**Lab#12 – Analyze the quality of your software by using software metrics**

# **วัตถุประสงค์การเรียนรู้**

1. ผู้เรียนสามารถอธิบายความหมายของ Software metrics ที่วัดออกมาได้อย่างถูกต้อง
2. ผู้เรียนสามารถใช้ Static code analysis tools ในการวิเคราะห์ปัญหาคุณภาพของซอฟต์แวร์ได้
3. ผู้เรียนสามารถอธิบายความแตกต่างระหว่าง Static testing และ Dynamic testing ได้อย่างถูกต้อง

**กิจกรรมที่ 12.1: การวิเคราะห์ปัญหาในโค้ดด้วย PMD**

1. ติดตั้ง PMD plugin ลงใน Eclipse ให้เรียบร้อย
2. สร้าง local repository ของ git และตั้งค่าให้เรียบร้อย Workspace ให้เรียบร้อย
3. Clone โค้ดตั้งต้นจาก

<https://github.com/ChitsuthaCSKKU/SQA/tree/2025/Assignment/Lab12_SoftwareMetrics>

1. ศึกษาการทำงานและโครงสร้างของ Source code ที่กำหนดให้
2. วิเคราะห์ปัญหาใน Source code โดยใช้ PMD
3. Capture หน้าจอ หรือ Export รายงานผลการทดสอบ แล้วตอบคำถามต่อไปนี้

**Q1: PMD พบ Error และ Warning ใน Whitboard.java กี่ตัว (Capture หน้าจอประกอบคำตอบ)**

ตอบA screenshot of a computer

AI-generated content may be incorrect. 43 ตัว

**Q2: จงอธิบายเกี่ยวกับปัญหา “Local variable 'managerB' could be declared final” ว่าเป็นปัญหาอะไร และมีแนวทางในการแก้ไขอย่างไร**

ตอบ ปัญหา: “Local variable 'managerB' could be declared final” มาจากกฎ PMD ชื่อ LocalVariableCouldBeFinal ซึ่งเตือนว่าตัวแปรโลคัล managerB ไม่เคยถูกกำหนดค่าใหม่หลังจากประกาศ ดังนั้นสามารถประกาศให้เป็น final ได้ เพื่อป้องกันการถูกเปลี่ยนค่าโดยไม่ตั้งใจและสื่อเจตนาให้ผู้อ่านรู้ว่าค่านี้ไม่ควรถูกแก้ไข

แนวทางแก้ไข

-เพิ่ม final หน้า type ของตัวแปรที่ไม่ถูก reassigned

**Q3: PMD ใช้ Metrics อะไรบ้างในการระบุ God class และแต่ละตัวมีความหมายว่าอะไรบ้าง**

ตอบ

* WMC (Weighted Methods per Class) ความหมาย: ผลรวมความซับซ้อนของเมธอดทั้งหมดในคลาส (โดยทั่วไปถ่วงด้วย Cyclomatic Complexity ของแต่ละเมธอด) ตีความ: ค่ายิ่งสูง = คลาสยิ่งใหญ่/ซับซ้อน/มีพฤติกรรมเยอะ
* ATFD (Access To Foreign Data) ความหมาย: จำนวนการเข้าถึงข้อมูลของคลาสอื่น เช่น เข้าถึงฟิลด์โดยตรงหรือผ่าน getter ของอ็อบเจ็กต์อื่น ตีความ: ค่ายิ่งสูง = คลาสพึ่งพาข้อมูลภายนอกมาก มีแนวโน้มทำหน้าที่ควบคุมหลายอย่างเกินไป
* TCC (Tight Class Cohesion) ความหมาย: สัดส่วนของคู่เมธอดภายในคลาสที่ใช้ฟิลด์ร่วมกัน ต่อจำนวนคู่เมธอดทั้งหมด ตีความ: ค่าใกล้ 1 = cohesion สูง (เมธอดเกี่ยวข้องกันมาก), ค่าต่ำ = cohesion ต่ำ (เมธอดแยกส่วน ไม่สัมพันธ์กัน)

**กิจกรรมที่ 12.2: การวิเคราะห์ปัญหาในโค้ดด้วย Checkstyle**

1. ติดตั้ง checkstyle plugin ลงใน Eclipse ให้เรียบร้อย
2. สร้าง local repository ของ git และตั้งค่าให้เรียบร้อย Workspace ให้เรียบร้อย
3. Clone โค้ดตั้งต้นจาก

<https://github.com/ChitsuthaCSKKU/SQA/tree/2025/Assignment/Lab12_SoftwareMetrics>

1. ศึกษาการทำงานและโครงสร้างของ Source code ที่กำหนดให้
2. วิเคราะห์ปัญหาใน Source code โดยใช้ checkstyle
3. Capture หน้าจอ หรือ Export รายงานผลการทดสอบ แล้วตอบคำถามต่อไปนี้

**Q4: จงระบุปัญหาการเขียน Code ที่ไม่ตรงกับ Coding standard ของ Java ที่ Checkstyle ระบุมา 3 รายการ พร้อมคำอธิบายว่า Code ส่วนนี้ไม่สอดคล้องกับมาตรฐานอย่างไร (Capture หน้าจอประกอบด้วย)**

ตอบ

**A screen shot of a computer

AI-generated content may be incorrect.**

1.JavadocType – ขาด Javadoc ของคลาสสาธารณะ

* ปัญหา: class Whiteboard เป็น public แต่ไม่มี Javadoc อธิบายหน้าที่ของคลาส
* ทำไมไม่สอดคล้อง: มาตรฐานกำหนดให้ public class ต้องมี Javadoc เพื่ออธิบายวัตถุประสงค์และบริบทการใช้งาน
* แนวแก้: เพิ่ม Javadoc เหนือประกาศคลาส /\*\*
  + Entry point for whiteboard demo. \*/

2.JavadocMethod – ขาด Javadoc ของเมธอดสาธารณะและพารามิเตอร์

* ปัญหา: public static void main(String[] args) ไม่มี Javadoc และไม่อธิบายพารามิเตอร์ args
* ทำไมไม่สอดคล้อง: public methods ต้องมี Javadoc และระบุ @param ให้ครบตามมาตรฐาน
* แนวแก้: /\*\*
  + Application entry point.
  + @param args command-line arguments \*/

A screenshot of a computer screen

AI-generated content may be incorrect.

3.MissingJavadocMethod – เมธอด/คอนสตรัคเตอร์แบบ public/protected ไม่มี Javadoc

* ตำแหน่ง: protected UserWhiteboard(String userId), public String getUserId(), public void addContent(String text), public String getContent(), public String toString()
* เหตุผล: มาตรฐานกำหนดให้ public/protected method ต้องมี Javadoc ระบุคำอธิบาย พร้อมแท็ก @param/@return/@since/@throws ตามความเหมาะสม
* แนวแก้: ใส่ Javadoc ให้แต่ละสมาชิก โดยอธิบายพฤติกรรม พารามิเตอร์ และค่าที่ส่งกลับ

**Q5: จาก Q4 หากต้องการแก้ไข Source code เพื่อให้ผ่านมาตรฐานการเขียนโค้ดและการตรวจสอบโดย Checkstyle จะต้องแก้ไข Code ในบรรทัดนั้น ๆ อย่างไร**

ตอบ

1. Whiteboard.java – JavadocType (ขาด Javadoc ของคลาสสาธารณะ) ให้เพิ่มบรรทัด Javadoc เหนือประกาศคลาส Whiteboard

/\*\*

* Entry point for whiteboard demo. \*/ public class Whiteboard { ... }

1. Whiteboard.java – JavadocMethod (ขาด Javadoc ของเมธอด main และ @param) ให้เพิ่มบรรทัด Javadoc เหนือเมธอด main

/\*\*

* Application entry point.
* @param args command-line arguments \*/ public static void main(String[] args) { ... }

1. UserWhiteboard.java – MissingJavadocMethod (เมธอด/คอนสตรัคเตอร์แบบ public/protected ไม่มี Javadoc) ให้เพิ่มบรรทัด Javadoc เหนือสมาชิกที่ระบุ

สำหรับคอนสตรัคเตอร์ /\*\*

* Creates a UserWhiteboard for the specified user.
* @param userId the unique identifier of the user \*/ protected UserWhiteboard(String userId) { ... }

สำหรับ getUserId /\*\*

* Returns the user ID that owns this whiteboard.
* @return the user ID \*/ public String getUserId() { ... }

สำหรับ addContent /\*\*

* Adds the given text to this whiteboard's content.
* @param text the text to add \*/ public void addContent(String text) { ... }

สำหรับ getContent /\*\*

* Returns the current content of this whiteboard.
* @return the content \*/ public String getContent() { ... }

สำหรับ toString /\*\*

* Returns a string representation of this UserWhiteboard.
* @return a string representation \*/ @Override public String toString() { ... }