Scala Abstract Class

A class which is declared with abstract keyword is known as abstract class. An abstract class can have abstract methods and non-abstract methods as well. Abstract class is used to achieve abstraction. Abstraction is a process in which we hide complex implementation details and show only functionality to the user.

In scala, we can achieve abstraction by using abstract class and trait. We have discussed about these in detail here.

Scala Abstract Class Example

In this example, we have created a Bike abstract class. It contains an abstract method. A class Hero extends it and provides implementation of its run method.

A class that extends an abstract class must provide implementation of its all abstract methods. You can't create object of an abstract class.

```
abstract class Bike{
   def run()
}

class Hero extends Bike{
   def run(){
      println("running fine...")
   }
}

object MainObject{
   def main(args: Array[String]){
      var h = new Hero()
```

```
        h.run()
    }
}
```

Output:

```
running fine...
```

Scala Abstract Class Example: Having Constructor, Variables and Abstract Methods

```
abstract class Bike(a:Int){            // Creating constructor
    var b:Int = 20                     // Creating variables
    var c:Int = 25
    def run()                          // Abstract method
    def performance(){                 // Non-abstract method
        println("Performance awesome")
    }
}

class Hero(a:Int) extends Bike(a){
    c = 30
    def run(){
        println("Running fine...")
        println("a = "+a)
        println("b = "+b)
        println("c = "+c)
    }
}
```

```
object MainObject{
   def main(args: Array[String]){
      var h = new Hero(10)
      h.run()
      h.performance()
   }
}
```

Output:

```
Running fine...
a = 10
b = 20
c = 30
Performance awesome
```

Scala Abstract Class Example: Abstract Method is not implemented

In this example, we didn't implement abstract method run(). Compiler reports an error during compilation of this program. Error message is given below in output section.

```
abstract class Bike{
   def run()             // Abstract method
}

class Hero extends Bike{       // Not implemented in this class
   def runHero(){
      println("Running fine...")
   }
```

```
}


object MainObject{
   def main(args: Array[String]){
      var h = new Hero()
      h.runHero()
   }
}
```

Output:

```
error: class Hero needs to be abstract, since method run in class Bike of type ()Unit is not defined
class Hero extends Bike{
    ^
one error found
```