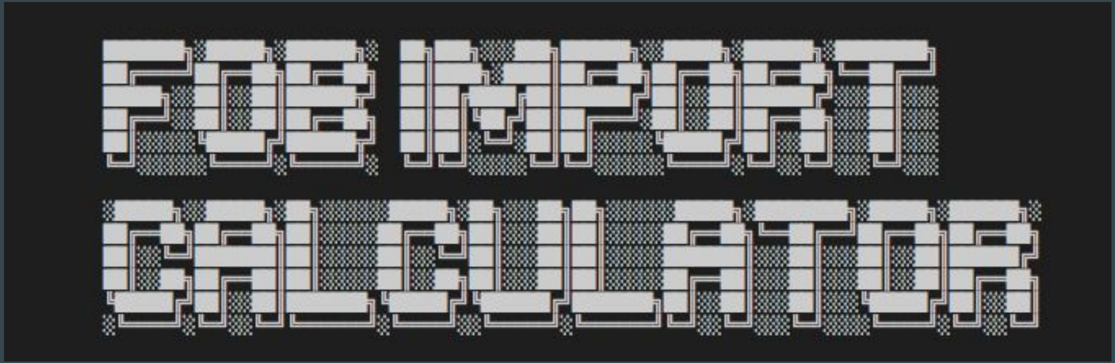
Freight Cost Calculator

T1A3_Darko_Mihajlovic

Overview



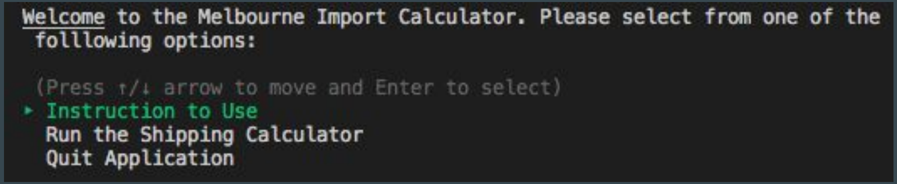
- Ocean Freight Cost Calculator
- Imports into Melbourne Port
- Includes prices from 97 main ports
- Offers 2 different shipping modes
    - Full Container Load (FCL) - 20’ & 40’ containers
    - Less than container Load (LCL)

Features

- Instructions
- Shipment Details <- User Input
- Calculates and outputs best / most cost effective mode of transport
- Load Port <- User Input
- Load Port price from CSV file
- Outputs total cost based on Load Port and calculated mode of transport (FCL/LCL)

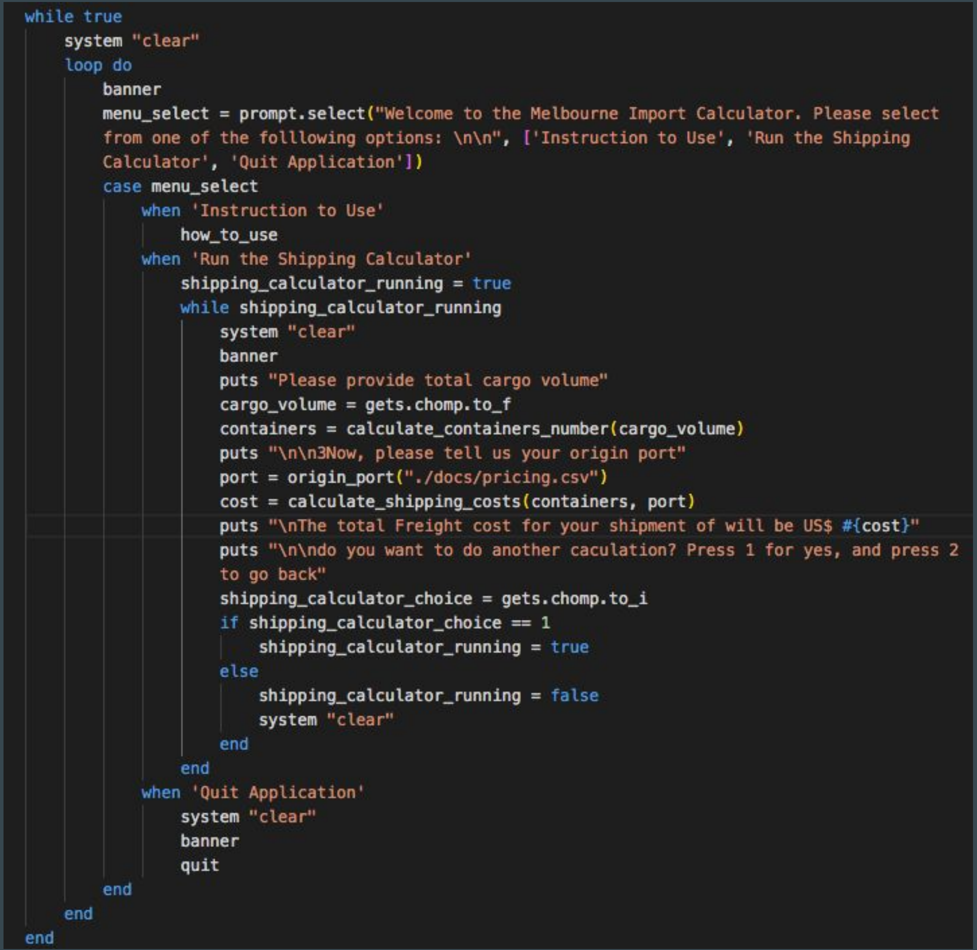
App Logic

- App Menu



```
Welcome to the Melbourne Import Calculator. Please select from one of the
 folllowing options:

 (Press ↑/↓ arrow to move and Enter to select)
‣ Instruction to Use
  Run the Shipping Calculator
  Quit Application
```

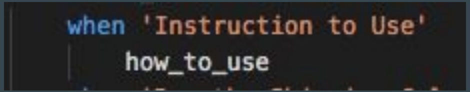
- Tty:prompt
- Loop / Case statement



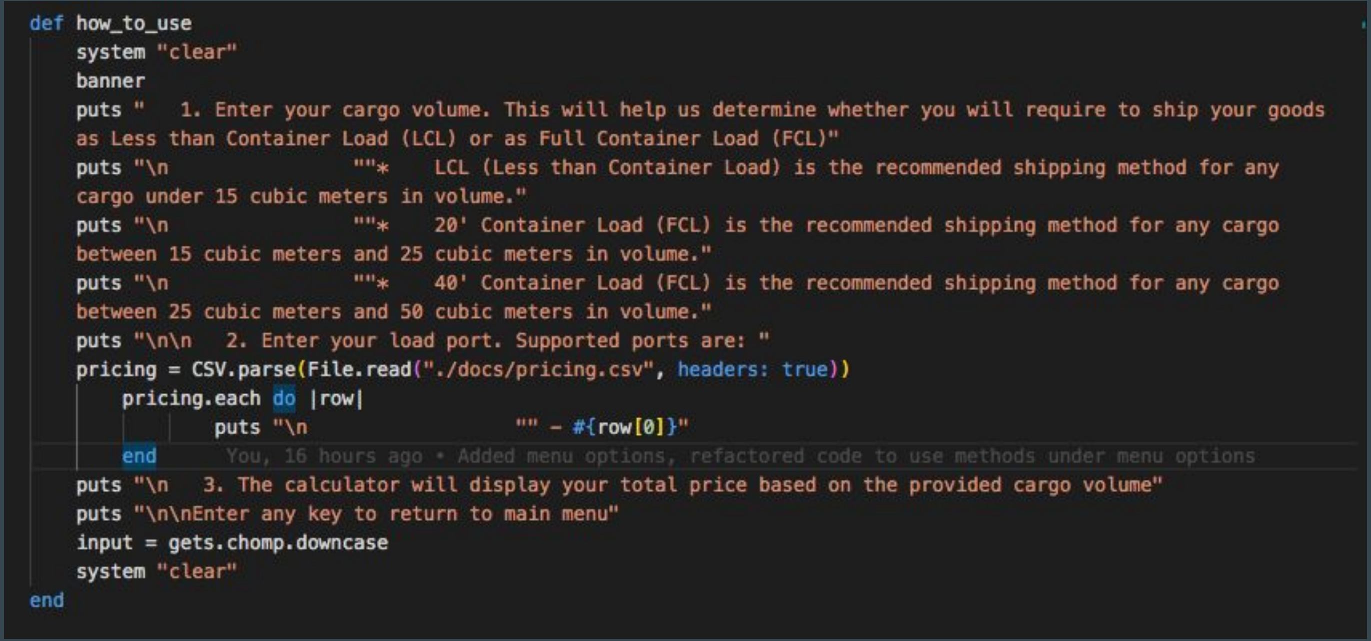
```ruby
while true
    system "clear"
    loop do
        banner
        menu_select = prompt.select("Welcome to the Melbourne Import Calculator. Please select
        from one of the folllowing options: \n\n", ['Instruction to Use', 'Run the Shipping
        Calculator', 'Quit Application'])
        case menu_select
            when 'Instruction to Use'
                how_to_use
            when 'Run the Shipping Calculator'
                shipping_calculator_running = true
                while shipping_calculator_running
                    system "clear"
                    banner
                    puts "Please provide total cargo volume"
                    cargo_volume = gets.chomp.to_f
                    containers = calculate_containers_number(cargo_volume)
                    puts "\n\n3Now, please tell us your origin port"
                    port = origin_port("./docs/pricing.csv")
                    cost = calculate_shipping_costs(containers, port)
                    puts "\nThe total Freight cost for your shipment of will be US$ #{cost}"
                    puts "\n\ndo you want to do another caculation? Press 1 for yes, and press 2
                    to go back"
                    shipping_calculator_choice = gets.chomp.to_i
                    if shipping_calculator_choice == 1
                        shipping_calculator_running = true
                    else
                        shipping_calculator_running = false
                        system "clear"
                    end
                end
            when 'Quit Application'
                system "clear"
                banner
                quit
        end
    end
end
```

App Logic

- Methods - “Instructions to Use”



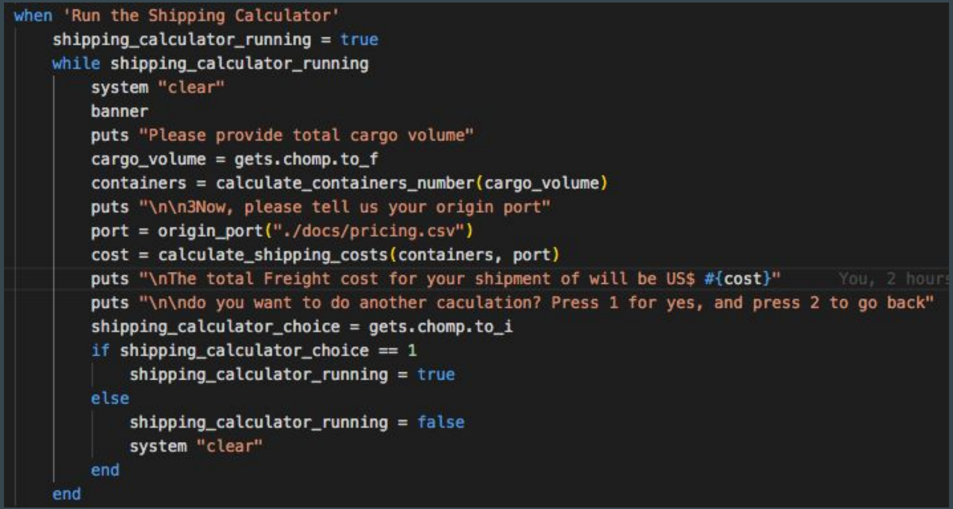
```ruby
when 'Instruction to Use'
    how_to_use
```



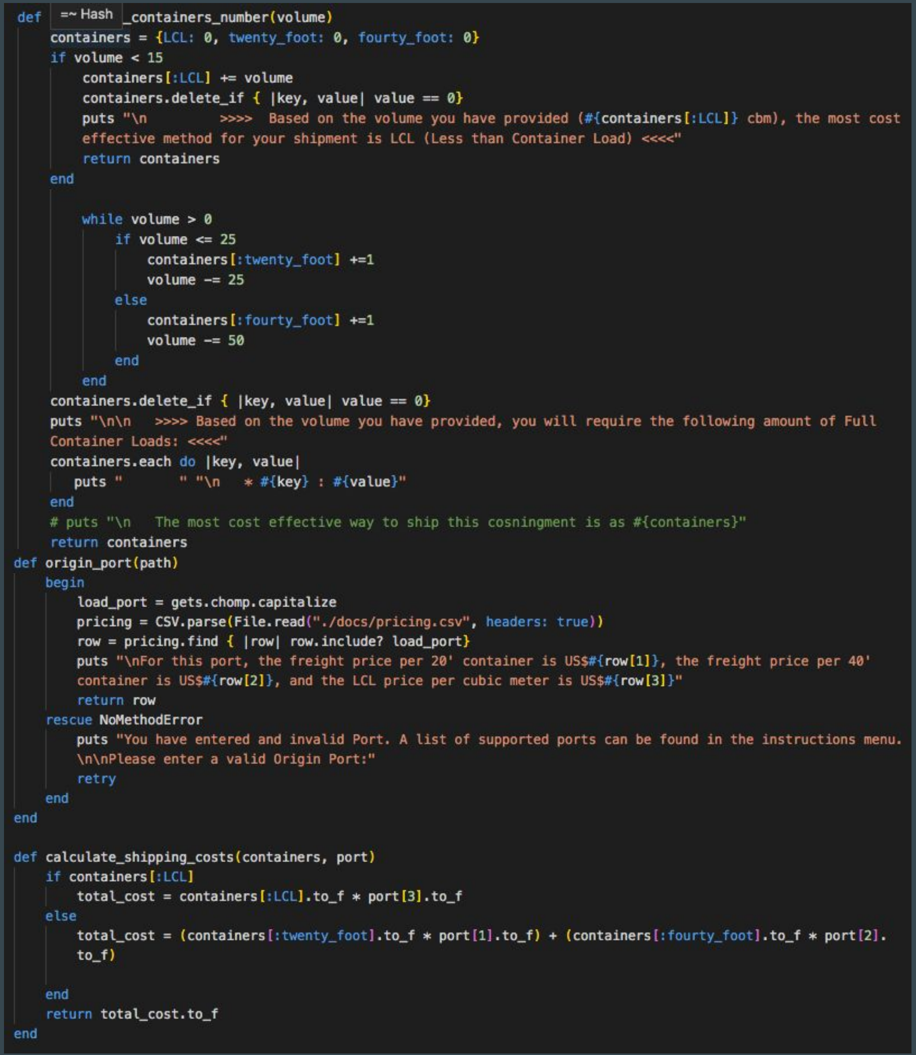
```ruby
def how_to_use
    system "clear"
    banner
    puts "   1. Enter your cargo volume. This will help us determine whether you will require to ship your goods
    as Less than Container Load (LCL) or as Full Container Load (FCL)"
    puts "\n                ""*    LCL (Less than Container Load) is the recommended shipping method for any
    cargo under 15 cubic meters in volume."
    puts "\n                ""*    20' Container Load (FCL) is the recommended shipping method for any cargo
    between 15 cubic meters and 25 cubic meters in volume."
    puts "\n                ""*    40' Container Load (FCL) is the recommended shipping method for any cargo
    between 25 cubic meters and 50 cubic meters in volume."
    puts "\n\n   2. Enter your load port. Supported ports are: "
    pricing = CSV.parse(File.read("./docs/pricing.csv", headers: true))
        pricing.each do |row|
            puts "\n                  "" - #{row[0]}"
        end       You, 16 hours ago • Added menu options, refactored code to use methods under menu options
    puts "\n   3. The calculator will display your total price based on the provided cargo volume"
    puts "\n\nEnter any key to return to main menu"
    input = gets.chomp.downcase
    system "clear"
end
```

App Logic

- Methods - “Run Shipping Calculator”
  - Variable
  - User Input



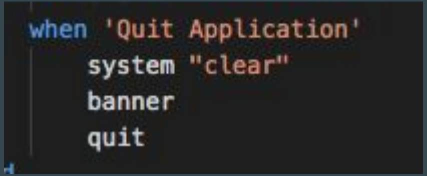
```ruby
when 'Run the Shipping Calculator'
    shipping_calculator_running = true
    while shipping_calculator_running
        system "clear"
        banner
        puts "Please provide total cargo volume"
        cargo_volume = gets.chomp.to_f
        containers = calculate_containers_number(cargo_volume)
        puts "\n\n3Now, please tell us your origin port"
        port = origin_port("./docs/pricing.csv")
        cost = calculate_shipping_costs(containers, port)
        puts "\nThe total Freight cost for your shipment of will be US$ #{cost}"      You, 2 hours
        puts "\n\ndo you want to do another caculation? Press 1 for yes, and press 2 to go back"
        shipping_calculator_choice = gets.chomp.to_i
        if shipping_calculator_choice == 1
            shipping_calculator_running = true
        else
            shipping_calculator_running = false
            system "clear"
        end
    end
```



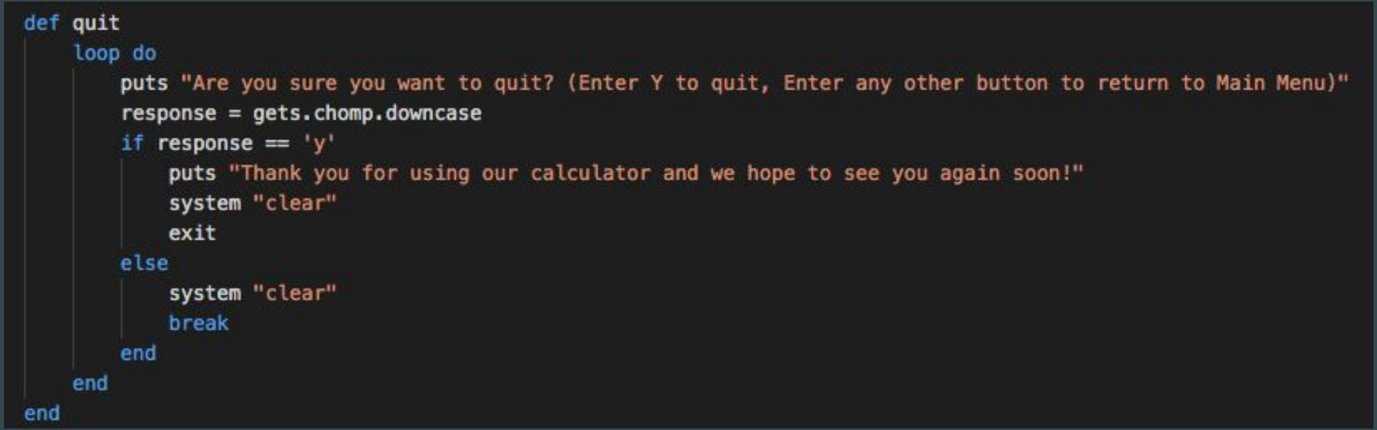
```ruby
def  =~ Hash _containers_number(volume)
    containers = {LCL: 0, twenty_foot: 0, fourty_foot: 0}
    if volume < 15
        containers[:LCL] += volume
        containers.delete_if { |key, value| value == 0}
        puts "\n          >>>>  Based on the volume you have provided (#{containers[:LCL]} cbm), the most cost
        effective method for your shipment is LCL (Less than Container Load) <<<<"
        return containers
    end

        while volume > 0
            if volume <= 25
                containers[:twenty_foot] +=1
                volume -= 25
            else
                containers[:fourty_foot] +=1
                volume -= 50
            end
        end
    containers.delete_if { |key, value| value == 0}
    puts "\n\n   >>>> Based on the volume you have provided, you will require the following amount of Full
    Container Loads: <<<<"
    containers.each do |key, value|
       puts "       " "\n   * #{key} : #{value}"
    end
    # puts "\n   The most cost effective way to ship this cosningment is as #{containers}"
    return containers
def origin_port(path)
    begin
        load_port = gets.chomp.capitalize
        pricing = CSV.parse(File.read("./docs/pricing.csv", headers: true))
        row = pricing.find { |row| row.include? load_port}
        puts "\nFor this port, the freight price per 20' container is US$#{row[1]}, the freight price per 40'
        container is US$#{row[2]}, and the LCL price per cubic meter is US$#{row[3]}"
        return row
    rescue NoMethodError
        puts "You have entered and invalid Port. A list of supported ports can be found in the instructions menu.
        \n\nPlease enter a valid Origin Port:"
        retry
    end
end

def calculate_shipping_costs(containers, port)
    if containers[:LCL]
        total_cost = containers[:LCL].to_f * port[3].to_f
    else
        total_cost = (containers[:twenty_foot].to_f * port[1].to_f) + (containers[:fourty_foot].to_f * port[2].
        to_f)

    end
    return total_cost.to_f
end
```

App Logic

- Methods - “Quit”



```ruby
when 'Quit Application'
    system "clear"
    banner
    quit
```



```ruby
def quit
    loop do
        puts "Are you sure you want to quit? (Enter Y to quit, Enter any other button to return to Main Menu)"
        response = gets.chomp.downcase
        if response == 'y'
            puts "Thank you for using our calculator and we hope to see you again soon!"
            system "clear"
            exit
        else
            system "clear"
            break
        end
    end
end
```

Demo

Summary

- Fully functional App & industry ready
- Address real life problems
- Suggests methods for inexperienced importers
- Adaptable
- Allows for further enhancements:
  - API (If available)
  - Link to SQL Databases
  - Use auto-update CSV files

Thank You !

•••

T1A3_Darko_Mihajlovic