Ruby Lab Exercise – 2

Suryakumar P 21MIS1146

1. Method with and without Arguments:

Code:

#Method With Arguments

module Methods

    class MethodArg

        def sum(a,b)

            c = a+b

            puts "Sum of #{a} and #{b} is #{c}"

        end

    end

    class MethodNoArg

        def sum()

            a=10

            b=12

            c=30

            d = a+b+c

            puts "Sum of Non Argument Method is #{d}"

        end

    end

end

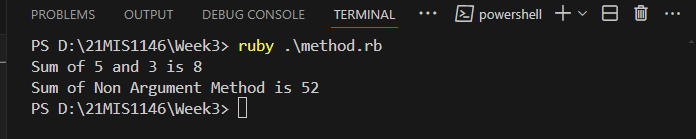
s1 = Methods::MethodArg.new

s2 = Methods::MethodNoArg.new

s1.sum(5,3)

s2.sum()

Output:



1. Create methods with two different symbols other than ‘?’ and ‘=’

Code:

class MethodSymbols

    def hello;

        puts "Hello Suryakumar"

    end

    def bye!(fname="Surya",lname="kumar")

        puts "Bye #{fname+lname}"

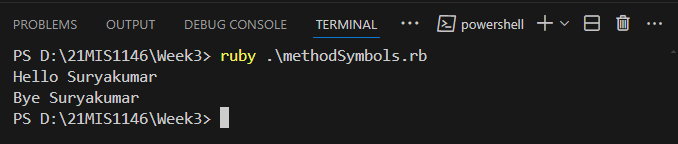
    end

end

name = MethodSymbols.new

name.hello;

name.bye!

Output:  


1. A Program using Instance Method

Code:

class Dog

    def initialize(name, breed)

      @name = name

      @breed = breed

    end

    def fetch(item)

      puts "#{@name} fetches the #{item}."

    end

    def bark

      puts "#{@breed} barks Woof woof!"

    end

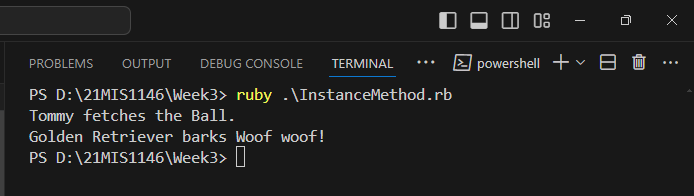
end

my\_dog = Dog.new("Tommy", "Golden Retriever")

my\_dog.fetch("Ball")

my\_dog.bark

Output:



1. Example for Class Method

Code:

class Circle

    PI = 3.14159

*# Class method to calculate the area of a circle*

    def self.area(radius)

      PI \* radius \* radius

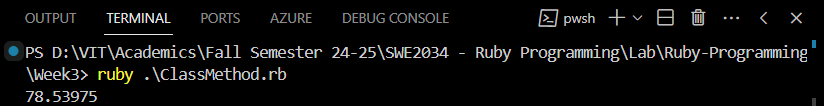
    end

  end

*# Calling the class method*

  puts Circle.area(5)

Output:



1. Bank Account System

Code:

class BankAccount

    attr\_reader :account\_number, :balance

    def initialize(account\_number, initial\_balance)

      @account\_number = account\_number

      @balance = initial\_balance

      puts "\t Welcome to VIJAY MALLYA BANK"

    end

    def withdraw(amount)

      if amount > @balance

        puts "Insufficient balance!"

      else

        @balance -= amount

        puts "Withdrawal successful. New balance: #{@balance}"

      end

    end

    def display\_details

      puts "Your Account Number: #{@account\_number}"

      puts "Your Current Balance: #{@balance}"

    end

  end

  account = BankAccount.new("1234567890", 10000)

  account.display\_details

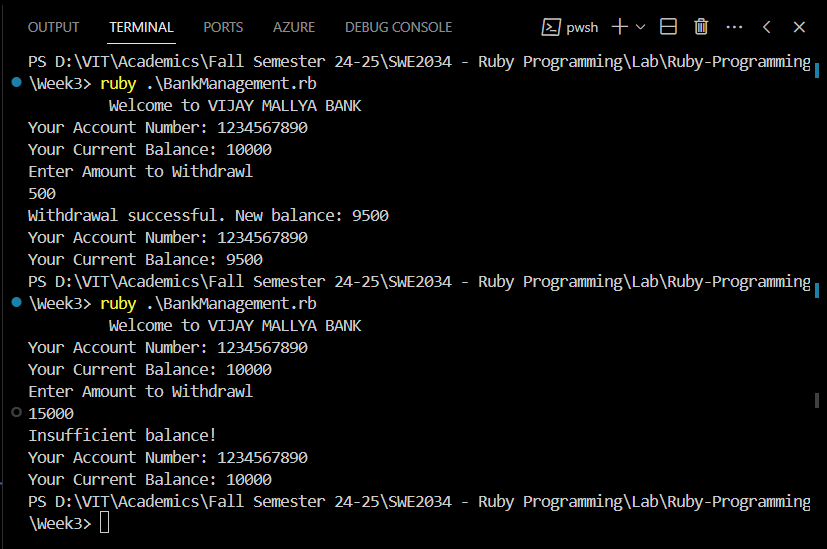
  puts "Enter Amount to Withdrawl"

  amt = gets.chomp.to\_i

  account.withdraw(amt)

  account.display\_details

Output:



1. Student Grade System:

Code:

class Student

    def initialize(name, marks)

      @name = name

      @marks = marks

    end

    def calculate\_grade

      case @marks

      when 90..100

        "S"

      when 80..89

        "A"

      when 70..79

        "B"

      when 60..69

        "C"

      when 50..59

        "D"

      when 40..49

        "E"

      when 0..39

        "F"

      else

        "Invalid marks"

      end

    end

    def display\_grade

      grade = calculate\_grade

      puts "#{@name}'s Grade: #{grade}"

    end

  end

  puts "Enter student name:"

  name = gets.chomp

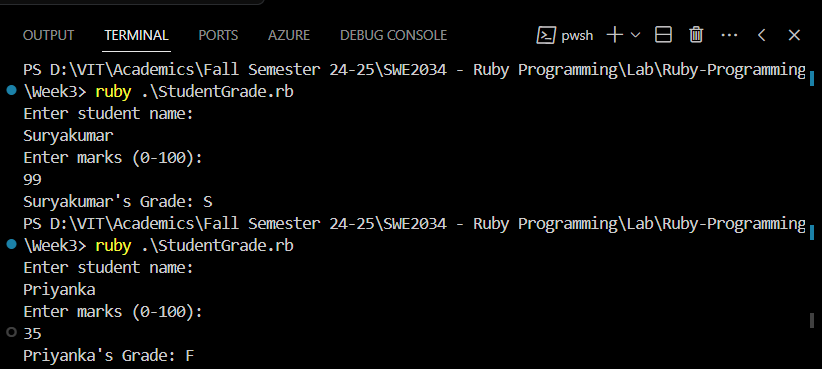
  puts "Enter marks (0-100):"

  marks = gets.chomp.to\_i

  student = Student.new(name, marks)

  student.display\_grade

Output:



1. Employee Salary Management System

Code:

class Employee

    def initialize(name, base\_pay)

      @name = name

      @base\_pay = base\_pay

    end

    def calculate\_net\_salary

      hra, da, tax = calculate\_slabs

      gross\_salary = @base\_pay + hra + da

      net\_salary = gross\_salary - tax

      net\_salary

    end

    def display\_salary

      net\_salary = calculate\_net\_salary

      puts "#{@name}'s Net Salary: Rs. #{net\_salary}"

    end

    private

    def calculate\_slabs

      case @base\_pay

      when 0..20000

        hra = 0.20 \* @base\_pay  *# 20% of base pay*

        da = 0.10 \* @base\_pay   *# 10% of base pay*

        tax = 0.05 \* @base\_pay  *# 5% of base pay*

      when 20001..50000

        hra = 0.25 \* @base\_pay  *# 25% of base pay*

        da = 0.15 \* @base\_pay   *# 15% of base pay*

        tax = 0.10 \* @base\_pay  *# 10% of base pay*

      else

        hra = 0.30 \* @base\_pay  *# 30% of base pay*

        da = 0.20 \* @base\_pay   *# 20% of base pay*

        tax = 0.15 \* @base\_pay  *# 15% of base pay*

      end

      [hra, da, tax]

    end

  end

  puts "Enter employee name:"

  name = gets.chomp

  puts "Enter base pay:"

  base\_pay = gets.chomp.to\_f

  employee = Employee.new(name, base\_pay)

  employee.display\_salary

Output:

