**Roles and Uses of Modules**

**Introduction:**

Since the beginning of Ruby Programming language, it has gained so much of popularity with time. It is the basis of its great framework Rails. It is basically an Object Oriented Programming language which is based upon various components such as classes, object, variables etc. It also provides a nice building block for applications which is known as module. A module is a collection of methods and constants. It defines a namespace in which other methods and constant can’t step on your methods and constants.

**Purpose of a module:**

Ruby module is a component to regroup similar things. Ruby methods, classes and constants can be grouped by similarity with the help of modules. Ruby modules provide two benefits:

* Modules provide ‘namespace’, it basically helps to prevent name clashes.
* Ruby’s mixin facility is implemented with the help of modules.

The basic syntax of module is:

*module Identifier*

*statement1*

*statement2*

*...........*

*End*

**Uses:**

Ruby module mainly functions as a namespace. It lets us define various methods for the actions that will perform. When a method which are defined inside a module will not be clashed with other methods that are written anywhere else, though they are having the same names.

Module constants are named like class constants with an initial uppercase letter. Module methods are also defined like class methods. Here is an example.

*module MyModule*

*def method*

*puts “hello”*

*end*

*end*

To access the methods and constants inside a module in a class *include* key word is used.

*class Customer < ActiveRecord::Base*

*include MyModule*

*end*

To use the method that is defined inside a module, specify the module name followed by a dot and then the method name.

**Ruby Mixins and Ruby Modules:**

Ruby is purely an Object Oriented Programming language. But it does not support multiple inheritance directly, which is handled beautifully by Modules. They provide a facility called mixin which eliminate the requirement of multiple inheritance. In Ruby when mixins are called and used in proper manner they provide high degree of versatility functionality.

A module mixin generally consists of several lines of codes to set up conditions where the module can mix in with a class or classes to improve the functionality of the class or itself too. Here is an example of a module mixin.

*module A*

*def a1*

*end*

*def a2*

*end*

*end*

*module B*

*def b1*

*end*

*def b2*

*end*

*end*

*class MyClass*

*include A*

*include B*

*def s1*

*end*

*end*

*obj = Objet.new*

*obj.a1*

*obj.a2*

*obj.b1*

*obj.b2*

*obj.s1*

Here module A and B are consists of two methods individually. They are included in the classMyClass. Now MyClass can access all the four methods a1,a2,b1,b2. So it can be said that Myclass inherits from multiple modules. Thus multiple inheritance is implemented with the help of module’s mixin facility.

**Conclusion:**

Modules don’t have any direct analogy in any mainstream programing language. They are used mostly in Ruby language. They are one of the key feature making Ruby’s design beautiful. With the facility of namespacing and mixin modules make Ruby more flexible Object Oriented Programming language. They also make the application highly secure as classes and their objects inherit from modules and modules are having mixins.