**164 Lab2**

2023 spring

1. Following are two class definitions (Push class) in two libraries.

gym.rb  
  
  
dojo.rb

class Push

def up

40

end

end

class Push

def up

30

end

end

They are loaded into a ruby program.

require "gym"

gym\_push = Push.new

p gym\_push.up # up returns 40

require "dojo"

dojo\_push = Push.new

p dojo\_push.up # up returns 30

As the dojo library is loaded after gym, it has overridden gym's class definition of Push and therefore creates an instance of Push defined in dojo.

Use modules to solve this problem so that both Push class can be used to create different objects for these two different classes.

1. When you subclass, the instance method will be inherited. However, the "initialize" method in the parent class might not be called automatically if the subclass has "initialize" itself.

Read the following code:

class Parent

def initialize(name="nobody")

@name = name

end

end

class Child < Parent

attr\_accessor :name, :grade

def initialize(name, grade)

@grade = grade

end

end

y = Child.new("yuan", 100)

print "name is: ", y.name

puts

puts y.grade

y.grade = 90

puts y.grade

Add a statement to make the Child object be able to initialize the "@name" attribute.

1. Constant lookup  
     
   Following is a code with some constants:

module Dojo

A = 4

module Kata

B = 8

module Roulette

class ScopeIn

def push

15

end

end

end

end

end

A = 16

B = 23

C = 42

Print all regular constants (other than class/module names), and create an object to call   
 “push()” method to print 15.

1. The following is a module definition

module Greetings

def english

puts "Hello!"

end

def french

puts "Bonjour!"

end

def spanish

puts "Hola!"

end

end

a. Define a class ‘Hello”. The object of this class should be able to call #english, #french,   
 #spanish instance method to output different languages "hello"

For example, if "hello" is an object of Hello, then

hello.spanish

=> Hola!

b. Modify the class ‘Hello’ so that you can call these methods directly,

For example:

Hello.spanish

=> Hola

1. For each of the following regular expression, give 2 examples of matching string (unless it defines a language that has less then 2 strings), use =~ in IRB to verify the match.

/hello/

/love | hate/

/colou?r/

/gr[ae]y/

/b[aeiou]bble/

/go\*gle/

/go+gle/

/g(oo)+gle/

/x{3}/

/x{6,10}/

/w/

/\w/

/d/

/\d/

/\d{5}/

/\d+(\.\d\d)?/

/hello\d+/

/sh[^io]t/

/^ruby/

/ruby$/

/^ruby$/

1. Find and substitute:

string = “abc12def34ghi56jkl78mn98op76qrs”

* + - * replace all number with “-“
      * output each number
      * print the first number
      * replace all non digit char with “-“

1. Implement the Array#map method and demonstrate it the array [1,2,3,4,5].

Here is an example output given that you called map on the array:

1 - hello

2 - hello

3 - hello

4 - hello

5 - hello

[3, 6, 9, 12, 15]

1. Check out the following regular expressions (as many as you can depend on your time), try to understand it. (No submission for this).

"one,two-three".split(/,|-/)

"12345".scan /\d/

"12345".scan /\d/ do |i|

puts i

end

"hiho hiho".scan /(hi)(ho)/

"hiho" =~ /hi/

"hiho" =~ /ho/

/hi/ === "hiho"

"(123) 456-7890".match /\(\d{3}\) \d{3}-\d{4}/

/\(\d{3}\) \d{3}-\d{4}/.match "(123) 456-7890"

str1 = "Joe Schmo, Plumber"

str2 = "Stephen Harper, Prime Minister"

re = /(\w\*)\s(\w\*),\s?([\w\s]\*)/

match1 = str1.match re

match2 = str2.match re

match1[1]

match1[2]

match1[3]

match1[4]

match2[1]

match2[2]

match2[3]

$1

$2

$3

match1.regexp

"some string".sub /string/, "message"

"The man in the park".gsub /the/, "a"

"The man in the park".gsub( /the/, "a")

original = "My name is Andrew."

new = original.sub /My name is/, "Hi, I'm"

puts original

puts new

original = "Who are you?"

original.sub! /Who are/, "And"

puts original

"WHAT'S GOING ON?".gsub(/\S\*/) {|s| s.downcase }

"WHAT'S GOING ON?".gsub(/\S\*/) {|s| puts s }

s = "i love you very much"

s.scan(/\W+/)

s.scan(/\w/)

s.scan(/\w+/)

s.scan(/\w\*/)

s.scan(/…/)

s.scan(/(…)/)

s.scan(/(..)(..)/)

s.scan(/\w+/) {|w| puts w}

s.scan(/(..)(..)/) {|a,b| print a,'-', b, '-'}

/abc/ =~ "xxxabc"

/abc/ =~ "xxxxxc"

/[abc]/ =~ "abc"

/\d\d:\d\d AM/ =~ "07:10 AM"

/\d\d:\d\d AM/ =~ "---07:10 AM"

names = "erik kalle johan anders erik kalle johan anders"

names[/kalle/]

String1 = "<name> <substring>"

"<name> <substring>".scan( /<([^>]\*)>/)

"<name> <substring>".scan( /<([^>]\*)>/).last

"<name> <substring>".scan( /<([^>]\*)>/).last.first

"<name> <substring>"[/.\*<([^>]\*)/,1]

"<name> <substring>".scan(/\w+/)

"<name> <substring>".scan(/\w+/)[0]

"<name> <substring>".scan(/\w+/)[1]

"<name> <substring>".scan(/\w+/) {|w| puts w}

"xxabcxxabcxxabcxxabc".scan(/abc/)

"xx1xx2xx3xx4".scan(/\d/)

m = /(?<this>\w+)\k<this>/.match("mississippi")

/(?<this>\w+)\k<this>/ =~ ("mississippi")

this