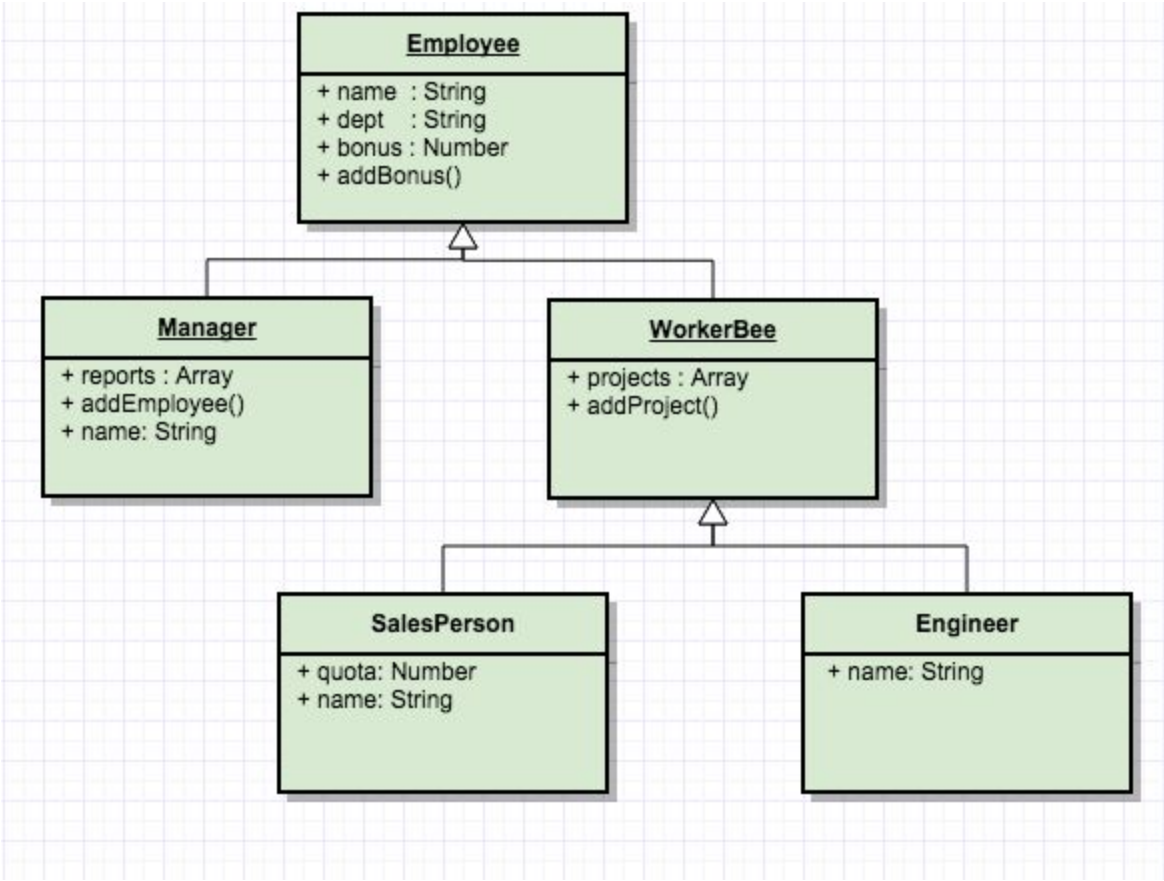
Mini Lab 2

In this lab you will create the following object inheritance structure using the two different methods of inheritance discussed in the slides (Object.create() and Constructor functions). In each case, you will need to create a superclass called *Employee* and then two subclasses, *Manager* and *WorkerBee* which will extend from *Employee* using inheritance. Create two additional subclasses of WorkerBee called *SalesPerson* and *Engineer*. Implement each method and property shown in the diagram. The point of this lab is to make sure you understand how prototypal inheritance works in JavaScript.



Part 1: Object.create() inheritance

In this first part, use the Object.create() method to achieve your hierarchical structure. Make sure you are using the prototype chain to access these properties.

1) Create an Employee object with an *addBonus(bonus)* method that, after validating that the parameter is a Number, sets the **bonus** member.

2) Create two objects that extend from Employee using Object.create(), *Manager* and *WorkerBee*. The *Manager* object should be created with a **reports** property that is an array, and a method called **addEmployee(employee)** which accepts an employee object. Use *Object.isPrototypeOf(object)* to make sure the passed in object is an instance of *Employee*. If it passes then add it to the **reports** array. If not, log a console message. The *WorkerBee* object needs a **projects** array, and an *addProject(project)* method that adds a project string to the **projects** array.
3) Create two more objects that extend from *WorkerBee*, 1) *SalesPerson* and 2) *Engineer*. *SalesPerson* should be initialized with a **dept** property set to "sales", and a quota number value and a name.The *Engineer* object should be initialized with a **dept** property set to "engineer" and a name.
4) When you are finished with your structure, create two managers, and 4 employees (2 Sales & 2 Engineers). Add your sales instances to one manager and the engineers to the other. Add some projects to the WorkBee employees, and give the managers a bonus.

Part 2: Constuctor Functions

In the second part, use constructor functions with the "new" keyword the the function's prototype member to achieve inheritance. Mimic the same functionality as in the first part but do not use Object.create().

Part 3 (optional): Extend method

Create an ***extends*** method that accepts a destination object as the first parameter, and then *n* number of source objects. For each source object, copy each member to the source object if and only if the the source object does not already have that member. Remember to use the **hasOwnProperty()** method when looping through the object properties so you don't dredge through the inheritance chain.