This program contains 6 main java classes and 5 java test classes. Below we briefly explain what each class does and how a user can use this program.

**Main Classes:**

**Class Client:**

This class represents the clients of CIBC bank who are willing to open an account. As a client, this class holds variable SIN, showing the SIN number of this client and name, representing the name of this client.

**Class DemandLoanAccount:**

This class makes an object of demand loan account when the bank decides to terminate a client’s account or when a client decides to cancel their account. An object of this class holds the balance of the cancelled/terminated account as the credit/debt of that client depends on if the balance has been positive or negative. Variables of this class are:

**Client**: the client of bank owning this demand loan account.

**Amount**: showing how much debt/credit is inside this account.

**Boolean isDebt**: represents if the amount is credit for client, if it is false, or if it is debt, if it is true.

**Abstract class Account:**

The main abstract class as a super class for credit accounts and checking accounts, by having and initializing all the common fields and methods.

This class has below variables:

**Owner**: an object of class client showing the owner of this account.

**isSuspended**: a boolean variable showing if the account is suspended.

**accountNumber**: account number of this account

**Balance**: balance of this account.

This class initiate the common variables of other classes and defines the methods suspend, reactivate, getBalance, create account, deposit, terminate and cancel for all the accounts. In the definition of all these methods, an object of class AccountActivity with proper arguments passing to its constructor is created to be saved in the static field activityLog of AccountActivity class. The more details about this class and different constructors of this class is available in its own section.

Besides, two methods terminateAccount and cancelAccount both create an object of class DemandLoanAccount after the cancellation/termination of main account to hold the amount of debt/credit for that client. Also this abstract class has getters methods for its fields and abstractly introduces withdraw, getAccountTyoe and transfer for its child classes.

**Class CreditAccount:**

This class extends the abstract class Account and is creating objects for credit accounts. Beside the variables of class Account, it has creditType to show what type of credit account it is, either high credit account or low credit account, and variable creditLimit representing how much credit amount this account has. Plus, it has some more static fields designed based on requirements of CIBC bank. The constructor of this class initiate the variables of parent class and then variables of this class.

This class has following methods:

**setCreditLimit**: setting amount of credit limit based on argument and then based on its amount, it sets the type of credit account to low or high, if it is greater than 1000 or not.

**getAccountTyoe**: return the type of this account as a string.

**Withdraw**: implement a definition for the super class method, based on the type of credit account, this method applies a exceeding credit limit fee for withdrawals than exceed the credit limit and then decline for high credit account and just decline for low credit account and then update the account balance. This method also create its own account activity object for showing the withdrawal activity.

**Transfer**: this method first makes an account activity object for transfer action, then withdraw the amount from main account and deposit it to destined account.

**NOTE**: implementation of this class requires that after creating the object of class, the user has to set this account’s credit limit by its method.

**Class CheckingAccount:**

This account extends the Account class and creates objects of checking account. Beside the variables of super class it also has field overdraftOption, indicating which option for overdraft debt is chosen for this account( among 3 possible options) and overdraftLimit showing how much limit its overdraft debts has. It also has some more static fields that are designed based on CIBC requirements. This class has following methods:

**setOverdraftOption**: sets the type of overdraft indebtedness option for this account.

**setOverdraftLimit**: sets the amount of overdraft indebtedness determined for this account.

**getAccountTyoe**: return the type of this account as a string.

**Withdraw**: implement a definition for the super class method, based on the type of checking account. This method applies a non sufficient fee for withdrawals that cause the balance to be negative, or if a protection plan is chosen, withdrawals that leas to balance be less than overdraft limit, and then decline it. This method also create its own account activity class object passing requires arguments to it.

**transfer**: this method first makes an account activity object for transfer action, then withdraw the amount from main account and deposit it to destined account.

**NOTE**: implementation of this class requires that after an object of this class is constructed, then the overdraft option and overdraft limit for this account have to be set by calling their own methods.

**Class AccountActivity:**

This class is responsible for creating an object for every activity of credit and checking accounts. In order to do so, this class has its own variables which are as below:

**activityType**: it represents what has been the type of activity of an account’ transaction.

it represents:

1 for withdraw an amount

2 for deposit an amount

3 for creating an account

4 for cancelling an account

5 for suspending an account

6 for reactivating an account

7 for getting balance of an account

8 for terminating an account

9 for setting overdraft option of an account

10 for setting credit limit or overdraft limit of an account

11 for transferring an amount

**mainAccount**: main account that has done this activity.

**mainAccountType**: type of the account doing this activity.

**Time**: time that this activity has been processed.

**Balance**: balance of this account after this activity.

**Amount**: amount of money that has been deposited or withdrawn or transferred, depends on the type of this activity. Only used for related activities.

**demandLoan**: demand loan account object resulted from cancellation/termination of an account. Only used for related activities.

**isSuccessful**: boolean variable showing if the withdrawal has been successful. Only used for related activities.

**destinedAccount**: desalinated account of a transfer activity. Only used for related activities.

**newLimit**: newly set limit of a credit account or newly set of overdraft limit of a checking account. Only used for related activities.

**Static activityLog**: an arraylist of account activity objects which each one holds an activity that an account has done. All the activities are saved in this arraylist.

This class has different constructors to create an accountactivity object for different type of activities.

**C1** is used for create an account, suspend an account, reactivate an account, getting the balance of an account.

**C2** is used for cancelling and terminating an account

**C3** is used for depositing an amount to an account

**C4** is used to withdraw an amount from an account

**C5** is used money transfer.

**C6** is used to set the overdraft option of an account

**C7** is used for setting the overdraft/credit limit of an account.

This class also has all the necessary getter methods to return the variables of objects of this class. The following static methods are responsible for processes that CIBC bank asked for:

**Static method sortActivityLog:**

static method to sort all the accountactivities inside the activitylog arraylist based on the increasing order of SIN number of clients who own the credit or checking accounts. in case of two or more activity have the same Sin number (so they have the same owner client) they are sorted by increasing order of time of implementation of activity.

**Static method endOfDayProcess:**

static method responsible for completing the end of the day transactions. this method apply pay per use fee for all the checking accounts that have had a withdrawal in this day that caused their debt be increased or their account balance become negative while their overdraft option have been set on pay per use. also this method implement all the account terminations with creating a demand loan account for that client and make the balance of their account equal to zero. This method returns an arraylist of demand loan account objects for clients whose accounts have been terminated in this day.

**Static method endOfMonthProcess:**

static method responsible for completing the end of the day transactions. this method applies monthly fixed fee to checking account which overdraft option of fixed monthly paty. this method returns an arraylist of demand loan account objects for clients who have cancelled their account within this month. besides, this method applies the interest amount as a withdraw to all the credit and checking accounts that are in debt( with negative balance). also this method implement all the account cancellations with creating a demand loan account for that client and make the balance of their account equal to zero.

**Static method saveActivityLog:**

This method saves all the activities inside the activityLog inside a file named activityLog.ser at the project folder.

**Static method retrieveActivityLog:**

This method retrieves all the activities inside the file named activityLog.ser at the project folder and saves it in an arraylist of accountactivities.

**NOTE** that all these classes are that they are implement the interface Serializable to ensure that saving and retrieving the objects to/from file is possible. Besides all these classes are meant to be inside the package CIBC\_design.

**Test Classes:**

This test classes are responsible for testing the accuracy of incomes of each classes and making sure that classes create correct outputs specially in the special cases. Beside testing the created objects of classes, These tests classes also evaluate the outputs of methods within classes to ensure they produce the proper result.

**Class TestFirst:**

This class firstly creates some objects of class clients to use those to initiate some objects of checking account and credit account classes. After that, by printing the output of results, this class first assess the correction of different methods like getBalance or suspend for both type of classes. It also assess the outcome of getClient method of accounts to make sure it is the same created client for that account.

After that we further initiate the accounts by setting overdrafLimit and OverdraftOption for them and in case of credit account, setting creditLimit. Then, we test the methods like deposit and withdraw and transfer to ensure that these methods work properly in changing the balance of account and keeping the methods and class invariants. We also implemented some tests to make sure they apply the non sufficient fee and penalties in case if there is not enough money in an account for withdrawal or transfer. We implemented these test for all different overdraft options of checking accounts and credit account types.

**Class TestSecond:**

This class starts by creating some objects of credit account and checking account classes. This class is willing to test the copy constructors of checking and credit account classes an in order to do so, this class make new objects by use of copy constructors from the previously made objects, and then it evaluate if the two objects have the exact same fields and arguments.

**Class TestThird:**

This test class is mainly responsible to evaluate if all the considered methods in both credit account and checking account classes create objects of related account activity classes and will successfully save it into the static variable of account activity class, named activityLog. It also tests to guarantee that the created account activity objects has the proper arguments and holds the correct information to pass all the necessary information about that accomplished transaction or activity.

In order to do so, we made objects of credit and checking classes, then implement different activities on them, for example deposit, transfer, reactivate, termination or in general any action which is required to create an account activity object. Then by getting the related element of activityLog arrayList and extracting the related information from it by getter methods and printing out its result, we made sure that outcome matches the activity.

**Class TestForth:**

This class applies some tests to assess the correctness of insertion sort method of class accountActivity. This methods name is sortActivityLog and to test it, this class first creates some object of different accounts and then does some activities on it and then it calls the the sortActivityLog method. After that, by getting the random element of sorted arrayList of account activities, by printing out different information about those like SIN number or the time of activity, it checks if the elements are properly sorted by increasing order of SIN number of clients and in case of equality of SIN numbers, by increasing order of time of activity.

**Class TestFifth:**

This class is attempting to test the other 4 methods of class accountActivity. It starts by instantiating some objects of both accounts and then does some different transactions and activities on them and then it calls the endOfDayProcess, which is responsible for completing the end of day processes on accounts. After that this class evaluates if this method has successfully terminated all the accounts that have had the termination activity in this day and also if this class made an arrayList of objects demandLoanAccount class for the Loan account of client who’s account is terminated. It test to make sure it is the same owner with same amount of indebtedness. Besides that, it checks if this method applies the pay per use fee on checking accounts with related overdraft option that has done a transaction that day that leads their balance become negative, or increase their indebtedness.

Then it calls the endOfMonthProcess method which is responsible for doing the required bank processes on accounts at the end of the month. then, this test class evaluates if all the in debt account, either credit or checking, have been charged with monthly interest in portion of their debt and based on the interest rate of that type of account. It also test to see if the monthly fixed fee option for the checking accounts with related protection plan is applied. moreover, this class also test to see if the process of account cancellation for accounts that have been cancelled is properly processed or not.

And at the end, this test class checks to ensure if the saving and retrieving of activityLog by means of two methods, saveAccountLog and retrieveAccountLog, is successfully done by calling them and evaluating the output.