4. Junit Spring Test exercises

Exercise 1: Basic Unit Test for a Service Method

CODE

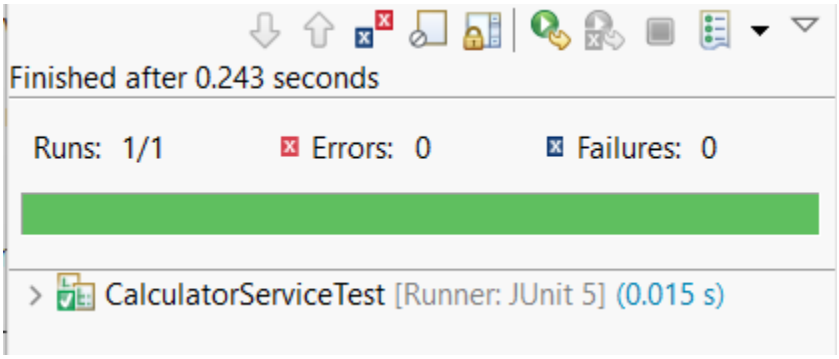
File name: CalculatorService.java

```
package com.example.service;


import org.springframework.stereotype.Service;


@Service
public class CalculatorService {
   public int add(int a, int b) {
      return a + b;
   }
}
```

File name: CalculatorServiceTest

```
package com.example.service;


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


public class CalculatorServiceTest {


   CalculatorService calculatorService = new CalculatorService();


   @Test
   public void testAdd() {
      int result = calculatorService.add(2, 3);
      assertEquals(5, result);
   }
}
```

OUTPUT



Exercise 2: Mocking a Repository in a Service Test

CODE

File name: User.java

```
package com.example.model;


import jakarta.persistence.Entity;
import jakarta.persistence.Id;


@Entity
public class User {
   @Id
   private Long id;
   private String name;


   //  Add this: Getter and Setter for id
   public Long getId() {
      return id;
   }


   public void setId(Long id) {
      this.id = id;
   }


   // Add this: Getter and Setter for name
   public String getName() {
      return name;
   }
```

```
    public void setName(String name) {
        this.name = name;
    }
}
```

File name: UserRepository.java

```
package com.example.repository;

import com.example.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    // you can add custom queries here later
}
```

File name: UserService.java

```
package com.example.service;

import com.example.model.User;
import com.example.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public User getUserById(Long id) {
        Optional<User> optionalUser = userRepository.findById(id);
```

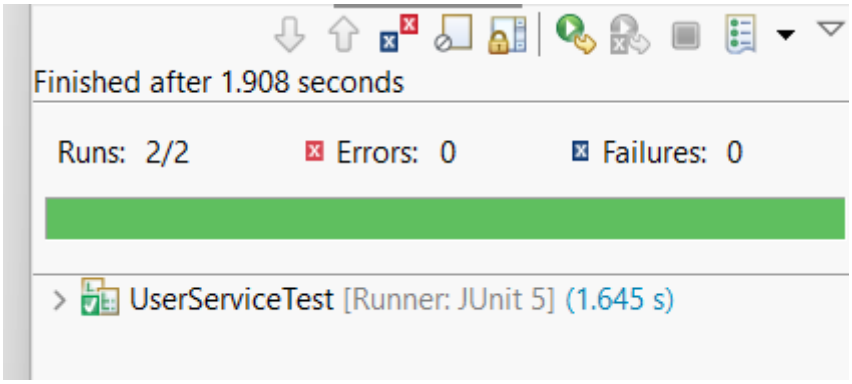
```
        return optionalUser.orElse(null);
    }

    // You can add more methods, like saveUser(), updateUser(), etc., later
}
```

File name: UserServiceTest.java

```
package com.example.service;

import com.example.model.User;
import com.example.repository.UserRepository;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Optional;

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

@ExtendWith(MockitoExtension.class)
public class UserServiceTest {

    @Mock
    private UserRepository userRepository;

    @InjectMocks
    private UserService userService;

    @Test
    public void testGetUserById_found() {
        // Arrange
        User user = new User();
```

```
        user.setId(1L);
        user.setName("Alice");

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

        // Act
        User result = userService.getUserById(1L);

        // Assert
        assertNotNull(result);
        assertEquals("Alice", result.getName());
    }

    @Test
    public void testGetUserById_notFound() {
        Mockito.when(userRepository.findById(99L)).thenReturn(Optional.empty());

        User result = userService.getUserById(99L);

        assertNull(result);
    }
}
```

OUTPUT



Exercise 3: Testing a REST Controller with MockMvc

CODE

File name: User.java

```
package com.example.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class User {

    @Id
    private Long id;
    private String name;

    // Getters and setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

File name: UserRepositroy.java

```
package com.example.repository;

import com.example.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
   // You can add custom query methods later
}
```

File name: UserService.java

```
package com.example.service;

import com.example.model.User;
import com.example.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {

   @Autowired
   private UserRepository userRepository;

   // Used in Exercise 2 and 3
   public User getUserById(Long id) {
      Optional<User> userOptional = userRepository.findById(id);
      return userOptional.orElse(null);
   }
```

```
    // Used in Exercise 5 (create user)
    public User saveUser(User user) {
        return userRepository.save(user);
    }
}
```

File name: UserController.java

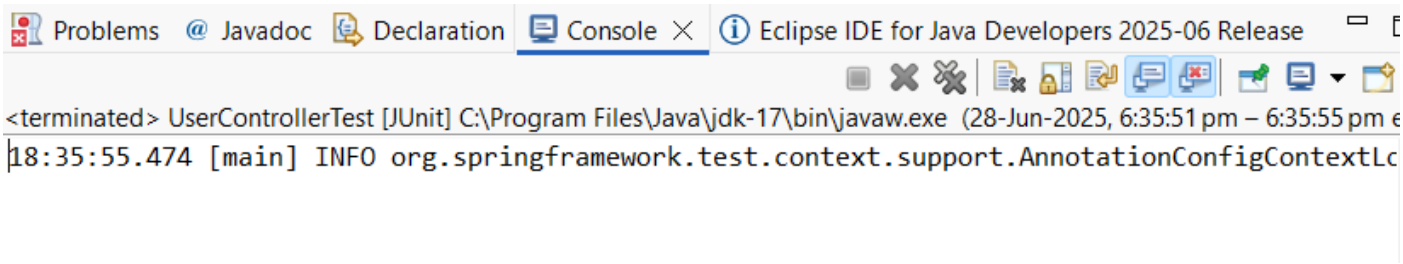
```
package com.example.controller;

import com.example.model.User;
import com.example.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    public ResponseEntity<User> getUser(@PathVariable Long id) {
        User user = userService.getUserById(id);
        return ResponseEntity.ok(user);
    }
}
```

File name: UserControllerTest.java

```
package com.example.controller;

import com.example.model.User;
import com.example.service.UserService;
```

```
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.http.MediaType;
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.*;

@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("Test User");

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

        mockMvc.perform(get("/users/1"))
                .andExpect(status().isOk())
                .andExpect(jsonPath("$.name").value("Test User"));
    }
}
```

OUTPUT


Problems
Javadoc
Declaration
Console
Eclipse IDE for Java Developers 2025-06 Release
<terminated> UserControllerTest [JUnit] C:\Program Files\Java\jdk-17\bin\javaw.exe (28-Jun-2025, 6:35:51 pm – 6:35:55 pm
18:35:55.474 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLc