# Mockito

Mockito

- It is built on the top of JUnit Tool.
- It is given to perform unit testing by mocking the Local Dependent or external Dependent objects.

```
Service class --------------------------> DAO class --------------> DB s/w
        I->business methods              |-> Persistence methods
        (having business logic)          (persistence logic)
```

- Let us assume DAO class coding is not completed, but we Service coding is completed and we want finish unit testing of service class. Then we need to create Mock object/ Fake object/ dummy object for DAO class and assign/ inject to Service class, in order write test cases on service class methods.
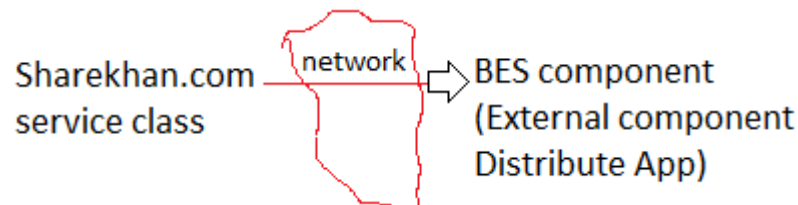
- When Sharekhan.com website is under development, we cannot take subscription of BSE component because they charge money for that. Generally, this subscription will take after hosting/ releasing the Sharekhan.com till that we need to take one mock BSE component and assign to Service class of Sharekhan.com to perform Unit Testing.
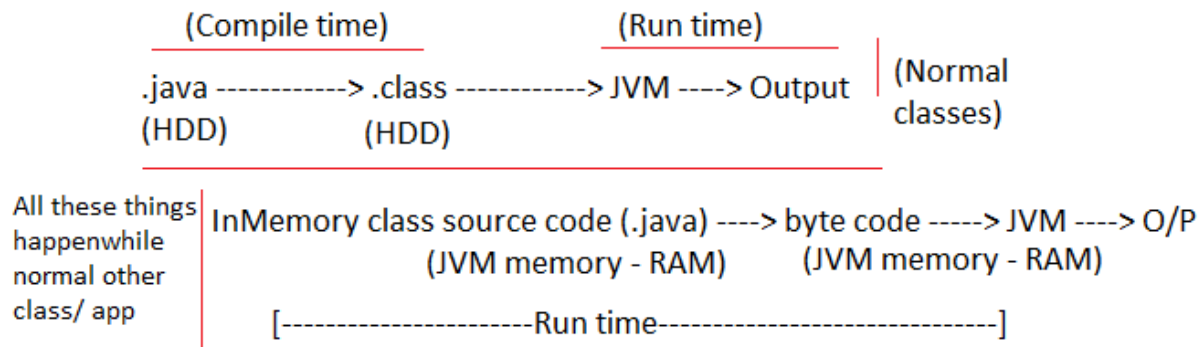
Note: Mock objects are for temporary needs, mostly they will be used in the Unit Testing. These mock objects created in test methods or Test case class does not real code.

We can do this Mocking in 3 ways:
   a. Using Mock object/ Fake object (Provides Temporary object)
   b. Using Stub object (Providing some Functionality for the methods of mock object like specifying for certain inputs or certain output should come)
   c. Using Spy object (It is called Partial Mock object/ Half mock object that means if you provide new functionality to method that will be used otherwise real object functionality will be used).

   Note: While working with Spy object will be having real object also.

✚ Instead of creating classes manually to prepare Mock, Stub and Spy objects, we can use mocking frameworks available in the market which are capable generate such classes dynamically at runtime as InMemory classes (That classes that are generated in the JVM memory of RAM).

```
        (Compile time)                (Run time)
        _____                   _____
   .java -----------> .class -----------> JVM ----> Output | (Normal
   (HDD)            (HDD)                                   | classes)
   _____

All these things | InMemory class source code (.java) ----> byte code -----> JVM ----> O/P
happenwhile      |             (JVM memory - RAM)                   (JVM memory - RAM)
normal other     |
class/ app       |    [---------------------Run time----------------------------]
```

List of Mocking Frameworks:
- Mockito (popular)
- JMockito
- EasyMock
- PowerMock
  And etc.

Example Application setup:
[Testing LoginMgmtService class without keeping LoginDAO class ready]

Step 1: Create maven standalone App
        File -> Maven Project -> next -> select maven-archetype-quickstart ->
                GroupId: nit
                ArtifactId: MockitoUniTesting-LoginApp
                Default package: com.nt.service

Step 2: Do following operations in pom.xml file
        change java version to 13
        Add the following dependencies (jars)
            - junit-jupiter-api.5.7.0.jar
            - junit-jupiter-engine.5.7.0.jar
            - mockito-core.3.6.28.jar

Step 3: Develop service interface, service class
        com.nt.service
                |-> ILoginMgmtService.java (I)

```
              |-> LoginMgmtServiceImpl.java (c)
        com.nt.dao
              |-> LoginDAO.java
```

Step 4: Develop Mockito based Test classes and DAO interface
Step 5: Run Tests.

Directory Structure of JUnitTestProject01:

```
v 🗂 MockitoUnitTesting-LoginApp
   v 🗁 src/main/java
      v ⊞ com.nt.dao
         > 📄 ILoginDAO.java
      v ⊞ com.nt.service
         > 📄 ILoginMgmtService.java
         > 📄 LoginMgmtServiceImpl.java
   v 🗁 src/test/java
      v ⊞ com.nt.test
         > 📄 TestLoginMgmtService.java
   > 🗄 JRE System Library [JavaSE-13]
   > 🗄 Maven Dependencies
   > 📂 src
   > 📂 target
     📄 pom.xml
```

- Develop the above directory Structure and package, class, XML file and add the jar dependencies in pom.xml file then use the following code with in their respective file.

pom.xml

```xml
    <dependencies>
            <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-
    jupiter-api -->
            <dependency>
                    <groupId>org.junit.jupiter</groupId>
                    <artifactId>junit-jupiter-api</artifactId>
                    <version>5.7.0</version>
                    <scope>test</scope>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-
    jupiter-engine -->
            <dependency>
                    <groupId>org.junit.jupiter</groupId>
                    <artifactId>junit-jupiter-engine</artifactId>
```

```xml
                    <version>5.7.0</version>
                    <scope>test</scope>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.mockito/mockito-
core -->
            <dependency>
                    <groupId>org.mockito</groupId>
                    <artifactId>mockito-core</artifactId>
                    <version>3.6.28</version>
                    <scope>test</scope>
            </dependency>
    </dependencies>
```

ILoginDAO.java

```java
package com.nt.dao;

public interface ILoginDAO {
        public int authenticate(String username, String password);
}
```

ILoginMgmtService.java

```java
package com.nt.service;

public interface ILoginMgmtService {
        public boolean login(String username, String password);
}
```

LoginMgmtServiceImpl.java

```java
package com.nt.service;

import com.nt.dao.ILoginDAO;

public class LoginMgmtServiceImpl implements ILoginMgmtService {

        private ILoginDAO loginDAO;

        public LoginMgmtServiceImpl(ILoginDAO loginDAO) {
                this.loginDAO = loginDAO;
        }

        @Override
        public boolean login(String username, String password) {
                if (username.equals("")||password.equals(""))
```

```java
                    throw new IllegalArgumentException("Empty
credentials");
            //use DAO
            int count =loginDAO.authenticate(username, password);
            if (count==0)
                    return false;
            else
                    return true;
    }

}
```

TestLoginMgmtService.java

```java
package com.nt.test;

import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

import com.nt.dao.ILoginDAO;
import com.nt.service.ILoginMgmtService;
import com.nt.service.LoginMgmtServiceImpl;

public class TestLoginMgmtService {

    private static ILoginMgmtService loginSevice;
    private static ILoginDAO loginDAOMock;

    @BeforeAll
    public static void setupOnce() {
        /* Create Mock/ Fake/ Dummy Object
         * mock(-) generates InMemory class implementing
         * ILoginDAO(I) having null method definitions for
         *authenticate(-,-) method */
        loginDAOMock = Mockito.mock(ILoginDAO.class);
        // Create Service class object
        loginSevice = new LoginMgmtServiceImpl(loginDAOMock);
    }
```

```java
        @AfterAll
        public static void clearOnce() {
                loginDAOMock = null;
                loginSevice = null;
        }

        //Test Methods
        @Test
        public void testLoginWithValidCredentials() {
                // Provide stub (Temporary functionality) for DAO's
authenticate method
                Mockito.when(loginDAOMock.authenticate("raja",
"rani")).thenReturn(1);
                assertTrue(loginSevice.login("raja", "rani"));
        }

        @Test
        public void testLoginWithInvalidCredentials() {
                // Provide stub (Temporary functionality) for DAO's
authenticate method
                Mockito.when(loginDAOMock.authenticate("raja",
"rani1")).thenReturn(0);
                assertFalse(loginSevice.login("raja", "rani1"));
        }

        @Test
        public void testLoginWithNoCredentials() {
                assertThrows(IllegalArgumentException.class, ()->{
                        loginSevice.login("", "");
                });
        }

}
```
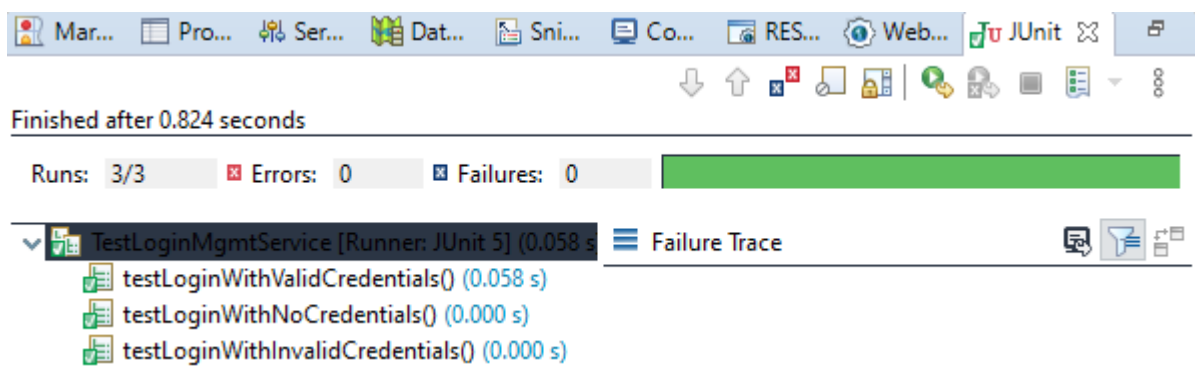
Mar...   Pro...   Ser...   Dat...   Sni...   Co...   RES...   Web...   JUnit

Finished after 0.824 seconds

Runs: 3/3   Errors: 0   Failures: 0

TestLoginMgmtService [Runner: JUnit 5] (0.058 s)   Failure Trace
    testLoginWithValidCredentials() (0.058 s)
    testLoginWithNoCredentials() (0.000 s)
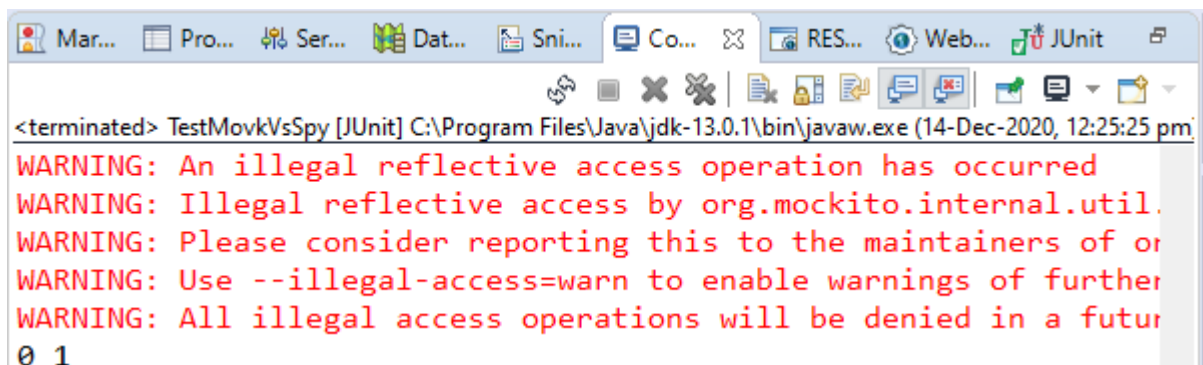   testLoginWithInvalidCredentials() (0.000 s)

TestMockVsSpy.java

```java
package com.nt.test;

import java.util.ArrayList;
import java.util.List;

import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

public class TestMockVsSpy {

    @Test
    public void testList() {
        List<String> listMock = Mockito.mock(ArrayList.class);
        List<String> listSpy = Mockito.spy(new ArrayList());
        listMock.add("table");
        listSpy.add("table");
        System.out.println(listMock.size()+" "+listSpy.size());
    }

}
```

```
Mar...   Pro...   Ser...   Dat...   Sni...   Co...   RES...   Web...   JUnit

<terminated> TestMovkVsSpy [JUnit] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (14-Dec-2020, 12:25:25 pm)
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.mockito.internal.util.
WARNING: Please consider reporting this to the maintainers of or
WARNING: Use --illegal-access=warn to enable warnings of further
WARNING: All illegal access operations will be denied in a futur
0 1
```

Note: Spy objects are useful to check how many time methods are called whether they are called or not. Because Spy object is always linked with real object (for this use Mockito.verify(-,-) method).
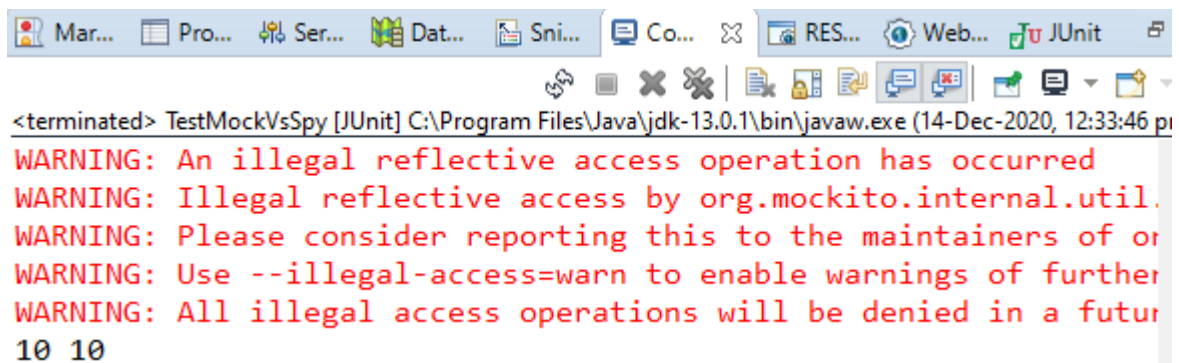
Stubbing on Mock and Spy object:
TestMockVsSpy.java

```java
    @Test
    public void testList() {
        List<String> listMock = Mockito.mock(ArrayList.class);
        //mock
        List<String> listSpy = Mockito.spy(new ArrayList());//spy
```

```java
        listMock.add("table");
        //stubbing on mock
        Mockito.when(listMock.size()).thenReturn(10);
        listSpy.add("table");
        //stubbing on spy
        Mockito.when(listSpy.size()).thenReturn(10);
        System.out.println(listMock.size()+" "+listSpy.size());
    }
```

```
<terminated> TestMockVsSpy [JUnit] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (14-Dec-2020, 12:33:46 p
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.mockito.internal.util.
WARNING: Please consider reporting this to the maintainers of or
WARNING: Use --illegal-access=warn to enable warnings of further
WARNING: All illegal access operations will be denied in a futur
10 10
```

ILoginDAO.java

```java
        public int addUser(String username, String role);
```

ILoginMgmtService.java

```java
        public String registerUser(String username, String role);
```

LoginMgmtServiceImpl.java
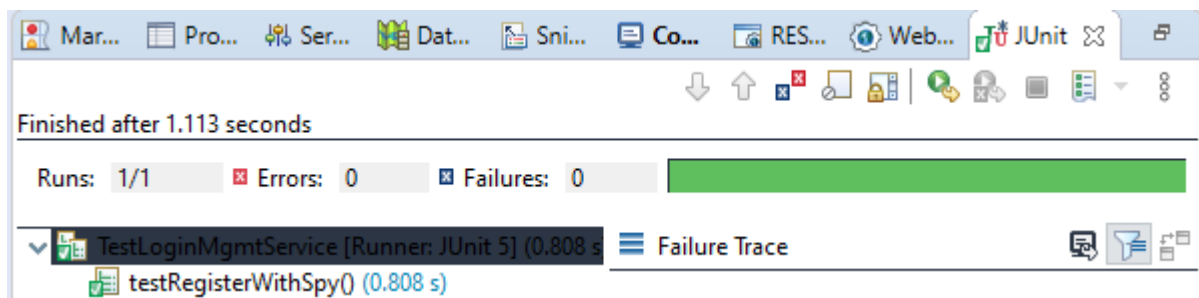
```java
        @Override
        public String registerUser(String username, String role) {
                if
(!role.equalsIgnoreCase("")&&!role.equalsIgnoreCase("visitor")) {
                        loginDAO.addUser(username, role);
                        return "User Added";
                }
                else {
                        return "User Not Added";
                }

        }
```

TestLoginMgmtService.java

```java
    @Test
    public void testRegisterWithSpy() {
            //spy object
            ILoginDAO loginDAOSpy = Mockito.spy(ILoginDAO.class);
            ILoginMgmtService loginService = new
LoginMgmtServiceImpl(loginDAOSpy);
            loginService.registerUser("raja", "admin");
            loginService.registerUser("suresh", "visitor");
            loginService.registerUser("jani", "");
            Mockito.verify(loginDAOSpy, Mockito.times(1)).addUser("raja",
"admin");

            Mockito.verify(loginDAOSpy,
Mockito.times(0)).addUser("suresh", "visitor");
            Mockito.verify(loginDAOSpy, Mockito.never()).addUser("jani",
"");
    }
```

![JUnit icons] Mar... Pro... Ser... Dat... Sni... Co... RES... Web... JUnit	

Finished after 1.113 seconds

Runs: 1/1   Errors: 0   Failures: 0

TestLoginMgmtService [Runner: JUnit 5] (0.808 s   ≡ Failure Trace
        testRegisterWithSpy() (0.808 s)

Mockito Annotations:
- @Mock: To generate mock object
- @Spy: To generate spy object
- @InjectMocks: To Inject Mock or Spy Objects Service class.
- MockitoAnnotations.openMocks(this); - call this method in @Before or constructor Testcase class in order to activate Mockito Annotations.

ILoginMgmtService.java

```java
    @InjectMocks
    private LoginMgmtServiceImpl loginSevice;
    @Mock
    private static ILoginDAO loginDAOMock;
    public AnnoTestLoginMgmtService() {
            MockitoAnnotations.openMocks(this);
    }
```

Note: To write stub functionality according agile user stories/ JIRA user stories (given, when, then)

```
BDDMockito.given(loginDAOMock.authenticate("raja", "rani")).willReturn(1);
```

⇩

```
Mockito.when(loginDAOMock.authenticate("raja", "rani")).thenReturn(1);
```