รายงานโครงการวิจัยประจำปีงบประมาณ 2548

เรื่อง

การออกแบบเครื่องมือแปลงภาษาจาวาเป็นแผนภาพ-

Design of JAVA – Diagram Transformation Tool

โดย

พรฤดี เนติโสภากุล

และ

เอื้ออนงค์ พลชนะ

โครงการสำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2548

การออกแบบเครื่องมือแปลงภาษาจาวาเป็นแผนภาพ

Design of JAVA – Diagram Transformation Tool

พรฤดี เนติโสภากุล และ เอื้ออนงค์ พลชนะ

บทคัดย่อ

ในการทำความเข้าใจโค้ดเป็นสิ่งที่ยากมากที่จะเข้าใจในรายละเอียดต่างๆของโค้ดได้อย่างครบถ้วนและถูกต้องทั้งหมดได้ วิธีการหนึ่งที่ใช้ในการทำความเข้าใจโค้ดซึ่งเป็นภาษาในการเขียนโปรแกรมนั้น คือ การทำความเข้าใจโดยสื่อออกมาเป็นแผนภาพ ลักษณะของการแปลงรูปจากโค้ดเป็นลักษณะภาษาแผนภาพนี้ เรียกว่า เทคนิคการรีเวอร์ส เอ็นจีเนียริ่ง ในงานวิจัยนี้มุ่งเน้นในการสร้างเครื่องมือที่มีประโยชน์ทางด้าน รีเวร์ส เอ็นจีเนียริ่ง โดยสามารถทำความเข้าใจโค้ดจาวาด้วยการนำเสนอในรูปแบบเชิงภาพ ทั้งหมด 3 แผนภาพ คือ คลาสไดอะแกรม, ซีเควนซ์ไดอะแกรม, และ Structure Chart ในการแปลงโค้ดภาษาจาวาไปเป็นแผนภาพได้มีการคำนึงถึงการแลกเปลี่ยนระหว่างเครื่องมืออื่นๆด้วยจึงนำมาตรฐานในการแลกเปลี่ยนที่มีชื่อว่า เอ็กซ์เอ็มไอ(XMI-XML Metadata Interchange) มาช่วยในการเป็นตัวกลางในการแสดงออกของโค้ดภาษาจาวาก่อนที่จะนำมาแมพเป็นแผนภาพ

ABSTRACT

Program code understanding is a very difficult task because there are many details. One approach to understand programs is to draw diagrams which represent the program. Techniques which transform codes to diagrams are called reverse-engineering. This research develops a transformation tool that parses JAVA code and transform to three types of diagrams. Those are class diagram, sequence diagram, and structure chart. In addition, the intermediate representation standard is considered. XMI metadata interchange is used as an intermediate representation for mapping java code to diagram.

สารบัญ

สารบัญ(ต่อ)

เรื่อง **หน้า**

สารบัญรูป

หน้า

สารบัญรูป(ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาของปัญหาและความสำคัญของโครงการพัฒนาระบบงาน

เทคโนโลยีการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) ได้ถูกเสนอขึ้นตั้งแต่ทศวรรษ 1980 เป็นต้นมา และเป็นที่นิยมใช้กันอย่างแพร่หลายจนปัจจุบัน ถึงแม้เทคโนโลยีการโปรแกรมเชิงวัตถุจะช่วยทำให้ประสิทธิภาพในการพัฒนาซอฟต์แวร์ดีขึ้นเป็นอย่างมาก แต่เทคโนโลยีดังกล่าวก็ก่อให้เกิดปัญหาในการทดสอบ และ บำรุงรักษาซอฟต์แวร์ (Netisopakul 2003) อันเกิดจากความสลับซับซ้อนของสถาปัตยกรรมซอฟต์แวร์เชิงวัตถุเป็นส่วนใหญ่

ระบบซอฟต์แวร์เชิงวัตถุประกอบไปด้วยวัตถุ (Object) จำนวนมาก (Lorenz and Kid 1994) ซึ่งวัตถุเหล่านี้จะมีคุณสมบัติ (Attribute) และพฤติกรรม (Method) ของตัวเอง และวัตถุเหล่านี้จะมีการโต้ตอบหรือติดต่อ (Interaction) กันกับวัตถุอื่นๆผ่านกลไกการส่งแมสเสจ (Massage) ซึ่งเป็นเรื่องยากที่จะทำความเข้าใจ ปัจจุบันปัญหาของการทดสอบโปรแกรมเชิงวัตถุมีมากมายหลายด้าน เช่น ปัญหาของการถ่ายทอดคุณสมบัติของคลาส และเมธตอด ซึ่งคุณสมบัตินี้เป็นประโยชน์อย่างมากในการนำกลับมาใช้ใหม่ (reuse) , ปัญหาในด้านการบำรุงรักษาระบบโปรแกรมเชิงวัตถุ และรวมถึงปัญหาของ Dynamic Binding , การเป็นอิสระของโปรแกรม , โครงสร้างโปรแกรมที่กระจัดกระจาย , การควบคุมโพลีมอร์ฟีซึ่ม (Polymorphism) และการเข้าใจในรายละเอียดของโปรแกรม จากปัญหาต่าง ๆ ที่กล่าวมาทั้งหมดนี้เป็นสาเหตุที่ทำให้ผู้พัฒนาและผู้ทดสอบโปรแกรมไม่สามารถทำความเข้าใจรายละเอียดของโปรแกรมได้

ลักษณะงานที่เป็นการบำรุงรักษาหรือการพัฒนาที่เป็นการทำความเข้าใจของโปรแกรมที่เป็นโค้ดที่เขียนด้วยภาษาเชิงวัตถุใดๆก็ตามนั้นเป็นเรื่องยากและถ้ามีการขาดการจัดการลักษณะงานทางด้านเอกสารที่ดีแล้วยิ่งเป็นเรื่องที่ยากลำบากเป็นอย่างยิ่ง บ่อยครั้งที่การทำความเข้าใจโค้ดนั้นไม่ตรงกับสิ่งที่ได้ทำการออกแบบไว้ตั้งแต่ต้น เทคนิคที่เป็นการทำความเข้าใจในตัวโค้ดนั้นมักมีการแสดงออกเพื่อให้เกิดความเข้าใจด้วยรูปแบบเชิงภาพหรือแบบจำลอง (model) การทำการแปลงจากโค้ดสู่ลักษณะเชิงแบบจำลองเป็นเทคนิคที่เรียกว่า รีเวอร์ส เอ็นจีเนียริ่ง (reverse engineering)

เทคนิคทางด้านรีเวอร์ส เอ็นจีเนียริ่ง เป็นเทคนิคที่ผู้วิเคราะห์สามารถผลิตโมเดลที่ออกแบบได้ ซึ่งสามารถถูกใช้เพื่อเข้าใจหรือเปิดเผยให้เห็นถึงพฤติกรรมหรือสถาปัตยกรรมอีกทั้งยังสามารถเข้าใจภาพรวมทั้งหมดของโปรแกรมหรือระบบนั้นๆได้

ยูเอ็มแอล (UML ย่อมาจาก Unified Modeling Language)เป็นภาษาเชิงแบบจำลองที่นิยมใช้ในการออกแบบและในทางกลับกันก็เป็นที่นิยมในการทำรีเวอร์ส เอ็นจีเนียริ่งด้วย ยูเอ็มแอล ได้กลายมาเป็นแนวทางส่วนใหญ่ในระบบซอฟแวร์ซึ่งทำให้เกิดเครื่องมือที่รองรับยูเอ็มแอลอย่างมากมาย อีกทั้งเครื่องมือต่างๆเหล่านั้นยังมีความสามารถในการทำรีเวอร์ส เอ็นจีเนียริ่งจากโค้ดไปสู่ยูเอ็มแอลได้ แต่ว่าผลลัพธ์ที่ได้นั้นไม่สมบูรณ์หรือขาดความสามารถในการแลกเปลี่ยนระหว่างเครื่องมืออื่นๆที่มีความสามารถในการทำรีเวอร์ส เอ็นจีเนียริ่ง

แนวทางหนึ่งที่นำมาแก้ไขปัญหาในการทำรีเวอร์ส เอ็นจีเนียริ่งจากจาวาโค้ดไปสู่ยูเอ็มแอลโมเดลในโครงงานนี้คือการนำตัวกลางในการแลกเปลี่ยนรูปแบบหนึ่งที่จัดได้ว่าเป็นมาตรฐาน (standard) ที่เป็นที่ยอมรับของเครื่องมือต่างๆที่มีความสามารถทางด้านรีเวอร์ส เอ็นจีเนียริ่งแล้ว มาตรฐานนี้มีชื่อว่า เอ็กซ์เอ็มไอ(XMI ย่อมาจาก XML Metadata Interchange)

1.2 วัตถุประสงค์ของการพัฒนาระบบ

คือ การพัฒนาเครื่องมือในการแปลงภาษาจาวาเป็นแผนภาพ(diagram)ที่อิงกับมาตรฐานยูเอ็มแอล โดยทำการแปลงโค้ดภาษาจาวาไปเป็น คลาสไดอะแกรม(class diagram), ซีเควนซ์ไดอะแกรม(sequence diagram), และ Structure Chart

1.3 สมมุติฐานของการพัฒนาระบบ

- ซอร์สโค้ดที่นำมากระทำการทำความเข้าใจนั้นต้องเป็นซอร์สโค้ดจาวาที่ได้รับการตรวจสอบไวยากรณ์ต่างๆถูกต้องแล้ว
- ซอร์สโค้ดจาวาต้องมีรูปแบบการเขียนที่เป็นสากลทั่วไป
- ซอร์สโค้ดจาวานั้นถูกพัฒนาด้วย JDK 1.0 – 1.4

1.4 ขอบเขตการพัฒนาระบบ

ในการศึกษาการพัฒนาเครื่องมือนี้เป็นการเอามาตรฐานกลางในการแลกเปลี่ยนเอ็กซ์เอ็มไอมาช่วยในการศึกษาและวิเคราะห์ในการแปลงโค้ดจาวาเป็นแผนภาพต่างๆตามมาตรฐานยูเอ็มแอลโดยกำหนดขอบเขตของการศึกษาไว้ดังนี้

1) การแปลงโค้ดจาวาเป็นแผนภาพต่างๆตามมาตรฐานทางยูเอ็มแอลได้เลือกเฉพาะแผนภาพที่เป็นที่นิยมใช้และเหมาะสมในการสื่อให้เกิดความเข้าใจในโค้ดนั้น แผนภาพที่เหมาะสมและได้ทำการพัฒนาในโครงการนี้ คือ คลาสไดอะแกรม, ซีเควนซ์ไดอะแกรม, และ Structure Chart

2) การออกแบบและพัฒนาระบบได้แบ่งการศึกษาออกเป็นสองส่วน โดยส่วนแรกเป็นการวิเคราะห์และออกแบบระบบในการแมพภาษาจาวาเป็นลักษณะตามมาตรฐานกลางเอ็กซ์เอ็มไอ โดยมีการใช้ตัวจำแนกที่มีชื่อว่า JavaCC(Java Complier Complier) มาใช้ในส่วนนี้ และส่วนที่สองเป็นส่วนของการวิเคราะห์และออกแบบในการแปลงตัวกลางเอ็กซ์เอ็มไอให้เป็นแผนภาพ โดยมีการใช้ตัวผู้วิเคราะห์ SAX (Simple API for XML)

3) การศึกษาการการสร้างเอกสารทางเอ็กซ์เอ็มไอ เพื่อเป็นตัวกลางในการแลกเปลี่ยนระหว่างโค้ดจาวากับแผนภาพ

1.5 ขั้นตอนการดำเนินการพัฒนาระบบ

- ศึกษารายละเอียดแผนภาพต่างๆ, ศึกษารายละเอียดของ JavaCC(Java Complier Complier), และศึกษารายละเอียดมาตรฐานทางเอ็กซ์เอ็ม
- ศึกษาเครื่องมืออื่นๆที่มีความสามารถในการแมพภาษาจาวาไปสู่แผนภาพโดยใช้มาตรฐานกลางเอ็กซ์เอ็มไอมาช่วยในการแลกเปลี่ยน
- ออกแบบตัวต้นแบบที่มีความสามารถในการแปลงโค้ดภาษาจาวาไปเป็นเอกสารทางเอ็กซ์เอ็มไอโดยใช้ JavaCC (Java Complier Complier)
- ออกแบบตัวต้นแบบในการแปลงเอกสารทางเอ็กซ์เอ็มไอไปสู่โครงสร้างแผนภาพ
- ออกแบบอินเตอร์เฟสในการติดต่อกับผู้ใช้
- ทดสอบและประเมินผลที่ได้รับ
- จัดทำเอกสารสรุปการทำโครงการ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

- ได้เข้าใจและประยุกต์ใช้เทคนิครีเวอร์สเอ็นจีเนียริ่ง
- เครื่องมือนี้สามารถอำนวยความสะดวกให้กับผู้พัฒนาระบบที่ต้องบำรุงรักษาหรือพัฒนาระบบที่มีการขาดจัดการเอกสารที่ดีได้

1.7 เครื่องมือที่ใช้

- EditPlus
- Rational Rose 2002
- Java2 SDK, Standard Edition Version 1.4.2
- NotePad++
- XMI Toolkit
- JavaCC(Java Complier Complier)
- JBuilder 9

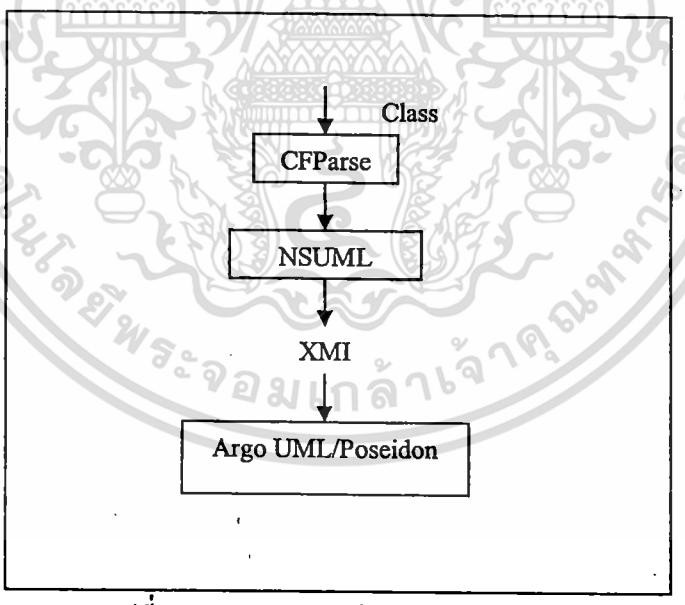
บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 การศึกษาเครื่องมืออื่นๆ

2.1.1 Java2UML (Pechanec. 2000)

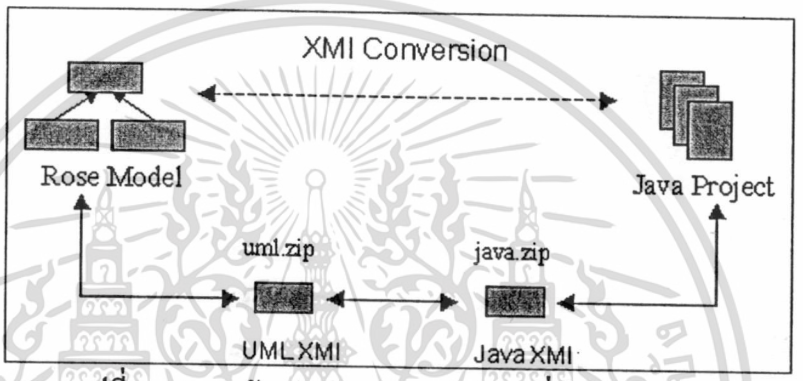
เครื่องมือนี้มีแนวคิดในการทำรีเวอร์สเอ็นจิเนียริ่ง ดังรูปที่ 2.1 โดยอินพุตของเครื่องมือนี้เป็นไฟล์คลาส (.class) มีตัวผู้วิเคราะห์(parser)ทั้งหมด 2 อย่าง คือ CFParse เป็นตัวผู้วิเคราะห์ที่ทำการวิเคราะห์คลาสไฟล์ ต่อจากนั้นเมื่อได้ผลลัพธ์จะทำการเรียกตัวผู้วิเคราะห์ที่มีชื่อว่า NSUML(Novo Soft UML) ทำการวิเคราะห์ ได้ผลลัพธ์เป็นการแสดงออกทางเอ็กซ์เอ็มไอ ต่อจากนั้นนำผลลัพธ์ที่ได้นี้ไปทดสอบบนเครื่องมือที่มีชื่อว่า ArgoUML หรือ Poseidon for UML



รูปที่ 2.1 แสดงโครงสร้างของ Java2UML

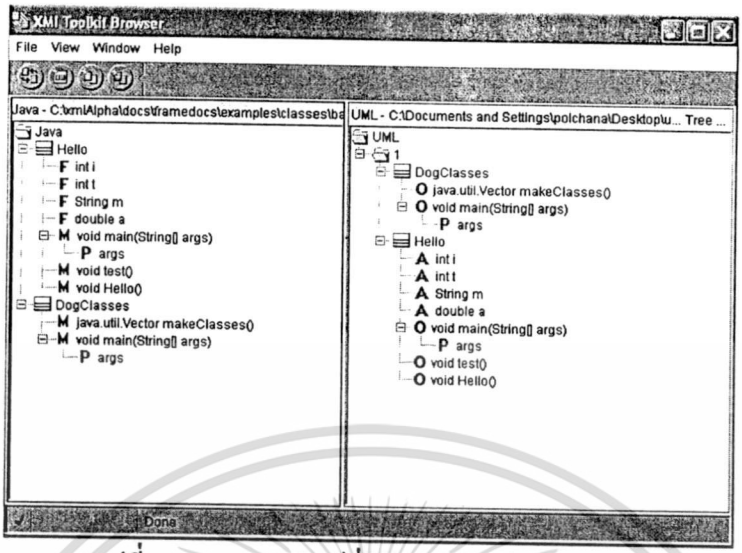
2.1.2 XMI Toolkit (IBM. 1999)

เป็นเครื่องมือที่ถือกำเนิดขึ้นมาเพื่อมาตรฐานของเอ็กซ์เอ็มไอ เครื่องมือนี้นอกจากมีความสามารถในการทำรีเวอร์ส เอ็นจิเนียริ่งแล้วยังมีความสามารถในการแปลงจากลักษณะทางโมเดลไปเป็นโค้ดภาษาจาวา ในการแปลงของทั้ง 2 เทคนิค(จากโค้ดไปสู่โมเดลและจากโมเดลไปสู่โค้ด) นั้นเครื่องมือนี้ทำการแปลงเข้าสู่มาตรฐานกลางที่มีชื่อว่า เอ็กซ์เอ็มไอ ดังที่แสดงในรูปที่ 2.2 ที่แสดงลักษณะของความสามารถที่มีในเครื่องมือนี้



รูปที่ 2.2 แสดงลักษณะของความสามารถที่มี XMIToolkit

ตัวอย่างของผลลัพธ์ที่ได้จากการใช้เครื่องมือนี้ แสดงในรูปที่ 2.3 โดยเป็นการใช้ตัวอย่างที่เป็นโค้ดจาวา 2 ไฟล์คือ Hello.java และ DogClasses.java มาใช้ในการสร้างแบบจำลองโดยในที่นี้ให้ชื่อเป็น "1.mdl" ในการแสดงผลที่ได้จากการแปลงนั้นเห็นได้ว่ามีการแสดงผลในรูปแบบของต้นไม้(tree)ซึ่งสามารถแบ่งออกเป็น 2 ส่วนคือ 1) ส่วนจาวาโค้ด มีการใช้ตัวย่อ F แทนคุณสมบัติ(field), M แทนพฤติกรรม(method), และ P แทนพารามิเตอร์(parameter) 2) ส่วนยูเอ็มแอล มีการใช้ตัวย่อ O แทนพฤติกรรม(operation), P แทนพารามิเตอร์, และ A แทนคุณสมบัติ(attribute) สำหรับไฟล์ที่เป็นรูปแบบมาตรฐานกลางเอ็กซ์เอ็มไอนั้นมีการเก็บเป็นไฟล์ซิพ(zip file) ซึ่งได้เป็นไฟล์ที่ชื่อว่า java.zip(จากตัวอย่างข้างในไฟล์ซิพนี้ได้ไฟล์ Hello.xml และDogClasses.xml) และ uml.zip โดยภายในของซิพไฟล์จะเป็นไฟล์ที่เป็นสกุลเอ็กซ์เอ็มแอล(.xml)



รูปที่ 2.3 แสดงผลลัพธ์ที่ได้จากการใช้ XMIToolkit

2.2 เอ็กซ์เอ็มแอล (XML -Extensible Markup Language) (สุวัฒนา. 2545.)

XML ย่อมาจาก Extensible Markup Language ซึ่งเป็นภาษา Markup คล้ายกับ SGML (Standard Generalize markup Language) ซึ่งเป็นต้นกำเนิดของภาษา HTML ซึ่งเป็นภาษาที่ได้รับการออกแบบมาเพื่อใช้ในการกำหนดลักษณะ และโครงสร้างข้อมูล ซึ่งเป็นข้อมูลที่มีความหมายอยู่ในตัวมันเอง (Self - Describe data) โดยอาศัยการสร้างแท็ก (tag) ที่ผู้ใช้งานกำหนดขึ้นเองเพื่อนิยามข้อมูล ทำให้สามารถนำมาเป็นรูปแบบที่การแลกเปลี่ยนข้อมูลระหว่างหน่วยงานหรือองค์กรได้ โดยไม่ต้องกังวลถึงเรื่องความแตกต่างของรูปแบบข้อมูล

2.2.1 โครงสร้างเอกสารเอ็กซ์เอ็มแอล (XML Document Structure)

โครงสร้างของเอกสารเอ็กซ์เอ็มแอลจะต้องมีรูปแบบที่ถูกต้องตามข้อกำหนดของมาตรฐาน (XML Specification) ที่องค์กร W3C (World Wide Web Consortium) ได้กำหนด โดยรูปที่ 2.4 เป็นการแสดงตัวอย่างของโครงสร้างเอกสารทางเอ็กซ์เอ็มแอล ลักษณะของเอกสารที่เป็นรูปแบบของโครงสร้างที่ดี (Well-Formed Document) นั้นมีคุณสมบัติดังนี้ (Homer. 1999)

- เอกสารเอ็กซ์เอ็มแอลหนึ่ง ๆ จะมีรูทอิลิเมนต์ (root element) ได้เพียงแค่หนึ่งเดียวเท่านั้น ซึ่งทำหน้าที่ควบคุมอิลิเมนต์(element)อื่น ๆ ทั้งหมด
- แท็กเปิดและแท็กปิดต้องเหมือนกัน ต่างกันในส่วนของแท็กปิดที่ต้องมีเครื่องหมาย slash (/) นำหน้าชื่อแท็กเท่านั้น
- ห้ามระบุแท็กเหลื่อมซ้อนกัน คือ แท็กที่เปิดก่อนต้องปิดทีหลัง
- ชื่อแท็กมีคุณสมบัติที่เรียกว่า Case-Sensitive คือตัวอักษรพิมพ์เล็ก – ใหญ่ถือว่าแตกต่าง

- สำหรับแท็กว่างคือแท็กที่ไม่มีข้อมูลข้างใน มีวิธีเขียนได้ 2 แบบ
  แบบที่หนึ่ง เป็นการเขียนแท็กว่างด้วยแท็กเพียงแท็กเดียว เช่น <title/>
  แบบที่สองเป็นการเขียนแท็กว่างด้วยอิลิเมนต์หนึ่งอิลิเมนต์ เช่น <title></title>
- ค่าของแอตทริบิวต์(attribute)ต้องอยู่ในเครื่องหมาย double quote (“) หรือ single quote (‘) อย่างใดอย่างหนึ่ง
- การตั้งชื่อแท็กจะต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย Underscore (_) เท่านั้น สามตัวแรกของชื่อห้ามเป็นคำว่า xml
- การประกาศเอ็กซ์เอ็มแอล(XML Declaration) เป็นการประกาศให้รู้ว่าไฟล์นั้นๆเป็นไฟล์เอกสาร XML โดยทุกตัวอักษรห้ามเป็นตัวพิมพ์ใหญ่ มีลักษณะการประกาศดังนี้
  <?xml version=”1.0”?>
- ข้อความที่เป็นคอมเมนต์(comment)ต้องอยู่ระหว่างเครื่องหมาย <! และ !> โดยห้ามแทรกอยู่ในส่วนหัวของเอกสารและห้ามแทรกอยู่ในแท็ก

```
<?xml version="1.0">
<BookStore>
    <Book category='Computer'>
        <Title>XML for Beginner</Title>
        <Author>
            <FirstName>John</FirstName>
            <LastName>Smith</LastName>
        </Author>
    </Book>
    </Book>
</BookStore>
```

รูปที่ 2.4 ตัวอย่างเอกสารเอ็กซ์เอ็มแอล

2.2.2 การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอล

จากที่เอกสารทางเอ็กซ์เอ็มแอลได้กลายเป็นมาตราฐานที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างหน่วยงาน หรือองค์กร ดังนั้นทั้งฝ่ายส่งและฝ่ายรับเอกสารเอ็กซ์เอ็มแอลของแต่ละองค์กรจะต้องมั่นใจว่า เอกสารนั้นมีความถูกต้อง และสามารถนำไปใช้งานได้ตามข้อตกลงร่วมกัน ซึ่งการกำหนดโครงสร้างเอกสารก็เพื่อให้เอกสารเอ็กซ์เอ็มแอลนั้นมีรูปแบบที่ถูกต้องพร้อมที่จะนำไปใช้งาน ซึ่งวิธีการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลที่นิยมใช้กันมี 2 วิธี ดังนี้

2.2.2.1 Document Type Definition (DTD) (สราวุธ. 2544.)

DTD ใช้ไวยากรณ์ (syntax) ในลักษณะที่เป็นทางการในการอธิบายโครงสร้างไวยากรณ์ภาษาของเอกสารเอ็กซ์เอ็มแอล โดยการใช้ DTD เป็นการแยกคำอธิบายข้อมูลเอ็กซ์เอ็มแอล ออกจากแอพพลิเคชันใด ๆ ทำให้แอพพลิเคชันที่ทำงานร่วมกันทั้งหมด สามารถใช้คำอธิบายใด ๆ ของข้อมูลร่วมกันได้ โดยที่รูปที่ 2.5 เป็นการแสดงลักษณะของ XML DTD

การกำหนด DTD อาจกำหนดรวมอยู่ภายในเอกสาร XML (เรียกว่าสับเซตภายในหรือ Internal DTD) หรือกำหนดเป็นไฟล์แยกจากเอกสาร XML ซึ่งมีนามสกุลเป็น .dtd (เรียกว่าสับเซตภายนอกหรือExternal DTD) หรืออาจจะกำหนดไว้แบบผสมผสานกันทั้งแบบภายในและภายนอก

```
<!ELEMENT BookStore (Book)* >
<!ELEMENT Book(Title,Author)>
<!ELEMENT Title(#PCDATA)>
<!ELEMENT Author(FirstName,LastName)>
<!ELEMENT FirstName(#PCDATA)>
<!ELEMENT LastName(#PCDATA)>
```

รูปที่ 2.5 ตัวอย่าง XML DTD ของ BookStore

2.2.2.2 เอ็กซ์เอ็มแอลสกีมา(XML Schema) (พอเจตน์. 2546.)

เอ็กซ์เอ็มแอลสกีมาทำหน้าที่เหมือนกับ DTD แต่เป็นอีกวิธีหนึ่งซึ่งเป็นแนวโน้มใหม่ที่จะเข้ามาแทนที่ DTD เนื่องจากมีความยืดหยุนกว่า อีกทั้งใช้ไวยากรณ์คล้ายกับเอ็กซ์เอ็มแอลมาตราฐานอีกด้วย

เอ็กซ์เอ็มแอลสกีมาเป็นมาตราฐานการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลของ W3C ประกอบด้วย 2 ส่วนหลัก ๆ คือชนิดข้อมูล และโครงสร้าง

2.2.3 วิธีดึงข้อมูลจากเอกสารเอ็กซ์เอ็มแอลมาใช้งาน

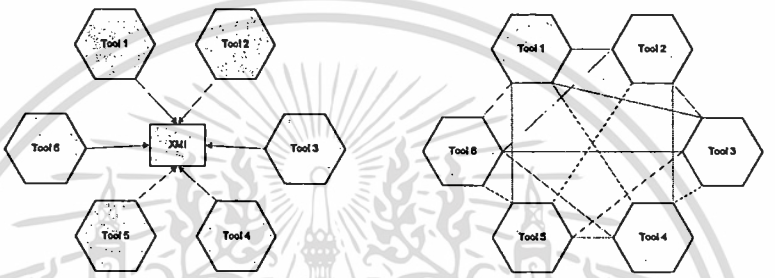
การแปลเอกสารเอ็กซ์เอ็มแอลละต้องอาศัยโปรแกรมที่เรียกกันว่า Parser(ตัวผู้วิเคราะห์) มาช่วยในการวิเคราะห์หรือดึงข้อมูล ดังนั้นการแปลความความหมายหรือต้องการการใช้ประโยชน์เอกสารทางเอ็กซ์เอ็มแอลนั้นจึงต้องมีการใช้ตัวผู้วิเคราะห์ ตัวผู้วิเคราะห์นี้มักเรียวกันว่า XML Parser โดยในที่นี้ทำการกล่าวถึง XML Parser 2 ชนิด คือ DOM และ SAX โดยสามารถดูรายละเอียดได้ในหัวข้อ 2.6

2.3 เอ็กซ์เอ็มไอ(XMI - XML Metadata Interchange) (พอเจตน์. 2003.)

เอ็กซ์เอ็มไอถูกสร้างขึ้นโดยโอเอ็มจี(OMG - Object Management Group) ซึ่งเป็นองค์กรระดับชาติที่ได้รับการสนับสนุนจากสมาชิกกว่า 600 กลุ่ม ซึ่งมีทั้ง Information System Vendor, นักพัฒนาซอฟแวร์ และกลุ่มผู้ใช้ทั่วไป โอเอ็มจีได้ก่อตั้งขึ้นในปี 1989 โดยมีแนวปฏิบัติ คือให้การสนับสนุนทฤษฎีและทำการทดสอบทาง Object Oriented Technology ในด้านการพัฒนาซอฟแวร์ โดยจุดประสงค์หลักของงานในองค์กรนี้ คือ การนำกลับมาใช้ใหม่ได้(reusable) , การเคลื่อนย้ายข้อมูลโปรแกรมจากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง(portable) และความสามารถในการติดต่อกับซอฟท์แวร์ที่มีโครงสร้างแบบอ๊อบเจท ในลักษณะที่สามารถใช้งานได้แบบกระจายในสภาวะแวดล้อมที่แตกต่างกัน

ในการเชื่อมโยงของโลกปัจจุบันทำให้การติดต่อสื่อสาร การแลกเปลี่ยนข้อมูลต่างๆเกิดขึ้นได้อย่างไม่จำกัดและใน มาตรฐานของเอ็กซ์เอ็มไอ เป็นมาตรฐานในการแสดงผ่านตัวกลางที่นำมาใช้ในการแลกเปลี่ยนเชื่อมโยงโปรแกรมจากหลายๆผู้จำหน่าย(vendor)ที่ใช้งานอยู่เข้าด้วยกันได้ การที่ไม่มีการแสดงผ่านตัวกลางในการเชื่อมโยงการแลกเปลี่ยนนั้นมีความยุ่งยากเกิดขึ้น ดูได้จากรูปที่ 2.6 เห็นได้ว่าการที่โปรแกรมสามารถเชื่อมโยงการแลกเปลี่ยนซึ่งกันและกันได้นั้น แต่ละโปรแกรมต้องรู้จักทุกๆเครื่องมือที่แตกต่างกันอยู่ในการเชื่อมโยงนี้ ดังนั้นการแลกเปลี่ยนที่เกิดขึ้นภายในเครื่องมือทั้ง 6 ชนิดนี้ ต้องมีจุดที่เชื่อมโยง(bridge)ทั้งหมด 30 จุด โดยคำนวณจุดเชื่อมโยงจากสูตร N*N-N (N คือ vendor) ดังนั้น ในความเป็นจริงการใช้งานเครื่องมือนั้นมีอยู่เป็นจำนวนมากมายหลากหลายชนิดทำให้การเชื่อมโยงการแลกเปลี่ยนต้องมีการสร้างจุดที่เชื่อมโยงเป็นจำนวนหลายแสนจุดในการติดต่อซึ่งกันและกัน จากปัญหาเหล่านี้ทำให้มีการใช้ตัวกลางในการแลกเปลี่ยนเกิดขึ้นโดยการเชื่อมโยงของแต่ละเครื่องมือที่มีการนำตัวกลางในการแลกเปลี่ยนเอ็กซ์เอ็มไอมาใช้ทำให้ในการแลกเปลี่ยนระหว่างเครื่องมือทั้ง 6 ชนิดนี้มีจุดเชื่อมโยงทั้งหมดเพียง 6 จุด(Brodsky. 1999)

เอ็กซ์เอ็มไอ ย่อมาจากคำว่า XML Metadata Interchange โดยที่คำว่า Metadata (รายละเอียดข้อมูล) หมายถึง การจำกัดความ, การอธิบายข้อมูล หรือ ข้อมูลที่เกี่ยวกับข้อมูล(data about data)(XML Journal. 2002.) เอ็กซ์เอ็มไอ เป็นมาตรฐานหนึ่งสำหรับการเก็บและการแบ่งปันโปรแกรมที่เป็นเชิงวัตถุ, การนำโค้ดกลับมาใช้ใหม่ หรือการออกแบบข้อมูล จุดประสงค์หลักของเอ็กซ์เอ็มไอคือ การทำให้การแลกเปลี่ยนของรายละเอียดข้อมูลเป็นสิ่งที่ง่ายขึ้น ระหว่าง เครื่องมือที่มีลักษณะสัมพันธ์เชิงแบบจำลอง และส่วนประกอบของรายละเอียดข้อมูล ภายใต้สภาพแวดล้อมที่แตกต่างกัน(OMG. 2002)



รูปที่ 2.6 แสดงการเชื่อมโยงของเครื่องมือที่มีและไม่มีการแสดงผ่านตัวกลางการแลกเปลี่ยน

เอ็กซ์เอ็มไอ เป็นมาตรฐานเป็นมาตรฐานที่ผสมผสานมาตรฐานทั้ง 3 มาตรฐาน คือ เอ็กซ์เอ็มแอล(XML – Extensible Markup Language), ยูเอ็มแอล (UML - Unified Modeling Language), และ เอ็มโอเอฟ(MOF- Meta Object Facility)

เอ็กซ์เอ็มแอล(สามารถดูรายละเอียดได้ที่หัวข้อ 2.2) เป็นมาตรฐานที่ใช้ในการแลกเปลี่ยนข้อมูลบนอินเตอร์และองค์กรที่รับผิดชอบมาตรฐานนี้ คือ W3C(World Wide Web Consortium) ลักษณะที่สัมพันธ์กันกับ เอ็กซ์เอ็มไอ คือ เอ็กซ์เอ็มไอ ใช้ เอ็กซ์เอ็มแอล เพื่อการเก็บและนำวัตถุจากเอกสาร เอ็กซ์เอ็มไอ ได้กำหนดลักษณะหลายด้านที่สำคัญเพื่อใช้ในการอธิบายอ็อบเจท ด้วยภาษาเอ็กซ์เอ็มแอล เป็นต้นว่า

- ทำการนำเสนอ object ในส่วนของ เอ็กซ์เอ็มแอล element และ attribute เป็นหลัก
- เนื่องจากอ็อบเจทต้องมีการติดต่อระหว่างกันเป็นหลัก เอ็กซ์เอ็มไอ จึงรวมเอากลไกพื้นฐานในการติดต่อทั้งการติดต่อของอ็อบเจทภายในไฟล์เดียวกัน และระหว่างไฟล์
- ระบุตัวแทนของอ็อบเจท(identity) เป็นการอ้างถึงจากอ็อบเจทอื่น ในแบบ IDs และ UUIDs
- Validation ของ เอ็กซ์เอ็มไอ document เรียกใช้จาก DTDs และ Schemas

ยูเอ็มแอล เป็นมาตรฐานในการแสดงรายละเอียดแบบจำลองเชิงวัตถุและโอเอ็มจีเป็นองค์กรที่รับผิดชอบมาตรฐานนี้ ลักษณะที่สัมพันธ์กับ เอ็กซ์เอ็มไอ คือ ยูเอ็มแอลทำการจำกัดโครงสร้างของอ๊อบเจทและคลาส เพื่ออธิบายว่าเอ็กซ์เอ็มไอแสดงถึงอ๊อบเจทและคลาสในเอ็กซ์เอ็มแอลได้อย่างไร(สามารถดูรายละเอียดของมาตรฐานนี้ได้ที่หัวข้อ 2.4)

เอ็มโอเอฟ เป็นมาตรฐานที่จัดแนวทางของการอธิบายรายละเอียดแบบจำลอง(Metamodel) และโอเอ็มจีเป็นองค์กรที่รับผิดชอบมาตฐานนี้เช่นกัน ลักษณะที่สัมพันธ์กับ เอ็กซ์เอ็มไอ คือ มาตรฐานนี้ทำการเลือกกลุ่มย่อย(subset)ของยูเอ็มแอล ที่มีความเหมาะสมสำหรับรายละเอียดข้อมูลด้วยแบบจำลอง(XML Journal. 2002.)

การทำความเข้าใจในเรื่องโครงสร้างรายละเอียดแบบจำลอง(metamodel architecture) สามารถอธิบายการแบ่งระดับของรายละเอียด(meta-level)ได้ดังตารางที่ 2.1 (Stevens. 2000)

ตารางที่ 2.1 ระดับของรายละเอียด (meta-level)

ระดับ	คำจำกัดความของ MOF
M0	Data
M1	Metadata model
M2	meta-metadata metamodel
M3	Meta-metamodel

M0 เป็น ข้อมูล คือเป็นระดับที่รายละเอียดน้อยที่สุด

M1 เป็นแบบจำลองของรายละเอียดข้อมูล เช่น แบบจำลองทางยูเอ็มแอล (ถ้าเทียบกับเอ็กซ์เอ็มไอ คือ เอกสารทางเอ็กซ์เอ็มไอ) (Jiang. and Tarja. 2002.)

M2 เป็นระดับที่มีความละเอียดมากกว่า M1 เช่น รายละเอียดแบบจำลองยูเอ็มแอล(UML metamodel) (ถ้าเทียบกับ เอ็กซ์เอ็มไอ คือ XMI DTD หรือ XMI Schema) (Jiang. and Tarja. 2002.)

M3 เป็นรายละเอียดของรายละเอียดแบบจำลอง เช่น แบบจำลองเอ็มโอเอฟ

2.3.1 โครงสร้างเอกสารเอ็กซ์เอ็มไอ(XMI Document)

โครงสร้างของทุกๆเอกสารทางเอ็กซ์เอ็มไอ ประกอบด้วย XML version, การประกาศการเข้ารหัส(encoding), คำสั่งในการประมวลผล(processing instruction หรือ PI) อื่นๆ และการประกาศ DTD ดังตัวอย่างในรูปที่ 2.8 ที่เป็นการแสดงลักษณะของเอกสารทางเอ็กซ์เอ็มไอ โดยมี

รายละเอียดดังนี้ Person เป็น class ที่มีข้อมูลภายใน class ที่ประกอบด้วย FirstName (ชื่อ), LastName (นามสกุล), Mobile (เบอร์มือถือ), Tel (เบอร์บ้าน)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XMI SYSTEM "person.dtd">
<XMI xmi.version="1.0" >
   <XMI.header>
     <XMI.documentation>
        An example of an person.
     </XMI.documentation>
   </XMI.header>
   <XMI.content>
     <Person>
       < Person.FirstName>John</ Person.FirstName >
       < Person.LastName >Smith</ Person.LastName >
       < Person.Mobile>017777777</ Person.Mobile >
       < Person.Tel>02999999</ Person.Tel >
     </ Person >
   </XMI.content>
</XMI>
```

รูปที่ 2.7 ตัวอย่างเอกสารทางเอ็กซ์เอ็มไอ

จากรูปที่ 2.7 นี้สามารถสังเกตเห็น prefix คำว่า 'XMI.' กับ 'xmi.' มีความหมายการใช้ดังต่อไปนี้ คือเอ็กซ์เอ็มแอลอิลิเมนต์(XML element) ถูกระบุโดยข้อกำหนดของมาตรฐานเอ็กซ์เอ็มไอ(XMI Specification) โดยใช้ prefix 'XMI.' ส่วนเอ็กซ์เอ็มไอแอตทริบิวต์(XML attributes) ถูกระบุด้วย prefix ว่า 'xmi.'

ในการอ่านหรือต้องการดึงข้อมูลของเอกสารทางเอ็กซ์เอ็มไอนั้น อย่างที่กล่าวมาแล้วว่าเอ็กซ์เอ็มไอมีการอิงกับมาตรฐานของเอ็กซ์เอ็มแอลทำให้ในการอ่านหรือดึงข้อมูลของเอกสารทางเอ็กซ์เอ็มไอมาใช้นั้นสามารถตัวผู้วิเคราะห์ของทางเอ็กซ์เอ็มแอลได้เช่นกัน(อย่างเช่น DOM หรือ

SAX สามารถนำมาวิเคราะห์เอกสารทางเอ็กซ์เอ็มไอได้เช่นกัน) นอกจากนี้ยังมีตัวผู้วิเคราะห์ที่ถูกออกแบบให้เป็นตัวผู้วิเคราะห์ของทางเอ็กซ์เอ็มไอโดยเฉพาะเช่นกันอย่างเช่น JOB (Java Object Bridge)

2.3.2 การกำหนดโครงสร้างเอกสาร XMI

2.3.2.1 XMI DTD

DTD ใช้ไวยากรณ์ในลักษณะที่เป็นทางการในการอธิบายโครงสร้างและไวยากรณ์ภาษาของเอกสาร ซึ่งการใช้ DTD จะเป็นการแยกคำอธิบายข้อมูลออกจากแอพลิเคชันใดๆ จะทำให้ แอพลิเคชันทำงานร่วมกันทั้งหมด สามารถใช้คำอธิบายใดๆของข้อมูลร่วมกันได้ โดยที่การกำหนด DTD อาจกำหนดรวมอยู่ภายในเอกสาร (internal DTD) หรืออ้างถึง DTD ที่อยู่ภายนอกเอกสาร (external DTD) โดยในรูปที่ 2.8 เป็นการแสดงลักษณะของ DTD ของเอ็กซ์เอ็มไอ (OMG. 2002.)

```
<!ELEMENT Person (Person.FirstName, Person.LastName, Person.Mobile,
Person.Tel, XMI.extension*)? >
<!ATTLIST Person  %XMI.element.att; %XMI.link.att;>
<!ELEMENT Person.FirstName (#PCDATA | XMI.reference)* >
<!ELEMENT Person.LastName (#PCDATA | XMI.reference)* >
<!ELEMENT Person.Mobile (#PCDATA | XMI.reference)* >
<!ELEMENT Person.Tel (#PCDATA | XMI.reference)* >
```

รูปที่ 2.8 ตัวอย่าง XMI DTD ของ Person

2.3.2.2 เอ็กซ์เอ็มไอสกีมา(XMI Schema)

เอ็กซ์เอ็มไอสกีมาบ่งบอกถึงสิ่งที่ XML Processor นำไปตรวจสอบเพื่อหาความถูกต้องของไวยากรณ์ต่าง ๆ และบอกถึงความหมายของเอกสารเอ็กซ์เอ็มแอลอีกด้วย ซึ่งรายละเอียดในส่วนนั้นยังบอกถึงกฎที่ซึ่งสกีมาสามารถใช้เพื่อสร้างเอกสารให้เป็น Valid XMI-Transmittable MOF-based Metamodel แต่อย่างไรก็ตาม ในการใช้สกีมานี้เป็นเพียงทางเลือก ซึ่งจริง ๆ แล้วเอกสารเอ็กซ์เอ็มแอลไม่จำเป็นต้องอ้างถึงสกีมาเสมอไป ซึ่งจะช่วยให้กระบวนการทำเอกสารเร็วขึ้น แต่จะขาดความเชื่อมั่นในคุณภาพของเอกสารนั้น (พอเจตน์. 2546.)

ในแต่ละสกีมาที่ใช้เอ็กซ์เอ็มไอจะต้องเป็นไปตามข้อกำหนดเหล่านี้

- ทุกๆอิลิเมนต์และแอตทริบิวต์ ถูกกำหนดตามกฎพื้นฐานของเอ็กซ์เอ็มไอ ซึ่งถูกอิมพอร์ต(import) เข้ามาในสกีมาและไม่สามารถใช้ได้โดยตรงในสกีมาเหล่านั้นเอง เนื่องจากสามารถมีเพียงหนึ่ง target namespace ต่อหนึ่งสกีมาเท่านั้น การสร้าง metamodel จะต้องสอดคล้องกับการประกาศอิลิเมนต์และแอตทริบิวต์ ซึ่งในการสร้างจะต้องมีการประกาศ complextype และอาจจะมี group, attribute group, type เพิ่มตามมาอีกด้วย

2.3.3 Example of Representing XMI in UML

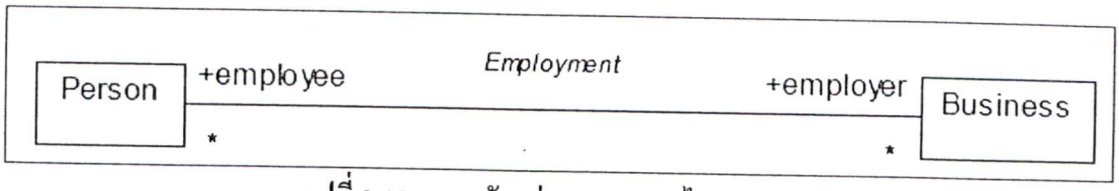
ในที่นี้ได้ยกตัวอย่างของเอกสารทางเอ็กซ์เอ็มไอดังรูปที่ 2.9 ที่รูปลักษณะของการแมพเอกสารเอ็กซ์เอ็มไอให้เข้ากับรูปแบบของแผนภาพเพียงแผนภาพเดียวเท่านั้น นั้นคือแผนภาพที่เป็นคลาสไดอะแกรมดังรูปที่รูปที่ 2.10

ในรูปที่ 2.9 เป็นการแสดงตัวอย่างเอกสารทางเอ็กซ์เอ็มไอที่เป็นการแมพแผนภาพจากคลาสไดอะแกรมแมพมาจากคลาสไดอะแกรมในรูปที่ 2.10

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE XMI SYSTEM 'UML_1.4_XMI_1.1.dtd'>
<XMI xmi.version='1.2' xmlns:UML='omg.org/UML/1.4'>
<XMI.header>
        <XMI.metamodel xmi.name='UML' xmi.version='1.4'/>
</XMI.header>
<XMI.content>
        <UML:Model xmi.id='S.1' name='Employment Model' visibility='public'
        isSpecification='false' isRoot='false' isLeaf='false' isAbstract='false'>
        <UML:Namespace.ownedElement>
        <UML:Class xmi.id='S.2' name='Person' visibility='public' isSpecification='false'
        namespace='S.1' isRoot='true' isLeaf='true' isAbstract='false' isActive='false'/>
        <UML:Class xmi.id='S.3' name='Business' visibility='public' isSpecification='false'
        namespace='S.1' isRoot='true' isLeaf='true' isAbstract='false' isActive='false'/>
        <UML:Association xmi.id='G.1' name='Employment' visibility='public'
        isSpecification='false' isRoot='false' isLeaf='false' isAbstract='false'>
        <UML:Association.connection>
```

```
<UML:AssociationEnd name='employer' visibility='public' isSpecification='false'
isNavigable='true' ordering='unordered' aggregation='none' targetScope='instance'
changeability='changeable' participant='S.3' association='G.1'>
<UML:AssociationEnd.multiplicity>
        <UML:Multiplicity>
                <UML:Multiplicity.range>
                <UML:MultiplicityRange lower='0' upper='-1'/>
                </UML:Multiplicity.range>
        </UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
</UML:AssociationEnd>
<UML:AssociationEnd name='employee' visibility='public' isSpecification='false'
isNavigable='true' ordering='unordered' aggregation='none' targetScope='instance'
changeability='changeable' participant='S.2' association='G.1'>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
        <UML:Multiplicity.range>
                <UML:MultiplicityRange lower='0' upper='-1'/>
        </UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
</UML:Namespace.ownedElement>
</UML:Model>
</XMI.content>
</XMI>
```

รูปที่ 2.9 แสดงรูปแบบของเอ็กซ์เอ็มไอที่เป็นตัวแทนของคลาสไดอะแกรม



รูปที่ 2.10 แสดงตัวอย่างของคลาสไดอะแกรม

2.4 ยูเอ็มแอล (UML – Unified Modeling Language) (จุฬาภา. 2547.และ ชาลี. 2544.)

ยูเอ็มแอลจัดเป็นภาษาภาษาหนึ่งที่มีลักษณะของรูปภาพหรือสัญลักษณ์(graphical language) ที่ใช้จำลองหรือถ่ายทอดสิ่งที่ต้องจะสื่อออกมา ทำไมจึงจัดว่ายูเอ็มแอลเป็นภาษา ก่อนอื่นต้องเข้าใจลักษณะของภาษาก่อน ภาษาโดยทั่วๆไปนั้นต้องมีโครงสร้างที่สำคัญ 2 ส่วน นั้นคือ คำศัพท์(Vocabulary) และ ไวยากรณ์(syntax) ซึ่งโครงสร้างทั้งสองนี้ก็มีอยู่ในยูเอ็มแอลเช่นกัน

องค์ประกอบของยูเอ็มแอลสามารถแบ่งได้เป็น 3 กลุ่ม ดังตารางที่ 2.2 (พนิดา และ กิตติ. 2548) โดยที่ สัญลักษณ์(Things) หมายถึง สิ่งที่ได้จากการจำลองเสมือน(abstract) ซึ่งสิ่งที่ได้จากการจำลองเสมือนก็จะถูกนำมาแทนด้วยสัญลักษณ์ และสัญลักษณ์จัดเป็นหน่วยที่เล็กที่สุดในยูเอ็มแอล, ความสัมพันธ์(Relationship) คือการเชื่อมโยงวัตถุเข้าด้วยกัน, และ แผนภาพ(Diagrams) คือการนำสัญลักษณ์และความสัมพันธ์มารวมกันแล้วนำมาแสดงเป็นแผนภาพตามมุมมองที่ต้องการ

ตารางที่ 2.2 แสดงองค์ประกอบของยูเอ็มแอล(บรรจง . และ ญาณวรรณ. 2000.)

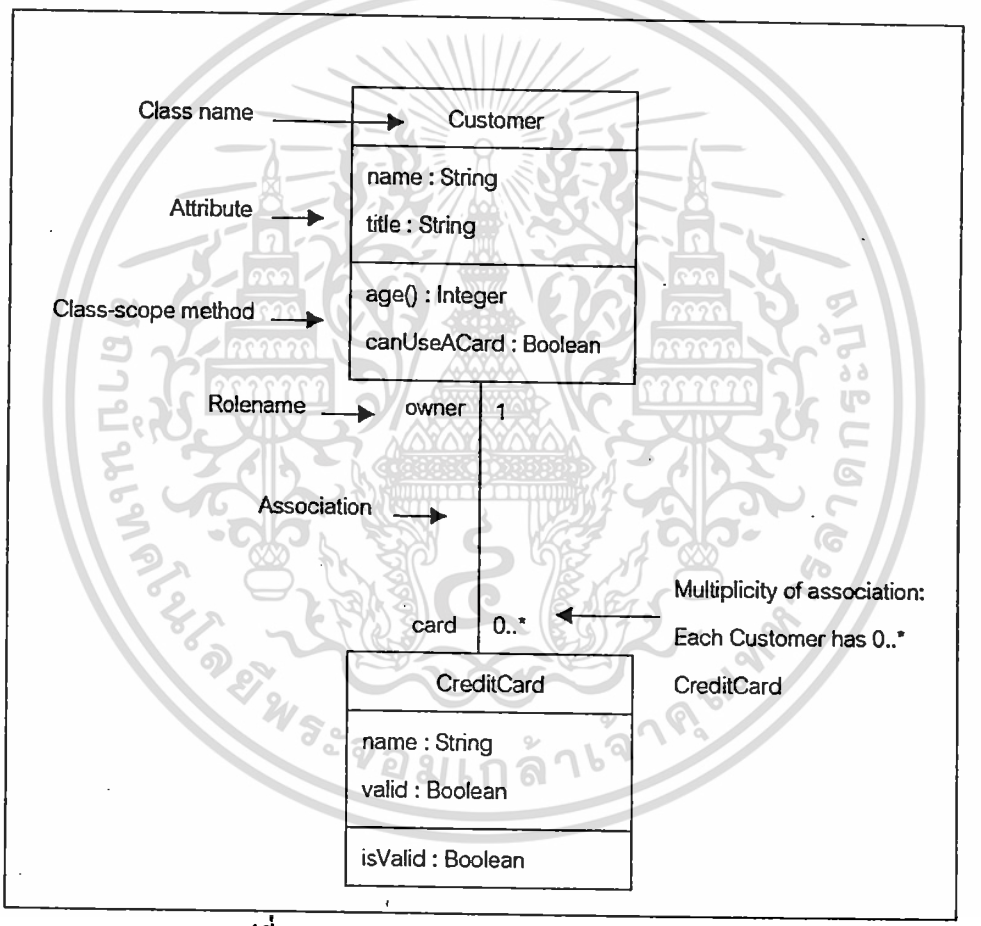
สัญลักษณ์(Things)	ความสัมพันธ์(Relationships)	แผนภาพ(Diagrams)
1. Structural Things • Class • Interface • Collaboration • Use Case • Active Class • Component • Node 2. Behavioral Things • Interaction • State Machine 3. Grouping Things • Package 4. Annotational Things • Note	• Dependency • Association • Generalization • Realization	1. Structural Diagram • Class Diagram • Object Diagram • Component Diagram • Deployment Diagram 2. Behavioral Diagram • Use Case Diagram • Sequence Diagram • Collaboration Diagram • Statechart Diagram • Activity Diagram

จากตารางที่ 2.2 แผนภาพของยูเอ็มแอลนั้นถูกแบ่งตามลักษณะในการจำแนกมุมมองเชิงโครงสร้าง โดยองค์ประกอบของแผนภาพประเภทนี้เป็นองค์ประกอบที่มีลักษณะเป็น สแตติค (Static) และมุมมองเชิงพฤติกรรม จัดว่าองค์ประกอบของแผนภาพประเภทนี้มีลักษณะเป็น ไดนามิค(Dynamic) (สุนทริน. 2537.) โดยในที่นี้ ทำการอธิบายรายละเอียดของแผนภาพที่เป็นผลลัพธ์ของเครื่องมือในการแปลงภาษาจาวาเป็นแผนภาพเท่านั้น เนื่องจากว่าบางแผนภาพของยูเอ็มแอลเป็นมุมมองระดับสูงที่ไม่สามารถพิจารณาจากตัวโค้ดได้(การพิจารณาโค้ดเป็นลักษณะของมุมมองระดับต่ำ) อย่างเช่น use case เป็นแผนภาพที่ใช้ในการวิเคราะห์ความต้องการของระบบ

2.4.1 คลาสไดอะแกรม(Class Diagram)

การสร้างคลาสไดอะแกรม เป็นไปเพื่อการแสดงโครงสร้างของระบบว่ามีการประกอบด้วยคลาส(คือ ชนิดของกลุ่มอ๊อบเจท)ใดบ้าง และมีความสัมพันธ์ระหว่างคลาส อย่างไร แผนภาพนี้จัดว่าเป็นแผนภาพที่จำเป็นและมีประโยชน์มากที่สุดสำหรับระบบที่มีการพัฒนาระบบโดยมีการเขียนโปรแกรมด้วยภาษาเชิงวัตถุ

-- องค์ประกอบของคลาสไดอะแกรม สามารถดูตัวอย่างขององค์ประกอบต่างได้ในรูปที่ 2.13 ที่แสดงตัวอย่างของคลาสไดอะแกรม มีรายละเอียดดังต่อไปนี้



รูปที่ 2.11 แสดงตัวอย่างของคลาสไดอะแกรม

- คลาส(Class)

เป็นการอธิบายถึงกลุ่มของอ๊อบเจทที่มีลักษณะและพฤติกรรมที่เหมือนกันหรือร่วมกัน โดยสัญลักษณ์ที่แทนคลาสนั้นเป็นรูปสี่เหลี่ยม และในสัญลักษณ์รูปสี่เหลี่ยมมีการแบ่งออกเป็น 3 ส่วน คือ

1) ชื่อของคลาส (Class Name) (สุนทริน. 2537.)

สามารถแบ่งลักษณะของชื่อคลาสได้ 2 แบบ คือ Simple Name เป็นลักษณะของคำๆเดียวหรือออาจจะเป็นวลี และ Path Name เป็นการตั้งชื่อที่นำแพ็กเกจ(package)ของคลาสมานำหน้า ตัวอย่างเช่น "java::util::Vector" โดยสองส่วนแรกเป็นชื่อแพ็กเกจ อีกส่วนเป็นชื่อคลาส

นอกจากนี้ในส่วนยังมีการใช้ Stereotype(เป็นเทคนิคในการขยายโมเดลอย่างหนึ่งในมาตรฐานยูเอ็มแอล ทำให้สามารถใช้องค์ประกอบทางโมเดลที่มีอยู่ในการกำหนดองค์ประกอบต่างๆ โดยมุ่งเน้นในเรื่องของการสื่อความหมายพิเศษ) ที่ระบุว่าคลาสนั้นๆเป็น abstract class โดยใส่ <<Abstract>>

2) คุณสมบัติหรือแอตทริบิวต์ (Attributes) เช่น "+ speed : Integer = 0" โดยจากตัวอย่างสามารถแบ่งส่วนประกอบของแอตทริบิวต์ได้ 4 ส่วน ดังนี้

2.1) การอ้างอิงถึงการเข้าถึง(Visibility) มีการใช้สัญลักษณ์ดังนี้

- " + " แทนการเข้าถึงแบบ "public"
- " - " แทนการเข้าถึงแบบ "private"
- " # " แทนการเข้าถึงแบบ "protected"

2.2) ชื่อของแอตทริบิวต์ จากตัวอย่าง

2.3) ประเภทของแอตทริบิวต์ (Attribute Type) โดยในการระบุชนิดของแอตทริบิวต์นั้นจะระบุต่อจากเครื่องหมายโคล่อน (Colon – " : ")

2.4) ค่าเริ่มต้นของแอตทริบิวต์

3) พฤติกรรมหรือ Methods เช่น " +age(year : Integer) : Integer " จากตัวอย่างนี้สามารถแบ่งส่วนประกอบของพฤติกรรมได้ 4 ส่วนดังนี้

3.1) การอ้างอิงถึงการเข้าถึง มีรายละเอียดเช่นเดียวกับการอ้างอิงถึงการเข้าถึงของแอตทริบิวต์

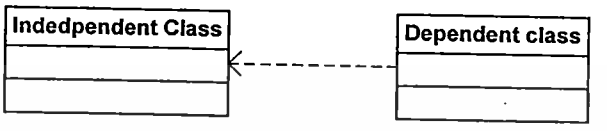
3.2) ชื่อของ method

3.3) พารามิเตอร์(parameters) รูปแบบการเขียนพารามิเตอร์นั้น คือ ชนิดของพารามิเตอร์ ตามด้วยเครื่องหมายโคล่อน และตามด้วยชนิดของพารามิเตอร์นั้นๆ ในกรณีที่มีพารามิเตอร์หลายๆตัวให้คั่นด้วยเครื่องหมายลูกน้ำ

- ความสัมพันธ์ระหว่างคลาส(Relationships) สามารถแบ่งได้ 5 รูปแบบดังนี้

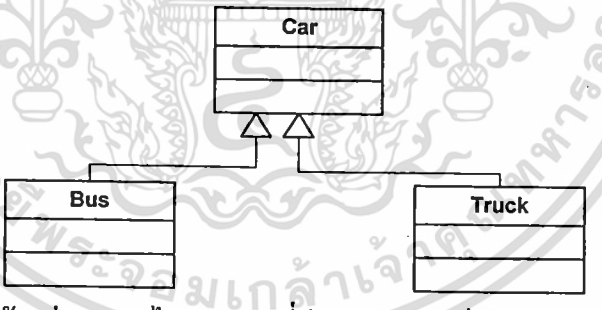
1) ความสัมพันธ์แบบพึ่งพิง(Dependency) เป็นความสัมพันธ์ที่บ่งบอกถึงความสัมพันธ์ซึ่งกันและกัน อย่างเช่น รูปที่ 2.14 ที่แสดงความสัมพันธ์แบบ

พึ่งพิง โดยการใช้สัญลักษณ์ของความสัมพันธ์แบบนี้ คือเส้นประพร้อมหัวลูกศรชี้ไปยังคลาสที่เป็นให้พึ่งพิง จากรูปที่ 2.14 สามารถกล่าวได้ว่าถ้าคลาส Independent class มีการเปลี่ยนแปลงใดๆก็ตาม คลาส Dependent class จะมีผลกระทบด้วย



รูปที่ 2.12 แสดงความสัมพันธ์แบบพึ่งพิง

2) ความสัมพันธ์แบบ Generalization เป็นลักษณะของการสืบทอด มีความสัมพันธ์ระหว่างซูเปอร์คลาส(Super class หรือคลาสแม่) กับ ซับคลาส (SubClass หรือคลาสลูก) โดยคลาสลูกจะได้รับการถ่ายทอดคุณสมบัติและพฤติกรรมจากคลาสแม่ สามารถเรียกความสัมพันธ์นี้ว่าเป็นความสัมพันธ์แบบ “is-a” สำหรับสัญลักษณ์ที่ใช้แทนความสัมพันธ์ดูได้จากตัวอย่างในรูปที่ 2.15 ที่แสดงตัวอย่างของคลาสที่มีความสัมพันธ์แบบ Generalization



รูปที่ 2.13 แสดงตัวอย่างคลาสไดอะแกรมที่มีความสัมพันธ์แบบ Generalization

3) Association เป็นการบ่งบอกถึงความสัมพันธ์ของกันและกันระหว่างนั้นๆ (ชาลี. และ เทพฤทธิ์. 2544.) โดยในรูปที่ 2.16 เป็นการแสดงตัวอย่างประกอบของรายละเอียดต่างในความสัมพันธ์แบบ Association

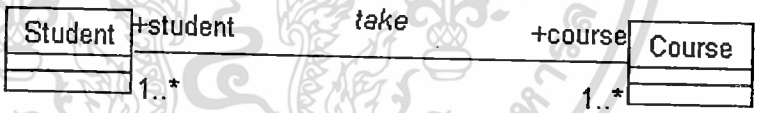
- เส้นตรงทึบ คือมีการแสดงว่าคลาสต่างๆมีความสัมพันธ์กันด้วยสัญลักษณ์ที่เป็นเส้นตรงทึบเชื่อมระหว่างกันแล้วซึ่งโดยสามารถบ่งบอกความสัมพันธ์แบบสองทิศทางหรือมากกว่า(Binary or N-ary association) ใน

บางครั้งพบว่ามีการใช้เส้นตรงที่มีการใช้หัวลูกศร เป็นการบ่งบอกถึงการใช้คุณสมบัติ Navigation(กิติ. และ กิติพงษ์. 2548) ที่แสดงถึงความสามารถในการเข้าถึง กล่าวคือ อ๊อบเจทของคลาสที่มีไม่มีหัวลูกศรสามารถเข้าถึงอ๊อบเจทของคลาสที่มีหัวลูกศรได้ แต่ในทางกลับกันอ๊อบเจทของคลาสฝั่งที่มีหัวลูกศรไม่สามารถเข้าถึงอีกฝั่งได้ ดังนั้น ความสัมพันธ์แบบ Association สามารถแยกเป็น Unidirectional Association(เป็นเส้นตรงทึบและมีหัวลูกศร) และ Bidirectional Association(เป็นเส้นตรงทึบ)

- ชื่อความสัมพันธ์ โดยที่ชื่อเหล่านี้มักเป็นคำกิริยาเป็นส่วนใหญ่ สำหรับชื่อที่ใช้กำกับความสัมพันธ์ไม่จำเป็นต้องใช้ได้

- ปริมาณของคลาสหรืออ๊อบเจทที่มีความสัมพันธ์กันโดยใช้ Multiplicity ซึ่งมีรูปแบบการบ่งบอกปริมาณที่มีความสัมพันธ์หลายรูปแบบ อย่างเช่น 0..1, จำนวนเต็มบวก, Many(*), และ 1..* เป็นต้น

- บทบาท(Role) สามารถบ่งบอกชื่อของบทบาทความสัมพันธ์ โดยเขียนชื่อบทบาทไว้ที่ฝั่งที่ต้องการแสดงชื่อบทบาท นอกจากนี้ยังสามารถการเข้าถึงหรือการมองเห็นของบทบาท (เป็นการใช้ Visibility)

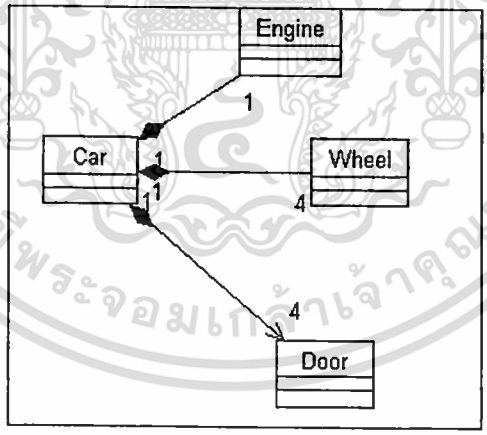


รูปที่ 2.14 แสดงตัวอย่างความสัมพันธ์แบบ Association

4) Aggregation เป็นการบ่งบอกถึงความสัมพันธ์ที่รวมกัน(Whole-Part) ซึ่งสัญลักษณ์ในการแสดงความสัมพันธ์เป็นเส้นตรงทึบโดยที่ปลายด้านหนึ่งเป็นรูปสี่เหลี่ยมขนมเปียกปูนโปร่ง ถ้าสัญลักษณ์ขนมเปียกปูนอยู่ติดที่ฝั่งของคลาสใดแสดงว่าคลาสนั้นมีสถานะที่เป็นเจ้าของอีกคลาสหนึ่งที่มีความสัมพันธ์ด้วยกันอยู่ ในรูปที่ 2.17 เป็นการแสดงตัวอย่างที่แสดงความสัมพันธ์แบบ Agrregation

รูปที่ 2.15 แสดงตัวอย่างที่แสดงความสัมพันธ์แบบ Aggregation

5) Composition เป็นการบ่งบอกความสัมพันธ์ที่มีลักษณะรวมกัน(Whole-Part) เช่นกันแต่ว่าความสัมพันธ์แบบ Composition เป็นการระบุว่าคลาสที่เป็น Part ไม่สามารถมีหรืออยู่ได้ถ้าปราศจากคลาสที่เป็น Whole(ขณะที่คลาสที่เป็น Part สามารถมีได้ถ้าปราศจากคลาสที่เป็น Whole ในความสัมพันธ์แบบ Aggregation) สัญลักษณ์ที่ใช้แทนความสัมพันธ์คล้ายกับ Aggregation เพียงแต่สัญลักษณ์สี่เหลี่ยมขนมเปียกปูนไม่โปร่งแสง ในรูปที่ 2.18 เป็นการแสดงตัวอย่างของความสัมพันธ์ Composition ที่แสดงถึงคลาสรถยนต์ที่ รถ 1 คันมีส่วนประกอบอะไรบ้าง(เช่น ล้อ, เครื่องยนต์, และประตู เป็นต้น) ซึ่งได้ไม่มีคลาสรถยนต์จะไม่สามารถมีส่วนประกอบของคลาสอื่นๆเกิดขึ้นได้



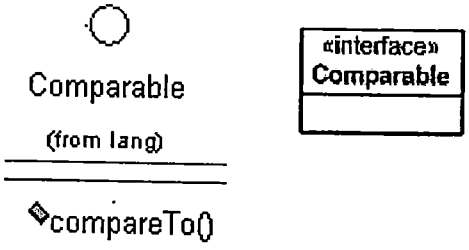
รูปที่ 2.16 แสดงตัวอย่างของความสัมพันธ์แบบ Composition

- สัญลักษณ์อื่นๆ

2อินเตอร์เฟส(Interface) สามารถใช้สัญลักษณ์แทนอินเตอร์เฟสได้ 2 แบบ ดังนี้

1) แสดงด้วยวงกลมและเขียนชื่ออินเตอร์เฟสกำกับไว้ด้านนอก

2) แสดงด้วยสัญลักษณ์ที่คล้ายกับสัญลักษณ์คลาสที่ตัดส่วนที่สองของคลาสพร้อมทั้งเขียน <<interface>> ไว้ด้านบนของส่วนที่หนึ่งของสัญลักษณ์



รูปที่ 2.17 แสดงตัวอย่างของสัญลักษณ์อินเตอร์เฟส

3แพ็กเกจ(Package) เป็นการจัดกลุ่มของคลาส ให้อยู่ภายใต้กลุ่มเดียวกัน(แพ็กเกจ) โดยสัญลักษณ์ที่ใช้ได้แสดงไว้ในรูปที่ 2.20 ที่แสดงตัวอย่างของสัญลักษณ์แพ็กแกจโดยตัวอย่างนี้เป็นค่า default ของโปรแกรม Rational Rose เมื่อผู้ใช้เลือกใช้สภาพแวดล้อมแบบ J2EE นอกจากนี้แพ็กเกจยังมีการใช้ลักษณะของการบ่งบอกถึงความสัมพันธ์เหมือนคลาสด้วย โดยความสัมพันธ์ที่ใช้ในแพ็จเกจ มี 2 ประเภท คือ Dependency(โดยสามารถ Stereotype ที่เป็น <<import>> และ <<access>> บนเส้นความสัมพันธ์) และGeneralization

รูปที่ 2.18 แสดงตัวอย่างของสัญลักษณ์แพ็กเกจ

2.4.2 ซีเควนซ์ไดอะแกรม(Sequence Diagram)

ซีเควนซ์ไดอะแกรม เป็นการแสดงการติดต่อสื่อสารระหว่างอ๊อบเจทต่างๆ โดยเน้นที่ ณ ช่วงเวลา เป็นการอธิบายด้วยสัญลักษณ์ที่ประกอบด้วยการอธิบายแบบ 2 มิติ(มองอ๊อบเจทจากซ้ายไปขวาและมองช่วงเวลาจากบนไปล่าง) ตัวอย่างของแผนภาพนั้นแสดงในรูปที่ 2.21 องค์ประกอบของซีเควนซ์ไดอะแกรมมีองค์ประกอบดังนี้

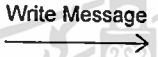
- Lifeline เป็นส่วนที่เป็นสัญลักษณ์เส้นประที่ลากจากอ๊อบเจทในแนวตั้ง
- อ๊อบเจทสามารถแบ่งออกได้เป็น 2 ประเภท คือ Concrete Object เป็นอ๊อบเจทที่สามารถระบุลักษณ์เฉพาะเจาะจง(Unique Identity) และ Prototypical Object เป็นอ๊อบเจทที่ไม่สามารถระบุเฉพาะเจาะจง

การใช้สัญลักษณ์เป็นการใช้รูปสี่เหลี่ยมผืนผ้า โดยที่การแยกประเภทระหว่างอ๊อบเจทที่เป็น Concrete Object กับ Protypical Object อยู่ที่ชื่อของอ๊อบเจท(เขียนไว้ภายในสัญลักษณ์และมีการเส้นใต้) รูปแบบของชื่ออ๊อบเจทที่เป็น Protypical Object เป็น " : ชื่ออ๊อบเจท" ส่วนรูปแบบของชื่ออ๊อบเจทที่เป็น Concrete Object เป็น "ชื่ออ๊อบเจท : ชื่อคลาส"

- Activation เป็นสัญลักษณ์รูปสี่เหลี่ยมเล็กๆที่ตั้งอยู่บนเส้น Lifeline โดยที่ Acitivation เป็นตัวแทนการทำงานต่างๆที่อ๊อบเจทและยังสื่อถึงเวลาที่อ๊อบเจทนั้นๆทำงานด้วย(ถ้าใช้ระยะเวลาในการทำงานมาก ความยาวของ Activation บน Lifeline จะยาวตามด้วย)
- เมสเสจ(Message) เป็นการสื่อสารถึงการติดต่อกันระหว่างอ๊อบเจทหรือภายในอ๊อบเจท และบนเมสเสจอาจจะมีการระบุข้อความ สามารถแบ่งชนิดของเมสเสจในการติดต่อสื่อสารดังนี้

- Simple

Write Message



เป็นสัญลักษณ์ของเมสเสจนี้ โดยแสดงว่ามีการย้ายการทำงานไปอีกอ๊อบเจทหนึ่ง

- Synchronous

Message1

เป็นสัญลักษณ์ของเมสเสจนี้ โดยแสดงถึงการติดต่อกับอ๊อบเจทที่ต้องรอคอยการตอบกลับของอ๊อบเจทนั้นก่อนจึงสามารถดำเนินงานต่อไปได้

4Asynchronous

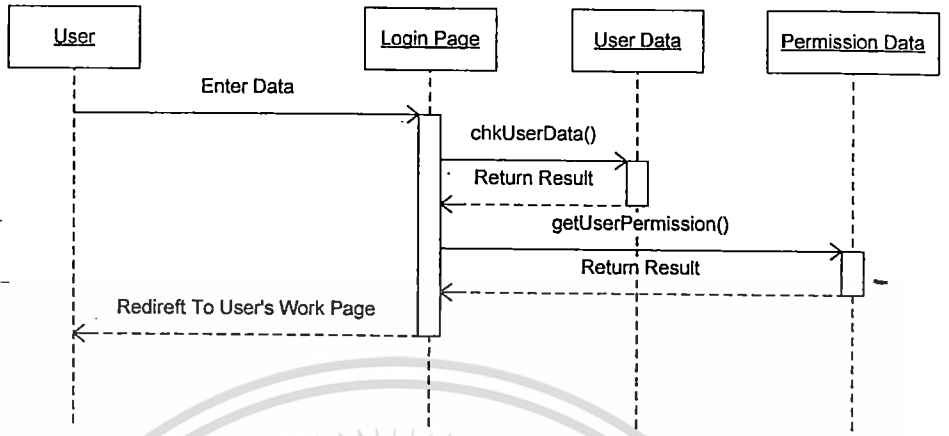
Message2

เป็นสัญลักษณ์ของเมสเสจนี้ โดยเป็นการแสดงที่ตรงกันข้ามกับ Synchronous คือไม่ต้องรอคอยการตอบกลับจากอ๊อบเจทที่ติดต่อไป ยังสามารถดำเนินงานของอ๊อบเจท

- Return

Message3

เป็นสัญลักษณ์ของเมสเสจนี้ โดยแสดงถึงการส่งกลับ

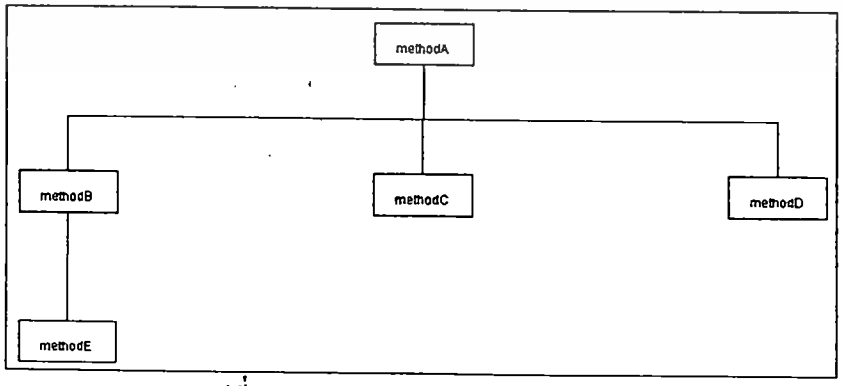


รูปที่ 2.19 แสดงตัวอย่างของ ซีเควนซ์ไดอะแกรม

2.5 Structure Chart

Structure Chart เป็นมุมมองระดับเมธตอดสคือเป็นแผนภาพที่แสดงให้เห็นถึงโครงสร้างการทำงานของแต่ละเมธตอด โดยที่ Structure Chart = (N,E) ขณะที่ N คือเซตของเมธตอด ซึ่งใช้สัญลักษณ์สี่เหลี่ยมผืนผ้าแทนแต่ละเมธตอด โดยภายในสัญลักษณ์สี่เหลี่ยมผืนผ้าจะมีชื่อของเมธตอดประกอบอยู่ และขณะที่ $E \subseteq N \times N$ คือเซตของแมจเสลที่ส่งจากเมธตอดหนึ่งไปยังอีกเมธตอดหนึ่ง โดยจะใช้ลูกศรแนวนอนที่ชี้จากเมธตอดหนึ่งไปอีกเมธตอดหนึ่ง ซึ่งจะมีชื่อพารามิเตอร์ที่ส่งไปกำกับอยู่บนลูกศรเพื่อแสดงว่าแมจเสลที่ส่งไปนั้นมีการส่งพารามิเตอร์ใดไปบ้าง

ในกรณีที่โครงสร้างของโปรแกรมมีลักษณะเป็นแบบการทำซ้ำนั้น (iteration) ข้อความที่กำกับอยู่บนลูกศรจะใช้สัญลักษณ์ * และในกรณีที่โครงสร้างของโปรแกรมมีลักษณะเป็นแบบการเลือกทำนั้น (selection) ข้อความที่กำกับอยู่บนลูกศรจะใช้สัญลักษณ์ [] เพื่อแสดงถึงเงื่อนไข



รูปที่ 2.20 แสดงตัวอย่าง Structure Chart

2.6 Parser

ตัวผู้วิเคราะห์(parser) คือ อ๊อบเจทที่แสดงโครงสร้างของภาษาและแปลแต่ละโครงสร้างไปสู่ผลลัพธ์ที่มีประโยชน์ (Metsker. 2001)

2.6.1 XML Parser

2.6.1.1 DOM (Document Object Model) (สราวุธ. 2544.)

DOM เป็นกระบวนการดึงข้อมูลในเอกสารเอ็กซ์เอ็มแอล มาใช้งานในแอพพลิเคชัน โดยมีหลักการคือ นำข้อมูลจากเอกสารมาวางเป็นโครงสร้างลักษณะลำดับชั้นต้นไม้ในหน่วยความจำของเครื่องคอมพิวเตอร์ แล้วใช้วิธีการท่องไปในต้นไม้เพื่อดึงข้อมูลมา ซึ่งวิธีการโหลดข้อมูลทั้งเอกสารเอ็กซ์เอ็มแอลมาไว้ในหน่วยความจำนั้นทำให้เกิดเป็นข้อเสียอย่างหนึ่งของ DOM เพราะจะกลายเป็นข้อจำกัดในเรื่องของหน่วยความจำในกรณีที่เอกสารเอ็กซ์เอ็มแอล มีขนาดใหญ่มาก ๆ จะต้องเปลืองเนื้อที่ในหน่วยความจำมาก และอาจจะไม่มีเนื้อที่เหลือพอที่จะทำงาน

2.6.1.2 SAX (Simple API for XML) (สุวัฒนา. 2545.)

SAX ได้ถูกพัฒนาขึ้นมาเพื่อการวิเคราะห์เอกสารทางเอ็กซ์เอ็มแอลที่มีขนาดใหญ่มาก ให้มีประสิทธิภาพมากขึ้น เพราะว่าการโหลดเอกสารขนาดใหญ่มาไว้ในหน่วยความจำจะทำให้ต้องเสียเนื้อที่การทำงานมากเกินความจำเป็น จึงทำให้ SAX เกิดขึ้นมาเพื่อแก้ปัญหานี้

การทำงานของ SAX เป็นกระบวนการดึงข้อมูลที่มีวิธีการดังนี้ อ่านเอกสารเอ็กซ์เอ็มแอลตั้งแต่ต้นเอกสาร และตัวผู้วิเคราะห์ จะสร้างเหตุการณ์ให้กับจุดต่าง ๆ ที่สำคัญของเอกสารทุก ๆ จุด จึงเรียกการทำงานแบบนี้ว่า Event – Driven Parser ดังนั้นการทำงานกับ SAX จะต้องมี XML Parser ที่รองรับการทำงานนี้ด้วย

2.6.2 XMI Parser(Grose. et al. 2002.)

ในการวิเคราะห์เอกสารทางเอ็กซ์เอ็มไอ สามารถทำได้โดยการใช้ API(Application Programming Interface คือ interface ของ software) ซึ่ง XMI Framework เป็น API เช่นกันและเป็น API ที่ถูกออกแบบเพื่อการสนับสนุนเอ็กซ์เอ็มไอโดยเฉพาะ นอกจากนี้ยังมี API อื่นๆที่มีความสามารถตรงนี้ อย่างเช่น DOM(Document Object Model) เป็น API หนึ่งที่ถูกออกแบบให้ทำการสนับสนุน XML หรือ JOB(Java Object Bridge) เป็น API ที่ถูกออกแบบเพื่อการสนับสนุนทาง XMI กับลักษณะทาง Java คือ JOB อนุญาตให้ทำการสร้างและเก็บเอกสารทางเอ็กซ์เอ็มไอ ที่บรรจุวัตถุทางจาวา

2.6.3 Parser Generator

การสร้างตัวผู้วิเคราะห์สำหรับภาษาใดภาษาหนึ่งนั้น สามารถใช้ตัวจำแนก(generator) ในการสร้างตัวผู้วิเคราะห์ โดยอาจจะต้องเข้าใจความรู้พื้นฐานของ Programming Language เบื้องต้น ตัวจำแนกตัวผู้วิเคราะห์(parser generator). สามารถดาวน์โหลดได้ ซึ่งมีตัวจำแนกอยู่อย่างหลากหลาย อย่างเช่น ATLR(Another Tool for Language Recognition), YACC(Yet Another Compiler Compiler), และ JAVACC(Java Compiler Compiler) เป็นต้น โดย JAVACC มีความสามารถที่เป็นทั้งตัวจำแนกตัวผู้วิเคราะห์และ Lexical Analyzer และได้นำมาประยุกต์ในโครงงานพัฒนาเครื่องมือนี้

2.7 UML and Java

ในการแปลงโค้ดภาษาจาวาไปเป็นแผนภาพที่อิงกับมาตรฐานยูเอ็มแอลนั้น ต้องศึกษาลักษณะที่เหมือนกันหรือต่างกันของทั้งสอง โดยแสดงได้ดังตารางที่ 2.3(Grose. et al. 2002.) ที่แสดงแนวคิดหลักๆของทั้งยูเอ็มแอลและจาวา

ตารางที่ 2.3 เป็นการแสดงการเปรียบเทียบแนวคิดของยูเอ็มแอลและจาวา

UML concept	Java concept
Class	Class
Attribute	Attribute
Association	None
Single inheritance	Inheritance
Multiple inheritance	None
Instance	Instance
Package	Package
Datatype	Primitive type

จากตารางที่ 2.3 เห็นได้ว่าการเปรียบเทียบแนวคิดของยูเอ็มแอลและจาวา จะเห็นได้ว่า สิ่งที่แนวคิดของจาวาไม่มีในแนวคิดของยูเอ็มแอล คือ Association กับ Multiple inheritance จากปัญหาตรงนี้จึงได้ทำการวิเคราะห์แนวทางการแก้ปัญหาความตรงกันของทั้งแนวคิด โดยที่การวิเคราะห์การปัญหานั้นได้ทำการวิเคราะห์โดยอิงมุมมองของคลาสไดอะแกรม

Association ไม่มีในแนวคิดของจาวาแต่สามารถวิเคราะห์ได้จากตัวโค้ดของภาษาจาวา โดยดูจากแอตทริบิวต์ของคลาสว่ามีชนิดของข้อมูลเป็นแบบใด(ชนิดของข้อมูลในส่วนของแอตทริบิวต์ของภาษาจาวาสามารถแบ่งออกได้เป็น 2 ส่วน คือ แบบที่หนึ่งเป็นแบบที่เรียกว่า primitive type อย่างเช่น int, boolean, และ double เป็นต้น และแบบที่สองเป็นแบบที่เรียกว่า reference type) ถ้าแอตทริบิวต์นั้นๆมีชนิดของข้อมูลเป็นแบบ reference type แล้วจัดได้ว่าคลาสนั้นมีความสัมพันธ์แบบ Association

นอกจากความสัมพันธ์แบบ Association แล้ว ในมุมมองของคลาสไดอะแกรมยังมีมุมมองแบบอื่นๆที่ใช้ในแผนภาพแบบคลาสไดอะแกรม ดังนี้

- ความสัมพันธ์แบบ Dependency
  สามารถทำการวิเคราะห์ความสัมพันธ์แบบ Dependency ที่เป็นการแสดงออกในรูปแบบของโค้ดที่เป็นภาษาจาวาได้โดยดูประโยคของ import statement
- ความสัมพันธ์แบบ Aggregation และ Composition
  เป็นลักษณะของความสัมพันธ์ที่วิเคราะห์ได้ยาก ขึ้นอยู่กับการวิเคราะห์ของผู้ใช้ โดยที่ Aggregation เป็นรูปแบบของความสัมพันธ์ Association แต่เป็นลักษณะของความสัมพันธ์ที่เรียกว่า whole/part ความสัมพันธ์ระหว่าง Aggregation และ Association ในจาวานั้นจัดได้ว่าไม่มีความแตกต่างแต่อย่างใดในแง่ของความหมาย จึงทำให้มีการใช้ Association มากกว่า Aggregation (Knoernschild. 2000.) ส่วน Composition คือ การที่คลาสหนึ่งคลาสใดเป็นส่วนหนึ่งของคลาสอื่น (ธนา. 2002.) สามารถจัดว่า Composition เป็น Aggregation ในแง่ whole/part ที่ผูกพันกันอย่างแข็งแกร่ง

Multiple inheritance ไม่มีในแนวคิดของภาษาจาวา โดยในมุมมองของแนวคิดจาวานั้นมองว่าการที่มีการถ่ายทอด(inheritance) มาจากหลายๆคลาสนั้นสามารถเกิดปัญหาได้จึงมีแนวทางในการแก้ปัญหานี้ โดยการนำอินเตอร์เฟสมาช่วยในการใช้ Multiple inheritance การถ่ายทอดในภาษาจาวาสามารถวิเคราะห์จากตัวโค้ดจาวาได้โดยดูจากคีย์เวิร์ด คือ extends(ตรงกับความสัมพันธ์แบบ Generalization ของคลาสไดอะแกรม) ส่วนการใช้อินเตอร์เฟสนั้นมีคีย์เวิร์ดที่ใช้คือ implements (ตรงกับความสัมพันธ์แบบ Realization ของคลาสไดอะแกรม)

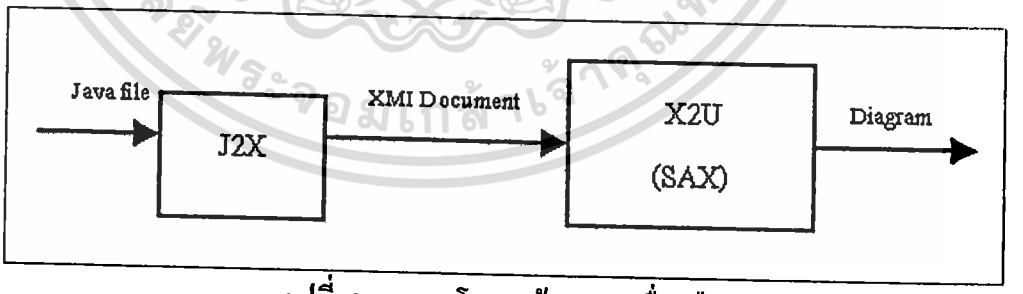
บทที่ 3

การวิเคราะห์และออกแบบระบบงาน

3.1 การออกแบบโครงสร้างของเครื่องมือในโครงการพัฒนาระบบ

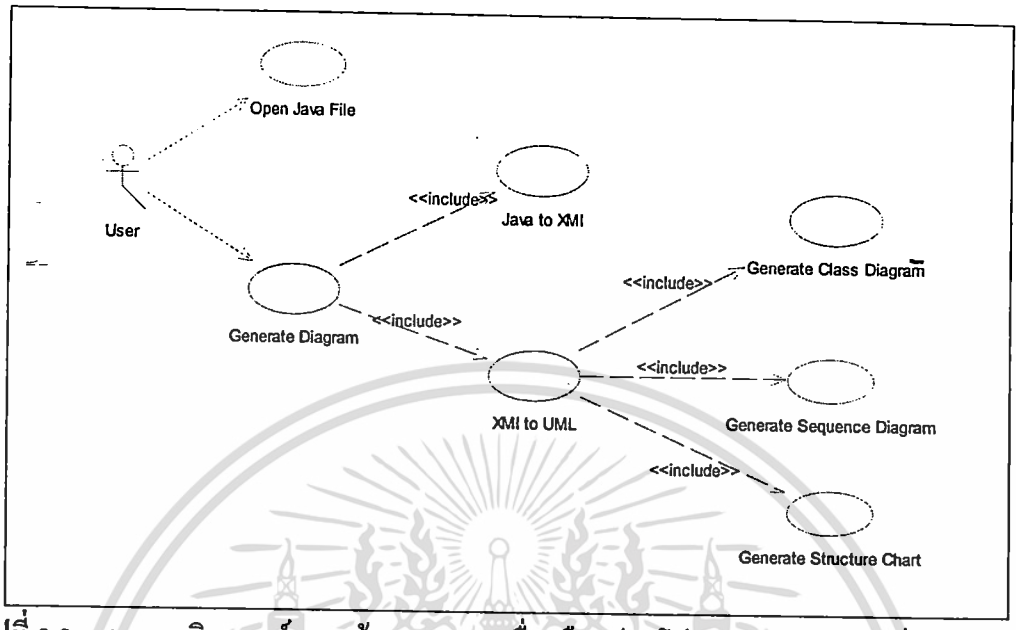
เครื่องมือที่สร้างในโครงการพัฒนาระบบได้ถูกตั้งชื่อว่า JMD ซึ่งย่อมาจากคำว่า Java Map to Diagram โครงสร้างที่เป็นภาพรวมของเครื่องมือนี้ได้แสดงไว้ในรูปที่ 3.1 โดยมีการแบ่งการทำงานของเครื่องมือออกเป็น 2 ส่วนหลักๆ ดังนี้

- ส่วนที่เป็นการแปลงข้อมูลจากจาวาเป็นเอกสารทางเอ็กซ์เอ็มไอ โดยเครื่องมือนี้มีอินพุตเป็น ไฟล์จาวา(.java) ต่อจากนั้นไฟล์จาวานั้นๆผ่านโปรแกรมที่มีชื่อว่า J2X(Java to XMI) ซึ่งผลลัพธ์ที่ได้คือ เอกสารเอ็กซ์เอ็มไอ(XMI Document)
- นำผลลัพธ์ที่เป็นเอกสารทางเอ็กซ์เอ็มไอมาทำการวิเคราะห์ด้วยตัวผู้วิเคราะห์ที่มีความสามารถในการวิเคราะห์เอ็กซ์เอ็มแอล เนื่องจากเอ็กซ์เอ็มไอเป็นมาตรฐานที่ตั้งอยู่บนพื้นฐานของเอ็กซ์เอ็มแอลนั้นเอง จึงสามารถนำตัวผู้วิเคราะห์มาทำการวิเคราะห์ได้ ในการเลือกตัวผู้วิเคราะห์นั้นเครื่องมือนี้ใช้ตัวผู้วิเคราะห์ที่มีชื่อว่า SAX ผลลัพธ์ที่ได้จากการทำการวิเคราะห์นั้นจะถูกนำไปประมวลผลเพื่อแสดงผลในรูปแบบแผนภาพ



รูปที่ 3.1 แสดงโครงสร้างของเครื่องมือ

3.2 การวิเคราะห์ความต้องการของเครื่องมือแปลงจาวาเป็นแผนภาพ



รูปที่ 3.2 แสดงการวิเคราะห์ความต้องการของเครื่องมือแปลงโปรแกรมภาษาจาวาเป็นแผนภาพ

จากแผนภาพแสดงการวิเคราะห์ความต้องการของเครื่องมือนี้จะประกอบไปด้วย

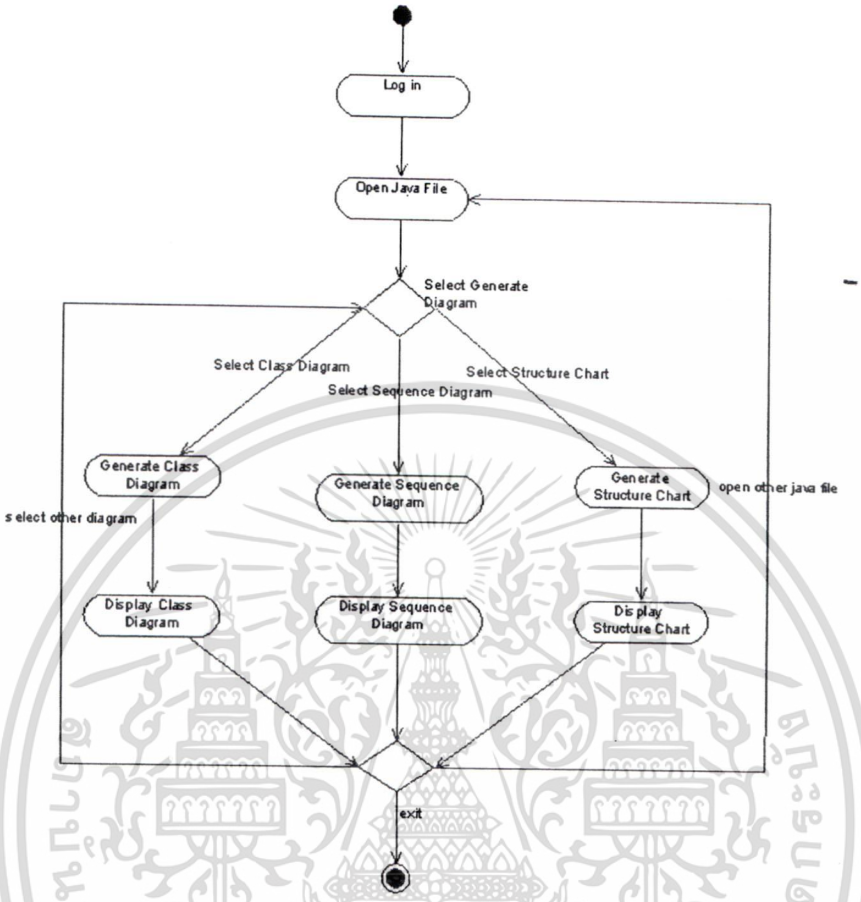
- แอคเตอร์ (Actor) เป็นส่วนที่แสดงถึงบุคคลหรือระบบอื่น ๆ ที่เกี่ยวข้องกับระบบ สำหรับใน Use Case Diagram ของระบบนี้จะมี 1 แอคเตอร์ คือ ผู้ใช้ (User) ซึ่งหมายถึง ผู้พัฒนาระบบหรือผู้ทดสอบระบบที่ต้องการสร้างแผนภาพจากโปรแกรมภาษาจาวา ซึ่ง แอคเตอร์นี้จะเป็นผู้ใช้งานระบบนี้โดยตรง
- ยูสเคส (Use Case) เป็นส่วนของฟังก์ชั่นการทำงานหลักของระบบ ซึ่งประกอบไปด้วย รายละเอียด ดังตารางที่ 3.1

ตารางที่ 3.1 แสดงหน้าที่การทำงานของยูสเคส

ยูสเคส	หน้าที่การทำงาน
Open Java File	เปิดโปรแกรมภาษาจาวา
Generate Graph	สร้างแผนภาพจากโปรแกรมภาษาจาวา
Java to XMI	แปลงโปรแกรมภาษาจาวาเป็นเอกสาร XMI
XMI to UML	การแปลงเอกสาร XMI เป็นแผนภาพ
Generate Class Diagram	สร้าง Class Diagram
Generate Sequence Diagram	สร้าง Sequence Diagram
Generate Structure Chart	สร้าง Structure Chart

3.3 ขั้นตอนการพัฒนาระบบ

ในการพัฒนาเครื่องมือนี้จะใช้ Activity Diagram เพื่ออธิบายขั้นตอนการทำงานของระบบ โดยมีรายละเอียดดังนี้คือ เมื่อผู้ใช้งานได้เปิดเครื่องมือนี้ขึ้นมาผู้ใช้สามารถเปิดโปรแกรมภาษาจาวาที่จะนำไปแปลงเป็นแผนภาพที่ต้องการ ดังนั้นผู้ใช้จะต้องเลือกประเภทแผนภาพที่ต้องการจะแสดง เช่น Class Diagram , Sequence Diagram หรือ Structure Chart หลังจากที่ผู้ใช้เลือกแผนภาพที่จะแสดงแล้วเครื่องมือจะทำการสร้างแผนภาพตามประเภทที่ผู้ใช้เลือกที่สอดคล้องกับโปรแกรมภาษาจาวาให้ หลังจากนั้นผู้ใช้สามารถที่จะเลือกประเภทของแผนภาพที่จะแสดงใหม่ได้ หรือผู้ใช้สามารถที่จะเลือกโปรแกรมภาษาจาวาที่จะนำไปแปลงเป็นแผนภาพใหม่ได้ โดยรูปที่ 3.3 เป็นแสดง Activity Diagram



รูปที่ 3.3 Activity Diagram ของเครื่องมือสร้างแผนภาพจากโปรแกรมภาษาจาวา

3.4 Java to XMI Document(J2X)Class Diagram

ในการออกแบบส่วนของการแปลงโค้ดภาษาจาวาไปเป็นเอกสารทางเอ็กซ์เอ็มไอ มีการนำ JavaCC มาช่วยในการแปลงส่วนนี้ด้วย

ในการวิเคราะห์ซอร์สโค้ดภาษาจาวาหรือข้อมูลที่มีลักษณะเป็นเท็กซ์ไฟล์นั้น มีการนำข้อมูลเหล่านั้นมาทำการแยกออกเป็นส่วนๆ(ที่เรียกว่า token) แล้วนำสิ่งที่ได้ไปวิเคราะห์ให้เกิดความเข้าใจในโครงสร้างหรือความหมายของข้อมูลนั้นๆ ตัวอย่างเช่น class ShowToken {int left,right;} สามารถทำการวิเคราะห์ออกเป็น token ได้ดังนี้ “class”, “ “, “ShowToken”, “ “, “{“, “int”, “ “, “left”, “,”, “right”, “;”, และ “}” ซึ่งกระบวนการวิเคราะห์อย่างนี้ เรียกว่า Lexical Analysis ต่อจากนั้นจึงนำผลลัพธ์ที่ได้เหล่านี้ไปทำการวิเคราะห์(parsing) ให้เกิดความเข้าใจในความหมาย โครงสร้าง หรือนำไปสู่ผลลัพธ์อื่นๆ

ความสามารถในการทำ Lexical Analysis นี้ในภาษาจาวาได้จัดทำคลาสที่เป็นมาตรฐานที่บรรจุไว้ในภาษาจาวาถึง 2 คลาส คือ StringTokenizer (ใช้กับอ๊อบเจทที่เป็นสตริง) และ StreamTokenizer(ใช้กับอ๊อบเจทที่เป็น InputStream) นอกจากนี้ยังมีอีกแนวทางหนึ่งที่มีความสามารถในการช่วยทำ Lexical Analysis คือการใช้ตัวจำแนก(generator) ที่มีความสามารถในการจำแนก Lexical Analyzer ได้ด้วย

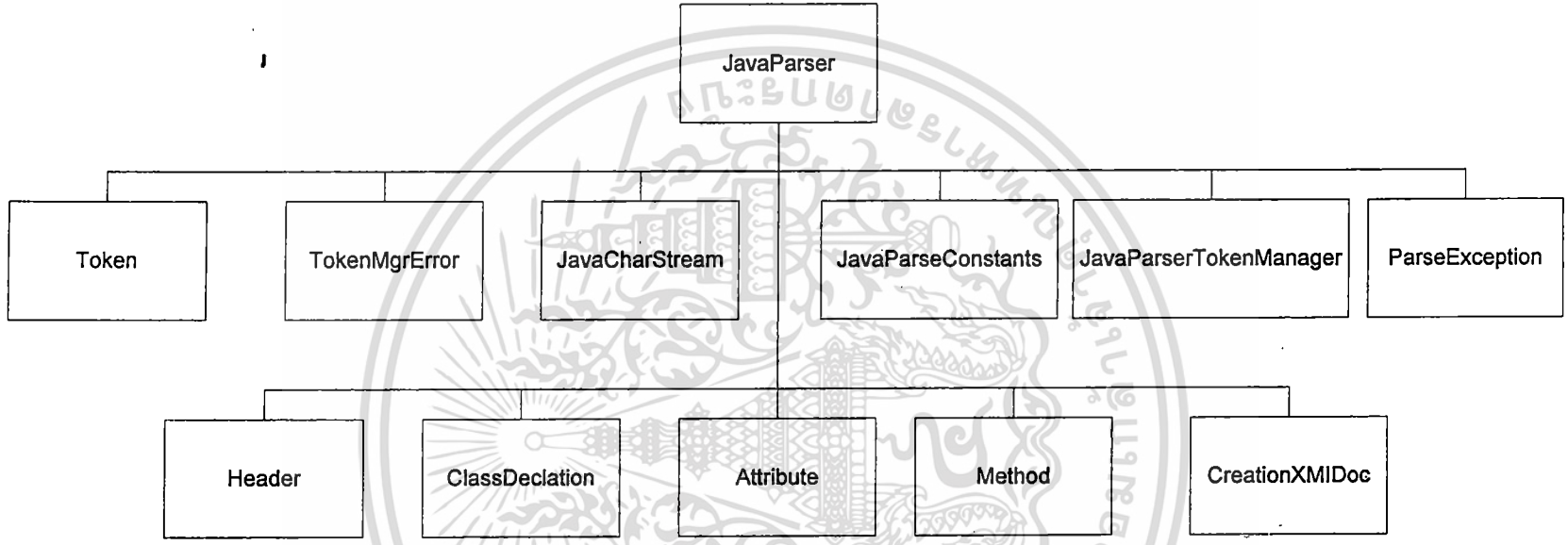
JavaCC (ย่อมาจาก Java Compiler Compiler) เป็นเครื่องมือที่มีความสามารถที่เป็นทั้ง Lexical Analyzer generator และ parser generator(เป็นโปรแกรมที่ยอมรับข้อกำหนดไวยากรณ์ในฐานะที่เป็นอินพุตและจำแนก parser สำหรับไวยากรณ์นั้นๆให้เป็นเอาท์พุต) นอกจากนี้ยังเป็นเครื่องมือที่เป็นฟรีแวร์ อีกทั้งยังสามารถดาวน์โหลดไฟล์ของ JavaCC(ที่เป็น .jj) ที่เป็นรูปแบบสำเร็จในการวิเคราะห์ในการวิเคราะห์ภาษาต่างๆได้ อย่างเช่น XML, Java, และ HTML เป็นต้น

การออกแบบในส่วนของการแปลงจาวาเป็นเอกสารทางเอ็กซ์เอ็มไอนี้ได้นำไฟล์ของ JavaCC ที่มีชื่อว่า java1.4.jj มาประยุกต์ใช้

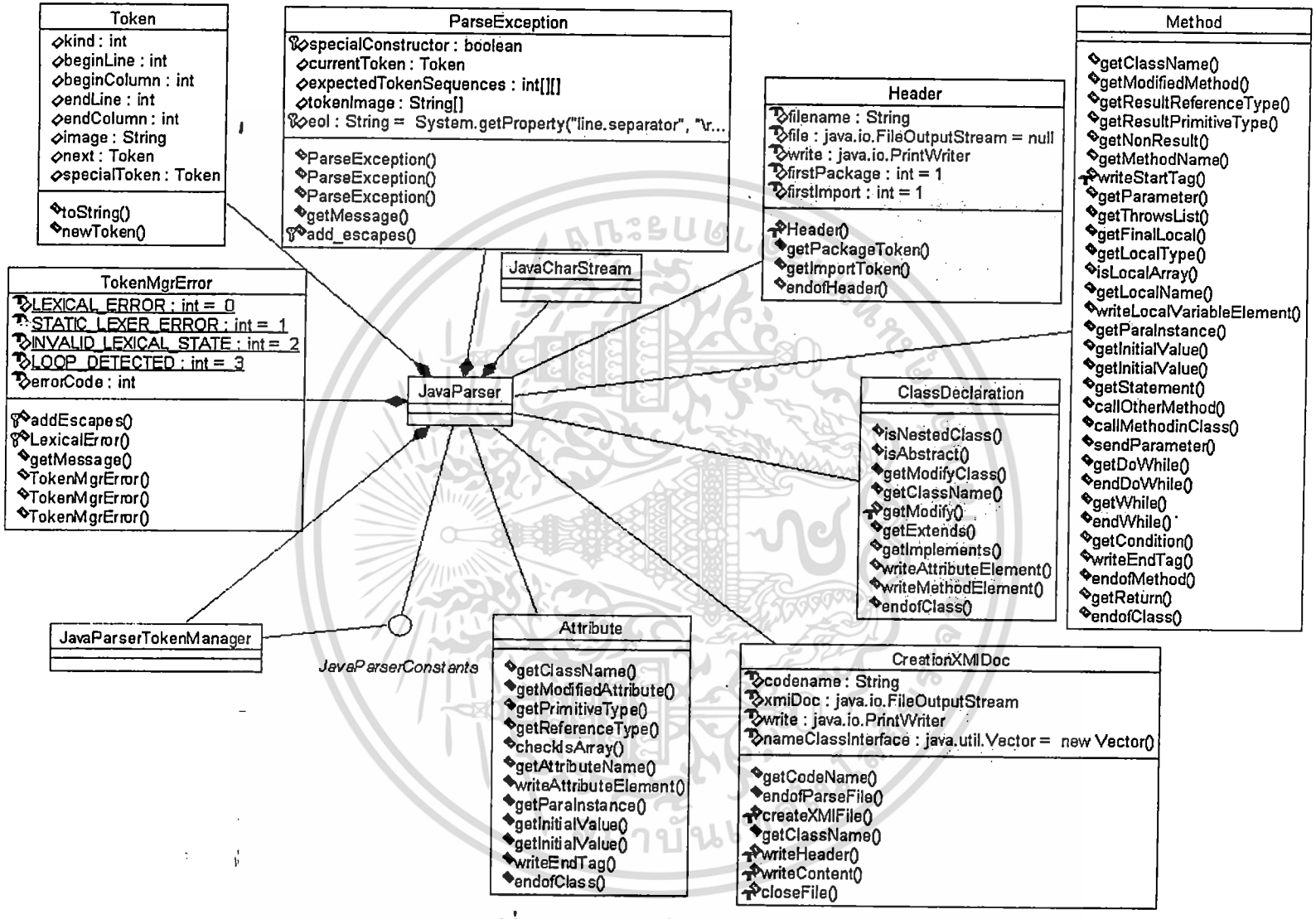
java1.4.jj เป็นตัวจำแนก Lexical Analyzer และตัวผู้วิเคราะห์ที่ความสามารถในการวิเคราะห์โค้ดภาษาจาวา โดยที่ไฟล์นี้เป็นไฟล์ที่มีการปรับปรุงอยู่ตลอด คือ เดิมทีนั้นเป็น java1.1.jj ซึ่งเป็นไฟล์ที่เขียนขึ้นตามรูปแบบและลักษณะการใช้งานของ JDK1.0 และเมื่อ JDK(Java Developer Kit) มีการพัฒนาเป็นเวอร์ชั่นต่างๆเรื่อยมาจน ทำให้ java1.1.jj ได้เพิ่มเติมรูปแบบลักษณะต่างๆตามJDK เช่นกัน จนกระทั่ง java1.4.jj เป็นการพัฒนาให้เหมาะกับรูปแบบและลักษณะการใช้ของ JDK 1.4

การนำ java1.4.jj มาช่วยในการทำงานที่ทำการวิเคราะห์โค้ดภาษาจาวา โดยได้คงส่วนของ Lexical Specification ไว้ นอกนั้นได้ทำการปรับปรุงเปลี่ยนแปลงให้มีความเหมาะสมกับวัตถุประสงค์ในการสร้างเครื่องมือนี้ สามารถดูรายละเอียดของไฟล์ .jj (ที่ถูกตั้งชื่อว่า J2X.jj) เป็นการนำ java1.4.jj มาเป็นแนวทางในการสร้าง

คลาสที่ได้จากการจำแนก J2X.jj(มี7 คลาส คือ JavaParser, JavaParserConstants, JavaParserTokenManager, ParseException, Token, JavaCharStream, และ TokenMgrError) และคลาสที่อื่นๆที่ทำหน้าที่ในการแมพภาษาจาวาไปเป็นเอกสารทางเอ็กซ์เอ็มไอ สามารถแสดงรายละเอียดในรูปแบบของ Structure Chart ได้ดังรูปที่ 3.4 และรูปที่ 3.5 แสดงคลาสไดอะแกรมของ J2X



รูปที่ 3.4 แสดง Structure Chart ของการแปลงโค้ดจาวาไปสู่เอกสารทางเอ็กซ์เอ็มไอ



รูปที่ 3.5 แสดงคลาสไดอะแกรมของ J2X

สำหรับรายละเอียดของต่างๆของคลาสไดอะแกรม มีดังนี้

- JavaParser มีเมธตอดที่สำคัญดังนี้

เมธตอด	หน้าที่
parseCode(filename: String)	เป็นเมธตอดที่รับข้อมูลจากอินเตอร์เฟส โดยเป็นการส่งชื่อของไฟล์ที่ต้องการจำแนกแผนภาพ
PackageDeclaration()	วิเคราะห์ package statement
ImportDeclaration()	วิเคราะห์ import statement
TypeDeclaration()	วิเคราะห์ว่าสิ่งที่ถูกวิเคราะห์เป็นอินเตอร์เฟสหรือคลาส
ClassDeclaration()	วิเคราะห์ส่วนของโค้ดที่เป็นการประกาศว่าเป็นคลาส
InterfaceDeclaration()	วิเคราะห์ส่วนของโค้ดที่เป็นการประกาศว่าเป็นอินเตอร์เฟส
FieldDeclaration()	วิเคราะห์ในส่วนของแอตทริบิวต์
MethodDeclaration()	วิเคราะห์ในส่วนของเมธดอต

- Token เป็นคลาสที่เป็นตัวแทน token ในแต่ละอ๊อบเจท Token (สิ่งที่ระบุว่าเป็น TOKEN ใน Lexical Specification จะถูกสร้างเป็นอ๊อบเจท Token)

เมธตอด	หน้าที่
toString() : String	แปลงอ๊อบเจทให้เป็นสตริงโดยการส่งค่ากลับ
newToken(ofKind : int)	นำค่าที่เป็นตัวเลข(ที่ได้มาจากการสร้าง Lexical Specification) ให้เป็นสตริง

- JavaParserConstants เป็นอินเตอร์เฟสคลาส โดยที่คลาสนี้ไม่มีเมธตอด เป็นอินเตอร์เฟสคลาสที่เก็บค่าแอตทริบิวต์ต่างๆ(เกี่ยวกับส่วน Lexical Specification)

- JavaCharStream เป็นคลาสที่แสดงสตรีมของ input character โดยมีการติดต่อกับ JavaParser class ที่รับไฟล์ที่จะทำการวิเคราะห์

- JavaParserTokenManager เป็น Lexical Analyzer

- ParseException เป็นคลาสที่ตรวจจับข้อผิดพลาดที่เกี่ยวกับรูปแบบ Grammar Specification

เมธตอด	หน้าที่
getMessage() : String	จับความผิดพลาดในการวิเคราะห์ token ในไฟล์นั้นพร้อมแสดงความผิดพลาดว่าเกิดจากการไม่สามารถวิเคราะห์ token ตัวใดที่ไม่ตรงตาม รูปแบบที่ระบุไว้ใน Grammar Specification

- TokenMgrError เป็นคลาสที่จัดการสิ่งที่ผิดพลาดในการวิเคราะห์ token

เมธตอด	หน้าที่
LexicalError() : String	รายงานความผิดพลาดที่เกี่ยวกับ Lexical Specification ว่าไม่สามารถวิเคราะห์ token ได้ที่ token ใด

- Header เป็นคลาสที่จัดการในส่วนของต้นโปรแกรมของไฟล์ที่ถูกวิเคราะห์ ได้แก่ package และ import statement

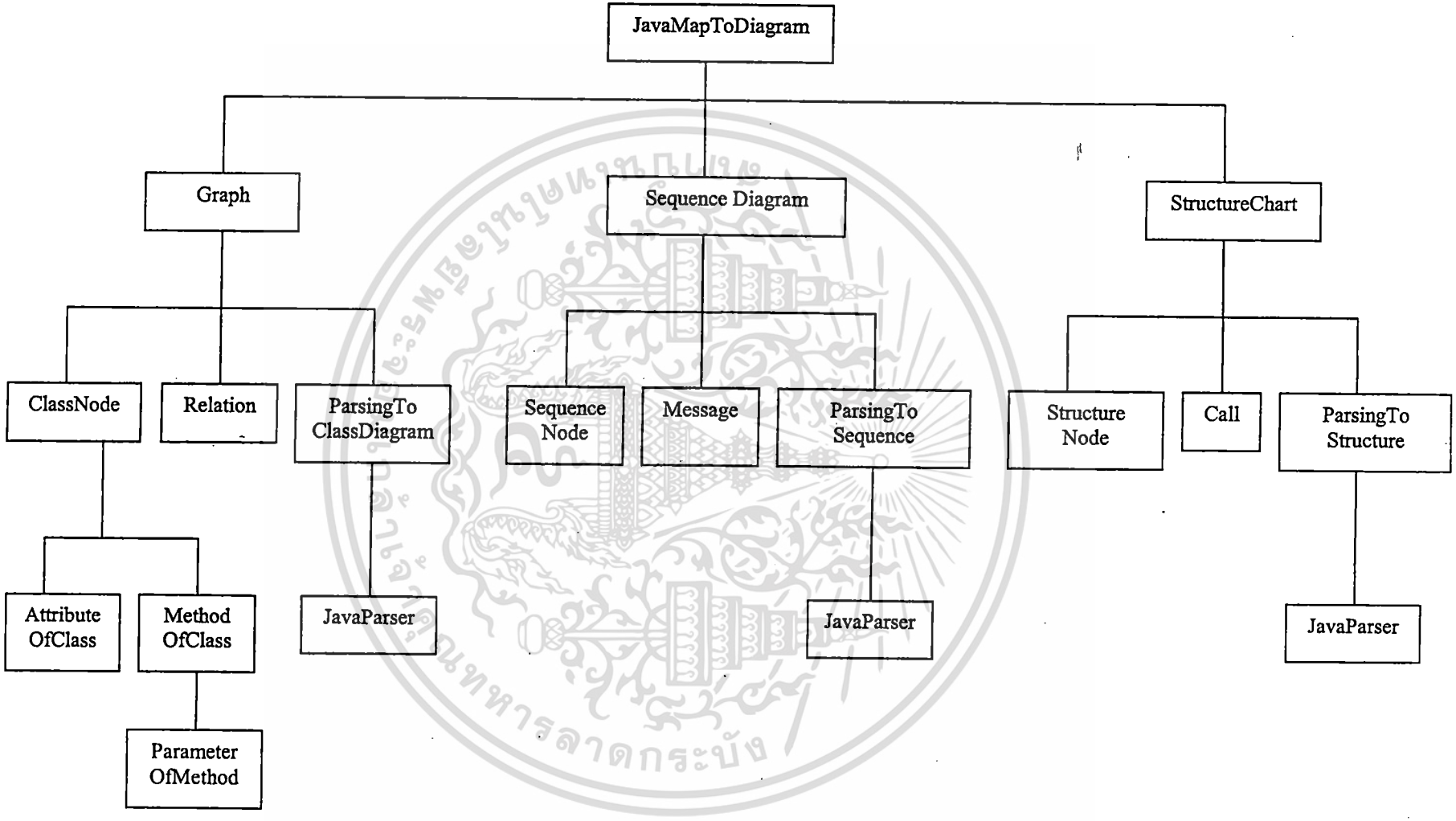
เมธตอด	หน้าที่
Header(name : String)	- รับค่าที่ชื่อไฟล์ที่ต้องการวิเคราะห์ - เก็บข้อมูลที่เป็นรายละเอียดต่างๆลงไฟล์ในรูปแบบของอิลิเมนต์ import และ package
getPackageToken(packageTok :Vector)	- เก็บข้อมูลข้อมูลของ package statement
checkClass(String instanceObject)	- เป็นการตรวจสอบว่าชื่อคลาสเป็นส่วนของคลาสที่อยู่ใน java library หรือเป็นคลาสที่สร้างขึ้นเอง
getImportToken(packageTok :Vector)	- เก็บข้อมูลข้อมูลของ import statement

- ClassDeclaration

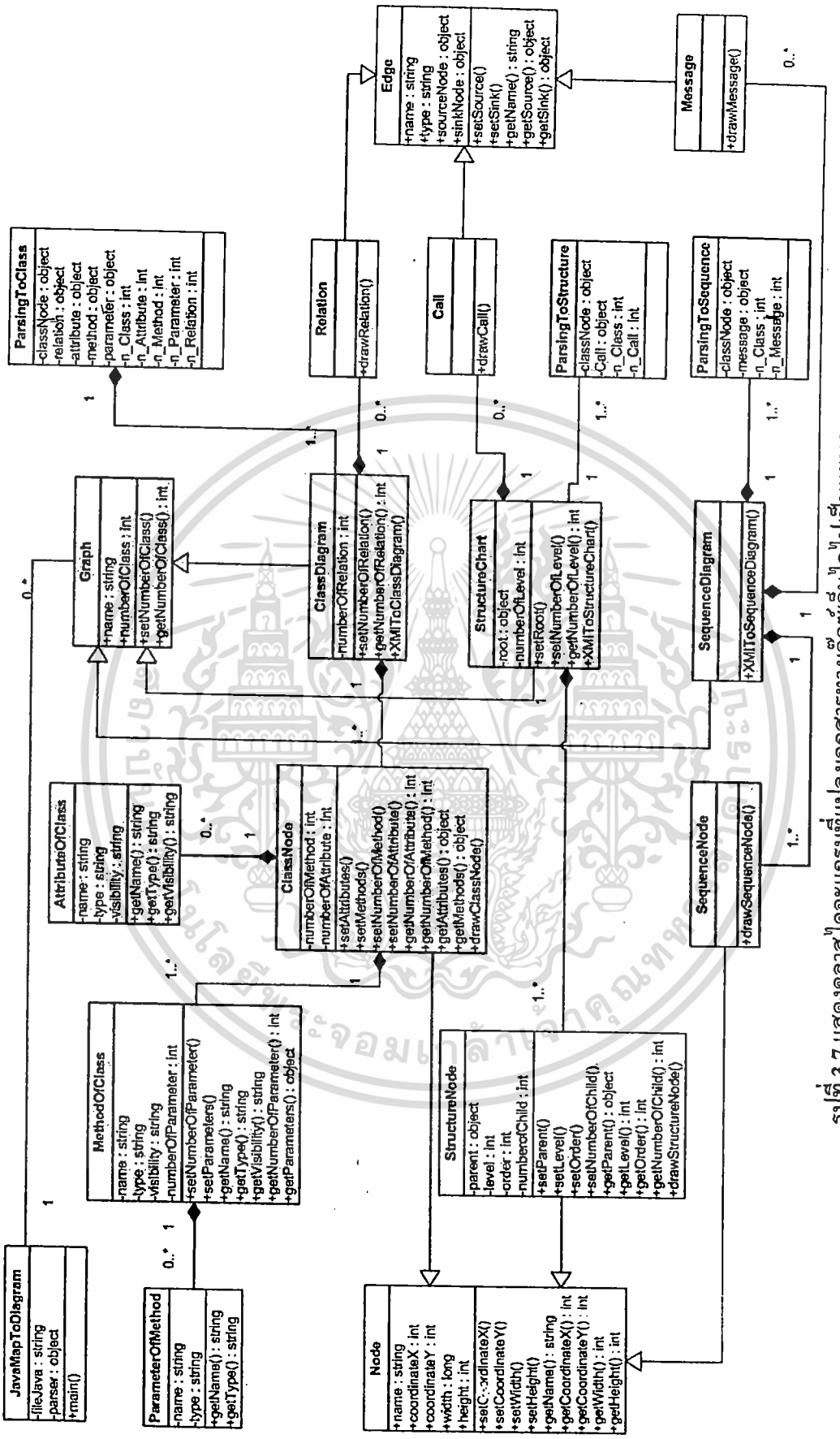
เมธตอด	หน้าที่
isAbstract()	- ตรวจสอบว่าคลาสนั้นๆเป็นคลาส abstract
getModifyClass(modFinal : String, modPublic : String, defStrictfp : String)	- เก็บข้อมูลที่เป็น modifier ของคลาส
getExtends(name : String)	- รับข้อมูลที่เป็นชื่อของคลาสที่เป็นคลาสแม่
getImplements()	- รับข้อมูลที่เป็นรายชื่อของ อินเตอร์เฟสต่างๆที่มีการใช้
writeAttributeElement()	- เขียนแอตทริบิวต์ของคลาส
writeMethodElement()	- เขียนเมธตอดของคลาส

3.5 XMI Document to Diagram (X2U)

ในส่วนของ X2U เป็นการแปลงเอกสารทางเอ็กซ์เอ็มไอไปเป็นแผนภาพ โดยรูปที่ 3.5 เป็นการแสดง Structure chart ที่ทำงานในการวิเคราะห์เอกสารทางเอ็กซ์เอ็มไอไปสู่แผนภาพรูปที่ 3.6 เป็นการแสดงลักษณะของคลาสไดอะแกรมของส่วนงานที่แปลงเอกสารทางเอ็กซ์เอ็มไอไปเป็นแผนภาพ สำหรับรายละเอียดของ Class Diagram ในส่วนของ X2U มีดังต่อไปนี้



รูปที่ 3.6 แสดง Structure Chart ของการแปลงเอกสารทางเอ็กซ์เอ็มไอเป็นแผนภาพ



รูปที่ 3.7 แสดงคลาสไดอะแกรมที่แปลงเอกสารทางเอกซ์เอ็มไอไปเป็นแผนภาพ

- JavaMapToDiagram มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
main()	จัดการการทำงานต่าง ๆ ในเครื่องมือนี้ เช่นเลือกเปิดไฟล์ภาษาจาวาจากเมนู , ออกจากเครื่องมือนี้ เป็นต้น

- Graph มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setNumberOfClass(int n)	กำหนดค่าจำนวนคลาสทั้งหมดในแผนภาพนั้น ๆ
getNumberOfClass():int	คืนค่าจำนวนคลาสทั้งหมดในแผนภาพนั้น ๆ

- ClassDiagram เป็นคลาสที่ได้รับการถ่ายทอดคุณสมบัติมาจากคลาส Graph ซึ่งมีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setNumberOfRelation(int n)	กำหนดค่าจำนวนความสัมพันธ์ทั้งหมดในแผนภาพนั้น ๆ
getNumberOfRelation():int	คืนค่าจำนวนความสัมพันธ์ทั้งหมดในแผนภาพนั้น ๆ
XMIToClassDiagram()	อ่านเอกสาร XMI ขึ้นมาวิเคราะห์เพื่อสร้าง Class Diagram

- SequenceDiagram เป็นคลาสที่ได้รับการถ่ายทอดคุณสมบัติมาจากคลาส Graph ซึ่งมีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
XMIToSequenceDiagram()	อ่านเอกสาร XMI ขึ้นมาวิเคราะห์เพื่อสร้าง Sequence Diagram

- StructureChart เป็นคลาสที่ได้รับการถ่ายทอดคุณสมบัติมาจากคลาส Graph ซึ่งมีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setRoot(StructureNode s)	กำหนดค่าโหนด root ของแผนภาพนั้น ๆ
setNumberOfLevel(int n)	กำหนดค่าจำนวน level ทั้งหมดในแผนภาพนั้น ๆ
getNumberOfLevel():int	คืนค่าจำนวน level ทั้งหมดในแผนภาพนั้น ๆ
XMIToStructureChart()	อ่านเอกสาร XMI ขึ้นมาวิเคราะห์เพื่อสร้าง Class Diagram

- AttributeOfClass มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
getName():String	คืนค่าชื่อของแอตทริบิวต์นั้น ๆ
getType():String	คืนค่าประเภทข้อมูลของแอตทริบิวต์นั้น ๆ
getVisibility():String	คืนค่า visibility ของแอตทริบิวต์นั้น ๆ

- Relation มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
drawRelation()	วาดเส้นที่แสดงถึงความสัมพันธ์ระหว่างคลาส

- ParameterOfMethod มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
getName():String	คืนค่าชื่อของพารามิเตอร์ตัวนั้น ๆ
getType():String	คืนค่าประเภทข้อมูลของพารามิเตอร์ตัวนั้น ๆ

- Node มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setCoordinateX(int x)	กำหนดค่าโคออร์ดิเนต X ของโหนดนั้น ๆ
setCoordinateY(int y)	กำหนดค่าโคออร์ดิเนต Y ของโหนดนั้น ๆ
setWidth(int w)	กำหนดค่าความกว้างของโหนดนั้น ๆ
setHeight(int h)	กำหนดค่าความสูงของโหนดนั้น ๆ
getName():String	คืนค่าชื่อของโหนดนั้น ๆ
getCoordinateX():int	คืนค่าโคออร์ดิเนต X ของโหนดนั้น ๆ
getCoordinateY():int	คืนค่าโคออร์ดิเนต Y ของโหนดนั้น ๆ
getWidth():int	ค่าความกว้างของโหนดนั้น ๆ
getHeight():int	คืนค่าความสูงของโหนดนั้น ๆ

- Edge มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setSource(Node n)	กำหนดค่าโหนดต้นทางของ edge นั้น ๆ
setSink(Node n)	กำหนดค่าโหนดปลายทางของ edge นั้น ๆ
getName():String	คืนค่าชื่อของ edge นั้น ๆ
getSource():Node	คืนค่าโหนดต้นทางของ edge นั้น ๆ
getSink():Node	คืนค่าโหนดปลายทางของ edge นั้น ๆ

- ClassNode มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setAttributes(AttributeOfClass[] a)	กำหนดค่าแอตทริบิวต์ของคลาสนั้น ๆ
setMethods(MethodOfClass[] m)	กำหนดค่าเมธตอดของคลาสนั้น ๆ
setNumberOfAttribute(int n)	กำหนดค่าจำนวนแอตทริบิวต์ทั้งหมดของคลาสนั้น ๆ
setNumberOf Method(int n)	กำหนดค่าจำนวนเมธตอดทั้งหมดของคลาสนั้น ๆ
getAttributes():AttributeOfClass	คืนค่าแอตทริบิวต์ของคลาสนั้น ๆ
getMethods():MethodOfClass	คืนค่าเมธตอดของคลาสนั้น ๆ
getNumberOfAttribute():int	คืนค่าจำนวนแอตทริบิวต์ทั้งหมดของคลาสนั้น ๆ
getNumberOf Method():int	คืนค่าจำนวนเมธตอดทั้งหมดของคลาสนั้น ๆ
drawClassNode()	วาดโหนดคลาสของ Class Diagram

- MethodOfClass มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setParameters(ParameterOfMethod[] p)	กำหนดค่าพารามิเตอร์ของเมธตอดนั้น ๆ
setNumberOfParameter(int n)	กำหนดค่าจำนวนพารามิเตอร์ของเมธตอดนั้น ๆ
getName():String	คืนค่าชื่อของเมธตอดนั้น ๆ
getType():String	คืนค่าประเภทข้อมูลของเมธตอดนั้น ๆ
getVisibility():String	คืนค่า visibility ของเมธตอดนั้น ๆ
getParameters():ParameterOfMethod	คืนค่าพารามิเตอร์ของเมธตอดนั้น ๆ
getNumberOfParameter():int	คืนค่าจำนวนพารามิเตอร์ของเมธตอดนั้น ๆ

- Message มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
drawMessage()	วาดเส้นที่แสดงถึงแมจเสลที่ส่งกันระหว่างคลาส

- Call มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
drawCall()	วาดเส้นที่แสดงถึงการเรียกคลาสอื่นมาทำงาน

- SequenceNode มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
drawSequenceNode ()	วาดโหนดคลาสของ Sequence Diagram

- StructureNode มีรายละเอียดดังต่อไปนี้

เมธตอด	หน้าที่
setParent(StructureNode s)	กำหนดค่าโหนดแม่ของโหนดนั้น ๆ
setLevel(int n)	กำหนดค่าระดับชั้นของโหนดนั้น ๆ
setOrder(int n)	กำหนดค่าลำดับของโหนดในระดับชั้นนั้น ๆ
setNumberOfChild(int n)	กำหนดค่าจำนวนโหนดลูกของโหนดนั้น ๆ
getParent():StructureNode	คืนค่าโหนดแม่ของโหนดนั้น ๆ
getLevel():int	คืนค่าระดับชั้นของโหนดนั้น ๆ
getOrder():int	คืนค่าลำดับของโหนดในระดับชั้นนั้น ๆ
getNumberOfChild():int	คืนค่าจำนวนโหนดลูกของโหนดนั้น ๆ
drawStructureNode()	วาดโหนดคลาสของ Structure Chart

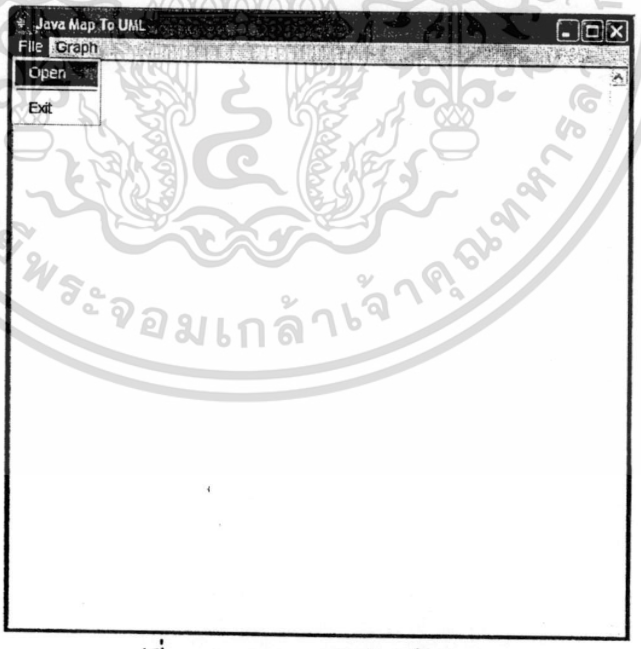
ParsingToClassDiagram , ParsingToSequenceDiagram และ ParsingToStructureChart เป็นคลาสที่ทำหน้าที่ในการวิเคราะห์เอกสารทางเอ็กซ์เอ็มไอเพื่อที่จะนำข้อมูลที่วิเคราะห์ได้ไปสร้างแผนภาพ ซึ่งเป็นคลาสที่ extends มาจากคลาส DefaultHandler ดังนั้นจึงต้อง implement บางเมธอดเองเช่น startDocument() , endDocument() , startElement() , endElement() เป็นต้น

บทที่ 4

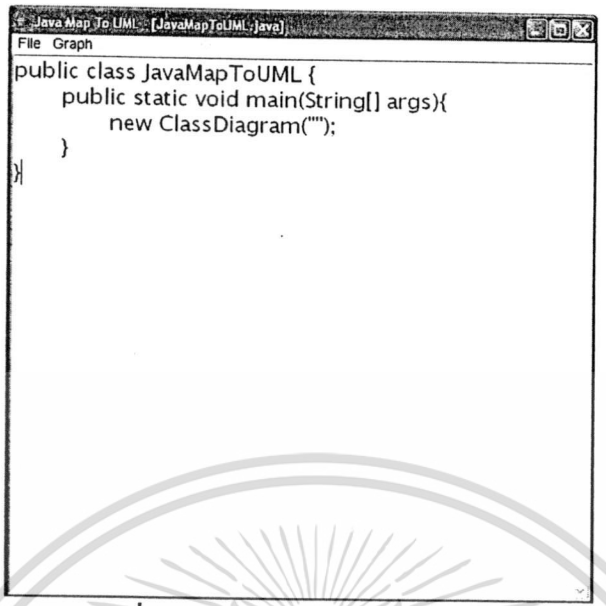
ผลการทดลอง

4.1 อินเตอร์เฟสของเครื่องมือ JMD

- เครื่องมือที่สร้างในโครงการพัฒนาระบบได้ถูกตั้งชื่อว่า JMD ซึ่งย่อมาจากคำว่า Java Map to Diagram เมื่อเริ่มใช้งานเครื่องมือ(โดยใช้การรัน JavaMaptoDiagram.java)
- เมื่อเริ่มใช้งานผู้ใช้ทำการเลือกเปิดไฟล์ของโค้ดภาษาจาวาได้ที่เมนู "File" แล้วเลือก "Open" โดยมีลักษณะของอินเตอร์เฟสดังรูปที่ 4.1 ต่อจากนั้นทำการเลือกไฟล์จาวาที่ต้องการแปลงเป็นแผนภาพ และเครื่องมือจะทำการแสดงโค้ดจาวาที่ถูกเลือกทางอินเตอร์เฟส ดังรูปที่ 4.2



รูปที่ 4.1 แสดงการเปิดไฟล์โค้ดจาวา



รูปที่ 4.2 แสดงโค้ดภาษาจาวาที่ถูกเลือก

- ต่อจากนั้นจึงทำการเลือกแผนภาพที่ต้องการ ดังรูปที่ 4.3 ที่แสดงเมนูการเลือกแผนภาพที่สามารถสร้างได้ของเครื่องมือนี้

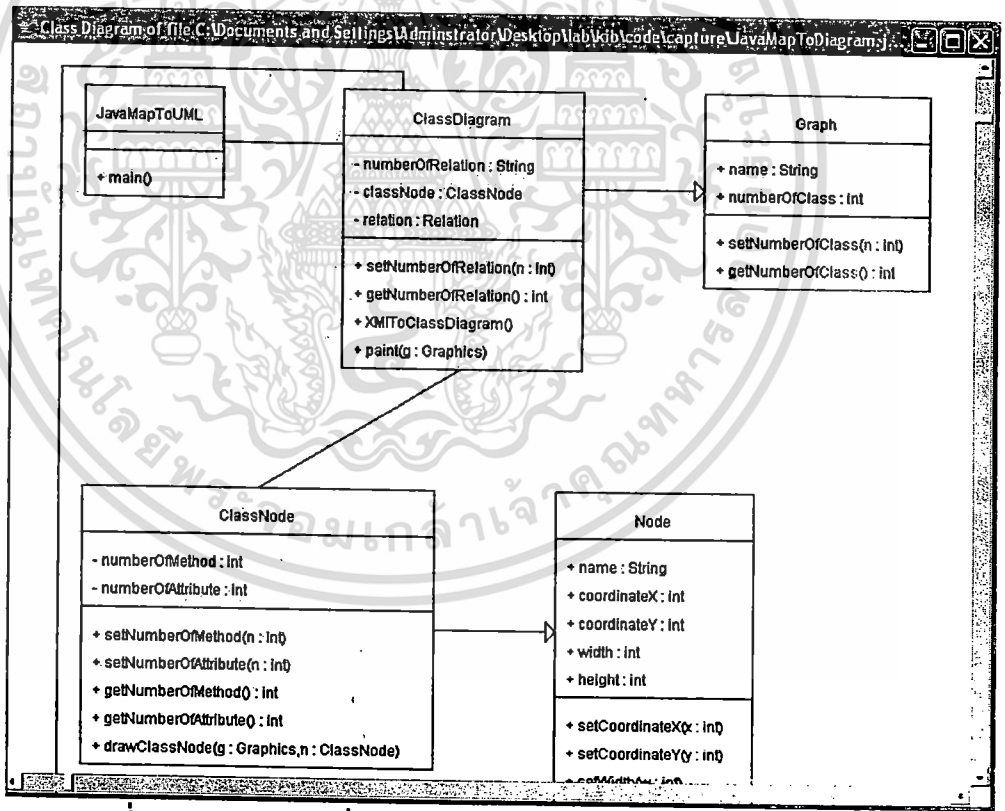
รูปที่ 4.3 แสดงเมนูของแผนภาพต่างๆ

4.2 ลักษณะของเอกสารทางเอ็กซ์เอ็มไอ

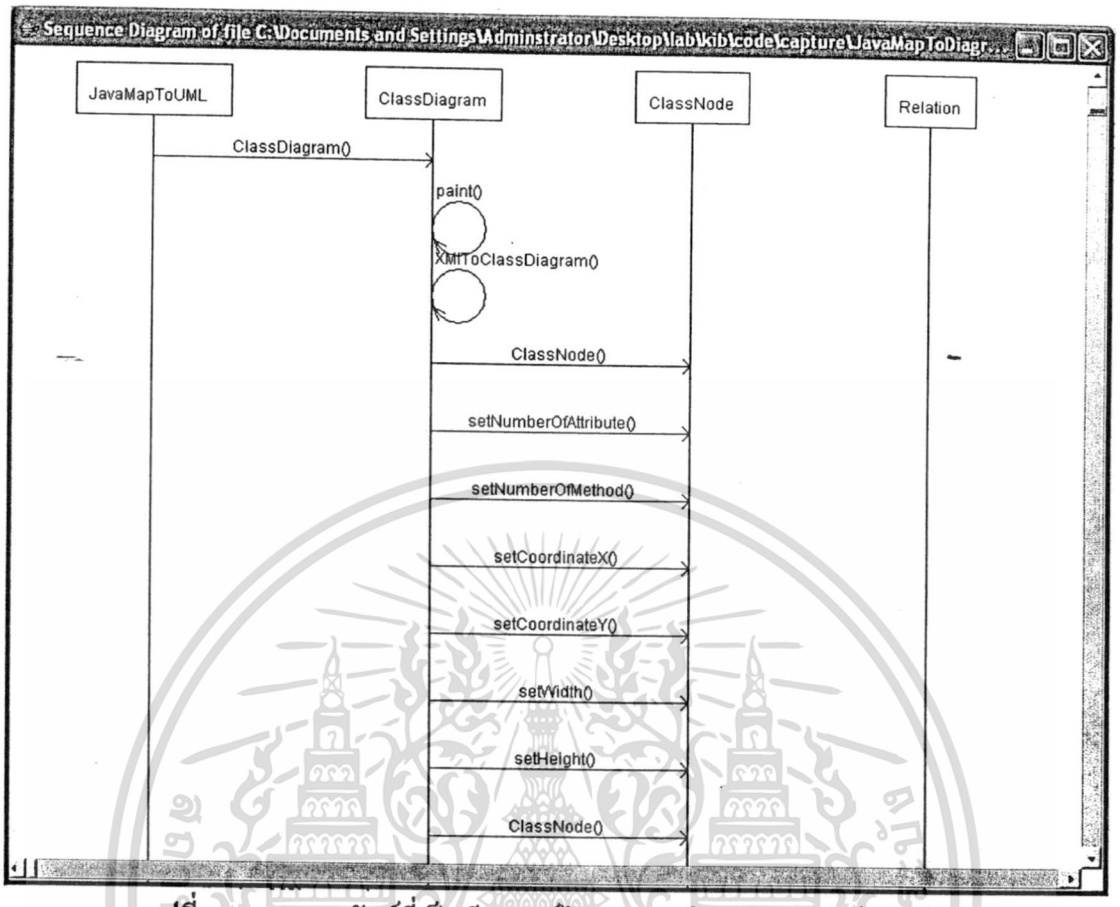
เมื่อผู้ใช้ทำการเลือกไฟล์จาวาที่ต้องการแปลงเป็นแผนภาพ และเครื่องมือได้ทำการแสดงโค้ดที่ถูกเลือก สิ่งที่เป็นผลลัพธ์ที่ได้อย่างอัตโนมัติคือเอกสารทางเอ็กซ์เอ็มไอ สำหรับเอกสารทางเอ็กซ์เอ็มไอที่ได้จากการแปลงไฟล์โค้ดจาวาที่มีชื่อว่า JavaMapToUML สามารถดูรายละเอียดของเอกสารเอ็กซ์เอ็มไอที่ได้จากแปลงไฟล์จาวาดังกล่าวและรายละเอียดของโค้ดจาวาต่างๆที่เกี่ยวข้องได้ที่ ภาคผนวก ค

4.3 ผลลัพธ์

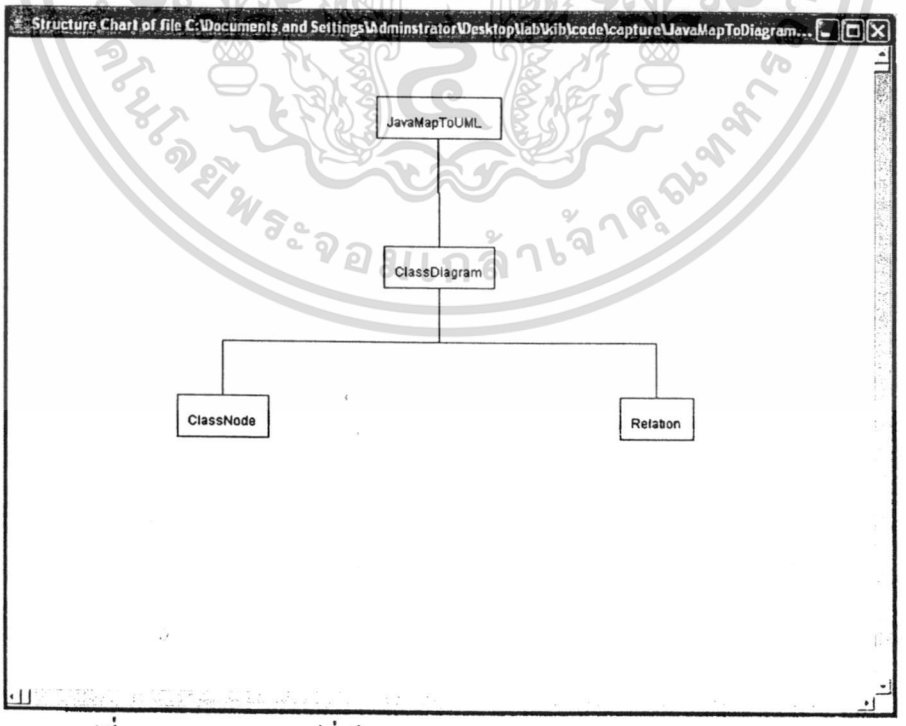
ผลลัพธ์ที่ได้จากเครื่องมือนี้ สามารถสร้างแผนภาพจากโค้ดจาวาได้ทั้งหมด 3 แผนภาพ ดังรูปที 4.4 แสดงผลลัพธ์ที่เป็นแผนภาพคลาสไดอะแกรม, รูปที่ 4.5 แสดงผลลัพธ์ที่เป็นแผนภาพซีเควนซ์ไดอะแกรม, และ รูปที่ 4.6 แสดงผลลัพธ์ที่เป็น Structure Chart



รูปที่ 4.4 แสดงผลลัพธ์ที่เป็นคลาสไดอะแกรมในการแปลงโค้ดจาวา



รูปที่ 4.5 แสดงผลลัพธ์ที่เป็นซีเควนซ์ไดอะแกรมในการแปลงโค้ดจาวา



รูปที่ 4.6 แสดงผลลัพธ์ที่เป็น Structure Chart ในการแปลงโค้ดจาวา

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 สรุปการพัฒนาระบบ

โครงการพัฒนาระบบนี้ เป็นการพัฒนาเครื่องมือที่แปลงโค้ดภาษาจาวาไปเป็นแผนภาพ ซึ่งผลที่ได้รับจากการพัฒนาระบบมีดังนี้

- เครื่องมือสำหรับให้ผู้พัฒนาโปรแกรมหรือผู้ทดสอบโปรแกรมสามารถใช้เป็นเครื่องมือในการแปลงโปรแกรมภาษาจาวาเป็นแผนภาพ โดยมีการทำงานตามลำดับดังต่อไปนี้
  - แปลงโปรแกรมภาษาจาวาที่ผู้ใช้ต้องการไปเป็นเอกสารทางเอ็กซ์เอ็มไอที่สอดคล้องกับโปรแกรมภาษาจาวา
  - แปลงเอกสารทางเอ็กซ์เอ็มไอ ที่ได้จากขั้นตอนก่อนหน้าไปเป็นแผนภาพตามประเภทที่ผู้ใช้เลือก เช่น Class Diagram , Sequence Diagram และ Structure Chart
- การใช้เครื่องมือแปลงโปรแกรมภาษาจาวาเป็นแผนภาพจะช่วยให้ผู้พัฒนาและผู้ทดสอบระบบสามารถเข้าใจลักษณะการทำงานของโปรแกรมได้ดีมากยิ่งขึ้น และช่วยให้ผู้พัฒนาระบบชุดใหม่ ๆ เข้าใจถึงลักษณะการทำงานของโปรแกรมที่ผู้พัฒนาชุดเก่าได้พัฒนาไว้
- ได้รับความรู้จากการศึกษาโครงสร้างไวยากรณ์ของภาษาจาวา ยูเอ็มแอล เอ็กซ์เอ็มไอ รวมถึงได้รับความรู้จากการศึกษาเครื่องมืออื่น ๆ ที่ใช้ในการพัฒนาระบบนี้ด้วย เช่น JavaCC เป็นต้น

5.2 ข้อเสนอแนะ

การพัฒนาเครื่องมือในการแปลงภาษาจาวาไปเป็นแผนภาพที่มีการนำมาตรฐานกลางมาเป็นตัวกลางในการแปรรูปโค้ดไปสู่แผนภาพ มีข้อเสนอแนะดังต่อไปนี้

- ในการศึกษาเครื่องมือแปลงโค้ดจาวาไปเป็นแผนภาพ สามารถทำการแปลงไปสู่แผนภาพอื่นๆได้ นอกจาก 3 แผนภาพที่เครื่องมือนี้ได้เลือกใช้
- นอกเหนือจากมาตรฐานกลางที่มีชื่อว่าเอ็กซ์เอ็มไอแล้ว เครื่องมือที่แปลงโค้ดจาวาเป็นแผนภาพอื่นๆ มีการใช้มาตรฐานกลางตัวกลางในการแลกเปลี่ยนแบบอื่นๆมาใช้ได้
- การศึกษาเอกสารทางเอ็กซ์เอ็มไอ โดยการแปลงกับเครื่องมือที่มีความสามารถทางด้านการแปลงจาวาโค้ดไปสู่แผนภาพโดยผ่านมาตรฐานกลางเอ็กซ์เอ็มไอ ไม่สามารถทำได้ในกรณีที่เวอร์ชั่นของเอ็กซ์เอ็มไอไม่ตรงกัน
- JMD เป็นเครื่องมือที่มีความสามารถทางด้าน reverse engineering ซึ่งในการพัฒนาต่อไปควรมีความสามารถในการทำ forward engineering และ ลักษณะของการทำ round trip engineering
- นอกจากนี้ พัฒนา JMD ให้มีความสามารถในการเป็น editor ด้วย

บรรณานุกรม

พอเจตน์ ตันธนกุล. 2546. **ศึกษามาตรฐานของ XMI และ ebXML.** คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

พนิดา พานิชกุล. และ กิตติ ภักดีวัฒนะกุล. 2548. **คัมภีร์การพัฒนาระบบเชิงวัตถุด้วย UML และ Java.** กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์.

กิติ ภักดีวัฒนะกุล. และ กิติพงษ์ กลมกล่อม. 2544. **UML วิเคราะห์และออกแบบระบบเชิงวัตถุ.** กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์.

กิติ ภักดีวัฒนะกุล. และ กิติพงษ์ กลมกล่อม. 2548. **คัมภีร์การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML.** กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์.

สุนทริน วงศ์ศิริกุล. 2537. **พัฒนาโมเดลยุคใหม่ UML(Unified Modeling Language) มาตราฐานการสร้างโมเดลระบบงาน.** กรุงเทพ. ซัคเซสมีเดีย .

ธนา สุขวารี. 2002. **ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาจาวา.** กรุงเทพฯ : แมคกรอฮิลล์.

ชาลี วรกุลพิพัฒน์. และ เทพฤทธิ์ บัณฑิตวัฒนาวงศ์. 2544. **UML ภาษามาตรฐานเพื่อผู้พัฒนาซอฟแวร์.** กรุงเทพ. ซีเอ็ดยูเคชั่น.

ดร.บรรจง หะรังสี. และ นางญาณวรรณ สินธุภิญโญ. 2000. **UML (Unified Modeling Language).** [Online]. Available : http://www.thaiall.com/uml.

สราวุธ อ้อยศรีสกุล. 2544. **เริ่มคิด-เริ่มสร้าง-เริ่มใช้ XML.** กรุงเทพฯ. วิตตี้ กรุ๊ป.

สุวัฒนา สุขสมจินตน์. 2545. **คัมภีร์การใช้ XML ฉบับสมบูรณ์.** กรุงเทพ. ซีเอ็ดยูเคชั่น.

จุฬาภา ชูอำไพ. 2547. **เครื่องมือทดสอบซอฟต์แวร์แบบกึ่งอัตโนมัติด้วยยูนิเวร์แซลเทสต์ไดรเวอร์.** คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

Brodsky, S. 1999. **XMI Opens Application Interchange.** [Online]. Available : http://www .cin. ufpe.br/ ~aad/garbage/xmiwhite0399.pdf.

Grose, T. et al. 2002. **Mastering XMI- Java Programming with XMI, XML and UML.** John Wiley & Sons Inc.

Homer, A. 1999. **XML in IE5 Programmer's Reference.** Indianapolis: Wrox Press.

IBM, 1999. **IBM alphaWorks laboratory releases XMI Framework.** [Online]. Available: http://xml.coverpages.org/ni2001-03-20-c.html

Jiang, Juan-Juan. and Tarja, Systä. 2002. **UML model exchange using XMI.** [Online]. Available : http://ww.cs.tut.fi/~xmlohj/linkit/XMIReport.pdf.

Knoernschild, Kirk. 2000. **Modeling Java Application using UML.** [Online]. Available : http://www.kirkk.com/old/ReferenceCard.html.

Lorenz, Mark and Kid, Jeff. 1994. **Object-Oriented Software Metrics,** Prentice Hall.

Metsker, Steven John. 2001. **Building Parsers with Java.** Addison-Wesley.

Netisopakul, Ponrudee. 2003. **Visualizing Dynamic Objects in Object-Oriented Program,** *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics.* July 2003. USA. Vol. XIII. Pp 321-325.

OMG. 2002. **XMI specification** 1.2. [Online] Available: http:// **www.omg.org/** technology/ documents/formal/xmi.htm.

Pechanec, Jan. 2000. **Reverse Engineering of Program in Java Language.** [Online] Available : http://www.sweb.cz/pechanec/java2uml.html.

Russell, C.R. and Dewar, R.G. 2001. **XML Encoded Reverse Engineering of Java to UML.** [Online] Available: http://www.macs.hw.ac.uk:8080/techreps/docs/files/HW-MACS-TR-0007.pdf.

Stevens, Perdita. 2000. **XMI and MOF: a mini-tutorial.** [Online]. Available: http://www.dcs.ed.ac.uk/home/pxs.

XML Journal. 2002. **XML feature (XMI: the OMG's XML Metadata Interchange Standard).** [Online] Available: http://www.sys-con.com/xml/articleprint.cfm?id=45.

ภาคผนวก ก

รูปแบบของ JavaCC

รูปแบบโครงสร้างของ JavaCC มีลักษณะโครงสร้างดังรูปที่ ก.1 ที่แสดงรูปแบบโครงสร้างของ JavaCC ที่สามารถแบ่งโครงสร้างได้เป็น 4 ส่วนหลักๆ ดังนี้

```
options { ... }

PARSER_BEGIN(X)
public class X {
    public static void main(String args []) {
        ...
        X parser = new X(System.in);
        try { parser. firstProduction (); } ...
    }
}
PARSER_END(X)
// Token definitions (optional)
...
// Production rules
void firstProduction () :
{ ... }   // declarations
{ ... }   // rules and actions
// more productions
```

รูปที่ ก.1 แสดงรูปแบบโครงสร้าง JavaCC

1) options {...}

เป็นส่วนของการตั้งค่าต่างๆที่จะให้มีรูปแบบและลักษณะการทำงานต่างๆ ซึ่งตัวโปรแกรมได้ค่า default ต่างๆไว้แล้วแต่ถ้าต้องการใช้หรือเปลี่ยนลักษณะการทำงานนั้นสามารถที่จะเปลี่ยนแปลงค่าได้ เช่น LOOKAHEAD = 1; เป็นค่า default ของโปรแกรมนี้ ดังนั้นถ้าต้องการให้โปรแกรมทำการ look ahead เป็นจำนวน 3 token ก่อนที่จะตัดสินใจถึงแนวทางการวิเคราะห์ในโปรแกรม สามารถกำหนดค่าเป็น LOOKAHEAD = 3;

2) PARSER_BEGIN(X)

.....

class X...{

.....

}

PARSER_END(X)

เป็นส่วนที่แสดงถึงรายละเอียดของคลาสที่เป็นตัวผู้วิเคราะห์(parser) โดยที่โปรแกรมจะจำแนกคลาสที่เป็นตัวผู้วิเคราะห์ที่มีชื่อตามที่กล่าวไว้ในโครงสร้างตรงส่วนนี้ (คือ ในส่วนของที่เป็น X นั่นเอง)

3) Lexical Specification

เป็นส่วนของบรรทัดที่มีข้อความว่า //Token definition(optional) ในรูปที่ 1 โดยที่ส่วนนี้เป็นการระบุข้อกำหนดต่างๆที่เป็นลักษณะของ token ของอินพุตที่ต้องการ ในส่วนนี้สามารถกำหนดรูปแบบการทำ lexical analysis ได้ทั้งหมด 4 รูปแบบ ดังนี้

3.1) SKIP

เป็นการกำหนดว่าข้อมูลแบบใดที่ไม่ต้องสนใจในการวิเคราะห์ เช่น SKIP : {" "} เป็นการกำหนดว่าไม่สนใจข้อมูลที่เป็นช่องว่าง(" ")

3.2) TOKEN

เป็นการกำหนดรูปแบบของ token ซึ่งถ้าตรงกับลักษณะของข้อมูลที่เป็นอินพุตจะทำการสร้างเป็นอ๊อบเจตและเป็นหนึ่งในรูปแบบที่ใช้กำหนดข้อกำหนดของรูปแบบการนำข้อมูลเข้ามาวิเคราะห์ เช่น TOKEN : { <CLASS : "class">}

3.3) SPECIAL_TOKEN

คล้ายกับ TOKEN แต่ว่าการกำหนดให้ลักษณะของข้อมูลใดเป็นแบบนี้แล้วข้อมูลเหล่านั้นจะไม่ถูกส่งเข้ามาในส่วนของการวิเคราะห์

3.4) MORE

เมื่อกำหนดลักษณะให้เป็นลักษณะนี้ จะไม่สนใจข้อมูลเหล่านั้นจนกว่าจะเจอข้อมูลที่เป็นลักษณะของข้อมูลที่เป็น TOKEN หรือ SPECIAL_TOKEN

4) Grammar Specification

เป็นส่วนของ production rule ที่กำหนดรูปแบบ(pattern) ต่างๆในการวิเคราะห์ข้อมูลต่างๆ ตัวอย่างเช่น

```
void CompilationUnit() :
{}
{
      [ PackageDeclaration() ]
      ( ImportDeclaration() )*
      ( TypeDeclaration() )*
      <EOF>
```

}

จากตัวอย่างนี้เป็นรูปแบบของ production ที่เรียกว่า bnf production โดยจะมีรูปแบบ ดังนี้ java_return_type java_identifier (java_parameter_list) : java_block { expansion choice choice}

ลักษณะของโครงสร้างที่มีการใช้ใน JavaCC ที่เป็นส่วน Lexical Specification และ Grammar Specification มีรายละเอียดดังนี้

- E1|E2|E3|... แสดงถึงทางเลือก(คล้ายกับการใช้การเปรียบเทียบ OR)
- (...)+ แสดงถึงการมีโอกาสการเกิดสิ่งที่อยู่ภายในรูปแบบนี้ตั้งแต่ หนึ่งถึงหลายๆครั้ง
- (...)* แสดงถึงการมีโอกาสการเกิดสิ่งที่อยู่ภายในรูปแบบนี้ตั้งแต่ ศูนย์ถึงหลายๆครั้ง
- (...)? แสดงถึงลักษณะของ "ถ้ามี" กับสิ่งที่ระบุภายในโครงสร้างนี้
- [...] แสดงรูปแบบที่ได้ระบุไว้ภายในโครงสร้างนี้(ซึ่งเป็นรูปแบบของตัวอักษรหรือช่วงของตัวอักษร)
- ~[...] แสดงถึงรูปแบบใดๆก็ตามที่ไม่ตรงกับสิ่งที่อยู่ภายในโครงสร้างนี้

เมื่อมีผลลัพธ์ของ JavaCC ที่เป็นไฟล์ .jj แล้วต่อจากนั้นต้องมาดำเนินการจำแนกไฟล์ต่างๆ ออกจากไฟล์ .jj นั้นๆ โดยการพิมพ์คำสั่ง "javacc ชื่อ.jj" ซึ่งผลลัพธ์ที่ได้นั้นเป็นไฟล์จาวาทั้งหมด 7 ไฟล์(ไฟล์ X.java, XTokenManager.java, และ XConstants.java โดยที่เครื่องหมาย X เป็นสัญลักษณ์แทนชื่อที่อยู่ในโครงสร้างส่วนที่ 2 ส่วนของPARSER_BEGIN(JavaParser) จนถึง PARSER_END(JavaParser)) โดยรูปที่ ก.2 เป็นการแสดงตัวอย่างของคลาสต่างที่ถูกสร้างขึ้นจากไฟล์ .jj ไฟล์หนึ่ง(ซึ่งในโครงสร้างส่วนที่ 2 เป็น PARSER_BEGIN(ShowingGeneration) .... PARSER_END(ShowingGeneration) เมื่อทำการรันคำสั่ง javacc ตามด้วยชื่อไฟล์(.jj) จะมีคลาสที่เกิดขึ้นอย่างอัตโนมัติทั้งหมด 7 คลาส ซึ่งแต่ละคลาสมีหน้าที่แตกต่างกันไป มีรายละเอียดดังนี้

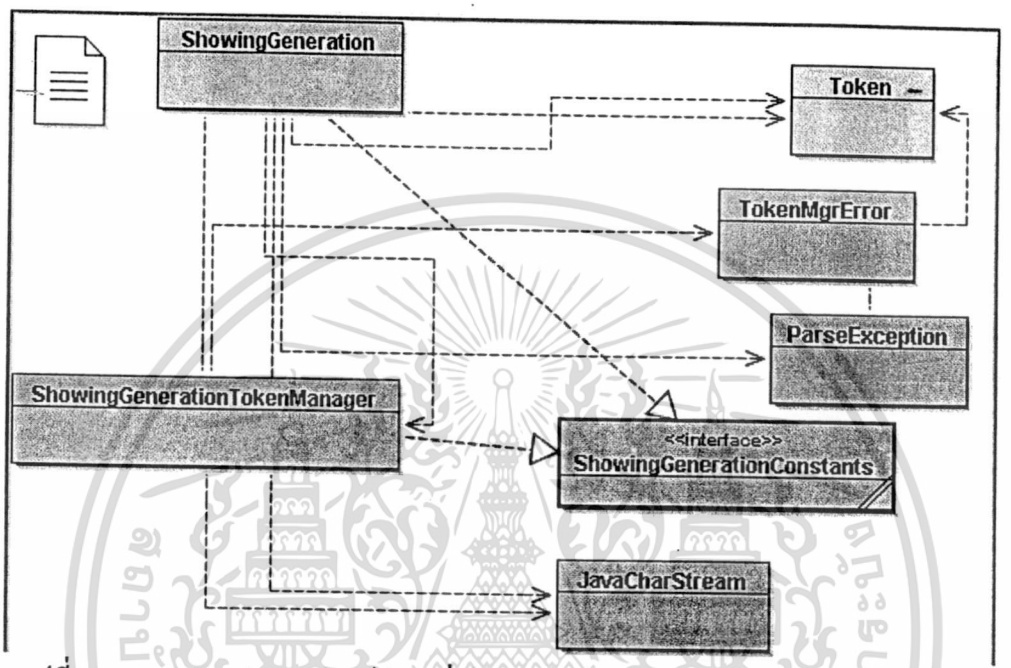
- TokenMgrError.java ส่งรายละเอียดของข้อความที่เกี่ยวข้องกับข้อผิดพลาดเมื่อมีความผิดพลาดเกี่ยวกับการ Lexical Analysis

- Token.java เป็นคลาสที่เป็นตัวแทน token ในแต่ละอ๊อบเจท Token (สิ่งที่ระบุว่าเป็น TOKEN ใน Lexical Specification จะถูกสร้างเป็นอ๊อบเจท Token)

-ParseException.java เป็นคลาสที่ตรวจจับข้อผิดพลาดที่เกี่ยวกับรูปแบบ Grammar Specification

- JavaCharStream แสดงสตรีมของ input character

- JavaParser.java เป็นคลาสของตัวผู้วิเคราะห์
- JavaParserTokenManager เป็น Lexical Analyzer
- JavaParserConstants.java เป็นอินเตอร์เฟสที่เชื่อมโยง token กับสัญลักษณ์ของ token



รูปที่ ก.2 แสดงตัวอย่างของไฟล์ต่างๆที่เป็นผลลัพธ์ของ JavaCC(ในรูปแบบของคลาส)

ภาคผนวก ข

รายละเอียดของ J2X.jj

```
/*JMU -- Java Maps to UML
This parser is created to JMU Tool. It is parser to analyse java code and store token of the code. The stored token will translate to XMI format and save it to ".xmi"(XMI Document). We created the parser with modification "java1.4.jj". For original parser, you can download and it is free.
*/
options {
  JAVA_UNICODE_ESCAPE = true;
  STATIC = false;//set option of method that is not static(default of JavaCC -- set option of all method to static)
}
PARSER_BEGIN(JavaParser)
import java.util.*;
class JavaParser
{

          Header header = new Header();
          ClassDeclaration classTok = new ClassDeclaration();
          Attribute att = new Attribute();
          Method met = new Method();
          CreationXMIDoc doc = new CreationXMIDoc();
          String codename;
          //start parse and take filename from J2X.java
          public void parseCode(String filename) {
                    //cut java
                    // to take file name of source code that cut ".java"
                    StringTokenizer cutName = new StringTokenizer(filename, ".java");
                    codename = cutName.nextToken();//this function return String
                    //sent code name to Header, method, Attribute, CreationXMIDoc
                    header.getName(codename);
                    doc.getCodeName(codename);

          //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                    //Parsing
                    //set time
                    long initTime = 0;
                    long parseTime = 0;
                    long startTime = 0;
                    long stopTime = 0;
               try//when it can completly parse, display time in parsing process
               {
                startTime = System.currentTimeMillis();
                System.out.println(" JMU -- Java Maps to UML");
                System.out.println("This is parsing java code to XMI Document: ");
                System.out.println(codename + " is parsed to ......");
                //start parse
               CompilationUnit();
```

```
44                      stopTime = System.currentTimeMillis();
45                      parseTime = stopTime - startTime;
46                      System.out.println("  Java program parsed " + codename + " successfully in " + (
    initTime + parseTime) + " ms.");
47                      System.out.println("     parser initialization time was " + initTime + " ms.");
48                      System.out.println("     parser parse time was " + parseTime + " ms.");
49                    } catch (ParseException e)//display error when it cannot parse
50                    {
51                      System.out.println(e.getMessage());
52                      System.out.println("Java Parser :  Encountered errors during parse.");
53                    }
54    }//end of parseCode
55    public String getFileXMI() {
56                String XMIFile = codename + ".xml";
57                return XMIFile;
58    }
59  }//end of class JavaParser
60  PARSER_END(JavaParser)
61  //*************************************************************************
62  //                                  Lexical Analysis
63  //*************************************************************************
64  /* WHITE SPACE */
65  SKIP :
66  {
67    " "
68  | "\t"
69  | "\n"
70  | "\r"
71  | "\f"
72  }
73  /* COMMENTS */
74  MORE :
75  {
76    "//" : IN_SINGLE_LINE_COMMENT
77  |
78    <"/**" ~["/"]> { input_stream.backup(1); } : IN_FORMAL_COMMENT
79  |
80    "/*" : IN_MULTI_LINE_COMMENT
81  }
82
83  <IN_SINGLE_LINE_COMMENT>
84  SPECIAL_TOKEN :
85  {
86    <SINGLE_LINE_COMMENT: "\n" | "\r" | "\r\n" > : DEFAULT
87  }
88
89  <IN_FORMAL_COMMENT>
90  SPECIAL_TOKEN :
91  {
92    <FORMAL_COMMENT: "*/" > : DEFAULT
93  }
94
95  <IN_MULTI_LINE_COMMENT>
96  SPECIAL_TOKEN :
97  {
98    <MULTI_LINE_COMMENT: "*/" > : DEFAULT
99  }
```

```
100
101   <IN_SINGLE_LINE_COMMENT,IN_FORMAL_COMMENT,IN_MULTI_LINE_COMMENT>
102   MORE :
103   {
104     < ~[] >
105   }
106
107   /* RESERVED WORDS AND LITERALS */
108
109   TOKEN :
110   {
111     < ABSTRACT: "abstract" >
112   | < BOOLEAN: "boolean" >
113   | < BREAK: "break" >
114   | < BYTE: "byte" >
115   | < CASE: "case" >
116   | < CATCH: "catch" >
117   | < CHAR: "char" >
118   | < CLASS: "class" >
119   | < CONST: "const" >
120   | < CONTINUE: "continue" >
121   | < _DEFAULT: "default" >
122   | < DO: "do" >
123   | < DOUBLE: "double" >
124   | < ELSE: "else" >
125   | < EXTENDS: "extends" >
126   | < FALSE: "false" >
127   | < FINAL: "final" >
128   | < FINALLY: "finally" >
129   | < FLOAT: "float" >
130   | < FOR: "for" >
131   | < GOTO: "goto" >
132   | < IF: "if" >
133   | < IMPLEMENTS: "implements" >
134   | < IMPORT: "import" >
135   | < INSTANCEOF: "instanceof" >
136   | < INT: "int" >
137   | < INTERFACE: "interface" >
138   | < LONG: "long" >
139   | < NATIVE: "native" >
140   | < NEW: "new" >
141   | < NULL: "null" >
142   | < PACKAGE: "package">
143   | < PRIVATE: "private" >
144   | < PROTECTED: "protected" >
145   | < PUBLIC: "public" >
146   | < RETURN: "return" >
147   | < SHORT: "short" >
148   | < STATIC: "static" >
149   | < SUPER: "super" >
150   | < SWITCH: "switch" >
151   | < SYNCHRONIZED: "synchronized" >
152   | < THIS: "this" >
153   | < THROW: "throw" >
154   | < THROWS: "throws" >
155   | < TRANSIENT: "transient" >
156   | < TRUE: "true" >
```

```
157     | < TRY: "try" >
158     | < VOID: "void" >
159     | < VOLATILE: "volatile" >
160     | < WHILE: "while" >
161     | < STRICTFP: "strictfp" >
162     // added by Andrea Gini
163     | < ASSERT: "assert" >
164     }
165
166     /* LITERALS */
167
168     TOKEN :
169     {
170       < INTEGER_LITERAL:
171           <DECIMAL_LITERAL> (["l","L"])?
172         | <HEX_LITERAL> (["l","L"])?
173         | <OCTAL_LITERAL> (["l","L"])?
174       >
175     |
176       < #DECIMAL_LITERAL: ["1"-"9"] (["0"-"9"])* >
177     |
178       < #HEX_LITERAL: "0" ["x","X"] (["0"-"9","a"-"f","A"-"F"])+ >
179     |
180       < #OCTAL_LITERAL: "0" (["0"-"7"])* >
181     |
182       < FLOATING_POINT_LITERAL:
183           (["0"-"9"])+ "." (["0"-"9"])* (<EXPONENT>)? (["f","F","d","D"])?
184         | "." (["0"-"9"])+ (<EXPONENT>)? (["f","F","d","D"])?
185         | (["0"-"9"])+ <EXPONENT> (["f","F","d","D"])?
186         | (["0"-"9"])+ (<EXPONENT>)? ["f","F","d","D"]
187       >
188     |
189       < #EXPONENT: ["e","E"] (["+","-"])? (["0"-"9"])+ >
190     |
191       < CHARACTER_LITERAL:
192           "'"
193         (   (~["'","\\","\n","\r"])
194           | ("\\"
195               ( ["n","t","b","r","f","\\","'","\""]
196               | ["0"-"7"] ( ["0"-"7"] )?
197               | ["0"-"3"] ["0"-"7"] ["0"-"7"]
198               )
199             )
200         )
201           "'"
202       >
203     |
204       < STRING_LITERAL:
205           "\""
206         (   (~["\"","\\","\n","\r"])
207           | ("\\"
208               ( ["n","t","b","r","f","\\","'","\""]
209               | ["0"-"7"] ( ["0"-"7"] )?
210               | ["0"-"3"] ["0"-"7"] ["0"-"7"]
211               )
212             )
213         )*
```

```
214         "\""
215       >
216     }
217
218     //added by Uaharong
219     TOKEN :
220     {
221     <STRING : "String">
222     }
223
224     /* IDENTIFIERS */
225
226     TOKEN :
227     {
228       < IDENTIFIER: <LETTER> (<LETTER>|<DIGIT>)* >
229     |
230       < #LETTER:
231         [
232           "\u0024",
233           "\u0041"-"\u005a",
234           "\u005f",
235           "\u0061"-"\u007a",
236           "\u00c0"-"\u00d6",
237           "\u00d8"-"\u00f6",
238           "\u00f8"-"\u00ff",
239           "\u0100"-"\u1fff",
240           "\u3040"-"\u318f",
241           "\u3300"-"\u337f",
242           "\u3400"-"\u3d2d",
243           "\u4e00"-"\u9fff",
244           "\uf900"-"\ufaff"
245         ]
246       >
247     |
248       < #DIGIT:
249         [
250           "\u0030"-"\u0039",
251           "\u0660"-"\u0669",
252           "\u06f0"-"\u06f9",
253           "\u0966"-"\u096f",
254           "\u09e6"-"\u09ef",
255           "\u0a66"-"\u0a6f",
256           "\u0ae6"-"\u0aef",
257           "\u0b66"-"\u0b6f",
258           "\u0be7"-"\u0bef",
259           "\u0c66"-"\u0c6f",
260           "\u0ce6"-"\u0cef",
261           "\u0d66"-"\u0d6f",
262           "\u0e50"-"\u0e59",
263           "\u0ed0"-"\u0ed9",
264           "\u1040"-"\u1049"
265         ]
266       >
267     }
268
```

```
269     /* SEPARATORS */
270
271     TOKEN :
272     {
273       < LPAREN: "(" >
274     | < RPAREN: ")" >
275     | < LBRACE: "{" >
276     | < RBRACE: "}" >
277     | < LBRACKET: "[" >
278     | < RBRACKET: "]" >
279     | < SEMICOLON: ";" >
280     | < COMMA: "," >
281     | < DOT: "." >
282     }
283
284     /* OPERATORS */
285
286     TOKEN :
287     {
288       < ASSIGN: "=" >
289     | < GT: ">" >
290     | < LT: "<" >
291     | < BANG: "!" >
292     | < TILDE: "~" >
293     | < HOOK: "?" >
294     | < COLON: ":" >
295     | < EQ: "==" >
296     | < LE: "<=" >
297     | < GE: ">=" >
298     | < NE: "!=" >
299     | < SC_OR: "||" >
300     | < SC_AND: "&&" >
301     | < INCR: "++" >
302     | < DECR: "--" >
303     | < PLUS: "+" >
304     | < MINUS: "-" >
305     | < STAR: "*" >
306     | < SLASH: "/" >
307     | < BIT_AND: "&" >
308     | < BIT_OR: "|" >
309     | < XOR: "^" >
310     | < REM: "%" >
311     | < LSHIFT: "<<" >
312     | < RSIGNEDSHIFT: ">>" >
313     | < RUNSIGNEDSHIFT: ">>>" >
314     | < PLUSASSIGN: "+=" >
315     | < MINUSASSIGN: "-=" >
316     | < STARASSIGN: "*=" >
317     | < SLASHASSIGN:  =  >
318     | < ANDASSIGN:  =  >
319     | < ORASSIGN:  =  >
320     | < XORASSIGN:  =  >
321     | < REMASSIGN:  =  >
322     | < LSHIFTASSIGN:  <<=  >
323     | < RSIGNEDSHIFTASSIGN: ">>=" >
324     | < RUNSIGNEDSHIFTASSIGN: ">>>=" >
325     }
326
327
```

```
/*****************************************
 * THE JAVA LANGUAGE GRAMMAR STARTS HERE *
 *****************************************/

/*
 * Program structuring syntax follows.
 */

void CompilationUnit() :
{}
{
  [ PackageDeclaration() ]
  ( ImportDeclaration() )*
  ( TypeDeclaration() )*
  <EOF>
  {doc.endofParseFile();}
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          Header
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void PackageDeclaration() :
{Vector token = new Vector();}
{
  //"package"Name() ";"
  "package"getHeader(token)";"{header.getImportToken(token);}
}

void ImportDeclaration() :
{Vector token = new Vector();}
{
  //"import" Name() [ "." "*" ] ";"
  "import" getHeader(token)";"{header.getImportToken(token);}
}
//it store token continuously util it find ";"
JAVACODE
void getHeader(Vector t) {
          Token tok;
          tok = getToken(1);
          while(tok.kind != SEMICOLON) {
                    t.addElement(tok);
                    tok = getNextToken();
                    //if this line missed, token will repeat
                    //For example -- java.swing.*  will result -- [javax,javax,.,swing,.,*]
                    tok = getToken(1);
          }
}
//this method decide that java code composed of class, inter, or ";"
void TypeDeclaration() :
{}
{

  LOOKAHEAD( ( "abstract" | "final" | "public" | "strictfp" )* "class" )
  ClassDeclaration()
  {header.endofHeader();} //to send signal tell Header class(end of work)
|
  InterfaceDeclaration() {header.endofHeader();}
//| ";" -- in original jj file has this line but in JMU does not use it. Also, if this line exists, ImportDeclaration
and PackageDeclaration do not put ";" in them.
}

```

```
386  /*
387   * Declaration syntax follows.
388   */
389  void ClassDeclaration() :
390  {
391          String  modFinal = " ", modPublic = " ", defStrictfp = " ";
392  }
393  {
394    //( "abstract" | "final" | "public" | "strictfp")*
395    (
396          "abstract"{classTok.isAbstract(); }//classTok.getIsAbstract ();}
397          |"final"{modFinal = "final";}
398          | "public"{modPublic = "public";}
399          | "strictfp"{defStrictfp = "strictfp";}
400    )*
401    {classTok.getModifyClass(modFinal, modPublic, defStrictfp); }
402    UnmodifiedClassDeclaration()
403  }
404
405  void UnmodifiedClassDeclaration() :
406  {Token t;}
407  {
408    "class"
409    //<IDENTIFIER>
410    t = <IDENTIFIER>
411    {
412          String nameofClass = t.image;
413          classTok.getClassName(nameofClass);
414          doc.getClassName(nameofClass);//store name
415          att.getClassName(nameofClass);
416          met.getClassName(nameofClass);
417    }
418    //[ "extends" Name() ]
419    [
420          "extends"
421          //name of class that is inheritted
422          t = <IDENTIFIER>
423          {String inheritedName = t.image; classTok.getExtends(inheritedName);}
424    ]
425    //[ "implements" NameList() ]
426    [ "implements"
427          //declar vector to store name
428          {Vector list = new Vector();}
429          getNameList(list)
430          {classTok.getImplements(list);}
431    ]
432    ClassBody()
433  }
434  void ClassBody() :
435  {}
436  {
437    "{" ( ClassBodyDeclaration() )* "}"
438    //end of class
439    {
440          att.endofClass();
441          classTok.writeAttributeElement();
442          met.endofClass();
443          classTok.writeMethodElement();
```

```
444                 classTok.endofClass();
445             }
446         }
447         void NestedClassDeclaration() :
448         {}
449         {
450             ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")*
451             UnmodifiedClassDeclaration()
452             {classTok.isNestedClass();}
453         }
454
455         void ClassBodyDeclaration() :
456         {}
457         {
458             //LOOKAHEAD(2)
459             // Initializer()|
460             LOOKAHEAD( ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")* "class" )
461             NestedClassDeclaration()
462         |
463             LOOKAHEAD( ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")*
464         "interface" )
465             NestedInterfaceDeclaration()
466         |
467             LOOKAHEAD( [ "public" | "protected" | "private" ] <IDENTIFIER> "(" )
468             myConstructorDeclaration()
469         |
470             LOOKAHEAD( MethodDeclarationLookahead() )
471             MethodDeclaration()
472         |
473
474             FieldDeclaration()
475         }
476
477         void InterfaceDeclaration() :
478         {}
479         {
480             ( "abstract" | "public" | "strictfp")*
481             UnmodifiedInterfaceDeclaration()
482         }
483
484         void NestedInterfaceDeclaration() :
485         {}
486         {
487             ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")*
488             UnmodifiedInterfaceDeclaration()
489         }
490
491         void UnmodifiedInterfaceDeclaration() :
492         {Vector list = new Vector();}
493         {
494             "interface" <IDENTIFIER> [ "extends" getNameList(list) ]
495             "{" ( InterfaceMemberDeclaration() )* "}"
496         }
497
```

```
498     void InterfaceMemberDeclaration() :
499     {}
500     {
501       LOOKAHEAD( ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")* "class" )
502       NestedClassDeclaration()
503     |
504       LOOKAHEAD( ( "static" | "abstract" | "final" | "public" | "protected" | "private" | "strictfp")*
        "interface" )
505       NestedInterfaceDeclaration()
506     |
507       LOOKAHEAD( MethodDeclarationLookahead() )
508       MethodDeclaration()
509     |
510
511       FieldDeclaration()
512
513     }
514     // This production is to determine lookahead only.
515     void MethodDeclarationLookahead() :
516     {}
517     {
518       ( "public" | "protected" | "private" | "static" | "abstract" | "final" | "native" | "synchronized" |
        "strictfp")*
519       ResultType() <IDENTIFIER> "("
520     }
521     void Type() :
522     {}
523     {
524       ( PrimitiveType() | <IDENTIFIER> ) ( "[" "]" )*
525     }
526
527     void PrimitiveType() :
528     {}
529     {
530       "boolean"
531     |
532       "char"
533     |
534       "byte"
535     |
536       "short"
537     |
538       "int"
539     |
540       "long"
541     |
542       "float"
543     |
544       "double"
545     |
546     }
547
548     void ResultType() :
549     {}
550     {
551       "void"
552     |
553       Type()
554     }
```

```
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//
FIELD

void FieldDeclaration() :
{
        String modify=" ",defStatic=" ",defFinal=" ",defTransient=" ",defVolatile=" ",type;
        Token t;
}
{
 //( "public" | "protected" | "private" | "static" | "final" | "transient" | "volatile" )*
 (
        "public"{modify = "public";}
      | "protected"{modify = "protected";}
      | "private"{modify = "private";}
      | "static"{defStatic = "static";}
      | "final"{defFinal = "final";}
      | "transient"{defTransient = "transient";}
      | "volatile"{defVolatile = "volatile"; }
 )*
 {att.getModifiedAttribute(modify,defStatic,defFinal,defTransient,defVolatile);}
 //Type()
 (
        type = myPrimitiveType(){ att.getPrimitiveType(type);}
        |
        t = <IDENTIFIER>
        {
                        //t = getToken(1);
                        type = t.image;
                        //t = getNextToken();
                        att.getReferenceType(type);
                        //t = getToken(1);
        }
 )
 ( "[" "]" {att.checkIsArray();})*
 myVariableDeclarator() ( "," myVariableDeclarator() )* ";"
}
String myPrimitiveType() :
{}
{
  "boolean"{String type = "boolean"; return type;}
 |
  "char"{type = "char"; return type;}
 |
  "byte"{type = "byte"; return type;}
 |
  "short"{type = "short"; return type;}
 |
  "int"{type = "int"; return type;}
 |
  "long"{type = "long"; return type;}
 |
  "float"{type = "float"; return type;}
 |
  "double"{type = "double"; return type;}
  //added by Uchanong
 |"String"{type = "String"; return type;}
}
```

```
612   void myVariableDeclarator() :
613   {Token t; String name;}
614   {
615    //<IDENTIFIER>( "[" "]" )*
616    t = <IDENTIFIER>
617    {
618                  name = t.image;
619                  att.getAttributeName(name);
620    }
621    ( "[" "]" {att.checkIsArray();})*{att.writeAttributeElement();}
622    ["="myVariableInitializer()]{att.writeEndTag();}
623   }
624   void myVariableInitializer() :
625   {Token t;}
626   {
627    //case 1 -- "new"
628                  "new"
629                  t = <IDENTIFIER>{String nameObject = t.image; att.getInstanceName(nameObject);}
630                  "("
631                  //it will occur two case - "new", parameter
632                   (
633                              "new"
634                              t = <IDENTIFIER>{String name = t.image;}
635                              "("
636                                        {
637                                                  t = getToken(1);
638                                                  String parameter ="";
639                                                  while(t.image !=")") {
640                                                            parameter = parameter+t.image;
641                                                            t = getNextToken();
642                                                            t = getToken(1);
643                                                  }
644                                        att.getNewInstanceonParameter(nameObject,
      parameter);
645                                        }
646                              ")"
647                   |
648                                        {
649                                                  t = getToken(1);
650                                                  parameter ="";
651                                                  while(t.image !=")") {
652                                                            parameter = parameter+t.image;
653                                                            t = getNextToken();
654                                                            t = getToken(1);
655                                                  }
656                                                  att.getParaInstance(parameter);
657                                        }
658                              )
659                   ")"
660    |//case 2-- "{ }"
661                  "{"
662                              {
663                                        Vector value = new Vector();
664                                        t = getToken(1);
665                                        while(t.image !="}") {
666                                                  if(t.image != ",") {
667                                                            value.addElement(t.image);
668                                                  }
```

```
669                                         t = getNextToken();
670                                         t = getToken(1);
671                                     }
672                                     att.getInitialValue(value);
673                             }
674                     "}"
675     |//case 3 -- assign value
676             {
677                             t = getToken(1);
678                             String v = t.image;
679                             att.getInitialValue(v);
680                             t = getNextToken();
681             }
682     }
683     //###################################                 METHOD
684
685     void MethodDeclaration() :
686     {
687             String modify=" ",defStatic=" ",defFinal=" ",defTransient=" ",defVolatile=" ",defAbstract=
        "",defNative="",defSyn="",defStrictfp="",nameofMethod;
688             Token t;
689     }
690     {
691     //( "public" | "protected" | "private" | "static" | "abstract" | "final" | "native" | "synchronized" | "strictfp")*
692     (
693                     "public"{modify = "public";}
694                     | "protected"{modify = "protected";}
695                     | "private"{modify = "private";}
696                     | "static"{defStatic = "static";}
697                     | "abstract"{defAbstract = "abstract";}
698                     | "final"{defFinal = "final";}
699                     |"native"{defNative = "native";}
700                     | "synchronized"{defSyn="synchronized";}
701                     | "strictfp"{defStrictfp = "strictfp"; }
702     )*
703     {met.getModifiedMethod(modify,defStatic,defAbstract,defFinal,defNative,defSyn,defStrictfp);}
704     //return type
705     myResultType()
706     //myMethodDeclarator() //[ "throws" NameList() ]
707     //name of method
708     t = <IDENTIFIER>
709     { nameofMethod = t.image; met.getMethodName(nameofMethod);}
710     //parameter
711      //stor every thing in "(...)" except "main" method, i don't care anything( "(..)") in main method
712      "("
713             {
714                             t = getToken(1);
715                             int i =1;//use in analysing pass parameter to getParameter Method
716                             Vector type = new Vector();
717                             Vector namePara = new Vector();
718                             while(t.image != ")") {
719                                     //not store "," in Vector
720                                     if(t.image !=",") {
721                                             //if result is odd number -- it's type
722                                             //if result is even number -- it's namePara
723                                             if(i%2 == 1) {//odd
724                                                     if(nameofMethod !=
        "main") type.addElement(t.image);
725                                     }
```

```
726  {
727                                                          else {
                                                                    if(nameofMethod !=
     "main") namePara.addElement(t.image);
728                                                         }
729                                                  }//end if -- t.image != ","
730                                                  ++i;
731                                                  t = getNextToken();
732                                                  t = getToken(1);
733                                          }//end while -- t.image != ")"
734                                          //in call method of Method class -- call 2 method
735  --                                      if(!nameofMethod.equals("main")) {
736  --                                              met.getParameter(type,namePara);
737                                          }
738              }
739    ")"
740    [ "throws"{Vector throwsList = new Vector();getNameList(throwsList);met.getThrowsList(
     throwsList);}]
741              (Block() | ";" )
742     {met.endofMethod();}
743  }
744  void myResultType() :
745  {Token t;String resultType;}
746  //it has 3 choose to analyse
747  {
748  //case 1 -- primitive type
749              resultType = myPrimitiveType()
750              {met.getResultPrimitiveType(resultType);}
751  |
752  //case 2 -- reference type
753              t = <IDENTIFIER>
754              {
755                      resultType = t.image;
756                      met.getResultReferenceType(resultType);
757              }
758  //case 3 -- nothing return type
759  |
760              "void"
761              {resultType = "void"; met.getNonResult();}
762  }
763  void Block() :
764  {}
765  {
766    "{" ( BlockStatement() )* "}"
767  }
768
769  void BlockStatement() :
770  {}
771  {
772    LOOKAHEAD([ "final" ] Type() <IDENTIFIER>)
773    myLocalVariableDeclaration() ";"
774  |
775    myStatement()
776
777  }
778
```

```
779   void myLocalVariableDeclaration() :
780   {Token t; String type;}
781   {
782    [ "final"{met.getFinalLocal();}]
783    //Type()
784    (
785                  type = myPrimitiveType()
786                  |
787                  t = <IDENTIFIER>
788                   {type = t.image;}
789    ){met.getLocalType(type);}
790    ( "[" "]" {met.isLocalArray();})*
791    //VariableDeclarator() ( "," VariableDeclarator() )*
792    localVariableDeclarator() ( "," localVariableDeclarator() )*
793   }
794   void localVariableDeclarator() :
795   {}
796   {
797    localVariableDeclaratorId()
798    {met.writeLocalVariableElement();}
799    [ "=" localVariableInitializer() ]
800    {met.writeEndLocalTag();}
801   }
802   void localVariableDeclaratorId() :
803   {Token t; String name;}
804   {
805    //<IDENTIFIER>( "[" "]" )*
806    t = <IDENTIFIER>
807    {
808                  name = t.image;
809                  met.getLocalName(name);
810    }
811    ( "[" "]" {met.isLocalArray();})*
812   }
813   void localVariableInitializer() :
814   {Token t;}
815   {
816    //case 1 -- "new"
817          "new"
818                  newObject()
819    |//case 2-- "{  }"
820                  "{"
821                        {
822                              Vector value = new Vector();
823                              t = getToken(1);
824                              while(t.image !="}") {
825                                    if(t.image != ",") {
826                                          value.addElement(t.image);
827                                    }
828                                    t = getNextToken();
829                                    t = getToken(1);
830                              }
831                              met.getInitialValue(value);
832                        }
833                  "}"
834    |//case 3 -- assign value
835                  {
836                        t = getToken(1);
837                        String v = t.image;
```

```
838                         met.getInitialValue(v);
839                         t = getNextToken();
840                 }
841     }
842     void myStatement() :
843     {}
844     {
845       LOOKAHEAD(<IDENTIFIER>"."<IDENTIFIER>"(")
846       callMethod()
847       |
848       LOOKAHEAD(<IDENTIFIER>"(")
849       callMethodinClass()
850       |
851       returnStatement()
852       |
853     //when instance object in body of method with " name = new ....."
854       LOOKAHEAD(<IDENTIFIER>"=" "new" <IDENTIFIER>"(")
855       instanceStatement()
856       |
857       TryStatement()
857       TryStatement()
858     // added by Andrea Gini
859       |
860       AssertStatement()
861       |
862       myDoWhileStatement()
863       |
864       myWhileStatement()
865       |
866       myIfStatement()
867       |
868       mySwitchStatement()
869       |
870       myForStatement()
871       |
872       getStatement()
873     }
874     void instanceStatement():
875     {Token t;}
876     {
877                 t = <IDENTIFIER>
878                 {String name = t.image;met.getNameInstanceStatement(name);}
879                 "=" "new"
880                 newObject()
881                 ";"{met.getEndInstanceTag();}
882     }
883     void newObject():
884     {Token t;}
885     {
886                 t = <IDENTIFIER>{String nameObject = t.image; met.getInstanceName(nameObject);}
887                 "("
888                   //it will occur two case - "new", parameter
889                   (
890                             "new"
891                             t = <IDENTIFIER>{String name = t.image;}
892                             "("
893                                       {
894                                                 t = getToken(1);
```

```
895                                                    String parameter ="";
896                                                    while(t.image !=")") {
897                                                                parameter = parameter+t.image;
898                                                                t = getNextToken();
899                                                                t = getToken(1);
900                                                    }
901                                                    met.getNewInstanceonParameter(nameObject,
    parameter);
902                                                    }
903                                        ")"
904                                 |
905                                                    {
906                                                    t = getToken(1);
907                                                    parameter ="";
908                                                    while(t.image !=")") {
909                                                                parameter = parameter+t.image;
910                                                                t = getNextToken();
911                                                                t = getToken(1);
912                                                    }
913                                                    met.getParaInstance(parameter);
914                                                    }
915                                        )
916                                 ")"
917     }
918     void callMethod() :
919     {Token t; String objectName,methodName;}
920     {
921                 t = <IDENTIFIER>
922                 {objectName = t.image;}
923                 "."
924                 t = <IDENTIFIER>
925                 {methodName = t.image;met.callOtherMethod(objectName, methodName);}
926                 "("
927                 sendParameter()
928                 ";"
929     }
930     void callMethodinClass():
931     {Token t; String methodName;}
932     {
933                 t = <IDENTIFIER>
934                 {methodName = t.image;}
935                 "("
936                 sendParameter()
937                 ";"
938                 {met.callMethodinClass(methodName);}
939     }
940     JAVACODE
941     void sendParameter() {
942                 Token t;
943                 t = getToken(1);
944                 Vector para = new Vector();
945                 while(t.image != ";") {
946                             para.addElement(t.image);
947                             t = getNextToken();
948                             t = getToken(1);
949                 }
950                 if(!para.isEmpty()) met.sendParameter(para);
951     }
```

```
952     void returnStatement( ):
953     { Token t; String returnName;}
954     {
955                 "return"
956                 t = <IDENTIFIER>
957                 { returnName = t.image;}
958                 ";"
959                 {met.getReturn(returnName);}
960     }
961     void myDoWhileStatement( ) :
962     {Token t;}
963     {
964                 "do"{met.getDoWhile( );}
965                 {met.getStartBody( );}
966                             Block( )
967                 {met.getEndBody( );}
968                 "while"
969                 "("
970                             {
971                                         Vector condition = new Vector( );
972                                         //to check one more condition
973                                         Stack parenthesis = new Stack( );
974                                         parenthesis.push("(");
975                                         condition.addElement(new String("("));
976                                         boolean endofCondition = false;
977                                         t = getToken(1);
978                                         while(endofCondition != true) {
979                                                     condition.addElement(t.image);
980                                                     t = getNextToken( );
981                                                     t = getToken(1);
982                                                     if(t.image == ")") {
983                                                                 parenthesis.pop( );
984                                                                 if(parenthesis.empty( ))
        endofCondition = true;
985                                                     }
986                                                     if(t.image == "(")parenthesis.push(t.image);
987                                         }
988                                         met.getCondition(condition);
989                             }
990                 ")"
991                 ";"{met.endDoWhile( );}
992     }
993     void myWhileStatement( ) :
994     {Token t;}
995     {
996                 "while"{met.getWhile( );}
997                 "("
998                             {
999                                         Vector condition = new Vector( );
1000                                        //to check one more condition
1001                                        Stack parenthesis = new Stack( );
1002                                        parenthesis.push("(");
1003                                        condition.addElement(new String("("));
1004                                        boolean endofCondition = false;
1005                                        t = getToken(1);
```

```
1006                                    while(endofCondition != true) {
1007                                            condition.addElement(t.image);
1008                                            t = getNextToken();
1009                                            t = getToken(1);
1010                                            if(t.image == ")") {
1011                                                    parenthesis.pop();
1012                                                    if(parenthesis.empty())
        endofCondition = true;
1013                                            }
1014                                            if(t.image == "(")parenthesis.push(t.image);
1015                                    }
1016                                    met.getCondition(condition);
1017                    }
1018            ")"
1019            //body of while
1020            (
1021                    {met.getStartBody();}
1022                      Block()
1023                    {met.getEndBody();}
1024                    |
1025                    {met.getStartBody();}
1026                    getBodyInLine()
1027                    ";"{met.getEndBody();}
1028            )
1029            {met.endWhile();}
1030    }
1031    void myIfStatement():
1032    {
1033            Token t;
1034    }
1035    {
1036            "if"{met.getIf();}
1037            "("
1038                    {
1039                            Vector condition = new Vector();
1040                            //to check one more condition
1041                            Stack parenthesis = new Stack();
1042                            parenthesis.push("(");
1043                            condition.addElement(new String("("));
1044                            boolean endofCondition = false;
1045                            t = getToken(1);
1046                            while(endofCondition != true) {
1047                                    condition.addElement(t.image);
1048                                    t = getNextToken();
1049                                    t = getToken(1);
1050
1051                                    if(t.image == ")") {
1052                                            parenthesis.pop();
1053                                            if(parenthesis.empty())
        endofCondition = true;
1054                                    }
1055                                    if(t.image == "(")parenthesis.push(t.image);
1056                            }
1057                            met.getCondition(condition);
1058                    }
1059            ")"
```

```
1060            //body of if
1061            (
1062                    {met.getStartBody();}
1063                       Block()
1064                    {met.getEndBody();}
1065            |
1066                    {met.getStartBody();}
1067                    getBodyInLine()
1068                    ";"{met.getEndBody();}
1069            )
1070            {met.endIf();}
1071            (
1072                    "else"{met.getElse();}
1073                    (
1074                            {met.getStartBody();}
1075                               Block()
1076                            {met.getEndBody();}
1077                    |
1078                            {met.getStartBody();}
1079                            getBodyInLine()
1080                            ";"{met.getEndBody();}
1081                    )
1082                    {met.endElse();}
1083            )*
1084    }
1085    void mySwitchStatement() :
1086    {
1087        Token t;
1088    }
1089    {
1090            "switch"{met.getSwitch();}
1091            "("
1092                    {
1093                            Vector expression = new Vector();
1094                            t = getToken(1);
1095                            while(t.image != ")") {
1096                                    expression.addElement(t.image);
1097                                    t = getNextToken();
1098                                    t = getToken(1);
1099                            }
1100                            met.getExpression(expression);
1101                    }
1102            ")"
1103            "{"
1104            //case
1105            (
1106                    "case"
1107                    //constant
1108                    {
1109                            t = getToken(1);
1110                            String constant = t.image;
1111                            t = getNextToken();
1112                            met.getConstantCase(constant);
1113                    }
1114                    ":"
1115                    (getStatement())*
1116                    {met.endCase();}
1117            )*
```

```
1117                )*
1118                //default
1119                ("default" ":"(getStatement())*)
1120                "}"
1121    }
1122
1123    void myForStatement() :
1124    {
1125                Token t;
1126    }
1127    {
1128                "for"{met.getFor();}
1129                "("
1130                        {
1131
1132                                t = getToken(1);
                                ?
1133                                int checkInitValue = 0;// to check that more than one value
1134
1135                                Vector init = new Vector();
1136                                //store initial value
1137                                while(t.image != ";") {
1138                                        if(t.image == ",") {
1139                                                ++checkInitValue;
1140                                        }
1141                                        init.addElement(t.image);
1142                                        t = getNextToken();
1143                                        t = getToken(1);
1144                                }
1145                                met.getInitForStatement(init);
1146                        }
1147                        ";"
1148                        //store condition
1149                        {
1150                                t = getToken(1);
1151                                Vector condition = new Vector();
1152                                while(t.image != ";") {
1153                                        condition.addElement(t.image);
1154                                        t = getNextToken();
1155                                        t = getToken(1);
1156                                }
1157                                met.getCondition(condition);
1158                        }
1159                        ";"
1160                        //increment or decrement count
1161                        {
1162                                t = getToken(1);
1163                                Vector count = new Vector();
1164                                while(t.image != ")") {
1165                                        count.addElement(t.image);
1166                                        t = getNextToken();
1167                                        t = getToken(1);
1168                                }
1169                                met.getCountForStatement(count);
1170                        }
                //body of for loop
```

```
1171            (
1172                    {met.getStartBody();}
1173                       Block()
1174                    {met.getEndBody();}
1175            |
1176                     {met.getStartBody();}
1177                     getBodyInLine()
1178                     ";"{met.getEndBody();}
1179            )
1180            {met.endFor();}
1181    }
1182    void getStatement() :
1183    {
1184            Vector statement = new Vector();
1185            Token t;
1186    }
1187    {
1188      t = <IDENTIFIER>
1189      {
1190            statement.addElement(t.image);
1191            while (t.image != ";") {
1192                                    statement.addElement(t.image);
1193                                    t = getNextToken();
1194                                    t = getToken(1);
1195            }
1196            met.getStatement(statement);
1197      }
1198      ";"
1199      |
1200      t = <INCR>| t= <DECR>
1201      {
1202            t = getToken(1);
1203            while (t.image != ";") {
1204                                    statement.addElement(t.image);
1205                                    t = getNextToken();
1206                                    t = getToken(1);
1207            }
1208            met.getStatement(statement);
1209      }
1210      ";"
1211    }
1212    JAVACODE
1213    void getBodyInLine() {
1214                            Token t;
1215                             t = getToken(1);
1216                             Vector body = new Vector();
1217                             while (t.image != ";") {
1218                                    body.addElement(t.image);
1219                                    t = getNextToken();
1220                                    t = getToken(1);
1221                             }
1222                             // met.getBodyInLine(body);
1223    }
1224
```

```
1225    void myConstructorDeclaration( ) :
1226    {Token t; String modify="";}
1227  {
1228    [
1229                  "public" {modify = "public"; met.getModifyConstructor(modify);}
1230                | "protected"{modify = "protected"; met.getModifyConstructor(modify);}
1231                | "private"{modify = "private"; met.getModifyConstructor(modify);}
1232    ]
1233    t = <IDENTIFIER>
1234    {String name = t.image; met.getNameConstructor(name);}
1235    //FormalParameters() [ "throws" NameList() ]"{"
1236    //[ LOOKAHEAD(ExplicitConstructorInvocation()) ExplicitConstructorInvocation() ]
1237     //( BlockStatement() )*"}"
1238     "("
1239              {
1240                                    t = getToken(1);
1241                                    int i = 1;//use in analysing pass parameter to getParameter Method
1242                                    Vector type = new Vector( );
1243                                    Vector namePara = new Vector( );
1244                                    while(t.image != ")") {
1245                                          //not store "," in Vector;
1246                                          if(t.image !=",") {
1247                                                //if result is odd number -- it's type
1248                                                //if result is even number -- it's namePara
1249                                                if(i%2 == 1) {//odd
1250                                                      type.addElement(t.image
      );
1251                                                }
1252                                                else {
1253                                                      namePara.addElement(t.
      image);
1254                                                }
1255                                          }//end if -- t.image != ","
1256                                          ++i;
1257                                          t = getNextToken( );
1258                                          t = getToken(1);
1259                                    }//end while -- t.image != ")"
1260                                    //in call method of Method class -- call 2 method
1261                                    met.getParameter(type,namePara);
1262              }
1263     ")"
1264     [ "throws"{Vector throwsList = new Vector( );getNameList(throwsList);met.getThrowsList(
      throwsList);}]
1265     //(Block() | ";" )
1266     Block( )
1267     {met.endofMethod( );}
1268  }
1269
```

ภาคผนวก ค

เอกสารทางเอ็กซ์เอ็มไอและโค้ดจาวาที่นำมาทดลอง

เอกสารทางเอ็กซ์เอ็มไอที่ได้จากการแปลง JavaMapToUML.java

```
<?xml version="1.0"?>
<xmi xmi.version="1.1">
<xmi.header>
        <xmi.doucumentation>
                This is XMI Document that transformed JavaMapToUML.java
        </xmi.doucumentation>
</xmi.header>
<xmi.content>
        <Class name = "JavaMapToUML" visibility = "public">
                <Method name = "main" visibility = "public" return = "void">
                        <New name="ClassDiagram" type = "DefinedClass"/>
                        <LocalVariable name = "ClassDiagram" type = "ClassDiagram"
/>
                        <CallOtherMethod objectName="ClassDiagram" methodName
="ClassDiagram" return = "false"/>
                </Method>
        </Class>

        <Class name = "ClassDiagram" visibility = "public">
                <Extends name="Graph" type = "DefinedClass"/>
                <Attribute name = "numberOfRelation" visibility = "private" type =
"String"/>
                <Attribute name = "classNode" visibility = "private" type = "ClassNode"/>
                <New name="ClassNode" type = "DefinedClass"/>
                <Attribute name = "relation" visibility = "private" type = "Relation"/>
                <New name="Relation" type = "DefinedClass"/>
                <Constructor name = "ClassDiagram" visibility = "public" return = "void">
                        <Parameter name = "n" type = "String"/>
                        <CallMethod methodName ="paint"/>
                </Constructor>
                <Method name = "setNumberOfRelation" visibility = "public" return =
"void">
                        <Parameter name = "n" type = "int"/>
                </Method>
                <Method name = "getNumberOfRelation" visibility = "public" return = "int"
/>
                <Method name = "XMIToClassDiagram" visibility = "public" return =
"void">
                        <New name="ClassNode" type = "DefinedClass"/>
                        <CallOtherMethod objectName="classNode" methodName =
"ClassNode" return = "false"/>
                        <CallOtherMethod objectName="classNode" methodName =
"setNumberOfAttribute" return = "false"/>
                        <CallOtherMethod objectName="classNode" methodName =
"setNumberOfMethod" return = "false"/>
```

```
37                                  <CallOtherMethod objectName="classNode" methodName =
"setCoordinateX" return = "false"/>
38                                  <CallOtherMethod objectName="classNode" methodName =
"setCoordinateY" return = "false"/>
39                                  <CallOtherMethod objectName="classNode" methodName =
"setWidth" return = "false"/>
40                                  <CallOtherMethod objectName="classNode" methodName =
"setHeight" return = "false"/>
41                                  <New name="ClassNode" type = "DefinedClass"/>
42                                  <CallOtherMethod objectName="classNode" methodName =
"ClassNode" return = "false"/>
43                                  <CallOtherMethod objectName="classNode" methodName =
"setNumberOfAttribute" return = "false"/>
44                                  <CallOtherMethod objectName="classNode" methodName =
"setNumberOfMethod" return = "false"/>
45                                  <CallOtherMethod objectName="classNode" methodName =
"setCoordinateX" return = "false"/>
46                                  <CallOtherMethod objectName="classNode" methodName =
"setCoordinateY" return = "false"/>
47                                  <CallOtherMethod objectName="classNode" methodName =
"setWidth" return = "false"/>
48                                  <CallOtherMethod objectName="classNode" methodName =
"setHeight" return = "false"/>
49
50                                  <New name="Relation" type = "DefinedClass"/>
51                                  <CallOtherMethod objectName="relation" methodName =
"Relation" return = "false"/>
52                                  <CallOtherMethod objectName="relation" methodName =
"setSource" return = "false"/>
53                                  <CallOtherMethod objectName="relation" methodName =
"setSink" return = "false"/>
54                          </Method>
55                          <Method name = "paint" visibility = "public" return = "void">
56                                  <Parameter name = "g" type = "Graphics"/>
57                                  <CallMethod methodName ="XMIToClassDiagram"/>
58                                  <CallOtherMethod objectName="classNode" methodName =
"drawClassNode" return = "false"/>
59                                  <CallOtherMethod objectName="relation" methodName =
"drawRelation" return = "false"/>
60                          </Method>
61                  </Class>
62
63                  <Class name = "Graph" visibility = "public">
64                          <Attribute name = "name" visibility = "public" type = "String" />
65                          <Attribute name = "numberOfClass" visibility = "public" type = "int"/>
66                          <Constructor name = "Graph" visibility = "public" return = "void">
67                                  <Parameter name = "n" type = "String"/>
68                          </Constructor>
69                          <Method name = "setNumberOfClass" visibility = "public" return = "void">
70                                  <Parameter name = "n" type = "int"/>
71                          </Method>
72                          <Method name = "getNumberOfClass" visibility = "public" return = "int"/>
73                  </Class>
74
```

```
<Class name = "ClassNode" visibility = "public">
        <Extends name="Node" type = "DefinedClass"/>
        <Attribute name = "numberOfMethod" visibility = "privatc" type = "int"/>
        <Attribute name = "numberOfAttribute" visibility = "private" type = "int"/>
        <Constructor name = "ClassNode" visibility = "public" return = "void">
                <Parameter name = "n" type = "String"/>
        </Constructor>
        <Method name = "setNumberOfMethod" visibility = "public" return =
"void">
                <Parameter name = "n" type = "int"/>
        </Method>
        <Method name = "setNumberOfAttribute" visibility = "public" return =
"void">
                <Parameter name = "n" type = "int"/>
        </Method>
        <Method name = "getNumberOfMethod" visibility = "public" return = "int"/>
        <Method name = "getNumberOfAttribute" visibility = "public" return = "int"
/>
        <Method name = "drawClassNode" visibility = "public" return = "void">
                <Parameter name = "g" type = "Graphics"/>
                <Parameter name = "n" type = "ClassNode"/>
        </Method>
</Class>

<Class name = "Node" visibility = "public">
        <Attribute name = "name" visibility = "public" type = "String" />
        <Attribute name = "coordinateX" visibility = "public" type = "int"/>
        <Attribute name = "coordinateY" visibility = "public" type = "int"/>
```

```
                <Parameter name = "n" type = "int"/>
        </Method>
        <Method name = "setNumberOfAttribute" visibility = "public" return =
"void">
                <Parameter name = "n" type = "int"/>
        </Method>
        <Method name = "getNumberOfMethod" visibility = "public" return = "int"/>
        <Method name = "getNumberOfAttribute" visibility = "public" return = "int"
/>
        <Method name = "drawClassNode" visibility = "public" return = "void">
                <Parameter name = "g" type = "Graphics"/>
                <Parameter name = "n" type = "ClassNode"/>
        </Method>
</Class>

<Class name = "Node" visibility = "public">
        <Attribute name = "name" visibility = "public" type = "String" />
        <Attribute name = "coordinateX" visibility = "public" type = "int"/>
        <Attribute name = "coordinateY" visibility = "public" type = "int"/>
        <Attribute name = "width" visibility = "public" type = "int"/>
        <Attribute name = "height" visibility = "public" type = "int"/>
        <Constructer name = "Node" visibility = "public" return = "void">
                <Parameter name = "n" type = "String"/>
        </Constructer>
```

```
100                     <Attribute name = "width" visibility = "public" type = "int"/>
101                     <Attribute name = "height" visibility = "public" type = "int"/>
102                     <Constructor name = "Node" visibility = "public" return = "void">
103                             <Parameter name = "n" type = "String"/>
104                     </Constructor>
105                     <Method name = "setCoordinateX" visibility = "public" return = "void">
106                             <Parameter name = "x" type = "int"/>
107                     </Method>
108                     <Method name = "setCoordinateY" visibility = "public" return = "void">
109                             <Parameter name = "y" type = "int"/>
110                     </Method>
111                     <Method name = "setWidth" visibility = "public" return = "void">
112                             <Parameter name = "w" type = "int"/>
113                     </Method>
114                     <Method name = "setHeight" visibility = "public" return = "void">
115                             <Parameter name = "h" type = "int"/>
116                     </Method>
117                     <Method name = "getName" visibility = "public" return = "String"/>
118                     <Method name = "getCoordinateX" visibility = "public" return = "int"/>
119                     <Method name = "getCoordinateY" visibility = "public" return = "int"/>
120                     <Method name = "getWidth" visibility = "public" return = "int"/>
121                     <Method name = "getHeight" visibility = "public" return = "int"/>
122             </Class>
123
124             <Class name = "Relation" visibility = "public">
125                     <Extends name="Edge" type = "DefinedClass"/>
126                     <Constructor name = "Relation" visibility = "public" return = "void">
127                             <Parameter name = "n" type = "String"/>
128                     </Constructor>
129                     <Method name = "drawRelation" visibility = "public" return = "void">
130                             <Parameter name = "g" type = "Graphics"/>
131                             <Parameter name = "e" type = "Edge"/>
132                             <Parameter name = "classNode" type = "ClassNode"/>
133                             <Parameter name = "n_Class" type = "int"/>
134                             <CallMethod methodName ="getSource"/>
135                             <CallMethod methodName ="getSink"/>
136                             <CallOtherMethod objectName="classNode" methodName =
"getCoordinateX" return = "true"/>
137                             <CallOtherMethod objectName="classNode" methodName =
"getWidth" return = "true"/>
138                             <CallOtherMethod objectName="classNode" methodName =
"getCoordinateY" return = "true"/>
139                             <CallOtherMethod objectName="classNode" methodName =
"getHeight" return = "true"/>
140                     </Method>
141             </Class>
142
143             <Class name = "Edge" visibility = "public">
144                     <Attribute name = "name" visibility = "public" type = "String" />
145                     <Attribute name = "type" visibility = "public" type = "String"/>
146                     <Attribute name = "sourceNode" visibility = "public" type = "Node"/>
147                     <Attribute name = "sinkNode" visibility = "public" type = "Node"/>
148                     <Constructor name = "Relation" visibility = "public" return = "void">
```

```
149                     <Parameter name = "n" type = "String"/>
150             </Constructor>
151             <Constructor name = "Relation" visibility = "public" return = "void">
152                     <Parameter name = "n" type = "String"/>
153                     <Parameter name = "t" type = "String"/>
154             </Constructor>
155             <Method name = "setSource" visibility = "public" return = "void">
156                     <Parameter name = "n" type = "Node"/>
157             </Method>
158             <Method name = "setSink" visibility = "public" return = "void">
159                     <Parameter name = "n" type = "Node"/>
160             </Method>
161             <Method name = "getName" visibility = "public" return = "String"/>
162             <Method name = "getType" visibility = "public" return = "String"/>
163             <Method name = "getSource" visibility = "public" return = "Node"/>
164             <Method name = "getSink" visibility = "public" return = "Node"/>
165     </Class>
166 </xmi.content>
167 </xmi>
168
169
```

รายละเอียดโค้ดจาวา

- JavaMapToUML.java

```
1  public class JavaMapToDiagram {
2          public static void main(String[] args){
3                  new ClassDiagram("");
4          }
5  }
```

- Edge.java

```
1  public class Edge
2  {
3          public String name;
4          public String type;
5          public Node sourceNode;
6          public Node sinkNode;
7
8          public Edge(String n){
9                  name = n;
10         }
11         //-----------------------------------------------------------
12         public Edge(String n,String t){
13                 name = n;
14                 type = t;
15         }
16         //-----------------------------------------------------------
17         public void setSource(Node n){
18                 sourceNode = n;         }
19         //-----------------------------------------------------------
20         public void setSink(Node n){
21                 sinkNode = n;           }
22         //-----------------------------------------------------------
23         public String getName(){        return name;            }
24         //-----------------------------------------------------------
25         public String getType(){ return type;           }
26         //-----------------------------------------------------------
27         public Node getSource(){        return sourceNode;      }
28         //-----------------------------------------------------------
29         public Node getSink(){  return sinkNode;        }
30         //-----------------------------------------------------------
31 }//end class Edge
```

- ClassNode

```
1   import java.awt.*;
2   public class ClassNode extends Node
3   {
4               private int numberOfMethod;
5               private int numberOfAttribute;
6
7               public ClassNode(String n){
8                       super(n);
9               }
10
            //-------------------------------------------------------------
                //delete (no use)
11              public void setNumberOfMethod(int n){          numberOfMethod = n;   }
12              //-------------------------------------------------------------
13              public void setNumberOfAttribute(int n){       numberOfAttribute = n;  }
14              //-------------------------------------------------------------
15              public int getNumberOfMethod(){    return numberOfMethod;          }
16              //-------------------------------------------------------------
17              public int getNumberOfAttribute(){   return numberOfAttribute;     }
18              //-------------------------------------------------------------
19              public void drawClassNode(Graphics g,ClassNode n){
20                      int spaceX = 10;
21                      int spaceY = 10;
22                      Font font = new Font("Cordia New",Font.PLAIN,20);
23                      g.getFont();
24                      FontMetrics fm = g.getFontMetrics(font);
25                      g.drawRect(coordinateX,coordinateY,width,height);
26                      int y = coordinateY+spaceY+fm.getAscent();
27                      g.drawString(name,coordinateX+width/2-fm.stringWidth(name)/2,y);
28                      g.drawLine(coordinateX,coordinateY+2*spaceY+fm.getAscent(),coordinateX+
    width,coordinateY+2*spaceY+fm.getAscent());
29
30                      y = y - (spaceY/2+fm.getAscent()) + spaceY;
31                      g.drawLine(coordinateX,y,coordinateX+width,y);
32
33              }
34  }//end class ClassNode
```

- Node

```
1   public class Node
2   {
3           public String name;
4           public int coordinateX;
5           public int coordinateY;
6           public int width;
7           public int height;
8
9           public Node(String n){   name = n;   }
10          //-------------------------------------------------------------------
11          public void setCoordinateX(int x){    coordinateX = x;          }
12          //-------------------------------------------------------------------
13          public void setCoordinateY(int y){    coordinateY = y;          }
14          //-------------------------------------------------------------------
15          public void setWidth(int w){          width = w;                }
16          //-------------------------------------------------------------------
17          public void setHeight(int h){         height = h;   }
18          //-------------------------------------------------------------------
19          public String getName(){              return name;              }
20          //-------------------------------------------------------------------
21          public int getCoordinateX(){          return coordinateX;       }
22          //-------------------------------------------------------------------
23          public int getCoordinateY(){          return coordinateY;       }
24          //-------------------------------------------------------------------
25          public int getWidth(){     return width;              }
26          //-------------------------------------------------------------------
27          public int getHeight(){    return height;             }
28  }//end class Node
```

- Graph

```
1   import java.awt.*;
2   import java.awt.event.*;
3   public class Graph extends Canvas {
4           public String name;
5           public int numberOfClass;
6
7
8
9
10          public Graph(String n){
11                  int widthBar = 15;
12                  int widthTitle = 30;
13                  int widthFrame = 800;
14                  int heightFrame =600;
15                  String name;
16                  int numberOfClass;
17                  Frame f = new Frame();
18                  name = n;
19
20                  f.setSize(widthFrame+widthBar+5,heightFrame+widthTitle+widthBar+5);
21                  f.setVisible(true);
22                  f.addWindowListener(new CloseWindow());
23                  this.setSize(widthFrame,heightFrame);
24                  this.setLocation(0,widthTitle);
25                  f.setLayout(null);
26
27                  f.add(this);
28          }
29          //-------------------------------------------------------
30          public void setNumberOfClass(int n){        numberOfClass = n;    }
31          //-------------------------------------------------------
32          public int getNumberOfClass(){      return numberOfClass;   }
33          //-------------------------------------------------------
34
35  }
```

- Relation

```
1   import java.awt.*;
2   public class Relation extends Edge
3   {
4           public Relation(String name){
5                   super(name);
6           }
7           //-------------------------------------------------------------------
8           public Relation(String name,String type){
9                   super(name,type);
10          }
11          //-------------------------------------------------------------------
12          public void drawRelation(Graphics g,Relation e,ClassNode[] classNode,int n_Class){
13                  Node source = getSource();
14                  Node sink = getSink();
15                  int x0 = source.getCoordinateX() + source.getWidth();
16                  int y0 = source.getCoordinateY() + source.getHeight()/2;
17                  g.drawLine(x0,y0,x0+100,y0);
18          }//method
19  }//end class Relation
```

- ClassDiagram

```
1   import java.awt.*;
2   public class ClassDiagram extends Graph{
3           private int numberOfRelation;
4           private ClassNode classNode[] = new ClassNode[20];
5           private Relation relation[] =new Relation[30];
6
7           public ClassDiagram(String n){
8                   super(n);
9           }
10          //-------------------------------------------------------------------
11          public void setNumberOfRelation(int n){          numberOfRelation = n;  }
12          //-------------------------------------------------------------------
13          public int getNumberOfRelation(){    return numberOfRelation;          }
14          //-------------------------------------------------------------------
15          public void XMIToClassDiagram(Graphics g){//parse XMI document
16                  //create ClassNode Object
17                  classNode[0] = new ClassNode("test0");
18                  classNode[0].setNumberOfAttribute(0);
19                  classNode[0].setNumberOfMethod(0);
20                  classNode[0].setCoordinateX(50);
21                  classNode[0].setCoordinateY(50);
22                  classNode[0].setWidth(100);
23                  classNode[0].setHeight(80);
24                  classNode[1] = new ClassNode("test1");
25                  classNode[1].setNumberOfAttribute(0);
26                  classNode[1].setNumberOfMethod(0);
27                  classNode[1].setCoordinateX(250);
28                  classNode[1].setCoordinateY(50);
29                  classNode[1].setWidth(100);
30                  classNode[1].setHeight(80);
31
32                  numberOfClass = 2;
33                  //create Relation Object
34                  relation[0] = new Relation("r0","new");
35                  relation[0].setSource(classNode[0]);
36                  relation[0].setSink(classNode[1]);
37
38                  numberOfRelation = 1;
39          }
40          //-------------------------------------------------------------------
41          public void paint(Graphics g){
42                  XMIToClassDiagram(g);
43                  for(int i=0;i<numberOfClass;i++){
44                          classNode[i].drawClassNode(g,classNode[i]);
45                  }
46
47                  //draw Relation
48
49                  for(int i=0;i<numberOfRelation;i++){
50                          relation[i].drawRelation(g,relation[i],classNode,numberOfClass);
51                  }
52          }
53          //-------------------------------------------------------------------
54  }//end class ClassDiagram
```