**Assignment No 1**

**Name: Vighnesh Sanjay Sawant**

**CWID: A20399823**

**Date: 09/16/2017**

**Title: Bank Account Simulator Program.**

**Lab No: 01**

**Source code:**

**Important note: The following program is not implemented by switch case so the implementation is sequential. It is required to run the program again for newer trials.**

AccountHolder.java

**import** java.util.Scanner; //importing scanner class from util package for tokenizing or taking inputs from user

**public** **class** AccountHolder { // Name of the class

**private** **double** annualInterestRate; //Declaration of instance variables

**int** month;

**private** **double** intbal;

Scanner sc = **new** Scanner(System.***in***); // Scanner class for taking inputs from user regarding deposit, withdrawal

AccountHolder() //declaration of a default constructor

{

**if**(intbal < 0.0) //condition given for checking whether the user has entered balance less than 0.0

**throw** **new** IllegalArgumentException("balance must be non negative"); //throwing exception of an illegal argument

}

**double** totbal; //declaration of instance variable

//declaration of instance methods begin

**public** **double** initialbalance(**double** intbal) //declaration of method named as initial balance taking initial balance as a parameter

{

**if**(intbal < 0.0) //condition given for checking whether the user has entered balance less than 0.0

**throw** **new** IllegalArgumentException("balance must be non negative"); //throwing exception of an illegal argument

**else**

{

**this**.intbal = intbal; // self reference by using this keyword

}

totbal = totbal + intbal;

**return** totbal;

}

**public** **double** deposit(**double** bal) //declaration of a method named as deposit taking balance as a parameter to calculate the total balance after depositing some amount

{

totbal = totbal + bal; // balance after depositing certain amount

**return** totbal;

}

**public** **double** withdrawal(**double** intbal) //declaration of method named as withdrawal taking balance as a parameter to calculate the total balance after withdrawing some amount

{

totbal = totbal - intbal; // balance after withdrawing certain amount

**return** totbal;

}

**public** **void** monthlyinterest(**double** annualInterest) //declaration of method named as monthly interest taking annual interest as a parameter to calculate monthly interest on the total balance

{

**if**(totbal > 500.0 ) {

**for** (**int** i = 1; i<= 12; i++) { // for loop which calculate monthly interest for 12 months of a year for total balance greater than $500

**double** interest = totbal \* 4/1200; // calculating interest according to 4% per annum

totbal = totbal + interest; // adding the calculated interest to the total balance

System.***out***.printf("Month %d is $%6.2f%n", i , totbal); // displaying the annual interest over the period of 12 months in a columnar format upto 2 digits of precision

month = i;

}

}**else** {

**double** totbal1 = totbal - 50.0;

**for** (**int** i = 1; i<= 12; i++) { // for loop which calculate monthly interest for 12 months of a year for total balance less than $500

**double** interest = totbal \* 4/1200; // calculating interest according to 4% per annum

totbal1 = totbal1 + interest; // adding the calculated interest to the total balance

System.***out***.printf("Month %d is $%6.2f%n", i , totbal1); // displaying the annual interest over the period of 12 months in a columnar format upto 2 digits of precision

month = i;

}

}

annualInterestRate = 5; // setting annual interest rate at 5%

**double** interest = totbal \* annualInterestRate/1200; // after completing 12 months of a year,for the next year the annual interest rate goes at 5%

totbal = totbal + interest; // adding the calculated interest at 5% to the total balance

System.***out***.printf("After 12 months as per the updated interest rate of 0.05, the amount is $ %6.2f %n", totbal); // displaying the annual interest at 5%

}

**public** **void** modifymonthlyinterest(**double** New\_Interest\_Rate) // declaration of a method to update the interest

{

**if** (New\_Interest\_Rate < 0 || New\_Interest\_Rate > 100)

System.***out***.println("specify interest rate between 0 and 100");

**else**

{

annualInterestRate = New\_Interest\_Rate;

}

}

**public** String toString() // The toString method of String class by using format function is used to display output in a stated format

{

**return** String.*format*("Account balance $%.2f",intbal); // account balance will be displayed upto 2 digits of precision

}

**public** **double** getBalance()

{

**return** intbal;

}

**public** **double** getInitialInterest()

{

**return** annualInterestRate;

}

}

AccountHolderTest.java

**import** java.util.Scanner; // importing scanner class from utility package to take inputs from user

**import** java.text.SimpleDateFormat; // importing text package and using SimpleDateFormat as a class member to add date and time

**import** java.util.Calendar; // importing calender class from utility package to use and record specific instance of time

**public** **class** AccountHolderTest { // A test class which will link or call all the methods from the AccountHolder class

**public** **static** **void** main(String[] args) { // A main method

// **TODO** Auto-generated method stub\

{

**try** //start of try block

{

AccountHolder ach = **new** AccountHolder(); // creating new object and reference of that object of AccountHolder class

Scanner sc=**new** Scanner(System.***in***); // scanner class for taking inputs from user

**double** intbal;

System.***out***.println("Enter the initial balance : $");

intbal = sc.nextDouble(); // taking the initial balance from the user

**double** Total\_Initial\_Balance = ach.initialbalance(intbal); // // calling the initial balance method

System.***out***.println("Initial balance: $"+Total\_Initial\_Balance); // displaying the initial balance

{

**double** Amount\_Deposited; // declaring variable

System.***out***.println("Enter amount to be deposited :");

Amount\_Deposited = sc.nextDouble(); // taking the amount to be deposited into the balance as an input from the user

**double** Total\_Balance = ach.deposit(Amount\_Deposited); // calling the deposit method

System.***out***.println(String.*valueOf*("Final balance after depositing $"+Amount\_Deposited +" is = $" +Total\_Balance)); // displaying the final balance after depositing some amount

**double** Amount\_Withdrawn; // declaring variable

System.***out***.println("Enter amount to be withdrawn = ");

Amount\_Withdrawn=sc.nextDouble(); // taking the amount to be withdrawn from the balance as a input from the user

**double** Amount\_After\_Withdraw = ach.withdrawal(Amount\_Withdrawn); // calling the withdrawal method

**if** (Amount\_After\_Withdraw < 100.0) // checking whether the balance after withdrawal is below $100 or not

{

System.***out***.println("Alert! Since you are low on balance(less than $100 after withdrawing) you are prevented from doing any further withdrawals. Thankyou "); // if the balance after withdrawal is below 100 then informing the user about low funds

System.*exit*(0); // if the user is low on funds then not permitting him to withdraw further

}

**else** **if**(Amount\_After\_Withdraw < 500.0) // checking if the balance after withdraw is less than 500

{

Amount\_After\_Withdraw = Amount\_After\_Withdraw - 50.0; // $50 will be deducted from the balance if the balance is less than $500

System.***out***.println("$50 fee is deducted and your current balance is = $"+Amount\_After\_Withdraw); // informing the user about deducting $50 from its balance

};

System.***out***.println(String.*valueOf*("Final balance after withdrawinng $" +Amount\_Withdrawn +" is $" +Amount\_After\_Withdraw)); // displaying final balance after withdrawing some amount

System.***out***.println("To see monthly balance for one year at 4%"); // prompting the user that the system is about to show the monthly interest

System.***out***.println("base = $" +Amount\_After\_Withdraw);

ach.monthlyinterest(4.0); // calling the monthly interest method to calculate interest for each month at 4%

**double** New\_Interest\_Rate = 5; // setting the new rate of interest as 5% after 12 months

ach.modifymonthlyinterest(New\_Interest\_Rate); // calling the method and passing the new rate of interest

sc.close(); // close method of scanner class invoked to no longer take input from user

}

} // end of try block

// begin of multiple catch block

**catch**(NumberFormatException e) //to check whether the number is in the correct format or not

{

System.***out***.println("Wrong data");

}

**catch**(ArithmeticException e) // to check whether arithmetic operations are correct or not

{

System.***out***.println("invalid Operation");

}

**catch**(IllegalArgumentException | NullPointerException e) // multi-catch block and to check whether the argument is illegal or not and check for null value assign to reference of an object

{

System.***out***.println("incorrect input, balance must be non negative");

}

**catch**(Exception e) // calling superclass of all exceptions to check for any other random errors

{

System.***out***.println("some other error");

}

**finally** // the above catch blocks may or may not run but finally will always run

{

System.***out***.println("Program Execution ended!");

}

String timeStamp = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.*getInstance*().getTime()); // using the SimpleDateFormat class and calender class to get the specified instance of time and to display it in yyyy/MM/dd HH:MM:SS by using format method

System.***out***.println("Cur dt=" + timeStamp + "\nProgrammed by Vighnesh Sanjay Sawant\n"); // displaying the person who programmed the code

System.*exit*(0);// used to exit the program.

}

}

}

**Junit Testing:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class JunitTest {

AccountHolder acc = new AccountHolder();

@Test

public void testBalance()

{

assertTrue(acc.getBalance() >= 0.0);

}

@Test

public void testInitialInterest()

{

assertTrue(acc.getInitialInterest() >= 0.0);

}

@Test

public void testupdatedinterest(){

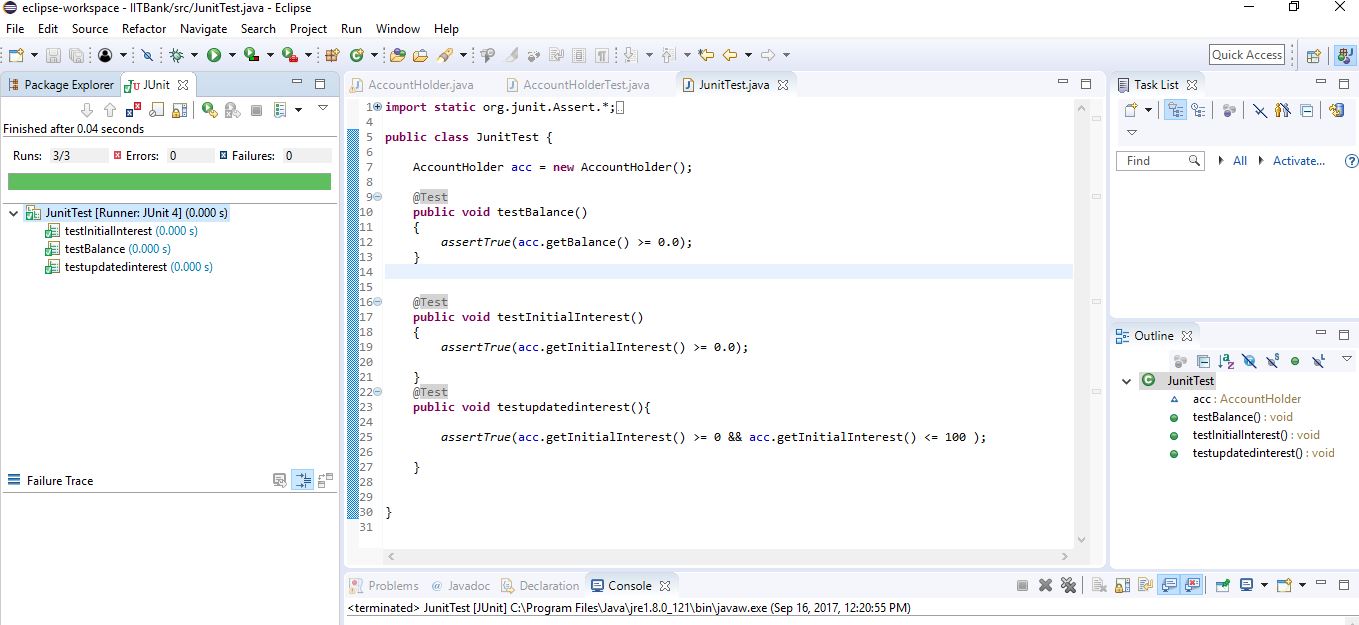
assertTrue(acc.getInitialInterest() >= 0 && acc.getInitialInterest() <= 100 );

}

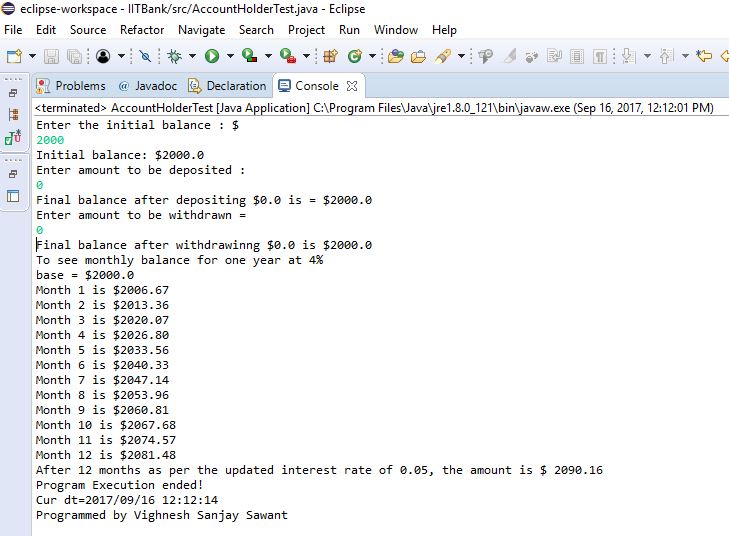
}

**The following are the snapshots of output:**

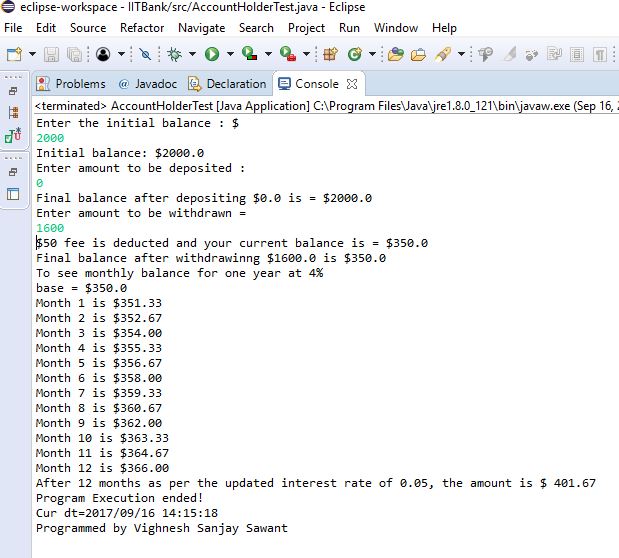
**Junit Testing:**



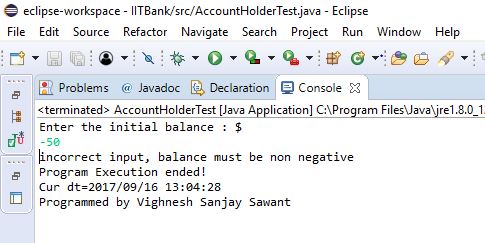
**1st execution for initial balance $2000, depositing and withdrawing $0:**



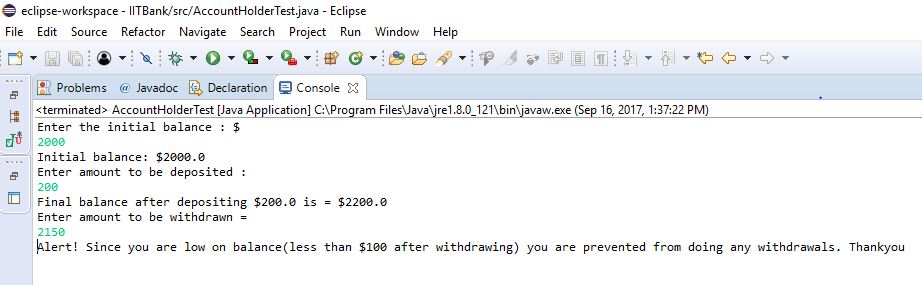
**2nd execution where balance after withdrawing goes below $500**



**3rd execution where entered initial balance is negative:**



**4th execution where you try to keep balance below $100:**



**All the classes are in the same package:**

