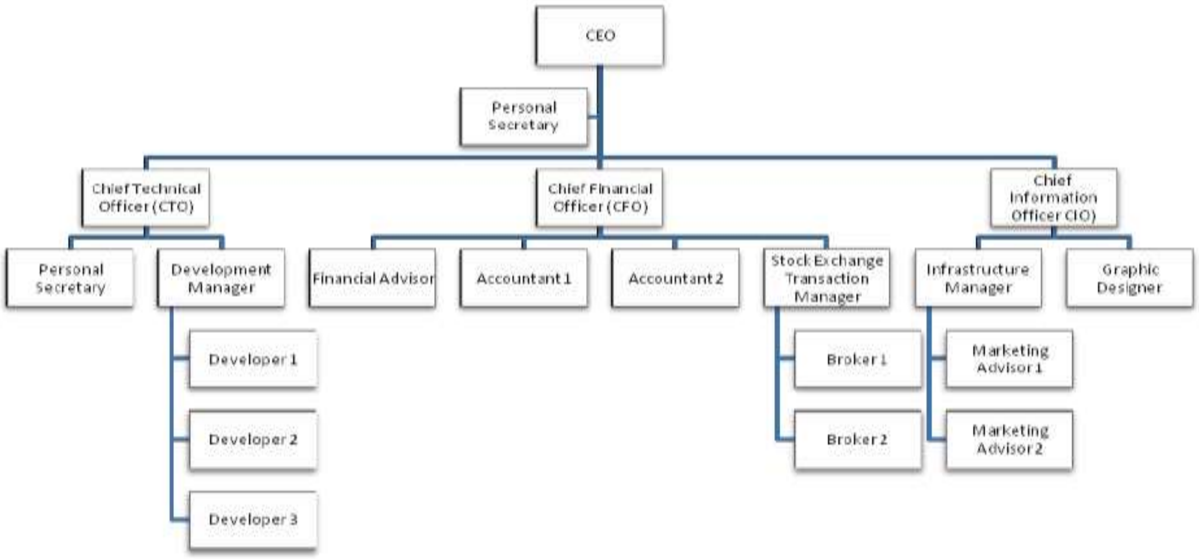
Practical Object oriented design (Composite, Visitor & Iterator)

Solve the following problems.

Problem1: Corporate Bureaucracy

A manager is an employee who leads engineers, technicians and support personnel, all of whom are also employees. A higher level manager can also lead other lower level managers. A typical hierarchy at the CyberCorp Inc will look similar to the one presented below:



a) Which design pattern you would use to implement our recursive corporate hierarchy? Provide the code that implements the above hierarchy and displays it as well.

b) Extend the solution from a) and show how the company average salary could be computed as well as the total cumulative income (the sum of all salaries) for all the employees at CyberCorp.

Problem2: Portfolio Manager

For this exercise you will implement a small portfolio application. A portfolio consists of accounts and other portfolios. An account is composed of securities classes: stocks, bonds, money market. The current value of a portfolio is the sum of the current values of the different securities it contains. You will need to write software to enable a user to build a portfolio of his choice and evaluate the total value of the portfolio. Do the following:

Practical Object oriented design (Composite, Visitor & Iterator)

a) Write the interface for all components of a portfolio(both account and other portfolios). Operations to add a component, delete a component and return its current value, are part of the interface. You will also need an accept operation to allow the visitor to work.

b) Implement classes called Account and Portfolio which must implement the interface you created in part-a. Note that this may be recursive, in that a composite may contain both leaf objects as well as other composite objects.

c) You will also need a visitor called PricingVisitor which values the different securities and sums up their current value. The PricingVisitor defines operations (e.g. visitStock, visitBond,..) to evaluate each kind of component.

d) Implement an iterator called PortfolioIterator that escorts the visitor around the composite portfolio. The PortfolioIterator defines operations like hasNext(), next().

e) Finally you need to define a class called PortfolioManager that is a Singleton class. This would contain operations to initialize (build) a portfolio, and evaluate it. To evaluate the portfolio, the PortfolioManager would use the iterator to navigate the portfolio and the visitor to value each component. The structure and creation of the composite structure in memory can be hardcoded in the PortfolioManager. No client GUI is necessary.

f) Test your software by building a couple of portfolios and evaluating them.