1) JUnit_Basic Testing Exercises

Exercise 1: Setting Up JUnit

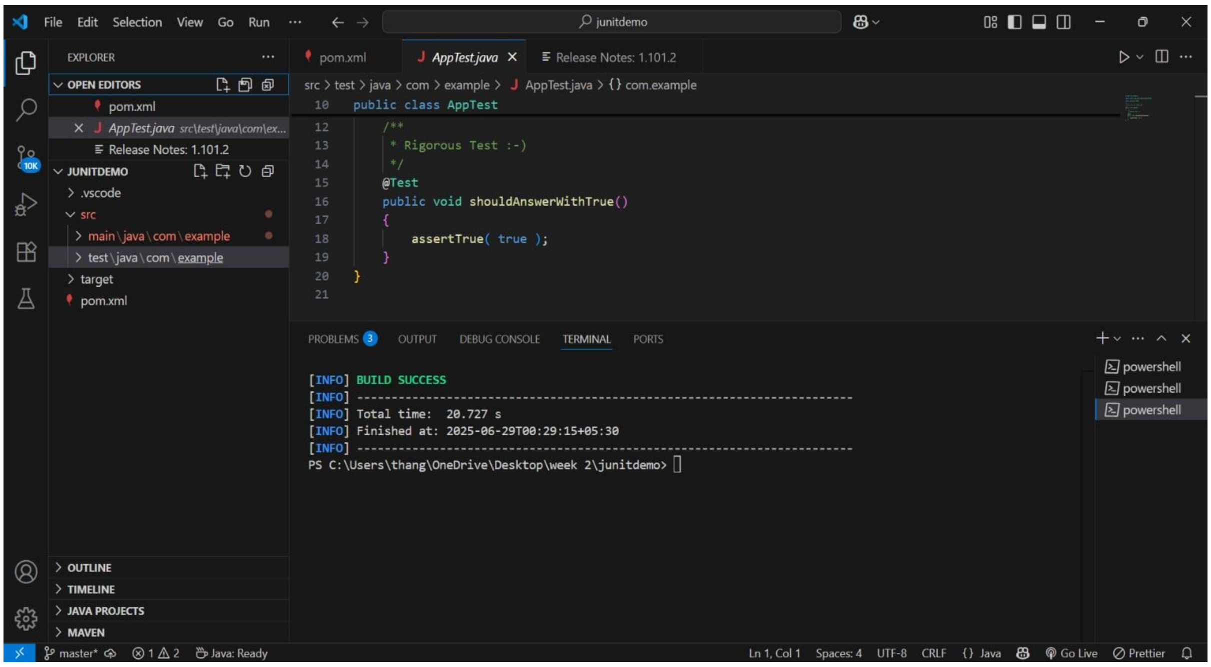
Calculator:

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

CalculatorTest:

```
package com.example;
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class CalculatorTest {
    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        int result = calc.add(2, 3);
        System.out.println("Result: " + result);
        assertEquals(5, result);
    }
}
```

OUTPUT:



```java
public class AppTest
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.727 s
[INFO] Finished at: 2025-06-29T00:29:15+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\thang\OneDrive\Desktop\week 2\junitdemo>
```

Exercise 3: Assertions in JUnit

Calculator:

```
package com.example;

public class Calculator {

public int add(int a, int b) {

        return a + b;

    }

}
```

AssertionTest:

```
import org.junit.jupiter.api.Test;

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


 public class AssertionsTest {

     @Test
```

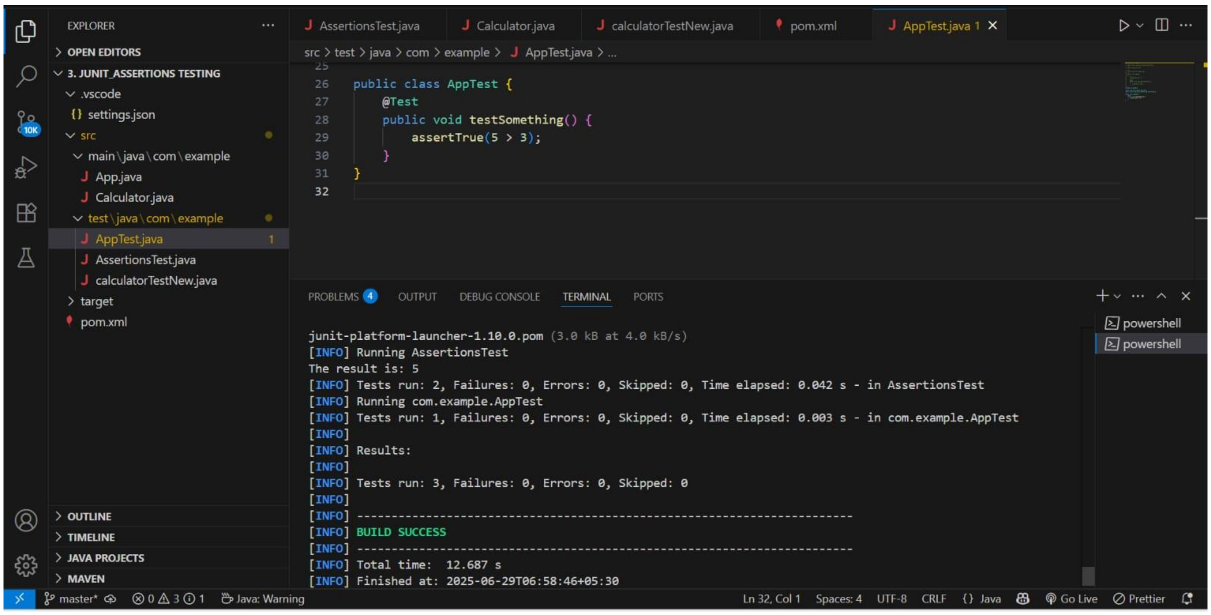
```
    public void testAssertions() {
        // Assert equals
        assertEquals(5, 2 + 3);

        // Assert true
        assertTrue(5 > 3);

        // Assert false
        assertFalse(5 < 3);

        // Assert null
        assertNull(null);

        // Assert not null
        assertNotNull(new Object());
    }
}
```

AppTest:

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

public class AppTest {
    @Test
    public void testSomething() {
        assertTrue(5 > 3);
    }
```

}

OUTPUT:



```java
public class AppTest {
    @Test
    public void testSomething() {
        assertTrue(5 > 3);
    }
}
```

```
junit-platform-launcher-1.10.0.pom (3.0 kB at 4.0 kB/s)
[INFO] Running AssertionsTest
The result is: 5
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.042 s - in AssertionsTest
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.687 s
[INFO] Finished at: 2025-06-29T06:58:46+05:30
```

Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

Calculator:

package com.example;

public class Calculator {

public int add(int a, int b) {

    return a + b;

  }

}

App:

package com.example;

public class App

{

  public static void main(String[] args)

  {

```
        System.out.println( "Hello World!" );
    }
}
```

AppTest:

```
package com.example;
import static org.junit.Assert.assertTrue;
import org.junit.Test;
public class AppTest (
    @Test
    public void shouldAnswerWithTrue() {
        System.out.println("Running the test...");
        assertTrue(true);
    }
}
```

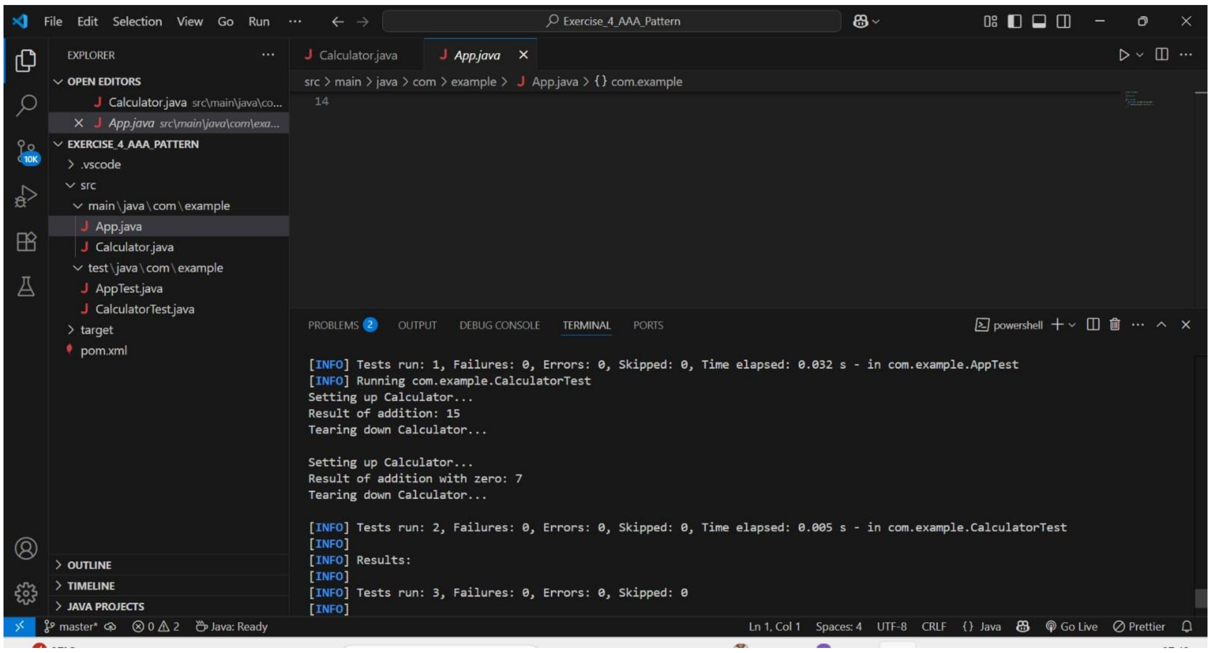
CalculatorTest:

```
package com.example;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class CalculatorTest {

    private Calculator calc;

    @Before
```

```
public void setUp() {
    System.out.println("Setting up Calculator...");
    calc = new Calculator(); // Arrange
}

@After
public void tearDown() {
    System.out.println("Tearing down Calculator...\n");
    calc = null;
}

@Test
public void testAddition() {
    // Act
    int result = calc.add(10, 5);

    // Assert
    assertEquals(15, result);
    System.out.println("Result of addition: " + result);
}

@Test
public void testAdditionWithZero() {
    // Act
    int result = calc.add(0, 7);

    // Assert
```

```
        assertEquals(7, result);
        System.out.println("Result of addition with zero: " + result);
    }
}
```

OUTPUT:



File Edit Selection View Go Run
Exercise_4_AAA_Pattern
EXPLORER
OPEN EDITORS
Calculator.java src\main\java\co...
App.java src\main\java\com\exa...
EXERCISE_4_AAA_PATTERN
.vscode
src
main \ java \ com \ example
App.java
Calculator.java
test \ java \ com \ example
AppTest.java
CalculatorTest.java
target
pom.xml
OUTLINE
TIMELINE
JAVA PROJECTS
Calculator.java
App.java
src > main > java > com > example > App.java > {} com.example
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell
master* ⊗ 0 ⚠ 2 Java: Ready
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Java Go Live Prettier

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.032 s - in com.example.AppTest
[INFO] Running com.example.CalculatorTest
Setting up Calculator...
Result of addition: 15
Tearing down Calculator...

Setting up Calculator...
Result of addition with zero: 7
Tearing down Calculator...

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.005 s - in com.example.CalculatorTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
```

Mockito exercises:
Exercise 1: Mocking and Stubbing

App:

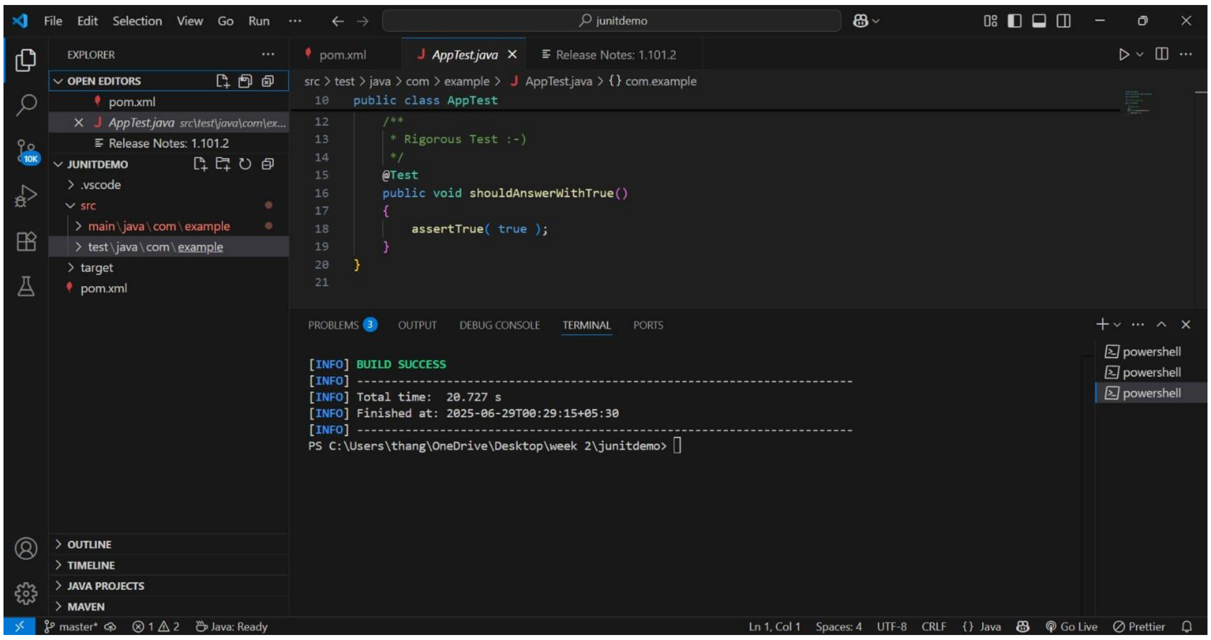
```
package com.example;
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

ExternalApi:

```
package com.example;

public interface ExternalApi {

    String getData();
```

```
}
MyService:
package com.example;

public class MyService {
    private ExternalApi api;

    public MyService(ExternalApi api) {
        this.api = api;
    }

    public String fetchData() {
        return api.getData();
    }
}
MyServiceTest:
package com.example;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

public class MyServiceTest {

    @Test
    public void testExternalApi() {
        // Step 1: Create mock
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        // Step 2: Stub method
        when(mockApi.getData()).thenReturn("Mock Data");

        // Step 3: Use mock in service
        MyService service = new MyService(mockApi);

        // Step 4: Call and assert
        String result = service.fetchData();
        assertEquals("Mock Data", result);

        System.out.println("Result: " + result);
    }
}
```

OUTPUT:



```
public class AppTest
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.727 s
[INFO] Finished at: 2025-06-29T00:29:15+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\thang\OneDrive\Desktop\week 2\junitdemo>
```

2) Verifying Interactions:

App:

```
package com.example;
public class App
{
   public static void main( String[] args )
   {
      System.out.println( "Hello World!" );
   }
}
```

ExternalApi:

```
package com.example;

public interface ExternalApi {

   String getData();

}
```

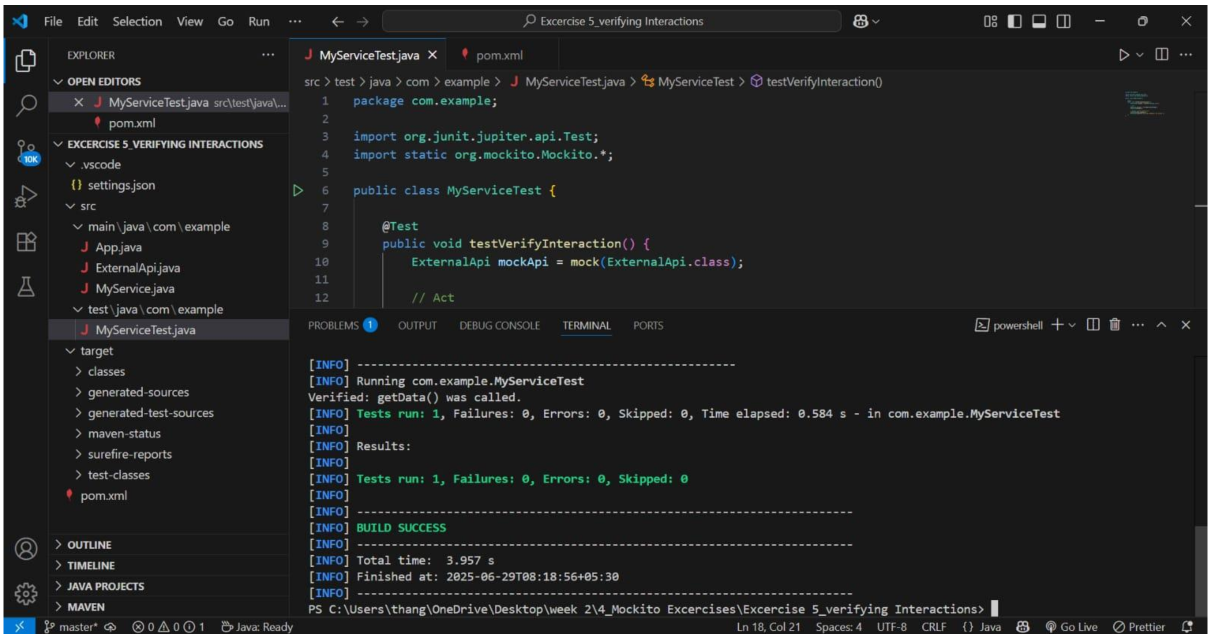
MyService:

```
package com.example;


public class MyService {

   private ExternalApi api;


   public MyService(ExternalApi api) {

      this.api = api;

   }


   public String fetchData() {

      return api.getData();

   }

}
```

MyServiceTest:

```
package com.example;
```

```
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;

public class MyServiceTest {

    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = mock(ExternalApi.class);

        // Act
        MyService service = new MyService(mockApi);
        service.fetchData();

        // Assert: verify interaction
        verify(mockApi).getData();
        System.out.println("Verified: getData() was called.");
    }
}
```
OUTPUT:


Excercise 5_verifying Interactions
MyServiceTest.java
pom.xml
src > test > java > com > example > MyServiceTest.java > MyServiceTest > testVerifyInteraction()
EXPLORER
OPEN EDITORS
EXCERCISE 5_VERIFYING INTERACTIONS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell

```java
package com.example;

import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;

public class MyServiceTest {

    @Test
    public void testVerifyInteraction() {
        ExternalApi mockApi = mock(ExternalApi.class);

        // Act
```

```
[INFO] -------------------------------------------------------
[INFO] Running com.example.MyServiceTest
Verified: getData() was called.
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.584 s - in com.example.MyServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.957 s
[INFO] Finished at: 2025-06-29T08:18:56+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\thang\OneDrive\Desktop\week 2\4_Mockito Excercises\Excercise 5_verifying Interactions>
```

6. SL4J Logging exercises

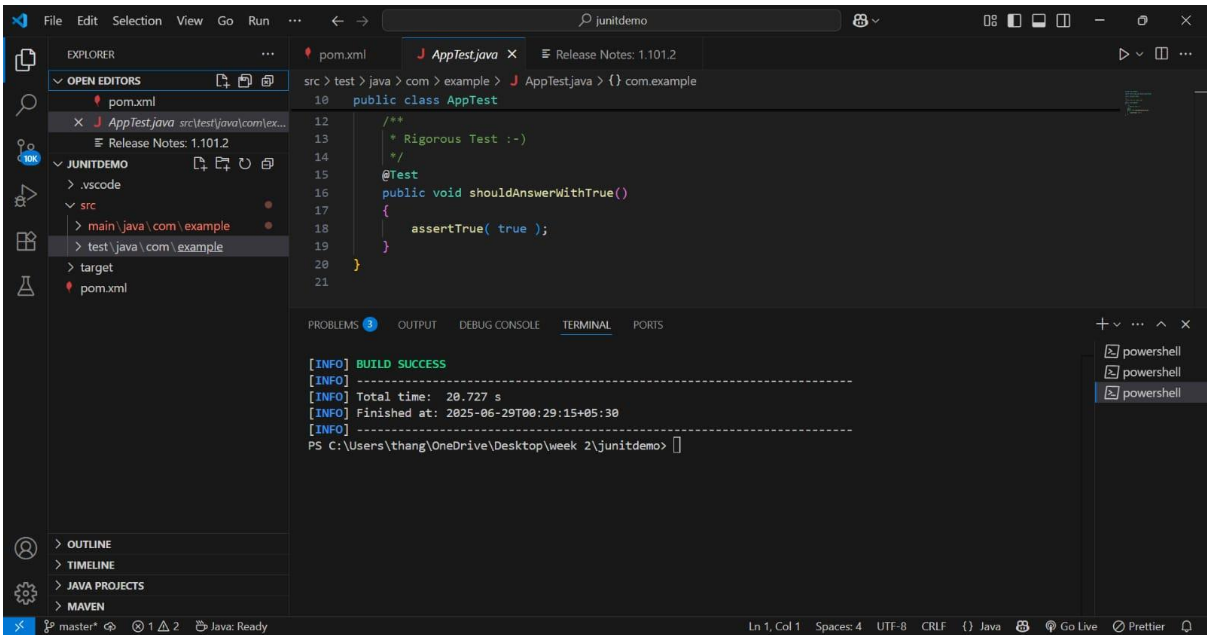
Exercise 1: Logging Error Messages and Warning Levels

App.java:

```java
package com.example;

public class App

{

    public static void main( String[] args )

    {

        System.out.println( "Hello World!" );

    }

}
```

LoggingExample:

```java
package com.example;


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("This is an error message");
        logger.warn("This is a warning message");
    }
}
```
OUTPUT:



```java
public class AppTest
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.727 s
[INFO] Finished at: 2025-06-29T00:29:15+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\thang\OneDrive\Desktop\week 2\junitdemo>
```