Scala IV

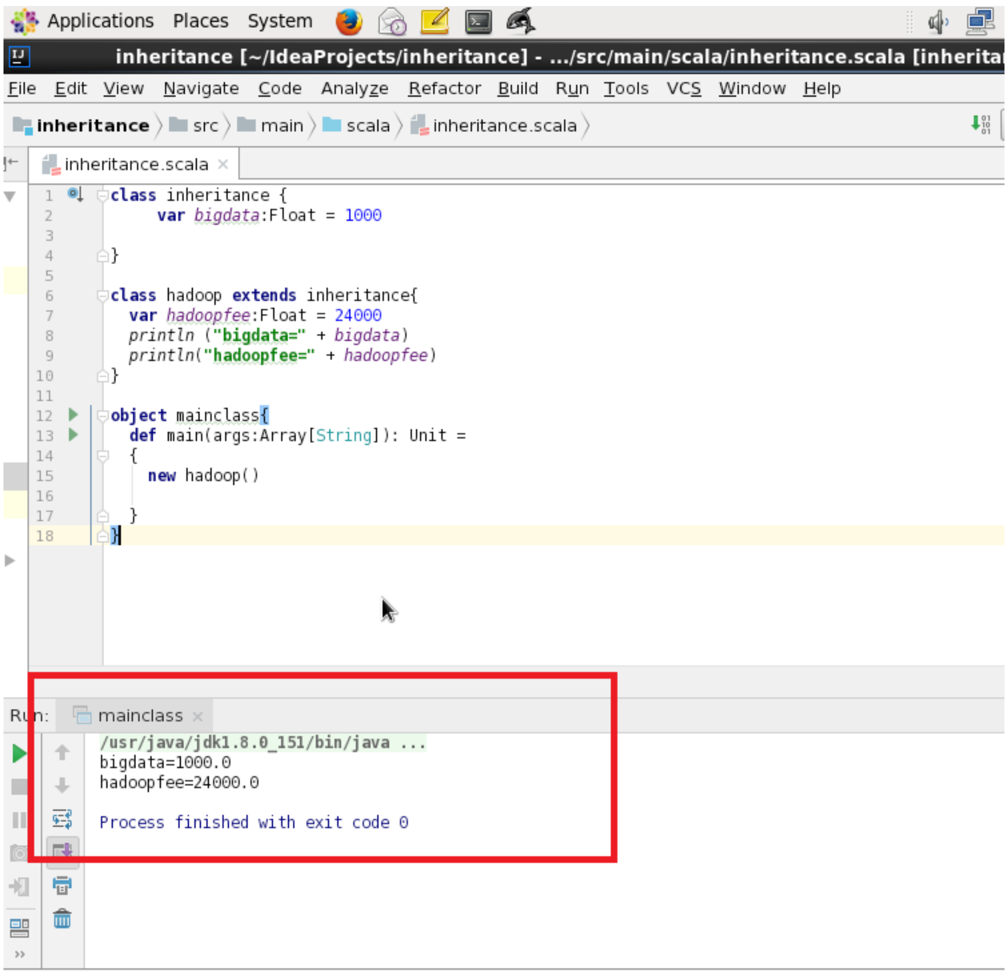
Task 1

Write a simple program to show inheritance in scala.

Code:

```
class inheritance {
     var bigdata:Float = 1000

}

class hadoop extends inheritance{
  var hadoopfee:Float = 24000
  println ("bigdata=" + bigdata)
  println("hadoopfee=" + hadoopfee)
}

object mainclass{
  def main(args:Array[String]): Unit =
  {
    new hadoop()

  }
}
```

Output:


Applications Places System
inheritance [~/IdeaProjects/inheritance] - .../src/main/scala/inheritance.scala [inherita
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
inheritance src main scala inheritance.scala
inheritance.scala
Run: mainclass

```
class inheritance {
      var bigdata:Float = 1000

}

class hadoop extends inheritance{
   var hadoopfee:Float = 24000
   println ("bigdata=" + bigdata)
   println("hadoopfee=" + hadoopfee)
}

object mainclass{
   def main(args:Array[String]): Unit =
   {
     new hadoop()

   }
}
```

```
/usr/java/jdk1.8.0_151/bin/java ...
bigdata=1000.0
hadoopfee=24000.0

Process finished with exit code 0
```

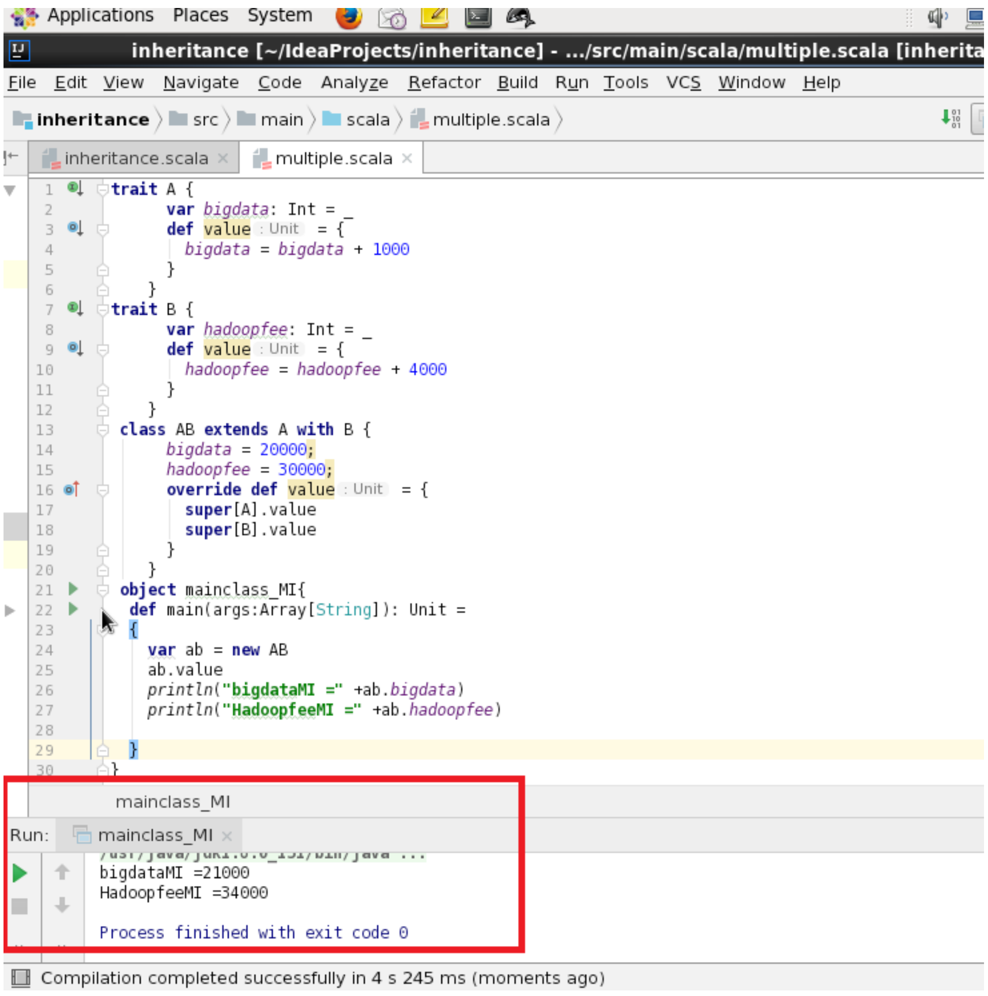
Task 2

Write a simple program to show multiple inheritance in scala

Code:

```
trait A {
      var bigdata: Int = _
      def value = {
        bigdata = bigdata + 1000
      }
    }
trait B {
      var hadoopfee: Int = _
      def value = {
        hadoopfee = hadoopfee + 4000
      }
    }
 class AB extends A with B {
      bigdata = 20000;
      hadoopfee = 30000;
      override def value = {
        super[A].value
        super[B].value
      }
    }
 object mainclass_MI{
  def main(args:Array[String]): Unit =
  {
    var ab = new AB
    ab.value
    println("bigdataMI =" +ab.bigdata)
    println("HadoopfeeMI =" +ab.hadoopfee)

  }
}
```

Output:



Applications Places System
inheritance [~/IdeaProjects/inheritance] - .../src/main/scala/multiple.scala [inherita
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
inheritance src main scala multiple.scala
inheritance.scala multiple.scala

```
trait A {
        var bigdata: Int = _
        def value : Unit  = {
          bigdata = bigdata + 1000
        }
    }
trait B {
        var hadoopfee: Int = _
        def value : Unit  = {
          hadoopfee = hadoopfee + 4000
        }
    }
  class AB extends A with B {
        bigdata = 20000;
        hadoopfee = 30000;
        override def value : Unit  = {
          super[A].value
          super[B].value
        }
    }
  object mainclass_MI{
   def main(args:Array[String]): Unit =
   {
     var ab = new AB
     ab.value
     println("bigdataMI =" +ab.bigdata)
     println("HadoopfeeMI =" +ab.hadoopfee)

   }
}
```

```
/usr/java/jdk1.8.0_151/bin/java ...
bigdataMI =21000
HadoopfeeMI =34000

Process finished with exit code 0
```

Compilation completed successfully in 4 s 245 ms (moments ago)

Task 3

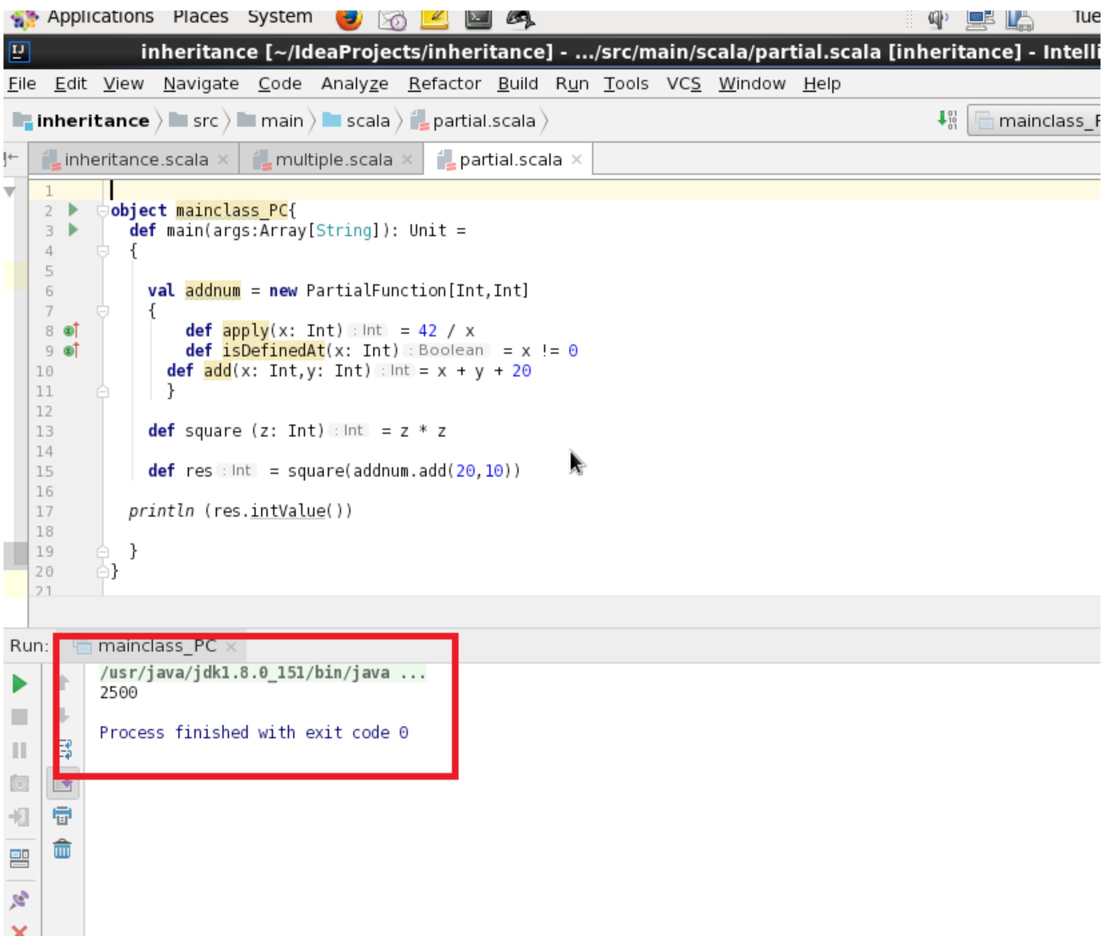
Write a partial function to add three numbers in which one number is constant and two numbers can be passed as inputs and define another method which can take the partial function as input and squares the result.

Code:

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

    val addnum = new PartialFunction[Int,Int]
    {
        def apply(x: Int) = 42 / x
        def isDefinedAt(x: Int) = x != 0
```

```
      def add(x: Int,y: Int)= x + y + 20
      }

    def square (z: Int) = z * z

    def res = square(addnum.add(20,10))

  println (res.intValue())

  }
}
```

Output:



Applications Places System
inheritance [~/IdeaProjects/inheritance] - .../src/main/scala/partial.scala [inheritance] - Intelli
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
inheritance src main scala partial.scala
mainclass_F
inheritance.scala
multiple.scala
partial.scala
Run: mainclass_PC

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

    val addnum = new PartialFunction[Int,Int]
    {
        def apply(x: Int) : Int  = 42 / x
        def isDefinedAt(x: Int) : Boolean  = x != 0
      def add(x: Int,y: Int) : Int = x + y + 20
      }

    def square (z: Int) : Int  = z * z

    def res : Int  = square(addnum.add(20,10))

  println (res.intValue())

  }
}
```

```
/usr/java/jdk1.8.0_151/bin/java ...
2500

Process finished with exit code 0
```

Task 4

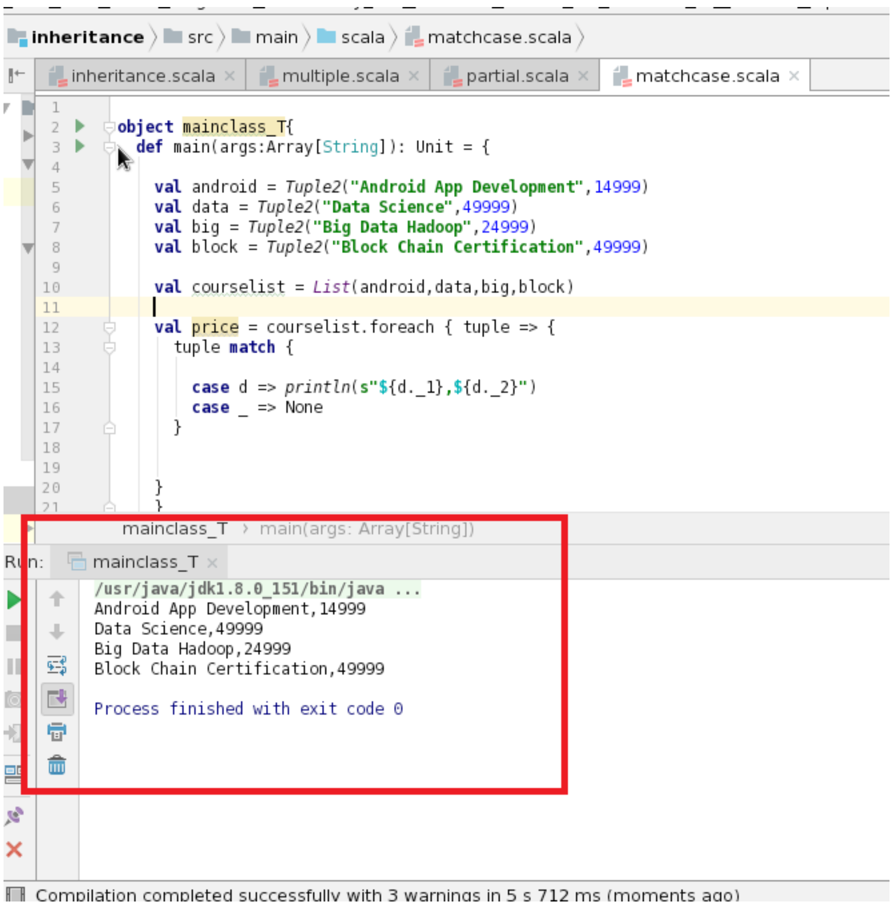
Write a program to print the prices of 4 courses of Acadgild:

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.

Code:

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

    val android = Tuple2("Android App Development",14999)
    val data = Tuple2("Data Science",49999)
    val big = Tuple2("Big Data Hadoop",24999)
    val block = Tuple2("Block Chain Certification",49999)

    val courselist = List(android,data,big,block)

    val price = courselist.foreach { tuple => {
      tuple match {

        case d => println(s"${d._1},${d._2}")
        case _ => None
      }


    }
    }

  }
  }
```

Output:



inheritance › src › main › scala › matchcase.scala
inheritance.scala
multiple.scala
partial.scala
matchcase.scala
mainclass_T › main(args: Array[String])
Run: mainclass_T

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

    val android = Tuple2("Android App Development",14999)
    val data = Tuple2("Data Science",49999)
    val big = Tuple2("Big Data Hadoop",24999)
    val block = Tuple2("Block Chain Certification",49999)

    val courselist = List(android,data,big,block)

    val price = courselist.foreach { tuple => {
      tuple match {

        case d => println(s"${d._1},${d._2}")
        case _ => None
      }


    }
    }
```

```
/usr/java/jdk1.8.0_151/bin/java ...
Android App Development,14999
Data Science,49999
Big Data Hadoop,24999
Block Chain Certification,49999

Process finished with exit code 0
```

Compilation completed successfully with 3 warnings in 5 s 712 ms (moments ago)