เครื่องมือแปลงภาษาจาวาเป็นยูเอ็มแอล

Mapping JAVA – UML Tool

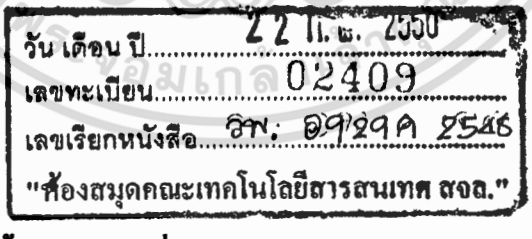
โดย

เอื้ออนงค์ พลชนะ

รหัส 46066707

อาจารย์ที่ปรึกษา

ผศ.ดร.พรฤดี เนติโสภากุล



รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 2 ปีการศึกษา 2548
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การออกแบบเครื่องมือแปลงภาษาจาวาเป็นยูเอ็มแอล
นักศึกษา	นางสาว เอื้ออนงค์ พลชนะ
อาจารย์ที่ปรึกษา	ผศ.ดร.พรฤดี เนติโสภากุล
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขวงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2548

บทคัดย่อ

การพัฒนาที่เป็นการทำความใจโค้ดนั้นเป็นเรื่องยาก วิธีการหนึ่งที่ใช้ในการทำความเข้าใจโค้ดซึ่งเป็นภาษาในการเขียนโปรแกรมนั้น คือ การทำความเข้าใจโดยสื่อออกมาเป็นภาษาเชิงภาพ/โมเดล โดยที่ภาษาเชิงภาพที่เป็นที่นิยมในการทำความเข้าใจในโค้ดนั้น คือ ภาษาเชิงภาพที่อยู่ในมาตรฐานของยูเอ็มแอล(UML – Unified Modeling Language) ในโครงงานนี้ได้ทำการสร้างเครื่องมือในการแมพโค้ดจาวาเป็นยูเอ็มแอล(คลาสไดอะแกรม) โดยมีการแมพเข้าสู่มาตรฐานเอ็กซ์เอ็มไอ(XMI – XML Metadata Interchange) เพื่อความสามารถในการแลกเปลี่ยนระหว่างเครื่องมืออื่นๆ นอกจากนี้ ในการวิเคราะห์โค้ดจาวาได้นำ JavaCC(Java Complier Complier) มาช่วยในการวิเคราะห์ สำหรับการนำมาตรฐานเอ็กซ์เอ็มไอมาเป็นตัวกลางในการแมพโค้ดจาวาเป็นยูเอ็มแอล(คลาสไดอะแกรม)นั้น ได้ทำการแมพโค้ดจาวาเป็นเอกสารเอ็กซ์เอ็มไอ ซึ่งเอกสารเอ็กซ์เอ็มไอนี้ใช้ DOM(Document Object Model) และในการวิเคราะห์เอกสารเอ็กซ์เอ็มไอได้นำ SAX(Simple API XML) มาใช้ในการวิเคราะห์เอกสารเอ็กซ์เอ็มไอเพื่อนำข้อมูลไปแสดงคลาสไดอะแกรม

Title	Mapping JAVA –UML Tool
Student	Miss Uahanong Polchana
Advisor	Asst. Prof. Dr. Ponrudee Netisopakul
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2005

ABSTRACT

Program understanding is essential for program maintainer. The modern object-oriented analysis and design techniques use UML(Unified Modeling Language) standard, which is a set of diagram. This project develops a tool that transforms Java code to UML(Class Diagram) and uses XMI(XML Metada Interchange) standard for transformation. Using XMI standard can interchange on other tools. Also JavaCC(Java Complier Complier) is applied to analyse Java code, which maps into XMI document. XMI document is created by DOM(Document Object Model). In order to represent class diagram, XMI document is parsed by SAX(Simple API XML).

กิตติกรรมประกาศ

โครงการนี้สำเร็จลุล่วงได้ด้วยการได้รับความช่วยเหลือและความกรุณาจากบุคคลต่าง ๆจึงขอขอบพระคุณบุคคลต่างๆ ดังนี้ บิดา มารดา ที่ให้โอกาสในการศึกษาเล่าเรียนอย่างเต็มที่ รวมทั้งคอยให้กำลังใจ ช่วยเหลือ และให้คำปรึกษาต่าง ๆ, ขอขอบพระคุณ ผศ.ดร.พรฤดี เนติโสภากุล อาจารย์ที่ปรึกษาเป็นอย่างมาก ที่ได้กรุณาให้คำปรึกษา คำแนะนำ และแก้ไขในสิ่งที่บกพร่องในการพัฒนาโครงการนี้ รวมทั้งอาจารย์ทุกท่านที่ให้ความรู้ต่าง ๆ เพื่อนำความรู้มาใช้ในการพัฒนาโครงการนี้, และขอขอบคุณเพื่อน ๆ ทุกคนที่ให้กำลังใจในการพัฒนาโครงการนี้

นางสาว เอื้ออนงค์ พลชนะ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญรูป

หน้า

รูปที่

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญรูป(ต่อ)

หน้า

รูปที่

สารบัญรูป(ต่อ)

หน้า

รูปที่

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหาและความสำคัญของโครงการพัฒนาระบบงาน

ลักษณะงานที่เป็นการบำรุงรักษาหรือการพัฒนาที่เป็นการทำความเข้าใจโค้ดที่เขียนด้วยภาษาใดๆก็ตามนั้นเป็นเรื่องยากและถ้ามีการขาดการจัดการลักษณะงานทางด้านเอกสารที่ดีแล้วยิ่งเป็นเรื่องที่ยากลำบากเป็นอย่างยิ่ง บ่อยครั้งที่การทำความเข้าใจโค้ดนั้นไม่ตรงกับสิ่งที่ได้ทำการออกแบบไว้ตั้งแต่ต้น เทคนิคที่เป็นการทำความเข้าใจในตัวโค้ดนั้นมักมีการแสดงออกเพื่อให้เกิดความเข้าใจด้วยรูปแบบเชิงภาพหรือโมเดล การทำการแปลงโค้ดสู่ลักษณะเชิงโมเดลเป็นเทคนิคที่เรียกว่า รีเวอร์ส เอ็นจีเนียริ่ง(Reverse Engineering)

เทคนิคทางด้านรีเวอร์ส เอ็นจีเนียริ่ง เป็นเทคนิคที่ผู้วิเคราะห์สามารถผลิตโมเดลที่ออกแบบได้ โดยสามารถถูกใช้เพื่อทำความเข้าใจหรือเปิดเผยให้เห็นถึงพฤติกรรมหรือสถาปัตยกรรมอีกทั้งยังสามารถเข้าใจภาพรวมทั้งหมดของโปรแกรมหรือระบบนั้นๆได้

สำหรับโมเดลที่นิยมใช้ในในเทคนิครีเวอร์สเอ็นจิเนียริ่ง(จากจาวาโค้ดไปเป็นโมเดล) คือ ยูเอ็มแอล(UML ย่อมาจาก Unified Modeling Language) ซึ่งเป็นภาษาเชิงโมเดลที่นิยมใช้ในการออกแบบและในทางกลับกันก็เป็นที่นิยมในการทำรีเวอร์ส เอ็นจีเนียริ่งด้วย ยูเอ็มแอลได้กลายมาเป็นแนวทางส่วนใหญ่ในระบบซอฟแวร์ซึ่งทำให้เกิดเครื่องมือที่รองรับยูเอ็มแอลอย่างมากมาย อีกทั้งเครื่องมือต่างๆเหล่านั้นยังมีความสามารถในการทำรีเวอร์สเอ็นจีเนียริ่งจากโค้ดไปสู่ยูเอ็มแอลได้ แต่ว่าผลลัพธ์ที่ได้นั้นมีลักษณะของการขึ้นอยู่กับเครื่องมือนั้น จากปัญหาเหล่านี้ทำให้โครงงานพัฒนาระบบงานนี้ได้สร้างเครื่องมือในการแมพโค้ดจาวาไปเป็นยูเอ็มแอล(คลาสไดอะแกรม) โดยที่มีการคำนึงถึงการขจัดปัญหาดังกล่าว

แนวทางหนึ่งที่นำมาแก้ไขปัญหาของผลลัพธ์ที่ได้จากเครื่องมือที่ทำรีเวอร์สเอ็นจีเนียริ่งที่มีการขาดความสามารถในการแลกเปลี่ยนระหว่างเครื่องมืออื่นๆนั้น ได้มีการนำมาตรฐานที่ใช้ในการแลกเปลี่ยนที่มีชื่อว่า เอ็กซ์เอ็มไอ(XMI ย่อมาจาก XML Metadata Interchange) มาเป็นตัวกลางในการแมพโค้ดจาวาเป็นยูเอ็มแอล

1.2 วัตถุประสงค์ของการพัฒนาระบบ

คือ การพัฒนาเครื่องมือในการแปลงภาษาจาวาเป็นไดอะแกรม ที่อิงกับมาตรฐานยูเอ็มแอล โดยทำการแปลงโค้ดภาษาจาวาไปเป็น คลาสไดอะแกรม(Class Diagram) โดยมีการใช้มาตรฐานเอ็กซ์เอ็มไอมาประยุกต์ใช้ในการแมพโค้ดจาวาไปเป็นยูเอ็มแอล(คลาสไดอะแกรม)

1.3 สมมุติฐานของการพัฒนาระบบ

- ซอร์สโค้ดที่ทำการทำความเข้าใจนั้นต้องเป็นซอร์สโค้ดจาวาที่ได้รับการตรวจสอบไวยากรณ์ต่างๆถูกต้องแล้ว

- ซอร์สโค้ดจาวาต้องมีรูปแบบการเขียนที่เป็นสากลทั่วไป

-ซอร์สโค้ดจาวานั้นถูกพัฒนาด้วย JDK 1.0 – 1.4

1.4 ขอบเขตการพัฒนาระบบ

ในการออกแบบและพัฒนาเครื่องมือในการแปลงโค้ดภาษาจาวาไปเป็นยูเอ็มแอล(คลาสไดอะแกรม) ได้มีการกำหนดขอบเขตของการศึกษาไว้ดังนี้

1) การแปลงโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม ได้ทำการศึกษารายละเอียดรูปแบบของเอกสารทางเอ็กซ์เอ็มไอที่ทำการแมพจากตัวโค้ดจาวาไปเป็นเอกสารเอ็กซ์เอ็มไอที่อิงกับโครงสร้างของยูเอ็มแอลเมต้าโมเดล

2) การออกแบบและพัฒนาระบบได้แบ่งการศึกษาออกเป็นสองส่วน โดยส่วนแรกเป็นการวิเคราะห์และออกแบบระบบในการแมพภาษาจาวาเป็นลักษณะของมาตรฐานกลางเอ็กซ์เอ็มไอ โดยมีการนำเจนเนอเรเตอร์(generator)ที่มีชื่อว่า JavaCC(Java Complier Complier) มาประยุกต์ใช้ในการวิเคราะห์โค้ดจาวาและทำการสร้างเอกสารเอ็กซ์เอ็มไอด้วยพาร์เซอร์(parser)ที่มีชื่อว่า DOM(Document Object Model) และส่วนที่สองเป็นส่วนของการวิเคราะห์และออกแบบในการแปลงตัวกลางเอ็กซ์เอ็มไอให้เป็นคลาสไดอะแกรม โดยมีการใช้พาร์เซอร์ที่มีชื่อว่า SAX (Simple API for XML)

3) การศึกษาการสร้างเอกสารเอ็กซ์เอ็มไอ เพื่อเป็นตัวกลางในการแลกเปลี่ยนระหว่างโค้ดจาวากับยูเอ็มแอล(คลาสไดอะแกรม) โดยผลลัพธ์ของเอกสารเอ็กซ์เอ็มไอที่ได้จากโครงงานนี้จะนำไปทดสอบบนเครื่องมืออื่นๆที่มีความสามารถในการแมพโค้ดจาวาไปสู่คลาสไดอะแกรม(โดยที่เครื่องมือนั้นรองรับมาตรฐานเอ็กซ์เอ็มไอ) นอกจากนี้ยังนำเอกสารเอ็กซ์เอ็มไอที่ถูกสร้างโดยเครื่องมืออื่น มาทดสอบการสร้างคลาสไดอะแกรมบนเครื่องมือของโครงงานนี้

1.5 ขั้นตอนการดำเนินการพัฒนาระบบ

- ศึกษารายละเอียดไดอะแกรมต่างๆ, ศึกษารายละเอียดของ JavaCC(Java Complier Complier), และศึกษารายละเอียดมาตรฐานต่างๆที่เกี่ยวข้อง(มาตรฐานเอ็กซ์เอ็มแอล,เอ็กซ์เอ็มไอและยูเอ็มแอล)
- ศึกษาเครื่องมืออื่นๆที่มีความสามารถในการแมพภาษาจาวาไปสู่คลาสไดอะแกรมโดยใช้มาตรฐานกลางเอ็กซ์เอ็มไอมาช่วยในการแลกเปลี่ยน
- ออกแบบตัวต้นแบบที่มีความสามารถในการแปลงโค้ดภาษาจาวาไปเป็นเอกสารทางเอ็กซ์เอ็มไอโดยใช้ JavaCC (Java Complier Complier) ช่วยวิเคราะห์โค้ดจาวาและมีการใช้พาร์เซอร์DOM สร้างเอกสารเอ็กซ์เอ็มไอ
- ออกแบบตัวต้นแบบในการแปลงเอกสารทางเอ็กซ์เอ็มไอไปสู่โครงสร้างคลาสไดอะแกรมโดยใช้พาร์เซอร์SAX วิเคราะห์เอกสารเอ็กซ์เอ็มไอ
- ออกแบบอินเตอร์เฟสในการติดต่อกับผู้ใช้ ทดสอบและประเมินผลที่ได้รับ และจัดทำเอกสารสรุปการทำโครงการ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

- ได้เข้าใจและประยุกต์ใช้เทคนิครีเวอร์สเอ็นจีเนียริ่ง
- เครื่องมือนี้สามารถอำนวยความสะดวกให้กับผู้พัฒนาระบบที่ต้องบำรุงรักษาหรือพัฒนาระบบที่มีการขาดจัดการเอกสารที่ดีได้

1.7 เครื่องมือที่ใช้

- NetBeans 5.0
- Rational Rose 2002
- Java2 SDK, Standard Edition Version 1.6
- NotePad++
- JavaCC(Java Complier Complier)
- DOM Parser
- SAX Parser

บทที่ 2

UML XMI Parser

2.1 ยูเอ็มแอล (UML – Unified Modeling Language) (OMG. 2003)

ยูเอ็มแอลจัดเป็นมาตรฐานหนึ่งที่เกี่ยวกับภาษาที่อธิบายด้วยโมเดล โดยทั่วไปคำว่า "ภาษา" จะถูกเข้าใจว่าเป็นลักษณะของตัวหนังสือ (Text) และการสื่อความหมายของตัวหนังสือของภาษานั้นจะมีการนำไวยากรณ์ (Syntax) มาใช้ ยูเอ็มแอลจัดว่าเป็นภาษาภาษาหนึ่งเช่นกัน(เพียงแต่ใช้รูปภาพสื่อความหมาย) ภาษาโดยทั่วไปนั้นมีโครงสร้างที่สำคัญ 2 ส่วน คือ คำศัพท์(Vocabulary) และ ไวยากรณ์

คำศัพท์ของยูเอ็มแอลนั้นอ้างอิงกับองค์ประกอบของยูเอ็มแอลซึ่งสามารถแบ่งได้เป็น 3 กลุ่ม ดังตารางที่ 2.1 โดยที่สัญลักษณ์ (Things) หมายถึง สิ่งที่ได้จากการจำลองเสมือน(abstract) ซึ่งสิ่งที่ได้จากการจำลองเสมือนก็จะถูกนำมาแทนด้วยสัญลักษณ์ และสัญลักษณ์จัดเป็นหน่วยที่เล็กที่สุดในยูเอ็มแอล, ความสัมพันธ์(Relationship) คือการเชื่อมโยงวัตถุเข้าด้วยกัน, และ ไดอะแกรม (Diagrams) คือการนำสัญลักษณ์และความสัมพันธ์มารวมกันแล้วนำมาแสดงเป็นไดอะแกรมตามมุมมองที่ต้องการ

ตารางที่ 2.1 แสดงองค์ประกอบของยูเอ็มแอล

สัญลักษณ์(Things)	ความสัมพันธ์(Relationships)	แผนภาพ(Diagrams)
1. Structural Things ได้แก่ Class, Interface, Collaboration, Use Case, Active Class, Node, Component 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 และ Activity Diagram

จากตารางที่ 2.1 ไดอะแกรมของยูเอ็มแอลนั้นถูกแบ่งตามลักษณะโดยในการจำแนกมุมมองเชิงโครงสร้างนั้นองค์ประกอบของไดอะแกรมประเภทนี้เป็นองค์ประกอบที่มีลักษณะเป็น สแตติค (Static) และมุมมองเชิงพฤติกรรม ที่จัดองค์ประกอบของไดอะแกรมประเภทนี้มีลักษณะเป็น ไดนามิค(Dynamic) ในการพิจารณารายละเอียดไดอะแกรมต่างๆในมุมมองของการทำรีเวอร์สเอ็นจีเนียริ่ง พบว่าบางไดอะแกรมของยูเอ็มแอลเป็นมุมมองระดับสูงที่ไม่สามารถพิจารณาจากแหล่งที่มาของตัวโค้ดได้(การพิจารณาโค้ดเป็นลักษณะของมุมมองระดับต่ำ) อย่างเช่น use case เป็นไดอะแกรมที่ใช้ในการวิเคราะห์ความต้องการของระบบ หรือบางไดอะแกรมไม่เป็นที่นิยมใช้ อย่างเช่นอ๊อบเจทไดอะแกรม(Object Diagram) เป็นไดอะแกรมที่แสดงความสัมพันธ์ระหว่างอินสแตนซ์(Instance) ที่เชื่อมโยงในช่วงเวลาหนึ่งเท่านั้น ประกอบกับ อ๊อบเจทไดอะแกรมและคลาสไดอะแกรมมีความคล้ายคลึงกัน ดังนั้นจึงมักนิยมใช้คลาสไดอะแกรมมากกว่า

ในโครงงานพัฒนาระบบงานนี้ได้ทำการเลือกใช้ตัวแทนไดอะแกรมที่เป็นตัวแทนในการทำความเข้าใจโค้ดจาวาไปเป็นยูเอ็มแอลด้วยคลาสไดอะแกรม สำหรับรายละเอียดของคลาสไดอะแกรมอยู่ในภาคผนวก ก

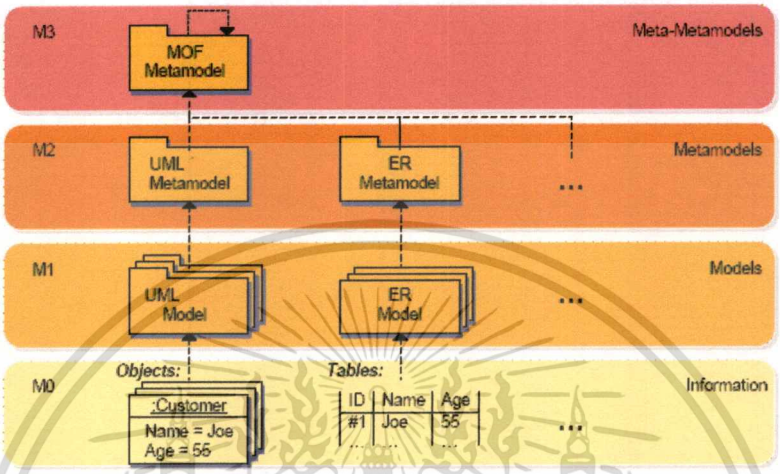
2.1.1 UML Metamodel

เนื่องจากยูเอ็มแอลเป็นภาษาที่ใช้รูปภาพในการสื่อความหมายจึงมักถูกเข้าใจว่า ยูเอ็มแอลเป็นเพียงสัญลักษณ์ที่ใช้เพียงแค่อธิบายถึงระบบงานเท่านั้น แต่จริงๆแล้วยูเอ็มแอลเป็นลักษณะของการใช้ตามกฎของเมต้าโมเดล(Metamodel) (สุทธิน. 2537.)

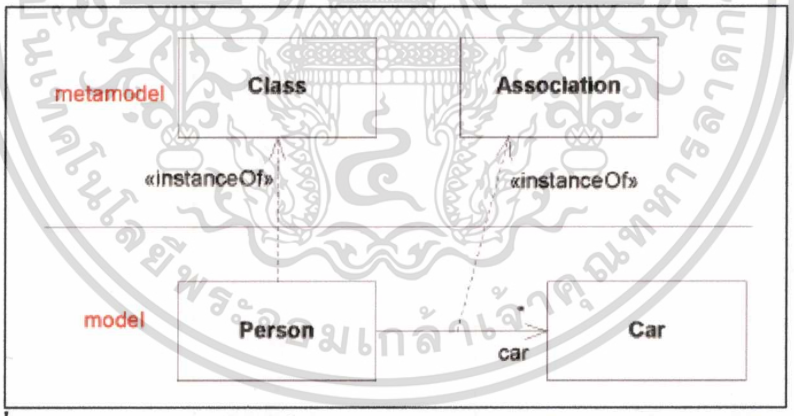
ยูเอ็มแอลเมต้าโมเดล เป็นลักษณะของมุมมองที่เป็นลักษณะระดับของรายละเอียด(Meta-Level) ซึ่งมีรายละเอียดดังรูปที่ 2.1 โดยที่ระดับที่ M0 เป็นระดับที่มีรายละเอียดน้อยที่สุด ตัวอย่างเช่นซอร์สโค้ด, ระดับที่ M1 เป็นระดับที่มีรายละเอียดมากกว่า M0 เป็นระดับที่เรียกว่า Metadata หรือ Model เช่น UML Model, ระดับที่ M2 เป็นระดับที่ละเอียดมากกว่า M1 เช่น UML Metamodel, และระดับที่ M3 เป็นระดับที่ละเอียดมากกว่า M2 เช่น MOF Model และระดับที่ M3 เป็นระดับที่เรียกว่า Meta-metamodel

จากรูปที่ 2.1 การใช้ยูเอ็มแอลทั่วไปเป็นโมเดลของระดับรายละเอียด M1 แต่ว่าการที่ใช้ไดอะแกรมต่างๆของยูเอ็มแอลนั้นเป็นไปตามกฎ(Rules)ต่างๆในยูเอ็มแอลเมต้าโมเดล(อยู่ในระดับ M2) ตัวอย่างเช่นในรูปที่ 2.2 เป็นการแสดงความสัมพันธ์ระหว่างระดับรายละเอียดของ M1(Model) และ M2(Metamodel) จะเห็นได้ว่าระดับความละเอียดแบบโมเดลมีการแสดงคลาส 2 คลาส(คือ คลาส Person และคลาส Car)ที่มีความสัมพันธ์กัน แบบ Association(มีการแสดงความสัมพันธ์ด้วยเส้นพร้อมหัวลูกศร) ระดับโมเดลเป็นการใช้ที่เป็นไปตามกฎของระดับความละเอียดแบบยูเอ็มแอล

เมต้าโมเดล ที่บ่งบอกว่าคลาส Car และ Person เป็นอ๊อบเจทของคลาสที่ชื่อว่า Class และความสัมพันธ์ที่เกิดขึ้นเป็นอ๊อบเจทของคลาส Association

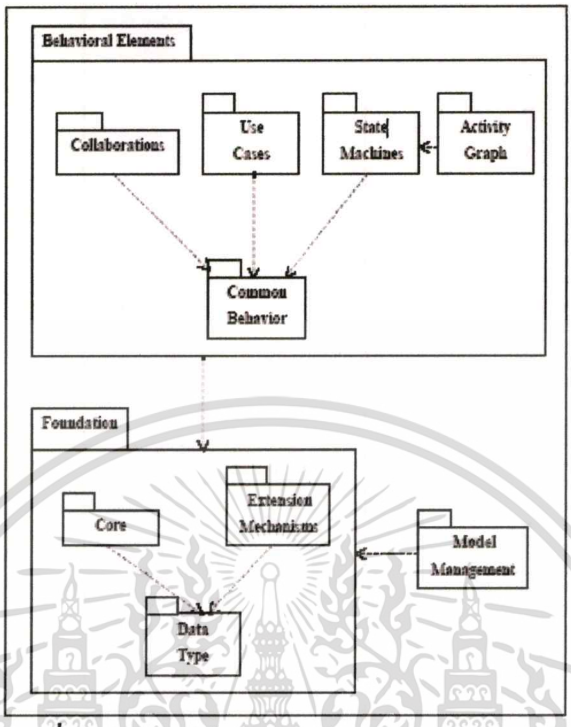


รูปที่ 2.1 แสดงระดับของรายละเอียด (Meta-Level)



รูปที่ 2.2 แสดงความสัมพันธ์ของระดับของรายละเอียดโมเดลและเมต้าโมเดล

ยูเอ็มแอลเมต้าโมเดลในระดับบนสุด(Top Level Package) ประกอบด้วยแพ็กเกจหลักๆ 3 แพ็กเกจ มีรายละเอียดดังรูปที่ 2.3 ซึ่งแสดงแพ็กเกจของยูเอ็มแอลเมต้าโมเดล ประกอบด้วย แพ็กเกจที่เป็นองค์ประกอบพื้นฐาน (Foundation Package) ที่ระบุโครงสร้างแบบสแตติคของยูเอ็มแอล, แพ็กเกจที่เก็บองค์ประกอบที่เกี่ยวข้องกับพฤติกรรม (Behavioral Elements Package) ที่ระบุโครงสร้างแบบไดนามิคของยูเอ็มแอล และ แพ็กเกจที่เก็บองค์ประกอบที่ใช้สำหรับจัดกลุ่ม (Model Management Package)(บุญประเสริฐ. 2546.)



รูปที่ 2.3 แสดงยูเอ็มแอลเมต้าโมเดลระดับบนสุด

- Foundation Package เป็นการรวบรวมโครงสร้างของไดอะแกรมที่มีโครงสร้างแบบสแตติกประกอบด้วย

  - Core Package เป็นส่วนที่แสดงรายละเอียดพื้นฐานของส่วนต่างๆ ซึ่งกฎสำหรับการตรวจสอบรูปแบบที่ถูกต้องของภาษายูเอ็มแอลถูกจำแนกตามรายละเอียดย่อยต่างๆ สิ่งเหล่านี้รวมถึง Classifier (เป็นโครงสร้างโมเดลที่อธิบาย Thing), เนื้อหา(Content) เช่น คุณสมบัติ(Attribute), และความสัมพันธ์(Relation) เช่น Association

  - DataType Package อธิบาย data type class ของยูเอ็มแอลเมต้าโมเดล

  - Extension Mechanisms Package อธิบายเทคนิค Constraint, สเตอริโอไทป์(Stereo Type), และค่าของแท็กต่างๆ เช่น ข้อจำกัด, เป็นต้น

- Behavioral Element Package ประกอบด้วย

  - Common Behavior Package อธิบาย ซิกแนล(signal), โอเปอร์เรชั่น(Operation), และแอคชั่น(Actor)

  - Collaborations Package อธิบายรายละเอียดต่างๆเช่น คอลแลบอเรชั่น(collaboration), แอสโซซิเอชั่น โรล(association), คลาสซิไฟล์เออร์ โรล(classifier role), อินเตอร์แอ็คชั่น(interaction), และเมสเสจ(message)เป็นต้น

- Use Case Package อธิบายตามรายละเอียดย่อยต่างๆ เช่น แอคเตอร์(actor)และยูสเคส (use case)

-State Machines Package อธิบายรายละเอียดย่อยต่างๆตามสเตตแมชชีน และทรานซิชั่น (transition) เป็นต้น

- Model Management Package อธิบาย แพ็กเกจ(package), โมเดล(model), และซับซิสเต็ม (subsystem)

2.1.2 UML Model Interchange

รายละเอียดของเมต้าโมเดลนั้น ทางองค์กรที่รับผิดชอบมาตรฐานยูเอ็มแอล(OMG) ได้มีการนำรายละเอียดของยูเอ็มแอลในเรื่องของเมต้าโมเดลมาประยุกต์ใช้ 2 มุมมอง คือ มุมมองที่เป็นในแง่ของความหมาย (UML Semantic) และมุมมองของการแลกเปลี่ยนโมเดล(UML Model Interchange) สำหรับเมต้าโมเดลของมุมมองความหมายนั้นเป็นลักษณะของเมต้าโมเดลที่กล่าวไว้ในหัวข้อ 2.1.1

เมต้าโมเดลของ UML Model Interchange และ UML Semantic มีความใกล้เคียงกัน โดยที่จะมีการเปลี่ยนแปลงหรือเพิ่มเติมบางส่วนของเมต้าโมเดลในมุมมองความหมาย อย่างเช่น ชื่อของแพ็กเกจหลักของเมต้าโมเดล UML Semantic ที่เป็นช่องว่างจะเปลี่ยนเป็นชื่อของแพ็กเกจสำหรับเมต้าโมเดล UML Interchange โดยการใช้ "_" แทนช่องว่าง ตัวอย่างเช่น แพ็กเกจModel Management เป็น แพ็กเกจ Model_Management สำหรับเมต้าโมเดลของ UML Model Interchange

UML Model Interchange ได้มีการนำมาตรฐานที่มีชื่อว่า เอ็กซ์เอ็มไอมาใช้ในการแลกเปลี่ยนโมเดลระหว่างเครื่องมือ โดยที่การแลกเปลี่ยนนั้นเป็นการแลกเปลี่ยนเอกสารเอ็กซ์เอ็มไอที่มีรูปแบบในการแมพอิงกับเมต้าโมเดล

2.2 XMI

XMI ย่อมาจาก XML Metadata Interchange เป็นมาตรฐานที่นำมาใช้ในการแลกเปลี่ยนรายละเอียดข้อมูลที่ตั้งอยู่บนพื้นฐานของมาตรฐานที่มีชื่อว่า XML (Extensible Markup Language) โดยได้ถือกำเนิดขึ้นเพื่อการแลกเปลี่ยนข้อมูลเชิงวัตถุ

2.2.1 เอ็กซ์เอ็มแอล(XML – Extensible Markup Language) (สุวัฒนา. 2545.)

เอ็กซ์เอ็มแอลเป็นมาตรฐานที่ใช้ในการแลกเปลี่ยนข้อมูลบนอินเตอร์เน็ตและมีองค์กรที่รับผิดชอบมาตรฐานนี้ คือ W3C (World Wide Web Consortium) วัตถุประสงค์ของมาตรฐานนี้คือ

เพื่อช่วยในการนิยามข้อมูล เพื่อการแลกเปลี่ยนข้อมูลผ่านอินเทอร์เน็ต ระหว่างแอพพลิเคชันหรืออุปกรณ์

ภาษาที่เป็นมาร์คอัพ(Markup) เป็นลักษณะของภาษาที่เป็นการอธิบายรายละเอียด การมาร์คอัพเป็นลักษณะการห่อหุ้มเอกสาร โดยมีวัตถุประสงค์ในการจัดรูปแบบ (Format) ของข้อความและให้คำอธิบาย คำนิยาม คำจำกัดความ กำหนดโครงสร้างในแต่ละส่วนของเอกสาร

เอ็กซ์เอ็มแอลประกอบด้วย 2 ส่วน คือ เอกสารเอ็กซ์เอ็มแอล(XML Document) และ XML DTD(Document Type Definition) หรือ XML Schema เอกสารเอ็กซ์เอ็มแอลจะบรรจุข้อมูลในรูปแบบของกลุ่มของแท็ก(Tag) ส่วน XML DTD/Schema เป็นการระบุกฎของแท็กต่างๆที่ถูกใช้ในเอกสารเอ็กซ์เอ็มแอล

2.2.2 เอ็กซ์เอ็มไอ (XMI –XML Matadata Language)

เอ็กซ์เอ็มไอถูกสร้างขึ้นโดยโอเอ็มจี(OMG - Object Management Group ซึ่งเป็นองค์เดียวกับผู้รับผิดชอบมาตรฐานยูเอ็มแอล) ซึ่งเป็นองค์กรระดับชาติที่ได้รับการสนับสนุนจากสมาชิกกว่า 600 กลุ่ม ซึ่งมีทั้ง Information System Vendor, นักพัฒนาซอฟแวร์ และกลุ่มผู้ใช้ทั่วไป โอเอ็มจีได้ก่อตั้งขึ้นในปี 1989 โดยมีแนวปฏิบัติ คือให้การสนับสนุนทฤษฎีและทำการทดสอบทาง Object Oriented Technology ในด้านการพัฒนาซอฟแวร์ โดยจุดประสงค์หลักของงานในองค์กรนี้ คือ การนำกลับมาใช้ใหม่ได้(reusable) , การเคลื่อนย้ายข้อมูลโปรแกรมจากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง (portable) และความสามารถในการติดต่อกับซอฟท์แวร์ที่มีโครงสร้างแบบอ็อบเจท ในลักษณะที่สามารถใช้งานได้แบบกระจายในสภาวะแวดล้อมที่แตกต่างกัน(พอเจตน์. 2003.)

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

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

เอ็มโอเอฟ(MOF-Meta Object Facility) เป็นมาตรฐานที่จัดแนวทางของการอธิบายเมต้าโมเดลโดยมีลักษณะที่สัมพันธ์กับเอ็กซ์เอ็มไอ คือ มาตรฐานนี้ทำการเลือกกลุ่มย่อย (subset)ของยูเอ็มแอลที่มีความเหมาะสมสำหรับเมต้าโมเดล

2.2.3 Architecture of XMI (Brodsky.1999.)

เนื่องจากเอ็กซ์เอ็มไอตั้งอยู่บนพื้นฐานของเอ็กซ์เอ็มแอลทำให้เอ็กซ์เอ็มไอมีส่วนประกอบอยู่ 2 ส่วนเช่นเดียวกับเอ็กซ์เอ็มแอล คือ ส่วนที่เป็นเอกสารทางเอ็กซ์เอ็มไอ(XMI Document) และส่วนที่เป็นตรวจสอบเอกสารทางเอ็กซ์เอ็มไอ(XMI DTD/Schema) โดยที่กฎการสร้างเอกสารทางเอ็กซ์เอ็มไอและXMI DTD มีอยู่ในข้อกำหนดของมาตรฐานเอ็กซ์เอ็มไอ(XMI Specification) ที่เป็นเวอร์ชั่นที่ขึ้นต้นด้วย 1.X(1.0, 1.1, และ1.2) ส่วนการสร้าง Schema มีในข้อกำหนดของมาตรฐานเอ็กซ์เอ็มไอที่เป็นเวอร์ชั่น 2.X(2.0 และ 2.1) สำหรับเอกสารทางเอ็กซ์เอ็มไอจัดว่ามีระดับรายละเอียดในระดับที่ M1

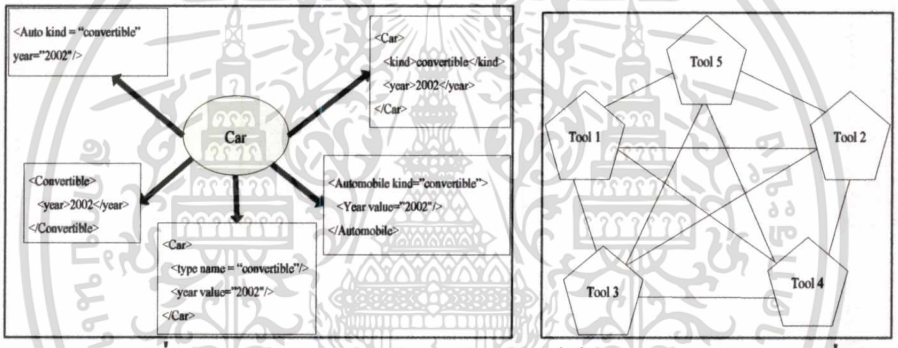
ในการสร้างเอกสารทางเอ็กซ์เอ็มไอมีรายละเอียดของอิลิเมนต์ที่สามารถแบ่งได้เป็น 4 ส่วน คือ 1) Header เป็นอิลิเมนต์ที่บรรจุการประกาศเวอร์ชั่นและเอกสารหรือรายละเอียดที่ใช้ในการแลกเปลี่ยน 2) Content เป็นอิลิเมนต์ที่บรรจุข้อมูลหลักที่ได้รับการแลกเปลี่ยน(เป็นเนื้อหาของโมเดล) 3) Differences เป็นอิลิเมนต์ที่ระบุความแตกต่างระหว่างเอกสารทางเอ็กซ์เอ็มไอของทั้งสองเอกสารที่จะนำมาใช้ในการแลกเปลี่ยน 4) Extensions เป็นอิลิเมนต์ที่สามารถแลกเปลี่ยนของข้อมูลที่เครื่องมือนั้นมีอยู่กับการแสดง DTD ที่มี

2.2.4 การเปรียบเทียบระหว่าง XMI กับ XML

ในการเชื่อมโยงของโลกปัจจุบันทำให้การติดต่อสื่อสาร การแลกเปลี่ยนข้อมูลต่างๆเกิดขึ้นได้อย่างไม่จำกัด การแทนข้อมูลในรูปแบบของตัวกลางตามมาตรฐานต่างๆเพื่อใช้ในการแลกเปลี่ยนจึงมีความสำคัญ มาตรฐานที่นำมาใช้เป็นตัวแทนแสดงข้อมูลเพื่อใช้ในการแลกเปลี่ยนนั้นมีอยู่ 2 มาตรฐาน คือ เอ็กซ์เอ็มแอลและเอ็กซ์เอ็มไอ

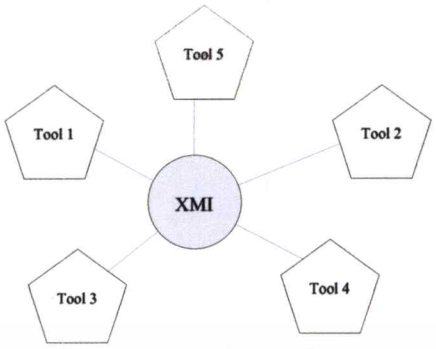
มาตรฐานเอ็กซ์เอ็มแอลเป็นมาตรฐานที่นิยมใช้ในการเป็นตัวกลางการแทนข้อมูลเพื่อการแลกเปลี่ยน เอ็กซ์เอ็มแอลมีจุดเด่นในการแสดงข้อมูลที่ไม่มีรูปแบบที่แน่นอน สามารถกำหนดลักษณะการนำเสนอข้อมูลที่ต้องการเองได้ จากข้อดีของเอ็กซ์เอ็มแอลที่ลักษณะของความยืดหยุ่นในการใช้งาน ทำให้เกิดปัญหาในการแลกเปลี่ยนข้อมูล ดังนั้นการแก้ปัญหาของความยืดหยุ่นในเอ็กซ์เอ็มแอลจะต้องมีการระบุข้อมูลที่จะถูกใช้ในการแลกเปลี่ยนและระบุถึงลักษณะการแทนข้อมูลในเอ็กซ์เอ็มแอล ดังตัวอย่างในรูป 2.4 เป็นการแสดงการแทนข้อมูลและการแลกเปลี่ยนเอ็กซ์เอ็มแอลระหว่างเครื่องมือต่างๆ ข้อมูลที่ใช้แสดงในรูปแบบเอ็กซ์เอ็มแอลเป็นข้อมูลของชนิดรถยนต์และปีที่ผลิตโดยการนำเสนอในรูปแบบของเอ็กซ์เอ็มแอล ซึ่งสามารถแทนข้อมูลดังกล่าวใน

รูปแบบของเอ็กซ์เอ็มแอลได้ 5 แบบ โดยในแต่ละแบบเป็นรูปแบบของเอ็กซ์เอ็มที่ได้จากแต่ละเครื่องมือทำให้ในการแลกเปลี่ยนระหว่างเครื่องมือทั้งห้านั้นจำเป็นต้องรู้ถึงรูปแบบการแทนข้อมูลของแต่ละเครื่องมือที่ทำการแลกเปลี่ยน ทำให้ต้องมีการเชื่อมโยง(bridge)ที่เป็นแบบ two-way bridges ทั้งหมด 10 จุด หรือ one-way bridges ทั้งหมด 20 จุด ทำให้การแลกเปลี่ยนข้อมูลของเอกสารเอ็กซ์เอ็มแอลเป็นเรื่องยาก แม้ว่าในเอ็กซ์เอ็มแอลมีกระบวนการการตรวจสอบ (XML Validation) ให้กับเอกสารเอ็กซ์เอ็มแอลเพื่อใช้ในการแลกเปลี่ยน โดยสามารถใช้ DTD/Schema ได้ แต่ว่าการสร้าง DTD/Schema ที่มีประสิทธิภาพนั้น ต้องมีการระบุลำดับเนื้อหาของ เอ็กซ์เอ็มแอลอิลิเมนต์ต่างๆซึ่งการระบุลำดับจะทำให้การสร้างเอกสารเอ็กซ์เอ็มแอลเป็นเรื่องที่ยากมากขึ้น(Grose. 2002.)



รูปที่ 2.4 แสดงการแทนข้อมูลด้วยเอ็กซ์เอ็มแอลและการแลกเปลี่ยน

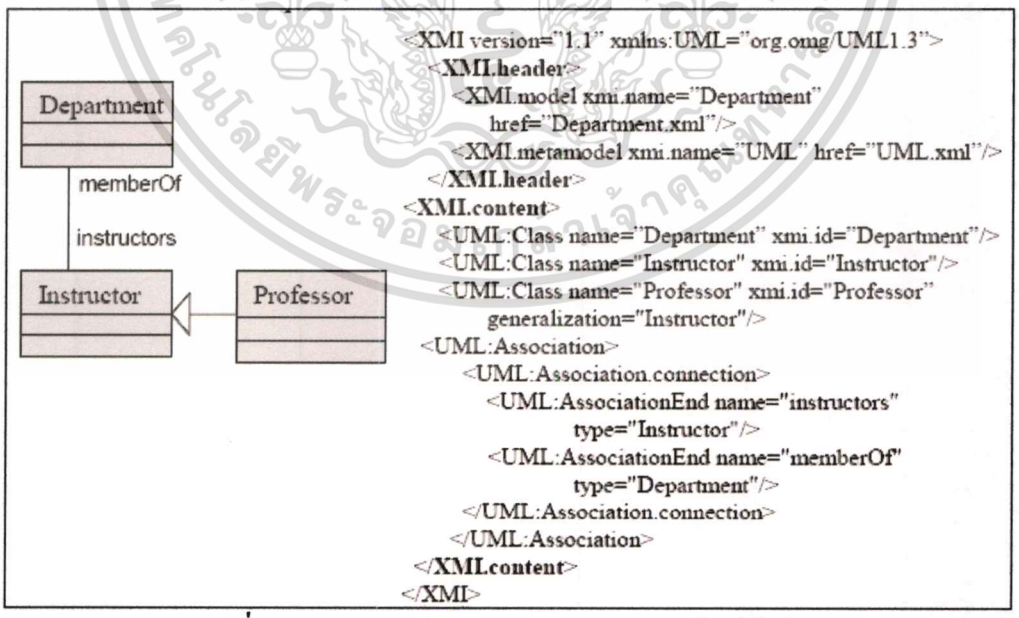
มาตรฐานเอ็กซ์เอ็มไอ เป็นตัวกลางที่นำประยุกต์ใช้ในโครงงานนี้ โดยเอ็กซ์เอ็มไอเป็นมาตรฐานที่ถูกนำมาเป็นตัวกลางของตัวแทนข้อมูลเชิงโมเดลเพื่อการแลกเปลี่ยน ซึ่งมาตรฐานนี้มีจุดเด่นที่เป็นมาตรฐานที่ตั้งอยู่บนพื้นฐานของเอ็กซ์เอ็มแอล เอ็กซ์เอ็มไอเป็นมาตรฐานในการแสดงผ่านตัวกลางที่นำมาใช้ในการแลกเปลี่ยนเชื่อมโยงโปรแกรมจากหลายๆผู้จำหน่าย(vendor)ที่ใช้งานอยู่เข้าด้วยกันได้ จากรูปที่ 2.5 แสดงการใช้เอ็กซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนทำให้ในการแลกเปลี่ยนระหว่างเครื่องมือทั้ง 5 ชนิดนี้มีการจุดเชื่อมโยงทั้งหมด 5 จุด



รูปที่ 2.5 แสดงการเชื่อมโยงของเครื่องมือที่มีการใช้เอ็กซ์เอ็มไอ

2.2.5 ตัวอย่างการนำ XMI มาประยุกต์ใช้กับ UML

เอ็กซ์เอ็มไอถูกนำมาใช้ในการแสดงรายละเอียดของโมเดล เพื่อความเป็นมาตรฐานในการแลกเปลี่ยนโมเดลระหว่างเครื่องมือต่างๆ ได้ โดยในการแลกเปลี่ยนระหว่างเครื่องมือที่มีความสามารถในการใช้ยูเอ็มแอล(สนับสนุนมาตรฐานการแลกเปลี่ยนเอ็กซ์เอ็มไอ) เป็นการนำเอกสารทางเอ็กซ์เอ็มไอที่รูปแบบการเขียนที่อิงรูปแบบของรายละเอียดเมต้าโมเดล ซึ่งในรูปที่ 2.6 เป็นการแสดงตัวอย่างที่ทำการเปรียบเทียบรูปแบบของเอกสารทางเอ็กซ์เอ็มไอกับรูปแบบของโมเดล



รูปที่ 2.6 แสดงตัวอย่างของการเอกสารทางเอ็กซ์เอ็มไอ

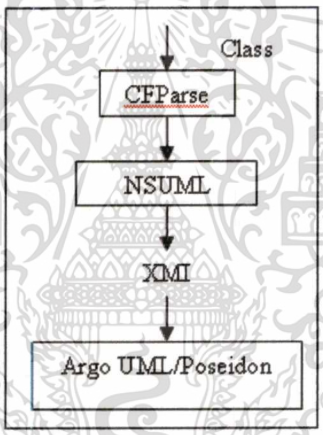
จากรูปที่ 2.6 ในส่วนของเอกสารทางเอ็กซ์เอ็มไอสามารถสังเกตเห็น prefix คำว่า 'XMI.' กับ 'xmi.' มีความหมายในการใช้ดังต่อไปนี้ คือเอ็กซ์เอ็มแอลอิลิเมนต์(XML element) ถูกระบุโดย

ข้อกำหนดของมาตรฐานเอ็กซ์เอ็มไอ โดยใช้ prefix 'XML.' ส่วนเอ็กซ์เอ็มไอแอตทริบิวต์(XML attributes) ถูกระบุด้วย prefix ว่า 'xmi.'

2.2.5 ตัวอย่างเครื่องมือที่นำ XMI มาประยุกต์

- Java2UML (Pechanec. 2000)

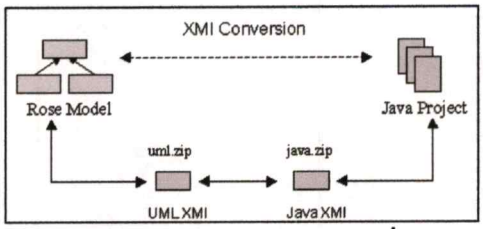
เครื่องมือนี้มีแนวคิดในการทำรีเวอร์สเอ็นจิเนียริ่ง ดังรูปที่ 2.7 โดยอินพุตของเครื่องมือนี้เป็นไฟล์คลาส(.class) มีพาร์เซอร์ทั้งหมด 2 อย่าง คือ CFParse เป็นพาร์เซอร์ที่ทำการวิเคราะห์ไฟล์คลาส ต่อจากนั้นเมื่อได้ผลลัพธ์จะทำการเรียกพาร์เซอร์ที่มีชื่อว่า NSUML(Novo Soft UML) ทำการวิเคราะห์ ให้ได้ผลลัพธ์เป็นเอกสารเอ็กซ์เอ็มไอ ต่อจากนั้นนำผลลัพธ์ที่ได้นี้ไปทดสอบบนเครื่องมืออื่นอย่างเช่น ArgoUML หรือ Poseidon for UML



รูปที่ 2.7 แสดงโครงสร้างของ Java2UML

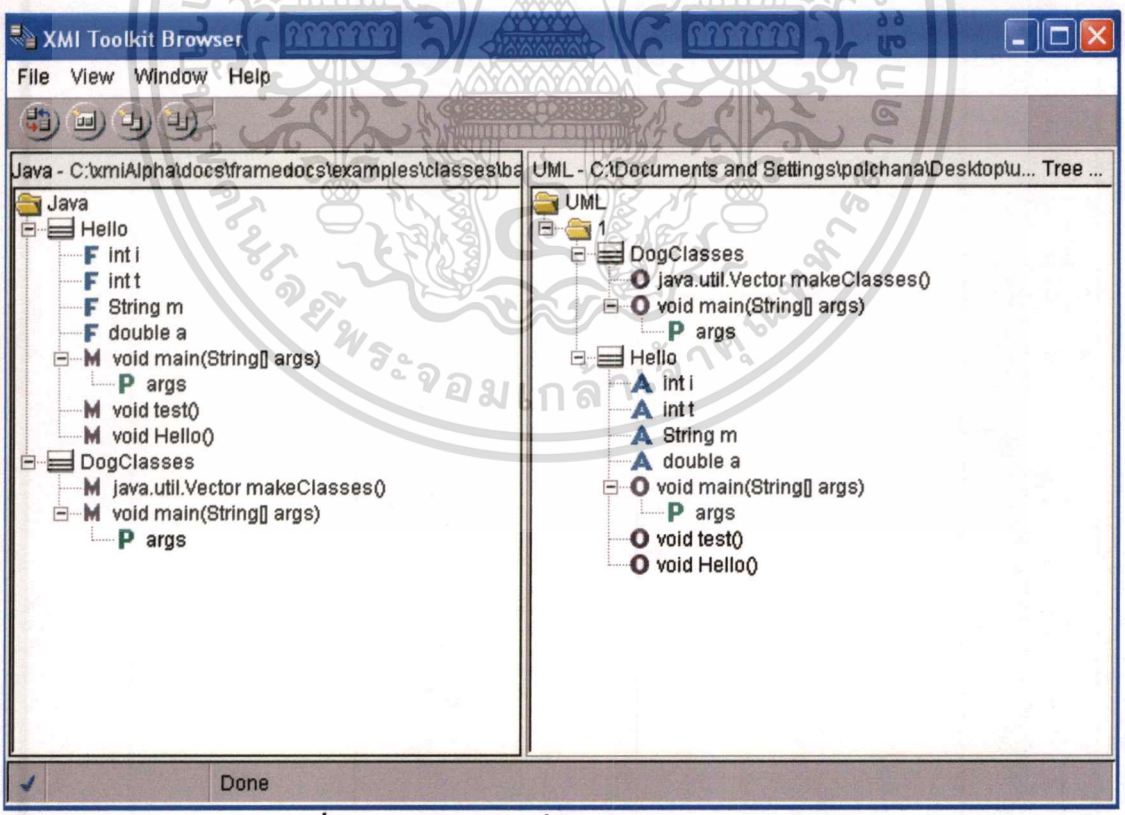
- XMI Toolkit (IBM. 1999)

เป็นเครื่องมือที่ถือกำเนิดขึ้นมาเพื่อมาตรฐานของเอ็กซ์เอ็มไอ เครื่องมือนี้นอกจากมีความสามารถในการทำรีเวอร์ส เอ็นจิเนียริ่งแล้วยังมีความสามารถในการแปลงจากลักษณะทางโมเดลไปเป็นโค้ดภาษาจาวา ในการแปลงของทั้ง 2 เทคนิค(จากโค้ดไปสู่โมเดลและจากโมเดลไปสู่โค้ด) นั้นเครื่องมือนี้ทำการแปลงเข้าสู่มาตรฐานกลางที่มีชื่อว่า เอ็กซ์เอ็มไอ ดังที่แสดงในรูปที่ 2.8 ที่แสดงลักษณะของความสามารถที่มีในเครื่องมือนี้



รูปที่ 2.8 แสดงลักษณะของความสามารถที่มี XMIToolkit

ตัวอย่างของผลลัพธ์ที่ได้จากการใช้เครื่องมือนี้ แสดงในรูปที่ 2.9 โดยเป็นการใช้ตัวอย่างที่เป็นโค้ดจาวา 2 ไฟล์คือ Hello.java และ DogClasses.java มาใช้ในการสร้างแบบจำลองโดยในที่นี้ให้ชื่อเป็น "1.mdl" ในการแสดงผลที่ได้จากการแปลงนั้นเห็นได้ว่ามีการแสดงผลในรูปแบบของทรี (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)



รูปที่ 2.8 แสดงผลลัพธ์ที่ได้จากการใช้ XMIToolkit

สำหรับการสร้างยูเอ็มแอล(คลาสไดอะแกรม) ของเครื่องมือนี้ ทำได้ 2 วิธี คือจากไฟล์ .mdl และไฟล์ที่เป็นเอกสารเอ็กซ์เอ็มไอ(.xml) ในการนำไฟล์ที่เป็นสกุล .mdl โดยทำการเปิดด้วยโปแกรม Rational Rose ดังตัวอย่างรูปที่ 2.9 ที่แสดงตัวอย่างของการเปิดไฟล์ "1.mdl" ซึ่งผลลัพธ์ที่ได้จะปรากฏอยู่ทางด้านซ้ายของเครื่องมือ Rational Rose (เป็นลักษณะโครงสร้างต้นไม้) ต่อจากนั้นก็สามารถลากโครงสร้างเหล่านั้นมาแสดงเป็นรูปคลาสไดอะแกรม(ทางขวาของ Rational Rose) สำหรับการเปิดด้วยไฟล์เอกสารเอ็กซ์เอ็มไอด้วยโปรแกรม Rational Rose นั้นต้องมีการ add-in โปรแกรมที่มีชื่อ Unisys XMI Tool(สามารถดาวน์โหลดได้ที่ www.rationalrose.com) ก่อน ต่อจากนั้นจึงสามารถทำการอิมพอร์ตไฟล์เอกสารเอ็กซ์เอ็มไอเพื่อแสดงคลาสไดอะแกรมได้

รูปที่ 2.9 แสดงผลลัพธ์ที่ได้จาก XMIToolkit โดยใช้ Rational Rose

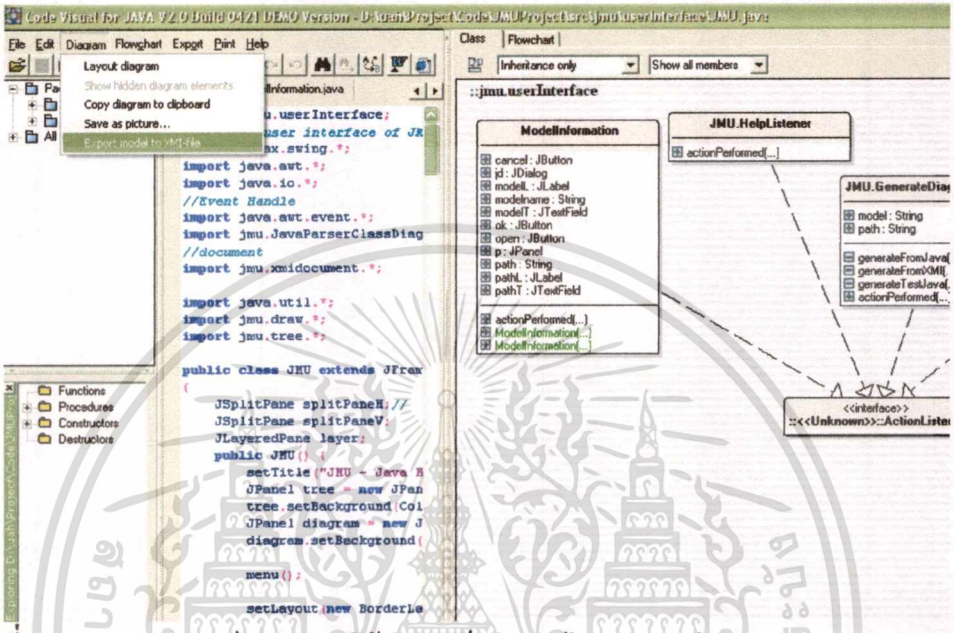
- Code Visual for Java 2.0 (Fate Software. 2005.)

Code Visual for Java 2.0 เป็น commercial software ที่ผลิตโดย Fate Software โปรแกรมมนี้มีความสามารถในการทำรีเวอร์สเอ็นจิเนียริ่งจากโค้ดจาวาสู่ยูเอ็มแอลคลาสไดอะแกรม(สามารถทำการ export คลาสไดอะแกรมในรูปแบบของเอกสารเอ็กซ์เอ็มไอได้) โปรแกรมนี้สามารถรับอินพุตจากไฟล์คลาส(.class)ของจาวาได้ ในรูปที่ 2.10 เป็นการแสดงรูปแบบของผลลัพธ์การใช้เครื่องมือในการทำรีเวอร์สเอ็นจิเนียริ่งจากโค้ดจาวาไปเป็นยูเอ็มแอลคลาสไดอะแกรม(โดยสามารถทำการ export คลาสไดอะแกรมเป็นเอกสารเอ็กซ์เอ็มไอ) นอกจากนี้ เครื่องมือนี้ยังสามารถสร้าง flowchart ได้

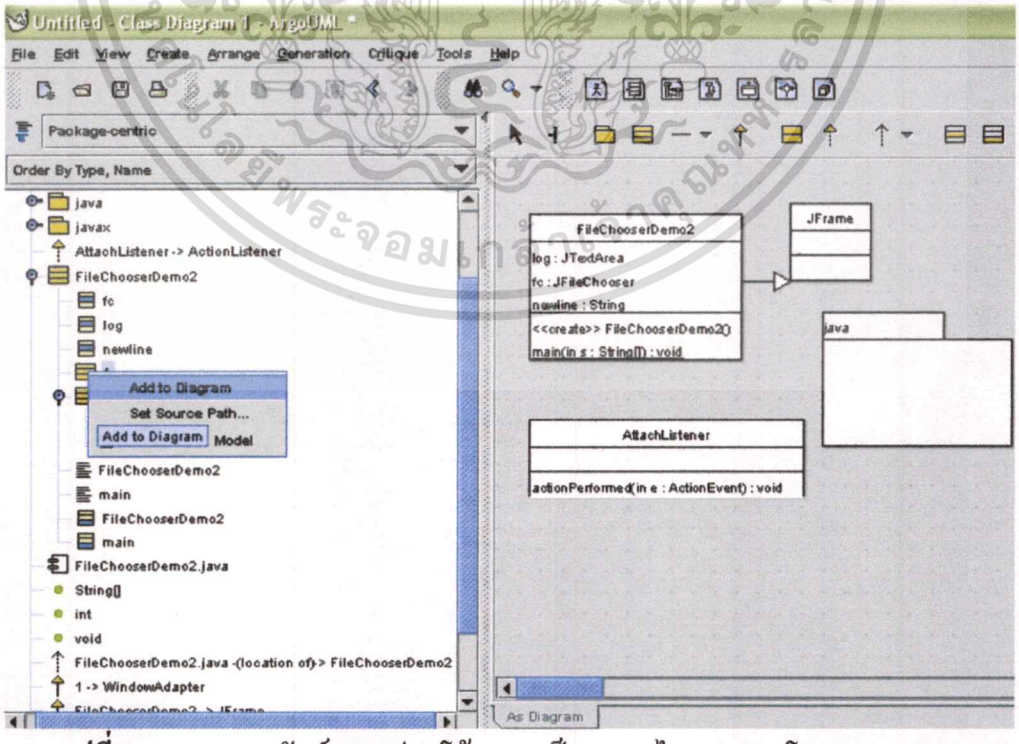
- ArgoUML (ArgoUML. 2006.)

ArgoUML เป็นฟรีซอฟแวร์ที่มีความสามารถในการสร้างยูเอ็มแอล และนอกจากนี้ยังมีความสามารถในการทำรีเวอร์สเอ็นจิเนียริ่งด้วย โดยรูปที่ 2.11 เป็นการแสดงผลลัพธ์รีเวอร์สเอ็นจิเนียริ่งจากโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม ซึ่งผลลัพธ์ที่ได้จะปรากฏในลักษณะของต้นไม้(ทางด้านซ้าย) ต่อจากนั้นทำการ add diagram ให้แสดงคลาสไดอะแกรม โปรแกรมนี้สามารถทำ

การ export คลาสไดอะแกรมที่ได้เป็นเอกสารเอ็กซ์เอ็มไอได้ นอกจากนี้ ArgoUML มีการใช้ GEF(Graph Editing Framework) ใช้ในการวาดไดอะแกรมต่างๆ



รูปที่ 2.10 แสดงผลลัพธ์การแปลงโค้ดจาวาเป็นคลาสไดอะแกรมโดย Code Visual for Java



รูปที่ 2.11 แสดงผลลัพธ์การแปลงโค้ดจาวาเป็นคลาสไดอะแกรมโดย ArgoUML

2.3 พาร์เซอร์(Parser)

ในการนำเอกสารเอ็กซ์เอ็มแอลและเอกสารทางเอ็กซ์เอ็มไอไปประยุกต์ใช้นั้นสามารถทำได้โดยการใช้พาร์เซอร์(Parser คือ อ๊อบเจทที่แสดงโครงสร้างของภาษาและแปลแต่ละโครงสร้างไปสู่ผลลัพธ์ที่มีประโยชน์)(Metsker. 2001)

2.3.1 XML Parser

2.3.1.1 DOM (Document Object Model)

DOM เป็นกระบวนการดึงข้อมูลในเอกสารเอ็กซ์เอ็มแอล มาใช้งานในแอพพลิเคชัน โดยมีหลักการคือ นำข้อมูลจากเอกสารมาเก็บเป็นโครงสร้างลักษณะลำดับชั้นต้นไม้ในหน่วยความจำของเครื่องคอมพิวเตอร์ แล้วใช้วิธีการท่องไปในต้นไม้เพื่อดึงข้อมูลมา ซึ่งวิธีการโหลดข้อมูลทั้งเอกสาร เอ็กซ์เอ็มแอลมาไว้ในหน่วยความจำนั้นทำให้เกิดเป็นข้อเสียอย่างหนึ่งของ DOM เพราะจะกลายเป็นข้อจำกัดในเรื่องของหน่วยความจำในกรณีที่เอกสารเอ็กซ์เอ็มแอล มีขนาดใหญ่มาก ๆ จะต้องเปลืองเนื้อที่ในหน่วยความจำมาก และอาจจะไม่มีเนื้อที่เหลือพอที่จะทำงาน

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

SAX ได้ถูกพัฒนาขึ้นมาเพื่อการวิเคราะห์เอกสารทางเอ็กซ์เอ็มแอลที่มีขนาดใหญ่มาก ให้มีประสิทธิภาพมากขึ้น เพราะว่าการโหลดเอกสารขนาดใหญ่มาไว้ในหน่วยความจำจะทำให้ต้องเสียเนื้อที่การทำงานมากเกินความจำเป็น จึงทำให้ SAX เกิดขึ้นมาเพื่อแก้ปัญหานี้

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

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

ในการวิเคราะห์เอกสารเอ็กซ์เอ็มไอ 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 อนุญาตให้ทำการสร้างและเก็บเอกสารเอ็กซ์เอ็มไอ ที่บรรจุวัตถุทางจาวา

2.3.3 พาร์เซอร์เจนเนอเรเตอร์(Parser Generator)

การสร้างพาร์เซอร์สำหรับภาษาใดภาษาหนึ่งนั้น สามารถใช้เจนเนอเรเตอร์(generator) ในการสร้างพาร์เซอร์ ซึ่งการใช้พาร์เซอร์เจนเนอเรเตอร์นั้นต้องเข้าใจความรู้พื้นฐานของ Programming Language เบื้องต้น ตัวอย่างของพาร์เซอร์เจนเนอเรเตอร์ เช่น ATLR(Another Tool for Language Recognition), YACC(Yet Another Compiler Compiler), และ JAVACC(Java Compiler Compiler) เป็นต้น

ในโครงงานนี้ได้นำพาร์เซอร์เจนเนอเรเตอร์ที่มีชื่อ JavaCC (ย่อมาจาก Java Compiler Compiler) มาประยุกต์ใช้เพื่อช่วยในการวิเคราะห์จาวาโค้ด JavaCC เป็นเครื่องมือที่มีความสามารถที่เป็นทั้ง Lexical Analyzer generator พาร์เซอร์เจนเนอเรเตอร์(เป็นโปรแกรมที่ยอมรับข้อกำหนดไวยากรณ์ในฐานะที่เป็นอินพุตและสร้างพาร์เซอร์สำหรับไวยากรณ์นั้นๆให้เป็นเอาท์พุต) นอกจากนี้ยังเป็นเครื่องมือที่เป็นฟรีแวร์ อีกทั้งยังสามารถดาวน์โหลดไฟล์ของ JavaCC(ที่เป็น .jj) ที่เป็นรูปแบบสำเร็จในการวิเคราะห์ภาษาต่างๆได้ อย่างเช่น XML, Java, และ HTML เป็นต้น

บทที่ 3

การวิเคราะห์และออกแบบระบบงาน

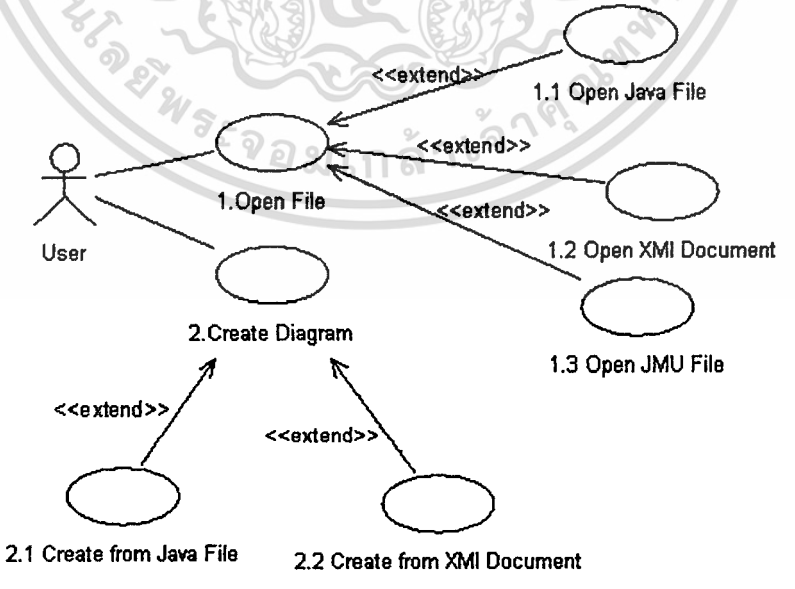
3.1 การวิเคราะห์และการออกแบบความต้องการของระบบ

ในการวิเคราะห์ความต้องการของเครื่องมือในการแปลงโค้ดจาวาเป็นคลาสไดอะแกรม สามารถกล่าวรายละเอียดได้ดังนี้

1) ระบบสามารถแสดงไฟล์ โดยแบ่งขั้นตอนในการแสดงไฟล์ออกเป็น 3 รูปแบบ คือ แสดงไฟล์จาวา, ไฟล์ .jmu และแสดงไฟล์เอกสารทางเอ็กซ์เอ็มไอ

2) ระบบสามารถสร้างคลาสไดอะแกรมได้จากไฟล์ 2 ชนิด คือ

- สร้างคลาสไดอะแกรมจากไฟล์จาวา โดยเครื่องมือจะทำการสร้างเอกสารเอ็กซ์เอ็มไออย่างอัตโนมัติ
- สร้างคลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอที่สร้างโดยเครื่องมืออื่นๆ(รวมทั้งเอกสารเอ็กซ์เอ็มไอที่ถูกสร้างโดยเครื่องมือนี้) ในโครงงานนี้ได้เลือกเครื่องมือที่มีชื่อว่า Rational Rose เป็นโปรแกรมที่สร้างเอกสารเอ็กซ์เอ็มไอ



รูปที่ 3.1 การออกแบบความต้องการของเครื่องมือแปลงโปรแกรมภาษาจาวาเป็นยูเอ็มแอล

ในการออกแบบ JMU (Java Map to UML – Class Diagram) สามารถสร้างเป็นยูสเคสไดอะแกรมดังรูปที่ 3.1 ซึ่งแสดงการออกแบบกระบวนการทำงานตามการวิเคราะห์ความต้องการของเครื่องมือในการแปลงโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม โดยมีการอธิบายรายละเอียดของการทำงานด้วย Use Case Description มีรายละเอียดดังต่อไปนี้

ตารางที่ 3.1 แสดงคำอธิบายยูสเคสที่ 1

Use Case	**1. Open File**
Brief Description	เปิดไฟล์จาวา, ไฟล์เอกสารเอ็กซ์เอ็มไอ และไฟล์ ".jmu
Actor	User
Pre-condition	
Post-condition	ระบบแสดงรายละเอียดของไฟล์จาวา, เอกสารทางเอ็กซ์เอ็มไอ, หรือไฟล์ ".jmu"
Primary scenario	1. ผู้ใช้เลือกเมนู Open 2. ระบบแสดงทางเลือกของการแสดงรายละเอียดไฟล์จาวา, เอกสารทางเอ็กซ์เอ็มไอ, และไฟล์ ".jmu" ตามลำดับ 3. ระบบแสดงหน้าต่างเลือกไฟล์และไดเร็คทอรี่ 4. ผู้ใช้เลือกไฟล์ 5. ระบบแสดงรายละเอียดของไฟล์ถูกเลือก

ตารางที่ 3.2 แสดงคำอธิบายยูสเคสที่ 2

Use Case	**2. Create Diagram**
Brief Description	สร้างไดอะแกรมจากไฟล์จาวาและเอกสารทางเอ็กซ์เอ็มไอ
Actor	User
Pre-condition	
Post-condition	ระบบแสดงคลาสไดอะแกรม
Primary scenario	1. ผู้ใช้เลือกเมนู Create Diagram 2. ระบบแสดงทางเลือกของชนิดไฟล์ที่ต้องการแปลงไปเป็นคลาสไดอะแกรม 3. ระบบแสดงหน้าต่างไฟล์และไดเร็คทอรี่ 4. ผู้ใช้เลือกไฟล์ 5. ระบบแสดงคลาสไดอะแกรม

ตารางที่ 3.3 แสดงคำอธิบายยูสเคสที่ 3

Use Case	**1.1 Open Java File**
Brief Description	แสดงไฟล์จาวา
Actor	User
Pre-condition	ผู้ใช้เลือกไฟล์จาวา
Post-condition	ระบบแสดงรายละเอียดของไฟล์จาวา
Primary scenario	1. ผู้ใช้เลือกเมนู Open 2. ผู้ใช้เลือก Open Java File 3. ระบบแสดงหน้าต่างไฟล์และไดเร็คทอรี่ 4.ผู้ใช้เลือกไฟล์จาวา 5. ระบบตรวจสอบว่าเป็นชนิดไฟล์ไฟล์จาวา 6. ระบบแสดงไฟล์จาวา

ตารางที่ 3.4 แสดงคำอธิบายยูสเคสที่ 4

Use Case	**1.2 Open XMI Document**
Brief Description	แสดงเอกสารทางเอ็กซ์เอ็มไอ
Actor	User
Pre-condition	
Post-condition	ระบบแสดงรายละเอียดของไฟล์เอกสารทางเอ็กซ์เอ็มไอ
Primary scenario	1. ผู้ใช้เลือกเมนู Open 2. ผู้ใช้เลือก Open XMI Document 3. ระบบแสดงไฟล์และไดเร็คทอรี่ 4. ผู้ใช้เลือกไฟล์เอ็กซ์เอ็มไอ 5. ระบบตรวจสอบชนิดของไฟล์ 6. ระบบแสดงไฟล์เอกสารทางเอ็กซ์เอ็มไอ

ตารางที่ 3.5 แสดงคำอธิบายยูสเคสที่ 5

Use Case	**1.3 Open JMU File**
Brief Description	แสดงไฟล์ JMU
Actor	User
Pre-condition	
Post-condition	ระบบแสดงรายละเอียดของไฟล์ ".jmu"
Primary scenario	1. ผู้ใช้เลือกเมนู Open 2. ผู้ใช้เลือกOpen JMU File 3. ระบบแสดงไฟล์และไดเร็คทอรี่ 4. ผู้ใช้เลือกไฟล์ ".jmu" 5. ระบบตรวจสอบชนิดของไฟล์ 6. ระบบแสดงไฟล์ JMU

ตารางที่ 3.6 แสดงคำอธิบายยูสเคสที่ 6

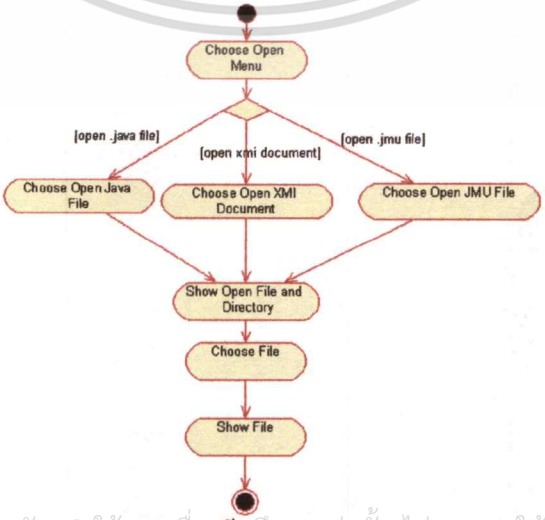
Use Case	**2.1 Create from Java File**
Brief Description	สร้างไดอะแกรมจากไฟล์จาวา
Actor	User
Pre-condition	
Post-condition	ระบบแสดงคลาสไดอะแกรมจากไฟล์จาวา
Primary scenario	1. ผู้ใช้เลือกเมนู Create Diagram 2. ผู้ใช้เลือกCreate from Java File 3. ระบบแสดงไฟล์และไดเร็คทอรี่ 4. ผู้ใช้เลือกไฟล์หรือกลุ่มไฟล์จาวา 5. ระบบตรวจสอบชนิดของไฟล์ 6. ระบบแสดงไดอะล็อคบ๊อกซ์ข้อมูลของโมเดล 7. ผู้ใช้เลือกไดเร็คทอรี่ที่ต้องการเก็บไฟล์เอกสารทางเอ็กซ์เอ็มไอและไฟล์ ".jmu" 8. ผู้ใช้ตั้งชื่อไฟล์ 9. ระบบทำการตรวจสอบข้อมูลไดเร็คทอรี่และไฟล์ 10. ระบบทำการสร้างไฟล์เอกสารทางเอ็กซ์เอ็มไอ 11. ระบบทำการสร้างไฟล์ ".jmu" 12. ระบบแสดงเอกสารทางเอ็กซ์เอ็มไอ 13. ระบบแสดงคลาสไดอะแกรม

ตารางที่ 3.7 แสดงคำอธิบายยูสเคสที่ 7

Use Case	**2.2 Create from XMI Document**
Brief Description	สร้างไดอะแกรมจากเอกสารทางเอ็กซ์เอ็มไอ
Actor	User
Pre-condition	
Post-condition	ระบบแสดงคลาสไดอะแกรมจากเอกสารทางเอ็กซ์เอ็มไอ
Primary scenario	1. ผู้ใช้เลือกเมนู Create Diagram 2. ผู้ใช้เลือก Create from XMI File 3. ระบบแสดงไฟล์ไดเร็คทอรี่ 4. ระบบตรวจสอบชนิดไฟล์ 5. ผู้ใช้เลือกไฟล์เอ็กซ์เอ็มไอ 6. ระบบแสดงคลาสไดอะแกรม

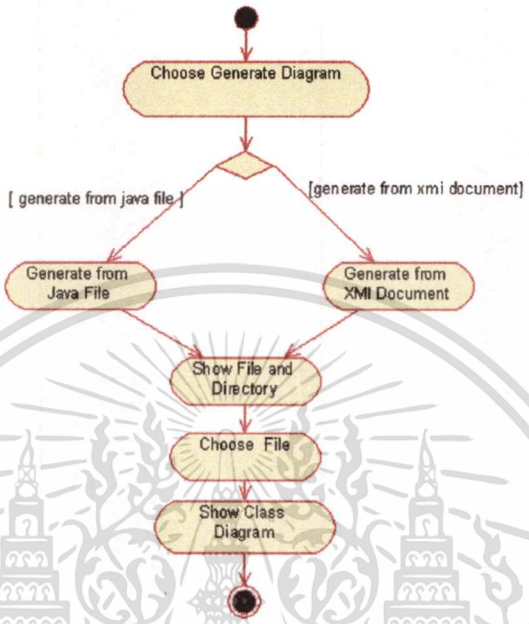
ขั้นตอนการทำงานของเครื่องมือแปลงโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรมสามารถแสดงขั้นตอนการทำงานได้ดังแอ็คทิวิตี้ไดอะแกรม ดังต่อไปนี้

- แอ็คทิวิตี้ไดอะแกรมของยูสเคส 1. Open File แสดงรายละเอียดดังรูปที่ 3.2



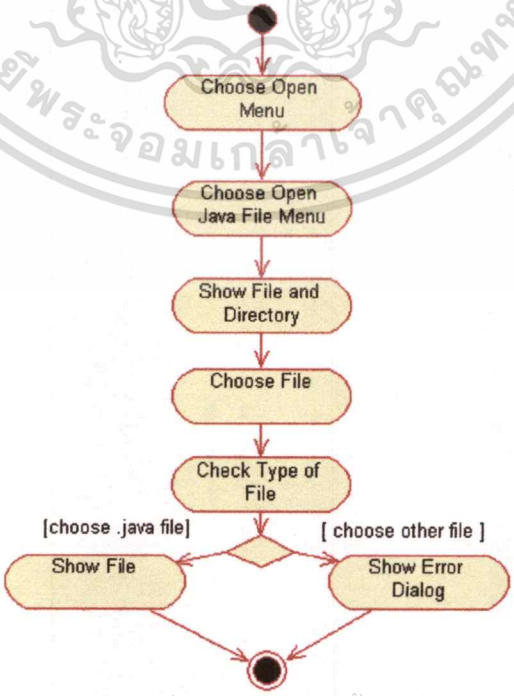
รูปที่ 3.2 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 1

-แอ็คทิวิตี้ไดอะแกรมของยูสเคส 2. Create Diagram แสดงรายละเอียดดังรูปที่ 3.3



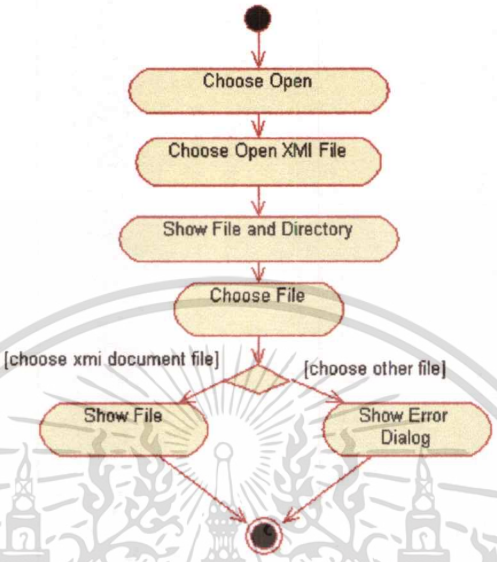
รูปที่ 3.3 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 2

-แอ็คทิวิตี้ไดอะแกรมของยูสเคส 1.1 Open Java File แสดงรายละเอียดดังรูปที่ 3.4



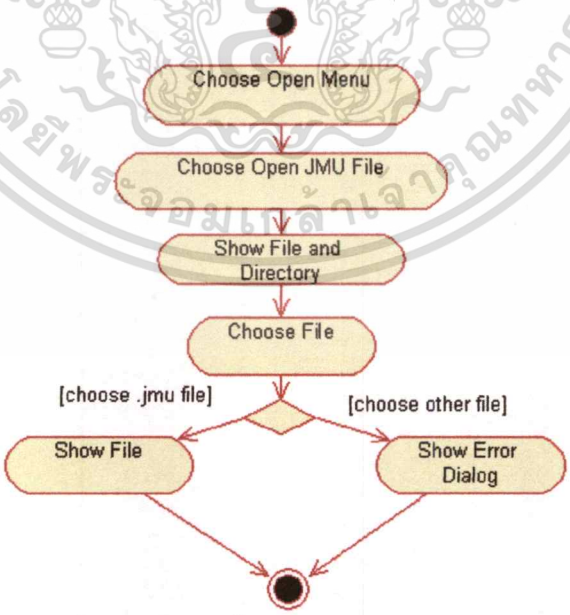
รูปที่ 3.4 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 1.1

-แอ็คทิวิตี้ไดอะแกรมของยูสเคส 1.2 Open XMI Document แสดงรายละเอียดดังรูปที่ 3.5



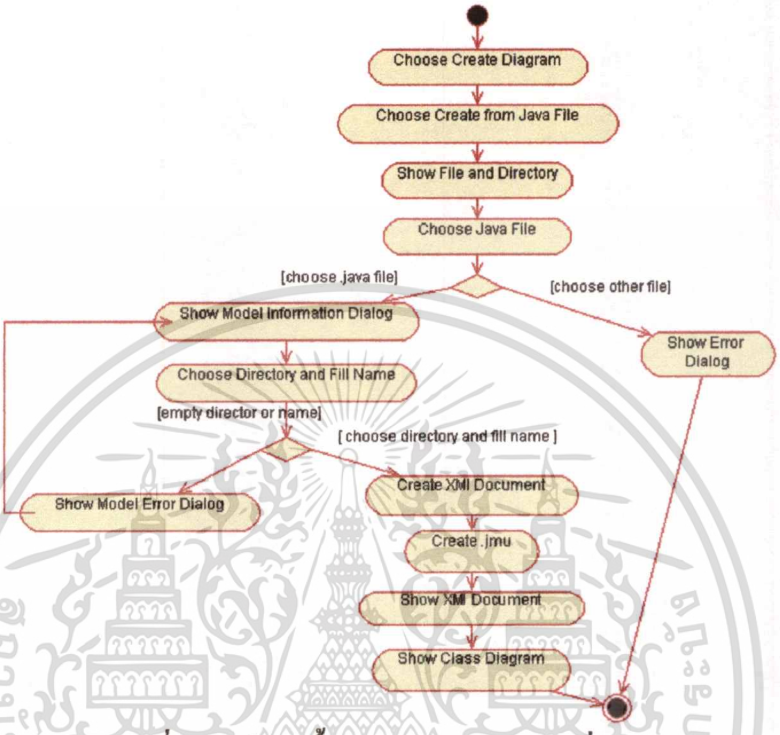
รูปที่ 3.5 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 1.2

-แอ็คทิวิตี้ไดอะแกรมของยูสเคส 1.3 Open JMU File แสดงรายละเอียดดังรูปที่ 3.6



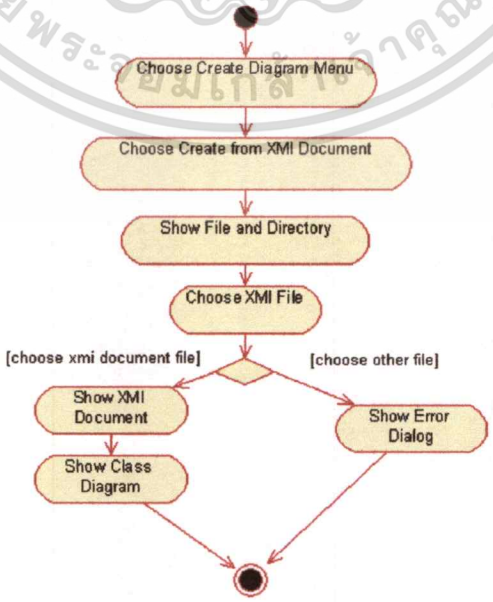
รูปที่ 3.6 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 1.3

-แอ็คทิวิตี้ไดอะแกรมของยูสเคส 2.1 Create from Java File แสดงรายละเอียดดังรูปที่ 3.7



รูปที่ 3.7 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 2.1

- แอ็คทิวิตี้ไดอะแกรมของยูสเคส 2.2 Create from XMI Document ดังรูปที่ 3.8



รูปที่ 3.8 แอ็คทิวิตี้ไดอะแกรมของยูสเคสที่ 2.2

3.2 เครื่องมือและขั้นตอนในการแมพโค้ดจาวาเป็นคลาสไดอะแกรม

3.2.1 เครื่องมือที่นำมาใช้ในการแมพโค้ดจาวาเป็นคลาสไดอะแกรม

1) Notepad++

-เป็นเท็กซ์อิดิเตอร์

-นำมาใช้ในการสร้างและประยุกต์ใช้ไฟล์ .jj(เป็นไฟล์ที่จะสร้างพาร์เซอร์สำเร็จการวิเคราะห์โค้ดจาวา)

2) JavaCC(Java Complier Complier)

-เป็นพาร์เซอร์เจนเนอเรเตอร์ที่นำมาใช้สร้าง JavaParser

-สำหรับการวิเคราะห์ข้อมูลเพื่อสร้างคลาสไดอะแกรมของโครงงานพัฒนาระบบนี้ ได้มีการนำไฟล์ที่มีชื่อว่า java1.4.jj มาประยุกต์เพื่อสร้าง JavaParser

-การใช้งานนั้นจะต้องมีไฟล์ .jj โดยในที่นี้ได้ประยุกต์ใช้ java1.4.jj(ในส่วนของ Lexical Specification จะไม่ทำการปรับปรุงใดๆ) ต่อจากนั้น ทำการ run command promt และพิมพ์คำสั่ง "javacc ชื่อไฟล์.jj"

-เมื่อไฟล์ได้ผ่านคำสั่ง "javacc" และไฟล์ .jj ไม่มีข้อผิดพลาดหรือคำเตือนใดปรากฏ จะมีการสร้างไฟล์ต่างๆที่ถูกสร้างโดยคำสั่งนี้ 7 ไฟล์ สำหรับรายละเอียดการใช้งาน JavaCC จะกล่าวในภาคผนวก ข

3) jdk 1.6 สามารถทำการดาวน์โหลดได้ฟรี

4) DOM (Document Object Model)

-เป็นเอ็กซ์เอ็มแอลพาร์เซอร์

-นำมาประยุกต์ใช้ในการสร้างไฟล์เอกสารเอ็กซ์เอ็มไอ โดยได้เลือก DOM ที่เป็นของ Xerces

-วิธีการใช้ต้องทำการนำเข้า(ใช้ import statement)

5) SAX (Simple API XML)

-เป็นเอ็กซ์เอ็มแอลพาร์เซอร์

-นำมาประยุกต์ใช้ในการวิเคราะห์เอกสารเอ็กซ์เอ็มไอ โดยได้ทำการเลือกใช้ SAX ที่อยู่ในจาวา

-วิธีการใช้ต้องทำการนำเข้า(ใช้ import statement)

6) NetBean 5.0 เป็นเครื่องมือหลักในการเขียนโค้ดที่ใช้ในการพัฒนาโครงงานนี้

7) Rational Rose 2002

-เป็นเครื่องมือที่มีความสามารถในการรองรับมาตรฐานยูเอ็มแอล

-เป็นเครื่องมือที่นำมาใช้ในการทดสอบการแลกเปลี่ยนเครื่องมือในโครงงานพัฒนาระบบนี้กับเครื่องมือนี้

3.2.2 ขั้นตอนในการแมพโค้ดจาวาเป็นคลาสไดอะแกรม

เครื่องมือที่สร้างในโครงการพัฒนาระบบได้ถูกตั้งชื่อว่า JMU ซึ่งย่อมาจากคำว่า Java Map to UML(Class Diagram) เครื่องมือนี้มีขั้นตอนการทำงานดังรูปที่ 3.9 ที่เป็นการแสดงขั้นตอนในการแมพโค้ดจาวาเป็นคลาสไดอะแกรม โดยสามารถแบ่งการทำงานของเครื่องมือออกเป็น 2 ส่วนหลักๆ ดังนี้

1) ส่วนที่เป็นการแปลงข้อมูลจากโค้ดจาวาเป็นเอกสารเอ็กซ์เอ็มไอ มีขั้นตอนดังนี้

-รับข้อมูลเป็นไฟล์หรือกลุ่มไฟล์จาวา

-นำไฟล์จาวาที่ได้รับไปวิเคราะห์ด้วยการใช้ JavaParser ถูกสร้างขึ้นจากไฟล์ java1.4.jj และใช้พาร์เซอร์เจนเนอเรเตอร์ที่มีชื่อว่า JavaCC เป็นตัวสร้าง JavaParser

-ข้อมูลที่ผ่าน JavaParser สามารถแบ่งการวิเคราะห์เป็นส่วนต่างๆของคลาสไดอะแกรม เช่น แอตทริบิวต์ เมื่อ JavaParser ทำการวิเคราะห์ว่าข้อมูลเป็นรูปแบบของแอตทริบิวต์จะนำข้อมูลดังกล่าวไปสร้างแอตทริบิวต์อิลิเมนต์และทำการหาความสัมพันธ์

-ในขั้นการวิเคราะห์ข้อมูลของโค้ดจาวานี้ นอกจากจะมีการวิเคราะห์เป็นส่วนต่างๆของคลาสไดอะแกรมยังทำการวิเคราะห์หาความสัมพันธ์ของโค้ดจาวาด้วย อย่างเช่น ส่วนประกอบของคลาสไดอะแกรมที่เป็นส่วนของแอตทริบิวต์ เมื่อทำการวิเคราะห์ข้อมูลในส่วนนี้จะทำการวิเคราะห์หาความสัมพันธ์ด้วย โดยดูจากชนิดของแอตทริบิวต์ ถ้าชนิดของแอตทริบิวต์เป็นแบบ Reference Type จะแสดงให้เห็นถึงความสัมพันธ์แบบ Association

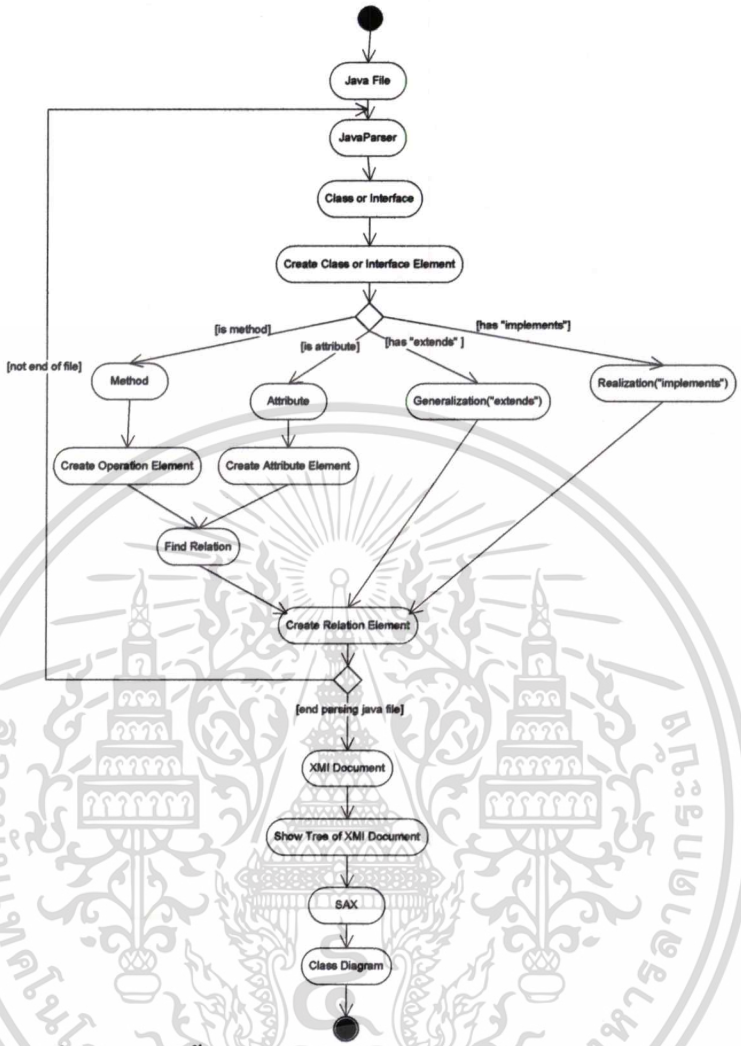
-การสร้างอิลิเมนต์ต่างๆในรูปแบบของเอกสารเอ็กซ์เอ็มไอได้นำ DOM มาประยุกต์ใช้ในการสร้างเอกสารเอ็กซ์เอ็มไอ

2) นำเอกสารเอ็กซ์เอ็มไอแปลงเป็นคลาสไดอะแกรม มีขั้นตอนดังนี้

- นำเอกสารเอ็กซ์เอ็มไอมาแสดงด้วยรูปแบบทรี(โดยใช้JTree)

- นำเอกสารเอ็กซ์เอ็มไอมาสร้างคลาสไดอะแกรม โดยการวิเคราะห์เอกสารเอ็กซ์เอ็มไอได้ทำการใช้ SAX ประยุกต์ในการวิเคราะห์ข้อมูลเพื่อสร้างคลาสไดอะแกรม

นอกจากนี้ ในโครงงานนี้จะนำเอกสารเอ็กซ์เอ็มไอที่ได้นั้น ไปทดสอบบนเครื่องมือที่มีชื่อว่า Rational Rose โดยจะนำเอกสารเอ็กซ์เอ็มไอที่ได้จากโครงงานนี้ นำไปวาดคลาสไดอะแกรมบน Rational Rose ได้



รูปที่ 3.9 แสดงขั้นตอนการแมพจาวาโค้ดเป็นคลาสไดอะแกรม

3.3 การออกแบบคลาสไดอะแกรม JMU(Java Map to UML – Class Diagram)

การออกแบบเครื่องมือในการแมพโค้ดภาษาจาวาไปเป็นยูเอ็มแอลคลาสไดอะแกรมโดยนำมาตรฐานเอ็กซ์เอ็มไอมาประยุกต์ใช้ ได้ทำการออกแบบคลาสไดอะแกรมของเครื่องมือนี้ไว้ดังรูปที่ 3.8 แสดงรายละเอียดคลาสไดอะแกรมของเครื่องมือ JMU โดยมีรายละเอียดดังต่อไปนี้

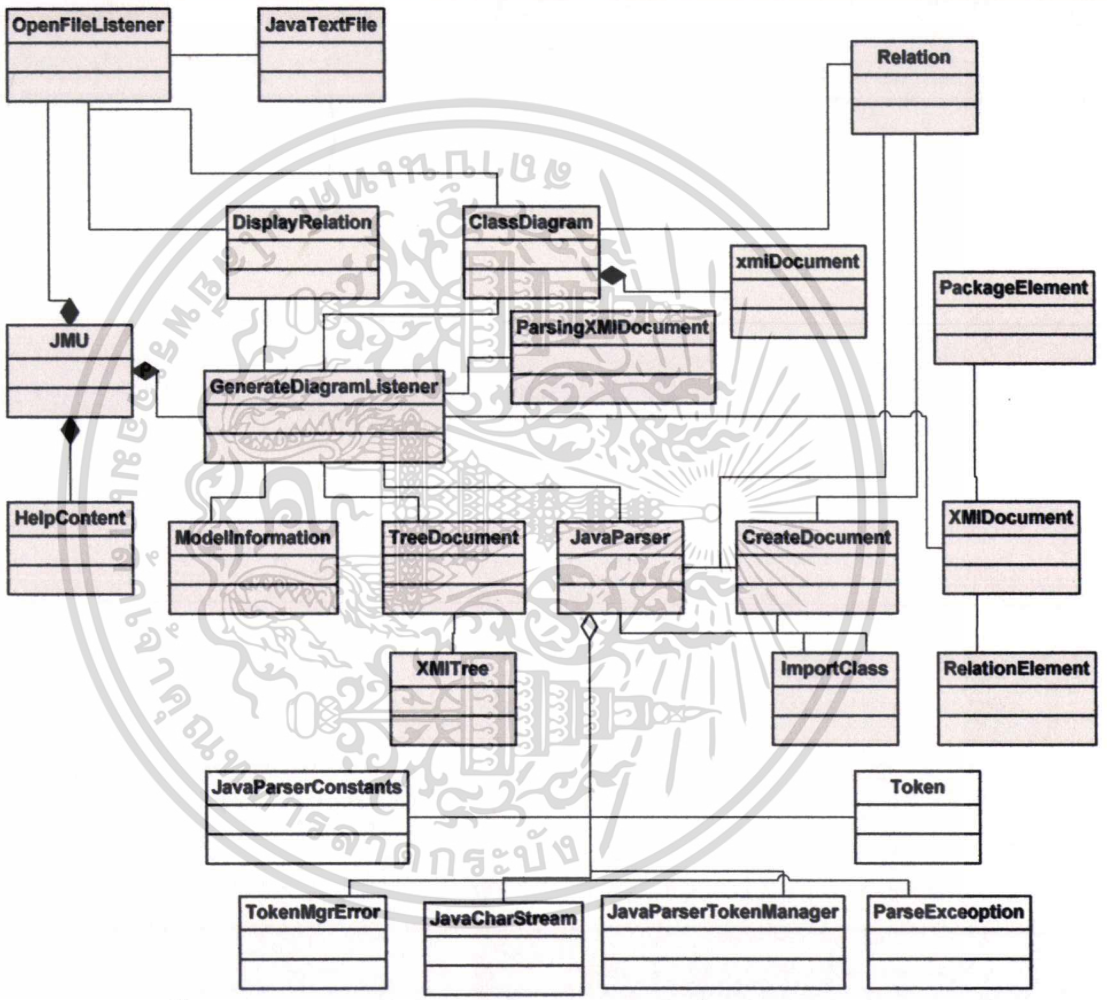
- JMU เป็นคลาสการทำงานหลักของเครื่องมือนี้ ทำงานเกี่ยวกับการทำงานของอินเตอร์เฟสที่ติดต่อกับผู้ใช้ และเป็นส่วนประสานงานโปรแกรมทั้งหมด
- คลาส OpenFileListener เป็นคลาสที่ทำหน้าที่ต่างๆเมื่อผู้ใช้เลือกรายการต่างๆในเมนู Open (Open Java File, Open XMI Document, และ Open JMU File) โดยมีเมธตอดที่สำคัญดังตารางที่ 3.8

- คลาส GenerateDiagramListener เป็นคลาสที่ทำหน้าที่ต่างๆเมื่อผู้ใช้เลือกรายการต่างๆ ในเมนู Generate Diagram(Generate from Java File และ Generate from XMI Document)

ตารางที่ 3.8 แสดงเมธอดที่สำคัญของคลาส OpenFileListener

เมธอด	หน้าที่
openJavaFile()	ดำเนินการเปิดไฟล์จาวา
openXMIFile()	ดำเนินการเปิดไฟล์เอกสารเอ็กซ์เอ็มไอ
openJMUFile()	ดำเนินการเปิดไฟล์ .jmu

- คลาส HelpContent เป็นคลาสที่ทำหน้าแสดงรายละเอียดของเมนูของการแสดงความช่วยเหลือและรายละเอียดการใช้งานต่างเครื่องมือ JMU
- คลาส ModelInformation เป็นคลาสเกี่ยวกับ Model Information Dialog Box โดยข้อมูลที่ได้จากคลาสนี้เป็นไดเร็คทอรี่และชื่อไฟล์เอกสารทางเอ็กซ์เอ็มไอที่เกิดขึ้นในขั้นตอนของการแมพ โค้ดจาวาเป็นคลาสไดอะแกรม
- JavaParserConstants เป็นอินเตอร์เฟสคลาส โดยที่คลาสนี้ไม่มีเมธอด เป็นอินเตอร์เฟสคลาสที่เก็บค่าแอตทริบิวต์ต่างๆ(เกี่ยวกับส่วน Lexical Specification)
- JavaCharStream เป็นคลาสที่แสดงสตรีมของ input character โดยมีการติดต่อกับ JavaParser class ที่รับไฟล์ที่จะทำการวิเคราะห์
- JavaParserTokenManager เป็น Lexical Analyzer
- Token เป็นคลาสที่เป็นตัวแทน token ในแต่ละอ๊อบเจท Token (สิ่งที่ระบุว่าเป็น TOKEN ใน Lexical Specification จะถูกสร้างเป็นอ๊อบเจท Token) สำหรับเมธอดที่สำคัญของคลาสนี้ เช่น toString() : String ทำหน้าที่แปลงอ๊อบเจทให้เป็นสตริงโดยการส่งค่ากลับ และ newToken(ofKind : int) ทำหน้าที่นำค่าที่เป็นตัวเลข(ที่ได้มาจากการสร้าง Lexical Specification) ให้เป็นสตริง
- ParseException เป็นคลาสที่ตรวจจับข้อผิดพลาดที่เกี่ยวกับรูปแบบ Grammar Specification คลาส ParseException มีเมธอดที่สำคัญเช่น getMessage() : String ทำหน้าที่จับความผิดพลาดในการวิเคราะห์ token ในไฟล์นั้นพร้อมแสดงความผิดพลาดว่าเกิดจากการไม่สามารถวิเคราะห์ token ตัวใดที่ไม่ตรงตาม รูปแบบที่ระบุไว้ใน Grammar Specification



รูปที่ 3.8 แสดงโครงสร้างคลาสไดอะแกรมของเครื่องมือ JMU

- TokenMgrError เป็นคลาสที่จัดการสิ่งที่ผิดพลาดในการวิเคราะห์ token คลาส TokenMgrError มีเมธตอดที่สำคัญ เช่น LexicalError() : String ทำหน้าที่รายงานความผิดพลาดที่เกี่ยวกับ Lexical Specification ว่าไม่สามารถวิเคราะห์ token ได้ที่ token ใด

- JavaParser เป็นคลาสหลักที่ใช้ทำการวิเคราะห์โค้ดจาวา คลาส JavaParser มีเมธตอดที่สำคัญดังตารางที่ 3.9

ตารางที่ 3.9 แสดงเมธตอดที่สำคัญของคลาส JavaParser

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

- Relation เป็นคลาสที่จัดการเกี่ยวกับการเก็บข้อมูลของความสัมพันธ์ต่างๆที่วิเคราะห์ได้จากโค้ดจาวา มีรายละเอียดเมธตอดที่สำคัญดังตารางที่ 3.10

- CreateDocument เป็นคลาสที่ทำหน้าที่ในการสร้างเอกสารเอ็กซ์เอ็มไอ โดยข้อมูลที่ใช้ในการสร้างเอกสารทางเอ็กซ์เอ็มไอเป็นข้อมูลของโค้ดจาวาที่ผ่านการวิเคราะห์แล้ว โดยในคลาสนี้มีเมธตอดที่สำคัญดังตารางที่ 3.11

- ImportClass เป็นคลาสที่รวบรวมชื่อไฟล์ของจาวาไลบารี่(java library) เพื่อให้ใช้ในการตรวจสอบความสัมพันธ์แบบ Dependency โดยในคลาส ImportClass มีเมธตอดที่สำคัญดังตารางที่ 3.12

ตารางที่ 3.10 แสดงเมธอดที่สำคัญของคลาส Relation

เมธอด	หน้าที่
checkTypeParaforRelation(type: String)	ตรวจสอบชนิดของพารามิเตอร์ของชนิดพารามิเตอร์
setPackage(name: String, frompackage: String)	ตรวจสอบและเก็บค่าของข้อมูลในส่วนของ package statement
setGeneralization(name:String, special:String)	เก็บค่าของค่าที่การใช้คีย์เวิร์ด extends โดยจะทำการเก็บค่าของชื่อคลาสและชื่อคลาสที่คลาสนั้นทำการ Generalization

ตารางที่ 3.11 แสดงเมธอดที่สำคัญของคลาส CreateDocument

เมธอด	หน้าที่
assignIdClass(name:String):String	กำหนดไอดี
setGeneralizationElement(afterextends:String,nameClass:String)	สร้างอิลิเมนต์ UML:Generalization
createAttributeElement()	สร้างอิลิเมนต์แอทริบิวต์ของคลาส

ตารางที่ 3.12 แสดงเมธอดที่สำคัญของคลาส ImportClass

เมธอด	หน้าที่
loadClass(packageName:String, className:String)	เก็บค่าของอิมพอร์ตไฟล์ที่มีการเรียกใช้ในโค้ดจาวา
checkRefType(name:String):boolean	เป็นการตรวจสอบว่าชนิดของแอทริบิวต์หรือชนิดของพารามิเตอร์ว่าเป็น Reference Type หรือไม่
checkClassFromPackage(nameofPackage:String ,name:String):boolean	ตรวจสอบ name ว่าเป็นคลาสที่อยู่ในแพ็จเกจหรือไม่

- PackageElement เป็นคลาสที่สร้างไฟล์ที่บรรจุอิลิเมนต์ทั้งหมดของ <UML:Package> โดยที่จะมีเฉพาะคลาสที่ได้มีการเรียกใช้โค้ดจาวาที่ทำการแปลงเป็นคลาสไดอะแกรม มีรายละเอียดของเมธอดดังตารางที่ 3.13

ตารางที่ 3.13 แสดงเมธตอดที่สำคัญของคลาส PackageElement

เมธตอด	หน้าที่
getIdClassinPackage(classinpack:String) :String	เก็บค่าไอดีของคลาสที่เป็นคลาสในแพ็กเกจ
findIdClassinPackage(classinpack:String):String	หาค่าของไอดีของคลาสในแพ็กเกจ
createPackageElement()	สร้างอิลิเมนต์และรายละเอียดของแพ็กเกจ

- RelationElement เป็นคลาสในการสร้างอิลิเมต์ต่างๆที่สื่อถึงความสัมพันธ์ต่างๆ เช่น อิลิเมนต์ <UML:Association> เป็นต้น คลาส RelationElement มีเมธตอดที่สำคัญดังตารางที่ 3.14

ตารางที่ 3.14 แสดงเมธตอดที่สำคัญของคลาส RelationElement

เมธตอด	หน้าที่
createAssociaitionElement()	ทำการสร้างอิลิเมนต์ UML:Association ที่เป็นการระบุความสัมพันธ์แบบ Association
createRealizationElement()	สร้างแอทริบิวต์ client และ supplier ในอิลิเมนต์คลาสเพื่อระบุความสัมพันธ์แบบ Realization
createAggregationElement()	ทำการสร้างอิลิเมนต์ UML:Association ที่เป็นการระบุความสัมพันธ์แบบ Aggregation
createDependencyElement()	ทำการสร้างอิลิเมนต์ UML:Dependency ที่เป็นการระบุความสัมพันธ์แบบ Dependency

- XMIDocument เป็นคลาสที่ทำการสร้างเอกสารเอ็กซ์เอ็มไอ โดยทำการสร้าง header และ content ต่างๆของเอกสารทางเอ็กซ์เอ็มไอ
- ClassDiagram เป็นคลาสที่ทำการสร้างคลาสไดอะแกรม มีเมธตอดสำคัญดังตารางที่ 3.15
- xmiDocument เป็นคลาสที่ทำหน้าที่ในการวิเคราะห์เอกสารทางเอ็กซ์เอ็มไอแล้วนำสิ่งที่ได้จากการวิเคราะห์ไปวาดคลาสไดอะแกรม ในคลาสนี้มีเมธตอดที่สำคัญดังตารางที่ 3.16

ตารางที่ 3.15 แสดงเมธตอดที่สำคัญของคลาส ClassDiagram

เมธตอด	หน้าที่
startDrawingDiagram()	เริ่มต้นกระบวนการทำงานในการวาดคลาสไดอะแกรม
ClassDiagram(path:String, name:String)	เป็นคอนสตรัคเตอร์สำหรับกรณีสร้างคลาสไดอะแกรมจากการแมพโค้ดจาวาเป็นคลาสไดอะแกรม
ClassDiagram(jmu:File)	เป็นคอนสตรัคเตอร์สำหรับกรณีสร้างคลาสไดอะแกรมจากเปิดไฟล์ ".jmu"

ตารางที่ 3.16 แสดงเมธตอดที่สำคัญของคลาส xmiDocument

เมธตอด	หน้าที่
parseDocument (path:String, modelfile:String)	รับข้อมูลที่ชื่อไฟล์ที่ต้องการวิเคราะห์
startDocument( )	เริ่มต้นการวิเคราะห์เอกสาร
createAttributeVertex()	นำข้อมูลที่ได้จากการวิเคราะห์แอทริบิวต์อิลิเมนต์มาใช้ในการสร้างภาพ

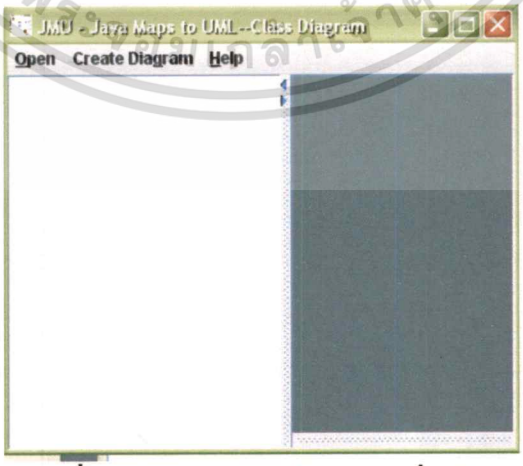
- DisplayRelation ทำการสร้างข้อมูลแสดงรายละเอียดความสัมพันธ์ โดยจะถูกเรียกใช้งานเมื่อมีการแสดงคลาสไดอะแกรมจะมีการแสดงชนิดเส้นความสัมพันธ์ไว้ด้านล่างของหน้าต่าง
- TreeXMIDocument แสดงเอกสารทางเอ็กซ์เอ็มไอบนหน้าจอ
- XMLTree วิเคราะห์เอกสารทางเอ็กซ์เอ็มไอโดยมีการใช้ JTree เพื่อให้วิเคราะห์เอกสารทางเอ็กซ์เอ็มไอแล้วแสดงออกในรูปแบบของทรี

บทที่ 4

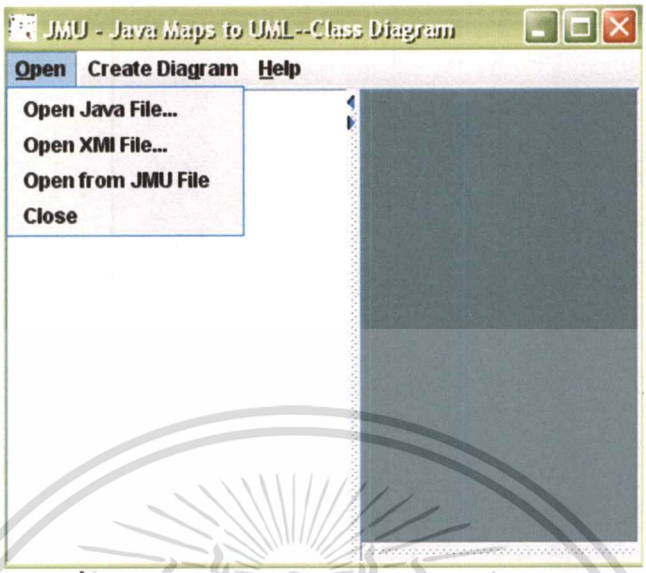
ผลการทดลอง

4.1 อินเตอร์เฟสของเครื่องมือ JMU

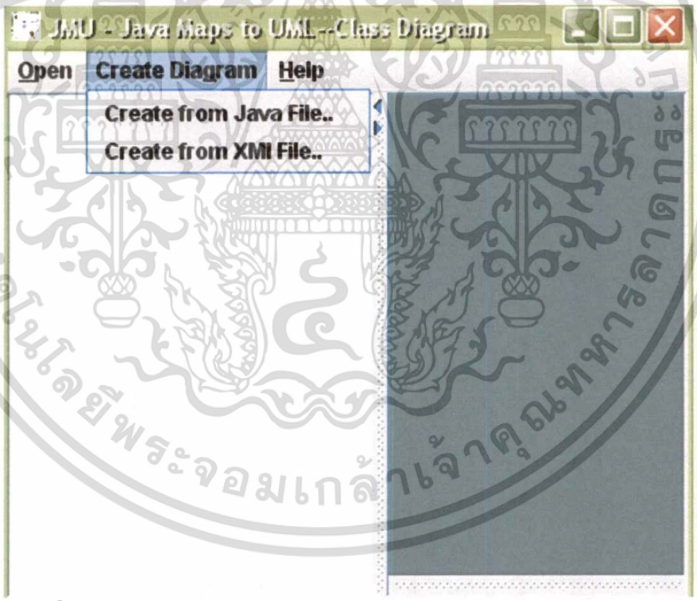
- เครื่องมือที่สร้างในโครงการพัฒนาระบบงานได้ถูกตั้งชื่อว่า JMU ซึ่งย่อมาจากคำว่า Java Map to UML โดยในการใช้งานเครื่องมือนี้จะทำการรันไฟล์ที่มีชื่อ JMU.java
- เครื่องมีการรองรับการทำงาน 2 ส่วน ซึ่งมีรายละเอียดดังรูปที่ 4.1 แสดงอินเตอร์เฟสเมื่อเปิดเครื่องมือ JMU สำหรับรายละเอียดการทำการมีดังต่อไปนี้
    1) การแสดงไฟล์(โดยการใช้เมนู Open) โดยเครื่องมือนี้สามารถแสดงไฟล์ได้ 3 ชนิด คือ ไฟล์จาวา, ไฟล์เอกสารทางเอ็กซ์เอ็มไอ, และไฟล์".jmu"(ซึ่งเป็นไฟล์โมเดลคลาสไดอะแกรม) ดังรูปที่ 4.2 แสดงความสามารถในการทำงานของเครื่องมือ JMU ในส่วนของการแสดงไฟล์
    2) การสร้างไดอะแกรม(โดยการใช้เมนู Generate Diagram) โดยสามารถสร้างไดอะแกรมได้จากแหล่งข้อมูล 2 ส่วน คือ ไฟล์หรือกลุ่มไฟล์จาวา และ ไฟล์เอกสารเอ็กซ์เอ็มไอ ดังรูปที่ 4.2 ที่แสดงความสามารถในการทำงานของโปรแกรม JMU ในส่วนของการสร้างไดอะแกรม



รูปที่ 4.1 แสดงอินเตอร์เฟสของเครื่องมือ



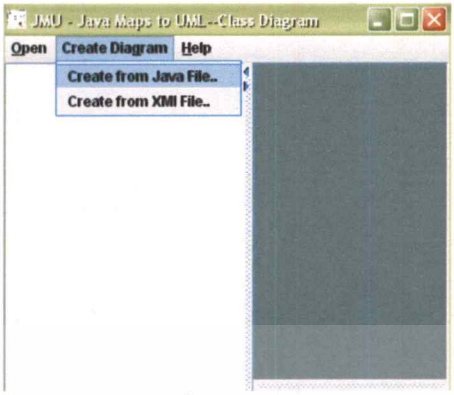
รูปที่ 4.2 แสดงอินเตอร์เฟสในส่วนของการเปิดไฟล์



รูปที่ 4.3 แสดงอินเตอร์เฟสในส่วนของการสร้างไดอะแกรม

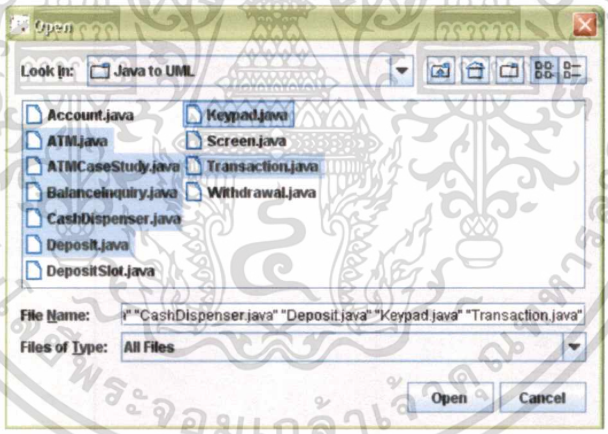
4.2 การแมพโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม

เมื่อผู้ใช้มีความต้องการในการแปลงโค้ดจาวาเป็นคลาสไดอะแกรม ผู้ใช้ต้องทำการเลือกเมนู Generate Diagram ต่อจากนั้นเลือก Generate from Java File ดังรูปที่ 4.4 ที่แสดงอินเตอร์เฟสในเริ่มต้นใช้งาน การสร้างคลาสไดอะแกรมจากไฟล์จาวา



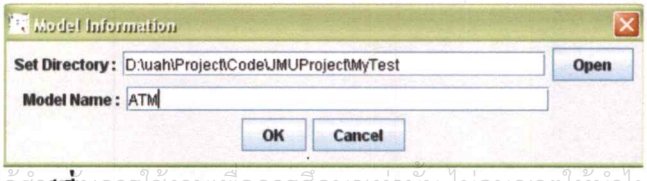
รูปที่ 4.4 แสดงอินเตอร์เฟสการเริ่มต้นการสร้างไดอะแกรมจากไฟล์จาวา

ต่อจากนั้นระบบจะแสดงไฟล์และไดเร็คทอรี่ให้ผู้ใช้เลือกไฟล์หรือกลุ่มไฟล์จาวา ดังรูปที่ 4.5 ที่แสดง Open Dialog Box ให้ผู้ใช้เลือกกลุ่มหรือไฟล์จาวาที่ต้องการแปลงเป็นคลาสไดอะแกรม (รูปแบบไฟล์จาวาที่นำเข้าอยู่ใน ภาคผนวก ค)



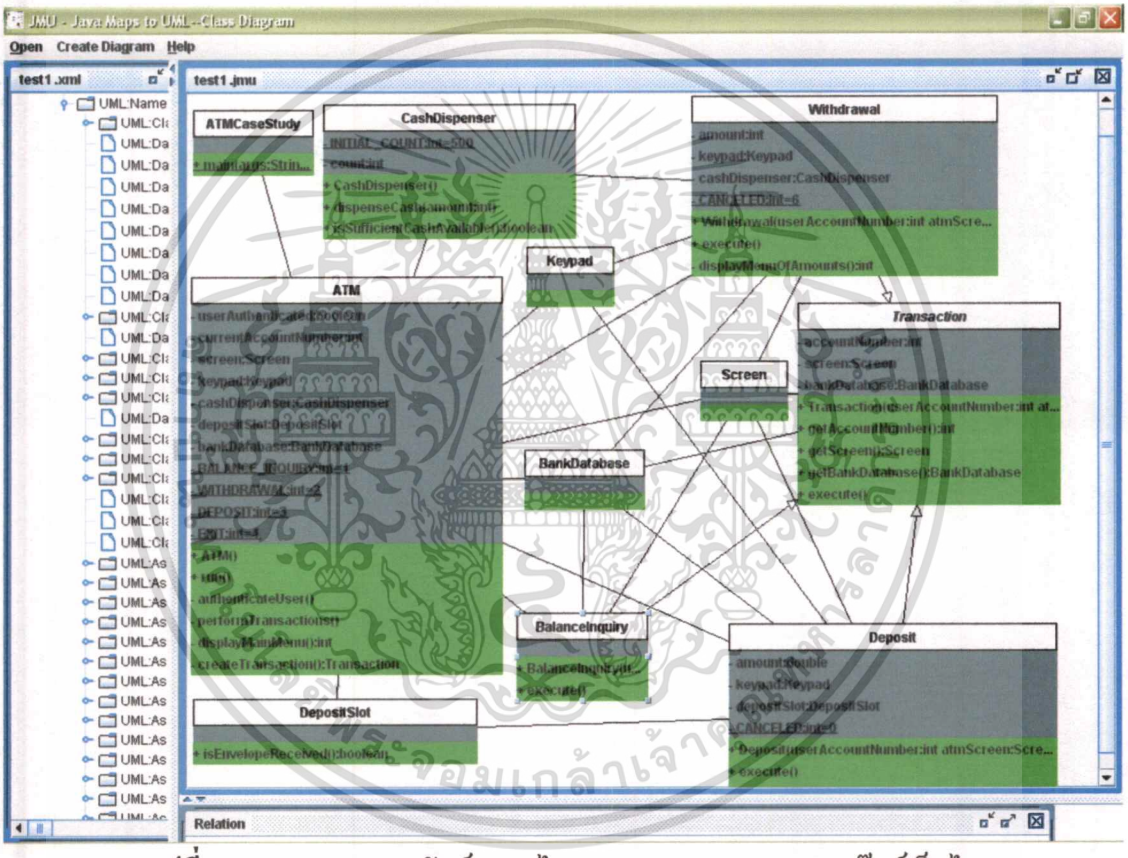
รูปที่ 4.5 ระบบแสดง Open Dialog Box เลือกไฟล์หรือกลุ่มไฟล์จาวา

เมื่อเลือกไฟล์หรือกลุ่มไฟล์ของจาวาที่ต้องการแปลงเป็นคลาสไดอะแกรมแล้วระบบจะแสดง Model Information Dialog Box ดังรูปที่ 4.6 เพื่อให้ผู้ใช้ระบุไดเร็คทอรี่และชื่อไฟล์ ที่ต้องการเก็บไฟล์ “ชื่อไฟล์.xml”(เป็นเอกสารทางเอ็กซ์เอ็มแอล) และ “ชื่อไฟล์.jmu”(เป็นคลาสไดอะแกรมที่ถูกสร้างด้วยระบบ)



รูปที่ 4.6 ระบบแสดง Model Information Dialog Box

ต่อจากนั้นระบบจะทำการสร้างไฟล์เอกสารทางเอ็กซ์เอ็มไอและไฟล์ ".jmu" ไว้ในไดเร็คทอรี่ที่ผู้ใช้ได้กำหนดไว้ พร้อมกับระบบทำการแสดงไฟล์เอกสารทางเอ็กซ์เอ็มไอที่ได้ทางด้านซ้ายของเฟรมในรูปแบบของทรี และแสดงคลาสไดอะแกรมที่ได้จากการแมพโค้ดจาวาไว้ด้านขวาของเฟรมดังรูปที่ 4.7 ที่แสดงผลลัพธ์คลาสไดอะแกรมและเอกสารเอ็กซ์เอ็มไอ ส่วนรายละเอียดของผลลัพธ์เอกสารเอ็กซ์เอ็มไอจะแสดงในภาคผนวก ค

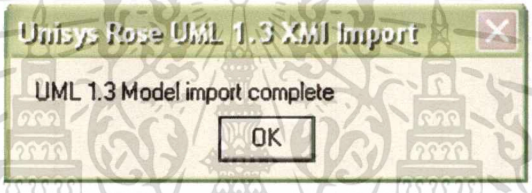


รูปที่ 4.7 ระบบแสดงผลลัพธ์คลาสไดอะแกรมและเอกสารทางเอ็กซ์เอ็มไอ

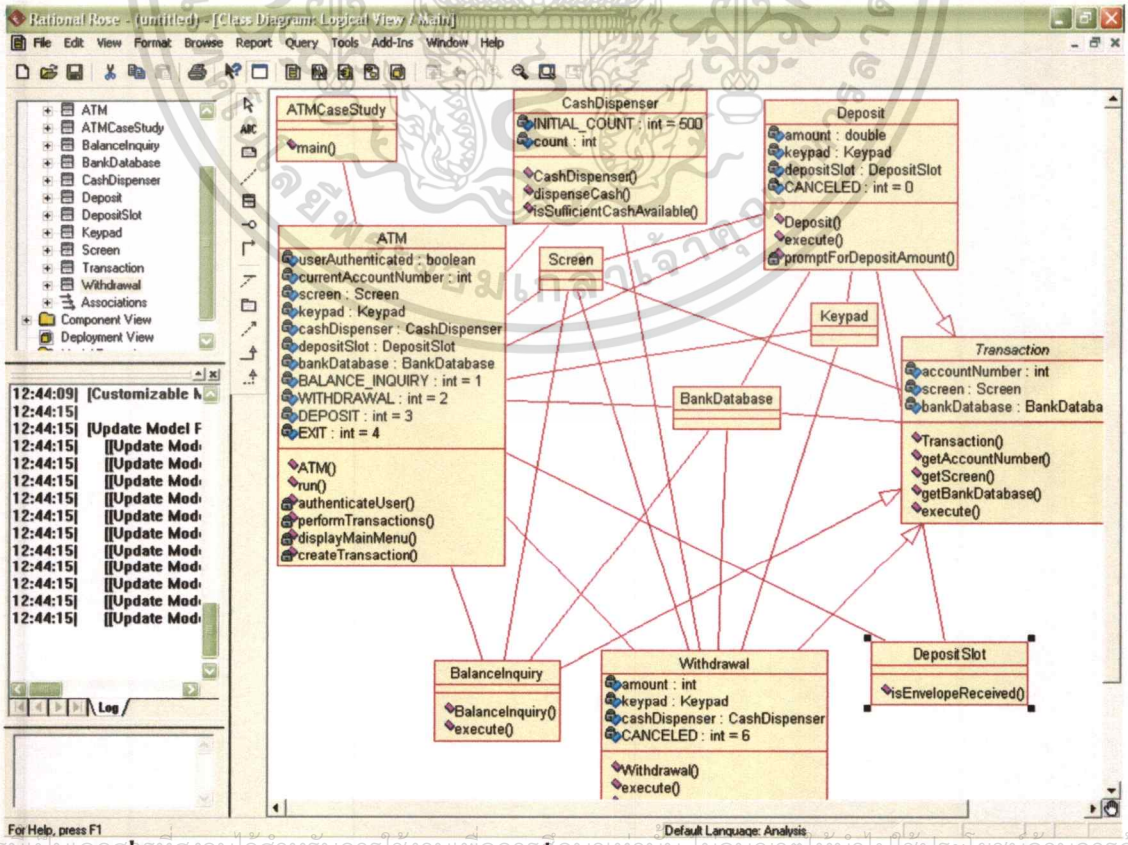
การทดสอบนี้เป็นการนำเอกสารทางเอ็กซ์เอ็มไอที่ได้จากขั้นตอนการแมพโค้ดจาวาเป็นคลาสไดอะแกรมในหัวข้อ 4.2 มาทดสอบบนเครื่องมืออื่นที่สนับสนุนมาตรฐานเอ็กซ์เอ็มไอ ซึ่งในโครงงานนี้ได้นำเครื่องมือที่มีชื่อว่า Rational Rose มาใช้ในการทดสอบผลลัพธ์ที่ได้จากเครื่องมือ JMU (เอกสารทางเอ็กซ์เอ็มไอ)

เนื่องจาก Rational Rose 2002 ได้ทำการรองรับยูเอ็มแอลเมต้าโมเดลเวอร์ชั่น 1.3 ดังนั้นจึงได้ทำการเปลี่ยน <XMI.metamodel xmi.name="UML" xmi.version="1.5"/> เป็น <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>

การทำการทดสอบบนโปรแกรม Rational Rose ต้องทำการ add-in Unisys XMI Tool ต่อจากนั้นจึงทำการอิมพอร์ตไฟล์เอ็กซ์เอ็มไอจากการเลือกเมนู Tool -> UML 1.3 XMI Addin -> UML 1.3 XMI Import ต่อจากนั้นจึงทำการเลือกไฟล์เอกสารเอ็กซ์เอ็มไอ โดยเมื่อการนำเข้าเอกสารเอ็กซ์เอ็มไอกับเครื่องมือ Rational Rose ไม่มีข้อผิดพลาดในการวิเคราะห์ไฟล์เอกสารเอ็กซ์เอ็มไอนี้จะปรากฏข้อความตามรูปที่ 4.8 ที่แสดงข้อความการนำเข้าไฟล์เอกสารเอ็กซ์เอ็มไอที่ถูกสร้างโดย JMU สำเร็จ โปรแกรมจะทำการสร้างคลาสไดอะแกรมดังรูปที่ 4.9 ที่แสดงผลลัพธ์ของการนำเอกสารทางเอ็กซ์เอ็มไอที่ได้จากหัวข้อ 4.2 โดยผลลัพธ์ที่ได้จะแบ่งเป็นสองส่วน คือ ผลลัพธ์ทั้งหมดจะปรากฏทางด้านซ้ายของเครื่องมือ ต่อจากนั้น สามารถนำมาวาดคลาสไดอะแกรมได้โดยลากผลลัพธ์ทางด้านซ้ายมายังด้านขวาของเครื่องมือ



รูปที่ 4.8 แสดงข้อความการนำเข้าเอกสารเอ็กซ์เอ็มไอสำเร็จ



รูปที่ 4.9 แสดงผลลัพธ์คลาสไดอะแกรมที่ได้จากการแมพไฟล์เอกสารทางเอ็กซ์เอ็มไอ

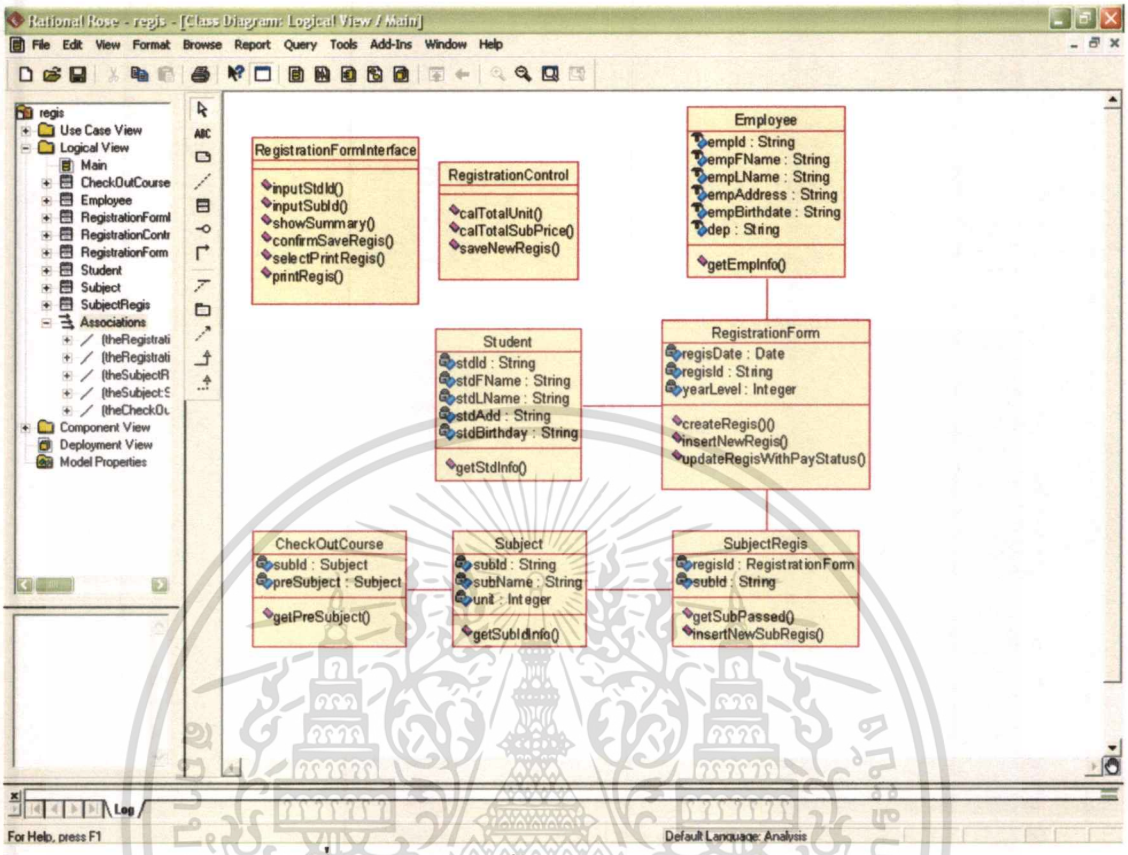
สรุปการทดสอบเอกสารทางเอ็กซ์เอ็มไอที่ถูกสร้างจากเครื่องมือของโครงการพัฒนาระบบนี้สามารถนำไปสร้างไดอะแกรมกับเครื่องมือ Rational Rose ได้ แต่คลาสไดอะแกรมที่ได้ไม่สามารถที่แสดงบนเครื่องมือ Rational Rose ไม่สามารถแสดงแอตทริบิวต์หรือโอเปอร์ชั่นที่เป็น static ได้ (คือ แอตทริบิวต์หรือโอเปอร์เรชั่นนั้นจะมีการขีดเส้นใต้)

4.3 การแปลงเอกสารทางเอ็กซ์เอ็มไอเป็นคลาสไดอะแกรม

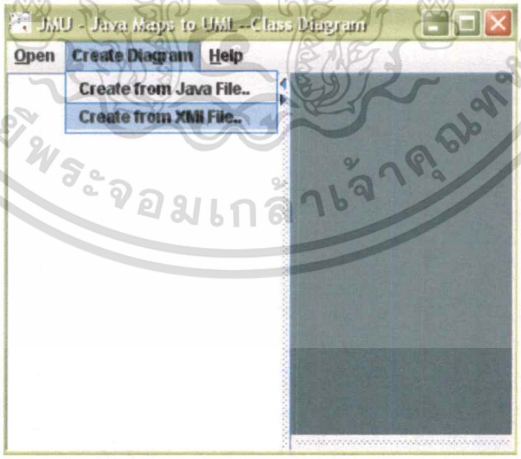
โครงงานนี้เป็นการพัฒนาเครื่องมือในการทำรีเวอร์สเอ็นจีเนียริ่งของ โค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรมโดยมีการนำมาตรฐานเอ็กซ์เอ็มไอมาประยุกต์ใช้ เพื่อแก้ปัญหาของผลลัพธ์ที่ได้จากกการทำรีเวอร์สเอ็นจีเนียริ่งที่ส่วนมากมีลักษณะของผลลัพธ์ที่ขึ้นอยู่กับเครื่องมือนั้น ไม่มีความสามารถในการแลกเปลี่ยน ซึ่งเครื่องมือ JMU นี้ถูกพัฒนาให้มีความสามารถในการทำรีเวอร์สเอ็นจีเนียริ่งและนำเอกสารเอ็กซ์เอ็มไอที่ได้จากการถูกจำแนกโดยเครื่องมือนี้นำไปเปิดใช้บนเครื่องมืออื่นๆ(อย่างเช่น Ration Rose)ได้ นอกจากนี้ยังสามารถนำเอกสารเอ็กซ์เอ็มไอที่ถูกจำแนกจากเครื่องมืออื่นๆมาแสดงคลาสไดอะแกรมในเครื่องมือ JMU ได้ โดยที่เอกสารเอ็กซ์เอ็มไอที่ถูกจำแนกโดยเครื่องมืออื่นนั้นอาจจะเป็นเอกสารเอ็กซ์เอ็มไอที่เป็นการ export คลาสไดอะแกรมให้อยู่ในรูปแบบของการแสกเปลี่ยน(เอกสารเอ็กซ์เอ็มไอ) ก็ได้

การทดสอบความสามารถในการนำเข้าเอกสารเอ็กซ์เอ็มไอ ให้สามารถแสดงผลลัพธ์เป็นคลาสไดอะแกรม โดยได้นำเอกสารเอ็กซ์เอ็มไอที่ถูกจำแนกโดยเครื่องมือที่มีชื่อ Rational Rose โดยรูปที่ 4.10 เป็นการแสดงคลาสไดอะแกรมบน Rational Rose ต่อจากนั้นทำการ export เป็นเอกสารเอ็กซ์เอ็มไอที่มีชื่อว่า regis.xml

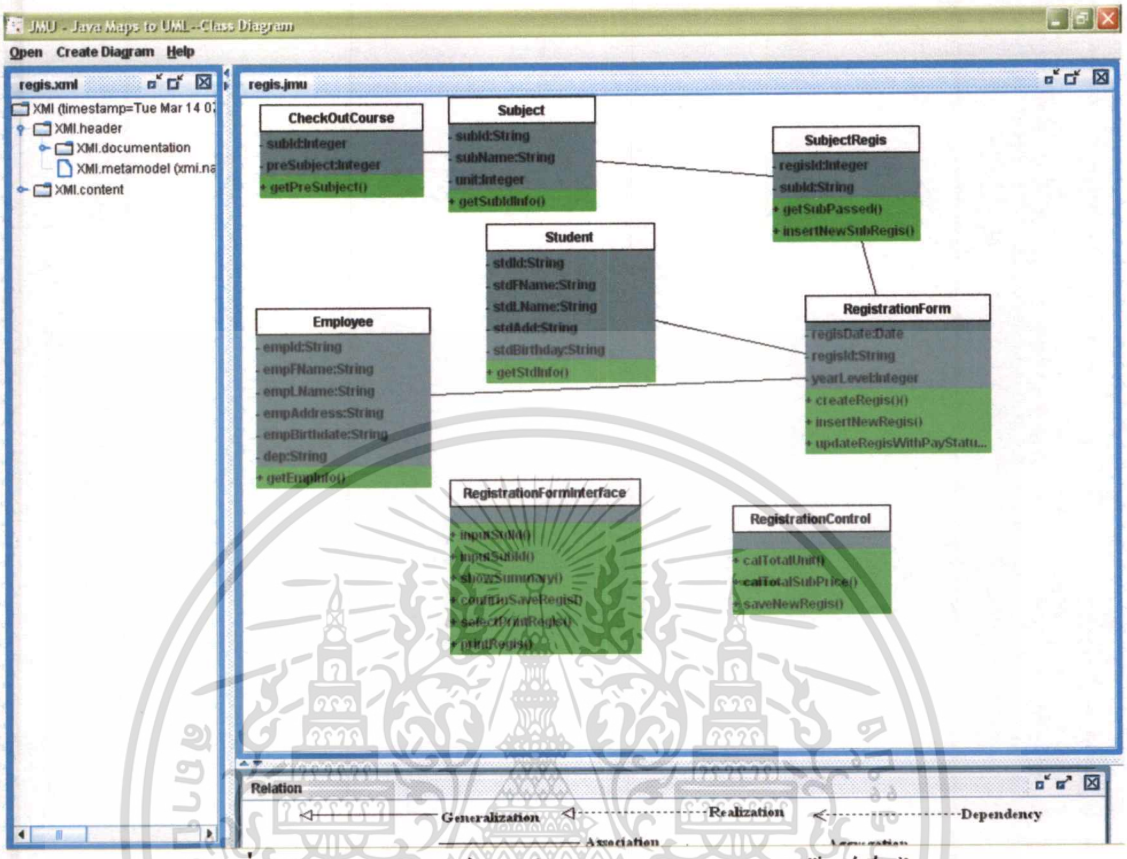
จากรูปที่ 4.11 เป็นอินเตอร์เฟสเริ่มต้นของการสร้างคลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอที่ถูกสร้างจากเครื่องมืออื่น โดยการใช้งานเลือกเมนู Create Diagram -> Create from XMI Document และรูปที่ 4.12 เป็นการแสดงผลลัพธ์คลาสไดอะแกรมที่จากเอกสารเอ็กซ์เอ็มไอ



รูปที่ 4.9 แสดงคลาสไดอะแกรมบน Rational Rose



รูปที่ 4.10 แสดงเริ่มต้นการใช้งานการสร้างคลาสไดอะแกรมด้วยเอกสารเอ็กซ์เอ็มไอ



รูปที่ 4.11 แสดงผลลัพธ์คลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอ

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปการพัฒนาระบบ

โครงการพัฒนาระบบงานนี้ เป็นการพัฒนาเครื่องมือที่ใช้การแมพโค้ดไปเป็นยูเอ็มแอลคลาสไดอะแกรม โดยในการพัฒนาเครื่องมือนี้ได้คำนึงถึงแนวทางในการแก้ปัญหาซึ่งเป็นปัญหาที่เกิดขึ้นในเครื่องมือที่มีความสามารถในการแปลงโค้ดจาวาสู่คลาสไดอะแกรม คือ ปัญหาของผลลัพธ์ที่ได้นั้นมักจะเป็นผลลัพธ์ที่ขึ้นอยู่กับเครื่องมือที่ผลิตผลลัพธ์นั้นๆ จากปัญหาที่เกิดขึ้นนี้ จะเป็นปัญหาที่ถูกนำมาพิจารณาในการสร้างเครื่องมือของโครงงานพัฒนาระบบนี้

จากการศึกษาเครื่องมืออื่นๆที่มีความสามารถในการทำรีเวอร์สเอ็นจิเนียริ่งและการศึกษามาตรฐานยูเอ็มแอล พบว่า มาตรฐานที่มีชื่อว่า เอ็กซ์เอ็มไอ(XMI-XML Metadata Interchange) เป็นมาตรฐานที่ถือกำเนิดขึ้นมาเพื่อการแลกเปลี่ยนรายละเอียดข้อมูลเชิงโมเดล อีกทั้งทางโอเอ็มจี ซึ่งเป็นองค์กรที่รับผิดชอบยูเอ็มแอลได้ผนวกมาตรฐานนี้(เพื่อให้ในการแลกเปลี่ยน)เข้าสู่ข้อกำหนดของยูเอ็มแอล ทำให้โครงการงานพัฒนาระบบนี้ ได้ทำการพัฒนาเครื่องมือในการแมพโค้ดจาวาเป็นคลาสไดอะแกรม โดยได้นำมาตรฐานเอ็กซ์เอ็มไอมาประยุกต์เพื่อแก้ปัญหาของลักษณะผลลัพธ์ที่ขึ้นอยู่กับเครื่องมือ

ในการวิเคราะห์โค้ดภาษาจาวาได้มีการนำ JavaCC (โดยการประยุกต์ java1.4.jj) เพื่อการวิเคราะห์โค้ดจาวาแมพเข้าสู่เอกสารทางเอ็กซ์เอ็มไอ(ใช้ DOM parser ในการสร้างเอกสาร) ต่อจากนั้นจึงนำเอกสารทางเอ็กซ์เอ็มไอมาวิเคราะห์ด้วย SAX parser โดยข้อมูลที่ได้จะถูกนำไปประมวลผลเพื่อแสดงคลาสไดอะแกรม

เอกสารทางเอ็กซ์เอ็มไอที่ได้จากการสร้างโดยเครื่องมือของโครงงานนี้(JMU) ได้นำไปทดสอบบนเครื่องมือที่มีความสามารถในการทำรีเวอร์สเอ็นจิเนียริ่งและรองรับมาตรฐานเอ็กซ์เอ็มไอ โดยในโครงงานนี้ได้เลือกเครื่องมือที่มีชื่อว่า Rational Rose ให้เป็นเครื่องมือในการทดสอบ ซึ่งการทดสอบนั้น ได้นำไฟล์เอกสารเอ็กซ์เอ็มไอที่ JMU ได้ทำการสร้างในขั้นตอนของการแมพ โค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม มารันบนโปรแกรม Rational Rose ปรากฏว่าโปรแกรม Rational Rose สามารถวาดคลาสไดอะแกรมได้เหมือนกับโปรแกรม JMU

นอกจากนี้ในการทดสอบความสามารถในการแลกเปลี่ยนได้นำเอกสารเอ็กซ์เอ็มไอที่ถูกสร้างโดยเครื่องมืออื่น(โดยได้นำเอกสารเอ็กซ์เอ็มไอที่ถูกสร้างด้วยโปรแกรม Rational Rose)

5.2 ประโยชน์ที่ได้จากการศึกษาและพัฒนาระบบ

-เข้าใจรายละเอียดของมาตรฐานยูเอ็มแอล, เอ็กซ์เอ็มแอล, และ เอ็กซ์เอ็มไอ

- ได้เข้าใจและประยุกต์ใช้เทคนิครีเวอร์สเอ็นจีเนียริ่ง

- เครื่องมือนี้สามารถอำนวยความสะดวกให้กับผู้พัฒนาระบบที่ต้องบำรุงรักษาหรือพัฒนาระบบที่มีการขาดจัดการเอกสารที่ดีได้

5.2 ข้อเสนอแนะ

การพัฒนาเครื่องมือในการแปลงโค้ดภาษาจาวาไปเป็นยูเอ็มแอลคลาสไดอะแกรมโดยการประยุกต์มาตรฐานเอ็กซ์เอ็มไอให้เป็นตัวกลางในการแมพโค้ดภาษาจาวาเป็นยูเอ็มแอลคลาสไดอะแกรมนั้น มีข้อเสนอแนะดังต่อไปนี้

- การพัฒนาเครื่องมือในการแมพโค้ดจาวาเป็นยูเอ็มแอล ควรมีความสามารถในการแมพโค้ดภาษาจาวาเป็นไดอะแกรมอื่นๆของยูเอ็มแอล เช่น ซีเควนซ์ไดอะแกรม เป็นต้น
- การใช้มาตรฐานเอ็กซ์เอ็มไอเป็นตัวกลางในการแลกเปลี่ยนโค้ดจาวาไปสู่ยูเอ็มแอลนั้น ในข้อกำหนดยูเอ็มแอล 2.0 ได้กำหนดลักษณะของตัวกลางมาตรฐานเอ็กซ์เอ็มไอให้มีความสามารถเพิ่มขึ้นโดยจะมีการลักษณะของ เอ็กซ์เอ็มไอและ SVG(Scale Vector Graphic) รวมเข้าด้วยกัน
- การแสดงคลาสไดอะแกรมที่ได้จากการแมพโค้ดจาวา ควรมีความสามารถในการทำการแก้ไขคลาสไดอะแกรมได้ และการแก้ไขนั้นจะต้องแก้ไขโค้ดจาวาของคลาสนั้นๆด้วย (เป็นเทคนิค forward engineering)

บรรณานุกรม

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

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

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

บุญประเสริฐ สุรักษ์รัตนสกุล. 2546. **วิธีการตรวจสอบวากยสัมพันธ์ของแผนภาพยูเอ็มแอล.** คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

ArgoUML. 2006. **ArgoUML.** [Online]. Available:http://argouml.tigris.org/

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

Fate Software. 2005. **Code Visual for Java 2.0.** [Online]. Available:http://www.fatesoftware.com.

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

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

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

OMG. 2002. **XMI Specification 1.2.** [Online]. Available : http:// www.omg.org

OMG. 2003. **UML Specification 1.5.** [Online]. Available : http:// www.omg.org

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

ภาคผนวก ก

UML Class Diagram

ภาคผนวก ก นี้เป็นการอธิบายรายละเอียดของคลาสไดอะแกรม โดยจะมีอ้างอิงกับข้อกำหนดของยูเอ็มแอล 1.5 ซึ่งอยู่ในส่วนที่ 3 (UML Notation)

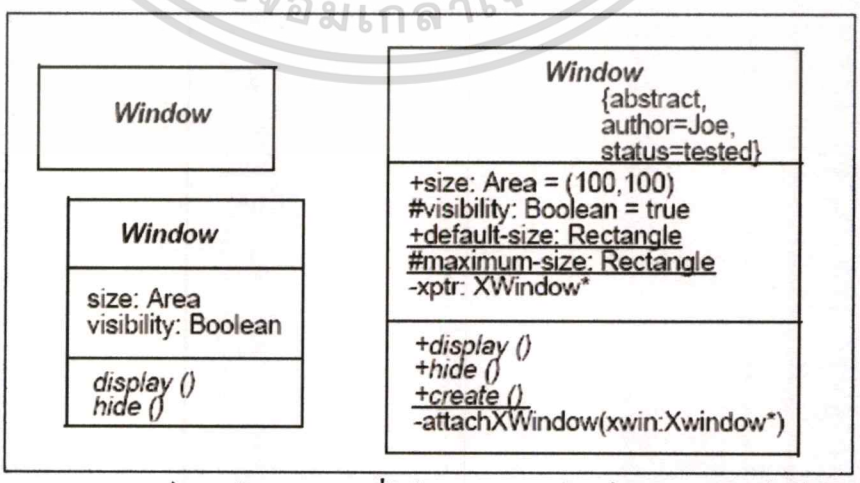
คลาสไดอะแกรมเป็นรูปแบบการแสดงออกของสัญลักษณ์ในมุมมองแบบสแตติก เป็นการแสดงโครงสร้างของระบบว่ามีการประกอบด้วยคลาส(คือ ชนิดของกลุ่มอ๊อบเจท)ใดบ้าง และมีความสัมพันธ์ระหว่างคลาอย่างไร คลาสไดอะแกรมจัดว่าเป็นและมีประโยชน์มากที่สุดสำหรับระบบที่มีการพัฒนาระบบที่มีการเขียนโปรแกรมด้วยภาษาเชิงวัตถุ

องค์ประกอบของคลาสไดอะแกรมมีรายละเอียดดังต่อไปนี้

- Class

  -เป็นการอธิบายถึงกลุ่มของอ๊อบเจทที่มีลักษณะและพฤติกรรมที่เหมือนกันหรือร่วมกัน คลาสมีการแสดงโครงสร้างข้อมูลและพฤติกรรมและความสัมพันธ์กับอิลิเมนต์อื่นๆ

  -การใช้สัญลักษณ์ เป็นการใช้สัญลักษณ์เป็นรูปสี่เหลี่ยม โดยที่สัญลักษณ์รูปสี่เหลี่ยมนี้สามารถนำเสนอได้หลายรูปแบบ ดังรูปที่ ก.1 เป็นการแสดงสัญลักษณ์ของคลาส ที่มีการใช้รูปสี่เหลี่ยมที่แสดงเฉพาะชื่อคลาสและรูปสี่เหลี่ยมที่แสดงส่วนประกอบ 3 ส่วน(ชื่อคลาส, แอทริบิวต์, และ โอเปอร์เรชั่น) สำหรับในกรณีที่ไม่มีแอทริบิวต์หรือโอเปอร์เรชั่นสามารถไม่แสดงส่วนนั้นได้



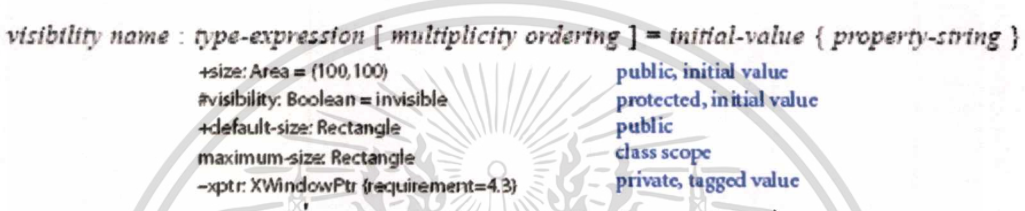
รูปที่ ก.1 แสดงผลลัพธ์คลาสไดอะแกรมที่ได้จากการแมพไฟล์เอกสารทางเอ็กซ์เอ็มไอ

-ชื่อคลาส

- แสดงชื่อของคลาสด้วยหนา
- ในกรณีที่เป็น abstract class ให้แสดงชื่อคลาสนั้นด้วยตัวเอียง

-แอตทริบิวต์

- การเขียนแอตทริบิวต์ มีลักษณะดังรูปที่ ก.2 ที่แสดงรูปแบบและตัวอย่างของแอทริบิวต์



รูปที่ ก.2 แสดงรูปแบบและตัวอย่างแอทริบิวต์

- visibility เป็นการแทนการเข้าถึงโดยมีการเข้าถึงทั้งหมด 3 ระดับ คือ "+" แทนการเข้าถึงแบบ public visibility, "-" แทนการเข้าถึงแบบ private visibility, และ "#" เป็นการแทนการเข้าถึงแบบ protected visibility นอกจากนี้ยังสามารถแทนด้วยคีย์เวิร์ด public, private, และ protected ได้
ในบางกรณีที่ไม่บ่งบอก visibility ไว้ ไม่ได้หมายความว่าแอตทริบิวต์นั้นมีการเข้าถึงแบบ public
- name เป็นการแทนชื่อของแอตทริบิวต์
- เมื่อมีการชื่อของแอตทริบิวต์และตามด้วยชนิดของแอตทริบิวต์โดยมีเครื่องหมาย ":" อยู่ระหว่างชื่อและชนิดของแอตทริบิวต์
- type-expression เป็นการแทนชนิดของแอตทริบิวต์ โดยที่การแทนชนิดนี้เป็นลักษณะของคำ, ชื่อของคลาส, หรือจะเป็นคำที่ขึ้นอยู่กับภาษาโปรแกรมมิ่งก็ได้
- multiplicity ordering เป็นการแสดง multiplicity และ ordering ของแอตทริบิวต์ โดยส่วนนี้สามารถละเว้นได้(และในกรณีที่ละเว้นส่วนนี้ไว้ หมายความว่าแอตทริบิวต์นั้น มี multiplicity เป็น 1..1 และ มี ordering ที่มีค่าแบบ unordered) ordering ของแอตทริบิวต์มีประโยชน์ในการบ่งบอกลักษณะของ multiplicity ที่มีขอบเขตบนมากกว่าหนึ่ง (โดยสามารถบ่งบอก ordering ด้วยคีย์เวิร์ด ordered และ unordered)

■ initial value เป็นค่าของแอตทริบิวต์ โดยการใส่เครื่องมือ "=" และตามด้วยค่าของแอตทริบิวต์นั้น

■ property string เป็นการระบุค่าคุณสมบัติที่ใช้ โดยการใช้ property string ต้องมีการใช้เครื่องมือ "{}" ในแอตทริบิวต์ สามารถมีค่าคุณสมบัติได้ดังนี้

-changeable เป็นการระบุว่าแอตทริบิวต์นั้นสามารถมีการเปลี่ยนแปลงค่าได้ และถือเป็นค่าคุณสมบัติ default ของแอตทริบิวต์

-addOnly เป็นการระบุว่าแอตทริบิวต์นั้นไม่สามารถทำการเปลี่ยนแปลงค่าที่มีอยู่ได้ แต่สามารถเพิ่มค่าของแอตทริบิวต์ได้

-frozen เป็นการระบุว่าแอตทริบิวต์ไม่สามารถเปลี่ยนแปลงหรือเพิ่มเติมค่าได้

■ สำหรับแอตทริบิวต์ ที่มีค่าเป็น "static" จะมีการขีดเส้นใต้แอตทริบิวต์(จะเรียกแอตทริบิวต์นี้เป็น class-scope attribute) ส่วนแอตทริบิวต์ที่ไม่ได้ขีดเส้นใต้ ซึ่งถือเป็นค่า default (เรียกแอตทริบิวต์นี้ เป็น instance-scope attribute)

-โอเปอร์เรชั่น

■ การเขียนโอเปอร์เรชั่น มีรูปแบบการเขียนได้ดังรูปที่ ก.3 ที่แสดงรูปแบบและตัวอย่างของโอเปอร์เรชั่น

visibility name (parameter-list) : return-type-expression { property-string }

```
+display (): Location
+hide ()
+create ()
-attachXWindow(xwin:Xwindow )
```

รูปที่ ก.3 แสดงรูปแบบและตัวอย่างแอทริบิวต์

■ visibility เป็นการแทนการเข้าถึง โดยมีการเข้าถึงทั้งหมด 3 ระดับ คือ "+" แทนการเข้าถึงแบบ public visibility, "-" แทนการเข้าถึงแบบ private visibility, และ "#" เป็นการแทนการเข้าถึงแบบ protected visibility นอกจากนี้ยังสามารถแทนด้วยคีย์เวิร์ด public, private, และ protected ได้

ในบางกรณีที่ไม่บ่งบอก visibility ไว้ ไม่ได้หมายความว่าโอเปอร์เรชั่นนั้นมีการเข้าถึงแบบ public

■ name เป็นการระบุชื่อแอตทริบิวต์

■ parameter list เป็นการระบุพารามิเตอร์ โดยพารามิเตอร์จะเขียนอยู่ในเครื่องหมายวงเล็บและในกรณีที่มีพารามิเตอร์มากกว่าหนึ่งตัวให้แยกพารามิเตอร์ด้วยเครื่องหมาย "," รูปแบบการเขียนพารามิเตอร์ มีรูปแบบดังนี้

kind name : type-expression = default-value

-kind เป็นการทิศทางข้อมูลการไหลของพารามิเตอร์ มีการกำหนดได้ 3 แบบ คือ 1) in เป็นการกำหนดว่าพารามิเตอร์รับค่าเข้ามาโดยการ passed by value ทำให้เมื่อมีการเปลี่ยนแปลงในโอเปอร์เรชั่นจะไม่ส่งผลกับผู้เรียก(เป็นการระบุ default ของพารามิเตอร์) 2) out เป็นการกำหนดว่าโอเปอร์เรชั่นสามารถกำหนดและเปลี่ยนค่าของพารามิเตอร์และส่งค่าของพารามิเตอร์ให้กับผู้เรียกใช้โอเปอร์เรชั่น 3) inout เป็นการบอกว่าโอเปอร์เรชั่นมีการใช้ค่าของพารามิเตอร์และอาจจะทำการเปลี่ยนแปลงค่าของพารามิเตอร์แล้วทำการส่งกลับให้กับผู้เรียกใช้

-name เป็นการประกาศชื่อของพารามิเตอร์

-type-expression เป็นการระบุชนิดของพารามิเตอร์

-default value เป็นการระบุค่า default ของพารามิเตอร์

■ return type expression เป็นการระบุชนิดของค่าที่ส่งกลับคืนให้กลับผู้เรียก และสามารถละเว้นได้ในกรณีที่ไม่มีค่าส่งกลับคืน

property string เป็นการระบค่าคุณสมบัติที่ใช้ โดยการใช้ property string ต้องมีการใช้เครื่องมือ "{}" ในโอเปอร์เรชั่น สามารถมีค่าคุณสมบัติได้ดังนี้

-query เป็นการระบุว่าโอเปอร์เรชั่นไม่สามารถเปลี่ยนแปลงค่าของแอทริบิวต์

-concurrent, sequential, และ guarded เป็นการบ่งบอกลักษณะที่เกิดขึ้นใน thread โดยในการเขียนค่าคุณสมบัติของ thread สามารถเขียนได้ดังรูปแบบนี้ "{ concurrency = name }" โดย name เป็นการแทนคีย์เวิร์ดของ concurrent, sequential, และ guarded

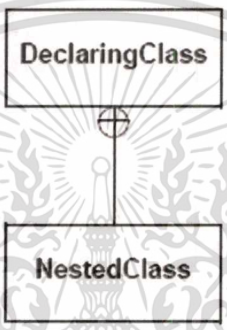
■ สำหรับโอเปอร์เรชั่น ที่มีค่าเป็น "static" จะมีการขีดเส้นใต้โอเปอร์เรชั่น(จะเรียกโอเปอร์เรชั่นนี้เป็น class-scope operation) ส่วนโอเปอร์เรชั่นไม่ได้ขีดเส้นใต้ ซึ่งถือเป็นค่า default (เรียกโอเปอร์เรชั่นนี้ เป็น instance-scope operation)

● Nested Class Declaration

-การประกาศคลาสซ้อน เป็นการการประกาศคลาสไว้ภายในขอบเขตของคลาสอื่น

-ในการประยุกต์ใช้ภาษาเชิงวัตถุ อย่างเช่น ภาษาจาวาสามารถทำการประกาศคลาสไว้ภายในขอบเขตของคลาสอื่นได้ โดยเรียกคลาสที่อยู่ภายในขอบเขตอื่นนั้นว่า nested class

-ในการใช้สัญลักษณ์ที่จะบ่งบอกว่าคลาสเป็นซ้อนอยู่ภายใต้ขอบเขตของคลาสอื่น มีการใช้สัญลักษณ์ดังรูปที่ ก.4 แสดงการใช้สัญลักษณ์ที่บ่งบอกว่าเป็น Nested Class Declaration

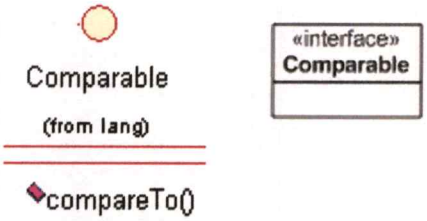


รูปที่ ก.4 แสดงรูปแบบการใช้ Nested Class Declaration

- Interface

-อินเตอร์เฟสสามารถใช้สัญลักษณ์แทนอินเตอร์เฟสได้ 2 แบบดังรูปที่ ก.5 แสดงตัวอย่างการใช้สัญลักษณ์แทนอินเตอร์เฟสที่สามารถแทนด้วยสัญลักษณ์ 2 แบบ คือ แสดงด้วยวงกลมและเขียนชื่ออินเตอร์เฟสกำกับไว้ด้านนอก กับการแสดงด้วยสัญลักษณ์ที่คล้ายกับสัญลักษณ์คลาสที่ตัดส่วนที่สองของคลาส พร้อมทั้งเขียน <<interface>> ไว้ด้านบนของส่วนที่หนึ่งของสัญลักษณ์

-ถ้าต้องการแสดงรายละเอียดของโอเปอร์เรชั่นควรใช้สัญลักษณ์อินเตอร์เฟสที่เป็นรูปสี่เหลี่ยม นอกจากนี้การแสดงรูปแบบของอินเตอร์เฟสโดยใช้สัญลักษณ์สี่เหลี่ยม (ที่อิงรูปแบบคลาส)สามารถละเว้นไม่แสดงส่วนแอตทริบิวต์ได้



รูปที่ ก.5 แสดงตัวอย่างของสัญลักษณ์อินเตอร์เฟส

- Association

-เป็นการบ่งบอกถึงความสัมพันธ์ของกันและกันระหว่างนั้นๆ โดยในรูปที่ ก.6 เป็นแสดงตัวอย่างของความสัมพันธ์แบบ Association โดยที่เส้นตรงที่เชื่อมระหว่างคลาสทั้งสองเป็นการบอกถึงความสัมพันธ์นี้

- ความสัมพันธ์แบบ Association สามารถแยกเป็น Unidirectional Association(เป็นเส้นตรงทึบและมีหัวลูกศร) และ Bidirectional Association(เป็นเส้นตรงทึบ)

-จุดจบของความสัมพันธ์ Association เป็นการเชื่อมต่อคลาส ซึ่งปลายของการเชื่อมต่อของทั้งสองด้านเรียกว่า association end

-Association End เป็นจุดจบของของการเชื่อมโยงความสัมพันธ์ Association โดยที่ในแต่ละ Association ต้องมี Association End เป็นสอง association end หรือมากกว่าสอง association end

- ชื่อความสัมพันธ์ โดยที่ชื่อเหล่านี้มักเป็นคำกิริยาเป็นส่วนใหญ่ สำหรับชื่อที่ใช้กำกับความสัมพันธ์ไม่จำเป็นต้องใช้ได้

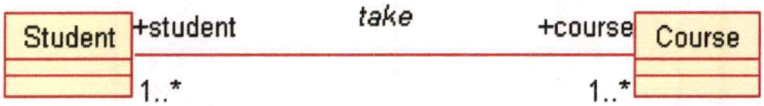
- multiplicity เป็นการบอกปริมาณของคลาสหรืออ๊อบเจทที่มีความสัมพันธ์กันโดยมีรูปแบบการบ่งบอกปริมาณที่มีความสัมพันธ์หลายรูปแบบ อย่างเช่น 0..1, จำนวนเต็มบวก, Many(*), และ 1..* เป็นต้น

-ordering เป็นการบ่งบอกในกรณีของการมี multiplicity ที่มีค่ามากกว่าหนึ่งว่ามีชุดของความสัมพันธ์ที่เกิดขึ้นเป็นลักษณะแบบ ordered และ unordered(ซึ่งเป็นค่า default)

-การที่ความสัมพันธ์แบบ Association ที่จุดปลายของความสัมพันธ์มีการใช้หัวลูกศรเป็นการแสดงว่าความสัมพันธ์นี้ มีคุณสมบัติของ association end ที่เรียกว่า navigability

-นอกจากนี้ที่จุดปลายของความสัมพันธ์แบบ Association สามารถมีการใช้สัญลักษณ์ diamond ที่จุดปลายโดยการใช้สัญลักษณ์นี้ เป็นการแสดงคุณสมบัติของ association end ที่เรียกว่า aggregation (ซึ่งคุณสมบัติแบ่งเป็นการใช้สัญลักษณ์ diamond ที่เป็นแบบโปร่งและแบบทึบ)

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



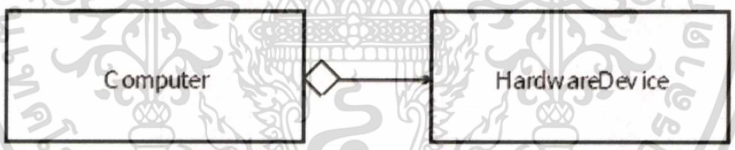
รูปที่ ก.6 แสดงตัวอย่างความสัมพันธ์แบบ Association

- Aggregation

  -เป็นการบ่งบอกถึงความสัมพันธ์ที่รวมกัน(Whole-Part)

  -สัญลักษณ์ในการแสดงความสัมพันธ์เป็นเส้นตรงทึบ โดยที่ปลายด้านหนึ่งเป็นสัญลักษณ์ diamond แบบโปร่ง

  -ถ้าสัญลักษณ์ diamond แบบโปร่ง อยู่ติดที่ฝั่งของคลาสใดแสดงว่าคลาสนั้นมีสถานะที่เป็นเจ้าของอีกคลาสหนึ่งที่มีความสัมพันธ์ด้วยกันอยู่ ในรูปที่ ก.7 เป็นการแสดงตัวอย่างที่แสดงความสัมพันธ์แบบ Aggregation



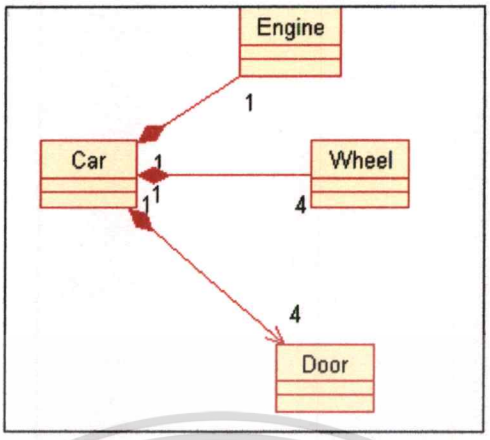
รูปที่ ก.7 แสดงตัวอย่างที่แสดงความสัมพันธ์แบบ Aggregation

- Composition

  -เป็นการบ่งบอกความสัมพันธ์ที่มีลักษณะรวมกัน(Whole-Part)

  -ความแตกต่างระหว่าง Aggregation และ Composition คือความสัมพันธ์แบบ Composition เป็นการระบุว่าคลาสที่เป็น Part ไม่สามารถมีหรืออยู่ได้ถ้าปราศจากคลาสที่เป็น Whole ในขณะที่คลาสที่เป็น Part สามารถมีได้ถ้าปราศจากคลาสที่เป็น Whole ในความสัมพันธ์แบบ Aggregation

  -สัญลักษณ์ที่ใช้แทนความสัมพันธ์นี้ คือสัญลักษณ์ diamond แบบทึบดังรูปที่ ก.8 เป็นการแสดงตัวอย่างของความสัมพันธ์ Composition ที่แสดงถึงคลาสรถยนต์ที่ รถ 1 คันมีส่วนประกอบอะไรบ้าง(เช่น ล้อ, เครื่องยนต์, และประตู เป็นต้น) ซึ่งได้ไม่มีคลาสรถยนต์จะไม่สามารถมีส่วนประกอบของคลาสอื่นๆเกิดขึ้นได้



รูปที่ ก.8 แสดงตัวอย่างของความสัมพันธ์แบบ Composition

- Dependency

  - เป็นความสัมพันธ์ที่บ่งบอกถึงความสัมพันธ์ซึ่งกันและกัน

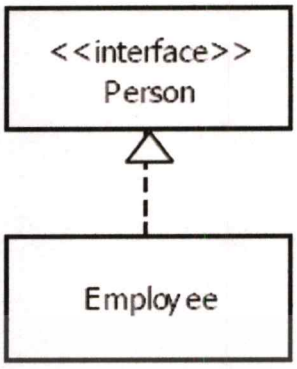
  -การใช้สัญลักษณ์ของความสัมพันธ์แบบนี้ คือเส้นประพร้อมหัวลูกศรชี้ไปยังคลาสที่เป็นให้พึ่งพิง ดังรูปที่ ก.9 ที่แสดงความสัมพันธ์แบบ Dependency โดยสามารถกล่าวได้ว่าถ้าคลาส Independent class มีการเปลี่ยนแปลงใดๆก็ตาม คลาส Dependent class จะมีผลกระทบด้วย

รูปที่ ก.9 แสดงความสัมพันธ์แบบพึ่งพิง

- Realization

  -เป็นการอธิบายความสัมพันธ์ระหว่างสองสิ่ง โดยที่จะมีสิ่งหนึ่งทำหน้าที่ในการดำเนินการตามโอเปอร์เรชั่น

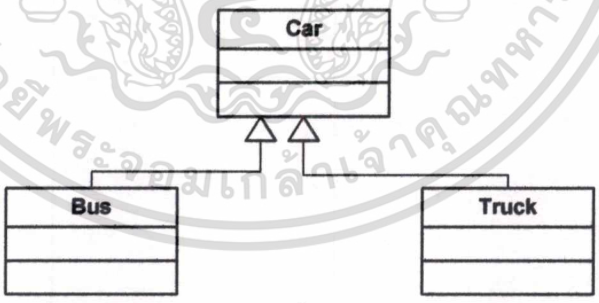
  -การแสดงความสัมพันธ์นี้มีสัญลักษณ์ดังรูปที่ ก.10 ที่เป็นการแสดงความสัมพันธ์แบบ Realization โดยที่ความสัมพันธ์นี้เป็นการเชื่อมความสัมพันธ์ระหว่างคลาสกับอินเตอร์เฟส



รูปที่ ก.10 แสดงความสัมพันธ์แบบ Realization

- Generalization

  - เป็นลักษณะของการสืบทอด มีความสัมพันธ์ระหว่างซูเปอร์คลาส(Super class หรือคลาสแม่) กับ ซับคลาส(SubClass หรือคลาสลูก) โดยคลาสลูกจะได้รับการถ่ายทอดคุณสมบัติและพฤติกรรมจากคลาสแม่
  -สามารถเรียกความสัมพันธ์นี้ว่าเป็นความสัมพันธ์แบบ “is-a”
  -สำหรับสัญลักษณ์ที่ใช้แทนความสัมพันธ์นี้ คือสัญลักษณ์ดังรูปที่ ก.11 ที่แสดงตัวอย่างของคลาสที่มีความสัมพันธ์แบบ Generalization



รูปที่ ก.11 แสดงตัวอย่างคลาสไดอะแกรมที่มีความสัมพันธ์แบบ Generalization

ภาคผนวก ข

JavaCC (Java Complier Complier)

การวิเคราะห์ซอร์สโค้ดภาษาจาวา เป็นการนำข้อมูลเหล่านั้นมาทำการแยกออกเป็นส่วนๆ (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 นี้ในภาษาจาวาได้จัดทำคลาสที่เป็นมาตรฐานที่บรรจุไว้ในภาษาจาวา 2 คลาส คือ StringTokenizer (ใช้กับอ๊อบเจทที่เป็นสตริง) และ StreamTokenizer(ใช้กับอ๊อบเจทที่เป็น InputStream) นอกจากนี้ยังมีอีกแนวทางหนึ่งที่มีความสามารถในการช่วยทำ Lexical Analysis คือการใช้พาร์เซอร์เจนเนอเรเตอร์ที่มีความสามารถในการจำแนก Lexical Analyzer ได้ด้วย โดยในโครงงานนี้ได้นำพาร์เซอร์เจนเนอเรเตอร์ที่มีชื่อว่า JavaCC มาประยุกต์ใช้ในการวิเคราะห์โค้ดจาวา

ในโครงงานนี้ได้นำไฟล์ที่มีชื่อว่า java1.4.jj (โดย .jj เป็นรูปแบบของไฟล์ที่นำมาใช้พาร์เซอร์เจนเนอเรเตอร์) มาประยุกต์ใช้ การนำ java1.4.jj มาช่วยในการทำงานที่ทำการวิเคราะห์โค้ดภาษาจาวา โดยได้คงส่วนของ Lexical Specification ไว้ นอกนั้นได้ทำการปรับปรุงเปลี่ยนแปลงให้มีความเหมาะสมกับการวิเคราะห์ซอร์สโค้ดภาษาจาวาเพื่อสร้างเอกสารทางเอ็กซ์เอ็มไอสำหรับคลาสไดอะแกรม

รูปแบบโครงสร้างของ JavaCC มีลักษณะโครงสร้างดังรูปที่ ข.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 ข.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 ข.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 แล้วต่อจากนั้นต้องมาดำเนินการจำแนกไฟล์ต่างๆออกจากไฟล์ .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

ภาคผนวก ค

รูปแบบไฟล์จาวาและเอกสารเอ็กซ์เอ็มไอ

รูปแบบของเอกสารทางเอ็กซ์เอ็มไอที่ได้จากการสร้างโดยเครื่องมือ JMU ที่มาจากการแมพ โค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรมในหัวข้อที่ 4.2 มีรายละเอียดดังต่อไปนี้

รูปแบบไฟล์จาวา

ไฟล์จาวาที่นำมาทดลองนี้ได้นำมาจาก

ATM.java

```
1   // ATM.java
2   // Represents an automated teller machine
3
4   public class ATM
5   {
6      private boolean userAuthenticated; // whether user is authenticated
7      private int currentAccountNumber; // current user's account number
8      private Screen screen; // ATM's screen
9      private Keypad keypad; // ATM's keypad
10     private CashDispenser cashDispenser; // ATM's cash dispenser
11     private DepositSlot depositSlot; // ATM's deposit slot
12     private BankDatabase bankDatabase; // account information database
13
14     // constants corresponding to main menu options
15     private static final int BALANCE_INQUIRY = 1;
16     private static final int WITHDRAWAL = 2;
17     private static final int DEPOSIT = 3;
18     private static final int EXIT = 4;
19
20     // no-argument ATM constructor initializes instance variables
21     public ATM( )
22     {
23        userAuthenticated = false; // user is not authenticated to start
24        currentAccountNumber = 0; // no current account number to start
25        screen = new Screen( ); // create screen
26        keypad = new Keypad( ); // create keypad
27        cashDispenser = new CashDispenser( ); // create cash dispenser
28        depositSlot = new DepositSlot( ); // create deposit slot
29        bankDatabase = new BankDatabase( ); // create acct info database
30     } // end no-argument ATM constructor
```

```
31
32    // start ATM
33    public void run( )
34    {
35       // welcome and authenticate user; perform transactions
36       while ( true )
37       {
38          // loop while user is not yet authenticated
39          while ( !userAuthenticated )
40          {
41             screen.displayMessageLine( "\nWelcome!" );
42             authenticateUser( ); // authenticate user
43          } // end while
```

```
44
45          performTransactions( ); // user is now authenticated
46          userAuthenticated = false; // reset before next ATM session
47          currentAccountNumber = 0; // reset before next ATM session
48          screen.displayMessageLine( "\nThank you! Goodbye!" );
49       } // end while
50    } // end method run
51
52    // attempts to authenticate user against database
53    private void authenticateUser( )
54    {
55       screen.displayMessage( "\nPlease enter your account number: " );
56       int accountNumber = keypad.getInput( ); // input account number
57       screen.displayMessage( "\nEnter your PIN: " ); // prompt for PIN
58       int pin = keypad.getInput( ); // input PIN
59
60       // set userAuthenticated to boolean value returned by database
61       userAuthenticated =
62          bankDatabase.authenticateUser( accountNumber, pin );
63
64       // check whether authentication succeeded
65       if ( userAuthenticated )
66       {
67          currentAccountNumber = accountNumber; // save user's account #
68       } // end if
69       else
70          screen.displayMessageLine(
71             "Invalid account number or PIN. Please try again." );
72    } // end method authenticateUser
73
74    // display the main menu and perform transactions
75    private void performTransactions( )
```

```
76  {
77     // local variable to store transaction currently being processed
78     Transaction currentTransaction = null;
79
80     boolean userExited = false; // user has not chosen to exit
81
82     // loop while user has not chosen option to exit system
83     while ( !userExited )
84     {
85        // show main menu and get user selection
86        int mainMenuSelection = displayMainMenu( );
87
88        // decide how to proceed based on user's menu selection
89        switch ( mainMenuSelection )
90        {
91           // user chose to perform one of three transaction types
92           case BALANCE_INQUIRY:
93           case WITHDRAWAL:
94           case DEPOSIT:
95
96              // initialize as new object of chosen type
97              currentTransaction =
98                 createTransaction( mainMenuSelection );
99
100             currentTransaction.execute( ); // execute transaction
101             break;
102          case EXIT: // user chose to terminate session
103             screen.displayMessageLine( "\nExiting the system..." );
104             userExited = true; // this ATM session should end
105             break;
106          default: // user did not enter an integer from 1-4
107             screen.displayMessageLine(
```

```
108                "\nYou did not enter a valid selection. Try again." );
109             break;
110          } // end switch
111       } // end while
112    } // end method performTransactions
113
114    // display the main menu and return an input selection
115    private int displayMainMenu( )
116    {
117       screen.displayMessageLine( "\nMain menu:" );
118       screen.displayMessageLine( "1 - View my balance" );
119       screen.displayMessageLine( "2 - Withdraw cash" );
120       screen.displayMessageLine( "3 - Deposit funds" );
121       screen.displayMessageLine( "4 - Exit\n" );
122       screen.displayMessage( "Enter a choice: " );
123       return keypad.getInput( ); // return user's selection
124    } // end method displayMainMenu
125
126    // return object of specified Transaction subclass
127    private Transaction createTransaction( int type )
128    {
129       Transaction temp = null; // temporary Transaction variable
130
131       // determine which type of Transaction to create
132       switch ( type )
133       {
134          case BALANCE_INQUIRY: // create new BalanceInquiry transaction
135             temp = new BalanceInquiry(
136                currentAccountNumber, screen, bankDatabase );
137             break;
138          case WITHDRAWAL: // create new Withdrawal transaction
139             temp = new Withdrawal( currentAccountNumber, screen,
140                bankDatabase, keypad, cashDispenser );
141             break;
142          case DEPOSIT: // create new Deposit transaction
143             temp = new Deposit( currentAccountNumber, screen,
144                bankDatabase, keypad, depositSlot );
145             break;
146       } // end switch
147
148       return temp; // return the newly created object
149    } // end method createTransaction
150 } // end class ATM
```

ATMCaseStudy.java

```
1  // ATMCaseStudy.java
2  // Driver program for the ATM case study
3
4  public class ATMCaseStudy
5  {
6    // main method creates and runs the ATM
7    public static void main( String[] args )
8    {
9      ATM theATM = new ATM( );
10     theATM.run( );
11   } // end main
12 } // end class ATMCaseStudy
```

BalanceInquiry.java

```
1  // BalanceInquiry.java
2  // Represents a balance inquiry ATM transaction
3
4  public class BalanceInquiry extends Transaction
5  {
6    // BalanceInquiry constructor
7    public BalanceInquiry( int userAccountNumber, Screen atmScreen,
8      BankDatabase atmBankDatabase )
9    {
10     super( userAccountNumber, atmScreen, atmBankDatabase );
11   } // end BalanceInquiry constructor
12
13   // performs the transaction
14   public void execute( )
15   {
16     // get references to bank database and screen
17     BankDatabase bankDatabase = getBankDatabase( );
18     Screen screen = getScreen( );
19
20     // get the available balance for the account involved
21     double availableBalance =
22       bankDatabase.getAvailableBalance( getAccountNumber( ) );
23
24     // get the total balance for the account involved
25     double totalBalance =
26       bankDatabase.getTotalBalance( getAccountNumber( ) );
27
28     // display the balance information on the screen
29     screen.displayMessageLine( "\nBalance Information:" );
30     screen.displayMessage( " - Available balance: " );
31     screen.displayDollarAmount( availableBalance );
32     screen.displayMessage( "\n - Total balance:     " );
33     screen.displayDollarAmount( totalBalance );
34     screen.displayMessageLine( "" );
35   } // end method execute
36 } // end class BalanceInquiry
```

CashDispenser.java

```
1  // CashDispenser.java
2  // Represents the cash dispenser of the ATM
3
4  public class CashDispenser
5  {
6     // the default initial number of bills in the cash dispenser
7     private final static int INITIAL_COUNT = 500;
8     private int count; // number of $20 bills remaining
9
10    // no-argument CashDispenser constructor initializes count to default
11    public CashDispenser( )
12    {
13       count = INITIAL_COUNT; // set count attribute to default
14    } // end CashDispenser constructor
15
16    // simulates dispensing of specified amount of cash
17    public void dispenseCash( int amount )
18    {
19       int billsRequired = amount / 20; // number of $20 bills required
20       count -= billsRequired; // update the count of bills
21    } // end method dispenseCash
22
23    // indicates whether cash dispenser can dispense desired amount
24    public boolean isSufficientCashAvailable( int amount )
25    {
26       int billsRequired = amount / 20; // number of $20 bills required
27
28       if ( count >= billsRequired )
29          return true; // enough bills available
30       else
31          return false; // not enough bills available
32    } // end method isSufficientCashAvailable
33 } // end class CashDispenser
```

Deposit.java

```
1  // Deposit.java
2  // Represents a deposit ATM transaction
3
4  public class Deposit extends Transaction
5  {
6     private double amount; // amount to deposit
7     private Keypad keypad; // reference to keypad
8     private DepositSlot depositSlot; // reference to deposit slot
9     private final static int CANCELED = 0; // constant for cancel option
10
11    // Deposit constructor
12    public Deposit( int userAccountNumber, Screen atmScreen,
13       BankDatabase atmBankDatabase, Keypad atmKeypad,
14       DepositSlot atmDepositSlot )
15    {
```

```
16        // initialize superclass variables
17        super( userAccountNumber, atmScreen, atmBankDatabase );
18
19        // initialize references to keypad and deposit slot
20        keypad = atmKeypad;
21        depositSlot = atmDepositSlot;
22     } // end Deposit constructor
23
24     // perform transaction
25     public void execute( )
26     {
27        BankDatabase bankDatabase = getBankDatabase( ); // get reference
28        Screen screen = getScreen( ); // get reference
29
30        amount = promptForDepositAmount( ); // get deposit amount from user
31
32        // check whether user entered a deposit amount or canceled
33        if ( amount != CANCELED )
34        {
35           // request deposit envelope containing specified amount
36           screen.displayMessage(
37              "\nPlease insert a deposit envelope containing " );
38           screen.displayDollarAmount( amount );
39           screen.displayMessageLine( "." );
40
41           // receive deposit envelope
42           boolean envelopeReceived = depositSlot.isEnvelopeReceived( );
43
44           // check whether deposit envelope was received
45           if ( envelopeReceived )
46           {
47              screen.displayMessageLine( "\nYour envelope has been " +
48                 "received.\nNOTE: The money just deposited will not " +
49                 "be available until we verify the amount of any " +
50                 "enclosed cash and your checks clear." );
51
52              // credit account to reflect the deposit
53              bankDatabase.credit( getAccountNumber( ), amount );
54           } // end if
55           else // deposit envelope not received
56           {
57              screen.displayMessageLine( "\nYou did not insert an " +
58                 "envelope, so the ATM has canceled your transaction." );
59           } // end else
60        } // end if
61        else // user canceled instead of entering amount
62        {
63           screen.displayMessageLine( "\nCanceling transaction..." );
64        } // end else
65     } // end method execute
66
67     // prompt user to enter a deposit amount in cents
68     private double promptForDepositAmount( )
```

```
69    {
70       Screen screen = getScreen( ); // get reference to screen
71
72       // display the prompt
73       screen.displayMessage( "\nPlease enter a deposit amount in " +
74          "CENTS (or 0 to cancel): " );
75       int input = keypad.getInput( ); // receive input of deposit amount
76
77       // check whether the user canceled or entered a valid amount
78       if ( input == CANCELED )
79          return CANCELED;
80       else
81       {
82          return ( double ) input / 100; // return dollar amount
83       } // end else
84    } // end method promptForDepositAmount
85 } // end class Deposit
```

DepositSlot.java

```
1  // DepositSlot.java
2  // Represents the deposit slot of the ATM
3
4  public class DepositSlot
5  {
6     // indicates whether envelope was received (always returns true,
7     // because this is only a software simulation of a real deposit slot)
8     public boolean isEnvelopeReceived( )
9     {
10       return true; // deposit envelope was received
11    } // end method isEnvelopeReceived
12 } // end class DepositSlot
13
```

Transaction.java

```
1  // Transaction.java
2  // Abstract superclass Transaction represents an ATM transaction
3
4  public abstract class Transaction
5  {
6     private int accountNumber; // indicates account involved
7     private Screen screen; // ATM's screen
8     private BankDatabase bankDatabase; // account info database
9
10    // Transaction constructor invoked by subclasses using super()
11    public Transaction( int userAccountNumber, Screen atmScreen,
12       BankDatabase atmBankDatabase )
13    {
14       accountNumber = userAccountNumber;
15       screen = atmScreen;
16       bankDatabase = atmBankDatabase;
17    } // end Transaction constructor
18
```

```
19    // return account number
20    public int getAccountNumber( )
21    {
22       return accountNumber;
23    } // end method getAccountNumber
24
25    // return reference to screen
26    public Screen getScreen( )
27    {
28       return screen;
29    } // end method getScreen
30
31    // return reference to bank database
32    public BankDatabase getBankDatabase( )
33    {
34       return bankDatabase;
35    } // end method getBankDatabase
36
37    // perform the transaction (overridden by each subclass)
38    abstract public void execute( );
39 } // end class Transaction
```

Withdrawal.java

```
1  // Withdrawal.java
2  // Represents a withdrawal ATM transaction
3
4  public class Withdrawal extends Transaction
5  {
6     private int amount; // amount to withdraw
7     private Keypad keypad; // reference to keypad
8     private CashDispenser cashDispenser; // reference to cash dispenser
9
10    // constant corresponding to menu option to cancel
11    private final static int CANCELED = 6;
12
13    // Withdrawal constructor
14    public Withdrawal( int userAccountNumber, Screen atmScreen,
15       BankDatabase atmBankDatabase, Keypad atmKeypad,
16       CashDispenser atmCashDispenser )
17    {
18       // initialize superclass variables
19       super( userAccountNumber, atmScreen, atmBankDatabase );
20
21       // initialize references to keypad and cash dispenser
22       keypad = atmKeypad;
23       cashDispenser = atmCashDispenser;
24    } // end Withdrawal constructor
25
26    // perform transaction
27    public void execute( )
28    {
29       boolean cashDispensed = false; // cash was not dispensed yet
30       double availableBalance; // amount available for withdrawal
31
```

```
32    // get references to bank database and screen
33    BankDatabase bankDatabase = getBankDatabase( );
34    Screen screen = getScreen( );
35
36    // loop until cash is dispensed or the user cancels
37    do
38    {
39       // obtain a chosen withdrawal amount from the user
40       amount = displayMenuOfAmounts( );
41
42       // check whether user chose a withdrawal amount or canceled
43       if ( amount != CANCELED )
44       {
45          // get available balance of account involved
46          availableBalance =
47             bankDatabase.getAvailableBalance( getAccountNumber( ) );
48
49          // check whether the user has enough money in the account
50          if ( amount <= availableBalance )
51          {
52             // check whether the cash dispenser has enough money
53             if ( cashDispenser.isSufficientCashAvailable( amount ) )
54             {
55                // update the account involved to reflect withdrawal
56                bankDatabase.debit( getAccountNumber( ), amount );
57
58                cashDispenser.dispenseCash( amount ); // dispense cash
59                cashDispensed = true; // cash was dispensed
60
61                // instruct user to take cash
62                screen.displayMessageLine(
63                   "\nPlease take your cash now." );
64             } // end if
65             else // cash dispenser does not have enough cash
66                screen.displayMessageLine(
67                   "\nInsufficient cash available in the ATM." +
68                   "\n\nPlease choose a smaller amount." );
69          } // end if
70          else // not enough money available in user's account
71          {
72             screen.displayMessageLine(
73                "\nInsufficient funds in your account." +
74                "\n\nPlease choose a smaller amount." );
75          } // end else
76       } // end if
77       else // user chose cancel menu option
78       {
79          screen.displayMessageLine( "\nCanceling transaction..." );
80          return; // return to main menu because user canceled
81       } // end else
82    } while ( !cashDispensed );
83
```

```
83
84    } // end method execute
85
86    // display a menu of withdrawal amounts and the option to cancel;
87    // return the chosen amount or 0 if the user chooses to cancel
88    private int displayMenuOfAmounts( )
89    {
90       int userChoice = 0; // local variable to store return value
91
92       Screen screen = getScreen( ); // get screen reference
93
94       // array of amounts to correspond to menu numbers
95       int amounts[] = { 0, 20, 40, 60, 100, 200 };
96
97       // loop while no valid choice has been made
98       while ( userChoice == 0 )
99       {
100         // display the menu
101         screen.displayMessageLine( "\nWithdrawal Menu:" );
102         screen.displayMessageLine( "1 - $20" );
103         screen.displayMessageLine( "2 - $40" );
104         screen.displayMessageLine( "3 - $60" );
105         screen.displayMessageLine( "4 - $100" );
106         screen.displayMessageLine( "5 - $200" );
107         screen.displayMessageLine( "6 - Cancel transaction" );
108         screen.displayMessage( "\nChoose a withdrawal amount: " );
109
110         int input = keypad.getInput( ); // get user input through keypad
111
112         // determine how to proceed based on the input value
113         switch ( input )
114         {
115            case 1: // if the user chose a withdrawal amount
116            case 2: // (i.e., chose option 1, 2, 3, 4 or 5), return the
117            case 3: // corresponding amount from amounts array
118            case 4:
119            case 5:
120               userChoice = amounts[ input ]; // save user's choice
121               break;
122            case CANCELED: // the user chose to cancel
123               userChoice = CANCELED; // save user's choice
124               break;
125            default: // the user did not enter a value from 1-6
126               screen.displayMessageLine(
127                  "\nInvalid selection. Try again." );
128         } // end switch
129      } // end while
130
131      return userChoice; // return withdrawal amount or CANCELED
132   } // end method displayMenuOfAmounts
133 } // end class Withdrawal
```

รูปแบบไฟล์เอกสารเอ็กซ์เอ็มไอที่สร้างโดย JMU

```
<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.2" xmlns:XMI="http://www.omg.org/XMI" xmlns:xmi="http://www.omg.org/XMI" xmlns:UML=
"http://www.omg.org/UML" xmi.timestamp="Sun Mar 19 23:44:19 MHT 2006">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>JMU</XMI.exporter>
      <XMI.exporterVersion>1.0</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
  </XMI.header>
  <XMI.content>
  <UML:Model xmi.id="G.0" name="test1 " visibility="public" isRoot="false" isLeaf="false" isAbstract="false">
  <UML:Namespace.ownedElement>
  <!--*****************************************************************************-->
    <UML:Class isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="ATM" namespace="G.0"
      visibility="public" xmi.id="C.1">
      <UML:Classifier.feature>
        <UML:Attribute isSpecification="false"
          name="userAuthenticated" ownerScope="instance"
          type="G.2" visibility="private" xmi.id="A.1"/>
        <UML:Attribute isSpecification="false"
          name="currentAccountNumber" ownerScope="instance"
          type="G.3" visibility="private" xmi.id="A.2"/>
        <UML:Attribute isSpecification="false" name="screen"
          ownerScope="instance" type="G.4" visibility="private" xmi.id="A.3"/>
        <UML:Attribute isSpecification="false" name="keypad"
          ownerScope="instance" type="G.8" visibility="private" xmi.id="A.4"/>
        <UML:Attribute isSpecification="false" name="cashDispenser"
          ownerScope="instance" type="G.12" visibility="private" xmi.id="A.5"/>
        <UML:Attribute isSpecification="false" name="depositSlot"
          ownerScope="instance" type="G.16" visibility="private" xmi.id="A.6"/>
        <UML:Attribute isSpecification="false" name="bankDatabase"
          ownerScope="instance" type="G.20" visibility="private" xmi.id="A.7"/>
        <UML:Attribute isSpecification="false"
          name="BALANCE_INQUIRY" ownerScope="classifier"
          type="G.3" visibility="private" xmi.id="A.8">
          <UML:Attribute.initialValue>
            <UML:Expression body="1"/>
          </UML:Attribute.initialValue>
        </UML:Attribute>
        <UML:Attribute isSpecification="false" name="WITHDRAWAL"
          ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.9">
          <UML:Attribute.initialValue>
            <UML:Expression body="2"/>
          </UML:Attribute.initialValue>
        </UML:Attribute>
        <UML:Attribute isSpecification="false" name="DEPOSIT"
          ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.10">
          <UML:Attribute.initialValue>
            <UML:Expression body="3"/>
          </UML:Attribute.initialValue>
        </UML:Attribute>
        <UML:Attribute isSpecification="false" name="EXIT"
          ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.11">
          <UML:Attribute.initialValue>
            <UML:Expression body="4"/>
          </UML:Attribute.initialValue>
        </UML:Attribute>
        <UML:Operation isAbstract="false" name="ATM"
          ownerScope="instance" visibility="public" xmi.id="M.0"/>
        <UML:Operation isAbstract="false" name="run"
          ownerScope="instance" visibility="public" xmi.id="M.1"/>
        <UML:Operation isAbstract="false" name="authenticateUser"
          ownerScope="instance" visibility="private" xmi.id="M.2"/>
        <UML:Operation isAbstract="false" name="performTransactions"
          ownerScope="instance" visibility="private" xmi.id="M.3"/>
```

```
67          <UML:Operation isAbstract="false" name="displayMainMenu"
68            ownerScope="instance" visibility="private" xmi.id="M.4">
69            <UML:BehavioralFeature.parameter>
70              <UML:Parameter kind="return"
71                name="displayMainMenuReturn" type="G.3"
72                visibility="public" xmi.id="M.5"/>
73            </UML:BehavioralFeature.parameter>
74          </UML:Operation>
75          <UML:Operation isAbstract="false" name="createTransaction"
76            ownerScope="instance" visibility="private" xmi.id="M.6">
77            <UML:BehavioralFeature.parameter>
78              <UML:Parameter kind="return" name="nullReturn"
79                type="G.39" visibility="public" xmi.id="M.8"/>
80            </UML:BehavioralFeature.parameter>
81          </UML:Operation>
82        </UML:Classifier.feature>
83      </UML:Class>
84      <UML:DataType isSpecification="false" name="boolean" xmi.id="G.2"/>
85      <UML:DataType isSpecification="false" name="int" xmi.id="G.3"/>
86      <UML:DataType isSpecification="false" name="Screen" xmi.id="G.4"/>
87      <UML:DataType isSpecification="false" name="Keypad" xmi.id="G.8"/>
88      <UML:DataType isSpecification="false" name="CashDispenser" xmi.id="G.12"/>
89      <UML:DataType isSpecification="false" name="DepositSlot" xmi.id="G.16"/>
90      <UML:DataType isSpecification="false" name="BankDatabase" xmi.id="G.20"/>
91      <UML:DataType isSpecification="false" name="Transaction" xmi.id="G.39"/>
92    <!--*************************************************************-->
93      <UML:Class isAbstract="false" isLeaf="false" isRoot="false"
94        isSpecification="false" name="ATMCaseStudy" namespace="G.0"
95        visibility="public" xmi.id="C.2">
96        <UML:Classifier.feature>
97          <UML:Operation isAbstract="false" name="main"
98            ownerScope="classifier" visibility="public" xmi.id="M.9">
99            <UML:BehavioralFeature.parameter>
100             <UML:Parameter kind="inout" name="args" type="G.49"
101               visibility="public" xmi.id="M.10"/>
102           </UML:BehavioralFeature.parameter>
103         </UML:Operation>
104       </UML:Classifier.feature>
105     </UML:Class>
106     <UML:DataType isSpecification="false" name="String[]" xmi.id="G.49"/>
107   <!--*************************************************************-->
108     <UML:Class generalization="G.53" isAbstract="false" isLeaf="false"
109       isRoot="false" isSpecification="false" name="BalanceInquiry"
110       namespace="G.0" visibility="public" xmi.id="C.3">
111       <UML:Namespace.ownedElement>
112         <UML:Generalization child="C.3" name="" parent="C.4" xmi.id="G.53"/>
113       </UML:Namespace.ownedElement>
114       <UML:Classifier.feature>
115         <UML:Operation isAbstract="false" name="BalanceInquiry"
116           ownerScope="instance" visibility="public" xmi.id="M.11">
117           <UML:BehavioralFeature.parameter>
118             <UML:Parameter kind="inout" name="userAccountNumber"
119               type="G.3" visibility="public" xmi.id="M.12"/>
120             <UML:Parameter kind="inout" name="atmScreen"
121               type="G.4" visibility="public" xmi.id="M.13"/>
122             <UML:Parameter kind="inout" name="atmBankDatabase"
123               type="G.20" visibility="public" xmi.id="M.14"/>
124           </UML:BehavioralFeature.parameter>
125         </UML:Operation>
126         <UML:Operation isAbstract="false" name="execute"
127           ownerScope="instance" visibility="public" xmi.id="M.15"/>
128       </UML:Classifier.feature>
129     </UML:Class>
130   <!--*************************************************************-->
```

```
<UML:Class isAbstract="false" isLeaf="false" isRoot="false"
  isSpecification="false" name="CashDispenser" namespace="G.0"
  visibility="public" xmi.id="C.5">
  <UML:Classifier.feature>
    <UML:Attribute isSpecification="false" name="INITIAL_COUNT"
      ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.12">
      <UML:Attribute.initialValue>
        <UML:Expression body="500"/>
      </UML:Attribute.initialValue>
    </UML:Attribute>
    <UML:Attribute isSpecification="false" name="count"
      ownerScope="instance" type="G.3" visibility="private" xmi.id="A.13"/>
    <UML:Operation isAbstract="false" name="CashDispenser"
      ownerScope="instance" visibility="public" xmi.id="M.16"/>
    <UML:Operation isAbstract="false" name="dispenseCash"
      ownerScope="instance" visibility="public" xmi.id="M.17">
      <UML:BehavioralFeature.parameter>
        <UML:Parameter kind="inout" name="amount" type="G.3"
          visibility="public" xmi.id="M.18"/>
      </UML:BehavioralFeature.parameter>
    </UML:Operation>
    <UML:Operation isAbstract="false"
      name="isSufficientCashAvailable" ownerScope="instance"
      visibility="public" xmi.id="M.19">
      <UML:BehavioralFeature.parameter>
        <UML:Parameter kind="return" name="nullReturn"
          type="G.2" visibility="public" xmi.id="M.21"/>
      </UML:BehavioralFeature.parameter>
    </UML:Operation>
  </UML:Classifier.feature>
</UML:Class>
<!--**********************************************************************-->
<UML:Class generalization="G.60" isAbstract="false" isLeaf="false"
  isRoot="false" isSpecification="false" name="Deposit"
  namespace="G.0" visibility="public" xmi.id="C.6">
  <UML:Namespace.ownedElement>
    <UML:Generalization child="C.6" name="" parent="C.4" xmi.id="G.60"/>
  </UML:Namespace.ownedElement>
  <UML:Classifier.feature>
    <UML:Attribute isSpecification="false" name="amount"
      ownerScope="instance" type="G.61" visibility="private" xmi.id="A.14"/>
    <UML:Attribute isSpecification="false" name="keypad"
      ownerScope="instance" type="G.8" visibility="private" xmi.id="A.15"/>
    <UML:Attribute isSpecification="false" name="depositSlot"
      ownerScope="instance" type="G.16" visibility="private" xmi.id="A.16"/>
    <UML:Attribute isSpecification="false" name="CANCELED"
      ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.17">
      <UML:Attribute.initialValue>
        <UML:Expression body="0"/>
      </UML:Attribute.initialValue>
    </UML:Attribute>
    <UML:Operation isAbstract="false" name="Deposit"
      ownerScope="instance" visibility="public" xmi.id="M.22">
      <UML:BehavioralFeature.parameter>
        <UML:Parameter kind="inout" name="userAccountNumber"
          type="G.3" visibility="public" xmi.id="M.23"/>
        <UML:Parameter kind="inout" name="atmScreen"
          type="G.4" visibility="public" xmi.id="M.24"/>
        <UML:Parameter kind="inout" name="atmBankDatabase"
          type="G.20" visibility="public" xmi.id="M.25"/>
        <UML:Parameter kind="inout" name="atmKeypad"
          type="G.8" visibility="public" xmi.id="M.26"/>
        <UML:Parameter kind="inout" name="atmDepositSlot"
          type="G.16" visibility="public" xmi.id="M.27"/>
```

```
195                </UML:BehavioralFeature.parameter>
196              </UML:Operation>
197              <UML:Operation isAbstract="false" name="execute"
198                ownerScope="instance" visibility="public" xmi.id="M.28"/>
199              <UML:Operation isAbstract="false"
200                name="promptForDepositAmount" ownerScope="instance"
201                visibility="private" xmi.id="M.29">
202                <UML:BehavioralFeature.parameter>
203                  <UML:Parameter kind="return"
204                    name="promptForDepositAmountReturn" type="G.61"
205                    visibility="public" xmi.id="M.30"/>
206                </UML:BehavioralFeature.parameter>
207              </UML:Operation>
208            </UML:Classifier.feature>
209          </UML:Class>
210          <UML:DataType isSpecification="false" name="double" xmi.id="G.61"/>
211        <!--*************************************************************************-->
212          <UML:Class isAbstract="false" isLeaf="false" isRoot="false"
213            isSpecification="false" name="DepositSlot" namespace="G.0"
214            visibility="public" xmi.id="C.7">
215            <UML:Classifier.feature>
216              <UML:Operation isAbstract="false" name="isEnvelopeReceived"
217                ownerScope="instance" visibility="public" xmi.id="M.31">
218                <UML:BehavioralFeature.parameter>
219                  <UML:Parameter kind="return"
220                    name="isEnvelopeReceivedReturn" type="G.2"
221                    visibility="public" xmi.id="M.32"/>
222                </UML:BehavioralFeature.parameter>
223              </UML:Operation>
224            </UML:Classifier.feature>
225          </UML:Class>
226        <!--*************************************************************************-->
227          <UML:Class isAbstract="true" isLeaf="false" isRoot="false"
228            isSpecification="false" name="Transaction" namespace="G.0"
229            specialization="G.53 , G.53" visibility="public" xmi.id="C.4">
230            <UML:Classifier.feature>
231              <UML:Attribute isSpecification="false" name="accountNumber"
232                ownerScope="instance" type="G.3" visibility="private" xmi.id="A.18"/>
233              <UML:Attribute isSpecification="false" name="screen"
234                ownerScope="instance" type="G.4" visibility="private" xmi.id="A.19"/>
235              <UML:Attribute isSpecification="false" name="bankDatabase"
236                ownerScope="instance" type="G.20" visibility="private" xmi.id="A.20"/>
237              <UML:Operation isAbstract="false" name="Transaction"
238                ownerScope="instance" visibility="public" xmi.id="M.33">
239                <UML:BehavioralFeature.parameter>
240                  <UML:Parameter kind="inout" name="userAccountNumber"
241                    type="G.3" visibility="public" xmi.id="M.34"/>
242                  <UML:Parameter kind="inout" name="atmScreen"
243                    type="G.4" visibility="public" xmi.id="M.35"/>
244                  <UML:Parameter kind="inout" name="atmBankDatabase"
245                    type="G.20" visibility="public" xmi.id="M.36"/>
246                </UML:BehavioralFeature.parameter>
247              </UML:Operation>
248              <UML:Operation isAbstract="false" name="getAccountNumber"
249                ownerScope="instance" visibility="public" xmi.id="M.37">
250                <UML:BehavioralFeature.parameter>
251                  <UML:Parameter kind="return"
252                    name="getAccountNumberReturn" type="G.3"
253                    visibility="public" xmi.id="M.38"/>
254                </UML:BehavioralFeature.parameter>
255              </UML:Operation>
256              <UML:Operation isAbstract="false" name="getScreen"
257                ownerScope="instance" visibility="public" xmi.id="M.39">
258                <UML:BehavioralFeature.parameter>
```

```
259                 <UML:Parameter kind="return" name="getScreenReturn"
260                   type="G.4" visibility="public" xmi.id="M.40"/>
261               </UML:BehavioralFeature.parameter>
262             </UML:Operation>
263             <UML:Operation isAbstract="false" name="getBankDatabase"
264               ownerScope="instance" visibility="public" xmi.id="M.41">
265               <UML:BehavioralFeature.parameter>
266                 <UML:Parameter kind="return"
267                   name="getBankDatabaseReturn" type="G.20"
268                   visibility="public" xmi.id="M.42"/>
269               </UML:BehavioralFeature.parameter>
270             </UML:Operation>
271             <UML:Operation isAbstract="false" name="execute"
272               ownerScope="instance" visibility="public" xmi.id="M.43"/>
273           </UML:Classifier.feature>
274         </UML:Class>
275       <!--*************************************************************************-->
276         <UML:Class generalization="G.92" isAbstract="false" isLeaf="false"
277           isRoot="false" isSpecification="false" name="Withdrawal"
278           namespace="G.0" visibility="public" xmi.id="C.8">
279           <UML:Namespace.ownedElement>
280             <UML:Generalization child="C.8" name="" parent="C.4" xmi.id="G.92"/>
281           </UML:Namespace.ownedElement>
282           <UML:Classifier.feature>
283             <UML:Attribute isSpecification="false" name="amount"
284               ownerScope="instance" type="G.3" visibility="private" xmi.id="A.21"/>
285             <UML:Attribute isSpecification="false" name="keypad"
286               ownerScope="instance" type="G.8" visibility="private" xmi.id="A.22"/>
287             <UML:Attribute isSpecification="false" name="cashDispenser"
288               ownerScope="instance" type="G.12" visibility="private" xmi.id="A.23"/>
289             <UML:Attribute isSpecification="false" name="CANCELED"
290               ownerScope="classifier" type="G.3" visibility="private" xmi.id="A.24">
291               <UML:Attribute.initialValue>
292                 <UML:Expression body="6"/>
293               </UML:Attribute.initialValue>
294             </UML:Attribute>
295             <UML:Operation isAbstract="false" name="Withdrawal"
296               ownerScope="instance" visibility="public" xmi.id="M.44">
297               <UML:BehavioralFeature.parameter>
298                 <UML:Parameter kind="inout" name="userAccountNumber"
299                   type="G.3" visibility="public" xmi.id="M.45"/>
300                 <UML:Parameter kind="inout" name="atmScreen"
301                   type="G.4" visibility="public" xmi.id="M.46"/>
302                 <UML:Parameter kind="inout" name="atmBankDatabase"
303                   type="G.20" visibility="public" xmi.id="M.47"/>
304                 <UML:Parameter kind="inout" name="atmKeypad"
305                   type="G.8" visibility="public" xmi.id="M.48"/>
306                 <UML:Parameter kind="inout" name="atmCashDispenser"
307                   type="G.12" visibility="public" xmi.id="M.49"/>
308               </UML:BehavioralFeature.parameter>
309             </UML:Operation>
310             <UML:Operation isAbstract="false" name="execute"
311               ownerScope="instance" visibility="public" xmi.id="M.50"/>
312             <UML:Operation isAbstract="false"
313               name="displayMenuOfAmounts" ownerScope="instance"
314               visibility="private" xmi.id="M.51">
315               <UML:BehavioralFeature.parameter>
316                 <UML:Parameter kind="return"
317                   name="displayMenuOfAmountsReturn" type="G.3"
318                   visibility="public" xmi.id="M.52"/>
319               </UML:BehavioralFeature.parameter>
320             </UML:Operation>
321           </UML:Classifier.feature>
322         </UML:Class>
```

```
<!--*****************************Relation*************************************************-->
    <UML:Class name="Screen" xmi.id="C.9"/>
    <UML:Class name="Keypad" xmi.id="C.10"/>
    <UML:Class name="BankDatabase" xmi.id="C.11"/>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.5">
      <UML:Association.connection>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.9" visibility="public" xmi.id="G.6"/>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.1" visibility="public" xmi.id="G.7"/>
      </UML:Association.connection>
    </UML:Association>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.6">
      <UML:Association.connection>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.10" visibility="public" xmi.id="G.7"/>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.1" visibility="public" xmi.id="G.9"/>
      </UML:Association.connection>
    </UML:Association>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.7">
      <UML:Association.connection>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.5" visibility="public" xmi.id="G.9"/>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.1" visibility="public" xmi.id="G.10"/>
      </UML:Association.connection>
    </UML:Association>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.9">
      <UML:Association.connection>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.7" visibility="public" xmi.id="G.10"/>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.1" visibility="public" xmi.id="G.11"/>
      </UML:Association.connection>
    </UML:Association>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.10">
      <UML:Association.connection>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.11" visibility="public" xmi.id="G.11"/>
        <UML:AssociationEnd aggregation="none" isNavigable="false"
          isSpecification="false" name="" ordering="unordering"
          targetScope="instance" type="C.1" visibility="public" xmi.id="G.13"/>
      </UML:Association.connection>
    </UML:Association>
    <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
      isSpecification="false" name="" visibility="public" xmi.id="G.11">
      <UML:Association.connection>
```

```
385             <UML:AssociationEnd aggregation="none" isNavigable="false"
386               isSpecification="false" name="" ordering="unordering"
387               targetScope="instance" type="C.9" visibility="public" xmi.id="G.13"/>
388             <UML:AssociationEnd aggregation="none" isNavigable="false"
389               isSpecification="false" name="" ordering="unordering"
390               targetScope="instance" type="C.1" visibility="public" xmi.id="G.14"/>
391           </UML:Association.connection>
392         </UML:Association>
393         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
394           isSpecification="false" name="" visibility="public" xmi.id="G.13">
395           <UML:Association.connection>
396             <UML:AssociationEnd aggregation="none" isNavigable="false"
397               isSpecification="false" name="" ordering="unordering"
398               targetScope="instance" type="C.10" visibility="public" xmi.id="G.14"/>
399             <UML:AssociationEnd aggregation="none" isNavigable="false"
400               isSpecification="false" name="" ordering="unordering"
401               targetScope="instance" type="C.1" visibility="public" xmi.id="G.15"/>
402           </UML:Association.connection>
403         </UML:Association>
404         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
405           isSpecification="false" name="" visibility="public" xmi.id="G.14">
406           <UML:Association.connection>
407             <UML:AssociationEnd aggregation="none" isNavigable="false"
408               isSpecification="false" name="" ordering="unordering"
409               targetScope="instance" type="C.5" visibility="public" xmi.id="G.15"/>
410             <UML:AssociationEnd aggregation="none" isNavigable="false"
411               isSpecification="false" name="" ordering="unordering"
412               targetScope="instance" type="C.1" visibility="public" xmi.id="G.17"/>
413           </UML:Association.connection>
414         </UML:Association>
415         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
416           isSpecification="false" name="" visibility="public" xmi.id="G.15">
417           <UML:Association.connection>
418             <UML:AssociationEnd aggregation="none" isNavigable="false"
419               isSpecification="false" name="" ordering="unordering"
420               targetScope="instance" type="C.7" visibility="public" xmi.id="G.17"/>
421             <UML:AssociationEnd aggregation="none" isNavigable="false"
422               isSpecification="false" name="" ordering="unordering"
423               targetScope="instance" type="C.1" visibility="public" xmi.id="G.18"/>
424           </UML:Association.connection>
425         </UML:Association>
426         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
427           isSpecification="false" name="" visibility="public" xmi.id="G.17">
428           <UML:Association.connection>
429             <UML:AssociationEnd aggregation="none" isNavigable="false"
430               isSpecification="false" name="" ordering="unordering"
431               targetScope="instance" type="C.11" visibility="public" xmi.id="G.18"/>
432             <UML:AssociationEnd aggregation="none" isNavigable="false"
433               isSpecification="false" name="" ordering="unordering"
434               targetScope="instance" type="C.1" visibility="public" xmi.id="G.19"/>
435           </UML:Association.connection>
436         </UML:Association>
437         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
438           isSpecification="false" name="" visibility="public" xmi.id="G.18">
439           <UML:Association.connection>
440             <UML:AssociationEnd aggregation="none" isNavigable="false"
441               isSpecification="false" name="" ordering="unordering"
442               targetScope="instance" type="C.3" visibility="public" xmi.id="G.19"/>
443             <UML:AssociationEnd aggregation="none" isNavigable="false"
444               isSpecification="false" name="" ordering="unordering"
445               targetScope="instance" type="C.1" visibility="public" xmi.id="G.21"/>
446           </UML:Association.connection>
447         </UML:Association>
448         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
449           isSpecification="false" name="" visibility="public" xmi.id="G.19">
450           <UML:Association.connection>
```

```
451             <UML:AssociationEnd aggregation="none" isNavigable="false"
452               isSpecification="false" name="" ordering="unordering"
453               targetScope="instance" type="C.8" visibility="public" xmi.id="G.21"/>
454             <UML:AssociationEnd aggregation="none" isNavigable="false"
455               isSpecification="false" name="" ordering="unordering"
456               targetScope="instance" type="C.1" visibility="public" xmi.id="G.22"/>
457           </UML:Association.connection>
458         </UML:Association>
459         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
460           isSpecification="false" name="" visibility="public" xmi.id="G.21">
461           <UML:Association.connection>
462             <UML:AssociationEnd aggregation="none" isNavigable="false"
463               isSpecification="false" name="" ordering="unordering"
464               targetScope="instance" type="C.6" visibility="public" xmi.id="G.22"/>
465             <UML:AssociationEnd aggregation="none" isNavigable="false"
466               isSpecification="false" name="" ordering="unordering"
467               targetScope="instance" type="C.1" visibility="public" xmi.id="G.23"/>
468           </UML:Association.connection>
469         </UML:Association>
470         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
471           isSpecification="false" name="" visibility="public" xmi.id="G.22">
472           <UML:Association.connection>
473             <UML:AssociationEnd aggregation="none" isNavigable="false"
474               isSpecification="false" name="" ordering="unordering"
475               targetScope="instance" type="C.1" visibility="public" xmi.id="G.23"/>
476             <UML:AssociationEnd aggregation="none" isNavigable="false"
477               isSpecification="false" name="" ordering="unordering"
478               targetScope="instance" type="C.2" visibility="public" xmi.id="G.24"/>
479           </UML:Association.connection>
480         </UML:Association>
481         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
482           isSpecification="false" name="" visibility="public" xmi.id="G.23">
483           <UML:Association.connection>
484             <UML:AssociationEnd aggregation="none" isNavigable="false"
485               isSpecification="false" name="" ordering="unordering"
486               targetScope="instance" type="C.9" visibility="public" xmi.id="G.24"/>
487             <UML:AssociationEnd aggregation="none" isNavigable="false"
488               isSpecification="false" name="" ordering="unordering"
489               targetScope="instance" type="C.3" visibility="public" xmi.id="G.25"/>
490           </UML:Association.connection>
491         </UML:Association>
492         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
493           isSpecification="false" name="" visibility="public" xmi.id="G.24">
494           <UML:Association.connection>
495             <UML:AssociationEnd aggregation="none" isNavigable="false"
496               isSpecification="false" name="" ordering="unordering"
497               targetScope="instance" type="C.11" visibility="public" xmi.id="G.25"/>
498             <UML:AssociationEnd aggregation="none" isNavigable="false"
499               isSpecification="false" name="" ordering="unordering"
500               targetScope="instance" type="C.3" visibility="public" xmi.id="G.26"/>
501           </UML:Association.connection>
502         </UML:Association>
503         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
504           isSpecification="false" name="" visibility="public" xmi.id="G.25">
505           <UML:Association.connection>
506             <UML:AssociationEnd aggregation="none" isNavigable="false"
507               isSpecification="false" name="" ordering="unordering"
508               targetScope="instance" type="C.10" visibility="public" xmi.id="G.26"/>
509             <UML:AssociationEnd aggregation="none" isNavigable="false"
510               isSpecification="false" name="" ordering="unordering"
```

```
511                 targetScope="instance" type="C.6" visibility="public" xmi.id="G.27"/>
512               </UML:Association.connection>
513             </UML:Association>
514             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
515               isSpecification="false" name="" visibility="public" xmi.id="G.26">
516               <UML:Association.connection>
517                 <UML:AssociationEnd aggregation="none" isNavigable="false"
518                   isSpecification="false" name="" ordering="unordering"
519                   targetScope="instance" type="C.7" visibility="public" xmi.id="G.27"/>
520                 <UML:AssociationEnd aggregation="none" isNavigable="false"
521                   isSpecification="false" name="" ordering="unordering"
522                   targetScope="instance" type="C.6" visibility="public" xmi.id="G.28"/>
523               </UML:Association.connection>
524             </UML:Association>
525             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
526               isSpecification="false" name="" visibility="public" xmi.id="G.27">
527               <UML:Association.connection>
528                 <UML:AssociationEnd aggregation="none" isNavigable="false"
529                   isSpecification="false" name="" ordering="unordering"
530                   targetScope="instance" type="C.9" visibility="public" xmi.id="G.28"/>
531                 <UML:AssociationEnd aggregation="none" isNavigable="false"
532                   isSpecification="false" name="" ordering="unordering"
533                   targetScope="instance" type="C.6" visibility="public" xmi.id="G.29"/>
534               </UML:Association.connection>
535             </UML:Association>
536             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
537               isSpecification="false" name="" visibility="public" xmi.id="G.28">
538               <UML:Association.connection>
539                 <UML:AssociationEnd aggregation="none" isNavigable="false"
540                   isSpecification="false" name="" ordering="unordering"
541                   targetScope="instance" type="C.11" visibility="public" xmi.id="G.29"/>
542                 <UML:AssociationEnd aggregation="none" isNavigable="false"
543                   isSpecification="false" name="" ordering="unordering"
544                   targetScope="instance" type="C.6" visibility="public" xmi.id="G.30"/>
545               </UML:Association.connection>
546             </UML:Association>
547             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
548               isSpecification="false" name="" visibility="public" xmi.id="G.29">
549               <UML:Association.connection>
550                 <UML:AssociationEnd aggregation="none" isNavigable="false"
551                   isSpecification="false" name="" ordering="unordering"
552                   targetScope="instance" type="C.10" visibility="public" xmi.id="G.30"/>
553                 <UML:AssociationEnd aggregation="none" isNavigable="false"
554                   isSpecification="false" name="" ordering="unordering"
555                   targetScope="instance" type="C.6" visibility="public" xmi.id="G.31"/>
556               </UML:Association.connection>
557             </UML:Association>
558             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
559               isSpecification="false" name="" visibility="public" xmi.id="G.30">
560               <UML:Association.connection>
561                 <UML:AssociationEnd aggregation="none" isNavigable="false"
562                   isSpecification="false" name="" ordering="unordering"
563                   targetScope="instance" type="C.7" visibility="public" xmi.id="G.31"/>
564                 <UML:AssociationEnd aggregation="none" isNavigable="false"
565                   isSpecification="false" name="" ordering="unordering"
566                   targetScope="instance" type="C.6" visibility="public" xmi.id="G.32"/>
567               </UML:Association.connection>
568             </UML:Association>
569             <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
570               isSpecification="false" name="" visibility="public" xmi.id="G.31">
571               <UML:Association.connection>
572                 <UML:AssociationEnd aggregation="none" isNavigable="false"
573                   isSpecification="false" name="" ordering="unordering"
574                   targetScope="instance" type="C.9" visibility="public" xmi.id="G.32"/>
```

```
575             <UML:AssociationEnd aggregation="none" isNavigable="false"
576               isSpecification="false" name="" ordering="unordering"
577               targetScope="instance" type="C.4" visibility="public" xmi.id="G.33"/>
578           </UML:Association.connection>
579         </UML:Association>
580         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
581           isSpecification="false" name="" visibility="public" xmi.id="G.32">
582           <UML:Association.connection>
583             <UML:AssociationEnd aggregation="none" isNavigable="false"
584               isSpecification="false" name="" ordering="unordering"
585               targetScope="instance" type="C.11" visibility="public" xmi.id="G.33"/>
586             <UML:AssociationEnd aggregation="none" isNavigable="false"
587               isSpecification="false" name="" ordering="unordering"
588               targetScope="instance" type="C.4" visibility="public" xmi.id="G.34"/>
589           </UML:Association.connection>
590         </UML:Association>
591         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
592           isSpecification="false" name="" visibility="public" xmi.id="G.33">
593           <UML:Association.connection>
594             <UML:AssociationEnd aggregation="none" isNavigable="false"
595               isSpecification="false" name="" ordering="unordering"
596               targetScope="instance" type="C.9" visibility="public" xmi.id="G.34"/>
597             <UML:AssociationEnd aggregation="none" isNavigable="false"
598               isSpecification="false" name="" ordering="unordering"
599               targetScope="instance" type="C.4" visibility="public" xmi.id="G.35"/>
600           </UML:Association.connection>
601         </UML:Association>
602         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
603           isSpecification="false" name="" visibility="public" xmi.id="G.34">
604           <UML:Association.connection>
605             <UML:AssociationEnd aggregation="none" isNavigable="false"
606               isSpecification="false" name="" ordering="unordering"
607               targetScope="instance" type="C.11" visibility="public" xmi.id="G.35"/>
608             <UML:AssociationEnd aggregation="none" isNavigable="false"
609               isSpecification="false" name="" ordering="unordering"
610               targetScope="instance" type="C.4" visibility="public" xmi.id="G.36"/>
611           </UML:Association.connection>
612         </UML:Association>
613         <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
614           isSpecification="false" name="" visibility="public" xmi.id="G.35">
615           <UML:Association.connection>
616             <UML:AssociationEnd aggregation="none" isNavigable="false"
617               isSpecification="false" name="" ordering="unordering"
618               targetScope="instance" type="C.10" visibility="public" xmi.id="G.36"/>
619             <UML:AssociationEnd aggregation="none" isNavigable="false"
```

```
639                 isSpecification="false" name="" ordering="unordering"
640                 targetScope="instance" type="C.9" visibility="public" xmi.id="G.38"/>
641               <UML:AssociationEnd aggregation="none" isNavigable="false"
642                 isSpecification="false" name="" ordering="unordering"
643                 targetScope="instance" type="C.8" visibility="public" xmi.id="G.40"/>
644             </UML:Association.connection>
645           </UML:Association>
646           <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
647             isSpecification="false" name="" visibility="public" xmi.id="G.38">
648             <UML:Association.connection>
649               <UML:AssociationEnd aggregation="none" isNavigable="false"
650                 isSpecification="false" name="" ordering="unordering"
651                 targetScope="instance" type="C.11" visibility="public" xmi.id="G.40"/>
652               <UML:AssociationEnd aggregation="none" isNavigable="false"
653                 isSpecification="false" name="" ordering="unordering"
654                 targetScope="instance" type="C.8" visibility="public" xmi.id="G.41"/>
655             </UML:Association.connection>
656           </UML:Association>
657           <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
658             isSpecification="false" name="" visibility="public" xmi.id="G.40">
659             <UML:Association.connection>
660               <UML:AssociationEnd aggregation="none" isNavigable="false"
661                 isSpecification="false" name="" ordering="unordering"
662                 targetScope="instance" type="C.10" visibility="public" xmi.id="G.41"/>
663               <UML:AssociationEnd aggregation="none" isNavigable="false"
664                 isSpecification="false" name="" ordering="unordering"
665                 targetScope="instance" type="C.8" visibility="public" xmi.id="G.42"/>
666             </UML:Association.connection>
667           </UML:Association>
668           <UML:Association isAbstract="false" isLeaf="false" isRoot="false"
669             isSpecification="false" name="" visibility="public" xmi.id="G.41">
670             <UML:Association.connection>
670             <UML:Association.connection>
671               <UML:AssociationEnd aggregation="none" isNavigable="false"
672                 isSpecification="false" name="" ordering="unordering"
673                 targetScope="instance" type="C.5" visibility="public" xmi.id="G.42"/>
674               <UML:AssociationEnd aggregation="none" isNavigable="false"
675                 isSpecification="false" name="" ordering="unordering"
676                 targetScope="instance" type="C.8" visibility="public" xmi.id="G.43"/>
677             </UML:Association.connection>
678           </UML:Association>
679           <UML:Stereotype baseClass="Abstraction" extendedElement=""
680             name="realize" xmi.id="stereotype"/>
681         </UML:Namespace.ownedElement>
682         </UML:Model>
683         </XMI.content>
684       </XMI>
```

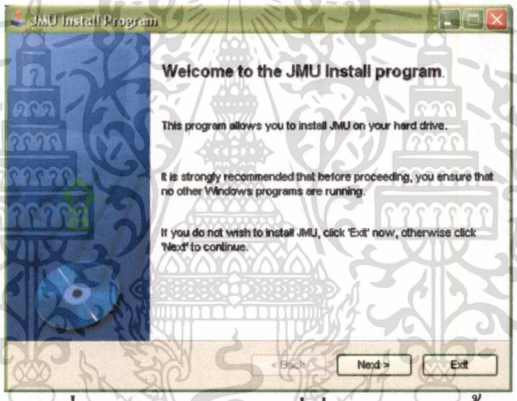
ภาคผนวก ง

การติดตั้งและการใช้งานโปรแกรม JMU

ในการใช้งานและติดตั้งเครื่องมือ JMU ซึ่งย่อมาจาก Java Map to UML – Class Diagram เป็นเครื่องมือที่มีสามารถในการทำรีเวอร์สเอ็นจีเนียริ่งที่มีการสนับสนุนมาตรฐานเอ็กซ์เอ็มไอ(XMI – XML Metadata Interchange) โดยเครื่องมือนี้สามารถทำการสร้างคลาสไดอะแกรมจากโค้ดจาวาและเอกสารทางเอ็กซ์เอ็มไอ(ไฟล์ .xml หรือ .xmi)

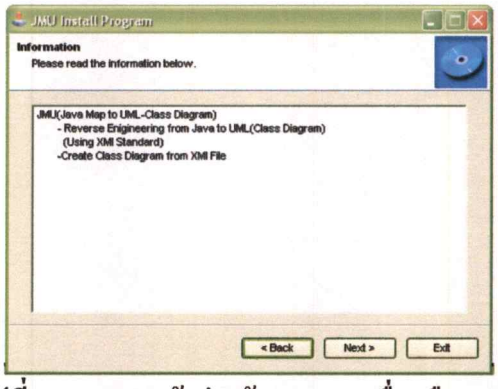
การติดตั้ง JMU

-ไฟล์ติดตั้งของเครื่องมือนี้มีชื่อว่า JMU.exe ในการติดตั้งให้ทำการดับเบิ้ลคลิกที่ไฟล์นี้ ต่อจากนี้จะปรากฏหน้าต่างการใช้งานดังรูปที่ ง.1 เป็นการแสดงหน้าต่างเริ่มต้นการติดตั้ง



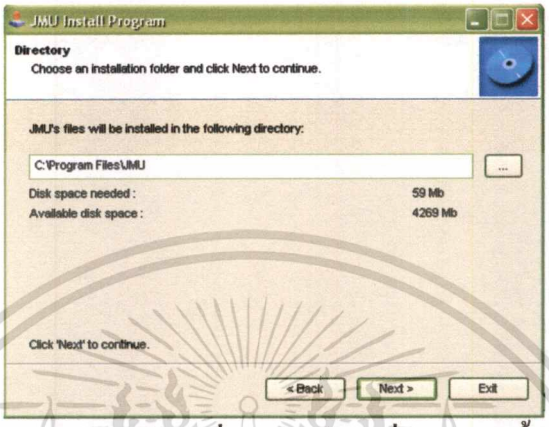
รูปที่ ง.1 แสดงหน้าต่างที่เริ่มต้นการติดตั้ง

- ต่อจากนั้นทำการคลิกปุ่ม Next จะปรากฏหน้าต่างดังรูปที่ ง.2 ที่แสดงหน้าต่างข้อมูลของเครื่องมือ



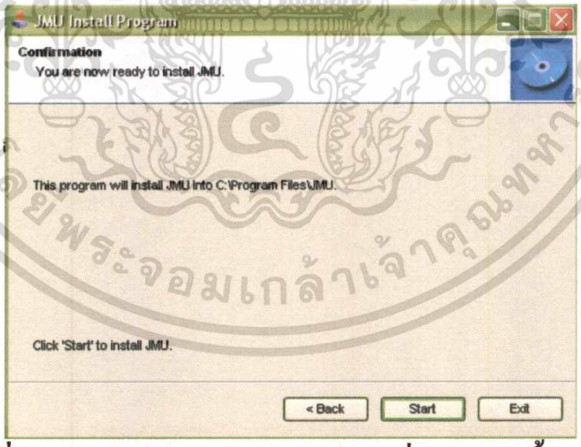
รูปที่ ง.2 แสดงหน้าต่างข้อมูลของเครื่องมือ JMU

- ต่อจากนั้นเมื่อกดปุ่ม Next จะปรากฏหน้าต่างดังรูปที่ ง.3 ที่แสดงหน้าต่างของการติดตั้งว่าจะทำการติดตั้งเครื่องมือ JMU ไว้ที่ไดเร็คทอรี่ใด



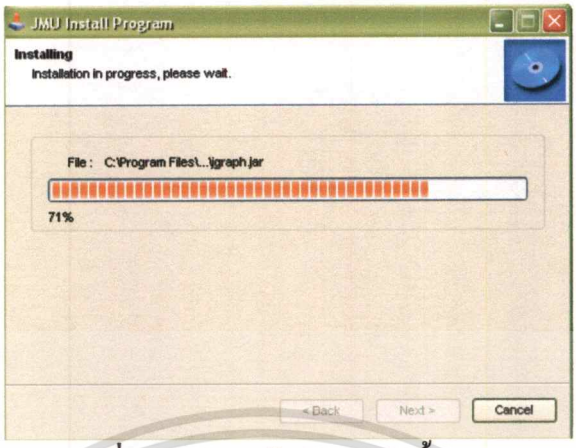
รูปที่ ง.3 แสดงหน้าต่างที่บอกไดเร็คทอรี่ในการติดตั้ง JMU

- เมื่อทำการเลือกไดเร็คทอรี่ในการติดตั้ง JMU แล้ว ให้กดปุ่ม Next จะปรากฏหน้าต่างดังรูปที่ ง.4 เป็นการแสดงหน้าต่างการยืนยันการติดตั้ง JMU ณ ไดเร็คทอรี่ที่ได้ทำการเลือกไว้

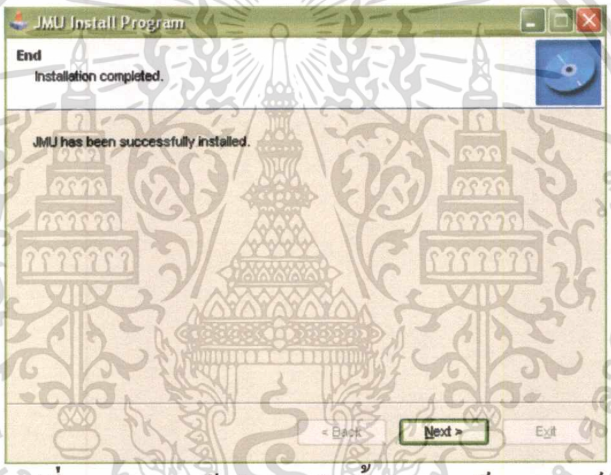


รูปที่ ง.4 แสดงหน้าต่างยืนยันไดเร็คทอรี่ในการติดตั้ง JMU

- เมื่อทำการกดปุ่ม Start จะปรากฏหน้าต่างการติดตั้งเครื่องมือดังรูปที่ ง.5 ที่แสดงหน้าต่างการติดตั้งเครื่องมือ JMU และเมื่อการติดตั้งเครื่องมือ JMU สิ้นสุดจะปรากฏหน้าต่างดังรูปที่ ง.6 ที่แสดงหน้าต่างการติดตั้งเครื่องมือ JMU ได้ถูกติดตั้งแล้ว



รูปที่ ง.5 แสดงหน้าต่างการติดตั้ง JMU



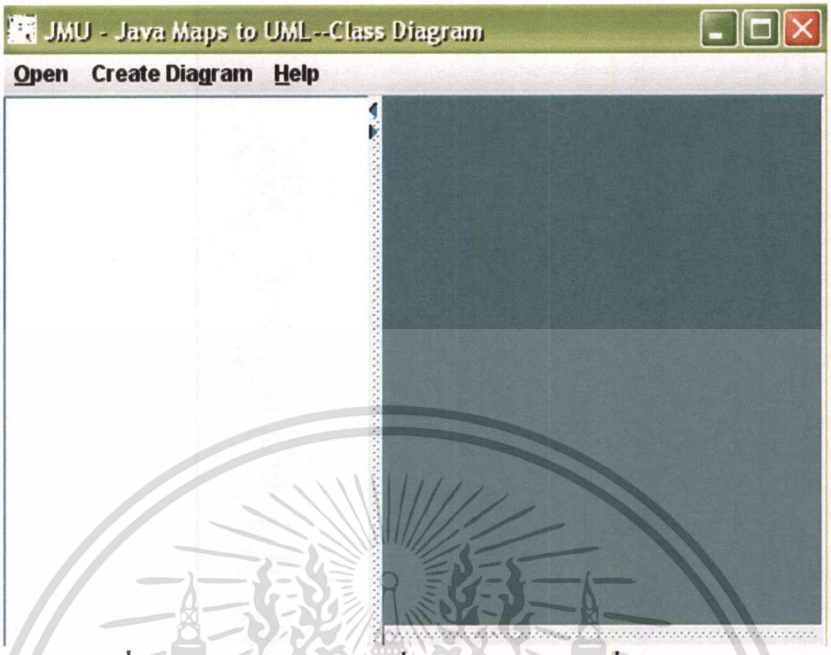
รูปที่ ง.6 แสดงหน้าต่างการติดตั้ง JMU เสร็จสมบูรณ์

การใช้งาน

การใช้งานเครื่องมือ JMU สามารถแบ่งการทำงานได้ 2 ส่วน คือ

- การแสดงไฟล์จาวา, ไฟล์เอกสารทางเอ็กซ์เอ็มไอ, และ ไฟล์ .jmu
- การสร้างคลาสไดอะแกรม โดยสามารถสร้างจากไฟล์จาวาและไฟล์เอกสารทางเอ็กซ์เอ็มไอ

เมื่อเริ่มต้นการใช้งานเครื่องมือ JMU จะปรากฏอินเตอร์เฟสดังรูปที่ ง.7 ที่แสดงอินเตอร์การเริ่ต้นการใช้งานเครื่องมือนี้ ซึ่งจะอธิบายรายละเอียดการใช้งานเครื่องมือ JMU นี้ตามเมนูย่อยที่ปรากฏในอินเตอร์เฟสรูปที่ ก.7 คือ เมนู Open, เมนู Create Diagram, และเมนู Help



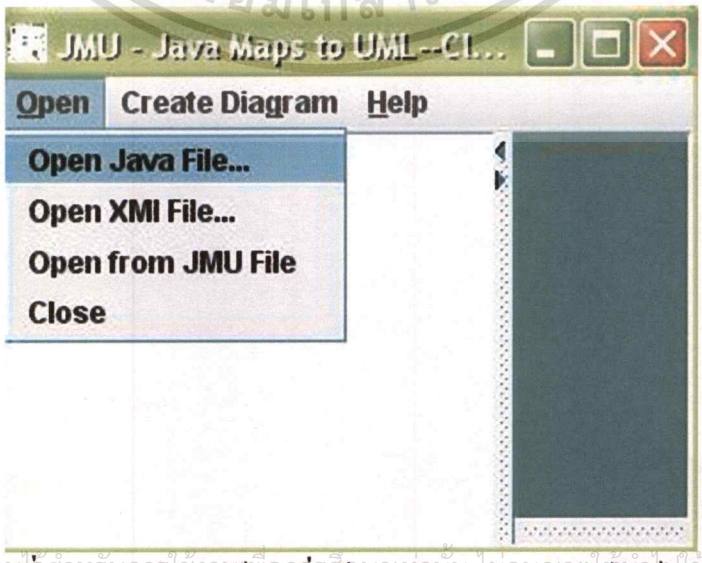
รูปที่ ง.7 แสดงอินเตอร์เฟสเริ่มต้นการใช้งานเครื่องมือ JMU

Open Menu

Open Menu เป็นเมนูที่ทำหน้าที่ในการเปิดไฟล์จาวา, ไฟล์เอกสารทางเอ็กซ์เอ็มไอ, และไฟล์ .jmu มีรายละเอียดการทำงานดังต่อไปนี้

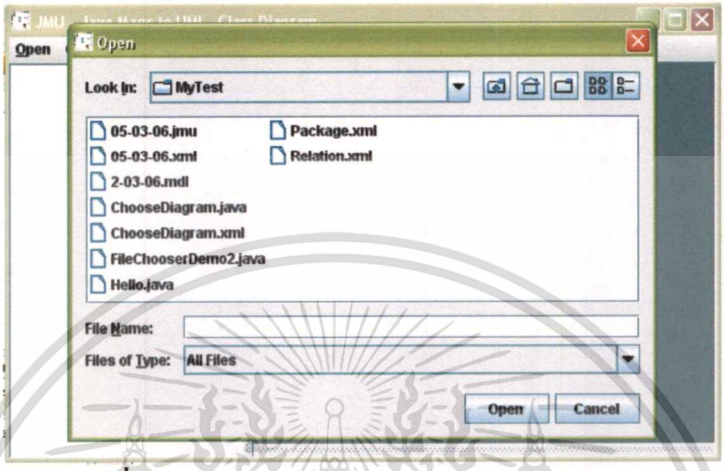
การเปิดไฟล์จาวา

1) เริ่มต้นการทำงานโดยการเลือกเมนู Open และ เลือก Open Java File ดังรูปที่ ง.8 ที่อินเตอร์เฟสเริ่มต้นการทำงานในการเปิดไฟล์จาวา



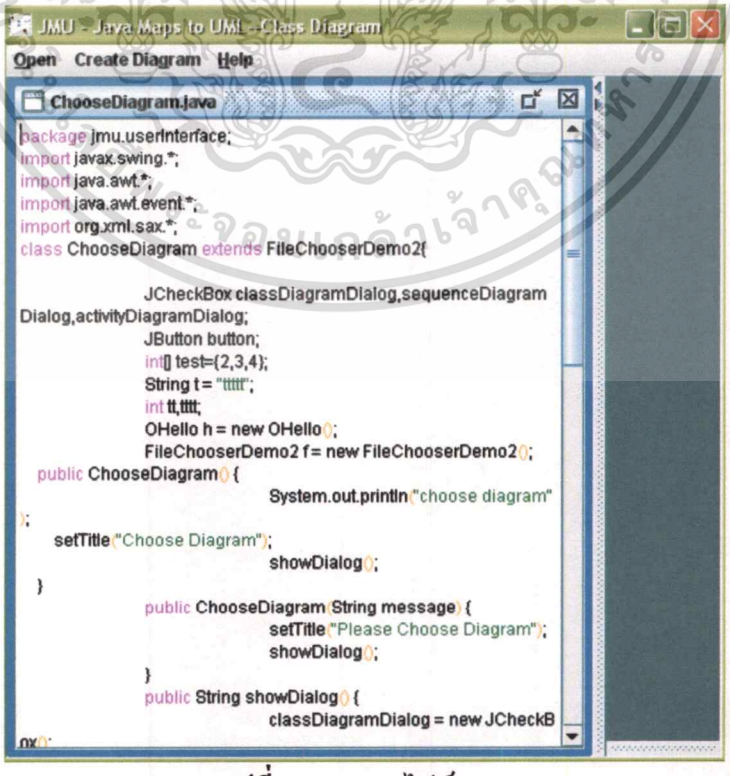
รูปที่ ง.8 แสดงอินเตอร์เฟสเริ่มต้นการทำงานในการเปิดไฟล์จาวา

2) เมื่อเลือก Open Java File... จะปรากฏ Open Dialog ดังรูปที่ ง.9 แสดง Open Dialog ในการเลือกเปิดไฟล์จาวา



รูปที่ ง.9 แสดง Open Dialog ในการเลือกไฟล์จาวา

3) เมื่อเลือกไฟล์จาวาที่ต้องการแล้วต่อจากนั้น กดปุ่ม Open ไฟล์จาวาที่ถูกเลือกจะแสดงบนเครื่องมือทางด้านซ้าย ดังรูปที่ ง.10 ที่แสดงการไฟล์จาวาที่เปิดบนเครื่องมือ JMU



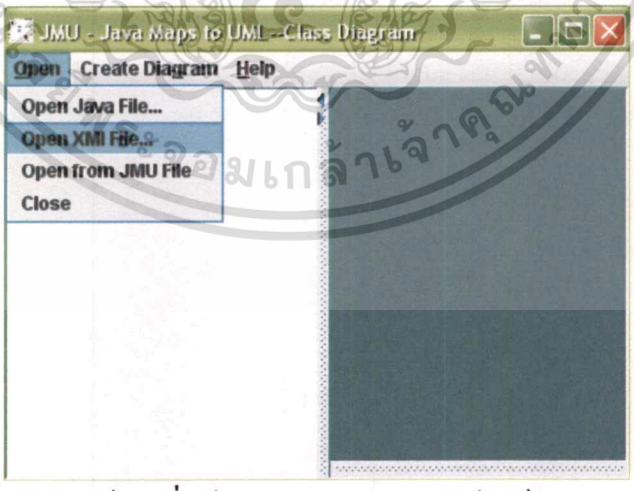
รูปที่ ง.10 แสดงไฟล์จาวา

4) ในกรณีที่ไฟล์ที่ถูกเลือกไม่ได้เป็นไฟล์จาวาระบบจะแสดงข้อมูลเตือนว่าไฟล์ที่เลือกนั้นไม่ได้เป็นไฟล์จาวา ดังรูปที่ ง.11 แสดงข้อความเตือนว่าไฟล์เลือกนั้นไม่ได้เป็นไฟล์จาวา

รูปที่ ง.11 แสดงข้อความเตือนในการเปิดไฟล์จาวา

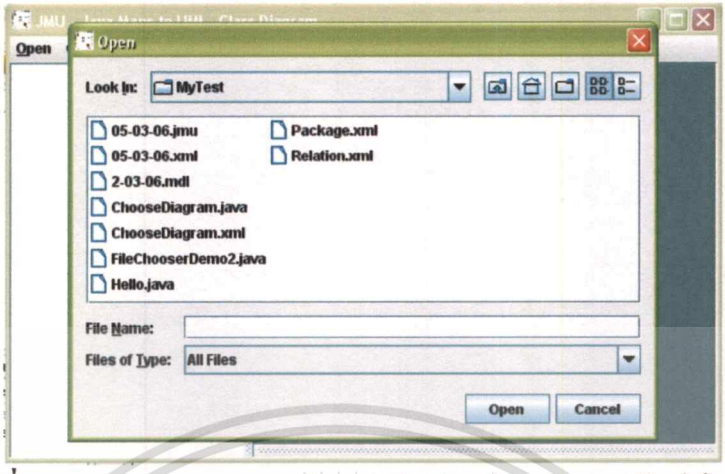
การเปิดไฟล์เอกสารทางเอ็กซ์เอ็มไอ

1) เริ่มต้นการทำงานโดยการเลือกเมนู Open และ เลือก Open XMI File ดังรูปที่ ง.12 ที่อินเตอร์เฟสเริ่มต้นการทำงานในการเปิดไฟล์เอกสารเอ็กซ์เอ็มไอ โดยไฟล์เอกสารทางเอ็กซ์เอ็มไอสามารถเป็นไฟล์ที่เป็นนามสกุล .xml และ .xmi



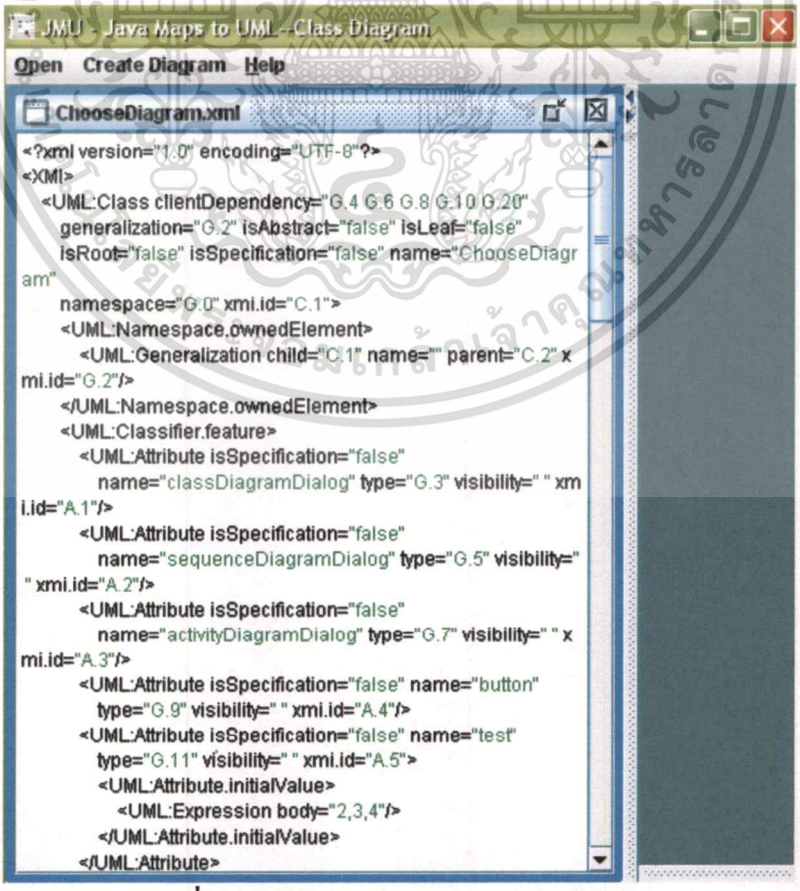
รูปที่ ง.12 แสดงอินเตอร์เฟสเริ่มต้นการทำงานในการเปิดไฟล์เอกสารทางเอ็กซ์เอ็มไอ

2) เมื่อเลือก Open XMI File... จะปรากฏ Open Dialog ดังรูปที่ ง.13 แสดง Open Dialog ในการเลือกเปิดไฟล์เอกสารเอ็กซ์เอ็มไอ



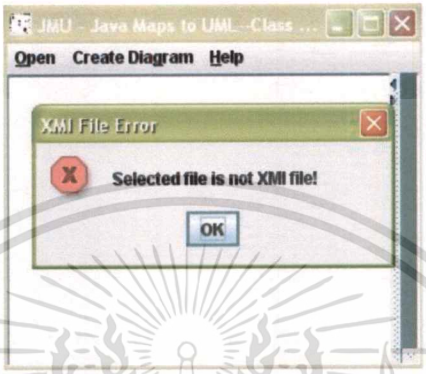
รูปที่ ง.13 แสดง Open Dialog ในการเลือกไฟล์เอกสารทางเอ็กซ์เอ็มไอ

3) เมื่อเลือกไฟล์เอกสารทางเอ็กซ์เอ็มไอที่ต้องการแล้วต่อจากนั้น กดปุ่ม Open ไฟล์เอกสารทางเอ็กซ์เอ็มไอที่ถูกเลือกจะแสดงบนเครื่องมือทางด้านซ้าย ดังรูปที่ ง.14 ที่แสดงการไฟล์เอกสารทางเอ็กซ์เอ็มไอที่เปิดบนเครื่องมือ JMU



รูปที่ ง.14 แสดงไฟล์เอกสารทางเอ็กซ์เอ็มไอ

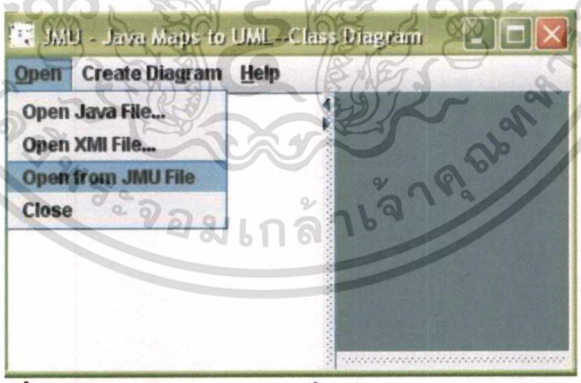
4) ในกรณีที่ไฟล์ที่ถูกเลือกไม่ได้เป็นไฟล์ที่มีนามสกุล .xml และ .xmi ระบบจะแสดงข้อมูลเตือนว่าไฟล์ที่เลือกนั้นไม่ได้เป็นไฟล์เอกสารเอ็กซ์เอ็มไอ ดังรูปที่ ง.15 แสดงข้อความเตือนว่าไฟล์เลือกนั้นไม่ได้เป็นไฟล์เอกสารทางเอ็กซ์เอ็มไอ



รูปที่ ง.15 แสดงข้อความเตือนในการเปิดไฟล์เอกสารทางเอ็กซ์เอ็มไอ

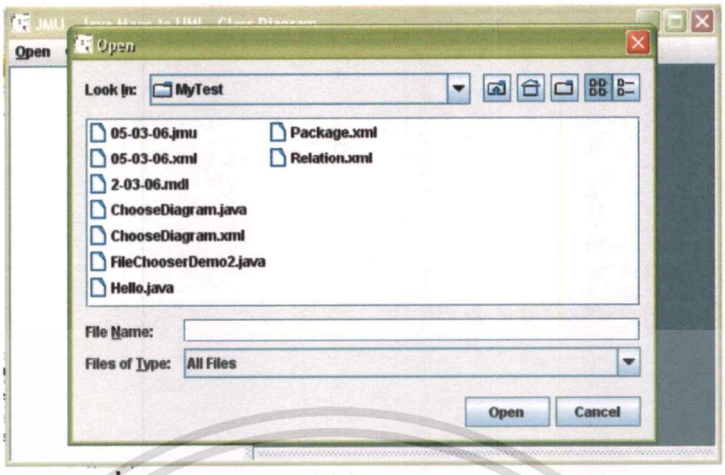
การเปิดไฟล์ .jmu

1) เริ่มต้นการทำงานโดยการเลือกเมนู Open และ เลือก Open JMU File ดังรูปที่ ง.16 ที่อินเตอร์เฟสเริ่มต้นการทำงานในการเปิดไฟล์ .jmu



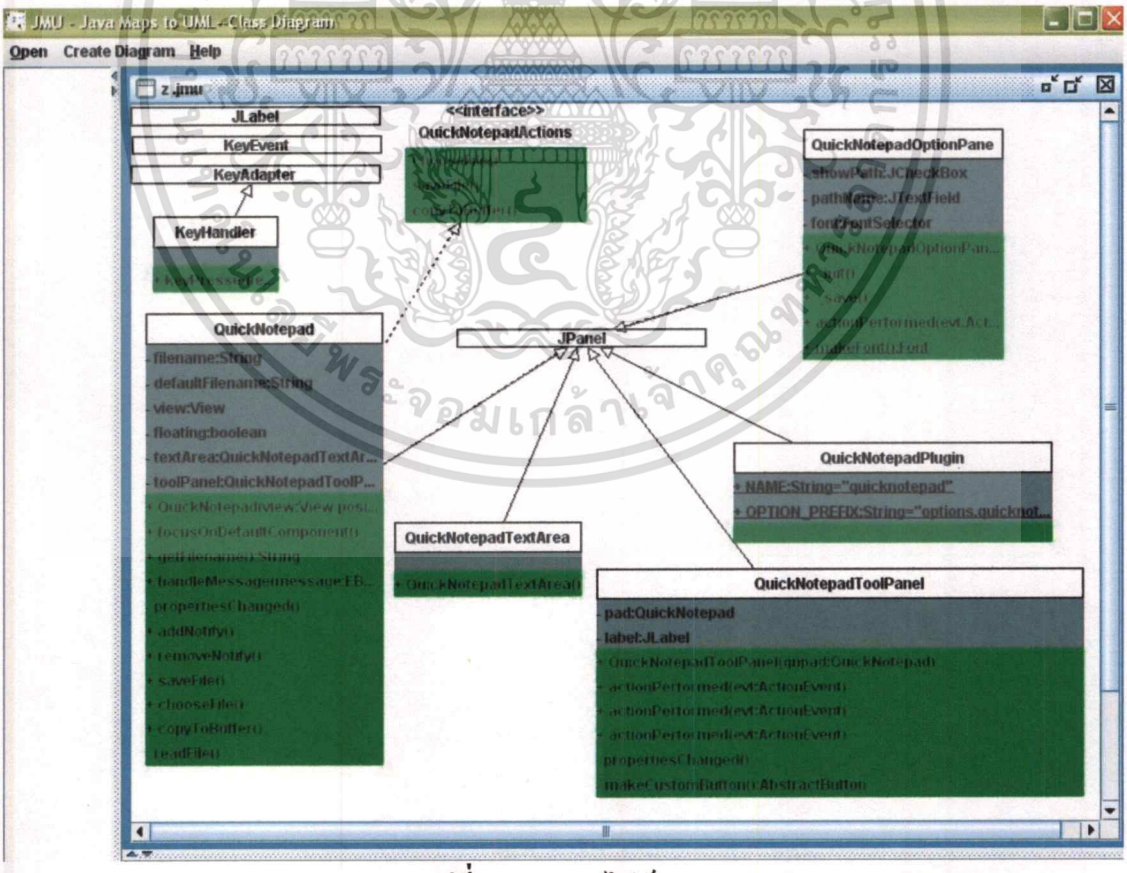
รูปที่ ง.16 แสดงอินเตอร์เฟสเริ่มต้นการใช้งานไฟล์ .jmu

2) เมื่อเลือก Open JMU File... จะปรากฏ Open Dialog ดังรูปที่ ง.17 แสดง Open Dialog ในการเลือกเปิดไฟล์ .jmu



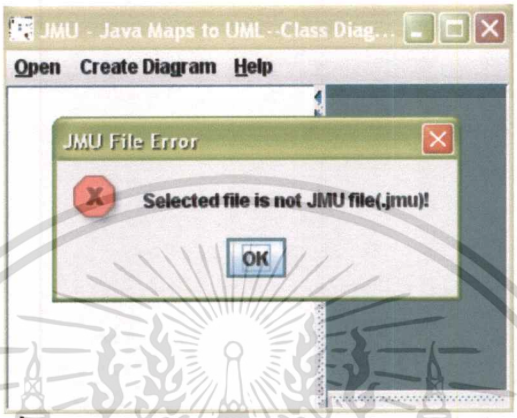
รูปที่ ง.17 แสดง Open Dialog ในการเลือกไฟล์ .jmu

3) เมื่อเลือกไฟล์ .jmu ที่ต้องการแล้วต่อจากนั้น กดปุ่ม Open ไฟล์ .jmu ที่ถูกเลือกจะแสดงบนเครื่องมือทางด้านขวา ดังรูปที่ ง.18 ที่แสดงการไฟล์ .jmu ที่เปิดบนเครื่องมือ JMU



รูปที่ ง.18 แสดงไฟล์.jmu

4) ในกรณีที่ไฟล์ที่ถูกเลือกไม่ได้เป็นไฟล์ .jmu ระบบจะแสดงข้อมูลเตือนว่าไฟล์ที่เลือกนั้นไม่ได้เป็นไฟล์ .jmu ดังรูปที่ ง.19 แสดงข้อความเตือนว่าไฟล์เลือกนั้นไม่ได้เป็นไฟล์.jmu



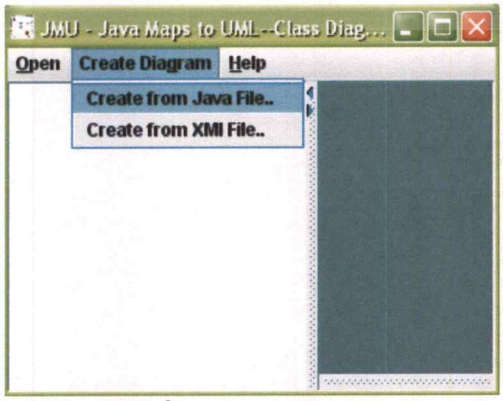
รูปที่ ง.19 แสดงข้อความเตือนในการเปิดไฟล์ .jmu

Create Diagram

ในการสร้างคลาสไดอะแกรมของเครื่องมือ JMU นี้ สามารถทำการสร้างจากโค้ดจาวาและเอกสารเอ็กซ์เอ็มไอ ทั้งนี้เอกสารเอ็กซ์เอ็มไอที่นำมาเปิดกับเครื่องมือนี้ ไม่ได้จำกัดว่าเอกสารเอ็กซ์เอ็มไอนั้นต้องเป็นเอกสารเอ็กซ์เอ็มไอที่ได้จากการทำรีเวอร์เอ็นจีเนียริ่ง เครื่องมือ JMU นี้สามารถแปลงเอกสารเอ็กซ์เอ็มไอที่ได้มาจากการ export ของคลาสไดอะแกรมได้

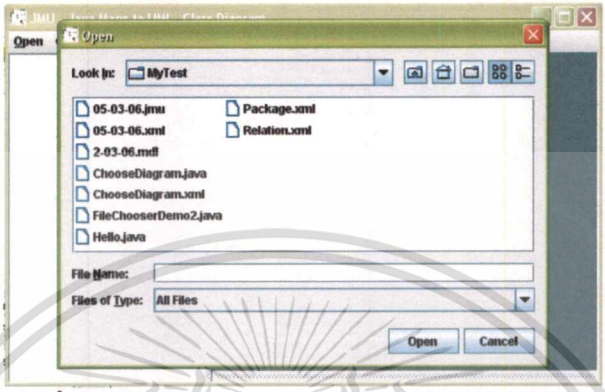
การสร้างคลาสไดอะแกรมจากโค้ดจาวา

1) เริ่มต้นการทำงานโดยการเลือกเมนู Create Diagram และ เลือก Create from Java File... ดังรูปที่ ง.20 ที่อินเตอร์เฟสเริ่มต้นการสร้างคลาสไดอะแกรมจากโค้ดจาวา



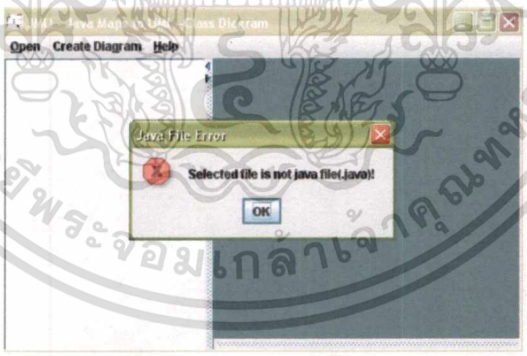
รูปที่ ง.20 แสดงอินเตอร์เฟสเริ่มต้นการสร้างคลาสไดอะแกรมจากโค้ดจาวา

2) เมื่อเลือก Create from Java File... จะปรากฏ Open Dialog ดังรูปที่ ง.21 ที่แสดง Open Dialog เพื่อเลือกไฟล์จาวาในการสร้างคลาสไดอะแกรม



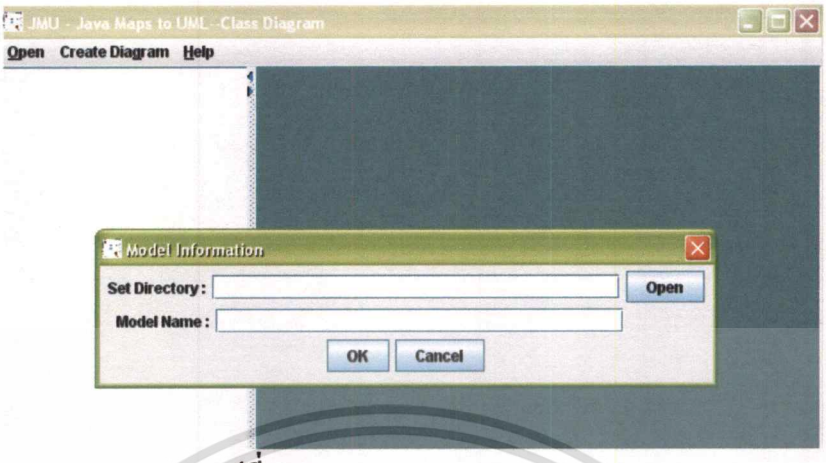
รูปที่ ง.21 แสดง Open Dialog ในการเลือกไฟล์จาวา

3) ในกรณีไฟล์ที่เลือกไม่ได้เป็นไฟล์จาวาจะมีข้อความเตือนว่าไฟล์ที่เลือกไม่ได้เป็นไฟล์จาวา ดังรูปที่ ง.22 ที่แสดงข้อความเตือนในกรณีที่ไฟล์ที่เลือกไม่ได้เป็นไฟล์จาวา ต่อจากนั้นทำการกดปุ่ม OK แล้วทำการเริ่มต้นขั้นตอนตั้งแต่ขั้นตอนที่ 1



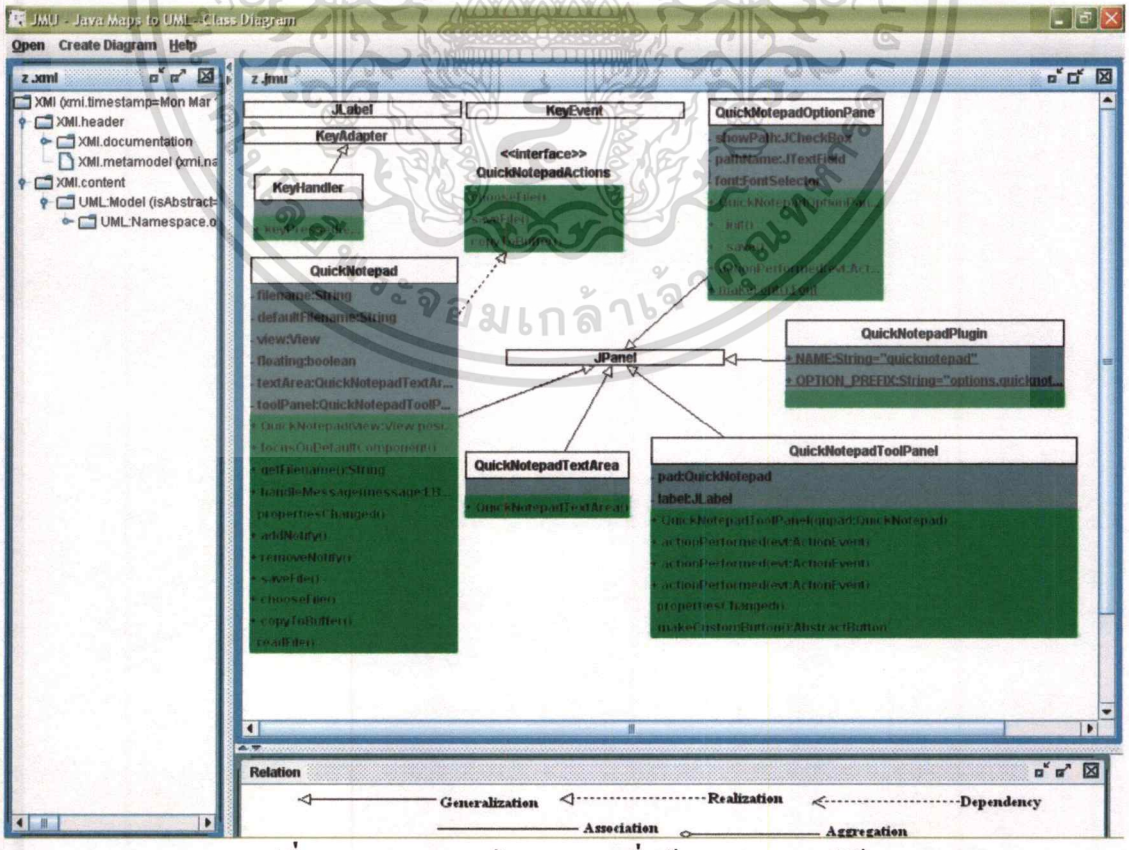
รูปที่ ง.22 แสดงข้อความเตือนว่าไฟล์ที่เลือกไม่ได้เป็นไฟล์จาวา

4) ในการเลือกไฟล์จาวาสามารถเลือกไฟล์จาวาหนึ่งไฟล์หรือมากกว่าหนึ่งไฟล์ได้ ต่อจากนั้นกดปุ่ม Open จะปรากฏหน้าต่างในการกรอกข้อมูลของคลาสไดอะแกรม ดังรูปที่ ง.23 ที่แสดงหน้าต่างในกรอกข้อมูลของคลาสไดอะแกรม โดยต้องทำการเลือกไดเร็คทอรี่ที่ต้องการเก็บไฟล์คลาสไดอะแกรม ต่อจากนั้นทำการตั้งชื่อของคลาสไดอะแกรม



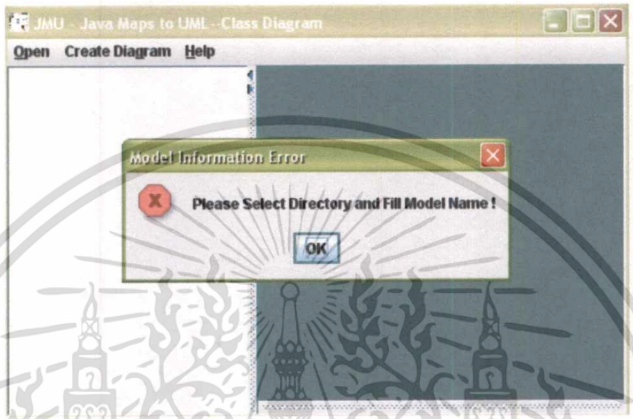
รูปที่ ง.23 แสดง Model Information

5) หลังจากเลือกไดเร็คทอรี่และตั้งชื่อของคลาสไดอะแกรมแล้ว ทำการกดปุ่ม OK จะปรากฏผลลัพธ์ ดังรูปที่ ง.24 ที่แสดงอินเตอร์ของผลลัพธ์ที่ได้จากการแมพโค้ดจาวาเป็นยูเอ็มแอลคลาสไดอะแกรม โดยที่ทางด้านขวาของอินเตอร์เฟสเป็นการแสดงเอกสารเอ็กซ์เอ็มไอในรูปแบบของทรี และผลลัพธ์คลาสไดอะแกรมจะถูกแสดงทางด้านซ้าย



รูปที่ ง.24 แสดงคลาสไดอะแกรมที่สร้างมาจากจาวาโค้ด

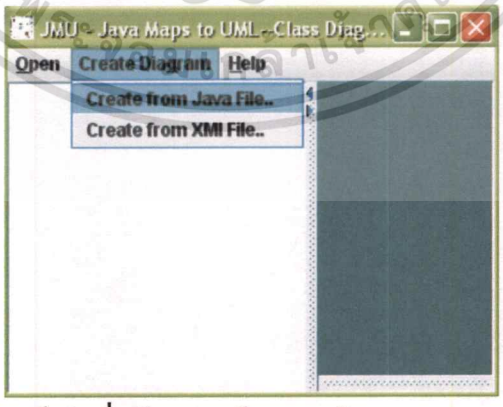
6) ในกรณีที่ไม่ได้เลือกไดเร็คทอรี่และตั้งชื่อคลาสไดอะแกรม หรือ ข้อมูลใดข้อมูลหนึ่งในหน้าต่าง Model Information ไม่ครบ จะปรากฏข้อความเตือนดังรูปที่ ง.25 ที่แสดงข้อความเตือนในกรณีที่ข้อมูลของ Model Information ไม่ครบ ให้ทำการกดปุ่ม OK จะมีการปรากฏหน้าต่าง Model Information



รูปที่ ง.26 แสดงข้อความเตือนในกรณีที่ Model Information ไม่ครบ

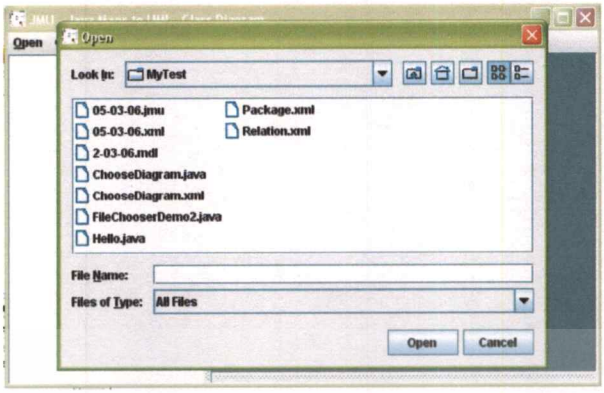
การสร้างคลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอ

1) เริ่มต้นการทำงานโดยการเลือกเมนู Create Diagram และ เลือก Create from XMI File... ดังรูปที่ ง.27 ที่อินเตอร์เฟสเริ่มต้นการสร้างคลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอ



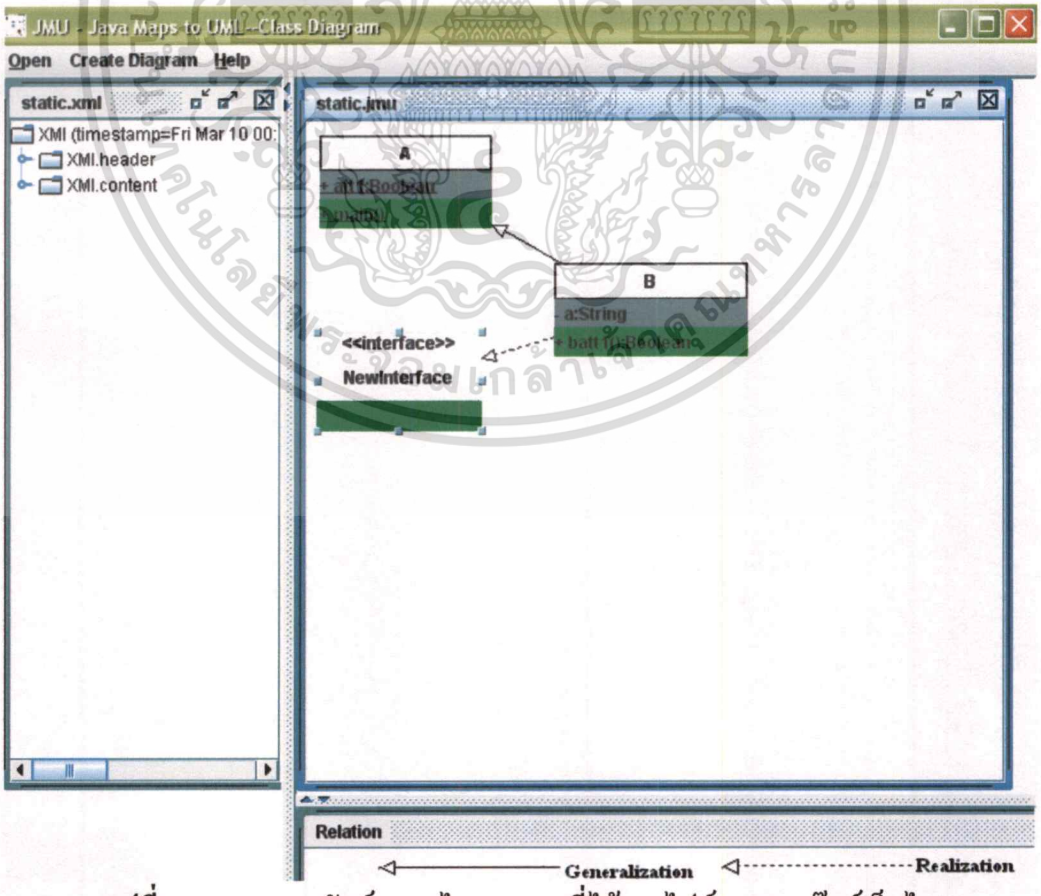
รูปที่ ง.27 แสดงอินเตอร์เฟสเริ่มต้นการสร้างคลาสไดอะแกรมจากเอกสารเอ็กซ์เอ็มไอ

2) เมื่อเลือก Create XMI File จะปรากฏ Open Dialog ดังรูปที่ ง.28 ที่แสดง Open Dialog ในการเลือกไฟล์เอกสารเอ็กซ์เอ็มไอที่ต้องการแสดงคลาสไดอะแกรม



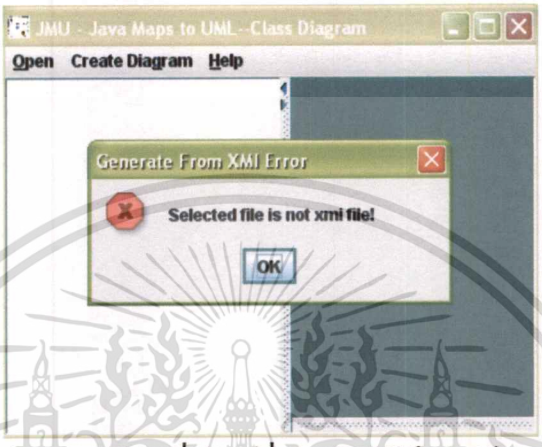
รูปที่ ง.28 แสดง Open Dialog ในการเลือกไฟล์เอกสารเอ็กซ์เอ็มไอ

3) เมื่อทำการเลือกไฟล์เอกสารเอ็กซ์เอ็มไอ กดปุ่ม Open เครื่องมือจะทำการแสดงผลลัพธ์ที่เป็นเอกสารเอ็กซ์เอ็มไอในรูปแบบของทรีทางด้านซ้าย และทำการแสดงคลาสไดอะแกรมทางด้านขวา ดังรูปที่ ง.29 ที่แสดงผลลัพธ์คลาสไดอะแกรมที่ได้จากเอกสารเอ็กซ์เอ็มไอ



รูปที่ ง.29 แสดงผลลัพธ์คลาสไดอะแกรมที่ได้จากไฟล์เอกสารเอ็กซ์เอ็มไอ

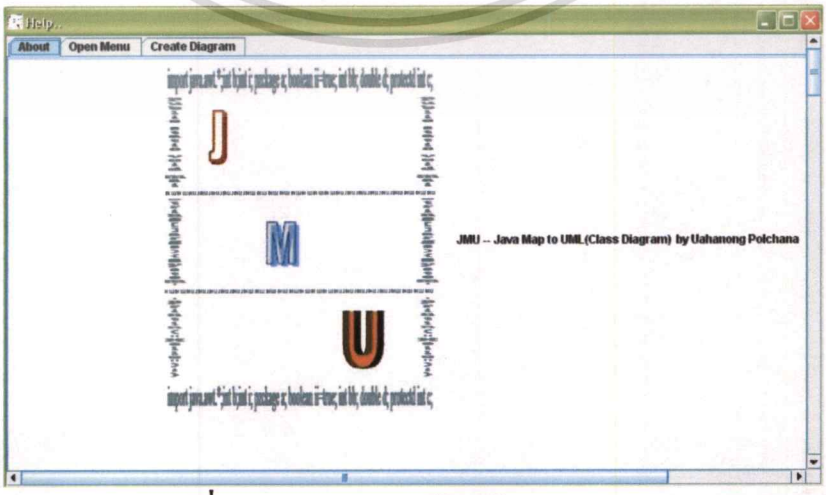
4) ในกรณีที่ไฟล์ที่เลือกไม่ได้เป็นไฟล์เอกสารเอ็กซ์เอ็มไอ เครื่องมือจะทำการแสดงข้อความเตือน ดังรูปที่ ง.30 ที่แสดงข้อความเตือนเมื่อไฟล์ที่เลือกไม่ได้เป็นไฟล์เอกสารเอ็กซ์เอ็มไอ



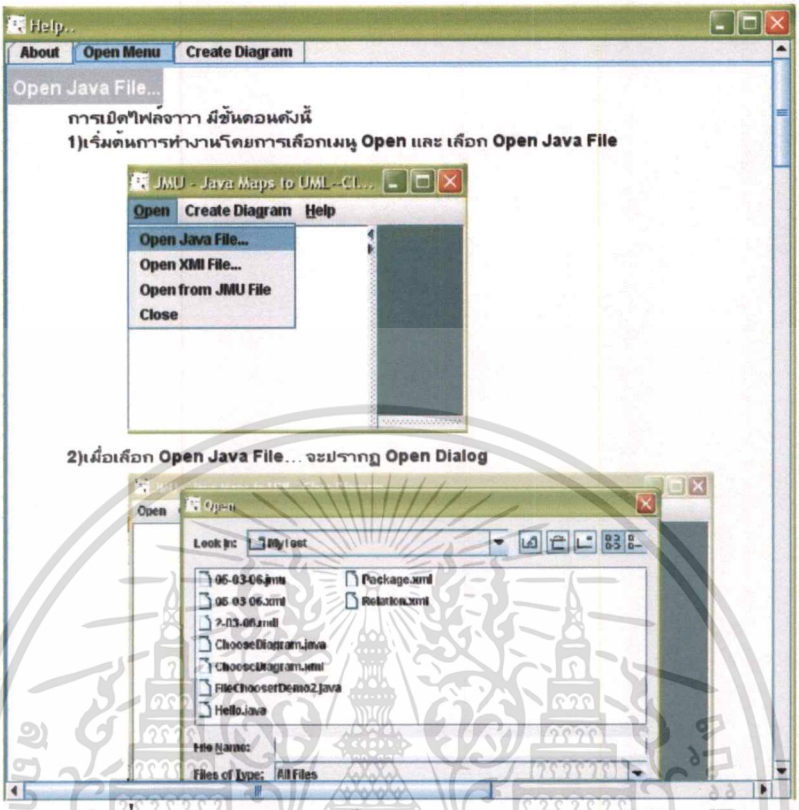
รูปที่ ง.30 แสดงข้อความเตือนเมื่อไฟล์ที่เลือกไม่ได้เป็นไฟล์เอกสารเอ็กซ์เอ็มไอ

Help Menu

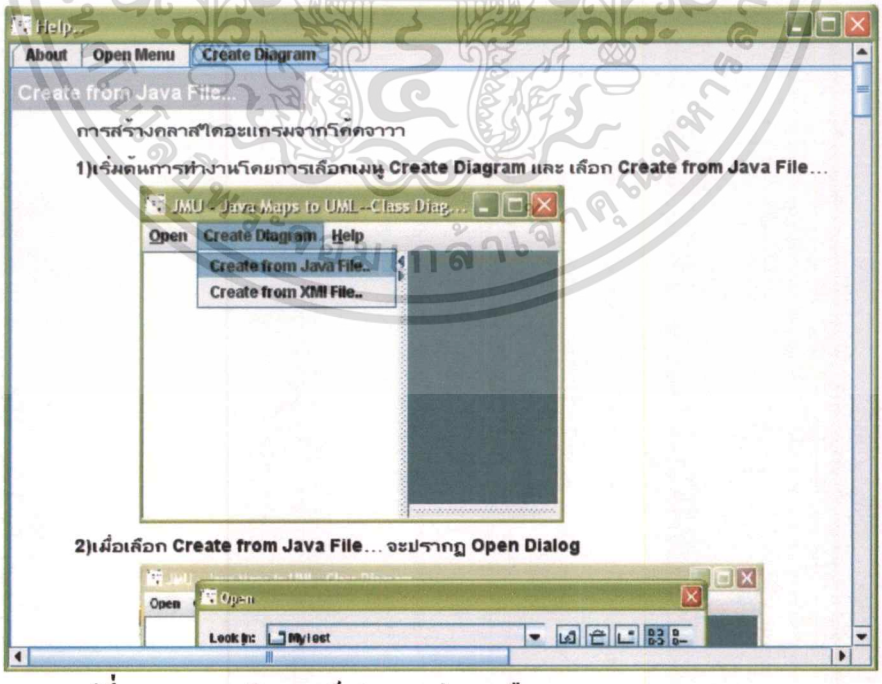
เมนูนี้เป็นการแสดงข้อมูลของเครื่องมือ JMU และรายละเอียดการใช้งานของเครื่องมือ JMU (เลือกเมนู Help -> Help Content) โดยรูปที่ ง.31 – ง.33 เป็นการแสดงอินเตอร์เฟสของเมนู Help โดยรูปที่ ง.31 เป็นการแสดงอินเตอร์เฟสข้อมูลของ JMU, รูปที่ ง.32 เป็นการแสดงอินเตอร์เฟสรายละเอียดการใช้งานเมนู Open , และ รูปที่ ง.33 เป็นการแสดงอินเตอร์เฟสรายละเอียดการใช้งานเมนู Create Diagram



รูปที่ ง.31 แสดงอินเตอร์เฟสข้อมูลของ JMU



รูปที่ ง.32 แสดงอินเตอร์เฟสการช่วยเหลือของเมนู Open



รูปที่ ง.33 แสดงอินเตอร์เฟสการช่วยเหลือของเมนู Create Diagram

ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาว เอื้ออนงค์ พลชนะ
วันเกิด	3 พฤศจิกายน 2524
สถานที่เกิด	กรุงเทพฯ
วุฒิการศึกษาระดับปริญญาตรี	มนุษยศาตรบัณฑิต
สถานที่สำเร็จการศึกษา	คณะมนุษยศาสตร์ มหาวิทยาลัยรามคำแหง
ปีที่สำเร็จการศึกษา	2545