Digital Product Architecture and Design – Coding Assignment #1

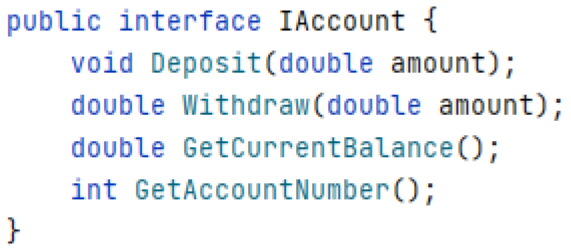
In this assignment, you will create a bank.

Our Bank has three different account types: standard, basic and premium.

Each of the account types should be represented as a class

Account Interface

Each of the account classes must implement the IAccount Interface



```
public interface IAccount {
    void Deposit(double amount);
    double Withdraw(double amount);
    double GetCurrentBalance();
    int GetAccountNumber();
}
```

Deposit: adds the amount provided as a double to the account balance.

Withdraw: withdraw the requested amount from the account according to the account restrictions, the value returned is the actual amount that has been withdrawn as a double.

GetCurrentBalance: Returns the current account's balance as double.

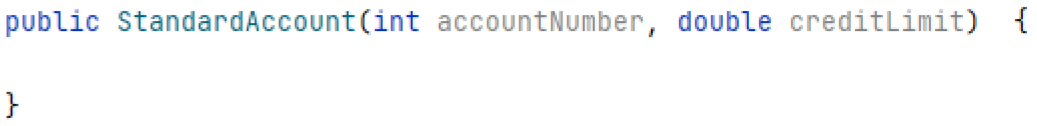
GetAccountNumber: Returns the account number as an integer

Standard Account

In standard account we can deposit and withdraw money up to our credit limit, we should not allow withdrawals that exceeds the credit limit and if we try to withdraw more we should only provide the amount allowed.
For example if the credit limit is -100, our balance is +100, and we try to withdraw 500 – We will only get 200.

Class name: StandardAccount

Constructor:



```
public StandardAccount(int accountNumber, double creditLimit)  {

}
```

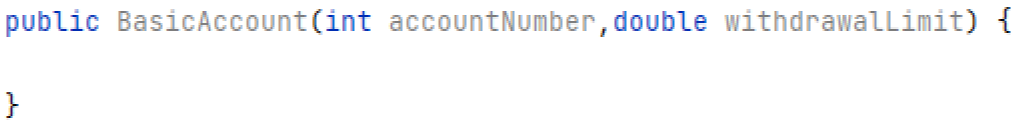
This Constructor gets as an arguments the account number represented as int and the acount creditLimit as double.
Note: the credit limit must be negative and represents the overdraft that allowed in this account, if a positive creditLimit is sent you should treat it as Zero.

Basic Account

Basic accounts cannot have a credit limit and it is always sets as zero.
Basic accounts also have a single withdrawal limit so we should not provide withdrawals more than that amount.
For Example if the withdrawal limit is 100 and we try to withdraw 500, we will get only 100. The credit limit rules also applies to this account

Class name: BasicAccount

Constructor:



```
public BasicAccount(int accountNumber,double withdrawalLimit) {

}
```

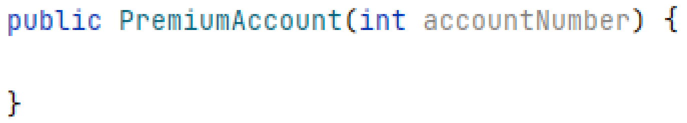
This Constructor gets as an arguments the account number represented as int and the withdrawal Limit as a positive double.

Premium Account

Premium accounts are similar to standard accounts with the one difference that they do not have credit limit – we can withdraw as much as we want

Class name: PremiumAccount

Constructor:



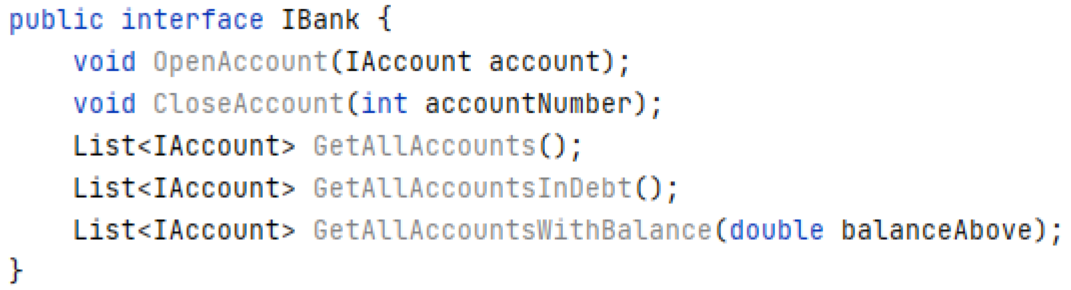
```
public PremiumAccount(int accountNumber) {

}
```

This Constructor gets as an arguments the account number represented as int.

The Bank

Now that we have our account we need a class that will represent our bank. Our bank will hold a list of accounts and will be able to preform several actions The bank class must implement the following interface:



```
public interface IBank {
    void OpenAccount(IAccount account);
    void CloseAccount(int accountNumber);
    List<IAccount> GetAllAccounts();
    List<IAccount> GetAllAccountsInDebt();
    List<IAccount> GetAllAccountsWithBalance(double balanceAbove);
}
```

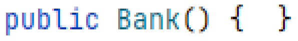
OpenAccount: gets an account object and adds it to the bank accounts list.

CloseAccount: gets account number as an integer, look him up in the bank account list, if we didn't find it, we ignore the request. If we found the account, we check if the balance is positive and removes him from the bank account list. If the balance is negative we print a message that the account is not closed due to debt, you can print using System.out.println() method.

GetAllAccounts: returns all of the accounts in the bank as List of accounts

GetAllAccountsInDebt: returns a list of accounts with negative balance.

GetAllAccountsWithBalance: returns a list of accounts with balance bigger than the supplied double

Constructor: 

```
public Bank() {  }
```

Submission and Grading

The Code of this assignment will be submitted via GitHub.

You must push the final program into a repository. There is no obligation to make commits during the work on the program although it is recommended.

You will submit in Moodle an explanation page about your program in PDF Format, This explanation page will include link to the GitHub repository as well as basic explanation about how you’ve built your program, this should be no longer than 1-2 pages.

Your program will be graded as follows:

- 50% Automatic tests.
  - A set of 10 automatic tests will run on your program to test its behavior. 5 points will be awarded for each success.
  - In order for your program to comply with the automatic tests you must use the exact class and interface names and constructors structure that was described in this assignment
- 50% General Code Quality

Late submissions allowed up to 5 days, each day would incur 10 points penalty.