# Software Design and Construction 2
# SOFT3202 / COMP9202

Introduction

Software Testing

Dr. Basem Suleiman
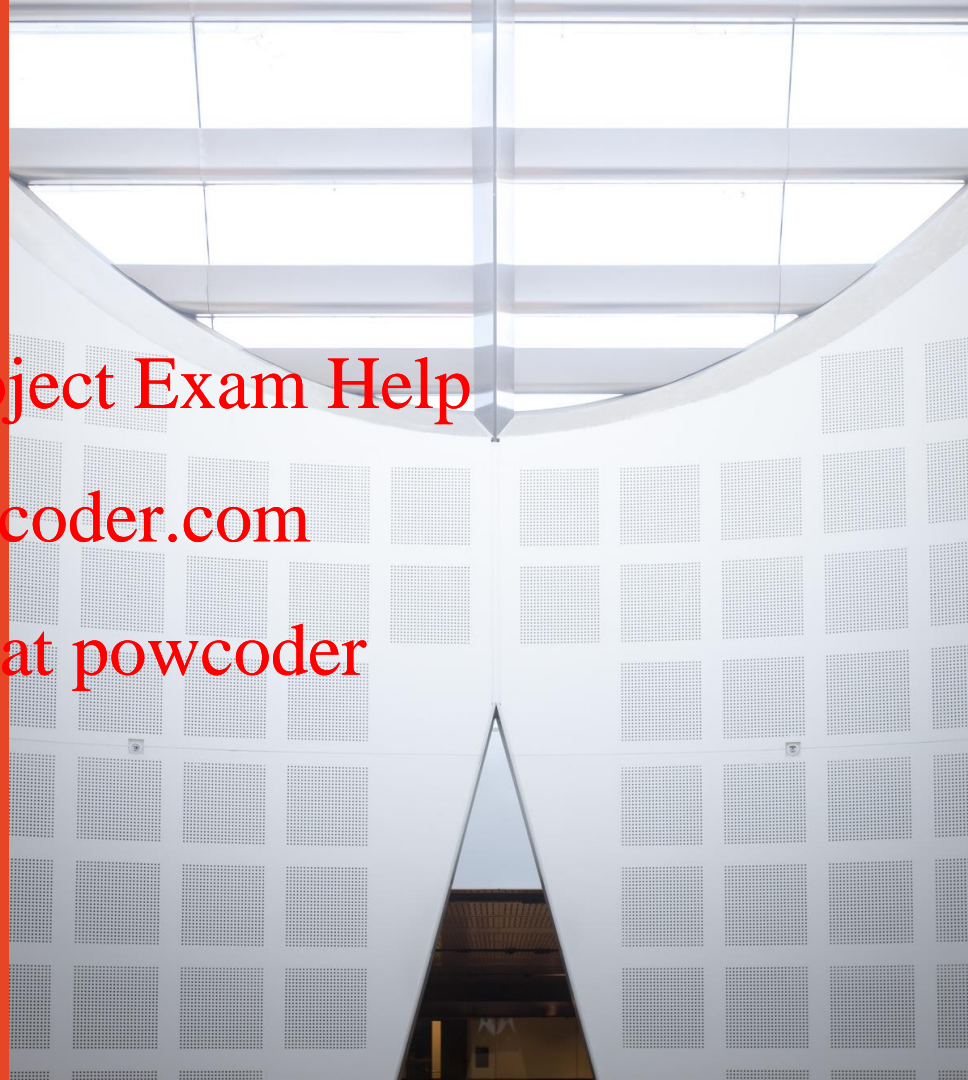
School of Information Technologies

THE UNIVERSITY OF
SYDNEY

# Copyright Warning

# Agenda

- Software Engineering

- Software Testing

- Unit Testing

# Testing in Software Engineering

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Software is Everywhere!

- Societies, businesses and governments dependent on SW systems

  - Power, Telecommunication, Education, Government, Transport, Finance, Health
  - Work automation, communication, control of complex systems

- Large software economies in developed countries

  - IT application development expenditure in the US more than $250bn/year[1]
  - Total value added GDP in the US[2]: $1.07 trillion

- Emerging challenges

  - Security, robustness, human user-interface, and new computational platforms

[1] Chaos Report, Standish group Report, 2014
[2] softwareimpact.bsa.org

# Software Engineering Body of Knowledge

- Software Requirements

- **Software Design / Modelling**

- **Software Construction**

- **Software Testing**

- Software Maintenance

- Software Configuration Management

- Software Engineering Process

- Software Engineering Tools and Methods

- Software Quality

IEEE computer society

SWEBOK®

Software Engineering Body of Knowledge (SWEBOK) https://www.computer.org/web/swebok/

# Why Software Engineering?

**Need to build high quality software systems under resource constraints**

- Social
  - Satisfy user needs (e.g., functional, reliable, trustworthy)
  - Impact on people's lives (e.g., software failure, data protection)
- Economical
  - Reduce cost; open up new opportunities
  - Average cost of IT development ~$12.3m, ~$1.3m and ~$434k for large, medium and small companies respectively[3]
- Time to market
  - Deliver software on-time

[3] Chaos Report, Standish group Report, 2014

# Software Quality Assurance

- Quality (products)
  - "Fitness for use"
- Software quality
  - Satisfying end use's needs: correct behaviour, easy to use, does not crash, etc
  - Easy to the developers to debug and enhance
- Quality Assurance (QA)
  - Processes and standards that lead to manufacturing high quality products
- Software Quality Assurance
  - Ensuring software under development have high quality and creating processes and standards in organization that lead to high quality software
  - Software quality is often determined through Testing

Juran and Gryna 1998

# Software Testing

**Why software testing?**

THE UNIVERSITY OF
SYDNEY

# Therac-25 Overdose*



- **What happened?**
  - Therac-25 radiation therapy machine
  - Patients exposed to overdose of radiation (100 times more than intended) - 3 lives!!

- **Why Did happen?**
  - Particular nonstandard sequence of keystrokes was entered within 8 seconds
  - Operator override a warning message with error code ("MALFUNCTION" followed by a number from 1 to 64), that explained in the user manual
  - Software checks safety replacing hardware interlocks in previous Therac versions
  - Absence of independent software code review
  - software and hardware integration has never been tested until assembled in the hospital (*Big Bang Testing*')

*https://en.wikipedia.org/wiki/Therac-25#Problem_description

# Software Failure – Ariane 5 Disaster[5]

## What happened?

- European large rocket - 10 years development, ~$7 billion
- Unmanaged software exception resulted from a data conversion from 64-bit floating point to a 16-bit signed integer
- Backup processor failed straight after using the same software
- Exploded 37 seconds after lift-off

## Why did it happen?

- *Inadequate validation and verification, testing and reviews, design error, incorrect analysis of changing requirements, ….*

[5] http://iansommerville.com/software-engineering-book/files/2014/07/Bashar-Ariane5.pdf

# Nissan Recall – Airbag Defect*



- **What happened?**
  - ~ 3.53 million vehicles recall of various models 2013-2017
  - Front passenger airbag may not deploy in an accident
- **Why Did happen?**
  - Software that activates airbags deployment improperly classify occupied passenger seat as empty in case of accident
  - Software defect that could lead to improper airbag function (failure)
  - No warning that the airbag may not function properly
  - Software sensitivity calibration due to combination of factors (high engine vibration and changing seat status)

http://www.reuters.com/article/us-autos-nissan-recall/nissan-to-recall-3-53-million-vehicles-air-bags-may-not-deploy-idUSKCN0XQ2A8
https://www.nytimes.com/2014/03/27/automobiles/nissan-recalls-990000-cars-and-trucks-for-air-bag-malfunction.html

# Software Project Failures

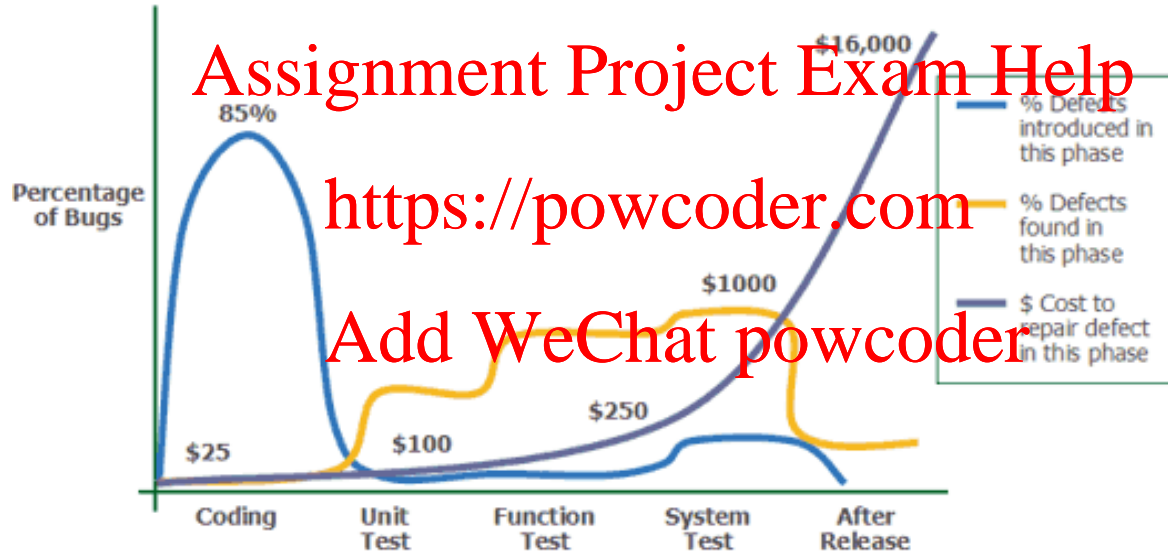| Project | Duration | Cost | Failure/Status |
|---|---|---|---|
| Pust Siebel - Swedish Police case management (Swedish Police) | 2011 – 2014 | $55m (actual) | Permanent failure – scraped due to poor functioning, inefficient in work environments |
| US Federal Government Health Care Exchange Web application | 2013 – ongoing | $93.7m (expected), $1.5bn (actual) | Ongoing problems - too slow, poor performance, people get stuck in the application process (frustrated users) |

# Why Software Testing?

- – Software development and maintenance costs
  - – Big financial burden

- – Total costs of inadequate software testing on the US economy is $59.5bn
  - – NIST study 2002*
  - – One-third of the cost could be eliminated by *improved software testing*

- – Need to develop functional, robust and reliable software systems
  - – Human/social factor - society dependency on software in every aspect of their lives
    - • Critical software systems - medical devices, flight control, traffic control
  - – Meet user needs and solve their problems
  - – Small software errors could lead to disasters

* https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf

# Software Testing – Costs



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Legend:
- % Defects introduced in this phase
- % Defects found in this phase
- $ Cost to repair defect in this phase

Data points: $25, $100, $250, $1000, $16,000

Phases: Coding, Unit Test, Function Test, System Test, After Release

85%

Percentage of Bugs

Capers Jones, Applied software measurement (2nd ed.): assuring productivity and quality, (1997), McGraw-Hill

# Software Testing

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

**What is software testing?**

# Software Testing

- Software process to
  - Demonstrate that software meets its requirements (*validation testing*)

  - Find incorrect or undesired behavior caused by defects/bugs (defect testing)
    - E.g., System crashes, incorrect computations, unnecessary interactions and data corruptions

- Part of software verification and validation (V&V) process

# Testing (Levels)

| Testing level | Description |
|---|---|
| Unit / Functional Testing | The process of verifying functionality of software components (functional units, subprograms) independently from the whole system |
| Integration Testing | The process of verifying interactions/communications among software components. Incremental integration testing vs. "Big Bang" testing |
| System Testing | The process of verifying the functionality and behaviour of the entire software system including security, performance, reliability, and external interfaces to other applications |
| Acceptance Testing | The process of verifying desired acceptance criteria are met in the system (functional and non-functional) from the user point of view |

# Software Verification and Validation

- Software testing is part of software Verification and Validation (V&V)

- The goal of V&V is to establish confidence that the software is "*fit for purpose*"

- Software Validation
  - Are we building the right product?
  - Ensures that the software meets customer expectations

- Software Verification
  - Are we building the product right?
  - Ensures that the software meets its stated functional and non-functional requirements

# Unit Testing

THE UNIVERSITY OF
SYDNEY

# Unit Testing

– The process of verifying functionality of software components independently

- Unit → methods, functions or object classes
- Verify that each functional unit behaves as expected (defect testing)
- Carried out by developers / SW testers
- First level of testing

# Why Unit Testing?

- Change and maintain code at smaller scale

- Discover defects early and fix it at cheaper costs

- Ease debugging

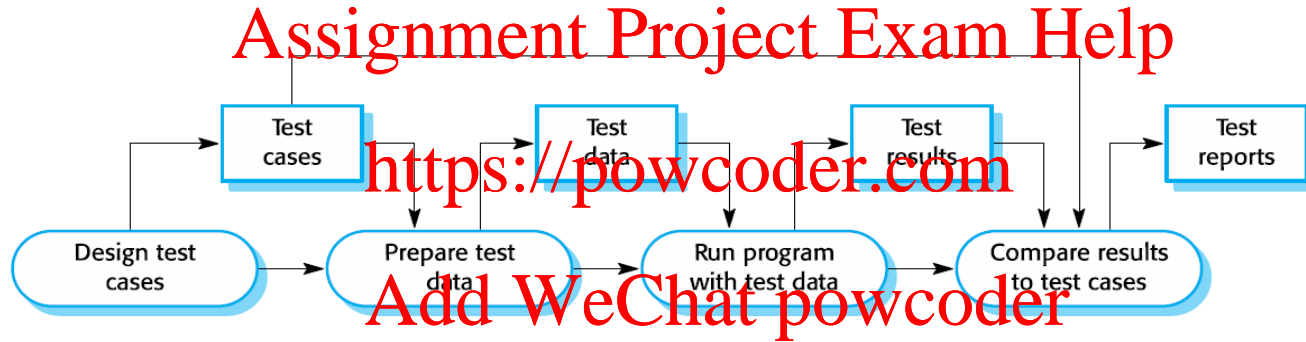- Code reusability

- Reduce integration testing

# Unit Testing – How?

1. Design test cases

2. Prepare test data

3. Run test cases using test data

4. Compare results to test cases

5. Prepare test reports

# Software Testing Process



Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Ian Sommerville. 2016. *Software Engineering* (10th ed.). Addison-Wesley, USA.

# Designing Test Cases

- Effective test cases show:
  - The unit under test does what it spoused to do
  - Reveal defects in the unit, if there is any

- Design two types of test cases
  - Test normal operation of the unit
  - Test abnormality (common problems/defects)

# Choosing Test Cases – Techniques

- Partition testing (equivalence partitioning)
  - Identify groups of inputs that have common characteristics
  - From within each of these groups, choose tests
  - Use program specifications, documentation and/or experience

- Guideline-based testing
  - Use testing guidelines based on previous experience of the kinds of errors often made

# Equivalence Partitioning

— Different groups with common characteristics
  – E.g., positive numbers, negative numbers

— Program behave in a comparable way for all members of a group

— Choose test cases from each of the partitions

# Test Case Selection

- Understanding developers thinking
  - Developers tend to think of typical values of input
  - Developers sometimes overlook atypical values of input

- Choose test cases that are:
  - On the boundaries of the partitions
  - Close to the midpoint of the partition

# Test Cases – Identifying Partitions

– Consider the following brief program specification:

   – A program accepts 4 to 8 input which are five-digits integers greater than 10,000

– Exercise:

   – Identify the input partitions and possible test input values

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Choose Test Cases – Testing Guidelines

– Knowledge of type of test cases effective for finding errors

– Example of test cases for testing sequences, arrays or lists:

    – Test single value

    – Test different sequences of different sizes in different tests

    – Test partition boundaries (first, middle and last elements)

# General Testing Guidelines – Examples

- Choose inputs that force the system to generate all error messages;

- Design inputs that cause input buffers to overflow

- Repeat the same input or series of inputs numerous times

- Force invalid outputs to be generated

- Force computation results to be too large or too small

J. A. Wittaker (2009) Exploratory Software Testing, Adison-Wesley

# Unit Testing

THE UNIVERSITY OF
SYDNEY

# Unit Testing – Terminology

- Code under test

- Unit test
  - a piece of code written by a developer that executes a specific functionality in the code under test and asserts a certain behavior or state
  - Small unit of code e.g., a method or class
  - External dependencies are removed (mocks can be used)
  - Not suitable for complex user interface or component interaction

- Test Fixture
  - The context for testing
    - Usually shared set of testing data
    - Methods for setup those data
    - E.g., a fixed string (test fixture), which is used as input for a method

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Unit Testing Frameworks for Java

- Junit
- TestNG
- Jtest (commercial)
- Many others …

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#Java

# Unit Testing Frameworks – Junit

- An open source framework for writing and running tests for Java
  - Test cases and fixtures, test suites, test runner

- One of the unit testing frameworks collectively known as xUnit

- Uses annotations to identify methods that specify a test

- Can be integrated with Eclipse and build automation tools (e.g., Ant, Maven, Gradle)

https://github.com/junit-team/junit4

# Junit – Constructs

- **Junit test (Test class)**
  - A method contained in a class which is only used for testing (called *Test class*)

- **Test suite**
  - Contains several test classes which will be executed all in the specified order

- **Test Annotations**
  - To define/denote test methods (e.g., @Test, @Before)
  - Such methods execute the code under test

- **Assertion methods (assert)**
  - To check an expected result versus actual results
  - Variety of methods
  - Provide meaningful messages in assert statements

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Junit – Annotations

| JUnit 4* | Description |
|---|---|
| import org.junit.* | Import statement for using the following annotations |
| @Test | Identifies a method as a test method |
| @Before | Executed before each test. To prepare the test environment (e.g., read input data, initialize the class) |
| @After | Executed after each test. To cleanup the test environment (e.g., delete temporary data, restore defaults) and save memory |
| @BeforeClass | Executed once, before the start of all tests. To perform time intensive activities, e.g., to connect to a database. Need to be defined as static to work with Junit |
| @AfterClass | Executed once, after all tests have been finished. To perform clean-up activities, e.g., to disconnect from a database. Need to be defined as static to work with Junit |

*See Junit 5 annotations and compare them https://junit.org/junit5/docs/current/user-guide/#writing-tests-annotations

# Junit – Assert Class

- Assert class provides *static* methods to test for certain conditions

- Assertion method compares the actual value returned by a test to the expected value
  - Specify the expected and actual results and the error message
  - Throw an *AssertionException* if the comparison fails

# Junit – Methods to Assert Test Results

| Method | Description |
|--------|-------------|
| fail([message]) | Let the method fail. E.g., to check that a certain part of the code not reached or to have a failing test before the test code implemented |
| assertTrue([message,] boolean condition)<br>assertFalse([message,] boolean condition) | Checks that the Boolean condition is true<br>Checks that the Boolean condition is false |
| assertEquals([message,] expected, actual) | Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays. |
| assertEquals([message,] expected, actual, tolerance) | Test that float or double values match. The tolerance is the number of decimals which must be the same |
| assertNull([message,] object)<br>assertNotNull([message,] object) | Checks that the object is null<br>Checks that the object is not null |

*Also check assertions in Junit 5 – https://junit.org/junit5/docs/current/user-guide/#writing-tests-assertions

# Junit Test – Example

```
1
2  //Class to be tested
3  public class MyClass{
4      public int multiply (int m, int n){
5          return m * n;
6      }
7
8  }
9
```

*MyClass' multiply*(int, int) method

```
1  import static org.junit.Assert.assertEquals;
2
3  import org.junit.Test;
4
5  public class MyClassTests {
6
7      @Test
8      public void multiplicationOfTwoIntegersShouldReturnZero() {
9          MyClass tester = new MyClass(); // MyClass is tested
10
11         // assert statements
12         assertEquals(0, tester.multiply(10, 0), "10 x 0 must be 0");
13         assertEquals(0, tester.multiply(0, 10), "0 x 10 must be 0");
14         assertEquals(0, tester.multiply(0, 0), "0 x 0 must be 0");
15     }
16 }
```

MyClassTests class for testing the method multiply(int, int)

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Junit – Executing Tests

- Tests can be executed from the command line
  - *runClass()* of the *org.junit.runner.JUnitCore* class allows developers to run one or several test classes
  - Information about tests can be retrieved using the *org.junit.runner.**Result*** object

- Test automation
  - Build tools such as Maven or Gradle along with a Continuous Integration Server (e.g., Jekins) can be configured to run tests automatically on a regular basis
  - Essential for regular daily tests (agile development)

# Junit – Executing Tests

– To run tests from the command line

```
 1  import org.junit.runner.JUnitCore;
 2  import org.junit.runner.Result;
 3  import org.junit.runner.notification.Failure;
 4
 5  public class MyTestRunner {
 6    public static void main(String[] args) {
 7      Result result = JUnitCore.runClasses(MyTestSuite.class);
 8      for (Failure failure : result.getFailures()) {
 9        System.out.println(failure.toString());
10      }
11    }
12  }
```

# Junit – Test Suites

– **Test suite** contains several test classes which will be executed all in the specified order

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
 1  import org.junit.runner.RunWith;
 2  import org.junit.runners.Suite;
 3  import org.junit.runners.Suite.SuiteClasses;
 4
 5  @RunWith(Suite.class)
 6  @SuiteClasses({
 7          MyClassTest.class;
 8          MySecondClassTest.class })
 9
10  public class AllTests {
11
12  }
```

# Junit – Static Import

– Static import is a feature that allows fields and methods in a class as public static to be used without specifying the class in which the field s defined

```
// without static imports you have to write the following statement
Assert.assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));

// alternatively define assertEquals as static import
import static org.junit.Assert.assertEquals;

// more code

// use assertEquals directly because of the static import
assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
```

# Junit – Eclipse Support

– Create Junit tests via wizards or write them manually

Assignment Project Exam Help

– Eclipse IDE also supports executing tests interactively

   – Run-as Junit Test will starts JUnit and execute all test methods in the selected
     https://powcoder.com
   class

Add WeChat powcoder

– Extracting the failed tests and stack traces


– Create test suites

# Automated Test Frameworks

– Key sign of good practice is that tests are kept as an asset of the process
  – and can be run automatically, and frequently

– Also, reports should be easy to understand
  – Big Green or Red Signal

– A framework is some software that allows test cases to be described in standard form and run automatically

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Tests Automation – Junit with Gradle

– To use Junit in your Gradle build, add a testCompile dependency to your build file

– Gradle adds the test task to the build graph and needs only appropriate Junit JAR to be added to the classpath to fully activate the test execution

```
1  ...
2  apply plugin: 'java'
3
4  repositories {
5      mavenCentral()
6  }
7  dependencies {
8      testCompile 'junit:junit:4.8.2'
9  }
```

**Test Summary**

2 | 0 | 0.002s
tests | failures | duration

**100%**
successful

| Packages | Classes |
| --- | --- |

| Package | Tests | Failures | Duration | Success rate |
| --- | --- | --- | --- | --- |
| default-package | 2 | 0 | 0.002s | 100% |

# Junit with Gradle – Parallel Tests

```
 1  ...
 2  apply plugin: 'java'
 3
 4  repositories {
 5      mavenCentral()
 6  }
 7  dependencies {
 8      testCompile 'junit:junit:4.8.2'
 9  }
10  test {
11      maxParallelForks = 5
12      forkEvery = 50
13  }
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

maximum simultaneous JVMs spawned

causes a test-running JVM to close and be replaced by a brand new one after the specified number of tests have run under an instance

# Next week ....

**Theory of Testing, Design of Test Cases and more unit testing**

Tutorial - Unit Testing

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

THE UNIVERSITY OF
SYDNEY

# References

- Ian Sommerville. 2016. Software Engineering (10th ed.) Global Edition. Pearson, Essex England

- Junit 4, Project Documentation, https://junit.org/junit4/

- Vogella GmbH, JUnt Testing with Junit – Tutorial (Version 4.3,21.06.2016 ) http://www.vogella.com/tutorials/JUnit/article.html