**Exercise 1: Basic Unit Test for a Service Method**

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

import org.junit.jupiter.api.Test;

public class CalculatorServiceTest {

@Test

public void testAdd() {

CalculatorService service = new CalculatorService();

assertEquals(5, service.add(2, 3));

}

}

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

import static org.mockito.Mockito.\*;

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

import java.util.Optional;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("John");

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

User result = userService.getUserById(1L);

assertEquals("John", result.getName());

}

}

**Exercise 3: Testing a REST Controller with MockMvc**

import static org.mockito.Mockito.\*;

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

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

import org.junit.jupiter.api.Test;

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

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

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

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

@WebMvcTest(UserController.class)

public class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("John");

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

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

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("John"));

}

}

**Exercise 4: Integration Test with Spring Boot**

import org.junit.jupiter.api.Test;

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

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

import org.springframework.boot.test.web.client.TestRestTemplate;

import org.springframework.boot.web.server.LocalServerPort;

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

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)

public class UserIntegrationTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Test

public void testGetUser() {

User user = restTemplate.getForObject("http://localhost:" + port + "/users/1", User.class);

assertNotNull(user);

}

}

**Exercise 5: Test Controller POST Endpoint**

@Test

public void testCreateUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Alice");

when(userService.saveUser(any(User.class))).thenReturn(user);

mockMvc.perform(post("/users")

.contentType("application/json")

.content("{\"id\":1,\"name\":\"Alice\"}"))

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("Alice"));

}

**Exercise 6: Test Service Exception Handling**

@Test

public void testUserNotFound() {

when(userRepository.findById(anyLong())).thenReturn(Optional.empty());

User result = userService.getUserById(99L);

assertNull(result);

}

**Exercise 7: Test Custom Repository Query**

@Test

public void testFindByName() {

List<User> users = List.of(new User(1L, "Alice"));

when(userRepository.findByName("Alice")).thenReturn(users);

List<User> result = userRepository.findByName("Alice");

assertEquals(1, result.size());

assertEquals("Alice", result.get(0).getName());

}

**Exercise 8: Test Controller Exception Handling**

@Test

public void testExceptionHandler() throws Exception {

when(userService.getUserById(99L)).thenThrow(new NoSuchElementException());

mockMvc.perform(get("/users/99"))

.andExpect(status().isNotFound())

.andExpect(content().string("User not found"));

}

**Exercise 9: Parameterized Test with JUnit**

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvSource;

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

public class CalculatorServiceParamTest {

private final CalculatorService service = new CalculatorService();

@ParameterizedTest

@CsvSource({

"1, 2, 3",

"5, 5, 10",

"10, -5, 5"

})

void testAdd(int a, int b, int expected) {

assertEquals(expected, service.add(a, b));

}

}