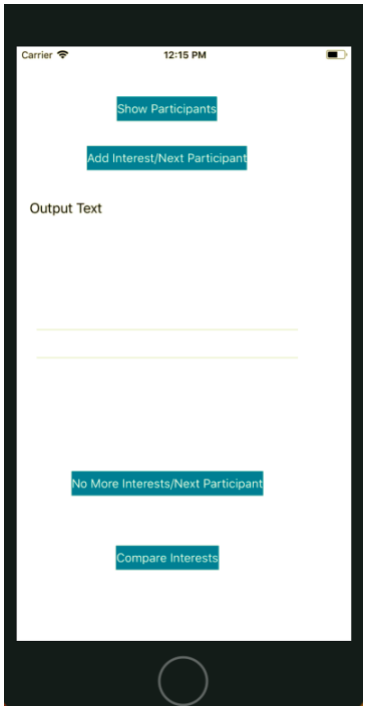
Curious Katie

Initial view & Operation



Button/Output	Use
Show Participants	Show a list of participants on the XCode debugger console
Add Interest/Next Participant	Participants click this button to add an interest and the next participant is randomly selected
Output Text	This UILabel will show the participant and advise when interest selection is complete
UIPickerView	This holds the interests the participants can select
No More Interests/ Next Participant	When a participant has decided they do not wish to add any further interests, this button will mark their interests as complete
Compare interests	This button shows a list of the participants with all of their interests and a suggested conversation pairing based on interests. All shown on the XCode debugger console

When the application is run pressing the *Show Participants* button shall print a list of participants, between 2 and 12, in the XCode debugger console with attributes for each.

Next each participant can select up to 10 interests from the picker view, one participant at a time being randomly selected. Once they have finished picking their interests they can press the *No More Interests* button, meaning they will not be asked to add any further interests.

Once all the participants have selected their interests, the *Compare Interests* button can be pushed, and this will show all the participants with their seletced interests in the XCode debugger console, followed by paring suggestions for each.

XCode

What I have used any why…

Object Orientated Programming

Using OOP allowed me to create the objects for person and interests using methods (generateParticipants and generateGeneralHobbies) to create the data to be used in the rest of the project, whilst using private lets to ensure only that file can access the data (encapsulation & abstraction). Using *inheritance* to use the interests object in the person object.

Enum

Within the Person class I have used enum for Gender. Enumerating allows me to create a group of related values, giving a type-safe way to use the data throughout the code.

Equatable & Hashable

Class Person required Hashable for the Set to compare the custom objects. In the Interest Class I have used Equatable to allow me to easily compare the custom objects in ViewController to check for non-equal participant interests.

Filter

I have used the filter function to filter out the participants with the finishedAddingInterests flag as true, to only show the interest in the UIPickerView that have not been selected by the current participant and to compare non-equal interest between participants.