COMP348 PRINCIPLES OF PROGRAMMING LANGUAGES

LISP: TUTORIAL - 8

TOPIC : RUBY

INHERITANCE

Exercise #9:

Consider class Person which defines the name and age of a person

```
class Person
    attr_accessor :name,:age
    @@total=0

    def initialize (name, age)
        @@total+=1
        @name=name
        @age=age
    end

    def to_person
        return "(#@name, #@age)"
    end

    def Person.total
        return "number of persons: #@@total"
    end
end
```

INHERITANCE

Consider class Student which defines the school of a Person

```
require " ./Person"
class Student < Person
    attr_accessor :school
    @@newtotal=0

    def initialize(name,age,school)
        super(name,age)
        @school=school
        @@newtotal+=1
    end

    def to_student
        return "(#@name, #@age, #@school)"
    end

    def Student.total
        return "number of  students: #@@newtotal"
    end
end
```

INHERITANCE

1. Use VIM on Linux to create class Person and class Student
2. Add following statements into the file,

```
require " ./Person"
require " ./Student"
if __FILE__== $0
        p1 = Person.new("test1",25)
        puts p1.name
        puts p1.age
        puts p1.to_person
        puts " “

        s1 = Student.new("test2",26,“Concordia")
        puts s1.name
        puts s1.age
        puts s1.to_student
        puts " “

        puts Person.total
        puts Student.total
End
```

3. save the file as “**inher.rb**”.
4. Execute “**ruby inher.rb**” to check the result

ITERATOR

Exercise #10:

Consider a array a=[“test”, “if”, “it”,”is”, “working”, “fine”], use keyword “each” to display each elements in this array

Exercise #11:

Consider a array a=[“test”, “if”, “it”,”is”, “working”, “fine”], use keyword “collect” to desplay each elements in this array

Exercise #12:

Consider a array a=[“test”, “if”, “it”,”is”, “working”, “fine”], use keyword “find” to find the element “if” and display this element

REGULAR EXPRESSIONS

Exercise #13:

Use the file “inher.rb”we created in Exercise#9, and define a pattern to display lines with a string containing one or more cs, followed by a or e or l.

Reflection

Exercise #13:

Use classes defined as Person and Student, execute the following commands in ruby interaction console.

Execute:

```
load "./inher.rb"
p1=Person.new("Alex",20)
s1=Student.new("Stephen",21, "Concordia")
ObjectSpace.each_object(Person){|p| puts p.inspect}
ObjectSpace.each_object(Student){|p| puts p.inspect}
```

Reflection (Cont.'s)

Exercise #14:

Based on Exercise13, find out the results of executing following commands

Execute	Results
puts p1.respond_to?(“name”)	
puts p1.respond_to?("setName")	
puts p1.class	
puts s1.class	
print s1.instance_variables	
puts p1.kind_of? Person	
puts p1.kind_of? Student	
puts p1.instance_of? Person	
puts Person.class_variables	
puts p1.instance_variables	

Module

Exercise #15:

Consider following module Week in Ruby where you will define FIRST_DAY = “Sunday” , a method called weeks_in_month which will output "You have four weeks in a month” and another method weeks_in_year which will output "You have 52 weeks in a year”

Now consider a class Decade which will call this module. In addition it will calculate number of months in the method no_of_months for a given number of years.

Solution

```
module Week
  FIRST_DAY = "Sunday"
  def Week.weeks_in_month
    puts "You have four weeks in a month"
  end
  def Week.weeks_in_year
    puts "You have 52 weeks in a year"
  end
end
```

Module (Cont.'s)

```
class Decade
    include Week
   def no_of_months
        puts Week::FIRST_DAY
        number=10*12
        puts number
   end
End
```

Execute:

d1=Decade.new

puts Week::FIRST_DAY

Week.weeks_in_month

Week.weeks_in_year

d1.no_of_months