

SWE 2034 RUBY LAB ASSESSMENT 2

21MIS1021

VIMAL KUMAR S

1. A program that has a method with and without argument

CODE:

```
new - Notepad

File Edit View

def no_arguments
  puts "This is a method without arguments"
end
def with_arguments(program)
  puts "This is a #{program} program"
end
no_arguments
with_arguments("Ruby")
```

OUTPUT:

```
C:\Users\Dell>cd C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S

C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby new.rb
This is a method without arguments
This is a Ruby program
```

2. Method name ends with 2 different types of symbols other than “?” and “=”

CODE:

```
symbol - Notepad

File Edit View

class MyClass
  def Vit!
    puts "VIT!"
  end
  def add(*num)
    num.inject(0) {|s,n|s+n}
  end
end
obj=MyClass.new
puts obj.Vit!
puts obj.add(1,2,5,6)
```

21MIS1021 VIMAL KUMAR S

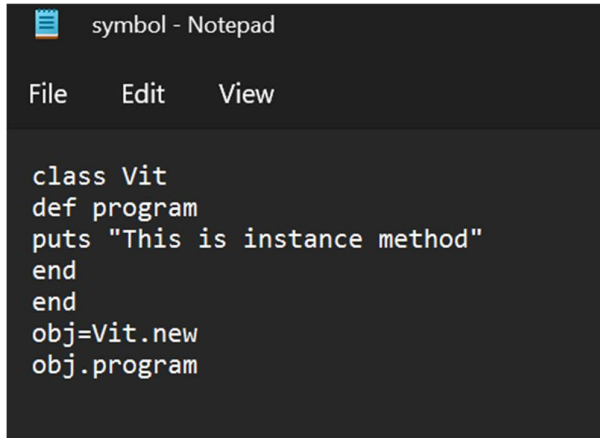
OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby symbol.rb
VIT!

14
```

3. Example program of Instance Method

CODE:



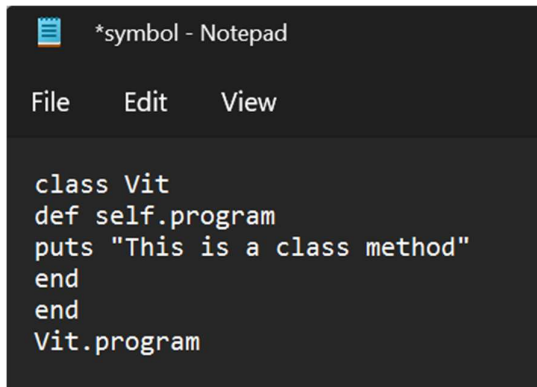
```
class Vit
  def program
    puts "This is instance method"
  end
end
obj=Vit.new
obj.program
```

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby symbol.rb
This is instance method
```

4. Example program of Class Method

CODE:



```
class Vit
  def self.program
    puts "This is a class method"
  end
end
Vit.program
```

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby symbol.rb
This is instance method
```

5. Bank Account System (Get Account details, check the balance before withdraw and update balance)

CODE:

```
bank - Notepad
File Edit View

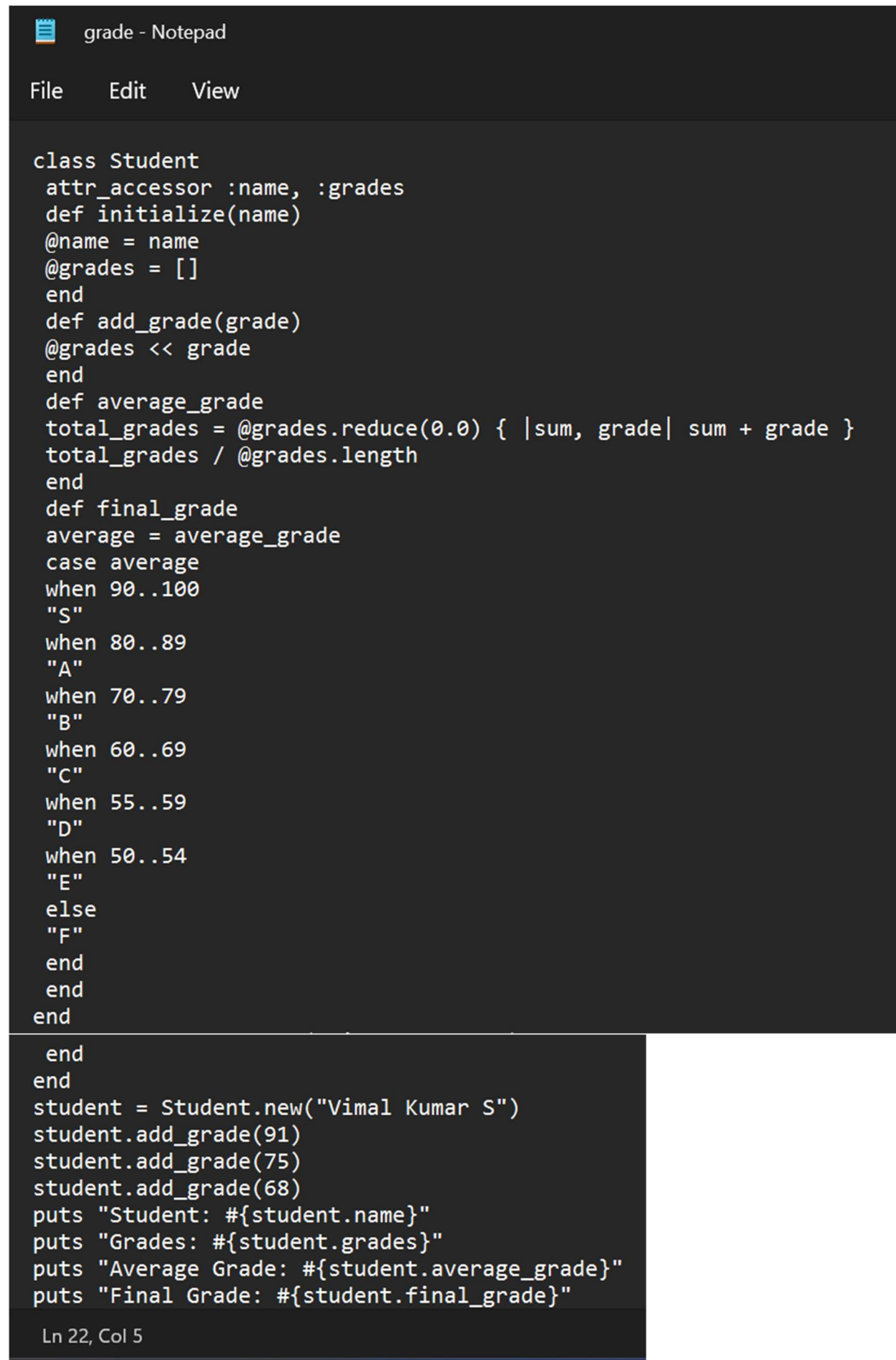
class BankAccount
  attr_accessor :acc_number, :acc_holder, :balance
  def initialize(acc_number, acc_holder, balance)
    @acc_number = acc_number
    @acc_holder = acc_holder
    @balance = balance
  end
  def get_account_details
    "Account Number: #{@acc_number}\nAccount Holder: #{@acc_holder}\nBalance:
rs.#{@balance}"
  end
  def check_balance
    "Current Balance: rs.#{@balance}"
  end
  def withdraw(amount)
    if amount > 0 && amount <= @balance
      @balance -= amount
      "Withdrawal successful. Updated balance: rs.#{@balance}"
    else
      "Insufficient funds or invalid withdrawal amount."
    end
  end
end
account = BankAccount.new("HDFC156740", "Vimal Kumar S", 22000)
puts account.get_account_details
puts account.check_balance
withdrawal_amount = 5000
puts account.withdraw(withdrawal_amount)
```

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby bank.rb
Account Number: HDFC156740
Account Holder: Vimal Kumar S
Balance:
rs.22000
Current Balance: rs.22000
Withdrawal successful. Updated balance: rs.17000
```

6. Student Grade book System

CODE:



```
class Student
  attr_accessor :name, :grades
  def initialize(name)
    @name = name
    @grades = []
  end
  def add_grade(grade)
    @grades << grade
  end
  def average_grade
    total_grades = @grades.reduce(0.0) { |sum, grade| sum + grade }
    total_grades / @grades.length
  end
  def final_grade
    average = average_grade
    case average
    when 90..100
      "S"
    when 80..89
      "A"
    when 70..79
      "B"
    when 60..69
      "C"
    when 55..59
      "D"
    when 50..54
      "E"
    else
      "F"
    end
  end
end

student = Student.new("Vimal Kumar S")
student.add_grade(91)
student.add_grade(75)
student.add_grade(68)
puts "Student: #{student.name}"
puts "Grades: #{student.grades}"
puts "Average Grade: #{student.average_grade}"
puts "Final Grade: #{student.final_grade}"
```

Ln 22, Col 5

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby grade.rb
Student: Vimal Kumar S
Grades: [91, 75, 68]
Average Grade: 78.0
Final Grade: B
```

7. Employee Salary Management System

CODE:

```
emp - Notepad
File Edit View

class Employee
  attr_accessor :name, :salary
  def initialize(name, salary)
    @name = name
    @salary = salary
  end
  def calculate_bonus
    @salary * 0.2
  end
  def to_s
    "Employee: #{@name}\nSalary: Rs.#{@salary}"
  end
end

class SalaryManagementSystem
  def initialize
    @employees = []
  end
  def add_employee(employee)
    @employees << employee
  end
  def calculate_total_salary
    @employees.reduce(0) { |total, employee| total + employee.salary }
  end
  def average_salary
    total_salary = calculate_total_salary
    total_salary / @employees.length
  end
  def display_employees
    @employees.each do |employee|
      puts employee
      puts "Bonus: Rs.#{employee.calculate_bonus}"
    end
  end
end

end
end
end

system = SalaryManagementSystem.new
employee1 = Employee.new("VIMAL", 15000)
employee2 = Employee.new("PRIYANKA", 10000)
system.add_employee(employee1)
system.add_employee(employee2)
puts "Total Salary: Rs.#{system.calculate_total_salary}"
puts "Average Salary: Rs.#{system.average_salary}"
system.display_employees

Ln 37, Col 35
```

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021 VIMAL KUMAR S>ruby emp.rb
Total Salary: Rs.25000
Average Salary: Rs.12500
Employee: VIMAL
Salary: Rs.15000
Bonus: Rs.3000.0
Employee: PRIYANKA
Salary: Rs.10000
Bonus: Rs.2000.0
```