



AccountHolder.java

**package** lab1;

/\*\*

\* @author henryfbp

\* @date 09/15/17

\* @class ITMD411

\* @lab 01

\*

\* This program simulates a command-line banking application.

\*/

**public** **class** **AccountHolder**

{

**static** **double** annualInterestRate;

**double** balance;

/\*\*\*

\*

\* @param balance The initial balance for the account. Must be >= 0.

\*/

**public** **AccountHolder**(**double** balance)

{

**this**.balance = **0**;

**if**(balance >= **0**)

{

**this**.balance = balance;

}

**else**

{

System.out.println("Balance '" + balance +"' is negative and cannot be used! Using '0' instead.");

}

}

/\*\*\*

\*

\* @param toDeposit A double representing how much will be deposited into the bank account.

\*/

**public** **void** **deposit**(**double** toDeposit)

{

**this**.balance += toDeposit; //add toDeposit to current funds

}

/\*\*\*

\*

\* @param toWithdraw A double representing how much will be withdrawn from the bank account.

\*/

**public** **void** **withdrawal**(**double** toWithdraw)

{

**double** projectedBalance = **this**.balance - toWithdraw;

**if**(projectedBalance < **500.00** && projectedBalance >= **100**)

{//charge for being under 500 dollars

**this**.balance -= toWithdraw;

**this**.balance -= **50**; //transaction fee for under $500

**return**;

}

**else** **if**(projectedBalance < **100**)

{//don't withdraw, tell user

System.out.println("Attempted transation of '"+toWithdraw+"' not withdrawn, as it would cause balance to be '"+projectedBalance+"', which is under '$100'.");

**return**;

}

**else**

{//user is withdrawing money from an account that will have over $500 after it's done

**this**.balance -= toWithdraw;

}

}

/\*\*\*

\*

\* @param balance The balance to be used in the interest calculation.

\* This method updates this account's balance to be (balance + (balance \* annualInterestRate / 12.0)).

\*/

**public** **void** **monthlyInterest**(**double** balance)

{

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

}

**public** **static** **void** **modifyMonthlyInterest**(**double** rateUpdate)

{

**if**(rateUpdate >= **0** && rateUpdate <= **1.0**)

{

annualInterestRate = rateUpdate;

}

}

**@Override**

**public** String **toString**()

{

**return** String.format("$%.2f",**this**.balance); //returns a balance double formatted as "$2.35" even if it's 2.356324. Thanks String.format!

}

}

AccountHolderTest.java

**package** lab1;

/\*\*

\* @author henryfbp

\* @date 09/15/17

\* @class ITMD411

\* @lab 01

\*/

**import** **java.text.SimpleDateFormat**;

**import** **java.util.Calendar**;

**import** **java.util.Scanner**;

**public** **class** **AccountHolderTest** {

**static** **final** String nlc = "\n > ";

/\*\*

\* @param args the command line arguments

\*/

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

{

**int** testVal = **140**;

AccountHolder myTestAccount = **new** AccountHolder(testVal);

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

System.out.println("Enter your initial account balance:"+nlc);

AccountHolder myAccount = **new** AccountHolder(scan.nextDouble());

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

myAccount.deposit(scan.nextDouble());

System.out.println("Enter amount to be withdrawn:"+nlc);

myAccount.withdrawal(scan.nextDouble());

System.out.println("The following is what happens when one attempts to "

+ "make a withdrawal that would drop your account balance below"

+ " $100,\n in this case we have $"+testVal+" and want to take out $50:");

myTestAccount.withdrawal(**50**);

System.out.println("Monthly balances for one year at 0.04");

myAccount.modifyMonthlyInterest(**0.04**);

System.out.println("Balances:");

System.out.println("Account Balance w. Interest");

System.out.printf("Base %15s\n",(myAccount.toString()));

**for**(**int** i = **1**; i <= **12**; i++)

{

myAccount.monthlyInterest(myAccount.balance);

System.out.printf("Month %2d: %10s\n", i,(myAccount.toString()));

}

System.out.println("After setting interest rate to .05 and calculating monthly interest");

System.out.println("Balances:");

System.out.println("Account Balance w. Interest");

myAccount.modifyMonthlyInterest(**0.05**);

myAccount.monthlyInterest(myAccount.balance);

System.out.println(myAccount.toString());

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

System.out.println("Cur dt=" + timeStamp + "\nProgrammed by Henry Post\n");

}

}

Junit.java

**package** lab1;

**import** **static** org.junit.Assert.\*;

**import** **org.junit.Before**;

**import** **org.junit.BeforeClass**;

**import** **org.junit.Test**;

**public** **class** **Junit** {

**private** AccountHolder account;

**@BeforeClass**

**public** **static** **void** **setUpBeforeClass**() **throws** Exception

{

}

**@Before**

**public** **void** **setUp**() **throws** Exception

{

account = **new** AccountHolder(**0**);

}

**@Test**

**public** **void** **testInitialInterest**()

{

**if**(account.annualInterestRate != **0**)

{

fail("Account initial interest is not zero. It is '" + account.annualInterestRate + "'");

}

}

**@Test**

**public** **void** **testBalance**()

{

account = **new** AccountHolder(-**269863945**);

**if**(account.balance < **0**)

{

fail("Account balance is less than zero.");

}

}

**@Test**

**public** **void** **testModifyInterest**()

{

**double** rate = **0.05**;

**double** bal = **1000**;

account = **new** AccountHolder(bal);

account.modifyMonthlyInterest(rate);

**for**(**int** i = **0**; i < **999**; i++)

{

bal = bal + (bal \* (rate / **12.0**)); //compound interest

account.monthlyInterest(account.balance);

**if**(account.balance != bal)

{

fail("Acc.bal != bal aka '" + account.balance + "' != '" + bal + "'");

}

}

}

**@Test**

**public** **void** **testNullObject**()

{

account = **null**;

**if**(account != **null**)

{

fail("Assigning null value to account somehow failed!");

}

}

**protected** **void** **tearDown**() **throws** Exception

{

System.out.println("Running teardown.");

account = **null**;

assertNull(account);

}

}