Some might say that testing private methods should be avoided because it means not testing the contract, that is the interface implemented by the class, but the internal implementation of the class itself. Still, not all classes were designed with testability in mind, so real life compromises sometimes demand such a trick.

When writing unit tests in C# with MSTest, the PrivateObject class lets you easily call private methods:

```
1. [TestMethod]
2. public void TestLPRead()
3. {
4.  var Logger = A.Fake<ILogger>();
5.  var Telemetry = A.Fake<ITelemetry>();
6.  DefaultDataModel DM = new DefaultDataModel(Logger, Telemetry);
7.  PrivateObject obj = new PrivateObject(DM);
8.  List<LPRead> ReadsList = (List<LPRead>)obj.Invoke("GetReads");
```

In the code above, a PrivateObject instance is created passing an instance of the class to be tested

```
1. PrivateObject obj = new PrivateObject(DM);
```

then the invocation of the private method, that would be

```
1. List<LPRead> ReadsList = DM.GetReads();
```

if the method were public, becomes

```
1. List<LPRead> ReadsList = (List<LPRead>)obj.Invoke("GetReads");
```