**Testing private members or methods.**

In computer science, reflection is the ability of a computer program to examine (see type introspection) and modify the structure and behaviour (specifically the values, meta-data, properties and functions) of an object at runtime.

Java's Reflection API's makes it possible to inspect classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It is also possible to instantiate new objects, invoke methods and get/set field values using reflection.

<http://tutorials.jenkov.com/java-reflection/index.html>

<http://java.sun.com/developer/technicalArticles/ALT/Reflection/>

So, is there a way to test private and protected fields and methods without altering the actual class in any way ?  
Yes there is. And the solution lies with Reflection, the same thing that drives JUnit.  
  
Using Java reflection, you can access and even alter private and protected fields of any class. You can effectively break encapsulation with this in a very naughty way. I guess we shouldn't go about using this in actual applications, but for unit testing our own classes, it will come in quite handy, and the code is very simple, just a few lines.

<http://www.jroller.com/CoBraLorD/entry/junit_testing_private_fields_and>

**import static** org.junit.Assert.\*;  
  
**import** org.junit.Test;  
  
**import** java.lang.reflect.Field;  
**import** java.lang.reflect.Method;  
**import static** org.junit.Assert.\*;  
  
**public class** RobotPrivateTest {  
  
 */\*\*  
 \* Accessing a private member  
 \* Test of age member, of class Robot.  
 \*/*

@Test  
 **public void** testSetAge() **throws** Exception {  
 System.***out***.println(**"testsetAge"**);  
 Robot target = **new** Robot(**"robbie"**, 5);  
 Class secretClass = target.getClass();  
  
 */\*Retrieve the field age \*/* Field f = secretClass.getDeclaredField(**"age"**);

*/\* make sure the field is accessible. \*/* f.setAccessible(**true**);

*/\* get the value of the field \*/* System.***out***.println(**"The value in f (age) is "** + f.get(target));

**int** result = f.getInt(target);  
 *assertEquals*(**"The ages should be equal"**, 5, result);  
 }

*/\*\*  
 \* Accessing a private member  
 \* Test of setAge method, of class robot.  
 \*/*

@Test  
 **public void** testsetAge1() **throws** Exception {  
 System.***out***.println(**"setAge1"**);  
 */\* Set up a new Robot \*/* Robot target = **new** Robot(**"robbie"**, 3);  
  
 */\* get the setAge method details \*/* Method method = Robot.**class**.getDeclaredMethod(**"setAge"**, **int**.**class**);  
  
 */\* make the method assessible \*/* method.setAccessible(**true**);  
  
 */\* invoke the method setAge with the value 7 \*/* method.invoke(target, 7);  
  
 */\*access the field age and check its value is set to 7\*/* Class secretClass = target.getClass();

Field f = secretClass.getDeclaredField(**"age"**);

f.setAccessible(**true**);

**int** result = f.getInt(target);  
 System.***out***.println(**"The value in f (age) is "** + f.get(target));  
 *assertEquals*(**"The ages should be equal"**, 7, result);  
 }  
}

**Exercise**

In the driver class write a test to test the setDriverNum method.