1. Complete this on your own, then review the videos and code in the class website: Employee Class Programming Challenge #1 from Chapter 6.

Write a class named `Employee` that has the following fields:

- `name`. The `name` field references a `String` object that holds the employee's name.
- `idNumber`. The `idNumber` is an `int` variable that holds the employee's ID number.
- `department`. The `department` field references a `String` object that holds the name of the department where the employee works.
- `position`. The `position` field references a `String` object that holds the employee's job title.

The class should have the following constructors:

- (1) A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name, employee's ID number, department, and position.
- (2) A constructor that accepts the following values as arguments and assigns them to the appropriate fields: employee's name and ID number. The `department` and `position` fields should be assigned an empty string ("").
- (3) A no-arg constructor that assigns empty strings ("") to the `name`, `department`, and `position` fields, and 0 to the `idNumber` field.

Write appropriate mutator methods that store values in these fields and accessor methods that return the values in these fields.

Draw a UML class diagram for the class.

Once you have written the class, write a separate program that creates three `Employee` objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.

Draw UML object diagrams for the three objects and draw a sequence/interaction diagram showing the program executing.

2. Answer the following questions about the program and diagrams:

a) If you had only the class diagram and no code to work from, give the public and private members of the class and explain how you know which is which.

Public members	Private members

_____

_____

_____

b) Describe the scope of the parameter to `setIDNumber()` and the scope of `idNumber`. List the methods in which each is visible.

_____

_____

_____

_____

setIDNumber parameter	idNumber

c) Are there any overloaded methods? Which? When reading the code, how do you know they're overloaded? How does the compiler know which one to bind to a method call?

_____

_____

_____

_____

_____

_____

_____

_____

d) List the methods that have results.

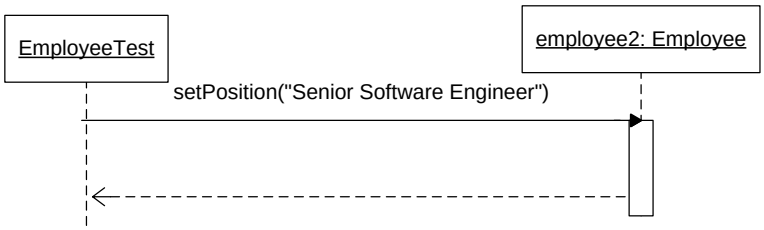
_____

_____

_____

_____

e) Show how the object diagram for the Mark Jones `Employee` object (called `employee2` here) looks after the following takes place:



3. On your own, enhance the `Employee` class as follows, and then post your solution to the discussion board.

Add the following fields:

- `payRate`. `payRate` is the employee's hourly pay rate. This is a `double` type.
- `hoursPerWeek`. `hoursPerWeek` is the employee's allowed hours per week. This is also a `double`.

Enhance all three constructors. The no-arg constructor (3) should initialize `payRate` and `hoursPerWeek` to 0.0. The constructor that takes two arguments (2) should also intialize `payRate` and `hoursPerWeek` to 0.0. The final constructor (1) should be modified to accept parameters for all of the object's fields.

Add accessors and mutators for the new fields. In addition, add a method called `grossPay()`. `grossPay()` should multiply `payRate` by `hoursPerWeek` and return the result.

Modify the program that tests the `Employee` class to include the following information for each employee (in addition to the information previously included):

Name	**Pay rate**	**Hours per week**
Susan Meyers	85.00	40.0
Mark Jones	50.00	60.0
Joy Rogers	45.00	37.5

The program, when printing employee information, should print the name, id number, department, position, and gross pay of each employee. (*Not* pay rate or hours per week.)

You should draw the following diagrams:

- Class diagram for `Employee`
- Object diagrams for the three `Employee` objects
- A sequence diagram showing the program executing