JUnit Advanced Testing Exercises - Solutions

# Exercise 1: Parameterized Tests

Steps:

1. Create a class EvenChecker with isEven method.  
2. Write a parameterized test class using @ParameterizedTest and @ValueSource.

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

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

# Exercise 2: Test Suites and Categories

Steps:  
1. Create a test suite class.  
2. Add test classes using @Suite and @SelectClasses.

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

# Exercise 3: Test Execution Order

Steps:  
1. Use @TestMethodOrder and @Order to define order of tests.

import org.junit.jupiter.api.\*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
public class OrderedTests {  
  
 @Test  
 @Order(2)  
 void secondTest() {  
 System.out.println("Second test");  
 }  
  
 @Test  
 @Order(1)  
 void firstTest() {  
 System.out.println("First test");  
 }  
  
 @Test  
 @Order(3)  
 void thirdTest() {  
 System.out.println("Third test");  
 }  
}

# Exercise 4: Exception Testing

Steps:  
1. Test that a method throws the expected exception.

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

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

# Exercise 5: Timeout and Performance Testing

Steps:  
1. Ensure method completes within time limit using assertTimeout.

public class PerformanceTester {  
 public void performTask() {  
 try {  
 Thread.sleep(100); // simulate task  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 }  
 }  
}

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
import java.time.Duration;  
  
public class PerformanceTesterTest {  
  
 @Test  
 void testTaskCompletesWithinTime() {  
 PerformanceTester tester = new PerformanceTester();  
 assertTimeout(Duration.ofMillis(200), tester::performTask);  
 }  
}