**WEEK-2**

**ADDITIONAL TASKS**

**1. Parameterized Tests**

**EvenChecker.java**

public class EvenChecker{

    public boolean isEven(int number){

        return number%2==0;

    }

}

**EvenCheckerTest.java**

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

import org.junit.jupiter.params.ParameterizedTest;

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

public class EvenCheckerTest{

    private final EvenChecker evenChecker=new EvenChecker();

    @ParameterizedTest

    @ValueSource(ints={2,4,6,8,10,100})

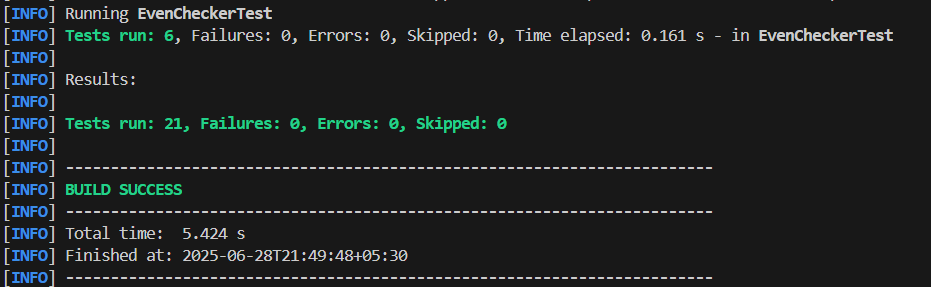
    void testIsEven(int number){

        assertTrue(evenChecker.isEven(number),number+" should be even");

    }

}

**Output:**

****

**2. Test Suites and Categories**

**EvenChecker.java**

package com.example;

public class EvenChecker{

    public boolean isEven(int number){

        return number%2==0;

    }

}

**EvenCheckerTest.java**

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

import org.junit.jupiter.params.ParameterizedTest;

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

public class EvenCheckerTest{

    private final EvenChecker evenChecker=new EvenChecker();

    @ParameterizedTest

    @ValueSource(ints={2,4,6,8,10,100})

    void testIsEven(int number){

        assertTrue(evenChecker.isEven(number),number+" should be even");

    }

}

**OddCheckerTest.java**

package com.example;

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

import org.junit.jupiter.params.ParameterizedTest;

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

public class OddCheckerTest{

    private final EvenChecker evenChecker=new EvenChecker();

    @ParameterizedTest

    @ValueSource(ints={1,3,5,7})

    void testIsNotEven(int number){

        assertFalse(evenChecker.isEven(number),number+" should be odd");

    }

}

**AllTests.java**

package com.example;

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

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

@Suite

@SelectClasses({

        EvenCheckerTest.class,

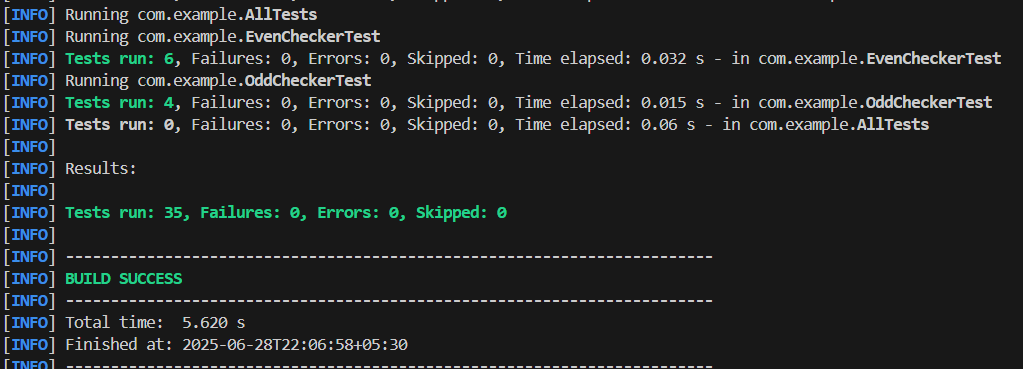
        OddCheckerTest.class

})

public class AllTests {

}

**Output:**

****

**3. Test Execution Order**

**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;

@TestMethodOrder(OrderAnnotation.class)

public class OrderedTests {

@Test

    @Order(1)

    void testFirst() {

        System.out.println("Running testFirst");

    }

@Test

    @Order(3)

    void testThird() {

        System.out.println("Running testThird");

    }

@Test

    @Order(2)

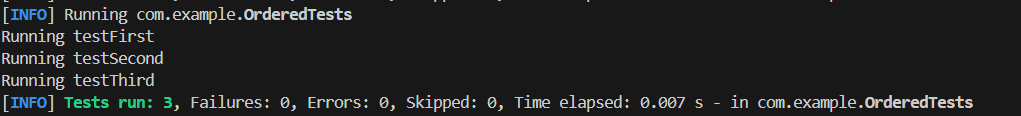
    void testSecond() {

        System.out.println("Running testSecond");

    }

}

**Output:**

****

**4. Exception Testing**

**ExceptionThrower.java**

package com.example;

public class ExceptionThrower{

public void throwException(){

        throw new IllegalArgumentException("This is an expected exception");

    }

}

**ExceptionThrowerTest.java**

package com.example;

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

import org.junit.jupiter.api.Test;

public class ExceptionThrowerTest{

    @Test

    void testThrowException(){

        ExceptionThrower thrower=new ExceptionThrower();

assertThrows(IllegalArgumentException.class,()->{

            thrower.throwException();

        });

    }

}

**Output:**

****

**5. Timeout and Performance Testing**

**PerformanceTester.java**

package com.example;

public class PerformanceTester{

    public void performTask(){

        try {

            Thread.sleep(300);

        } catch (InterruptedException e){

            Thread.currentThread().interrupt();

        }

    }

}

**PerformanceTesterTest.java**

package com.example;

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

import java.time.Duration;

import org.junit.jupiter.api.Test;

public class PerformanceTesterTest{

    @Test

    void testPerformTaskTimeout(){

        PerformanceTester tester=new PerformanceTester();

        assertTimeout(Duration.ofMillis(500),()->{

            tester.performTask();

        });

    }

}

**Output:**

