Pogress on CLIMate

CLIMate Overview

- A command line weather app that enables the user to search for place names (Nominatim API)
  and view the current or weekly forecast weather data for that location (if valid)
- Maintains a list of:
  - User locations
  - Favourites
  - History
- Weekly weather forecasts can be exported in the form of PDF documents.

App Architecture

Object Oriented Approach

- CLIMate includes a custom built ArgumentParser class for parsing command line arguments
- All app logic is organised in an App class, which is initiated by calling its ‘exec’ method.
- App configuration is managed by a ConfigManager class:
  - Reads from and writes to four JSON configuration files, and maintains instance variables that store the configuration state for each of these files.
- These three classes are instantiated in main.rb – the app is bootstrapped and initial command line argument processing takes place

main.rb

ConfigManager is instantiated and initialised. App will exit if errors

Parser is instantiated and its ‘parse’ method is called with ARGV. App will exit if errors

The app does not accept non-option arguments, so exit if they are present

main.rb

src > main.rb

```ruby
require_relative "./lib/console"
require_relative "./config_manager"
require_relative "./parser/parser"
require_relative "./app"

config_manager = ConfigManager.new
config_errors = config_manager.init

if config_errors
  Console.info("Oops! There were some configuration errors:")
  config_errors.each do |error_message|
    Console.error(error_message)
  end
  Console.info("Please run the setup script to sort things out!")
  exit
end

parser = Parser.new # Initialize command line argument parser.
results = parser.parse(ARGV)

if results[:errors].length > 0
  # There were parsing errors, so display them and exit:
  Console.info("Please fix the following command line argument errors:")
  results[:errors].each_with_index do |error, i|
    Console.error("#{i + 1}. #{error}")
  end
  exit
end

arguments = results[:args]

if arguments.length > 0
  # This application does not support
  # non-option arguments. Display errors
  # and exit:
  arguments.each do |arg|
    Console.error("Unexpected token: '#{arg}'")
  end
  exit
end
```

main.rb continued

The app only accepts one option at a time. Exit if multiple options have been provided.

```
42  options = results[:options]
43
44  if options.length > 1
45    # This application does not support
46    # multiple options in the one command.
47    # Display errors and exit:
48    Console.error("Please provide CLIMate with ONLY ONE option.")
49    Console.info("You can use the '--help' or '-h' option for help :)")
50    exit
51  end
52
53  app = App.new(config_manager)
54  app.exec(options[0])
```

Here we instantiate the app class with the ConfigManager object and call its ‘exec’ method with the first option (this may be nil)

Modules

- CLIMate source includes four modules:
  - Console
    - Input/output methods
  - Timezone
    - Methods for configuring the user's timezone – worldtimeapi.org
  - Geocode
    - Methods for searching Nominatim for geocode information required to fetch weather
  - Weather
    - All weather API and helper methods – using Open-Meteo for weather data

Gems

- Using 5 gems for this project:
  - http – for making API requests
  - json – for parsing and generating JSON
  - tty-prompt – for gathering user input
  - colorize – for colouring console output
  - prawn – for generating PDF forecast reports

exec

When nil is passed as the option argument, ‘main_loop’ is executed, which provides the primary logic for the application. If an option is present, ‘exec’ invokes the corresponding handler method with the option’s arguments

```
19
20  def exec(option)
21    if !option
22      self.main_loop
23    else
24      case option[:name]
25      when "help"
26        self.help
27      when "config"
28        self.config(option[:args])
29      when "history"
30        self.history(option[:args])
31      end
32    end
33  end
```

The config handler is not interactive. The history handler is (like ‘main_loop’)

main_loop

Begin by fetching the user's timezone (required for weather API queries)

User selects the location type, then the app guides them through chosing/searching for a location to view the weather for

Please excuse the lack of comments in these screenshots!

No location was decided on – ask if the user wants to try another location

app.rb

src > app.rb

```ruby
def main_loop
  self.print_welcome_message
  timezone = self.get_timezone
  if !timezone
    Console.info("CLIMate needs to know your timezone to fetch forecasts.")
    self.exit_gracefully
  end
  begin
    while true
      location_type = self.select_location_type
      location_info = nil

      case location_type
      when LOCAL
        location_info = self.select_location_from_user_locations
        if !location_info
          # The user elected to search for a new location:
          message = "Enter a place name for your current location:"
          location_info = self.get_location_from_user(message)
          if location_info && !self.in_user_locations(location_info)
            save_location = Console.yes?("Would you like to save this location?")
            if save_location
              self.save_user_location(location_info)
            end
          end
        end
      when ELSEWHERE
        location_info = self.select_location_from_favourites
        if !location_info
          # The user elected to search for a new location:
          message = "Enter a place name to search for:"
          location_info = self.get_location_from_user(message)
          if location_info && !self.in_favourites(location_info)
            save_location = Console.yes?("Would you like to save this location?")
            if save_location
              self.save_favourite(location_info)
            end
          end
        end
      end

      if !location_info
        if !Console.yes?("Would you like to view the weather for a different location?")
          self.exit_gracefully
        end
        next
      end
```

main_loop continued

At this point in the main loop a location has been chosen The user selects a forecast type and the app fetches weather data according to their choice

```
      forecast_type = self.select_forecast_type
      current_weather = nil
      weekly_forecast = nil

      case forecast_type
      when CURRENT
        current_weather = self.get_current_weather(timezone, location_info)
        if !current_weather
          message = "Sorry - CLIMate couldn't get weather data for #{location_info["display_name"]}"
          Console.info(message)
        else
          Console.success("Success!")
          self.print_current_weather(location_info["display_name"], current_weather)
        end
      when COMING_WEEK
        weekly_forecast = self.get_coming_week_weather(timezone, location_info)
        if !weekly_forecast
          message = "Sorry - CLIMate couldn't get weather data for #{location_info["display_name"]}"
          Console.info(message)
        else
          Console.success("Success!")
          output_type = self.select_output_type
          case output_type
          when PRINT_TO_CONSOLE
            self.print_coming_week_weather(location_info["display_name"], weekly_forecast)
          when EXPORT_TO_PDF
            self.generate_forecast_pdf(location_info["display_name"], weekly_forecast)
          end
        end
      end

      if current_weather || weekly_forecast
        if Console.yes?("Would you like to save this weather data to your history?")
          if current_weather
            self.save_to_history(location_info["display_name"], forecast_type, current_weather)
          elsif weekly_forecast
            self.save_to_history(location_info["display_name"], forecast_type, weekly_forecast)
          end
        end
      end

      if !Console.yes?("Would you like to view the weather for another location?")
        self.exit_gracefully
      end
    end
  rescue SignalException
    self.exit_gracefully
  end
end
```

If weather data was successfully retrieved (either current weather or the coming week's forecast) the user has the option to save the data to their history

Ask the user if they would like to view more weather. Exit the app if not

Development Process

Challenges

- Processing all the data returned from Open-Meteo was probably the biggest challenge – lots of typing…
- I was able to make the code fairly generic using iteration with hash keys.
- Keeping my Trello board up-to-date has been another challenge!

Ethical Issues

- The main ethical issue I can think of is that the app can be used to save a history of the user's previous locations (maybe if someone found that information on a stolen computer the could use it in data aggregations to build a more detailed profile of the user). However, this is an optional feature.

Favourite Parts

- Without a doubt my favourite parts were designing the app architecture, and creating the configuration system and argument parser.
- Also loved making the setup bash script!

Thank you for your time :)