Rahul Nagaraju | **A20543969 |** [rnagaraju@hawk.iit.edu](mailto:rnagaraju@hawk.iit.edu)

**Lab1**

**Object-oriented App Development ( ITMD-510 )**

**PROJECT Bank account simulator program 100 points**

Objective: To write a program that performs various bank transactions.

***PROJECT DESCRIPTION***

Bank of IIT has contacted you to write, compile and execute a complete program that creates bank account information and executes various transaction details for their clients.

Your program will prompt users for options such as creating an initial balance, entering deposits or withdrawals. Also, your program will allow for the printing of account information including interest at various interest rates.

Use loops, user defined methods, conditional and relational logic and the basics of OOP to accomplish the objectives of this program.

Error trapping will be part of your grade so don’t forget to include some basic error trapping logic! Comment your code thoroughly as well for maximum points.

Source Code:

**AccountHolder.java**

/\*

--------------------------------------------------------------------

- Author Rahul Nagaraju

- Assignment: Lab1

- FileName: AccountHolder.java

- Course: ITMD-510 Object-Oriented App Development

- Instructor: James Papademas

----------------------------------------------------------------------

\*/

// Import necessary libraries

import java.text.DecimalFormat;

// Creating AccountHolder Class

public class AccountHolder{

    // Declaration of variables

    private double annualInterestRate=(4.0/100.0);

    private double balance;

    // Constructor to initiate balance

    public  AccountHolder(double balance){

        try{

        if(balance<0){

            System.out.println("Balance cannot be initiated to a negative number");

        }

        else{

            this.balance=balance;

        }

        }catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

    // Method to deposit the amount and update the balance in the account

    public void deposit(double amount){

        balance+=amount;

    }

    // Method to withdraw the amount and update the balance in the account

    public void withdrawal(double amount){

        // Making sure the balance is 50 before allowing for withdrawal

        try{

        if(balance-amount<50){

            throw new Exception ("You can't withdraw as your minimum balance should be 50");

        }else{

            balance-=amount;

        }

        }catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

    // Method to calculate the interest and update the balance in the account

    public void monthlyInterest(){

        balance += balance \* (annualInterestRate / 12.0);

    }

    // Getter method to fetch and return the balance

    public double getBalance(){

        return balance;

    }

    // Method to print balances for each month for one year

    public void DisplayBalanceForYear(){

        System.out.println("");

        System.out.println("Monthly balances for one year at 0.04 interest is as follows:");

        System.out.println("Account Balance with Interest:");

        // Variable to format and round off balance to 2 decimal places

        DecimalFormat deci = new DecimalFormat("##.00");

        for(int i=1;i<=12;i++){

            monthlyInterest();

            System.out.println("Month "+i+":\t$"+deci.format(balance));

        }

        System.out.println("");

    }

}

**AccountHolderTest.java**

/\*

--------------------------------------------------------------------

- Author Rahul Nagaraju

- Assignment: Lab1

- FileName: AccountHolderTest.java

- Course: ITMD-510 Object-Oriented App Development

- Instructor: James Papademas

- The program in this file will

1. Ask for the account holder's name

2. Request the account holder for their initial balance.

3. Deposit the amount requested by the account holder.

4. Support the account holder in maintaining a minimum balance of $50.

5. Help the account holder with withdrawals.

6. Calculate the accumulated interest at the end of the month.

7. Extra credit - Calculate the accumulated interest for one year (month by month).

----------------------------------------------------------------------

\*/

// Import all necessary libraries

// Date and Scanner

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Scanner;

import java.util.concurrent.ThreadPoolExecutor.DiscardOldestPolicy;

// Creating AccountHolderTest Class

public class AccountHolderTest{

    // Creating variable to hold Account Holder Name

    private static String NameOfAccountHolder;

    // Main function

    public static void main(String[] args)    {

        // Scanner object named sc to get necessary inputs from the end user

        Scanner sc=new Scanner(System.in);

        // To greet the user and ask name as an input to proceed

        System.out.println("We welcome you to the Bank of IIT!!");

        System.out.println("We kindly request you to enter your name:");

        NameOfAccountHolder = sc.nextLine();

        // Request user to enter the initial balance amount

        System.out.println("Hi "+NameOfAccountHolder+", Please enter your initial balance amount:");

        // Creating AccountHolder object by calling the constructor in the AccountHolder

        AccountHolder Account1=new AccountHolder(sc.nextDouble());

        // Request user to enter the deposit amount

        System.out.println(NameOfAccountHolder+", Please enter a deposit amount:");

        Account1.deposit(sc.nextDouble());

        // Request user to enter the withdrawal amount

        System.out.println(NameOfAccountHolder+", Please enter a withdrawal amount:");

        Account1.withdrawal(sc.nextDouble());

        // Closing the Scanner Object

        sc.close();

        // Update the balance to include interest by calling the method monthly interest

        Account1.monthlyInterest();

        // Fetch the updated balance and store in the balance variable

        double balance=Account1.getBalance();

        // Variable to format and round off balance to 2 decimal places

        DecimalFormat deci = new DecimalFormat("##.00");

        System.out.println("Balance with interest applied = "+deci.format(balance));

        // Display balances for each month for the whole year

        Account1.DisplayBalanceForYear();

        // Greetings after transactions

        System.out.println("Thanks for visiting us " + NameOfAccountHolder+"!!");

        // Display current timestamp and name of the owner who wrote the code

        String timeStamp = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.getInstance().getTime());

        System.out.println("\nCur dt=" + timeStamp + "\nProgrammed by Rahul Nagaraju\n");

    }

}

**AccountHolder.java [Extra credit]**

    // Method to print balances for each month for one year

    public void DisplayBalanceForYear(){

        System.out.println("");

        System.out.println("Monthly balances for one year at 0.04 interest is as follows:");

        System.out.println("Account Balance with Interest:");

        // Variable to format and round off balance to 2 decimal places

        DecimalFormat deci = new DecimalFormat("##.00");

        for(int i=1;i<=12;i++){

            monthlyInterest();

            System.out.println("Month "+i+":\t$"+deci.format(balance));

        }

        System.out.println("");

    }

**Snapshot 1: - Balance is above 50 after withdrawal**

**A computer screen with blue text

Description automatically generated**

**Snapshot 2: Balance is below 50 after withdrawal (Hence, withdrawal not accepted)**

**A computer screen with blue text

Description automatically generated**

**Snapshot 3: [Extra credit] – Display balances for each month after calculated interest**

**A screen shot of a computer

Description automatically generated**