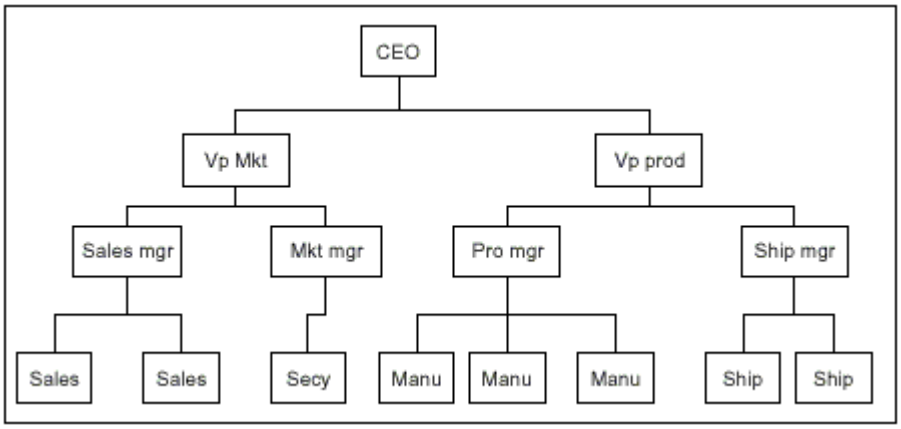
Exercise 3

01.11.2001 **Due date: 14.11.2002**

Assignment 3.1

Let?s consider a small company. It may have started with a single person who got the business going. He was, of course, the CEO, although he may have been too busy to think about it at first. Then he hired a couple of people to handle the marketing and manufacturing. Soon each of them hired some additional assistants to help with advertising, shipping and so forth, and they became the company?s first two vice-presidents. As the company?s success continued, the firm continued to grow until it has the organizational chart we see below:



Now, if the company is successful, each of these company members receives a salary, and we could at any time ask for the cost of any employee to the company. We define the cost as the salary of that person and those of all his subordinates.

Implement the organizatinal chart using the composite design pattern. Your application should allow
1. drawing the organizational chart
2. asking for the salary of a company's member.

It is sufficient to draw the organizational chart in a text-based form, such as:

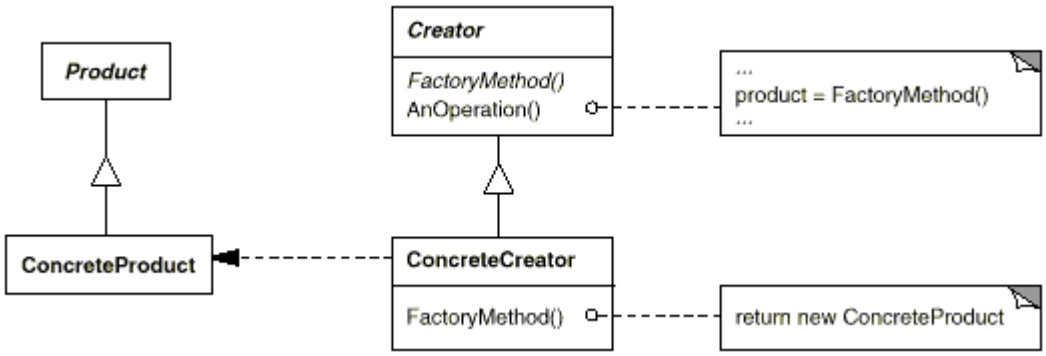
-CEO
-- Vp Mkt
--- Sales mgr
---- Sales 1
---- Sales 2 .....

Assignment 3.2

The factory method design pattern is used when one wants to construct an instance of a product, but

let subclasses decide which concrete product to instantiate. Factory method lets a class (the creator) defer instantiation to subclasses (concrete creator). In order to avoid subclassing the class containing the factory method (the creator), we need a mechanism for creating instances of related products without necessarily knowing which will be instantiated. This can be implemented by defining an interface which has all the common methods and properties of the product and is then implemented by the concrete product. The client calls a method of the creator with the name of the concrete product it wants to instantiate and the creator instantiates the class representing the concrete product and returns back the interface to the client which then works on that interface. This is done using reflection mechanism. The biggest advantage of using reflection in the factory method design pattern is that any number of implementing concrete classes can be plugged-in without breaking any existing code.

The figure below shows the original factory method design pattern from the book "Gamma et. al, *Design Patterns: Elements of Reusable Object-Oriented Software*"without using reflection.



Implement the factory method design pattern using reflection mechanism in the creator class (instantiation of the concrete product using reflection) instead of subclassing the creator class with a concrete creator that instantiates the concrete product.
