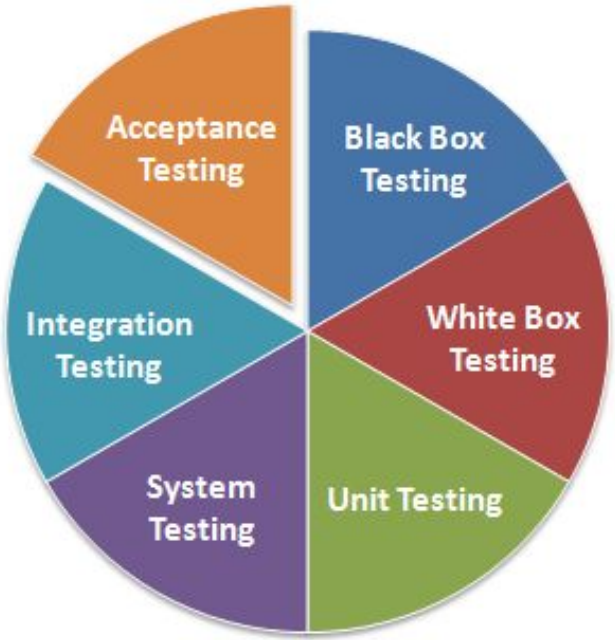
Write Unit Testing with JUnit

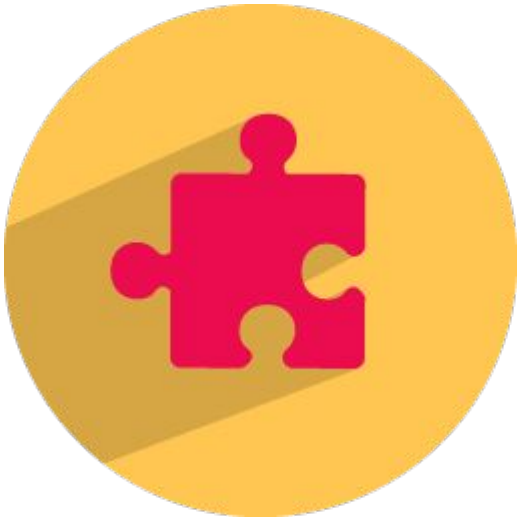
pokpitch.patcharadamrongkul@stream.co.th

Stream
it consulting

Type of Software Testing



Introduction to Unit Testing



Unit testing is a software testing method by which individual units of source code.

“Focus on testing whether a method follows the terms of its *API contract*”

“Confirm that the method accepts the expected range of input and that the returns the expected value for each input”

Starting from scratch

The simple calculator class

```
public class Calculator {
        public double add(double number1, double number2) {

                return number1 + number2;

        }

}
```

A Simple Test Calculator Program

```
public class CalculatorTest {
        public static void main(String[] args) {
                Calculator calculator = new Calculator();
                Double result = calculator.add(10,50);
                if(result != 60) {
                        System.out.println(“Bad result: ” + result);
                }
        }
}
```

A (Slightly) Better Test Program

```
public class CalculatorTest {

    private int nbError = 0;

    public void testAdd() {
        Calculator calculator = new Calculator();
        double result = calculator.add(10,50);
        if(result != 60) {
            throw new IllegalStateException("Bad result: " + result);
        }
    }
```

A (Slightly) Better Test Program (Cont)

```
        public static void main(String[] args) {
                CalculatorTest test = new CalculatorTest();
                try {
                        test.testAdd();
                } catch (Throwable e) {
                        test.nbError++;
                        e.printStackTrace();
                }
                if(test.nbError > 0) {
                        throw new IllegalStateException("There were " + test.nbError + "
                error(s)");
                }
        }
}
```

Unit Testing Best Practices

- Always Write Isolated Test Case
- Test One Thing Only In One Test Case
- Use a Single Assert Method per Test Case
- Use a Naming Conversions for Test Cases
- Use Arrange-Act-Assert or Given-When-Then Style

What is JUnit?

is a simple, open source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.

Latest Stable Version: 4.12 (18 April 2016)

Understanding Unit Testing frameworks

Unit testing frameworks should follow several best practices. These seemingly minor improvements in the *CalculatorTest* program highlight **three** rules that all unit testing frameworks should follow.

Each unit test run independently of all other unit tests.

The framework should detect and report errors test by test.

It should be easy to define which unit tests will run.

JUnit design goals

The JUnit team has defined three discrete goals for the framework

The framework must help us write useful tests.

The framework must help us create tests
that retain their value over time.

The framework must help us lower the cost of
writing tests by reusing code.

Downloading and Installing JUnit

Plain-Old JAR

maven

MAVEN

Plain-Old JAR

Download the following JARs and put them on your test classpath

- junit.jar
- hamcrest-core.jar

MAVEN

Add a dependency to junit:junit in test scope

pom.xml

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>unit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

Testing with JUnit

JUnit has many features that make it easy to write and run tests. You'll see these features at work throughout this example.

- Separate test class instances and class loaders for each unit test to avoid side effect.
- JUnit annotations to provide resource initialization and reclamation methods: @Before, @BeforeClass, @After and @AfterClass
- A variety of assert methods to make it easy to check the results of your tests.
- Integration with popular tools like Ant and Maven, and popular IDEs like Eclipse, NetBeans, IntelliJ, and JBuilder

The JUnit CalculatorTest Program

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

public class CalculatorTest {
	@Test
	public void testAdd() {
		Calculator calculator = new Calculator();
		double result = calculator.add(10,50);
		assertEquals(60, result, 0);
	}
}
```

Run JUnit Test Case (Command line)

```
D:\experiment\stream-it-junit4-training\002-junit-calculator-test>javac -cp junit-4.12.jar *.java

D:\experiment\stream-it-junit4-training\002-junit-calculator-test>ls
Calculator.class  CalculatorTest.class  hamcrest-core-1.3.jar
Calculator.java   CalculatorTest.java   junit-4.12.jar
```

```
D:\experiment\stream-it-junit4-training\002-junit-calculator-test>java -cp .;junit-4.12.jar;hamcrest
-core-1.3.jar org.junit.runner.JUnitCore CalculatorTest
JUnit version 4.12
.
Time: 0.007

OK (1 test)


D:\experiment\stream-it-junit4-training\002-junit-calculator-test>
```

Run JUnit Test Case (Maven)

```
D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building 003-junit-calculator-test-maven-project 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ 003-junit-calculator-test-maven-project ---
[WARNING] Using platform encoding (MS874 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ 003-junit-calculator-test-maven-project ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding MS874, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ 003-junit-calculator-test-maven-project ---
[WARNING] Using platform encoding (MS874 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ 003-junit-calculator-test-maven-project ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding MS874, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ 003-junit-calculator-test-maven-project ---
[INFO] Surefire report directory: D:\experiment\stream-it-junit4-training\003-junit-calculator-test-maven-project\target\surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.stream.CalculatorTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.082 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 7.252 s
```

Exploring Core JUnit

Mastering JUnit

Test Coverage and Development

Testing with Mock Objects

Running Unit Tests from MAVEN/ANT

Continuous Integration Tools

JUnit Extensions

Workshop