

การออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับ โปรแกรมเชิงวัตถุ



นาย สมหวัง แซ่ตั้ง

สถาบันวิทยบริการ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต  
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2543

ISBN 974-347-100-6

ลิขสิทธิ์ของ จุฬาลงกรณ์มหาวิทยาลัย

DESIGN AND IMPLEMENTATION OF A MEASUREMENT TOOL  
FOR OBJECT-ORIENTED PROGRAMS



MR. SOMWANG SAE-TANG

สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements  
for the Degree of Master of Science in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2000

ISBN 974-347-100-6

หัวข้อวิทยานิพนธ์      การออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุ  
โดย                              นาย สมหวัง แซ่ตั้ง  
สาขาวิชา                      วิทยาศาสตร์คอมพิวเตอร์  
อาจารย์ที่ปรึกษา              อาจารย์ ดร.พรศิริ หมั่นไชยศรี

---

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้หัวข้อวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการ  
ศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์  
( ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว )

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ  
( รองศาสตราจารย์ ดร.วันชัย รวีไพบูลย์ )

.....อาจารย์ที่ปรึกษา  
( อาจารย์ ดร.พรศิริ หมั่นไชยศรี )

.....กรรมการ  
( อาจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ )

จุฬาลงกรณ์มหาวิทยาลัย

.....กรรมการ  
( อาจารย์ ดร.ทวีชัย เสนิงวงศ์ ณ อยุธยา )

สมหวัง แซ่ตั้ง : การออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุ

(DESIGN AND IMPLEMENTATION OF A MEASUREMENT TOOL FOR OBJECT-ORIENTED PROGRAMS) อาจารย์ที่ปรึกษา : อาจารย์ ดร. พรศิริ หมั่นไชยศรี, 77 หน้า. ISBN 974-347-100-6.

วิทยานิพนธ์นี้เป็นการออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุ ซึ่งเป็นเครื่องมือที่ใช้วัดขนาดและความซับซ้อนของโปรแกรมต้นฉบับ ที่พัฒนาด้วยภาษาจาวา คำวัดต่าง ๆ ที่วัดได้จากเครื่องมือนี้ได้แก่ จำนวนบรรทัดของโปรแกรมต้นฉบับ จำนวนเมทอดในแต่ละคลาส ระดับของการขาดความสัมพันธ์ภายในคลาส และค่าวัดของไซโคลเมตริกของแมคเคลบ เป็นต้น ผู้พัฒนาซอฟต์แวร์สามารถใช้เครื่องมือนี้เพื่อติดตามความก้าวหน้าในการพัฒนาโปรแกรม และสามารถวิเคราะห์ความซับซ้อนของแต่ละเมทอดเพื่อให้ผู้พัฒนาสามารถเลือกปรับปรุงเมทอดที่มีค่าความซับซ้อนมาก ๆ ได้ นอกจากนี้ ผู้วิเคราะห์ระบบสามารถใช้เครื่องมือนี้ในการประมาณขนาดและความซับซ้อนของโปรเจกต์ต่อไปได้ โดยเฉพาะอย่างยิ่งโปรเจกต์ที่มีลักษณะคล้ายคลึงกัน ผู้วิจัยได้พัฒนาเครื่องมือนี้โดยการอ่านโปรแกรมต้นฉบับแล้วแปลงเป็นชินแท็กซ์ทรี และทำการท่องไปบนชินแท็กซ์ทรีเพื่อเก็บข้อมูลคุณสมบัติต่างๆ ของโปรแกรมต้นฉบับที่ต้องการ จากนั้นจึงนำค่าต่าง ๆ ไปใช้ในการคำนวณหาคำวัดและแสดงผลคำวัด

สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา วิศวกรรมคอมพิวเตอร์ .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ .....

ปีการศึกษา 2543 .....

ลายมือชื่อนิสิต .....

ลายมือชื่ออาจารย์ที่ปรึกษา .....

##4071492921 : MAJOR COMPUTER SCIENCE

KEY WORD : size / complexity / Measurement / Metrics / Object-Oriented Program / method / Lack of Cohesion / McCabe's Cyclomatic Complexity Metrics

SOMWANG SAE-TANG : DESIGN AND IMPLEMENTATION OF A MEASUREMENT TOOL FOR OBJECT-ORIENTED PROGRAMS. THESIS ADVISOR : DR. PORNSIRI MUENCHAISRI, 77 pp. ISBN 974-347-100-6.

This thesis is the design and implementation of a measurement tool for object-oriented programs, which is a tool for measuring size and complexity of Java source programs. This tool reports line of code (LOC), number of methods per class, lack of cohesion, McCabe's cyclomatic complexity metrics etc. Software developers can use this tool to monitor the progress of software development and to analyze the complexity of each method. It helps developers to identify complicated methods to be improved and also helps system analysts to estimate size and complexity of next projects especially the projects similar to the previous ones. This tool starts the process by reading and transforming a source program into the syntax tree. Then the tool traverses the tree to collect the program's attributes and computes the metrics to display on the output screen.



Department Computer Engineering .....

Field of study Computer Science .....

Academic year 2543 .....

Student's signature .....

Advisor's signature .....

## กิตติกรรมประกาศ

ข้าพเจ้าใคร่ขอกราบขอบพระคุณอาจารย์ ดร.พรศิริ หมั่นไชยศรี อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้าที่ท่านเป็นผู้แนะนำให้ความรู้ คำปรึกษา ความช่วยเหลือต่างๆ ตลอดจนคอยดูแลการทำวิจัยของข้าพเจ้าอย่างดียิ่งจนสำเร็จลุล่วงลงได้ด้วยดี

ขอกราบขอบพระคุณรองศาสตราจารย์ ดร.วันชัย ธีรไพบูลย์ เป็นประธานกรรมการ อาจารย์ ดร.ธราทิพย์ สุวรรณศาสตร์ และอาจารย์ ดร.ทวีชัย เสนิงวงษ์ ณ อุษยา เป็นกรรมการสอบวิทยานิพนธ์ ซึ่งได้สละเวลาและให้คำแนะนำต่าง ๆ ในการสอบวิทยานิพนธ์ของข้าพเจ้าได้อย่างดียิ่ง

ขอขอบคุณเพื่อนร่วมงาน และเพื่อน ๆ ที่เคยได้ศึกษาคู่ด้วยกันมา ที่ได้ให้คำแนะนำและกำลังใจต่าง ๆ แก่ข้าพเจ้า และขอขอบคุณบริษัท รอยเตอร์ (ประเทศไทย) จำกัด ที่ได้ให้มีโอกาสศึกษาต่อในสาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ของภาควิชาวิศวกรรมคอมพิวเตอร์แห่งนี้

ท้ายที่สุด ข้าพเจ้าใคร่ขอกราบขอบพระคุณบิดา มารดา และพี่ น้อง ที่ได้ให้โอกาสและสนับสนุนในด้านการเงินและกำลังใจแก่ข้าพเจ้าเสมอมา

นายสมหวัง แซ่ตั้ง

สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

## สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	ง
บทคัดย่อภาษาอังกฤษ.....	จ
กิตติกรรมประกาศ.....	ฉ
สารบัญตาราง.....	ญ
สารบัญรูป.....	ฎ
บทที่ 1 บทนำ.....	1
1.1 ความเป็นมาและความสำคัญของปัญหา.....	1
1.2 วัตถุประสงค์.....	1
1.3 ขอบเขตของงานวิจัย.....	2
1.4 ขั้นตอนและวิธีการดำเนินงาน.....	2
1.5 ประโยชน์ที่คาดว่าจะได้รับ.....	2
บทที่ 2 แนวคิดและทฤษฎีที่เกี่ยวข้อง.....	3
2.1 ตัวแปลภาษา (Compiler).....	3
2.1.1 เลกซิกัลลอนาไลเซอร์.....	3
2.1.2 ซินแทกซ์ลอนาไลเซอร์.....	3
2.2 ตัววัดเชิงวัตถุ (Object –Oriented Metrics).....	4
2.2.1 จำนวนบรรทัด (Lines of code – LOC).....	4
2.2.2 จำนวนเมทอดต่อคลาส (Weighted methods per class – WMC).....	4
2.2.3 ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ (Depth of inheritance hierarchy – DIT)....	5
2.2.4 จำนวนคลาสลูก (Number of children – NOC).....	ผิดพลาด! ที่คั่นหนังสือไม่ได้กำหนด
2.2.5 ขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling between objects – CBO).....	ผิดพลาด! ที่คั่นหนังสือไม่ได้กำหนด
2.2.6 ระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of cohesion of methods – LCOM).....	6
2.2.7 ค่าวัชโรลเมตริกของแมคเคบ (Cyclomatic complexity – V(G)).....	7
บทที่ 3 การวิเคราะห์และออกแบบระบบ.....	9
3.1 โมเดลการใช้งาน.....	9
3.1.1 การดูค่าตัววัด (View Metrics).....	10
3.1.2 การดูค่าตัววัดในรูปแบบตาราง (View Tables).....	10

3.2 แผนภาพแสดงลำดับการทำงาน .....	10
3.3 แผนภาพแสดงกิจกรรม .....	12
3.4 แผนภาพคลาส .....	13
3.4.1 แพ็กเกจส่วนติดต่อผู้ใช้ .....	13
3.4.2 แพ็กเกจคอมไพเลอร์ .....	14
3.4.3 แพ็กเกจคำนวณค่าตัววัด .....	16
3.4.4 แพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด .....	16
บทที่ 4 การพัฒนาระบบ .....	23
4.1 การสร้างแพ็กเกจคอมไพเลอร์ .....	23
4.2 การสร้างแพ็กเกจคำนวณค่าตัววัด .....	24
4.2.1 คลาสตัววัด (Metrics) .....	27
4.2.2 คลาสหน่วยรู้จำเกี่ยวกับโปรเจกต์ (Project) .....	28
4.2.3 คลาสหน่วยรู้จำเกี่ยวกับคลาส (JClass) .....	29
4.2.4 คลาสหน่วยรู้จำเกี่ยวกับเมธอด (Method) .....	30
4.2.5 คลาสหน่วยรู้จำเกี่ยวกับตัวแปร (Variable) .....	31
4.3 การเก็บค่าคุณสมบัติและคำนวณค่าตัววัด .....	32
4.3.1 ชุดคำสั่งการนับจำนวนสเตทเมนต์ .....	32
4.3.2 ชุดคำสั่งหาค่าวัดของไซโคลเมตริกของแมคเคบ .....	33
4.3.3 ชุดคำสั่งหาค่าวัดจำนวนบรรทัดของโปรเจกต์ .....	34
บทที่ 5 การใช้งานระบบ MTOOP .....	36
5.1 การอ่านโปรแกรมต้นฉบับ .....	36
5.2 การดูค่าตัววัด .....	37
5.3 การเก็บและเรียกดูค่าตัววัดจากฐานข้อมูล .....	40
บทที่ 6 การทดสอบ .....	42
6.1 การเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดและการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับการวัดด้วยมือ .....	42
6.1.1 ผลการทดสอบโปรแกรมที่ 1 .....	42
6.2 การเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับเครื่องมือวัดที่มีใช้อยู่ในขณะนี้ .....	46
6.2.1 ผลการทดสอบโปรแกรมที่ 2 .....	46
6.2.2 ผลการทดสอบโปรแกรมที่ 3 .....	51
6.2.3 ผลการทดสอบโปรแกรมที่ 4 .....	55
6.2.4 ผลการทดสอบโปรแกรมที่ 5 .....	59
6.2.5 ผลการทดสอบโปรแกรมที่ 6 .....	63



6.3 สรุปผลการทดสอบ .....	69
บทที่ 7 บทสรุปและข้อเสนอแนะ.....	70
7.1 บทสรุป.....	70
7.2 ข้อเสนอแนะ .....	71
7.3 ผลงานตีพิมพ์.....	71
รายการอ้างอิง.....	72
ภาคผนวก ก. ผลงานตีพิมพ์.....	74
ประวัติผู้เขียนวิทยานิพนธ์.....	77



สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย



## สารบัญรูป

	หน้า
รูปที่ 2.1 หน่วยย่อย ๆ ของการสร้างตัวแปลภาษา.....	3
รูปที่ 2.2 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าขนาดความสัมพันธ์ระหว่างวัตถุ .....	6
รูปที่ 2.3 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าระดับของการขาดความสัมพันธ์ภายในคลาส .....	7
รูปที่ 2.4 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าวัชโรคมติของแมคเคลบ.....	8
รูปที่ 3.1 โมเดลการใช้งาน.....	9
รูปที่ 3.2 แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้ต้องการดูค่าตัววัด .....	11
รูปที่ 3.3 แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้ต้องการเก็บค่าตัววัดลงฐานข้อมูล.....	11
รูปที่ 3.4 แผนภาพแสดงกิจกรรมของเครื่องมือ MTOOP .....	12
รูปที่ 3.5 แผนภาพแสดงความสัมพันธ์ระหว่างแพ็คเกจต่าง ๆ ของระบบ .....	13
รูปที่ 3.6 แผนภาพคลาสในภาพรวมของแพ็คเกจส่วนติดต่อผู้ใช้.....	14
รูปที่ 3.7 แผนภาพคลาสของแพ็คเกจคอมไพเลอร์ .....	14
รูปที่ 3.8 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่มาจากการสร้างซึนแท็กซ์ทรีของคลาสจาวาพาร์เซอร์ .....	15
รูปที่ 3.8 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่มาจากการสร้างซึนแท็กซ์ทรีของคลาสจาวาพาร์เซอร์ (ต่อ).....	16
รูปที่ 3.9 แผนภาพคลาสของแพ็คเกจคำนวณค่าตัววัด.....	17
รูปที่ 3.10 แผนภาพคลาสของแพ็คเกจจัดรูปแบบการแสดงผลค่าตัววัด.....	17
รูปที่ 4.1 แผนภาพแสดงขั้นตอนเพื่อสร้างตัวสร้างซึนแท็กซ์ทรี .....	23
รูปที่ 4.2 การรับโปรแกรมต้นฉบับจากส่วนติดต่อผู้ใช้ และเริ่มสร้างซึนแท็กซ์ทรี.....	24
รูปที่ 4.3 ความแตกต่างระหว่างก่อนแก้ไขและหลังแก้ไข PackageDeclaration() .....	25
รูปที่ 4.4 การเก็บค่า packages ในตัวแปรเวคเตอร์.....	25
รูปที่ 4.5 ความแตกต่างระหว่างก่อนแก้ไขและหลังแก้ไข ImportDeclaration().....	26
รูปที่ 4.6 การเก็บค่า importCollection ในตัวแปรเวคเตอร์ .....	26
รูปที่ 4.7 รายละเอียดภายในคลาส "Metrics" .....	27
รูปที่ 4.8 รายละเอียดภายในคลาส "Project" .....	28
รูปที่ 4.9 รายละเอียดภายในคลาส "JClass" .....	29
รูปที่ 4.10 รายละเอียดภายในคลาส "Method" .....	30
รูปที่ 4.11 รายละเอียดภายในคลาส "Variable" .....	31
รูปที่ 4.12 รายละเอียดชุดคำสั่งการนับจำนวนสเตทเมนต์.....	32
รูปที่ 4.13 รายละเอียดชุดคำสั่งหาค่าวัชโรคมติของวัชโรคมติของแมคเคลบ.....	33
รูปที่ 4.14 รายละเอียดชุดคำสั่งหาค่าวัชโรคมติจำนวนบรรทัดของโปรแกรม.....	34

รูปที่ 5.1	เฟรมเพื่อเลือกโปรแกรมต้นฉบับภาษาจาวา .....	36
รูปที่ 5.2	ทรีและโหนดต่างๆ ที่ได้จากการเลือกโปรแกรมต้นฉบับ .....	37
รูปที่ 5.3	การดูค่าตัววัดตามประเภทของโปรเจกต์เมื่อคลิกโหนดที่ชื่อ "Project" .....	37
รูปที่ 5.4	การดูค่าตัววัดตามประเภทของแพ็คเกจเมื่อคลิกโหนดที่ชื่อ "None" .....	38
รูปที่ 5.5	การดูค่าตัววัดตามประเภทของคลาสเมื่อคลิกโหนดที่ชื่อ "SortItem" .....	38
รูปที่ 5.6	การดูค่าตัววัดตามประเภทของเมธอดเมื่อคลิกโหนดที่ชื่อ "paint(Graphics)" .....	39
รูปที่ 5.7	การดูค่าตัววัดตามประเภทของตัวแปรอินสแตนซ์เมื่อคลิกโหนดที่ชื่อ "parent" .....	39
รูปที่ 5.8	ตัวอย่างการดูค่าตัววัดรูปแบบตาราง .....	40
รูปที่ 5.9	การกำหนดค่าค่าดัชนีซอร์สเนมและไฟล์ค่าดาเบสในเครื่องมือจัดการ โอดีบีซี .....	40
รูปที่ 5.10	ตัวอย่างการใช้ฟังก์ชันการบันทึกข้อมูลลงฐานข้อมูล .....	41
รูปที่ 5.11	ตัวอย่างการใช้ฟังก์ชันดูค่าตัววัดจากฐานข้อมูล .....	41
รูปที่ 6.1	โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 1 .....	42
รูปที่ 6.1	โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 1 (ต่อ) .....	43
รูปที่ 6.2	ค่าตัววัดสำหรับโปรเจกต์ ของโปรแกรมทดสอบที่ 1 .....	43
รูปที่ 6.3	ค่าตัววัดสำหรับแพ็คเกจที่ชื่อ "None" ของโปรแกรมทดสอบที่ 1 .....	44
รูปที่ 6.4	ค่าตัววัดสำหรับคลาสที่ชื่อ "CardLayoutTest" ของโปรแกรมทดสอบที่ 1 .....	44
รูปที่ 6.5	ค่าตัววัดสำหรับเมธอดที่ชื่อ "main( array )" ของโปรแกรมทดสอบที่ 1 .....	44
รูปที่ 6.6	ค่าตัววัดสำหรับตัวแปรอินสแตนซ์ที่ชื่อ "cards" ของโปรแกรมทดสอบที่ 1 .....	45
รูปที่ 6.7	ค่าตัววัดของแพ็คเกจในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 1 .....	45
รูปที่ 6.8	ค่าตัววัดของคลาสในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 1 .....	45
รูปที่ 6.9	ค่าตัววัดของเมธอดในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 1 .....	46
รูปที่ 6.10	ค่าตัววัดของแพ็คเกจในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 2 .....	47
รูปที่ 6.11	ค่าตัววัดของคลาสในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 2 .....	48
รูปที่ 6.12	ค่าตัววัดของเมธอดในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 2 .....	49
รูปที่ 6.13	ตัวอย่างการนับจำนวนบรรทัดของเมธอด "_parseArg(String[])" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 2 .....	50
รูปที่ 6.14	ค่าตัววัดของแพ็คเกจในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 3 .....	51
รูปที่ 6.14	ค่าตัววัดของแพ็คเกจในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 3 (ต่อ) .....	52
รูปที่ 6.15	ค่าตัววัดของคลาสในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 3 .....	52
รูปที่ 6.15	ค่าตัววัดของคลาสในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 3 (ต่อ) .....	53
รูปที่ 6.16	ค่าตัววัดของเมธอดในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 3 .....	53
รูปที่ 6.17	ตัวอย่างการนับจำนวนบรรทัดของเมธอด "informRead(HttpURLConnection, Hashtable)" ของ เครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 3 .....	54
รูปที่ 6.18	ค่าตัววัดของแพ็คเกจในโปรเจกต์ สำหรับโปรแกรมทดสอบที่ 4 .....	55

รูปที่ 6.18 ค่าตัววัดของแฟ็กเกจในโปรเจค สำหรับโปรแกรมทดสอบที่ 4 (ต่อ) .....	56
รูปที่ 6.19 ค่าตัววัดของคลาสในโปรเจค สำหรับโปรแกรมทดสอบที่ 4.....	56
รูปที่ 6.20 ค่าตัววัดของเมทอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 4 .....	57
รูปที่ 6.21 ตัวอย่างการนับจำนวนบรรทัดของเมทอด "of(Document)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 4 .....	58
รูปที่ 6.22 ค่าตัววัดของแฟ็กเกจในโปรเจค สำหรับโปรแกรมทดสอบที่ 5.....	59
รูปที่ 6.22 ค่าตัววัดของแฟ็กเกจในโปรเจค สำหรับโปรแกรมทดสอบที่ 5 (ต่อ).....	60
รูปที่ 6.23 ค่าตัววัดของคลาสในโปรเจค สำหรับโปรแกรมทดสอบที่ 5.....	61
รูปที่ 6.24 ค่าตัววัดของเมทอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 5 .....	62
รูปที่ 6.25 ตัวอย่างการนับจำนวนบรรทัดและการหาค่าไซโคลเมตริกของเมคเคบของเมทอด "reloadDirectory (String)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของ โปรแกรมต้นฉบับทดสอบที่ 5 .....	63
รูปที่ 6. 26 ค่าตัววัดของแฟ็กเกจในโปรเจค สำหรับโปรแกรมทดสอบที่ 6.....	64
รูปที่ 6.26 ค่าตัววัดของแฟ็กเกจในโปรเจค สำหรับโปรแกรมทดสอบที่ 6 (ต่อ).....	65
รูปที่ 6. 27 ค่าตัววัดของคลาสในโปรเจค สำหรับโปรแกรมทดสอบที่ 6.....	65
รูปที่ 6. 27 ค่าตัววัดของคลาสในโปรเจค สำหรับโปรแกรมทดสอบที่ 6.....	66
รูปที่ 6.28 ค่าตัววัดของเมทอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 6 .....	66
รูปที่ 6.28 ค่าตัววัดของเมทอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 6 .....	67
รูปที่ 6.29 ตัวอย่างการนับจำนวนบรรทัดและการหาค่าไซโคลเมตริกของเมคเคบของเมทอด "loadMenu(String, boolean)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของ โปรแกรมต้นฉบับทดสอบที่ 6.....	68

# บทที่ 1

## บทนำ

### 1.1 ความเป็นมาและความสำคัญของปัญหา

แม้ว่าปัจจุบันนี้จะมีเทคโนโลยีทางคอมพิวเตอร์และวิธีการใหม่ ๆ มากมายมาช่วยในการพัฒนาซอฟต์แวร์ แต่ในการพัฒนาซอฟต์แวร์ยังคงประสบปัญหาต่างๆ เหล่านี้ เช่น การพัฒนาซอฟต์แวร์ใช้ระยะเวลาในการพัฒนาไม่เป็นไปตามกำหนด ซอฟต์แวร์ที่พัฒนาได้มีขนาดใหญ่และความซับซ้อนมาก ทำให้ยากต่อการทำความเข้าใจ การนำกลับมาใช้ใหม่ การแก้ไขและการบำรุงรักษา เป็นต้น ทั้งนี้อาจเนื่องมาจากว่าผู้พัฒนาไม่มีการใช้เครื่องมือวัดซอฟต์แวร์ในการควบคุมและติดตามความก้าวหน้าของการพัฒนาซอฟต์แวร์ ผู้วิจัยตระหนักถึงความสำคัญของการวัด (Measurement) ของซอฟต์แวร์ ว่ามีความจำเป็นมากสำหรับการพัฒนาซอฟต์แวร์ การวัดช่วยในการวางแผนการประมาณ การควบคุมโปรเจกต์ ช่วยปรับปรุงคุณภาพของซอฟต์แวร์ให้ดีขึ้น ด้วยการลดขนาดและการลดความซับซ้อนของซอฟต์แวร์ที่มีขนาดใหญ่หรือมีความซับซ้อนมาก นอกจากนี้ยังจะช่วยในการตัดสินใจว่าจะดำเนินงานโปรเจกต์ต่อไปหรือไม่

ดังนั้นผู้ทำวิจัยจึงได้ทำการออกแบบและพัฒนาเครื่องมือที่จะนำมาใช้ในการวัดโปรแกรมเชิงวัตถุ (Object-Oriented Program) ที่พัฒนาด้วยภาษาจาวา การวัดจะแสดงผลของการนับและการคำนวณค่าต่าง ๆ ตามสูตรของการวัดที่ต้องการ โดยการอ่านโปรแกรมต้นฉบับแล้วแปลงเป็นซินแท็กซ์ทรี และทำการท่องไปบนซินแท็กซ์ทรีเพื่อเก็บข้อมูลคุณสมบัติต่างๆ ของโปรแกรมต้นฉบับที่ต้องการ จากนั้นจึงนำค่าต่าง ๆ ไปใช้ในการคำนวณหาค่าวัดและแสดงผลค่าวัด นอกจากนี้ยังสามารถนำค่าตัววัดที่ได้ทำการบันทึกลงในฐานข้อมูล เพื่อนำมาแสดงผลในภายหลังได้ และนำค่าที่ได้มาหาความสัมพันธ์หรือเปรียบเทียบกันเพื่อใช้ในการวิเคราะห์ต่อไป

### 1.2 วัตถุประสงค์

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อสร้างเครื่องมือวัดซอฟต์แวร์ที่

1.2.1. ช่วยในการวัดซอฟต์แวร์ที่พัฒนาด้วยโปรแกรมภาษาจาวา

1.2.2. ช่วยให้ทราบค่าวัดต่าง ๆ ตามตัววัดของซอฟต์แวร์ที่ใช้ในงานวิจัยนี้

1.2.3. นำค่าวัดต่าง ๆ ที่ได้เก็บเป็นฐานข้อมูล เพื่อนำไปเป็นข้อมูลในการพัฒนาซอฟต์แวร์อื่น ๆ และเพื่อเปรียบเทียบค่าที่วัดได้ของเครื่องมือวัดซอฟต์แวร์ที่พัฒนากับค่าที่วัดได้ของเครื่องมือวัดซอฟต์แวร์ที่มีใช้อยู่ในขณะนี้

### 1.3 ขอบเขตของงานวิจัย

วิทยานิพนธ์นี้มีขอบเขตของการวิจัยดังนี้

- 1.3.1. ใช้โปรแกรมภาษาจาวาในการพัฒนาเครื่องมือ
- 1.3.2. โปรแกรมต้นฉบับที่นำมาหาค่าตัววัด ต้องเป็นโปรแกรมภาษาจาวาและผ่านการคอมไพล์แล้ว
- 1.3.3. พัฒนาและใช้เครื่องมือวัดซอฟต์แวร์นี้บนระบบปฏิบัติการวินโดวส์ (WINDOWS) ตั้งแต่รุ่น 95 ขึ้นไป
- 1.3.4. สามารถประมวลผลได้มากกว่า 1 แฟ้มข้อมูล ขึ้นอยู่กับการนำแฟ้มข้อมูลเข้าไปประมวลผล แต่จะไม่ทำการค้นหาแฟ้มข้อมูลทั้งหมดที่เกี่ยวข้องให้
- 1.3.5. จำนวนซอฟต์แวร์ที่ใช้ในการทดสอบมีอย่างน้อย 5 ซอฟต์แวร์
- 1.3.6. ขนาดของซอฟต์แวร์ที่ใช้ในการทดสอบต้องมีขนาดอย่างน้อย 2 KLOC แต่ไม่เกิน 50 KLOC
- 1.3.7. จำนวนตัววัดที่ใช้ในการวัดค่าจะใช้อย่างน้อย 5 ตัววัด จากที่ได้ทำการศึกษาในงานวิจัยนี้
- 1.3.8. สามารถเก็บข้อมูลค่าตัววัดต่าง ๆ ลงฐานข้อมูลของไมโครซอฟต์แอคเซส (Microsoft Access) ได้
- 1.3.9. เครื่องมือที่ใช้ในการเปรียบเทียบได้แก่ JMetric หรือ JavaNCSS เป็นต้น

### 1.4 ขั้นตอนและวิธีการดำเนินงาน

วิทยานิพนธ์นี้มีขั้นตอนและวิธีการดำเนินงานดังนี้

- 1.4.1. ศึกษาแนวคิดและทฤษฎีที่เกี่ยวกับการวัดซอฟต์แวร์เชิงวัตถุ
- 1.4.2. ศึกษาเครื่องมือวัดซอฟต์แวร์ที่มีใช้อยู่ในขณะนี้
- 1.4.3. ศึกษางานวิจัยที่เกี่ยวข้อง ได้แก่ การออกแบบและพัฒนาเครื่องมือช่วยในการทำความเข้าใจโปรแกรมภาษาจาวา [12]
- 1.4.4. ศึกษาเครื่องมือที่ช่วยในการสร้างตัวแปลภาษา ที่มีการสร้างซอร์สโค้ดของตัวแปลภาษาเป็นภาษาจาวา ได้แก่ โปรแกรม Javacc ของบริษัท Sun Microsystems, Inc.
- 1.4.5. ศึกษาหลักไวยากรณ์ต่าง ๆ ของโปรแกรมภาษาจาวา เพื่อหาวิธีในการวัดซอฟต์แวร์
- 1.4.6. ออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์
- 1.4.7. ทดสอบเครื่องมือวัดซอฟต์แวร์
- 1.4.8. สรุปผลและเสนอแนะผลของการวิจัย

### 1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับจากวิทยานิพนธ์นี้และเครื่องมือวัดซอฟต์แวร์มีดังนี้

- 1.5.1. ช่วยให้ทราบคุณสมบัติต่าง ๆ ของซอฟต์แวร์ โดยไม่ต้องนับด้วยตนเอง
- 1.5.2. มีฐานข้อมูลที่ช่วยในการประมาณเวลาและกำลังคนที่ใช้ในการพัฒนาซอฟต์แวร์
- 1.5.3. เป็นแนวทางสำหรับการพัฒนาเครื่องมือที่ใช้วัดซอฟต์แวร์ที่มีสูตรการวัดนอกเหนือจากงานวิจัยนี้

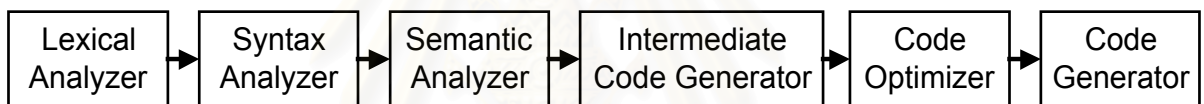
## บทที่ 2

### แนวคิดและทฤษฎีที่เกี่ยวข้อง

ในบทนี้ได้สรุปเนื้อหาของเรื่องต่าง ๆ ที่เกี่ยวข้องที่ได้ศึกษาเพื่อนำมาใช้ประกอบโครงงานวิจัยนี้ ได้แก่ ตัวแปลภาษา และตัววัดเชิงวัตถุ ดังรายละเอียดต่อไปนี้

#### 2.1 ตัวแปลภาษา (Compiler)

ตัวแปลภาษาเป็นโปรแกรมที่เปลี่ยนโปรแกรมภาษาคอมพิวเตอร์ภาษาหนึ่งไปเป็นโปรแกรมภาษาคอมพิวเตอร์อีกภาษาหนึ่ง โปรแกรมต้นฉบับที่ถูกแปลงเรียกว่าซอร์สโปรแกรม (Source program) ส่วนโปรแกรมที่ได้เรียกว่าออบเจกต์โปรแกรม (Object program) ในการสร้างตัวแปลภาษานิยมแบ่งเป็นหน่วยย่อย ๆ ดังรูปที่ 2.1



รูปที่ 2.1 หน่วยย่อย ๆ ของการสร้างตัวแปลภาษา

สามส่วนแรกได้แก่ เลกซิคัลลอนาไลเซอร์ (Lexical analyzer) ซินแทกซ์ลอนาไลเซอร์ (Syntax analyzer) และ ซีแมนติกลอนาไลเซอร์ (Semantic analyzer) เรียกว่าส่วน ฟรอนต์เอนด์ (Front end) ของตัวแปลภาษา มีหน้าที่วิเคราะห์การรู้จำ (Concept) จากตัวโปรแกรม สามส่วนหลังได้แก่ อินเตอร์มีเดียดโค้ดเจเนอเรเตอร์ (Intermediate code generator) โค้ดออปติไมเซอร์ (Code optimizer) และ โค้ดเจเนอเรเตอร์ (Code generator) เรียกว่าส่วน แบคเอนด์ (Back end) ของตัวแปลภาษา มีหน้าที่นำส่วนวิเคราะห์การรู้จำ ที่ได้ไปเปลี่ยนเป็นออบเจกต์โปรแกรม

ส่วนของตัวแปลภาษาที่ต้องใช้ใน โครงงานวิจัยนี้มี 2 ส่วนดังนี้

##### 2.1.1 เลกซิคัลลอนาไลเซอร์

เลกซิคัลลอนาไลเซอร์ หรือเรียกอีกชื่อหนึ่งว่า สแกนเนอร์ (Scanner) มีหน้าที่อ่านอักขระจากภาษาต้นแบบ แล้วจัดการแยกกลุ่มอักขระเหล่านั้นออกเป็นโทเคน (Token)

##### 2.1.2 ซินแทกซ์ลอนาไลเซอร์

ซินแทกซ์ลอนาไลเซอร์ หรือเรียกอีกชื่อหนึ่งว่า พาร์เซอร์ (Parser) มีหน้าที่ตรวจสอบว่าโทเคนที่ได้จากขั้นตอน เลกซิคัลลอนาไลเซอร์ เรียงกันถูกต้องตามหลักไวยากรณ์ของภาษาที่กำหนดไว้หรือไม่ โดยจะได้เป็นโครงสร้างต้นไม้ที่เรียกว่า เอเอสที (AST – Abstract Syntax Tree) เป็นโครงสร้างต้นไม้ที่มีการลดทอนให้เหลือแต่ส่วนที่จำเป็นต้องใช้ในการสร้างโค้ดเท่านั้น



ในแต่ละโหนด (Node) ของเอเอสทีจะเก็บข้อมูลหรือคุณสมบัติต่าง ๆ ที่อยู่ภายในโหนด เช่น เป็นคลาส เป็นเมธอด เป็นคำสั่งเงื่อนไข เป็นต้น ซึ่งเราสามารถหาข้อมูลหรือคุณสมบัติต่าง ๆ เหล่านี้ ด้วยการท่องไปตามโหนด (Traverse node) ต่าง ๆ และทำการนับเพื่อหาค่าวัดตามตัววัดที่เราต้องการได้

## 2.2 ตัววัดเชิงวัตถุ (Object –Oriented Metrics)

ในหัวข้อนี้สรุปเนื้อหาของงานวิจัยต่าง ๆ ที่เกี่ยวข้องกับการวิจัยนี้ ได้แก่ งานวิจัยของ McCabe ซึ่งได้ออกแบบค่าวัดไซโคลเมตริกของแมคเคบ (Cyclomatic complexity metric  $V(G)$ ) [5, 6, 7, 8, 9] และงานวิจัยของ Chidamber and Kemerer ได้ออกแบบชุดของการวัดสำหรับการออกแบบเชิงวัตถุ (Object-oriented design) โดยชุดของการวัดนี้มีด้วยกัน 6 ค่าคือ จำนวนเมธอดต่อคลาส (Weighted methods per class: WMC) ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ (Depth of the inheritance tree : DIT) จำนวนคลาสลูก (Number of children : NOC) ขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling between objects : CBO) ระดับการตอบสนองต่อคลาส (Response for a class : RFC) และระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of cohesion of methods : LCOM) [1, 2, 3, 4, 6, 7, 8]

ตัววัดที่เลือกใช้ในงานวิจัยเพื่อทำการหาค่าตัววัด ได้จากการศึกษาเครื่องมือวัดที่มีอยู่ในขณะนี้ เช่น JMetric เป็นต้น และได้จากการรวบรวมจากผลงานวิจัยของ McCabe และ Chidamber and Kemerer ดังรายละเอียดต่อไปนี้

### 2.2.1 จำนวนบรรทัด (Lines of code – LOC)

การวัดวิธีนี้เป็นวิธีที่ง่ายที่สุดในจำนวนการวัดทั้งหมดที่สามารถนำมากำหนดและคำนวณ การนับจำนวนบรรทัดของโค้ดเป็นวิธีการวัดที่มีมาตั้งแต่การเขียนโปรแกรมยังเป็นการโปรแกรมแบบโครงสร้าง (Structured programming) และยังมีนิยมใช้กันอย่างแพร่หลายมาจนถึงปัจจุบัน ขนาดของโปรแกรมต้นฉบับมีผลกระทบกับความยากง่ายในการทำความเข้าใจ ความสามารถนำมาใช้ใหม่ (Reusability) และความสามารถในการบำรุงรักษา (Maintainability) การคำนวณจำนวนบรรทัดของโปรแกรมต้นฉบับมีด้วยกันหลายวิธี เช่น อาจจะได้มาจากผลรวมของการนับจำนวนสเตตเมนต์ (Statement) จำนวนบรรทัดว่าง (Blank lines) จำนวนบรรทัดของคำอธิบาย (Lines of comments) และจำนวนบรรทัดที่เป็นวงเล็บปีกกา (Block delimiters) เป็นต้น ซึ่งขึ้นอยู่กับข้อกำหนดในการนับจำนวนบรรทัดว่าจะรวมค่าอะไรบ้าง

### 2.2.2 จำนวนเมธอดต่อคลาส (Weighted methods per class – WMC)

ถ้าจำนวนเมธอดในคลาสสามารถกำหนดขึ้นมาในช่วงของการออกแบบและการเขียนโมเดลคลาส (Design and modeling phase) ของโปรเจกต์ ค่าตัววัดนี้ก็สามารถจะนำมาเป็นตัวทำนายว่าจะใช้เวลาและความพยายาม (Effort) มากเพียงใดในการพัฒนา แก้ไข และบำรุงรักษา การวัดนี้ยังได้แก้ไขต่อไปโดยการถ่วงน้ำหนัก

ของความซับซ้อน (Weighting for complexity) ของแต่ละโมดูลโดยการถ่วงน้ำหนัก ปกติจะได้มาจากแบบค่าวัชไรซ์โคเลเมติกของแมทริกซ์ของเมทริกซ์นั้น

ชั้นคลาสของคลาสจะถ่ายทอดเมทริกซ์ที่เป็น public และ protected ทั้งหมด (อาจรวมถึงเมทริกซ์ที่เป็นแพ็คเกจด้วยก็ได้) ของคลาส ดังนั้นจำนวนของเมทริกซ์ในคลาสจะมีผลกระทบโดยตรงกับความซับซ้อนของชั้นคลาส คลาสที่มีจำนวนของเมทริกซ์มาก ๆ ก็มักจะมีลักษณะเฉพาะแน่นอนอนที่จะใช้กับแอปพลิเคชัน (Application) ใด ๆ ทำให้ลดความสามารถที่จะนำมาใช้ได้ใหม่กับแอปพลิเคชันอื่น ๆ

### 2.2.3 ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ (Depth of inheritance hierarchy – DIT)

การวัดวิธีนี้เป็นการหาค่าระดับความลึกของการสืบทอดคุณสมบัติ (Inheritance) ของคลาส ว่ามีการถ่ายทอดกันมากน้อยเพียงใด ดังนั้นเมื่อมีการสืบทอดคุณสมบัติ ชั้นคลาสแรกก็จะมีค่าเท่ากับ 1 และชั้นคลาสของชั้นคลาสแรกก็จะมีค่าเท่ากับ 2 และเพิ่มมากขึ้นเรื่อย ๆ ในชั้นคลาสตัวต่อ ๆ ไป

คลาสใด ๆ ที่มีความลึกในทรี (Tree) มากก็จะสามารถสืบทอดคุณสมบัติของเมทริกซ์และสถานะของตัวแปร (State variables) ได้มาก แต่อย่างไรก็ตามการสืบทอดคุณสมบัติก็จะยิ่งเพิ่มความซับซ้อนและทำให้ยากต่อการทำนายคุณสมบัติ (Behavior) ของคลาส และยากที่จะเข้าใจระบบที่มีการสืบทอดคุณสมบัติมาก ๆ อีกด้วย

### 2.2.4 จำนวนคลาสลูก (Number of children – NOC)

ขนาดโดยรวมของระบบสามารถนำมาประมาณค่าได้โดยการคำนวณจำนวนของคลาสที่มีอยู่ในระบบใหญ่ ๆ ที่มีคลาสมาก ๆ จะมีความซับซ้อนมากกว่าระบบเล็ก ๆ เนื่องจากว่าจำนวนของปฏิสัมพันธ์ที่มีระหว่างออปเจกต์สูงขึ้น ทำให้ความเข้าใจในระบบลดน้อยลงเป็นผลให้ยากที่จะทำการทดสอบ แก้ไขและบำรุงรักษา

ถ้าจำนวนของคลาสในระบบ สามารถถูกควบคุมไว้ได้ในช่วงของการออกแบบในระยะเริ่มต้นของโปรเจกต์ก็จะสามารถช่วยเป็นพื้นฐานในการนำมาประมาณความพยายาม ค่าใช้จ่ายในการพัฒนา การแก้ไขและการบำรุงรักษาระบบทั้งหมดได้

### 2.2.5 ขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling between objects – CBO)

เมื่อมีออปเจกต์หรือคลาสหนึ่งใช้ออปเจกต์หรือคลาสอื่น ๆ เราจะเรียกการกระทำเช่นนี้ว่าการเข้าคู่กัน (Coupling) แหล่งของการเข้าคู่กันส่วนใหญ่ก็จะเป็นการเข้าคู่กันระหว่างซูเปอร์คลาสและชั้นคลาส การเข้าคู่กันยังรวมไปถึงเมื่อเมทริกซ์หรือฟิลด์ในคลาสอื่น ๆ ถูกเรียกใช้ หรือเมื่อออปเจกต์ของคลาสอื่น ๆ มีการส่งคำสั่งร้องขอโดยผ่านการอ้างอิงถึงในเมทริกซ์ การวัดการเข้าคู่กันระหว่างออปเจกต์จะเป็นการวัดการเข้าคู่กันแบบไม่ถ่ายทอด (Non-inheritance coupling) ระหว่าง 2 ออปเจกต์

ถ้าค่าของการเข้าคู่กันมีค่าสูงจะลดความเป็นโมดูลของคลาสและจะทำให้การนำมาใช้ใหม่ทำได้ยากยิ่งขึ้น แสดงว่ายิ่งคลาสต่าง ๆ มีความเป็นอิสระกันมากเท่าใดก็ดูเหมือนจะมีความเป็นไปได้ที่จะทำให้คลาสนั้นสามารถนำกลับมาใช้ในส่วนอื่น ๆ ของระบบได้ เมื่อคลาสมีการเข้าคู่กันกับคลาสอื่นจะทำให้เกิดผลกระทบได้ง่ายในเวลาที่จะทำการเปลี่ยนแปลงใด ๆ ในคลาสนั้นและยังจะทำให้การบำรุงรักษาทำได้ยากด้วย นอกจากนี้คลาสนี้ขึ้นอยู่กับคลาสอื่นมากเกินไปก็จะทำให้เป็นการยากในการทำความเข้าใจและทดสอบคลาสนั้นแยกต่างหากออกมา

การหาค่าขนาดความสัมพันธ์ระหว่างวัตถุสามารถแสดงให้เห็นดังตัวอย่างต่อไปนี้

```
class A {
    public int x() {
        ...
    }
}

class B {
    private A a;
    public void y() {
        a.x();
        ...
    }
}

class C {
    private A a;
    private B b;
    public void z() {
        a.x();
        b.y();
        ...
    }
}
```

รูปที่ 2.2 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าขนาดความสัมพันธ์ระหว่างวัตถุ

จากตัวอย่างโปรแกรมในรูปที่ 2.2 ค่าขนาดความสัมพันธ์ระหว่างวัตถุของคลาส B จะมีค่าเท่ากับ 1 เพราะมีการเรียกใช้เมทอดในคลาส A และค่าขนาดความสัมพันธ์ระหว่างวัตถุของคลาส C จะมีค่าเท่ากับ 2 เพราะมีการเรียกใช้เมทอดในคลาส A และคลาส B

#### 2.2.6 ระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of cohesion of methods – LCOM)

การเกาะกันเป็นก้อน (Cohesion) ของคลาสคือการที่จะบอกว่าเมทอดของคลาสนั้นมีความสัมพันธ์กับเมทอดอื่น ๆ อย่างไร จะถูกกำหนดโดยการยกตัวอย่างรูปแบบของตัวแปรอินสแตนซ์ที่ถูกใช้งานในกลุ่มของเมทอด

ถ้าเมทอดทั้งหมดใช้งานตัวแปรอินสแตนซ์ที่ตัวเดียวกันแล้ว แสดงว่ามันมีการเกาะกันเป็นก้อนสูง ถ้าการใช้งานอยู่ในกลุ่มของตัวแปรที่แตกต่างกันแสดงว่ามีการเกาะกันเป็นก้อนต่ำ ตัวอย่างที่เห็นได้ชัดเจนที่สุดที่แสดงให้เห็นว่ามีการเกาะกันเป็นก้อนต่ำมากก็คือการที่ไม่มีเมทอดใดเลยที่ใช้ตัวแปรอินสแตนซ์

ถ้าคลาสแสดงให้เห็นว่ามีการเกาะกันเป็นก้อนต่ำจะชี้ให้เห็นว่าการออกแบบของคลาสมีบางสิ่งบางอย่างไม่ถูกต้องและสามารถทำให้ดีขึ้นได้โดยการแบ่งคลาสต่าง ๆ ออกเป็นคลาสย่อย ๆ ที่จะทำให้มีการเกาะกันเป็นก้อนสูงขึ้น หรือในอีกความหมายหนึ่งก็คือการที่มีค่าของการเกาะกันเป็นก้อนสูง (การขาดการเกาะกันเป็นก้อนต่ำ (Low lack of cohesion)) แสดงว่าคลาสของเรามีการออกแบบที่ดี การเกาะกันเป็นก้อนของคลาสจะเป็นการบอกแนวโน้มของการเ็นแคปซูลชัน (Encapsulation) ด้วยเช่นกัน ดังนั้นการขาดการเกาะกันเป็นก้อนสูงจะทำให้แนวโน้มของการเ็นแคปซูลชัน มีค่าต่ำ ซึ่งจะทำให้มีความซับซ้อนของคลาสสูง

จากผลงานวิจัยของ Brian Henderson-Sellers ได้เสนอสูตรในการหาค่าระดับของการขาดความสัมพันธ์ภายในคลาสดังนี้

$$\text{ค่าระดับของการขาดความสัมพันธ์ภายในคลาส} = \frac{\left\{ \frac{1}{a} \sum_{k=1}^a M(A_k) \right\} - m}{(1 - m)}$$

โดยที่ :

a = จำนวนตัวแปรอินสแตนซ์ที่ประกาศในคลาส (Instance variables in class)

m = จำนวนเมธอดในคลาส (Number of methods in class)

$M(A_k)$  = จำนวนครั้งที่ตัวแปรอินสแตนซ์ k ที่ประกาศในคลาสถูกใช้ในเมธอดใด ๆ ในคลาส  
(Instance variables accessed by method)

การหาค่าระดับของการขาดความสัมพันธ์ภายในคลาสสามารถแสดงให้เห็นดังตัวอย่างต่อไปนี้

```
class A {
    private int x;
    public int y;

    public void m1() {
        x = 1;
    }
    private void m2() {
        y = 1;
    }
    void m3() {
        x = 10;
    }
}
```

รูปที่ 2.3 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าระดับของการขาดความสัมพันธ์ภายในคลาส

จากตัวอย่างโปรแกรมในรูปที่ 2.3 ค่าระดับของการขาดความสัมพันธ์ภายในคลาส A สามารถหาค่าตามสูตรได้ดังนี้

$$LCOM(A) = \frac{\left\{ \frac{1}{2} (2 + 1) \right\} - 3}{(1 - 3)} = 0.75 ; \text{ โดยที่ } a = 2, m = 3, M(A_1) = 2, M(A_2) = 1$$

2.2.7 ค่าวัชไซโคลเมติกของแมคเคบ (Cyclomatic complexity – V(G))

การวัดวิธีนี้ได้ถูกแนะนำขึ้นมาในช่วงคริสต์ศตวรรษที่ 1970 เพื่อวัดความซับซ้อนของการใช้คำสั่งควบคุม ได้แก่ คำสั่ง if คำสั่ง while และคำสั่ง for การวัดนี้จะแสดงจำนวนทางผ่านของโปรแกรมต้นฉบับ ที่อาจจะถูกไหลผ่าน โปรแกรมที่มีความซับซ้อนมาก ๆ (มีการตรวจสอบเงื่อนไขมาก) จะทำให้ยากต่อการทำความเข้าใจและบำรุงรักษาซอฟต์แวร์ โดยทั่ว ๆ ไปแล้วยิ่งโปรแกรมมีความซับซ้อนมากเท่าใดก็ยิ่งเป็นการยากที่จะทำการตรวจสอบโปรแกรมด้วยและนี่ก็อาจส่งผลไปถึงความน่าเชื่อถือ (Reliability) ของโปรแกรม ค่าวัชไซโคลเมติกของแมคเคบกำหนดว่าแต่ละโมดูลไม่ควรมีค่าเกิน 10 [9]

การหาค่าวัชโรลเมติกของแมคเคบสามารถแสดงให้เห็นดังตัวอย่างต่อไปนี้

	V(G)
class A { private int x;  public void m() { if (x > 0 && x < 3) { switch(x) { case 1: ... case 2: ... } // end switch } // end if } }	1, 2 3 4  <u>4+1</u>

รูปที่ 2.4 ตัวอย่างโปรแกรมเพื่อแสดงการหาค่าวัชโรลเมติกของแมคเคบ

จากตัวอย่างโปรแกรมในรูปที่ 2.4 ค่าวัชโรลเมติกของแมคเคบของเมทอด m() ในคลาส A มีค่าเท่ากับ 5

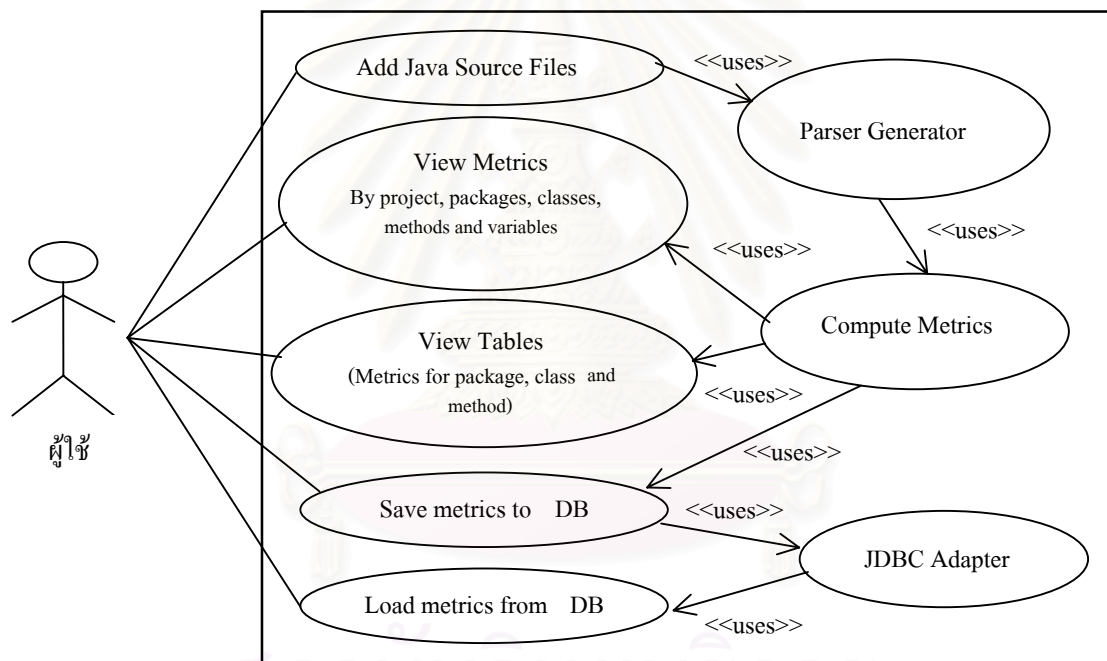
สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

## บทที่ 3

### การวิเคราะห์และออกแบบระบบ

ผู้วิจัยแสดงการวิเคราะห์และออกแบบเครื่องมือ MTOOP โดยใช้ภาษายูเอ็มแอล (Unified Modeling Language – UML) [10] ซึ่งในงานวิจัยนี้ได้นำเสนอภาพรวมของระบบ ได้แก่ โมเดลการใช้งาน (Use Case Modeling) แผนภาพแสดงลำดับการทำงาน (Sequence Diagram) แผนภาพแสดงกิจกรรม (Activity Diagram) แผนภาพของคลาส (Class Modeling) และการคำนวณค่าตัววัดต่าง ๆ ดังรายละเอียดต่อไปนี้

#### 3.1 โมเดลการใช้งาน



รูปที่ 3.1 โมเดลการใช้งาน

ผู้ใช้งานสามารถใช้งานและคิดค่าตัววัดได้ 5 กรณี ดังรูปที่ 3.1 คือผู้ใช้งานสามารถกำหนดโปรแกรมต้นฉบับภาษาจาวาที่ต้องการหาค่าตัววัด ผู้ใช้งานสามารถดูค่าตัววัดแยกตามโปรเจกต์ แพคเกจ คลาส เมททอด ตัวแปร ผู้ใช้งานสามารถดูค่าตัววัดในรูปแบบของตารางสำหรับแพคเกจ คลาสและเมททอด ผู้ใช้งานสามารถเก็บบันทึกข้อมูลค่าตัววัดลงฐานข้อมูลได้ และผู้ใช้งานสามารถดูข้อมูลค่าตัววัดที่มีอยู่ในฐานข้อมูลได้ การที่ผู้ใช้งานสามารถดูค่าตัววัดต่าง ๆ เหล่านี้ได้จะต้องมีการคำนวณค่าตัววัดก่อน การคำนวณค่าตัววัดนี้จะต้องอาศัยการแปลงโปรแกรมต้นฉบับเป็นแผนภูมิต้นไม้ แล้วเก็บข้อมูลคุณสมบัติต่างๆ เช่น ชื่อคลาส (Class name) ชนิดของคลาส (Type of class) ตัวแปรอินสแตนซ์ (Instance variables) ชื่อเมททอด (Method name) ชนิดของเมททอด (Type of method) ชนิดของการส่งค่ากลับของเมททอด (Return type of method) ตัวแปรที่ประกาศในเมททอด (Local variables) เป็นต้น ผู้ใช้งานสามารถดูข้อมูลคุณสมบัติของโปรแกรมต้นฉบับและค่าตัววัดต่างๆ ได้ 2 ทางคือ

### 3.1.1 การดูค่าตัววัด (View Metrics)

ผู้ใช้งานสามารถดูข้อมูลคุณสมบัติของโปรแกรมต้นฉบับและค่าตัววัดต่างๆ โดยแบ่งออกเป็น 5 ประเภทคือ

3.1.1.1 ค่าตัววัดของโปรเจก ได้แก่ จำนวนแพ็คเกจ จำนวนคลาส จำนวนเมทอด จำนวนเมทอดต่อคลาส จำนวนบรรทัด จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนซ์

3.1.1.2 ค่าตัววัดของแพ็คเกจ ได้แก่ จำนวนคลาส จำนวนเมทอด จำนวนเมทอดต่อคลาส จำนวนบรรทัด จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนซ์

3.1.1.3 ค่าตัววัดของคลาส ได้แก่ ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ จำนวนเมทอด จำนวนบรรทัด จำนวนสเตทเมนต์ จำนวนตัวแปรอินสแตนซ์ และค่าของการขาดความสัมพันธ์ภายในคลาส

3.1.1.4 ค่าตัววัดของเมทอด ได้แก่ จำนวนพารามิเตอร์ จำนวนบรรทัด จำนวนสเตทเมนต์ ค่าวัชโรลเมตริกของแมคเคลบ จำนวนตัวแปรเมทอด และขนาดความสัมพันธ์ระหว่างวัตถุ

3.1.1.5 ค่าตัววัดของตัวแปรอินสแตนซ์ ได้แก่ จำนวนครั้งที่ตัวแปรอินสแตนซ์ถูกเรียกใช้ และจำนวนเมทอดที่เรียกใช้ตัวแปรอินสแตนซ์

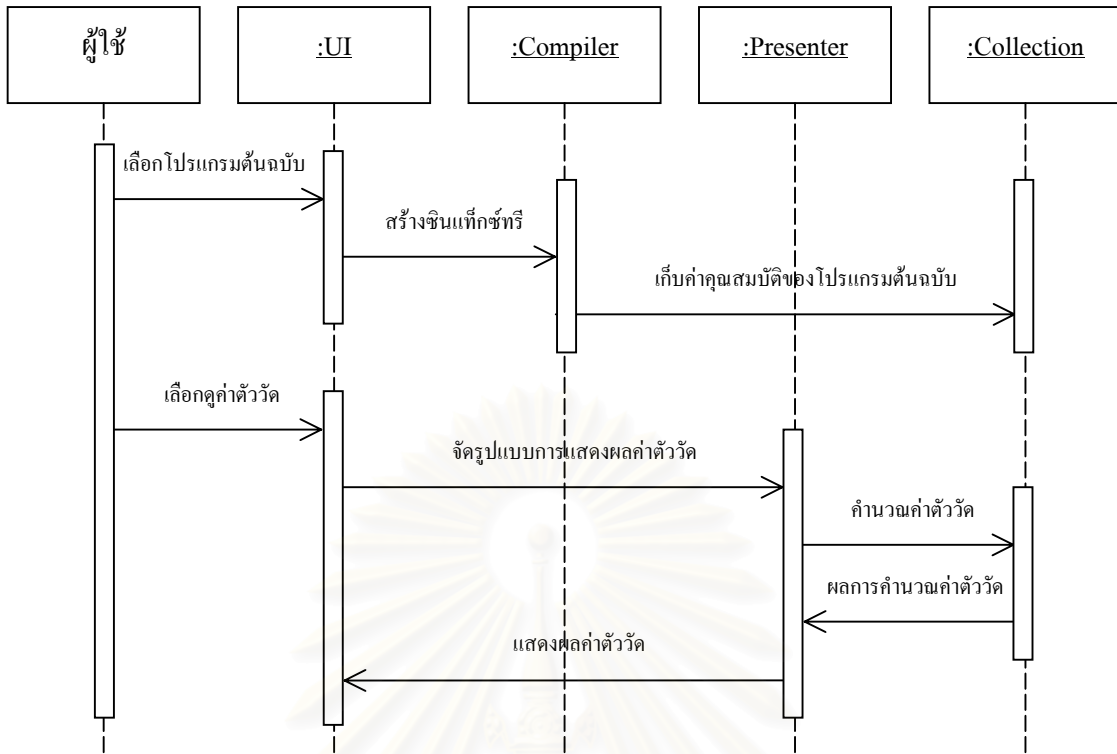
### 3.1.2 การดูค่าตัววัดในรูปแบบตาราง (View Tables)

ผู้ใช้งานสามารถดูข้อมูลคุณสมบัติของโปรแกรมต้นฉบับและค่าตัววัดต่าง ๆ ในรูปแบบของตารางโดยแบ่งออกเป็น 3 ประเภทคือ ค่าตัววัดสำหรับแพ็คเกจของโปรเจก ค่าตัววัดสำหรับคลาสของโปรเจก และค่าตัววัดสำหรับเมทอดของโปรเจก ซึ่งค่าตัววัดจะเหมือนกับในหัวข้อ 3.1.1

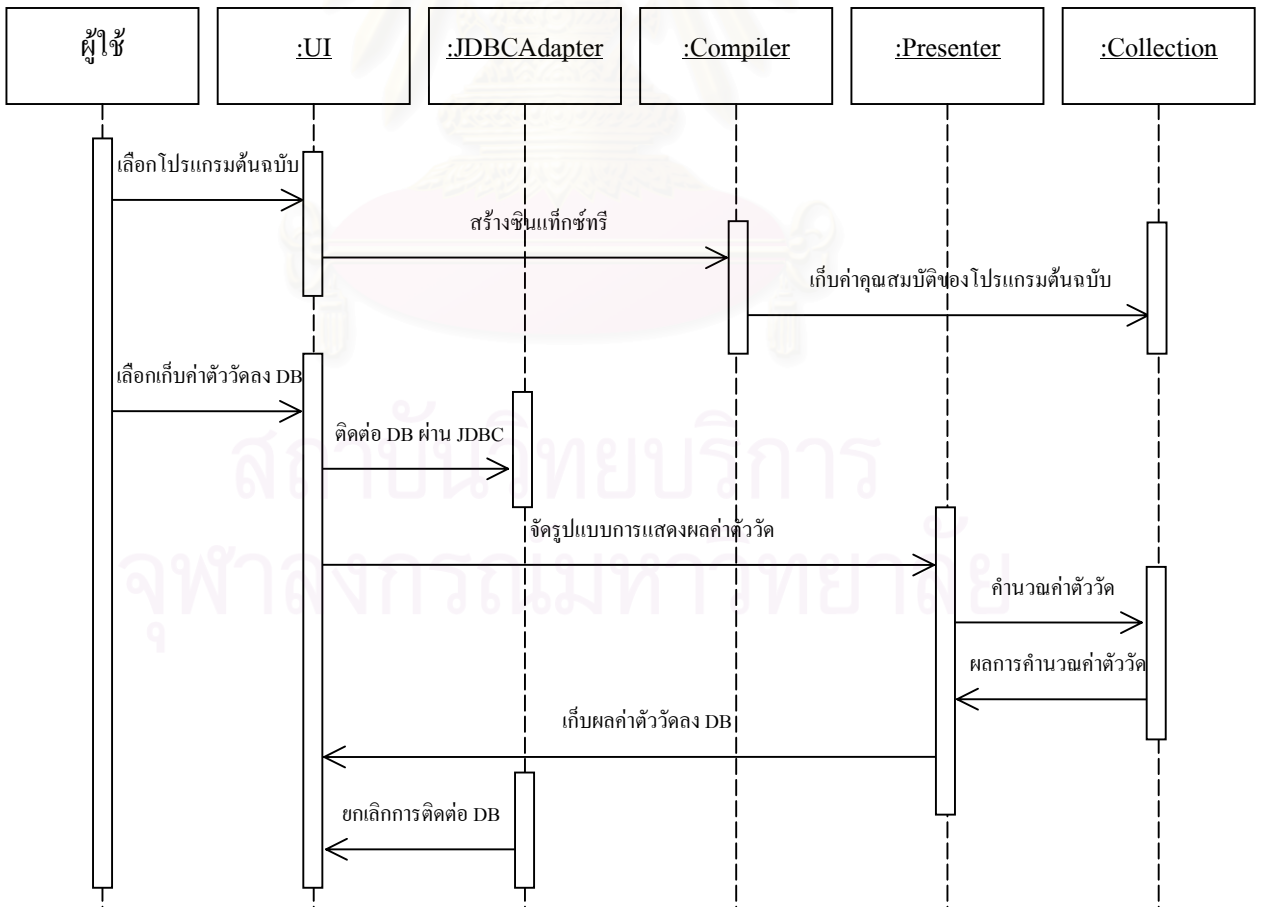
## 3.2 แผนภาพแสดงลำดับการทำงาน

แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้งานต้องการดูค่าตัววัด ดังรูปที่ 3.2 โดยเริ่มจากผู้ใช้งานทำการเลือกโปรแกรมต้นฉบับจากวัตถุ “UI” แล้วนำโปรแกรมต้นฉบับที่ผู้ใช้เลือกมาทำการสร้างซิงแทกซ์ทรีด้วยวัตถุ “Compiler” จากนั้นทำการเก็บค่าคุณสมบัติต่าง ๆ ของโปรแกรมต้นฉบับหลังจากที่ได้ทำการสร้างซิงแทกซ์ทรีด้วยวัตถุ “Collection” จากนั้นผู้ใช้ทำการเลือกดูค่าตัววัด หลังจากที่ได้เลือกโปรแกรมต้นฉบับเข้าสู่ระบบ โดยระบบจะทำการจัดรูปแบบการแสดงผลค่าตัววัดด้วยวัตถุ “Presenter” แล้วทำการส่งคำสั่งร้องขอเพื่อทำการคำนวณค่าตัววัดจากวัตถุ “Collection” แล้วส่งผลของค่าตัววัดกลับมายังวัตถุ “Presenter” และทำการแสดงผลค่าตัววัดด้วยวัตถุ “UI”

แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้งานต้องการเก็บค่าตัววัดลงฐานข้อมูล ดังรูปที่ 3.3 โดยเริ่มจากผู้ใช้งานทำการเลือกโปรแกรมต้นฉบับจากวัตถุ “UI” แล้วนำโปรแกรมต้นฉบับที่ผู้ใช้เลือกมาทำการสร้างซิงแทกซ์ทรีด้วยวัตถุ “Compiler” จากนั้นทำการเก็บค่าคุณสมบัติต่าง ๆ ของโปรแกรมต้นฉบับหลังจากที่ได้ทำการสร้างซิงแทกซ์ทรีด้วยวัตถุ “Collection” จากนั้นผู้ใช้ทำการเลือกเก็บค่าตัววัดลงฐานข้อมูล (DB – Database) หลังจากที่ได้เลือกโปรแกรมต้นฉบับเข้าสู่ระบบแล้ว ระบบจะทำการติดต่อกับฐานข้อมูลโดยการใช้จาวาดาต้าเบสคอนเนกตีวิตี (Java Database Connectivity – JDBC) ด้วยวัตถุ “JDBCAdapter” และจัดรูปแบบการแสดงผลค่าตัววัดด้วยวัตถุ “Presenter” แล้วทำการส่งคำสั่งร้องขอเพื่อทำการคำนวณค่าตัววัดจากวัตถุ “Collection” แล้วส่งผลของค่าตัววัดกลับมายังวัตถุ “Presenter” และทำการเก็บผลค่าตัววัดลงฐานข้อมูลและยกเลิกการติดต่อกับฐานข้อมูลด้วยวัตถุ “UI”



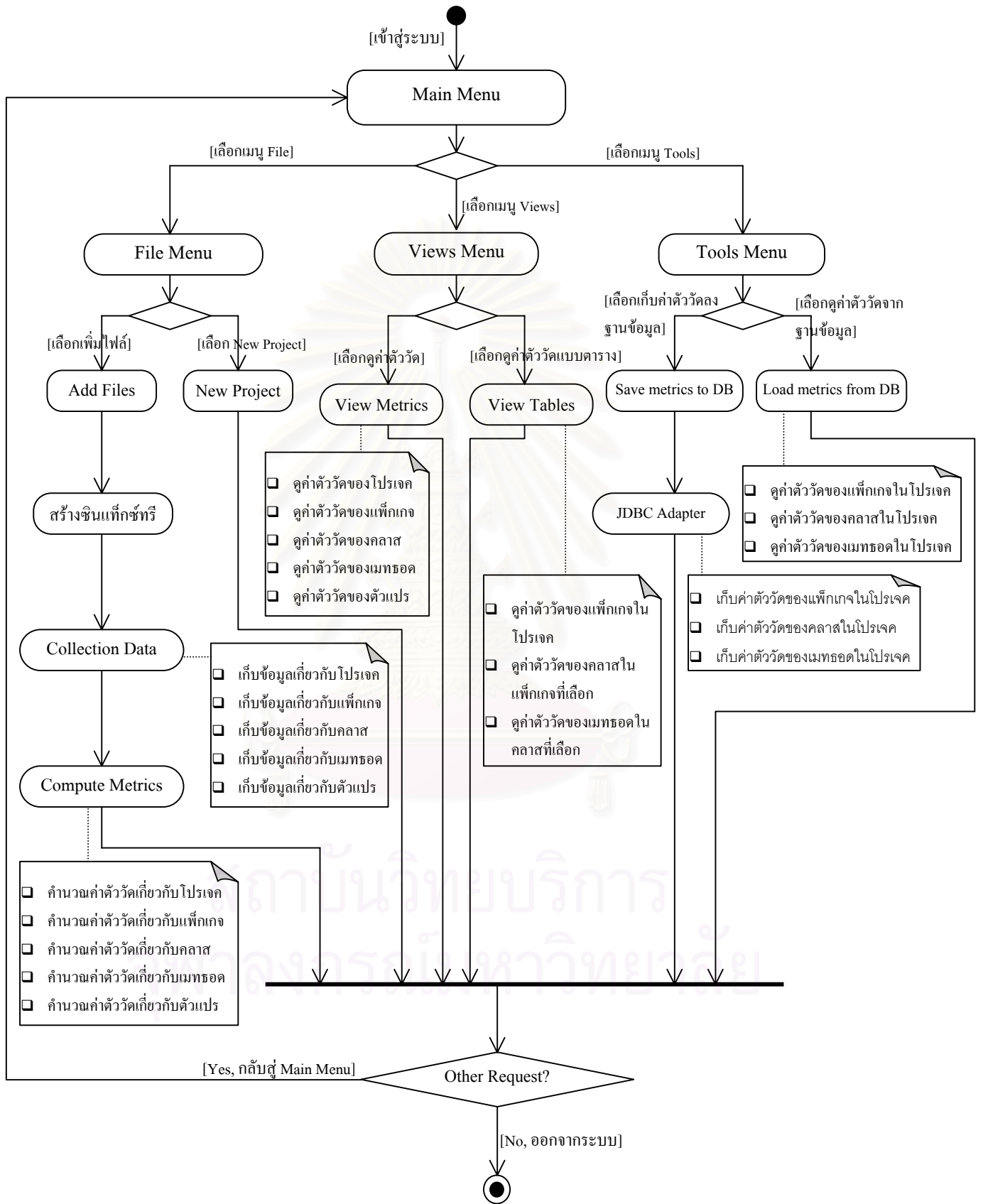
รูปที่ 3.2 แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้งานต้องการดูค่าตัววัด



รูปที่ 3.3 แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้งานต้องการเก็บค่าตัววัดลงฐานข้อมูล



### 3.3 แผนภาพแสดงกิจกรรม

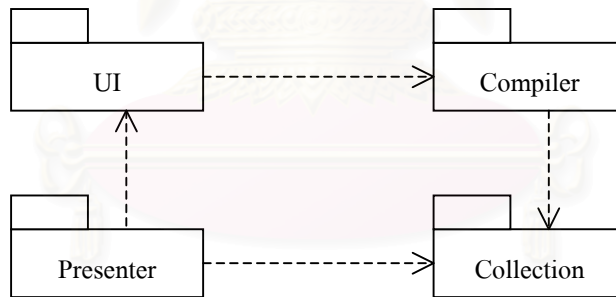


รูปที่ 3.4 แผนภาพแสดงกิจกรรมของเครื่องมือ MTOOP

แผนภาพแสดงกิจกรรมของเครื่องมือ MTOOP ดังแสดงในรูปที่ 3.4 แสดงถึงกิจกรรมการทำงานของระบบคือ เมื่อเข้าสู่ระบบ ผู้ใช้จะต้องเลือกโปรแกรมต้นฉบับเข้าสู่โปรเจก เมื่อเลือกโปรแกรมต้นฉบับแล้วระบบจะทำการสร้างซันแท็กซ์ทรี เพื่อเก็บข้อมูลคุณสมบัติต่างๆ ของโปรแกรมต้นฉบับซึ่งแยกตามโปรเจก แพ็กเกจ คลาส เมธอด และตัวแปรอินสแตนซ์ จากนั้นก็จะทำการคำนวณค่าตัววัดต่างๆ ตามที่ได้แยกประเภทไว้ เมื่อระบบทำการประมวลผลข้อมูลดังกล่าวเสร็จแล้ว ผู้ใช้สามารถที่จะเลือกโปรแกรมต้นฉบับเพิ่มเติมเข้าสู่โปรเจก หรือว่าจะสร้างโปรเจกใหม่ได้ โดยระบบจะทำการลบข้อมูลที่มีอยู่ในระบบออกไปเพื่อให้ผู้ใช้ทำการเลือกโปรแกรมต้นฉบับเข้าไปใหม่ หรือผู้ใช้สามารถที่จะดูข้อมูลค่าตัววัดของโปรเจกได้จากฟังก์ชันดูค่าตัววัดหรือดูค่าตัววัดในรูปแบบตาราง ดังรายละเอียดใน 3.1.1 และ 3.1.2 นอกจากนี้ผู้ใช้สามารถที่จะทำการเก็บข้อมูลค่าตัววัดลงฐานข้อมูล โดยจะเก็บแยกเป็น 3 ประเภทคือ ค่าตัววัดสำหรับแพ็กเกจในโปรเจก ค่าตัววัดสำหรับคลาสในโปรเจก และค่าตัววัดสำหรับเมธอดในโปรเจก ระบบจะทำการติดต่อกับฐานข้อมูลโดยการใช้จาวาดาต้าเบสคอนเน็กทิวตี หลังจากทำการเก็บลงฐานข้อมูลแล้ว ผู้ใช้สามารถดูค่าตัววัดจากฐานข้อมูลที่ทำการเก็บบันทึกไว้ได้

### 3.4 แผนภาพคลาส

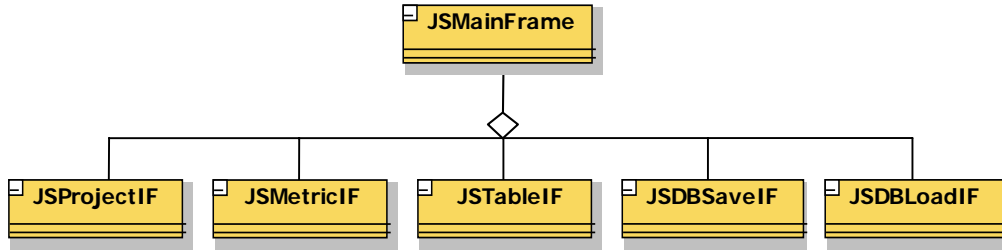
แผนภาพของระบบประกอบด้วยแผนภาพคลาส ซึ่งสามารถแบ่งออกเป็น 4 แพ็กเกจหลักคือแพ็กเกจส่วนติดต่อผู้ใช้ (UI) แพ็กเกจคอมไพเลอร์ (Compiler) แพ็กเกจคำนวณค่าตัววัด (Collection) และ แพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด (Presenter) ซึ่งสามารถแสดงความสัมพันธ์ระหว่างแพ็กเกจต่าง ๆ ของระบบดังรูปที่ 3.5



รูปที่ 3.5 แผนภาพแสดงความสัมพันธ์ระหว่างแพ็กเกจต่าง ๆ ของระบบ

#### 3.4.1 แพ็กเกจส่วนติดต่อผู้ใช้

แพ็กเกจส่วนติดต่อผู้ใช้ เป็นชุดของคลาสที่ทำหน้าที่ติดต่อกับผู้ใช้งานระบบ รูปที่ 3.6 แพ็กเกจส่วนติดต่อผู้ใช้ ได้แก่ คลาสเมนเฟรม (JSMainFrame) เป็นคลาสที่ทำการสร้างเมนูให้ผู้ใช้เลือกและเป็นเฟรมหลักที่ให้เฟรมอื่นๆ แสดงผลในเฟรมหลักนี้ ได้แก่ คลาสการเลือกโปรแกรมต้นฉบับภาษาจาวาเข้าสู่โปรเจก (JSProjectIF) คลาสการดูค่าตัววัด (JSMetricIF) คลาสการดูค่าตัววัดในรูปแบบตาราง (JSTableIF) คลาสการเก็บข้อมูลค่าตัววัดลงฐานข้อมูล (JSDBSaveIF) และคลาการดูค่าตัววัดจากฐานข้อมูล (JSDBLoadIF) ซึ่งคลาสต่างๆ ในแพ็กเกจนี้ สืบทอดคุณสมบัติมาจากแพ็กเกจสวิง (swing)



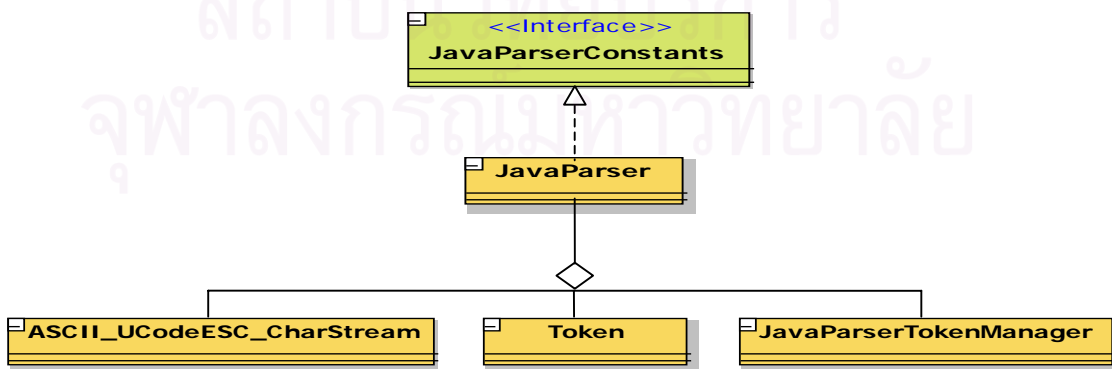
รูปที่ 3.6 แผนภาพคลาสในภาพรวมของแพ็คเกจส่วนติดต่อผู้ใช้

### 3.4.2 แพ็คเกจคอมไพเลอร์

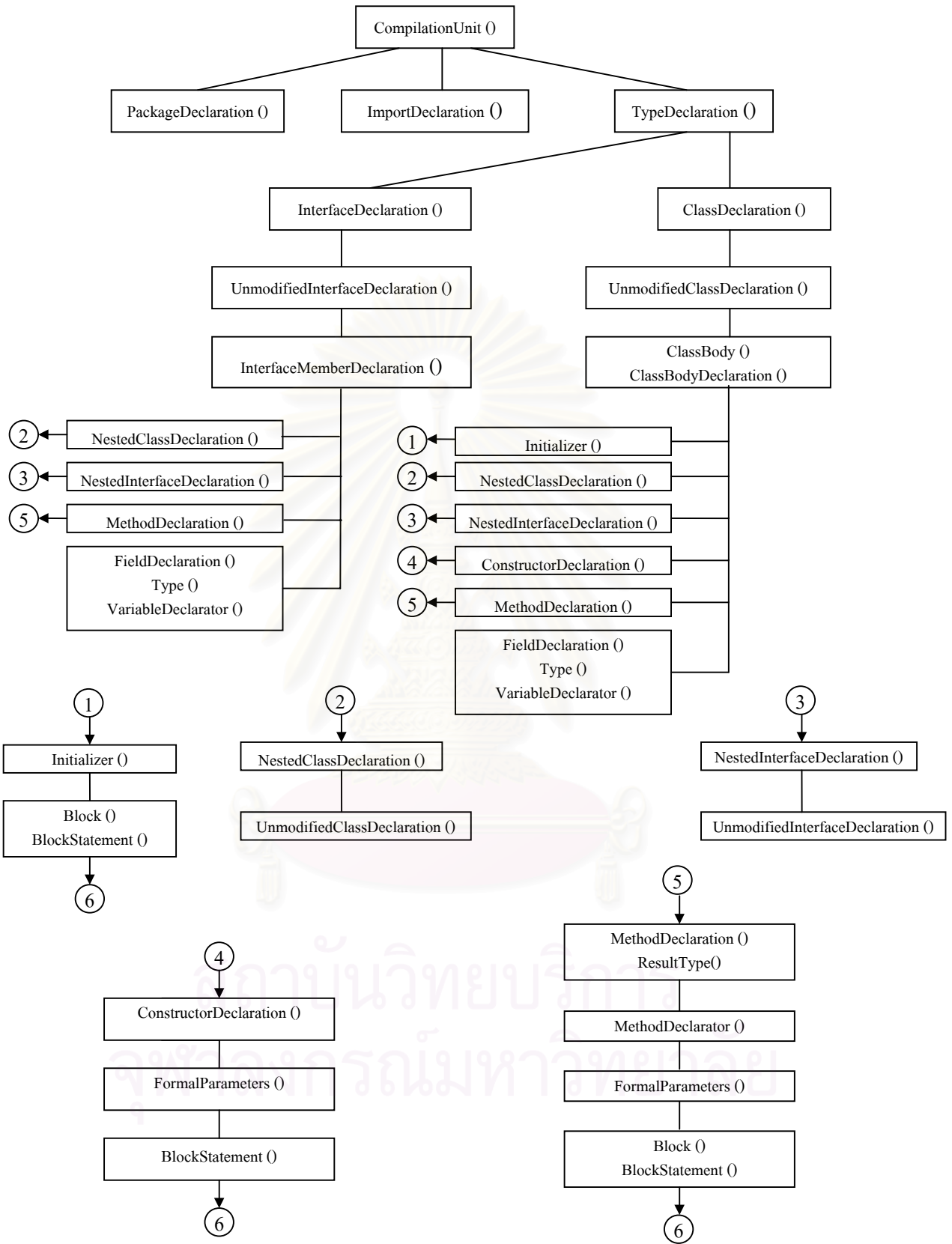
แพ็คเกจคอมไพเลอร์ เป็นชุดของคลาสที่ทำหน้าที่นำโปรแกรมต้นฉบับมาผ่านขบวนการสร้างตัวแปลภาษาเพื่อสร้างชิ้นแท็กซ์ทรี รูปที่ 3.7 แผนภาพคลาสของแพ็คเกจคอมไพเลอร์ ผู้วิจัยได้เลือกเครื่องมือช่วยสร้างตัวแปลภาษาที่ชื่อว่า จาวา คอมไพเลอร์ คอมไพเลอร์ (Java Compiler Compiler – JavaCC) โดยผลที่ได้จากการประมวลผลด้วยเครื่องมือนี้จะได้ชุดของคลาสต่างๆ ดังนี้

คลาสจาวาพาร์เซอร์ (JavaParser) จะเป็นคลาสหลักที่จะนำโปรแกรมต้นฉบับมาสร้างชิ้นแท็กซ์ทรี โดยมีอินเตอร์เฟซคลาสจาวาพาร์เซอร์คอนสแตนท์ส (JavaParserConstants) ที่มีการประกาศค่าคงที่ต่าง ๆ และโทเคนต่าง ๆ (โทเคน คือ กลุ่มอักขระตามไวยากรณ์ของภาษาจาวา) และมีคลาสโทเคน (Token) คลาสแอสกียูโค้ด (ASCII\_UCodeESC\_CharStream) และคลาสจาวาพาร์เซอร์โทเคนแมนเอเจอร์ (JavaParserTokenManager) ที่มีหน้าที่อ่านอักขระของโปรแกรมต้นฉบับและแบ่งให้เป็นโทเคน คลาสจาวาพาร์เซอร์ภายในจะประกอบไปด้วยตัวสร้างโหนดต่าง ๆ โดยมี CompilationUnit() เป็นโหนดราก (root node) โหนดกิ่ง (branch node) และโหนดใบ (leaf node) ดังแสดงในรูปที่ 3.8

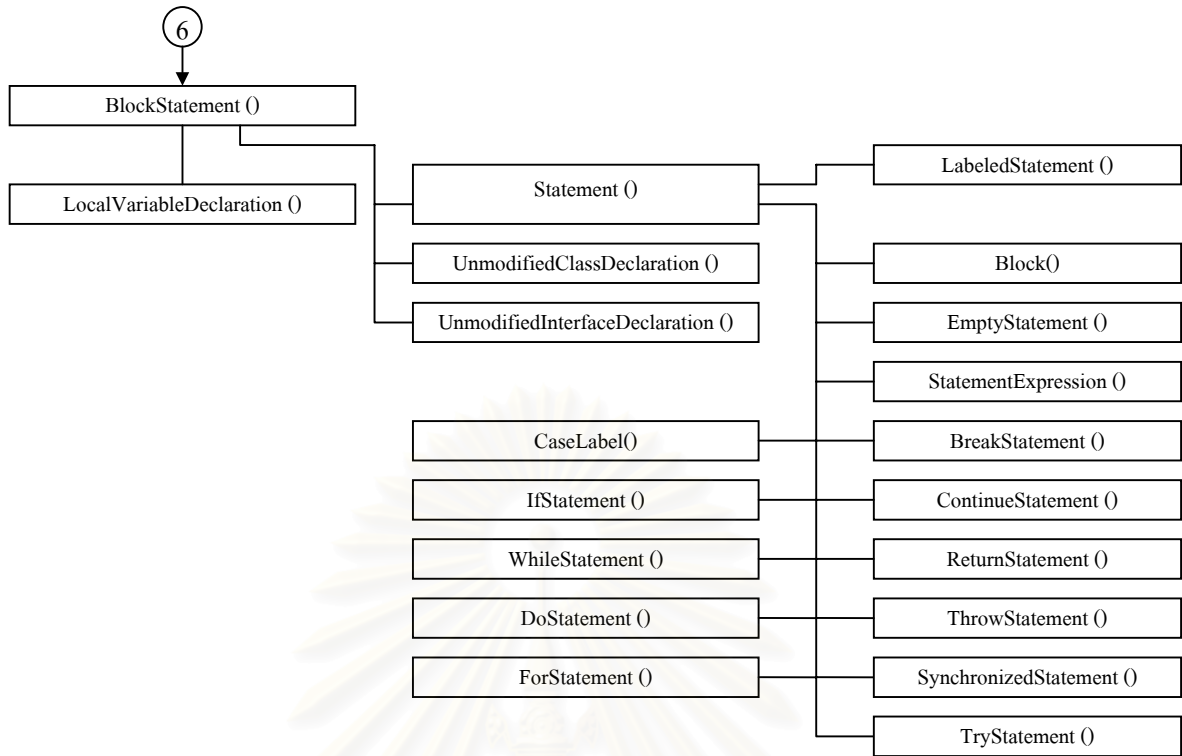
โหนดต่าง ๆ เป็นส่วนสำคัญในการวิเคราะห์ว่าจะสามารถคำนวณค่าตัววัดแต่ละค่าได้จากโหนดใบบ้าง เช่น ค่าวัดของไซโคลเมตริกของแมคเคลบ ก็คือการนับจำนวน โหนดของ CaseLabel() + IfStatement() + WhileStatement() + DoStatement() + ForStatement +  $\sum$ ConditionalOrExpression() +  $\sum$ ConditionalAndExpression() + 1 ภายในเมทรูด เป็นต้น



รูปที่ 3.7 แผนภาพคลาสของแพ็คเกจคอมไพเลอร์



รูปที่ 3.8 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่มาจากการสร้างชินแท็กซ์ทรีของคลาสวาพาร์เซอร์



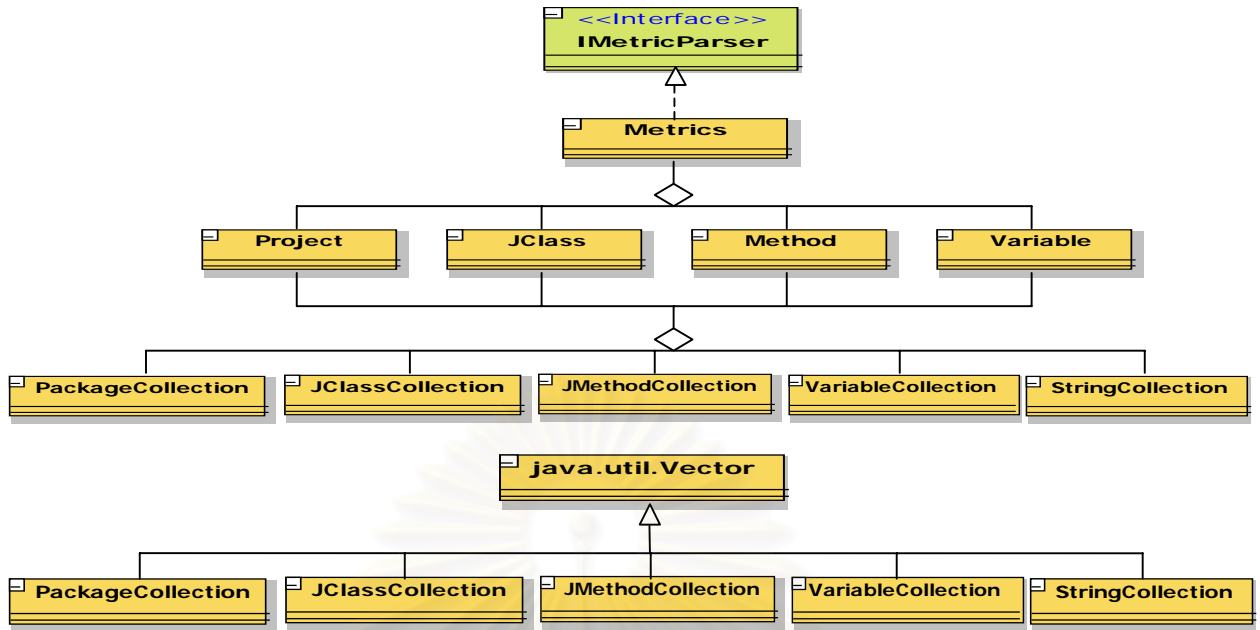
รูปที่ 3.8 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่มาจากการสร้างชั้นแท็กซ์ตรีของคลาสจาวาพาร์เซอร์ (ต่อ)

### 3.4.3 แพ็กเกจคำนวณค่าตัววัด

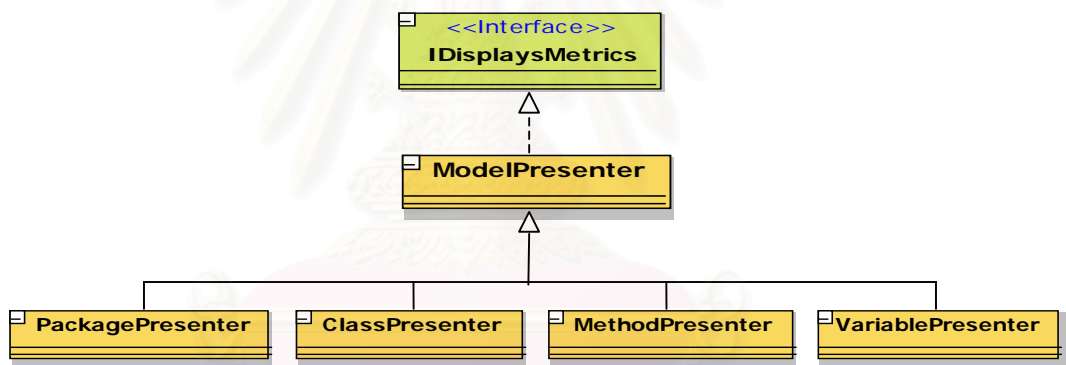
แพ็กเกจคำนวณค่าตัววัด เป็นชุดของคลาสหน่วยรู้จำ โดยมีคลาสตัววัด (Metrics) ทำหน้าที่ท่องไปบนชั้นแท็กซ์ตรี เพื่อเก็บข้อมูลต่าง ๆ จากโหนดต่าง ๆ ตามรูปที่ 3.8 และคำนวณค่าตัววัดต่างๆ ดังที่ได้ยกตัวอย่างค่าตัววัดของไซโคลเมตริกของแมคเคปใน 3.4.2 ซึ่งแพ็กเกจนี้ได้แบ่งออกเป็นหน่วยย่อย ๆ คือ คลาสหน่วยรู้จำเกี่ยวกับโปรเจก (Project) คลาสหน่วยรู้จำเกี่ยวกับคลาส (JClass) คลาสหน่วยรู้จำเกี่ยวกับเมทอด (Method) และคลาสหน่วยรู้จำเกี่ยวกับตัวแปร (Variable) ซึ่งจะถูกรวบรวมอยู่ในรูปแบบของตัวแปรแบบเวกเตอร์ โดยมีคลาสที่สืบทอดคุณสมบัติมาจากคลาสเวกเตอร์ (java.util.Vector) เช่น คลาสแพ็กเกจคอลเลกชัน (PackageCollection) คลาสจาวาคลาสคอลเลกชัน (JClassCollection) คลาสเมทอดคอลเลกชัน (MethodCollection) เป็นต้น รูปที่ 3.9 แสดงให้เห็นถึงความสัมพันธ์ของคลาสต่าง ๆ ในชุดแพ็กเกจนี้

### 3.4.4 แพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด

แพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด เป็นชุดของคลาสที่ทำหน้าที่นำค่าตัววัดจากแพ็กเกจคำนวณค่าตัววัด มาจัดรูปแบบตามค่าตัววัดที่ต้องการ เพื่อส่งต่อให้แพ็กเกจส่วนติดต่อผู้ใช้ นำไปแสดงผลค่าตัววัดตามที่ได้จัดรูปแบบไว้ โดยได้แยกรูปแบบการแสดงผลค่าตัววัดตามประเภทของหน่วยรู้จำ ได้แก่ คลาสแพ็กเกจพรีเซ็นเตอร์ (PackagePresenter) คลาสคลาสพรีเซ็นเตอร์ (ClassPresenter) คลาสเมทอดพรีเซ็นเตอร์ (MethodPresenter) และคลาสวาริเอเบิลพรีเซ็นเตอร์ (VariablePresenter) รูปที่ 3.10 แสดงให้เห็นถึงความสัมพันธ์ของคลาสต่างๆ ในชุดแพ็กเกจนี้



รูปที่ 3.9 แผนภาพคลาสของแพ็คเกจคำนวณค่าตัววัด



รูปที่ 3.10 แผนภาพคลาสของแพ็คเกจจัดรูปแบบการแสดงผลค่าตัววัด

การคำนวณค่าตัววัดต่างๆ แบ่งออกเป็น 5 ประเภทดังนี้

3.4.4.1 ตัววัดค่าสำหรับ โปรเจก (Metrics for project) ได้แก่ จำนวนแพ็คเกจ จำนวนคลาส จำนวนเมธอด จำนวนเมธอดต่อคลาส จำนวนบรรทัด จำนวนสแตทเมนต์ และจำนวนตัวแปรอินสแตนท์ ซึ่งได้แสดงสูตรการคำนวณค่าตัววัดที่มาจากกรนับจำนวนโหนดต่างๆ จากรูปที่ 3.8 ดังต่อไปนี้

1) การนับจำนวนแพ็คเกจของโปรเจก (Number of packages) คือการนับจำนวนโหนดของ PackageDeclaration() โดยที่ชื่อของแพ็คเกจต้องไม่ซ้ำกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนแพ็คเกจของโปรเจก} = \sum \text{PackageDeclaration()}; \text{ชื่อของแพ็คเกจต้องไม่ซ้ำกัน}$$

- 2) การนับจำนวนคลาสของโปรเจก (Number of classes – NOC) คือการนับจำนวนโหนดของ ClassDeclaration() รวมกับจำนวนโหนดของ InterfaceDeclaration() ของทุกแพ็คเกจ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนคลาสของโปรเจก} = \sum \left\{ \sum \text{ClassDeclaration}() + \sum \text{InterfaceDeclaration}() \right\}_{\text{แต่ละ Package}}$$

- 3) การนับจำนวนเมทอดของโปรเจก (Number of methods – NOM) คือการนับจำนวนโหนดของ Initializer() รวมกับจำนวนโหนดของ ConstructorDeclaration() และรวมกับจำนวนโหนดของ MethodDeclaration() ของทุกคลาสและของทุกแพ็คเกจ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนเมทอดของโปรเจก} = \sum \left\{ \sum \left[ \sum \text{Initializer}() + \sum \text{ConstructorDeclaration}() + \sum \text{MethodDeclaration}() \right]_{\text{แต่ละ Class}} \right\}_{\text{แต่ละ Package}}$$

- 4) การหาค่าจำนวนเมทอดต่อคลาสของโปรเจก (Weighted methods per class – WMC) คือจำนวนเมทอดของโปรเจก หารด้วย จำนวนคลาสของโปรเจก สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{ค่าจำนวนเมทอดต่อคลาสของโปรเจก} = \frac{\text{จำนวนเมทอดของโปรเจก}}{\text{จำนวนคลาสของโปรเจก}}$$

- 5) การนับจำนวนบรรทัดของโปรเจก (Lines of code – LOC) คือการนับจำนวนโหนดของ ImportDeclaration() รวมกับจำนวนบรรทัดของคลาส ของทุกแพ็คเกจ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนบรรทัดของโปรเจก} = \sum \left\{ \sum \text{ImportDeclaration}() + \text{Classes LOC} \right\}_{\text{แต่ละ Package}}$$

- 6) การนับจำนวนสเตทเมนต์ของโปรเจก (Number of statements – NOS) คือการนับจำนวนโหนดของ Statement() ของคลาส ของทุกแพ็คเกจ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนสเตทเมนต์ของโปรเจก} = \sum \left\{ \sum \left\{ \sum \text{Statement}() \right\}_{\text{แต่ละ Method}} \right\}_{\text{แต่ละ Class}} \right\}_{\text{แต่ละ Package}}$$

- 7) การนับจำนวนตัวแปรอินสแตนซ์ของโปรเจก (Number of instance variables) คือการนับจำนวน โหนดของ FieldDeclaration() ของคลาส ของทุกแพ็คเกจ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนตัวแปรอินสแตนซ์ของโปรเจก} = \sum \left\{ \sum \text{FieldDeclaration} () \right\}_{\text{แต่ละ Package}}$$

- 3.4.4.2 ตัววัดค่าสำหรับแพ็คเกจ (Metrics for package) ได้แก่ จำนวนคลาส จำนวนเมทอด จำนวน เมทอดต่อคลาส จำนวนบรรทัด จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนซ์ ซึ่งได้แสดง สูตรการคำนวณค่าตัววัดที่มาจาก การนับจำนวน โหนดต่าง ๆ จากรูปที่ 3.8 ดังต่อไปนี้

- 1) การนับจำนวนคลาสของแพ็คเกจ (Number of Classes – NOC) คือการนับจำนวน โหนดของ ClassDeclaration() รวมกับจำนวน โหนดของ InterfaceDeclaration() ของแพ็คเกจที่ใช้ชื่อ แพ็คเกจเดียวกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\begin{aligned} & \text{จำนวนคลาสของแพ็คเกจ} \\ & = \sum \left\{ \sum \text{ClassDeclaration}() + \sum \text{InterfaceDeclaration}() \right\}_{\text{ชื่อ Package เดียวกัน}} \end{aligned}$$

- 2) การนับจำนวนเมทอดของแพ็คเกจ (Number of methods – NOM) คือการนับจำนวน โหนด ของ Initializer() รวมกับจำนวน โหนดของ ConstructorDeclaration() และรวมกับจำนวน โหนดของ MethodDeclaration() ของทุกคลาสและของแพ็คเกจที่ใช้ชื่อแพ็คเกจเดียวกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\begin{aligned} & \text{จำนวนเมทอดของแพ็คเกจ} \\ & = \sum \left\{ \sum \left[ \sum \text{Initializer}() + \sum \text{ConstructorDeclaration}() + \sum \text{MethodDeclaration}() \right]_{\text{แต่ละ Class}} \right\}_{\text{ชื่อ Package เดียวกัน}} \end{aligned}$$

- 3) การหาค่าจำนวนเมทอดต่อคลาสของแพ็คเกจ (Weighted methods per class – WMC) คือ จำนวนเมทอดของแพ็คเกจ หารด้วย จำนวนคลาสของแพ็คเกจ สามารถแสดงสูตรคำนวณได้ ดังต่อไปนี้

$$\text{ค่าจำนวนเมทอดต่อคลาสของแพ็คเกจ} = \frac{\text{จำนวนเมทอดของแพ็คเกจ}}{\text{จำนวนคลาสของแพ็คเกจ}}$$

- 4) การนับจำนวนบรรทัดของแพ็คเกจ (Lines of code – LOC) คือการนับจำนวนบรรทัดของคลาส ของแพ็คเกจที่ใช้ชื่อแพ็คเกจเดียวกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนบรรทัดของแพ็คเกจ} = \sum \left\{ \text{Classes LOC} \right\}_{\text{ชื่อ Package เดียวกัน}}$$



- 5) การนับจำนวนสเตทเมนต์ของแพ็คเกจ (Number of statements – NOS) คือการนับจำนวน โหนดของ Statement() ของคลาส ของแพ็คเกจที่ใช้ชื่อแพ็คเกจเดียวกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนสเตทเมนต์ของแพ็คเกจ} = \sum \left\{ \sum \left\{ \sum \text{Statement}() \right\}_{\text{แต่ละ Method}} \right\}_{\text{แต่ละ Class}} \left\{ \text{ชื่อ Package เดียวกัน} \right\}$$

- 6) การนับจำนวนตัวแปรอินสแตนท์ของแพ็คเกจ (Number of instance variables) คือการนับจำนวน โหนดของ FieldDeclaration() ของคลาส ของแพ็คเกจที่ใช้ชื่อแพ็คเกจเดียวกัน สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนตัวแปรอินสแตนท์ของแพ็คเกจ} = \sum \left\{ \sum \text{FieldDeclaration}() \right\}_{\text{ชื่อ Package เดียวกัน}}$$

3.4.4.3 ตัววัดค่าสำหรับคลาส (Metrics for class) ได้แก่ ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ จำนวนเมทอด จำนวนบรรทัด จำนวนสเตทเมนต์ จำนวนตัวแปรอินสแตนท์ และค่าของการขาดความสัมพันธ์ภายในคลาส ซึ่งได้แสดงสูตรการคำนวณค่าตัววัดที่มาจาก การนับจำนวน โหนดต่าง ๆ จากรูปที่ 3.8 ดังต่อไปนี้

- 1) การหาค่าระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ (Depth of the inheritance tree – DIT) คือการนับจำนวนของการสืบทอดคุณสมบัติ (Inheritance) ของคลาส โดยการตรวจสอบโทเคน "Extends" ใน โหนดของ ClassDeclaration() และ InterfaceDeclaration()
- 2) การนับจำนวนเมทอดของคลาส (Number of methods – NOM) คือการนับจำนวน โหนดของ Initializer() รวมกับจำนวน โหนดของ ConstructorDeclaration() และรวมกับจำนวน โหนดของ MethodDeclaration() ของคลาส สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\begin{aligned} & \text{จำนวนเมทอดของคลาส} \\ & = \sum \text{Initializer}() + \sum \text{ConstructorDeclaration}() + \sum \text{MethodDeclaration}() \end{aligned}$$

- 3) การนับจำนวนบรรทัดของคลาส (Lines of code – Classes LOC) คือการนับจำนวน โหนดของ ClassDeclaration() รวมกับจำนวน โหนดของ FieldDeclaration() และรวมกับจำนวนบรรทัดของเมทอด ของคลาสนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\begin{aligned} & \text{จำนวนบรรทัดของคลาส} \\ & = \text{ClassDeclaration}() + \sum \text{NestedClassDeclaration}() + \sum \text{FieldDeclaration}() + \text{Methods LOC} \end{aligned}$$

- 4) การนับจำนวนสเตทเมนต์ของคลาส (Number of Statements – NOS) คือการนับจำนวนโหนดของ Statement() ของเมทอด ของคลาสนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนสเตทเมนต์ของคลาส} = \sum \left\{ \sum \text{Statement() }_{\text{แต่ละ Method}} \right\}_{\text{ชื่อคลาส เดียวกัน}}$$

- 5) การนับจำนวนตัวแปรอินสแตนท์ของคลาส (Number of Instance Variables) คือการนับจำนวนโหนดของ FieldDeclaration() ของคลาสนั้น ๆ ดังจะแสดงสูตรต่อไปนี้

$$\text{จำนวนตัวแปรอินสแตนท์ของคลาส} = \sum \text{FieldDeclaration ()}$$

- 6) การหาค่าระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of cohesion of methods – LCOM) การหาค่า LCOM จะใช้สูตรการคำนวณของ Brian Henderson-Sellers ดังแสดงสูตรในหัวข้อ 2.2.6

3.4.4.4 ตัววัดค่าสำหรับเมทอด (Metrics for method) ได้แก่ จำนวนพารามิเตอร์ จำนวนบรรทัด จำนวนสเตทเมนต์ ค่าวัดไซโคลเมตริกของแมกเคบ จำนวนตัวแปรเมทอด และขนาดความสัมพันธ์ระหว่างวัตถุ ซึ่งได้แสดงสูตรการคำนวณค่าตัววัดที่มาจากกรนับจำนวนโหนดต่าง ๆ จากรูปที่ 3.8 ดังต่อไปนี้

- 1) การนับจำนวนพารามิเตอร์ (Number of parameters) คือการนับจำนวนโหนดของ FormalParameters() ของเมทอดนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนพารามิเตอร์} = \sum \text{FormalParameters ()}$$

- 2) การนับจำนวนบรรทัดของเมทอด (Lines of code – Methods LOC) คือการนับจำนวนโหนดของ MethodDeclaration() รวมกับจำนวนโหนดของ LocalVariableDeclaration() และรวมกับจำนวนโหนดของ Statement() ของเมทอดนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\begin{aligned} &\text{จำนวนบรรทัดของเมทอด} \\ &= \text{MethodDeclaration() } + \sum \text{LocalVariableDeclaration() } + \sum \text{Statement() } \end{aligned}$$

การนับจำนวนบรรทัดจะไม่นับรวมบรรทัดว่าง (EmptyStatement()) บรรทัดที่เป็นเครื่องหมายบล็อก (Block()) และบรรทัดที่เป็นคอมเมนต์ (Comment Line) การประกาศตัวแปรหลายตัวอยู่บนบรรทัดเดียวกัน จะนับจำนวนบรรทัดเท่ากับจำนวนตัวแปรที่ประกาศ เช่น `int x, y;` จะได้  $\text{LOC} = 2$

- 3) การนับจำนวนสเตทเมนต์ของเมทอด (Number of statements – Methods NOS) คือการนับจำนวนโหนดของ Statement() ของเมทอดนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนสเตทเมนต์ของเมทอด} = \sum \text{Statement()}$$

- 4) การหาค่าวัดของไซโคลเมตริกของแมคเคเบ – V(G) คือการนับจำนวนโหนดของเงื่อนไขต่าง ๆ ของเมทอดนั้น ๆ บวกด้วย 1 สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

ค่าวัดของไซโคลเมตริกของแมคเคเบ

$$= \sum \text{CaseLabel ()} + \sum \text{IfStatement()} + \sum \text{WhileStatement()} + \sum \text{DoStatement()} \\ + \sum \text{ForStatement()} + \sum \text{ConditionalOrExpression()} + \sum \text{ConditionalAndExpression()} + 1$$

- 5) การนับจำนวนตัวแปรของเมทอด (Number of local variables) คือการนับจำนวนโหนดของ LocalVariableDeclaration() ของเมทอดนั้น ๆ สามารถแสดงสูตรคำนวณได้ดังต่อไปนี้

$$\text{จำนวนตัวแปรของเมทอด} = \sum \text{LocalVariableDeclaration()}$$

- 6) การหาค่าขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling between objects – CBO) คือการนับจำนวนของการส่งค่ากลับ (Return type) ของเมทอดหรือตัวแปรที่เรียกใช้ที่ไม่ใช่ประเภทเดียวกันกับประเภทของคลาสนั้น ๆ ของแต่ละเมทอด

3.4.4.5 ตัววัดค่าสำหรับตัวแปรอินสแตนซ์ (Metrics for variable) ได้แก่ จำนวนครั้งที่ตัวแปรอินสแตนซ์ถูกเรียกใช้ และจำนวนเมทอดที่เรียกใช้ตัวแปรอินสแตนซ์ ซึ่งได้แสดงสูตรการคำนวณค่าตัววัดที่มาจากกรนับจำนวนโหนดต่าง ๆ จากรูปที่ 3.8 ดังต่อไปนี้

- 1) การนับจำนวนตัวแปรอินสแตนซ์ที่ถูกเรียกใช้ (Number of instance variables uses) คือการนับจำนวนครั้งที่ตัวแปรอินสแตนซ์ที่ถูกเรียกใช้ภายในคลาสนั้น ๆ
- 2) การนับจำนวนเมทอดที่เรียกใช้ตัวแปรอินสแตนซ์ (Number of instance variables used) คือการนับจำนวนเมทอดที่มีการเรียกใช้ตัวแปรอินสแตนซ์

## บทที่ 4

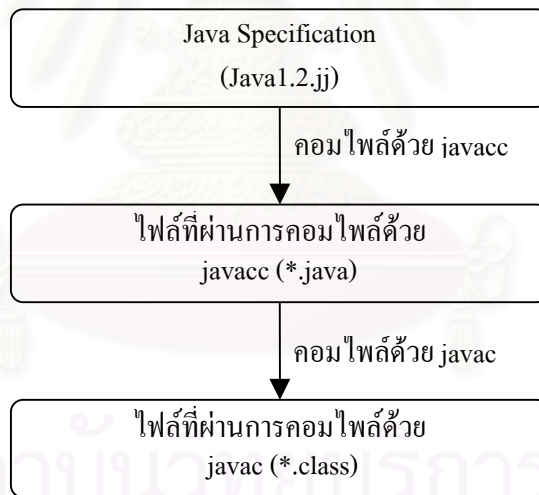
### การพัฒนาระบบ

เครื่องมือ MTOOP นี้พัฒนาขึ้นด้วยโปรแกรมภาษาจาวาบนระบบปฏิบัติการวินโดวส์ โดยใช้คลาสไลบรารีของจาวาดีเวลอปเม้นต์ทูลคิต (Java Development Toolkit: JDK) เวอร์ชัน 1.2 ซึ่งจะมีคลาสสวิง ที่ช่วยในการสร้างส่วนติดต่อกับผู้ใช้ (GUI)

ในบทนี้เป็นการอธิบายรายละเอียดการทำงานของคลาสต่างๆ ซึ่งจะอธิบายรายละเอียดเกี่ยวกับการสร้างแพ็คเกจคอมไพเลอร์ การสร้างแพ็คเกจคำนวณค่าตัววัด การเก็บค่าคุณสมบัติและคำนวณค่าตัววัด ดังจะแสดงรายละเอียดการพัฒนาต่อไปนี้

#### 4.1 การสร้างแพ็คเกจคอมไพเลอร์

การสร้างแพ็คเกจคอมไพเลอร์ มีขั้นตอน ดังนี้



รูปที่ 4.1 แผนภาพแสดงขั้นตอนเพื่อสร้างตัวสร้างซินแท็กซ์ทรี

จากรูปที่ 4.1 คือแผนภาพแสดงขั้นตอน (Activity Diagram) เพื่อสร้างตัวสร้างซินแท็กซ์ทรี โดยเริ่มจากไฟล์ข้อกำหนดของภาษาจาวา (Java Specification) จะมีรายละเอียดเกี่ยวกับข้อกำหนดโทเคน (lexical specifications) และข้อกำหนดไวยากรณ์ (grammar specifications) ซึ่งเขียนอยู่ในรูปแบบของบีเอ็นเอฟ (BNF : Backus Naur Form) ซึ่งไฟล์ข้อกำหนดของภาษาจาวานี้มีอยู่ในชุดของเครื่องมือ จาวาซีซี (JavaCC) ซึ่งเป็นเวอร์ชัน 1.2 (Java1.2.jj) จากนั้นนำไฟล์ข้อกำหนดของภาษาจาวา มาประมวลผลด้วยจาวาซีซี โดยจาวาซีซีจะประมวลผลโทเคนและไวยากรณ์ แล้วสร้างเป็นไฟล์โปรแกรมภาษาจาวาจำนวนหนึ่ง ได้แก่ ไฟล์จาวาพาร์เซอร์ (JavaParser.java) ไฟล์จาวาพาร์เซอร์คอนสแตนท์ (JavaParserConstants.java) ไฟล์โทเคน (Token.java) ไฟล์จาวาพาร์เซอร์โทเคนเมนเนเจอร์

(JavaParserTokenManager.java) และไฟล์แอสกียูโค้ด (ASCII\_UCodeESC\_CharStream.java) จากนั้นนำไฟล์โปรแกรมภาษาจาวาจำนวนนี้มาประมวลผลด้วยจาวาซี (javac) ซึ่งเป็นคอมไพเลอร์ของ เจดีเค (JDK) แต่ก่อนที่จะทำการคอมไพล์ จะต้องมีการแก้ไขปรับปรุง เพิ่มเติมไฟล์จาวาพาร์เซอร์ เพื่อเก็บค่าคุณสมบัติตามโหมดต่างๆ และให้แพ็คเกจคำนวณค่าตัววัด สามารถนำมาคำนวณค่าตัววัดต่างๆ ได้ ซึ่งจะอธิบายรวมไปกับการสร้างแพ็คเกจคำนวณค่าตัววัด ในหัวข้อ 4.2 ต่อไป

## 4.2 การสร้างแพ็คเกจคำนวณค่าตัววัด

ในหัวข้อนี้จะอธิบายถึงการแก้ไขปรับปรุง เพิ่มเติมไฟล์ JavaParser.java และการคำนวณค่าตัววัดดังกล่าวละเอียดต่อไป

```
// Class JavaParser
private static IMetricParser metrics;

public JavaParser(java.io.InputStream fileStream, IMetricParser m) {
    this(fileStream);
    metrics = m;           // เพิ่มตัวแปร metrics
    parseFile(fileStream);
}

public JavaParser(java.io.InputStream stream) {
    ...
}

public static void parseFile(java.io.InputStream fileStream) {
    try {
        ReInit(fileStream);
        CompilationUnit();
    }
    catch(Exception e) {
        System.out.println(e.getMessage());
        System.out.println(e.toString());
        System.out.println("Encountered errors during file parsing.");
    }
}
```

รูปที่ 4.2 การรับโปรแกรมต้นฉบับจากส่วนติดต่อผู้ใช้ และเริ่มสร้างชิ้นแท็กซ์ตรี

รูปที่ 4.2 การเพิ่มตัวแปรของคลาสชื่อ metrics ที่มีคุณสมบัติของอินเตอร์เฟซคลาส IMetricParser และได้เพิ่มคอนสตรัคเตอร์ JavaParser(java.io.InputStream fileStream, IMetricParser m) ที่มีพารามิเตอร์ 2 ตัวคือพารามิเตอร์สำหรับรับไฟล์สตรีมมาจากโปรแกรมต้นฉบับและพารามิเตอร์สำหรับเก็บค่าคุณสมบัติต่างๆ เพื่อคำนวณค่าตัววัด จากนั้นคอนสตรัคเตอร์จะไปเรียกโอเพอร์เรชัน parseFile(java.io.InputStream fileStream) ขึ้นมาทำงานเพื่อเริ่มสร้างโหนดราก CompilationUnit() โหนด CompilationUnit() จะทำการสร้างโหนดกิ่งและ

โหนดใบต่างๆ โดยเริ่มที่การตรวจสอบว่ามีการประกาศชื่อแพ็คเกจ (Package) ก็จะไปเรียกโอเปอร์เรชัน PackageDeclaration() ขึ้นมาทำงาน การประกาศอิมพอร์ต (Import) ก็จะไปเรียกโอเปอร์เรชัน ImportDeclaration() ขึ้นมาทำงาน หรือเป็นการประกาศประเภทอื่นๆ เช่น การประกาศคลาส การประกาศอินเตอร์เฟซคลาส ก็จะไปเรียกโอเปอร์เรชัน TypeDeclaration() ขึ้นมาทำงาน

รูปที่ 4.3 แสดงให้เห็นความแตกต่างระหว่างก่อนแก้ไข PackageDeclaration() ในรูปทางซ้ายและหลังแก้ไข PackageDeclaration() ในรูปทางขวา ซึ่งจะเห็นว่ามีการเก็บชื่อของแพ็คเกจ packageName = getName() โดยได้เพิ่มโอเปอร์เรชัน getName() ขึ้นมาเพื่อเก็บชื่อของโหนดที่เราต้องการ ในที่นี้คือชื่อของแพ็คเกจ จากนั้นได้ส่งชื่อแพ็คเกจที่ได้ไปเก็บค่าในตัวแปรเวกเตอร์ โดยเรียกโอเปอร์เรชัน metrics.addPackage(packageName) ซึ่งมีวิธีการเพิ่มค่าใน packages โดยมีการตรวจสอบชื่อของแพ็คเกจก่อนว่าซ้ำกันหรือไม่ ดังรูปที่ 4.4

<pre>static final public void PackageDeclaration() throws ParseException {     jj_consume_token(PACKAGE);     Name();     jj_consume_token(SEMICOLON); }</pre>	<pre>static final public void PackageDeclaration() throws ParseException {     jj_consume_token(PACKAGE);     String packageName = getName();     jj_consume_token(SEMICOLON);     metrics.addPackage(packageName); }</pre>
<pre>static final public void Name() throws ParseException {     jj_consume_token(IDENTIFIER);     label_19:     while (true) {         if (jj_2_14(2)) {             ;         }         else {             break label_19;         }         jj_consume_token(DOT);         jj_consume_token(IDENTIFIER);     } }</pre>	<pre>static final public String getName() throws ParseException {     Token t = jj_consume_token(IDENTIFIER);     String strName = t.image;     label_19:     while (true) {         if (jj_2_14(2)) {             ;         }         else {             break label_19;         }         jj_consume_token(DOT);         t = jj_consume_token(IDENTIFIER);         strName = strName + "." + t.image;     }     return strName; }</pre>

รูปที่ 4.3 ความแตกต่างระหว่างก่อนแก้ไขและหลังแก้ไข PackageDeclaration()

```
private StringCollection packages;
private String currentPackage;

public void addPackage(String name) {
    packages.addUnique(name);
    currentPackage = name;
}
```

รูปที่ 4.4 การเก็บค่า packages ในตัวแปรเวกเตอร์

```

static final public void ImportDeclaration() throws
ParseException {
    jj_consume_token(IMPORT);
    Name();
    switch ((jj_ntk===-1)?jj_ntk():jj_ntk) {
    case DOT:
        jj_consume_token(DOT);
        jj_consume_token(STAR);
        break;
    default:
        jj_la1[3] = jj_gen;
        ;
    }
    jj_consume_token(SEMICOLON);
}

```

```

static final public void ImportDeclaration() throws
ParseException {
    jj_consume_token(IMPORT);
    String importName = getName();
    switch ((jj_ntk===-1)?jj_ntk():jj_ntk) {
    case DOT:
        jj_consume_token(DOT);
        jj_consume_token(STAR);
        importName = importName + ".*";
        break;
    default:
        jj_la1[3] = jj_gen;
        ;
    }
    jj_consume_token(SEMICOLON);
    metrics.addImportStatement(importName);
}

```

รูปที่ 4.5 ความแตกต่างระหว่างก่อนแก้ไขและหลังแก้ไข ImportDeclaration()

จากรูปที่ 4.5 แสดงให้เห็นความแตกต่างระหว่างก่อนแก้ไข ImportDeclaration() รูปทางซ้ายและหลังแก้ไข ImportDeclaration() รูปทางขวา ซึ่งจะเห็นว่าการเก็บชื่อของการอิมพอร์ต importName = getName() จากนั้นได้ส่งชื่ออิมพอร์ตที่ได้ไปเก็บค่าในตัวแปรเวคเตอร์ โดยเรียกโอเปอเรชัน metrics.addImportStatement(importName) ซึ่งมีวิธีการเพิ่มค่าใน importCollection และแสดงให้เห็นวิธีการนับจำนวนโหนดของอิมพอร์ต getImportCount() ดังรูปที่ 4.6

```

private StringCollection importCollection;

public void addImportStatement(String name) {
    importCollection.addElement(name);
}

public int getImportCount() {
    return importCollection.size();
}

```

รูปที่ 4.6 การเก็บค่า importCollection ในตัวแปรเวคเตอร์

รายละเอียดการทำงานของคลาสต่างๆ ในแพ็คเกจคำนวณค่าตัววัดได้แก่ คลาสตัววัด (Metrics) คลาสหน่วยรู่จำเกี่ยวกับโปรเจก (Project) คลาสหน่วยรู่จำเกี่ยวกับคลาส (JClass) คลาสหน่วยรู่จำเกี่ยวกับเมทอด (Method) คลาสหน่วยรู่จำเกี่ยวกับตัวแปร (Variable) สามารถอธิบายได้ดังนี้

#### 4.2.1 คลาสตัววัด (Metrics)

รายละเอียดของคลาส "Metrics" แสดงในรูปที่ 4.7 เป็นคลาสที่ทำหน้าที่ท่องเที่ยวบนจีนเท็กซ์ทรี เพื่อเก็บข้อมูลต่าง ๆ จากโหนดต่าง ๆ ภายในคลาสประกอบด้วยการประกาศตัวแปรอินสแตนซ์ (Data Member) ที่ใช้เก็บข้อมูลและมีเมทอดต่าง ๆ ที่ใช้จัดการกับข้อมูลเหล่านั้นเช่น การจัดเก็บชื่อของแพ็คเกจ (addPackage) การจัดเก็บชื่อของคลาส (setClassName) และคุณสมบัติต่าง ๆ ของคลาส เป็นต้น จากรูปที่ 4.7 จะเห็นว่ามีการประกาศตัวแปรอินสแตนซ์ที่เป็นตัวแปรอ้างอิง (Reference Variable) ชื่อ currentClass ที่เป็นวัตถุในคลาส "JClass" เพื่อให้สามารถเข้าใช้ข้อมูลและเมทอดในคลาส "JClass" ได้ เพื่อทำการจัดเก็บข้อมูลและคำนวณค่าตัววัดซึ่งแยกตามประเภทต่าง ๆ ต่อไป

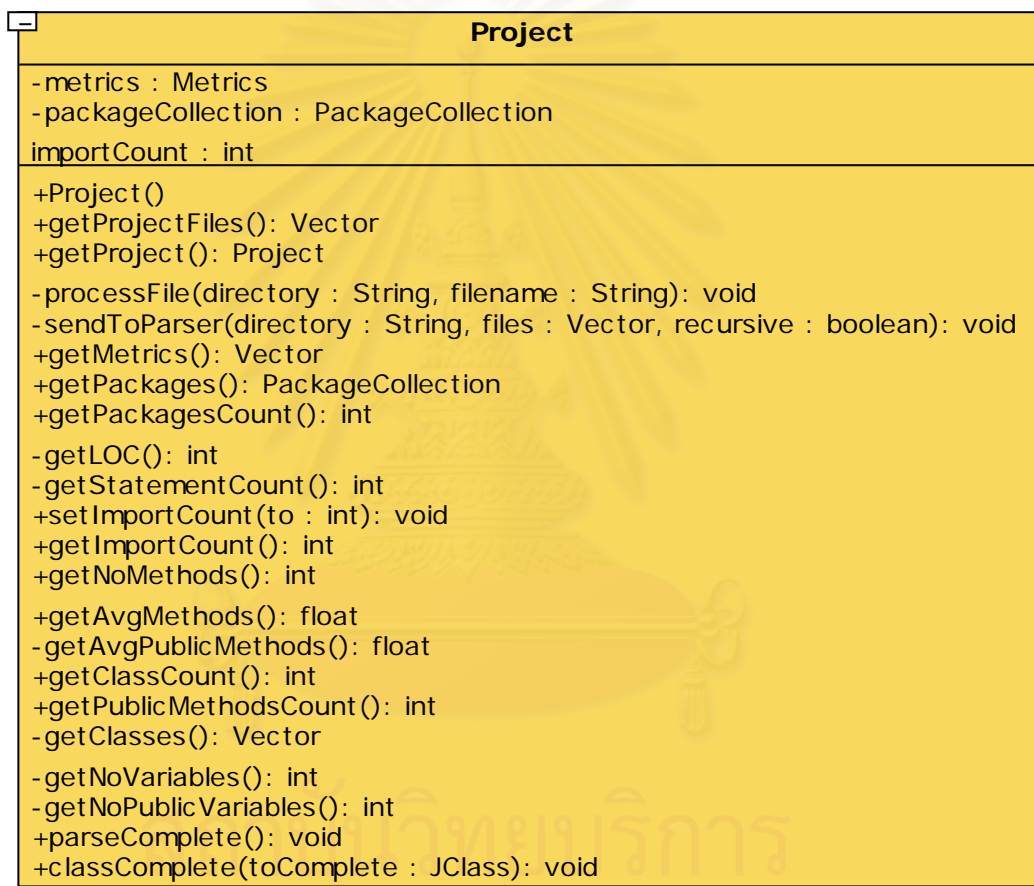
Metrics
-classCollection : JClassCollection -importCollection : StringCollection -currentClass : JClass -packages : StringCollection -currentPackage : String -classComplete : IClassComplete
+Metrics(to : IClassComplete) +setClassInterface(): void +setClassName(name : String): void +setClassPublic(): void +setClassPrivate(): void +setClassProtected(): void +startClass(): void +endClass(): void +startInnerClass(): void +endInnerClass(): void +addPackage(name : String): void +addMethod(name : String, isConstructor : boolean): void +startBlock(): void +endBlock(): void +addSuperClassCall(): void +addStatement(isEmpty : boolean, type : int): void +addImportStatement(name : String): void +addSuperClass(name : String): void +addInterface(name : String): void +addMethodParameter(): void +endMethod(): void +addInstanceVariable(): void +addLocalVariable(): void +getClasses(): Vector +getImportCount(): int +getPackages(): StringCollection

รูปที่ 4.7 รายละเอียดภายในคลาส "Metrics"



#### 4.2.2 คลาสหน่วยรู้จำเกี่ยวกับโปรเจก (Project)

รายละเอียดของคลาส "Project" แสดงในรูปที่ 4.8 เป็นคลาสที่มีหน้าที่ส่งคำร้องขอไปยังตัวสร้างชินแท็กซ์ทรี (JavaParser) โดยมีการส่งชื่อไคเรกทอรีที่เก็บโปรแกรมต้นฉบับ ชื่อของโปรแกรมต้นฉบับและสถานะการวนซ้ำ (recursive) เป็นพารามิเตอร์ เมธอดที่ทำหน้าที่นี้มีชื่อว่า "sendToParser" ภายในคลาส "Project" นี้จะมีการประกาศตัวแปร metrics ที่เป็นวัตถุของคลาส "Metrics" เพื่อใช้เก็บข้อมูลต่าง ๆ มีเมธอดที่เกี่ยวกับการกำหนดค่าการอ่านค่าเพื่อเก็บและเรียกใช้ค่าคุณสมบัติและค่าตัววัดต่าง ๆ ของโปรเจก โดยให้คลาสอื่นเข้ามาเรียกใช้คุณสมบัติและเมธอดต่าง ๆ ในคลาสนี้ได้ เช่น จำนวนแพ็คเกจ (getPackageCount) จำนวนคลาสของโปรเจก (getClassCount) จำนวนเมธอดของโปรเจก (getNoMethods) และจำนวนบรรทัดของโปรเจก (getLOC) เป็นต้น



รูปที่ 4.8 รายละเอียดภายในคลาส "Project"

#### 4.2.3 คลาสหน่วยรู้จำเกี่ยวกับคลาส (JClass)

รายละเอียดของคลาส "JClass" แสดงในรูปที่ 4.9 เป็นคลาสที่ทำหน้าที่เก็บรายละเอียดคุณสมบัติต่าง ๆ ที่เกี่ยวกับคลาส ภายในคลาสนี้จะมีเมธอดที่เกี่ยวกับการกำหนดค่า การอ่านค่าเพื่อเก็บและเรียกใช้ค่าคุณสมบัติ และค่าตัววัดต่าง ๆ ของคลาส โดยให้คลาสอื่นเข้ามาเรียกใช้คุณสมบัติและเมธอดต่าง ๆ ในคลาสนี้ได้ เช่น กำหนดชื่อคลาส (setName) อ่านชื่อคลาส (getName) การเพิ่มค่าให้กับเมธอด (addMethod) การอ่านและคำนวณค่าระดับของการขาดความสัมพันธ์ภายในคลาส (getLCOM) เป็นต้น จากรูปที่ 4.9 จะเห็นว่ามีการประกาศตัวแปรอินสแตนท์ที่เป็นตัวแปรอ้างอิง ชื่อ currentMethod ที่เป็นวัตถุในคลาส "Method" เพื่อให้สามารถเข้าใช้ข้อมูลและเมธอดในคลาส "Method" ได้ เพื่อทำการจัดเก็บข้อมูลและคำนวณค่าตัววัดซึ่งเป็นข้อมูลที่เกี่ยวข้องกับเมธอดในคลาสนั้นต่อไป



รูปที่ 4.9 รายละเอียดภายในคลาส "JClass"

#### 4.2.4 คลาสหน่วยรู้จำเกี่ยวกับเมทอด (Method)

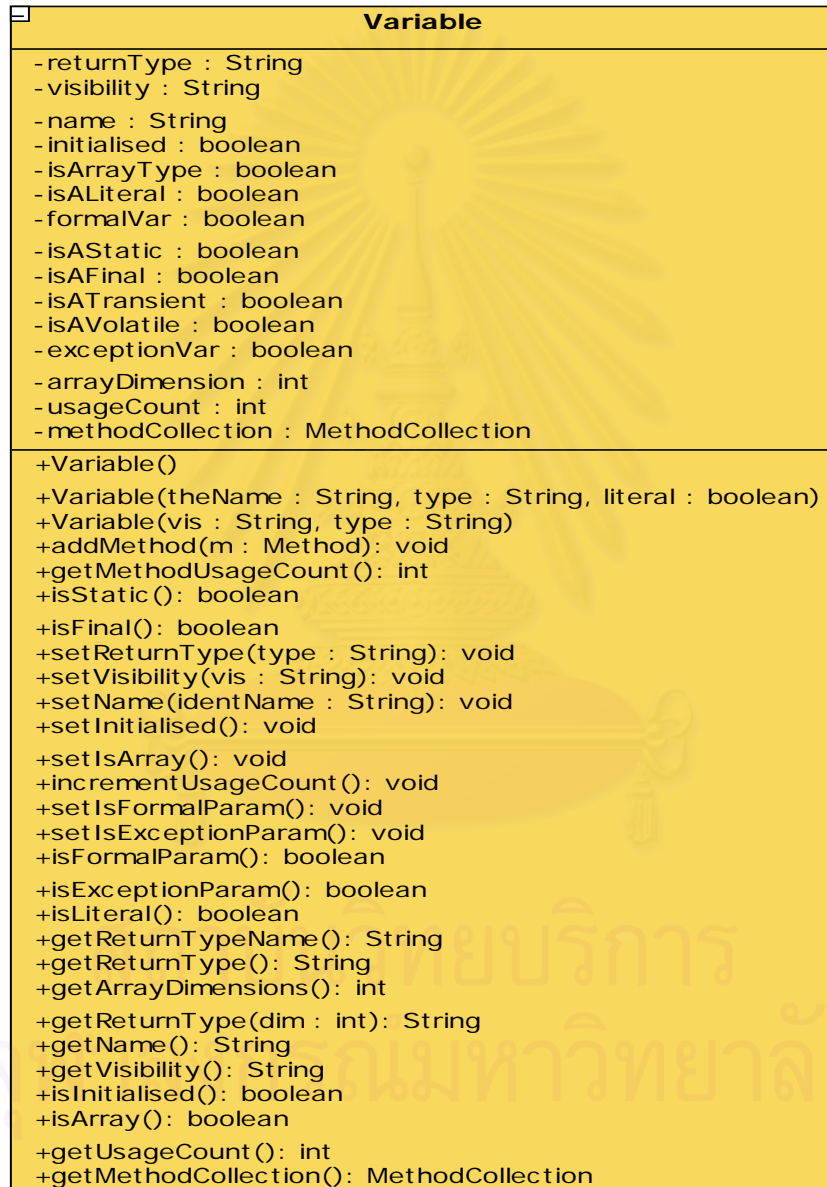
รายละเอียดของคลาส "Method" แสดงในรูปแบบที่ 4.10 เป็นคลาสที่ทำหน้าที่เก็บรายละเอียดคุณสมบัติต่าง ๆ ที่เกี่ยวกับเมทอด ภายในคลาสนี้จะมีเมทอดที่เกี่ยวกับการกำหนดค่า การอ่านค่าเพื่อเก็บและเรียกใช้ค่าคุณสมบัติ และค่าตัววัดต่าง ๆ ของเมทอด โดยให้คลาสอื่นเข้ามาเรียกใช้คุณสมบัติและเมทอดต่าง ๆ ในคลาสนี้ได้ เช่น การเพิ่มค่าจำนวนสเตทเมนต์ (addStatement) การอ่านค่าจำนวนสเตทเมนต์ (getStatementCount) การเพิ่มค่าจำนวนเงื่อนไขต่าง ๆ (addIfStatement, addCaseLabel, addWhileLoop, addForLoop) การอ่านค่าจำนวนเงื่อนไขต่าง ๆ (getIfStatementCount, getCaseLabelCount, getWhileLoopCount, getForLoopCount) เป็นต้น



รูปที่ 4.10 รายละเอียดภายในคลาส "Method"

#### 4.2.5 คลาสหน่วยรู้จำเกี่ยวกับตัวแปร (Variable)

รายละเอียดของคลาส " Variable " แสดงในรูปที่ 4.11 เป็นคลาสที่ทำหน้าที่เก็บรายละเอียดคุณสมบัติต่าง ๆ ที่เกี่ยวกับตัวแปร ภายในคลาสนี้จะมีเมธอดที่เกี่ยวกับการกำหนดค่า การอ่านค่าเพื่อเก็บและเรียกใช้ค่าคุณสมบัติ และค่าตัววัดต่าง ๆ ของตัวแปร โดยให้คลาสอื่นเข้ามาเรียกใช้คุณสมบัติและเมธอดต่าง ๆ ในคลาสนี้ได้ เช่น การกำหนดชื่อตัวแปร (setName) เพิ่มค่าเมธอดที่ใช้ตัวแปรนี้ (addMethod) อ่านค่าจำนวนเมธอดที่ใช้ตัวแปรนี้ (getMethodUsageCount) เพิ่มค่าจำนวนที่ตัวแปรถูกเรียกใช้ (incrementUsageCount)



รูปที่ 4.11 รายละเอียดภายในคลาส "Variable"

### 4.3 การเก็บค่าคุณสมบัติและคำนวณค่าตัววัด

ตัวอย่างรายละเอียดชุดคำสั่งของคลาสต่าง ๆ เพื่อใช้เก็บคุณสมบัติต่าง ๆ ของข้อมูลและคำนวณค่าตัววัด เพื่อให้คลาสจัดรูปแบบการแสดงผลค่าตัววัดนำไปแสดงผลตามที่ต้องการ สามารถอธิบายรายละเอียดได้ดังต่อไปนี้

#### 4.3.1 ชุดคำสั่งการนับจำนวนสเตทเมนต์

รายละเอียดของชุดคำสั่งเพื่อเก็บคุณสมบัติเกี่ยวกับการนับจำนวนสเตทเมนต์ ดังรูปที่ 4.12 เมื่อมีการสร้าง โหนด statement() ก็จะทำให้การเรียกโอเปอเรชัน addStatement(...) ซึ่งเป็นเมทอดของวัตถุ "Metrics" โดยมีหน้าที่ส่งคำร้องขอ (message) ต่อไปให้โอเปอเรชัน addStatement(...) ซึ่งเป็นเมทอดของวัตถุ "JClass" ถ้าค่าของ currInnerClass เท่ากับ null ก็จะส่งคำร้องขอไปยังโอเปอเรชัน addStatement(...) ซึ่งเป็นเมทอดของวัตถุ "Method" เพื่อทำการเพิ่มค่าให้กับตัวนับ ซึ่งตัวนับแบ่งออกเป็นประเภทต่าง ๆ เช่น statementCount, addIfStatement(), addCaseLabel(), addForLoop(), addWhileLoop() เป็นต้น

```
// Class Metrics
private JClass currentClass;
public void addStatement(boolean isEmpty, int type) {
    currentClass.addStatement(isEmpty, type);
}

// Class JClass
private Method currentMethod;
private JClass currInnerClass;
public void addStatement(boolean isBlank, int type) {
    if(currInnerClass == null)
        currentMethod.addStatement(isBlank, type);
    else
        currInnerClass.addStatement(isBlank, type);
}

// Class Method
public void addStatement(boolean isBlank, int type) {
    if(isBlank) {
        emptyStatementCount++;
        statementCount--;
    }
    else {
        switch(type) {
            case IF_STATEMENT:
                addIfStatement();
                break;
            case CASE_LABEL:
                addCaseLabel();
                break;
            case FOR_LOOP:
                addForLoop();
                break;
            case WHILE_LOOP:
                addWhileLoop();
                break;
            ...
            default:
                statementCount++;
                break;
        } // end switch
    } // end else
} // end addStatement
```

รูปที่ 4.12 รายละเอียดชุดคำสั่งการนับจำนวนสเตทเมนต์

#### 4.3.2 ชุดคำสั่งหาค่าวัคของไซโคลเมติกของแมคเคบ

รายละเอียดของชุดคำสั่งเพื่อหาค่าวัคของไซโคลเมติกของแมคเคบ ดังรูปที่ 4.13 เมื่อต้องการดูค่าวัคของไซโคลเมติกของแมคเคบ จะเริ่มจากการเรียกเมทอดของวัตถุ "MethodPresenter" เป็นคอนสตรัคเตอร์ (constructor) ที่จะมีการกำหนดค่าเริ่มต้นที่ได้มาจากวัตถุ "Method" ที่มีการอ่านค่าจากการนับค่าในหัวข้อ 4.3.1 การนับจำนวนสเตทเมนต์ โดยการเรียกโอเปอเรชันการอ่านค่าได้แก่ getIfStatementCount(), getCaseLabelCount(), getWhileLoopCount() และ getForLoopCount() โดยค่าเริ่มต้นที่ได้คือ ifStatementCount, caseLabelCount, whileLoopCount และ forLoopCount ตามลำดับ เพื่อใช้ในการคำนวณค่าวัคของไซโคลเมติกของแมคเคบ จากนั้นทำการเรียกโอเปอเรชัน makeCyclomaticComplexity() เพื่อทำการคำนวณค่าวัควัคของไซโคลเมติกของแมคเคบ แล้วกำหนดค่าให้กับตัวแปรอินสแตนซ์ที่ชื่อ cyclomaticComplexity เพื่อใช้ในการนำไปแสดงผลต่อไป

```
// Class MethodPresenter
private int cyclomaticComplexity;
public MethodPresenter(ClassPresenter parent, Method from) {
    ifStatementCount = from.getIfStatementCount();
    caseLabelCount = from.getCaseLabelCount();
    whileLoopCount = from.getWhileLoopCount();
    forLoopCount = from.getForLoopCount();
    ...
    makeCyclomaticComplexity()
}

private void makeCyclomaticComplexity() {
    int complexity = ifStatementCount + caseLabelCount;
    complexity += whileLoopCount;
    complexity += forLoopCount;
    cyclomaticComplexity = complexity + 1;
}

public int getCyclomaticComplexity() {
    return cyclomaticComplexity;
}

// Class Method
public int getIfStatementCount() {
    return ifStatementCount;
}

public int getCaseLabelCount() {
    return caseLabelCount;
}

public int getWhileLoopCount() {
    return whileLoopCount;
}

public int getForLoopCount() {
    return forLoopCount;
}
```

รูปที่ 4.13 รายละเอียดชุดคำสั่งหาค่าวัคของไซโคลเมติกของแมคเคบ

### 4.3.3 ชุดคำสั่งหาค่าวัดจำนวนบรรทัดของโปรเจก

```

// Class Project
private int getLOC() {
    int returnValue;
    Enumeration e = packageCollection.elements();
    while (e.hasMoreElements()) {
        returnValue += ((PackagePresenter)e.nextElement()).getLOC();
    }
    returnValue += getImportCount();
    return returnValue;
}

// Class PackagePresenter
public int getLOC() {
    int returnValue;
    Enumeration e = classCollection.elements();
    while (e.hasMoreElements()) {
        returnValue += ((ClassPresenter)e.nextElement()).getLOC();
    }
    return returnValue;
}

// Class ClassPresenter
public ClassPresenter(JClass from) {
    ...
    makeLOC();
}

public int getLOC() {
    return LOC;
}

public void makeLOC() {
    int returnValue = 0;
    Enumeration e = methodCollection.elements();
    while (e.hasMoreElements()) {
        returnValue += ((MethodPresenter)e.nextElement()).getLOC();
    }

    returnValue += getInnerClassLOC();
    returnValue = returnValue + variableCollection.size() + 1;
    LOC = returnValue;
}

// Class MethodPresenter
public MethodPresenter(ClassPresenter parent, Method from) {
    ...
    makeLOC()
}

public int getLOC() {
    return LOC;
}

private void makeLOC() {
    int LOC = getStatementCount() + localVarCollection.size();
    LOC++;
    this.LOC = LOC;
}

```

รูปที่ 4.14 รายละเอียดชุดคำสั่งหาค่าวัดจำนวนบรรทัดของโปรเจก

รายละเอียดของชุดคำสั่งเพื่อหาค่าวัดจำนวนบรรทัดของโปรเจกต์ ดังรูปที่ 4.14 เมื่อต้องการดูค่าตัววัดของจำนวนบรรทัดของโปรเจกต์ มีขั้นตอนการเรียกดูค่าดังต่อไปนี้

4.3.3.1 จำนวนบรรทัดของโปรเจกต์ ได้มาจากการเรียกโอเปอเรชันการอ่านค่า `getLOC()` ซึ่งเป็นเมธอดของวัตถุ "Project" ที่จะทำการนับผลรวมของจำนวนบรรทัดของแต่ละแพ็คเกจ (หัวข้อ 4.3.3.2) รวมกับจำนวนนับของอิมพอร์ต (`getImportCount()`)

4.3.3.2 จำนวนบรรทัดของแต่ละแพ็คเกจ ได้มาจากการส่งคำร้องขอให้กับโอเปอเรชันการอ่านค่า `getLOC()` ซึ่งเป็นเมธอดของวัตถุ "PackagePresenter" ที่จะทำการนับผลรวมของจำนวนบรรทัดของแต่ละคลาส (หัวข้อ 4.3.3.3)

4.3.3.3 จำนวนบรรทัดของแต่ละคลาส ได้มาจากการส่งคำร้องขอให้กับโอเปอเรชันการอ่านค่า `getLOC()` ซึ่งเป็นเมธอดของวัตถุ "ClassPresenter" ที่ได้มาจากการเก็บค่าจำนวนบรรทัดของคลาสที่มีการส่งคำร้องขอจากคอนสตรัคเตอร์ของวัตถุ "ClassPresenter" ไปยังโอเปอเรชัน `makeLOC()` ซึ่งจะทำการนับผลรวมของจำนวนบรรทัดของแต่ละเมธอดในคลาสนั้น ๆ (หัวข้อ 4.3.3.4) รวมกับจำนวนบรรทัดภายในอินเนอร์คลาส (`getInnerClassLOC()`) และรวมกับจำนวนบรรทัดที่มีการประกาศตัวแปรอินสแตนซ์ (`variableCollection.size()`) บวกด้วย 1 คือบรรทัดที่ใช้ในการประกาศชื่อคลาส

4.3.3.4 จำนวนบรรทัดของแต่ละเมธอด ได้มาจากการส่งคำร้องขอให้กับโอเปอเรชันการอ่านค่า `getLOC()` ซึ่งเป็นเมธอดของวัตถุ "MethodPresenter" ที่ได้มาจากการเก็บค่าจำนวนบรรทัดของเมธอดที่มีการส่งคำร้องขอจากคอนสตรัคเตอร์ของวัตถุ "MethodPresenter" ไปยังโอเปอเรชัน `makeLOC()` ซึ่งจะทำการนับผลรวมของจำนวนสเตตเมนต์ของเมธอดนั้น ๆ (`getStatementCount()`) รวมกับจำนวนบรรทัดของการประกาศตัวแปรภายในเมธอดนั้น ๆ (`localVarCollection.size()`)




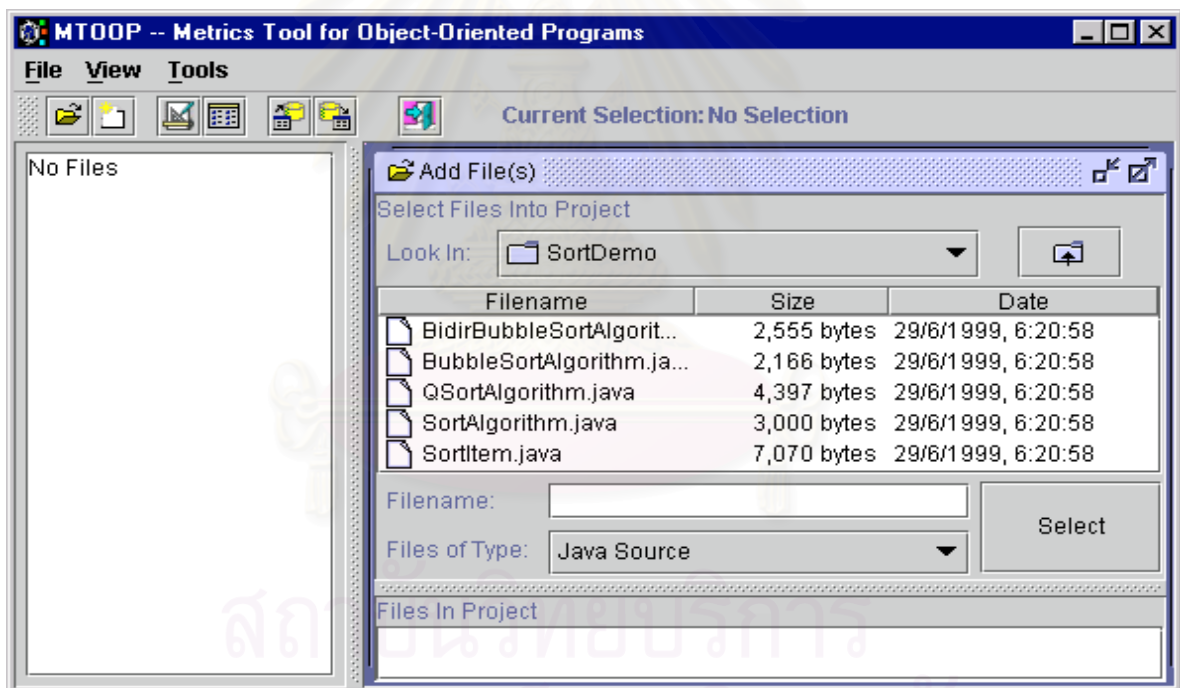
## บทที่ 5

### การใช้งานระบบ MTOOP




ในบทนี้จะอธิบายถึงการใช้งานระบบ MTOOP ซึ่งจะแบ่งออกเป็น 3 ส่วนคือ การอ่านโปรแกรมต้นฉบับ การดูค่าตัววัด และการเก็บและเรียกดูค่าตัววัดจากฐานข้อมูล ดังรายละเอียดต่อไปนี้

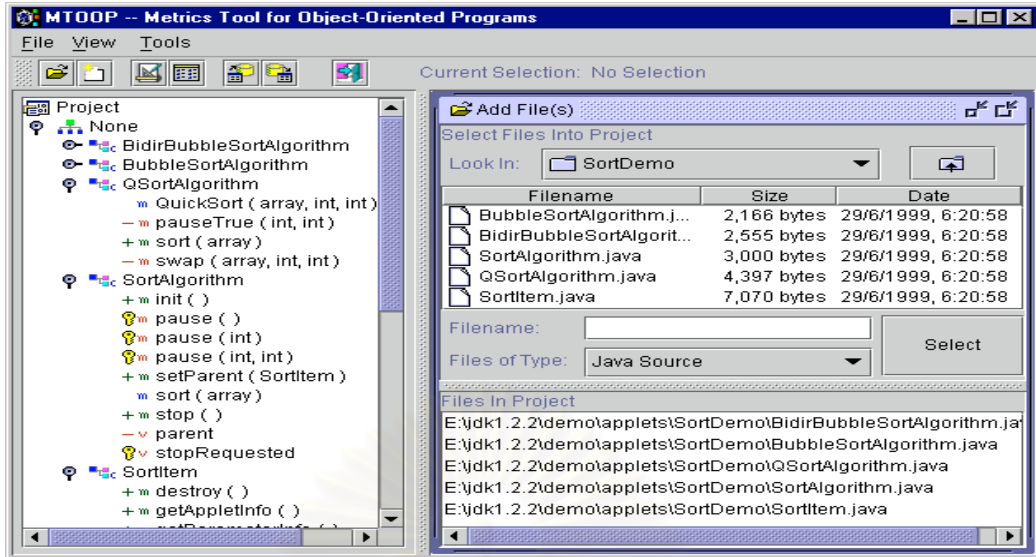
#### 5.1 การอ่านโปรแกรมต้นฉบับ

เมื่อผู้ใช้เข้าสู่ระบบ MTOOP ผู้ใช้สามารถเลือกโปรแกรมต้นฉบับได้จากฟังก์ชันเพิ่มไฟล์  (File/Add Files) เพื่อเลือกโปรแกรมต้นฉบับเข้าสู่โปรเจกต์ ผู้ใช้สามารถเลือกทีละไฟล์หรือเลือกทั้งไดเรกทอรี (directory) ที่มีโปรแกรมต้นฉบับอยู่ ดังรูปที่ 5.1




รูปที่ 5.1 เฟอร์มเพื่อเลือกโปรแกรมต้นฉบับภาษาจาวา

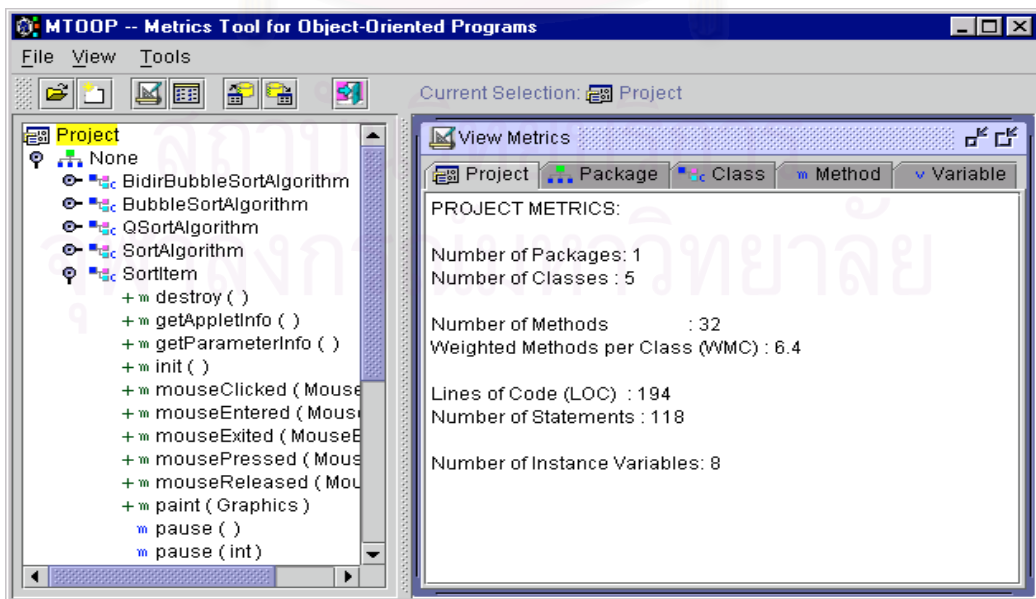
หลังจากผู้ใช้ได้เลือกโปรแกรมต้นฉบับภาษาจาวาเข้าสู่ระบบแล้ว ระบบจะแสดงชื่อของโปรเจกต์ แพ็กเกจ คลาส เมททอด และตัวแปรอินสแตนซ์ในรูปแบบต้นไม้ (Tree) ดังรูปที่ 5.2 ทางด้านซ้ายมือ เพื่อให้ผู้ใช้สามารถเลือกดูค่าตัววัดต่าง ๆ ตามที่ผู้ใช้ต้องการ หรือถ้าผู้ใช้ต้องการเลือกโปรแกรมต้นฉบับเพิ่มเติมเข้าสู่โปรเจกต์ ผู้ใช้ก็สามารถใช้ฟังก์ชันเพิ่มไฟล์  (File/Add Files) เพื่อเลือกโปรแกรมต้นฉบับ หรือถ้าผู้ใช้ต้องการที่จะเริ่มเลือกโปรแกรมต้นฉบับเข้าสู่โปรเจกต์ใหม่ตั้งแต่ต้น ผู้ใช้สามารถใช้ฟังก์ชันเริ่มโปรเจกต์ใหม่  (File/New Project) หรือถ้าผู้ใช้ต้องการที่จะออกจากระบบ ผู้ใช้สามารถใช้ฟังก์ชันออกจากระบบ  (File/Exit)



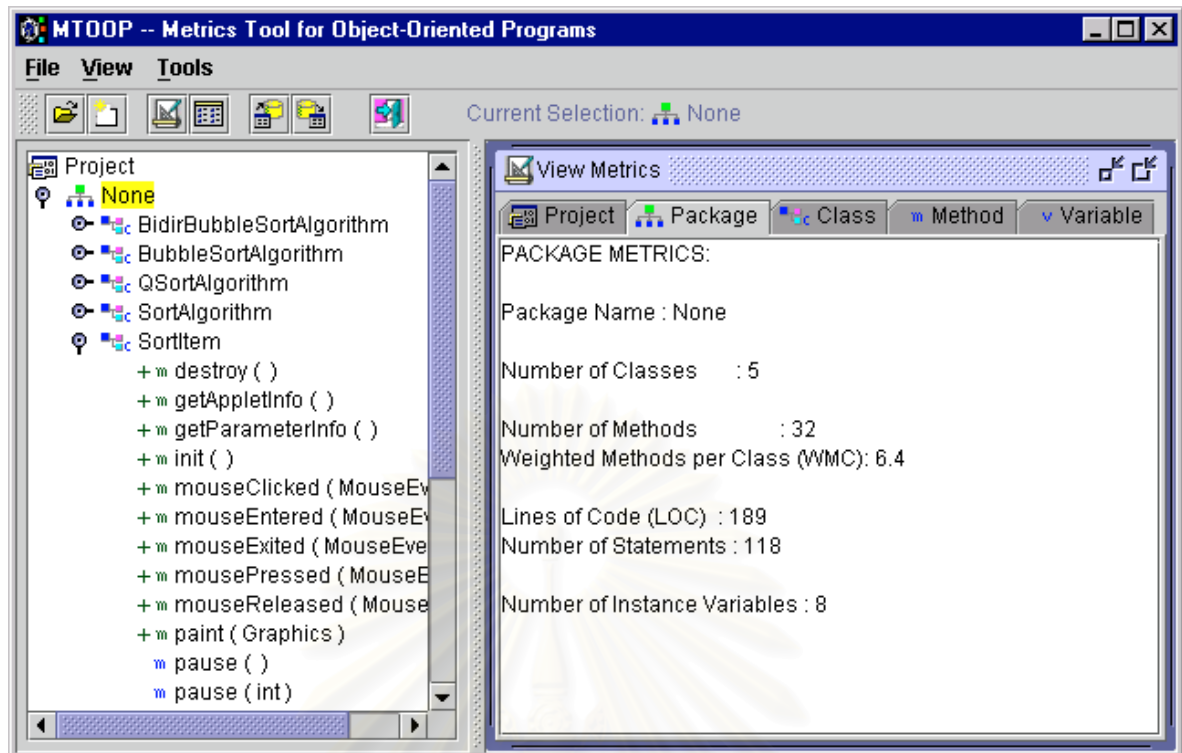
รูปที่ 5.2 ทริและโหนดต่างๆ ที่ได้จาก การเลือกโปรแกรมต้นฉบับ

## 5.2 การดูค่าตัววัด

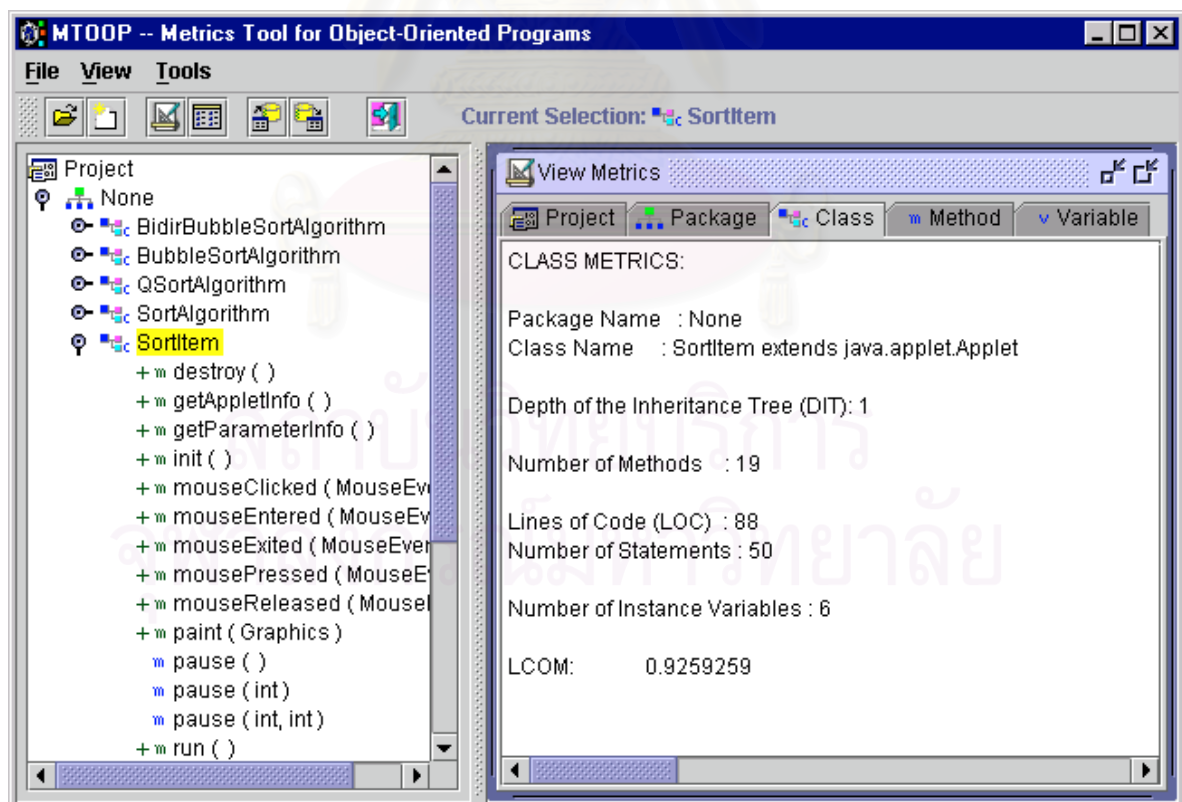
เมื่อผู้ใช้งานต้องการดูค่าตัววัด ผู้ใช้สามารถเลือกฟังก์ชันดูค่าตัววัด  (View/View Metrics) ระบบก็จะทำการแสดงเฟรมที่มีแท็บแบ่งตามประเภทของโปรเจกต์ แพ็กเกจ คลาส เมธอด และตัวแปรอินสแตนซ์ เมื่อผู้ใช้งานคลิกโหนดของต้นไม้ ค่าตัววัดในเฟรมที่แสดงค่าก็จะเปลี่ยนไปตามโหนดที่เลือก เช่น รูปที่ 5.3 เป็นการดูค่าตัววัดตามประเภทของโปรเจกต์เมื่อคลิกโหนดที่ชื่อ "Project" รูปที่ 5.4 เป็นการดูค่าตัววัดตามประเภทของแพ็กเกจเมื่อคลิกโหนดที่ชื่อ "None" รูปที่ 5.5 เป็นการดูค่าตัววัดตามประเภทของคลาสเมื่อคลิกโหนดที่ชื่อ "SortItem" รูปที่ 5.6 เป็นการดูค่าตัววัดตามประเภทของเมธอดเมื่อคลิกโหนดที่ชื่อ "paint(Graphics)" รูปที่ 5.7 เป็นการดูค่าตัววัดตามประเภทของตัวแปรอินสแตนซ์เมื่อคลิกโหนดที่ชื่อ "parent" เป็นต้น



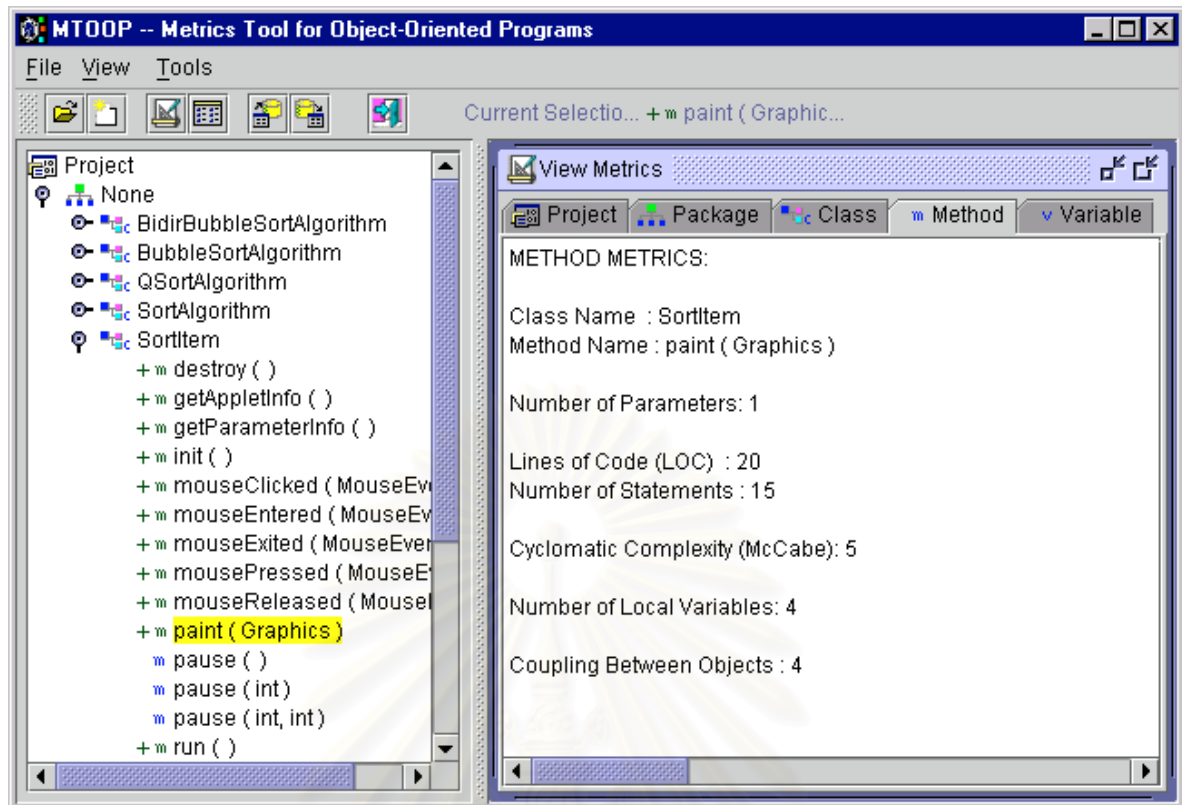
รูปที่ 5.3 การดูค่าตัววัดตามประเภทของโปรเจกต์เมื่อคลิกโหนดที่ชื่อ "Project"



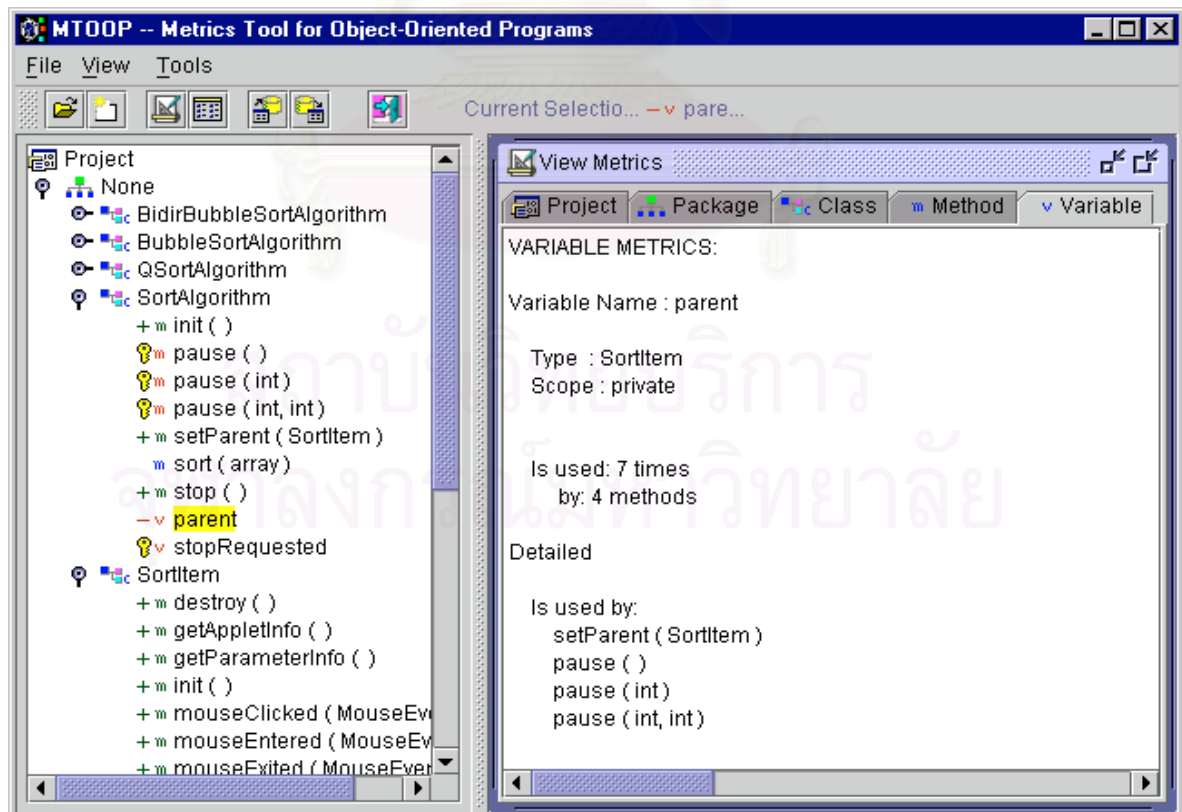
รูปที่ 5.4 การดูค่าตัววัดตามประเภทของแพ็คเกจเมื่อกดคลิกโหนดที่ชื่อ "None"



รูปที่ 5.5 การดูค่าตัววัดตามประเภทของคลาสเมื่อกดคลิกโหนดที่ชื่อ "SortItem"

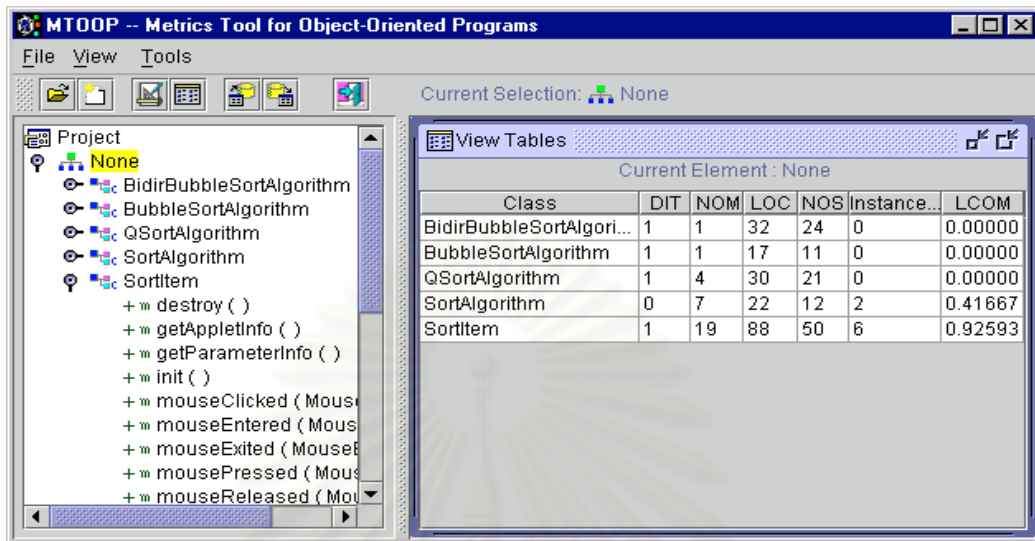


รูปที่ 5.6 การดูค่าตัววัดตามประเภทของเมทอดเมื่อคลิกโหนดที่ชื่อ "paint(Graphics)"



รูปที่ 5.7 การดูค่าตัววัดตามประเภทของตัวแปรอินสแตนซ์เมื่อคลิกโหนดที่ชื่อ "parent"

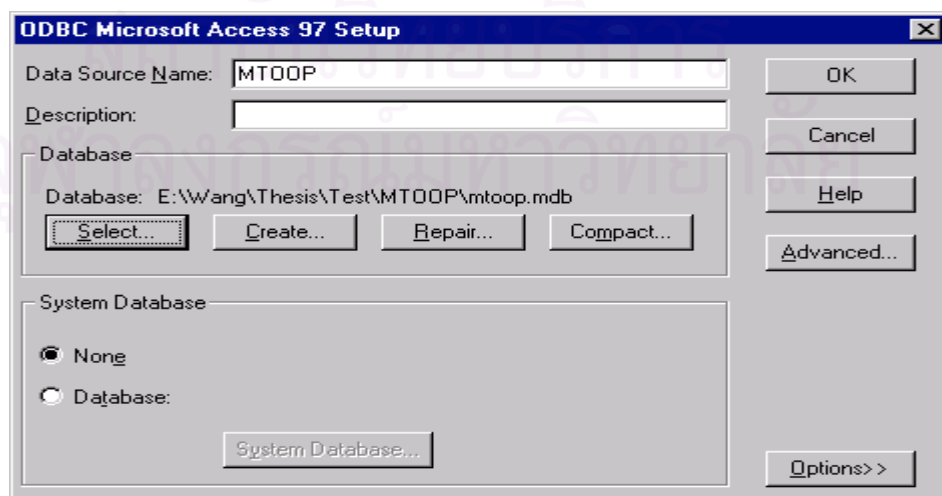
หรือถ้าผู้ใช้ต้องการดูค่าตัววัดในรูปของตาราง ผู้ใช้สามารถเลือกฟังก์ชันดูค่าตัววัดรูปแบบตาราง (View/View Tables) ระบบก็จะทำการแสดงเฟรมที่มีค่าตัววัดในรูปของตาราง ผู้ใช้สามารถดูค่าตัววัดเมื่อคลิกไปที่ โหนดของต้นไม้ ซึ่งสามารถดูได้เฉพาะ โหนดที่เป็น โปรเจกต์ แพ็กเกจและ โหนดที่เป็นคลาสเท่านั้น ดังรูปที่ 5.8




รูปที่ 5.8 ตัวอย่างการดูค่าตัววัดรูปแบบตาราง

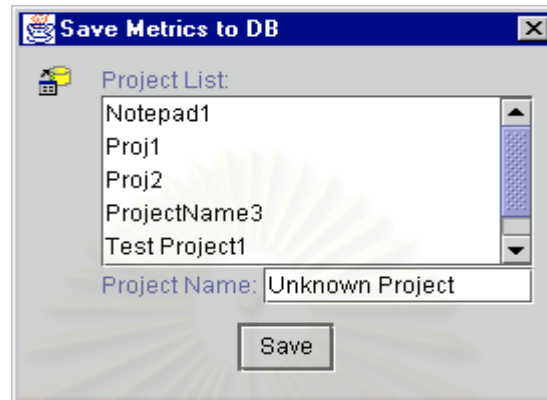
### 5.3 การเก็บและเรียกดูค่าตัววัดจากฐานข้อมูล

การเก็บค่าตัววัดของโปรเจกต์ลงฐานข้อมูล เนื่องจากเครื่องมือวัดนี้จะทำการบันทึกข้อมูลลงฐานข้อมูล ด้วยการผ่านเจดีบีซี (JDBC – Java Database Connectivity) ซึ่งต้องมีการกำหนดช่องทางติดต่อเพื่อใช้รับส่งข้อมูล ด้วยการใช้อีดีบีซี (ODBC – Open Database Connectivity) โดยใช้ไดรเวอร์ของไมโครซอฟต์แอคเซส (Microsoft Access Driver) และกำหนดค่าตัวซอร์สเนม (Data Source Name) ชื่อ "MTOOP" และไฟล์ค่าตัวเบส ชื่อ "mtoop.mdb" ในเครื่องมือจัดการ โอดีบีซี (ODBC Data Source Administrator) ผู้ใช้จึงจะสามารถเก็บบันทึกหรือร้องขอข้อมูลได้ ดังรูปที่ 5.9




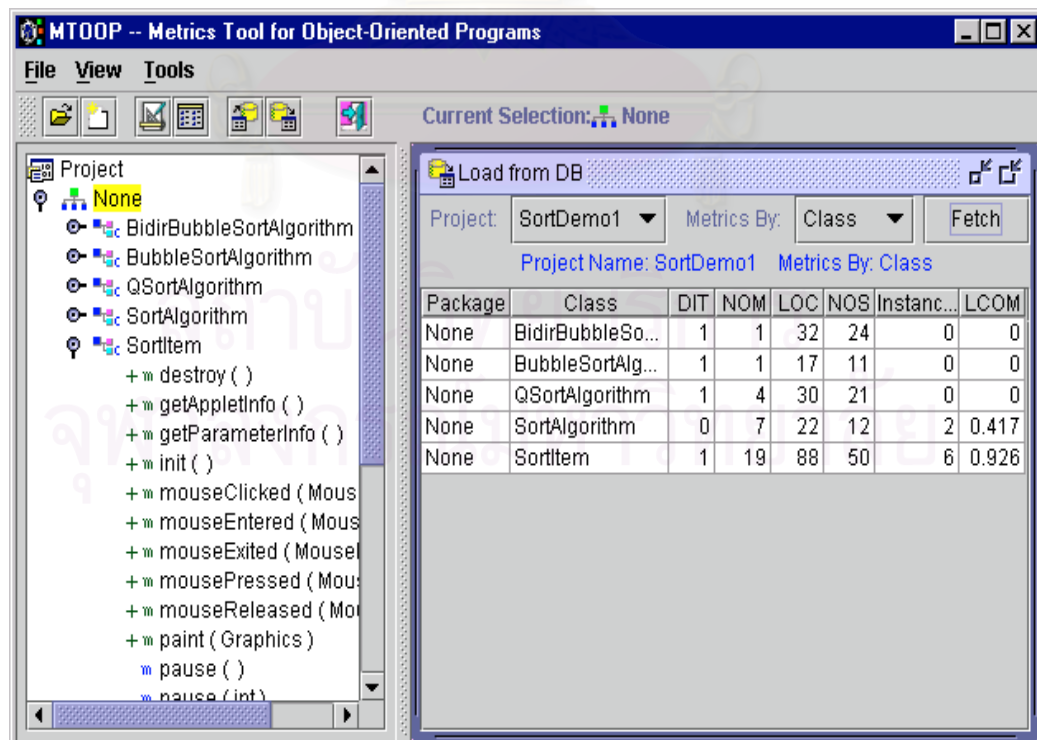
รูปที่ 5.9 การกำหนดค่าตัวซอร์สเนมและไฟล์ค่าตัวเบสในเครื่องมือจัดการ โอดีบีซี

เมื่อผู้ใช้เลือกฟังก์ชันการบันทึกข้อมูลลงฐานข้อมูล  (Tools/Save to DB) ระบบก็จะทำการแสดง ไดอะล็อกบ็อกซ์ (Dialog Box) เพื่อให้เลือกชื่อโปรเจกต์ที่ได้มีการบันทึกลงฐานข้อมูลไปแล้ว หรือให้กรอกชื่อโปรเจกต์ที่ต้องการบันทึกโดยระบบได้กำหนดชื่อโปรเจกต์ไว้คือ Unknown Project และมีปุ่มบันทึกข้อมูล (Save) เพื่อทำการบันทึกข้อมูลลงฐานข้อมูล ดังรูปที่ 5.10



รูปที่ 5.10 ตัวอย่างการใช้ฟังก์ชันการบันทึกข้อมูลลงฐานข้อมูล

หลังจากนั้นผู้ใช้สามารถเลือกฟังก์ชันดูค่าตัววัดจากฐานข้อมูล  (Tools/Load from DB) ระบบก็จะทำการแสดงเฟรมที่มีรายชื่อของโปรเจกต์และประเภทที่ต้องการดูค่าตัววัดที่แยกตาม แพ็กเกจ คลาส และเมทอด ให้เลือก จากนั้นทำการคลิกปุ่มแสดงข้อมูล (Fetch) ระบบก็จะนำเงื่อนไขที่เลือกไปขอข้อมูลจากฐานข้อมูลแล้วทำการแสดงค่าตัววัดในรูปของตารางดังรูปที่ 5.11



รูปที่ 5.11 ตัวอย่างการใช้ฟังก์ชันดูค่าตัววัดจากฐานข้อมูล

## บทที่ 6

### การทดสอบ

ผู้วิจัยได้ทำการทดสอบการทำงานของเครื่องมือที่ได้พัฒนาขึ้นมา ด้วยการนำโปรแกรมต้นฉบับที่มีขนาดต่าง ๆ กัน เป็นจำนวน 6 โปรแกรม มาทำการหาค่าตัววัดต่าง ๆ แล้วนำค่าตัววัดที่ได้มาเปรียบเทียบ โดยโปรแกรมทดสอบที่ 1 จะเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดและการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับการวัดด้วยมือ ส่วนโปรแกรมทดสอบที่ 2 ถึงโปรแกรมทดสอบที่ 6 จะเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับเครื่องมือวัดที่มีชื่ออยู่ในขณะนี้ ดังรายละเอียดต่อไปนี้

#### 6.1 การเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดและการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับการวัดด้วยมือ

โปรแกรมทดสอบที่ 1 แสดงให้เห็นผลของการใช้ฟังก์ชันการดูค่าตัววัดและการใช้ฟังก์ชันการดูค่าตัววัดแบบตาราง ซึ่งค่าตัววัดจะทำการตรวจสอบค่าด้วยการนับและการคำนวณด้วยมือ ดังนั้นจึงใช้ตัวอย่างโปรแกรมที่มีขนาดไม่ถึง 2 KLOC ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ได้ตามที่ได้ออกแบบไว้

##### 6.1.1 ผลการทดสอบโปรแกรมที่ 1

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 1 ได้มาจากตัวอย่างโปรแกรมในหนังสือชื่อ “Java AWT Reference” เขียนโดย John Zukowski [11] ซึ่งมีขนาด 1,602 ไบต์ ดังรูปที่ 6.1 เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลผลค่าตัววัดด้วยเครื่องมือ MTOOP จะได้ผลค่าตัววัดที่แยกตามประเภทต่าง ๆ

```
1. // This example is from the book _Java AWT Reference_ by John Zukowski.
2. // Written by John Zukowski. Copyright (c) 1997 O'Reilly & Associates.
3. // You may study, use, modify, and distribute this example for any purpose.
4. // This example is provided WITHOUT WARRANTY either expressed or
5. import java.awt.*;
6.
7. class CardPanel extends Panel {
8.     Panel create(LayoutManager layout) {
9.         Panel p = new Panel();
10.        p.setLayout(layout);
11.        p.add("North", new Button("this"));
12.        p.add("West", new Button("is"));
13.        p.add("South", new Button("a"));
14.        p.add("East", new Button("test"));
15.        p.add("Center", new Button("applet"));
16.        return p;
17.    }
18.    CardPanel() {
19.        setLayout(new CardLayout());
20.        add("flow", create(new FlowLayout()));
21.        add("border", create(new BorderLayout()));
22.        add("grid", create(new GridLayout(2, 2)));
23.    }
24. }
```


รูปที่ 6.1 โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 1

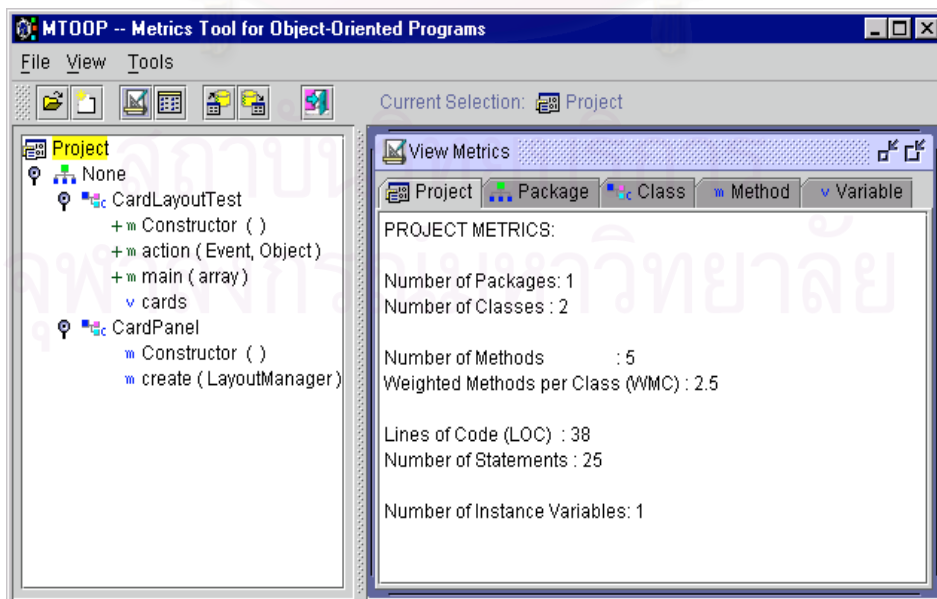
```

25. public class CardLayoutTest extends java.applet.Applet {
26.     private CardPanel cards;
27.     public CardLayoutTest() {
28.         setLayout(new BorderLayout());
29.         add("Center", cards = new CardPanel());
30.         Choice c = new Choice();
31.         c.addItem("flow");
32.         c.addItem("border");
33.         c.addItem("grid");
34.         add("South", c);
35.     }
36.     public boolean action(Event evt, Object arg) {
37.         if (evt.target instanceof Choice) {
38.             ((CardLayout)cards.getLayout()).show(cards,(String)arg);
39.         }
40.         return true;
41.     }
42.     public static void main(String args[]) {
43.         Frame f = new Frame("CardLayoutTest");
44.         CardLayoutTest card = new CardLayoutTest();
45.         card.init();
46.         card.start();
47.         f.add("Center", card);
48.         f.resize(300, 300);
49.         f.show();
50.     }
51. }

```

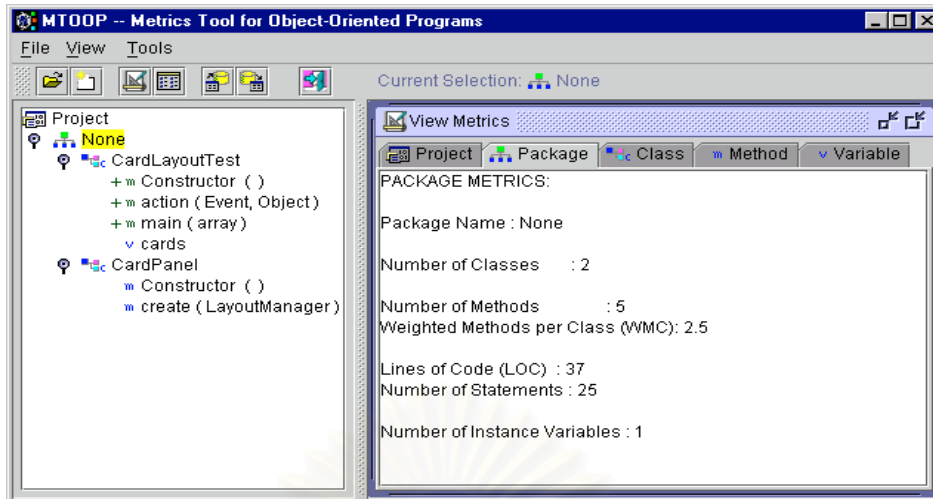
รูปที่ 6.1 โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 1 (ต่อ)

เมื่อใช้เครื่องมือ MTOOP ในการดูค่าตัววัดโดยการใช้ฟังก์ชันดูค่าตัววัด  (View/View Metrics) ซึ่งต้องการให้เห็นภาพของผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภทหลังจากที่ได้นำโปรแกรมต้นฉบับทดสอบที่ 1 มาทำการประมวลผลแล้วได้ผลลัพธ์ คือ ค่าตัววัดสำหรับโปรเจก แสดงในรูปที่ 6.2 ค่าตัววัดสำหรับแพ็คเกจที่ชื่อ "None" แสดงในรูปที่ 6.3 ค่าตัววัดสำหรับคลาสที่ชื่อ "CardLayoutTest" แสดงในรูปที่ 6.4 ค่าตัววัดสำหรับเมทอดที่ชื่อ "main( array )" แสดงในรูปที่ 6.5 และค่าตัววัดสำหรับตัวแปรอินสแตนซ์ที่ชื่อ "cards" แสดงในรูปที่ 6.6 ผลที่ได้เมื่อทำการตรวจสอบค่าตัววัดที่ได้เทียบกับการคำนวณด้วยมือ พบว่าเครื่องมือวัดสามารถคำนวณได้ถูกต้องตามที่ได้ออกแบบไว้

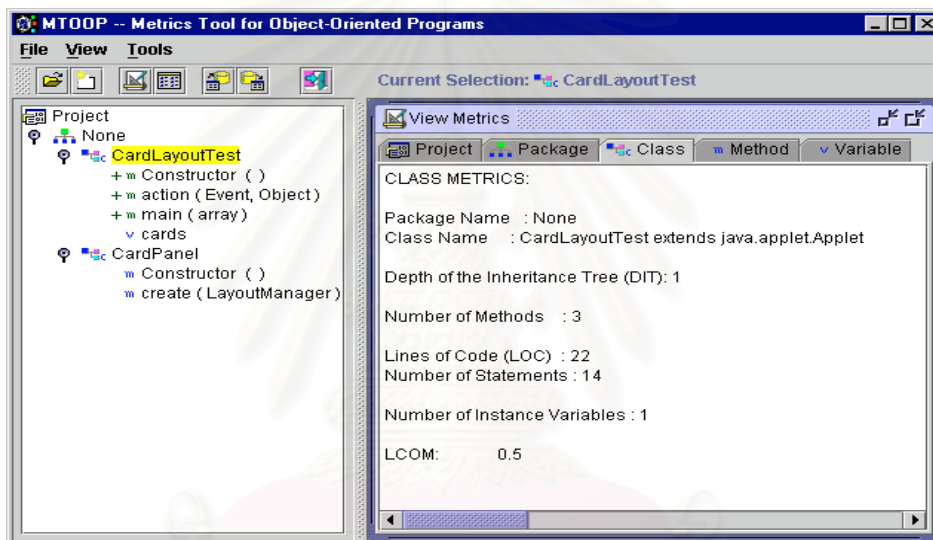


รูปที่ 6.2 ค่าตัววัดสำหรับโปรเจก ของโปรแกรมทดสอบที่ 1

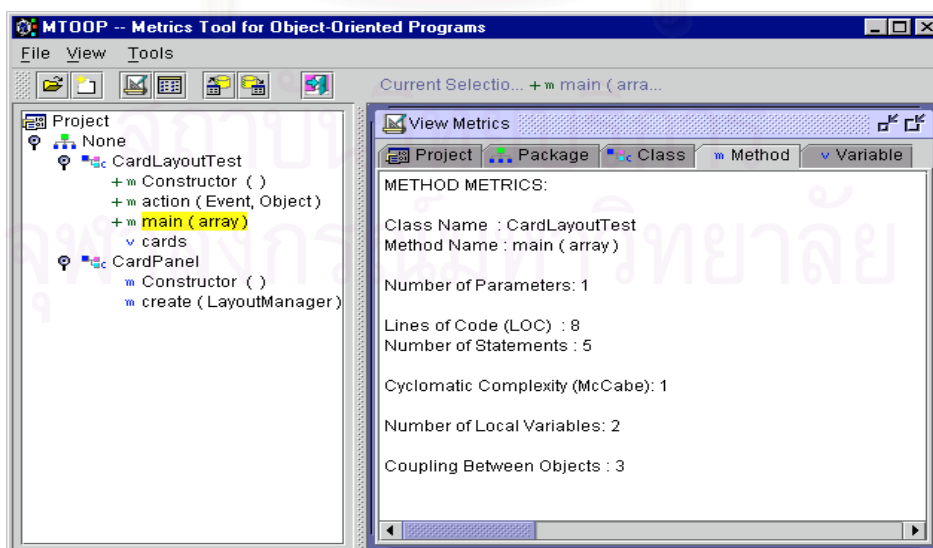




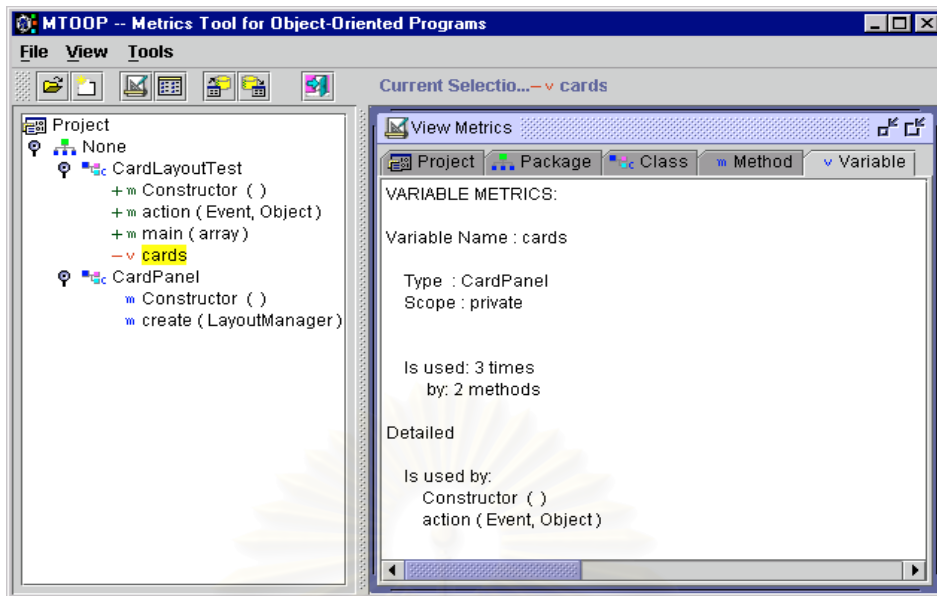
รูปที่ 6.3 ค่าตัววัดสำหรับแพ็คเกจที่ชื่อ "None" ของโปรแกรมทดสอบที่ 1



รูปที่ 6.4 ค่าตัววัดสำหรับคลาสที่ชื่อ "CardLayoutTest" ของโปรแกรมทดสอบที่ 1



รูปที่ 6.5 ค่าตัววัดสำหรับเมทอดที่ชื่อ "main( array )" ของโปรแกรมทดสอบที่ 1



รูปที่ 6.6 ค่าตัววัดสำหรับตัวแปรอินสแตนซ์ที่ชื่อ "cards" ของโปรแกรมทดสอบที่ 1

เมื่อใช้เครื่องมือ MTOOP ในการดูค่าตัววัดโดยใช้ฟังก์ชันค่าตัววัดในรูปแบบตาราง (View/View Tables) ซึ่งต้องการให้เห็นภาพของผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภทหลังจากที่ได้นำโปรแกรมต้นฉบับทดสอบที่ 1 มาทำการประมวลผลแล้วได้ผลลัพธ์ คือ ค่าตัววัดของแพ็คเกจในโปรเจก แสดงในรูปที่ 6.7 ค่าตัววัดของคลาสในโปรเจก แสดงในรูปที่ 6.8 และค่าตัววัดของเมธอดในโปรเจก แสดงในรูปที่ 6.9 ผลที่ได้เมื่อทำการตรวจสอบค่าตัววัดที่ได้เทียบกับการคำนวณด้วยมือ พบว่าเครื่องมือวัดสามารถคำนวณได้ถูกต้องตามที่ได้ออกแบบไว้

The screenshot shows the 'View Tables' window with 'Current Element : Project'. The table displays metrics for the 'None' package:

Package	NOC	NOM	WMC	LOC	NOS	Instance Var.
None	2	5	2.50	37	25	1

รูปที่ 6.7 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 1

The screenshot shows the 'View Tables' window with 'Current Element : None'. The table displays metrics for classes 'CardLayoutTest' and 'CardPanel':

Class	DIT	NOM	LOC	NOS	Instance Var.	LOCOM
CardLayoutTest	1	3	22	14	1	0.50000
CardPanel	1	2	15	11	0	0.00000

รูปที่ 6.8 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 1

The top screenshot shows the MTOOP interface with 'CardLayoutTest' selected. The 'View Tables' window displays the following data:

Method	Parameters	LOC	NOS	V(G)	Local Var.	CBO
Constructor ( )	0	8	6	1	1	2
action ( Event, Object )	2	4	3	2	0	4
main ( array )	1	8	5	1	2	3

The bottom screenshot shows the MTOOP interface with 'CardPanel' selected. The 'View Tables' window displays the following data:

Method	Parameters	LOC	NOS	V(G)	Local Var.	CBO
Constructor ( )	0	5	4	1	0	0
create ( LayoutManager )	1	9	7	1	1	2

รูปที่ 6.9 ค่าตัววัดของเมทอดในโปรเจก สำหรับโปรแกรมทดสอบที่ 1

## 6.2 การเปรียบเทียบผลการทดสอบการใช้ฟังก์ชันการดูค่าตัววัดแบบตารางกับเครื่องมือวัดที่มีข้อมูลในขณะนี้

โปรแกรมทดสอบที่ 2 ถึงโปรแกรมทดสอบที่ 6 ใช้สำหรับแสดงผลของค่าตัววัดแบบตาราง โดยนำค่าตัววัดที่ได้มาเปรียบเทียบกับเครื่องมือวัดที่มีข้อมูลในขณะนี้ ได้แก่ JMetric (Java Metrics Analyser) และ JavaNCSS (Java Non Commenting Source Statements) ซึ่งตัววัดต่าง ๆ ที่แต่ละเครื่องมือวัดแสดงผลจะมีจำนวนตัววัดที่แตกต่างกัน โดยจะทำการเปรียบเทียบเฉพาะตัววัดที่เหมือนกัน โปรแกรมที่ใช้ในการทดสอบจะมีขนาดอย่างน้อย 2 KLOC แต่ไม่เกิน 50 KLOC ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ แต่จะมีค่าตัววัดบางตัวที่คำนวณค่าได้แตกต่างกัน เช่น จำนวนบรรทัด เพราะข้อกำหนดในการนับที่แตกต่างกัน เพราะเมื่อทำการคำนวณค่าด้วยมือในเมทอดที่มีค่าตัววัดที่แตกต่างกัน เครื่องมือวัด MTOOP หาค่าได้ตามที่ได้ออกแบบไว้ โดยจะแสดงรายละเอียดของค่าตัววัดที่ได้ทำการทดสอบโปรแกรมทดสอบต่าง ๆ ดังต่อไปนี้

### 6.2.1 ผลการทดสอบโปรแกรมที่ 2

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 2 มีขนาด 2.5 KLOC เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลค่าตัววัดด้วยเครื่องมือ MTOOP, JMetric และ JavaNCSS แล้วจะได้ผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภท คือ

1. ค่าตัววัดของแพ็กเกจในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.10 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของแพ็กเกจที่ชื่อ "ptolemy.plot" ได้ดังตารางที่ 6.1 ปรากฏว่า

เครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด เมื่อเทียบกับเครื่องมือวัด JavaNCSS มีค่าไม่เท่ากัน เป็นเพราะว่า

- 1.1 มีการประกาศตัวแปรหลาย ๆ ตัวแปรในบรรทัดเดียวกัน ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับจำนวนบรรทัดเท่ากับจำนวนตัวแปรที่มีการประกาศ ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 1 บรรทัดไม่ว่าจะมีการประกาศตัวแปรกี่ตัวก็ตาม
- 1.2 มีการใช้คำสั่ง else if (...) เป็นจำนวนมาก ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด

รูปที่ 6.13 เป็นตัวอย่างการนับจำนวนบรรทัดของเมธอด "\_parseArg(String[])" ของโปรแกรมต้นฉบับทดสอบที่ 2

2. ค่าตัววัดของคลาสในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.11 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของคลาสที่ชื่อ "PlotApplication" ได้ดังตารางที่ 6.2 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก
3. ค่าตัววัดของเมธอดในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.12 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของเมธอดที่ชื่อ "\_parseArg(String[])" ได้ดังตารางที่ 6.3 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก

The image shows three screenshots of software tools used for code metrics analysis. The top screenshot is MTOOP (Metrics Tool for Object-Oriented Programs) showing a table of metrics for the 'Project' element. The middle screenshot is JMetric (Java Metrics Analyser) showing a table of metrics for the 'Metrics Project' element. The bottom screenshot is JavaNCSS showing a detailed summary of metrics for the project.

Package	NOC	NOM	WMC	LOC	NOS	Instance Var.
ptolemy.plot	12	146	12.17	2561	1847	133

Package	Lines of Co...	Statements	Classes	Methods
ptolemy.plot	2561	1847	12	146

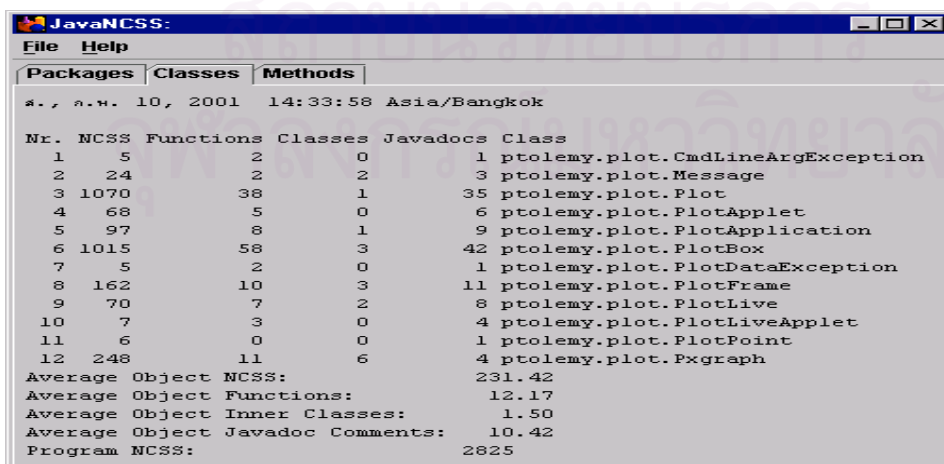
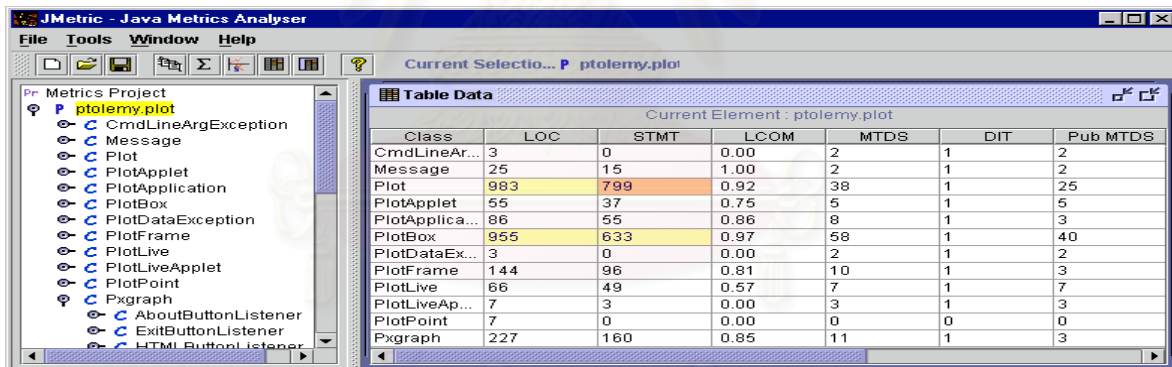
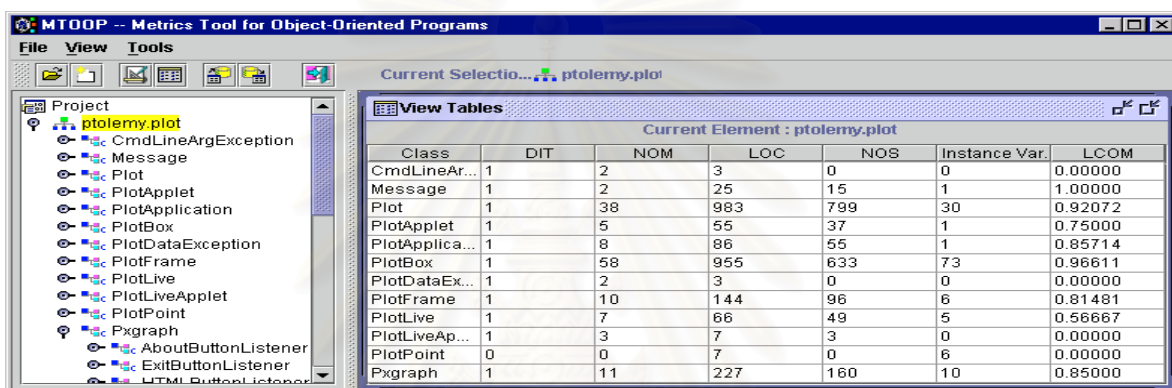
  

Packages	Classes	Methods	NCSS	per
1.00	12.00	146.00	2,825.00	Project
	12.00	146.00	2,825.00	Package
		12.17	235.42	Class
			19.35	Function

รูปที่ 6.10 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 2

ตารางที่ 6.1 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของแพ็คเกจ สำหรับโปรแกรมทดสอบที่ 2

ชื่อแพ็คเกจ	เครื่องมือวัด	จำนวนคลาส	จำนวนเมทอด	จำนวนเมทอดต่อคลาส	จำนวนบรรทัด	จำนวนสเตทเมนต์	จำนวนตัวแปรอินสแตนท์
ptolemy.plot							
	MTOOP	12	146	12.17	2561	1847	133
	JMetric	12	146	-	2561	1847	-
	JavaNCSS	12	146	12.17	2825	-	-



รูปที่ 6.11 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 2

ตารางที่ 6.2 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของคลาส สำหรับโปรแกรมทดสอบที่ 2

ชื่อคลาส	เครื่องมือวัด	ระดับความลึกของการสืบทอดฯ	จำนวนเมทอด	จำนวนบรรทัด	จำนวนสเตทเมนต์	จำนวนตัวแปรอินสแตนท์	LCOM
PlotApplication							
	MTOOP	1	8	86	55	1	0.85714
	JMetric	1	8	86	55	-	0.86
	JavaNCSS	-	8	97	-	-	-

MTOOP -- Metrics Tool for Object-Oriented Programs

Current Selection: PlotApplication

Method	Parameters	LOC	NOS	V(G)	Local Var.	CBO
Constructo...	0	1	0	1	0	0
Constructo...	1	11	9	4	1	5
_about ( )	0	3	1	1	1	1
_close ( )	0	2	1	1	0	0
_help ( )	0	3	1	1	1	1
_parseArg...	1	45	33	13	11	5
_usage ( )	0	12	7	3	4	3
main ( arra...	1	4	2	2	1	4

JMetric - Java Metrics Analyser

Current Selection: PlotApplication

Method	LOC	STMT	v(G)	Parameters	Local Var's	Var's Used
Constructo...	1	0	1	0	0	0
Constructo...	11	9	3	1	1	0
_about ( )	3	1	1	0	1	0
_close ( )	2	1	1	0	0	0
_help ( )	3	1	1	0	1	0
_parseArg...	45	33	12	1	11	1
_usage ( )	12	7	3	0	4	0
main ( arr...	4	2	2	1	1	1

JavaNCSS:

File Help

Mr. NCSS CCN JWDC Function

50	2	1	1	ptolemy.plot.PlotApplication.PlotApplication()
51	2	1	0	ptolemy.plot.PlotApplication.WindowAdapter.windowClosing(WindowEvent)
52	21	9	1	ptolemy.plot.PlotApplication.PlotApplication(String[])
53	6	3	1	ptolemy.plot.PlotApplication.main(String[])
54	3	1	1	ptolemy.plot.PlotApplication._about()
55	2	1	1	ptolemy.plot.PlotApplication._close()
56	3	1	1	ptolemy.plot.PlotApplication._help()
57	46	12	1	ptolemy.plot.PlotApplication._parseArgs(String[])
58	12	3	1	ptolemy.plot.PlotApplication._usage()
59	5	1	0	ptolemy.plot.PlotBox.PlotBox()
60	3	1	1	ptolemy.plot.PlotBox.addLegend(int, String)

รูปที่ 6.12 ค่าตัววัดของเมทอดในโปรเจก สำหรับโปรแกรมทดสอบที่ 2

ตารางที่ 6.3 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของเมทรูด สำหรับโปรแกรมทดสอบที่ 2

ชื่อเมทรูด	เครื่องมือวัด	จำนวนพารามิเตอร์	จำนวนบรรทัด	จำนวนสเตทเมนต์	V(G)	จำนวนตัวแปรเมทรูด	CBO
<code>_parseArg(String[])</code>							
	MTOOP	1	45	33	13	11	5
	JMetric	1	45	33	12	11	-
	JavaNCSS	-	46	-	12	-	-

	MTOOP	JavaNCSS
1. <code>/** Parse the command-line</code>		
2. <code>*/</code>		
3. <code>protected int _parseArgs(String args[]) throws CmdLineArgException,</code>	1	1
4. <code>FileNotFoundException, IOException {</code>		
5. <code>int i = 0, j, argsread;</code>	2,3,4	2
6. <code>String arg;</code>	5	3
7. <code>String title = "Ptolemy plot";</code>	6	4
8. <code>int width = 400; // Default width of the graph</code>	7	5
9. <code>int height = 300; // Default height of the graph</code>	8	6
10. <code></code>		
11. <code>while (i &lt; args.length) {</code>	9	7
12. <code>arg = args[i++];</code>	10	8
13. <code></code>		
14. <code>if (arg.equals("-help")) {</code>	11	9
15. <code>System.out.println(_usage());</code>	12	10
16. <code>continue;</code>	13	11
17. <code>} else if (arg.equals("-test")) {</code>	14	12,13
18. <code>_test = true;</code>	15	14
19. <code>continue;</code>	16	15
20. <code>} else if (arg.equals("-t")) {</code>	17	16,17
21. <code>title = args[i++];</code>	18	18
22. <code>continue;</code>	19	19
23. <code>} else if (arg.equals("-v"    arg.equals("-version")) {</code>	20	20,21
24. <code>_about();</code>	21	22
25. <code>continue;</code>	22	23
26. <code>} else if (arg.startsWith("=")) {</code>	23	24,25
27. <code>int xscreen = 1, yscreen = 1;</code>	24,25	26
28. <code>boolean screenlocationgiven = false;</code>	26	27
29. <code>StringTokenizer stoken =</code>	27	28
30. <code>new StringTokenizer(arg.substring(1, arg.length()),</code>		
31. <code>"=x-+");</code>		
32. <code>if (stoken.hasMoreTokens() {</code>	28	29
33. <code>width = (int)Integer.valueOf(stoken.nextToken()).</code>	29	30
34. <code>intValue();</code>		
35. <code>}</code>		
36. <code>if (stoken.hasMoreTokens() {</code>	30	31
37. <code>height = (int)Integer.valueOf(stoken.nextToken()).</code>	31	32
38. <code>intValue();</code>		
39. <code>}</code>		
40. <code>if (stoken.hasMoreTokens() {</code>	32	33
41. <code>xscreen = (int)Integer.valueOf(stoken.nextToken()).</code>	33	34
42. <code>intValue();</code>		
43. <code>screenlocationgiven = true;</code>	34	35
44. <code>}</code>		
45. <code>if (stoken.hasMoreTokens() {</code>	35	36
46. <code>yscreen = (int)Integer.valueOf(stoken.nextToken()).</code>	36	37
47. <code>intValue();</code>		
48. <code>screenlocationgiven = true;</code>	37	38
49. <code>}</code>		
50. <code>if (screenlocationgiven) {</code>	38	39
51. <code>setLocation(new Point(xscreen+1, yscreen+1));</code>	39	40
52. <code>}</code>		
53. <code>continue;</code>	40	41
54. <code>}</code>		
55. <code>}</code>		
56. <code></code>		
57. <code>setSize(width, height);</code>	41	42
58. <code>setTitle(title);</code>	42	43
59. <code>argsread = i++;</code>	43	44
60. <code>plot.parseArgs(args);</code>	44	45
61. <code>return argsread;</code>	45	46
62. <code>}</code>	46	

รูปที่ 6.13 ตัวอย่างการนับจำนวนบรรทัดของเมทรูด "`_parseArg(String[])`" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 2

### 6.2.2 ผลการทดสอบโปรแกรมที่ 3

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 3 มีขนาด 3.5 KLOC เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลค่าตัววัดด้วยเครื่องมือ MTOOP, JMetric และ JavaNCSS แล้วจะได้ผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภท คือ

1. ค่าตัววัดของแพ็คเกจในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.14 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของแพ็คเกจที่ชื่อ "Freenet.node" ได้ดังตารางที่ 6.4 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด เมื่อเทียบกับเครื่องมือวัด JavaNCSS มีค่าไม่เท่ากัน เป็นเพราะว่า โปรแกรมทดสอบที่ 3 นี้ มีการใช้คำสั่ง else if (...) เป็นจำนวนมาก ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด ดังรูปที่ 6.17 เป็นตัวอย่างการนับจำนวนบรรทัดของเมทอด "informRead(URLConnection c, Hashtable at)" ของโปรแกรมต้นฉบับทดสอบที่ 3
2. ค่าตัววัดของคลาสในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.15 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของคลาสที่ชื่อ "Data" ได้ดังตารางที่ 6.5 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก
3. ค่าตัววัดของเมทอดในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.16 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของเมทอดที่ชื่อ "informRead(URLConnection, Hashtable)" ได้ดังตารางที่ 6.6 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก

The image contains two screenshots of software tools. The top screenshot is titled "MTOOP -- Metrics Tool for Object-Oriented Programs". It shows a tree view on the left with a selected package "Freenet.support". The main window displays a table with the following data:

Package	NOC	NOM	WMC	LOC	NOS	Instance Var.
Freenet	34	151	4.44	851	463	77
Freenet.cli...	19	54	2.84	783	503	62
Freenet.co...	3	8	2.67	179	118	21
Freenet.crypt	1	22	22.00	238	188	8
Freenet.m...	17	84	4.94	539	316	34
Freenet.no...	9	51	5.67	387	247	21
Freenet.su...	17	70	4.12	440	269	38

The bottom screenshot is titled "JMetric - Java Metrics Analyser". It shows a tree view on the left with a selected package "Freenet.support". The main window displays a table with the following data:

Package	Lines of Co...	Statements	Classes	Methods
Freenet	851	463	34	151
Freenet.cli...	783	503	19	54
Freenet.co...	179	118	3	8
Freenet.crypt	238	188	1	22
Freenet.m...	539	316	17	84
Freenet.no...	387	247	9	51
Freenet.su...	440	269	17	70

รูปที่ 6.14 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 3



Nr.	Classes	Functions	NCSS	Package
1	34	151	1049	Freenet
2	19	54	971	Freenet.client
3	3	8	224	Freenet.contrib.fproxy
4	1	22	234	Freenet.crypt
5	17	84	734	Freenet.message
6	9	51	444	Freenet.node
7	17	70	506	Freenet.support
-----				
	100	440	4162	Total
-----				
Packages	Classes	Functions	NCSS	per
-----				
7.00	100.00	440.00	4,162.00	Project
	14.29	62.86	594.57	Package
		4.40	41.62	Class
			9.46	Function

รูปที่ 6.14 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 3 (ต่อ)

ตารางที่ 6.4 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของแพ็คเกจ สำหรับ โปรแกรมทดสอบที่ 3

ชื่อแพ็คเกจ	เครื่องมือวัด	จำนวนคลาส	จำนวนเมทอด	จำนวนเมทอดต่อคลาส	จำนวนบรรทัด	จำนวนสเตตเมนต์	จำนวนตัวแปรอินสแตนท์
Freenet.node							
	MTOOP	9	51	5.67	387	247	21
	JMetric	9	51	-	387	247	-
	JavaNCSS	9	51	-	444	-	-

**MTOOP -- Metrics Tool for Object-Oriented Programs**

Class	DIT	NOM	LOC	NOS	Instance Var.	LCOM
Data	0	11	53	30	4	0.75000
DataCB	0	2	6	2	1	1.00000
DataNotRe...	1	0	1	0	0	0.00000
DataStore...	0	7	15	7	0	0.00000
DataStoreIt...	0	2	10	4	3	0.00000
LimitedHa...	1	4	30	20	2	0.33333
Node	1	7	122	93	3	1.00000
StandardD...	0	14	110	66	5	0.80000
StandardM...	0	4	40	25	3	0.66667

**JMetric - Java Metrics Analyser**

Class	LOC	STMT	LCOM	MTDS	DIT	Pub MTDS
Data	53	30	0.75	11	0	9
DataCB	6	2	1.00	2	0	2
DataNotRe...	1	0	0.00	0	1	0
DataStore...	15	7	0.00	7	0	7
DataStoreIt...	10	4	0.00	2	0	2
LimitedHa...	30	20	0.33	4	1	4
Node	122	93	1.00	7	1	4
DataStore...	110	66	0.80	14	0	12
StandardM...	40	25	0.67	4	0	3

รูปที่ 6.15 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 3

Nr.	NCSS	Functions	Classes	Javadocs	Class
74	56	11	0	9	Freenet.node.Data
75	6	2	0	1	Freenet.node.DataCB
76	1	0	0	1	Freenet.node.DataNotReadyException
77	8	7	0	6	Freenet.node.DataStore
78	141	7	0	3	Freenet.node.Node
79	10	2	0	0	Freenet.node.DataStoreItem
80	119	14	0	8	Freenet.node.StandardDataStore
81	31	4	0	2	Freenet.node.LimitedHashtable
82	43	4	0	3	Freenet.node.StandardMessageHandler
83	43	4	0	1	Freenet.support.BinaryTree
84	53	6	0	0	Freenet.support.Branch

รูปที่ 6.15 ค่าตัววัดของคลาสในโปรเจค สำหรับโปรแกรมทดสอบที่ 3 (ต่อ)

ตารางที่ 6.5 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของคลาส สำหรับโปรแกรมทดสอบที่ 3

ชื่อคลาส	เครื่องมือวัด	ระดับความลึกของการสืบทอดๆ	จำนวนเมธอด	จำนวนบรรทัด	จำนวนสเตทเมนต์	จำนวนตัวแปรอินสแตนท์	LCOM
Data							
	MTOOP	0	11	53	30	4	0.75000
	JMetric	0	11	53	30	-	0.75
	JavaNCSS	-	11	56	-	-	-

Method	Parameters	LOC	NOS	V(G)	Local Var.	CBO
Constructo...	4	3	2	1	0	4
inform (Ha...	1	8	6	1	1	3
informRea...	2	17	10	7	6	7
informWrit...	1	6	4	2	1	2
main (Stri...	1	58	47	18	10	11
usage ( )	0	24	23	1	0	0
version ( )	0	2	1	1	0	0

Method	LOC	STMT	v(G)	Parameters	Local Var's	Var's Used
Constructo...	3	2	1	4	0	1
inform ( H...	8	6	1	1	1	0
informRea...	17	10	7	2	6	1
informWrit...	6	4	2	1	1	0
main ( Stri...	58	47	16	1	10	1
usage ( )	24	23	1	0	0	0
version ( )	2	1	1	0	0	0

Nr.	NCSS	CCN	JVDC	Function
353	73	23	0	Freenet.node.Node.main(String[])
354	6	2	0	Freenet.node.Node.informWrite(URLConnection)
355	18	7	0	Freenet.node.Node.informRead(URLConnection,Hashtable)
356	10	2	0	Freenet.node.Node.inform(Hashtable)
357	2	1	1	Freenet.node.Node.version()
358	24	1	1	Freenet.node.Node.usage()
359	4	1	0	Freenet.node.Node.Node(long,int,int,ListeningAddress)
360	15	3	0	Freenet.node.StandardDataStore.makeDataStore(Node,long,int)

รูปที่ 6.16 ค่าตัววัดของเมธอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 3

ตารางที่ 6.6 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของเมทรูด สำหรับโปรแกรมทดสอบที่ 3

ชื่อเมทรูด	เครื่องมือวัด	จำนวนพารามิเตอร์	จำนวนบรรทัด	จำนวนสเตทเมนต์	V(G)	จำนวนตัวแปรเมทรูด	CBO
informRead(URLConnection, Hashtable)							
	MTOOP	2	17	10	7	6	7
	JMetric	2	17	10	7	6	-
	JavaNCSS	-	18	-	7	-	-

	MTOOP	JavaNCSS
1. private static final void informRead(URLConnection c, Hashtable at) throws IOException	1	1
2. {		
3. if (params.getParam("informRead").equalsIgnoreCase("yes"))	2	2
4. {		
5. InputStreamReader ir = new InputStreamReader(c.getInputStream());	3	3
6. BufferedReader br = new BufferedReader(ir);	4	4
7. SHA1 sha=new SHA1(true);	5	5
8. while(br.ready())	6	6
9. {		
10. String addrstr=br.readLine();	7	7
11. addrstr=addrstr.trim();	8	8
12. if(addrstr.startsWith(BUILD))	9	
13. {		9
14. String latestbuild = addrstr.substring(BUILD.length());	10	10
15. if (buildNumber.compareTo(latestbuild) < 0)	11	11
16. System.err.println ( "Newer build "+latestbuild+" is available than this build "+buildNumber+" , please upgrade!");	12	12
17. }		
18. else if(!addrstr.equals(""))	13	13,14
19. {		
20. StringKey tname = new StringKey(sha.doHash(addrstr));	14	15
21. if(n.ds.searchRef(tname)==null)	15	16
22. at.put(addrstr, tname);	16	17
23. }		
24. }		
25. ir.close();	<u>17</u>	<u>18</u>
26. }		
27. }		

รูปที่ 6.17 ตัวอย่างการนับจำนวนบรรทัดของเมทรูด "informRead(URLConnection, Hashtable)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 3

### 6.2.3 ผลการทดสอบโปรแกรมที่ 4

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 4 มีขนาด 11 KLOC เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลค่าตัววัดด้วยเครื่องมือ MTOOP, JMetric และ JavaNCSS แล้วจะได้ผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภท คือ

1. ค่าตัววัดของแพ็คเกจในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.18 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของแพ็คเกจที่ชื่อ "org.openxml" ได้ดังตารางที่ 6.7 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด เมื่อเทียบกับเครื่องมือวัด JavaNCSS มีค่าไม่เท่ากัน เป็นเพราะว่า โปรแกรมทดสอบที่ 4 นี้ มีการใช้คำสั่ง else if (...) เป็นจำนวนมาก ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด ดังรูปที่ 6.21 เป็นตัวอย่างการนับจำนวนบรรทัดของเมทอด "of(Document)" ของโปรแกรมต้นฉบับทดสอบที่ 4
2. ค่าตัววัดของคลาสในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.19 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของคลาสที่ชื่อ "XMLBookmarks" ได้ดังตารางที่ 6.8 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก
3. ค่าตัววัดของเมทอดในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.20 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของเมทอดที่ชื่อ "of(Document)" ได้ดังตารางที่ 6.9 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก

Package	NOC	NOM	WMC	LOC	NOS	Instance Var.
org.openxml	7	38	5.43	228	148	13
org.openxml.dom	4	30	7.50	126	74	3
org.openxml.dom.ext	23	280	12.17	1541	1046	63
org.openxml.dom.html	1	4	4.00	9	4	0
org.openxml.dom.iterator	57	650	11.40	1953	1117	27
org.openxml.io	7	25	3.57	195	139	12
org.openxml.parser	8	98	12.25	828	611	52
org.openxml.source	11	104	9.45	1535	1246	61
org.openxml.source.holder	5	28	5.60	64	28	3
org.openxml.util	15	75	5.00	522	358	33
org.openxml.x3p	6	63	10.50	549	383	36
org.openxml.x3p.helpers	14	91	6.50	404	216	60
org.openxml.x3p.processor	3	28	9.33	171	114	14
org.openxml.x3p.publisher	9	54	6.00	366	250	12
org.w3c.dom	10	78	7.80	786	609	25
org.w3c.dom.fl	18	72	4.00	185	72	23
org.w3c.dom.html	3	4	1.33	16	4	7
org.xml.sax	56	584	10.43	1224	584	0
org.xml.sax.helpers	11	73	6.64	172	77	11
org.xml.sax.impl	3	24	8.00	74	36	7

Package	Lines of Co...	Statements	Classes	Methods
org.openxml	228	148	7	38
org.openxml.dom	126	74	4	30
org.openxml.dom.ext	1541	1046	23	280
org.openxml.dom.html	9	4	1	4
org.openxml.dom.iterator	1953	1117	57	650
org.openxml.io	195	139	7	25
org.openxml.parser	828	611	8	98
org.openxml.source	1535	1246	11	104
org.openxml.source.holder	64	28	5	28
org.openxml.util	522	358	15	75
org.openxml.x3p	549	383	6	63
org.openxml.x3p.helpers	404	216	14	91
org.openxml.x3p.processor	171	114	3	28
org.openxml.x3p.publisher	366	250	9	54
org.w3c.dom	786	609	10	78
org.w3c.dom.fl	185	72	18	72
org.w3c.dom.html	16	4	3	4
org.xml.sax	1224	584	56	584
org.xml.sax.helpers	172	77	11	73
org.xml.sax.impl	74	36	3	24

รูปที่ 6.18 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 4

Nr.	Classes	Functions	NCSS	Package
1	7	38	289	org.openxml
2	4	30	155	org.openxml.beans
3	23	280	1687	org.openxml.dom
4	1	4	6	org.openxml.dom.ext
5	57	650	2266	org.openxml.dom.html
6	7	25	238	org.openxml.dom.iterator
7	8	98	932	org.openxml.io
8	11	104	1818	org.openxml.parser
-----				
	118	1229	7391	Total
-----				
Packages	Classes	Functions	NCSS	per
8.00	118.00	1,229.00	7,391.00	Project
	14.75	153.63	923.88	Package
		10.42	62.64	Class
			6.01	Function

รูปที่ 6.18 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 4 (ต่อ)

ตารางที่ 6.7 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของแพ็คเกจ สำหรับ โปรแกรมทดสอบที่ 4

ชื่อแพ็คเกจ	เครื่องมือวัด	จำนวนคลาส	จำนวนเมทอด	จำนวนเมทอดต่อคลาส	จำนวนบรรทัด	จำนวนสเตตเมนต์	จำนวนตัวแปรอินสแตนท์
org.openxml							
	MTOOP	7	38	5.43	228	148	13
	JMetric	7	38	-	228	148	-
	JavaNCSS	7	38	-	289	-	-

Class	DIT	NOM	LOC	NOS	Instance Var.	LCOM
DOMFactory	0	17	137	98	3	0.96528
DTDDocu...	1	4	13	6	0	0.00000
XMLBookm...	0	7	51	34	3	0.55556
XMLCollect...	1	2	3	0	0	0.00000
XMLDocu...	1	3	12	6	0	0.00000
XMLElement	1	4	9	3	1	0.66667
XMLEleme...	0	1	3	1	0	0.00000

Class	LOC	STMT	LCOM	MTDS	DIT	Pub MTDS
DOMFactory	137	98	0.97	17	0	17
DTDDocu...	13	6	0.00	4	1	4
XMLBookm...	51	34	0.66	7	0	6
XMLCollect...	3	0	0.00	2	1	2
XMLDocu...	12	6	0.00	3	1	3
XMLElement	9	3	0.67	4	1	4
XMLEleme...	3	1	0.00	1	0	1

Nr.	NCSS	Functions	Classes	Javadocs	Class
1	159	17	0	17	org.openxml.DOMFactory
2	15	4	0	1	org.openxml.DTDDocument
3	54	7	0	1	org.openxml.XMLBookmarks
4	5	2	0	3	org.openxml.XMLCollection
5	13	3	0	2	org.openxml.XMLDocument
6	9	4	0	5	org.openxml.XMLElement
7	2	1	0	2	org.openxml.XMLElementFactory
Average Object NCSS: 36.71					
Average Object Functions: 5.43					
Average Object Inner Classes: 0.00					
Average Object Javadoc Comments: 4.43					
Program NCSS: 289					

รูปที่ 6.19 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 4

ตารางที่ 6.8 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของคลาส สำหรับโปรแกรมทดสอบที่ 4

ชื่อคลาส	เครื่องมือวัด	ระดับความลึกของการสืบทอดฯ	จำนวนเมทอด	จำนวนบรรทัด	จำนวนสเตทเมนต์	จำนวนตัวแปรอินสแตนท์	LCOM
XMLBookmarks							
	MTOOP	0	7	51	34	3	0.55556
	JMetric	0	7	51	34	-	0.56
	JavaNCSS	-	7	54	-	-	-

MTOOP -- Metrics Tool for Object-Oriented Programs

Current Selection: XMLBookmarks

Method	Parameters	LOC	NOS	V(G)	Local Var.	CBO
Constructo...	1	5	4	2	0	2
get ( String )	1	7	5	3	1	4
getDocum...	0	2	1	1	0	1
list ( )	0	11	7	2	3	4
of ( Docum...	1	10	8	3	1	3
set ( String,...	2	4	3	2	0	4
setUnique ...	2	8	6	4	1	5

JMetric - Java Metrics Analyser

Current Selection: XMLBookmarks

Method	LOC	STMT	v(G)	Parameters	Local Var's	Var's Used
Constructo...	5	4	2	1	0	2
get ( Strin...	7	5	2	1	1	2
getDocum...	2	1	1	0	0	1
list ( )	11	7	2	0	3	1
of ( Docu...	10	8	3	1	1	1
set ( Strin...	4	3	2	2	0	2
setUnique ...	8	6	3	2	1	2

JavaNCSS:

File Help

Packages Classes Methods

ส., ค.ศ. 10, 2001 16:29:31 Asia/Bangkok

Nr.	NCSS	CCN	JVDC	Function
22	5	3	0	org.openxml.XMLBookmarks.set(String,Element)
23	9	5	0	org.openxml.XMLBookmarks.setUnique(String,Element)
24	7	2	0	org.openxml.XMLBookmarks.get(String)
25	11	2	0	org.openxml.XMLBookmarks.list()
26	2	1	0	org.openxml.XMLBookmarks.getDocument()
27	11	3	0	org.openxml.XMLBookmarks.of(Document)
28	5	3	0	org.openxml.XMLBookmarks.XMLBookmarks(Document)

รูปที่ 6.20 ค่าตัววัดของเมทอดในโปรเจค สำหรับโปรแกรมทดสอบที่ 4

ตารางที่ 6.9 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของเมทรีค สำหรับโปรแกรมทดสอบที่ 4

ชื่อเมทรีค	เครื่องมือวัด	จำนวนพารามิเตอร์	จำนวนบรรทัด	จำนวนสเตทเมนต์	V(G)	จำนวนตัวแปรเมทรีค	CBO
of(Document)							
	MTOOP	1	10	8	3	1	3
	JMetric	1	10	8	3	1	-
	JavaNCSS	-	11	-	3	-	-

	MTOOP	JavaNCSS
1. public static XMLBookmarks of( Document doc )	1	1
2. {	2	2
3. XMLBookmarks bookmarks;	3	3
4. if ( _allBookmarks == null )	4	4
5. {	5	5
6. _allBookmark = new Hashtable();		6
7. bookmarks = null;		7
8. }		8
9. else		9
10. bookmarks = (XMLBookmarks) _allBookmarks.get( doc );		10
11. if ( bookmarks == null )		11
12. {		
13. bookmarks = new XMLBookmarks( doc );		
14. _allBookmarks.put( doc, bookmarks );		
15. }		
16. return bookmarks;		
17. }		
18. }		
19. }		
20. }		
	<u>10</u>	<u>11</u>

รูปที่ 6.21 ตัวอย่างการนับจำนวนบรรทัดของเมทรีค "of(Document)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 4

สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

6.2.4 ผลการทดสอบโปรแกรมที่ 5

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 5 มีขนาด 16.5 KLOC เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลค่าตัววัดด้วยเครื่องมือ MTOOP, JMetric และ JavaNCSS แล้วจะได้ผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภท คือ

1. ค่าตัววัดของแพ็คเกจในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.22 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของแพ็คเกจที่ชื่อ "org.gjt.sp.jedit.browser" ได้ดังตารางที่ 6.10 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด เมื่อเทียบกับเครื่องมือวัด JavaNCSS มีค่าไม่เท่ากัน เป็นเพราะว่าโปรแกรมทดสอบที่ 5 นี้ มีการใช้คำสั่ง else if (...) เป็นจำนวนมาก ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด
2. ค่าตัววัดของคลาสในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.23 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของคลาสที่ชื่อ "BrowserIORequest" ได้ดังตารางที่ 6.11 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็คเกจในโปรเจก
3. ค่าตัววัดของเมทอดในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.24 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของเมทอดที่ชื่อ "reloadDirectory(String)" ได้ดังตารางที่ 6.12 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด และค่าของไซโคลเมติกของแมคเคบ มีสาเหตุดังนี้
  - 3.1 การนับจำนวนบรรทัด คำสั่ง else if (...) เครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด
  - 3.2 การหาค่าของไซโคลเมติกของแมคเคบของ MTOOP ตามสูตรการคำนวณในหัวข้อ 3.4.4.4 สูตรที่ 4 จะมีการนับจำนวนโหนดของ ConditionalOrExpression() และ ConditionalAndExpression() แต่ของ JMetric และ JavaNCSS จะไม่ได้นับ ดังรูปที่ 6.25 เป็นตัวอย่างการนับจำนวนไซโคลเมติกของแมคเคบของเมทอด "reloadDirectory(String)" ของโปรแกรมต้นฉบับทดสอบที่ 5

Current Element : Project						
Package	NOC	NOM	WMC	LOC	NOS	Instance Var.
org.gjt.sp.jedit	29	511	17.62	5220	3441	166
org.gjt.sp.jedit.browser	7	66	9.43	1061	708	66
org.gjt.sp.jedit.gui	34	186	5.47	2664	1797	136
org.gjt.sp.jedit.io	7	84	12.00	673	425	37
org.gjt.sp.jedit.msg	11	28	2.55	94	34	21
org.gjt.sp.jedit.options	26	128	4.92	1895	1252	154
org.gjt.sp.jedit.proto.jedit...	2	5	2.50	44	29	3
org.gjt.sp.jedit.search	14	97	6.93	1070	726	59
org.gjt.sp.jedit.syntax	8	89	11.13	1102	758	102
org.gjt.sp.jedit.textarea	5	240	48.00	2353	1501	85
org.gjt.sp.util	5	44	8.80	324	195	34

รูปที่ 6.22 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 5



Package	Lines of Co...	Statements	Classes	Methods
org.gjt.sp.jedit	5220	3441	29	511
org.gjt.sp.jedit.browser	1061	708	7	66
org.gjt.sp.jedit.gui	2664	1797	34	186
org.gjt.sp.jedit.io	673	425	7	84
org.gjt.sp.jedit.msg	94	34	11	28
org.gjt.sp.jedit.options	1895	1252	26	128
org.gjt.sp.jedit.proto.jeditres...	44	29	2	5
org.gjt.sp.jedit.search	1070	726	14	97
org.gjt.sp.jedit.syntax	1102	758	8	89
org.gjt.sp.jedit.textarea	2353	1501	5	240
org.gjt.sp.util	324	195	5	44

Nr.	Classes	Functions	NCSS	Package
1	29	507	5768	org.gjt.sp.jedit
2	7	66	1208	org.gjt.sp.jedit.browser
3	34	183	3051	org.gjt.sp.jedit.gui
4	7	83	806	org.gjt.sp.jedit.io
5	11	28	136	org.gjt.sp.jedit.msg
6	26	128	2071	org.gjt.sp.jedit.options
7	2	5	60	org.gjt.sp.jedit.proto.jeditresource
8	14	97	1220	org.gjt.sp.jedit.search
-----				
	130	1097	14320	Total
-----				
	Packages	Classes	Functions	NCSS
	8.00	130.00	1,097.00	14,320.00
		16.25	137.13	1,790.00
			8.44	110.15
				13.05

รูปที่ 6.22 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 5 (ต่อ)

ตารางที่ 6.10 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของแพ็คเกจ สำหรับโปรแกรมทดสอบที่ 5

ชื่อแพ็คเกจ	เครื่องมือวัด	จำนวนคลาส	จำนวนเมทอด	จำนวนเมทอดต่อคลาส	จำนวนบรรทัด	จำนวนสเตตเมนต์	จำนวนตัวแปรอินสแตนท์
org.gjt.sp.jedit.browser							
	MTOOP	7	66	9.43	1061	708	66
	JMetric	7	66	-	1061	708	-
	JavaNCSS	7	66	-	1208	-	-

Current Selection: org.gjt.sp.jedit.brows...

Class	DIT	NOM	LOC	NOS	Instance Var.	LCOM
BrowserIO...	1	7	85	56	10	0.65000
BrowserLi...	0	2	5	2	0	0.00000
BrowserPo...	1	2	108	78	3	1.66667
BrowserVi...	1	10	243	169	10	0.83333
FileCellRe...	0	3	59	38	12	0.58333
VFSBrowser	1	36	426	274	25	0.96457
VFSFileCh...	1	6	135	91	6	0.80000

Current Selectio... P org.gjt.sp.jedit.brow...

Class	LOC	STMT	LCOM	MTDS	DIT	Pub MTDS
BrowserIO...	85	56	0.65	7	1	3
BrowserLi...	5	2	0.00	2	0	0
BrowserPo...	108	78	1.67	2	1	1
BrowserVi...	243	169	0.83	10	1	7
FileCellRe...	59	38	0.58	3	0	2
VFSBrowser	426	274	0.96	36	1	25
VFSFileCh...	135	91	0.80	6	1	5

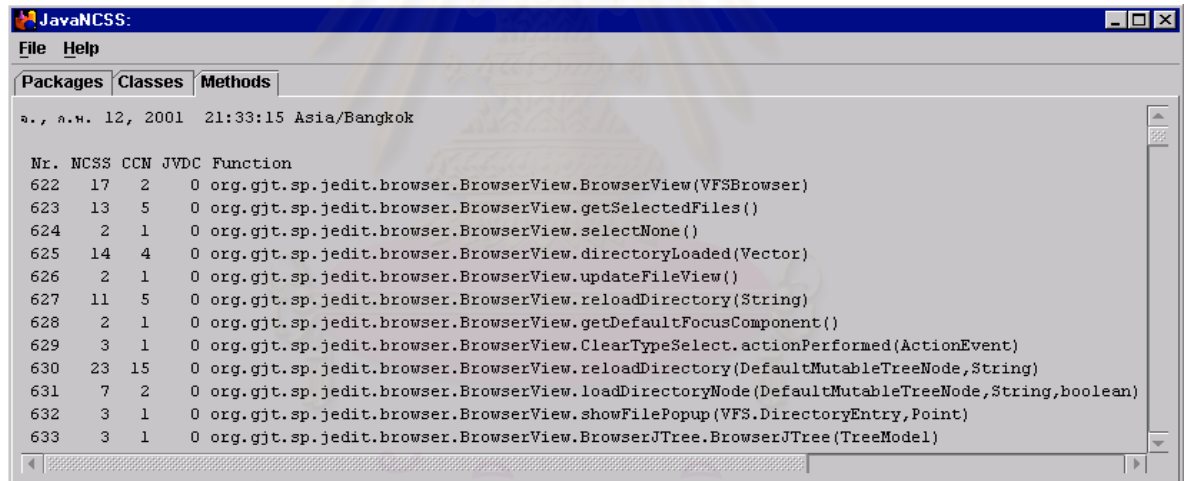
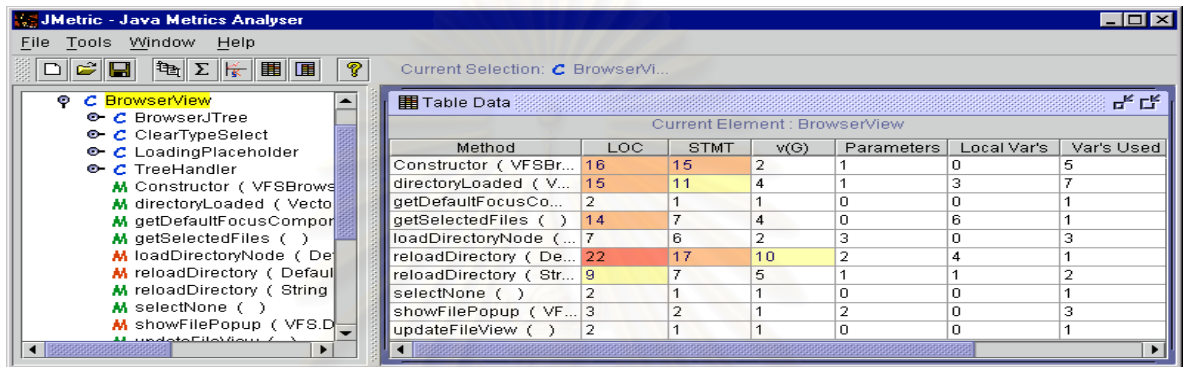
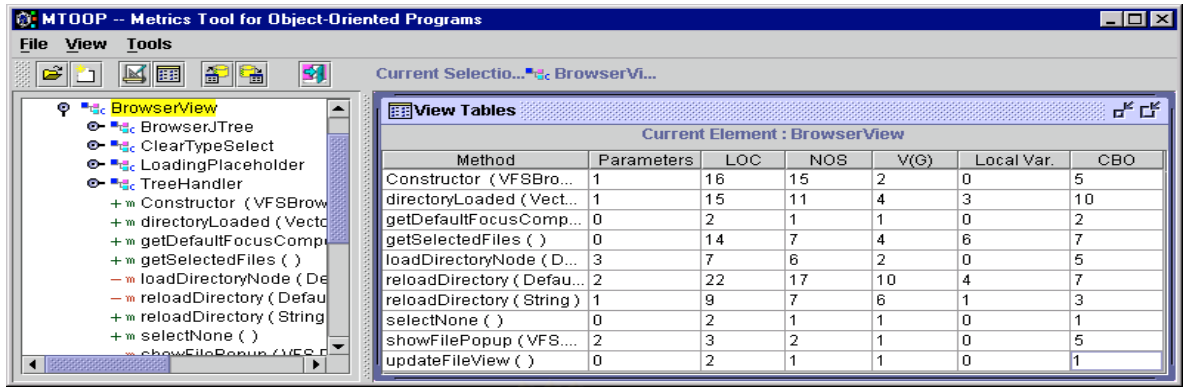
File Help

Mr.	NCSS	Functions	Classes	Javadocs	Class
30	118	7	0	2	org.gjt.sp.jedit.browser.BrowserIORequest
31	3	2	0	3	org.gjt.sp.jedit.browser.BrowserListener
32	123	2	1	1	org.gjt.sp.jedit.browser.BrowserPopupMenu
33	250	10	4	1	org.gjt.sp.jedit.browser.BrowserView
34	64	3	0	0	org.gjt.sp.jedit.browser.FileCellRenderex
35	447	36	5	3	org.gjt.sp.jedit.browser.VFSBrowser
36	142	6	3	1	org.gjt.sp.jedit.browser.VFSFileChooserDialog
37	39	3	1	0	org.gjt.sp.jedit.gui.AboutDialog

รูปที่ 6.23 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 5

ตารางที่ 6.11 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของคลาส สำหรับโปรแกรมทดสอบที่ 5

ชื่อคลาส	เครื่องมือวัด	ระดับความลึกของการสืบทอดฯ	จำนวนเมทอด	จำนวนบรรทัด	จำนวนสแตกเมนต์	จำนวนตัวแปรอินสแตนซ์	LCOM
BrowserIORequest							
	MTOOP	1	7	85	56	10	0.65000
	JMetric	1	7	85	56	-	0.65
	JavaNCSS	-	7	118	-	-	-



รูปที่ 6.24 ค่าตัววัดของเมทร็อคในโปรเจก สำหรับโปรแกรมทดสอบที่ 5

ตารางที่ 6.12 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของเมทร็อค สำหรับโปรแกรมทดสอบที่ 5

ชื่อเมทร็อค	เครื่องมือวัด	จำนวนพารามิเตอร์	จำนวนบรรทัด	จำนวนสเตทเมนต์	V(G)	จำนวนตัวแปรเมทร็อค	CBO
reloadDirectory(String)							
	MTOOP	1	9	7	6	1	3
	JMetric	1	9	7	5	1	-
	JavaNCSS	-	11	-	5	-	-

	MTOOP		JavaNCSS	
	LOC	V(G)	LOC	V(G)
1. public void reloadDirectory(String path)	1		1	
2. {				
3. // because this method is called for *every* VFS update,				
4. // we don't want to scan the tree all the time. So we				
5. // use the following algorithm to determine if the path				
6. // might be part of the tree:				
7. // - if the path starts with the browser's current directory,				
8. // we do the tree scan				
9. // - if the browser's directory is 'favorites:' -- we have to				
10. // do the tree scan, as every path can appear under the				
11. // favorites list				
12. // - if the browser's directory is 'roots:' and path is on				
13. // the local filesystem, do a tree scan				
14. String browserDir = browser.getDirectory();	2		2	
15. if(browserDir.startsWith(FavoritesVFS.PROTOCOL))	3	1	3	1
16. reloadDirectory(rootNode,path);	4		4	
17. else if(browserDir.startsWith(FileRootsVFS.PROTOCOL))	5	2	5,6	2
18. {				
19. if(!MiscUtilities.isURL(path)	6	3,4	7	3
20. MiscUtilities.getProtocolOfURL(path).equals("file"))				
21. reloadDirectory(rootNode,path);	7		8	
22. } else if(path.startsWith(browserDir))	8	<u>5+1</u>	9,10	<u>4+1</u>
23. reloadDirectory(rootNode,path);	<u>9</u>		<u>11</u>	
24. }				
25. }				

รูปที่ 6.25 ตัวอย่างการนับจำนวนบรรทัดและการหาค่าไซโคลเมตริกของเมคเคบของเมทอด "reloadDirectory (String)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมต้นฉบับทดสอบที่ 5

6.2.5 ผลการทดสอบโปรแกรมที่ 6

โปรแกรมต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่ 6 มีขนาด 12 KLOC เมื่อนำโปรแกรมต้นฉบับนี้เข้าไปประมวลค่าตัววัดด้วยเครื่องมือ MTOOP, JMetric และ JavaNCSS แล้วจะได้ผลลัพธ์ของค่าตัววัดที่ได้ในแต่ละประเภท คือ

- ค่าตัววัดของแพ็กเกจในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.26 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของแพ็กเกจที่ชื่อ "org.jext" ได้ดังตารางที่ 6.13 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด เมื่อเทียบกับเครื่องมือวัด JavaNCSS มีค่าไม่เท่ากัน เป็นเพราะว่า โปรแกรมทดสอบที่ 5 นี้ มีการใช้คำสั่ง else if (...) เป็นจำนวนมาก ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด
- ค่าตัววัดของคลาสในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.27 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของคลาสที่ชื่อ "GUIUtilities" ได้ดังตารางที่ 6.14 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด ซึ่งมีเหตุผลเดียวกันกับการหาค่าตัววัดของแพ็กเกจในโปรเจก

3. ค่าตัววัดของเมทรีคในโปรเจกของแต่ละเครื่องมือวัด จะได้ผลลัพธ์ดังรูปที่ 6.28 ซึ่งสามารถนำค่ามาสรุปเป็นตารางเพื่อเปรียบเทียบค่าของเมทรีคที่ชื่อ "loadMenu(String, boolean)" ได้ดังตารางที่ 6.15 ปรากฏว่าเครื่องมือวัด MTOOP สามารถหาค่าตัววัดต่าง ๆ ส่วนใหญ่ได้ค่าตรงกับเครื่องมือวัดอื่น ๆ ยกเว้นค่าของจำนวนบรรทัด และค่าของไซโคลเมตริกของแมคเคบ มีสาเหตุดังนี้

3.1 การนับจำนวนบรรทัด คำสั่ง else if (...) เครื่องมือวัด JavaNCSS จะนับเป็น 2 บรรทัด ซึ่งเครื่องมือวัด MTOOP และ JMetric จะนับเป็น 1 บรรทัด

3.2 การหาค่าของไซโคลเมตริกของแมคเคบของ MTOOP ตามสูตรการคำนวณในหัวข้อ 3.4.4.4 สูตรที่ 4 จะมีการนับจำนวนโหนดของ ConditionalOrExpression() และ ConditionalAndExpression() แต่ของ JMetric และ JavaNCSS จะไม่ได้นับ นอกจากนี้ JavaNCSS จะนับจำนวนโหนดของ return() และจะไม่มีการบวกด้วย 1 ดังรูปที่ 6.29 เป็นตัวอย่างการนับจำนวนไซโคลเมตริกของแมคเคบของเมทรีค " loadMenu(String,boolean)" ของโปรแกรมต้นฉบับทดสอบที่ 6

Package	NOC	NOM	VWMC	LOC	NOS	Instance Var.
org.jext	13	305	23.46	2664	1819	103
org.jext.actions	73	160	2.19	968	417	6
org.jext.console	4	55	13.75	677	464	50
org.jext.console.commands	11	33	3.00	140	77	12
org.jext.dawn	5	67	13.40	461	316	21
org.jext.dawn.javaccess	9	18	2.00	57	20	0
org.jext.dawn.array	4	8	2.00	82	63	0
org.jext.dawn.io	15	38	2.53	136	70	1
org.jext.dawn.math	7	16	2.29	179	110	1
org.jext.dawn.naming	3	6	2.00	194	160	0
org.jext.dawn.javaccess	21	42	2.00	123	43	0
org.jext.dawn.loop	4	8	2.00	86	66	0
org.jext.dawn.math	18	36	2.00	158	74	0
org.jext.dawn.stack	19	38	2.00	127	50	0
org.jext.dawn.string	20	40	2.00	195	97	0
org.jext.dawn.test	15	30	2.00	93	30	1
org.jext.dawn.util	2	6	3.00	37	10	19
org.jext.event	22	102	4.64	572	325	33
org.jext.gui	5	20	4.00	268	186	25
org.jext.menu	20	110	5.50	1878	1245	111
org.jext.misc	1	2	2.00	4	1	0
org.jext.protocol.jextresource	12	58	4.83	1004	668	75
org.jext.onclick	2	10	5.00	76	49	9
org.jext.options	2	10	5.00	71	49	3
org.jext.popup	2	5	2.50	36	22	1
org.jext.properties	15	30	2.00	115	40	0
org.jext.scripting	5	37	7.40	720	501	35
org.jext.search	5	10	2.00	125	63	7
org.jext.textarea	5	23	4.60	192	125	16
org.jext.toolbar	6	43	7.17	400	257	32
org.jext.xinsert						

Package	Lines of Co.	Statements	Classes	Methods
org.jext	2664	1819	13	305
org.jext.actions	968	417	73	160
org.jext.console	677	464	4	55
org.jext.console.commands	140	77	11	33
org.jext.dawn	461	316	5	67
org.jext.dawn.array	57	20	9	18
org.jext.dawn.err	82	63	4	8
org.jext.dawn.io	136	70	15	38
org.jext.dawn.math	179	110	7	15
org.jext.dawn.naming	194	160	3	6
org.jext.dawn.string	123	43	21	42
org.jext.dawn.test	86	66	4	8
org.jext.dawn.util	158	74	18	36
org.jext.event	127	50	19	38
org.jext.gui	185	87	20	40
org.jext.menu	93	30	15	30
org.jext.onclick	37	10	2	6
org.jext.options	572	325	22	102
org.jext.popup	268	186	5	20
org.jext.properties	1878	1245	20	110
org.jext.protocol.jextresource	4	1	1	2
org.jext.scripting	1004	668	12	58
org.jext.search	76	49	2	10
org.jext.textarea	71	49	2	10
org.jext.toolbar	36	22	2	5
org.jext.xinsert	115	40	15	30
org.jext.search	720	501	5	37
org.jext.textarea	125	63	5	10
org.jext.toolbar	192	125	5	23
org.jext.xinsert	400	257	6	43

รูปที่ 6. 26 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 6

Nr.	Classes	Functions	NCSS	Package
1	13	305	2924	org.jext
2	73	160	1409	org.jext.actions
3	4	55	720	org.jext.console
4	11	33	193	org.jext.console.commands
5	5	67	488	org.jext.dawn
6	9	18	92	org.jext.dawn.array
7	4	8	102	org.jext.dawn.err
8	15	38	204	org.jext.dawn.io
-----				
	134	684	6132	Total
-----				
Packages	Classes	Functions	NCSS	per
-----				
8.00	134.00	684.00	6,132.00	Project
	16.75	85.50	766.50	Package
		5.10	45.76	Class
			8.96	Function

รูปที่ 6.26 ค่าตัววัดของแพ็คเกจในโปรเจก สำหรับโปรแกรมทดสอบที่ 6 (ต่อ)

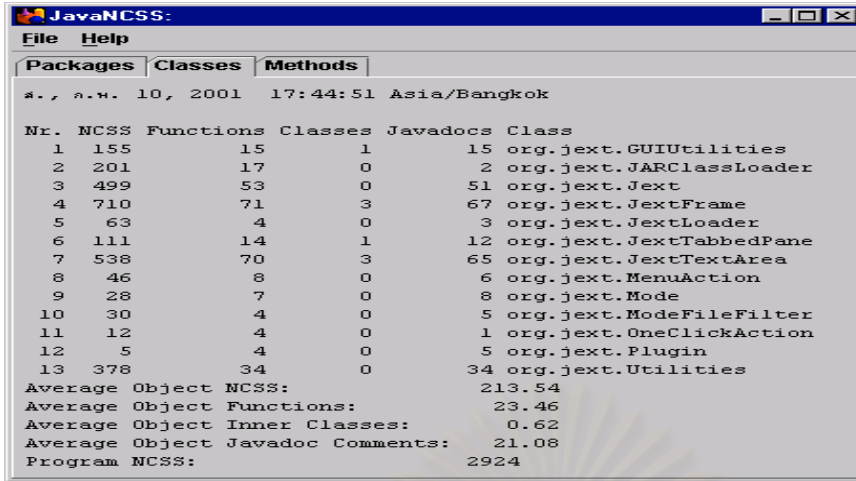
ตารางที่ 6.13 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่างๆ ของแพ็คเกจ สำหรับโปรแกรมทดสอบที่ 6

ชื่อแพ็คเกจ	เครื่องมือวัด	จำนวนคลาส	จำนวนเมทอด	จำนวนเมทอดต่อคลาส	จำนวนบรรทัด	จำนวนสเตตเมนต์	จำนวนตัวแปรอินสแตนท์
org.jext							
	MTOOP	13	305	23.46	2664	1819	103
	JMetric	13	305	-	2664	1819	-
	JavaNCSS	13	305	-	2924	-	-

Class	DIT	NOM	LOC	NOS	Instance Var.	LCOM
GUIUtilities	0	15	144	90	3	1.00000
JARClassL...	1	17	185	104	6	0.87500
Jext	0	53	488	340	24	0.96314
JextLoader	0	71	687	490	29	0.95813
JextTabbedPane	0	4	55	35	5	0.60000
JextTextArea	1	14	109	66	5	0.92308
JextTabbe...	1	30	509	358	16	0.94686
Mode	0	40	28	13	1	1.00000
ModeAction	0	7	27	13	4	0.91667
ModeFileFi...	1	4	25	13	3	0.66667
OneClickA...	1	4	10	4	1	1.00000
Plugin	0	9	9	4	0	0.00000
Utilities	0	34	376	274	4	1.01515

Class	LOC	STMT	LCOM	MTDS	DIT	Pub MTDS
GUIUtilities	144	90	1.00	15	0	15
JARClassL...	185	104	0.88	17	1	12
Jext	488	340	0.96	53	0	47
JextLoader	687	490	0.96	71	1	66
JextTabbedPane	55	35	0.60	4	0	2
JextTextArea	109	66	0.92	14	1	11
JextTabbe...	509	358	0.95	70	1	66
Mode	40	28	1.00	8	0	8
ModeAction	27	13	0.92	7	0	7
ModeFileFi...	25	13	0.67	4	1	4
OneClickA...	10	4	1.00	4	1	4
Plugin	9	4	0.00	4	0	4
Utilities	376	274	1.02	34	0	33

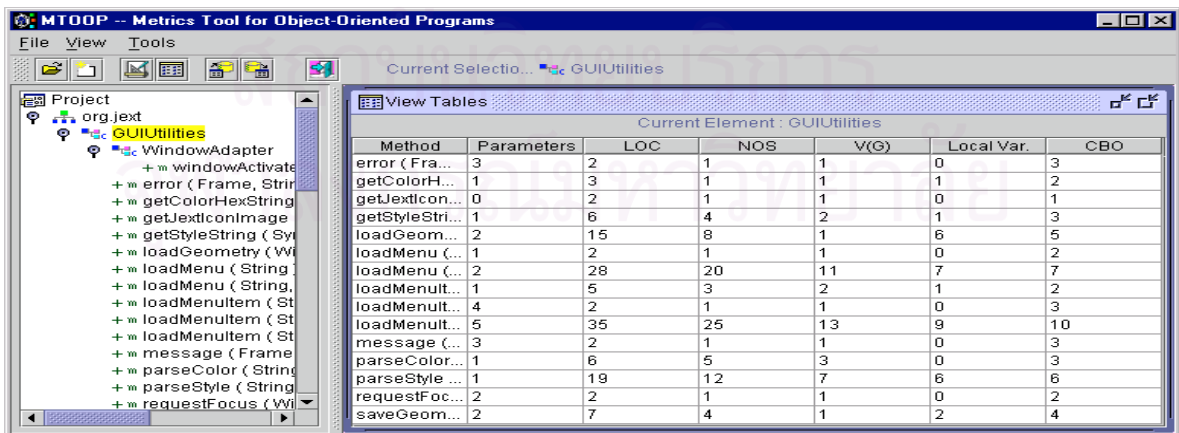
รูปที่ 6.27 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 6



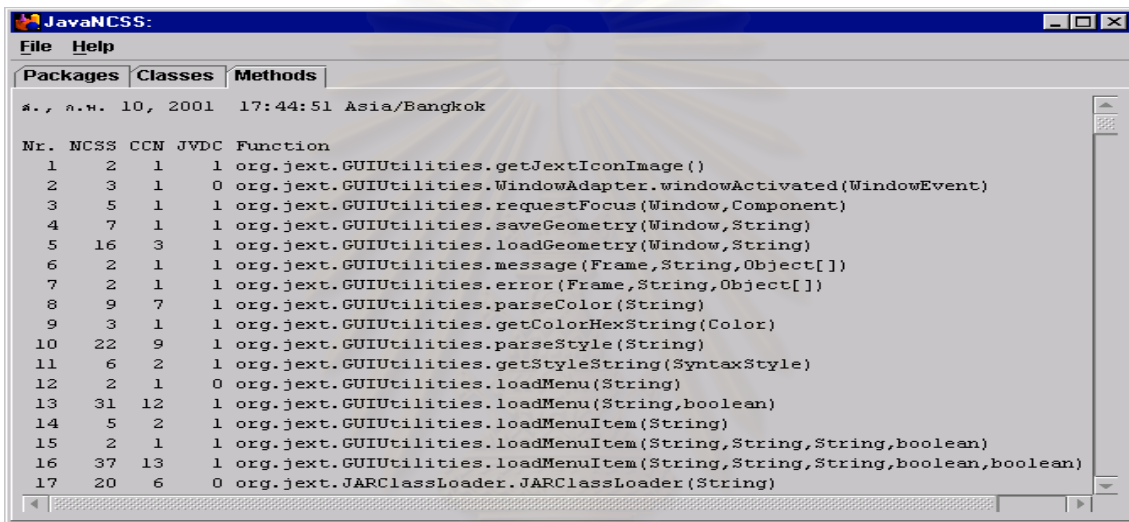
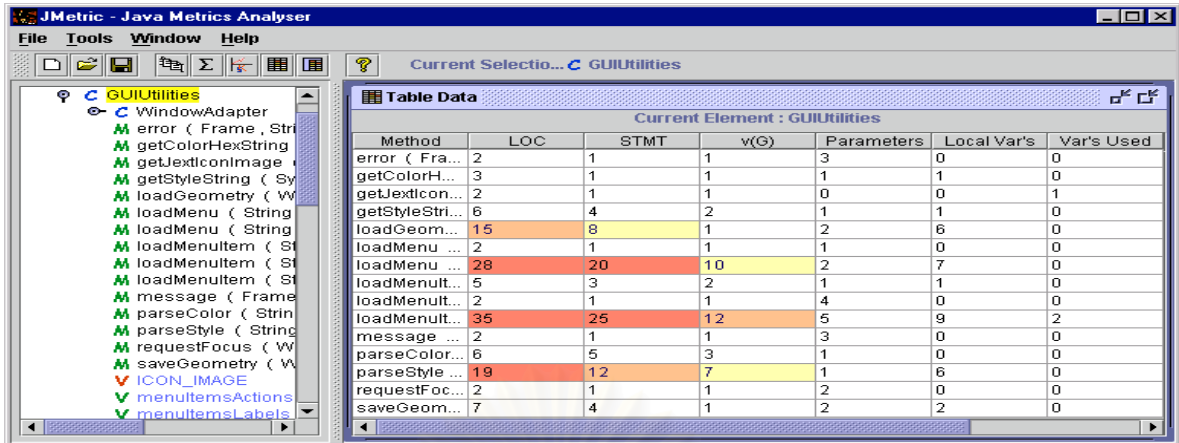
รูปที่ 6.27 ค่าตัววัดของคลาสในโปรเจก สำหรับโปรแกรมทดสอบที่ 6

ตารางที่ 6.14 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของคลาส สำหรับโปรแกรมทดสอบที่ 6

ชื่อคลาส	เครื่องมือวัด	ระดับความลึกของการสืบทอดฯ	จำนวนเมทรอด	จำนวนบรรทัด	จำนวนสเตตเมนต์	จำนวนตัวแปรอินสแตนท์	LCOM
GUIUtilities							
	MTOOP	0	15	144	90	3	1.00000
	JMetric	0	15	144	90	-	1.00
	JavaNCSS	-	15	155	-	-	-



รูปที่ 6.28 ค่าตัววัดของเมทรอดในโปรเจก สำหรับโปรแกรมทดสอบที่ 6



รูปที่ 6.28 ค่าตัววัดของเมทริกซ์ในโปรเจก สำหรับโปรแกรมทดสอบที่ 6

ตารางที่ 6.15 การเปรียบเทียบเครื่องมือวัดกับค่าตัววัดต่าง ๆ ของเมทริกซ์ สำหรับโปรแกรมทดสอบที่ 6

ชื่อเมทริกซ์	เครื่องมือวัด	จำนวนพารามิเตอร์	จำนวนบรรทัด	จำนวนสเตทเมนต์	V(G)	จำนวนตัวแปรเมทริกซ์	CBO
loadMenu(String,boolean)							
	MTOOP	2	28	20	11	7	7
	JMetric	2	28	20	10	7	-
	JavaNCSS	-	31	-	12	-	-



	MTOOP		JavaNCSS	
	LOC	V(G)	LOC	V(G)
26. public static JMenu loadMenu(String name, boolean isLabel)	1		1	
27. {	2	1	2	1
28. if (name == null)	3		3	2
29. return null;				
30.				
31. String label;	4		4	
32. if (!isLabel)	5	2	5	3
33. {				
34. label = Jext.getProperty(name + ".label");	6		6	
35. if (label == null)	7	3	7	4
36. label = name;	8		8	
37. }				
38. else			9	
39. label = name;	9		10	
40.				
41. JextMenu menu;	10		11	
42.				
43. int index = label.indexOf('\$');	11		12	
44. if (index != -1 && label.length() - index > 1)	12	4,5	13	5
45. {				
46. menu = new JextMenu(label.substring(0, index).concat(label.substring	13		14	
(++index));				
47. menu.setMnemonic(Character.toLowerCase(label.charAt(index)));	14		15	
48. }				
49. else			16	
50. menu = new JextMenu(label);	15		17	
51.				
52. if (isLabel)	16	6	18	6
53. return menu;	17		19	7
54.				
55. String menuItems = Jext.getProperty(name);	18		20	
56. if (menuItems != null)	19	7	21	8
57. {				
58. StringTokenizer st = new StringTokenizer(menuItems);	20		22	
59. while (st.hasMoreTokens())	21	8	23	9
60. {				
61. String menuItemName = st.nextToken();	22		24	
62. if (menuItemName.equals("-"))	23	9	25	10
63. menu.getPopupMenu().add(new JextMenuSeparator());	24		26	
//addSeparator();				
64. else			27	
65. {				
66. JMenuItem mi = loadMenuItem(menuItemName);	25		28	
67. if (mi != null)	26	<u>10+1</u>	29	11
68. menu.add(mi);	27		30	
69. }				
70. }				
71. }				
72.				
73. return menu;	<u>28</u>		<u>31</u>	<u>12</u>
74. }				
75.				

รูปที่ 6.29 ตัวอย่างการนับจำนวนบรรทัดและการหาค่าไซโคลเมตริกของแมคเคบของเมทอด "loadMenu(String, boolean)" ของเครื่องมือวัด MTOOP กับ JavaNCSS ของโปรแกรมค้นฉบับทดสอบที่ 6

### 6.3 สรุปผลการทดสอบ

ผลการทดสอบของเครื่องมือวัดซอฟต์แวร์ MTOOP สามารถสรุปได้ว่าเครื่องมือวัดนี้สามารถหาค่าตัววัดต่าง ๆ ได้ตามที่ได้ออกแบบไว้ และเมื่อนำค่าตัววัดที่ได้มาเปรียบเทียบกับค่าตัววัดที่ได้จากเครื่องมือวัดอื่น ๆ ที่มีชื่ออยู่ในขณะนี้คือ JMetric และ JavaNCSS สามารถสรุปได้ว่าค่าตัววัดต่าง ๆ ส่วนใหญ่มีค่าตรงกัน ส่วนตัววัดที่มีค่าไม่ตรงกันนั้นเป็นเพราะว่าข้อกำหนดในการหาค่าตัววัดบางอย่างไม่เหมือนกัน ได้แก่

6.3.1 การนับจำนวนบรรทัดของการประกาศตัวแปร เครื่องมือวัด MTOOP และ JMetric จะนับจำนวนบรรทัดเท่ากับจำนวนตัวแปรที่มีการประกาศ ส่วนเครื่องมือวัด JavaNCSS จะนับเป็น 1 บรรทัดถ้ามีการประกาศตัวแปรหลายตัวในบรรทัดเดียวกัน เช่น การประกาศตัวแปร `int x, y, z;` เครื่องมือวัด MTOOP และ JMetric จะได้  $LOC = 3$  ส่วนเครื่องมือวัด JavaNCSS จะได้  $LOC = 1$  เป็นต้น

6.3.2 การหาค่าไซโคลเมตริกของแมคเคบ เครื่องมือวัด MTOOP และ JMetric จะได้จากผลรวมของจำนวนโหนด  $CaseLabel() + IfStatement() + WhileStatement() + DoStatement() + ForStatement + ConditionalOrExpression() + ConditionalAndExpression()$  บวกด้วย 1 ในเมทริกสนั้น ๆ ส่วนเครื่องมือวัด JMetric และ JavaNCSS จะไม่นับรวมจำนวนโหนดของ  $ConditionalOrExpression()$  และ  $ConditionalAndExpression()$  เข้าไปด้วย นอกจากนี้ JavaNCSS จะนับจำนวนโหนดของ `return()` และถ้ามีการนับจำนวนโหนดของ `return()` ก็จะไม่มีการบวกด้วย 1

## บทที่ 7

### บทสรุปและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงบทสรุปและข้อเสนอแนะเกี่ยวกับงานวิจัย ที่ได้จากการออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์เชิงวัตถุ ดังรายละเอียดต่อไปนี้

#### 7.1 บทสรุป

ผู้วิจัยได้พัฒนาเครื่องมือวัดซอฟต์แวร์เชิงวัตถุ ที่มีโปรแกรมต้นฉบับเป็นภาษาจาวา เพื่อวัดขนาดและความซับซ้อนของซอฟต์แวร์ ซึ่งสูตรการคำนวณค่าตัววัดต่าง ๆ ได้มาจากการรวบรวมจากผลงานวิจัยของ McCabe ซึ่งได้ออกแบบค่าวัดไซโคลเมตริกของแมคเคบ และ Chidamber and Kemerer ได้ออกแบบชุดของการวัดสำหรับการออกแบบเชิงวัตถุ โดยชุดของการวัดมีด้วยกัน 6 ค่าคือ จำนวนเมทอดต่อคลาส ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ จำนวนคลาสลูก ขนาดความสัมพันธ์ระหว่างวัตถุ ระดับการตอบสนองต่อคลาสและระดับของการขาดความสัมพันธ์ภายในคลาส ส่วนการวิเคราะห์และออกแบบระบบได้ใช้ภาษายูเอ็มแอลเป็นเครื่องมือในการวิเคราะห์และออกแบบเพื่อใช้ในการพัฒนาเครื่องมือวัดนี้ การออกแบบระบบได้แบ่งส่วนประกอบชุดของคลาสออกเป็น 4 แพ็กเกจคือ แพ็กเกจส่วนติดต่อผู้ใช้ แพ็กเกจคอมไพเลอร์ เป็นส่วนสร้างซินแทกซ์ทรี แพ็กเกจคำนวณค่าตัววัด เป็นส่วนเก็บค่าคุณสมบัติและคำนวณค่าตัววัด และแพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด เป็นส่วนรวบรวมค่าตัววัดและจัดรูปแบบเพื่อส่งให้ส่วนติดต่อผู้ใช้งานไปแสดงผล ผู้วิจัยได้พัฒนาเครื่องมือวัดตามที่ได้ออกแบบไว้ซึ่งสามารถคำนวณหาค่าตัววัดได้ตามประเภทของ โปรเจกต์ แพ็กเกจคลาส เมทอด และตัวแปรอินสแตนซ์ และเครื่องมือวัดยังสามารถเก็บข้อมูลค่าตัววัดที่ได้บันทึกลงฐานข้อมูลได้ จากนั้นก็ได้ทำการทดสอบความสามารถของเครื่องมือวัด ซึ่งก็สามารถวัดขนาดและความซับซ้อนของซอฟต์แวร์ได้ตามที่ได้ออกแบบไว้

ผู้วิจัยสามารถสรุปประโยชน์ของเครื่องมือวัดซอฟต์แวร์ ได้ดังต่อไปนี้

1. สามารถใช้เครื่องมือในการวัดขนาดและความซับซ้อนของซอฟต์แวร์แบบอัตโนมัติ

เมื่อผู้ใช้เลือกโปรแกรมต้นฉบับภาษาจาวาเข้าสู่ระบบ ระบบจะทำการคำนวณค่าตัววัดต่าง ๆ ให้อัตโนมัติ ทำให้ผู้ใช้สามารถทราบขนาดและความซับซ้อนของซอฟต์แวร์ทันทีที่ต้องการ

2. สามารถใช้เครื่องมือในการติดตามความก้าวหน้าในการพัฒนาซอฟต์แวร์

ถ้าผู้พัฒนาซอฟต์แวร์ได้มีการประมาณขนาดและความซับซ้อนของซอฟต์แวร์ เครื่องมือวัดนี้จะ เป็นประโยชน์ในการช่วยติดตามความก้าวหน้าของการพัฒนาซอฟต์แวร์ได้ ว่าเป็นไปตามที่ได้ประมาณหรือไม่ อีกทั้งยังทำให้รู้ว่าการพัฒนาซอฟต์แวร์จะเสร็จทันตามกำหนดหรือไม่

### 3. สามารถใช้เครื่องมือในการเลือกปรับปรุงโปรแกรมที่มีขนาดและความซับซ้อนมาก

ถ้าผู้พัฒนาซอฟต์แวร์ได้มีการกำหนดขนาดและความซับซ้อนของซอฟต์แวร์ เช่น จำนวนบรรทัดของแต่ละเมทรودจะต้องมีค่าไม่เกิน 50 บรรทัด หรือค่าความซับซ้อนของไซโคลเมติกของแมคเคบจะต้องมีค่าไม่เกิน 10 เป็นต้น หลังจากใช้เครื่องมือวัดเพื่อดูค่าตัววัดแล้วพบว่ามีความซับซ้อนเกินจากที่กำหนดไว้ ผู้พัฒนาก็สามารถพิจารณาปรับปรุงให้มีขนาดหรือความซับซ้อนน้อยลงได้ เช่น แยกโมดูลออกเป็นโมดูลย่อย หรือลดโปรแกรมต้นฉบับที่ไม่มีความจำเป็นออกจากโปรแกรม เป็นต้น

### 4. สามารถใช้เครื่องมือในการประมาณขนาดและความซับซ้อนของโปรเจกต์ได้

เนื่องจากเครื่องมือวัดนี้สามารถเก็บค่าตัววัดต่าง ๆ ลงฐานข้อมูลได้ จึงสามารถนำข้อมูลไปวิเคราะห์และใช้ในการประมาณค่าของโปรเจกต์ได้ โดยเฉพาะอย่างยิ่งโปรเจกต์ที่มีลักษณะคล้ายคลึงกัน

## 7.2 ข้อเสนอแนะ

ผู้วิจัยมีข้อเสนอแนะเกี่ยวกับงานวิจัย ดังต่อไปนี้

1. เนื่องจากเครื่องมือวัดนี้จะคำนวณและแสดงค่าตัววัดต่าง ๆ เท่านั้น ซึ่งผู้ใช้จะต้องนำค่าต่าง ๆ ไปวิเคราะห์หาค่าตัววัดที่ได้เกินค่าที่กำหนดไว้หรือไม่ ดังนั้นเครื่องมือนี้จะเป็นประโยชน์มากขึ้นถ้าผู้ใช้สามารถกำหนดค่าต่าง ๆ ของตัววัดได้ และถ้าค่าตัววัดเกินค่าที่กำหนดอาจจะแสดงสีหรือข้อความแสดงให้รู้ว่าค่าตัววัดนั้น ๆ มีค่าเกินกว่าที่กำหนดไว้
2. เนื่องจากค่าตัววัดในงานวิจัยต่าง ๆ มีเป็นจำนวนมาก แต่เครื่องมือวัดนี้ได้เลือกมาเป็นบางตัวเท่านั้น ดังนั้นงานวิจัยนี้จึงเป็นแนวทางในการหาค่าตัววัดอื่น ๆ เพื่อเป็นประโยชน์ในการวิเคราะห์หาค่าคุณสมบัติและพัฒนาซอฟต์แวร์ให้ดีขึ้นต่อไป

## 7.3 ผลงานตีพิมพ์

ผลงานวิจัยนี้ได้รับคัดเลือกให้ถูกตีพิมพ์ในงานสัมมนาวิชาการ NCSEC 2000 เป็นการประชุมวิชาการทางด้านวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์แห่งชาติครั้งที่ 4 "The 4th National Computer Science and Engineering Conference (NCSEC 2000)" ซึ่งได้จัดขึ้นที่ศูนย์การประชุมแห่งชาติสิริกิติ์ ระหว่างวันที่ 16-17 พฤศจิกายน พ.ศ. 2543 โดยรายละเอียดแสดงอยู่ในภาคผนวก ก.

## รายการอ้างอิง

- [1] Chidamber, S.R., and Kemerer, C.F. A Metrics Suite for Object-Oriented Design. IEEE Trans. Software Engineering. 20, 6, (June 1994): 476-493.
- [2] Hitz, M. and Montazeri, B. Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective. IEEE Trans. Software Engineering. 22, 4, (April 1996): 267-271.
- [3] Jones, C. Applied Software Measurement: assuring productivity and quality. (NY: McGraw-Hill, 1991).
- [4] Lorenz, M. and Kidd, J. Object-Oriented Software Metrics. (NJ: Prentice-Hall, 1994).
- [5] McCabe, T.J. A Complexity Measure. IEEE Trans. Software Engineering. 2, 4, (December 1976): 308-320.
- [6] Pressman, R.S. Software Engineering: A Practitioner's Approach 4 th ed. (NY: McGraw-Hill, 1997).
- [7] Schach, S.R. Classical and Object-Oriented Software Engineering With UML and Java. 4 th ed. (NY: McGraw-Hill, 1999).
- [8] Henderson-Sellers, B. Object-Oriented Metrics: Measures of Complexity. (NJ: Prentice-Hall, 1996).
- [9] Watson, A.H. and McCabe, T.J. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. National Institute of Standards and Technology Special Publication 500-235, 123 pages (September 1996).
- [10] Rumbaugh, J., Jacobson, I., Booch, G. The Unified Modeling Language Reference Manual. (MA: Addison-Wesley, 1999).
- [11] Zukowski, J. Java AWT Reference. 1 st ed. (CA: O'Reilly & Associates, 1997).
- [12] ชนะ นิตยฤกษ์. การออกแบบและพัฒนาเครื่องมือช่วยในการทำความเข้าใจโปรแกรมภาษาจาวา (ปีการศึกษา 2541) สาขาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย. หลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย.



ภาคผนวก

สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก.

## ผลงานตีพิมพ์

ผลงานนี้ได้ถูกตีพิมพ์ในงานสัมมนาวิชาการ NCSEC 2000 เป็นการประชุมวิชาการทางด้านวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์แห่งชาติครั้งที่ 4 "The 4th National Computer Science and Engineering Conference (NCSEC 2000)" ซึ่งได้จัดขึ้นที่ศูนย์การประชุมแห่งชาติสิริกิติ์ ระหว่างวันที่ 16-17 พฤศจิกายน พ.ศ. 2543



สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย

# A Measurement Tool for Object-Oriented Programs

Somwang Sae-Tang  
Department of Computer Engineering  
Chulalongkorn University  
Bangkok, Thailand  
E-Mail: tang42@usa.net

Pornsiri Muenchaisri  
Department of Computer Engineering  
Chulalongkorn University  
Bangkok, Thailand  
E-Mail: muenchp@cp.eng.chula.ac.th

**Abstract:** Software measurement for object-oriented programs called MTOOP (Metrics Tool for Object-Oriented Programs) is a tool used for measuring size and complexity of Java source programs. This tool reports Line of code (LOC), Number of Method per Class, Lack of Cohesion, and McCabe's Cyclomatic Metrics etc. Software developers can use this tool to monitor the progress of software developing and analyze the complication of each method. It helps developers to select the very complicated methods to be improved and also helps system analyst to estimate size and complexity of the next projects especially the similar projects as the previous one. This tool starts the process by reading and transforming source programs into the syntax tree. Then it traverses the tree to collect the program's attributes and computes the metrics to display on the output screen.

**Key words:** size, complexity, Measurement, Metrics, Object-Oriented Program, method, Lack of Cohesion, McCabe's Cyclomatic Metrics

## 1. บทนำ

แม้ว่าปัจจุบันนี้จะมีเทคโนโลยีทางคอมพิวเตอร์และวิธีการใหม่ ๆ มากมายมาช่วยในการพัฒนาซอฟต์แวร์ แต่การพัฒนาซอฟต์แวร์ยังคงประสบปัญหาต่างๆ เหล่านี้ เช่น การพัฒนาซอฟต์แวร์ใช้ระยะเวลาในการพัฒนาไม่เป็นที่พอใจตามกำหนด ต้องลงทุนมากกว่าที่ได้ตั้งเอาไว้ ซอฟต์แวร์ที่พัฒนาได้มีคุณภาพต่ำไม่ตรงตามความต้องการและไม่เป็นที่พอใจของผู้ใช้งาน ทั้งนี้อาจเนื่องมาจากว่าผู้พัฒนาไม่มีการใช้เครื่องมือวัดซอฟต์แวร์ในการควบคุมและติดตามความก้าวหน้าของการพัฒนาซอฟต์แวร์ หรือผู้พัฒนาไม่ได้นำค่าวัดจากโครงการในอดีตมาช่วยในการประมาณค่าสำหรับโครงการปัจจุบันและโครงการในอนาคต ผู้วิจัยตระหนักถึงความสำคัญของการวัด (measurement) ซอฟต์แวร์ ที่มีความจำเป็นมากสำหรับการพัฒนาซอฟต์แวร์ การวัดช่วยในการวางแผน การประมาณ การควบคุมโครงการ ช่วยปรับปรุงคุณภาพของซอฟต์แวร์ให้ดีขึ้นและช่วยในการตัดสินใจในการดำเนินงานต่อไป

## 2. งานวิจัยที่เกี่ยวข้อง

งานวิจัยนี้ได้จากการรวบรวมผลงานวิจัยของ McCabe และ Chidamber and Kemerer โดย McCabe ได้ออกแบบค่าวัดไซโคลเมตริกของแมคเคบ (Cyclomatic Complexity metric V(G)) [3] Chidamber and Kemerer ออกแบบชุดของการวัดสำหรับการออกแบบเชิงวัตถุ (Object-Oriented Design) โดยชุดของการวัดนี้มีด้วยกัน 6 ค่าคือ จำนวนเมทอดต่อคลาส (Weighted Methods per Class: WMC) ระดับความลึกของแผนภูมิแสดงการสืบทอดคุณสมบัติ (Depth of the Inheritance Tree : DIT) จำนวนคลาสลูก (Number Of Children : NOC) ขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling Between Objects : CBO) ระดับการตอบสนองต่อคลาส (Response for a Class : RFC) และระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of Cohesion of Methods : LCOM) [1, 2, 4]

## 3. การวิเคราะห์และออกแบบเครื่องมือ MTOOP

ผู้วิจัยแสดงผลการวิเคราะห์และออกแบบเครื่องมือ MTOOP ด้วยโมเดลการใช้งาน (Use Case Modeling) และโมเดลของคลาส (Class Modeling) ดังต่อไปนี้



### 3.1 โมเดลการใช้งาน (Use Case Modeling)

ผู้ใช้สามารถใช้งานและติดต่อกับเครื่องมือนี้ได้ 3 กรณี คือผู้ใช้สามารถกำหนดซอร์สโปรแกรมภาษาจาวาที่ต้องการหาค่าตัววัด ผู้ใช้สามารถดูค่าวัดแยกตามโปรเจกต์ แพ็กเกจ คลาส เมททอด ตัวแปรและผู้ใช้สามารถดูค่าวัดในรูปของตารางสำหรับคลาสและสำหรับเมททอด การที่ผู้ใช้จะสามารถดูค่าวัดต่าง ๆ เหล่านี้ได้จะต้องมีการคำนวณค่าวัดก่อน การคำนวณค่าวัดนี้จะต้องอาศัยการแปลงซอร์สโปรแกรมเป็นแผนภูมิต้นไม้ แล้วเก็บข้อมูลคุณสมบัติต่างๆ

### 3.2 โมเดลของคลาส (Class Modeling)

โมเดลของคลาสสามารถแบ่งออกเป็น 4 แพ็กเกจหลักคือ แพ็กเกจส่วนติดต่อผู้ใช้ (UI) แพ็กเกจคอมไพเลอร์ (Compiler) เป็นส่วนสร้างชิ้นเท็กซ์ทีรี แพ็กเกจคำนวณค่าตัววัด (Collection) เป็นส่วนเก็บค่าคุณสมบัติและคำนวณค่าตัววัด และแพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด (Presenter) เป็นส่วนรวบรวมค่าตัววัดและจัดรูปแบบเพื่อส่งให้ส่วนติดต่อผู้ใช้ นำไปแสดงผล

## 4. การพัฒนาเครื่องมือ MTOOP

ผู้วิจัยได้พัฒนาตามที่ได้ออกแบบไว้ซึ่งสามารถคำนวณหาค่าตัววัดต่างๆ โดยแบ่งออกเป็น 5 ประเภทคือ

### 4.1 ค่าตัววัดของโปรเจกต์

ค่าตัววัดของโปรเจกต์ ได้แก่ จำนวนแพ็กเกจ จำนวนคลาส จำนวนเมททอด จำนวนเมททอดต่อคลาส จำนวนบรรทัด และจำนวนตัวแปรคลาส

### 4.2 ค่าตัววัดของแพ็กเกจ

ค่าตัววัดของแพ็กเกจ ได้แก่ จำนวนคลาส จำนวนเมททอด จำนวนเมททอดต่อคลาส จำนวนบรรทัด และจำนวนตัวแปรคลาส

### 4.3 ค่าตัววัดของคลาส

ค่าตัววัดของคลาส ได้แก่ จำนวนเมททอด จำนวนบรรทัด จำนวนตัวแปรคลาส และค่าของการขาดความสัมพันธ์ภายในคลาส

### 4.4 ค่าตัววัดของเมททอด

ค่าตัววัดของเมททอด ได้แก่ จำนวนพารามิเตอร์ จำนวนบรรทัด จำนวนตัวแปรเมททอด และค่าวัชโรลของเมททอด

### 4.5 ค่าตัววัดของตัวแปรคลาส

ค่าตัววัดของตัวแปรคลาส ได้แก่ จำนวนครั้งที่ตัวแปรคลาสถูกเรียกใช้ และจำนวนเมททอดที่เรียกใช้ตัวแปรคลาส

## 5. สรุป

ผู้วิจัยได้เสนอวิธีการออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุที่เป็นภาษาจาวา ซึ่งขั้นตอนการวิเคราะห์และออกแบบระบบได้แบ่งส่วนประกอบออกเป็น 4 แพ็กเกจคือ แพ็กเกจส่วนติดต่อผู้ใช้ แพ็กเกจคอมไพเลอร์ แพ็กเกจคำนวณค่าตัววัด และแพ็กเกจจัดรูปแบบการแสดงผลค่าตัววัด ในขั้นตอนการพัฒนา ผู้วิจัยได้พัฒนาตามที่ได้ออกแบบไว้ซึ่งสามารถคำนวณหาค่าตัววัดได้ตามประเภทของ โปรเจกต์ แพ็กเกจ คลาส เมททอด และตัวแปรคลาส งานวิจัยนี้เป็นแนวทางในการพัฒนาหาค่าตัววัดที่นอกเหนือจากในบทวิจัยนี้เช่น คำนวณค่าตัววัดขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling Between Object : CBO) และให้สามารถเก็บค่าตัววัดที่ได้จัดเก็บเป็นฐานข้อมูลเพื่อใช้ในการเปรียบเทียบ ประมาณขนาดและความซับซ้อนของโครงการอื่น ๆ เป็นต้น

## 6. หนังสืออ้างอิง

- [1] Chidamber S. R. and Kemerer C. F., "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Engineering, vol. 20, no. 6: pp. 476-493, June 1994.
- [2] Hitz M. and Montazeri B., "Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective," IEEE Trans. Software Engineering, vol. 22, no. 4: pp. 267-271, April 1996.
- [3] McCabe T. J., "A Complexity Measure," IEEE Trans. Software Engineering, vol. 2, no. 4: pp. 308-320, December 1976.
- [4] Pressman R. S., Software Engineering: A Practitioner's Approach 4<sup>th</sup> edition, McGraw-Hill, 1997.

## ประวัติผู้เขียนวิทยานิพนธ์

นายสมหวัง แซ่ตั้ง เกิดเมื่อวันที่ 17 พฤษภาคม พ.ศ. 2514 ที่จังหวัดอุทัยธานี สำเร็จการศึกษาหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วท.บ.) สาขาสถิติประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เมื่อปีการศึกษา 2536 หลังจากนั้นได้ทำงานในบริษัท รอยเตอร์ (ประเทศไทย) จำกัด จนถึงปัจจุบัน (พ.ศ. 2544) และเข้าศึกษาต่อหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2540



สถาบันวิทยบริการ  
จุฬาลงกรณ์มหาวิทยาลัย