CS313 Part A Testing Strategy

Test 1

**public** **class** Driver1 {

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

System.***out***.println("Scenario #1");

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

Account a1 = **new** Account(1, "Account #1");

a1.setBalance(85.0);

RunnablePrintBalance rpb1 = **new** RunnablePrintBalance(a1);

RunnablePrintBalance rpb2 = **new** RunnablePrintBalance(a1);

Thread rpbT1 = **new** Thread(rpb1);

Thread rpbT2 = **new** Thread(rpb2);

rpbT1.start();

rpbT2.start();

}

}

Overview

This test uses two threads. Both threads rpbT1 and rpbT2 will simulate two account holders trying to print the bank balance. Both threads will be running simultaneously.

Sample Output

The balance on Account #1 has been set to £85.0.

Thread with ID 11 (PB): Trying to print the balance on Account #1...

Thread with ID 11 (PB): Checking for any possible concurrent balance-changing operations to have terminated...

Thread with ID 12 (PB): Trying to print the balance on Account #1...

Thread with ID 12 (PB): Checking for any possible concurrent balance-changing operations to have terminated...

Thread with ID 11 (PB): There are no impending balance-changing operations.

Thread with ID 11 (PB): The balance on Account #1 is £85.0.

Thread with ID 12 (PB): There are no impending balance-changing operations.

Thread with ID 12 (PB): The balance on Account #1 is £85.0.

Input

Initial Balance: 85

Execution Sequence

The critical code for printing balance is executed first:

Expected: Wait to see if any other operations have to be done before printing balance.

Actual: Waits for a certain amount of time before continuing.

The critical code for printing balance a second time is executed second:

Expected: Wait to see if any other operations have to be done before printing balance.

Actual: Waits for a certain amount of time before continuing.

The critical code for printing balance has finished waiting:

Expected: Print balance as £85

Actual: Prints balance as £85

The critical code for printing balance a second time has finished waiting:

Expected: Print balance as £85

Actual: Prints balance as £85

Test Outcome = PASS

Test 2

**public** **class** Driver2 {

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

System.***out***.println("Scenario #2");

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

Account a2 = **new** Account(2, "Account #2");

a2.setBalance(126.0);

RunnablePrintBalance rpb = **new** RunnablePrintBalance(a2);

RunnableDeposit rd = **new** RunnableDeposit(a2, 40);

RunnableWithdraw rw = **new** RunnableWithdraw(a2, 50);

Thread rpbT = **new** Thread(rpb);

Thread rdT = **new** Thread(rd);

Thread rwT = **new** Thread(rw);

rpbT.start();

rdT.start();

rwT.start();

}

}

Overview

This test uses three threads. The first thread rpbT simulates an account holder trying to check the balance. The second thread rdT simulates another user trying to deposit money into the same account. The third thread rwT simulates another user trying to withdraw money from the same account. All of these actions are happening simultaneously.

Sample Output

The balance on Account #2 has been set to £126.0.

Thread with ID 11 (PB): Trying to print the balance on Account #2...

Thread with ID 11 (PB): Checking for any possible concurrent balance-changing operations to have terminated...

Thread with ID 12 (DE): Trying to deposit £40.0 on Account #2...

Thread with ID 12 (DE): £40.0 have been deposited on Account #2.

Thread with ID 12 (DE): The balance on Account #2 is now £166.0.

Thread with ID 13 (WI): Trying to withdraw £50.0 from Account #2...

Thread with ID 13 (WI): £50.0 have been withdrawn from Account #2.

Thread with ID 13 (WI): The balance on Account #2 is now £116.0.

Thread with ID 11 (PB): There are no impending balance-changing operations.

Thread with ID 11 (PB): The balance on Account #2 is £116.0.

Input

Initial Balance: £126

Deposit Amount: £40

Withdraw Amount: £50

Execution Sequence

The critical code for printing balance is executed first:

Expected: Wait to see if any other operations have to be done before printing balance.

Actual: Waits until Deposit and Withdraw operations are done.

The critical code for depositing is executed second:

Expected: Balance after deposit = £166

Actual: Balance after deposit = £166

The critical code for withdrawing is executed second:

Expected: Balance after deposit = £116

Actual: Balance after deposit = £116

The critical code for printing balance is continued:

Expected: Print balance as £116

Actual: Prints balance as £116

Test Outcome: PASS