**David Hazall-Farrell**

**CS-132-Lab 7**

**3 hours**

**It wasn’t too bad, just an extension of the previous bank account lab.**

3.

**public** **class** AltMain {

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

SavingsAccount savings = **new** SavingsAccount("Oliver Spire", 1000, "SA13555", 5);

// Test: Can you print a SavingsAccount object directly?

System.***out***.println(savings);

// Test: Try adding interest to a BankAccount object

BankAccount bank = **new** BankAccount("Oliver Spire", 1000, "SA13555");

//bank.addInterest();

// Test: Assign a SavingsAccount object to a BankAccount variable

BankAccount bankAccount = savings;

//bankAccount.addInterest(); // This will NOT work as BankAccount doesn't have addInterest() method

System.***out***.println("Balance using BankAccount reference: " + bankAccount.getBalance());

// instanceof checks

System.***out***.println(savings **instanceof** SavingsAccount); // true

System.***out***.println(savings **instanceof** BankAccount); // true

System.***out***.println(savings **instanceof** Object); // true

}

}

It prints out the Savings account just as I expected.

Adding interest doesn’t allow the program to run and an error is thrown.

It is possible, but it isn’t possible to add interest.

**public** **class** SavingsAccount **extends** BankAccount {

**private** **double** interestRate;

// Constructor

**public** SavingsAccount(String custName, **double** bal, String acctName ,**double** interestRate) {

**super**(custName, bal, acctName);

**this**.interestRate = interestRate;

}

// Add interest to the balance

**public** **void** addInterest() {

**double** interest = getBalance() \* (interestRate / 100);

deposit(interest);

}

@Override

**public** String toString() {

**return** **super**.toString() + ", Interest Rate: " + interestRate + "%";

}

}

**public** **class** CheckingAccount **extends** BankAccount {

**private** **int** transactionCount;

**private** **static** **final** **double** ***transactionFee*** = 0.25;

**public** CheckingAccount(String custName, **double** bal, String acctName) {

**super**(custName, bal, acctName);

**this**.transactionCount = 0;

}

@Override

**public** **void** deposit(**double** amount) {

**super**.deposit(amount);

transactionCount++;

}

@Override

**public** **void** withdraw(**double** amount) {

**super**.withdraw(amount);

transactionCount++;

}

**public** **void** deductFees() {

**double** totalFees = transactionCount \* ***transactionFee***;

deposit(-totalFees);

transactionCount = 0;

}

}

import static org.junit.Assert.\*;

import org.junit.Test;

public class SavingsAccountTest {

@Test

public void testAddInterest() {

SavingsAccount savings = new SavingsAccount("John Doe", 1000, "SA12345", 5);

savings.addInterest();

assertEquals(1050, savings.getBalance(), 0.01);

}

@Test

public void testToString() {

SavingsAccount savings = new SavingsAccount("John Doe", 1000, "SA12345", 5);

String expectedOutput = "Customer: John Doe, Balance: 1000.0, Account Number: SA12345, Interest Rate: 5.0%";

assertEquals(expectedOutput, savings.toString());

}

}

public class CheckingAccountTest {

@Test

public void testTransactionFees() {

CheckingAccount checking = new CheckingAccount("Jane Doe", 1000, "CA18936");

checking.deposit(200);

checking.withdraw(50);

checking.deductFees();

// After 2 transactions, the fee should be 2 \* 0.25 = 0.50, new balance should be 1000 + 200 - 50 - 0.50 = 1149.50

assertEquals(1149.50, checking.getBalance(), 0.01);

}

}