



Flower Shop App

T1A3 - Terminal Application
Ana (Mingyang Wang)
Student ID: 11945

A photograph showing a person's hands arranging a bouquet of tulips in a white vase. The bouquet includes red, pink, purple, and white tulips, along with greenery and twine. The background is a light-colored wooden surface.

Outline

- Terminal application features
- The logic and code of application
- A review of development process

A close-up photograph of a person's hand holding a bouquet of tulips. The bouquet includes red, white, and purple flowers, along with greenery and twine. The background is slightly blurred.

Main Features for A Flower Shop Application

Display Welcome

Welcome message displays name of the flower shop

Display Bouquets

Bouquets display bouquet items

Placing An Order

Validate item using bouquets

Add item to order

Display Order and Total Order Price

Display order items from order

Display order cost using prices from bouquets and items in order

A close-up photograph of a person's hand holding a bouquet of tulips. The bouquet includes red, pink, purple, and white flowers, along with greenery and twine. The background is a light-colored wooden surface.

Application Interacting with Users

ruby-flowershop.rb

- Greeting to customers: “Hello, Ana!”
- Display welcome: “Welcome to Blossom House!”
- Display the marketing slogan for the flower shop
- Display the bouquet description
- Display the list of bouquets with prices
- In a loop, until receive “done”:
 1. Prompt for order item
 2. Validate order item
 3. Prompt quantity
 4. Add item to order
- Print order
- Prompt bouquet delivery service
- Print contact detail of the flower shop



Using Ruby gems for:

- Gem colorize: colorizing text
- Gem tty-prompt: command line prompt with a robust API for getting and validating complex inputs
- Gem json (file): storing and displaying each bouquet description
- Gem pastel: terminal output styling
- Gem tyy-font: writing text in large stylized characters
- Gem rubocop: setting up coding style
- Gem RSpec: using for the TDD



Terminal Application

```
anawang@Anas-MBP ruby-flowershop-app % ruby ruby-flowershop.rb  
What is your name? Ana
```

```
Hello, Ana!
```

```
Welcome to Blossom House!
```

Feel the Blossom

Please choose your flowers from Bouquet:

Bouquet Description

PARIS: A romantic feminine floral design featuring fuchsia, lilac & pink colours.

LISBON: Named after the bright and colourful Portuguese seaside town, our Lisbon collection stands out for its pink and apricot pastel hues. Lisianthus & Carnation flowers with Monstera Leaves.

OSAKA: Japan's annual Cherry Blossom festival was the inspiration for this unique floral design we named "Osaka". Osaka features soft pink colours paired with delicate white puffs. Roses, Chrysanthemum Disbuds and Gypsophilia.

SYDNEY: Australian native flowers and foliage dominate our rustic Broome design. This arrangement features a strong scent of Eucalyptus.

CANNES: The delicate appearance of this beautiful arrangement belies its underlying strength. Carnations, Gypsophila, Chrysanthemum Disbud, Anthurium, Stock, Roses and Caladium Leaf.

LONDON: Perfect for pink lovers, our London design features bold pops of pink combined with lush green foliage. Snapdragon, Tulip and Chrysanthemum Disbud flowers with Cypress foliage.

BARCELONA: With its dusky pinks, sunny yellows and warm, earthy browns, this gorgeous design is radiant and colourful. Magnolia foliage, Bansksia, Lissianthus, Carnations, Yellow Pom Pom Disbuds and Local Chrysanthemum.

SANTORINI: Named after white colour of architecture with blue southern Aegean Sea, the design features textured pops of white blooms. Delicate clusters of white Lisianthus, Chrysanthemum Disbuds and Alstroemeria, backed by Magnolia foliage.

Bouquet

PARIS	...	130
LISBON	...	120
OSAKA	...	130
SYDNEY	...	120
CANNES	...	110
LONDON	...	100
BARCELONA	...	110
SANTORINI	...	100

What would you like to order? When you are finished, please type 'done'.

London

How many would you like?

2

Bouquet

PARIS	...	130
LISBON	...	120
OSAKA	...	130
SYDNEY	...	120
CANNES	...	110
LONDON	...	100
BARCELONA	...	110
SANTORINI	...	100

What would you like to order? When you are finished, please type 'done'.

done

Thank you! Here is your order:



Code for ruby-flowershop.rb

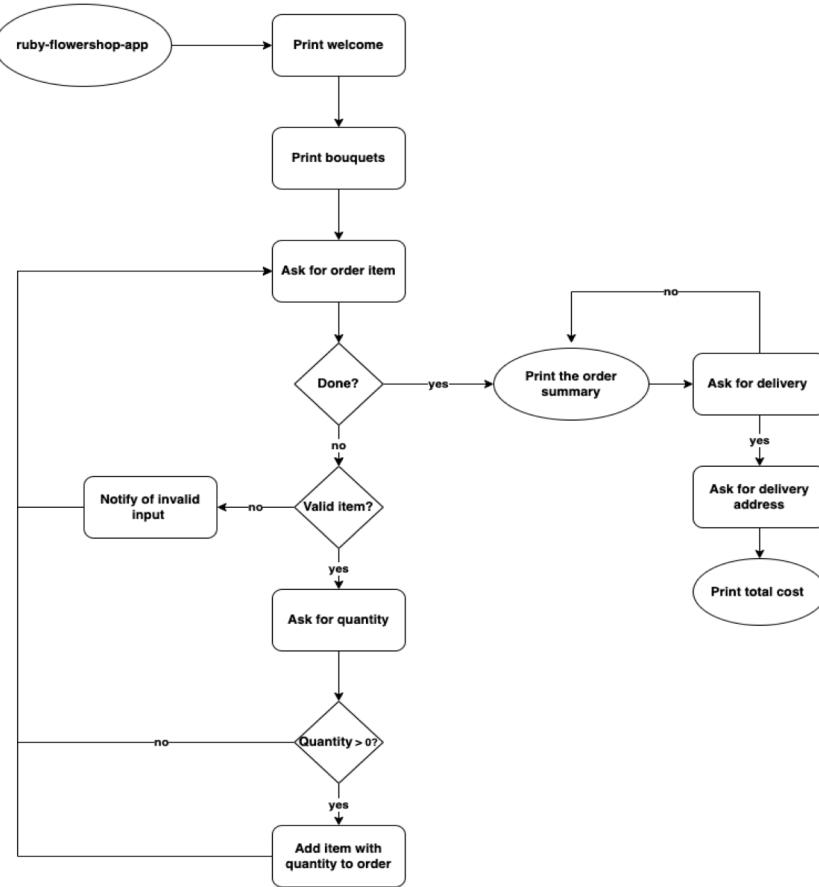
```
ruby-flowershop-app > █ ruby-flowershop.rb
1  require_relative './flowershop'
2  require 'colorize'
3  require 'tty-prompt'
4  prompt = TTY::Prompt.new
5  require 'tty-font'
6  font = TTY::Font.new(:doom)
7  require 'json'
8
9  # create an instance of flowershop
10 bouquet = { 'PARIS' => 130, 'LISBON' => 120, 'OSAKA' => 130, 'SYDNEY' => 120, 'CANNES' => 110, 'LONDON' => 100,
11   'BARCELONA' => 110, 'SANTORINI' => 100 }
12 flowershop = Flowershop.new('Blossom House', bouquet)
13
14 # say hello to customer
15 if ARGV[0]
16   customer_name = ARGV[0]
17 else
18   begin
19     customer_name = prompt.ask('What is your name?')
20     raise InvalidNameError if customer_name.empty?
21
22     puts '-----'
23     puts "Hello, #{customer_name}!".colorize(:blue)
24     puts ''
25   rescue StandardError
26     puts 'Please enter your name!'.colorize(:red)
27     retry
28   end
29 end
30
31 # print welcome message and the marketing slogan
32 flowershop.welcome
33 pastel = Pastel.new
34 puts pastel.red(font.write('Feel the Blossom'))
35 puts pastel.blue(pastel.underline('Please choose your flowers from Bouquet:'))
36 puts ''
37
38 # Print Bouquet Description from JSON file
39 puts 'Bouquet Description'.colorize(:red)
40 puts ''
41 file = File.read('../bouquet.json')
42 data_hash = JSON.parse(file)
43 data_hash.each do |key, value|
44   puts "#{key}: #{value}"
45   puts ''
46 end
47
48 loop do
49   # Print bouquet price list
50   flowershop.print_bouquet
51   puts ''
52   puts "What would you like to order? When you are finished, please type 'done'.":colorize(:blue)
53   input = gets.strip.upcase
54
55   # If users are done, break from loop
56   break if input == 'DONE'
```



Overall Structure of Application (The Logic of Application)

Ruby Flower Shop App

A simple app that takes an order from a user after displaying bouquets. When an order is completed, the order summary is displayed.





Test Driven Development

- Writing tests that fails for the code(Red)
- Make the tests pass (Green)
- Make the code better(Refactor)

```
ruby-flowershop-app > spec > ruby_flowershop_spec.rb
1  require_relative '../bouquet_item'
2  require_relative '../bouquet'
3  require_relative '../order'
4  require_relative '../flowershop'
5
6  describe BouquetItem do
7    it 'should return the price of the item' do
8      name = 'Paris'
9      price = 130
10     bouquet_item = BouquetItem.new(name, price)
11     expect(bouquet_item.price).to eq(price)
12   end
13   it 'should return the name of the item' do
14     name = 'Paris'
15     price = 130
16     bouquet_item = BouquetItem.new(name, price)
17     expect(bouquet_item.name).to eq(name)
18   end
19 end
```

```
anawang@Anas-MBP global-ruby % ls
index.rb                               ruby-flowershop-app
anawang@Anas-MBP global-ruby % cd ruby-flowershop-app
anawang@Anas-MBP ruby-flowershop-app % rspec -f d

BouquetItem
  should return the price of the item
  should return the name of the item

Bouquet
  should be able to get an item price
  should be able to add an item
  should return the item name for a valid item
  should return nil for an invalid item

Order
  should add an item to the order
  should update an item quantity

Flowershop
  should create a flowershop with a name
  should create a flowershop with a bouquet
  should add an item to order
  Welcome to Blossom House!
    should define a welcome method
Bouquet
-----
Paris    ... 130
Lisbon   ... 120
  should define a print_bouquet method
  should calculate order total

Finished in 0.00744 seconds (files took 0.3946 seconds to load)
14 examples, 0 failures
```



Identify Data Structure

Classes for this application: building from the lowest data structure

The BouquetItem class

- Attributes: price and name
- Actions: display item, get price

Code for BouquetItem Class

```
ruby-flowershop-app > bouquet_item.rb
1  class BouquetItem
2    attr_reader :price, :name
3
4    def initialize(name, price)
5      @name = name
6      @price = price
7    end
8
9    def to_s
10      @name.to_s + ' ' * (10 - @name.length) + "... #{@price}"
11    end
12  end
```



The Bouquet class:

- Attributes: bouquet items
- Actions: add bouquet item, display bouquets, get price for bouquet item, validate bouquet item

Code for Bouquet Class

```
ruby-flowershop-app > bouquet.rb
  1  require_relative './bouquet_item'
  2
  3  class Bouquet
  4    def initialize
  5      @bouquet_items = []
  6    end
  7
  8    def add_item(name, price)
  9      bouquet_item = BouquetItem.new(name, price)
 10      @bouquet_items << bouquet_item
 11    end
 12
 13    def get_price(name)
 14      item = @bouquet_items.find { |bouquet_item| bouquet_item.name == name }
 15      item.price
 16    end
 17
 18    def get_items
 19      @bouquet_items
 20    end
 21
 22    def display
 23      puts 'Bouquet'
 24      puts '-----'
 25      @bouquet_items.each { |item| puts item }
 26      nil
 27    end
 28
 29    def validate_item(name)
 30      @bouquet_items.each do |bouquet_item|
 31        return name if bouquet_item.name == name
 32      end
 33      nil
 34    end
 35  end
```



Code for Order Class

The Order class

- Attributes: order items
- Actions: add item, get items, display order items

```
ruby-flowershop-app > order.rb
```

```
1  class Order
2  def initialize
3  |   @order_items = Hash.new(0)
4  end
5
6  def add_item(name, quantity)
7  |   @order_items[name] += quantity
8  end
9
10 def get_items
11 |   @order_items
12 end
13 end
```

The Flowershop class

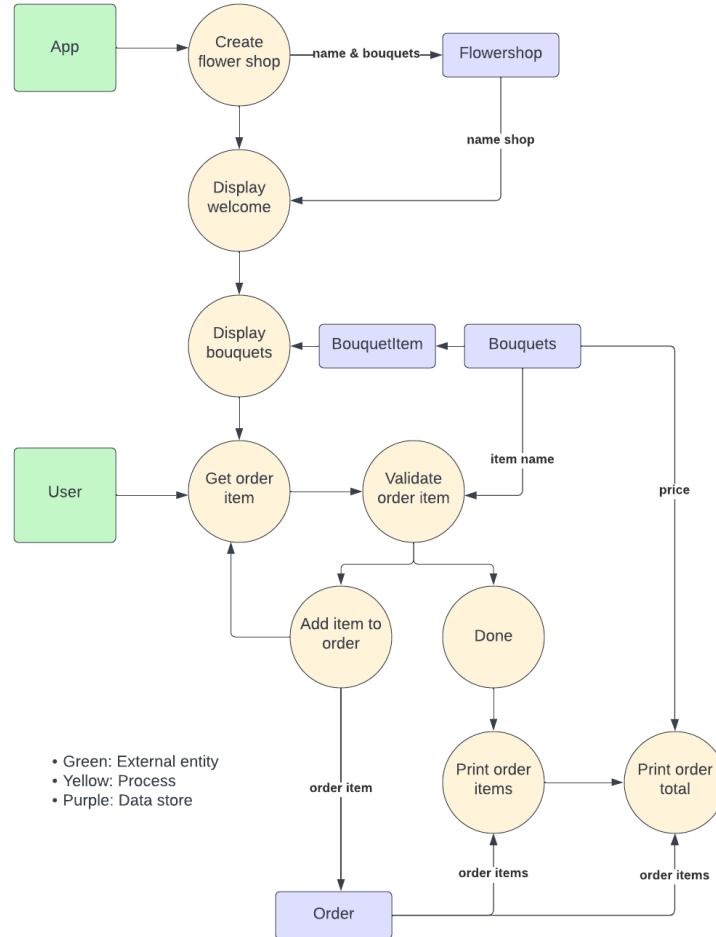
- Attributes: name, bouquet, order
 - Actions: display welcome, bouquets, and order with total, add item to order

Code for Flowershop Class

```
ruby-flowershop-app > ./flowershop.rb
1   require_relative './bouquet'
2   require_relative './order'
3
4 class Flowershop
5   attr_reader :name, :bouquet
6
7   def initialize(name, bouquet_items)
8     @name = name
9     @bouquet = Bouquet.new
10    populate_bouquet(bouquet_items)
11    @order = Order.new
12  end
13
14  def populate_bouquet(bouquet_items)
15    bouquet_items.each do |name, price|
16      @bouquet.add_item(name, price)
17    end
18  end
19
20  def add_to_order(item, quantity)
21    @order.add_item(item, quantity)
22  end
23
24  def get_order
25    @order
26  end
27
28  def welcome
29    puts "Welcome to #{@name}!"
30    puts
31  end
32
33  def print_bouquet
34    @bouquet.display
35  end
36
37  def order_total
38    total = 0
39    @order.get_items.each do |item, quantity|
40      total += @bouquet.get_price(item) * quantity
41    end
42    total
43  end
44
45  def print_order
46    if @order.get_items.size > 0
```



Data Flow Diagram (Development Process)



Development Process

- Challenges:
 - Decisions on features
 - Automated testing – time consuming
 - Coding process
- Favourite parts:
 - Making the tests pass
 - Using Ruby gems
 - Enjoy using git and pushing all commits to my repository on GitHub



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

commit ebb6a77e9783d6bd0c06403cdd3784aac8a4aaaa (HEAD -> master, origin/master)
Author: AnaBondiguel <michaelyangyang@hotmail.com>
Date: Mon Apr 18 17:37:35 2022 +1000

    added run_app.sh

commit b4aa32953a219200067f0210975970cd3143e9bf
Author: AnaBondiguel <michaelyangyang@hotmail.com>
Date: Wed Apr 13 20:40:11 2022 +1000

    Added new file ruby-flowershop to app

commit 3e122a88239d674e20d9afcaab6c29bd6c85ee35 (add-print-bouquet)
Author: AnaBondiguel <michaelyangyang@hotmail.com>
Date: Wed Apr 13 12:36:45 2022 +1000

    Adding print bouquet

commit c3173aebd26ac5ea1dab988f6687afb10c5e15e (add-flowershop-welcome)
Author: AnaBondiguel <michaelyangyang@hotmail.com>
Date: Wed Apr 13 11:15:11 2022 +1000

    Added welcome to flowershop

commit 6cf15438c6e6a792894e5041a5e239900840143f
Author: AnaBondiguel <michaelyangyang@hotmail.com>
Date: Tue Apr 12 22:13:09 2022 +1000

    initial commit
```