IT 114 - ADVANCED PROGRAMMING FOR INFORMATION TECHNOLOGY

Programming Assignment 1: Bank Account

Author: Jennifer Soh

1. Formulating the Problem

1.1 Assignment Description

Design and implement a Java program that creates a class that represents an individual's bank account. The class should store information about the account balance, the name of the account holder, and an account number. There are a number of operations that need to be implemented via methods they are as follows: withdraw money, deposit money, check the balance and get account information such as name, and account number. When creating a new account you should display the initial balance, the name of the owner, and the account number.

1.2 Verbalization

What is the goal?

Create a program that outputs the transaction amount (amount before and after the transaction). Transactions include deposit and withdraw, checking the account balance, and getting the account information such as the account number. Upon creation, the new account should display the initial balance, the name of the owner and the account number.

What are the givens?

The givens are the name, initial balance, and account number as well as the calculations required to add or subtract amounts from the balance.

What are the unknowns?

The balance after every transaction is unknown.

1.3 Information Elicitation

Goal

Collect the user's name, account number, and initial balance. Collect deposit and withdraw amounts and calculate the balance, then return the balance after every debit and credit, then display the account summary.

Givens

The givens are the name, initial balance, and account number as well as the calculations required to add or subtract amounts from the balance.

Unknowns
The balance after every transaction is unknown.

Conditions

2. Planning the Solution

2.1 Solution Strategy

The user will start a new bank account by inputting his/her name, the initial deposit amount, and generate an account number. An account summary with the starting account information is then displayed. The user is then prompted for a withdraw amount and a deposit amount. After each debit or credit, the new deposit is displayed. Finally, the account summary is posted.

2.2 Goal Decomposition

Sub-goal 1
Get the user's name, account number, and initial balance.

Sub-goal 2
Post account summary.

Sub-goal 3
Request a withdraw amount from the user.

Sub-goal 4
Deduct the withdraw amount from the balance and post the balance.

Sub-goal 5
Request a deposit amount from the user.

Sub-goal 6
Add the deposit amount to the balance and post the new balance.

Sub-goal 7
Post new account summary.

2.3 Resources

Relevant formulas
Deposit: accountBalance + deposit amount
Withdraw: accountBalance – withdraw amount

Formula Derivation

The user provides and initial balance. When you withdraw, you deduct the withdraw amount from the balance. When you deposit, you add the amount to the balance.

2.4 Data Organization and Description

Input (givens):

Name	Description	Origin	Used in Sub-goal #
Name	User Name	User	1
acctNum	Account Number	User	1
initBalance	Initial Balance deposited	User	1
Debit	Withdraw amount from balance	User	3
Credit	Deposit amount added to balance	User	5

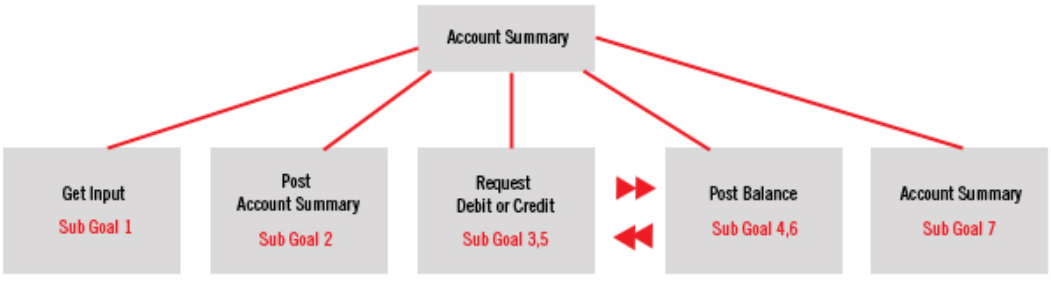
Output (unknowns):

Name	Description	Origin	Used in Sub-goal #
Summary	Post account summary	Screen	2,7
getBalance	Gets the Balance	Screen	4,6

3. Designing the Solution

3.1 Structure Chart

First Level Decomposition



The first level decomposition shows the broad outline of the whole program. The steps are to get input needed for a basic account, post the start-up account summary, request debit and credit information and display the balance after each transaction by adding and subtracting the debit and credit, then post an account summary.

Goal Refinement

Sub-goal 1

Get account details from the user to setup the account.

Sub-goal 2

Display the startup account.

Sub-goal 3

Request a withdraw amount

Sub-goal 3.1

Check if the withdraw amount is greater than the balance

Sub-goal 3.2

Deduct the amount if the withdraw amount is less than or equal to the balance.

Sub-goal 4

Post the balance

Sub-goal 5

Request a deposit amount

Sub-goal 5.1

Add the amount to the balance

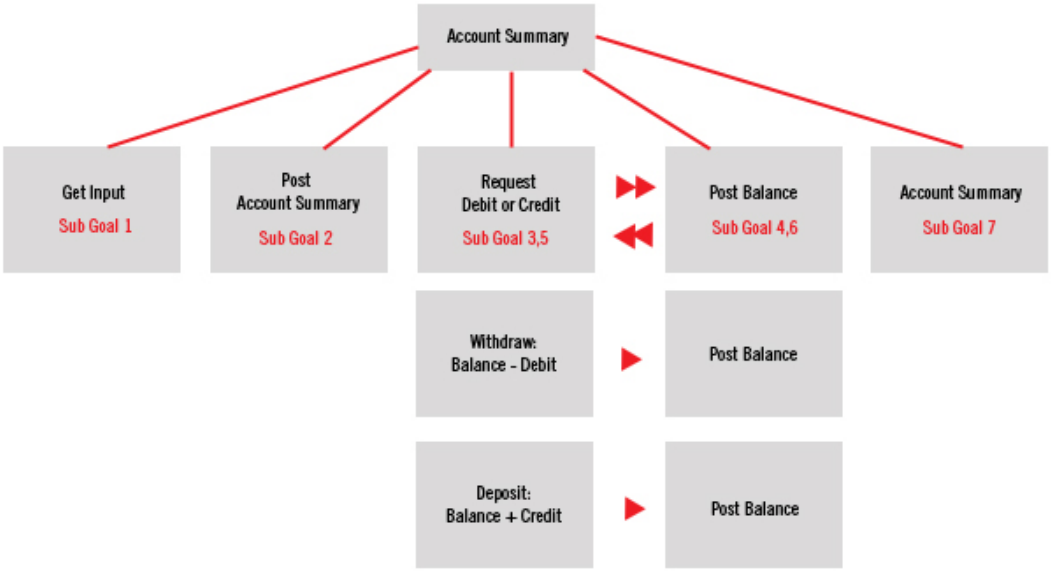
Sub-goal 6

Post the balance

Sub-goal 7

Display new account summary.

Second Level Decomposition



The second level shows the withdraw amount being deducted from the balance and then posted, followed by the deposit being added to the balance and posted.

3.2 Module and Data Specifications

Class: ManageAccounts
Name: Main function requests input from the user and provides display of information such as account summary, balance, etc.
Variables in the main function:
Acct1: Holds all the account information for this account.
name: accepts name input and references it
acctNum: accepts the account number input and references it.
Balance: holds the current balance.

Class: Accounts
Name: withdraw – tests if the withdraw amount is greater than balance and deducts the amount from the balance if it's less than or equal to the balance.
Input: None
Output: None
Logic: If the withdraw amount is less than or equal to the balance, deduct the withdraw amount.

Name: deposit – adds an amount to the balance
Input: None
Output: None
Logic: Add credit to the current balance.

Name: summary – return the current summary
Input: None
Output: None
Logic: Collect all the information and format it into a string, then store it in the accountInfo variable.

Name: accountNum – return account number if it's requested.
Input: None
Output: None
Logic: get account number if it's requested.

Data:

Name	Type	Structure
Balance	Real number	Simple
Name	String	Simple
acctNum	Real number	Simple

3.3 Algorithm

Logic

1.0 Request all required account information for a new account. (Name, Balance, Account Number)

2.0 Create a new account with all the input information and display the current account summary.

3.0 Request a withdraw amount from the user, if any.
- 3.1: Check to see if the withdraw amount is greater than the balance.
- 3.2: Deduct the amount from the balance.
- 3.3: Display the new balance to the user.

4.0 Request a deposit amount from the user, if any.
- 4.1: Add the deposit to the balance.
- 5.1: Display the new balance to the user.

5.0 Display the final account summary to the user.

Algorithm Description

There isn't much of an algorithm to this program. The program starts by requesting all the necessary information required to start a new account, such as the name on the account, the initial balance, and an account number. The new account that is created is an instantiation of the Account class. You then use the *summary* method in the Account class to show the user all the information about the new account. After that, the user is asked for a withdraw amount and the *withdraw* method is used on the new account. The withdraw *method* tests to see if the withdraw amount is more than what is in the account and returns a message if it's too much. If it isn't, the amount is deducted from the balance, and the new balance is returned. The program asks if the user has an amount to deposit. The *deposit* method is used to add the amount to balance. The new balance is then returned. Finally, the *summary* method is used to display the final account information and the summary is then output to the screen.

4. Translation

4.1 Source Code

```
//*******************************************************
// Account.java
//
// A bank account class with methods to deposit to, withdraw from,
// change the name on, charge a fee to, and print a summary of the account.
//*******************************************************
public class Account
{
private double balance;
private String name;
private long acctNum;
//-----------------------------------------
//Constructor -- initializes balance, owner, and account number
//-----------------------------------------
public Account(double initBal, String user, long number)
```

```
{
balance = initBal;
name = user;
acctNum = number;
}
//-----------------------------------------------
// Checks to see if balance is sufficient for withdrawal.
// If so, decrements balance by amount; if not, prints message.
//-----------------------------------------------
public void withdraw(double amount)
{
if (balance >= amount)
balance -= amount;
else
System.out.println("Insufficient funds");
}
//-----------------------------------------------
// Adds deposit amount to balance.
//-----------------------------------------------
public void deposit(double amount)
{
balance += amount;
}
//-----------------------------------------------
// Returns balance.
//-----------------------------------------------
public double getBalance()
{
return balance;
}
//-----------------------------------------------
// Returns a string containing the name, account number, and balance.
//-----------------------------------------------
public String summary()
{
String accountInfo = "Name: " + name + "\n" + "Account Number: " + acctNum +
"\n" + "Balance:
" + balance;
return accountInfo;
}
//-----------------------------------------------
// Gets Account Number
//-----------------------------------------------
public long accountNum()
{
return acctNum;
}
}
// *************************************************************
// ManageAccounts.java
// Author: Jennifer Soh ID: JS542
// Compiler Used: JGrasp
// Create and manage a bank account. Uses the Account class.
// *************************************************************
import java.util.Scanner;
public class ManageAccounts
{
public static void main(String[] args)
{
Scanner scan = new Scanner(System.in);
Account acct1;
System.out.println("Please enter your first and last name: ");
String name = scan.nextLine();
```

```
System.out.println("Please enter a 6 digit account number: ");
long acctNum = scan.nextLong();
System.out.println("Please enter your initial balance: ");
double initBalance = scan.nextDouble();
//create account1
acct1 = new Account(initBalance, name, acctNum);
//print summary for account
System.out.println("Your Current Account Summary: ");
System.out.println(acct1.summary());
// Make a withdraw
System.out.println("Amount to Withdraw from Your Account (Enter 0 if none): ");
double debit = scan.nextDouble();
acct1.withdraw(debit);
//print new balance using getBalance())
System.out.printf("New Account Balance after Deposit: $%.2f \n", +
acct1.getBalance());
// Make a deposit
System.out.println("Amount to deposit to your account (enter 0 if none): ");
double credit = scan.nextDouble();
acct1.deposit(credit);
//print new balance using getBalance())
System.out.printf("New Account Balance after Deposit: $%.2f \n",
acct1.getBalance());
//Retrieve account number
System.out.println("Retrieve your account number: " + acct1.accountNum());
//print summary for account
System.out.println("Your Current Account Summary: ");
System.out.println(acct1.summary());
}
}
```

4.2 Program and Module Description

withdraw

This method is used to check if the requested withdraw amount is greater than the balance. If the amount is greater, a rejection message is returned. If less than or equal to the balance, the amount is deducted from the balance and stored back in *balance*.

deposit

This method is used to add a deposit amount to the account balance. The value is then stored in *balance.*

getBalance

This method is used to get the current balance. This returns *balance*.

summary

This method returns the accountInfo variable holding a string with the account information (name, account number, and balance).

accountNum

Returns the account number if requested independently from summary.

Main

The main function first creates a scanner to accept input from the user, and declares a new Account variable for the first account. String outputs request name, account number, and initial balance information followed by input prompts after each request. A new Account object is created using the input collected from the user in the name, accountNum, and initBalance variables. The account summary is then output calling the summary method. A message requesting a withdraw amount is then output, followed by a prompt that accepts input and stores it in debit variable. The new account balance is retrieved using the getBalance method and is then output to the screen. A deposit amount is requested followed by a prompt that accepts deposit information from the user and assigns it to the credit variable. The *deposit* method is then used to add the amount to the account balance, which is then output after using the *getBalance* method. The accountNum method is tested to show that it's working, and then output. Finally, the main function uses the *summary* method is used to display the final account summary after all transactions.

5. Solution Testing

Test the program with following data domain:
Test string input for name but entering any string value.
I used my name.

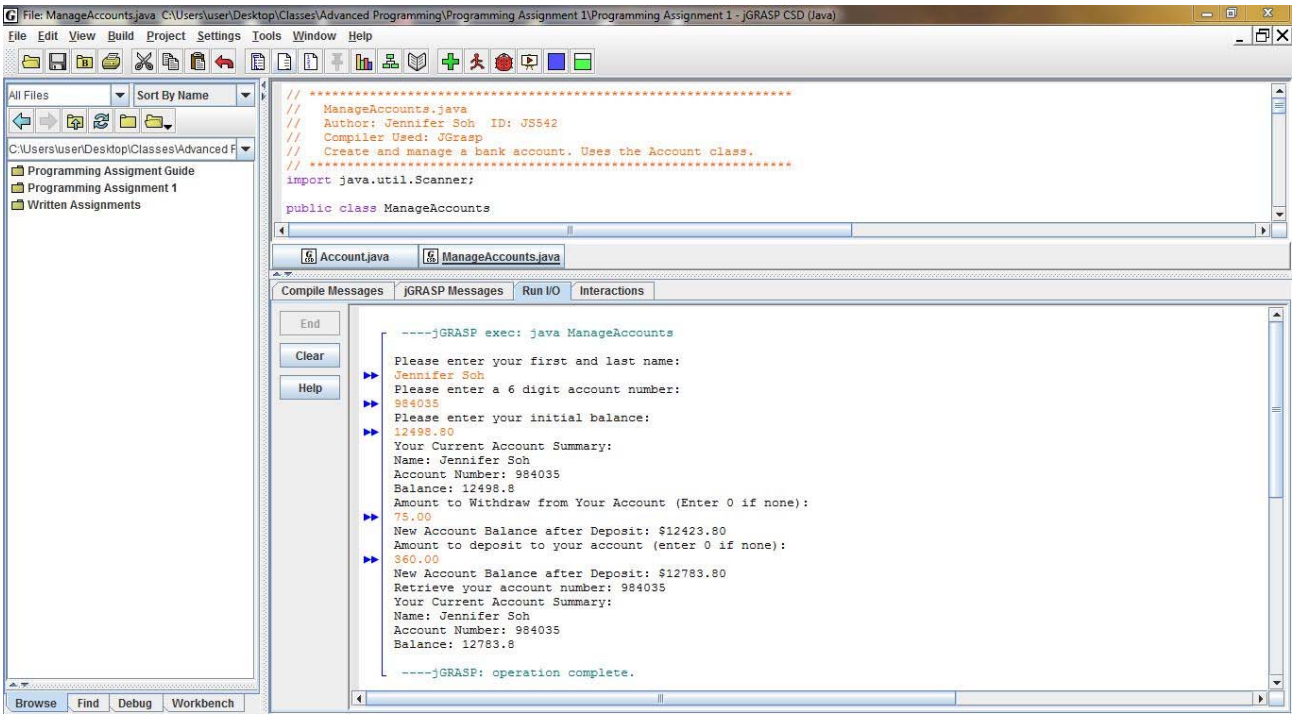
Test the program with following data:
Input a numeric amount for the account number. Input a numeric amount for the initial balance.
Input Account Number: 984035
Input Balance Amount: 12498.80
Withdraw: 75.00
Deposit: 360.00

6. Testing: Output

```
Please enter your first and last name:
Jennifer Soh
Please enter a 6 digit account number:
984035
Please enter your initial balance:
12498.80
Your Current Account Summary:
Name: Jennifer Soh
Account Number: 984035
Balance: 12498.8
Amount to Withdraw from Your Account (Enter 0 if none):
75.00
New Account Balance after Deposit: $12423.80
Amount to deposit to your account (enter 0 if none):
360.00
```

```
New Account Balance after Deposit: $12783.80
Retrieve your account number: 984035
Your Current Account Summary:
Name: Jennifer Soh
Account Number: 984035
Balance: 12783.8
```

7.1 Testing: Output Image



```
//*******************************************************
// Account.java
//
// A bank account class with methods to deposit to, withdraw from,
// change the name on, charge a fee to, and print a summary of the account.
//*******************************************************

public class Account
{
  private double balance;
  private String name;
  private long acctNum;

  //-----------------------------------------------
  //Constructor -- initializes balance, owner, and account number
  //-----------------------------------------------
  public Account(double initBal, String user, long number)
  {
    balance = initBal;
    name = user;
    acctNum = number;
  }

  //-----------------------------------------------
  // Checks to see if balance is sufficient for withdrawal.
  // If so, decrements balance by amount; if not, prints message.
  //-----------------------------------------------
  public void withdraw(double amount)
  {
    if (balance >= amount)
       balance -= amount;
    else
       System.out.println("Insufficient funds");
  }

  //-----------------------------------------------
  // Adds deposit amount to balance.
  //-----------------------------------------------
  public void deposit(double amount)
  {
    balance += amount;
  }

  //-----------------------------------------------
  // Returns balance.
  //-----------------------------------------------
  public double getBalance()
  {
    return balance;
  }


  //-----------------------------------------------
  // Returns a string containing the name, account number, and balance.
  //-----------------------------------------------
  public String summary()
  {
   String accountInfo = "Name: " + name + "\n" + "Account Number: " + acctNum + "\n" + "Balance:
" + balance;
   return accountInfo;
  }

  //-----------------------------------------------
  // Gets Account Number
  //-----------------------------------------------
  public long accountNum()

  {
```

```
   return acctNum;
  }

}
```

```
// ****************************************************************
//   ManageAccounts.java
//   Author: Jennifer Soh  ID: JS542
//   Compiler Used: JGrasp
//   Create and manage a bank account. Uses the Account class.
// ****************************************************************
import java.util.Scanner;

public class ManageAccounts
{
    public static void main(String[] args)
    {
      Scanner scan = new Scanner(System.in);
      Account acct1;

      System.out.println("Please enter your first and last name: ");
      String name = scan.nextLine();

      System.out.println("Please enter a 6 digit account number: ");
      long acctNum = scan.nextLong();

      System.out.println("Please enter your initial balance: ");
      double initBalance = scan.nextDouble();

      //create account1
      acct1 = new Account(initBalance, name, acctNum);

      //print summary for account
      System.out.println("Your Current Account Summary: ");
      System.out.println(acct1.summary());

      // Make a withdraw
      System.out.println("Amount to Withdraw from Your Account (Enter 0 if none): ");
      double debit = scan.nextDouble();
      acct1.withdraw(debit);

      //print new balance using getBalance())
      System.out.printf("New Account Balance after Deposit: $%.2f \n", + acct1.getBalance());

      // Make a deposit
      System.out.println("Amount to deposit to your account (enter 0 if none): ");
      double credit = scan.nextDouble();
      acct1.deposit(credit);

      //print new balance using getBalance())
      System.out.printf("New Account Balance after Deposit: $%.2f \n", acct1.getBalance());

      //Retrieve account number
      System.out.println("Retrieve your account number: " + acct1.accountNum());

      //print summary for account
      System.out.println("Your Current Account Summary: ");
      System.out.println(acct1.summary());

    }
}
```

```
 ----jGRASP exec: java ManageAccounts

Please enter your first and last name:
Jennifer Soh
Please enter a 6 digit account number:
984035
Please enter your initial balance:
12498.80
Your Current Account Summary:
Name: Jennifer Soh
Account Number: 984035
Balance: 12498.8
Amount to Withdraw from Your Account (Enter 0 if none):
75.00
New Account Balance after Deposit: $12423.80
Amount to deposit to your account (enter 0 if none):
360.00
New Account Balance after Deposit: $12783.80
Retrieve your account number: 984035
Your Current Account Summary:
Name: Jennifer Soh
Account Number: 984035
Balance: 12783.8

 ----jGRASP: operation complete.
```