CSC Match Requirements

1. Methodology

This project will be done in teams and will include the following phases:

A. Design and Design Review
B. Implementation and Code Review
C. Test Case Development and Testing

Team membership will be assigned by the instructor.

2. Problem Description

Every social group today has a way to connect with like-minded individuals (e.g. FarmersOnly.com). You've been tasked with designing, implementing and testing an application that can track CS students and their interests. This app will be used to find other students with similar interests for study sessions. You can find the design, functional and non-functional requirements below.

Design Requirements

1. Create class diagram(s) in StarUML, showing
    a. All of the Classes
    b. All of the Attributes
    c. All of the Operations
    d. All of the Relationships (inheritance, realization, association, dependency)
2. As appropriate, follow the good Design Principles discussed in class
    a. Separate the Common from the Specialized: Create a hierarchy (either Interfaces or Classes) that recognizes and exploits the common and the exceptional.
    b. Separate the Data from the Container: Use Generics in your Collection definitions and implementations.
    c. Separate the What from the How: Interfaces define what the concept is; Classes define how a concept is implemented.
3. Existing Code:
    a. You may use any class/interface from the Java Collections Framework (JCF) whether we covered it in class or not.
    b. You may use any code found in the book. Some of this can be found on Canvas, under our Project Module, in Collections Code. Anything else you want can be typed in.
    c. If using existing classes/interfaces are used, there is no need to include their attributes or operations in the design.
4. Other:
    a. Where appropriate, override toString() to assist in printing objects.

b. Your Member class should implement Comparable to allow Members to be matched, per the algorithm outlined below.
c. A Member's Interests should be maintained in a Sorted List.

Functional Requirements

1. The user should be presented with a menu of the following activities:
    a. Load the Members
    b. Save the Members
    c. List All Members
    d. Add a Member
    e. Remove a Member
    f. List Member
    g. Add an Interest to a Member
    h. Quit
2. Each Member includes a name, and year (1-5).
    a. A freshman is a 1, sophomore 2, etc. A CS student with a degree is a 5.
3. Each Interest includes a topic (e.g. "Assembly Language", "Java", "Digital Design", "Cyber Security", "Web Design", "C++", etc.) and an interest level, from 0-10 (least to most).
    a. There is no predefined list of Topics, the user can enter anything they want.
4. Each Member may have any number of Interests (including none).
5. When adding a new Member, make sure their name is unique.
6. When saving the Members, ask the user for a filename. If the file cannot be written, tell the user and ask for another name.
    a. Its ok to overwrite an existing file. No warning is necessary.
7. When loading the Members, ask the user for a filename. If the file cannot be read, tell the user and ask for another name.
8. When adding an Interest to a Member, if the Interest already exists, replace the Interest with the new one.
9. When Listing All the Members, list only their names in the order they were added to the application.
10. When Listing a Member, list his/her name, year and list of interests, from highest interest to lowest interest with the score. Also, list the name of his/her top five "Matches".
    a. The name and order of top matches may change as Members gain additional Interests, or their interest level's in certain topics change.
11. A Member is matched to another by computing a "Compatibility Score", using the rules listed below. Note that A's interest in B may not be the same as B's interest in A.
    a. If a topic matches, the score is increased by (my_interest x their_interest). A compatible interest!
    b. For any topics they have that I don't have, increase the score by (their_interest/2), rounded down. The other person is interesting in general.
    c. Example: What is the score for A compared to B?

Member A		Member B	
Topic 1	2	Topic 1	5
Topic 2	3	Topic 3	3
Topic 3	5	Topic 4	7
		Topic 5	1
		Topic 6	9

A's interest in B Score = (2 x 5) for Topic 1 match + (5 x 3) for Topic 3 match + 7/2 for Topic 4 + 1/2 for Topic 5 + 9/2 for Topic 6 = 10 + 15 + 3 + 0 + 4 = 32

d. Example: What is the score for B compared to A?

B's interest in A Score = (2 x 5) for Topic 1 match + (5 x 3) for Topic 3 match + 3/2 for Topic 2 = 10 + 15 + 1 = 26

12. When quitting, if the user has made any changes since they last saved, warn the user. If they still insist on leaving, allow it. Otherwise, cancel the "Quit" so the user can go back and save.
13. Any requirement which is unstated is left to the design team to invent. Document this.

Non-Functional Requirements

1. The main() must be in a class called CSCMatch.
2. You may use package(s) if this helps organize your solution.
3. Catch IOExceptions and handle them gracefully.
4. The application to be usable without prior training (the interface should be self-explanatory).

3. Required Artifacts

Each phase of the project requires that you produce and turn in artifacts. Be sure to understand the multiple assignments in Canvas for these artifacts.

A. Design and Design Review
   a. Design document: StarUML file, or Word document with embedded diagrams.
   b. Word document listing Design review comments and adjustments made.

B. Implementation and Code Review
   a. Turn in only your Java source files. However, be sure to turn in ALL source files (including any you may have taken from Canvas).
   b. Word document listing Defects found during the Code Review.

C. Test Case Development and Testing
   a. Test cases – Word document listing each of the test scenarios that the team intends to run.
   b. Test execution results – Word document with each test case listed, and the captured output.

D. Data File
   a. Create a sample datafile called "TestData.csc" with the following seven students and interests. I have included the matches that you should get if your program is working correctly:
      i. Member A: Freshman
         1. Topic 1: 2
         2. Topic 2: 3
         3. Topic 3: 5
      ii. Member B: Sophomore
         1. Topic 1: 5
         2. Topic 3: 3
         3. Topic 4: 7
         4. Topic 5: 1
         5. Topic 6: 9
      iii. Member C: Junior

1. Topic 1: 5
2. Topic 2: 10

iv. Member D: Senior
1. Topic 1: 4
2. Topic 3: 3
3. Topic 5: 1
4. Topic 7: 7

v. Member E: Graduate
1. Topic 2: 5
2. Topic 4: 5

vi. Member F: Freshman
1. Topic 3: 8

vii. Member G: Sophomore
1. Topic 2: 1
2. Topic 3: 5
3. Topic 4: 7

b. You should get the following matches:

Member A	C-40	F-40	B-32	G-31	D-26
Member B	G-64	E-37	D-33	C-30	A-26
Member C	E-52	A-42	B-33	D-24	G-15
Member D	B-37	C-25	A-24	F-24	G-18
Member E	C-52	B-42	G-42	A-18	D-6
Member F	G-43	A-42	B-33	D-29	C-7
Member G	B-70	E-40	F-40	A-29	D-20

E. Other

a. Describe the responsibilities for each member of your development team.

b. Describe any changes, additions or clarifications your team made to the requirements.