JUnit\_Basic Testing Exercises

Exercise 1: Setting Up Junit

Code:

1)Pom.xml code:

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

         http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>junit-example</artifactId>

    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>4.13.2</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.8.1</version>

                <configuration>

                    <source>1.8</source>

                    <target>1.8</target>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>

2)Test class code:

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

    @Test

    public void testAddition() {

        int result = 2 + 3;

        assertEquals(5, result);

    }

    @Test

    public void testSubtraction() {

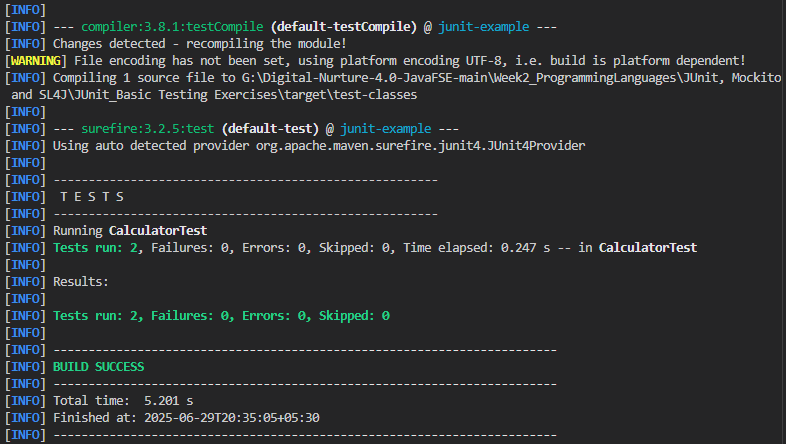
        int result = 10 - 4;

        assertEquals(6, result);

    }

}

Output:



Exercise 2: Writing Basic JUnit Tests:

Code:

1)MathUtils.java code:

public class MathUtils {

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("Cannot divide by zero");

return a / b;

}

}

2)MathUtilsTest.java code:

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

MathUtils math = new MathUtils();

@Test

public void testAdd() {

assertEquals(5, math.add(2, 3));

}

@Test

public void testSubtract() {

assertEquals(4, math.subtract(7, 3));

}

@Test

public void testMultiply() {

assertEquals(12, math.multiply(3, 4));

}

@Test

public void testDivide() {

assertEquals(5, math.divide(10, 2));

}

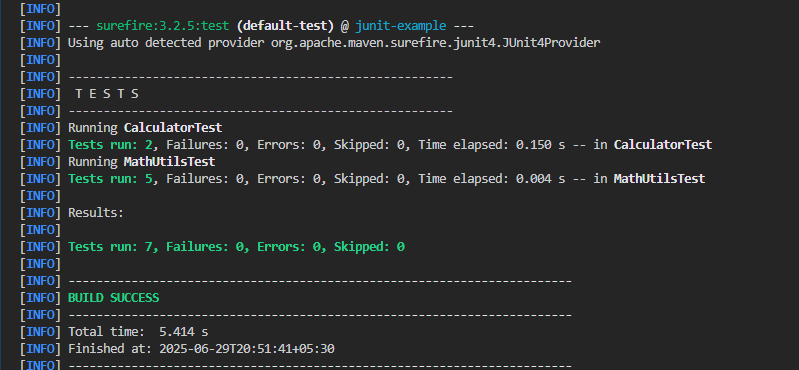
@Test(expected = IllegalArgumentException.class)

public void testDivideByZero() {

math.divide(10, 0);

}

}

Output: 

Exercise 3: Assertions in Junit:

Code:

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

    @Test

    public void testAssertions() {

        assertEquals("Sum should be 5", 5, 2 + 3);

        assertTrue("5 is greater than 3", 5 > 3);

        assertFalse("5 is not less than 3", 5 < 3);

        Object obj = null;

        assertNull("Object should be null", obj);

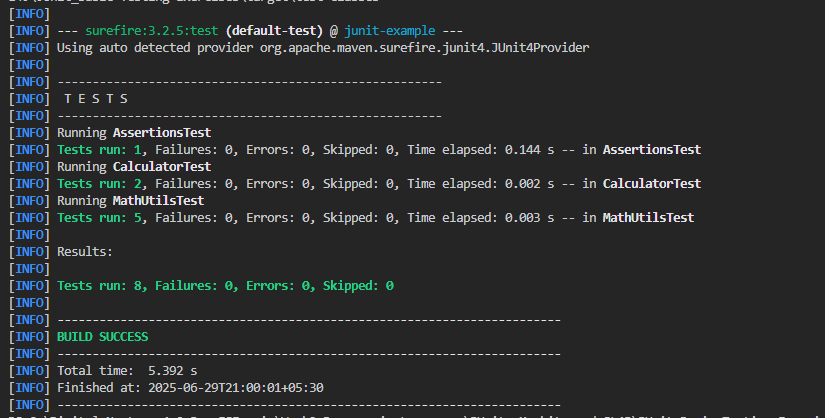
        Object notNullObj = new Object();

        assertNotNull("Object should not be null", notNullObj);

    }

}

Output:



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

Teardown Methods in Junit:

Code:

1)bankAccount.java code:

public class BankAccount {

    private int balance;

    public BankAccount(int initialBalance) {

        this.balance = initialBalance;

    }

    public void deposit(int amount) {

        balance += amount;

    }

    public void withdraw(int amount) {

        if (amount > balance) throw new IllegalArgumentException("Insufficient funds");

        balance -= amount;

    }

    public int getBalance() {

        return balance;

    }

    public void closeAccount() {

        balance = 0;

    }

}

2)BankAccountTest.java Code:

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class BankAccountTest {

    private BankAccount account;

    @Before

    public void setUp() {

        account = new BankAccount(100);

    }

    @After

    public void tearDown() {

        account.closeAccount();

    }

    @Test

    public void testDeposit() {

        account.deposit(50);

        assertEquals(150, account.getBalance());

    }

    @Test

    public void testWithdraw() {

        account.withdraw(40);

        assertEquals(60, account.getBalance());

    }

    @Test(expected = IllegalArgumentException.class)

    public void testWithdrawInsufficientFunds() {

        account.withdraw(200);

    }

}

Output: 