

Assignment Day 17

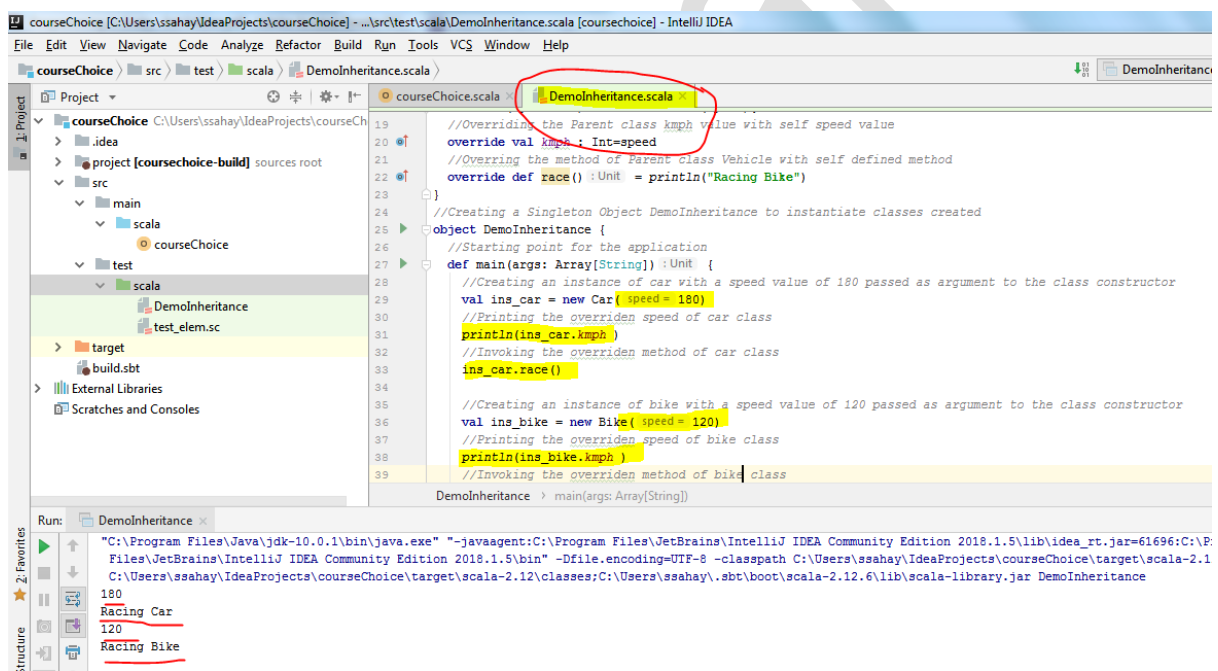
Task 1:

Write a simple program to show inheritance in Scala.

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

ScreenShot:



```
19 //Overriding the Parent class kmph value with self speed value
20 override val kmph : Int= speed
21 //Overriding the method of Parent class Vehicle with self defined method
22 override def race() : Unit = println("Racing Bike")
23
24 //Creating a Singleton Object DemoInheritance to instantiate classes created
25 object DemoInheritance {
26   //Starting point for the application
27   def main(args: Array[String]) : Unit {
28     //Creating an instance of car with a speed value of 180 passed as argument to the class constructor
29     val ins_car = new Car( speed = 180)
30     //Printing the overridden speed of car class
31     println(ins_car.kmph)
32     //Invoking the overridden method of car class
33     ins_car.race()
34
35     //Creating an instance of bike with a speed value of 120 passed as argument to the class constructor
36     val ins_bike = new Bike( speed = 120)
37     //Printing the overridden speed of bike class
38     println(ins_bike.kmph)
39     //Invoking the overridden method of bike class
40     ins_bike.race()
41   }
42 }
```

Run: DemoInheritance

```
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\lib\idea_rt.jar=61696:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\ssahay\IdeaProjects\courseChoice\target\scala-2.12\classes;C:\Users\ssahay\IdeaProjects\courseChoice\target\scala-2.12\lib\scala-library.jar DemoInheritance
```

180
Racing Car
120
Racing Bike

Task 2:

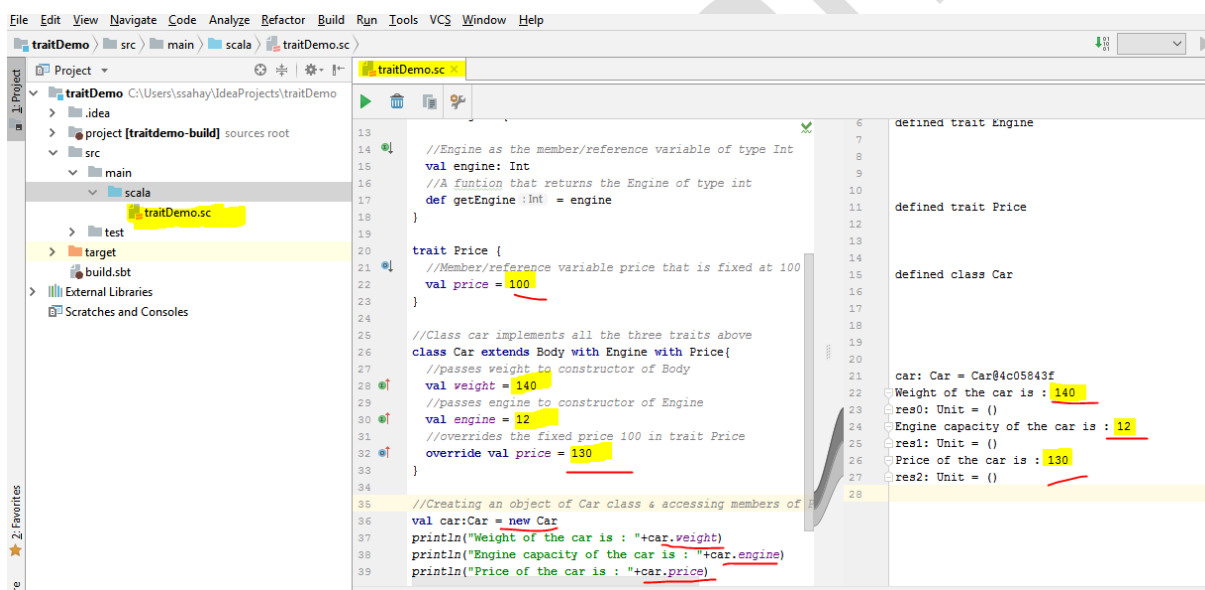
Write a simple program to show multiple inheritance in SCALA.

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

*Multiple Inheritance in Scala is achieved by Traits as shown below using **extends** & **with** keyword.*

ScreenShot:



The screenshot shows an IDE window with a Scala file named `traitDemo.sc`. The code defines three traits: `Engine`, `Price`, and `Body`. The `Car` class extends `Body` with `Engine` and `Price`. The `Car` class has a constructor that takes `weight`, `engine`, and `price` as parameters. The `Car` class overrides the `price` from the `Price` trait. The `Car` class has a `getEngine` method that returns the `engine` variable. The `Car` class has a `main` method that creates a `Car` object and prints its `weight`, `engine`, and `price`.

```
13 //Engine as the member/reference variable of type Int
14 val engine: Int
15 //A function that returns the Engine of type int
16 def getEngine :Int = engine
17
18
19
20 trait Price {
21   //Member/reference variable price that is fixed at 100
22   val price = 100
23 }
24
25 //Class car implements all the three traits above
26 class Car extends Body with Engine with Price{
27   //passes weight to constructor of Body
28   val weight = 140
29   //passes engine to constructor of Engine
30   val engine = 12
31   //overrides the fixed price 100 in trait Price
32   override val price = 130
33 }
34
35 //Creating an object of Car class & accessing members of
36 val car:Car = new Car
37 println("Weight of the car is : "+car.weight)
38 println("Engine capacity of the car is : "+car.engine)
39 println("Price of the car is : "+car.price)
```

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.

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

ScreenShot:

```
1 // singleton object to call the functions
2 object partialFun
3 {
4     //Square function that takes an integer & returns square of that integer.
5     def squareFun(x: Int): Unit = {
6         //printing the square
7         println("Square of added number is = " + x*x)
8     }
9     //A Partial function to add two integers passed during runtime with one pre-defined integer & returning the sum as Int
10    def partAdd(x: Int, y: Int, z: Int): Int = x+y+z
11    //Adding two dynamic & one static values & saving result in sum_val variable
12    val sum_val = partAdd(5, _, _)
13
14    //A function to call both the functions using values passed from main method.
15    def partialFunc(a: Int, b: Int): Unit = {
16
17        //Printing sum of all the the values adding a & b with the constant input.
18        println("Addition all the three numbers we get = " + sum_val(a,b))
19        //Calling the squareFun function & passing returned value of partial function as arguments to it.
20        squareFun(sum_val(a,b))
21    }
22 }
```

Run: partialFun

Enter the value of the numbers:

8
9

Addition = 22
Squares = 289

```
1 // singleton object to call the functions
2 object partialFun
3 {
4     //Square function that takes an integer & returns square of that integer.
5     def squareFun(x: Int): Unit = {
6         //printing the square
7         println("Square of added number is = " + x*x)
8     }
9     //A Partial function to add two integers passed during runtime with one pre-defined integer & returning the sum as Int
10    def partAdd(x: Int, y: Int, z: Int): Int = x+y+z
11    //Adding two dynamic & one static values & saving result in sum_val variable
12    val sum_val = partAdd(5, _, _)
13
14    //A function to call both the functions using values passed from main method.
15    def partialFunc(a: Int, b: Int): Unit = {
16
17        //Printing sum of all the the values adding a & b with the constant input.
18        println("Addition all the three numbers we get = " + sum_val(a,b))
19        //Calling the squareFun function & passing returned value of partial function as arguments to it.
20        squareFun(sum_val(a,b))
21    }
22 }
```

Run: partialFun

Enter the value of the numbers:

12
9

Addition all the three numbers we get = 21
Square of added number is = 441

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.

Ans:

Note: Program files (attached separately) are properly documented with a detailed description of each instruction used within the program.

ScreenShot:

The screenshot displays the IntelliJ IDEA IDE with the `courseChoice.scala` file open. The file defines a `courseMatch` function using a `match` expression to handle different course names and their prices. The `main` function calls `courseMatch` for four specific courses. The output window at the bottom shows the execution results for each course name.

```
object courseChoice
{
  //Defining a method courseMatch that takes String as input parameter to match against predefined cases
  def courseMatch(courseName:String):Unit = courseName match
  {
    case "Android App Development" => println("Price for AAD course is : 14,999 INR")
    case "Data Science" => println("Price for DS course is : 49,999 INR")
    case "Big Data Hadoop & Spark Developer" => println("Price for BD & Spark course is : 24,999 INR")
    case "Blockchain Certification" => println("Price for BCC course is : 49,999 INR")
    //The Default case if input does not match any of the above cases.
    case _ => println("This course is yet to come")
  }
}

// Entry point of the application
def main(args:Array[String]): Unit ={
  //Calling the course match function for different set of inputs.
  courseMatch( courseName = "Android App Development")
  courseMatch( courseName = "Data Science")
  courseMatch( courseName = "Big Data Hadoop & Spark Developer")
  courseMatch( courseName = "Blockchain Certification")
  courseMatch( courseName = "Teleporting Basics")
}
```

Run: courseChoice

```
Files\JetBrains\IntelliJ IDEA Community Edition 2018.1.5\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\ssahay\IdeaProjects\courseChoice\target\scala-2.12\
C:\Users\ssahay\.sbt\boot\scala-2.12.6\lib\scala-library.jar courseChoice
Price for AAD course is : 14,999 INR
Price for DS course is : 49,999 INR
Price for BD & Spark course is : 24,999 INR
Price for BCC course is : 49,999 INR
This course is yet to come
```

End
