SECURE TEST DRIVEN DEVELOPMENT

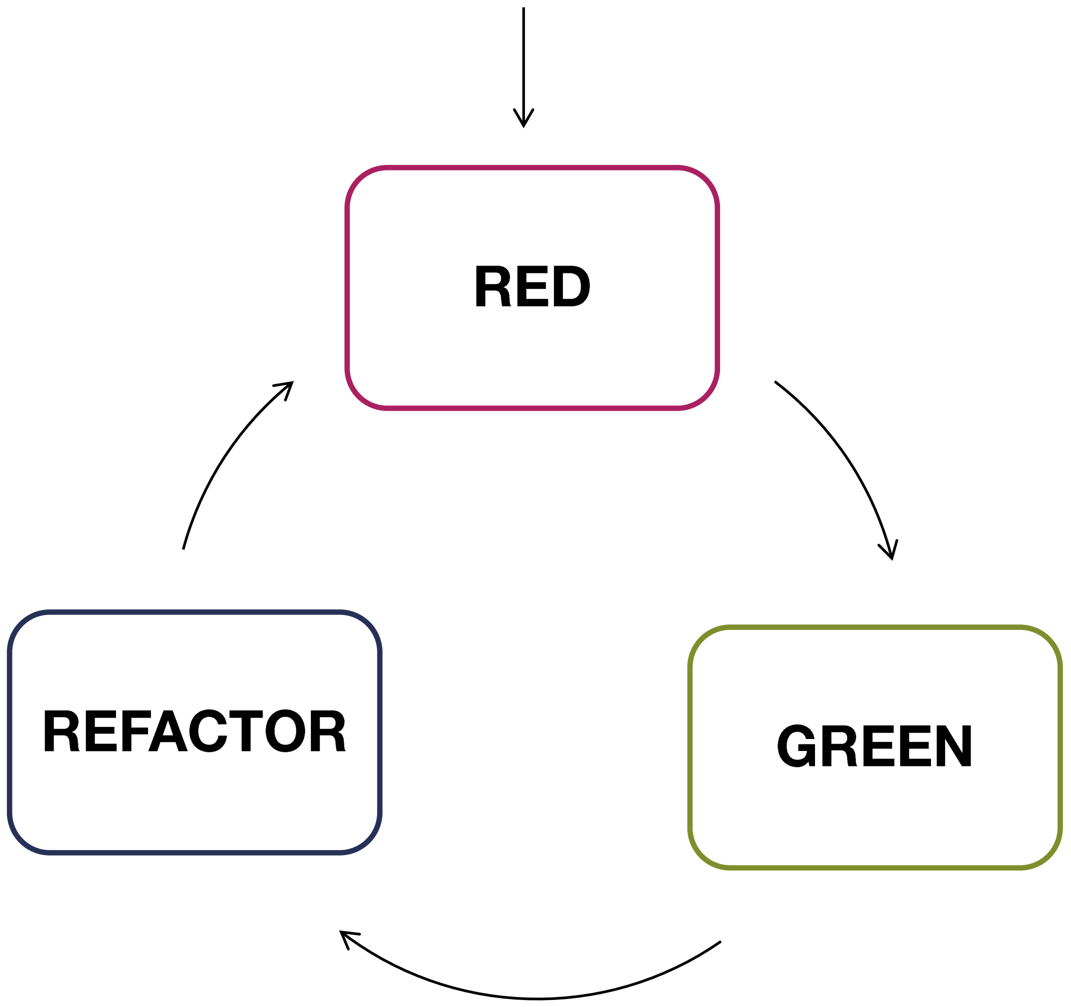
THE SECURITY YOU REALLY NEED

Nir Valtman & Alex Linder

Retalix
A Division of NCR Retail

1st October | Retail Future Ready®

WHAT IS TDD?

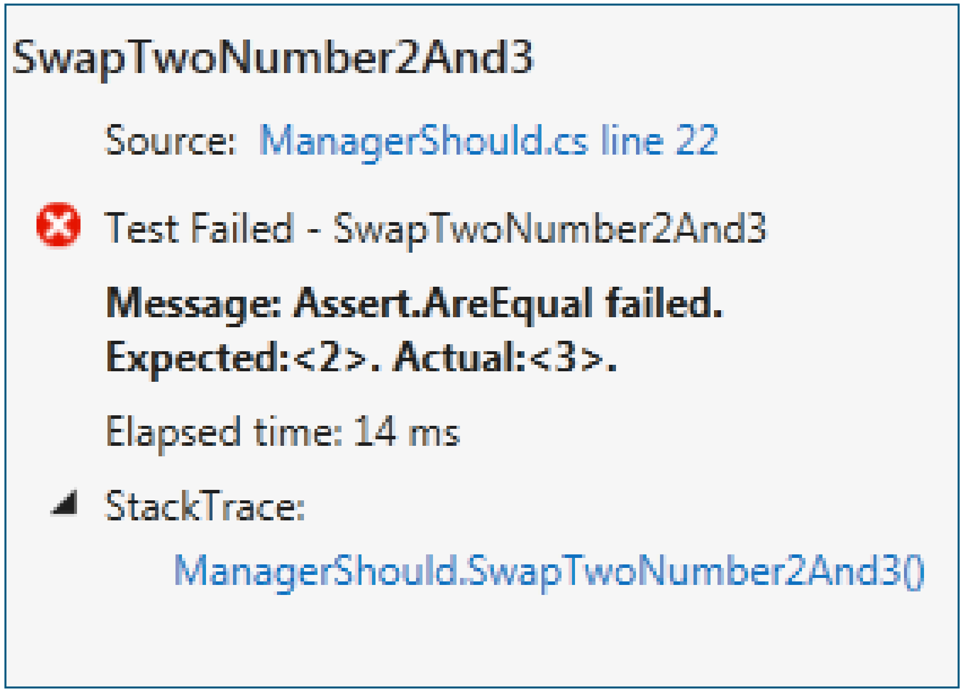

RED
GREEN
REFACTOR

TDD by example – 1st test

```
[TestClass]
public class ManagerShould
{
    private Numbers CreatNumberData(int x, int y)
    {
        Numbers numbers = new Numbers();
        numbers.X = x;
        numbers.Y = y;
        return numbers;
    }

    [TestMethod]
    public void SwapTwoNumber2And3()
    {
        Numbers numbers = CreatNumberData(2, 3);

        new Swap().Numbers(numbers);

        Assert.AreEqual(numbers.X, 3);
        Assert.AreEqual(numbers.Y, 2);
    }
```

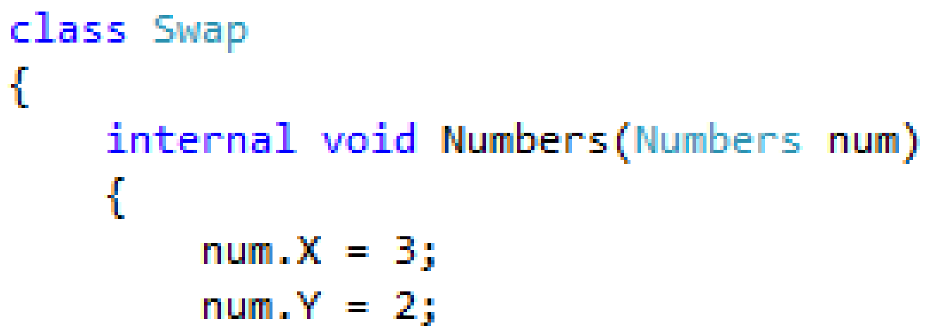
Retalix
A Division of NCR Retail

TDD by example – 1st fail

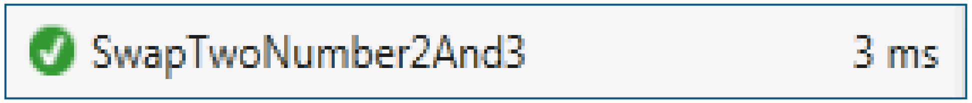


```
SwapTwoNumber2And3

    Source:  ManagerShould.cs line 22

    Test Failed - SwapTwoNumber2And3

    Message: Assert.AreEqual failed.
    Expected:<2>. Actual:<3>.

    Elapsed time: 14 ms

    StackTrace:
        ManagerShould.SwapTwoNumber2And3()
```

Build simple implementation

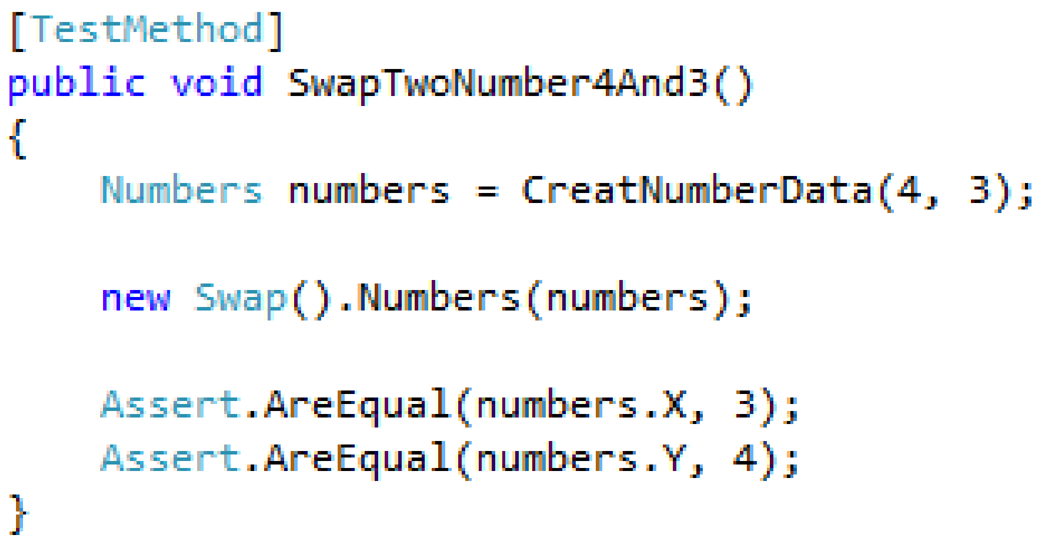


```
class Swap
{
    internal void Numbers(Numbers num)
    {
        num.X = 3;
        num.Y = 2;
```



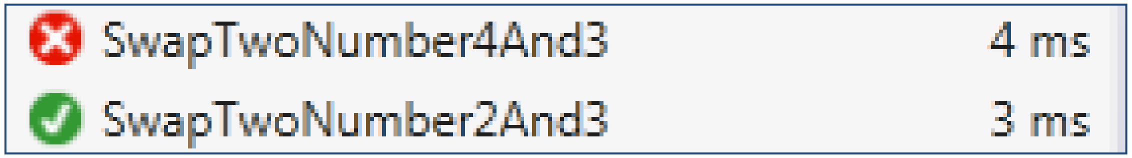
```
✓ SwapTwoNumber2And3                3 ms
```

Add a new test

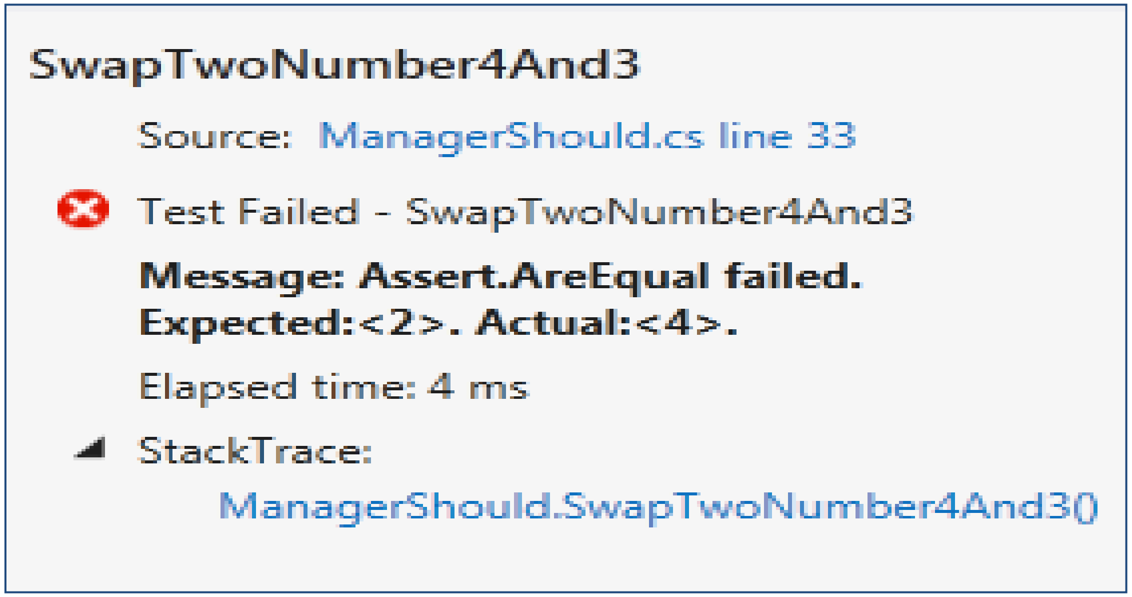


```
[TestMethod]
public void SwapTwoNumber4And3()
{
    Numbers numbers = CreatNumberData(4, 3);

    new Swap().Numbers(numbers);

    Assert.AreEqual(numbers.X, 3);
    Assert.AreEqual(numbers.Y, 4);
}
```

Fail again



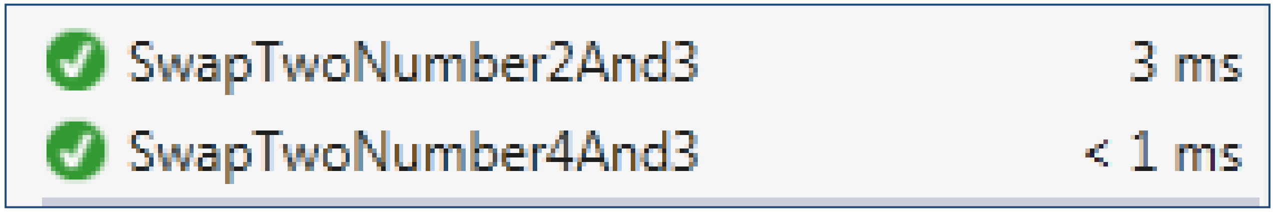
```
✗ SwapTwoNumber4And3    4 ms
✓ SwapTwoNumber2And3    3 ms
```



```
SwapTwoNumber4And3
    Source:  ManagerShould.cs line 33
  ✗ Test Failed - SwapTwoNumber4And3
    Message: Assert.AreEqual failed.
    Expected:<2>. Actual:<4>.
    Elapsed time: 4 ms
  ◢ StackTrace:
        ManagerShould.SwapTwoNumber4And3()
```

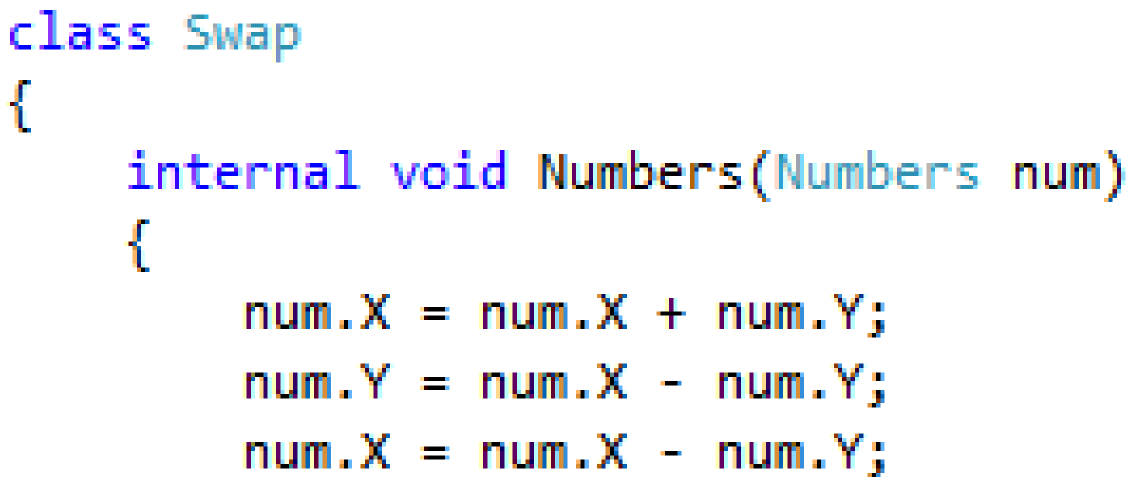
Fix the code

```
class Swap
{
    internal void Numbers(Numbers num)
    {
        int temp = num.X;
        num.X = num.Y;
        num.Y = temp;
```

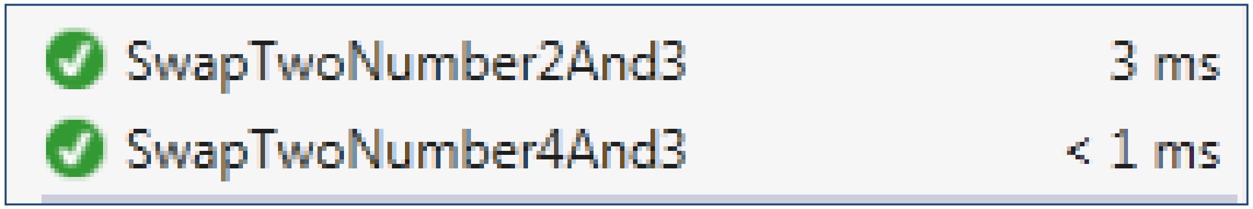


```
✓ SwapTwoNumber2And3      3 ms
✓ SwapTwoNumber4And3    < 1 ms
```

Refactor after NFR



```
class Swap
{
    internal void Numbers(Numbers num)
    {
        num.X = num.X + num.Y;
        num.Y = num.X - num.Y;
        num.X = num.X - num.Y;
```



```
✓ SwapTwoNumber2And3      3 ms
✓ SwapTwoNumber4And3    < 1 ms
```

LET'S TALK ABOUT STDD

MEET THE ACTORS

Happy developer

Security officer

... not that happy

One moment before we begin…

We develop a 3-tier social application

PL

→

BL

→

DAL

Business requirement #1
The system should be able to authenticate users

Foo authentication test

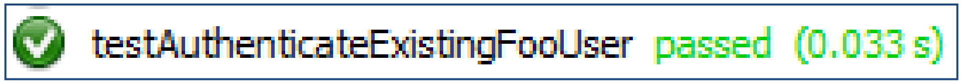
```
@Test
public void testAuthenticateExistingFooUser() {
    AuthenticationClaim authClaim =
            new AuthenticationClaim("foo", "bar");
    Identity result = null;
    try {
        IDM idm = new IDM();
        result = idm.Authenticate(authClaim);
    } catch (SQLException ex) {
        fail("Could not authenticate the user foo");
    }
    assertNotNull(result);
}
```

⚠ testAuthenticateExistingUser Failed: Could not authenticate the user foo

Foo Authentication implementation

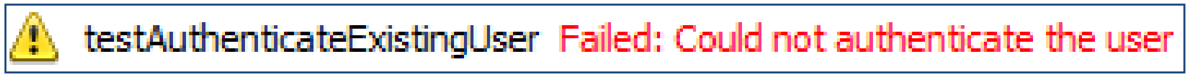
Business Logic

```
public Identity Authenticate(AuthenticationClaim authClaim)
        throws SQLException {
    if (authClaim.getUsername().equals("foo") &&
            authClaim.getPassword().equals("bar"))
       return new Identity("foo", "roleName");
    return null;
}
```


testAuthenticateExistingFooUser passed (0.033 s)

Authentication test

```
@Test
public void testAuthenticateExistingUser() {
    AuthenticationClaim authClaim =
            new AuthenticationClaim(testUserName, testPassword);
    Identity result = null;
    try {
        IDM idm = new IDM();
        result = idm.Authenticate(authClaim);
    } catch (SQLException ex) {
        fail("Could not authenticate the user " + testUserName);
    }
    assertNotNull(result);
}
```

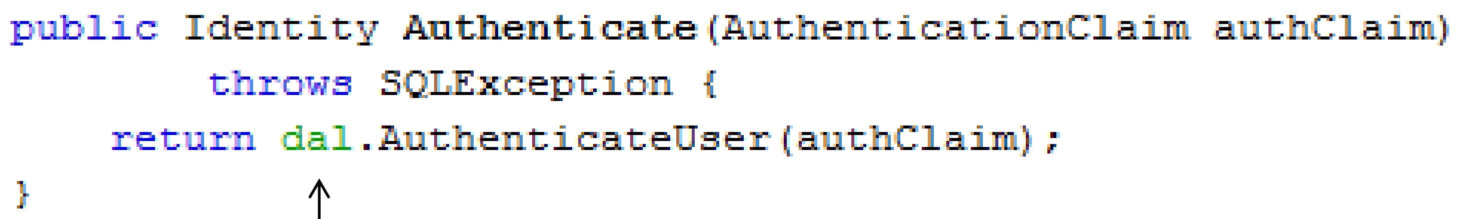


```
testAuthenticateExistingUser Failed: Could not authenticate the user
```

Retalix
A Division of NCR Retail

Authentication implementation

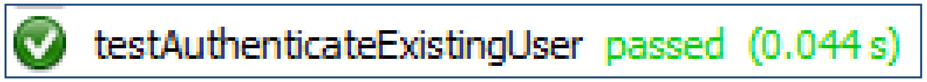
Business Logic



```
public Identity Authenticate(AuthenticationClaim authClaim)
        throws SQLException {
    return dal.AuthenticateUser(authClaim);
}
```

↑

We don't care how DAL works!



testAuthenticateExistingUser passed (0.044 s)

STOP
COLLABORATE AND LISTEN

Password hash test preparations

Let’s create our testing DAL

```
public class InjectedDAL implements IDAL {

    public AuthenticationClaim AuthClaim;
    String testUserName = "foo";
    String testPassword = "bar";

    @Override
    public Identity AuthenticateUser(AuthenticationClaim authClaim)
            throws SQLException {
        AuthClaim = authClaim;
        if (authClaim.getUsername().equals(testUserName) &&
            authClaim.getPassword().equals(DigestUtils.sha256Hex(testPassword)))
                return new Identity(testUserName, "DefaultRole");
        return null;
    }
```

Retalix
A Division of NCR Retail

Password hash BL test

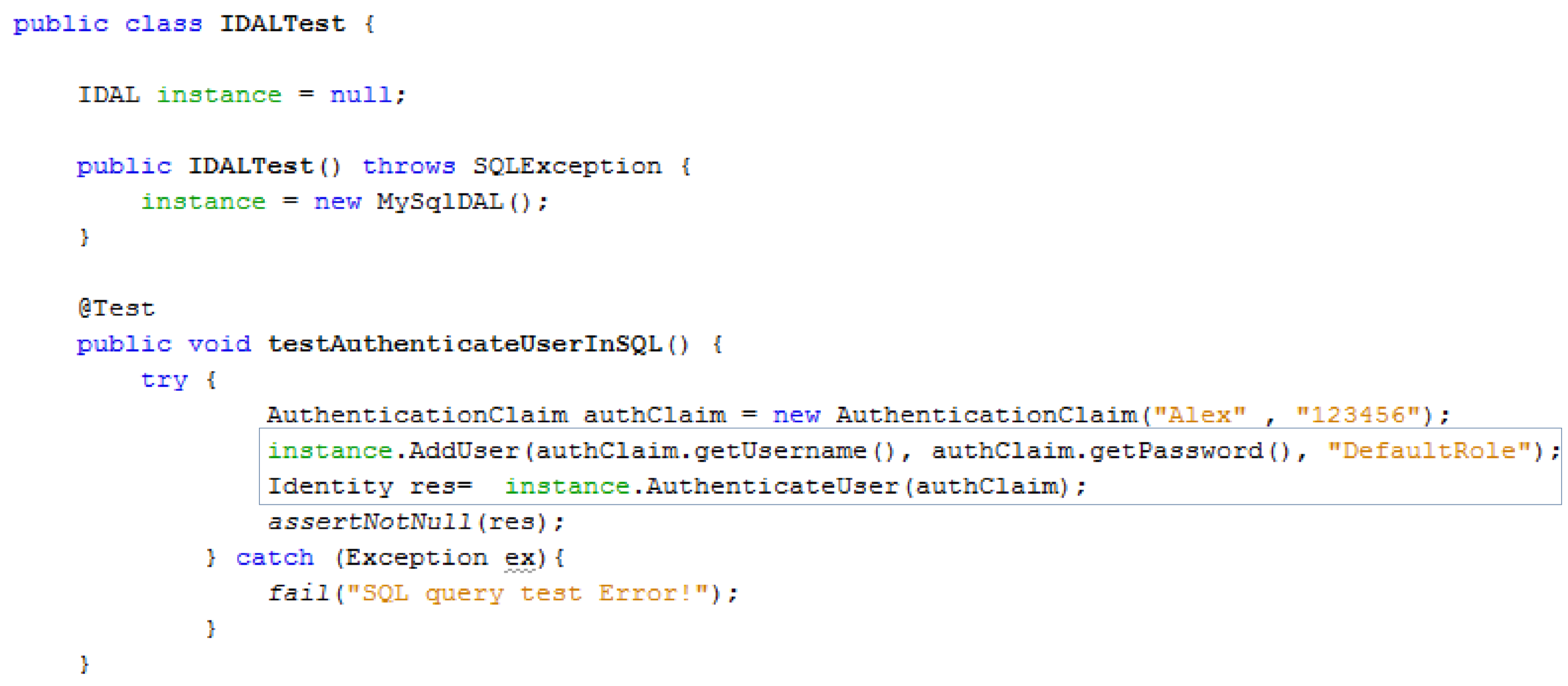
```
@Test
public void testHashPasswordsInDB() {
    InjectedDAL injectedDAL = new InjectedDAL();
    Identity identity = null;
    try {
        AuthenticationClaim websiteClaim =
                new AuthenticationClaim(testUserName, testPassword);
        IDM injectedIdm = new IDM();
        injectedIdm.SetDAL(injectedDAL);
        identity = injectedIdm.Authenticate(websiteClaim);
    } catch (SQLException ex) {
        fail("Test failed: sql exception");
    }
    assertNotNull(identity);
}
```

DEMO

Retalix
A Division of NCR Retail

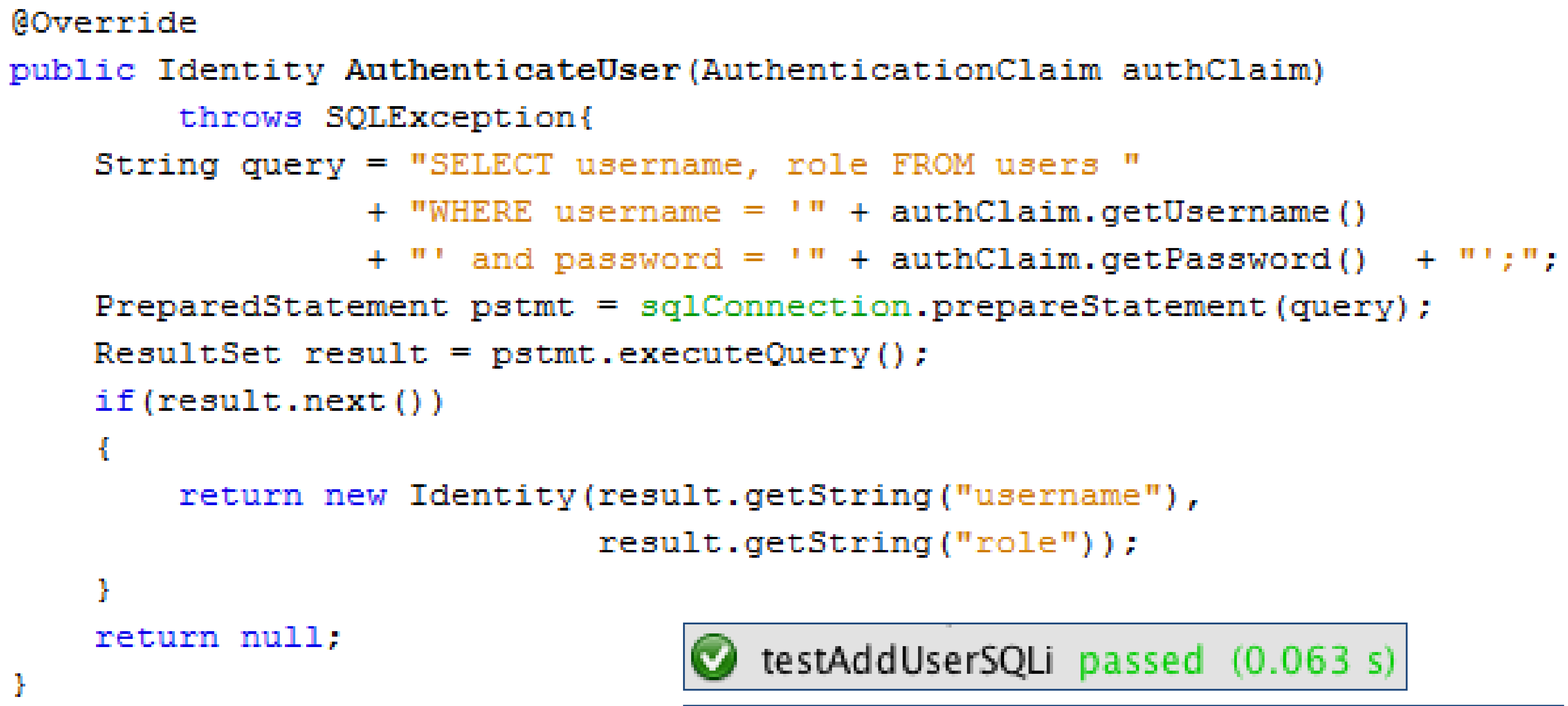
Business requirement #2
User data should be persistent

MySQL data access test

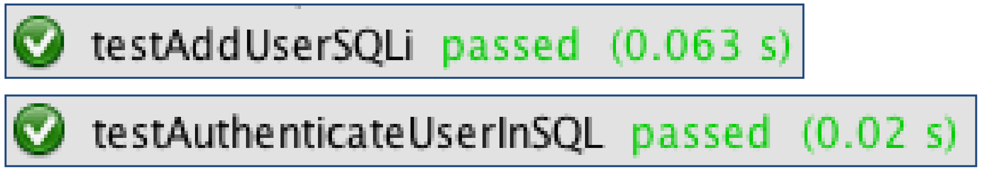


```
public class IDALTest {

    IDAL instance = null;

    public IDALTest() throws SQLException {
        instance = new MySqlDAL();
    }

    @Test
    public void testAuthenticateUserInSQL() {
        try {
                AuthenticationClaim authClaim = new AuthenticationClaim("Alex" , "123456");
                instance.AddUser(authClaim.getUsername(), authClaim.getPassword(), "DefaultRole");
                Identity res=  instance.AuthenticateUser(authClaim);
                assertNotNull(res);
            } catch (Exception ex){
                fail("SQL query test Error!");
            }
    }
```

MySQL data access implementation



```
@Override
public Identity AuthenticateUser(AuthenticationClaim authClaim)
        throws SQLException{
    String query = "SELECT username, role FROM users "
                 + "WHERE username = '" + authClaim.getUsername()
                 + "' and password = '" + authClaim.getPassword()  + "';";
    PreparedStatement pstmt = sqlConnection.prepareStatement(query);
    ResultSet result = pstmt.executeQuery();
    if(result.next())
    {
        return new Identity(result.getString("username"),
                            result.getString("role"));
    }
    return null;
}
```

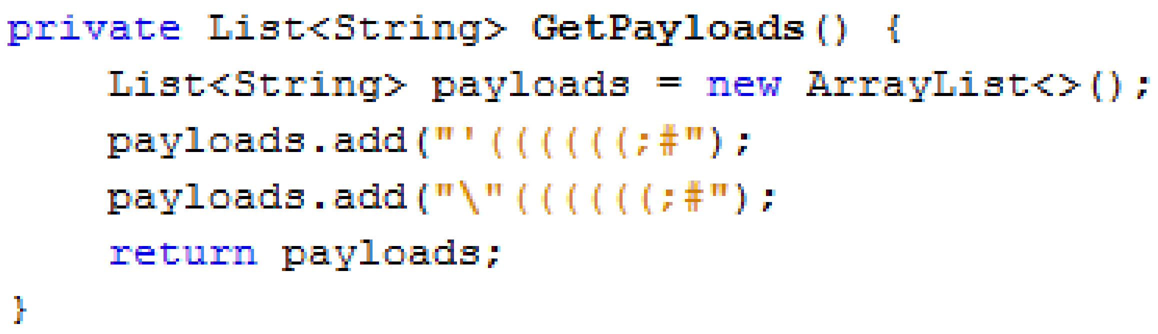


```
✓ testAddUserSQLi passed (0.063 s)
✓ testAuthenticateUserInSQL passed (0.02 s)
```

STOP!

SQL Injection test preparations

Let’s create SQL exception payloads



```
private List<String> GetPayloads() {
    List<String> payloads = new ArrayList<>();
    payloads.add("'((((((;#");
    payloads.add("\"((((((;#");
    return payloads;
}
```

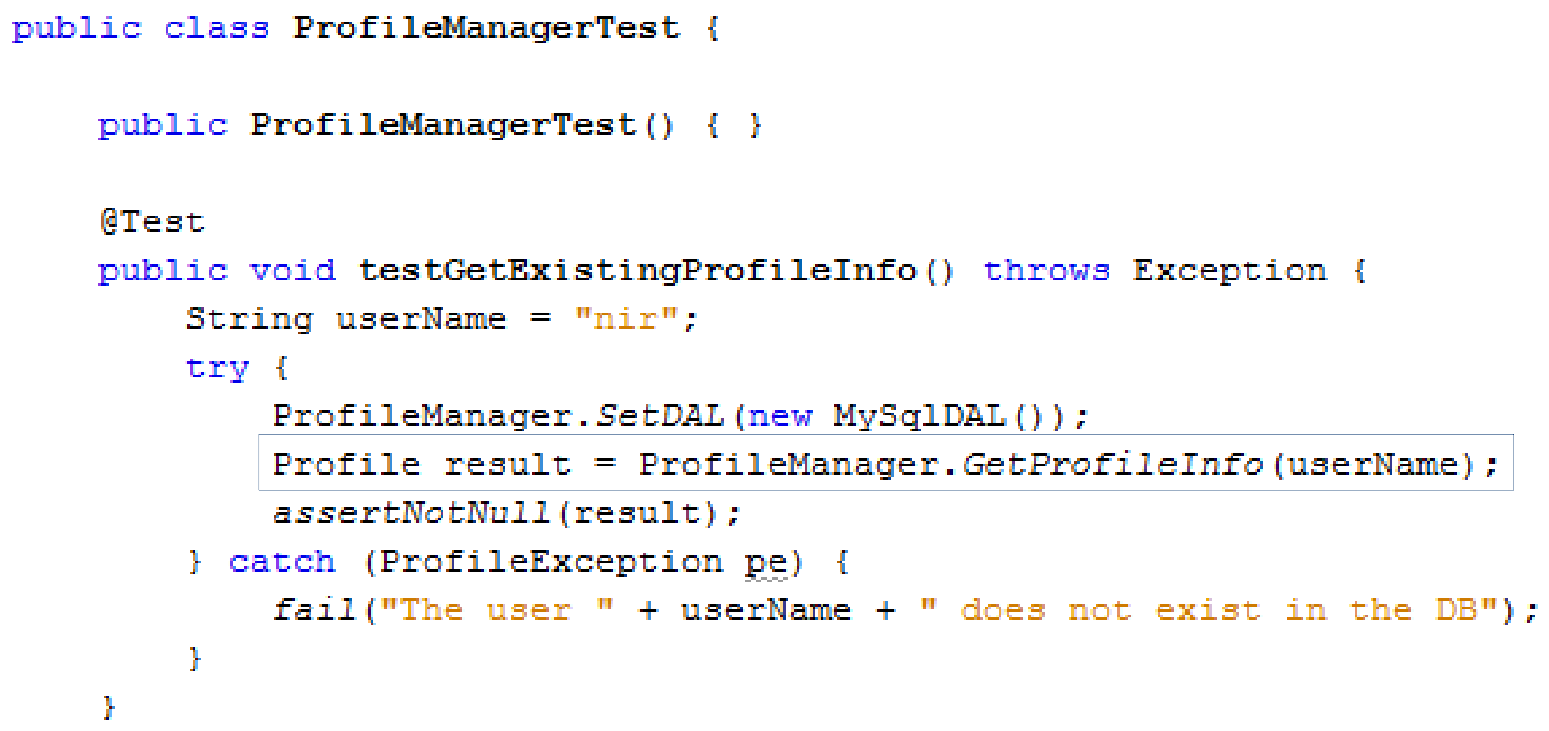
SQL Injection test

```
@Test
public void testAuthenticateUserSQLi() {
    List<String> payloads = GetPayloads();
    for (String payload : payloads) {
        try {
            AuthenticationClaim authClaim = new AuthenticationClaim(payload, payload);
            instance.AuthenticateUser(authClaim);
        } catch (Exception ex){
            fail("SQL injection found by running the payload " + payload);
        }
    }
    assertTrue(true);
}
```

DEMO

Business requirement #3
User profile should be public and accessible by all users

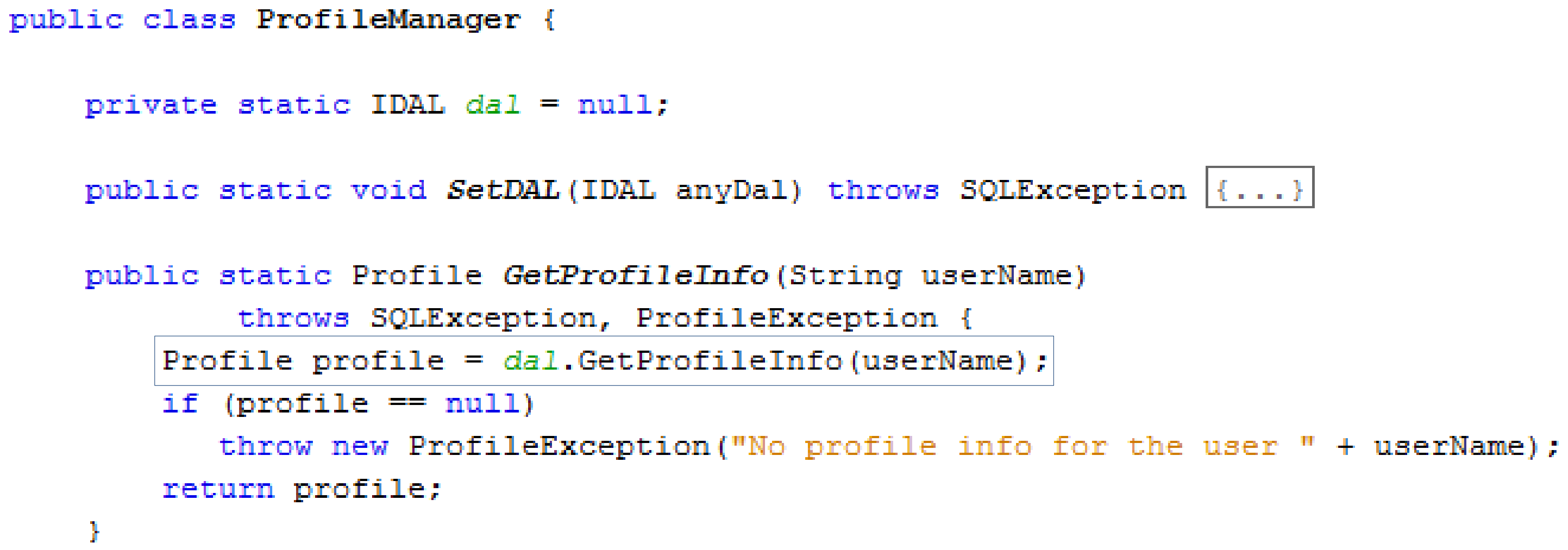
View profile test



```
public class ProfileManagerTest {

    public ProfileManagerTest() { }

    @Test
    public void testGetExistingProfileInfo() throws Exception {
        String userName = "nir";
        try {
            ProfileManager.SetDAL(new MySqlDAL());
            Profile result = ProfileManager.GetProfileInfo(userName);
            assertNotNull(result);
        } catch (ProfileException pe) {
            fail("The user " + userName + " does not exist in the DB");
        }
    }
```

Retalix
A Division of NCR Retail

View profile implementation



```
public class ProfileManager {

    private static IDAL dal = null;

    public static void SetDAL(IDAL anyDal) throws SQLException {...}

    public static Profile GetProfileInfo(String userName)
            throws SQLException, ProfileException {
        Profile profile = dal.GetProfileInfo(userName);
        if (profile == null)
           throw new ProfileException("No profile info for the user " + userName);
        return profile;
    }
```

Retalix
A Division of NCR Retail

DAD! STOP!
....IT"S NOT FUNNY!!!
Troll.me

XSS simple test

```
@Test
public void testGetProfileInfoHtmlEncodedError() throws Exception {
    String userNameXssLocator = "'';!--\"<XSS>=&{()}";
    try {
        ProfileManager.SetDAL(new MySqlDAL());
        ProfileManager.GetProfileInfo(userNameXssLocator);
        fail("The user " + userNameXssLocator + " exist in the DB");
    } catch (ProfileException pe) {
        if (pe.getMessage().contains(userNameXssLocator))
            fail("The user " + userNameXssLocator + " is not HTML encoded");
    }
    assertTrue(true);
}
```

DEMO

Retalix
A Division of NCR Retail

STDD

All code is vulnerable until proven secure

THANK YOU