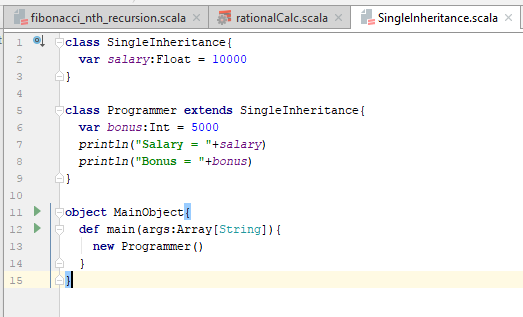
**ASSIGNMENT-15.1**

1. Single Inheritance in Scala.

Inheritance is an object oriented concept which is used to reusability of code. You can achieve inheritance by using extends keyword. To achieve inheritance a class must extend to other class. A class which is extended called super or parent class. a class which extends class is called derived or base class.

Screenshot below:

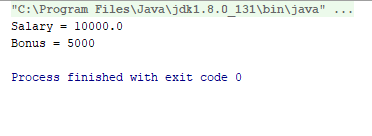


In the above program,

Class Single Inheritance contains a variable assignment, say “salary = 10000”. As , the class programmer extends SingleInheritance, it is able to access the value assigned for variable salary.

In the main, class Programmer is invoked, which prints the result of variables declared in both SingleInheritance and Programmer.

**Result Screenshot:**

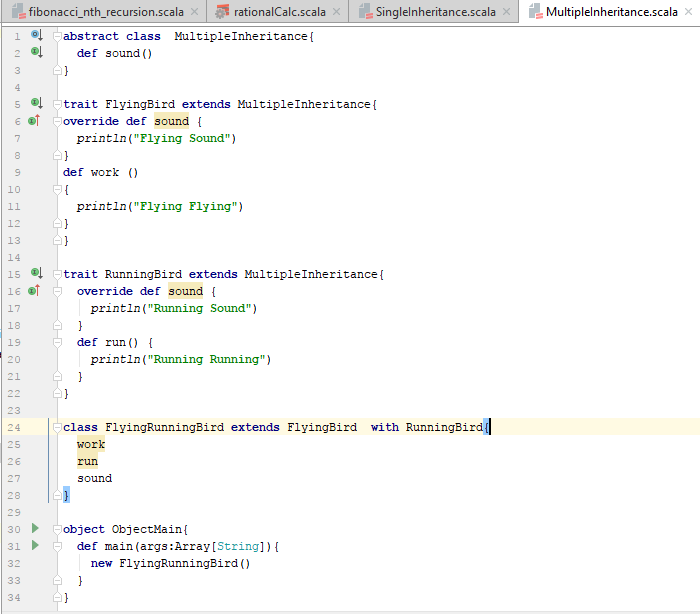
****

1. Write a simple program to show multiple inheritance in scala.

Scala uses scala traits and mixin concept to implement multiple inheritance.

Trait a new construct defined in Scala that lies halfway between an interface and a class. Traits are unusual in that a class can incorporate as many of them as desired, like interfaces, but they can also contain behaviour, like classes. In some way we can assume trait as abstract class with the additional property that is used in multiple inheritance. Also, like both classes and interfaces, traits can introduce new methods.

Screenshot below:



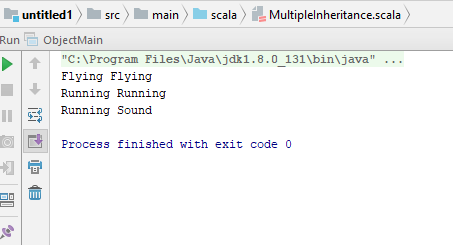
In the above example, Multiple Inheritance contains a method definition, which is implemented by extending the class by a trait.

Each of the method overrides the definition as per requirement. In the final invocation,

**class** FlyingRunningBird **extends** FlyingBird **with** RunningBird{  
 work  
 run  
 sound  
}

Here, the work and run implementation is present in only one of the above classes, whereas sound is overridden by all the classes. It would consider the final class after with keyword (as highlighted above) for implementation. Hence, “Running Sound ” is printed.

Below is the result screenshot:

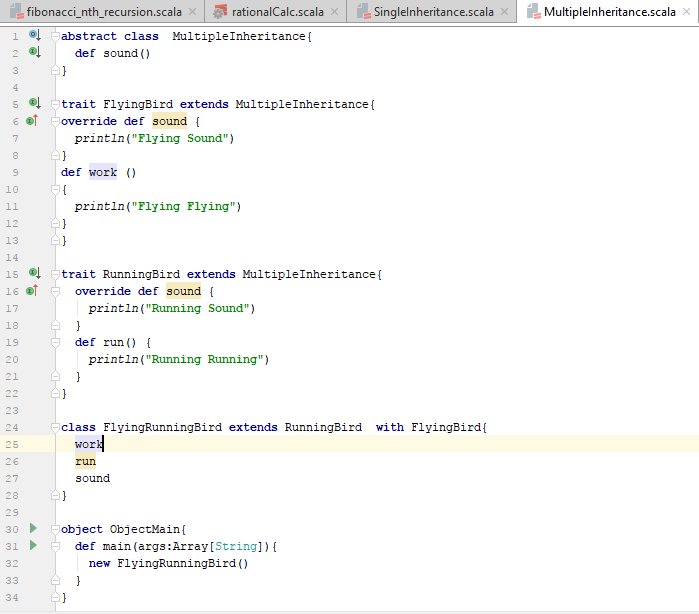


* The other way round to check the working of sound method:

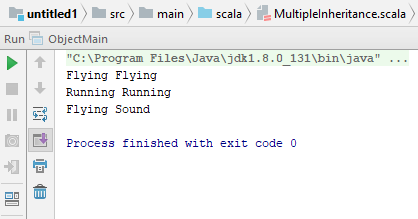
If the class extension is like the below example:

**class** FlyingRunningBird **extends RunningBird with** FlyingBird

Below is the program Screenshot:



RESULT:



*Attached are the files for reference:*

**