**WEEK-2**

**Advanced JUnit Testing Exercises**

**Exercise 1: Parameterized Tests**

**Code:**

**VoteChecker.java**

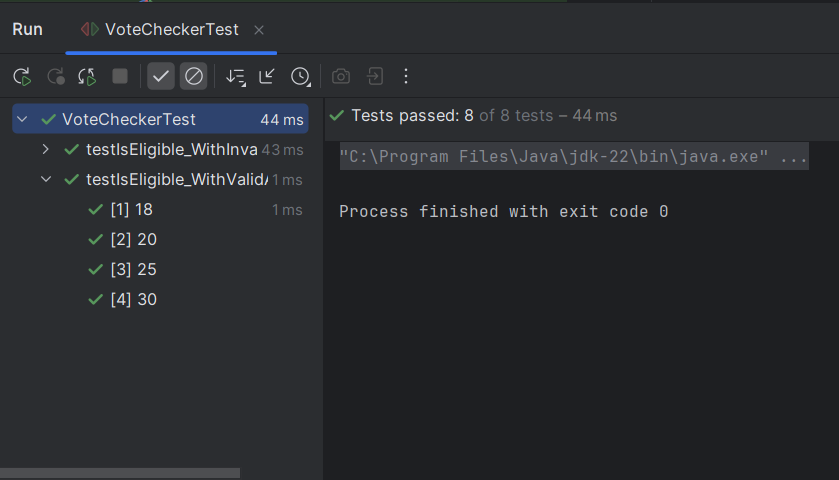
package com.example;  
  
public class VoteChecker {  
 public boolean isEligible(int age) {  
 return age >= 18;  
 }  
}

**VoteCheckerTest.java**

import com.example.VoteChecker;  
  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.ValueSource;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class VoteCheckerTest {  
  
 private final VoteChecker checker = new VoteChecker();  
  
 @ParameterizedTest  
 @ValueSource(ints = {18, 20, 25, 30})  
 public void testIsEligible\_WithValidAges(int age) {  
 *assertTrue*(checker.isEligible(age));  
 }  
  
 @ParameterizedTest  
 @ValueSource(ints = {0, 5, 10, 17})  
 public void testIsEligible\_WithInvalidAges(int age) {  
 *assertFalse*(checker.isEligible(age));

}  
}

**Output:**

****

**Exercise 2: Test Suites and Categories**

**Code:**

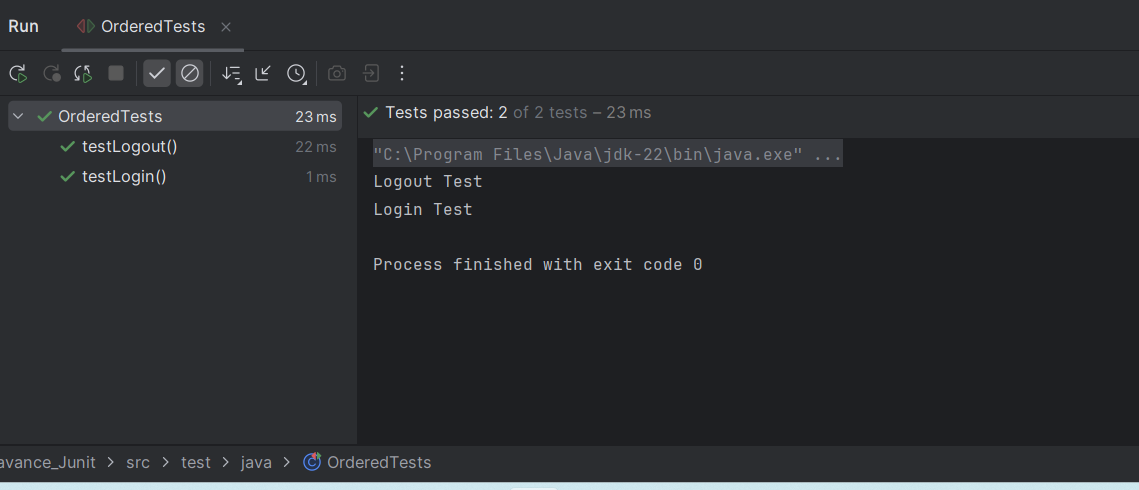
**Exercise 3: Test Execution Order**

**Code:**

**OrderedTests.java**

import org.junit.jupiter.api.\*;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
public class OrderedTests {  
  
 @Test  
 @Order(1)  
 public void testLogin() {  
 System.*out*.println("Login Test");  
 *assertTrue*(true);  
 }  
  
 @Test  
 @Order(2)  
 public void testLogout() {  
 System.*out*.println("Logout Test");  
 *assertTrue*(true);  
 }  
}

**Output:**

****

**Exercise 4: Exception Testing**

**Code:**

**ExceptionThrower.java**

package com.example;  
  
public class ExceptionThrower {  
  
 public void throwException(String input) {  
 if (input == null || input.isEmpty()) {  
 throw new IllegalArgumentException("Input must not be null or empty");  
 }  
 System.*out*.println("Input: " + input);  
 }  
}

**ExceptionThrowerTest.java**

import com.example.ExceptionThrower;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class ExceptionThrowerTest {  
  
 @Test  
 public void testThrowException\_WithNullInput() {  
 ExceptionThrower thrower = new ExceptionThrower();  
  
 Exception exception = *assertThrows*(IllegalArgumentException.class, () -> {  
 thrower.throwException(null);  
 });  
  
 *assertEquals*("Input must not be null or empty", exception.getMessage());  
 }  
  
 @Test  
 public void testThrowException\_WithEmptyInput() {  
 ExceptionThrower thrower = new ExceptionThrower();  
  
 Exception exception = *assertThrows*(IllegalArgumentException.class, () -> {  
 thrower.throwException("");  
 });  
  
 *assertEquals*("Input must not be null or empty", exception.getMessage());  
 }  
  
 @Test  
 public void testThrowException\_WithValidInput() {  
 ExceptionThrower thrower = new ExceptionThrower();  
  
 *assertDoesNotThrow*(() -> {  
 thrower.throwException("Thilagar");  
 });  
 } **}**

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 5: Timeout and Performance Testing**

**Code:**

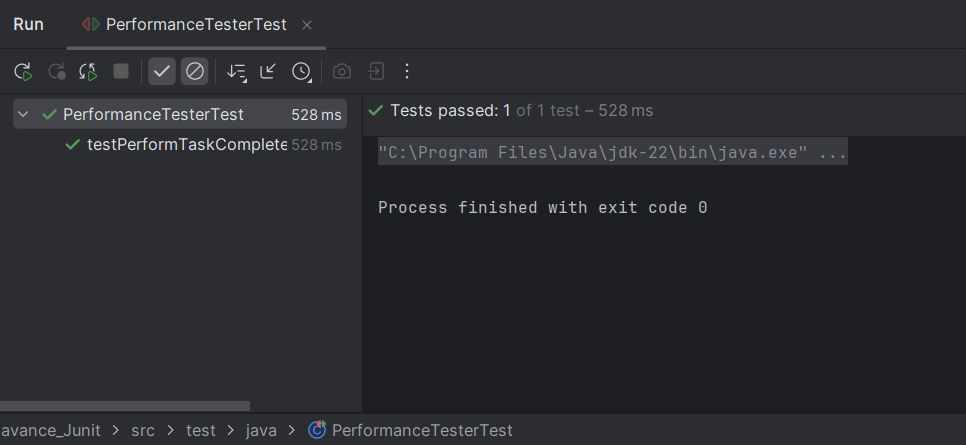
**PerformanceTester.java**

package com.example;  
  
public class PerformanceTester {  
  
 public void performTask() {  
 try {  
 Thread.*sleep*(500);   
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 }  
 }  
}

**PerformanceTesterTest.java**

import com.example.PerformanceTester;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertTimeout*;  
import java.time.Duration;  
  
public class PerformanceTesterTest {  
  
 @Test  
 public void testPerformTaskCompletesWithin1Second() {  
 PerformanceTester tester = new PerformanceTester();  
  
 *assertTimeout*(Duration.*ofSeconds*(1), () -> {  
 tester.performTask();  
 });  
 }  
}

**Output:**

****