Toh Ker Wei - Project Portfolio

Project: FitBiz

Overview

FitBiz is a fitness business management application, specially created for fitness coaches to manage their clients. It is a platform where coaches can store his clients' information, exercises done and weekly training schedule. FitBiz is primarily a desktop application where the user interacts with the program via the Command Line Interface (CLI), and views data via the Graphical User Interface (GUI). This project is written in Java 11, packaged using Gradle, and uses JavaFX for the GUI.

Summary of contributions

- **Major enhancement**: added the **filter** feature which allows user to filter and display the clients list using client's `Tag` and `Sport`
  - What it does: Allows user to filter through his list of clients using `Tag` or `Sport` keywords and return clients that have `Tag` or `Sport` matching all of the keywords specified.
  - Justification: Allows user to group and view clients based on their `Tag` or the `Sport` they play, which is a convenient way of looking at similar groups of client when planning exercises or scheduling clients with similar needs to train as a group. This increases the efficiency of viewing clients from a long list based on their attributes and to find clients' `INDEX` to view their information.
  - Highlights: Provides a simple yet important way of grouping and viewing clients from the list. The implementation was done in such a way that makes it easy to expand to filter all attributes of clients allowing user to make training plans more robust. Future expansion plan includes the abilities to filter clients by ranges of their current and target weight to group those with similar needs and filter by address to allow user to schedule clients living near each other on the same day.
- **Major enhancement**: helped to build the foundation of **schedule** feature
  - What it does: Allows user to add and view the training schedule of clients
  - Justification: Users will be able to add, store and view clients schedule and therefore eliminating the need to use another application for the same purpose. This increases the efficiency of users while planning for clients' trainings as the schedule, exercise history and information can all be viewed in the same page.
  - Highlights: We have essentially created a mini version of a calendar app that is simple yet maintain the important functionalities to add training schedules and display them in a weekly view. Many real life conditions for the training schedule added such as overlapping training time, whether to allow backlogging of training dates and clients having same training time have been resolved through multiple trials and discussions.

- **Minor enhancement**: added `Sport` attribute to `Client`
- **Code contributed**: [RepoSense]
- **Other contributions**:
  - Enhancements to existing features:
    - Wrote test code for classes regarding filter to improve test coverage (Pull request #231)
    - Wrote test codes for `Sport` attribute and part of `Schedule` classes to improve test coverage (Pull request #46, #62, #141)
  - Documentation:
    - Wrote the command guide for `add-c` and `filter-c` in User Guide (Pull request #260, #273)
    - Wrote the implementation of `filter-c` and `view-c` in Developer Guide (Pull request #248, #272)
    - Edited `Logic` to reflect the implementation of FitBiz in Developer Guide (Pull request #248)
  - Community:
    - Reported bugs and suggestions for other teams in the class that were resolved or implemented (1, 2, 3)

Contributions to the User Guide

Given below are the sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Filter clients based on attribute: `filter-c` --- Toh Ker Wei

When you have many clients and want to filter the list of clients to view a specific group, you can use the command `filter-c` to filter clients based on their tags or their sports.

Format

Format: `filter-c [t/TAG]... [s/SPORT]...

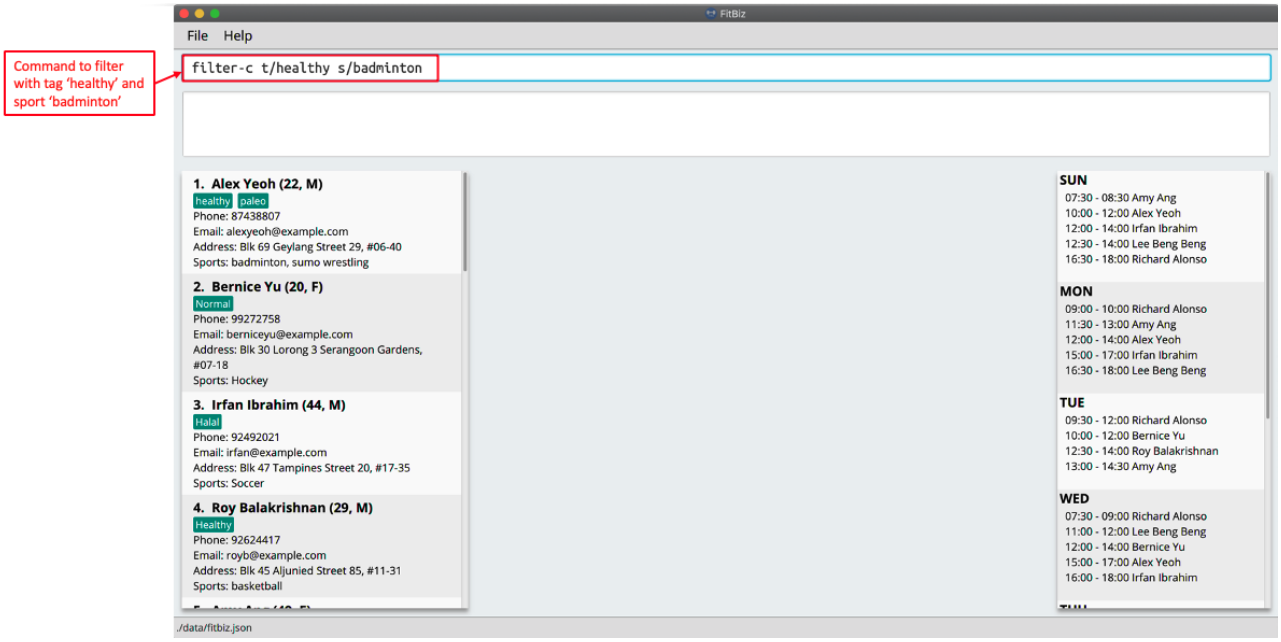
Parameter	Important notes
`t/TAG`	`TAG` is the tag of the clients you want to match and list. `TAG` is case-insensitive. e.g. `healthy` will match `Healthy` `TAG` should only contain letters or numbers. e.g. `monday` or `obese200kg`

Parameter	Important notes
`s/SPORT`	`SPORT` is the sport of the clients you want to match and list. `SPORT` is case-insensitive. e.g. `track and field` returns the same result as `Track And Field` `SPORT` should only contain letters, numbers or spaces. e.g. `sumo wrestling` or `100m sprint` Order of words in `SPORT` does not matter e.g. `track and field` returns the same result as `field and track`

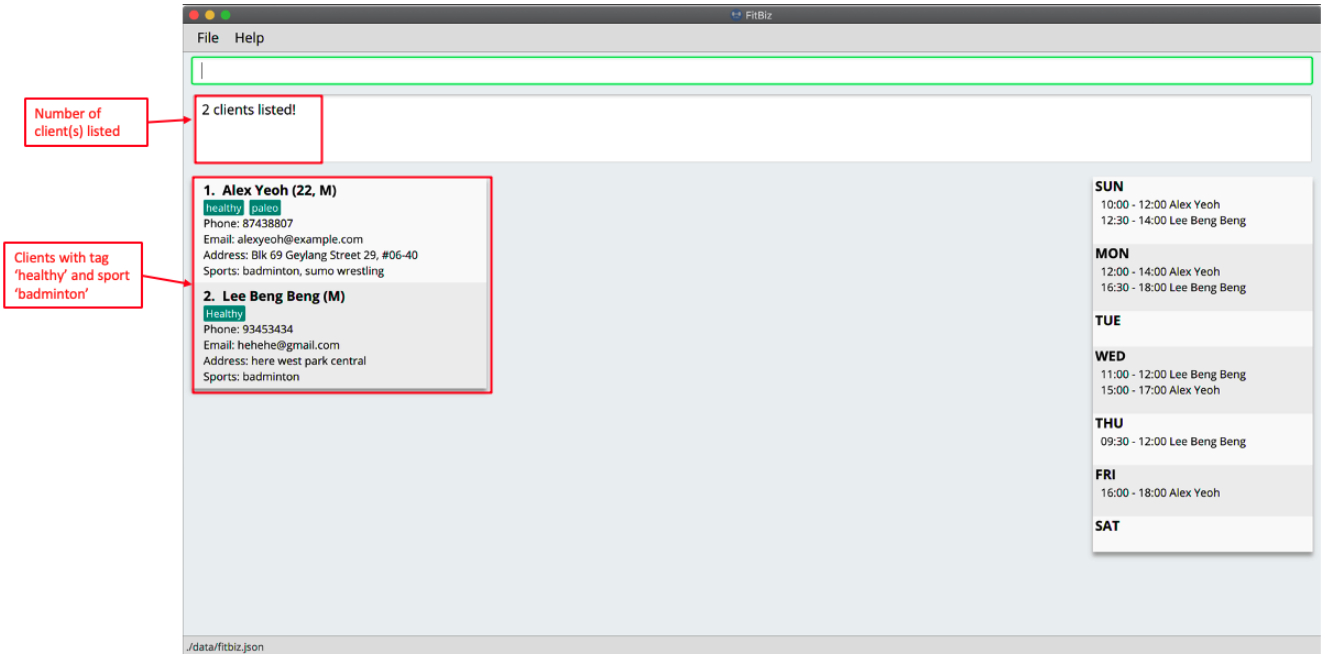
Example

Let's say you want to filter through your list of clients and only display those with the tag `healthy` and play the sport `badminton`.

1. Type the command `filter-c t/healthy s/badminton` into the command box.



2. Press enter to execute.
3. The clients with the matching tag and sport will be displayed as shown.

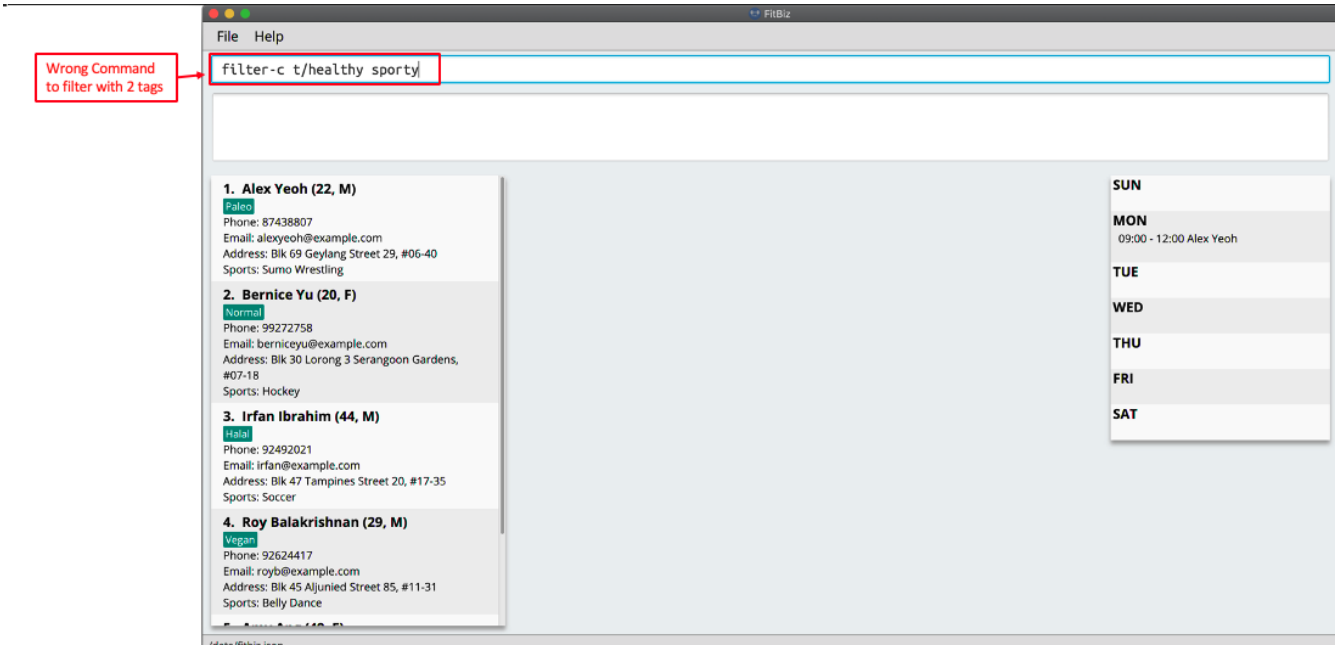


/ photo of listed clients

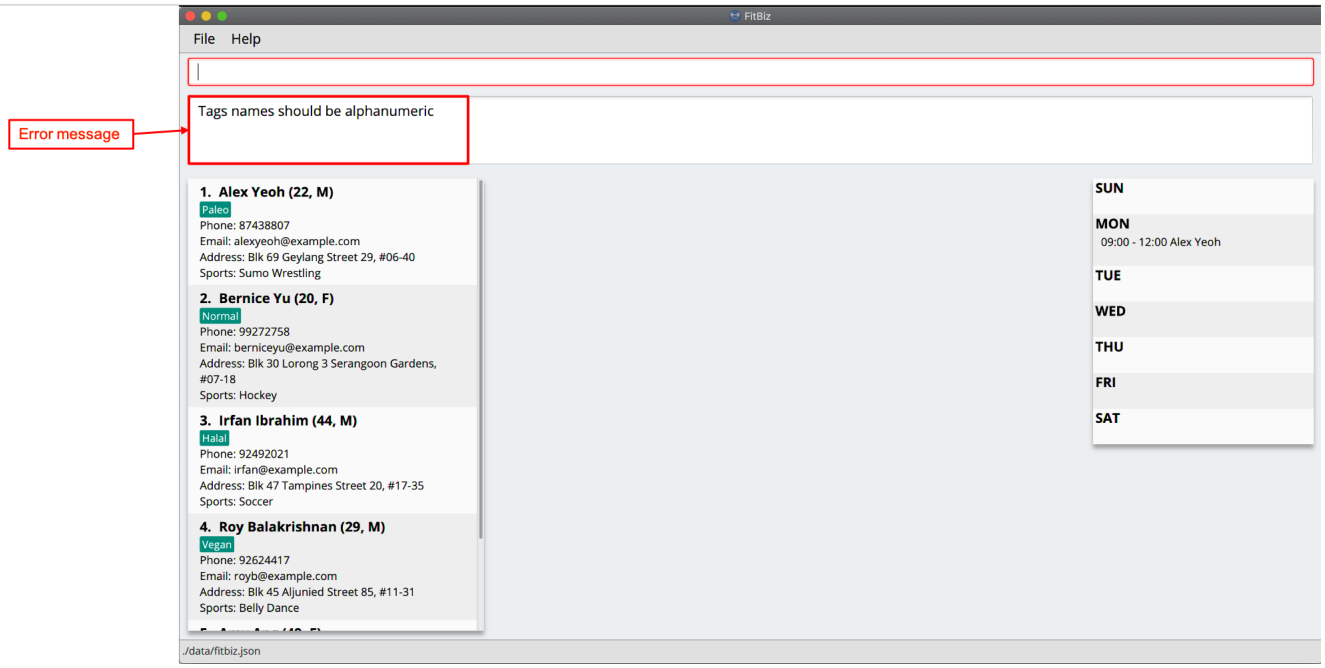
Common error/ problem

Tags with spaces

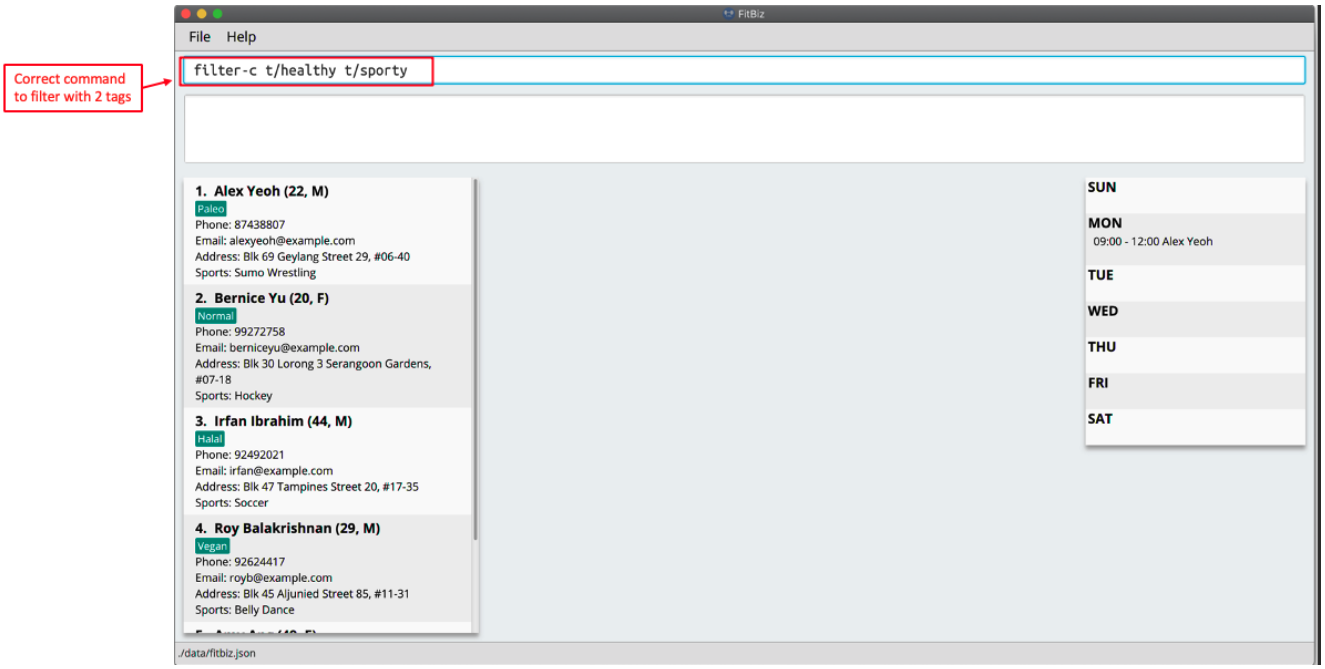
When you want to filter the clients list with multiple tags like `healthy` and `sporty`, you might enter the command shown below.



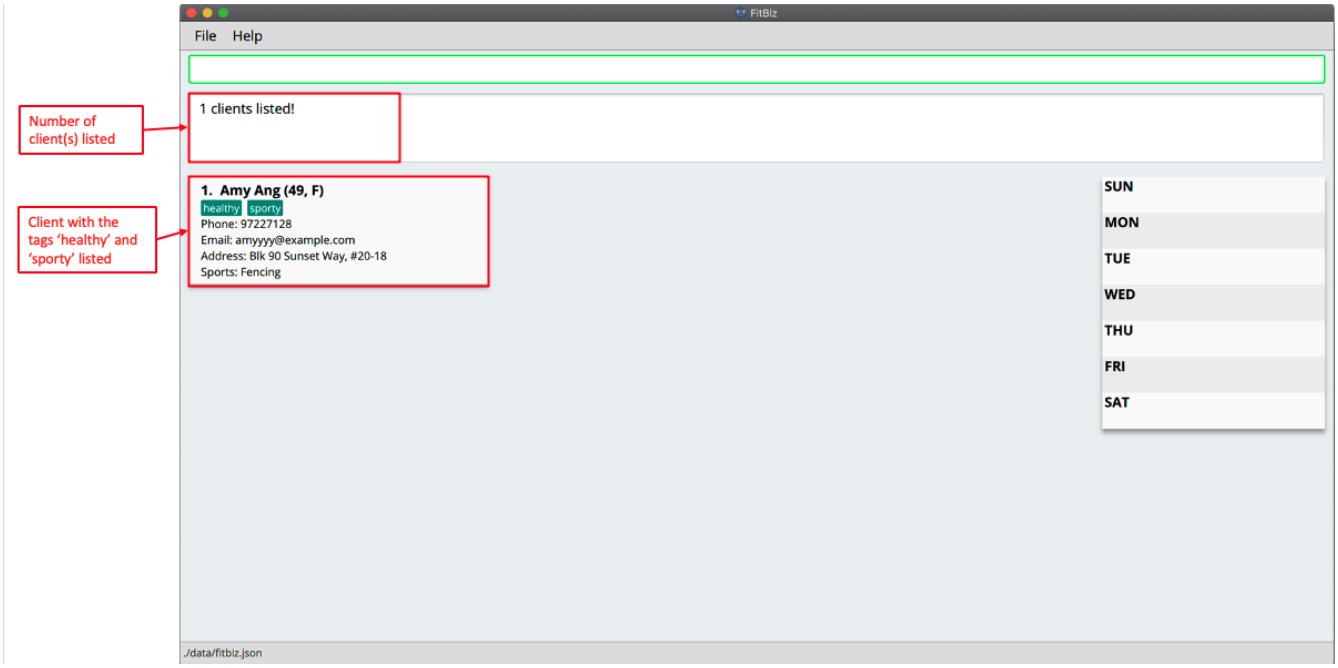
You will then encounter the error `Tags names should be alphanumeric`.



This error occurs because `TAG` only accepts letters and numbers but not spaces. To solve the problem, add an additional delimiter for each tag you want to specify. Note that sports does not require multiple delimiter.



The list of successfully filter clients will then be displayed.



Other contributions to the User Guide: `add-c`

Contributions to the Developer Guide

Given below are the sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

Filtering the list of clients --- Toh Ker Wei

This feature allows users to filter the list of clients by specifying the `Tag` or `Sport` of the clients they want to view.

Implementation

This filtering mechanism is facilitated by `TagAndSportContainsKeywordsPredicate`, that implements `Predicate<Client>` which is a wrapper class for a boolean. `FilterCommand` is associated with `Model` is responsible for calling `Model#updateFilteredClientList` based on `TagAndSportContainsKeywordsPredicate`. `TagAndSportContainsKeywordsPredicate` will call `test` on `Client` to check if the clients 'Tag' and `Sport` contains all the keyword. the relations between these classes are shown in the class diagram below.

[FilterClassDiagram] | *FilterClassDiagram.png*

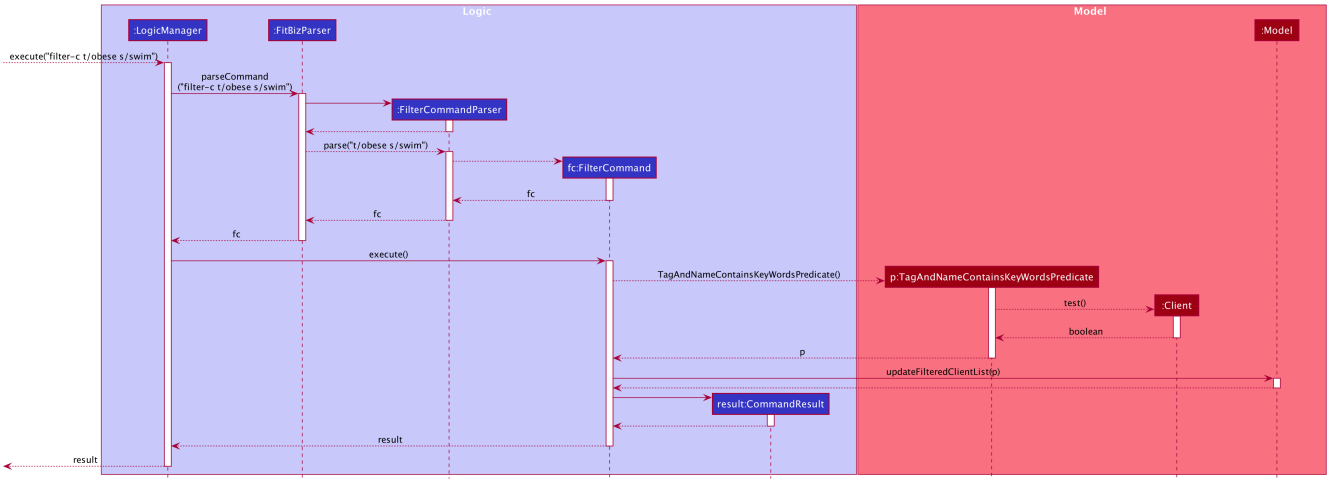
To further elaborate, `TagAndSportContainsKeywordsPredicate` contains 2 booleans:

1. `hasTag`: evaluates if the client has all the `Tag` specified
2. `hasSport`: evaluates if the client has all the `Sport` specified

If there is no keyword specified for either `Tag` or `Sport`, the corresponding boolean will return `true`. There must be at least 1 keyword specified, regardless of whether it is a `Tag` or `Sport`. `TagAndSportContainsKeywordsPredicate` will then evaluate and return the logical addition of `hasTag`

and `hasSport`.

In the following sequence diagram, we will be tracing the execution of the command `filter t/obese s/swim` entered by the user.



Design Considerations

Table 1. Table of Design Considerations

	Using separate booleans to check for `Tag` and `Sport` keywords (Chosen)	**Using one boolean to check for all keywords**
Ease of Implementation	Checks for client's `Tags` and `Sports` containing keywords can be done separately ensuring that individual results are correct before combining them	Simpler logic but errors are more difficult to pinpoint to either `TAG` or `SPORT`
Ease of Expanding Feature	Easier to add new parameters to filter since a separate check will be done before combining with the result of previous checks	Boolean conditions can get very complex and logical error will be prone to occur

We decided to use the first approach of checking if the client contains `Tag` specified and `Sport` specified separately.

Firstly, by separating the checks for each attributes, a correct implementation of checking `Tag` against the keywords will allow us to easily duplicate the logic to be done for `Sport`. This makes the code easier to debug as we can simply check the hasAttribute boolean to see if it gives the correct value.

Secondly, separating the checks for each attributes will allow us to add attributes of different types stored in different data structure easier. We could simply add another check on the attribute against the keyword specified then do a logical addition of the result against the others.

Therefore, as we foresee us adding more attributes to be filtered increasing the need to ensure

logical correctness, the first approach is the most ideal.

Other contributions to the Developer Guide:

- `Logic`
- `view-c`