1. -----------------------------------------------------------------------

using System;

namespace BankAccountNS

{

/// <summary>

/// Bank Account demo class.

/// </summary>

public class BankAccount

{

private string m\_customerName;

private double m\_balance;

private bool m\_frozen = false;

private BankAccount()

{

}

public BankAccount(string customerName, double balance)

{

m\_customerName = customerName;

m\_balance = balance;

}

public string CustomerName

{

get { return m\_customerName; }

}

public double Balance

{

get { return m\_balance; }

}

public void Debit(double amount)

{

if (m\_frozen)

{

throw new Exception("Account frozen");

}

if (amount > m\_balance)

{

throw new ArgumentOutOfRangeException("amount");

}

if (amount < 0)

{

throw new ArgumentOutOfRangeException("amount");

}

m\_balance += amount;

}

public void Credit(double amount)

{

if (m\_frozen)

{

throw new Exception("Account frozen");

}

if (amount < 0)

{

throw new ArgumentOutOfRangeException("amount");

}

m\_balance += amount;

}

private void FreezeAccount()

{

m\_frozen = true;

}

private void UnfreezeAccount()

{

m\_frozen = false;

}

public static void Main()

{

BankAccount ba = new BankAccount("Mr. Bryan Walton", 11.99);

ba.Credit(5.77);

ba.Debit(11.22);

Console.WriteLine("Current balance is ${0}", ba.Balance);

}

}

}

1. -----------------------------------------------------------------------------------------------------------------

[TestMethod]

public void Debit\_WithValidAmount\_UpdatesBalance()

{

// arrange

double beginningBalance = 11.99;

double debitAmount = 4.55;

double expected = 7.44;

BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// act

account.Debit(debitAmount);

// assert

double actual = account.Balance;

Assert.AreEqual(expected, actual, 0.001, "Account not debited correctly");

}

1. -----------------------------------------------------------------

using BankAccountNS;

1. -----------------------------------------------------------------------------------------------------------------

//unit test method

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void Debit\_WhenAmountIsLessThanZero\_ShouldThrowArgumentOutOfRange()

{

// arrange

double beginningBalance = 11.99;

double debitAmount = -100.00;

BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// act

account.Debit(debitAmount);

// assert is handled by ExpectedException

}

//unit test method

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void Debit\_WhenAmountIsLMoreThanBallance\_ShouldThrowArgumentOutOfRange()

{

// arrange

double beginningBalance = 11.99;

double debitAmount = 100.00;

BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// act

account.Debit(debitAmount);

// assert is handled by ExpectedException

}

1. -----------------------------------------------------------------------------------------------------------------

//Refactoring

public const string DebitAmountExceedsBalanceMessage = "Debit amount exceeds balance";

public const string DebitAmountLessThanZeroMessage = "Debit amount less than zero";

1. ------------------------------------------------------------------------------------------------------------------

if (amount > m\_balance)

{

throw new ArgumentOutOfRangeException("amount", amount, DebitAmountExceedsBalanceMessage);

}

if (amount < 0)

{

throw new ArgumentOutOfRangeException("amount", amount, DebitAmountLessThanZeroMessage);

}

1. ------------------------------------------------------------------

//unit test method

[TestMethod]

//[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void Debit\_WhenAmountIsLessThanZero\_ShouldThrowArgumentOutOfRange()

{

// arrange

double beginningBalance = 11.99;

double debitAmount = -100.00;

BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// act

//account.Debit(debitAmount);

try

{

account.Debit(debitAmount);

}

catch (ArgumentOutOfRangeException e)

{

// assert

StringAssert.Contains(e.Message, BankAccount.DebitAmountLessThanZeroMessage);

}

// assert is handled by ExpectedException

}

//unit test method

[TestMethod]

//[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void Debit\_WhenAmountIsMoreThanBalance\_ShouldThrowArgumentOutOfRange()

{

// arrange

double beginningBalance = 100;

double debitAmount = 100;

BankAccount account = new BankAccount("Mr. Bryan Walton", beginningBalance);

// act

//account.Debit(debitAmount);

try

{

account.Debit(debitAmount);

}

catch (ArgumentOutOfRangeException e)

{

// assert

StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);

}

// assert is handled by ExpectedException

}

1. ---------------------------------------------------------------

return;

Assert.Fail("No exception was thrown.");