|  |  |
| --- | --- |
| **WEEK 2 – Mockito Hands-On** | **Superset Id : 6429486**  **Name : Akshaya V** |

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

**Scenario:**

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

**Steps:**

1. Create a mock object for the external API.

2. Stub the methods to return predefined values.

3. Write a test case that uses the mock object.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

} }

**Code:**

**WeatherApi.java:**

package org.example;  
public interface WeatherApi{  
 String getTemperature(String city);  
 String getHumidity(String city);  
 String getCondition(String city);  
}

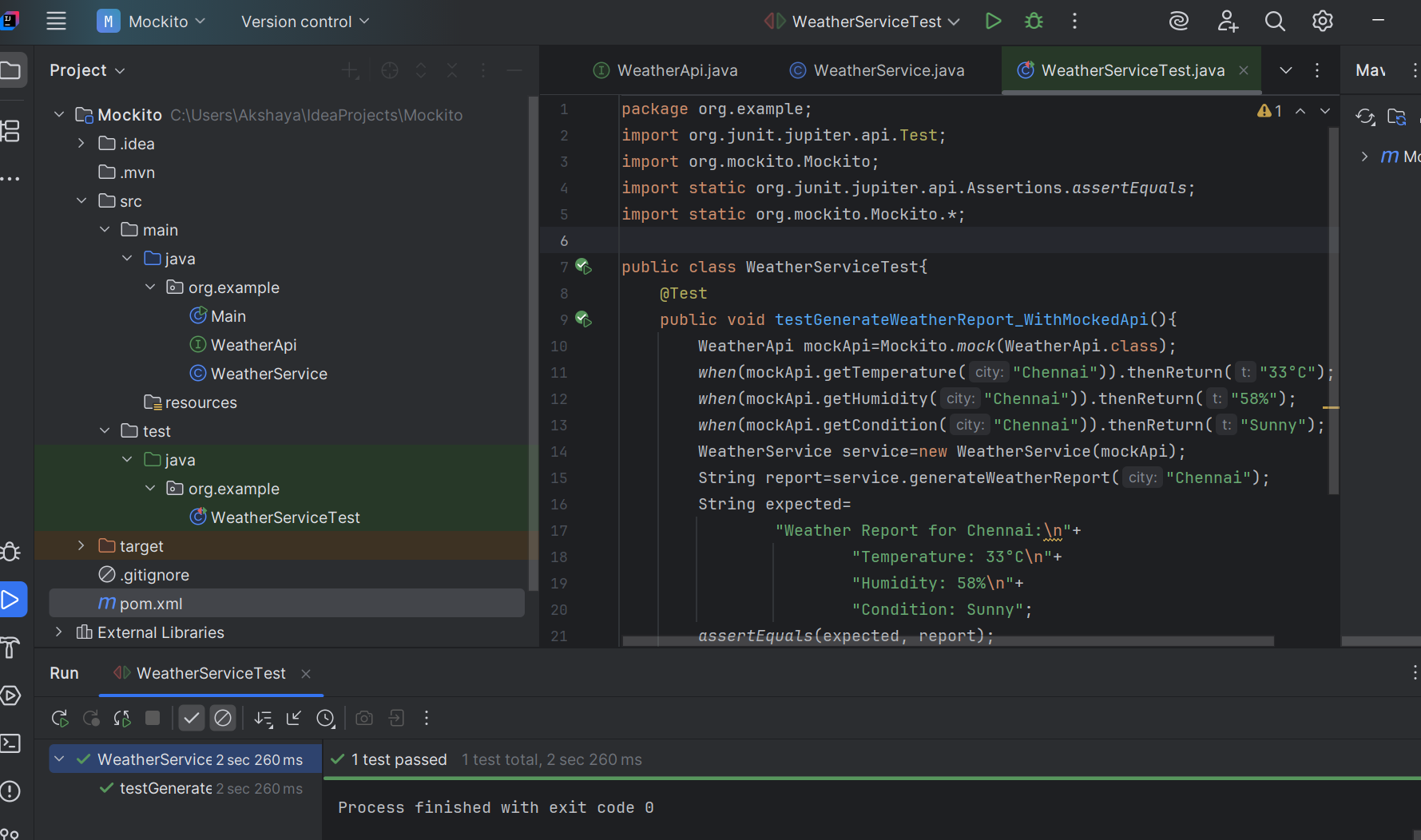
**WeatherService.java:**

package org.example;  
public class WeatherService{  
 private WeatherApi api;  
 public WeatherService(WeatherApi api){  
 this.api=api;  
 }  
 public String generateWeatherReport(String city){  
 String temp=api.getTemperature(city);  
 String humidity=api.getHumidity(city);  
 String condition=api.getCondition(city);  
  
 return "Weather Report for "+city+":\n"  
 + "Temperature: "+temp+"\n"  
 + "Humidity: "+humidity+"\n"  
 + "Condition: "+condition;  
 }  
}

**WeatherServiceTest.java:**

package org.example;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.\*;  
public class WeatherServiceTest{  
 @Test  
 public void testGenerateWeatherReport\_WithMockedApi(){  
 WeatherApi mockApi=Mockito.*mock*(WeatherApi.class);  
 *when*(mockApi.getTemperature("Chennai")).thenReturn("33°C");  
 *when*(mockApi.getHumidity("Chennai")).thenReturn("58%");  
 *when*(mockApi.getCondition("Chennai")).thenReturn("Sunny");  
 WeatherService service=new WeatherService(mockApi);  
 String report=service.generateWeatherReport("Chennai");  
 String expected=  
 "Weather Report for Chennai:\n"+  
 "Temperature: 33°C\n"+  
 "Humidity: 58%\n"+  
 "Condition: Sunny";  
 *assertEquals*(expected, report);  
 }  
}

**Output:**

****

**Exercise 2: Verifying Interactions**

**Scenario:**

You need to ensure that a method is called with specific arguments.

**Steps:**

1. Create a mock object.

2. Call the method with specific arguments.

3. Verify the interaction.

**Solution Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**Code:**

**EmailClient.java:**

package org.example;  
public interface EmailClient{  
 String getStatus(String emailId);  
 boolean sendEmail(String to,String subject,String body);  
 boolean isSpam(String emailId);  
 String deleteEmail(String emailId);  
}

**EmailService.java:**

package org.example;  
public class EmailService{  
 private final EmailClient client;  
 public EmailService(EmailClient client){  
 this.client=client;  
 }  
 public String checkStatus(String emailId){  
 return client.getStatus(emailId);  
 }  
 public boolean send(String to, String subject,String body){  
 return client.sendEmail(to,subject,body);  
 }  
 public boolean isSuspicious(String emailId){  
 return client.isSpam(emailId);  
 }  
 public String removeEmail(String emailId){  
 return client.deleteEmail(emailId);}}

**EmailServiceTest.java:**

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class EmailServiceTest {  
 @Test  
 public void testCheckStatus(){  
 EmailClient mockClient=*mock*(EmailClient.class);  
 *when*(mockClient.getStatus("e123")).thenReturn("SENT");  
  
 EmailService service=new EmailService(mockClient);  
 *assertEquals*("SENT",service.checkStatus("e123"));  
 }  
 @Test  
 public void testSend(){  
 EmailClient mockClient=*mock*(EmailClient.class);  
 *when*(mockClient.sendEmail("a@b.com","Hi","Hello")).thenReturn(true);  
 EmailService service=new EmailService(mockClient);  
 *assertTrue*(service.send("a@b.com","Hi","Hello"));  
 }  
 @Test  
 public void testIsSpam(){  
 EmailClient mockClient=*mock*(EmailClient.class);  
 *when*(mockClient.isSpam("e456")).thenReturn(true);  
 EmailService service=new EmailService(mockClient);  
 *assertTrue*(service.isSuspicious("e456"));  
 }  
 @Test  
 public void testRemoveEmail(){  
 EmailClient mockClient=*mock*(EmailClient.class);  
 *when*(mockClient.deleteEmail("e789")).thenReturn("Deleted");  
 EmailService service=new EmailService(mockClient);  
 *assertEquals*("Deleted", service.removeEmail("e789"));  
 }  
}

**Output:**

