**DIGITAL NURTURE 4.0 JAVA -FSE**

**Submitted By:**

**THAMIMUL ANSARI M**

**Superset ID: 6400989**

**1. JUnit\_Basic Testing Exercises**

### ✅ Exercise 1: Setting Up JUnit

**Files Created:**

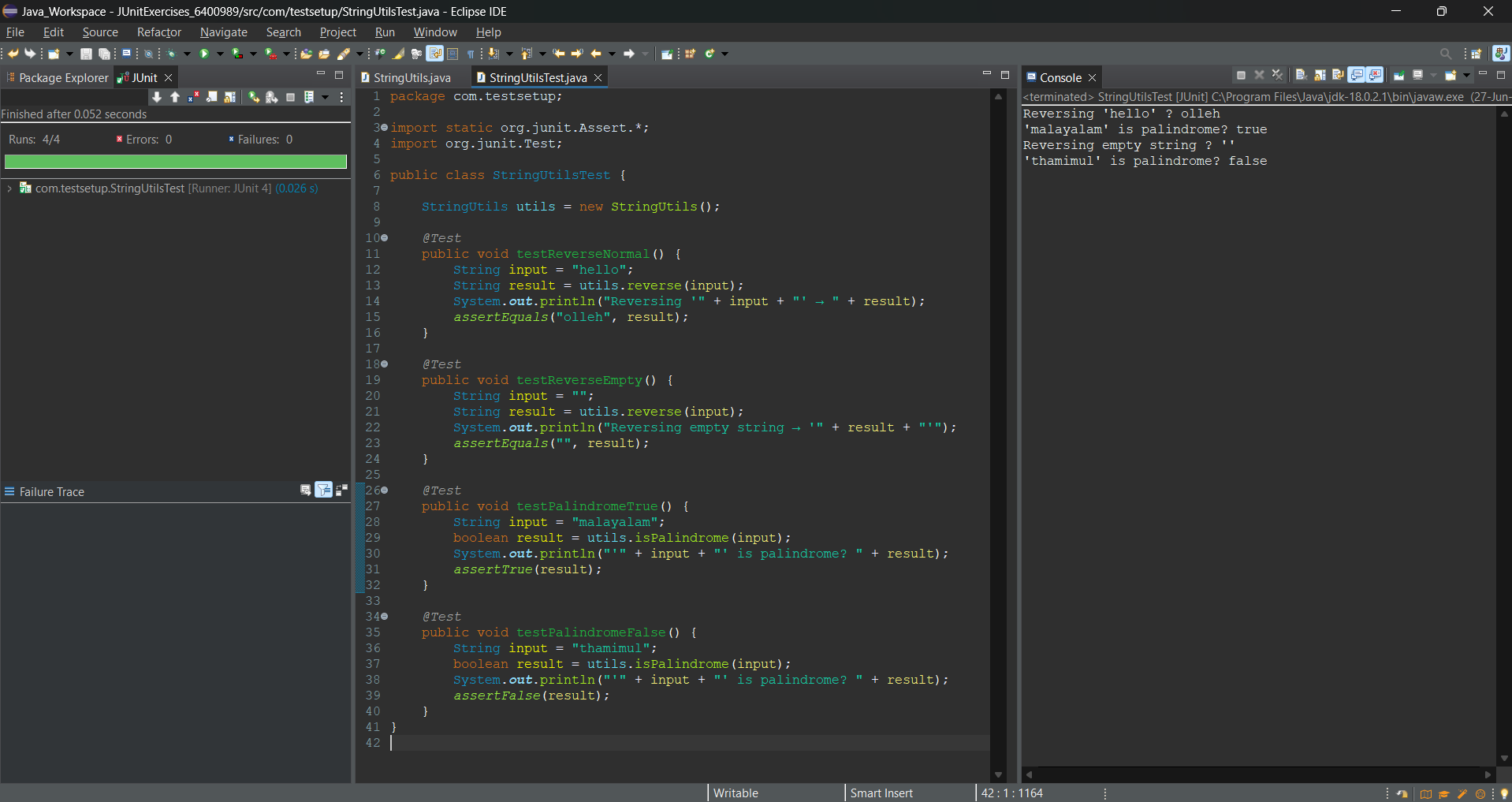
**StringUtils.java**

package com.testsetup;  
  
public class StringUtils {  
 public String reverse(String input) {  
 return new StringBuilder(input).reverse().toString();  
 }  
  
 public boolean isPalindrome(String input) {  
 String reversed = reverse(input);  
 return input.equalsIgnoreCase(reversed);  
 }  
}

**StringUtilsTest.java**

package com.testsetup;  
  
import static org.junit.Assert.\*;  
import org.junit.Test;  
  
public class StringUtilsTest {  
  
 StringUtils utils = new StringUtils();  
  
 @Test  
 public void testReverseNormal() {  
 String input = "hello";  
 String result = utils.reverse(input);  
 System.out.println("Reversing '" + input + "' → " + result);  
 assertEquals("olleh", result);  
 }  
  
 @Test  
 public void testReverseEmpty() {  
 String input = "";  
 String result = utils.reverse(input);  
 System.out.println("Reversing empty string → '" + result + "'");  
 assertEquals("", result);  
 }  
  
 @Test  
 public void testPalindromeTrue() {  
 String input = "malayalam";  
 boolean result = utils.isPalindrome(input);  
 System.out.println("'" + input + "' is palindrome? " + result);  
 assertTrue(result);  
 }  
  
 @Test  
 public void testPalindromeFalse() {  
 String input = "thamimul";  
 boolean result = utils.isPalindrome(input);  
 System.out.println("'" + input + "' is palindrome? " + result);  
 assertFalse(result);  
 }  
}

**Output Screenshot:**

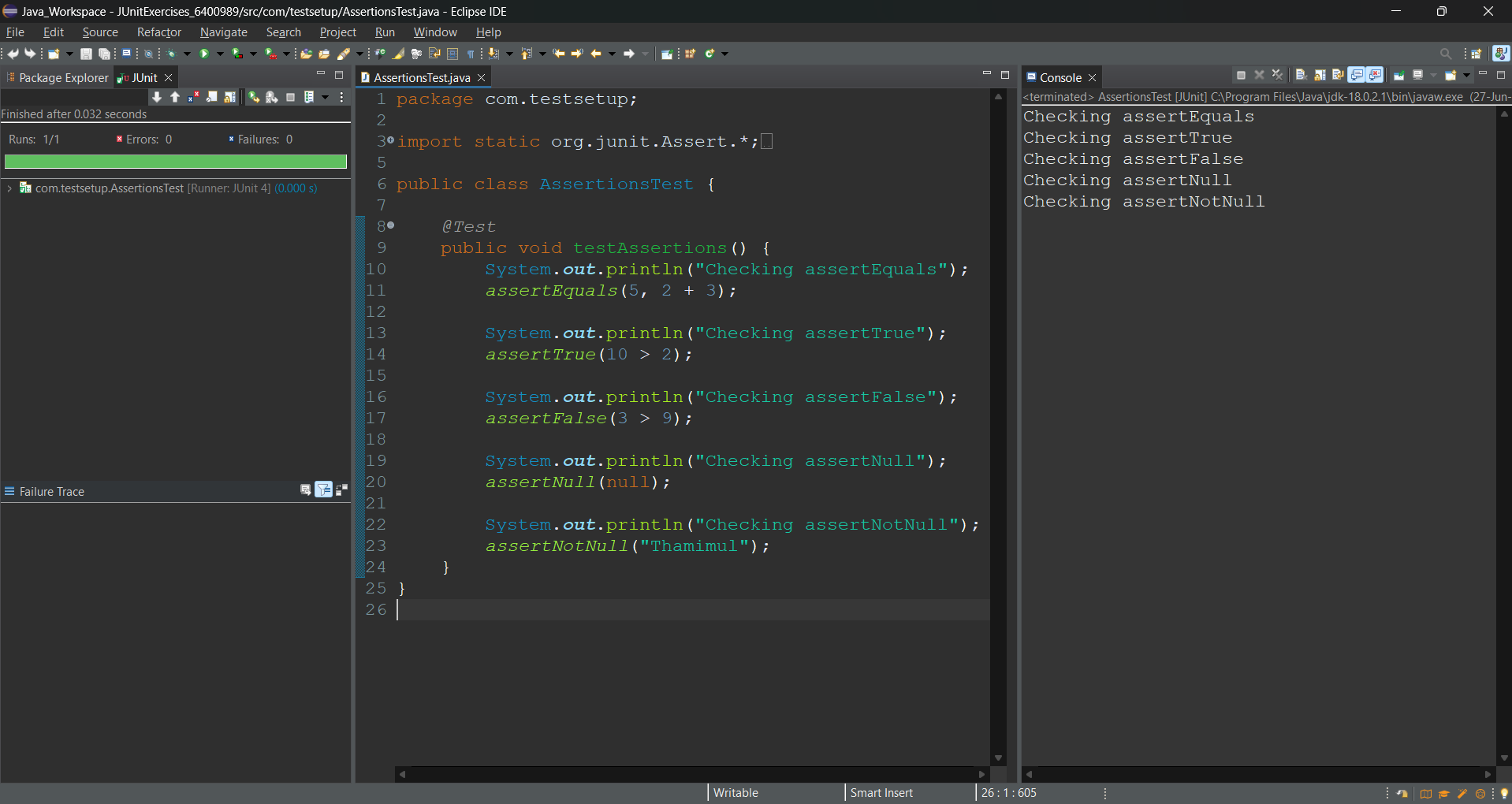


### ✅ Exercise 3: Assertions in JUnit

**AssertionsTest.java**

package com.testsetup;  
  
import static org.junit.Assert.\*;  
import org.junit.Test;  
  
public class AssertionsTest {  
 @Test  
 public void testAssertions() {  
 System.out.println("Checking assertEquals");  
 assertEquals(5, 2 + 3);  
  
 System.out.println("Checking assertTrue");  
 assertTrue(10 > 2);  
  
 System.out.println("Checking assertFalse");  
 assertFalse(3 > 9);  
  
 System.out.println("Checking assertNull");  
 assertNull(null);  
  
 System.out.println("Checking assertNotNull");  
 assertNotNull("Thamimul");  
 }  
}

**Output Screenshot:**



### 

### ✅ Exercise 4: AAA Pattern, Setup and Teardown (LoginService - Unique)

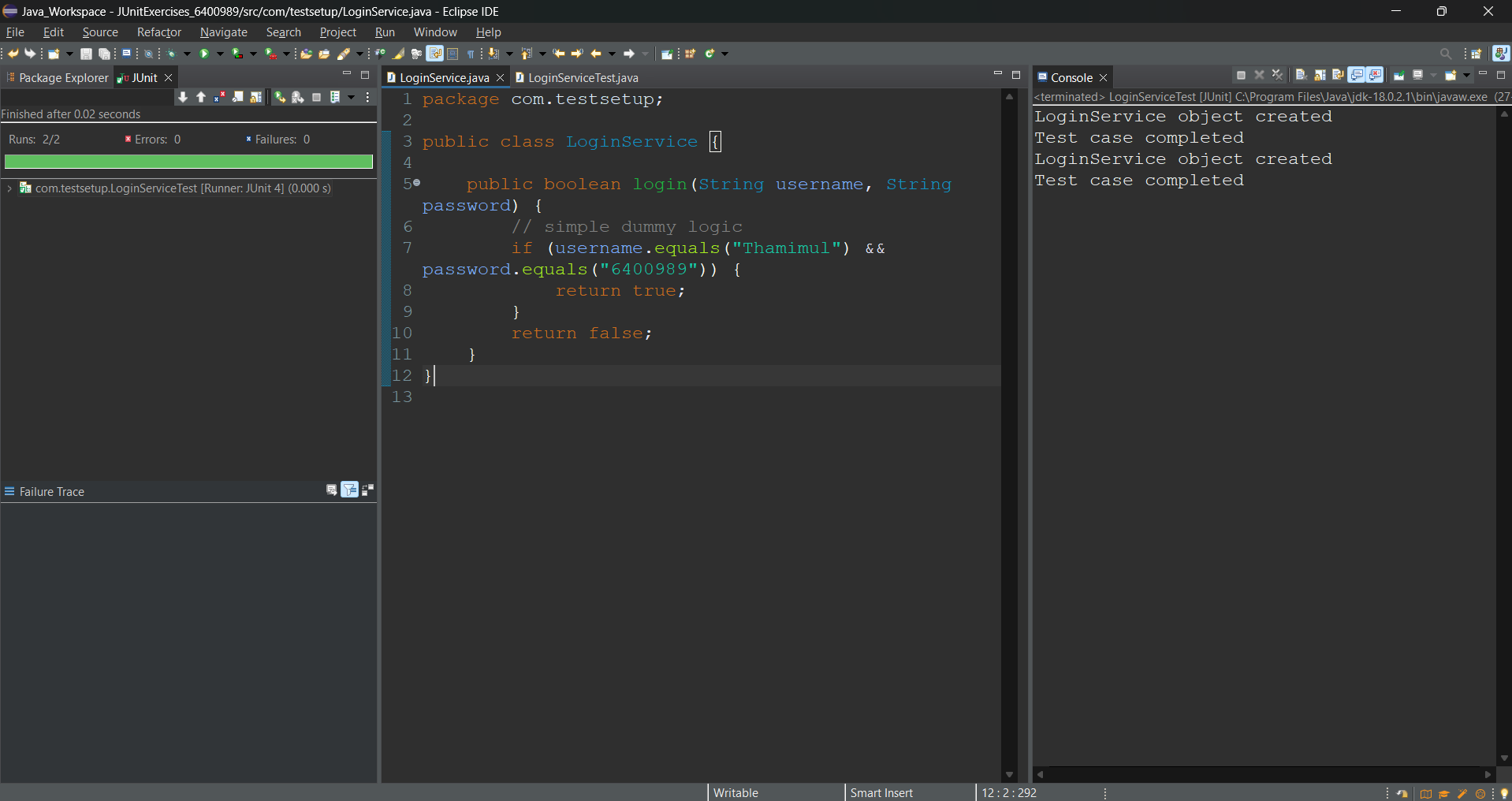
**LoginService.java**

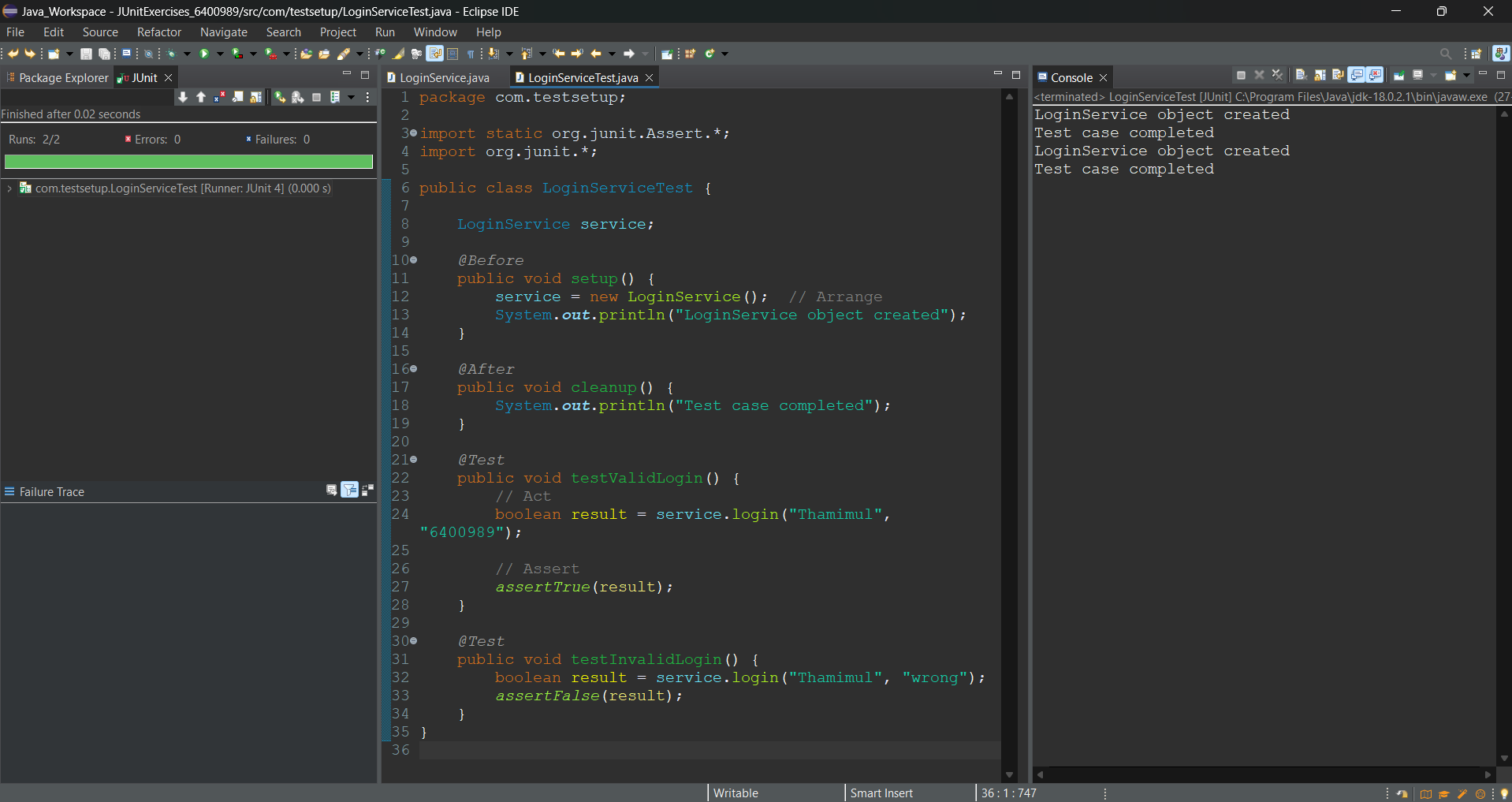
package com.testsetup;  
  
public class LoginService {  
 public boolean login(String username, String password) {  
 return username.equals("Thamimul") && password.equals("6400989");  
 }  
}

**LoginServiceTest.java**

package com.testsetup;  
  
import static org.junit.Assert.\*;  
import org.junit.\*;  
  
public class LoginServiceTest {  
  
 LoginService service;  
  
 @Before  
 public void setup() {  
 service = new LoginService();  
 System.out.println("LoginService object created");  
 }  
  
 @After  
 public void cleanup() {  
 System.out.println("Test case completed");  
 }  
  
 @Test  
 public void testValidLogin() {  
 boolean result = service.login("Thamimul", "6400989");  
 assertTrue(result);  
 }  
  
 @Test  
 public void testInvalidLogin() {  
 boolean result = service.login("user", "wrong");  
 assertFalse(result);  
 }  
}

**Output Screenshot:**





**3. Mockito exercises**

### ✅ Exercise 1: Mocking and Stubbing

**MyService.java**

package com.testsetup;  
  
public class MyService {  
 ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

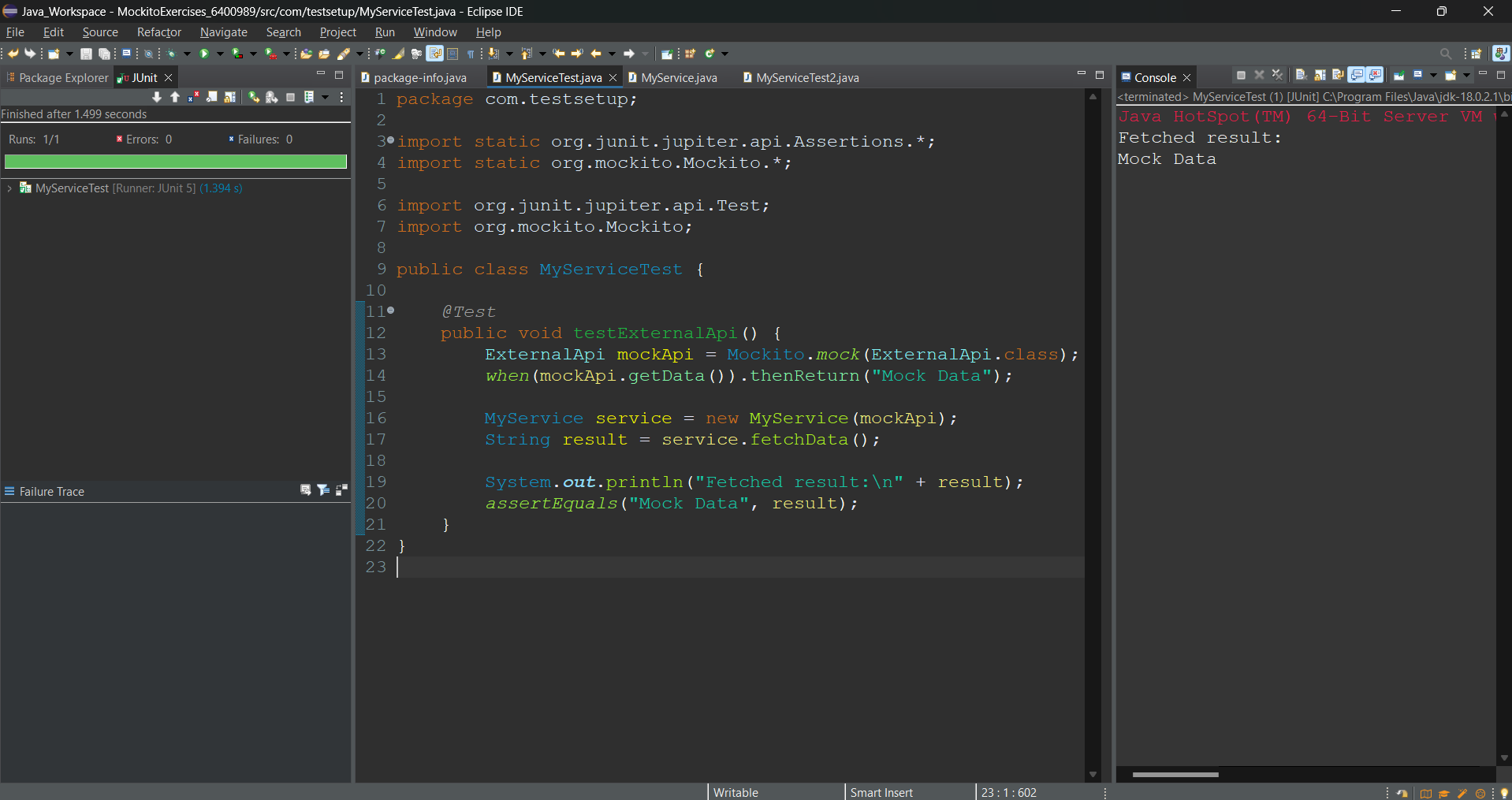
**ExternalApi.java**

package com.testsetup;  
  
public interface ExternalApi {  
 String getData();  
}

**MyServiceTest.java (Mocking Test)**

package com.testsetup;  
  
import static org.junit.jupiter.api.Assertions.\*;  
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();  
  
 System.out.println("Fetched result:\n" + result);  
 assertEquals("Mock Data", result);  
 }  
}

**Output Screenshot:**



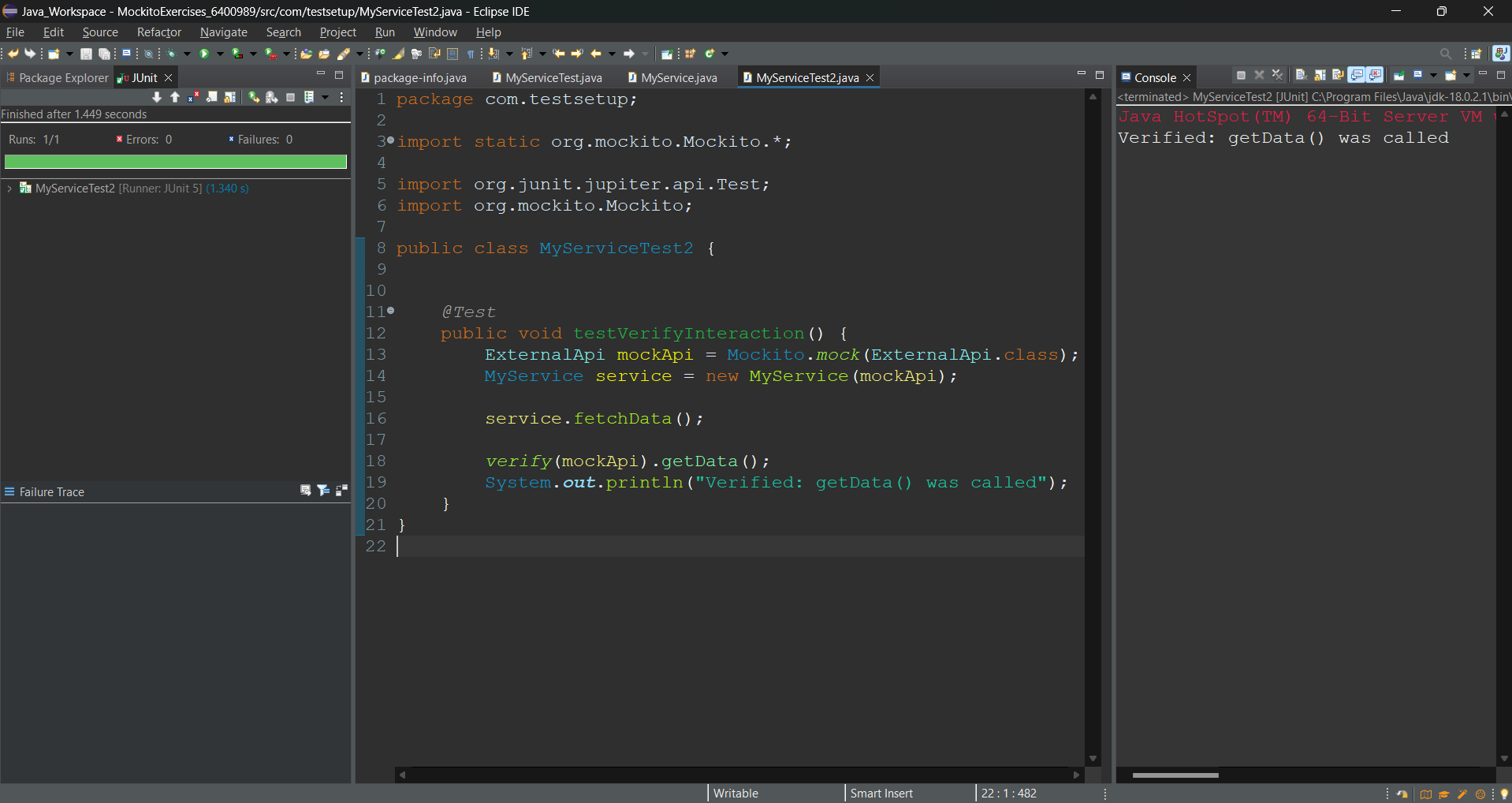
### 

### ✅ Mockito Exercise 2: Verifying Interaction

**MyServiceTest.java (Verification Test)**

@Test  
public void testVerifyInteraction() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 MyService service = new MyService(mockApi);  
  
 service.fetchData();  
  
 verify(mockApi).getData();  
 System.out.println("Verified: getData() was called");  
}

**Output Screenshot:**



### 

**6. SL4J Logging exercises**

### ✅ SLF4J Logging Exercise 1: Logging Error Messages and Warning Levels

**pom.xml (Dependencies)**

<dependencies>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
 </dependency>  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.3</version>  
 </dependency>  
</dependencies>

**LoggingExample.java**

package com.logdemo;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
public class LoggingExample {  
  
 private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);  
  
 public static void main(String[] args) {  
 logger.error("Error message using SL4J Logger");  
 logger.warn("Warning message using SL4J Logger");  
  
 System.out.println("Log messages sent. Check console.");  
 }  
}

**Output Screenshot:**

