RUBY: Eclipse 🡪 file 🡪 new 🡪 Ruby project

Ruby interpreter must be configured.

Write something on console

Puts: this is used to print something on console and line switch

Print: to print something on console and no line switch

Prog 1 :

Puts “This is line 1 and line switch”

Print “this Is line 2 and no line switch”

Prog 2: Comments in ruby:

Single line comment : put # in starting

Multiple line comment : =begin”….”………”..”=end

RUBY GEM:

Ruby providing lot of libraries which can be installed and use while working with ruby, these libraries are called Ruby Gem

Ruby gems are similar to Jar file which we are getting in Java.

Each gen has name, version, platform(on which platform gem can work)

Command prompt checking: gem list - -local

Set up a new gem : gem install cucumber (Enter in command prompt)

Un install gem: gem uninstall cucumber (Enter in command prompt)

Download gem : gem fetch cucumber (Enter in command prompt)

RUBY PROGRAMMING ON NOTE PAD: and run in command prompt with out eclipse.

Open note pad:

Print “this is start of note pad ruby programing”

Print “ This is End of programming”

Puts “With line”

Save with somename.rb (Extension should be .rb)

To execute this code: 1. copy the location and give cd location path(cd space location path) in command prompt.

2. Ruby file name.rb

INTERACTIVE RUBY:

Ruby is a interactive language, were we can write one line of code and can execute it and c its result.

Start command prompt 🡪 type

1. irb (enter)
2. print “Hello this is ruby” (write command in console and result will be shown in console directly)
3. puts “Testing world”
4. 2+5 (Any calculation on console)
5. 3\*7
6. quit (come out of the interactive environment)

RUBY START AND END EXECUTION BLOCKS:

END{

puts “This is end of code”

}

Puts “This is my file code”

BEGIN{

Puts “This is starting code”

}

print “Hello”

Result : Every time it starts with BEGIN, ends with END block

1. This is starting code

2. This is my file code

3. HelloThis is end of code

CLASS IN RUBY:

1. Class can be defined as container of Constructors, methos, variables and constants
2. Class name always starts with capitol letter
3. Class body end with “end” keyword
4. Class members can be accessed by creating object of class

Right click on project and press on Ruby Class

Name of class “A1”

Program

Class A1

def hello //def is a key word to define method

Puts “This is Class Method - Hello”

end

end

we need to create a empty Ruby class “Call.rb” to create an object

require ‘A1’ // Importing A1

obj = A1.new

obj.hello()

METHODS/FUNCTIONS IN RUBY:

1. Methods can be defined as collection of reusable code
2. Methods can be called by creating a class object
3. No need of this brackets {} if no argument is there
4. Return keyword is used to return value

Prog 2:

Class A1

Def Hello #method 1 ( not taking any arg and not returning value)

Puts “This is Method –No arguments/parametets”

end

def sum(a,b) # method 2 (Taking some arguments but not returning value)

puts a+b

end

def sub(a,b) # method 3 (Taking some arguments and returning value)

c = a-b

return c

end

Obj class

Require ‘A1’

Obj = A1.new

Obj.hello()

Obj.sum(100,200)

X = obj.sub(100, 200)

Puts x

Result: This is method – no Arguments/parameters

300

-100

Write a prog for (20-30)\*(100 – 20)

CONSTRUCTOR:

1. Similar to method
2. Group of code, used to perform specific task at the start
3. Automatically called when object is created.

Prog 3: # is automatically called when we create object of the class

Class A1

Def initialize # with out arguments

Puts “This is my constructor”

End

Def initialize(a) # with arguments

Put a

end

Objct class

Require ‘A1’

Obj = A1.new

Obj = A1.new(10)

Result : This is my contructor

10

CONDITION HANDLING:

1. If – else
2. If elsif else

Class A1

Def checkEvenOdd(a)

If(a%2==0)

Puts “This is Even Number”

Else

Puts “This is Odd Number”

End

End

End

Object class

Require ‘A1’

Obj=A1.new

Obj.checkEvenOdd(10)

IF ELSIF ELSE

Class A1

=begin

if Number is less than 0 display its negative

if Number is zero, display its zero

if Nubmer is greater than zero, check even and odd number

== for comparision

&& for logical and

|| for logical or

=end

def checkEvenodd(a)

if(a<0)

puts “This is Negative Number”

elsif(a=0)

puts “This is Zero”

elsif(a%2==0)

puts “This is Even Number”

else

puts “This is Odd Number”

end

end

end

object class

require ‘A1’

obj = A1.new

obj.checkEvenOdd(1) # Give multiple options in the bracket for result

LOGICAL AND(&&) AND LOGICAL OR (||)

PROG

=BEGIN

NUMBER <0 OR NUMBER> 100 – INVALID ID NUMBER

Number 0 -30 Its failed

Number 31 -60 Its passed

Number 61 - 100 Its passed with grade1

== for comparision

&& for logical and

|| for logical or

==end

def checkGradingSystem(a)

if(a<0 || a>100)

puts “This is invalid Number”

elsif (a>=0 && a<=30)

puts “You are failed”

else (a>30 && a<=60)

puts “Passed with grade1”

end

end

end

Object class

Require ‘A1’

Obj = A1.new

Obj.checkGradingSystem(1) #enter multiple values

WHILE LOOP:

Rather than writing code various times we use loops

Prog 1:

I=1

While(i<=10)

Puts i

I=i+1

End

Result: 1, 2 3 4 5 6 7 8 9 10

Prog 2:

Class A1

Def displayTable(a)

I=1

While(i<=10)

Puts a\*i

I=i+1

End

End

Def displayReverseTable(a)

i=10

while(i>=1)

puts i\*a

end

End

Object class

Require ‘A1’

Obj = A1.new

Obj.displayTable(8)

Obj.displayReverseTable(7)

Result :

display 8th table

display reverse 7th table

FOR LOOP

Prog

Class A1

# … (Three dots) excluding last value

# .. (Two dots) including last value

def printTable(a)

for I in 1…10

puts i\*a

end

end

end

object classs

require ‘A1’

obj = A1.new

obj.printTable(100)

result : 100,200,300,400,500,600,700,800 ,900

ARRAY

1. ARRAY is used to hold multiple data
2. In ruby, array can hold data of different data type
3. Arr = Array.new(20)

Prog :

# case 1

Arr = array.new(5)

arr = [“Hello”, 23, “Test”, “Dev”, 55]

puts arr[3]

result : Dev

#case 2

Arr = array.new(5)

# case 1

arr = [“Hello”, 23, “Test”, “Dev”, 55]

arr[4] = “change the value of 55 which is index 4”

puts arr[4]

result : change the value of 55 which is index 4

#case 3

Arr = Array.new(5)

Arr1 = Array.new(3)

# case 1

arr = [“Hello”, 23, “Test”, “Dev”, 55]

arr[4] = “change the value of 55 which is index 4”

# case 2

arr1[0]=””MyTesting”

arr1[2]=34

arr1[1]=23.33

puts arr1[1]

result :23.33

FOR EACH LOOP

Can be used with multiple data like array.

Prog

Arr1= Array.new(5)

Arr2 = Array.new(3)

# case 1

arr1= [“Hello”, 23, “Test”, “Dev”, 55]

arr1[4] = “change the value of 55 which is index 4”

arr1.each do |i|

puts i

end

result : hello,

23,

test,

dev,

change the value of 55 which is index 4

HASHES

PROG:

Hash1 = hash.new

#Case1

hash1 ={“K1”=>”Val1”, “K2”=>23, 21 +> “Testing”}

puts hash1.length() # find length of hash

puts hash1[“K1”] # pick data from hash

# add one more key for this length

hash1[“K4”]=”Added one more key”

puts hash1.length # Finding length

puts hash1[“K4”] # Picking k4

Result :

3

Val1

4

Added one morekey

STRING HANDLING

WE can declare string in 3 ways

1st way.

Str = ”Hello testing”

2nd way

str1 = String.new

str1 = “Hellooooo”

3rd way :

str2 = String.new(“Hello world”)

put str

result: Hello testing

prog

str = “Hello Testing”

puts str.upcase( ) # display string intp upper case

puts str.downcase() # display string intp lower case

puts str.length() # giving length of the string

puts str + india # concatenated (hello testing india)

puts str.concat(“India”) # concatenated (hello testing india)

str = “ Hello world “

puts str.lstrip().length() # remove space from left side

puts str.rstrip().length() # remove space from right side

puts str.strip().length() # remove all the spaces

puts str.reverse() # to reverse the value

# comparision of strings we can compare expected with actual result in testing

str = “hello”

str1= “Hello”

if(str.casecmo(str1)==0)

puts ‘’strings are equal’

else

puts “strings are miss matched”

end

TYPES OF VARIABLES IN RUBY

Local variable : we cannot use one method variable in another variable

Start with small letter or created inside a method can be usedinside method only

Globle variable :

Starts with $, can used in different class

Instance variable :

Starts with @, can b used any whee in a class for 1 object

Class variable:

Starts with @@, can b used anywhere in class for all class objects

EXCEPTION HANDLING:

IF WE GET ANY ERROR ON RUN TIME WHA ACTIONS WE NEED TO TAKE IS CALLED EXCEPTION HANDLING (10/0)

Prog:

Begin

a=10

b=0

puts a/b

rescue #for exception handling

puts ‘’ yes exception is there”

end

puts ‘’This is after exception Handling”

Result:

Yes exception is there

This is after exception handling

MODULES:

1. It’s a simple file with .rb extension
2. We can place ruby executable code here
3. We can create classes inside ruby module
4. We can create method/functions inside ruby module
5. When ever we use method name we cannot give simply as “sum” we need to give it as “moduleA.sum”

Create a new ruby project 🡪 right click on ruby project and create new ruby module

Module ModuleA

Puts “This is module excutable code”

Def ModuleA.sum(a,b) # when ever we create method in module class mention

Module.methodname

C=a+b

Puts c

End

Class Testing

Puts “This is constructor”

End

Def sub(a,b)

c=a-b

Puts c

end

end

end

create new empty ruby script

require ‘ModuleA’

ModuleA::sum(100,200)

Obj = ModuleA::Testing.new

Obj.sub(100,40)

OOPS CONCEPTS IN RUBY

INHERITANCE (<)

Parent class cannot call child method but child class can call both child and parent method

Constructor will be called only for its own constructor but all all parent classes.

Create a class A

Class A # add and sub methods

Def sum(a,b)

C=a+b

Puts c

End

Def sub(a,b)

C= a-b

Puts c

End

end

create class B

require A

class B #multiplication method

def mul(a,b)

c= a\*b

puts c

end

Create class C #display method

Require B

Classs C

Def display()

Puts “This is display method”

End

End

Create empty class Z to create object

Require ‘C’ # C is a child class which can call all the super classes A and B

Obj = c.new

Obj.display()

Obj.mul(10,20)

Obj.sum(10,20)

Obj.sub(10,20)

OVER RIDING

If parent and child class has same name and same signature method this is called overriding

A class has def mul(a\*b)

B class has def mul(a\*b)\*10

Prog:

Class A

Def mul(a,b)

C=a\*b

Puts c

End

Require ‘A’

Class B<A

Def mul(a,b)

C=a\*b\*10

Puts c

End

#Create empty object class

require ‘B’

obj=B.new

obj.sum(10,20)

obj.sub(2,3)

obj.mul(10,20) # executed from class B i.e “(a\*b\*10)”

result:

30

1

2000

READ/WRITE DATA

r: read only (starts at beginning of file)

w: write only (if the file exists, overwrites everything in the file)

w+: read and write (if the file exists, overwrites every thing in the file)

r+ : read write (if the file exists, starts at end of file. Other wise creates a new file)

Create a note pad and copy the path and file and paste it

Prog

How to Read data of the file in note pad

File = File.open(“C:\\ path of the file\\A1.txt”, ‘r’) # r = read

While !file.eof?

Puts file.readline()

End

How to write data to the file:

File = File.open(“testing.txt”, ‘w’) #w = write # create new .txt file

File.puts “Testing code”

File.close

READ DATA FROM EXCEL FILE

Start command prompt

Code: gem install spreadsheet

Eclipse🡪 windows🡪preference 🡪ruby 🡪interpreters🡪remove interpreters 🡪 adding them again which we installed 🡪 ok

Now we got spreadsheet libraries in to project

Prog

Create a spreadsheed excel and put some data into that so that we will get in to console

Require ‘spreadsheet’

Worksheet = spreadsheet.open(“c:/path/A1.xls”)

Worksheet = workbook.worksheet “sheet1”

Worksheet.each do |row| # running prog for all the rows

Puts “#{row[0]}” # for coloumn 0

Puts “#{row[1]}” # for coloumn 1

end

Write content to the excel sheet

Require ‘spreadsheet’

wk = Spreadsheet::Workbook.new

ws=wk.create\_worksheet

ws.name = “my sheet”

ws[1,1]=”Hello world”

wk.write(“Results.xls”)

Result: Hello world will excute in excel sheet on row 1 and coloumn 1

TEST DRIVEN DEVELOPMENT

VMWARE Instructor

SYSTEMS ADMINISTRATOR

Swapnica – id proof USA, opt expired.

Sanjay panocha: birth certificate,

Copy of any awards, appraisals, achievements or certification

Experience letters if any

Birth certificate if have

Sai ram: resume word, usa id proof,

Hitesh : ssn, id proof, pay stubs if any.