

Scala Assignment 15.1

For this assignment, I've used IntelliJ Idea scala.

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

Scala provides two structures for inheritance. Classes (abstract or not) and traits. Classes in Scala can extend other classes using the extends keyword. The overriding aspects are a bit more detailed compared to Java, because in Scala there are not just methods to override but also vals, vars.

There are a few restrictions added in Scala like:

- Overriding classes must use the “override” modifier. In Java one can use the @Override annotation to enforce correct overriding

Inheriting a class :

Created a class parent and created a class child which extends parent as below

```
class parent{  
}  
class child extends parent{  
}
```

Overriding a val:

Overriding val in the parent class, as below

```
class parent{  
  val age = 40  
}  
//Child class inheriting the parent class  
class child extends parent{  
  override val foo = 8  
}
```

Overriding a method :

In the below parent class 2 methods/functions are there, we want to override and add child specific implementation for method work(), we use override keyword before def as below in the child method, which is overridden. We can also create new methods in child class.

```
//super class parent  
class parent{  
  val age = 40  
  def work(): Unit = {  
    println("Parent working")  
  }  
  def earn(): Unit = {  
    println("Parent earning")  
  }  
}
```

```

}
//Child class inheriting the parent class
class child extends parent{
  override val age = 8

  override def work(): Unit = {
    println("child studing");
  }
  def play(): Unit = {
    println("child playing")
  }
}

```

Running the above class and testing :

```

object assignement_15_1 {
  def main(args: Array[String]) {
    var p = new parent();
    var child = new child();

    println("child age : " + child.age);
    print("child work: " + child.work())
    print("child earn: " + child.earn())
    print("child earn: " + child.play())

    println("parent age: " + child.age);
    print("parent work: " + p.work())
    print("parent earn: " + p.earn())
  }
}

```

Output printed in the console:

```

child age : 8

child studing

child work: ()Parent earning

child earn: ()child playing

child earn: ()parent age: 8

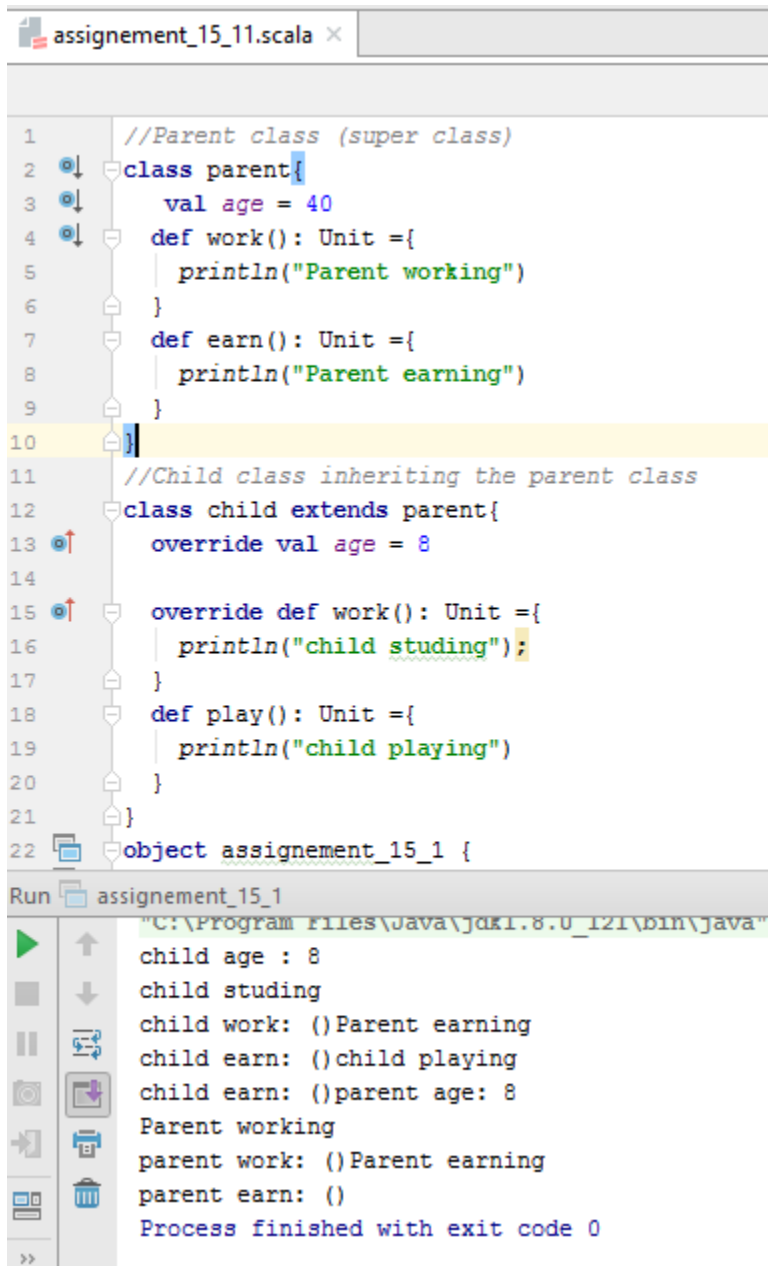
Parent working

parent work: ()Parent earning

parent earn: ()

```

Screenshot of IntelliJ Inheritance



The screenshot displays the IntelliJ IDEA interface with a Scala file named `assignment_15_11.scala`. The code defines a `parent` class and a `child` class that inherits from it. The `parent` class has a `val age = 40` and two methods: `work()` and `earn()`, both returning `Unit` and printing messages. The `child` class overrides `age` to 8 and adds a `play()` method. An `assignment_15_1` object is created. The bottom panel shows the execution output, which includes the output of the `child` object's methods followed by the `parent` object's methods, demonstrating method resolution and inheritance.

```
1 //Parent class (super class)
2 class parent{
3     val age = 40
4     def work(): Unit ={
5         println("Parent working")
6     }
7     def earn(): Unit ={
8         println("Parent earning")
9     }
10 }
11 //Child class inheriting the parent class
12 class child extends parent{
13     override val age = 8
14
15     override def work(): Unit ={
16         println("child studing");
17     }
18     def play(): Unit ={
19         println("child playing")
20     }
21 }
22 object assignment_15_1 {
23
24 }
25
26 Run assignment_15_1
27 "C:\Program Files\Java\jdk1.8.0_121\bin\java"
28 child age : 8
29 child studing
30 child work: ()Parent earning
31 child earn: ()child playing
32 child earn: ()parent age: 8
33 Parent working
34 parent work: ()Parent earning
35 parent earn: ()
36 Process finished with exit code 0
```

Source code: Inheritance

```
//Parent class (super class)
class parent{
    val age = 40
    def work(): Unit ={
        println("Parent working")
    }
    def earn(): Unit ={
        println("Parent earning")
    }
}
//Child class inheriting the parent class
```

```

class child extends parent{
  override val age = 8

  override def work(): Unit ={
    println("child studying");
  }
  def play(): Unit ={
    println("child playing")
  }
}
object assignement_15_1 {
  def main(args: Array[String]) {
    var p = new parent();
    var child = new child();

    println("child age : " + child.age);
    print("child work: " + child.work())
    print("child earn: " + child.earn())
    print("child earn: " + child.play())

    println("parent age: " + child.age);
    print("parent work: " + p.work())
    print("parent earn: " + p.earn())
  }
}

```

2) Write a simple program to show multiple inheritance in scala..

Scala uses trait to achieve multiple inheritance. Traits are used to share interfaces and fields between classes. They are similar to Java 8's interfaces. Classes and objects can extend traits but traits cannot be instantiated and therefore have no param

We have to use the extends keyword to extend a trait. Then implement any abstract members of the trait using the override keyword

Defining a trait:

Created a trait as below

```

trait Drawable {
  def draw
  def paint
}

```

Extending a trait:

We can extend the Drawable trait as below. It can be extended by a trait or a class. Here, Artist and Cartoonist inherit Drawable and override the methods draw and paint.

```

trait Cartoonist extends Drawable {
  override def draw() { println("Bang!") }
  override def paint() { println("I need this to be painted")}
}
trait Artist extends Drawable {
  override def draw() { println("A pretty painting") }
  override def paint() { println("I don't want this to be painted")}
}

```

Multiple Inheritance:

The below class, CartoonistArtist inherits both Cartoonist and Artist in the same order. To extend more than one class we have to use extends with keyword.

```
//child class extending from 2 parent class
class CartoonistArtist extends Cartoonist with Artist{
}
```

When we tried to excute the draw() method and paint() method by creating an instance of CartoonistArtist.The output is as below

```
object assignment_15_12 extends App {
  (new CartoonistArtist()).draw() // A pretty painting
  (new CartoonistArtist()).paint() // I don't want this to be painted
}
```

Output printed in the console:

A pretty painting

I don't want this to be painted

It took values from the Artist trait, because, when same method exists in 2 parent classes, Scala determines which method to be called based on the order in which its inherited and the methods from the right most trait would be executed.

Implementing methods from Different trait

To Call different methods from different traits in scala with multiple inheritance, We can use override keyword in the child (CartoonistArtist) class, If we want one of the method to be called from the one trait and another method from another trait,

For example, I want the draw method to be executed from Artist trait but I want the paint method to be executed from the Cartoonist trait.Then in the CartoonistArtist, class

```
class CartoonistArtist extends Cartoonist with Artist{
  override def paint = super[Cartoonist].paint
}

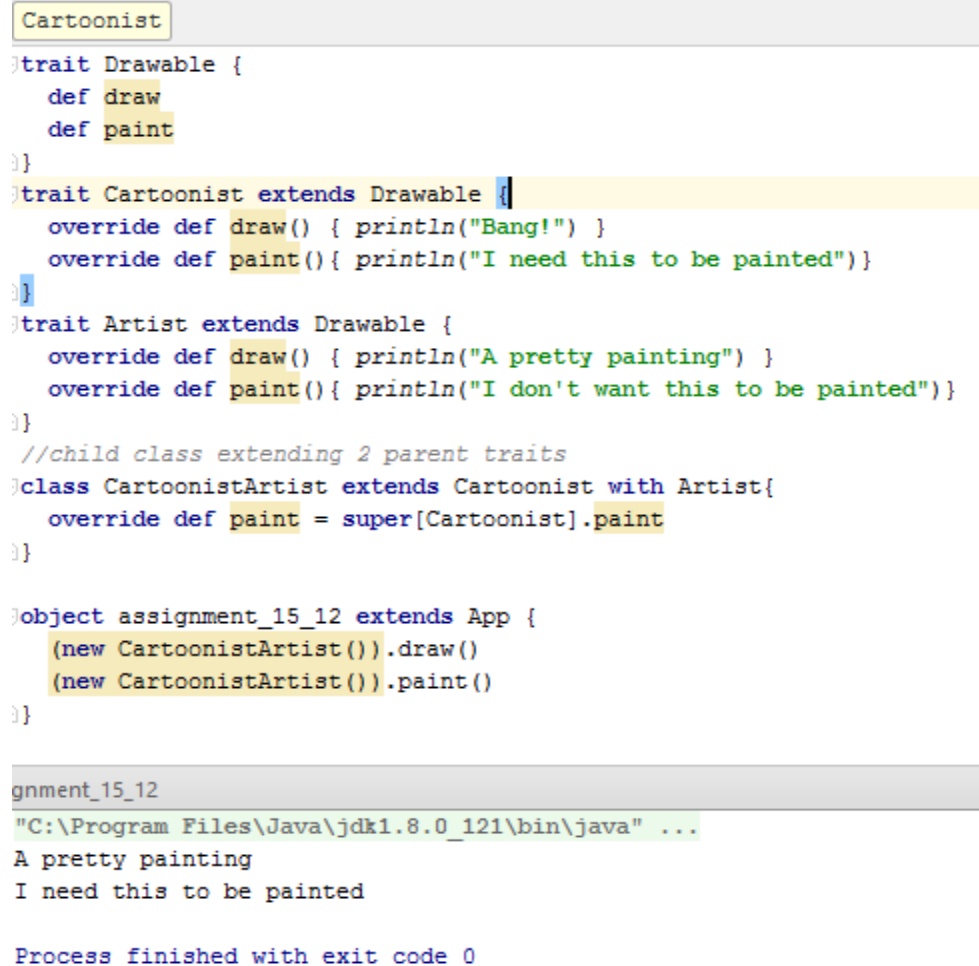
object assignment_15_12 extends App {
  (new CartoonistArtist()).draw() // A pretty painting
  (new CartoonistArtist()).paint() //I need this to be painted
}
```

Output printed in the console:

A pretty painting

I need this to be painted

Screenshot of IntelliJ and output:



The screenshot shows the IntelliJ IDE with a Scala file named 'Cartoonist'. The code defines a 'Drawable' trait with 'draw' and 'paint' methods, a 'Cartoonist' trait extending 'Drawable' with overridden methods, an 'Artist' trait extending 'Drawable' with overridden methods, and a 'CartoonistArtist' class extending both 'Cartoonist' and 'Artist'. An object 'assignment_15_12' extends 'App' and calls 'draw' and 'paint' on a 'CartoonistArtist' instance. The output window shows the execution results: 'A pretty painting' and 'I need this to be painted', followed by 'Process finished with exit code 0'.

```
trait Drawable {
  def draw
  def paint
}

trait Cartoonist extends Drawable {
  override def draw() { println("Bang!") }
  override def paint() { println("I need this to be painted")}
}

trait Artist extends Drawable {
  override def draw() { println("A pretty painting") }
  override def paint() { println("I don't want this to be painted")}
}

//child class extending 2 parent traits
class CartoonistArtist extends Cartoonist with Artist{
  override def paint = super[Cartoonist].paint
}

object assignment_15_12 extends App {
  (new CartoonistArtist()).draw()
  (new CartoonistArtist()).paint()
}
```

gnment_15_12

"C:\Program Files\Java\jdk1.8.0_121\bin\java" ...

A pretty painting

I need this to be painted

Process finished with exit code 0

Source code for Multiple Inheritance:

```
trait Drawable {
  def draw
  def paint
}

trait Cartoonist extends Drawable {
  override def draw() { println("Bang!") }
  override def paint() { println("I need this to be painted")}
}

trait Artist extends Drawable {
  override def draw() { println("A pretty painting") }
  override def paint() { println("I don't want this to be painted")}
}

//child class extending 2 parent traits
class CartoonistArtist extends Cartoonist with Artist{
  override def paint = super[Cartoonist].paint
}

object assignment_15_12 extends App {
  (new CartoonistArtist()).draw()
  (new CartoonistArtist()).paint()
}
```