

1.JUnit_Basic Testing Exercises

Exercise 1: Setting Up Junit

OUTPUT

```
jdbc - JUNIT/src/com/example/CalculatorTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit X
finished after 0.089 seconds
Runs: 1/1 Errors: 0 Failures: 0
com.example.CalculatorTest [Runner: JUnit 4] (0.001 s)

1 package com.example;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*; // This is required for JUnit
5
6 public class CalculatorTest {
7
8     @Test
9     public void testAdd() {
10         Calculator calc = new Calculator(); // Class must be public
11         int result = calc.add(2, 3);
12         assertEquals(5, result); // Now assertEquals will work
13     }
14 }
15
```

Exercise 2: Writing Basic JUnit Tests

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;
```

```
}
```

```
    public int multiply(int a, int b) {
```

```
        return a * b;
```

```
}
```

```
    public int divide(int a, int b) {
```

```
    if (b == 0) throw new IllegalArgumentException("Division by zero");
    return a / b;
}
```

File name:CalculatorTest.java

```
package com.example;
```

```
import org.junit.Test;
```

```
import static org.junit.Assert.*;
```

```
public class CalculatorTest {
```

```
    Calculator calc = new Calculator();
```

```
    @Test
```

```
    public void testAdd() {
```

```
        assertEquals(7, calc.add(3, 4));
```

```
}
```

```
    @Test
```

```
    public void testSubtract() {
```

```
        assertEquals(2, calc.subtract(5, 3));
```

```
}
```

```
    @Test
```

```
    public void testMultiply() {
```

```
        assertEquals(20, calc.multiply(4, 5));
```

```
}
```

```
    @Test
```

```
    public void testDivide() {
```

```
        assertEquals(5, calc.divide(10, 2));
```

```
}
```

```

    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        calc.divide(10, 0); // This should throw exception
    }
}

```

OUTPUT

The screenshot shows the Eclipse IDE interface with the JUnit view open. The status bar at the bottom left indicates "Finished after 0.037 seconds". The code editor on the right displays the `CalculatorTest.java` file with the following content:

```

8     Calculator calc = new Calculator();
9
10    @Test
11    public void testAdd() {
12        assertEquals(7, calc.add(3, 4));
13    }
14
15    @Test
16    public void testSubtract() {
17        assertEquals(2, calc.subtract(5, 3));
18    }
19
20    @Test
21    public void testMultiply() {
22        assertEquals(20, calc.multiply(4, 5));
23    }
24
25    @Test
26    public void testDivide() {
27        assertEquals(5, calc.divide(10, 2));
28    }
29
30    @Test(expected = IllegalArgumentException.class)
31    public void testDivideByZero() {
32        calc.divide(10, 0); // This should throw exception
33    }

```

Exercise 3: Assertions in Junit

CODE

File name: AssertionsTest.java

```
package com.example;
```

```
import org.junit.Test;
import static org.junit.Assert.*;
```

```
public class AssertionsTest {
```

```
    @Test
```

```
    public void testAssertions() {
```

```
        // assertEquals - check for equality
```

```
assertEquals(5, 2 + 3);

// assertTrue - check if condition is true
assertTrue(10 > 5);

// assertFalse - check if condition is false
assertFalse(3 > 10);

// assertNull - check if value is null
String nullStr = null;
assertNull(nullStr);

// assertNotNull - check if value is not null
String name = "JUnit";
assertNotNull(name);

// assertSame - check if two references point to the same object
String a = "Hello";
String b = a;
assertSame(a, b);

// assertNotSame - check if two references do not point to the same object
String x = new String("World");
String y = new String("World");
assertNotSame(x, y);
}

}
```

OUTPUT

The screenshot shows the Eclipse IDE interface with the following details:

- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project navigation.
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Left Sidebar:** Package Explorer (selected), JUnit (green checkmark icon).
- Bottom Status Bar:** Runs: 1/1, Errors: 0, Failures: 0.
- Central Editor:** Displays the code for `AssertionsTest.java`. The code contains various JUnit assertions like assertEquals, assertTrue, assertFalse, assertNull, assertNotNull, and assertSame.
- Bottom Left:** Failure Trace button.
- Bottom Right:** Icons for saving, closing, and other file operations.

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

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: CalculatorTest.java

```
package com.example;
```

```
import org.junit.Before;
```

```
import org.junit.After;
```

```
import org.junit.Test;
```

```
import static org.junit.Assert.*;
```

```
public class CalculatorTest {
```

```
    private Calculator calc;
```

```
    // setup method - runs BEFORE each test
```

```
    @Before
```

```
    public void setUp() {
```

```
        System.out.println("Setting up Calculator...");
```

```
        calc = new Calculator();
```

```
}
```

```
    // Teardown method - runs AFTER each test
```

```
    @After
```

```
    public void tearDown() {
```

```
        System.out.println("Cleaning up...");
```

```
        calc = null; //
```

```
}
```

```
    @Test
```

```
    public void testAddition() {
```

```
        // Act
```

```
        int result = calc.add(5, 3);
```

```
        // Assert
```

```
        assertEquals(8, result);
```

```
}
```

```
@Test  
public void testSubtraction() {  
    int result = calc.subtract(10, 4);  
    assertEquals(6, result);  
}  
}
```

OUTPUT

The screenshot shows the Eclipse IDE interface with the JUnit View open. The JUnit View has tabs for Problems, Javadoc, Declaration, and Console. The Console tab is selected, displaying the output of the test run:

```
<terminated> CalculatorTest [JUnit] C:\Users\Nelampal\p:  
Setting up Calculator...  
Cleaning up...  
Setting up Calculator...  
Cleaning up...
```

Below the Console tab, the JUnit View displays the results of the test run:

- Package Explorer tab is visible.
- JUnit tab is selected.
- Run status: Finished after 0.037 seconds.
- Test results: Runs: 2/2, Errors: 0, Failures: 0.
- Test suite: com.example.CalculatorTest [Runner: JUnit 4] (0.001 s)