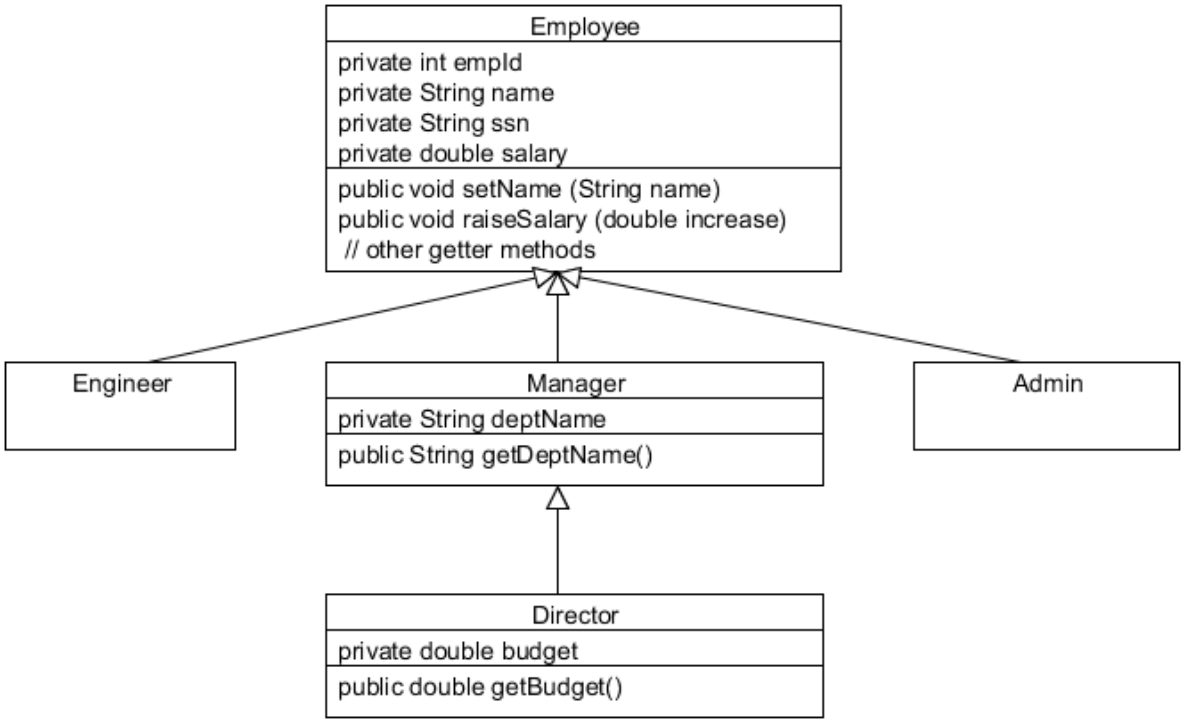
Overview

In this practice, you will create subclasses of Employee, including Manager, Engineer, and Administrative assistant (Admin). You will create a subclass of Manager called Director, and create a test class with a `main` method to test your new classes.

Assumptions

Use this Java class diagram to help guide this practice.



Apply encapsulation to the `Employee` class.

a. Make the fields of the `Employee` class private.
b. Replace the no-arg constructor in Employee with a constructor that takes `empId`, `name`, `ssn`, and `salary`.
c. Remove all the setter methods except `setName`.
d. Add a method named `raiseSalary` with a parameter of type `double` called `increase` to increment the salary.
e. Add a method named `printEmployee` to print the `Employee` object details.
f. Save `Employee.java`.

3. Create a subclass of `Employee` called `Manager` in the same package.
   a. Add a private String field to store the department name in a field called `deptName`.
   b. Create a constructor that includes all the parameters needed for Employee and `deptName`.
   c. Add a getter method for `deptName`.

4. Create subclasses of `Employee`: `Engineer` and `Admin` in the `com.example.domain` package. These do not need fields or methods at this time.
5. Create a subclass of `Manager` called `Director` in the `com.example.domain` package.
   a. Add a private field to store a double value `budget`.
   b. Create a constructor for Director that includes the parameters needed for Manager and the `budget` parameter.
   c. Create a getter method for this field.
6. Save all the classes.
7. Test your subclasses by modifying the `EmployeeTest` class. Have your code do the following:
   a. Remove the code that creates an instance of the “Jane Smith” Employee.
   b. Create an instance of an `Engineer` with the following information:

Field	Choices or Values
ID	101
Name	Jane Smith
SSN	012-34-5678
Salary	120_345.27

   c. Create an instance of a `Manager` with the following information:

Field	Choices or Values
ID	207
Name	Barbara Johnson
SSN	054-12-2367
Salary	109_501.36
Department	US Marketing

   Create an instance of an `Admin` with the following information:

Field	Choices or Values
ID	304
Name	Bill Munroe
SSN	108-23-6509
Salary	75_002.34

   e. Create an instance of a `Director`:

Field	Choices or Values
ID	12
Name	Susan Wheeler
SSN	099-45-2340
Salary	120_567.36

Department	Global Marketing
Budget	1_000_000.00

f. Use the `printEmployee` method to print out information about each of your Employee objects.

g. (Optional) Use the `raiseSalary` and `setName` methods on some of your objects to make sure that those methods work.

h. Save the `EmployeeTest` class and test your work.

8. (Optional) Improve the look of the salary print output using the `NumberFormat` class.

   a. In the `printEmployee()` method of `Employee.java`, use the following code to get an instance of a static `java.text.NumberFormat` class that you can use to format the salary to look like a standard US dollar currency:

```
NumberFormat.getCurrencyInstance().format((double)
getSalary()));
```

9. (Optional) Add additional business logic (data validation) to your `Employee` class.

   a. Prevent a negative value for the `raiseSalary` method.

   b. Prevent a null or empty value for the `setName` method.