Object-Oriented Programming

Lab Exercises - Inheritance

a.

Write a program PeopleTest with main() function as following:

```java
Employee newbie = new Employee
    ("Newbie", new Date("2/10/1989"), 1000000);
Manager boss = new Manager
    ("Boss", new Date("2/23/1986"), 4000000);
boss.setAssistant(newbie);
Manager biggerBoss = new Manager
    ("Big Boss", new Date("3/12/1969"), 1000000);
biggerBoss.setAssistant(boss);
System.out.println(newbie);
System.out.println(boss);
System.out.println(biggerBoss);
```

You need import java.util.Date in order to use the data type Date.

Implement classes Person, Employee, Manager as designed in the diagram so that you can run PeopleTest.
You can add more methods if it’s necessary.
Note that biggerBoss.setAssistant(boss) assign boss to be the assistant of biggerBoss.
- means private, + means public.
Make sure subclasses’s constructors call super class’s constructor to initialize inherited instance variables.
Subclasses’ toString() should call superclass’s version to avoid duplicated code.

Override toString() at the classes to suit their data.
You can check out an example of using class Date at:
https://bitbucket.org/chauttm/examples/src/73a7f25b097498048b8f557df37907831e8f3cd8/TestDate.java?at=master

b.

In function main() of PeopleTest, create an array to hold three objects newbie, boss, and bigBoss created above, then use a loop to print information about each person to the screen.

c.

Create a package named people. Move Person and Employee to that package, leaving Manager and PeopleTest in the default package. Fix access modifiers so that the program can run again.
Bank Account

Implement classes Account, FlatFee, NickelNDime, Gambler that model four types of bank accounts described as follows:

Data about each account includes current balance, number of transactions during the current month. Each account object responses to the following message:

• a constructor that create a new account given an initial balance as a parameter;

• boolean deposit(int) put some money to the account. It updates transaction count then returns true if successful

• boolean withdraw(int) get some money from the account. It updates transaction count then returns true if successful

• void endMonth() is called by client programs at the end of each month to charge monthly fee (amount calculated by endMonthCharge()), print account data including balance, transaction count, and fee, then reset transaction count to 0 ready for the next month.

• endMonthCharge() return the amount to be charged as fee for the last month. Fees depend on the type of account. FlatFee account has a fix fee of 10000 per month. NickelNDime accounts are charged 2000 for each successful withdraw. Fees for FlatFee and NickelNDime accounts are collected at the end of each month.

Gamble accounts don’t have monthly fee. But each withdraw is done in a gambling way. For a probability of 49%, a withdraw results in zero deduction from the account balance, the owner gets the withdrawn amount for free. For a probability of 51%, the amount deducted from the balance is twice the amount the account owner requests and receives.

Make sure you have a superclass BankAccount, and try to avoid code duplication as possible.