**Comp 1501 Assignment #6 (worth 8% of overall mark)**

**Given:** Thursday, April 2, 2015

**Due Date**:

Thursday, April 16, 2015, 11:59 p.m. sharp

Note: **no late assignments accepted**

**NOTE:** Read the entire assignment before starting anything! Thinking about how you will design things is extremely important in this assignment.

**Assignment Overview**

A program is required for a computing dating service. Clients will give you their information, including a number of interests. The system will keep track of clients using the service and match up compatible clients when requested to do so.

**Specifications**

1. **Client Class**

A class called **Client** is required; it should have the following fields:

* name: the person’s name (String).
* gender: the person’s gender (String – “M” or “F”).
* genderSought: the gender of person this client would like to be hooked up with (String – “M”, “F” or “E” – meaning either).
* interests: an ArrayList of String to hold this person’s interests.
* myMatchIndex: the index in the Client ArrayList (see MatchMaker class below) of the client currently being matched to the current client (int)

Realistically there should be others (e.g. id, phone), but to keep the assignment a reasonable size, this will be enough.

The methods required are:

* Client (String name, String gender, String genderSought)
  + creates a Client with no interests

standard getters / setters – no error checking required for setters (or for constructor)

* String getName()
* String getGender()
* String getGenderSought()
* int getMyMatchIndex()
* ArrayList<String> getInterests()
* void setName (String name)
* void setGender (String gender)
* void setGenderSought (String genderSought)
* void setMyMatchIndex (int myMatchIndex)
* void addInterest (String interest)
  + adds one interest to the client’s interests. If there is no more room in the interests ArrayList to store this, ignore it – clients know the maximum number of interests allowed. The number of interests must be between 3 and 10.
* boolean isGender (String gender)
  + returns true if client’s gender is the same as the gender passed in
* boolean isGenderSought (String genderSought)
  + returns true if client’s genderSought is the same as genderSought passed in
* int numMatchingInterests (Client otherClient)
  + returns the number of interests that this client has in common with otherClient
* String toString()
  + returns the client data (name, gender, gender sought, interests, matched?) in the the format illustrated below:

ANDREW JOHNSON M F PIANO POKER READING SAILING (Matched)

NANCY DREW F E GAMING PROGRAMMING WALKING (Not Matched)

1. **MatchMaker Class**

A class called **MatchMaker** is required; it should have the following fields:

* + allClients: **one** ArrayList of all the clients

The methods required are:

* MatchMaker (String fileName) throws IOException
  + this constructor initializes / instantiates the allClients array list. It also reads all the client data from the file specified. To do this, you will make a call to DatingSystemHelper.readResults (filename, allClients) which you will implement as well. The readResults throws an IOException, so you need the “throws” clause as shown in the function header.
* void printClients ()
  + prints all the data in the allClients array list.
* void printClients (String gender)
  + prints all the data on all men (if the gender is “M”) or all women (if the gender is “F”)
* void printMatches()
  + prints all the matches (i.e., the names of both of the people who are matched). There should only be one line for each matched pair (this requires a minor check – do not try anything complex – it’s not necessary).
* void printUnmatchedClients()
  + prints the names of all unmatched clients in the array list.
* void printUnmatchedClients (String gender)
  + prints the names of all unmatched men or all unmatched women of given gender
* String addClient (Client newbie)
  + This adds newbie to allClients array list. It then calls DatingSystemHelper.makeMatch (newbie, allClients) which is already implemented for you.

1. **A6**

A menu-driven driver program (A6) has been provided. Make no changes to it except as specified in the code (mostly just replacing comments with calls to DatingSystemHelper method and MatchMaker method). Use it to test that your program works as required.

1. **DatingSystemHelper**

* public static void readResults (String filename, ArrayList<Client> allClients) throws IOException
  + this does the actual reading. The file has been created by the program, so there is no need to error check. It contains all the data for all clients. The last line of the file ends in a carriage return. The data for each client is in the format:
    - line 1 – name (first last)
    - line 2 – client gender (M or F), followed by a space, followed by gender sought (M, F or E)
    - line 3 – number of interests, followed by the interests. A blank occurs between all data fields
    - line 4 – the index of the client with whom this client is matched, or -1 if not matched.
  + As you read each client you will add it to the allClients array list.
* public static String makeMatch (Client clientToMatch, ArrayList<Client> allClients)
  + This will find a match (if possible), and designate both persons as matched as described above. It returns a String which returns either the names of the newly matched partners, or an explanation of why a match was not made. This code is provided **so do not change it!**
* public static Client addClient (Scanner kb)
  + use terminal window with keyboard input to create a new client.

**SUBMISSION INSTRUCTIONS**

Copy your BlueJ project folder (with all its contents) to the “submit” folder, which appears under the “I:” drive each time you log in to a university PC. If you are submitting from a non-university computer (e.g. a home PC), you can access the submit folder via secure.mtroyal.ca (in which case you must upload a “.zip” archive of your folder). Your submitted folder name must follow the format illustrated below:

<LastName>\_<FirstName>\_A6