LAB 9 : ACCOUNT CLASS

```
Account
--------------------------------
accountName : string
accountNumber : string
accountType: int
accountBalance : float
--------------------------------
getBalance() : float
withdraw(float): void
deposit(float) : void
printDetail() : void
```

Write a complete C++ program that is able to do the following tasks:

1. Write a class definition based on the class diagram above. Declare the data members as private, while the member functions as public.
2. Define the constructor for class account by one of the following method:
   i. Overload constructors - Define the first constructor without parameter and the second constructor with parameter. Both constructor will initialize the data members with suitable values.

   ii. Constructor with default argumen - initialize the data members with suitable values.

3. Write implementation function for the following member function:

   i. **`getBalance( )`** – return the the account balance.

   ii. **`withdraw(float )`** – allow user to withdraw some money. Check whether the balance in the account is sufficient to be withdrawn.

   iii. **`deposit(float)`** – allow user to deposit some money and update the balance in the account.

   iv. **`printDetail()`** – print all the information in the account.

4. Write **`main()`** program that will declare 2 instances of object account. Declare the instances based on the parameters provided in the constructor.

5. Implement all member functions to show the transactions of the accounts using: **`getBalance( )`**, **`withdraw( )`**, **`deposit( )`** and **`printDetail()`** which are accessed through the objects.

6. Declare an array of account with size 10. Read information about the account from a data file. Add necessary function to initialize the array elements.

7. Add one member function for account class, named **`transfer(account a)`** that will allow transfer to be done from one account to another. Use appropriate passing value approach, either by value or reference. Ask the user to insert an account Number and the program should search for the account number. If the account does not exist, give proper message and if the account exists, implement the transfer function. Give appropriate message to show the transaction process has successfully being processed.

Overview

- This exercise is to be conducted **in-class individually.**

Problem

- Do the question **LAB 9 : Account Class** from the exercise book, page 50 – 51.
- Do tasks 1, 2, 3 , 4, 5 and 7 and some amendments below.
- Write the program in multiple files: Your program should comprise the following files:
    - Specification /header files (*.hpp)
    - Implementation / definition files (*.cpp)
    - Main file (e.g., main.cpp)

Amendments

- In Task 3(ii), as for the implementation of the method `withdraw(),` the method will throw an **exception** if there is not enough money to withdraw from the account. Define a dedicated exception for this requirement.
- Add another getter method to the class for the attribute `accountNumber.`
- In Task 7. Do only adding the method `transfer()` to the class. Also, do make use of the methods `withdraw()` and `deposit()` in the implementation of method `transfer()`
- In Task 4, as for the main function, include the following sub tasks:
    - You may use hard-code values to create the accounts.
    - `Print the initial balance of both accounts.`
    - `Deposit an amount of money into the first account`
    - `Withdraw an amount of money from the second account`
    - `Transfer an amount of money from the second account to the first account.`
    - `Print the details of both accounts`
- Up to this point your program should print the output as shown in Figure 1

- Define two regular functions in a separate file to perform the following file operations:
  - `saveAccount()`: To save an account object into a binary file. This function accepts the account to be saved as a parameter. It uses the account's number as the file's name.
  - `loadAccount()`: To read an account object from a binary file. This function accepts an account's number (of type string) as its parameter. It returns the account object read from the file.
- Use the functions you defined above to modify the existing program as follows:
  - Save both accounts, the first and second into files.
  - Declare another account object named `third` and load the data of the second account from the file and assign the object to the third account.
  - Print the details of the third account.
  - Example output from this task is shown in Figure 2.

Expected Output

```
INITIAL BALANCE
First Account balance: RM0
Second Account balance: RM1000

AFTER DEPOSITING RM 200 INTO THE FIRST ACCOUNT
First Account balance: RM200

AFTER WITHDRAWING RM 200 FROM THE SECOND ACCOUNT
Second Account balance: RM800

AFTER TRANSFERING RM 100 FROM THE SECOND ACCOUNT TO THE FIRST
ACCOUNT
ACCOUNT INFORMATION
Account Holder's Name:
Account Number:
Account Type:0
Current Balance: RM300

ACCOUNT INFORMATION
Account Holder's Name:Razali Ahmad
Account Number:123456
Account Type:1
Current Balance: RM700
```

Figure 1

```
THIRD ACCOUNT
ACCOUNT INFORMATION
Account Holder's Name:Razali Ahmad
```

```
Account Number:123456
Account Type:1
Current Balance: RM800
```

Figure 2