PROBLEM #1

Given the following interface:

```
public interface IAccountService
{
double GetAccountAmount(int _accountId);
}
```

...and the following class that depends on this interface:

```
public class AccountInfo
{
private readonly int _accountId;
private readonly IAccountService _accountService;
public AccountInfo(int accountId, IAccountService accountService)
{
_accountId = accountId;
_accountService = accountService;
}
public double Amount { get; private set; }
public void RefreshAmount()
{
Amount = _accountService.GetAccountAmount(_accountId);
}
}
```

REQUIRED: Write a unit test that asserts the behaviour of RefreshAmount() method.

Note: Not supplying an implementation of the service interface is deliberate; it is expected to create a *Test Double*.

PROBLEM #2

It has been determined that IAccountService.GetAccountAmount() is a potentially slow and unreliable remote network call and that it should be made asynchronous. Note that AccountInfo.RefreshAmount() *may* be invoked multiple times concurrently. Adjust IAccountService and / or

AccountInfo and your tests accordingly.

Answer:

Problem 1:

Use mocking technique. Assumption is IAccountService.GetAccountAmount does not throw exception over interface (usually the case, stated otherwise).

```
 [TestClass]
  public class TestMethod
  {
    [TestMethod]
    public void TestRefreshAmount()
    {
      var mock = new MockRepository();
      var accountServiceMock = mock.StrictMock<IAccountService>();

      var expectedAccountId = new Random().Next(100);
      var accountInfoToTest = new AccountInfo(expectedAccountId, accountServiceMock);

      var expectedReturnAmount = new Random().NextDouble();
      Expect.Call(accountServiceMock.GetAccountAmount(expectedAccountId)).Return(expe
ctedReturnAmount);
      mock.ReplayAll();

      accountInfoToTest.RefreshAmount();
      Assert.AreEqual(expectedReturnAmount, accountInfoToTest.Amount);

      mock.VerifyAll();
    }

  }
```

Problem 2:

Use mocking and autoresetEvent technique. Reset event is used for getting access to RefreshAmount function.

```
    public interface IAccountService
    {
        double GetAccountAmount(int accountId);
    }

    public class AccountInfo
    {
        private readonly int _accountId;
        private readonly IAccountService _accountService;
        private readonly AutoResetEvent getAccountAmountResetEvent;

        public AccountInfo(int accountId, IAccountService accountService, AutoResetEvent
accountAmountResetEvent)
        {
            _accountService = accountService;
            _accountId = accountId;
            getAccountAmountResetEvent = accountAmountResetEvent;
        }

        public double Amount { get; private set; }

        private delegate double GetAccountAmountDelegate(int accountId);

        public void RefreshAmount()
        {
            if (getAccountAmountResetEvent.WaitOne(0))
            {
                var getAccountAmountdlg = new GetAccountAmountDelegate(GetAccountAmount);
                getAccountAmountdlg.BeginInvoke(this._accountId, CallBack,
getAccountAmountdlg);
            }
        }

        private void CallBack(IAsyncResult ar)
        {
            var handler = (GetAccountAmountDelegate) ar.AsyncState;
            Amount = handler.EndInvoke(ar);

            getAccountAmountResetEvent.Set();
        }

        private double GetAccountAmount(int accountid)
        {
            return _accountService.GetAccountAmount(accountid);
        }
    }
```

```
  [TestClass]
  public class TestMethod
  {
    [TestMethod]
    public void TestRefreshAmount()
    {
      var mock = new MockRepository();
      var getAmountEventReceived = new AutoResetEvent(false);

      var accountServiceMock = mock.StrictMock<IAccountService>();
      var getAmountResetEventMock = mock.StrictMock<AutoResetEvent>();

      var expectedAccountId = new Random().Next(100);
      var accountInfoToTest = new AccountInfo(expectedAccountId, accountServiceMock,
getAmountResetEventMock);

      var expectedReturnAmount = new Random().NextDouble();
      Expect.Call(getAmountResetEventMock.WaitOne(0)).Return(true);
      Expect.Call(getAmountResetEventMock.WaitOne(0)).Return(false);
      Expect.Call(accountServiceMock.GetAccountAmount(expectedAccountId)).Return(expe
ctedReturnAmount);
      Expect.Call(getAmountResetEventMock.Set()).Return(true).WhenCalled(MethodInvoca
tion => getAmountEventReceived.Set());
      mock.ReplayAll();

      getAmountEventReceived.Reset();

      accountInfoToTest.RefreshAmount();
      accountInfoToTest.RefreshAmount();

      Assert.IsTrue(getAmountEventReceived.WaitOne(1000));
      Assert.AreEqual(expectedReturnAmount, accountInfoToTest.Amount);

      mock.VerifyAll();

    }

  }
```