BACS2023 Object-Oriented Programming

Practical 6: Inheritance and Polymorphism

Q1. (a) A bank offers its customers the following account types:

- The *savings account* earns interest that compounds monthly i.e. the interest is calculated based on the balance on the last day of the month.
- The *current account* has no interest, but the customer is given a small number of free transactions per month and is charged a nominal transaction fee for each additional transaction.

Create a superclass **`Account`** that has the properties *account number*, *balance*, and *date created*, and methods for *deposit* and *withdrawal*. Create two subclasses for savings and current accounts. The class for savings account should have methods to calculate the interest amount and to add the interest to the balance.

(b) Write a test program that creates objects of **`SavingsAccount`** and **`CurrentAccount`**. Test the deposit and withdraw methods in each class to ensure that they work correctly. For the **`SavingsAccount`** class, include statements to test the methods for calculating the interest amount and adding the interest to the balance.

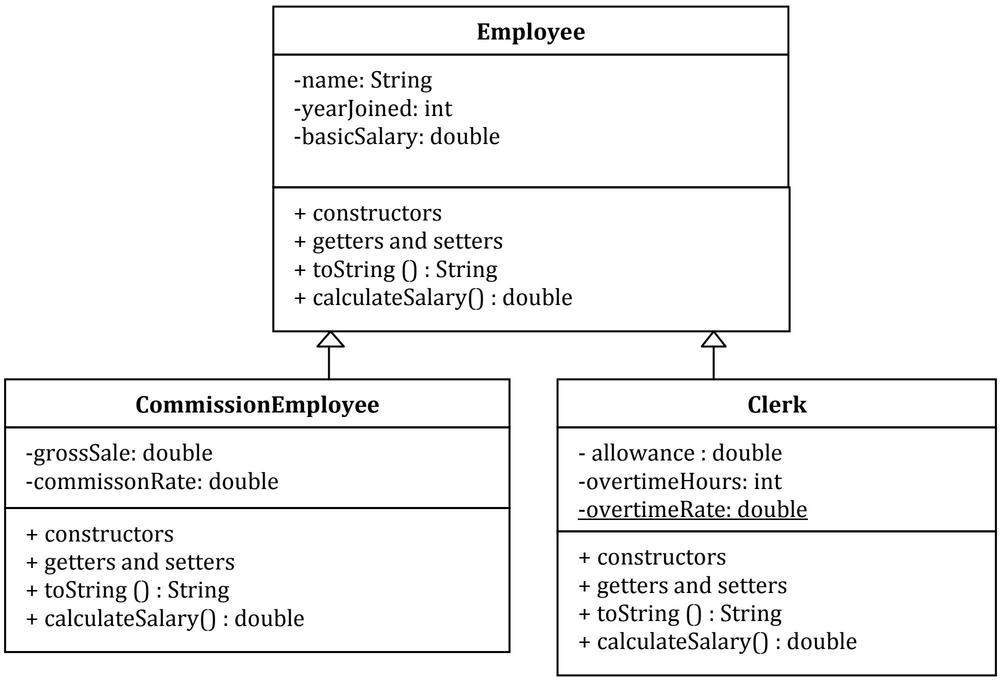
(c) Write a client program that has a simple menu for handling transactions for a current account. Your program should ensure that the transaction fee is deducted from the account balance if the number of free transactions has been exceeded for the month.

Q2. (a) Override the **`withdraw()`** and **`deposit()`** methods in the **`CurrentAccount`** class such that the transaction fee is automatically deducted for each transaction once the number of free transactions has been exceeded. Next, simplify your client program accordingly.

(b) Write the **`toString()`** method for the **`Account`** class such that it returns the object's data field values. Next, override the **`toString()`** method in the **`CurrentAccount`** class so that the object's transaction count is also returned as part of the string. Modify your test program from Q1(b) to test the **`toString()`** methods.

(c) Override the **`equals()`** method in the **`Account`** class such that it returns **`true`** if the current object has the same account number as the parameter. Test to ensure that your method works correctly.

Q3. Consider the diagram given below:



(a) Implement all the classes. The formula for calculating salary for each class is shown below:

- For the **`CommissionEmployee`** class ,
  salary = basic salary + gross sale * commission rate
- For the **`Clerk`** class , salary = basic salary + allowance + overtime pay,
  where overtime pay = overtime hours x overtime rate

(b) Write a client program that creates an array named **`empArray`** that stores an object of an **`Employee,`** a **`CommissionEmployee`** and a **`Clerk`**.

In your program, include a method called **`printElements()`** that takes an array as parameter and prints the type of employee, the object's data field values (by invoking the **`toString()`** method) and the monthly salary.

(c) Override the **`Object`** class's **`equals`** method in the **`Employee,`** **`CommissionEmployee`** and **`Clerk`** classes. For each class, assume that two objects are considered equal if they have the *same name*. Test the **`equals`** method on all derived types of **`Employee`**.