
DevOps

自動化持續整合測試與建置

郭忠義

jykuo@ntut.edu.tw

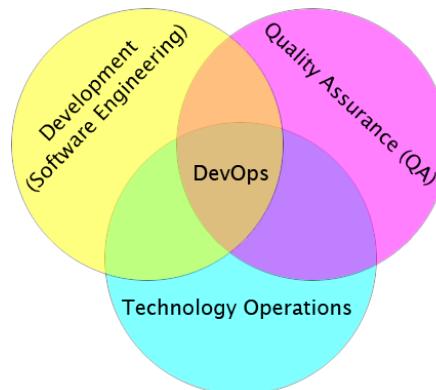
臺北科技大學資訊工程系

DevOps 目標

- 企業組織開發產品目標
 - 面對多種使用者、具備多樣功能。
 - 頻繁交付產品、部署週期短。
 - 虛擬化和雲端運算基礎設施。
- 開發和業務紛爭
 - 業務為滿足客戶，要求開發人員做「Change」。
- 開發和維運紛爭
 - 開發人員為滿足使用者需求，不斷更新版本，不管**硬體網路**部署問題，但**新版本**會造成新問題，增加維運人員工作量。
 - 維運人員為成功部署，可能變更設定，造成上線責任歸屬紛爭。
 - 維運遇到問題，軟體開發人員不一定協助處理，兩者間產生「困惑牆Wall of Confusion」。

DevOps 目標

- “開發Development”與“維運Operations”整合。
 - 強調溝通(Communication)、合作(Collaboration)、整合(Integration)及自動化(Automation)加強開發人員與維運和品質控制人員合作。
 - DevOps指導原則幫助解決Wall of Confusion問題，提高軟體品質、加快開發速度和避免重工。
- 敏捷軟體開發(Agile Software Development)方法。
 - 透過跨部門合作及自動化實現快速、頻繁發佈軟體。



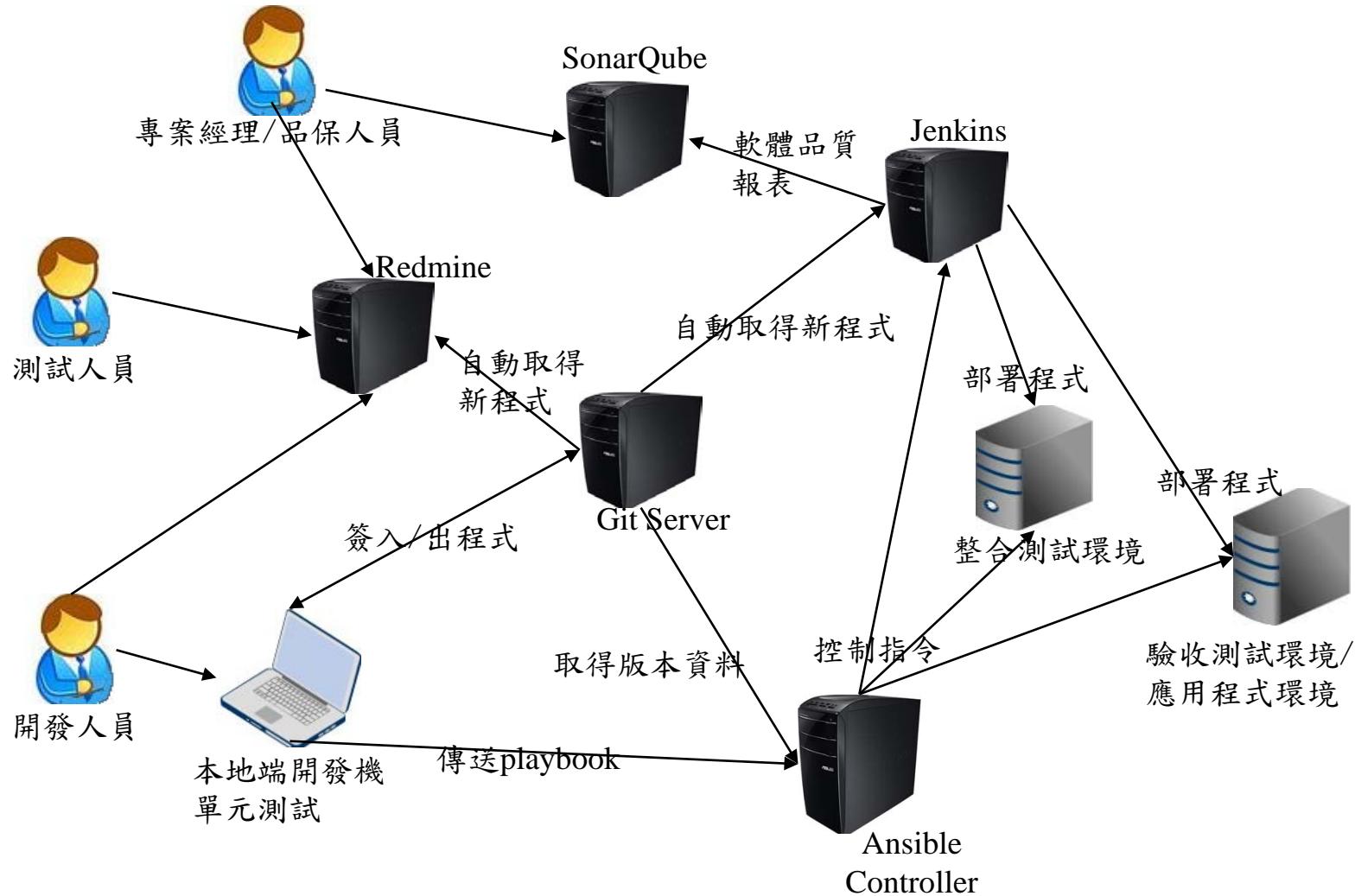
DevOps 目標

- CI/CD持續整合與持續發展
 - 扮演角色
 - 與傳統開發流程的不同
- 系統上線後如何持續監控狀態以修復程式
- 建立Git版本控管帳號
- 在Jenkins建立CI/CD的工作任務
 - 透過Jenkins自動將新版本部署至環境上

自動化建置與工具整合



DevOps 開發流程



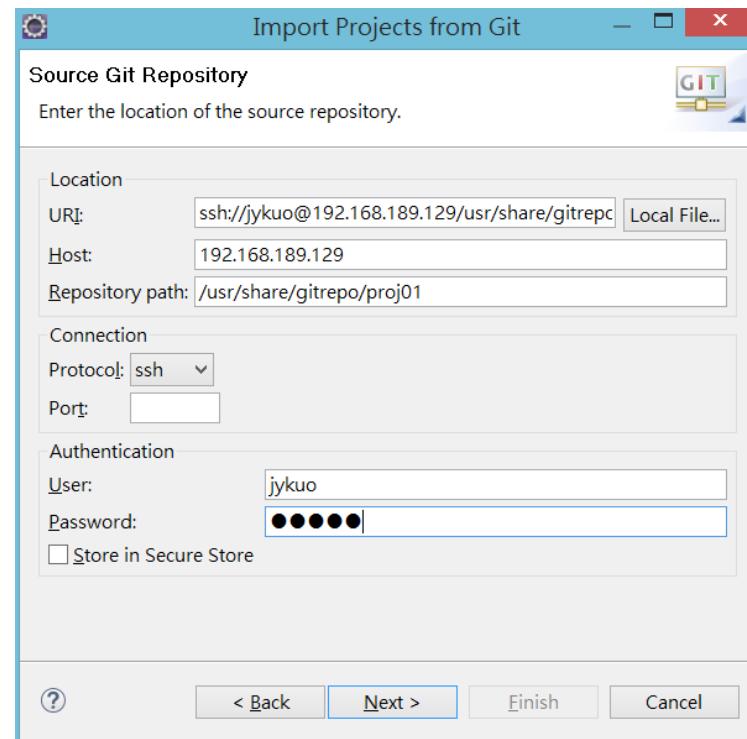
DevOps開發流程

□ DevOps開發流程

- 簽入(Commit)：程式在本地端開發機測試後，簽入版本控制系統 Git。
- 建置 (Build)：持續整合系統Jenkins CI，偵測到版本更新，自動從 Git 取得最新程式進行建置與測試。。
- 單元測試：Jenkins 完成建置後，自動執行指定的單元測試。
- 部署到測試環境 (Deploy to Test Environment)：完成單元測試後，Jenkins 將程式部署到跟應用環境 (Production Environment) 相近的驗收測試環境進行測試。
- 驗收測試：在驗收測試環境進行自動化功能測試，例如 Selenium 測試。開發人員或客戶手動進行跟實際情況類似的驗收測試。
- 部署到應用環境 (Deploy to Production Environment)：通過所有測試後，將最新版本部署到實際應用環境。

Eclipse簽出程式

- 啟動虛擬機器
- 使用Eclipse，以新的workspace，從Git簽出程式
 - Host: 在這裡鍵入方程式。192.168.xx.xx
 - Repository path: /usr/share/gitrepo/proj02



Eclipse改修程式

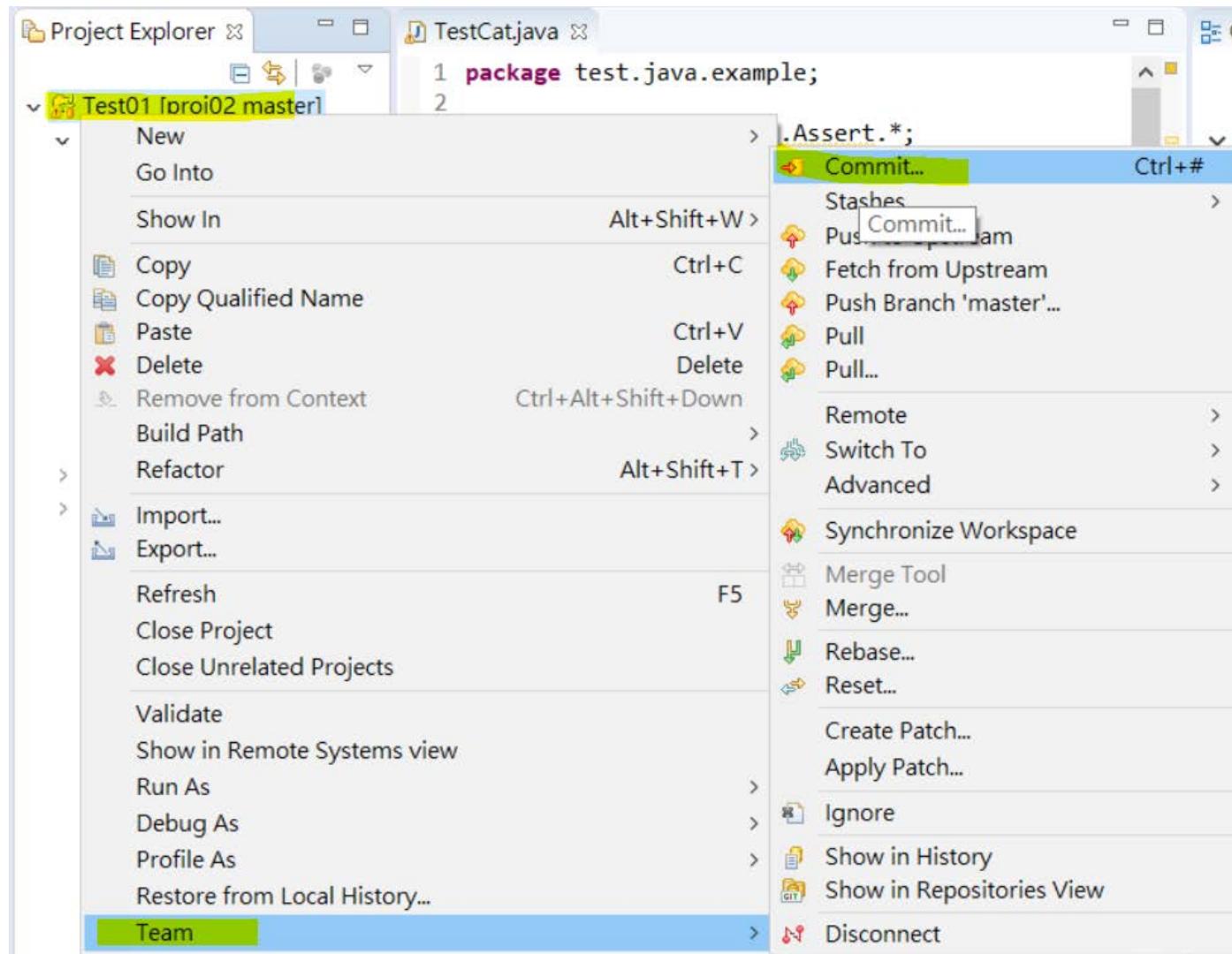
- TestCat
- 測試程式
 - 三個Cat
 - 改成Tiger

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer view, displaying a Java project named 'Test01'. The 'src' folder contains two packages: 'main.java.example' and 'test.java.example'. Under 'main.java.example', there are 'Cat.java' and 'Tiger.java'. Under 'test.java.example', there are 'CatTest.java' and 'TestCat.java'. The 'test.java.example' package is highlighted with a yellow selection bar. In the center-right is the code editor window for 'TestCat.java'. The code is as follows:

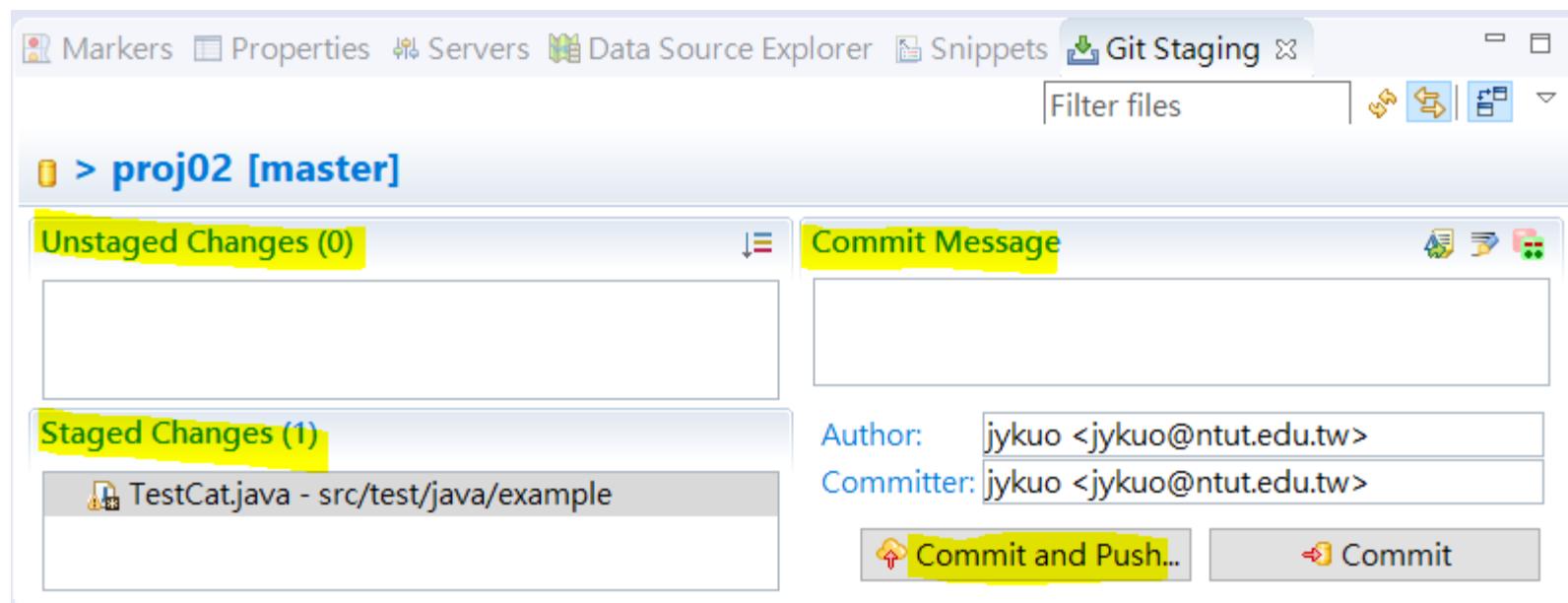
```
1 package test.java.example;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.After;
6 import org.junit.Before;
7 import org.junit.Test;
8
9 import main.java.example.Cat;
10
11 public class TestCat {
12
13     @Before
14     public void setUp() throws Exception {
15     }
16
17     @After
18     public void tearDown() throws Exception {
19     }
20
21     @Test
22     public void testEat() {
23         new Cat().eat();
24     }
25
26     @Test
27     public void testSound() {
28         new Cat().sound();
29     }
30 }
```

The code editor has several parts highlighted with yellow boxes: 'import main.java.example.Cat;', 'public class TestCat {', 'new Cat().eat();', and 'new Cat().sound();'.

簽入 Git



簽入 Git



持續整合測試查看Coverage

Jenkins

- 新增作業
- 使用者
- 建置歷程
- 管理 Jenkins
- 我的視景
- Credentials

管理 Jenkins

您的 Reverse Proxy 設定
新版的 Jenkins (2.73) 已可用

- 設定系統
- 設定全域設定
- 設定全域安全
- 保護 Jenkins

SonarQube servers

Environment variables

Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	Server URL
MySonar	http://192.168.78.128:9000/

儲存 Apply

持續整合測試查看Coverage



持續整合測試查看 Coverage

Jenkins > Test01 >

回到儀表板 狀態 變更 工作目錄 馬上建置 刪除專案 組態 Coverage Trend Move SonarQube

專案 Test01

My Test

SonarQube 工作區 最近變更

Code Coverage Trend

Build	Green Coverage (%)	Red Coverage (%)
#9	20	2
#10	20	7
#11	11	6
#12	15	7
#13	18	5
#14	19	4
#15	22	1

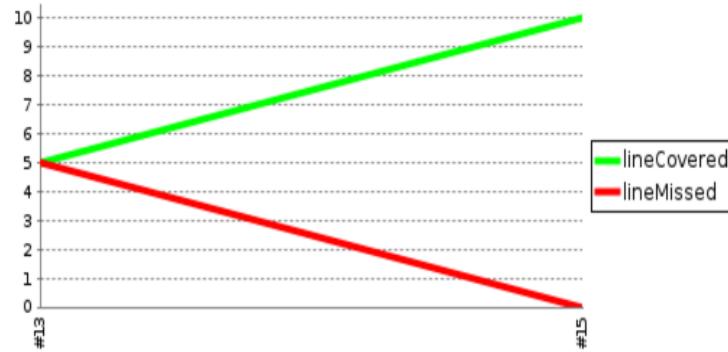
永久連結

- 最新建置 (#15), 49 秒 以前
- 最新穩定建置 (#15), 49 秒 以前
- 最新成功建置 (#15), 49 秒 以前
- 最新失敗建置 (#14), 9 分 51 秒 以前
- 最新不成功建置 (#14), 9 分 51 秒 以前

建置歷程 趨勢 find #15 2017/8/11 上午 6:51

持續整合測試查看Coverage

Package: main.java.example



Coverage Summary

name	instruction	branch	complexity	line	method	class
main.java.example	M: 0 C: 22 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 0 0%	M: 0 C: 6 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 10 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 6 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 2 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>

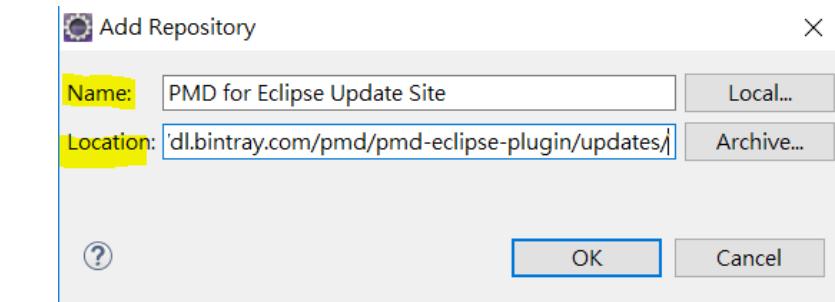
Coverage Breakdown by Source File

name	instruction	branch	complexity	line	method	class
<u>Cat</u>	M: 0 C: 11 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 0 0%	M: 0 C: 3 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 5 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 3 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 1 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>
<u>Tiger</u>	M: 0 C: 11 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 0 0%	M: 0 C: 3 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 5 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 3 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>	M: 0 C: 1 100% <div style="width: 100%; background-color: #2e7131; height: 10px;"></div>

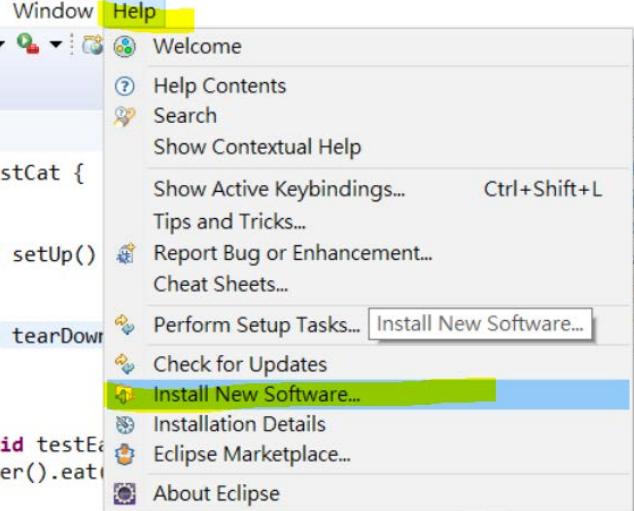
靜態程式碼分析 – PMD

□ Help –> Install New Software...

- Name: PMD for Eclipse Update Site
- <https://dl.bintray.com/pmd/pmd-eclipse-plugin/updates/>



The screenshot shows the 'Add Repository' dialog in Eclipse. The 'Name:' field contains 'PMD for Eclipse Update Site' and the 'Location:' field contains 'https://dl.bintray.com/pmd/pmd-eclipse-plugin/updates/'. The 'OK' button is highlighted with a blue border.



The screenshot shows the Eclipse 'Help' menu. The 'Install New Software...' option is highlighted with a green background and a blue border.

Work with: <https://dl.bintray.com/pmd/pmd-eclipse-plugin/updates/> Add...
Find more software by working with the "Available Software Sites" preferences
type filter text

Name	Version
<input checked="" type="checkbox"/> PMD for Eclipse 4	
<input checked="" type="checkbox"/> All items are installed	

Select All Deselect All

靜態程式碼分析 – PMD

The screenshot shows the Eclipse IDE interface with the PMD analysis results overlaid.

Top Bar: Shows the Eclipse menu bar with "File", "Edit", "Source", and "PMD". The "PMD" option is highlighted, and its submenu is open, showing "Clear Violation Reviews", "Clear Violations", and "Check Code".

Left Side: The "Package Explorer" view shows a project named "Test01" with a "src" folder containing "main.java.example", "Cat.java", "Tiger.java", and "test.java.example". Other entries like "JRE System Library [JavaSE-1.8]", "JUnit 4", and "build.gradle" are also visible.

Code Editor: Displays Java code with annotations for setup and teardown methods.

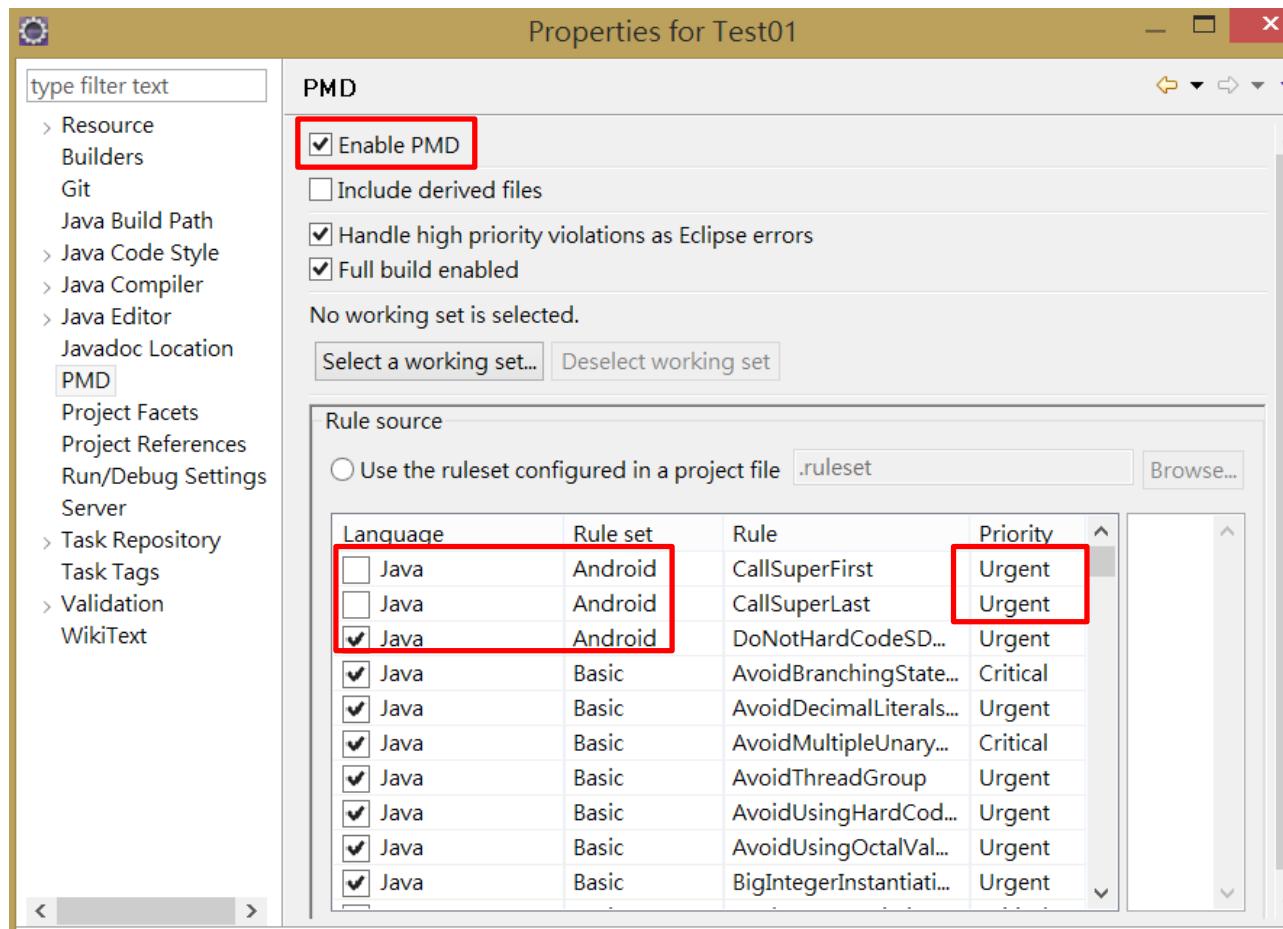
```
public void setUp() throws Exception {  
    //  
}  
@After  
public void tearDown() throws Exception {  
    //  
}
```

Violations Overview View: A table showing the violations found in the code.

Element	# Violations	# Violation...	# Violation...	Project
main.java.example	13	1625.0	3.25	Test01
Tiger.java	6	1500.0	3.00	Test01
▶ CommentRequired	3	750.0	1.50	Test01
▶ AtLeastOneConstructor	1	250.0	0.50	Test01
▶ SystemPrintln	2	500.0	1.00	Test01
Cat.java	7	1750.0	3.50	Test01
▶ ShortClassName	1	250.0	0.50	Test01
▶ CommentRequired	3	750.0	1.50	Test01
▶ AtLeastOneConstructor	1	250.0	0.50	Test01
▶ SystemPrintln	2	500.0	1.00	Test01

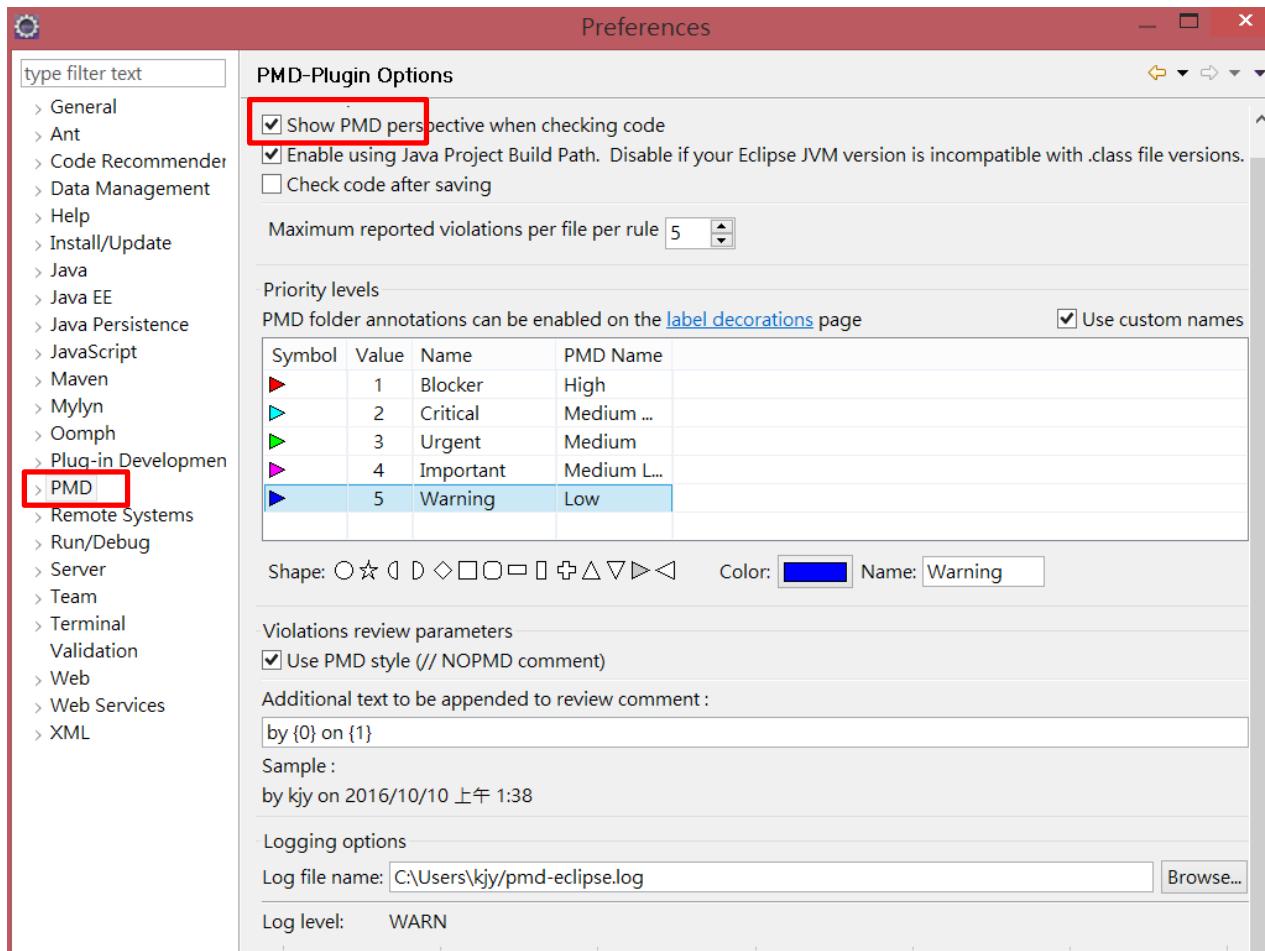
靜態程式碼分析 – PMD

- 點選Project名稱，按右鍵，選擇 Properties，設定Rules



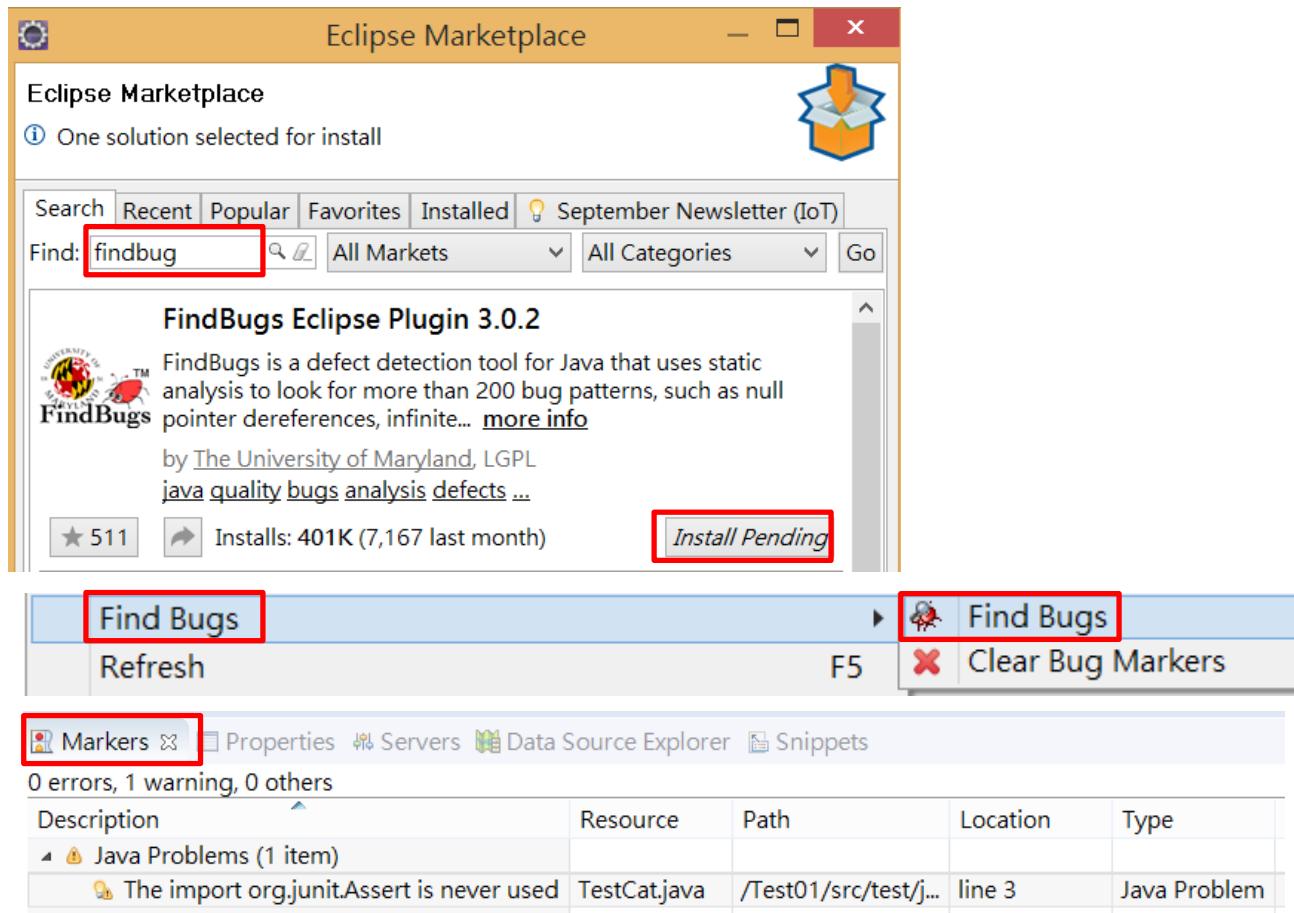
靜態程式碼分析 – PMD

□ Window – Preferences



靜態程式碼分析 – Find Bugs

- Help - Eclipse Marketplace... - Findbug
 - 右鍵 – Find Bugs



Sonarqube品質管理

- 啟動sonarqube，sudo docker start sonarqube
 - 查看sonarqube docker是否啟動，sudo docker ps
 - 192.168.x.x:9000，登入admin/admin，Software Testing

The screenshot shows the SonarQube dashboard at 192.168.189.129:9000. The top navigation bar includes links for Dashboards, Issues, Measures, Rules, Quality Profiles, Quality Gates, and More. The main content area displays the 'Home' page with a 'Welcome to SonarQube Dashboard' message and a 'PROJECTS' section listing the 'Software Testing' project.

QG	Name	Version	LOC	Bugs
✗	Software Testing	1.0	201	0

1 results

The screenshot shows the 'Software Testing' project details page. The top navigation bar includes links for Issues, Measures, Code, Dashboards, and Administration. The main content area displays the 'Quality Gate' status as 'Failed' (indicated by a red box). It also shows 'Bugs & Vulnerabilities' and 'Leak Period' information.

since previous version	since previous version
2 New Vulnerabilities > 0	6.9% Technical Debt Ratio on New Code > 5.0%

Leak Period: since previous version started 22天前

Bugs	Vulnerabilities	New Bugs	New Vulnerabilities
0 A	2 D	0	2

Key: moe-st
Quality Gate (Default) Sc
Quality Profile (Java) Sona (JavaScript)
Events

Jenkins 查看 Git 工作區 Code

- 點選 Test01，或回到專案；點選工作目錄
- 點選src、example；有 ShoppingCart.java



- 點選ShoppingCart.java，可以看到Code

Eclipse新增程式

- 增加程式 production code，在main.java.example，右鍵new
 - 購物車 ShoppingCart 類別
 - 資料屬性
 - cost，原始價格。
 - vip=1，會員，vipDiscount 會員折扣；vip=0，非會員。
 - 滿額 mass，massDiscount 滿額折扣。
 - 方法
 - 設定資料屬性，折扣0~100為0~100折。
 - 計算總價
 - » 會員，總價，打 vipDiscount 折。非會員無折扣。
 - » 滿額 mass，總價，再打 massDiscount 折扣。

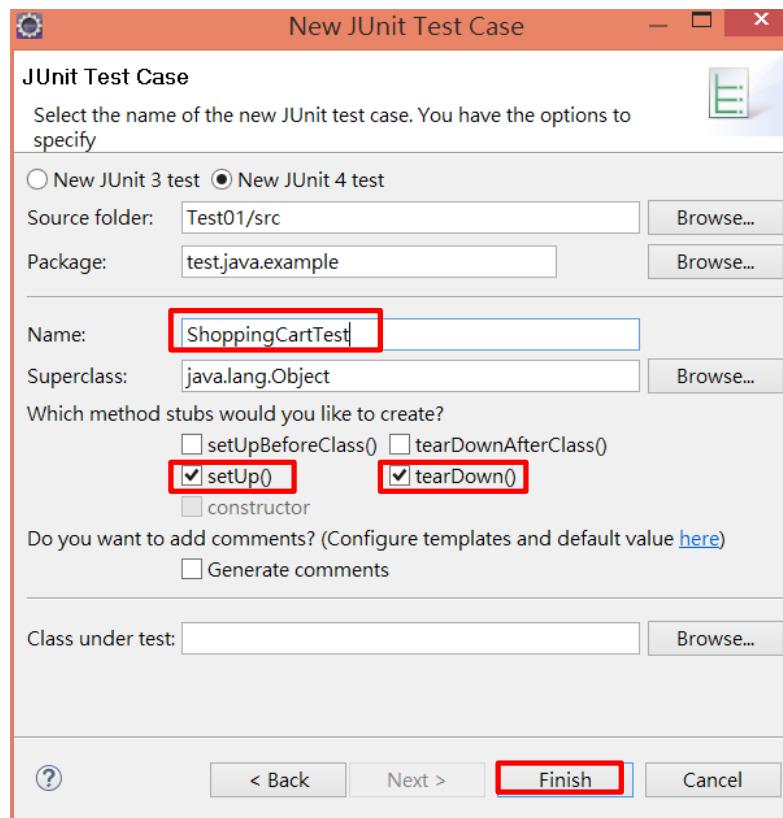
Eclipse新增程式

```
package main.java.example;
public class ShoppingCart {
    private int vip;
    private int vipDiscount;
    private int cost;
    private int mass;
    private int massDiscount;
    private int newCost;
    public ShoppingCart() {
        vip=0;    vipDiscount=0;
        cost = 0; newCost=0;
        mass = 0; massDiscount=100;
    }
    public void setVip(int vip, int vipDiscount) {
        this.vip = vip;
        this.vipDiscount = vipDiscount;
    }
}
```

```
public void setMass(int mass, int massDiscount) {
    this.mass = mass;
    this.massDiscount = massDiscount;
}
public void setCost(int cost) {
    this.cost = cost;
}
public void computeCost() {
    newCost = cost;
    if (vip==1) {
        newCost = newCost*vipDiscount;
    }
    if (cost>=mass) {
        newCost = newCost*massDiscount;
    }
}
public int getCost() {
    return newCost;
}
}
```

Eclipse新增程式

- 新增 test Driver，測試計算總價方法，test.java.example右鍵
 - New – Other... – Java – Junit – Junit Test Case [next]
 - Name: ShoppingCartTest [Finish]

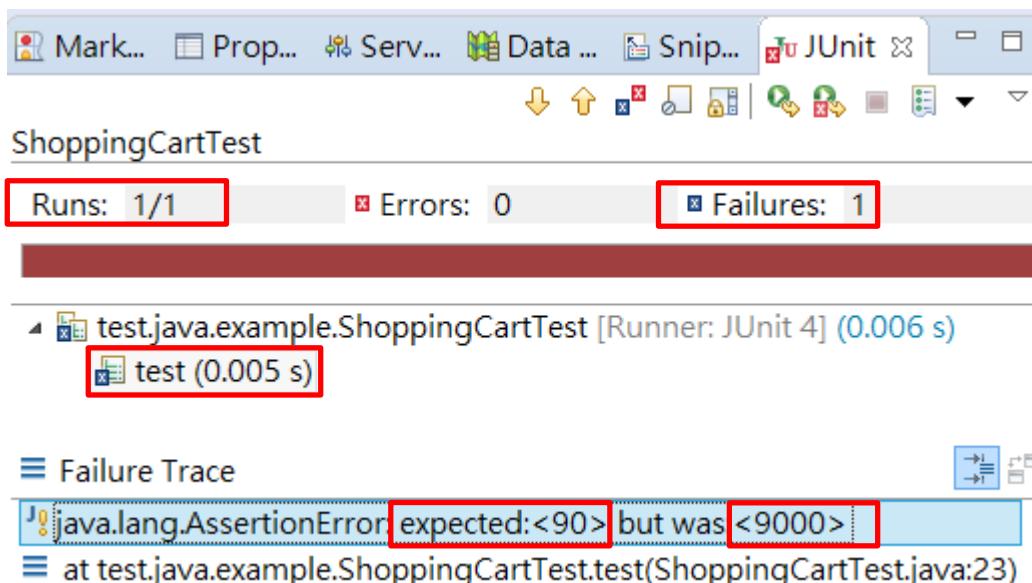


Eclipse新增程式

```
package test.java.example;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import main.java.example.*;
public class ShoppingCartTest {
    private ShoppingCart myCart;
    @Before
    public void setUp() throws Exception {
        myCart = new ShoppingCart();
    }
    @After
    public void tearDown() throws Exception {
        myCart = null;
    }
    @Test
    public void test() {
        myCart.setCost(100);
        myCart.setMass(100, 90);
        myCart.setVip(0, 100);
        myCart.computeCost();
        assertEquals(90,myCart.getCost());
    }
}
```

Eclipse新增程式

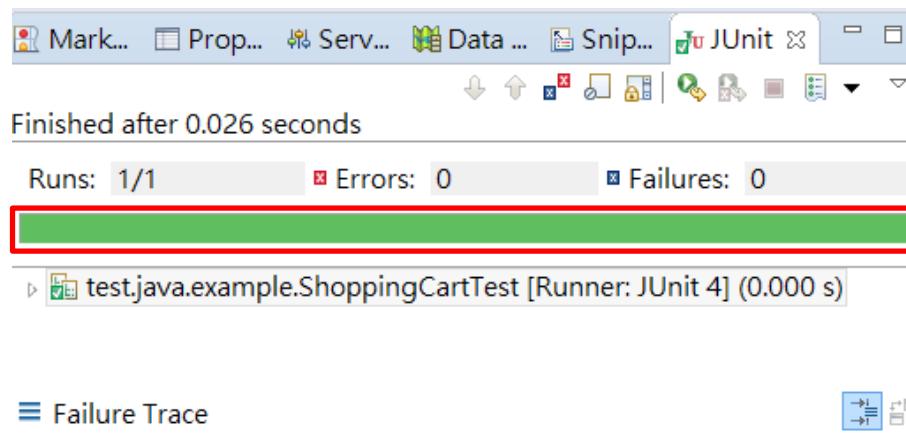
- 執行測試， ShoppingCartTest 右鍵



Eclipse改修程式

- 修改程式，重新測試，變成綠色

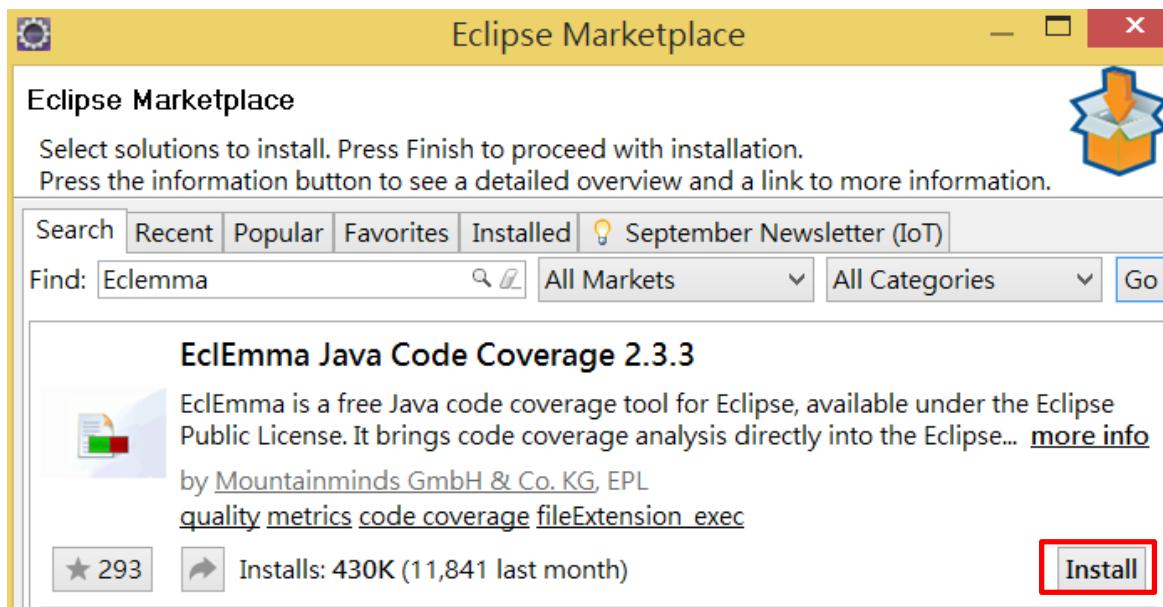
```
public void computeCost() {  
    newCost = cost;  
    if (vip==1) {  
        newCost = (newCost*vipDiscount)/100;  
    }  
    if (cost>=mass) {  
        newCost = (newCost*massDiscount)/100;  
    }  
}
```



Eclipse測試涵蓋度Coverage

□ 安裝Eclipse plugin，Eclemma

- Help – Eclipse Marketplace
- Find: Eclemma [Go][Install][Accept]
- 重新啟動Eclipse



Eclipse測試涵蓋度Coverage

- 重新測試，查看 Coverage；右鍵，點選 Coverage As

The screenshot shows the Eclipse IDE interface with the Coverage As tool open. At the top, there is a toolbar with a dropdown labeled "Coverage As" set to "JUnit" (highlighted with a red box), and a button "Coverage Configurations...". Below the toolbar, the code editor displays a Java snippet:

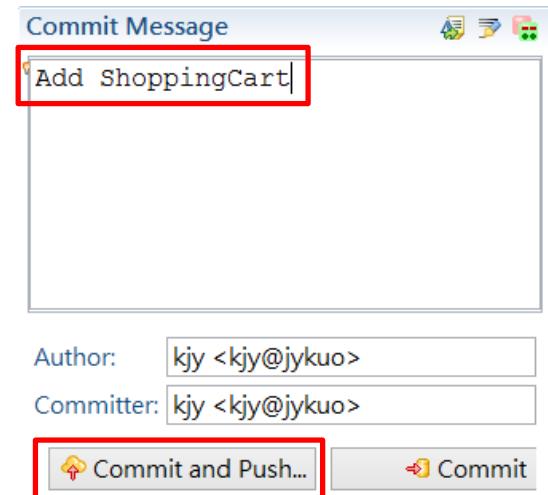
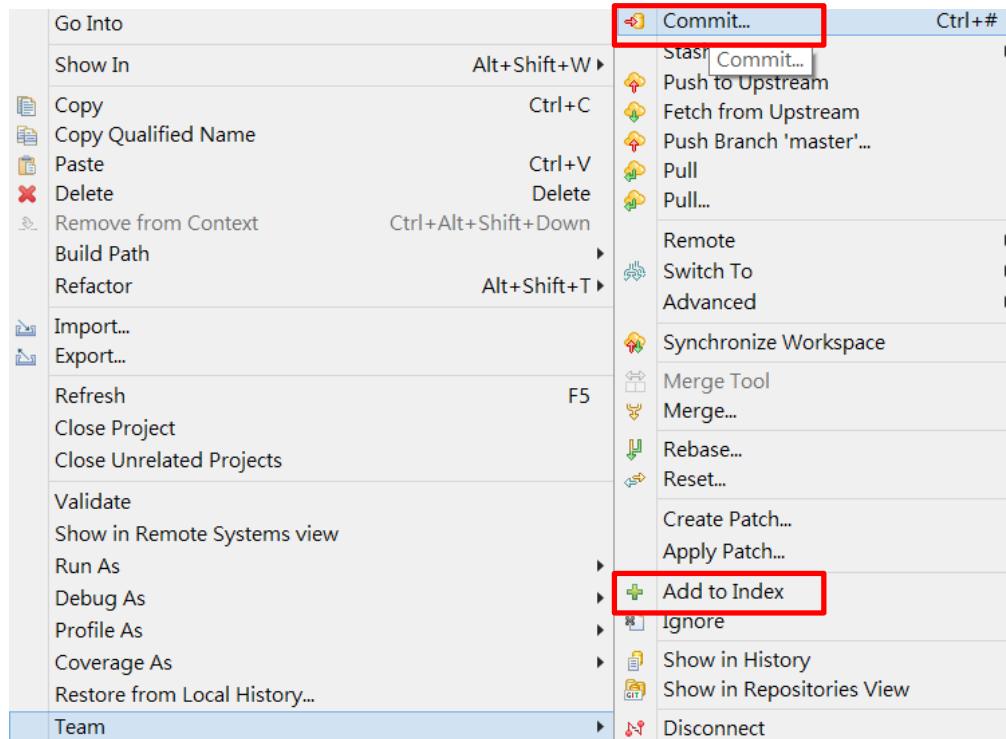
```
28     if (vip==1) {  
29         newCost = (newCost*vipDiscount)/100;  
30     }  
31     if (cost>=mass)  
32         newCost = (newCost*massDiscount)/100;  
33     }  
34 }  
35 public int getCost() {  
36     return newCost;  
37 }  
38 }
```

The code editor highlights certain lines in yellow, indicating they are covered by the current test. The bottom half of the screenshot shows the Coverage view in the Eclipse interface, which is a table of coverage statistics:

Element	Coverage	Covered Instr...	Missed Instru...	Total Instru...
Test01	63.4 %	102	59	161
src	63.4 %	102	59	161
main.java.example	67.7 %	65	31	96
Cat.java	0.0 %	0	11	11
Tiger.java	0.0 %	0	11	11
ShoppingCart.java	87.8 %	65	9	74
ShoppingCart	87.8 %	65	9	74
computeCost()	71.9 %	23	9	32
ShoppingCart()	100.0 %	21	0	21
getCost()	100.0 %	3	0	3
setCost(int)	100.0 %	4	0	4
setMass(int, int)	100.0 %	7	0	7
setVip(int, int)	100.0 %	7	0	7

Eclipse簽入修改程式碼

- 加入暫存區、簽入本地儲存區、遠端儲存區
 - 輸入密碼、[ok]



Eclipse 改修測試程式

- 新增 test ， ShoppingCartTest ，增加Code Coverage
- ShoppingCartTest.java

- 右鍵，Run As-Junit Test
 - 通過，綠色
- 右鍵，Coverage As-Junit Test

The screenshot shows the Eclipse IDE interface. At the top, there is a code editor with Java code for ShoppingCart.java. A specific line of code is highlighted with a red box: "if (cost>=mass) {". Below the code editor is a toolbar with various icons, and the "Coverage" icon is highlighted with a red box. The main workspace shows a JUnit perspective with tabs for "Markers", "Properties", "Servers", "Data Source Explorer", "Snippets", "JUnit", and "Coverage". The "Coverage" tab is active. Below the tabs is a status bar showing "ShoppingCartTest (2016/10/9 上午 11:40:42)". The bottom half of the screen is a table titled "Element" showing coverage statistics for various files and methods. The table includes columns for Element, Coverage, Covered Instructions, Missed Instructions, and Total Instructions. The "computeCost()" method in ShoppingCart.java has 100.0% coverage.

Element	Coverage	Covered Instr...	Missed Instru...	Total Instructi...
Test01	71.4 %	125	50	175
src	71.4 %	125	50	175
test.java.example	64.6 %	51	28	79
main.java.example	77.1 %	74	22	96
Cat.java	0.0 %	0	11	11
Tiger.java	0.0 %	0	11	11
ShoppingCart.java	100.0 %	74	0	74
ShoppingCart	100.0 %	74	0	74
computeCost()	100.0 %	21	0	21
computeCost()	100.0 %	32	0	32

```
public void test() {
    myCart.setCost(100);
    myCart.setMass(100, 90);
    myCart.setVip(0, 100);
    myCart.computeCost();
    assertEquals(90,myCart.getCost());
    myCart.setVip(1, 90);
    myCart.computeCost();
    assertEquals(81,myCart.getCost());
}
```

Eclipse 改修測試程式

- 新增 test ， ShoppingCartTest ， 增加Code Coverage
- ShoppingCartTest.java
 - 右鍵， Run As-Junit Test
 - 通過，綠色
 - 右鍵， Coverage As-Junit Test

The screenshot shows the Eclipse IDE interface. At the top, there's a toolbar with various icons. Below it is a navigation bar with tabs like 'Markers', 'Properties', 'Servers', 'Data Source Explorer', 'Snippets', 'JUnit', 'Coverage', etc. The main area displays the code for `ShoppingCartTest.java`:

```
26 public void computeCost() {  
27     newCost = cost;  
28     if (vip==1) {  
29         newCost = (newCost*vipDiscount)/100;  
30     }  
31     if (cost>=mass) {  
32         newCost = (newCost*massDiscount)/100;  
33     }  
34 }  
35 public int getCost() {  
36     return newCost;  
37 }  
38 }
```

A red box highlights the code between lines 28 and 34. Below the code editor is a table showing coverage statistics for different elements:

Element	Coverage	Covered Instr...	Missed Instru...	Total Instruc...
Test01	73.5 %	139	50	189
src	73.5 %	139	50	189
test.java.example	69.9 %	65	28	93
main.java.example	77.1 %	74	22	96
Cat.java	0.0 %	0	11	11
Tiger.java	0.0 %	0	11	11
ShoppingCart.java	100.0 %	74	0	74
ShoppingCart	100.0 %	74	0	74
ShoppingCart0	100.0 %	21	0	21
computeCost()	100.0 %	32	0	32

```
public void test() {  
    myCart.setCost(100);  
    myCart.setMass(100, 90);  
    myCart.setVip(0, 100);  
    myCart.computeCost();  
    assertEquals(90,myCart.getCost());  
    myCart.setVip(1, 90);  
    myCart.computeCost();  
    assertEquals(81,myCart.getCost());  
    myCart.setMass(200, 90);  
    myCart.computeCost();  
    assertEquals(90,myCart.getCost());  
}
```