Flower Shop App

T1A3 - Terminal Application

Ana (Mingyang Wang)
Student ID: 11945

Outline

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

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

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!

 _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____
|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|
|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|  ___|
|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|


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, Gypsophilia, 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, Banskia, Lissianthus, Carnations, Yellow Pom Pom Disbuds and Local Chrysanthemum.

SANTORINI: Named after white colour of architecture with blue sourthern 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

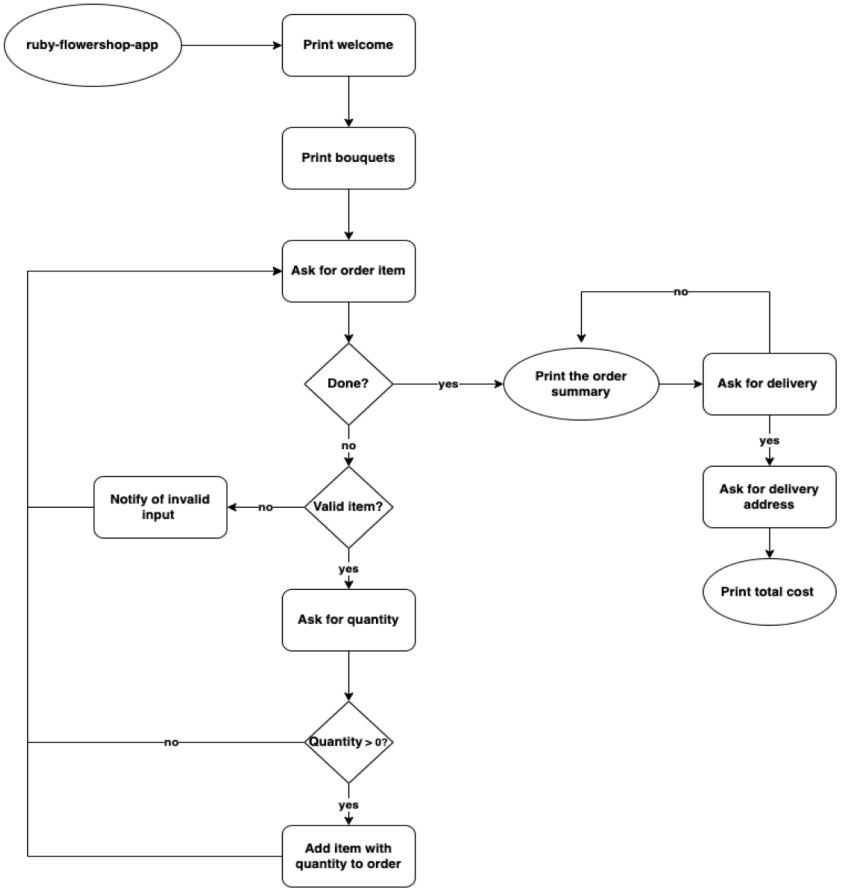
```ruby
require_relative './flowershop'
require 'colorize'
require 'tty-prompt'
prompt = TTY::Prompt.new
require 'tty-font'
font = TTY::Font.new(:doom)
require 'json'

# create an instance of flowershop
bouquet = {"PARIS" => 130, "LISBON" => 120, "OSAKA" => 130, "SYDNEY" => 120, "CANNES" => 110, "LONDON" => 100, "BARCELONA" => 110, "SANTORINI" => 100}
flowershop = Flowershop.new("Blossom House", bouquet)

# say hello to customers
if ARGV[0]
    customer_name = ARGV[0]
else
 begin
    customer_name = prompt.ask("What is your name?")
    raise InvalidNameError if customer_name.empty?
    puts "----------------"
    puts "Hello, #{customer_name}!".colorize(:blue)
    puts ""
rescue
    puts "Please enter your name!"
    retry
 end
end

# print welcome message and the marketing slogan
flowershop.welcome
pastel = Pastel.new
    puts pastel.red(font.write("Feel the Blossom"))
    puts pastel.blue(pastel.underline("Please choose your flowers from Bouquet:"))
    puts ""

#Print Bouquet Description from JSON file
puts "Bouquet Description".colorize(:red)
    puts ""
    file = File.read('./bouquet.json')
    data_hash = JSON.parse(file)
    data_hash.each do |key, value|
    puts "#{key}: #{value}"
    puts ""
end

loop do
#print bouquet list
    flowershop.print_bouquet
    puts
    puts "What would you like to order? When you are finished, please type 'done'.".colorize(:blue)
    input = gets.strip.upcase
```

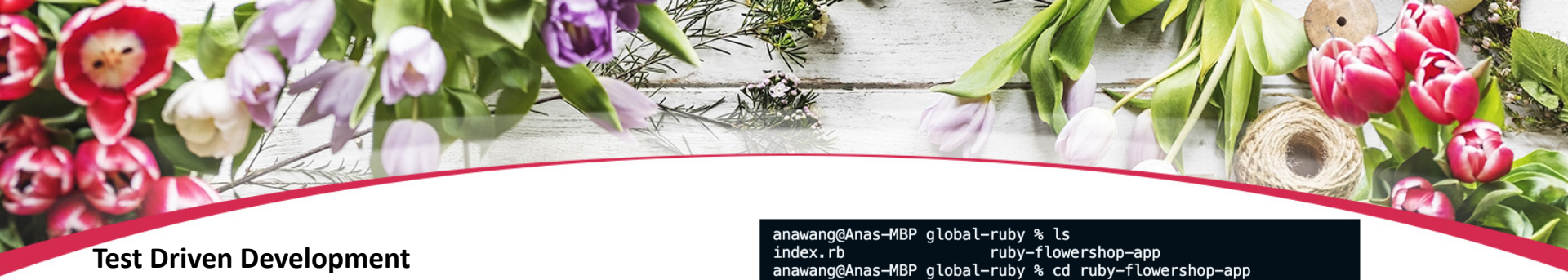
Overall Structure of Application

(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

```ruby
require_relative '../bouquet_item'
require_relative '../bouquet'
require_relative '../order'
require_relative '../flowershop'


describe BouquetItem do
    it 'should return the price of the item' do
        name = "Paris"
        price = 130
        bouquet_item = BouquetItem.new(name,price)
        expect(bouquet_item.price).to eq(price)
    end
    it 'should return the name of the item' do
        name = "Paris"
        price = 130
        bouquet_item = BouquetItem.new(name,price)
        expect(bouquet_item.name).to eq(name)
    end
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
class BouquetItem
    attr_reader :price, :name
    def initialize(name, price)
        @name = name
        @price = price
    end

    def to_s
        return "#{@name}" +" "*(10 - @name.length)+"... #{@price}"
    end
end
```

Code for Bouquet Class

The Bouquet class:

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

```ruby
require_relative './bouquet_item'

class Bouquet
    def initialize
        @bouquet_items = []
    end

    def add_item(name,price)
        bouquet_item = BouquetItem.new(name,price)
        @bouquet_items << bouquet_item
    end

    def get_price(name)
        item = @bouquet_items.find {|bouquet_item| bouquet_item.name==name}
        return item.price
    end

    def get_items
        return @bouquet_items
    end

    def display
        puts "Bouquet".colorize(:green)
        puts "----------".colorize(:green)
        @bouquet_items.each {|item| puts item}
        return nil
    end

    def validate_item(name)
        @bouquet_items.each do |bouquet_item|
          if bouquet_item.name == name
              return name
          end
        end
            return nil
    end

end
```

The Order class

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

Code for Order Class

```ruby
class Order
    def initialize
        @order_items = Hash.new(0)
    end

    def add_item(name,quantity)
        @order_items[name] += quantity
    end

    def get_items
            return @order_items
     end
end
```

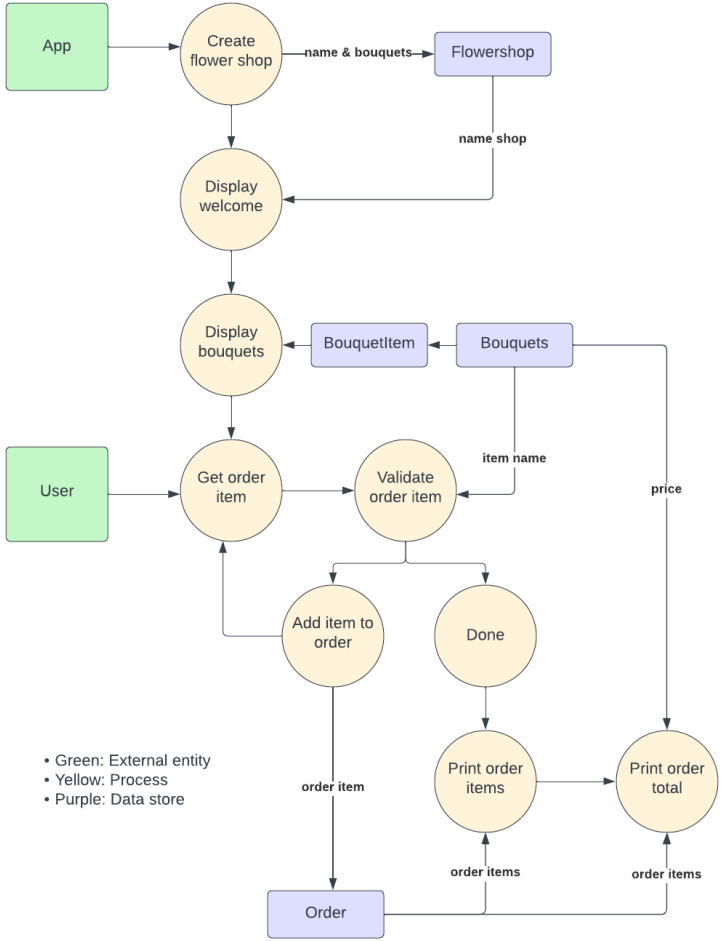
The Flowershop class

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

Code for Flowershop Class

```ruby
require_relative './bouquet'
require_relative './order'

class Flowershop
    attr_reader :name, :bouquet
    def initialize(name, bouquet_items)
        @name = name
        @bouquet = Bouquet.new
        populate_bouquet(bouquet_items)
        @order =Order.new
    end

    def populate_bouquet(bouquet_items)
        bouquet_items.each do |name, price|
            @bouquet.add_item(name,price)
        end
    end

    def add_to_order(item,quantity)
        @order.add_item(item, quantity)
    end

    def get_order
        return @order
    end

    def welcome
        puts "Welcome to #{@name}!".colorize(:red)
        puts
    end

    def print_bouquet
        @bouquet.display
    end

    def order_total
        total = 0
        @order.get_items.each do |item,quantity|
            total += @bouquet.get_price(item) * quantity
        end
        return total
    end

    def print_order
        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 ebb6a77e9783d6bd0c06403cdd3784aac8a4eaaa (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 c3173aebd26ac5ea1dab988f6687afbb10c5e15e (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
```