


# Mock private method call with PowerMock

 [automationrhapsody.com/mock-private-method-call-powermock](http://automationrhapsody.com/mock-private-method-call-powermock)

By Lyudmil Latinov

May 28, 2017

## Post summary: How to mock private method with PowerMock by using spy object.

This post is part of [PowerMock series examples](#). The code shown in examples below is available in GitHub [java-samples/junit](#) repository.

### Mock private method

In some cases, you may need to alter the behavior of private method inside the class you are unit testing. You will need to mock this private method and make it return what needed for the particular case. Since this private method is inside your class under test then mocking it is little more specific. You have to use spy object.

### Spy object

A spy is a real object which mocking framework has access to. Spied objects are partially mocked objects. Some their methods are real some mocked. I would say use spy object with great caution because you do not really know what is happening underneath and whether are you actually testing your class or mocked version of it.

### Code to be tested

Below is a simple code that has a private method which created new Point object based on given as argument one. This private method is used to demonstrate how private methods can be called in [Call private method with PowerMock](#) post. In the current example, there is also a public method which calls this private method with a Point object.

```
public class PowerMockDemo {
    public Point callPrivateMethod() {
        return privateMethod( new Point( 1 , 1 ));
    }
    private Point privateMethod(Point point) {
        return new Point(point.getX() + 1 , point.getY() + 1 );
    }
}
```

### Unit test

What we want to achieve in the unit test is to mock private method so that each call to it returns an object we have control over. The first thing to do is to annotate unit test with `@RunWith(PowerMockRunner.class)` telling JUnit to use PowerMock runner and with `@PrepareForTest(PowerMockDemo.class)` telling PowerMock to get inside **PowerMockDemo** class and prepare it for mocking. Then a spy object has to be

created with **`PowerMockito.spy(new PowerMockDemo())`**). Actually, this is real **`PowerMockDemo`** object, but PowerMock is spying on it. The mocking of the private method is done with following code: **`PowerMockito.doReturn(mockPoint).when(powerMockDemoSpy, "privateMethod", anyObject())`**. When **`"privateMethod"`** is called with whatever object then return **`mockPoint`** which is actually a mocked object. The full code example is shown below:

```
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.powermock.api.mockito.PowerMockito;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.mockito.Matchers.anyObject;
import static org.mockito.Mockito.mock;
@RunWith (PowerMockRunner. class )
@PrepareForTest (PowerMockDemo. class )
public class PowerMockDemoTest {
    private PowerMockDemo powerMockDemoSpy;
    @Before
    public void setUp() {
        powerMockDemoSpy = PowerMockito.spy( new PowerMockDemo());
    }
    @Test
    public void testMockPrivateMethod() throws Exception {
        Point mockPoint = mock(Point. class );
        PowerMockito.doReturn(mockPoint)
            .when(powerMockDemoSpy, "privateMethod" , anyObject());
        Point actualMockPoint = powerMockDemoSpy.callPrivateMethod();
        assertThat(actualMockPoint, is(mockPoint));
    }
}
```

## Conclusion

---

PowerMock provides a way to mock private methods by using spy objects. Mockito also has spy objects, but they are not so powerful as PowerMock's. One example is that PowerMock can spy on final objects.

## Related Posts

---

[PowerMock examples and why better not to use them](#)

Category: [Java](#), [Unit testing](#) | Tags: [JUnit](#), [PowerMock](#)