CSULB
CECS277

Hmw

We have implemented a simple BankAccount Class that simulates a bank account.

Customers can deposit and withdraw funds. If sufficient funds are not available for withdrawal, a $10 overdraft penalty is charged. At the end of the month, interest is added to the account. The interest rate can vary every month.

```
**
   A bank account has a balance that can be changed by
   deposits and withdrawals.
*/
public class BankAccount
{
   private double balance;

   /**
      Constructs a bank account with a zero balance.
   */
   public BankAccount()
   {
      balance = 0;
   }

   /**
      Constructs a bank account with a given balance.
      @param initialBalance the initial balance
   */
   public BankAccount(double initialBalance)
   {
      balance = initialBalance;
   }

   /**
      Deposits money into the bank account.
      @param amount the amount to deposit
   */
   public void deposit(double amount)
   {
      double newBalance = balance + amount;
      balance = newBalance;
   }

   /**
      Withdraws money from the bank account.
      @param amount the amount to withdraw
   */
   public void withdraw(double amount)
   {
      double newBalance = balance - amount;
      balance = newBalance;
   }
   /**
      Gets the current balance of the bank account.
      @return the current balance
   */
   public double getBalance()
   {
      return balance;
   }
}
```

Here is a simple test program that exercises all member functions:

```
int main() {

        BankAccount harrys_account(1000);

        harrys_account.deposit(500); // Balance is now $1500

        harrys_account.withdraw(2000); // Balance is now $1490

        harrys_account.add_interest(1); // Balance is now $1490 + 14.90

        cout << fixed << setprecision(2) << harrys_account.get_balance() << endl; return 0;

}

        Program Run 1504.90
```

Now create two subclasses for Checking and Saving accounts. A checking account has an overdraft limit (say $1,000 with a $25 fee charged), but a savings account **cannot** be overdrawn.

The following website provides you with what you need to know about bank accounts
http://www.diffen.com/difference/Checking_Account_vs_Savings_Account

Enhance the bank account class and the subclasses to simulate the followings se behaviors:

- The Bank charges a $1 fee for every transaction that occurs in a given month –after giving 5 free transactions-. The fee is applied at the end of the month. Each deposit and withdrawal counts as a transaction. Mercifully, there is no charge for getting the balance.
  the fee can be deducted by calling the **void endOfMonth()** method

- The Bank, also pays interest on the minimum balance during the month. For example, if the account started out at $1000, then dipped to $500, and then went to $2000, the interest is only paid on $500.
  Adding the interest does not count as a transaction with a fee. The interest is added before fees are deducted.

Remember that you need minimumBalance, initialBalance, and don't forget to count transactions.

Now, Instructors and students has bank accounts. Modify the previous (Instructor and Student) classes to show that. Use the objects from those classes to test your code.

Now your tester should be interactive (Use a menu). Requesting if you want to create a new account, deposit, withdraw… Exit.

Make sure to validate your entry.

Draw the UMLs

Effectively document your code.

Paste all of your classes in one word document file with sample output and upload.