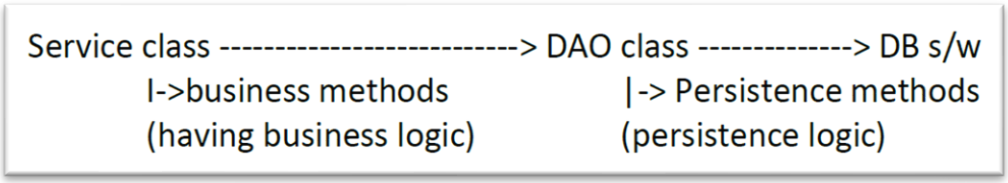
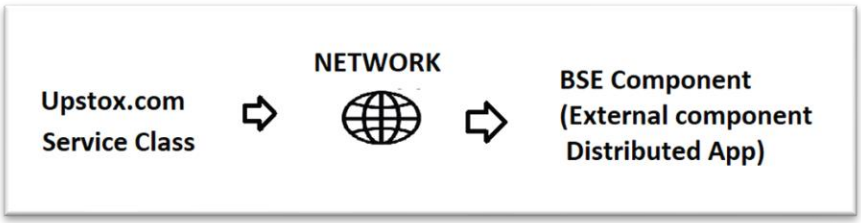
Mockito

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.



- Let us assume DAO class coding is not completed, but 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.



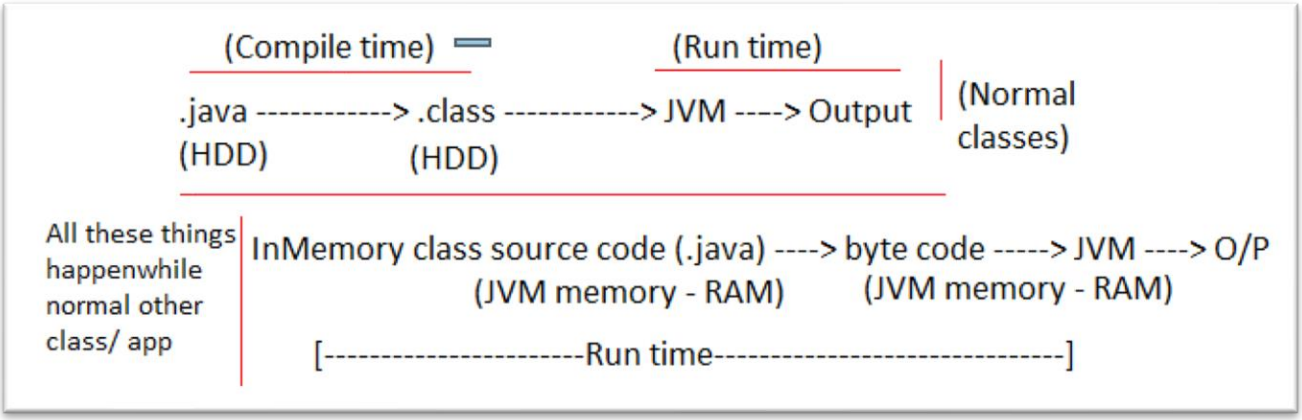
- When **Upstox.com** website is under development, they cannot take subscription of BSE component because they charge money for that. Generally, this subscription will take after hosting / releasing the Upstox.com till that we need to take one mock BSE component and assign to Service class of Upstox.com 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:**
  1. Using Mock object/ Fake object (Provides Temporary object)
  2. Using Stub object (Providing some Functionality for the methods of mock object like specifying for certain inputs or certain output should come)
  3. 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 in-memory classes (That classes that are generated in the JVM memory of RAM).

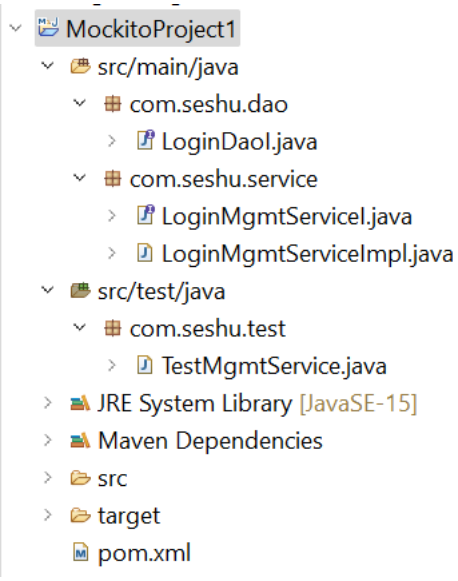


List of Mocking Frameworks:

1. Mockito (popular)
2. JMockito
3. EasyMock
4. PowerMock

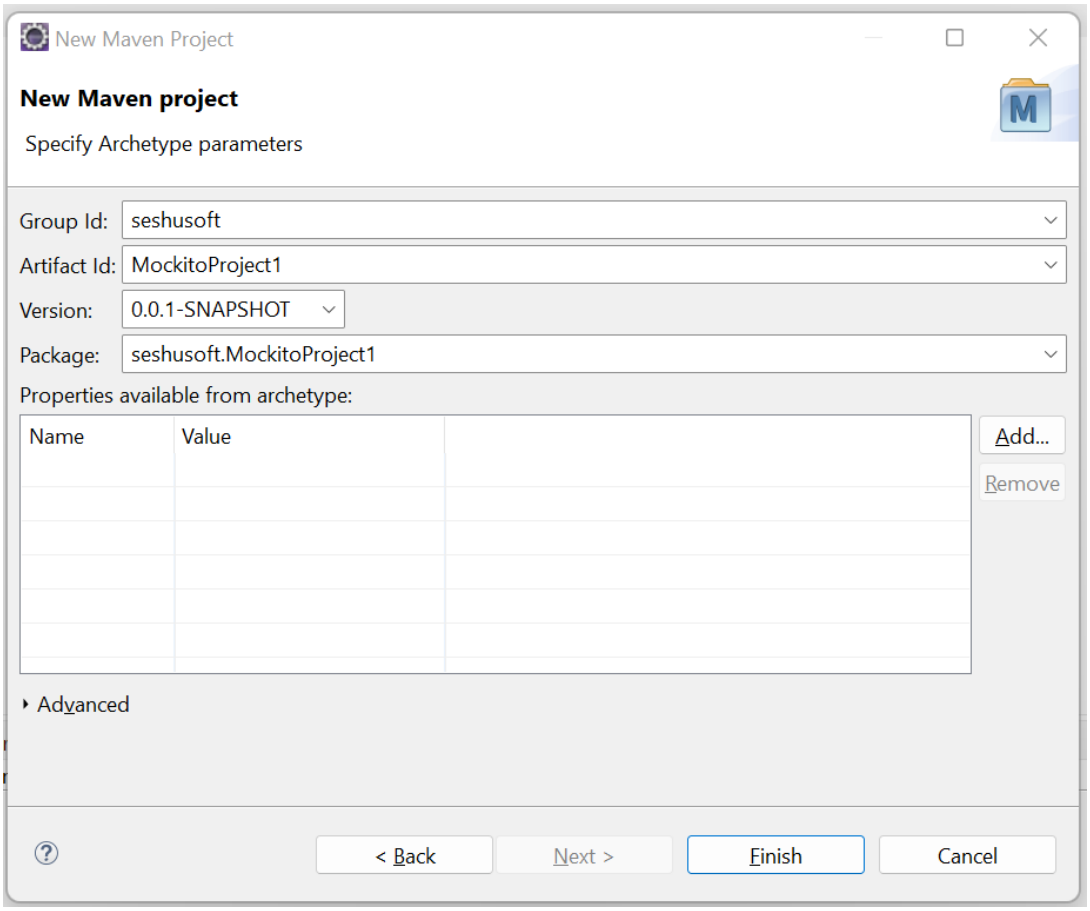
Example Application:

[Testing LoginMgmtService class without keeping LoginDAO class ready]



Step 1: Create maven standalone App

File -> Maven Project -> next -> select maven-archetype-quickstart ->



Step 2:

Open the pom.xml file and change java version to 15

Add the fallowing dependencies.

junit-jupiter-api.5.8.0.jar

junit-jupiter-engine.5.8.0.jar

mockito-core.4.0.0.jar

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>seshusoft</groupId>
	<artifactId>MockitoProject1</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>MockitoProject1</name>
	<url>http://maven.apache.org</url>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<maven.compiler.source>15</maven.compiler.source>
		<maven.compiler.target>15</maven.compiler.target>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.junit.jupiter</groupId>
			<artifactId>junit-jupiter-api</artifactId>
			<version>5.8.0</version>
			<scope>test</scope>
		</dependency>

		<dependency>
			<groupId>org.junit.jupiter</groupId>
			<artifactId>junit-jupiter-engine</artifactId>
			<version>5.8.0</version>
			<scope>test</scope>
		</dependency>

		<dependency>
			<groupId>org.mockito</groupId>
			<artifactId>mockito-core</artifactId>
			<version>4.0.0</version>
			<scope>test</scope>
		</dependency>

	</dependencies>
</project>
```

Step 3: Develop LoginMgmntServiceI interface, LoginMgmntServiceImpl class and LoginDaoI interface.

LoginDaoI.java

```
package com.seshu.dao;

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

LoginMgmtServiceI.java

```
package com.seshu.service;

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

LoginMgmtServiceImpl.java

```
package com.seshu.service;

import com.seshu.dao.LoginDaoI;

public class LoginMgmtServiceImpl implements LoginMgmtServiceI {
	private LoginDaoI loginDao;

	public LoginMgmtServiceImpl(LoginDaoI loginDao) {
		this.loginDao = loginDao;
	}

	public boolean login(String username, String password) {
		if (username.equals("") || password.equals(""))
			throw new IllegalArgumentException("Empty credentials");
		//use DAO
		int count = loginDao.authenticate(username, password);
		if (count == 0)
			return false;
		else
			return true;
	}
}
```

Step 4: Develop Mockito based Test classes and DAO interface

TestMgmtService.java

```
package com.seshu.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.seshu.dao.LoginDaoI;
```

```
import com.seshu.service.LoginMgmtServiceI;
import com.seshu.service.LoginMgmtServiceImpl;

public class TestMgmtService {
	private static LoginMgmtServiceI loginSevice;
	private static LoginDaoI loginDAOMock;

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

	@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("adi", "seshu")).thenReturn(1);
		assertTrue(loginSevice.login("adi", "seshu"));
	}

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

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

Step 5: Run Tests.
Right click on TestMgmtService.java -> Run As -> Junit Test