Assignment 17.1
Scala Basics 4

Task 1 :

Write a simple program to show inheritance in scala.

Here **vehicle** is parent class (Base class) and **car** is child class (Derived class). So we have inherited properties from vehicle class into car class.

Here you could see that we have used **override** keyword to override members of Parent class Vehicle.

Parent class vehicle has **parameter** with value 5. This gets overridden in car class with value 10. Similarly vehicle has **details** method. This gets overridden with details method in car class.

```scala
object Simple_Inheritance {

  class vehicle {

    val parameter: Int = 5

    def details: Unit = {
      println("Vehicle gets called")
    }
  }

  class car extends vehicle {

    override val parameter: Int = 10

    override def details {
      println("Car method gets called")
    }
  }

  def main(args: Array[String]): Unit = {

    val obj = new car()
    println("parameter value is "+ obj.parameter)
    obj.details
  }
}
```

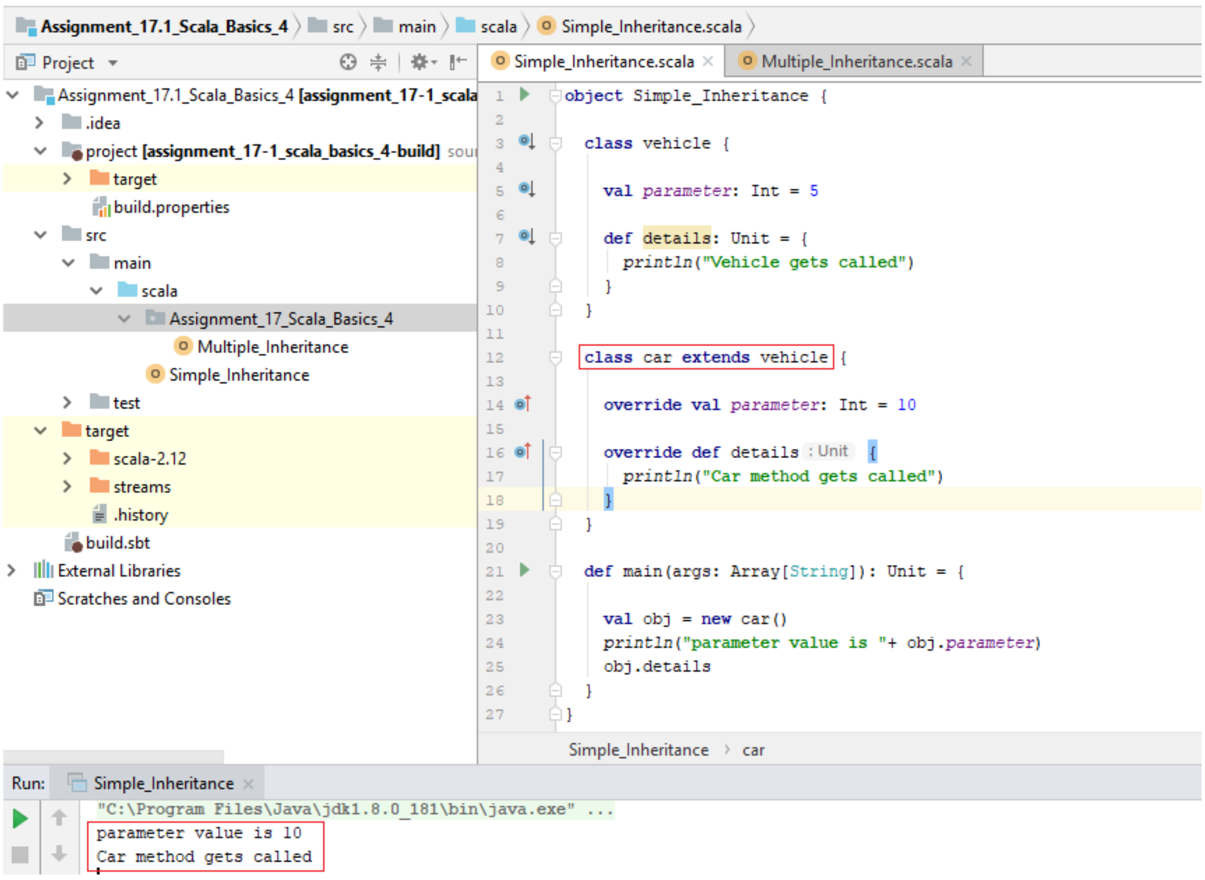
Output :

parameter value is 10

Car method gets called

Assignment 17.1
Scala Basics 4

Scala Program :



Assignment_17.1_Scala_Basics_4 › src › main › scala › Simple_Inheritance.scala

```
object Simple_Inheritance {

  class vehicle {

    val parameter: Int = 5

    def details: Unit = {
      println("Vehicle gets called")
    }
  }

  class car extends vehicle {

    override val parameter: Int = 10

    override def details : Unit  {
      println("Car method gets called")
    }
  }

  def main(args: Array[String]): Unit = {

    val obj = new car()
    println("parameter value is "+ obj.parameter)
    obj.details
  }
}
```

Run: Simple_Inheritance

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
parameter value is 10
Car method gets called
```

Assignment 17.1
Scala Basics 4

Task 2 :

Write a simple program to show multiple inheritance in scala

Here we are making use of two Traits to achieve Multiple Inheritance. We will inherit properties from both Traits A and B. **Show** method from trait A and **see** method from trait B.

```
package Assignment_17_Scala_Basics_4

object Multiple_Inheritance {

  trait A {
    def show: Unit
  }

  trait B{
    def see: Unit
  }

  class C extends A with B{
    override def show : Unit = {println("show method gets called from trait A")}

    override def see : Unit = {println("see method gets called from trait B")}
  }

  def main(args: Array[String]) : Unit = {
    val obj = new C
    obj.show
    obj.see
  }
}
```

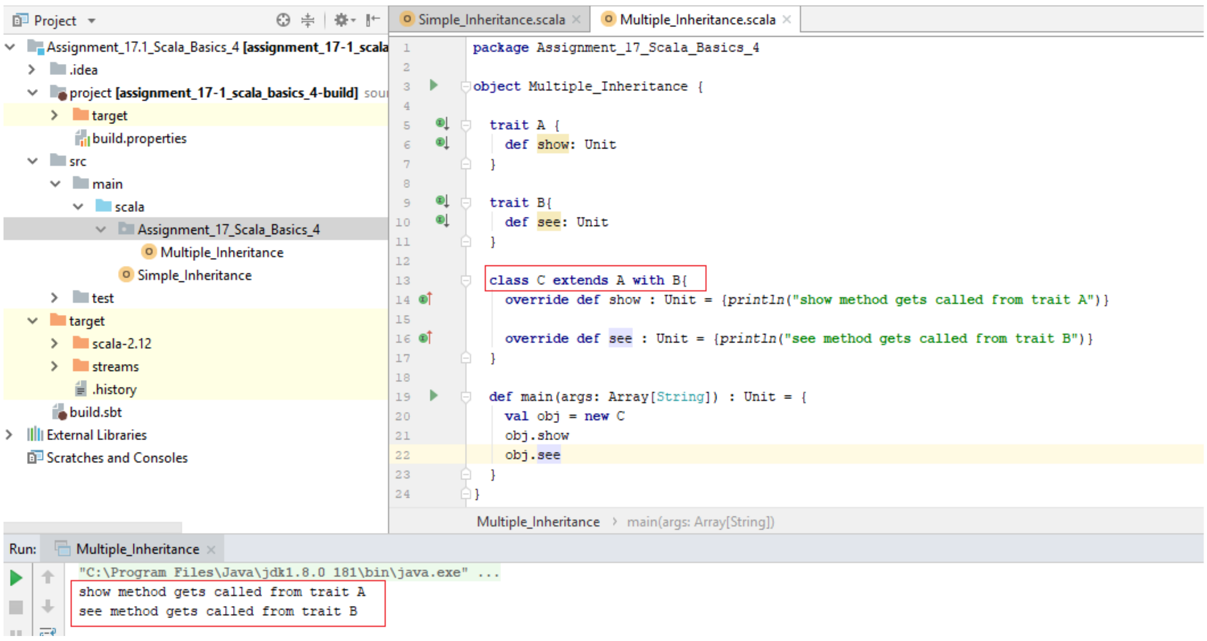
Output :

show method gets called from trait A

see method gets called from trait B

Assignment 17.1
Scala Basics 4

<u>Scala Output :</u>



Project
Assignment_17.1_Scala_Basics_4 [assignment_17-1_scala
.idea
project [assignment_17-1_scala_basics_4-build] sou
target
build.properties
src
main
scala
Assignment_17_Scala_Basics_4
Multiple_Inheritance
Simple_Inheritance
test
target
scala-2.12
streams
.history
build.sbt
External Libraries
Scratches and Consoles
Simple_Inheritance.scala
Multiple_Inheritance.scala
Multiple_Inheritance › main(args: Array[String])
Run: Multiple_Inheritance

```
package Assignment_17_Scala_Basics_4

object Multiple_Inheritance {

  trait A {
    def show: Unit
  }

  trait B{
    def see: Unit
  }

  class C extends A with B{
    override def show : Unit = {println("show method gets called from trait A")}

    override def see : Unit = {println("see method gets called from trait B")}
  }

  def main(args: Array[String]) : Unit = {
    val obj = new C
    obj.show
    obj.see
  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
show method gets called from trait A
see method gets called from trait B
```

<u>Task 3 :</u>

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.

Here we have created Partial Function with one constant **yourConstant** and two input parameters and returns the addition of these numbers as output.

Then we have created **Square** function which returns square of input parameter.

<u>Scala Program :</u>

```
object Partial_Function extends App{

  val yourConstant = 3

  val partial_fn: PartialFunction[(Int, Int), Int] = {
    case (x, y) => x + y + yourConstant
  }
  println("Output of Partial Function is "+partial_fn((5, 4)))

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

println("Square of "+partial_fn((5, 4))+ " is "+Square(partial_fn((5, 4))))

}
```

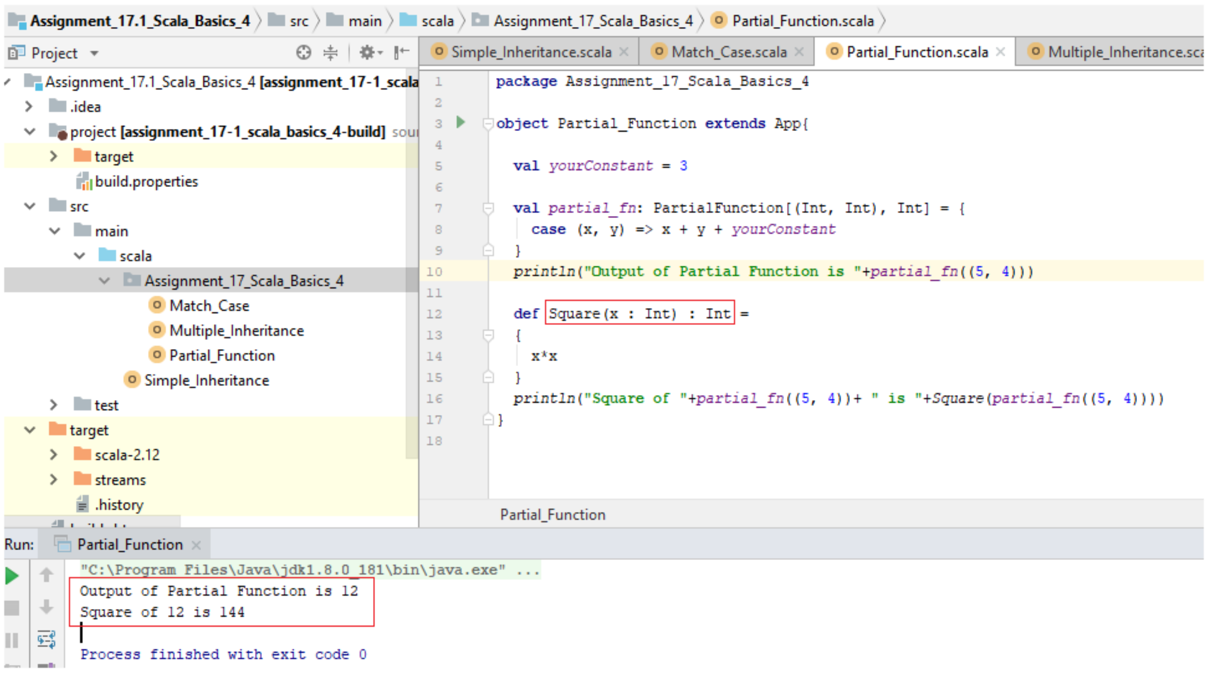
Assignment 17.1
Scala Basics 4

Output :

Output of Partial Function is 12

Square of 12 is 144

Scala Program :



```scala
package Assignment_17_Scala_Basics_4

object Partial_Function extends App{

  val yourConstant = 3

  val partial_fn: PartialFunction[(Int, Int), Int] = {
    case (x, y) => x + y + yourConstant
  }
  println("Output of Partial Function is "+partial_fn((5, 4)))

  def Square(x : Int) : Int =
  {
    x*x
  }
  println("Square of "+partial_fn((5, 4))+ " is "+Square(partial_fn((5, 4))))
}
```

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
Output of Partial Function is 12
Square of 12 is 144

Process finished with exit code 0
```

Assignment 17.1
Scala Basics 4

<u>**Task 4 :**</u>

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.

We have used **Pattern Matching** here. This includes a sequence of alternatives, each starting with the keyword **case**. Each alternative includes a pattern and expressions, which will be evaluated if the pattern matches. An arrow symbol => separates the pattern from the expressions.

If course_name matches with one of these cases then its corresponding expression would be printed. Else it will print **"Not Applicable, Please enter correct course".**

For below code, for course "**Blockchain Certification**", it shows course price is **"49,999 INR"**

and for course "DotNet", it shows course price as "**Not Applicable, Please enter correct course"**

<u>**Scala Program :**</u>

```
package Assignment_17_Scala_Basics_4

object Match_Case {

  def course_price(course_name: String) : Unit = {
    course_name 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("Not Applicable, Please 
enter correct course")
    }
  }

  def main(args : Array[String]) : Unit = {

    course_price("Blockchain Certification")
    course_price("DotNet")

  }
}
```

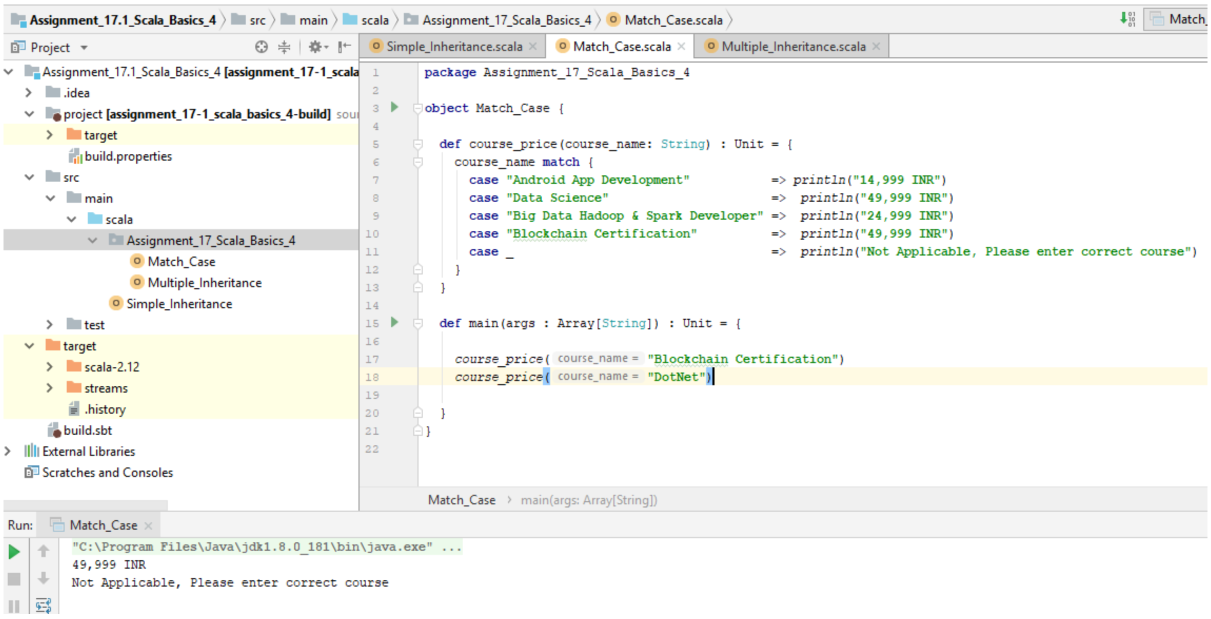
Assignment 17.1
Scala Basics 4

<u>Output :</u>

49,999 INR

Not Applicable, Please enter correct course

<u>Scala Program :</u>



Assignment_17.1_Scala_Basics_4 › src › main › scala › Assignment_17_Scala_Basics_4 › Match_Case.scala
Project
Simple_Inheritance.scala
Match_Case.scala
Multiple_Inheritance.scala
Assignment_17.1_Scala_Basics_4 [assignment_17-1_scala
.idea
project [assignment_17-1_scala_basics_4-build] sou
target
build.properties
src
main
scala
Assignment_17_Scala_Basics_4
Match_Case
Multiple_Inheritance
Simple_Inheritance
test
target
scala-2.12
streams
.history
build.sbt
External Libraries
Scratches and Consoles
Match_Case › main(args: Array[String])
Run:
Match_Case

```scala
package Assignment_17_Scala_Basics_4

object Match_Case {

  def course_price(course_name: String) : Unit = {
    course_name 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("Not Applicable, Please enter correct course")
    }
  }

  def main(args : Array[String]) : Unit = {

    course_price( course_name = "Blockchain Certification")
    course_price( course_name = "DotNet")

  }
}
```

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" ...
49,999 INR
Not Applicable, Please enter correct course
```