

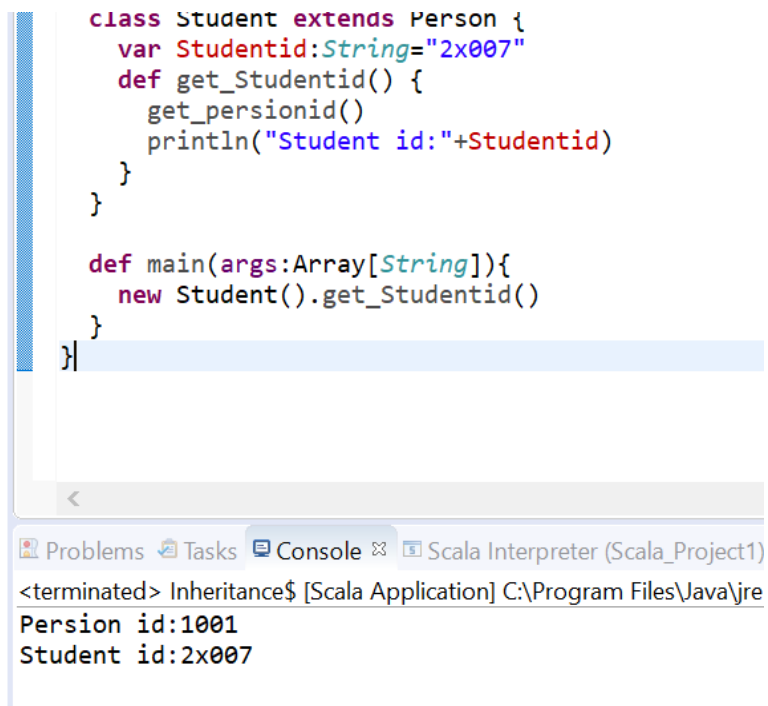
## Task 1

Write a simple program to show inheritance in scala.

Code-

```
object Inheritance {  
  
  class Person {  
    var Persionid:Int=1001  
    def get_persionid(){  
      println("Persion id:"+Persionid)  
    }  
  }  
  class Student extends Person {  
    var Studentid:String="2x007"  
    def get_Studentid() {  
      get_persionid()  
      println("Student id:"+Studentid)  
    }  
  }  
  
  def main(args:Array[String]){  
    new Student().get_Studentid()  
  }  
}
```

**Output-** class Student is inheriting the attributes and method of class Person



The screenshot shows an IDE with a Scala file named 'Inheritance.scala'. The code defines a 'Person' class with a 'Persionid' attribute and a 'get\_persionid()' method. A 'Student' class extends 'Person', adding a 'Studentid' attribute and a 'get\_Studentid()' method that calls 'get\_persionid()' and prints the student ID. The 'main' method creates a 'Student' object and calls 'get\_Studentid()'. The console output shows 'Persion id:1001' and 'Student id:2x007'.

```
class Student extends Person {  
  var Studentid:String="2x007"  
  def get_Studentid() {  
    get_persionid()  
    println("Student id:"+Studentid)  
  }  
}  
  
def main(args:Array[String]){  
  new Student().get_Studentid()  
}
```

Problems Tasks Console Scala Interpreter (Scala\_Project1)  
<terminated> Inheritance\$ [Scala Application] C:\Program Files\Java\jre  
Persion id:1001  
Student id:2x007

## Task 2

Write a simple program to show multiple inheritance in scala

Code-

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

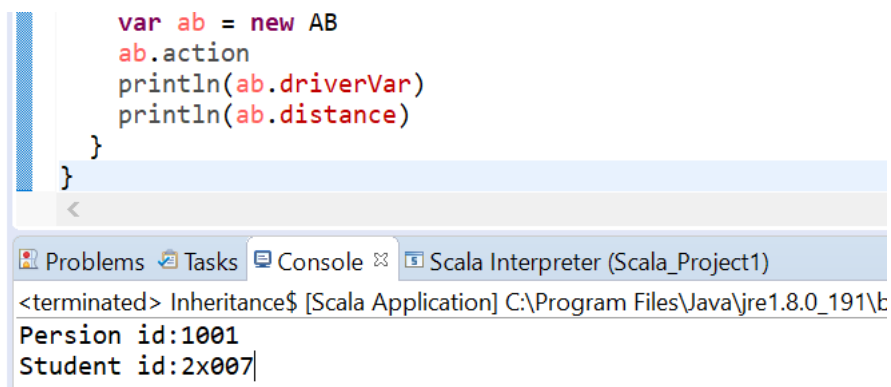
        trait A {
            var distance: Int = _
            def action = {
                distance = distance + 5
            }
        }

        trait B {
            var driverVar: Int = _
            def action = {
                driverVar = driverVar + 1
            }
        }

        class AB extends A with B {
            distance = 3;
            driverVar = 6;
            override def action = {
                super[A].action
                super[B].action
            }
        }

        var ab = new AB
        ab.action
        println(ab.driverVar)
        println(ab.distance)
    }
}
```

Output-



```
var ab = new AB
ab.action
println(ab.driverVar)
println(ab.distance)
}
```

The screenshot shows an IDE with a Scala file. The code defines a class `AB` that inherits from two traits, `A` and `B`. Trait `A` has a `distance` variable and an `action` method that increments it by 5. Trait `B` has a `driverVar` variable and an `action` method that increments it by 1. The `AB` class overrides the `action` method to call both `super[A].action` and `super[B].action`. In the `main` method, an instance `ab` of `AB` is created, `ab.action` is called, and the values of `ab.driverVar` and `ab.distance` are printed. The console output shows the program terminated successfully, with the output `6` and `10` (though the text in the image is partially obscured and appears as `id:1001` and `id:2x007`).

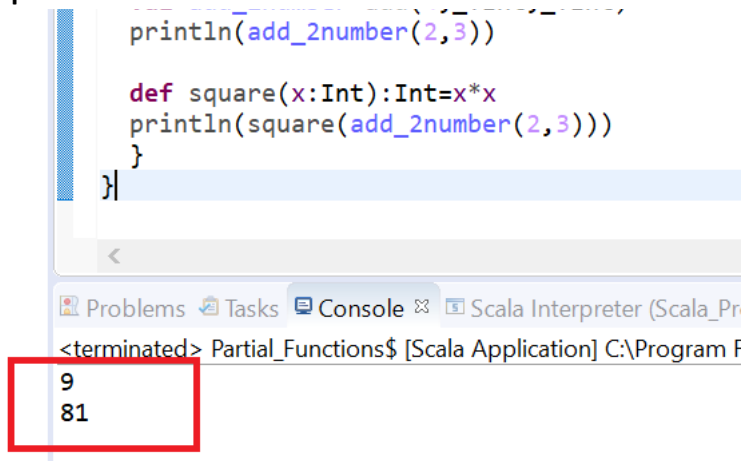
### Task 3

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 Partial_Functions {  
  
    def main(args:Array[String]){  
  
        def add(x:Int,y:Int,z:Int)=x+y+z  
        val add_2number=add(4,_,_:Int)  
        println(add_2number(2,3))  
  
        def square(x:Int):Int=x*x  
        println(square(add_2number(2,3)))  
    }  
}
```

Output-



The screenshot shows an IDE with a Scala file. The code defines a partial function `add_2number` and a `square` function. The output console shows the results of the program execution.

```
println(add_2number(2,3))  
  
def square(x:Int):Int=x*x  
println(square(add_2number(2,3)))  
}  
}
```

Problems Tasks Console Scala Interpreter (Scala\_Pr  
<terminated> Partial\_Functions\$ [Scala Application] C:\Program F  
9  
81

### 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 Match {
  def main(args:Array[String]){

    println("Enter course name:")
    var input=scala.io.StdIn.readLine()
    var price= input 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("price not available")
    }
  }
}

```

### Output-

```

<terminated> Match$ [Scala Application] C:\Program Files\Java\j
Enter course name:
Data Science
49,999 INR

```

```

Problems Tasks Console Scala Interpreter ($)
<terminated> Match$ [Scala Application] C:\Program File:
Enter course name:
Java Certification
price not available

```