The University of Melbourne
Department of Computer Science and Software Engineering
433-254 Software Design
Second Semester, 2003

Project B: Version 1.0

Due Date: 5pm, Oct 21, 2003

Motivation Statement

An Australian National Bank, OzBank which has been carrying out its operations using batch-processing based computer systems, would like to move to *interactive computing systems*. It also wants to enhance its services by supporting electronic banking and 7x24 banking via ATMs. It installs ATM machines in many locations in major cities. They would like to procure banking software that helps in automating all of their operations and also supports *interactive* banking. Their CIO (Chief Information Officer) found that they are many vendors who have been developing banking software and looking for customers. Instead of developing in house, the bank decided to procure the software through open and competitive process. The bank also prefers to procure from a vendor who is willing to study their current system and make sure that their software offer all required functionality. The bank also expects that the vendor should be willing to take feedback from them and update the software before they adopt it.

OzBank issues an open call for tenders. Many software vendors respond to this call with promise to supply interactive and innovative software at lower price.

Now imagine yourself as an Australian software vendor, read the project spec given below and develop a banking software, which is interactive (menu driven) and intuitive. Make your software as innovative as possible to win the OzBank contract. Note OzBank CIO may not have time to answer each and every query; you are expected to study the Spec carefully and produce output. If they are any errors in data entry or incorrect transactions, you need to report appropriate errors. The bank will pay for your software only after you demonstrate its functionality to the satisfaction of CIO and also nominated employees who will be using the software in running the day-to-day operations of the bank.

Now imagine yourself as a "software vendor", other students are competing software vendors, your lecturer as "CIO" and "markers" as employees of bank who are part of software acceptance certification team, and implement this Project B with open mind to win the contract. All the best!

Note: There is NO automated testing procedure, you will be demonstrating it to a real human evaluator (mostly lab-in-charge) during the lab class that follows project submission due date. The demo evaluator will ask you to demonstrate some or all functionalities stated in the spec and he/she (the demo evaluator) will "decide" functionalities to be demonstrated! If they are any specific procedure to be followed during the demo, they will be posted on the web a week before the submission due date.

Problem Statement for Project B

The Project B should be able to handle the following high level requirements.

1. The bank should be able to maintain records for all its customers.
2. The record for a customer should have the following information.
    a. A unique customer ID – 10 digit

b. Surname
c. First Name
d. Telephone number
e. email address
f. A list of accounts for the customer.

3. The bank can have 2 types of accounts, Checking and Savings. Each account will have the following information.
    a. Account Number
    b. Branch Name
    c. Account Balance

4. A customer could have up to a maximum of two accounts, one of each type. Each account has a minimum opening balance requirement (See detailed requirements).

5. Every customer in the Bank will be entitled to have one ATM (Automatic Teller Machine) card. The ATM card can be linked to one of the accounts for the same customer. An ATM card shall have the following information.
    a. Card ID
    b. Maximum Withdrawal limit (the amount of cash that can be withdrawn in one transaction).
    c. PIN Number

6. The following operations shall be supported in relation to ATM cards.
    a. Adding a new ATM card to a customer and linking it to an account.
    b. Linking an account (for the same customer) to an existing ATM card.
    c. Given the Customer ID getting ATM card information

7. The bank should support the following operations.
    a. Opening and closing accounts.
    b. Balance Inquiry
    c. Generating reports

8. The bank shall support the following operations.
    a. Cash Withdrawal
    b. Cash Deposit
    c. Fund Transfer from one account to another (both accounts should belong to the same customer)
    d. Cash Withdrawal from ATM

9. The Bank should keep a history of all the transactions specified above (specified in 8 (a) – (d)) that were performed on an account and should support the capability of printing a statement of all the transactions for the account.

10. Customer, Accounts, and Transaction detailed have to be maintained in a persistent storage (e.g., in files on hard disk) so system can be shutdown (stopped) and brought back into operation without losing any data. *It is recommended that all data be stored in binary format due to its support for efficient storage and retrieval.*

Functional Requirements

<ProjB-10> The bank shall have two different types of accounts.

- Checking Account
- Savings Account

<End ProjB-10>

<ProjB-20> Each account type shall have a minimum opening balance (the initial deposit to open the account) as specified below.

- Checking Account - $2000
- Savings Account - $500

<End ProjB-20>

<ProjB-30> The bank shall have the following information related to the account.

- Account Number (10 digit number)
- Branch Name
- Account Balance

<End ProjB-30>

<ProjB-35> The bank shall have the following information related to customers.

- A unique customer ID – 10 digit
- Surname
- First Name
- Telephone number
- email address
- A list of accounts

<End ProjB-35>

<ProjB-40> Account Number shall be a unique 10 digit number with the following constraints on the first digit

- Checking Account - 1
- Savings Account - 2

E.g. 1001200300 is a Checking Account. 2002345123 is Saving Account.

<End ProjB-40>

<ProjB-50> A customer may have up to a maximum of two accounts at the bank. However, only one account of a particular type is allowed.

<End ProA-50>

<ProjB-60> The following operations shall be supported.

- Open Account – OPEN_ACCOUNT
- Close Account - CLOSE_ACCOUNT
- Balance Inquiry – BALANCE_INQUIRY
- Generate Report – GENERATE_REPORT

<End ProjB-60>

<ProjB-70> An OPEN_ACCOUNT operation shall open a new account in the bank. The following information shall be provided as input parameters:

- Account Number

- Branch Name
- Opening Balance
- Customer ID
- All information related to customers specified in requirement <ProjB-35> if the account is for a new customer. If this information is not specified it can be assumed that this is a new account for an existing customer.

<End ProjB-70>

<ProjB-110> A CLOSE_ACCOUNT operation shall close an existing customer account. Customer ID and the Account Number being deleted. If the account deleted is the last account for the customer the customer information shall be removed from the system.

<End ProjB-110>

<ProjB-120> A BALANCE_INQUIRY operation shall report the account balance. The Customer ID and Account Number and the operation shall be provided as input parameters.

<End ProjB-120>

<ProjB-130> A GENERATE_REPORT operation shall report the bank account summary. The summary report shall show the account type, total number of accounts and the total balance for the 2 different types of accounts supported by the bank for each customer.

<End ProjB-130>

<ProjB-200> Every customer in the bank shall be entitled to one ATM card. The bank shall have the following information related to an ATM card.

- Card ID
- PIN number (4 digit number)
- Maximum Withdrawal limit

<End ProjB-200>

<ProjB-220> The Card ID shall be a unique 10 Digit number.

<End-ProjB-220>

<ProjB-230> The following operations shall be supported in relation to ATM cards.

- Add an ATM card to an existing customer - ADD_CARD
- Link an Account to an existing ATM Card for the customer – ADD_ACCOUNT_TO_CARD
- Get Card information given customer ID – GET_CARD_INFO

<End ProjB-230>

<ProjB-235> An ADD_CARD operation shall add a new ATM card to an existing customer and link it to the specified type of account for the customer. The following information shall be provided as input parameters.

- Customer ID
- Card ID

- PIN number
- Maximum Withdrawal limit
- The type of account to be linked to (Checking Account – CHK, Savings Account – SAV)

Note: The acronyms CHK and SAV mentioned in the rest of the document will refer to the particular types of accounts as specified in this requirement.

<End ProjB-235>

<ProjB-240> An ADD_CARD operation shall fail if an ATM card already exists for the customer or if the specified type of account does not exist for the customer.
<End ProjB-240>

<ProjB-270> An ADD_ACCOUNT_TO_CARD operation shall associate the specified account to an existing ATM Card. The following information shall be provided as input parameters.

- Card ID
- The type of account to be linked to (Checking Account – CHK, Savings Account – SAV)

Note: This operation is like updating/changing the attachment of the ATM card from one account type to another.
<End ProjB-270>

<ProjB-280> An ADD_ACCOUNT_TO_CARD operation shall fail if the specified type of account does not exist for the customer. If the specified type of account is already associated to the CARD the command shall succeed.
<End ProjB-280>

<ProjB-300> A GET_CARD_INFO operation shall get the card ID, Maximum Withdrawal limit and the account number linked to the card, for the specified Customer ID. The output shall be display on screen. The output shall have the following information.

- Customer ID
- Card ID
- Maximum Withdrawal limit
- Account Number that is linked to the card

<End ProjB-300>

<ProjB-310> The bank shall support the following operations for accounts.

- WITHDRAW_CASH
- DEPOSIT_CASH
- TRANSFER_FUNDS
- WITHDRAW_CASH_ATM

<End ProjB-310>

<ProjB-320> A WITHDRAW_CASH operation shall reduce the account balance by the amount specified in the withdrawal operation. The following parameters shall be provided as input arguments to the operation.

- Customer ID
- Date
- Account Type (CHK, SAV)
- Withdrawal Amount

<End ProjB-320>

<ProjB-330> A DEPOSIT_CASH operation shall add the amount specified in the transaction to the account balance. The following parameters shall be provided as input arguments to the operation.

- Customer ID
- Date
- Account Type (CHK, SAV)
- Deposit Amount

<End ProjB-330>

<ProjB-340> A TRANSFER_FUNDS operation shall transfer the specified amount of funds from one account to another (both accounts must belong to the same customer). The following parameters shall be specified as input arguments to the command.

- Customer ID
- Date
- From Account Type (CHK, SAV)
- To Account Type (CHK, SAV)
- Amount

<End ProjB-340>

<ProjB-350> A WITHDRAW_CASH_ATM operation shall reduce the account balance by the amount specified in the withdrawal operation. The following parameters shall be provided as input arguments to the operation.

- Card ID
- PIN Number
- Date
- Account Type (CHK, SAV)
- Amount

<End ProjB-350>

<ProjB-360> A WITHDRAW_CASH_ATM operation shall fail if the PIN number specified as an input argument to the operation does not match the PIN number of the card.

<End ProjB-360>

<ProjB-370> A WITHDRAW_CASH_ATM operation shall fail if the specified type of account is not linked to the ATM card.

<End ProjB-370>

<ProjB-380> A WITHDRAW_CASH_ATM operation shall fail if the specified withdrawal amount is greater than the maximum withdrawal limit.

<End ProjB-380>

<ProjB-400> WITHDRAW_CASH, TRANSFER_FUNDS and WITHDRAW_CASH_ATM operations shall fail if the funds available in the account as a result of the withdrawal or transfer shall fall below the minimum opening balance for the type of account.

<End ProjB-400>

<ProjB-410> The bank shall maintain a history of deposit, withdrawal and transfer type operations, that were performed on a particular account. Each transaction shall have,

- Date
- Description - Use command name as description. Following descriptions are allowed
  - WITHDRAW_CASH,DEPOSIT_CASH, TRANSFER_FUNDS, WITHDRAW_CASH_ATM
- Amount – A transaction which causes the account balance go down shall be shown as a negative amount and a transaction which adds to the balance shall be shown as a positive amount.

Note: TRANSFER_FUNDS operation shall result in two different transactions, one for the account the funds are transferred from and another for the account the funds are transferred to.

<End ProjB-410>

<ProjB-420> The bank shall support a ACCOUNT_HISTORY operation that shall print all the transaction information for a particular account. The following information shall be requested as input:

- Customer ID
- Account Type (CHK, SAV)

<End ProjB-420>

Input/Output Requirements

<ProjB-500> All input commands shall be read interactively from the standard input device (keyboard). You need to display appropriate text message with request to enter data.

NOTE: There is NO automated testing for this Project. You will be demonstrating Project B in operation.

<End ProjB-500>

<ProjB-510> Your program should be able to recognize bad input data and display appropriate error message. Then ask the user whether they want to continue further, if yes continue to request for data along with remaining data items. Otherwise, terminate the operation without storing incomplete data in the bank database.

After successful completion of selection operation, please go back to the main menu.

<End ProjB-510>

<ProjB-520> All output results or error messages shall be written to a standard output device (screen) unless otherwise mentioned.

<End ProjB-520>

<ProjB-530>

Customer, Accounts, and Transaction detailed have to be maintained in a persistent storage (e.g., in files on hard disk) so system can be shutdown (stopped) and brought back into operation without losing any data.

<End ProjB-530>

<ProjB-540>

Selection of various operations (see Table below) is made via an *interactive text-based menu*. That means, you need display a text menu listing all operations supported by your system and let the user select one of the option.

For Example: The main menu can be something like:

```
OzBank Operations System
************************
A)dd Account
G)enerate Report
...
E)Exit

Enter your command:
```

So if they type "A" (or "a" as the case may be), you'd them prompt for the info that you need to add an account. *Display error message if requested operation is NOT supported*.

Alternatively, the menu can also be something like:

```
OzBank Operations System
************************
[1]  Add Account
[2]  Generate Report
...

[13] Exit
Enter your command [1-13]:
```

Note 1: The above examples are just two samples. Please feel free to choose any other menu format that appeals to you and you think that also appeals others! Try to make it as simple and elegant as possible so that you can focus more on OO design.

Note 2: This project clearly specifies that you need to implement is text-based user-interface (UI). If someone wants to implement it as GUI, I am still fine with it. There is no change in spec. and no special marks will be given. However, I strong recommend that you do it as text-based UI to save time and energy for other purposes!

Type of Operation Requested	Input Items to be Requested / Output to be displayed 1. For each item, issue a separate request and read separately. There is NO need to supplying/reading all of them from a single input line. 2. Please display appropriate message/prompt with request to enter the input item.
OPEN_ACCOUNT	**Account for a New Customer:** CustomerID, AccountNumber, BranchName, OpeningBalance, SurName, FirstName, TelephoneNumber, Email
	New account for an existing customer: CustomerID, AccountNumber, BranchName, OpeningBalance
CLOSE_ACCOUNT	Customer ID, AccountNumber
BALANCE_INQUIRY	Customer ID, AccountNumber Display account balance information.
GENERATE_REPORT	Generate Report of Accounts Display information as specified in <ProjB-130> for each customer on a new line. Consider displaying the out in tabular format.
ADD_CARD	CustomerID, CardID, PINNumber, MaxWithdrawLimit, AccountType
ADD_ACCOUNT_TO_CARD	CardID, AccountType
GET_CARD_INFO	Customer ID Display card info, each item on new line with appropriate message.
WITHDRAW_CASH	CustomerID, Date, AccountType, Amount
DEPOSIT_CASH	CustomerID, Date, AccountType, Amount
TRANSFER_FUNDS	CustomerID, Date, FromAccountType, ToAccountType, Amount
WITHDRAW_CASH_ATM	CardID, PINNumber]Date, AccountType, Amount
ACCOUNT_HISTORY	CustomerID, AccountType If Customer with given ID had a Account of type specified, display transaction history information.

<End ProjB-540>

Use the object oriented design paradigm to design and develop a Java program to meet the above requirements.

Deliverables:

1. A UML Use Case diagram for the system.

2. UML Collaboration diagrams for the success scenarios of the following operations.
    a. OPEN_ACCOUNT
    b. DEPOSIT_CASH
    c. WITHDRAW_CASH

3. UML Sequence diagrams for the success scenarios of the following operations.
    a. OPEN_ACCOUNT
    b. DEPOSIT_CASH
    c. WITHDRAW_CASH

4. A UML Class diagram for the system showing the classes used in your design and the relationships.
    a. The class diagram should show all the classes used in the application.
    b. Class relationships should show the details such as the relationship name, roles, multiplicities and direction where applicable.
    c. Classes used in the class diagram show a few prominent members of the class.

5. Assumptions you had to make in the design/coding if you feel some requirements not explicitly specified.

6. Screenshot after successful completion of following operations:
    a. Opening of a new account (OPEN_ACCOUNT)
    b. GET_CARD_INFO

    You can get this by pressing "PrintScreen" and pasting into your WORD document or capture window and than save and print.

7. Well documented code that implements the above requirements.

8. As stated earlier, you need to demonstrate functionality of your system during the Lab class that follows your project submission due date.

Note:

- If you can familiarize yourself with any UML design tool which can be downloaded from the web (254 subject page has links to some tools) you can use the tool to draw the UML diagram mentioned above. Otherwise, you can use any diagram editor you are familiar with to draw the UML diagrams. One of the above methods is preferred but if you are not able to use either one of the above approaches for your diagrams you can draw your diagrams (clearly) by hand.

- Please put your design diagrams in the order specified above (1-6) with cover sheet (which will be provided) on top.

Assessment

1. Design - 8 marks.
2. Java Code - Well documented code should be submitted (12 marks), marks will be awarded for:
    (a) Style,
    (b) Readability,
    (c) Documentation
    (d) Correctness.
    (e) Implementations matching with the design.

Submission

You should submit your design and assumptions you had to make (if you had to make any) in hardcopy form to the submission box marked 254 Project B, in Level 1. Please make sure the name, login and student number are clearly written on the cover sheet.

All your source files (.java files) should be submitted using the submit command by the deadline

submit 254 B [File Names]

e.g. submit 254 B OzBank.java Account.java ……

Please make sure that the entry point of your program (main) is in a file called OzBank.java
i.e. your program will be invoked with the command

java OzBank

You can submit as many java files as you like but make sure that your main program is OzBank.java.

You should then verify the submission using the command:

verify 254 B

It is important to ensure your system can interpret the commands stated in the requirements.

Late submissions should be sent to A.late using the above commands. Late submissions will be penalized at 1 mark per day late.

Individual Work

You are reminded that all submitted work in this subject is to be your own individual work. Students submitting the work of others for assessment will be penalized by the Department, and risk prosecution under the University's Discipline Regulations. Students who allow other students access to their work will be penalized, even if they themselves are sole authors of the program in question. Automated similarity checks will be used to compare submissions.

Questions and further Information

The subject WW home page will be kept updated with any further information relevant to the project and will be considered as part of the specification for the project. **Questions about the project specification should be directed to the newsgroup cs.254.** Even with these resources, note that there may still be some cases where there is ambiguity about how the program should behave. In such situations, you should make a sensible assumption and justify and document it.

Please **direct any submission related issues/problems** such as extension due to illness to our head tutor (Mr. Saeed Araban). Since the project spec has been released 4 weeks before the submission deadline, please do your project early and avoid last minute excuses/problems.

After projects are evaluated and results are announced: If you have any real concerns on marks that you received, please contact our head tutor (Mr. Saeed Araban) since he is coordinating your submissions and marking.