Problem 1. Provide definitions for the following terms.

- abstraction
- encapsulation
- cohesion
- coupling
- Also, how does each of these terms apply to the object- oriented notion of a class? Provide examples of both good and bad uses of these terms in the design of a class or a set of classes.

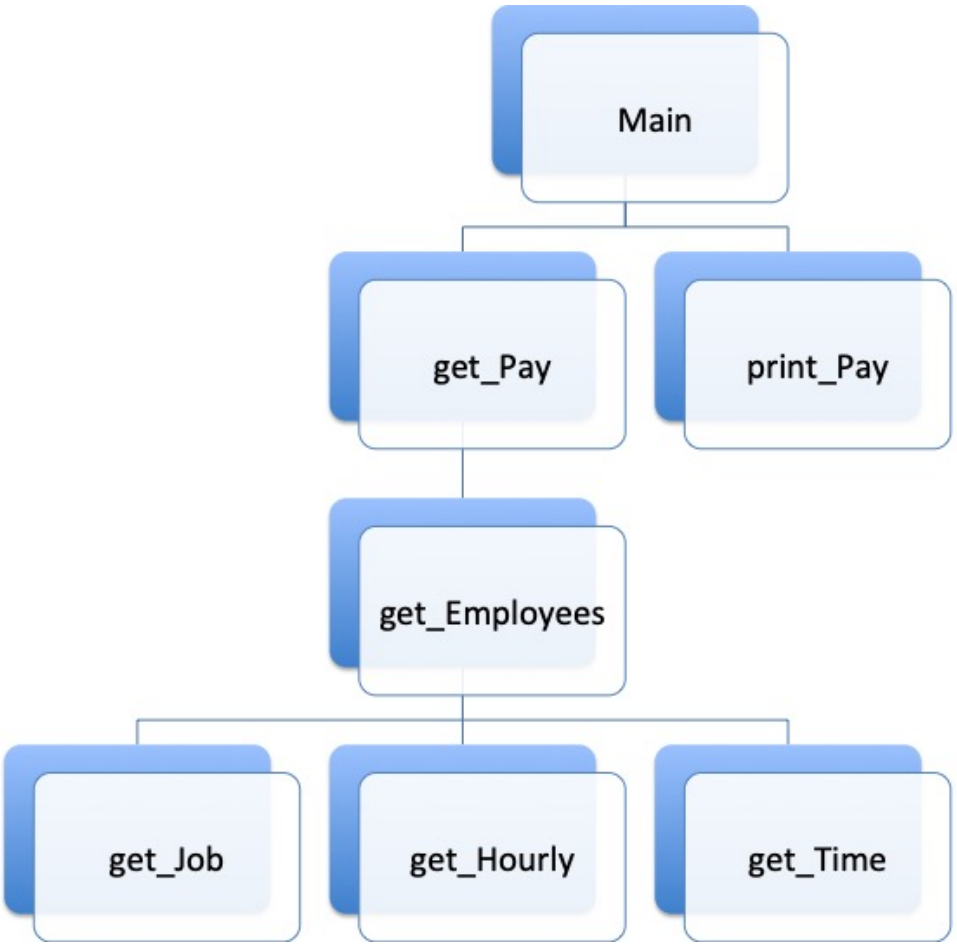
Solution:

- Abstraction
  - Abstraction refers to a set of concepts that are provided to you in order for you find a solution or finish a task. In other words, it is the process of showing only necessary information of an object to the user and hiding the unnecessary data. A good use of this is if you were designing a calculator. A calculator must be powered by battery and it shows you the battery life. However, it does not give information about specific implementation details of how the battery works. A bad use would be showing this information to the user because they have no idea what it means and doesn't help them at all.
- Encapsulation
  - Encapsulation refers to a set of language-level mechanisms or design techniques that hide implementation details of a class, module, or subsystem from other classes, modules, and subsystems. An example of a good use of this is the calculator. The calculator has buttons that have formulas that a pre-programmed and does not give you the chance to change these. This is a good use because it hides certain parts of the system while still exposing necessary information. A bad use of this is if one could change how the buttons worked on the calculator because if they did not know what they were doing it could affect the device.
- Cohesion
  - Cohesion refers to how closely the operations in a routine are related. Cohesion applies to the object-oriented notion of class because classes can have weak and strong cohesion. A good use of this term would be creating classes that do a well defined job. For example if you were writing a program to calculate grades for each class. A good use of cohesion would be having one class calculating grades and another class to visually show the results. This would be a good use because each class has a well defined job. A bad use of cohesion would be to do the opposite and combine all of these classes because doing mathematical operations and visually showing the results are not closely related. Writing 2 different classes for these will allow you to develop them separately which is a very good practice.

- Coupling
  - Coupling refers to the strength of a connection between two routines. Coupling applies to the object-oriented notion of class because in a program classes are connected to one another and can affect each other if one is changed. Classes are going to be dependent on each other to some degree but , we, the coders get to decide how dependent they are. Using the example before with two classes, one calculating the grades and another visually displaying them, results in loose coupling which good use of coupling because these classes aren't highly dependent on each other. On the other hand if there was only one class written then one small change could send ripple effects and affect the entire class whereas a small change in the grade calculator class won't have large changes, if any at all, on the visual class.

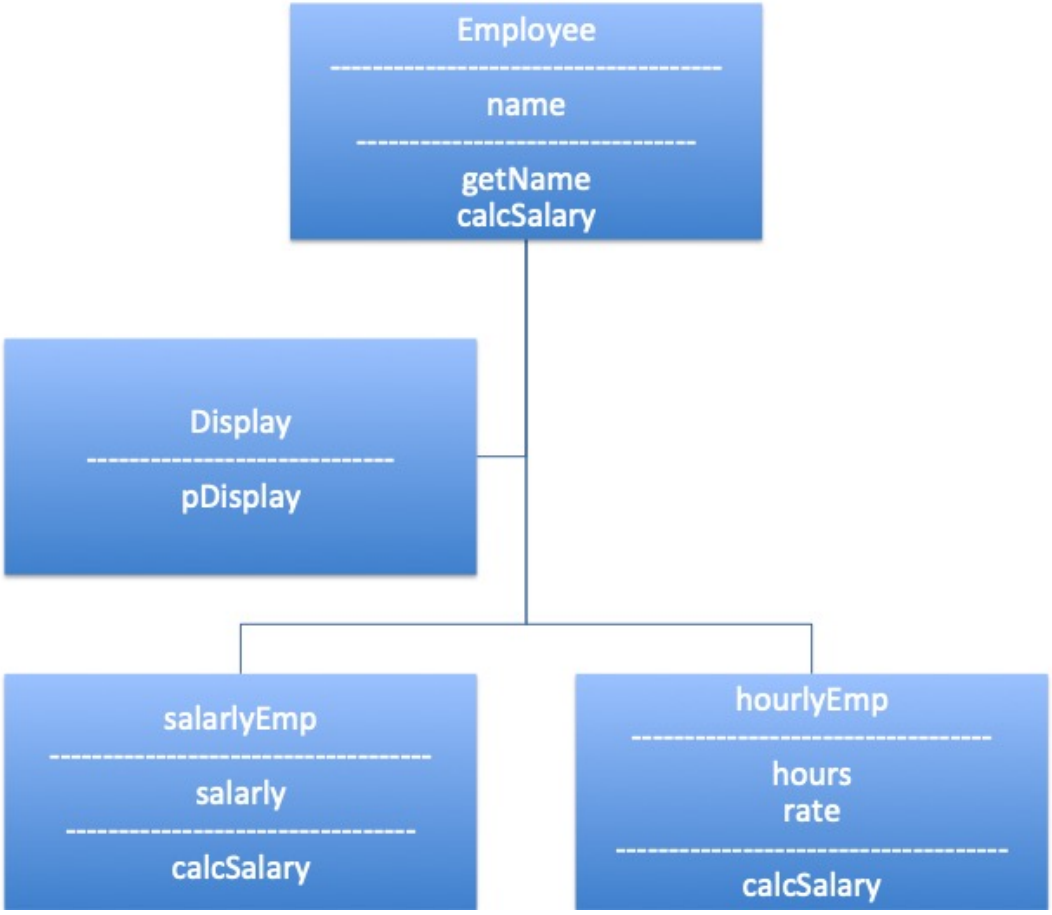
Problem 2. A company has asked us to design a payroll system that will pay employees for the work they perform each month. Using a level of abstraction, similar to that shown in Chapter 1 of the textbook and as shown on slide 6 of lecture 2, develop a design for this system using the functional decomposition approach. You can assume the existence of a database that contains all of the information you need on your employees. For your answer, first describe the functional decomposition approach, discuss what assumptions you are making concerning this problem, and then present your design.

Solution: In my functional decomposition approach I have broken down the payroll system into small steps that will retrieve the information from the database that is needed. The first step getPay will calculate all of the information from getEmployees. The next step getEmployees will get 3 separate steps, that will retrieve all of the information for each employee from the database in order to calculate their pay for the month. The final step is returning the of getPay.



Problem 3. Now develop a design for the payroll system using the object-oriented approach, keeping in mind the points discussed on slides 21-23 of lecture 2 as well as the discussion in Chapter 1 of the textbook. Identify the classes you would include in your design and their responsibilities. (As before, you can assume the existence of a database and that you'll be able to create objects based on the information stored in that database.) Then, identify what objects you would instantiate and in what order and how they would work together to fulfill the responsibilities associated with the payroll system.

Solution: The classes that I have included in my design are Employee class, Display Class, salary Employee and hourly employee. The object that would instantiate are salary employees and hour employees. For each employee their information would be pulled from the database and then used to calculate their salary. The display class would then display the employees name and what they made in the past month.


Employee
name
getName
calcSalary
Display
pDisplay
salarlyEmp
salarly
calcSalary
hourlyEmp
hours
rate
calcSalary