

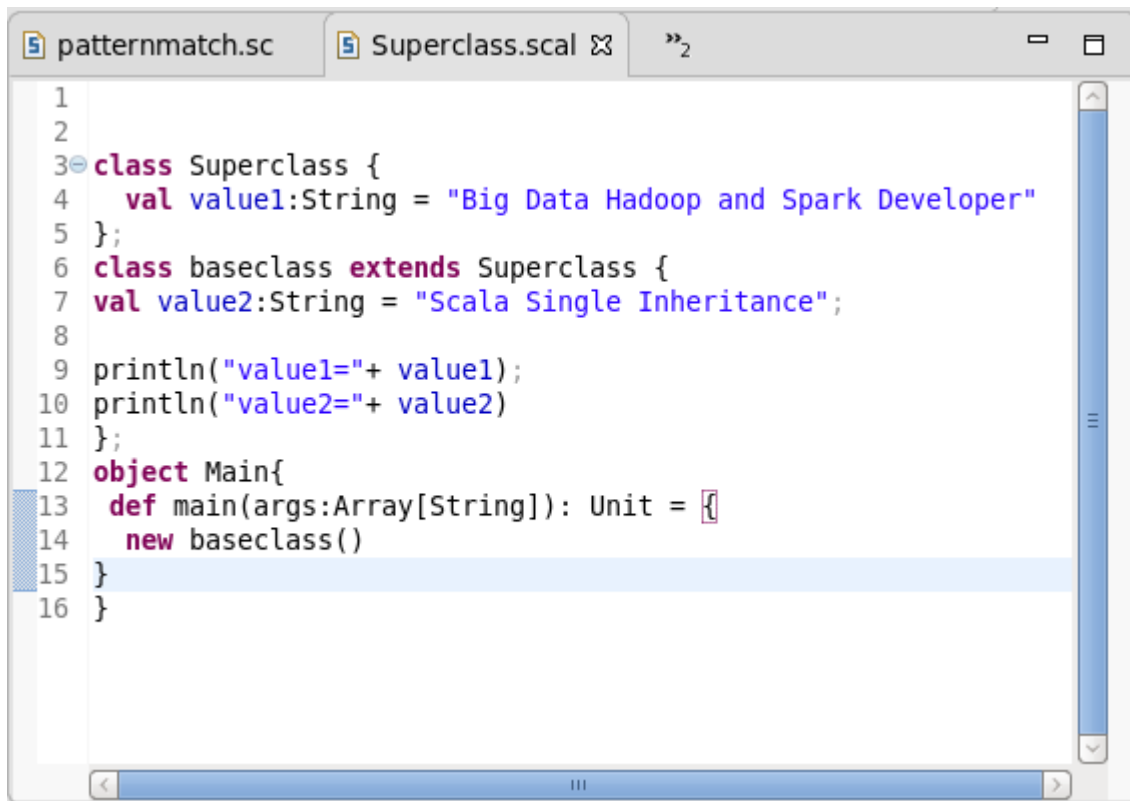
Assignment 17

Task 1

Write a simple program to show inheritance in scala.

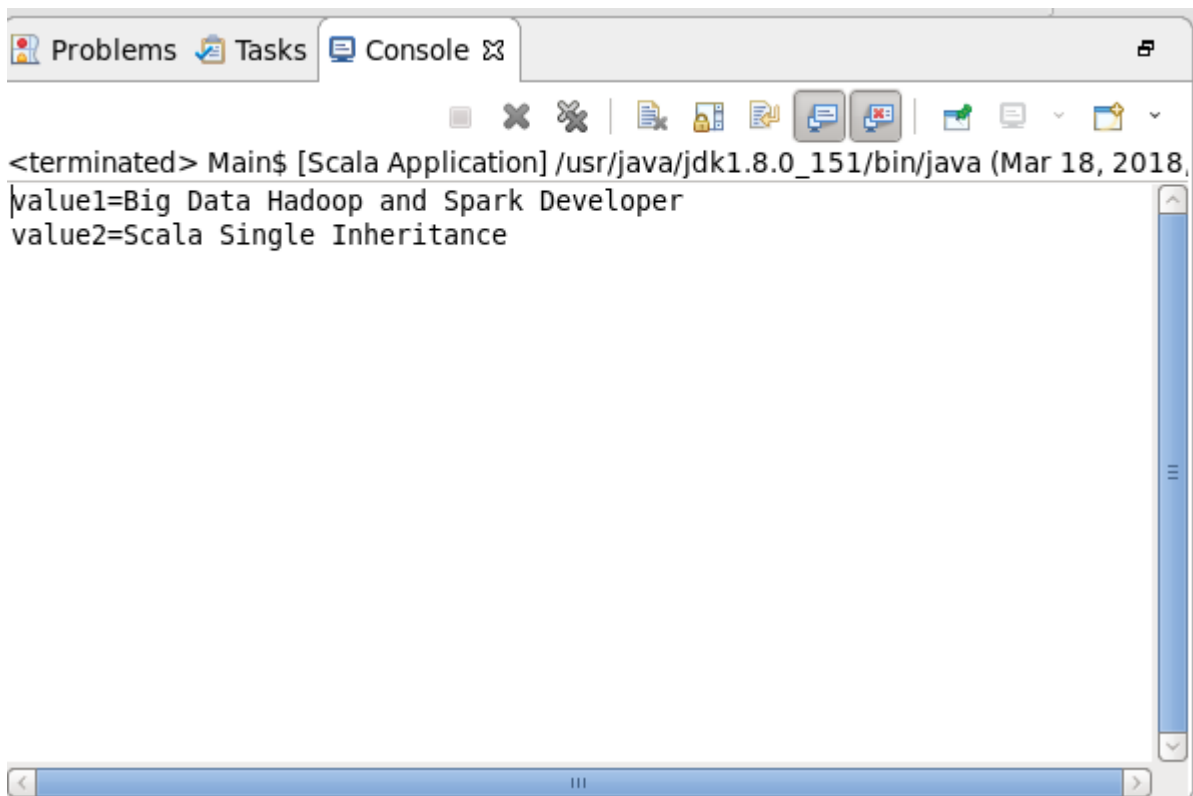
```
class Superclass
{
val value1:String = "Big Data Hadoop and Spark Developer"
}
class baseclass extends Superclass {
val value2:String = "Scala Single Inheritance"

println("value1="+ value1)
println("value2="+ value2)
}
object Main{
def main(args:Array[String]): Unit = {
new baseclass()
}
}
```



```
1
2
3 class Superclass {
4   val value1:String = "Big Data Hadoop and Spark Developer"
5 };
6 class baseclass extends Superclass {
7   val value2:String = "Scala Single Inheritance";
8
9   println("value1="+ value1);
10  println("value2="+ value2)
11 };
12 object Main{
13   def main(args:Array[String]): Unit = {
14     new baseclass()
15   }
16 }
```

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

```
Superclass.scala  superClass.scala  »_3
2
3 trait MultipleInheritance
4 {
5   def show()
6   {
7     println("Bigdata hadoop and Spark")
8   }
9 };
10 trait one extends MultipleInheritance
11 {
```

```
Superclass.scala  superClass.scala  »_3
12 override def show()
13 {
14   println("This won't be printed")
15 }
16 };
17 trait two extends MultipleInheritance
18 {
19 override def show()
20 {
21   println("Acadgild Scala Multiple Inheritance Example")
22 }
23 };
24 class three extends one with two;
25
26 object MainMulti{
27   def main(args:Array[String]) : Unit = {
28     var c:three = new three;
29     c.show()
30   }
31 }
```

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

Example 2

```
Superclass.scala  superClass.scala  »_3
3 trait MultipleInheritance
4 {
5   def show()
6   {
7     println("Bigdata hadoop and Spark")
8   }
9 };
10 trait one extends MultipleInheritance
11 {
12   override def show()
13   {
14     println("This won't be printed")
15   }
16 };
17 trait two extends MultipleInheritance
18 {
19   override def show()
20   {
21     println("Acadgild Scala Multiple Inheritance Example")
22   }
23 }
```

```
24 class three extends two with one;
25
26 object MainMulti{
27   def main(args:Array[String]) : Unit = {
28     var c:three = new three;
29     c.show()
30   }
31 }
```

```
Problems  Tasks  Console  »
<terminated> MainMulti$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 18, 2016 10:10:10 AM)
This won't be printed
```

Task 3

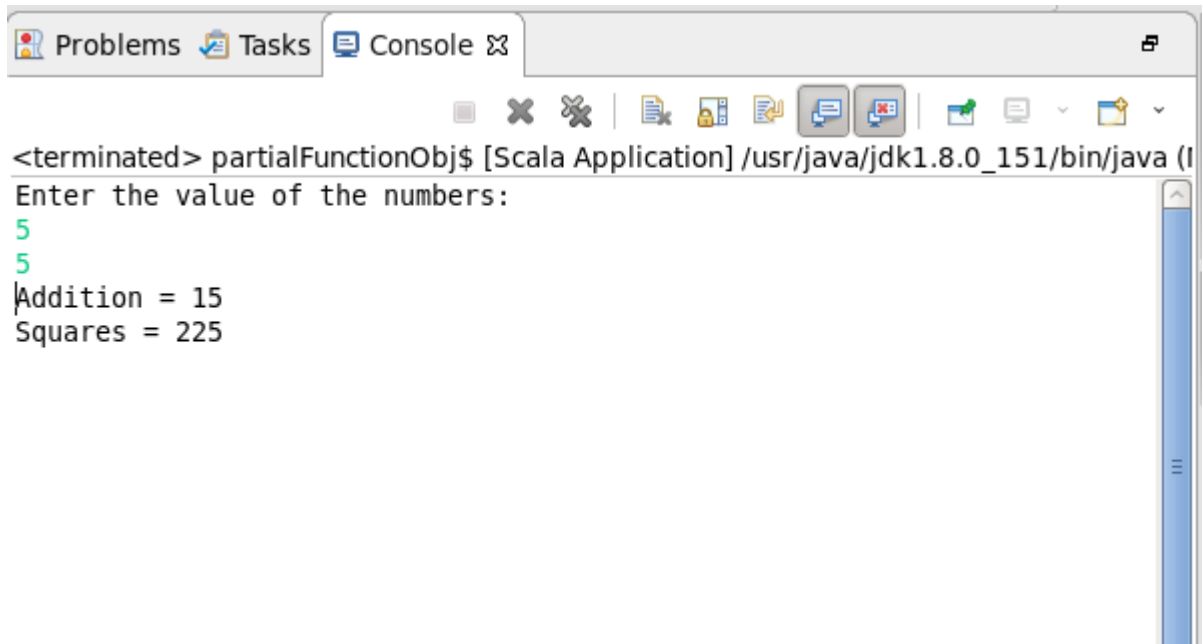
Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result

```
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)
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)
}
}
```

```
partialClass.scala  partialFunctionObj.scala
1
2
3 class partialClass {
4   def squareFunc(x:Int) : Unit = {
5     println("Squares = " + x*x)
6   };
7   def addition(x:Int ,y:Int,z:Int)=x+y+z;
8   val add = addition(5,_,_:Int);
9   def partialFunc(a: Int,b: Int) : Unit = {
10    println("Addition = " +add(a,b));
11    squareFunc(add(a,b))
12  }
13 }
```

```
partialClass.scala  partialFunctionObj.scala
1
2
3 object partialFunctionObj {
4   def main(args:Array[String]): Unit = {
5     println("Enter the value of the numbers:");
6     var a:Int = scala.io.StdIn.readLine().toInt;
7     var b:Int = scala.io.StdIn.readLine().toInt;
8     new partialClass().partialFunc(a,b)
9   }
10 }
```

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

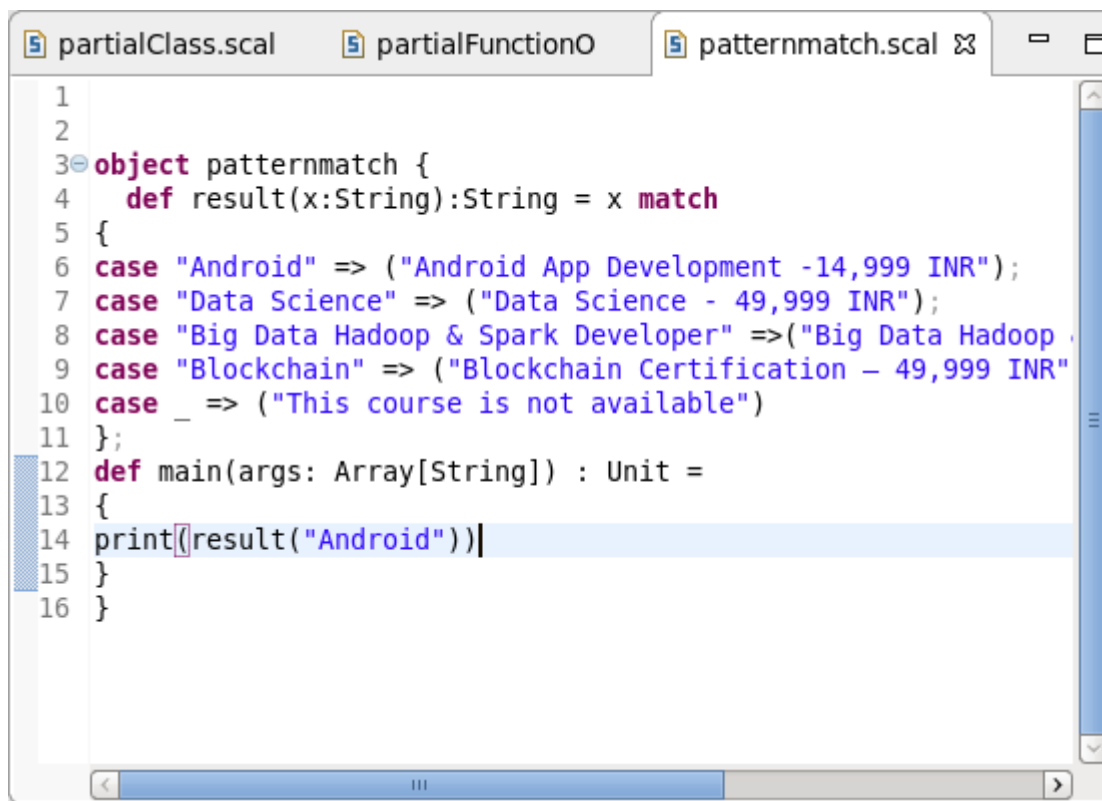
using match and add a default condition if the user enters any other course.

```
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")
  }
}
```



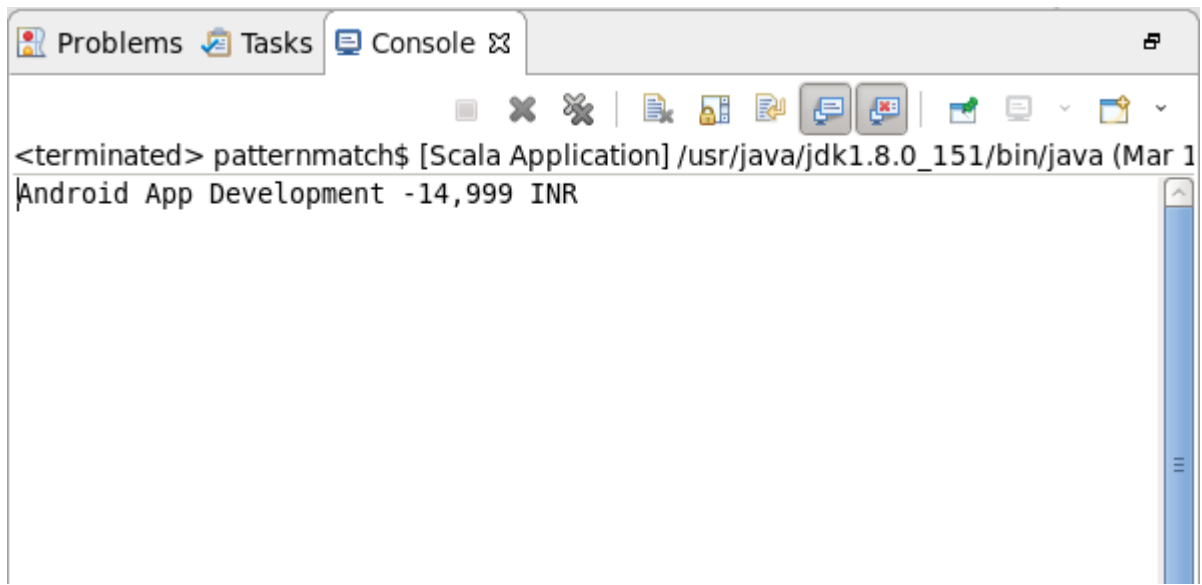
```
}  
  
def main(args: Array[String]) : Unit =  
{  
  print(result("Big Data Hadoop & Spark Developer"))  
}  
}
```

Example 1 :- print(result("Android"))



```
1  
2  
3 object patternmatch {  
4   def result(x:String):String = x match  
5   {  
6     case "Android" => ("Android App Development -14,999 INR");  
7     case "Data Science" => ("Data Science - 49,999 INR");  
8     case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop  
9     case "Blockchain" => ("Blockchain Certification – 49,999 INR"  
10    case _ => ("This course is not available")  
11  };  
12  def main(args: Array[String]) : Unit =  
13  {  
14    print(result("Android"))  
15  }  
16  }
```

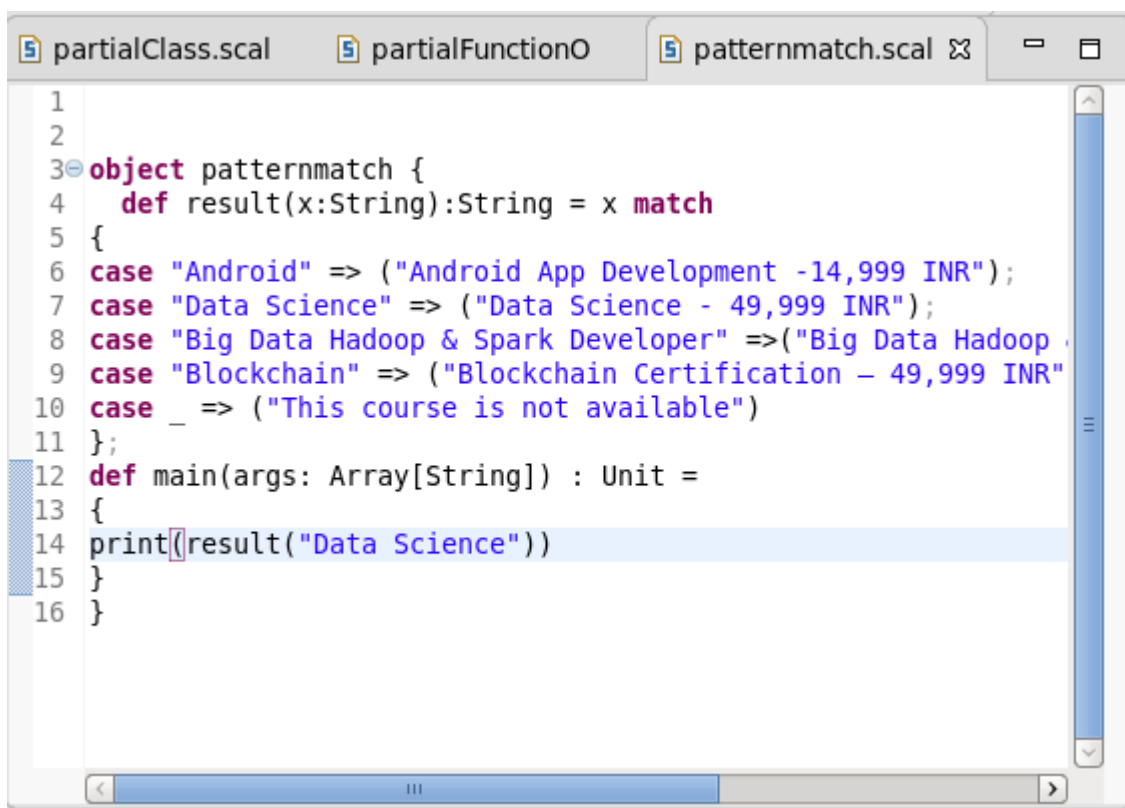
Output



The screenshot shows an IDE console window with tabs for 'Problems', 'Tasks', and 'Console'. The console output displays the result of a pattern match for 'Data Science'.

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

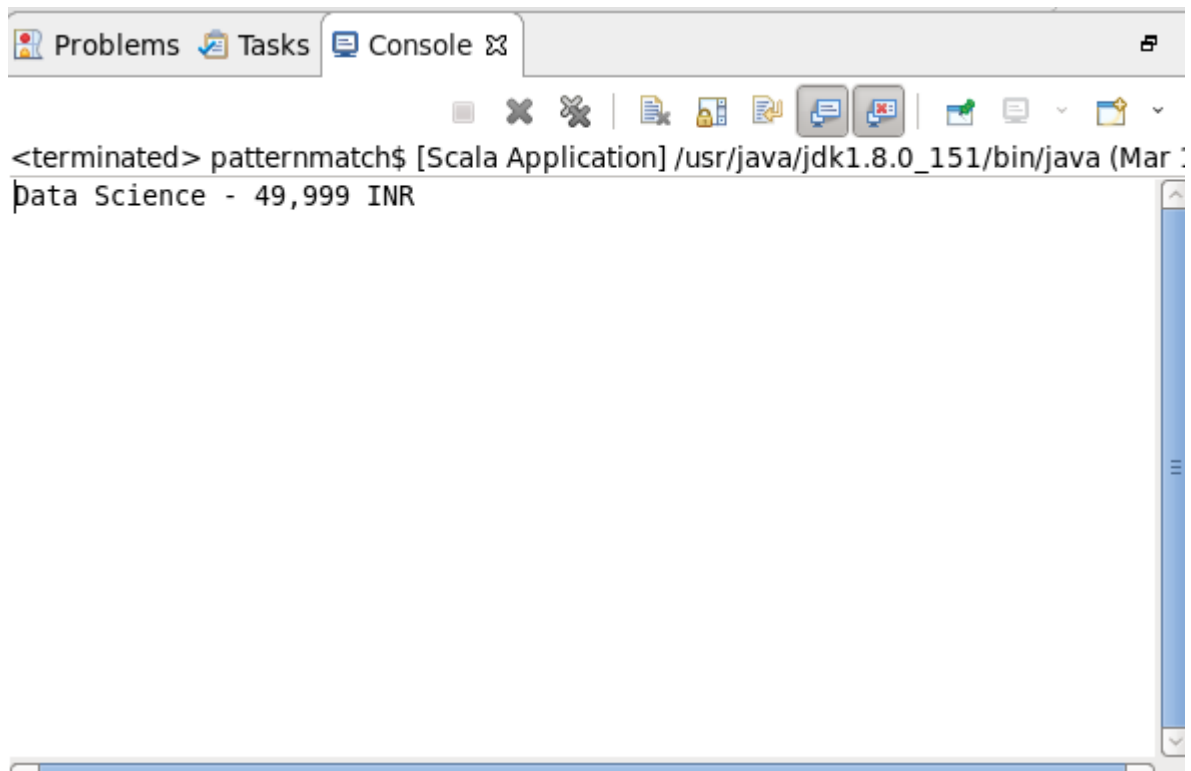
Example 2:- print(result("Data Science"))



The screenshot shows an IDE editor with three tabs: 'partialClass.scala', 'partialFunctionO', and 'patternmatch.scala'. The code in 'patternmatch.scala' defines a pattern match object and a main function that prints the result of a match for 'Data Science'.

```
1  
2  
3 object patternmatch {  
4   def result(x:String):String = x match  
5   {  
6     case "Android" => ("Android App Development -14,999 INR");  
7     case "Data Science" => ("Data Science - 49,999 INR");  
8     case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop  
9     case "Blockchain" => ("Blockchain Certification – 49,999 INR"  
10    case _ => ("This course is not available")  
11  };  
12  def main(args: Array[String]) : Unit =  
13  {  
14    print(result("Data Science"))  
15  }  
16 }
```


Output



The screenshot shows an IDE console window with tabs for Problems, Tasks, and Console. The console output displays the result of a pattern match for the input "Data Science", which is "Data Science - 49,999 INR". The prompt "<terminated> patternmatch\$" is visible at the top of the console.

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

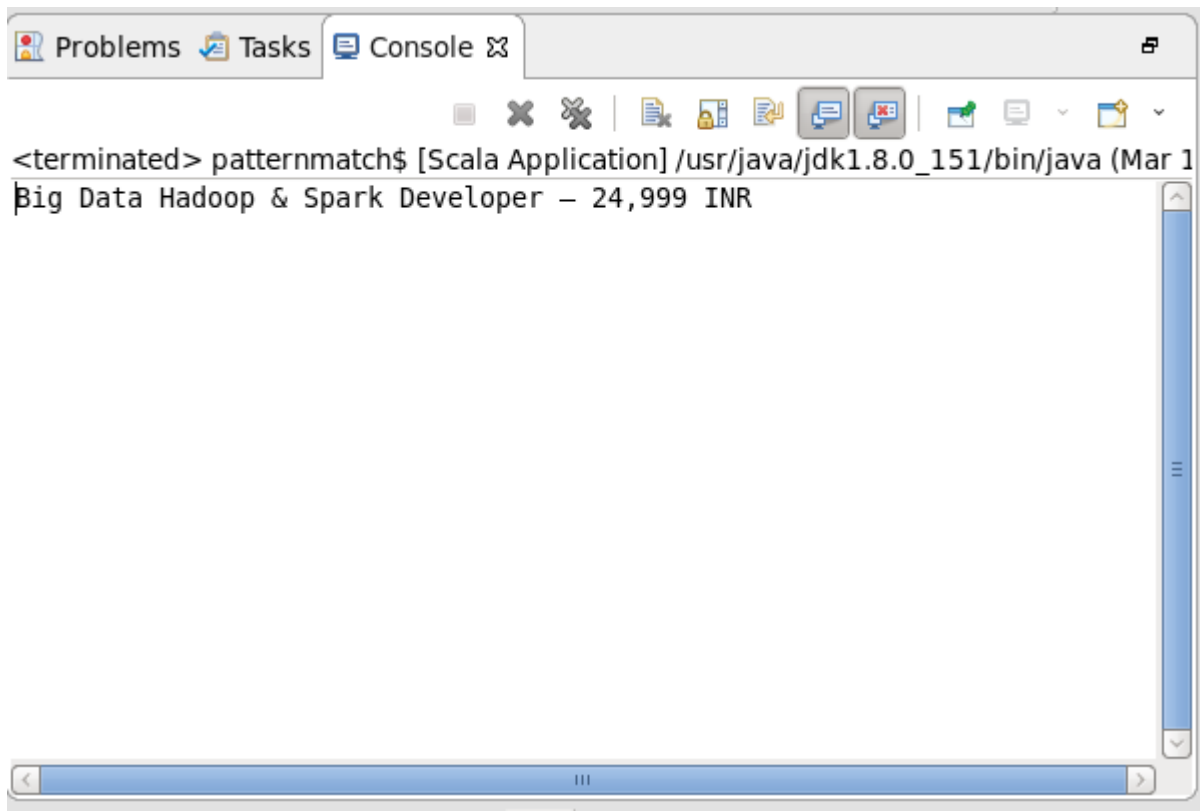
Example 3:- `print(result("Big Data Hadoop and Spark Developer"))`



The screenshot shows an IDE editor with three tabs: partialClass.scala, partialFunctionO, and patternmatch.scala. The code in patternmatch.scala defines an object with a match function and a main method that prints the result of a pattern match for "Big Data Hadoop & Spark Developer".

```
1  
2  
3 object patternmatch {  
4   def result(x:String):String = x match  
5   {  
6     case "Android" => ("Android App Development -14,999 INR");  
7     case "Data Science" => ("Data Science - 49,999 INR");  
8     case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop  
9     case "Blockchain" => ("Blockchain Certification – 49,999 INR"  
10    case _ => ("This course is not available")  
11  };  
12  def main(args: Array[String]) : Unit =  
13  {  
14    print(result("Big Data Hadoop & Spark Developer"))  
15  }  
16 }
```

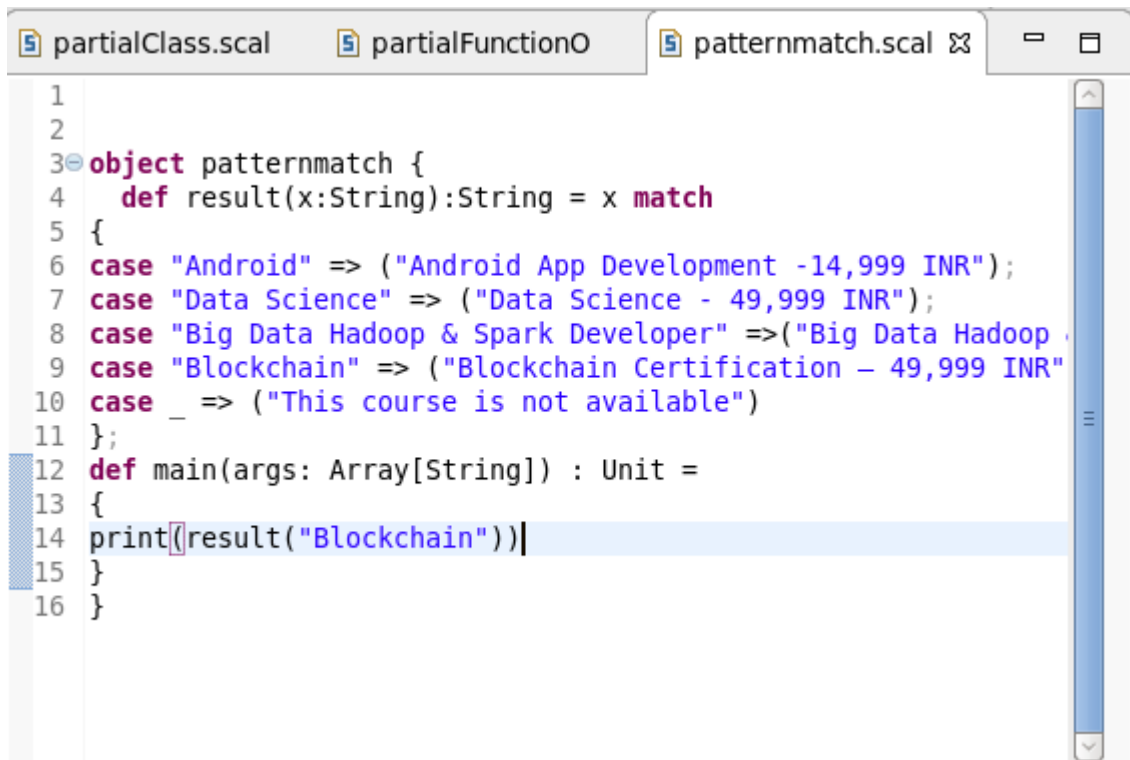
Output



The screenshot shows an IDE console window with tabs for 'Problems', 'Tasks', and 'Console'. The console output displays the command prompt for a Scala application, indicating it has terminated. The output text is: `<terminated> patternmatch$ [Scala Application] /usr/java/jdk1.8.0_151/bin/java (Mar 1` followed by a line break and `Big Data Hadoop & Spark Developer – 24,999 INR`. A vertical scrollbar is visible on the right side of the console area.

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

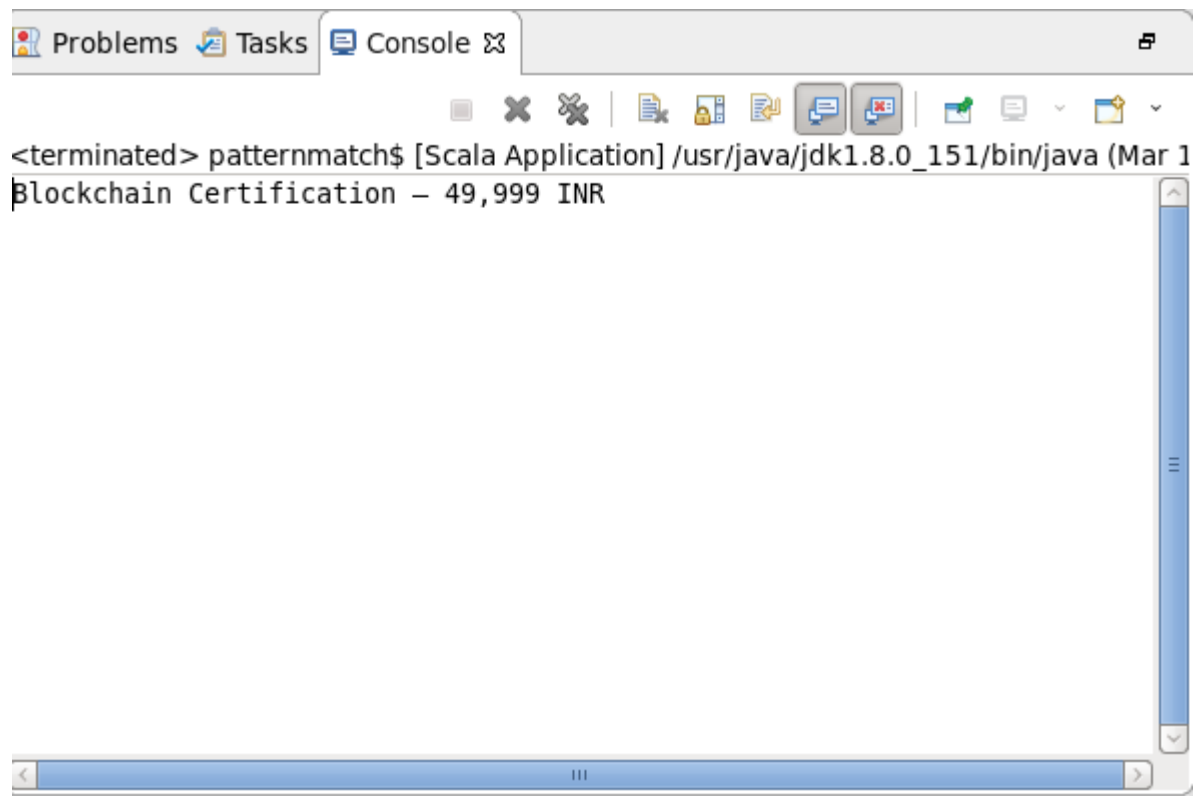
Example 4:- print(result("Blockchain"))



The screenshot shows an IDE editor with three tabs: 'partialClass.scala', 'partialFunctionO', and 'patternmatch.scala'. The 'patternmatch.scala' tab is active, showing the following Scala code: `1
2
3 object patternmatch {
4 def result(x:String):String = x match
5 {
6 case "Android" => ("Android App Development -14,999 INR");
7 case "Data Science" => ("Data Science - 49,999 INR");
8 case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop
9 case "Blockchain" => ("Blockchain Certification – 49,999 INR"
10 case _ => ("This course is not available")
11 };
12 def main(args: Array[String]) : Unit =
13 {
14 print(result("Blockchain"))
15 }
16 }`. Line 14 is highlighted with a blue background. A vertical scrollbar is visible on the right side of the editor area.

```
1
2
3 object patternmatch {
4   def result(x:String):String = x match
5   {
6     case "Android" => ("Android App Development -14,999 INR");
7     case "Data Science" => ("Data Science - 49,999 INR");
8     case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop
9     case "Blockchain" => ("Blockchain Certification – 49,999 INR"
10    case _ => ("This course is not available")
11  };
12  def main(args: Array[String]) : Unit =
13  {
14    print(result("Blockchain"))
15  }
16 }
```

Output



The screenshot shows an IDE console window with tabs for 'Problems', 'Tasks', and 'Console'. The console output displays the command prompt for a Scala application, followed by the output of a pattern match function. The output is:

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

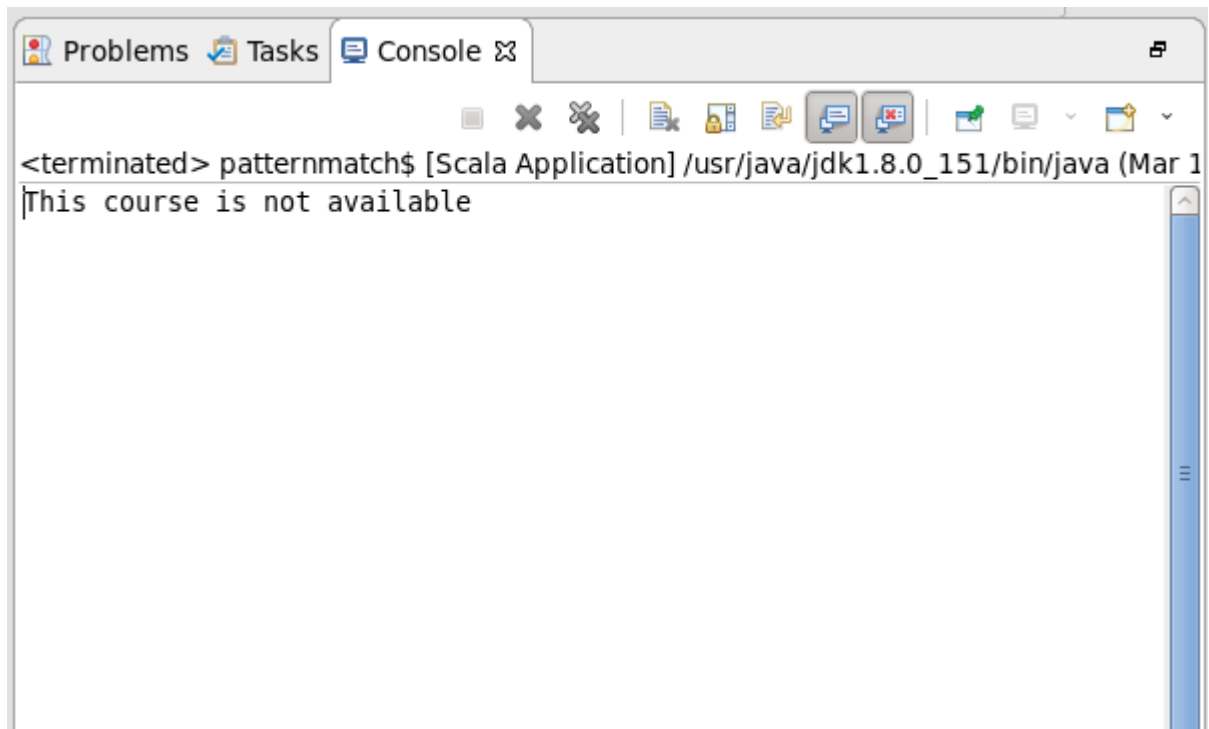
Example 5:- print(result("Networking"))



The screenshot shows an IDE editor with three tabs: 'partialClass.scal', 'partialFunctionO', and 'patternmatch.scal'. The code in the 'patternmatch.scal' tab is as follows:

```
1  
2  
3 object patternmatch {  
4   def result(x:String):String = x match  
5   {  
6     case "Android" => ("Android App Development -14,999 INR");  
7     case "Data Science" => ("Data Science - 49,999 INR");  
8     case "Big Data Hadoop & Spark Developer" =>("Big Data Hadoop  
9     case "Blockchain" => ("Blockchain Certification - 49,999 INR"  
10    case _ => ("This course is not available")  
11  };  
12  def main(args: Array[String]) : Unit =  
13  {  
14    print(result("Networking"))  
15  }  
16  }
```

Output



The screenshot shows an IDE's console window. At the top, there are tabs for 'Problems', 'Tasks', and 'Console'. The 'Console' tab is active. Below the tabs is a toolbar with various icons for file operations and debugging. The console output area contains the following text:

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

The text is displayed in a monospaced font. A vertical scrollbar is visible on the right side of the console area.