RUBY CLASS CASE STUDY

For your case study, you will create a Ruby Class called Customer and you will declare two methods:

- *display_details*: method will display the details of the customer.
- *total_no_of_customers*: method will display the total number of customers created in the system.

```
#!/usr/bin/ruby

class Customer
   @@no_of_customers=0
   def initialize(id, name, addr)
      @cust_id=id
      @cust_name=name
      @cust_addr=addr
   end
   def display_details()
      puts "Customer id #@cust_id"
      puts "Customer name #@cust_name"
      puts "Customer address #@cust_addr"
    end
    def total_no_of_customers()
       @@no_of_customers += 1
       puts "Total number of customers: #@@no_of_customers"
    end
end
```

The *display_details* method contains three puts statements displaying the Customer ID, the Customer name, and the Customer address. The puts statement:

```
puts "Customer id #@cust_id"
```

will display the text Customer id followed by the value of the variable @cust_id in a single line.

When you want to display the text and the value of the instance variable in a single line, you need to precede the variable name with the hash symbol # in the puts statement. The text and the instance variable along with the hash symbol # should be enclosed in double quotation marks.

The second method, total_no_of_customers, is a method that contains the class variable @@no_of_customers. The expression @@no_of_ customers+=1 adds 1 to the variable no_of_customers each time the method total_no_of_customers is called. In this way, you will always have the total number of customers in the class variable.

Now create two customers as follows:

```
cust1=Customer.new("1", "John", "Wisdom Apartments, Ludhiya")
cust2=Customer.new("2", "Poul", "New Empire road, Khandala")
```

Here, we create two objects of the Customer class as cust1 and cust2 and pass the necessary parameters with the new method. The initialize method is invoked, and the necessary properties of the object are initialized.

Once the objects are created, you need to call the methods of the class by using the two objects. If you want to call a method or any data member, you write the following:

```
cust1.display_details()
cust1.total_no_of_customers()
```

The object name should always be followed by a dot, which is in turn followed by the method name or any data member. We have seen how to call the two methods by using the cust1 object. Using the cust2 object, you can call both methods as shown below:

```
cust2.display_details()
cust2.total_no_of_customers()
```

Save and Execute the Code:

Now, put all this source code in main.rb file as follows:

```
#!/usr/bin/ruby

class Customer
   @@no_of_customers=0
   def initialize(id, name, addr)
      @cust_id=id
      @cust_name=name
      @cust_addr=addr
   end
   def display_details()
      puts "Customer id #@cust_id"
      puts "Customer name #@cust_name"
      puts "Customer address #@cust_addr"
   end
   def total_no_of_customers()
      @@no_of_customers += 1
      puts "Total number of customers: #@@no_of_customers"
   end
end

# Create Objects
cust1=Customer.new("1", "John", "Wisdom Apartments, Ludhiya")
cust2=Customer.new("2", "Poul", "New Empire road, Khandala")

# Call Methods
cust1.display_details()
cust1.total_no_of_customers()
cust2.display_details()
cust2.total_no_of_customers()
```

Now, run this program as follows:

```
$ ruby main.rb
```

This will produce the following result:

```
Customer id 1
Customer name John
Customer address Wisdom Apartments, Ludhiya
Total number of customers: 1
Customer id 2
Customer name Poul
Customer address New Empire road, Khandala
Total number of customers: 2
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js