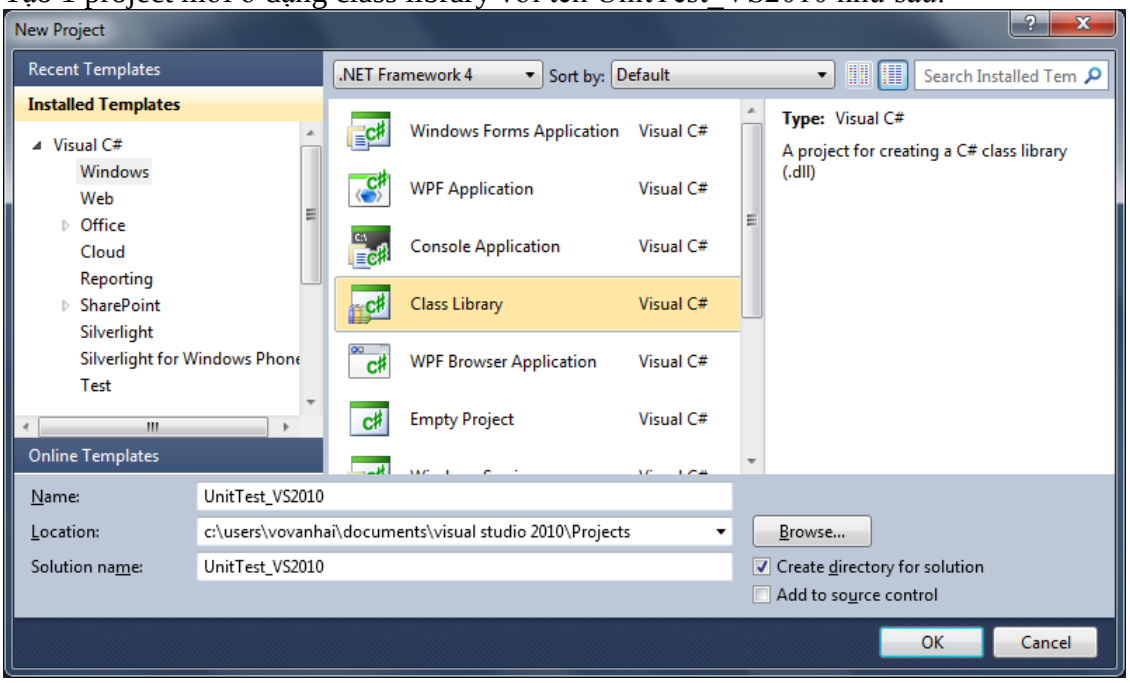
UNIT TEST VỚI VISUAL 2010

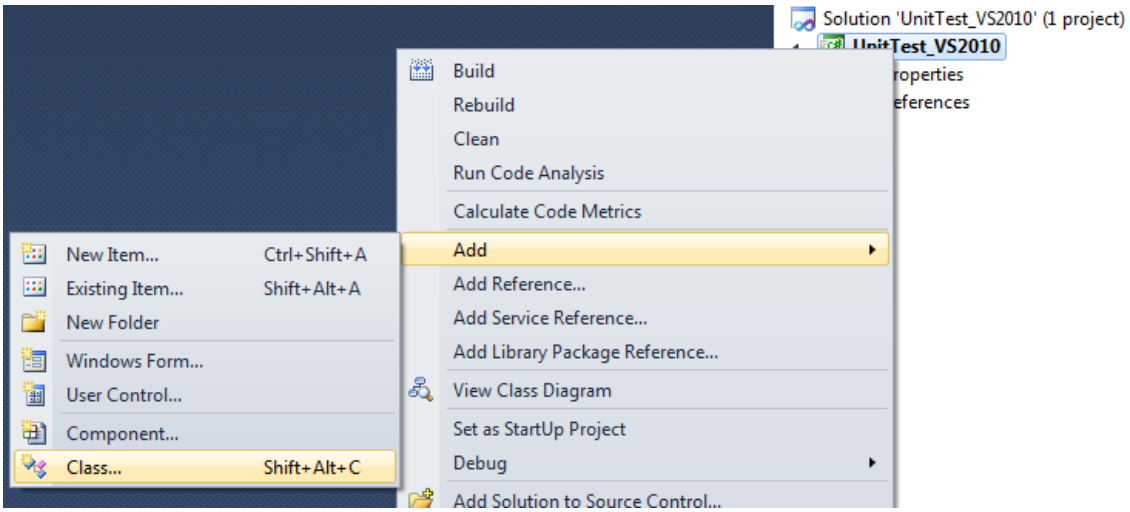
Giả sử ta phải viết 1 module đặc tả một đối tượng tài khoản trong ngân hàng. Mỗi tài khoản gồm các thuộc tính: số tài khoản, tên chủ tài khoản, số tiền. Mỗi tài khoản có các phương thức: rút tiền, nộp tiền, chuyển tiền. Có một vài ràng buộc như sau: số tiền rút 1 lần không được quá 5 triệu đồng, số tiền chuyển cho 1 lần không được quá 20 triệu đồng, số tiền nộp là không hạn chế. Yêu cầu viết unit testing cho module này với các testcase sử dụng phương pháp robustness testing.

Thiết kế chương trình (sử dụng C#)

Tạo 1 project mới ở dạng class library với tên UnitTest_VS2010 như sau:



Tạo 1 lớp mới có tên **Account**



Code cho lớp Account như sau:

```
using System;

namespace UnitTest_VS2010
{
    public class Account
    {
        private long accID;//số tài khoản
        private string ownerName;//tên người sở hữu
```

```
        private double balance;//số tiền có trong tài khoản

        public Account(long accID, string ownerName)
        {
            this.accID = accID;
            this.ownerName = ownerName;
            this.balance = 0;
        }
        /// <summary>
        /// Thiết lập giá trị thuộc tính tên chủ sở hữu tài khoản
        /// </summary>
        public string OwnerName
        {
            set
            {
                this.ownerName = value;
            }
            get
            {
                return this.ownerName;
            }
        }
        /// <summary>
        /// Thiết lập giá trị thuộc tính số tiền trong tài khoản
        /// </summary>
        public double Balance
        {
            set
            {
                this.balance = value;
            }
            get
            {
                return this.balance;
            }
        }
        /// <summary>
        /// Nộp tiền vào tài khoản
        /// </summary>
        /// <param name="amount">số tiền cần nộp</param>
        /// <returns>tài khoản hiện hành</returns>
        public Account Deposit(double amount)
        {
            if (amount <= 0)
                throw new Exception("số tiền nộp không hợp lệ");
            this.balance += amount;
            return this;
        }
        /// <summary>
        /// Rút tiền
        /// </summary>
        /// <param name="amount">số tiền cần rút</param>
        /// <returns>tài khoản hiện hành</returns>
        public Account Withdrawal(double amount)
        {
            if (amount <= 0)
                throw new Exception("số tiền rút không hợp lệ");
            if (amount > balance)
                throw new Exception("không đủ tiền để rút");
            this.balance -= amount;
            return this;
        }
```

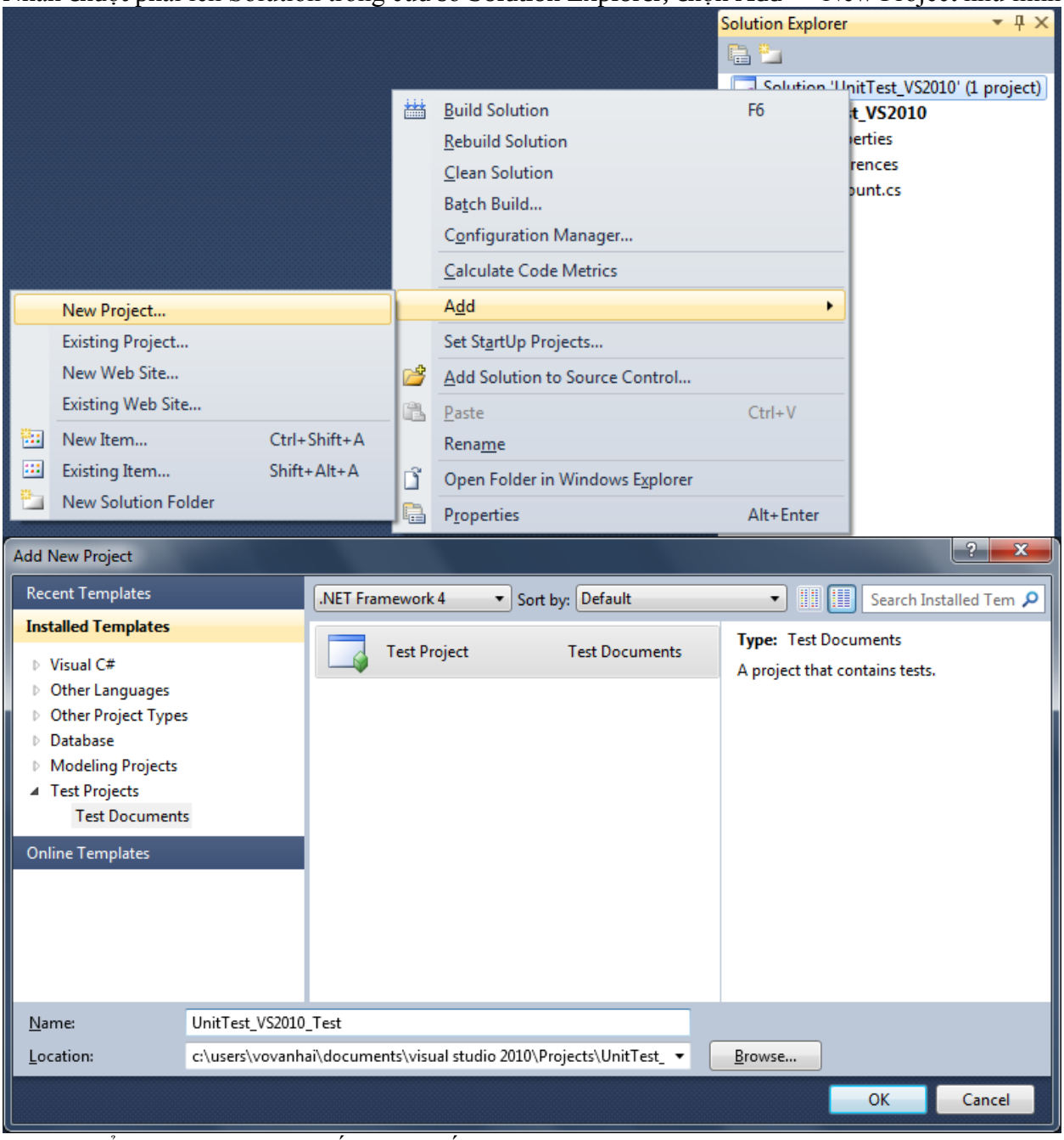
```
        /// <summary>
        /// Chuyển tiền
        /// </summary>
        /// <param name="to">tài khoản cần chuyển tới</param>
        /// <param name="amount">số tiền cần chuyển</param>
        /// <returns>tài khoản hiện hành</returns>
        public Account Transfer(Account to, double amount)
        {
            if (amount <= 0)
                throw new Exception("số tiền chuyển không hợp lệ");
            if (amount > this.balance)
                throw new Exception("không đủ tiền để chuyển");
            this.balance -= amount;
            to.Balance += amount;
            return this;
        }

        public override int GetHashCode()
        {
            return accID.GetHashCode();
        }
        public override bool Equals(object obj)
        {
            if (this == obj)
                return true;
            if (obj == null)
                return false;
            if (this.GetType() != obj.GetType())
                return false;
            Account other = (Account)obj;
            if (this.accID != other.accID)
                return false;
            return true;
        }
        public override string ToString()
        {
            return "Số tài khoản: " + accID +
                ". Tên chủ sở hữu: " + ownerName +
                ". Số tiền trong tài khoản:" + balance;
        }
    }
}
```

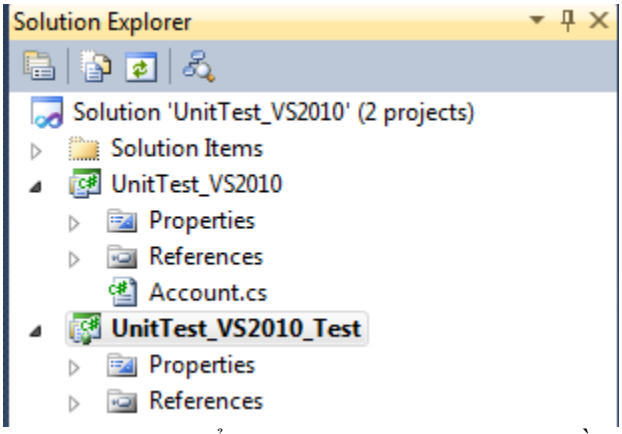
Biên dịch project ta được 1 file DLL có tên UnitTest_VS2010.dll

Bây giờ ta tiếp tục với việc tạo ra 1 Unit test project.
Thiết kế các testcases:
Các testcases theo phương pháp này cho quá trình rút tiền như sau: -1; 0; 1; 5,000,000, 9,999,999; 10,000,000; 10,000,001 (cho các trường hợp biên của điều kiện rút) và số tiền có trong tài khoản -1; số tiền có trong tài khoản; số tiền có trong tài khoản +1 (cho các trường hợp giới hạn tiền rút).
Tương tự cho quá trình chuyển tiền, các testcases sẽ là: -1; 0; 1; 10,000,000, 19,999,999; 20,000,000; 20,000,001 (cho các trường hợp biên của điều kiện rút) và số tiền có trong tài khoản -1; số tiền có trong tài khoản; số tiền có trong tài khoản +1 (cho các trường hợp giới hạn tiền rút).
Trong trường hợp nộp tiền, ta sẽ test cho cận dưới và 1 số ngẫu nhiên nào đó khác cận dưới. Testcases sẽ như sau: -1; 0; 1; 15,900,000;

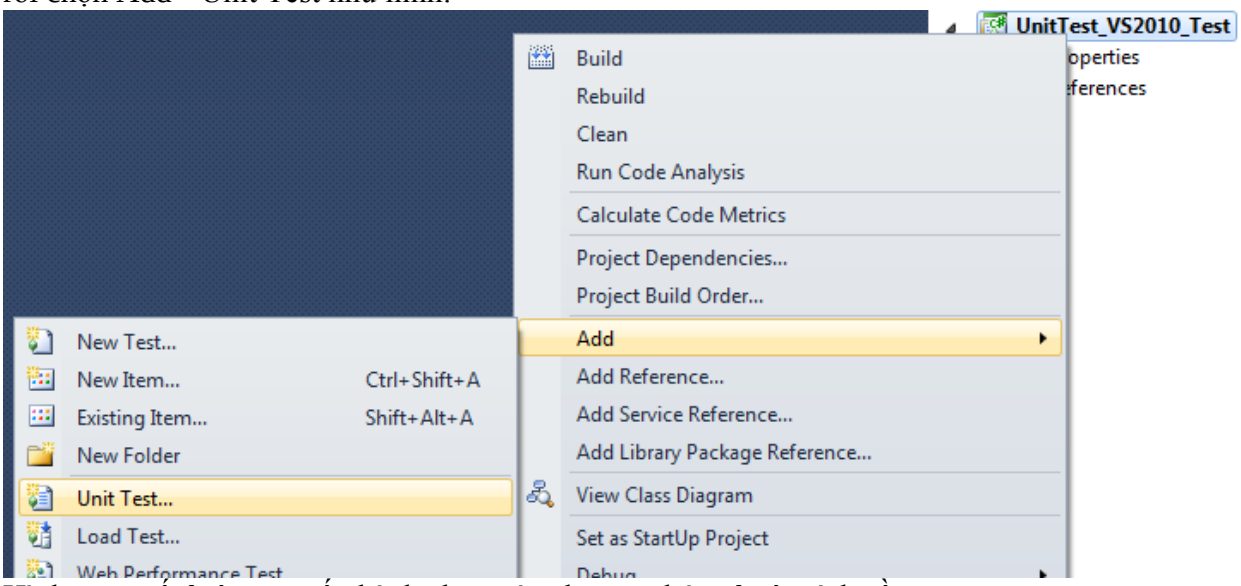
Tạo 1 Test Project trong VS 2010 như sau:
Nhấn chuột phải lên Solution trong cửa sổ Solution Explorer, chọn Add--> New Project như hình



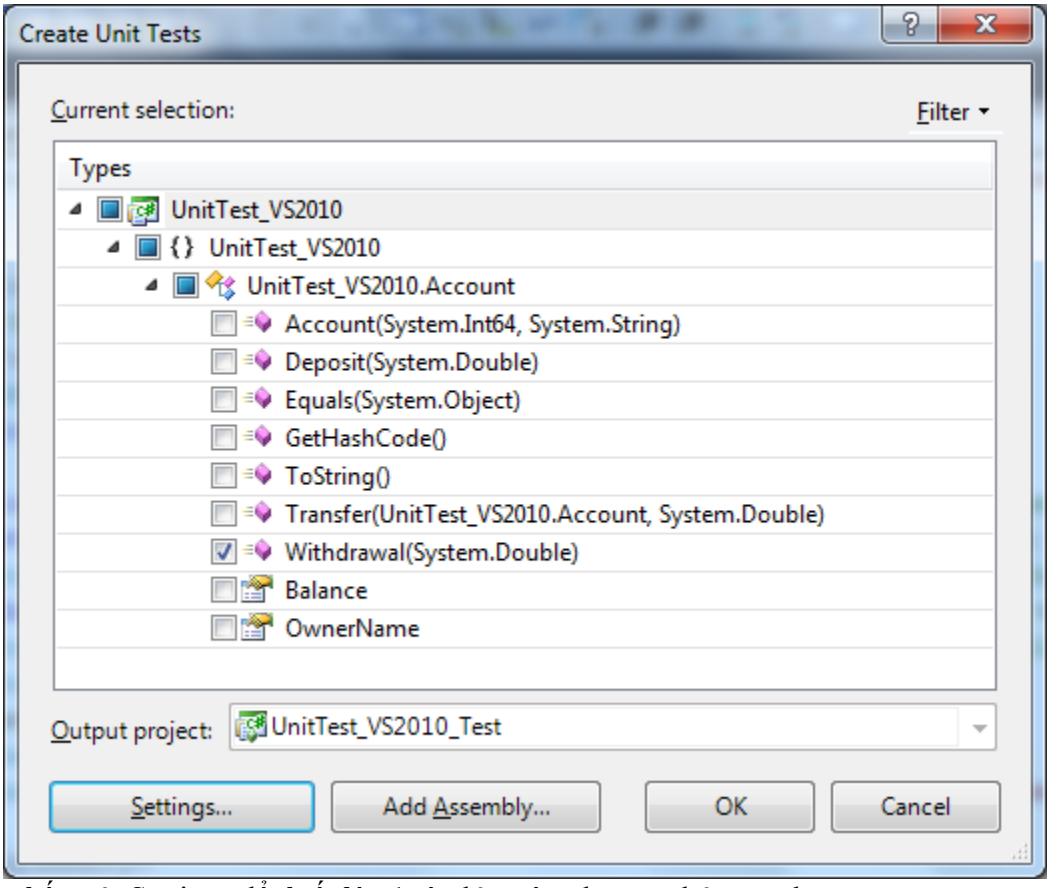
Bạn có thể xóa bỏ UnitTest1 nếu bạn muốn. Project chúng ta bây giờ như sau:



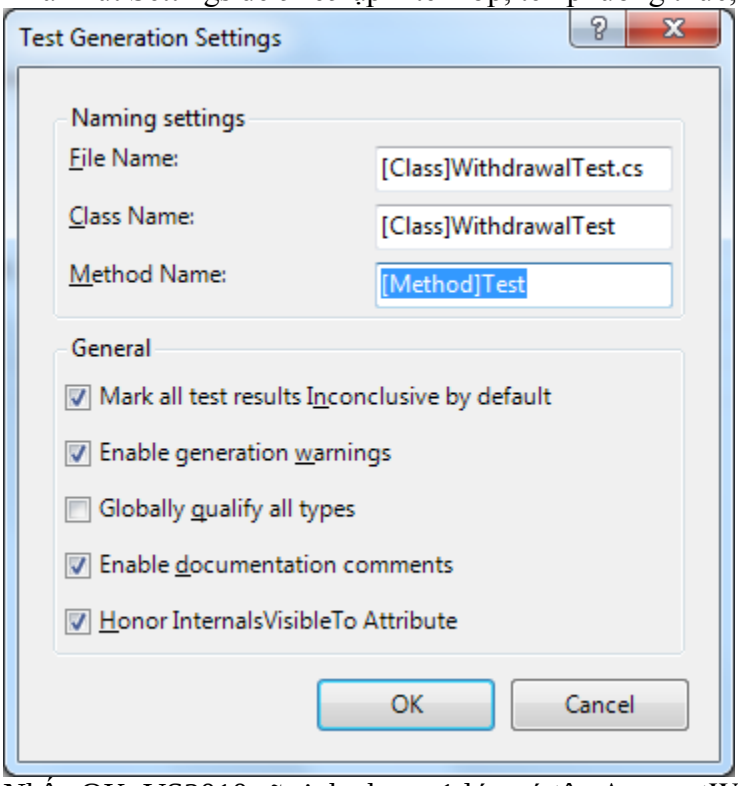
Bây giờ ta có thể thêm vào 1 unit test mới bằng cách nhấn phải chuột lên test project của chúng ta rồi chọn Add->Unit Test như hình:



Hình sau xuất hiện. Ta tiến hành chọn các phương thức, thuộc tính cần test:
Ta test cho phương thức Withdrawal trước như sau



Nhấn nút Settings để thiết lập 1 tên lớp, tên phương thức,… như sau



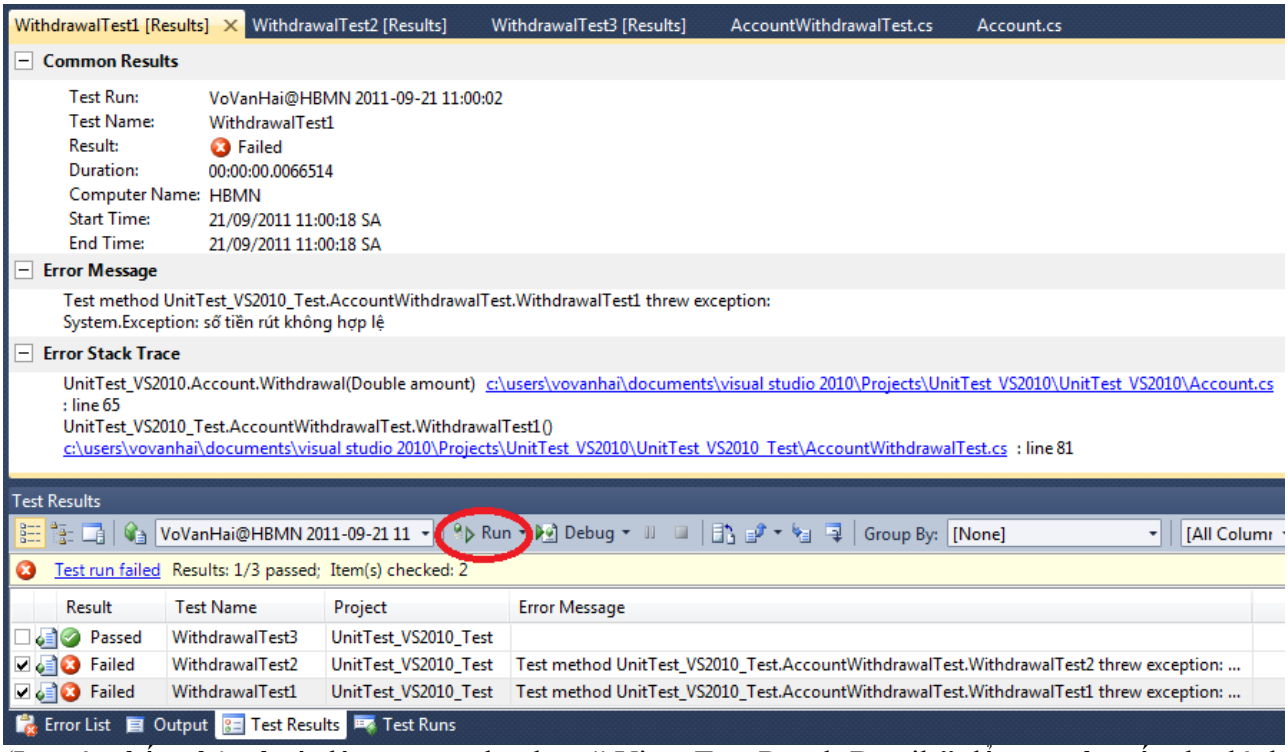
Nhấn OK. VS2010 sẽ sinh cho ta 1 lớp có tên AccountWithdrawalTest và một phương thức mẫu để test cho phương thức rút tiền.

Ta có tất cả 10 testcases cho trường hợp rút tiền. Ta tiến hành copy thành 10 phương thức và đặt tên theo thứ tự, sau đó đưa các giá trị thử vào.

Ta test thử cho trường hợp 1, 2 và 3. Code như sau:

```
[TestMethod()]
    public void WithdrawalTest1()
    {
        long accID = 1001;
        string ownerName = "Võ Văn Hải";
        Account target = new Account(accID, ownerName);
        target.Deposit(1000000);//nộp vào 1 triệu cho tài khoản thử
        double amount = -1;//testcase 1
        Account expected = new Account(accID,ownerName);
        expected.Balance = 1000001;
        Account actual;
        actual = target.Withdrawal(amount);
        Assert.AreEqual(expected.Balance, actual.Balance);
    }
[TestMethod()]
    public void WithdrawalTest2()
    {
        long accID = 1001;
        string ownerName = "Võ Văn Hải";
        Account target = new Account(accID, ownerName);
        target.Deposit(1000000);//nộp vào 1 triệu cho tài khoản thử
        double amount = 0; //testcase 2
        Account expected = new Account(accID, ownerName);
        expected.Balance = 1000000;
        Account actual;
        actual = target.Withdrawal(amount);
        Assert.AreEqual(expected.Balance, actual.Balance);
    }
[TestMethod()]
    public void WithdrawalTest3()
    {
        long accID = 1001;
        string ownerName = "Võ Văn Hải";
        Account target = new Account(accID, ownerName);
        target.Deposit(1000000);//nộp vào 1 triệu cho tài khoản thử
        double amount = 1;
        Account expected = new Account(accID, ownerName);
        expected.Balance = 999999;
        Account actual;
        actual = target.Withdrawal(amount);
        Assert.AreEqual(expected.Balance, actual.Balance);
    }
```

Chạy thử ta có kết quả:



(Lưu ý: nhấn phải chuột lên test result, chọn " View Test Result Details" để xem chi tiết như hình trên)

Tương tự ta thêm các testcase khác vào thử.

Và cũng tương tự ta xây dựng các lớp test cho phương thức Deposit và transfer