Advanced JUnit Testing Exercises - Solution

# Exercise 1: Parameterized Tests

Java class:

public class EvenChecker {  
 public boolean isEven(int number) {  
 return number % 2 == 0;  
 }  
}

JUnit test with parameterized input:

import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.ValueSource;  
  
public class EvenCheckerTest {  
  
 @ParameterizedTest  
 @ValueSource(ints = {2, 4, 6, 8, 10})  
 public void testIsEven(int number) {  
 EvenChecker checker = new EvenChecker();  
 assertTrue(checker.isEven(number));  
 }  
}

Output:  
All parameterized test cases passed successfully.

# Exercise 2: Test Suites and Categories

JUnit suite class:

import org.junit.platform.suite.api.SelectClasses;  
import org.junit.platform.suite.api.Suite;  
  
@Suite  
@SelectClasses({EvenCheckerTest.class, OrderedTests.class})  
public class AllTests {  
}

Output:  
All tests in EvenCheckerTest and OrderedTests ran successfully.

# Exercise 3: Test Execution Order

JUnit test class with ordered tests:

import org.junit.jupiter.api.\*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
public class OrderedTests {  
  
 @Test  
 @Order(1)  
 public void firstTest() {  
 System.out.println("Running first test");  
 }  
  
 @Test  
 @Order(2)  
 public void secondTest() {  
 System.out.println("Running second test");  
 }  
}

Output:  
Running first test  
Running second test

# Exercise 4: Exception Testing

Java class:

public class ExceptionThrower {  
 public void throwException() throws IllegalArgumentException {  
 throw new IllegalArgumentException("Invalid argument!");  
 }  
}

JUnit test class:

import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
  
public class ExceptionThrowerTest {  
  
 @Test  
 public void testException() {  
 ExceptionThrower et = new ExceptionThrower();  
 assertThrows(IllegalArgumentException.class, et::throwException);  
 }  
}

Output:  
Exception thrown and caught as expected.

# Exercise 5: Timeout and Performance Testing

Java class:

public class PerformanceTester {  
 public void performTask() throws InterruptedException {  
 Thread.sleep(500); // Simulate task  
 }  
}

JUnit test with timeout:

import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.Timeout;  
import java.util.concurrent.TimeUnit;  
  
public class PerformanceTesterTest {  
  
 @Test  
 @Timeout(value = 1, unit = TimeUnit.SECONDS)  
 public void testPerformTask() throws InterruptedException {  
 PerformanceTester pt = new PerformanceTester();  
 pt.performTask();  
 }  
}

Output:  
Test completed within timeout limit.