Assignment 6 Writeup

Main Relations between classes:

Our main class for this project is **RecommendationSystem**.java which is itself an abstract class and implemented by **RRecommendationSystem**(for random selection), **SRecommendationSystem**(for selecting users from start) and **ERecommendationSystem**(for selecting users from end). Upon running main function, first of all command line arguments will be parsed and if any input provided is wrong, then we throw appropriate exception(**CmdParser**). And if no error is present, we carry on with execution and then initialize our graph. We parse node csv file to get information of user and save all the users as vertices of this graph and then we draw edges between these nodes by parsing the edge csv file.
And in next step, we take the processing flag and call appropriate class. We have created one **Rule** interface that simply have one method -generateRecommendations() which is implemented by four classes corresponding to the 4 rule criterion which we have to use for generating recommendations. After processing recommendations from rule 1, we check the number of recommendations and if it is equal to the required recommendations, we stop there itself and otherwise we move to Rule number 2. And we perform this step for all the rules. There can be some users for which we can never generate the required number of recommendations, so for those users, we have introduced one check inside rule 4 class.

After all the required recommendations have been generated, we write all these lists into our output file. (**CsvGenerator**). And we use **Analyser** to get the top ten most frequently recommended node IDs and print their Ids to console.

How you handle errors and exceptions

If there is anything wrong with command line arguments , then we raise IllegalCmdArgumentException. And for all the users present in node csv file, if there is anything wrong with any detail of user, we raise appropriate exception. For any relation present in edge csv file, if we encounter some user whose id is not present in node csv file, then we simply print this information on console and does not stop the execution of program in any way. And if there is any user , for which we can never generate the required number of recommendations - As an example, consider the users with node id 96 and 99 in edges-small.csv. User 96 and User 99 already follow 92 and 87 people respectively and suppose our required number of recommendations is 15. So, the maximum number of recommendations system can provide is just 7 and 12 (considering, one person cannot follow himself), so we simply output just those recommended users and print on console this information and we do not stop execution in any way

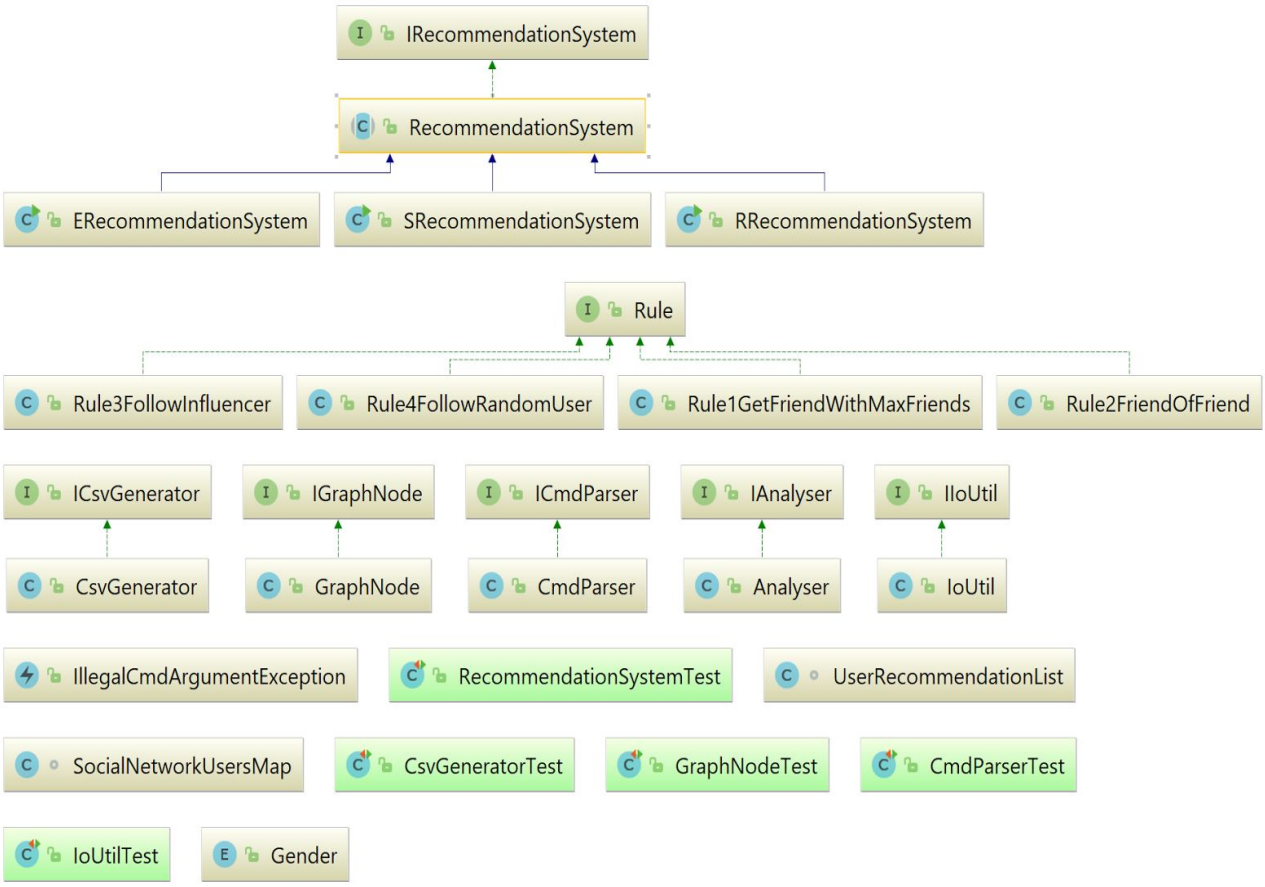
How Information is encoded

Users of the social network are stored as vertices of graph. Every user of social network has following information - a set of people he follows, number of people who follow him, number of times he was recommended etc. Edges between two nodes signify that one node follows the

other. So, for computing recommendations through rule 1, we simply check whether this user's profile is just a month old, if yes, we iterate over the friends of this user and find the user with maximum friends. And then we recommend all the friends of this user.

If there are not enough recommendations generated by Rule 1, then we try to find all the friends of user and try to recommended friends of friends to the user. And for generating recommendations by Rule 3, we maintain a priority queue of users and extract the root of priority queue to get the influencer and if that influencor is not already present in friends list, we recommend that user. And for rule 4, we simply get random users and try to recommend those users.
Complexity of our approach: The complexity of approach that we are using for our implementation is O(V+E) where V is the node/user of social network and E is the edge that signifies the relationship between two users. The complexity is linear because if we consider all the rules that we used to generate recommendations, we are branching out just for one set of edges and we are not exploring all the edges present in graph (thus, it cant be O(V*E))

UML Diagram



NOTE:
The csv files that we considered for this assignment - nodes_small.csv and nodes_10000.csv did not had any user whose profile creation date was a month old, due to which coverage for rule1 was low as that code was never hit due to all the users were created in almost 2009/2011. So, in order to cover rule1, we created a new csv file - nodes_rule1.csv and edited the profile creation dates to most recent dates. And then tried to generate recommendations for this new data in our tests.