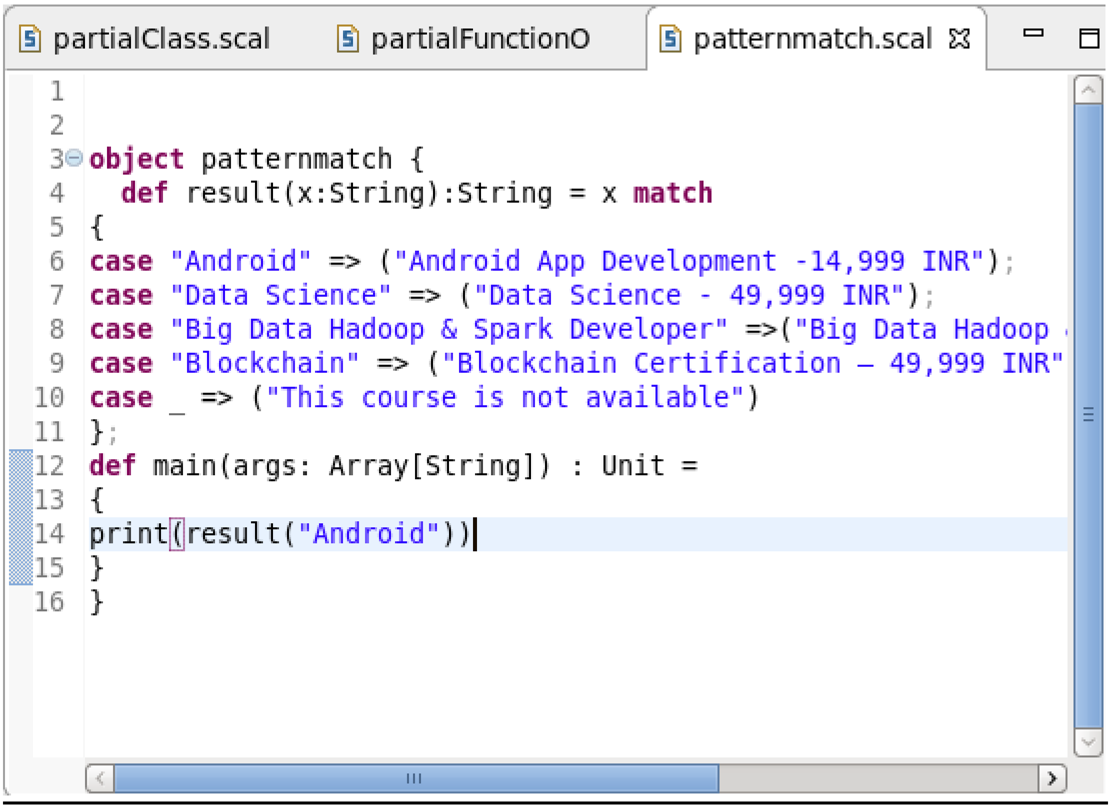
Big Data Hadoop & Spark Developer – 24,999 INR

Blockchain Certification – 49,999 INR

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

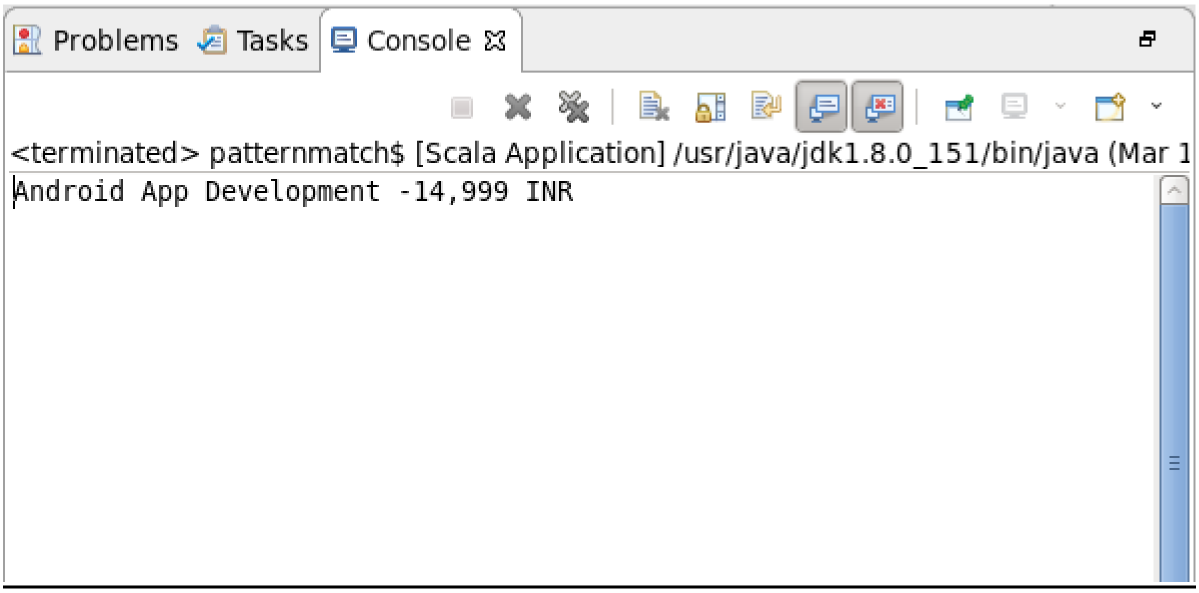
```
object patternmatch

{

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"))
}
}
```

<u>Example 1</u> :- print(result("Android"))



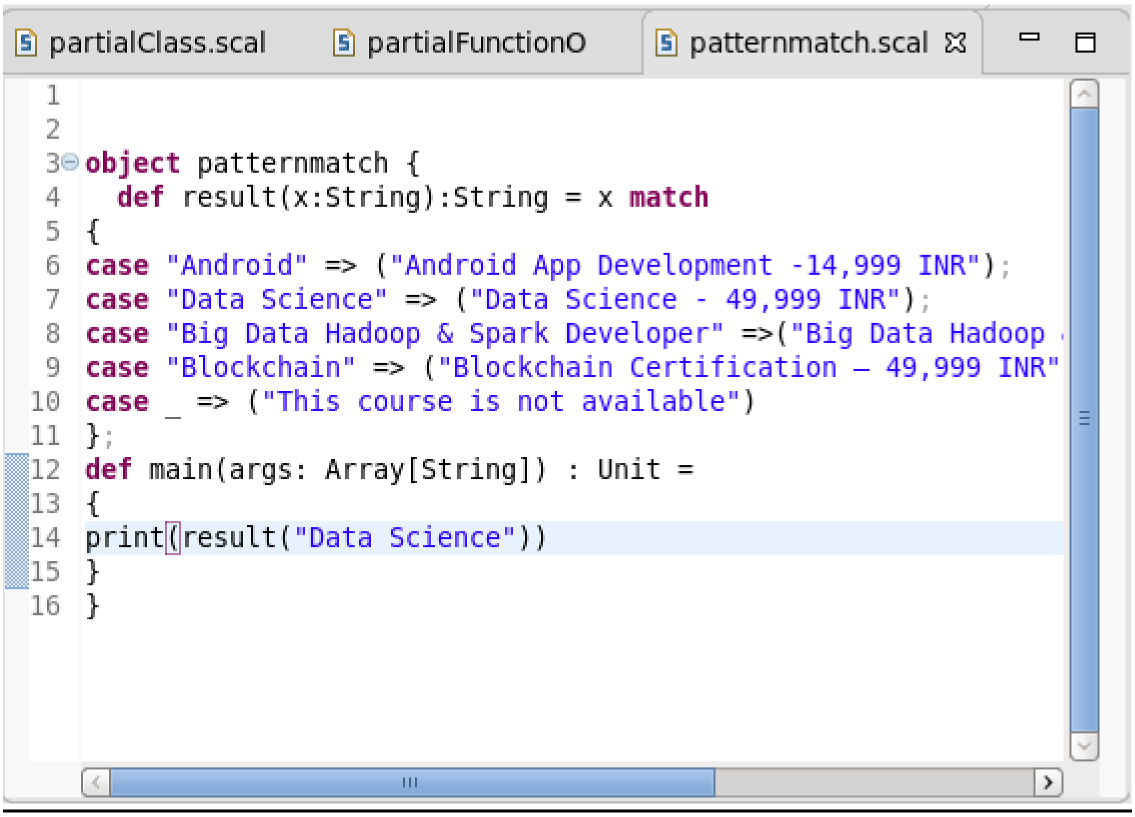
```


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
case "Blockchain" => ("Blockchain Certification – 49,999 INR"
case _ => ("This course is not available")
};
def main(args: Array[String]) : Unit =
{
print(result("Android"))
}
}
```

Output



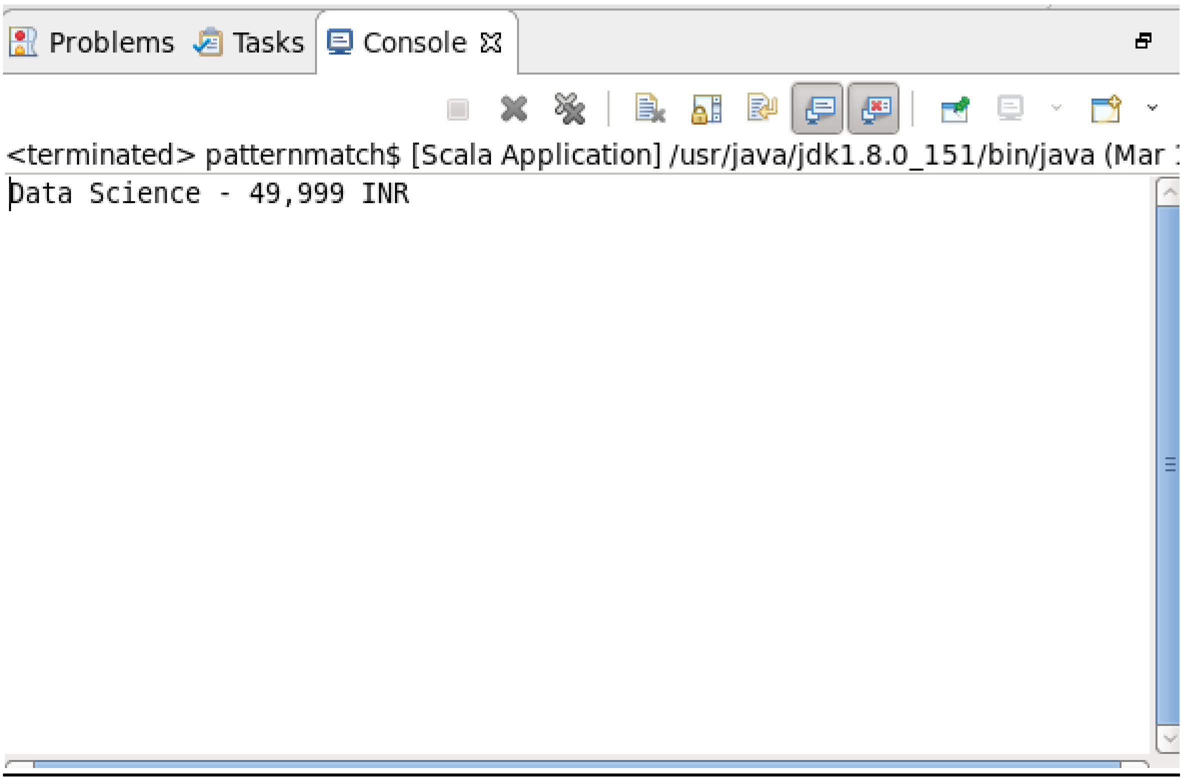
```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 1
Android App Development -14,999 INR
```

Example 2:- print(result(“Data Science”))



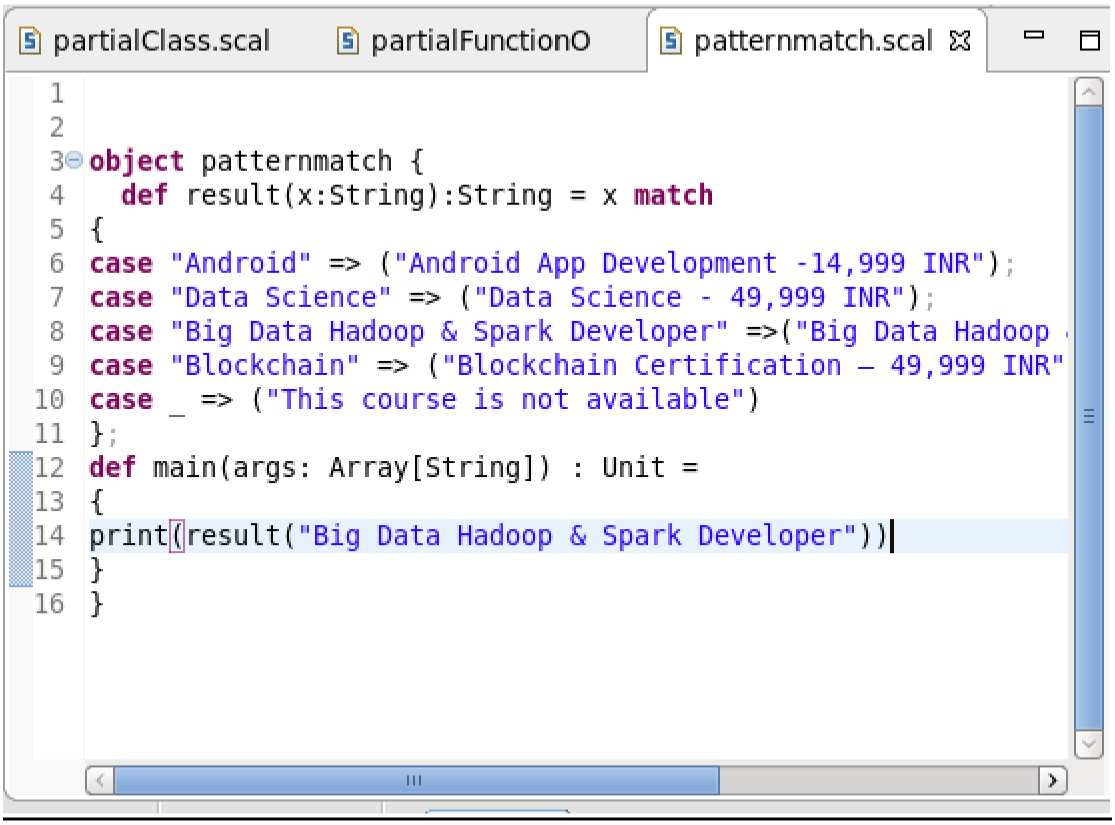
```


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
case "Blockchain" => ("Blockchain Certification – 49,999 INR"
case _ => ("This course is not available")
};
def main(args: Array[String]) : Unit =
{
print(result("Data Science"))
}
}
```

Output



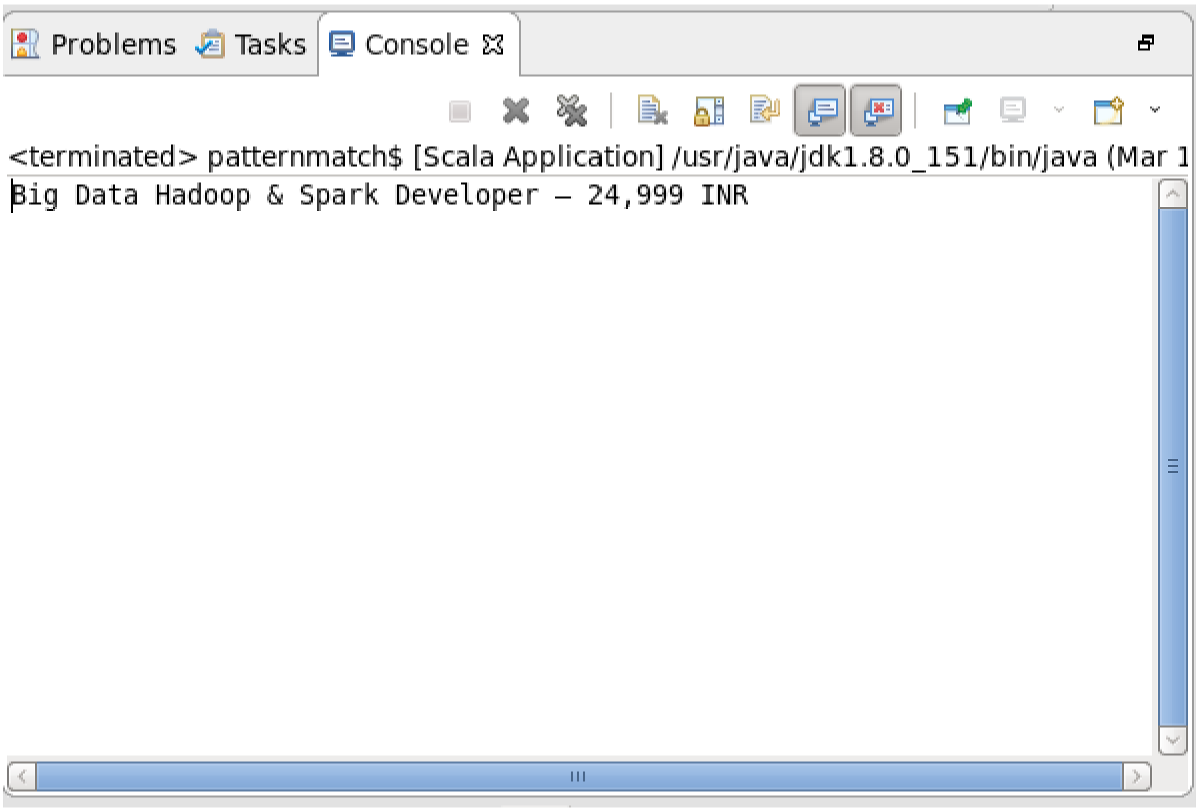
```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar
Data Science - 49,999 INR
```

Example 3:- print(result("Big Data Hadoop and Spark Developer"))



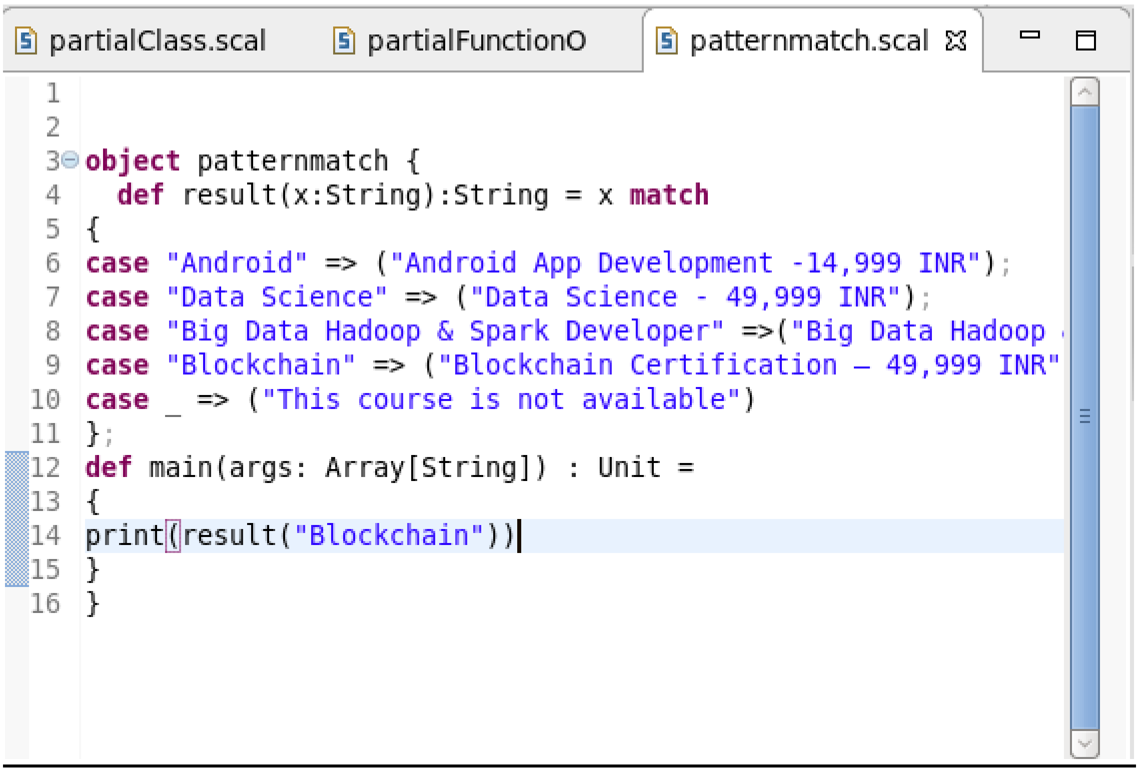
```


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
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



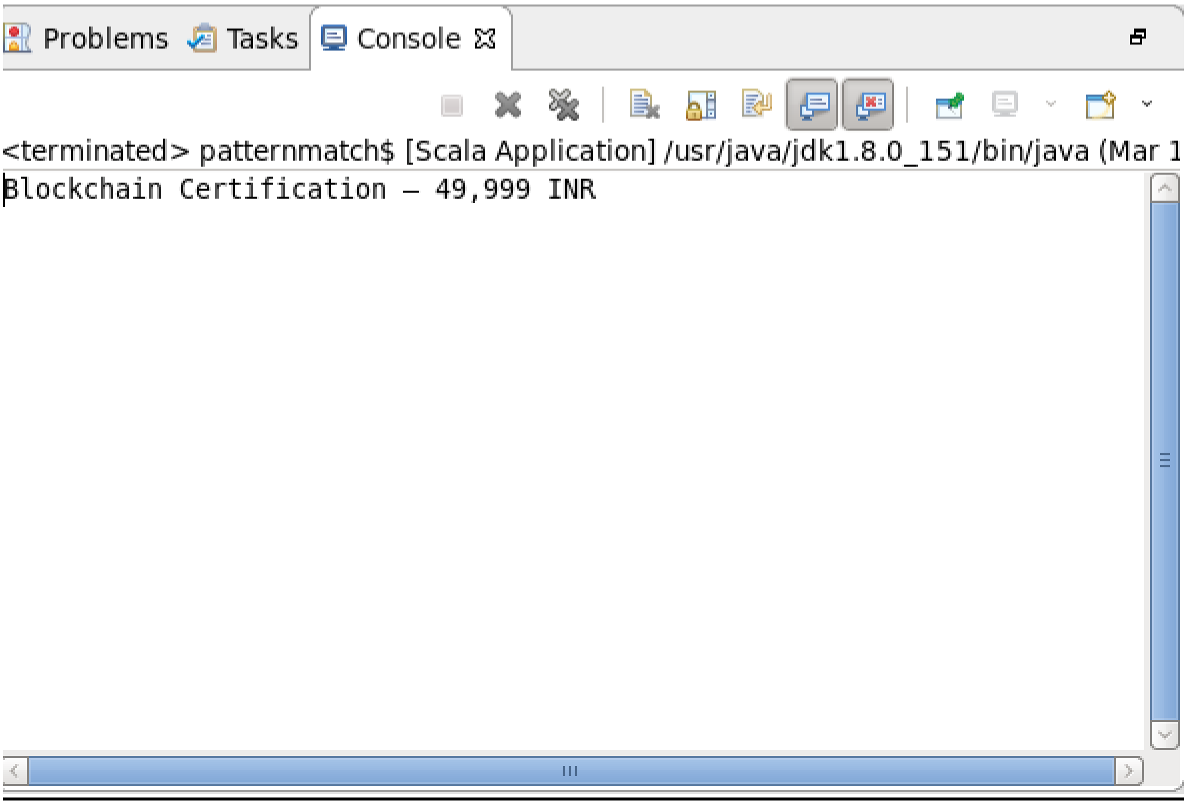
```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 1
Big Data Hadoop & Spark Developer – 24,999 INR
```

Example 4:- print(result(“Blockchain”))



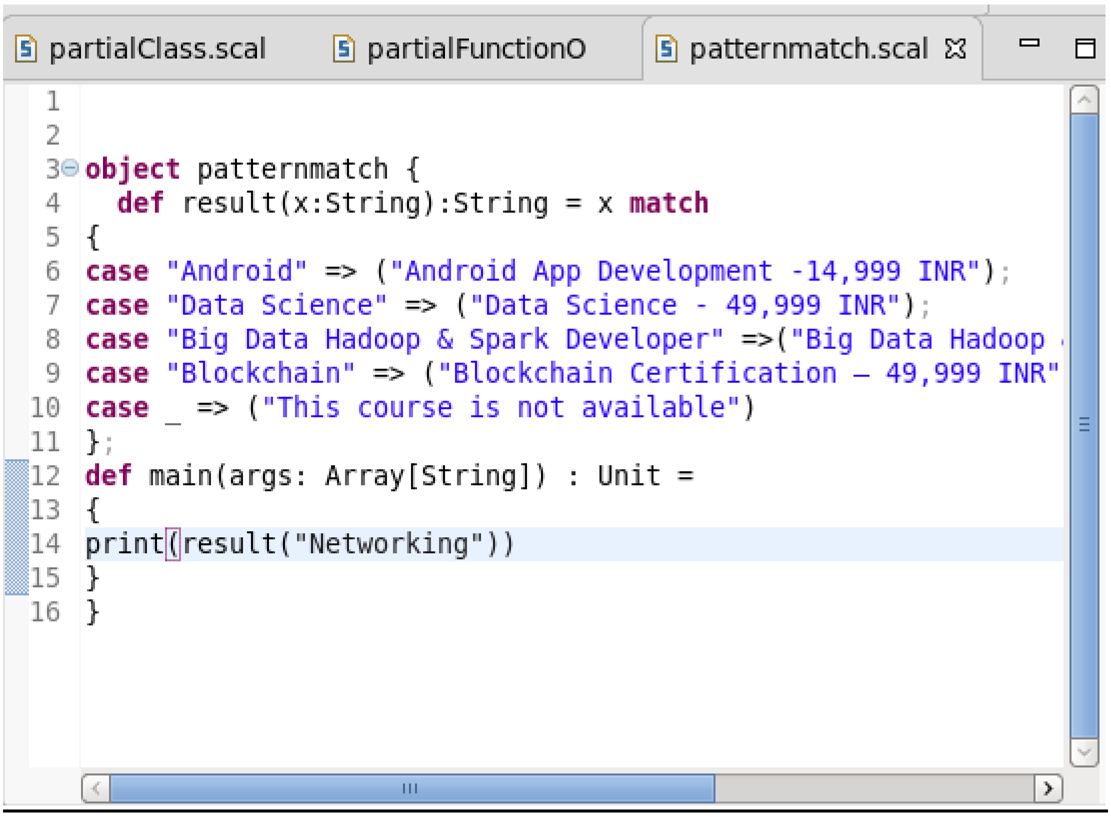
```


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
case "Blockchain" => ("Blockchain Certification – 49,999 INR"
case _ => ("This course is not available")
};
def main(args: Array[String]) : Unit =
{
print(result("Blockchain"))
}
}
```

Output



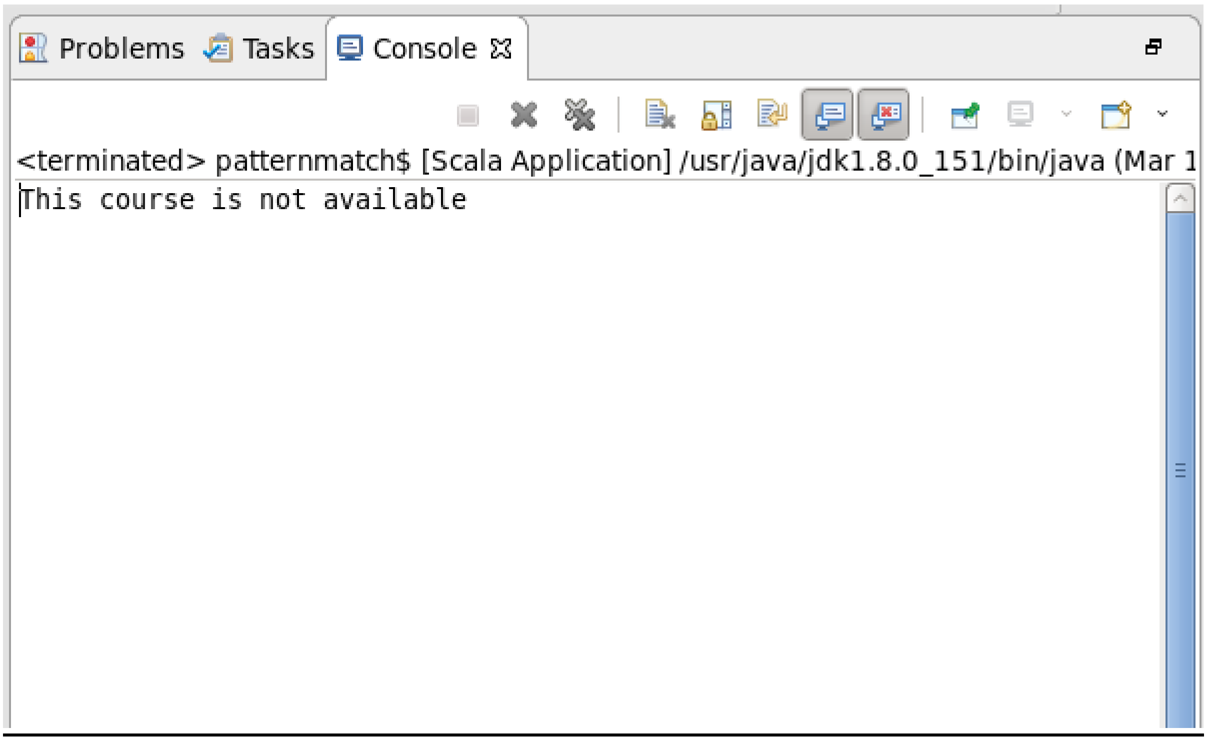
```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 1
Blockchain Certification – 49,999 INR
```

Example 5:- print(result(“Networking”))



```


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
case "Blockchain" => ("Blockchain Certification – 49,999 INR"
case _ => ("This course is not available")
};
def main(args: Array[String]) : Unit =
{
print(result("Networking"))
}
}
```

Output



```
<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 1
This course is not available
```