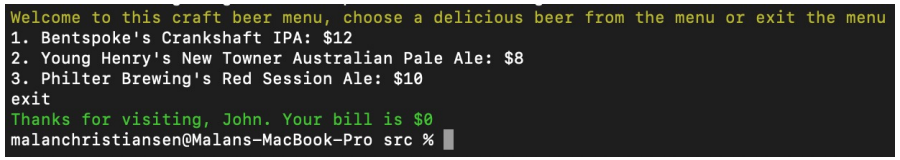
Terminal Application Beer Menu

Malan Christiansen

An overview of the terminal app

- Displays a welcome message incl. title, welcome, and statement to choose a beer
- Menu options: displays beer menu with 3 beers
- User input: prompts user for an order
- User input: prompts user for quantity of selected beer
- Displays goodbye message
- Exit option: user can quit menu
- Error handling: user is helped with invalid input



An overview of the terminal app

- Example of user typing ‘exit’ to exit the app. The gem colorize is used to make the text green for color coding.

- Example of user typing the wrong value, 5. The user is prompted with a message giving the correct values to choose from. The gem colorize is used to make the text red for color coding.

The main features and overall structure of the *app*

Welcome message

Menu options

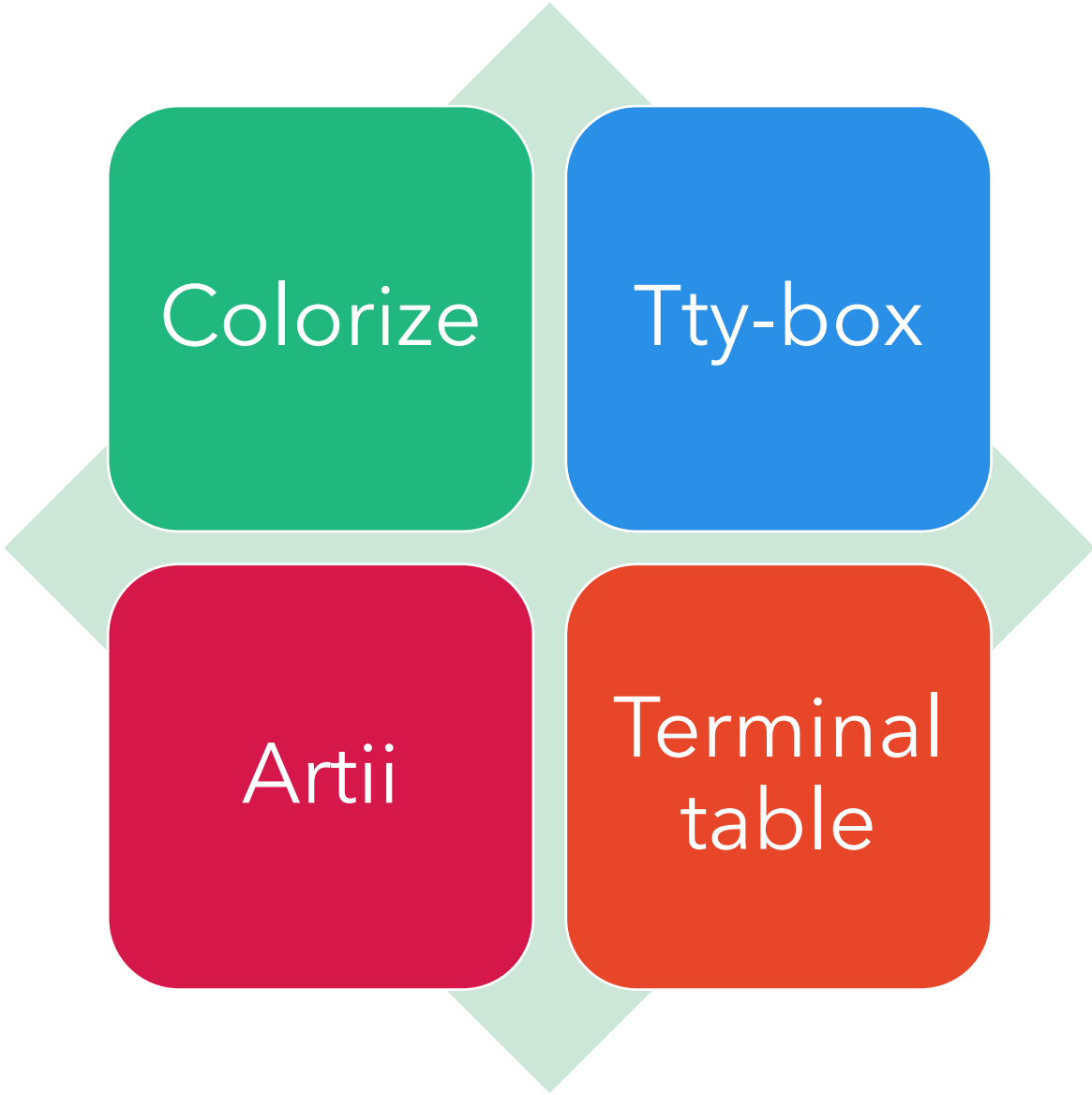
User Input

Calculation of final bill

Goodbye message

Exit option

Gems being used



An overview of the code

- Using a while loop for user input:
  - If the user 1, 2, or 3, the app continues and aske the user next question.
  - If the user types exit, the user exits the app and is displayed a goodbye message with a bill of $0 before the app stops running.
  - If user types an invalid value, e.g. a number that is not on the menu, the user is displayed a message that tells which numbers the user can choose or if the user wants to exit.

```ruby
ordering = true
while ordering
#get user input
    user_choice = gets.chomp.to_i

    case user_choice
        when 1,2,3
            return customer.place_order(user_choice)
        when 0
            ordering = false
            puts "Thanks for visiting! Your bill is $#{customer.bill}".colorize(:green)
        else
            puts "Invalid choice, please type 1,2,3 or exit to leave the app".colorize(:red)
    end
end
```

An overview of the code

- Created the class CustomerOrder
- Notice the beer menu is created in a hash
- To display the hash in a terminal table, the hash is added to the rows

```ruby
class CustomerOrder
    attr_reader :bill, :name
    def initialize(name)
        @name = name
        @bill = 0
    end
    def welcome_msg
        "Choose a delicious beer from the menu or exit the menu".colorize(:yellow)
    end
    def menu
        menu_items = {"1. Bentspoke's Crankshaft IPA" => 12, "2. Young Henry's New Towner Australian Pale Ale" => 8, "3.
        Philter Brewing's Red Session Ale" => 10}
        # update the menu list (contains item number and price) with the right price if the menu items are updated otherwise
        CODE WILL NOT WORK
        @menu_list = {1 => 12, 2 => 8, 3 => 10}
        menu_items.each do |item, price|
            "#{item}: $#{price}"
        end
    end
    def get_item_price(user_choice)
        @item_price = @menu_list[user_choice]
```

```ruby
rows = customer.menu
table = Terminal::Table.new :title => "BEER MENU", :headings => ['Beer on tap', 'price $'], :rows => rows
puts table
```

Error handling

```ruby
def get_quantity
    puts "How many would you like?".colorize(:blue)
    begin
        @quantity = gets.chomp.to_i
        if @quantity == 0
            raise TypeError
        end
    rescue
        puts "Please type a number greater than 0"
        retry
    end
end
```

- Error handling is being used when the app prompts the user for quantity. If the user types 0 or a string it is converted to an integer and a message is displayed to let the user retry to type a correct value.

Test Driven Development

- Before programming the app, I used test driven development (TDD) to aim for a bug free app
- I set up test cases for the features of the app
- I modified the code in order to pass the tests I had set up
- I used rspec for TDD

```ruby
require_relative "../customer_order.rb"
require 'colorize'

describe CustomerOrder do
    before(:each) do
        @customer = CustomerOrder.new("John")
    end
    it "instance must have a readable name" do
        expect(@customer.name).to eq("John")
    end
    it "displays welcome message" do
        msg = "Choose a delicious beer from the menu or exit the menu".colorize(:yellow)
        expect(@customer.welcome_msg).to eq(msg)
    end
    it "display the menu" do
        menu = {"1. Bentspoke's Crankshaft IPA" => 12, "2. Young Henry's New Towner Australian Pale Ale" =
        Brewing's Red Session Ale" => 10}
        expect(@customer.menu).to eq(menu)
    end
    it "get price on beer based on user choice" do
        @customer.menu
        user_choice = 1
        expect(@customer.get_item_price(user_choice)).to eq(12)
    end
    it "should get the quantity" do
        expect(@customer.get_quantity).to be > 0
    end

    it "should calculate the final bill" do
        @customer.menu
       user_choice = 1
       final_bill = @customer.get_item_price(user_choice) * @customer.get_quantity
        expect(@customer.calculate_bill).to eq(final_bill)
    end
end
```

Questions?