Ruby Lab Exercise – 2

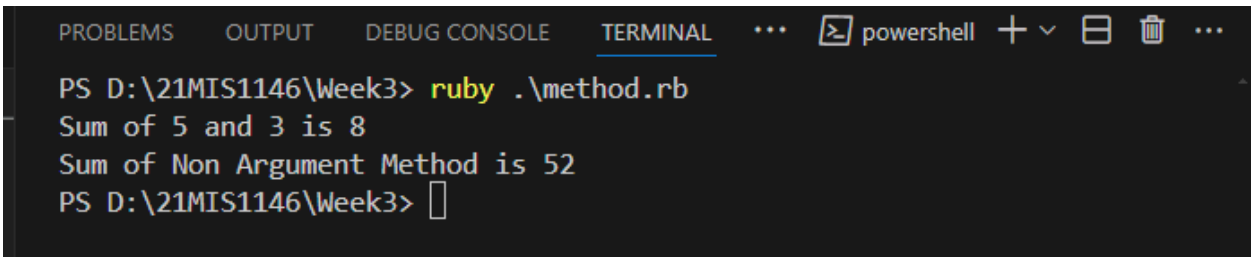
Suryakumar P 21MIS1146

1. Method with and without Arguments:

Code:

```ruby
#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:

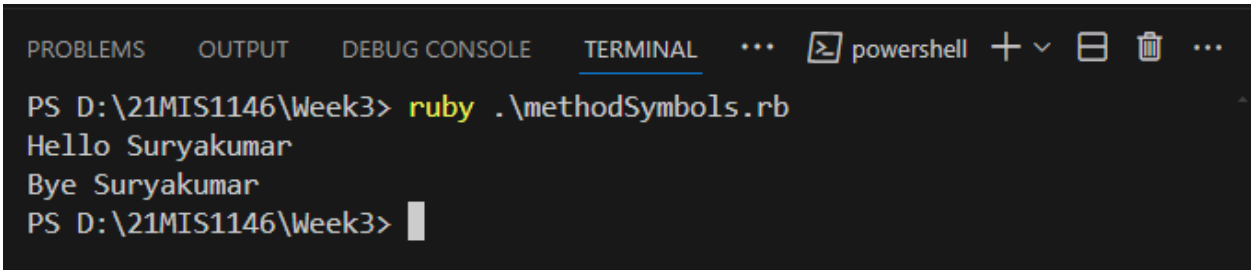


2. Create methods with two different symbols other than ‘?’ and ‘=’

Code:

```ruby
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:

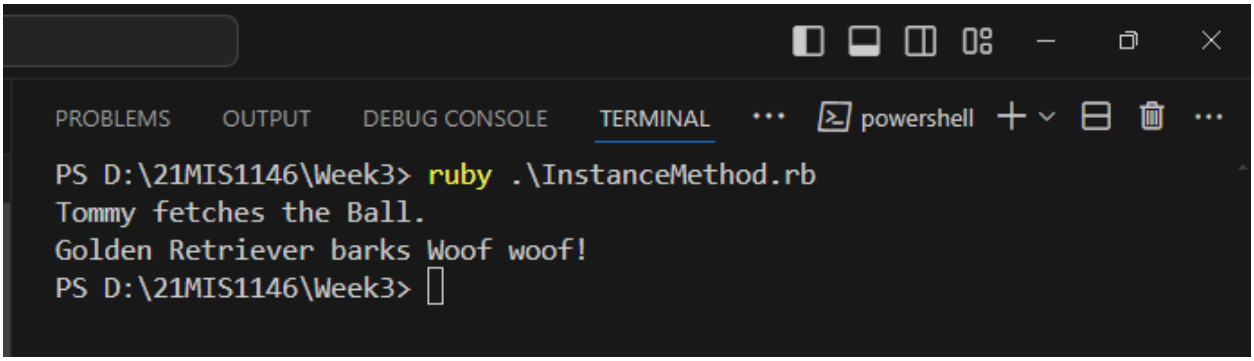


3. A Program using Instance Method

Code:

```ruby
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:

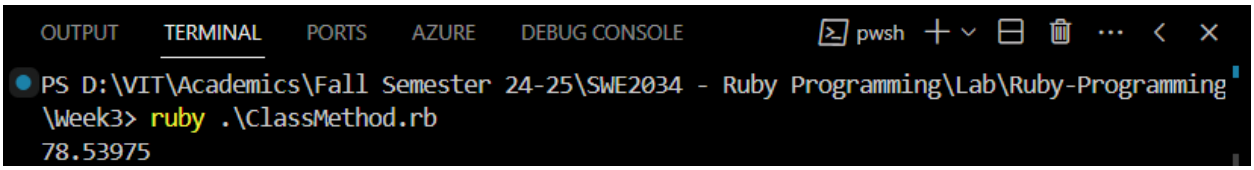


4. Example for Class Method

Code:

```ruby
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:



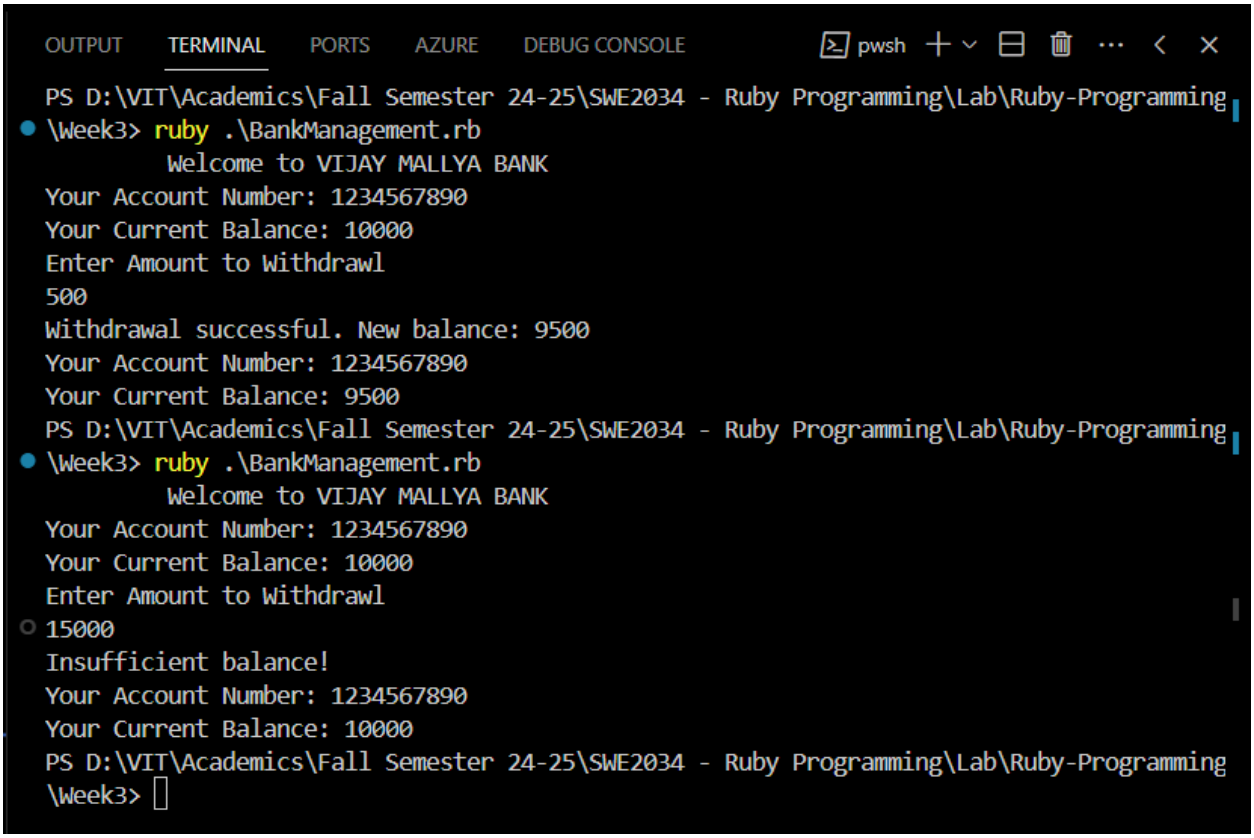
5. Bank Account System

Code:

```ruby
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}"
```

```ruby
    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:



6. Student Grade System:

Code:

```ruby
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:

7. Employee Salary Management System

Code:

```ruby
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
```

```ruby
    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: