2. JUnit_Advanced Testing exercices

Exercise 1: Parameterized Tests

CODE

File name: EvenChecker.java

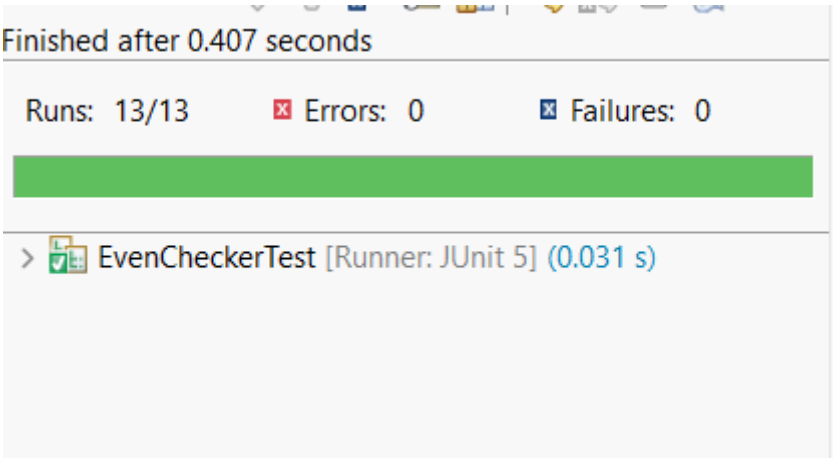
```
package com.example;

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

File name: EvenCheckerTest.java

```
package com.example;

import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;

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

public class EvenCheckerTest {

   EvenChecker checker = new EvenChecker();

   @ParameterizedTest
   @ValueSource(ints = {2, 4, 6, 8, 10, -2, 0})
   public void testIsEvenTrue(int input) {
      assertTrue(checker.isEven(input));
   }

   @ParameterizedTest
   @ValueSource(ints = {1, 3, 5, 7, -1, -3})
   public void testIsEvenFalse(int input) {
      assertFalse(checker.isEven(input));
   }
```

```
}
```

OUTPUT



Exercise 2: Test Suites and Categories

CODE

File name: Calculator.java

```
package com.example;

public class Calculator {
   public int add(int a, int b) {
      return a + b;
   }

   public int subtract(int a, int b) {
      return a - b;
   }
}
```

File name: AdditionTest.java

```
package com.example;

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

public class AdditionTest {

   Calculator calc = new Calculator();
```

```
    @Test
    public void testAdd() {
        assertEquals(7, calc.add(3, 4));
    }
}
```

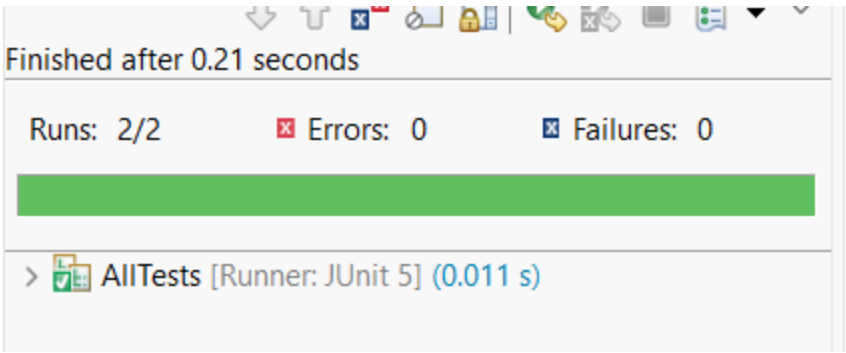
File name: Subtraction.java

```
package com.example;

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

public class SubtractionTest {

    Calculator calc = new Calculator();

    @Test
    public void testSubtract() {
        assertEquals(5, calc.subtract(10, 5));
    }
}
```

File name: AllTests.java

```
package com.example;

import org.junit.platform.suite.api.SelectClasses;
import org.junit.platform.suite.api.Suite;

@Suite
@SelectClasses({
    AdditionTest.class,
    SubtractionTest.class
})
public class AllTests {
    // No code needed – annotations handle everything
}
```

OUTPUT



Exercise 3: Test Execution Order

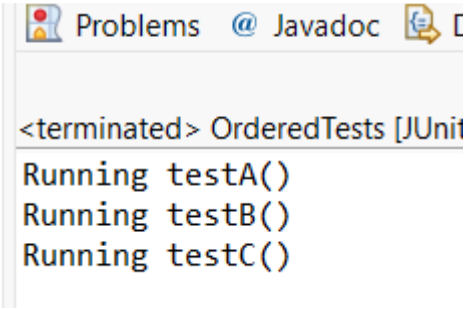
CODE

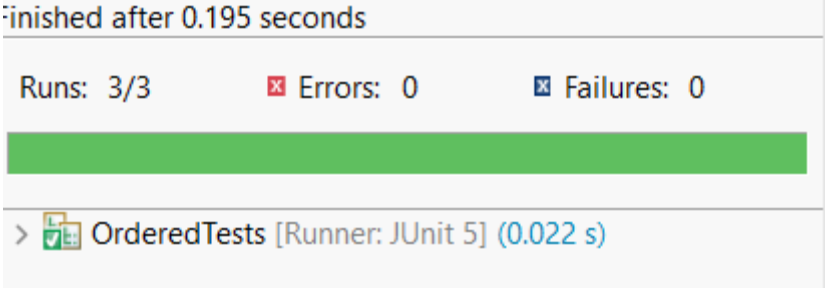
File name: OrderedTests.java

```
package com.example;

import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestMethodOrder;

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

@TestMethodOrder(OrderAnnotation.class)  // ✅ Enables ordered execution
public class OrderedTests {

    @Test
    @Order(3)
    public void testC() {
        System.out.println("Running testC()");
        assertTrue(true);
    }

    @Test
    @Order(1)
    public void testA() {
        System.out.println("Running testA()");
        assertEquals(4, 2 + 2);
```

```
    }

    @Test
    @Order(2)
    public void testB() {
        System.out.println("Running testB()");
        assertNotNull("JUnit");
    }
}
```

OUTPUT





Exercise 4: Exception Testing

CODE

File name: ExceptionThrower.java

```
package com.example;

public class ExceptionThrower {

    public void throwException(String input) {
        if (input == null || input.isEmpty()) {
            throw new IllegalArgumentException("Input cannot be null or empty");
        }
        // Otherwise, do something
        System.out.println("Valid input: " + input);
```

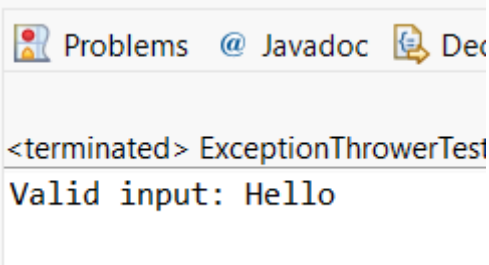
```
    }
}
```

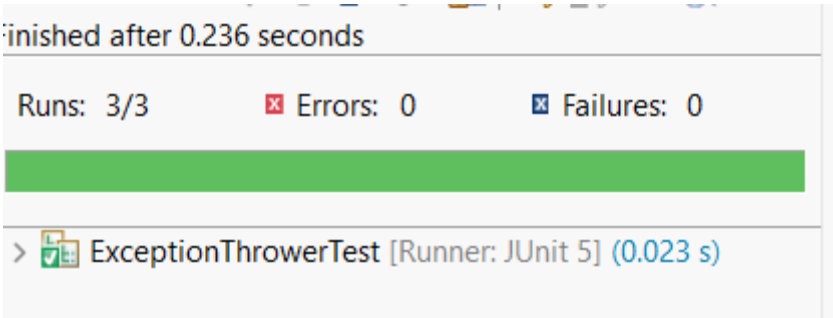
File name: ExceptionThrowerTest.java

```
package com.example;

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

public class ExceptionThrowerTest {

    ExceptionThrower thrower = new ExceptionThrower();

    @Test
    public void testThrowExceptionWithNull() {
        // Assert that exception is thrown when input is null
        assertThrows(IllegalArgumentException.class, () -> {
            thrower.throwException(null);
        });
    }

    @Test
    public void testThrowExceptionWithEmptyString() {
        // Assert that exception is thrown when input is empty
        assertThrows(IllegalArgumentException.class, () -> {
            thrower.throwException("");
        });
    }

    @Test
    public void testValidInputDoesNotThrow() {
        // Assert that no exception is thrown with valid input
        assertDoesNotThrow(() -> {
            thrower.throwException("Hello");
        });
```

```
    }
}
```

OUTPUT





Exercise 5: Timeout and Performance Testing

CODE

File name: PerformanceTester.java

```
package com.example;

public class PerformanceTester {

    public void performTask() {
        // Simulate a task that takes time
        try {
            Thread.sleep(100);  // 100 milliseconds
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

File name: PerformanceTesterTest.java

```
package com.example;

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

```
import java.time.Duration;

public class PerformanceTesterTest {

    PerformanceTester tester = new PerformanceTester();

    @Test
    public void testPerformTaskCompletesInTime() {
        // Task should complete within 500 milliseconds
        assertTimeout(Duration.ofMillis(500), () -> {
            tester.performTask();
        });
    }

    @Test
    public void testPerformTaskFailsIfTooSlow() {
        //  Example: this would fail if task took more than 50ms
        assertTimeout(Duration.ofMillis(50), () -> {
            tester.performTask();
        });
    }
}
```

OUTPUT