**Algorithm Implementation**

A greedy algorithm, as the name suggests, **always makes the choice that seems to be the best at that moment**. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution.

Assume that you have an objective function that needs to be optimized (either maximized or minimized) at a given point. A Greedy algorithm makes greedy choices at each step to ensure that the objective function is optimized. The Greedy algorithm has only one shot to compute the optimal solution so that it never goes back and reverses the decision.

Greedy algorithms have some advantages and disadvantages:

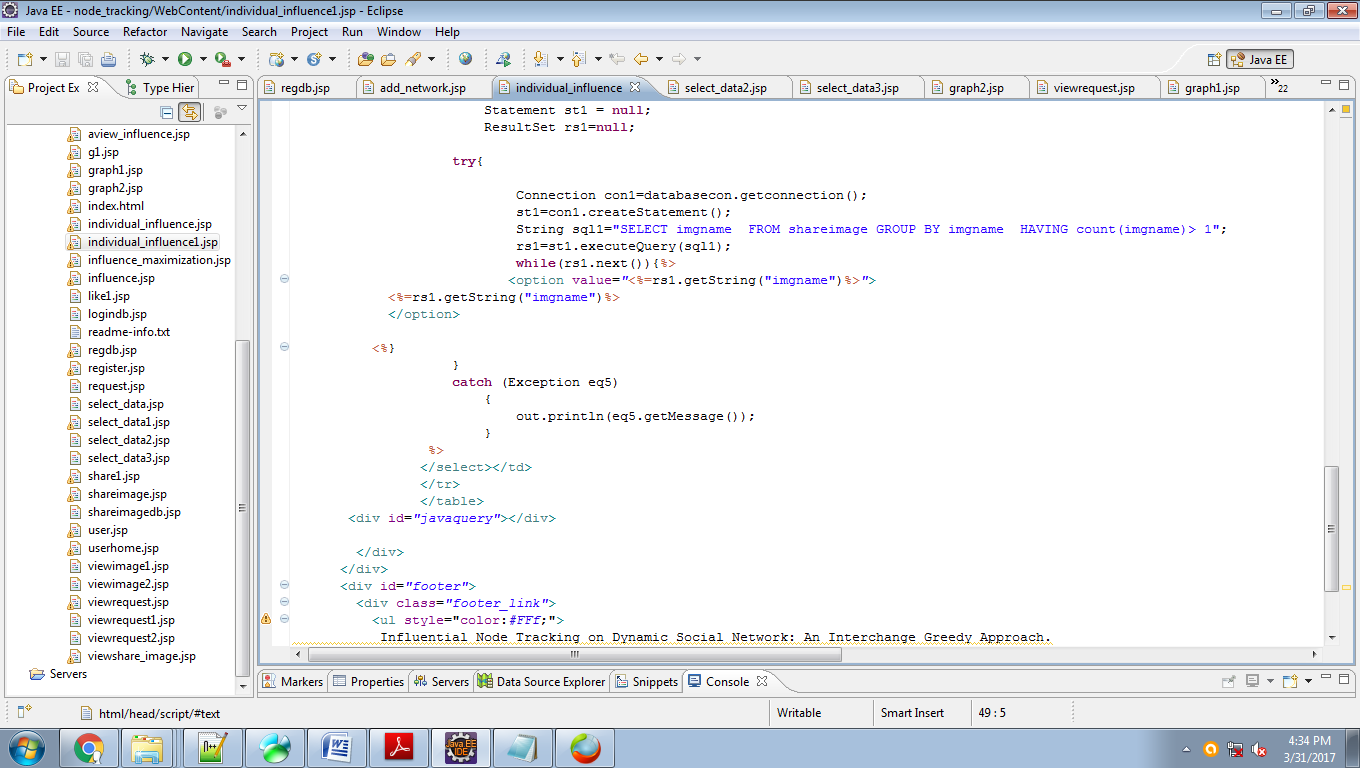
1. It is quite easy to come up with a greedy algorithm (or even multiple greedy algorithms) for a problem.
2. Analyzing the run time for greedy algorithms will generally be much easier than for other techniques (like Divide and conquer). For the Divide and conquer technique, it is not clear whether the technique is fast or slow. This is because at each level of recursion the size of gets smaller and the number of sub-problems increases.
3. The difficult part is that for greedy algorithms you have to work much harder to understand correctness issues. Even with the correct algorithm, it is hard to prove why it is correct. Proving that a greedy algorithm is correct is more of an art than a science. It involves a lot of creativity.

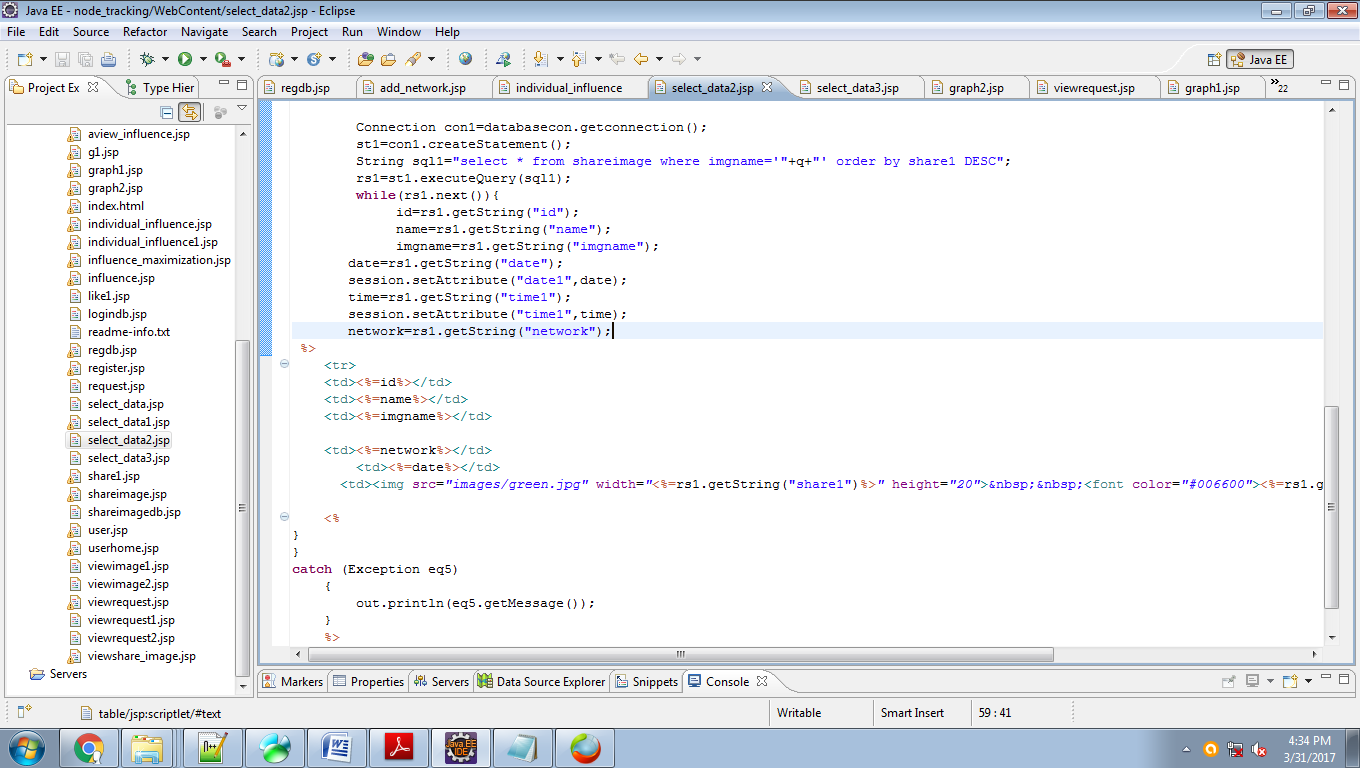
**Screen Shot**

****

****

**Source code**

****

****