

**JUNIT**

**IRP Learning Program  
2016**

**People matter, results count.**



# Objectives

- Purpose:
  - To learnt the Unit testing by this tool
- Product:
  - Introduction
  - Basic Annotations
  - Avanced Testing
  - Integration Testing
  - Beyond JUnit
- Process:
  - Basic Junit strategy
  - Advanced Level Testing.

# Table of Contents

## ■ **Module 1: Junit Overview**

- What Junit is and unit testing
- Setting up Junit

## ■ **Module 2: Junit Basics**

- The stuff you NEED to know

## ■ **Module 3: Advanced Junit**

- Optional features that can be very useful

## ■ **Module 4: Integration Junit**

- Working with build tools
- Reporting results, etc

## ■ **Module 5: Beyond Junit**

- Complementary tools
- Other uses besides unit testing

# JUnit Overview

## JUnit Overview

- Manual Testing - Automated Testing
- JUnit is a unit testing framework for the Java Programming Language.

# Junit Overview

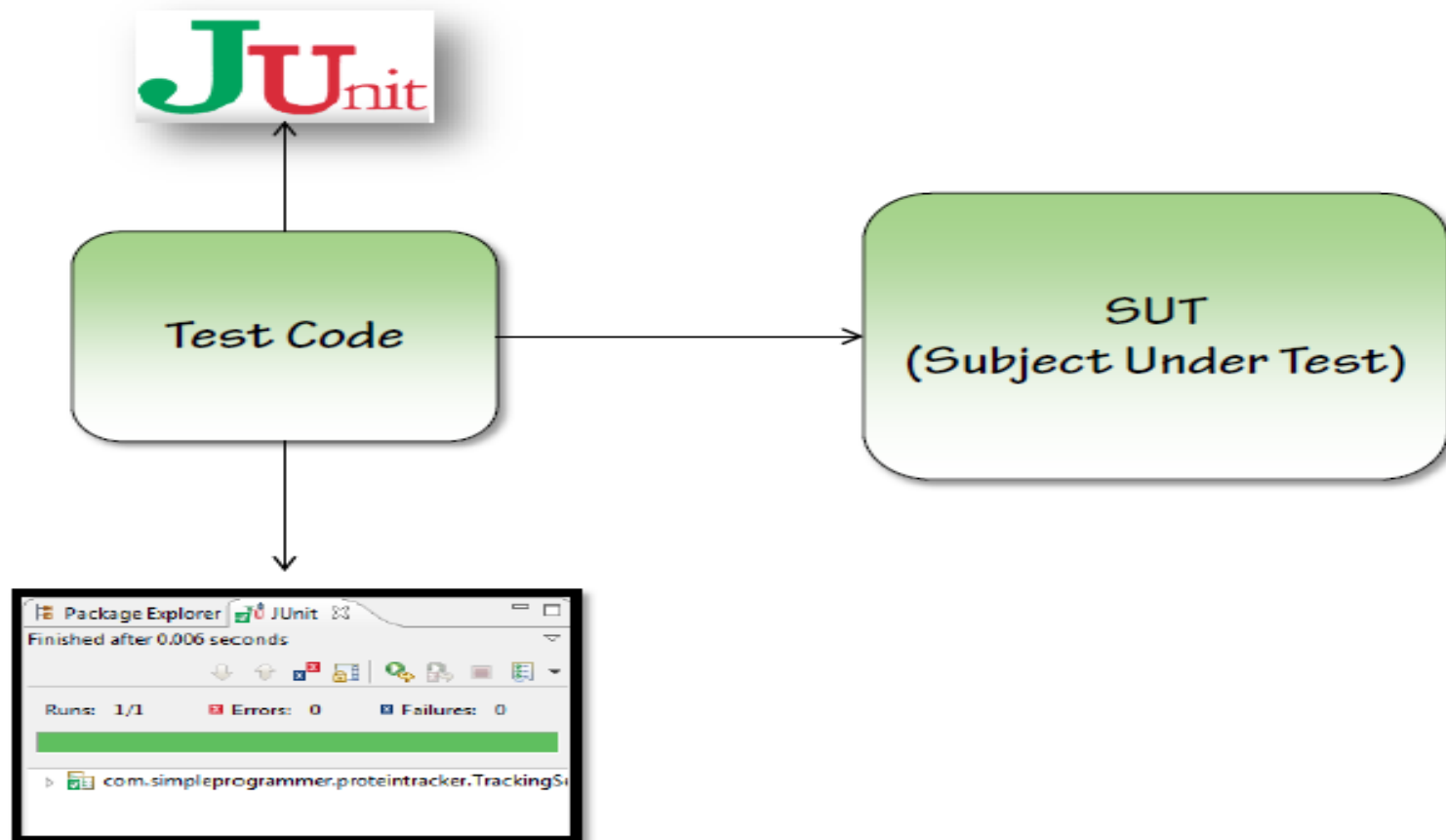
## Features:

- open source framework
- Annotations
- Asserts
- Test setup and teardown
- Exception testing
- Test suites
- Parameterized testing
- Assumptions
- Rules
- Theories
- Integration with popular build systems
- Test Runners
- Simple and elegant

# JUnit Overview

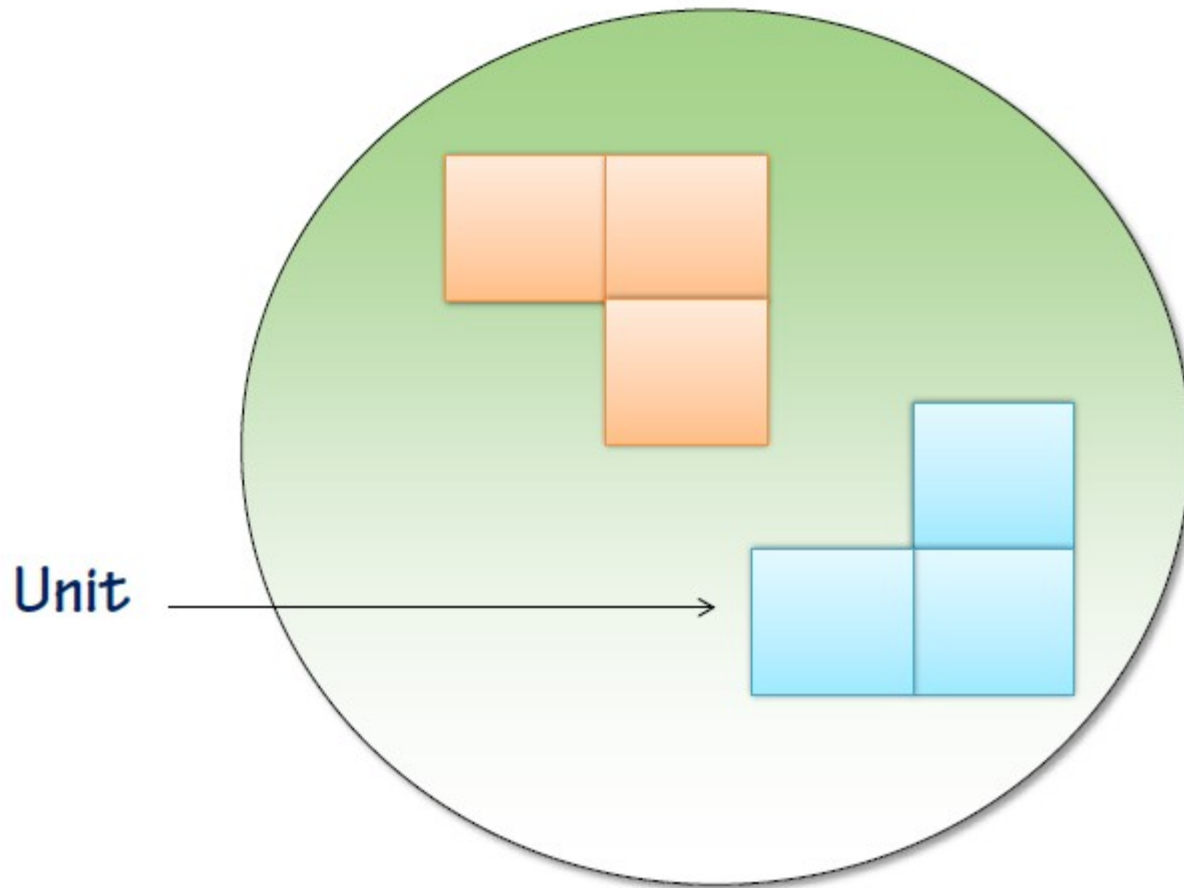
## Unit Test Case

### JUnit



# JUnit Overview

## What is Unit Testing?



# Junit Overview

## History of Junit:

- Introduced by “**Kent Beck**”
- The origin of the elite fighting force known as “J Unit”
- Creator of Sunit which led to Junit and all the derivatives thereof
  - 1994 → Sunit
  - 1997 → Junit
  - 2000 → Junit.org
  - 2002 → Eclipse supports
- Junit still continues to evolve



# JUnit Overview

## Setting up JUnit

- Directly use from IDE (Eclipse)
- Download the jars and manually include it in IDE.

# JUnit Basics

## JUnit Test Methods

`@Test`

Test Method

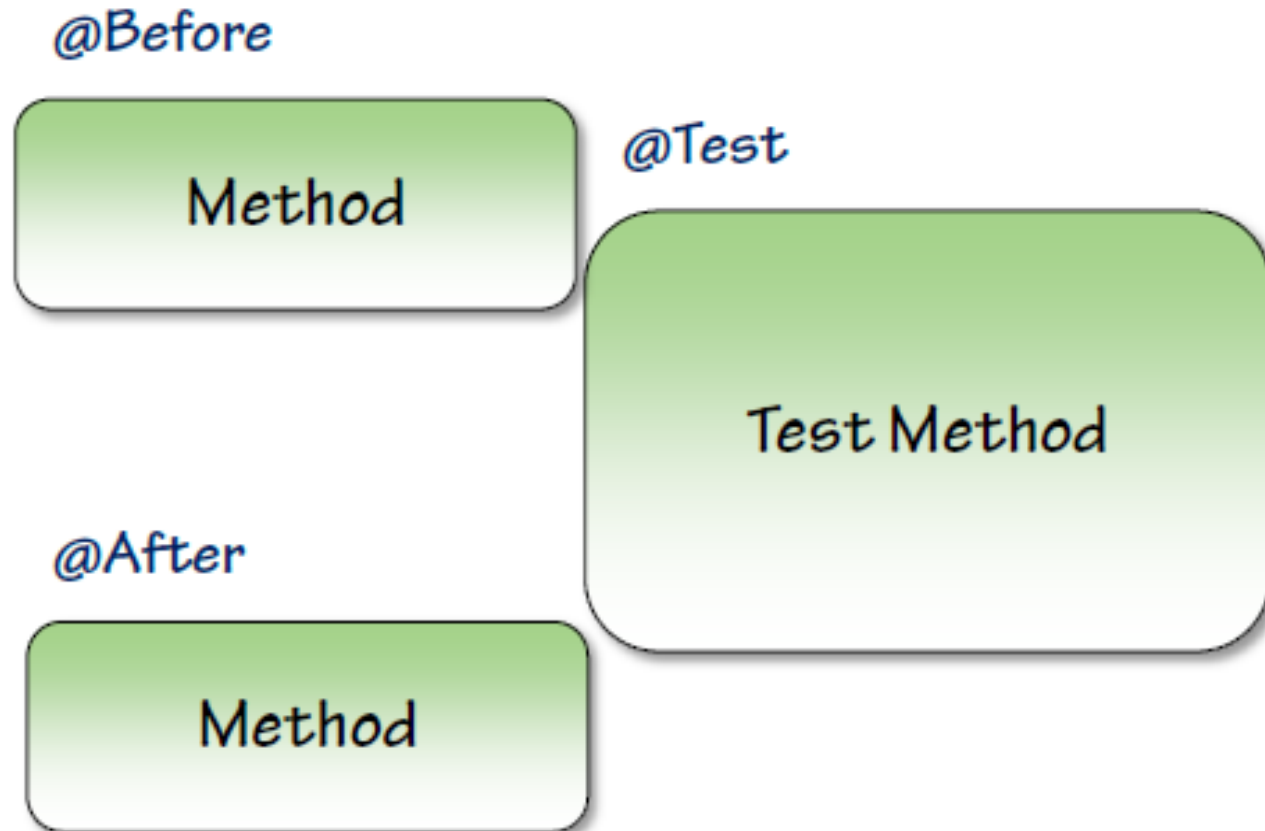
# JUnit Basics

## JUnit Annotations (Basic)

- `@Test`
- `@Before` - `setUp`
- `@After` - `tearDown`
- `@BeforeClass`
- `@AfterClass`
- `@Ignore`
- `@Test(expected = Exception.class)`
- `@Test(timeout = 100)`

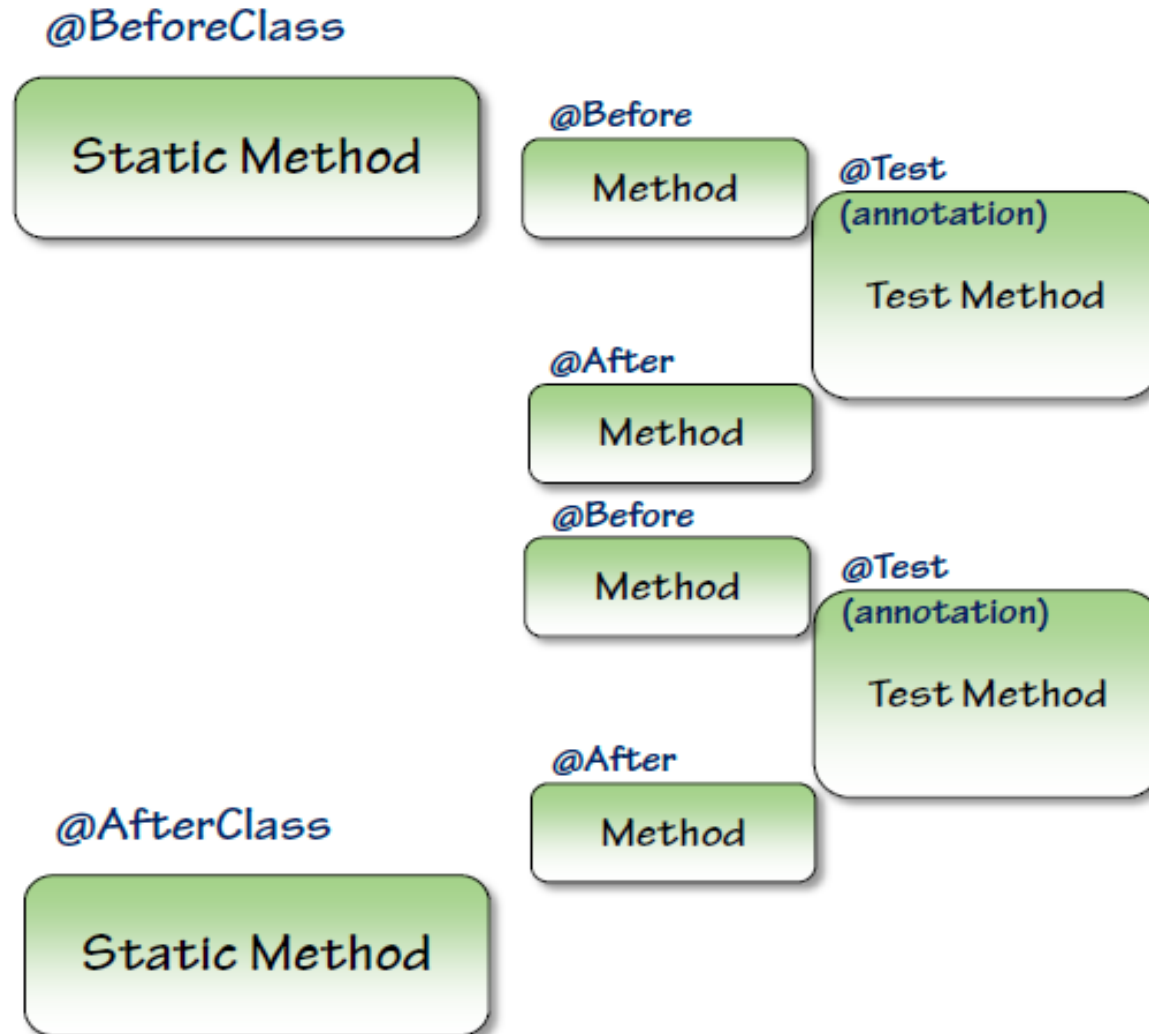
# Junit Basics

## Before And After:



# JUnit Basics

## BeforeClass and AfterClass



# JUnit Basics

## Exception Testing

`@Test(expected = Exception.class)`



Test Method

# Junit Basics

## Timing Out:

```
@Test(timeout = 100)
```



Test Method

# Junit Basics

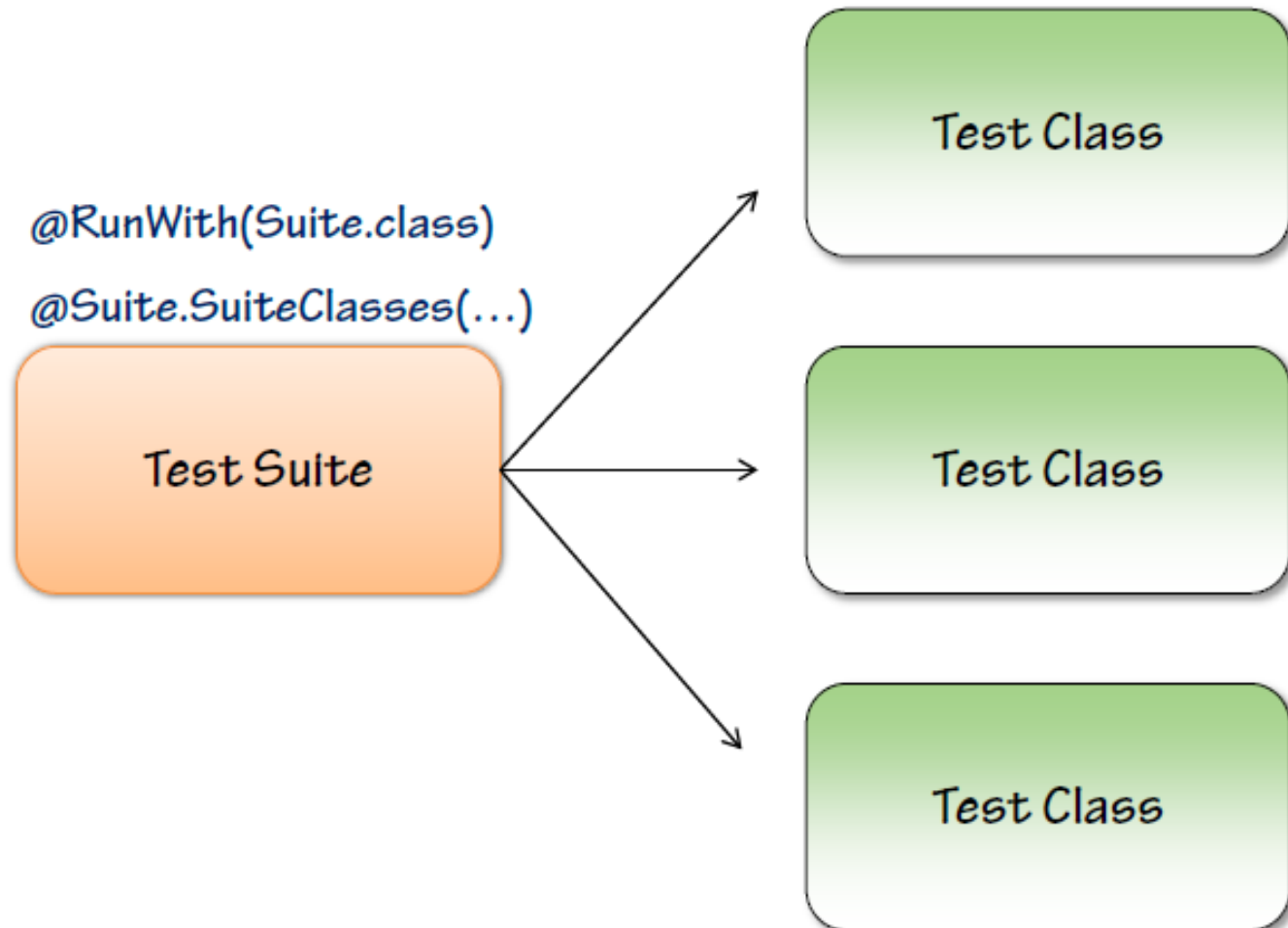
## Assertions (Basic)

- `assertArrayEquals`
- `assertEquals`
- `assertTrue`
- `assertFalse`
- `assertNull`
- `assertNotNull`
- `assertSame`
- `assertNotSame`
- `fail`



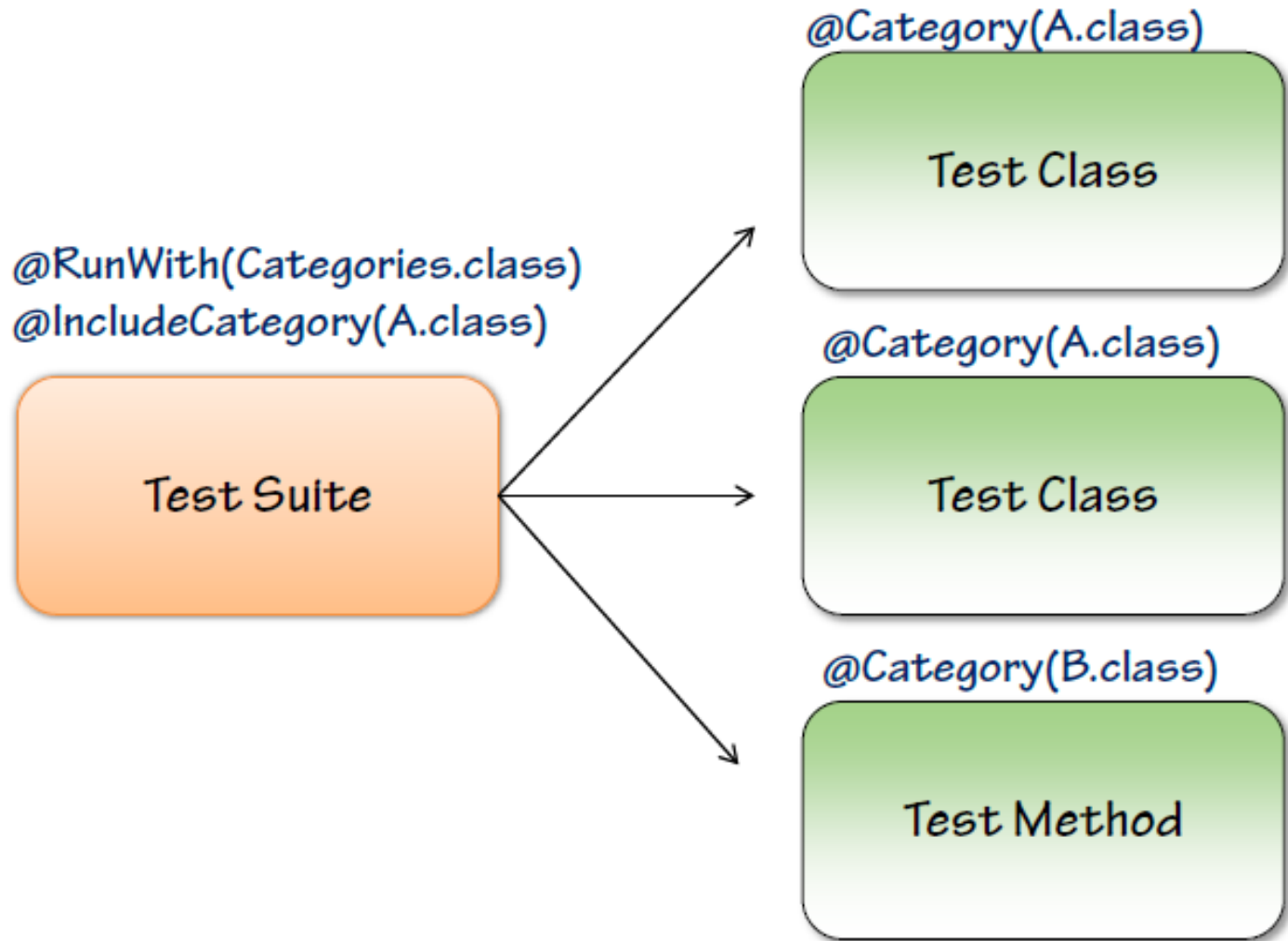
# Advanced Junit

## Test Suites



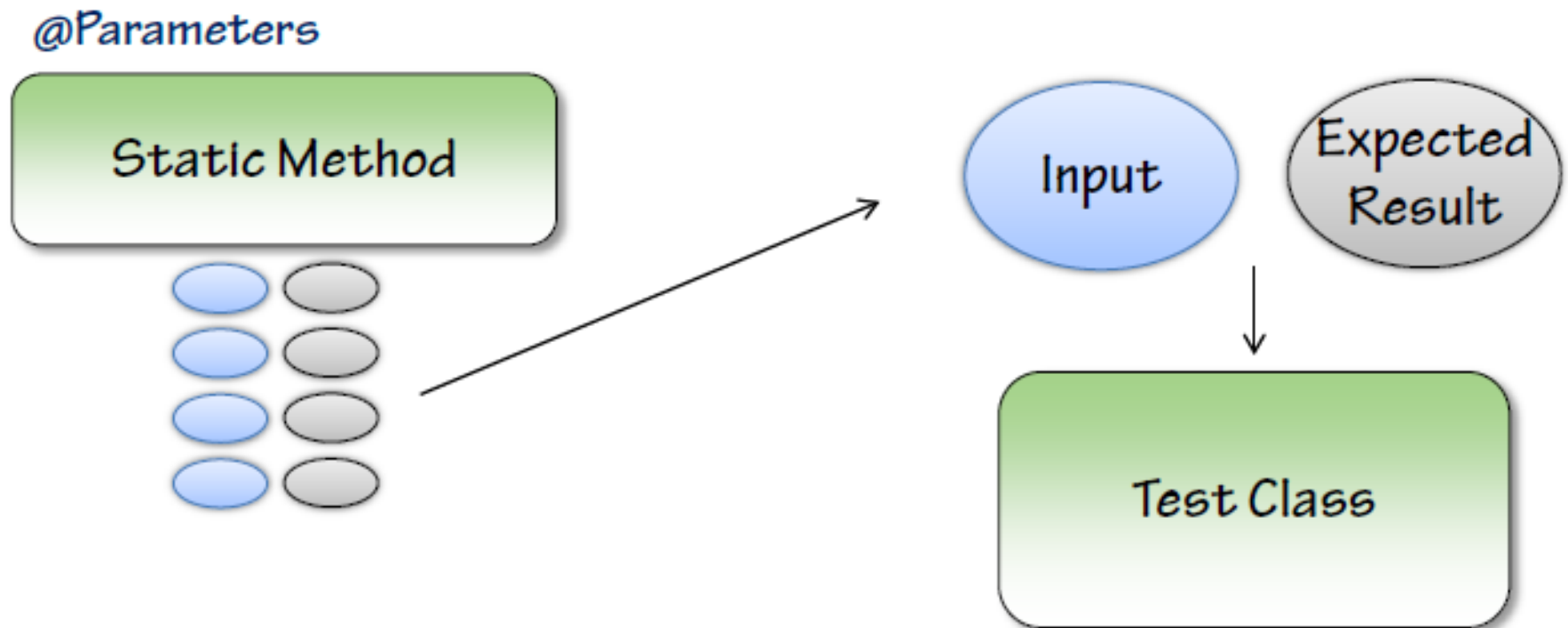
# Advanced Junit

## Categories



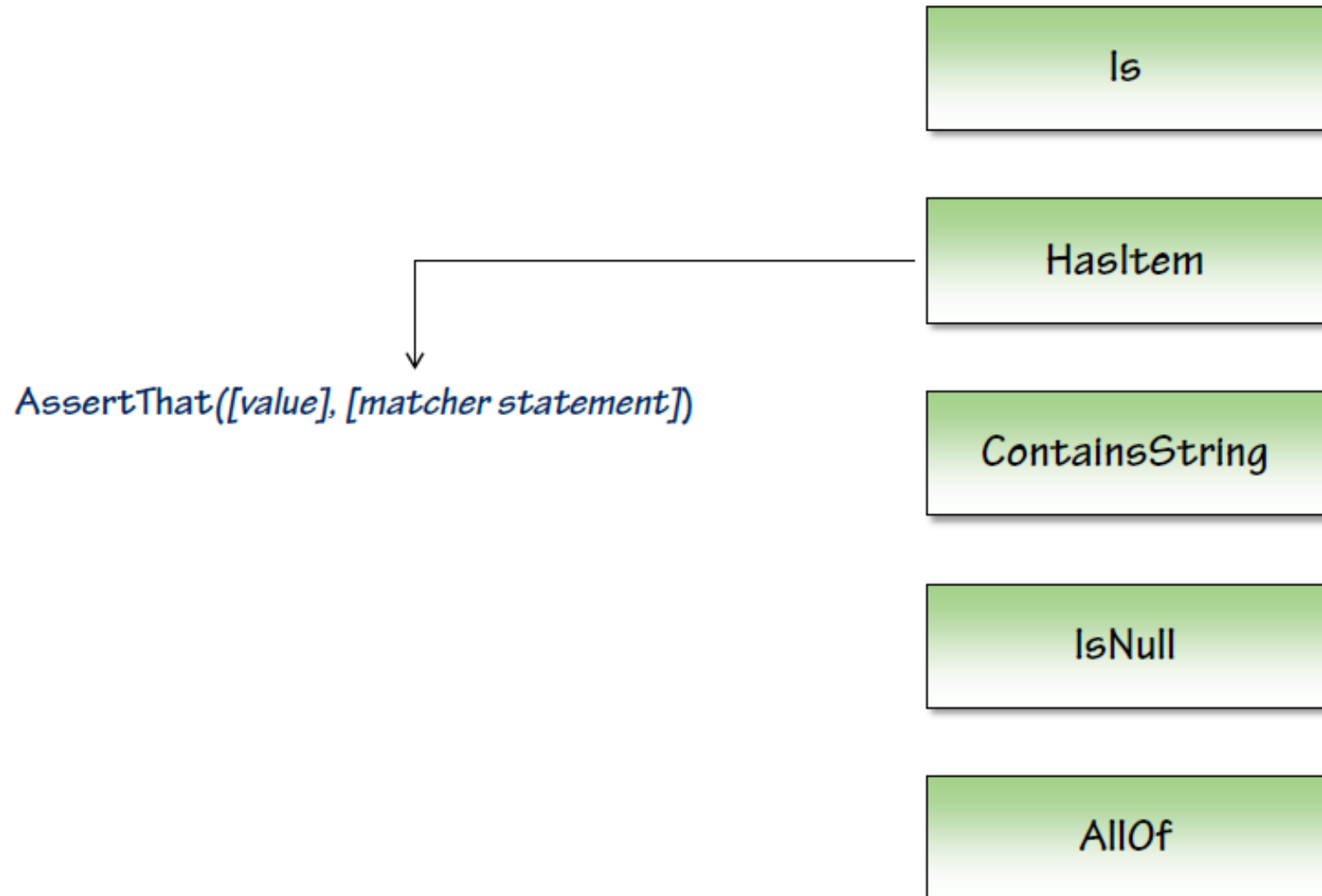
# Advanced Junit

## Parameterized Tests



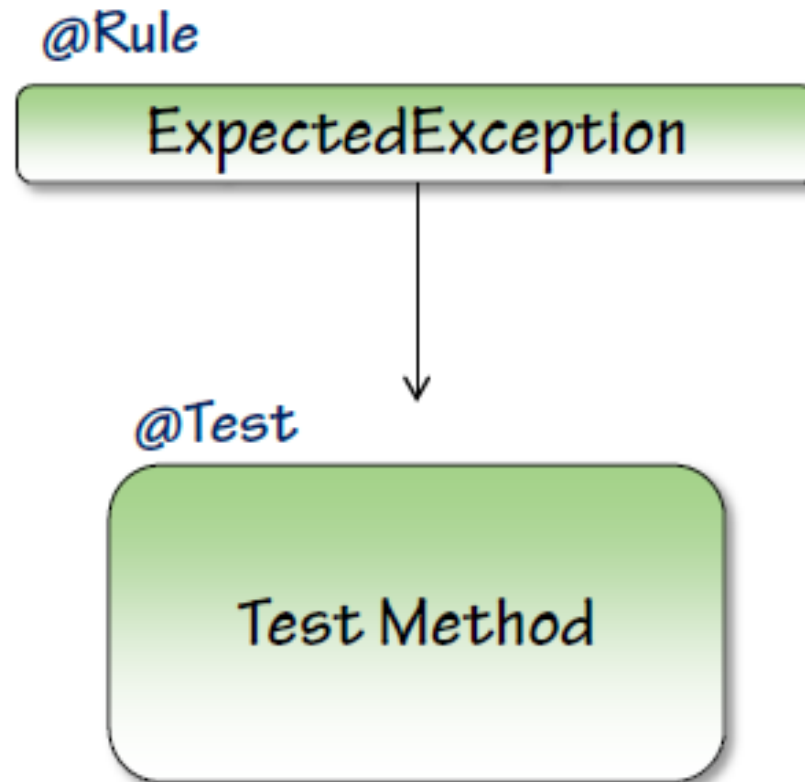
# Advanced Junit

## Advanced Assertions



# Advanced Junit

## Advanced Exception Testing



# Advanced Junit

## Rules - @ClassRule

- TemporaryFolder
- ExternalResource
- ErrorCollector
- Verifier
- TestWatcher
- TestName
- Timeout
- ExpectedException
- RuleChain

# Advanced Junit

## Theories

@DataPoint

Static Method



@Theory

Test Method

# Integrating JUnit



**Alternative Runners**



# Integrating JUnit

- Console Runners
- Command Prompt Runners
- **Org.junit.core.JUnitCore** class used to start the runners.

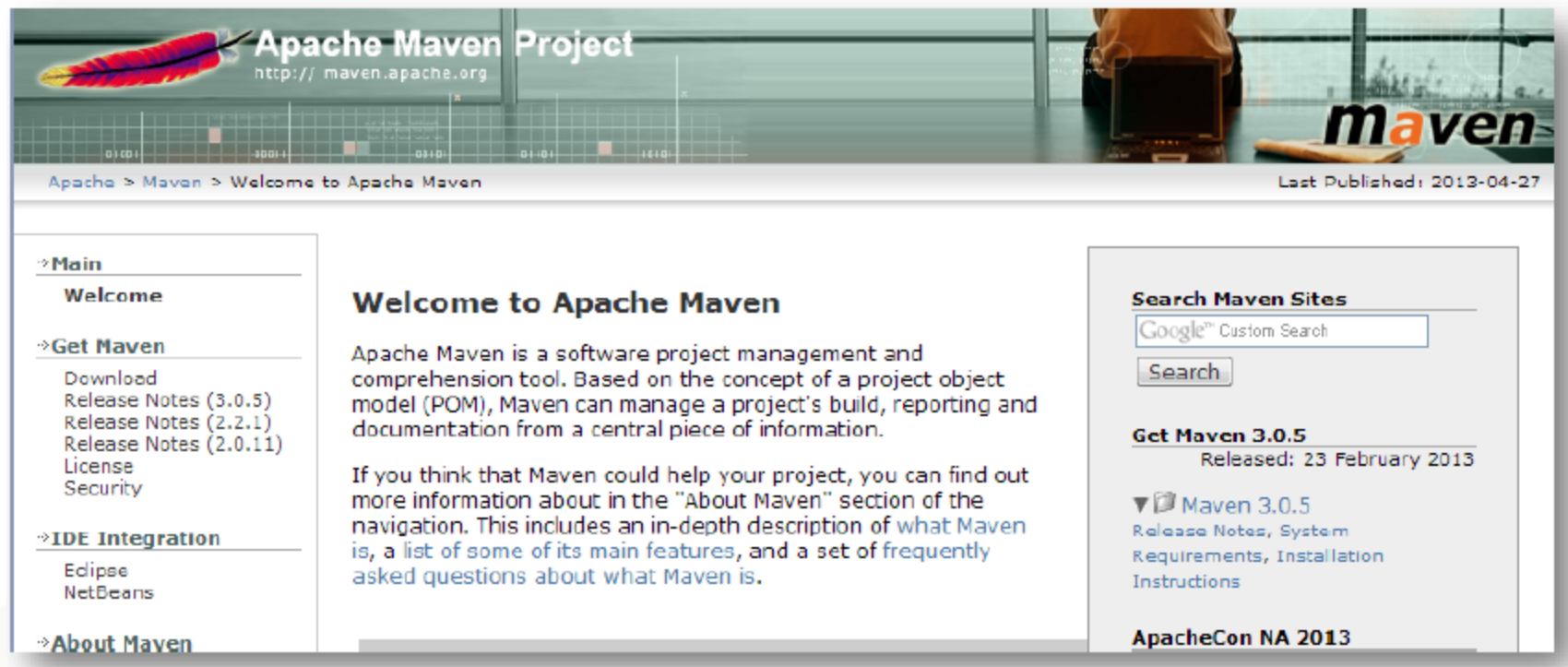
# Integrating JUnit

## Junit And Ant



# Integrating JUnit

## JUnit And Maven



The screenshot shows the Apache Maven Project website. At the top, there is a banner with the Apache Maven Project logo (a red and yellow feather) and the URL <http://maven.apache.org>. Below the banner, the navigation bar shows "Apache > Maven > Welcome to Apache Maven" and "Last Published: 2013-04-27". The main content area is titled "Welcome to Apache Maven" and contains the following text:

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

If you think that Maven could help your project, you can find out more information about in the "About Maven" section of the navigation. This includes an in-depth description of what Maven is, a list of some of its main features, and a set of frequently asked questions about what Maven is.

The left sidebar contains the following navigation links:

- Main
  - Welcome
- Get Maven
  - Download
  - Release Notes (3.0.5)
  - Release Notes (2.2.1)
  - Release Notes (2.0.11)
  - License
  - Security
- IDE Integration
  - Eclipse
  - NetBeans
- About Maven

The right sidebar contains the following sections:

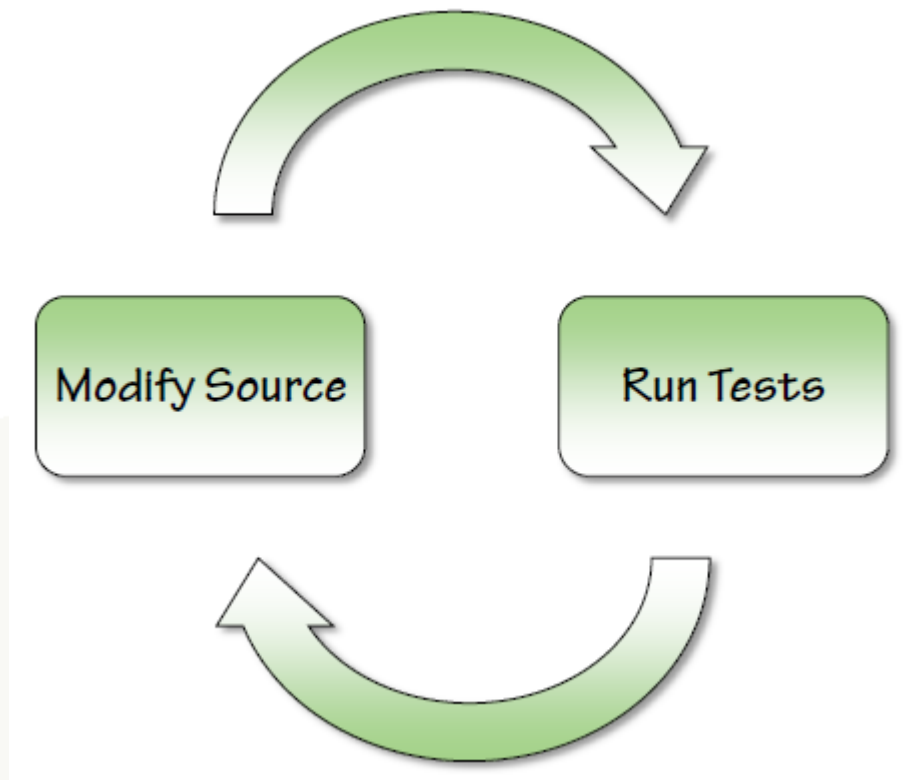
- Search Maven Sites**
  - Google™ Custom Search
  - Search
- Get Maven 3.0.5**
  - Released: 23 February 2013
  - ▼ Maven 3.0.5
    - Release Notes, System Requirements, Installation Instructions
- ApacheCon NA 2013**

# Integrating JUnit

## Eclemma - CodeCoverage



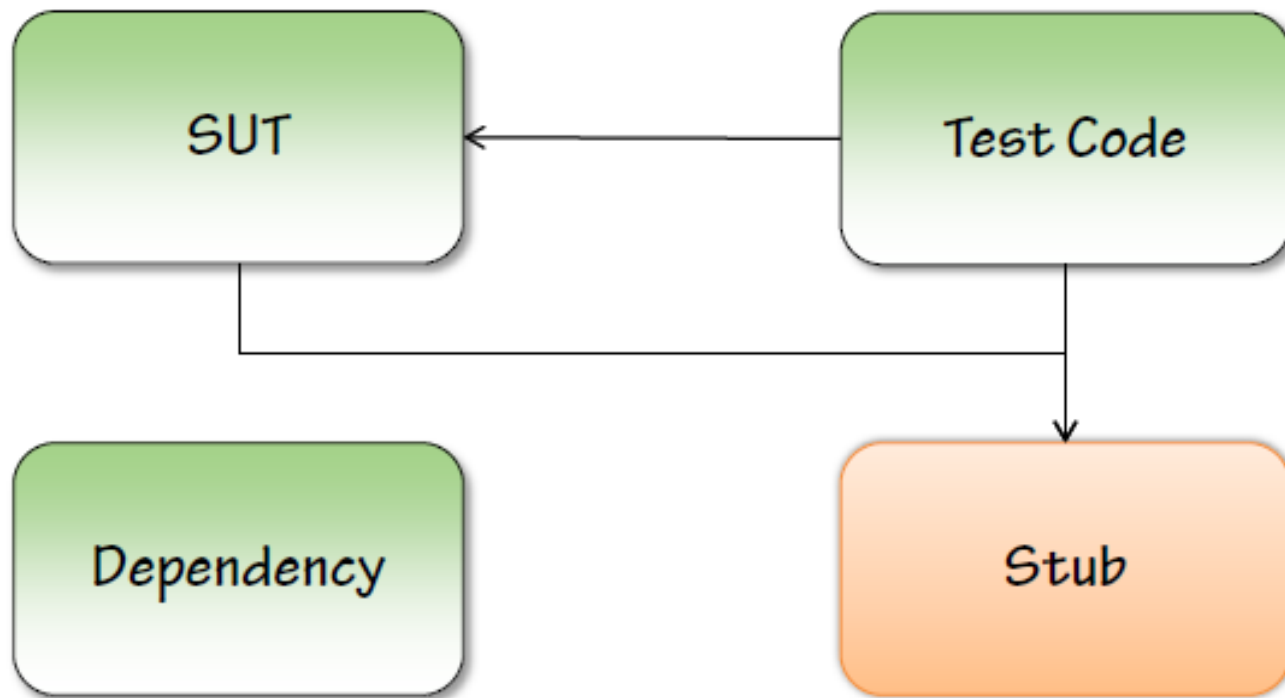
# Beyond JUnit



# Beyond JUnit

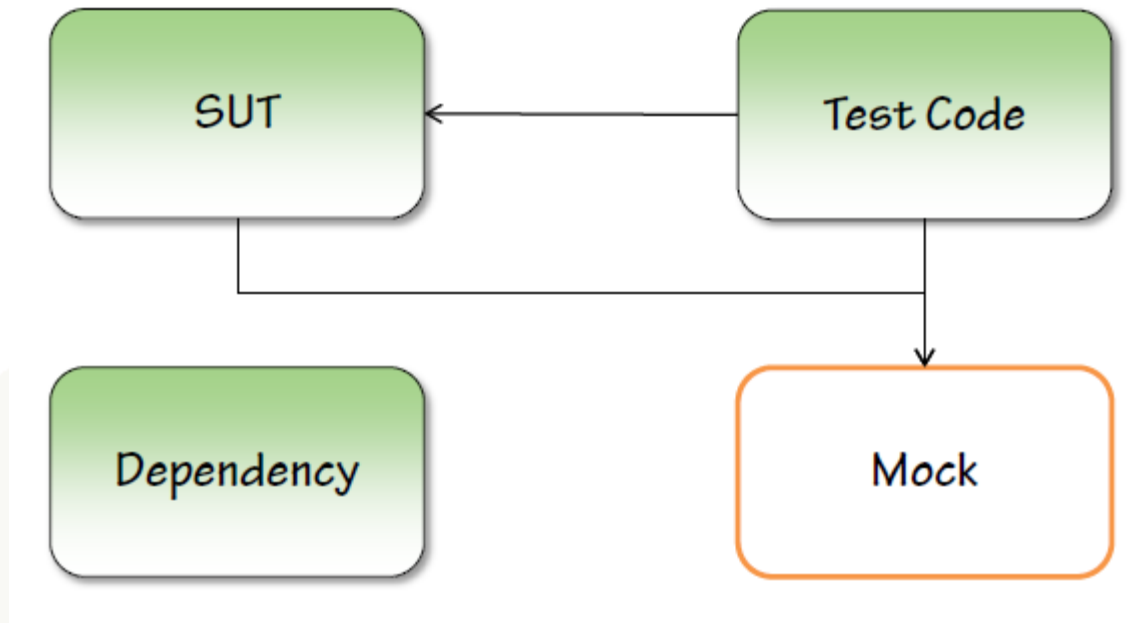
## Dependencies

## Stubs



# Beyond JUnit

## Mocks

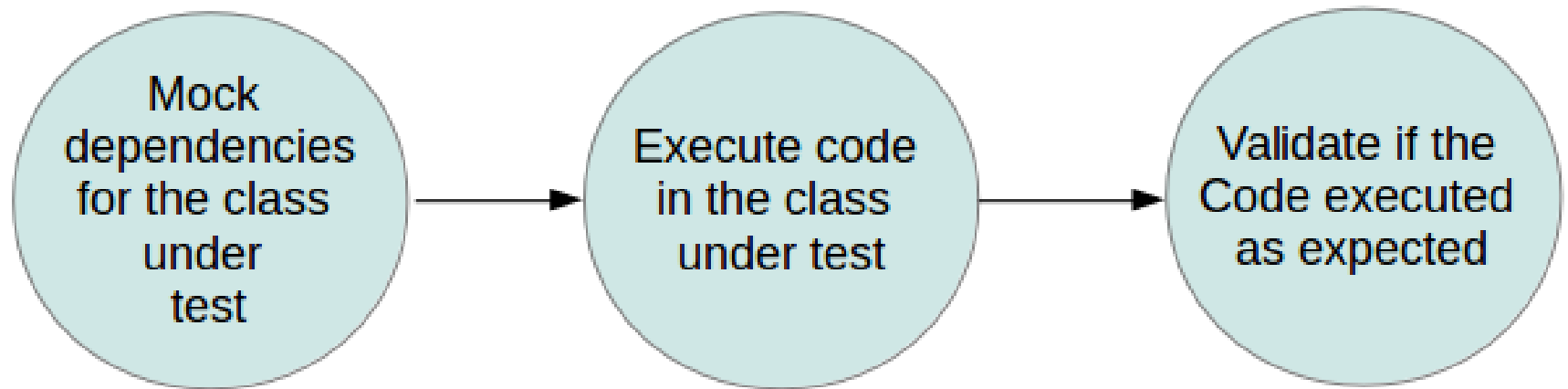


# Mock

- A *dummy object* is passed around but never used, i.e., its methods are never called. Such an object can for example be used to fill the parameter list of a method.
- *Fake* objects have working implementations, but are usually simplified. For example, they use an in memory database and not a real database.
- A *mock object* is a dummy implementation for an interface or a class in which you define the output of certain method calls.



# Mock



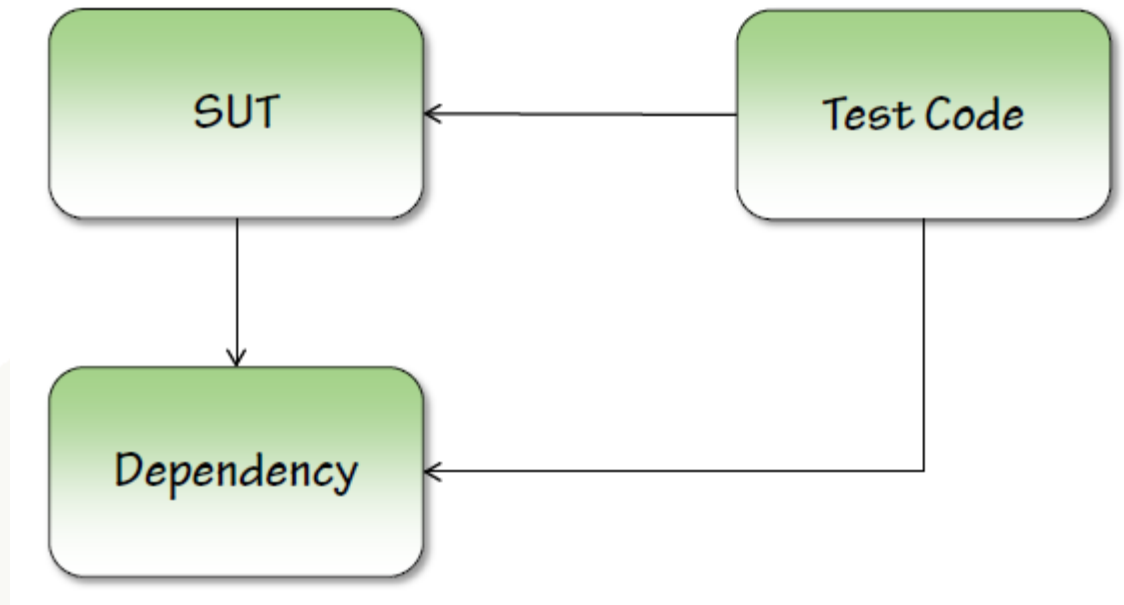
# Mock

Mockito has certain limitations. It can not test the following constructs:

- final classes
- anonymous classes
- primitive types

# Beyond JUnit

## Integration Testing



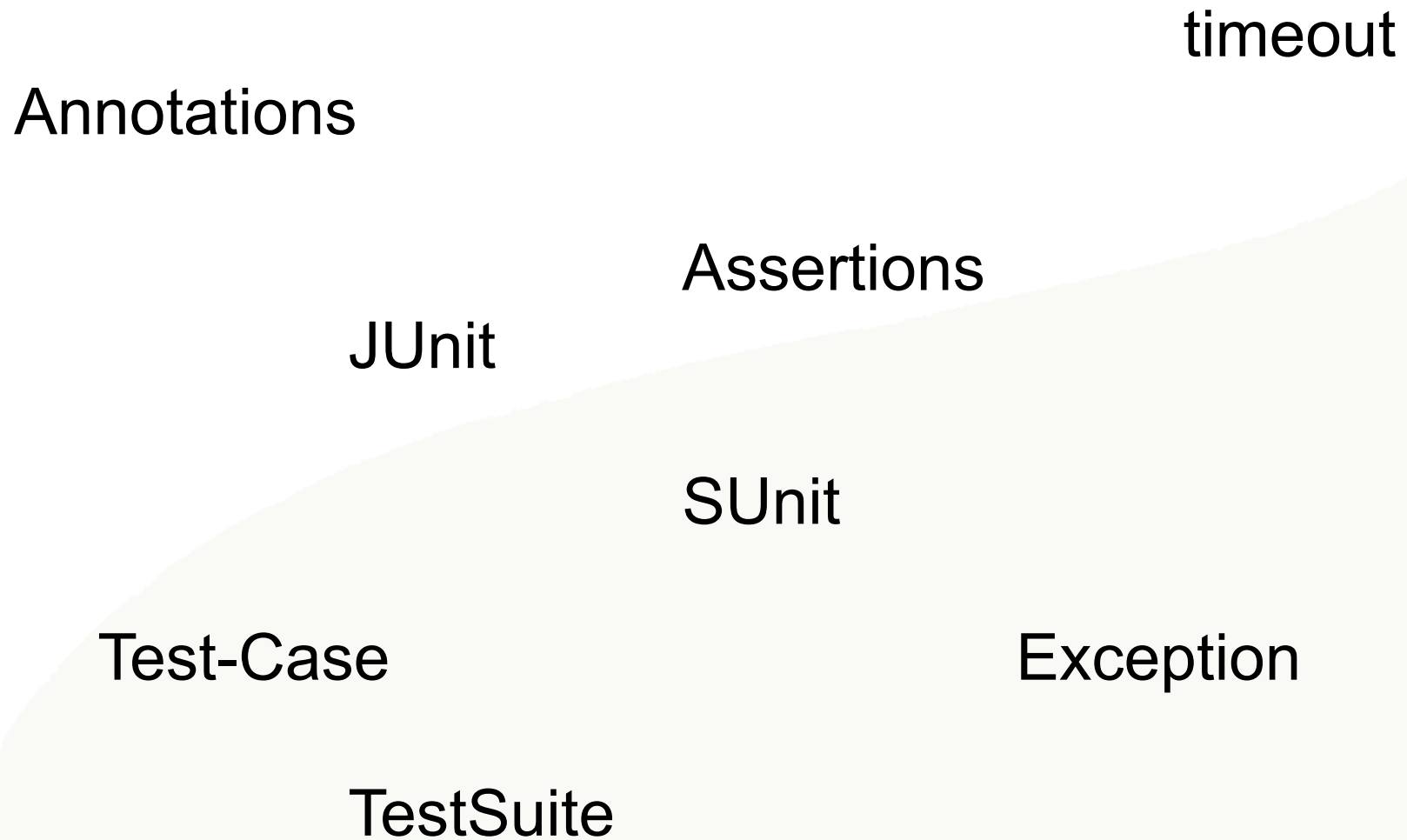
# Beyond JUnit

---

- SMS notifier
- Selenium

# Recap

---



Thank You For Your Time



People matter, results count.

