**Exercise 1: Mocking a Service Dependency in a Controller Test**

**Create the UserControllerTest:**

package controller;  
  
import model.User;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.springframework.http.ResponseEntity;  
import service.UserService;  
  
@ExtendWith(MockitoExtension.class)  
public class UserControllerTest {  
  
 @Mock  
 private UserService userService;  
  
 @InjectMocks  
 private UserController userController;  
  
 @Test  
 void getUser\_ShouldReturnUser\_WhenUserExists() {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Test User");  
  
 *when*(userService.getUserById(1L)).thenReturn(user);  
  
 ResponseEntity<User> response = userController.getUser(1L);  
  
 *assertEquals*(200, response.getStatusCodeValue());  
 *assertNotNull*(response.getBody());  
 *assertEquals*("Test User", response.getBody().getName());  
 }  
  
 @Test  
 void getUser\_ShouldReturnNotFound\_WhenUserDoesNotExist() {  
 *when*(userService.getUserById(1L)).thenReturn(null);  
  
 ResponseEntity<User> response = userController.getUser(1L);  
  
 *assertEquals*(404, response.getStatusCodeValue());  
 *assertNull*(response.getBody());  
 }  
}

**Exercise 2: Mocking a Repository in a Service Test**

**Create the UserServiceTest:**

package com.example.demo.service;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

void getUserById\_ShouldReturnUser\_WhenUserExists() {

User user = new User();

user.setId(1L);

user.setName("Mocked User");

when(userRepository.findById(1L)).thenReturn(Optional.of(user));

User result = userService.getUserById(1L);

assertNotNull(result);

assertEquals("Mocked User", result.getName());

verify(userRepository, times(1)).findById(1L);

}

@Test

void getUserById\_ShouldReturnNull\_WhenUserDoesNotExist() {

when(userRepository.findById(1L)).thenReturn(Optional.empty());

User result = userService.getUserById(1L);

assertNull(result);

verify(userRepository, times(1)).findById(1L);

}

}

**Exercise 3: Mocking a Service Dependency in an Integration Test**

**Create the UserIntegrationTest:**

package com.example.demo;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.test.web.servlet.MockMvc;

import static org.mockito.Mockito.when;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void getUser\_ShouldReturnUser\_WhenUserExists() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Integration User");

when(userService.getUserById(1L)).thenReturn(user);

mockMvc.perform(get("/users/1"))

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("Integration User"));

}

@Test

void getUser\_ShouldReturnNotFound\_WhenUserDoesNotExist() throws Exception {

when(userService.getUserById(1L)).thenReturn(null);

mockMvc.perform(get("/users/1"))

.andExpect(status().isNotFound());

}

}