Assignment 17: Scala Basics 4 Assignment Problems

<u>Problem Statement</u>

==

<u>Task 1</u>

Write a simple program to show inheritance in scala.

<u>Program:</u>

```
    class Employee{
        var salary:Float = 10000
    }

    class Programmer extends Employee{
        var bonus:Int = 5000
        println("Salary = "+salary)
        println("Bonus = "+bonus)
    }

    object MainObject{
        def main(args:Array[String]){
            new Programmer()
        }
    }
```

<u>Terminal Execution :</u>

```
scala> class Employee{
     | var salary:Float = 10000
     | }
defined class Employee

scala> class Programmer extends Employee{
     | var bonus:Int = 5000
     | println("Salary = "+salary)
     | println("Bonus = "+bonus)
     | }
defined class Programmer
```

```
scala> object MainObject{
     | def main(args:Array[String]){
     | new Programmer()
     | }
     | }
defined object MainObject

scala> new Programmer()
Salary = 10000.0
Bonus = 5000
res0: Programmer = Programmer@18e8568
```

<u>Explaination :</u>

Here, Employee is a base class and Programmer is a sub-class of Employee. Thus, Programmer class object is able to inheriet the salary from the Base class 'Employee' and able to display it on calling the function with reference variable of Employee instantiated using Programmer object.

--
==

<u>Task 2</u>

Write a simple program to show multiple inheritance in scala

<u>Program:</u>

```
   trait Printable{
        def print()
    }

    trait Showable{
       def show()
    }

    class A6 extends Printable with Showable{
        def print(){
            println("This is printable")
        }
        def show(){
            println("This is showable");
        }
    }
```

```
object MainObject{
    def main(args:Array[String]){
        var a = new A6()
        a.print()
        a.show()
    } }
```

Terminal Execution :

```
scala>  trait Printable{
     | def print()
     | }
defined trait Printable

scala> trait Showable{
     | def show()
     | }
defined trait Showable

scala> class A6 extends Printable with Showable{
     | def print(){
     | println("This is printable")
     | }
     | def show(){
     | println("This is showable");
     | }
     | }
defined class A6

scala> object MainObject{
     | def main(args:Array[String]){
     | var a = new A6()
     | a.print()
     | a.show()
     | }
     | }
defined object MainObject

scala> new A6()
res5: A6 = A6@3640d5

scala> new A6().print()
This is printable
```

```
scala> new A6().show()
This is showable
```

<u>Explaination :</u>

Here, we are implementing multiple inheritance by extending two parent class printable and showable in A6 class. Therefore, on declaring the print() and show() methods in parent class, we are defining their body in their sub-class.

--
==

<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.

<u>Terminal Execution :</u>

```
scala> class Calculate{ private var result = 0
     | def add(a:Int,b:Int,c:Int):Unit = result = a + b + c
     | def square(a:Int):Unit = result = a * a
     | def printResult():Int = result
     | }
defined class Calculate

scala> var cal = new Calculate()
cal: Calculate = Calculate@317c52

scala> var cal = new Calculate()
cal: Calculate = Calculate@317c52

scala> val sum  = cal.add(4,_:Int,_:Int)
sum: (Int, Int) => Unit = <function2>

scala> sum(6,8)

scala> val sumResult  = cal.printResult()
sumResult: Int = 18

scala> val squares = cal.square(_:Int)
squares: Int => Unit = <function1>
```

```
scala> squares(5)

scala> val squareResult  = cal.printResult()
squareResult: Int = 25
```

Explaination :

Here, we are defining the partial function in the class 'Claculate' and then getting Sum and square of the number by passing numbers at the methods as parameter at main method or at run time.

==

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.

Program :

```
object MainObject {
       def main(args: Array[String]) {
            var result = search ("Hello")
            print(result)
        }
        def printFee(a:Any):Any = a 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("Course Not Found")
            }
    }
```

Terminal Execution :

```
scala> def printFee(a:Any):Any = a 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("Course Not Found")
     | }
printFee: (a: Any)Any

scala> var result = printFee("Android App Development")
14,999 INR
result: Any = ()

scala> var result = printFee("Data Science")
49,999 INR
result: Any = ()

scala> var result = printFee("Big Data Hadoop & Spark Developer")
24,999 INR
result: Any = ()

scala> var result = printFee("Blockchain Certification")
49,999 INR
result: Any = ()

scala> var result = printFee("Angular")
Course Not Found
result: Any = ()
```

Explaination :

Here, we are passing the name of the course as parameter to the method printFee() and using 'match' , we are fetching and displaying the respective fee of that course. In case, if the name of the course doesn't match any of the define courses, “Course Not Found”message is displayed.