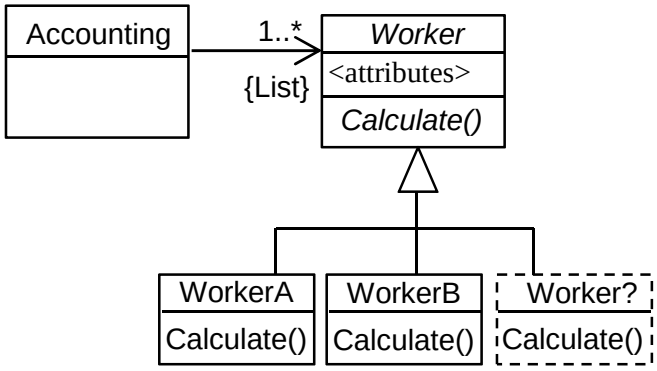
İTÜ
Computer Engineering Department
Assoc.Prof. Feza BUZLUCA

Object Oriented Modeling and Design Exemplary Final Questions and Solutions

QUESTION 1:



In a company, salaries of some workers are calculated using the method A, and salaries of other workers are calculated using the method B. These methods depend on some attributes of the works such as age, number of children etc.
The accounting system maintains a list of workers. Each month salaries of workers in this list are calculated.
In the future, these methods can change, a new method can be added to the system, and the method of an existing worker can be replaced by another method.
The software architect decides to use inheritance and constucts the given design.

a. What are the possible drawbacks of the given design?
b. To overcome these problems construct a better design and draw the UML class diagram. Show parameters of the methods. Mention the design principles and patterns used in your solution.
c. Explain what will hapen in your system if the calculation method of an existing worker is replaced by another existing calculation method?

QUESTION 2:

In a software system there is a counter object (:Counter) that counts some events. There are also other objects which perform some actions if the counter value exceeds two predetermined threshold values. For example, objects :A and :B perform their actions if the counter exceeds the threshold1 value; objects :C and :D perform their operations if the counter exceeds the threshold2 value. In run-time, the number of objects that are interested in the counter value can change. While objects (:A , :B , :C , :D) are created they get the address of the :Counter object in which they are interested. Classes (A,B,C,D) include two methods, namely startCount() and stopCount() to start and terminate their interest in the counter value, respectively.
In the future new classes (E, F) can be added to system that may be interested in the counter value.

a) Design the system to achieve the required flexibility and draw the UML class diagram. Which GoF design patterns did you use in your design?
b) Write the known parts of the program of class A in C++ or Java. Include necessary attributes, constructor, and methods startCount(), stopCount(), Operation() that is invoked if the counter value exceeds the threshold.
c) Assume that the class Conter includes a method eventHappened() that increments the counter value. Draw a UML sequential interaction diagram, that show interactions, which occur in the system if when the eventHappened() method is called.

QUESTION 3 :

To do: An object of class A must be created and an object of class B will use it.
According to GRASP Creator, "Assign class B the responsibility to create an instance of class A if B closely uses A".
In what cases this advice does not lead to proper solutions?
Create a better design to overcome this problem and express your solution by drawing a UML collaboration diagram.

SOLUTION 1:

a.

Actually, there are not different types of workers in the system; only the method to calculate the salary changes.

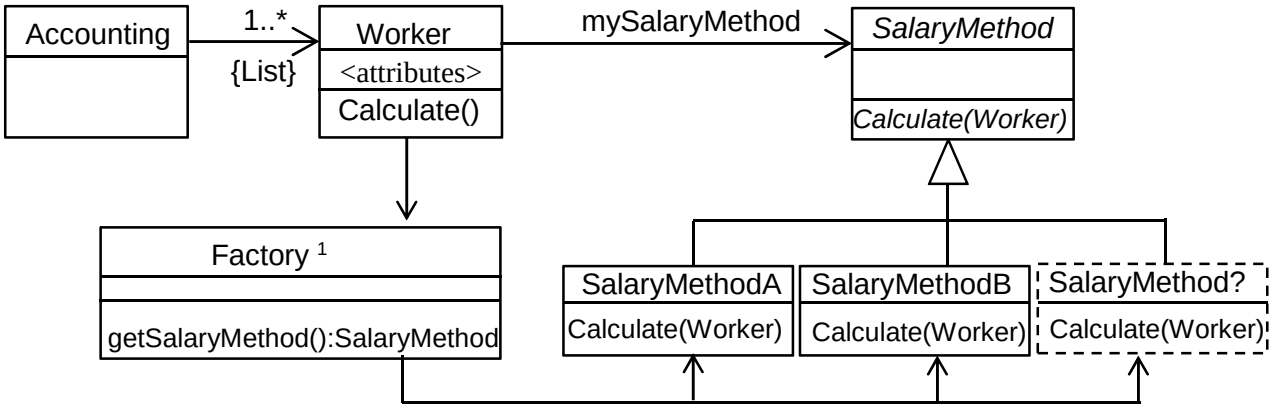
Varying parts (calculation methods) are inserted into stable class (worker).

Drawbacks:

- If we want to inset a new calculation method a new type (class) of worker must be written.
- Calculation method of a worker cannot be changed in run time. In such a case, we must destroy (delete) the existing worker object and then create a new one.

b.

To have a proper solution we will apply the Strategy pattern, which supports the "Favor composition over inheritance" principle.



Principles:

"Favor composition over inheritance" principle.

"Design to interface"

Patterns:

Strategy, Factory (singleton)

c. If we want to replace the calculation method of an existing worker by another existing calculation method, we only need to change the address (reference) of the SlaryMethod pointed by the pointer mySalaryMethod of the Worker object. We can change it in run time, without interrupting the system.

Details:

Worker object asks the Factory for a SalaryMethod pointed by calling the getSalaryMethod(). The Factory will decide the new method and return its address.

SOLUTION 2:

Observer pattern will be used.

Counter is publisher (:A , :B , :C , :D) are subscribers (listeners or observers).

One solution is to create two listeners-lists, namely one for threshold1 and the other one for threshold2.

Ore only one list can be created and the notified objects can check the reason for the notification (threshold1 or threshold2).

SOLUTION 3:

This advice can violate the "low coupling" principle. If B creates objects A then B is coupled to A. Class B must also know the details how to create objects of A (when, which parameters?).

It may also lower the cohesion of B. Class B may have different responsibilities from object creation.

Class A can be a subclass of a general type and according to some conditions objects of different subclasses can be creates such as A1, A2. Class B must include the decision logic. If subclasses or the decision rule change class B must also change.

It is better to apply the "Factory" pattern.