

**Advanced Java Programming**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Lab Report: 03**

**Submitted by:**

Name: Aashish Pokharel

Faculty: Bsc.CSIT,7th sem.

Roll no.: 20786/075

**Submitted to:**

Krishna Pandey

Department of CSIT , OIC

**Question**

1. Write classes to hold Account, SB-Account and Current-Account details. [Here implement

the concept of inheritance.]

The common properties of the account are Account number, name and amount.

Specifics of SB account is 4% interest to be paid per month.

ð Implement the run-time polymorphism by creating base class variable and derived class

object.

ð Ask the user for which type of account to be created then create the corresponding

account (Note: Use scanner class).

ð Implement function overriding by having deposit and withdraw functions and perform the

required action accordingly.

Ensure base class can’t be instantiated. (Note: Abstract keyword can be used).

2. Define the minimum balance for the both the type of accounts. Use final keyword to

create constants.

Ensure sb account class and current account class will cannot be used as base classes (Note:

You can use final classes).

**Implementation:**

package classwork; // Uses classwork Package

import java.util.Scanner; // for reading the input from user

public class Module4{

/\* Main Public function for module 4 \*/

public static void main(String[] args) {

//Implementing runtime polymorphism

int accType; // for taking input from the user

Account a; // Account type object for

Scanner s = new Scanner(System.in);

System.out.println("Enter the type of account:\n1. SB \n2. Current\n");

accType = s.nextInt();

if(accType == 1) {

a = new SBAccount();

}else {

a = new CurrentAccount();

}

a.deposit(20000.0); // deposit called

a.withdraw(3000); // withdraw called

s.close(); // closes Scanner

}

}

abstract class Account{

/\*

\* Abstract Class for the Account Handling

\* \*/

double bankBalance; // Variable to hold the balance in the bank

int accountNo; // Account No of the user

String Name; // Name of the user

final double minimumBalance = 2000.0; // Minimum balance - CONSTANT

public abstract void withdraw(double amount); // abstract method for handling withdraw

public abstract void deposit(double amount); // abstract method for handling deposit

}

final class SBAccount extends Account{

/\* Class to handle SB Accounts \*/

final double interest = 0.04;

@Override

public void withdraw(double amount) {

/\* A function that handles the withdrawing process

\* Parameters :

\* <amount> : Amount to be withdrawn

\* Returns:

\* <Void>

\*

\* \*/

//check if the money is available

if(this.bankBalance > amount + 2000.0) {

this.bankBalance -=amount;

System.out.println("\nWithdraw successful");

System.out.println("\nAmount Withdrawn: "+ amount);

System.out.println("\nRemaining Balance: "+this.bankBalance);

}else {

System.out.println("\nWithdraw unsuccessful! System shows not enough balance.");

}

}

@Override

public void deposit(double amount) {

/\* A function that handles the depositing process

\* Parameters :

\* <amount> : Amount to be deposited

\*

\* Returns:

\* <Void>

\* \*/

// TODO Auto-generated method stub

this.bankBalance +=amount;

System.out.println("\nDeposit successful.");

System.out.println("\nAmount Deposited: "+ amount);

System.out.println("\nRemaining Balance: "+this.bankBalance);

}

public void addInterest() {

/\* A function that handles the depositing process

\* Parameters :

\* <Void>

\*

\* Returns:

\* <Void>

\* \*/

this.bankBalance += this.bankBalance \* this.interest /12;

System.out.println("Interest Added");

}

}

final class CurrentAccount extends Account{

/\* Class to handle Current Accounts \*/

final double interest = 0.04;

@Override

public void withdraw(double amount) {

/\* A function that handles the withdrawing process

\* Parameters :

\* <amount> : Amount to be withdrawn

\* Returns:

\* <Void>

\* \*/

//check if the money is available

if(this.bankBalance > amount + 2000.0) {

this.bankBalance -=amount;

System.out.println("\nWithdraw successful");

System.out.println("\nAmount Withdrawn: "+ amount);

System.out.println("\nRemaining Balance: "+this.bankBalance);

}else{

System.out.println("Withdraw unsuccessful! System shows not enough balance.");

}

}

@Override

public void deposit(double amount) {

/\* A function that handles the depositing process

\* Parameters :

\* <amount> : Amount to be deposited

\*

\* Returns:

\* <Void>

\*

\*/

this.bankBalance +=amount;

System.out.println("\nDeposit successful");

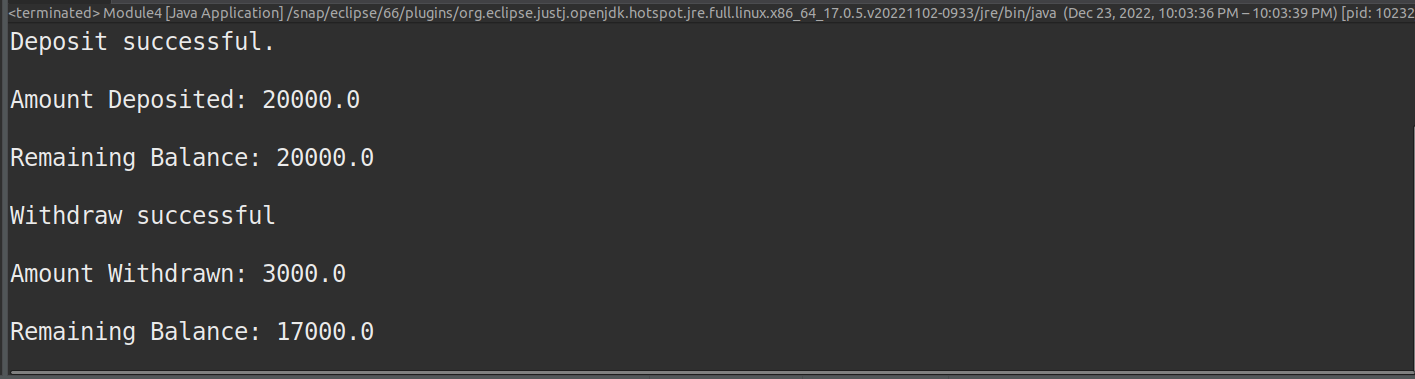
System.out.println("\nAmount Deposited: "+ amount);

System.out.println("\nRemaining Balance: "+this.bankBalance);

}

}

**Output:**

****