Using *PrivateObject* to access nonpublic instance members

Suppose you'd like to test the private field and method of the following class:

```
public class Example
{
        public Example() {}
        private string password = "letmein";
        private bool VerifyPassword(string password)
        {
                return (String.Compare(this.password, password, false) == 0);
        }
}
```

Because the field and method are marked private, a unit test will not have the ability to directly access them. How can you ensure that VerifyPassword is working correctly?
Team System introduces the PrivateObject class, which is a wrapper around reflection code that enables you to access nonpublic members in a fairly straightforward manner.

The following table summarizes the methods supported by PrivateObject.

Method	Description
GetArrayElement	Returns the selected item from a private array member. Supports multidimensional arrays with additional arguments.
GetField	Returns the value of the target field
GetFieldOrProperty	Returns the value of the target field or property
GetProperty	Returns the value of the target property
Invoke	Invokes the target method, optionally passing parameters
SetArrayElement	Assigns the given value to the indicated element of a private array Supports multidimensional arrays with additional arguments
SetField	Assigns the supplied object to the target field
SetFieldOrProperty	Assigns the supplied object to the target field or property
SetProperty	Assigns the supplied object to the target property

To use it, you first create an instance of the PrivateObject class, passing a Type object for the class you wish to work with:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
namespace Explorations
{
                [TestClass]
                public class ExampleTest
                {
                private PrivateObject privateObject;
                const string PASSWORD = "letmein";
                [TestInitialize]
                public void TestInitialize()
```

```
                {
                privateObject = new PrivateObject(typeof(Example));
                }
        }
}
```

Now you can create your tests. Use the GetField method of the ***PrivateObject*** instance to access non-public fields, supplying the name of the desired variable. Similarly, use Invoke to call methods, supplying the method name as the first argument, followed by any parameters to that method:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
namespace Explorations
{
        [TestClass]
        public class ExampleTest
        {
                const string PASSWORD = "letmein";
                [TestInitialize]
                public void TestInitialize()
                {
                        privateObject = new PrivateObject(typeof(Example));
                }
                [TestMethod]
                public void ComparePrivatePassword()
                {
                        string password = (string)privateObject.GetField("password");
                        Assert.AreEqual(PASSWORD, password);
                }
                [TestMethod]
                public void TestPrivateVerifyPassword()
                {
                        bool accepted = (bool)privateObject.Invoke("VerifyPassword", PASSWORD);
                        Assert.IsTrue(accepted);
                }
        }
}
```

Because ***PrivateObject*** uses reflection, you need to cast the results of these calls from the generic Object type back to the correct underlying type.

Using *PrivateType* to access nonpublic static members

PrivateObject is used to access instance-based members of a class. If you need to access static nonpublic members, you use the ***PrivateType*** class, which has a very similar interface and is a wrapper of reflection code.

The following table summarizes the methods exposed by ***PrivateType***.

Method	Description
GetStaticArrayElement	Returns the selected item from a private static array member. Supports multidimensional arrays with additional arguments.
GetStaticField	Returns the value of the target static field.
GetStaticProperty	Returns the value of the target static property.
InvokeStatic	Invokes the target static method, optionally passing parameters.
SetStaticArrayElement	Assigns the given value to the indicated element of a private static array. Supports multidimensional arrays with additional arguments.
SetStaticField	Assigns the supplied object to the target static field.
SetStaticProperty	Assigns the supplied object to the target static property.

The usage is very similar to the ***PrivateObject***. Create an instance of the ***PrivateType***, indicating which type you wish to work with, and then use the methods to access members as with ***PrivateObject***. Suppose you added a private static count of password failures with a wrapping private property called ***FailureCount***. The following code could read and test that property:

```
[TestMethod]
public void TestPrivateStaticFailureCount() {
        PrivateType example = new PrivateType(typeof(Example));
        int failureCount = (int)example.GetStaticProperty("FailureCount");
        Assert.AreEqual(failureCount, 0);
}
```

Again, you create an instance of ***PrivateType***, passing the type reference for the class you wish to access. Then you use that instance, invoking ***GetStaticProperty*** to retrieve the value you wish to test. Finally, you ensure that the value is zero as expected.

> ***Use caution when testing static data. Because static data is shared and is not automatically reset between your tests, sometimes the order of your tests will affect their results. In other words, if you test that a value is initialized to zero in one test and then set it to a test value in another test, if the order of those tests is reversed, the zero test will fail. Remember that you must be able to run your tests in any order and in any combination.***