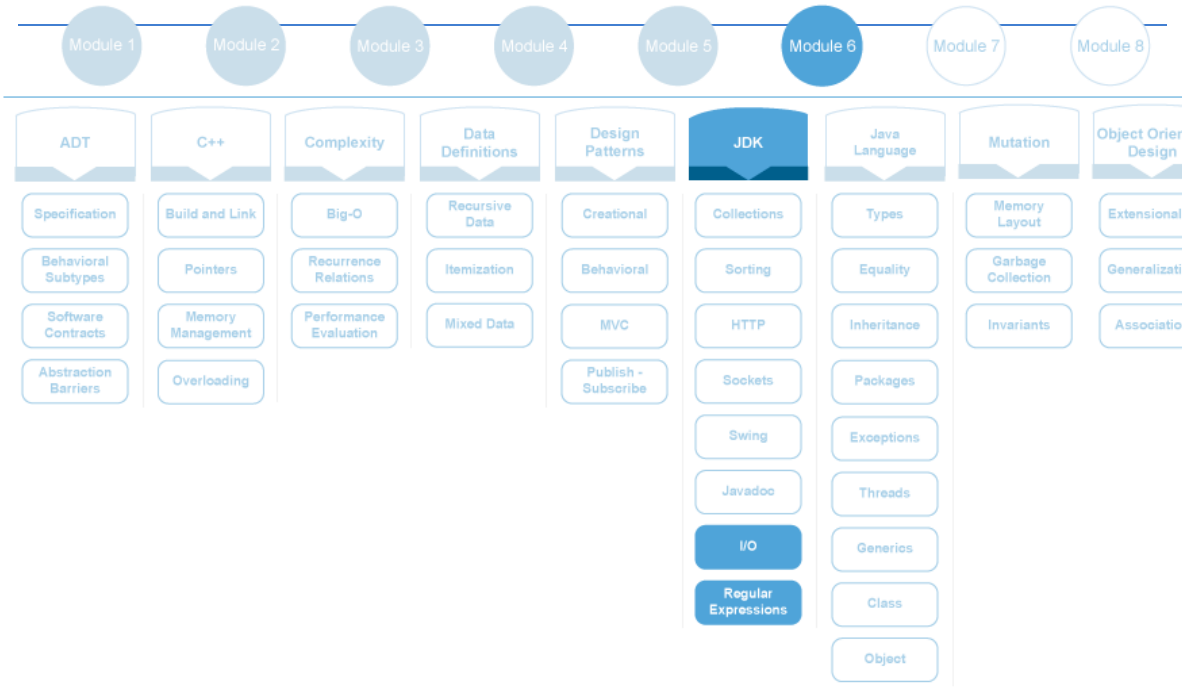
CS 5004: Object Oriented Design and Analysis

Northeastern University
College of Computer and Information Science

In this module, we start our conversation about graphs and trees in Java. We show several available formal representations, and explore some traversal and balancing algorithms.

We then start exploring how we can design programs that interact with the operating system. We introduce Java's Input and Output (I/O) standard libraries, and walk through an example Java program that consumes inputs from the command line.



1. Write a Java program that accepts input parameters on the command line.
2. Write a Java program that reads data from a file.
3. Write a Java program that writes data to a file.

Java Tutorial: Basic I/O
Java Tutorial: The Platform Environment

DUE DATE: Wednesday, March 28th at 11:59pm

Submission Criteria

Repository Contents

Your repositories must contain only the following:

1. pom.xml file and
2. src folder with appropriate sub-folders with your code and test code only.

Code Criteria

1. Your solution for each problem should be in its own package. The package names should follow this naming convention `edu.neu.ccs.cs5004.assignmentN.problemM` where you replace `N` with the assignment number and `M` with the problem number, e.g., all your code for problem 1 for this assignment must be in a package named `edu.neu.ccs.cs5004.assignment8.problem1`.
2. Your project should successfully build using maven generating all the default reports.
3. Your Javadoc generation should complete with no errors or warnings.
4. Your Checkstyle report must have no violations.
5. Your JaCoCo report must indicate 75% code coverage per package for "Branches" **and** "Instructions" or more.
6. Your FindBugs report must have no violations (except for violations categorized as PERFORMANCE or SECURITY, you can ignore those).
7. Your PMD report should have no violations.

Social Network Recommendation System

In this assignment, you will build a simple recommendation system for a Twitter-like social network. You will create a program `RecommendationSystem` that will take three required parameters as inputs:

1. `file1.csv` - a String, denoting the name of a CSV file containing information about the users (nodes) in the considered network.
2. `file2.csv`- a String, denoting the name of a CSV file containing information about the edges, where an edge {x, y} denotes that user x follows user.
3. `file3.csv` - a String, denoting the name of the CSV file used to store the output data.

The order of required arguments should be as defined above, and if one of the arguments is missing, the program should terminate with an appropriate error message. In addition to the required parameters, your program can take up to three input parameters, defined as follows:

1. `processingFlag` - a character from the set `{s, e, r}`, where `s` represents the flag to process users from the beginning, `e` a flag to process users from the end, and `r` a flag to process users in some random order.
2. `numberOfUsersToProcess` - an integer in the set `[1, 100]`, defining the number of users that need recommendations from your system. The default value for this parameter is 50. That means that, if not specified differently, your system should generate recommendations for 50 social network users.

3. `numberOfRecommendations` - an integer in the set [1, 100], defining the number of recommendations made for a single user. The default value is 15.

Based on the received input arguments, your program will then recommend a certain number of new people that an individual node may want to start following. In making such "Who-to-Follow" recommendations, your program should will rely on three ordered recommendation criteria, defined as follows:

1. **Friend of a friend is a friend** - when making recommendations for some user *A* under this criterion, your system should consider all of the friends of *A* (the nodes that *A* already follows), and it should find their friends, say some nodes in set *B*. It should then recommend as friends of *A* those nodes from *B* that are not already friends of *A*. If node *A* does not have any friends, or the system cannot make sufficient number of recommendations, your program should proceed to the next criterion. If, on the other hand, there are too many possible recommendations, then your program should release the required number of them in the ascending order, starting from the candidate node with the smallest node ID.
2. **Always follow the influencer** - when making recommendations for some user *A* under this criterion, your system should take a global view of the network, and recommend as friends those nodes that have more than 25 followers for the small data set (nodes small.csv, edges smallcsv), and more than 250 followers for the regular data set (nodes_10000.csv, edges_10000.csv). If the system cannot make sufficient number of recommendations, your program should proceed to the next criterion. On the other hand, if there are more potential candidates then the required number of recommendations, your program should again release the required number of them in the ascending order, starting from the candidate node with the smallest node ID.
3. **When in doubt, branch out** - Last criterion for your recommendation system does not really care about security, privacy and personalization. Under this criterion, your system is expected to grow a social network of some user *A* by randomly proposing some nodes from the network as potential friends, until the required number of recommendations for a user is reached.

Processing Input Files

In this assignment, we will use a modified version of the original Twitter data set, the Arizona State University Twitter Data Set, that was made available in 2009, by R. Zafarani and H. Liu, the members of the Arizona State University, School of Computing, Informatics and Decision System Engineering. The original dataset can be accessed here, and it consists of 11316811 nodes (users), and 85331846 edges (friends/followers relationships). The original data in the data set is organized in two files, and in this assignment, we will follow the same convention:

1. *nodes.csv* - representing the file of all the users. This file works as a dictionary of all the users in this data set, and in the original data set, it contains the node IDs of all the users in the data set. In our modified version, in addition to node IDs, it also contains:
    1. The date of profile creation in the format MM/DD/YY
    2. Gender, represented as one of the characters M, F or O, denoting respectively male, female and other.
    3. Age, represented as an integer in the range [0, 100]
    4. City, represented as a String
2. *edges.csv* - representing friendship/followership network among the users, where a follower/friend is represented as a directed edge

For convenience, we have provided two data sets:

1. nodes_small.csv, edges_small.csv, consisting of 100 users, and their corresponding followership network, and
2. nodes_10000.csv, edges_10000.csv, consisting of 10000 users, and their corresponding followership network.

You may want to use a smaller network during the development and debugging phase, but your program should finish in a reasonable amount of time for the larger network.

Generating Output Files

Your program should store the generated recommendations into the third provided CSV file, file3.csv (if the file does not exists, it should be created). The CSV file should contain header:

1. *Node ID* - representing the ID of a currently processed node
2. *Recommended nodes* - representing a space-separated list of node IDs to follow

For every processed social network user, the information indicated in the header should be provided as a row in the file.