Enter 1 to open a new Checking account

Enter 2 to open a new savings account

Enter 3 to make a transaction

q. Exit

3

a. Deposit in Checking

b. Withdraw in Checking

c. Deposit in Savings

d. Withdraw in Savings

a

Account Number

345

Amount in transaction

2345

Invalid option

Enter 1 to open a new Checking account

Enter 2 to open a new savings account

Enter 3 to make a transaction

q. Exit

1

Enter the opening Checking balance :

2345

Enter 1 to open a new Checking account

Enter 2 to open a new savings account

Enter 3 to make a transaction

q. Exit

1

Enter the opening Checking balance :

3456

**import** java.util.Arrays;

**import** java.util.Date;

**import** java.util.Scanner;

**import** java.util.Date;

**public** **class** Account {

Customer customer;

**int** accountNumber;

**private** **double** balance;

Transaction transaction[] = **new** Transaction[20];

Date date = **new** Date();

Scanner sc = **new** Scanner(System.***in***);

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

**return** balance;

}

**public** **void** setBalance(**double** balance) {

**this**.balance = balance;

}

**public** Customer getCustomer() {

**return** customer;

}

**public** **void** setCustomer() {

**this**.customer = customer;

}

**public** String toString() {

**return** "Account Number " + accountNumber + " Current balance is " + balance + "\nLast date of update: " + date;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** **void** reAllocate(Transaction transaction[]) {

**int** min = 0;

**int** prior = transaction.length;

**if** (min > prior) {

Object oldData[] = transaction;

**int** now = (prior \* 2);

**if** (now < min)

now = min;

transaction = Arrays.*copyOf*(transaction, now);

}

}

}

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** Customer {

**private** String name;

**private** String address;

**private** **int** age;

**private** **int** telephoneNumber;

**private** **int** customerNumber;

**private** **double** checking\_interest, savings\_interest, check\_charge;

Iterator<Account> itr;

**private** ArrayList<Account> accountList;

**public** String getName() {

**return** name;

}

**public** String setName() {

**return** **this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **int** getAge() {

**return** age;

}

**public** **int** setAge() {

**return** **this**.age = age;

}

**public** **int** getTelephoneNumber() {

**return** telephoneNumber;

}

**public** **int** setTelphoneNumber() {

**return** **this**.telephoneNumber = telephoneNumber;

}

**public** **int** getCustomerNumber() {

**return** customerNumber;

}

**public** **int** setCustomerNumber() {

**return** **this**.customerNumber = customerNumber;

}

**public** **double** getSavingsInterest() {

**return** savings\_interest;

}

**public** **double** getCheckinInterest() {

**return** checking\_interest;

}

**public** **double** getCheckCharge() {

**return** check\_charge;

}

}

**public** **class** SavingsAccount **extends** Account {

**static** **int** *amount*;

Account balance = **new** Account();

**public** **static** **boolean** deposit(Account balance){

System.***out***.println("Balance before deposit ");

System.***out***.println(balance.toString());

balance.setBalance(balance.getBalance() + *amount*);

System.***out***.println("Balance after deposit " + balance.toString());

balance.setBalance(balance.getBalance() + *amount*);

**return** **true**;

}

**public** **static** **boolean** withdraw(Account balance) {

**if** (balance.getBalance() < *amount*) {

System.***out***.println("Insufficient Balance.");

**return** **false** ;

}

System.***out***.println("Balance in Savings Account before withdraw ");

System.***out***.println(balance.toString());

balance.setBalance(balance.getBalance() - *amount*);

System.***out***.println("Balance in Savings Account after withdraw " + balance.toString());

**return** **true**;

}

**public** **void** addInterest(**double** interest) {

System.***out***.println("Balance in Savings Account before interest ");

System.***out***.println(balance.toString());

//Savings account has 1.00% interest rate

interest = (**int**) (balance.getBalance() + *amount*\*(.01));

balance.setBalance(interest);

System.***out***.println("Balance in Savings Account after interest" + balance.toString());

}

}

**public** **class** CheckingAccount **extends** Account {

**static** **int** *amount*;

Account balance = **new** Account();

**public** **static** **boolean** deposit(Account balance) {

System.***out***.println("Balance before deposit ");

System.***out***.println(balance.toString());

balance.setBalance(balance.getBalance() + *amount*);

System.***out***.println("Balance after deposit " + balance.toString());

balance.setBalance(balance.getBalance() + *amount*);

**return** **true**;

}

**public** **static** **boolean** withdraw(Account balance) {

**if** (balance.getBalance() < *amount*) {

System.***out***.println("Insufficient Balance.");

**return** **false**;

}

System.***out***.println("Balance in Checking Account before withdraw ");

System.***out***.println(balance.toString());

balance.setBalance(balance.getBalance() - *amount*);

System.***out***.println("Balance in Checking Account after withdraw " + balance.toString());

**return** **true**;

}

**public** **void** addInterest(**double** interest) {

System.***out***.println("Balance in Savings Account before interest ");

System.***out***.println(balance.toString());

//Checkings Account has a 1.00% interest rate.

interest = (**int**) (balance.getBalance() + *amount*\*(.01));

balance.setBalance(interest);

System.***out***.println("Balance in Savings Account after interest" + balance.toString());

}

}

**import** java.util.\*;

**public** **class** Main {

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

String choice, ch, operation;

Transaction trans = **new** Transaction();

Scanner sc = **new** Scanner(System.***in***);

**double** amount;

**long** accountNo;

**do** {

System.***out***.println("Enter 1 to open a new Checking account");

System.***out***.println("Enter 2 to open a new savings account");

System.***out***.println("Enter 3 to make a transaction");

System.***out***.println("q. Exit");

choice = sc.next();

**switch** (choice) {

**case** "1":

**double** openingChecking;

System.***out***.println("Enter the opening Checking balance :");

openingChecking = sc.nextDouble();

trans.transaction(0, "Opening", openingChecking);

**break**;

**case** "2":

**double** openingSavings;

System.***out***.println("Enter the opening Checking balance :");

openingSavings = sc.nextDouble();

trans.transaction(0, "Opening", openingSavings);

**break**;

**case** "3":

System.***out***.println("a. Deposit in Checking");

System.***out***.println("b. Withdraw in Checking ");

System.***out***.println("c. Deposit in Savings ");

System.***out***.println("d. Withdraw in Savings ");

ch = sc.next();

**if** (ch.equalsIgnoreCase("a"))

operation = "deposit checking";

**else** **if** (ch.equalsIgnoreCase("b"))

operation = "withdraw checking";

**else** **if** (ch.equalsIgnoreCase("c"))

operation = "deposit savings";

**else** **if** (ch.equalsIgnoreCase("c"))

operation = "withdraw savings";

**else** {

operation = "Invalid option";

}

System.***out***.println("Account Number ");

accountNo = sc.nextLong();

System.***out***.println("Amount transaction ");

amount = sc.nextDouble();

trans.transaction(accountNo, operation, amount);

**break**;

**case** "q":

System.***out***.println("Thank you!");

**break**;

**default**:

System.***out***.println("Wrong choice!!");

}

} **while** (choice != "q");

sc.close();

}

}**import** java.util.ArrayList;

**import** java.util.Date;

**public** **class** Bank {

**private** ArrayList<Account> accountList;

Account balance[] = **new** Account[100];

**private** String transactionType;

**double** amount;

**private** **long** accountNum;

**private** Date date;

Account account = **new** Account();

**public** **void** addAccount(Account e) {

accountList.add(e);

}

**public** **void** Bank(**long** accountNum, String transactionType, **double** amount) {

**this**.accountNum = accountNum;

**this**.transactionType = transactionType;

**this**.amount = amount;

date = **new** Date();

}

**public** **void** makeDeposit() {

**if** (transactionType.equalsIgnoreCase("deposit checking")) {

System.***out***.println("Balance before deposit \n:");

System.***out***.println(account.toString());

account.setBalance(account.getBalance() + accountNum);

System.***out***.println("Balance after deposit \n" + account.toString());

CheckingAccount.*deposit*(account);

}

**if** (transactionType.equalsIgnoreCase("deposit savings")) {

System.***out***.println("Balance before deposit \n:");

System.***out***.println(account.toString());

account.setBalance(account.getBalance() + accountNum);

System.***out***.println("Balance after deposit \n" + account.toString());

SavingsAccount.*deposit*(account);

}

}

**public** **void** makeWithdrawl() {

**if** (account.getBalance() < amount) {

System.***out***.println("Insufficient Balance.");

**return**;

}

**if** (transactionType.equalsIgnoreCase("withdraw checking ")) {

System.***out***.println("Balance before withdrawl ");

System.***out***.println(account.toString());

account.setBalance(account.getBalance() - accountNum);

System.***out***.println("Balance after withdrawal \n" + account.toString());

CheckingAccount.*withdraw*(account);

}

**if** (transactionType.equalsIgnoreCase("withdraw savings ")) {

System.***out***.println("Balance before withdrawl ");

System.***out***.println(account.toString());

account.setBalance(account.getBalance() - accountNum);

System.***out***.println("Balance after withdrawal \n" + account.toString());

SavingsAccount.*withdraw*(account);

}

}

**public** **long** getAccountNum() {

**return** accountNum;

}

}