The assignment is to write classes for this hierarchy shown:

SavingsAccount

CheckingAccount

BankAccount

1. Write the parent **BankAccount** class with the following:
   1. A private, double typed instance variable called **balance.** This will hold the account’s balance
   2. A constructor that receives an initial account balance as a parameter and sets the account’s balance using it.
   3. A method called **deposit** that receives a double parameter and adds that to the account’s balance. It should not return anything.
   4. A method called **withdraw** that receives a double parameter and reduces the account’s balance by that amount. It should not return anything.
   5. A method called **getBalance** that returns the account’s balance. It should not receive any parameters.
   6. An *abstract* method called **endOfMonth;** no parameters, no return type. This will require you to make the class abstract.
2. A **CheckingAccount** is a class that will have all the stuff that a **BankAccount** has, with the additional feature of having a limit on the number of free transactions allowed. A fee will be charged if there are more than 3 transactions issued on the account. Write **CheckingAccount** by inheriting **BanckAccount** and providing these additional things:
   1. A private, int typed instance variable called **transactionCount** that will be used to keep track of how many transactions have been issued on the account.
   2. A constructor that receives an initial account balance as a parameter. There is no variable in this class for it; you’ll need to remember what to do in the child constructor when the instance variables are in the parent. The constructor should also initialize the **transactionCount**  to 0.
   3. *Override* the parent’s **deposit** method. It should do what the parent does for a deposit **and** it should add one to the transaction count.
   4. *Override* the parent’s **withdraw** method. It should do what the parent does for a withdraw **and** it should add one to the transaction count.
   5. A method called **deductFees;** no parameters, no return type. They get 3 free transactions. If there are more than 3, deduct $2.50 from the account’s balance for each transaction over 3 (the first three transactions are free, they cost $2.50 per transaction after that). As it finishes, it should set the transaction count back to zero.
   6. *Override* the parent’s abstract **endOfMonth** method. Have it call **deductFees.**

**(over)**

1. A **SavingsAccount** is a class that will have all the stuff that a **BankAccount** has, with the additional feature of earning interest. Write **SavingsAccount** by inheriting **BanckAccount** and providing these additional things:
   1. A private, double typed instance variable called **interestRate.**
   2. A constructor that receives an initial account balance and an interest rate as parameters. Handle these two variables appropriately
   3. A method called **addInterest;** no parameters, no return type. Calculate interest as the account balance \* interestRate / 100. Add that amount to the account balance.
   4. *Override* the parent’s abstract **endOfMonth** method. Have it call **addInterest.**
2. Write a client that constructs the following two accounts:

SavingsAccount momsSavings = new SavingsAccount(10000.90, 0.5)

CheckingAccount harrysChecking = new CheckingAccount(100)

Then, perform the following operations:

* 1. Transfer $2000 from mom’s account to Harry’s by withdrawing from mom’s and depositing into Harry’s
  2. Withdraw $1500 from Harry’s account
  3. Withdraw $80.50 from Harry’s account
  4. Transfer $1000 from mom’s account to Harry’s by withdrawing from mom’s and depositing into Harry’s
  5. Simulate the end of the month by calling endOfMonth() on the accounts.
  6. Print the balances for each account using proper prompts (i.e. Mom’s account has $8500, etc).