การพัฒนาโปรแกรมประยุกต์เชิงวัตถุโดยใช้ VisualAge for Java

Object Oriented Application Implementation using VisualAge for Java

นายปกรณ์ นวลประเสริฐสุข
นายอภิวัฒน์ วิเชียรบรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุโดยใช้ VisualAge for Java

Object Oriented Application Implementation using VisualAge for Java

โดย

นายปกรณ์ นวลประเสริฐสุข

นายอภิวัฒน์ วิเชียรบรรณ

อาจารย์ที่ปรึกษา

อาจารย์ บรรจง ปิยธำรง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมประยุกต์เชิงวัตถุโดยใช้ VisualAge for Java

Object Oriented Application Implementation using VisualAge for Java

ผู้จัดทำ

1. นายปกรณ์ นวลประเสริฐสุข รหัสประจำตัว 40013256
2. นายอภิวัฒน์ วิเชียรบรรณ รหัสประจำตัว 40013278

_____อาจารย์ที่ปรึกษา

(อาจารย์ บรรจง ปิยธำรง)

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุโดยใช้ VisualAge for Java

นายปกรณ์ นวลประเสริฐสุข 40013256
นายอภิวัฒน์ วิเชียรบรรณ 40013278
อาจารย์ บรรจง ปิยธำรง อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของโครงงานวิจัยและพัฒนาการออกแบบโปรแกรมประยุกต์เชิงวัตถุ เพื่อสร้างระบบการจัดการฐานข้อมูลโครงการขายบ้านจัดสรร โดยระบบเดิมนั้นจะเป็นระบบที่ใช้คนในการจัดเก็บเอกสาร ซึ่งมีความยุ่งยากและล่าช้ากว่า จากปัญหาดังกล่าวจึงได้นำเอาเทคโนโลยีการออกแบบเชิงวัตถุมาใช้ เพื่อให้ระบบมีความสะดวก, ถูกต้องและรวดเร็วมากขึ้น

การพัฒนาจะเริ่มตั้งแต่ การกำหนดความต้องการของระบบ, วิเคราะห์และออกแบบระบบเชิงวัตถุ, ออกแบบและสร้างฐานข้อมูลของระบบ โดยจะใช้ระบบการจัดการฐานข้อมูลของออราเคิล 8 และพัฒนาโปรแกรมประยุกต์เชิงวัตถุนั้น โดยใช้เครื่องมือในการพัฒนาภาษาจาวาคือ “VisualAge for Java”

Object Oriented Application Implementation using VisualAge for Java

Pakorn Nualprasertsuk 40013256
Apiwat Wicheanbarn 40013278
Asst. Prof. Banjong Piyatamrong Advisor
1999

ABSTRACT

This thesis is the research, development and design Object-Oriented application for database management of village project. The legacy system managed document manually, which is complex and slow. For solving this problem, Object-Oriented Technology is used for convenient, inexpensive, and rapid development.

This development was started from requirement specification, system analysis and Object-Oriented design, to system database design and implementation using "Oracle 8" as the DBMS and "VisualAge for Java" for developing Object-Oriented programs.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี คณะผู้จัดทำก็ต้องขอขอบพระคุณอาจารย์และเพื่อนๆทุกคน ที่คอยให้ความช่วยเหลือเป็นอย่างดี ขอขอบพระคุณอาจารย์บรรจง ที่คอยให้คำปรึกษาและแนะนำในการทำโครงงานนี้ ขอขอบคุณน้องพฤกษ์ที่ช่วยแปลบทคัดย่อให้ ขอขอบคุณ ตาห้อย, ตาเถิก ที่ให้ข้อมูลในการจัดทำปริญญานิพนธ์ ขอขอบคุณ ตาโอ้ฟาม, ตาเหลี่ยม, ตาทัชจัง, ตาต้อมหมี, อาเกา ที่อยู่ทำโปรเจ็คด้วยกัน ขอบคุณน้องอาร์ทที่คอยให้กำลังใจ, ขอบคุณพ่อและแม่ที่ส่งกำลังใจมาให้

คณะผู้จัดทำ

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพ	VI
สารบัญตาราง	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ขอบเขตของงานวิจัย	2
1.4 วิธีการดำเนินงาน	3
บทที่ 2 ทฤษฎีและหลักการ	4
2.1 พื้นฐานของออบเจ็กต์-โอเรียนเต็ด (Object-Oriented)	4
2.2 การเขียนโปรแกรมเชิงวัตถุโดยใช้จาวา	6
2.3 VisualAge for Java 2.0 Enterprise Edition	7
2.3.1 คุณสมบัติของ VisualAge for Java Enterprise Edittion	15
2.3.2 การติดต่อกับฐานข้อมูลรีเลชันด้วย VisualAge for Java	16
2.3.3 Select Bean	16
2.3.4 DBNavigator Bean	17
2.4 ตัวอย่างโปรแกรมประยุกต์ที่ใช้ในการออกแบบ	17
2.5 เทคนิคการแมป (Mapping) จากออบเจ็กต์เป็นตาราง	19
2.5.1 การแมปจากออบเจ็กต์เป็นตาราง (Mapping Framework)	19
2.5.2 กลยุทธ์ในการแมปประเภทต่างๆ	19
2.6 การติดต่อกับฐานข้อมูลออราเคิลผ่าน JDBC และ SQLJ	21
2.6.1 ติดต่อกับฐานข้อมูลออราเคิลโดยภาษาจาวา	21
2.6.2 เจดีบีซี (JDBC)	22
2.6.3 SQLJ	25
2.7 ออราเคิล 8	27
บทที่ 3 การสร้างและออกแบบ	33
3.1 ความต้องการของระบบ (Requirement)	33
3.2 ยูสเคส (Use Case)	35

3.3 กำหนดออบเจ็กต์จากความต้องการของระบบ 41
3.4 กำหนดรายละเอียดและความสัมพันธ์ให้กับออบเจ็กต์ 43
3.5 การแมป (Mapping) ออบเจ็กต์เป็นฐานข้อมูลรีเลชันนัล 50
3.6 แสดงตาราง (Table) ของฐานข้อมูลจากระบบทั้งหมด 15 ตาราง 54
3.7 แสดงการใช้คำสั่ง SQL ในการสร้างตารางฐานข้อมูลในออราเคิล 60
3.8 การออกแบบหน้าจอในการติดต่อกับผู้ใช้ 67
3.9 การจัดเก็บแผนผังของโครงการ 72
3.10 โปรแกรมส่วนของเจดีบีซี (JDBC) 75
3.10.1 การสร้างการเชื่อมต่อกับฐานข้อมูล (Connect) 76
3.10.2 การโหลดข้อมูล (Load) 76
3.10.3 การลบข้อมูล (Remove) 78
3.10.4 การแก้ไขอัพเดตข้อมูล (Update) 79
3.10.5 การเพิ่มข้อมูล (Insert, Add) 79
บทที่ 4 การทดสอบและผลการทดลอง 80
4.1 การทดสอบหน้าจอและแอพพลิเคชัน 80
4.2 การทดสอบกลไกของระบบและการจัดเก็บลงฐานข้อมูล 83
บทที่ 5 การวิจารณ์และสรุป 87
5.1 สรุป 87
5.2 บทวิจารณ์ 87
5.3 ข้อเสนอแนะและแนวทางในการพัฒนาต่อ 88
บรรณานุกรม

สารบัญภาพ

หน้าที่

รูปที่ 2.1-1 แสดงข้อมูลในส่วนที่ Public และ Private ของออบเจ็กต์ 6
รูปที่ 2.3-1 แสดงให้เห็นหน้าจอการทำงานของ VisualAge 8
รูปที่ 2.3-2 แสดง Log Windows 9
รูปที่ 2.3-3 แสดง Workbench Windows ของ VisualAge for Java 10
รูปที่ 2.3-4 แสดงการแก้ไขโปรแกรมโดยใช้วินโดว์คนละตัวกับ Workbench 11
รูปที่ 2.3-5 แสดง Console Windows 11
รูปที่ 2.3-6 แสดง Debugger Windows 12
รูปที่ 2.3-7 แสดง Scrapbook Windows 13
รูปที่ 2.3-8 แสดง Option Windows 14
รูปที่ 2.3-9 แสดง Repository Explorer 15
รูปที่ 2.4-1 แสดงตัวอย่างการออกแบบ User-Interface 18
รูปที่ 2.5-1 แสดงการแมปจากคลาสเป็นตาราง 19
รูปที่ 2.5-2 แสดงการแมปจากคลาสทั้ง 3 ที่สืบทอดกันเป็น 1 ตาราง 20
รูปที่ 2.5-3 แสดงการแมปจากคลาสทั้ง 3 ที่สืบทอดกันเป็นตาราง Root และ Leaf 20
รูปที่ 2.5-4 แสดงการแมปคลาสที่มีความสัมพันธ์กันมาเป็นตารางโดยการมองจากล่างขึ้นบน 20
รูปที่ 2.5-5 แสดงการแมปคลาสที่มีความสัมพันธ์กันมาเป็นตารางโดยการมองจากบนลงล่าง 21
รูปที่ 2.5-6 แสดงความสัมพันธ์แบบ 1 to many ที่แมปออกมาเป็นตาราง 21
รูปที่ 2.6-1 แสดงเส้นทางของการติดต่อกับฐานข้อมูล 22
รูปที่ 2.6-2 แสดงลำดับชั้นการทำงานของ JDBC 23
รูปที่ 2.6-3 แสดงการติดต่อระหว่างฝั่งไคลเอ็นต์กับฐานข้อมูลฝั่งเซิร์ฟเวอร์ผ่าน JDBC ด้วยจาวา 23
รูปที่ 2.6-4 แสดงขั้นตอนการสร้าง SQLJ แอพพลิเคชัน 26
รูปที่ 2.3-3 แสดงโครงสร้างของชนิดข้อมูลต่างๆใน oracle 8 28
รูปที่ 3.2-1 แสดงความต้องการของระบบ 35
รูปที่ 3.2-2 แสดง use case หลักของระบบ 36
รูปที่ 3.2-3 แสดง Service Management use case 37
รูปที่ 3.2-4 แสดง Contract's Info Management use case 38
รูปที่ 3.2-5 แสดง Customer's Info Management use case 39
รูปที่ 3.2-6 แสดง All Information Management use case 39
รูปที่ 3.2-7 แสดง Project Management use case 40
รูปที่ 3.2-8 แสดง Register Management use case 40
รูปที่ 3.4-1 แสดงคลาส Company 43

รูปที่ 3.4-2 แสดงคลาส Project และ Project Type 43
รูปที่ 3.4-3 แสดงคลาสที่เกี่ยวกับ Home 44
รูปที่ 3.4-4 แสดงคลาส MapBackground กับ MapImage 44
รูปที่ 3.4-5 แสดงคลาสที่แสดงถึงบุคคลที่เกี่ยวข้องกับระบบ 45
รูปที่ 3.4-6 แสดงคลาสที่เกี่ยวกับสัญญาของลูกค้า 45
รูปที่ 3.4-7 แสดงคลาสที่เป็นองค์ประกอบของ Employee 46
รูปที่ 3.4-8 แสดงคลาสไดอะแกรมหน้าจอของระบบที่ใช้ติดต่อกับผู้ใช้ 46
รูปที่ 3.4-9 แสดงคลาสไดอะแกรมทั้งหมดของระบบ 48
รูปที่ 3.4-10 แสดงความสัมพันธ์กันระหว่างคลาสโดยไม่แสดงแอตทริบิวต์กับคุณสมบัติ 49
รูปที่ 3.5-1 แสดงความสัมพันธ์กันระหว่างคลาสและแอตทริบิวต์ของแต่ละคลาส 50
รูปที่ 3.5-2 แสดงความสัมพันธ์กันระหว่างคลาสของคลาส Person กับคลาสลูก 51
รูปที่ 3.5-3 แสดงความสัมพันธ์กันระหว่างคลาส Employee กับคลาสต่างๆ 52
รูปที่ 3.5-4 แสดงความสัมพันธ์กันระหว่างคลาส Customer, Contract และ ContractType 52
รูปที่ 3.5-5 แสดงความสัมพันธ์กันระหว่างคลาส Project กับคลาสต่างๆ 53
รูปที่ 3.7-1 แสดงการสร้างตาราง Customer ด้วยคำสั่ง SQL 60
รูปที่ 3.7-2 แสดงการสร้างตาราง Employee ด้วยคำสั่ง SQL 61
รูปที่ 3.7-3 แสดงการสร้างตาราง Account ด้วยคำสั่ง SQL 61
รูปที่ 3.7-4 แสดงการสร้างตาราง Contract ด้วยคำสั่ง SQL 62
รูปที่ 3.7-5 แสดงการสร้างตาราง ContractType ด้วยคำสั่ง SQL 62
รูปที่ 3.7-6 แสดงการสร้างตาราง Privilege ด้วยคำสั่ง SQL 62
รูปที่ 3.7-7 แสดงการสร้างตาราง Position ด้วยคำสั่ง SQL 62
รูปที่ 3.7-8 แสดงการสร้างตาราง Company ด้วยคำสั่ง SQL 63
รูปที่ 3.7-9 แสดงการสร้างตาราง Project ด้วยคำสั่ง SQL 63
รูปที่ 3.7-10 แสดงการสร้างตาราง ProjectType ด้วยคำสั่ง SQL 63
รูปที่ 3.7-11 แสดงการสร้างตาราง MapBackground ด้วยคำสั่ง SQL 64
รูปที่ 3.7-12 แสดงการสร้างตาราง MapImage ด้วยคำสั่ง SQL 64
รูปที่ 3.7-13 แสดงการสร้างตาราง HomeType ด้วยคำสั่ง SQL 64
รูปที่ 3.7-14 แสดงการสร้างตาราง Home ด้วยคำสั่ง SQL 65
รูปที่ 3.7-15 แสดงการสร้างตาราง HomeStatus ด้วยคำสั่ง SQL 65
รูปที่ 3.7-16 แสดงการกำหนด Foreign Key ของตาราง Customer ด้วยคำสั่ง SQL 65
รูปที่ 3.7-17 แสดงการกำหนด Foreign Key ของตาราง Employee ด้วยคำสั่ง SQL 65
รูปที่ 3.7-18 แสดงการกำหนด Foreign Key ของตาราง Contract ด้วยคำสั่ง SQL 66
รูปที่ 3.7-19 แสดงการกำหนด Foreign Key ของตาราง Privilege ด้วยคำสั่ง SQL 66
รูปที่ 3.7-20 แสดงการกำหนด Foreign Key ของตาราง Project ด้วยคำสั่ง SQL 66

รูปที่ 3.7-21 แสดงการกำหนด Foreign Key ของตาราง MapBackground ด้วยคำสั่ง SQL 66
รูปที่ 3.7-22 แสดงการกำหนด Foreign Key ของตาราง HomeType ด้วยคำสั่ง SQL 66
รูปที่ 3.7-23 แสดงการกำหนด Foreign Key ของตาราง Home ด้วยคำสั่ง SQL 66
รูปที่ 3.8-1 แสดงลำดับการทำงานของโปรแกรม 67
รูปที่ 3.8-2 แสดงหน้าต่าง Visual Composition 69
รูปที่ 3.8-3 แสดงหน้าต่าง Hierarchy 69
รูปที่ 3.8-4 แสดงตัวอย่างการสร้างหน้าจอการล็อกอินเข้าสู่ระบบ 70
รูปที่ 3.8-5 แสดงตัวอย่างการสร้างหน้าจอไว้ใช้ในการสร้างแผนผังของโครงการ 71
รูปที่ 3.8-6 แสดงตัวอย่างการสร้างหน้าจอของการเก็บข้อมูลลูกค้า 71
รูปที่ 3.9-1 แสดงคลาสไดอะแกรมของตัวโครงการ 72
รูปที่ 3.9-2 แสดงหน้าจอที่ใช้ในการสร้างแผนผังโครงการ 72
รูปที่ 3.9-3 แสดงส่วนหนึ่งของแผนผังและข้อมูลที่อยู่ในอาร์เรย์ 73
รูปที่ 3.9-4 แสดงโปรแกรมส่วนที่ใช้ในการจัดเก็บรายละเอียดต่างๆของโครงการ 74
รูปที่ 3.9-5 แสดงโปรแกรมส่วนที่จัดเก็บรายละเอียดในแผนผังลงฐานข้อมูล 75
รูปที่ 3.10-1 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการเชื่อมต่อกับฐานข้อมูลออราเคิล 76
รูปที่ 3.10-2 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการลบข้อมูล 78
รูปที่ 3.10-3 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการอัพเดตข้อมูล 79
รูปที่ 4.1-1 แสดงหน้าจอในส่วนของการล็อกอินเข้าสู่ระบบ 80
รูปที่ 4.1-2 แสดงหน้าจอที่ใช้ในการแก้ไขแผนผังโครงการ 81
รูปที่ 4.1-3 แสดงหน้าจอที่ใช้ในการเพิ่มเติมข้อมูลรายละเอียดของบ้านแต่ละหลังในโครงการ 81
รูปที่ 4.1-4 แสดงหน้าจอการให้บริการลูกค้าในการซื้อขายสั่งจองบ้าน 82
รูปที่ 4.1-5 แสดงหน้าจอการแก้ไขข้อมูลของลูกค้า 83
รูปที่ 4.2-1 แสดงหน้าจอฟ้องผิดพลาดเมื่อพนักงานใส่ข้อมูลในการล๊อกอินผิด 84
รูปที่ 4.2-2 แสดงการตรวจสอบข้อมูลของลูกค้าด้วย Oracle SQL*Plus 84
รูปที่ 4.2-3 แสดงการ Select ข้อมูลจากตาราง Project ขึ้นมาตรวจสอบ 85
รูปที่ 4.2-4 แสดงการ Select ข้อมูลจากตาราง MapBackground ขึ้นมาตรวจสอบ 85
รูปที่ 4.2-5 แสดงการ Select ข้อมูลจากตาราง Home ขึ้นมาตรวจสอบ 86
รูปที่ 4.2-6 แสดงการ Select ข้อมูลจากตาราง Contract ขึ้นมาตรวจสอบ 86

สารบัญตาราง

หน้าที่

ตารางที่ 3-1 แสดงตารางที่ใช้เก็บข้อมูลของลูกค้าที่ชื่อว่า Customer 54
ตารางที่ 3-2 แสดงตารางที่ใช้เก็บข้อมูลของพนักงานที่ชื่อว่า Employee 55
ตารางที่ 3-3 แสดงตารางที่ใช้เก็บรหัสผ่านเข้าสู่ระบบชื่อว่า Account 55
ตารางที่ 3-4 แสดงตารางที่ใช้เก็บสัญญาระหว่างลูกค้ากับโครงการชื่อว่า Contract 56
ตารางที่ 3-5 แสดงตารางที่ใช้เก็บชนิดของสัญญาชื่อว่า ContractType 56
ตารางที่ 3-6 แสดงตารางที่ใช้เก็บสิทธิเข้าสู่ระบบชื่อว่า Privilege 56
ตารางที่ 3-7 แสดงตารางที่ใช้เก็บชื่อของตำแหน่งพนักงานชื่อว่า Position 56
ตารางที่ 3-8 แสดงตารางที่ใช้เก็บชื่อของบริษัทชื่อว่า Company 57
ตารางที่ 3-9 แสดงตารางที่ใช้เก็บข้อมูลของโครงการชื่อว่า Project 57
ตารางที่ 3-10 แสดงตารางที่ใช้เก็บชนิดของโครงการชื่อว่า ProjectType 57
ตารางที่ 3-11 แสดงตารางที่ใช้เก็บแผนผังของโครงการชื่อว่า MapBackground 58
ตารางที่ 3-12 แสดงตารางที่ใช้เก็บชื่ออิมเมจ (Image) ที่ใช้แสดงในแผนผังชื่อว่า MapImage 58
ตารางที่ 3-13 แสดงตารางที่ใช้เก็บชนิดของบ้านชื่อว่า HomeType 58
ตารางที่ 3-14 แสดงตารางที่ใช้เก็บข้อมูลบ้านในโครงการชื่อว่า Home 59
ตารางที่ 3-15 แสดงตารางที่ใช้เก็บที่ใช้เก็บชื่อสถานะของบ้านชื่อว่า HomeStatus 59

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันความรู้และเทคโนโลยีในด้านคอมพิวเตอร์ ได้มีการพัฒนาอยู่ตลอดเวลา ไม่ว่าจะเป็นฮาร์ดแวร์หรือ ซอฟต์แวร์ ซึ่งถ้าจะกล่าวถึงในด้านซอฟต์แวร์แล้ว การออกแบบโปรแกรมก็ได้มีการพัฒนาไปสู่รูปแบบต่างๆ การออกแบบเชิงวัตถุ (Object-oriented) ก็เป็นการออกแบบโปรแกรมอีกรูปแบบหนึ่งที่ถูกพัฒนาขึ้นมา ที่มีคุณสมบัติดีกว่าการออกแบบโปรแกรมในรูปแบบเก่า ที่เป็นโพรซีเจอร์ (Procedure) อีกทั้งมีการพัฒนาภาษาต่างๆขึ้นมากมายเพื่อใช้ร่วมกับระบบที่ออกแบบมาเป็นออปเจ็กต์-โอเรียนเต็ด

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุ หรือที่เรียกกันว่า "ออบเจ็กต์" นั้นเป็นการมองโครงสร้างของระบบในรูปแบบใหม่ โดยจะมองเปรียบเทียบกับสิ่งที่มีอยู่จริงในโลก ซึ่งการมองโครงสร้างของระบบในรูปแบบเก่านั้นจะมีการเขียนโปรแกรมโดยแยกระบบเป็นส่วนต่างๆ แต่ก็ยังมีการใช้ข้อมูลต่างๆร่วมกัน แต่สำหรับการมองแบบ ออบเจ็กต์ นั้นจะมองเสมือนว่าสิ่งต่าง ๆ นั้นเป็นวัตถุ มันจะอิสระต่อกัน ไม่มีการใช้ข้อมูลร่วมกัน มีการซ่อนข้อมูลในตัวมันเองไว้ไม่ให้ผู้อื่นใช้ได้ หรือที่เรียกว่า Encapsulation มีการมองโครงสร้างเป็นลำดับชั้นลงไป หรือที่เรียกว่า Hierarchy มีการสืบทอดคุณสมบัติของแต่ละออบเจ็กต์ หรือที่เรียกว่า Inherit และยังนำสิ่งที่มีอยู่กลับมาใช้ใหม่ได้ หรือที่เรียกว่า Reuse สิ่งต่างๆเหล่านี้ ทำให้แนวความคิด (Concept) ของออบเจ็กต์-โอเรียนเต็ด นั้นเป็นที่แพร่หลายมากขึ้นในปัจจุบันและนั้นก็เป็นที่มาของการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ

สำหรับภาษาที่ใช้ในการพัฒนาระบบที่ออกแบบเชิงวัตถุนั้น มีภาษาอยู่หลายตัวที่สามารถเขียนเชิงออบเจ็กต์ได้ไม่ว่าจะเป็น เดลไฟล์ (Delphi), ซี++ (C++) แต่ที่เป็นที่นิยมมากที่สุดในปัจจุบันก็คือภาษาจาวา (Java) ที่มีลักษณะของภาษารองรับแนวคิดของออปเจ็กต์-โอเรียนเต็ด เช่น การทำสืบทอดแบบหลายคลาส (Multi-Inherit), การซ่อนข้อมูล (Encapsulation) เป็นต้น

แต่สำหรับการพัฒนาด้านซอฟต์แวร์ในปัจจุบันนั้น ไม่ใช่แต่เพียงการออกแบบเชิงวัตถุเท่านั้นที่มีการพัฒนาขึ้นมา ปัจจุบันบริษัททางด้านซอฟต์แวร์คอมพิวเตอร์ต่างๆได้มีการพัฒนาเครื่องมือที่ใช้ในการพัฒนาภาษาต่างๆขึ้นมามากมาย (Software Development Tool) ซึ่งสำหรับภาษาจาวานั้นก็มีบริษัทต่างๆที่ทำการพัฒนาเครื่องมือที่ใช้ในพัฒนาภาษาจาวา (Java Development Tool) ขึ้นมาเช่นกัน ไม่ว่าจะเป็น Jbuilder, J++, Visual Café และ VisualAge for Java เป็นต้น ซึ่ง Development Tool เหล่านี้จะช่วยทำให้เราสามารถพัฒนาโปรแกรมได้อย่างรวดเร็ว ไม่ว่าจะเป็นการช่วยในการสร้าง User Interface หรือว่าช่วยในการตรวจสอบ Syntax ทำให้โปรแกรมเมอร์ในปัจจุบันหันมาใช้เครื่องมือในการพัฒนาโปรแกรมเหล่านี้แทนการเขียนโปรแกรมที่อาศัยเอดิเตอร์ (Editor) ในการเขียนและนำมาคอมไพล์ (Compile) ด้วยคอมไพเลอร์ (Compiler) แบบในอดีต

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาการออกแบบโปรแกรมประยุกต์ด้วยแนวคิดการออกแบบเชิงวัตถุ ซึ่งเป็นแนวคิดใหม่ในการออกแบบระบบ

1.2.2 ศึกษาการใช้ระบบจัดการฐานข้อมูล (DBMS : Database Management System) ออราเคิล (Oracle) และการออกแบบฐานข้อมูลขึ้นมาเพื่อใช้งานร่วมกับระบบที่เป็นออบเจ็กต์-โอเรียนเต็ดได้

1.2.3 ศึกษาการเขียนโปรแกรมด้วยภาษาจาวา เพื่อใช้ในการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ ไม่ว่าจะเป็นการสร้าง User Interface, การควบคุมระบบ, การติดต่อกับรีเลชันนัลดาต้าเบส (Relational Database)

1.2.4 ศึกษาการใช้งาน VisualAge for Java คือเครื่องมือที่ช่วยในการพัฒนาภาษาจาวา (Java Development Tool) เพื่อมาช่วยในการสร้าง User Interface และจัดเก็บโปรแกรมให้เป็นระบบได้

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้เป็นการศึกษาถึงการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ (Object Oriented Application) ซึ่งเป็นการสร้างระบบจำลองขึ้นมา ที่ประกอบไปด้วย การกำหนดความต้องการของระบบ, การออกแบบระบบตามความต้องการของระบบที่ได้มาโดยจะนำมาแยกออกมาเป็นออบเจ็กต์ต่างๆ, การออกแบบฐานข้อมูลเพื่อใช้ในการจัดเก็บข้อมูล, การออกแบบหน้าจอ User-Interface, การทดสอบระบบ และสรุปผล ซึ่งระบบจำลองที่ใช้ในงานวิจัยนี้คือระบบการจัดเก็บฐานข้อมูลโครงการขายบ้านจัดสรร

ซึ่งสำหรับระบบการจัดเก็บข้อมูลโครงการขายบ้านจัดสรรนี้ จะเป็นระบบที่สามารถเพิ่มโครงการใหม่ๆให้กับระบบได้โดยไม่ต้องทำการออกแบบระบบใหม่ สามารถทำได้โดยการสร้างแผนผังโครงการใหม่ให้กับระบบ เมื่อลูกค้าคนใดสนใจที่จะซื้อบ้านภายในโครงการก็จะทำติดต่อกับเจ้าหน้าที่ของโครงการนั้น เจ้าหน้าที่ก็จะทำการแสดงตำแหน่งของบ้านหรืออาคารนั้น รวมทั้งรายละเอียดต่างๆของบ้านหรืออาคารหลังนั้นด้วยอาทิเช่น ราคา , พื้นที่ , จำนวนห้องนอน , จำนวนห้องน้ำ เป็นต้นโดยการแสดงตำแหน่งของบ้านหรืออาคารจะทำการจำลองโครงการออกมาเป็นแผนผังทางจอภาพ เพื่อให้ลูกค้าสามารถเข้าใจได้ง่ายและสามารถที่จะตัดสินใจในการซื้อได้ง่ายขึ้น ในส่วนของผู้จัดการ (Manager)โครงการหรือเจ้าหน้าที่ (Official) ที่ติดต่อกับลูกค้าจะมีหน้าที่ในการเพิ่มหรือแก้ไขข้อมูลของลูกค้า, แก้ไขข้อมูลของบ้านหรืออาคารภายในโครงการ และระบบยังมีการแสดงการซื้อขายของโครงการเพื่อเป็นการให้ข้อมูลแก่ผู้จัดการเมื่อต้องการได้อีกด้วย

และในส่วนของเจ้าหน้าที่ดูแลฐานข้อมูล (DBA : Database Administrator) จะมีหน้าที่ในการเพิ่มและแก้ไขข้อมูลของโครงการ, แก้ไขประเภทของบ้านหรืออาคารแต่ละประเภท รวมถึงกำหนดสิทธิและแก้ไขข้อมูลของเจ้าหน้าที่แต่ละคนด้วย

โดยในระบบการจัดเก็บฐานข้อมูลโครงการขายบ้านจัดสรรตัวนี้ จะถือเป็นระบบต้นแบบ (Prototype)ที่ทดลองสร้างขึ้น เพื่อศึกษาแนวความคิดในการออกแบบและพัฒนาโปรแกรมประยุกต์เชิง

วัตถุ ระบบจึงมีข้อจำกัดอยู่หลายประการเช่น การกำหนดขนาดความยาวของข้อมูล, ไม่สามารถแก้ไขหรือใช้งานได้ในบางส่วนที่ออกแบบไว้แล้ว, ความเร็วในการทำงานยังช้าอยู่ แต่ถึงอย่างไรก็ตามระบบก็ยังมีความสมบูรณ์ในระดับหนึ่งที่สามารถจะนำไปพัฒนาต่อเพื่อใช้งานจริงได้ในอนาคต

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะทำการศึกษาถึงพื้นฐานความรู้ต่างๆที่เกี่ยวข้องกับตัวโครงงาน ประกอบไปด้วย การค้นหาและกำหนดความต้องการของระบบ, การออกแบบระบบเชิงวัตถุ, การเขียนโปรแกรมเชิงวัตถุ, การใช้เครื่องมือในการพัฒนาโปรแกรม, การใช้งานระบบจัดการฐานข้อมูล และ การออกแบบกับการติดต่อกับฐานข้อมูล

โดยเราจะเริ่มด้วยการหาความต้องการของระบบการจัดเก็บฐานข้อมูลโครงการขายบ้านจัดสรรว่าประกอบไปด้วยความต้องการอะไรบ้าง, นำความต้องการของระบบที่ได้มานั้น มาออกแบบด้วยออบเจ็กต์-โอเรียนเต็ด โดยการกำหนดความต้องการของระบบออกมาในรูปของยูสเคส (Use Case) และหาออบเจ็กต์ของระบบออกมา แสดงในรูปของคลาสไดอะแกรม (Class Diagram) นำคลาสไดอะแกรมที่เป็นข้อมูลของระบบนั้นมาแปลงคลาสเป็นตาราง (Table) เพื่อใช้ในการสร้างและจัดเก็บฐานข้อมูลของระบบ แล้วก็จะทำการออกแบบและกำหนดลำดับการทำงานของระบบ เขียนโปรแกรมเพื่อสร้างคลาสไดอะแกรม ขึ้นมาตามที่ได้ออกแบบไว้

ต่อไปเราก็จะทำการออกแบบหน้าจอ User Interface ที่ใช้ในการติดต่อกับผู้ใช้และสร้างโดยใช้ VisualAge for Java และทำการเขียนโปรแกรมเพิ่มเติมเพื่อให้ระบบนั้นทำงานได้ สุดท้ายก็จะทำการทดสอบโปรแกรมที่เขียนขึ้นมา พร้อมทั้งปรับปรุงให้สมบูรณ์

บทที่ 2

ทฤษฎีและหลักการ

2.1 พื้นฐานของออบเจ็กต์-โอเรียนเต็ด (Object-Oriented)

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุ หรือที่เรียกกันว่า "ออบเจ็กต์" นั้นเป็นการมองโครงสร้างของโปรแกรมในรูปแบบใหม่ มองเปรียบเทียบกับสิ่งที่มีอยู่จริงในโลก ซึ่งการมองโครงสร้างของโปรแกรมในรูปแบบเก่านั้นจะมีการใช้ข้อมูลร่วมกัน แต่การมองแบบออบเจ็กต์ นั้นจะมองเสมือนว่าสิ่งต่าง ๆ นั้นเป็น ออบเจ็กต์ มันจะมีอิสระจากกัน ไม่ใช้ข้อมูลร่วมกัน มีการซ่อนของข้อมูลในตัวมันเอง ใช้ข้อมูลในส่วนของมัน มีการมองโครงสร้างเป็นลำดับชั้นลงไป มีการสืบทอดคุณสมบัติของแต่ละออบเจ็กต์ รวมไปถึงการนำสิ่งที่มีอยู่กลับมาใช้ใหม่ สิ่งต่างๆเหล่านี้ คือที่มาของการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ

2.1.1 เทคโนโลยีออบเจ็กต์-โอเรียนเต็ด (Object-Oriented Technology)

ในการพัฒนาระบบแบบโครงสร้างนั้น จะพยายามที่ให้นักพัฒนาระบบแยกปัญหาและแก้ปัญหาเป็นส่วน ๆ ซึ่งในการพัฒนาระบบแบบออบเจ็กต์โอเรียนเต็ด นั้นก็เช่นกันแต่จะมองปัญหาเป็นลักษณะของออบเจ็กต์ ซึ่งจะมีความใกล้ชิดกันน้อย ทำให้การดูแลและแก้ไขส่วนต่าง ๆ ของระบบ หรือออบเจ็กต์สามารถทำได้ง่าย และ มีผลกระทบต่อส่วนอื่น ๆ ของระบบน้อยที่สุด ด้วยการนำหลักการของ ออบเจ็กต์โอเรียนเต็ดมาใช้ จะทำให้ การออกแบบและพัฒนาซอฟต์แวร์สามารถตรวจสอบความผิดพลาดได้ง่าย , ช่วยเพิ่มประสิทธิภาพในการเอ็กซิคิวต์ (execute) คำสั่ง , ลดชนิดของคำสั่งของฮาร์ดแวร์ , มีการคอมไพล์จากภาษาระดับสูงเป็นภาษาระดับต่ำที่ไม่ซับซ้อนและยังช่วยลดพื้นที่ในการเก็บข้อมูลอีกด้วย

2.1.2 ออบเจ็กต์

ออบเจ็กต์เป็นกลุ่มของเอนติตี้ (Entity) ที่ประกอบด้วยคุณสมบัติ หรือ แอตทริบิวต์ (attribute), พฤติกรรมหรือเมธอด (method) และสถานะด้วย

การซ่อนรายละเอียด เป็นกระบวนการในการซ่อนข้อมูลและความซับซ้อนภายในออบเจ็กต์ ที่ไม่ต้องการที่จะให้ผู้ใช้หรือแม้แต่ผู้พัฒนาระบบเองเห็น โดยแต่บอกว่าออบเจ็กต์ทำอะไรได้ แต่ไม่จำเป็นต้องบอกว่าทำได้อย่างไร ด้วยหลักการนี้จึงทำให้ต้องแบ่งคลาส ออกเป็น 2 ส่วนคือส่วนอินเตอร์เฟซ (Interface) และส่วนอิมพลีเมนเทชั่น (Implementation) ซึ่งส่วนของอินเตอร์เฟซนั้นจะบอกว่าคลาสนั้นสามารถทำอะไรได้ ซึ่งจะเป็นมุมมองจากภายนอก และ ส่วนอิมพลีเมนเทชั่น จะเป็นส่วนที่บอกว่าคลาสนั้นทำงานได้อย่างไร ซึ่งจะเป็นมุมมองจากภายใน

คลาสคือการแยกแยะเอกลักษณ์พื้นฐานจากกลุ่มของออบเจ็กต์ที่เหมือนกัน คลาสจะเป็นตัวกำหนดแอททริบิวต์และเมธอดของออบเจ็กต์ซึ่งเป็นอินสแทนซ์ (Instance) แต่คลาสไม่มีความรับผิดชอบคุณสมบัติทั้งหมดของออบเจ็กต์ ซึ่งจะอยู่ในรูปของชนิด (type) ไม่ใช่ค่า (value)

2.1.3 ความสัมพันธ์ระหว่างคลาส (Class Relationships)

2.1.3.1 แอซโซซิเอชั่น (Association)

เป็นความสัมพันธ์ระหว่างออบเจ็กต์หนึ่งที่ทำการเรียกใช้บริการจากอีกออบเจ็กต์หนึ่ง ซึ่งการเรียกใช้จะอยู่ในรูปของแมสเซจ (Message)

2.1.3.2 การสืบทอด (Inheritance)

ในความสัมพันธ์แบบนี้ จะอยู่ในรูปของ "is a" คือคลาสหนึ่งจะมีคุณสมบัติของอีกคลาสหนึ่งอยู่นั่นหมายความว่าคลาสลูกจะมีคุณสมบัติทุกอย่างที่คลาสแม่มี ซึ่งคลาสลูกอาจจะมีคุณสมบัติและแอทตริบิวต์ที่เพิ่มขึ้นมาได้ หรือสามารถที่จะมีคุณสมบัติจากหลายคลาสอีกด้วย (Multi Inheritance) ซึ่งจะทำให้คลาสลูกมีคุณสมบัติมากกว่าคลาสพ่อแม่

2.1.3.3 อะกรีเกชั่น (Aggregation)

ในความสัมพันธ์แบบนี้จะอยู่ในรูปของ "part of" ซึ่งคลาสหนึ่งจะบรรจุอีกคลาสไว้ทั้งในทางความคิดและทางกายภาพ คลาสที่ใหญ่กว่านั้นจะเรียกว่าเจ้าของ (Owner,Whole) ส่วนคลาสที่เล็กกว่าจะเรียกว่า คลาสคอมโพเนนท์ (Component) โดยทั่วไป คลาสเจ้าของจะมีหน้าที่ในการสร้าง และ ทำลายคลาสคอมโพเนนท์

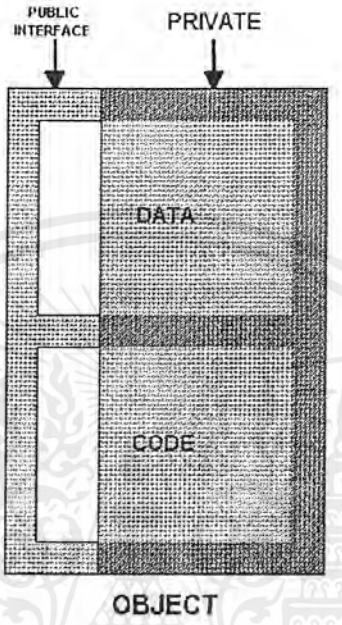
2.1.4 โพลิมอร์พิซึม (Polymorphism)

เป็นโอเปอเรชั่น (operation) ที่มีชื่อเรียกที่เหมือนกันแต่มีการทำงานที่ต่างกัน ซึ่งโพลิมอร์พิซึมจะแบ่งออกเป็น 2 แบบคือสเตติก (static) คือ คอมไพล์จะเลือกโอเปอเรชั่นขึ้นมาทำงานได้ตั้งแต่เวลาคอมไพล์โปรแกรม ส่วนอีกแบบคือ ไดนามิก (dynamic) คือ ตัวโปรแกรมจะทำหน้าที่ในการตัดสินใจว่าจะเลือกโอเปอเรชั่นใดขึ้นมาทำงานในเวลาที่รันโปรแกรมแล้ว โดยในการเลือกว่าจะให้โอเปอเรชั่นใดขึ้นมาทำงานนั้นจะขึ้นอยู่กับชนิดและจำนวนของพารามิเตอร์ (parameter) ในการเรียกใช้งาน

2.1.5 คุณสมบัติของ ออบเจ็กต์โอเรียนเทชัน (Object Orientation) ในการจัดการกับข้อมูล

ออบเจ็กต์โอเรียนเทชันเป็นรูปแบบที่มีการจัดการกับข้อมูล และ code ในรูปแบบใหม่ ไม่เหมือนเดิม เป็นการเปลี่ยนแปลงต่อการมองภาพรวมในส่วนโครงสร้างของ ซอฟต์แวร์ เป็นรูปแบบของการทดลองนำสิ่งที่มีอยู่จริงในโลกมาวิเคราะห์ ออบเจ็กต์ ในโลกความจริงเช่น คน, บ้าน , รถ , บริษัท , โลก

สิ่งเหล่านี้จะไม่มีการใช้ ข้อมูล ร่วมกัน นกและรถทั้งคู่ต่างก็มีตำแหน่งและทิศทางของการเคลื่อนที่ แต่ก็ไม่ได้เคลื่อนที่ไปในทางเดียวกัน ใน Model ของ ออบเจ็กต์ เราจำเป็นจะต้องมีการอธิบายถึงองค์ประกอบและ คุณสมบัติ เราจะทำการพัฒนาซอฟต์แวร์ โดยการกำหนดคลาสของออบเจ็กต์, กำหนดข้อมูลที่จำเป็นต้องใช้ และเมดตอดที่เตรียมให้แก่ ออบเจ็กต์



รูปที่ 2.1-1 แสดง ข้อมูลในส่วนที่ Public และ Private ของ ออบเจ็กต์

เพื่อแสดงคุณสมบัติ (behavior) ของมัน เมื่อเราทำการกำหนด ออบเจ็กต์ และ เตรียมในส่วนที่แสดงการติดต่อซึ่งกันและกันแล้ว มันก็เหมือนเราได้จัดการกับส่วนที่สำคัญของการพัฒนาเรียบร้อยแล้ว สำหรับการติดต่อข้อมูลจะมีส่วนที่จัดการกับข้อมูล ในการติดต่อกับข้อมูลนั้นเราจะต้องผ่าน "Public Interface" ส่วนอื่นที่ถูก private ไว้เราไม่สามารถจะเข้าถึงข้อมูลเหล่านั้นได้โดยตรงเรียกว่าการ Encapsulation เป็นวิธีการที่สำคัญในการซ่อนข้อมูล มันทำให้การติดต่อสื่อสารนั้นมีความคล่องตัว และควบคุมได้ง่าย ซึ่งจากรูปจะแสดงให้เห็นภาพว่าส่วนที่ประกาศ public นั้นสามารถจะเข้าไปจัดการตรงจุดนั้นได้ แต่ส่วนของ private ออบเจ็กต์ อื่นๆที่ไม่ใช่ตัวมันเองไม่สามารถเข้าไปยุ่งเกี่ยวได้

2.2 การเขียนโปรแกรมเชิงวัตถุโดยใช้ จาวา

ลักษณะเด่นของจาวา

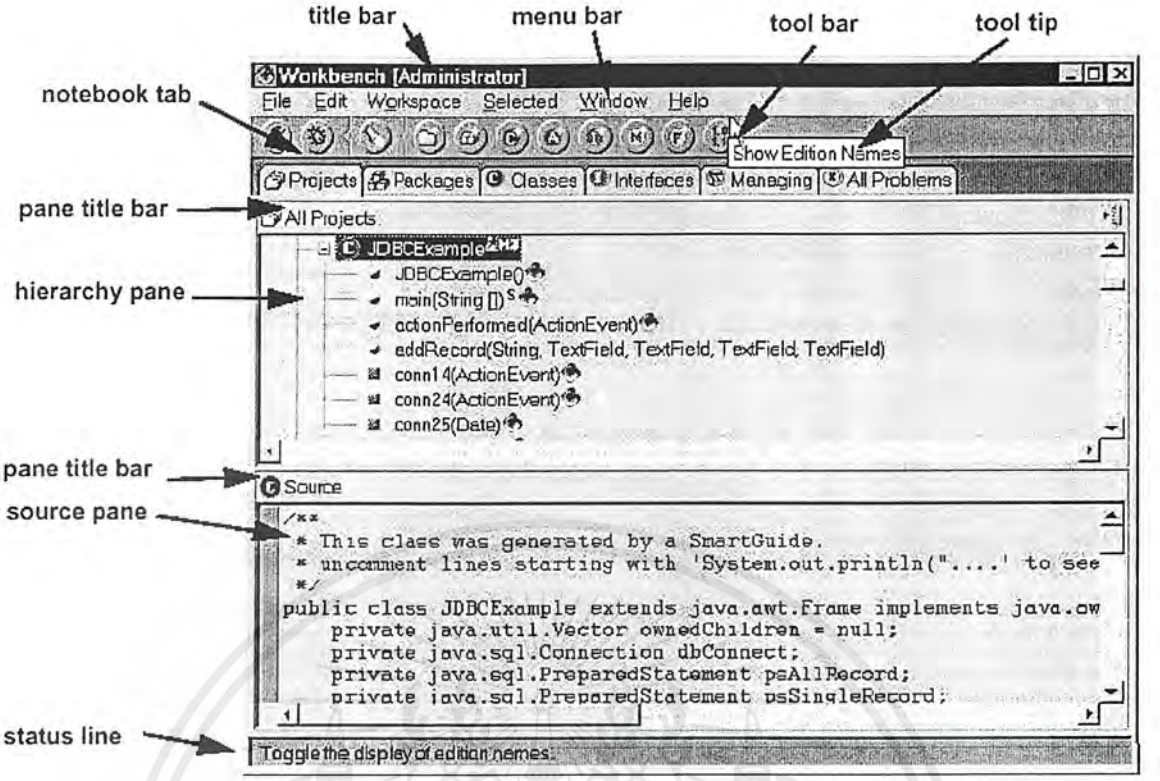
- **ง่าย** จาวาพยายามที่จะทำให้ใช้งานได้ง่าย ในขณะที่พยายามจะรักษาประสิทธิภาพ ซึ่งจะพยายามไม่เพิ่มคุณสมบัติที่ไม่จำเป็นเข้าไป
- **ใช้วิธีการออบเจ็กต์โอเรียนเท็ด** ทุกสิ่งที่อยู่ในจาวา มักจะอยู่ในรูปของ คลาส , เมธอด หรือ ออบเจ็กต์ จะมีก็เพียงแต่ประเภทของข้อมูลพื้นฐานส่วนใหญ่เท่านั้นที่ยังใช้ แบบเดิมอยู่

- **ทำงานได้กับหลายแพลตฟอร์ม (platform)** โปรแกรมจาวา จะทำการคอมไพล์ให้อยู่ในรูปของไบต์โค้ด (ByteCode) ซึ่งสามารถที่จะทำงานได้โดยอินเตอร์พลีเตอร์ (Interpreter) ที่อยู่บนหลายแพลตฟอร์ม ซึ่งรวมถึง วินโดวส์ 95 (Windows 95) , วินโดวส์เอ็นที (Windows NT) , โซลาริส (Solaris) , เอไดเอกส์ (AIX) และ โอเอสทู (OS/2)
- **มีประสิทธิภาพ** จาวาสามารถคอมไพล์ไปเป็นโค้ดเดิมได้
- **ทำงานแบบมัลติเทรด (MultiThread)** เทรดดิ้ง (Threading) จะสนับสนุนการสร้างภาษาโปรแกรมจาวาที่มีเพียงตัวเดียวก็สามารถที่จะมีการประมวลผลงานที่แตกต่างกันโดยที่ไม่เกี่ยวข้องกันและต่อเนื่องกัน

2.3 VisualAge for Java 2.0 Enterprise Edition (จากบริษัทไอบีเอ็ม)

สำหรับ Java Development Tool ที่ใช้ในการเขียนโปรแกรมและออกแบบ User Interface นั้นผู้เขียนได้เลือกใช้ทูล (Tool) ชื่อว่า VisualAge for Java ซึ่งเป็นทูลที่พัฒนาขึ้นโดยบริษัท IBM เป็น Enterprise Edition Version 2.0

ทูลที่ใช้ในการพัฒนาภาษาจาวานั้นมีด้วยกันหลายตัวไม่ว่าจะเป็น Visual Café จาก บริษัท Symantec, Visual J++ จาก Microsoft, Jbuilder จาก บริษัท Borland, รวมไปถึง VisualAge for Java จาก IBM ซึ่งล้วนแต่เป็นทูลที่เป็นเครื่องมือที่ช่วยในการพัฒนาโปรแกรมที่เป็นออบเจ็กต์-โอเรียนเต็ดที่ใช้ภาษาจาวา ช่วยให้เราสามารถเขียนโปรแกรมได้อย่างสะดวกมากขึ้นมีการจัด Windows หรือหน้าจอต่างๆ ไว้แก้ไขโปรแกรมที่ง่ายต่อการเข้าใจ เมื่อเทียบกับการใช้ JDK ร่วมกับเอดิเตอร์ทั่วไป แต่อาจจะมีข้อเสียตรงที่จะช้า มีการแสดงการจัดกลุ่มของแต่ละคลาสให้อยู่ในแพ็กเกจ (Package) รวบรวมแพ็กเกจต่างๆเป็นโปรเจ็ค และ ในแต่ละคลาสก็แบ่งเป็นเมธอดต่างๆ ซึ่งถ้าเทียบกับการใช้เอดิเตอร์ปรกติทั่วไปแล้วอาจจะทำให้ผู้พัฒนาเกิดความยุ่งยาก ซับซ้อน และสับสนได้เมื่อเกิดการแก้ไข, รวมไปถึงข้อดีในการออกแบบ User Interface ที่สามารถทำได้ในเวลาที่รวดเร็ว ไม่จำเป็นต้องมาเขียนโค้ดในส่วนนั้น โดยตัวทูลจะทำการแปลงโค้ดออกมาให้เราเอง เมื่อมีปัญหาและต้องการแก้ไข ส่วนของหน้าจอก็สามารถที่จะทำได้ง่าย และรวดเร็ว แต่การใช้ทูลในการพัฒนาเหล่านี้ ผู้ใช้ควรที่จะมีความรู้พื้นฐาน เกี่ยวกับภาษาจาวา อยู่พอสมควรมิเช่นนั้นแล้วก็จะทำให้เกิดการไม่เข้าใจ หรือเกิดปัญหาขึ้นได้ได้ขณะใช้งาน เพราะตัวทูลเองไม่ได้ช่วยจัดการเกี่ยวกับโค้ดให้กับเราทุกอย่าง เราจะต้องทำการแก้ไขเองด้วย



รูปที่ 2.3-1 แสดงให้เห็นหน้าจอการทำงานของ VisualAge

จากที่กล่าวไปแล้วในขั้นต้น ต่อจากนี้จะขอกล่าวถึงส่วนต่างๆ และการใช้งานของ VisualAge ที่สำคัญๆ โดยแบ่งออกเป็นส่วนหลักๆดังต่อไปนี้

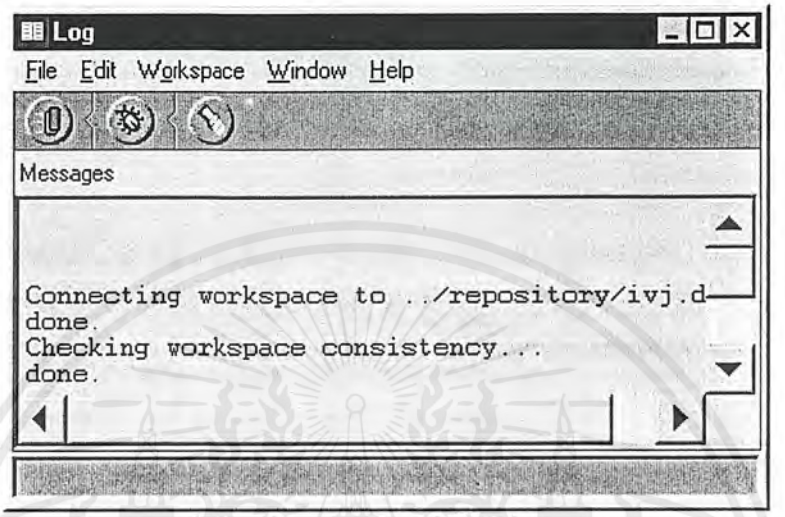
- วินโดว์หลักต่างๆ เช่น Workbench, Console, Debugger, Log เป็นต้น
- การเซต เช่น การเซตหน้าจอ, เซตฟอนต์ เป็นต้น
- การจัดการเกี่ยวกับคลาสต่างๆเช่น การ Import, Export, คอมไพล์ เป็นต้น
- เทคนิคการใช้งานต่างๆ

ซึ่งก่อนจะไปดูรายละเอียดต่างๆ ก็ขอกล่าวถึงคำศัพท์ที่จะต้องใช้ เพื่อให้เข้าใจ

- Workspace : โค้ดหรือโปรแกรมที่เขียนขึ้นจะถูกเก็บไว้ใน Workspace ดังนั้นทุกครั้งหลังจากที่ทำการเขียนโปรแกรมเสร็จและ ต้องการออกจากโปรแกรม ก็จะต้องทำการเซพข้อมูลทั้งหมดลง Workspace ก่อน
- Log : เป็นวินโดว์ที่ผู้ใช้จะเปิดขึ้นมาเพื่อทำการเซพ Workspace ให้โดยอัตโนมัติ เมื่อผู้ใช้ออกจากโปรแกรม โดยจะต้องปิดวินโดว์ที่เปิดขึ้นโดย VisualAge for Java ให้หมดก่อนแล้วค่อยปิดวินโดว์ Log
- Workbench : เป็นวินโดว์ที่เป็นหน้าจอหลักในการแก้ไขโปรแกรมทั้งหมด
- Console : เป็นหน้าจอที่ใช้แสดงเอาต์พุทหรือรับอินพุทของระบบ
- Repository : ข้อมูลที่ถูกลบไป จะเก็บไว้ใน Repository สามารถนำกลับมาใช้ใหม่ได้

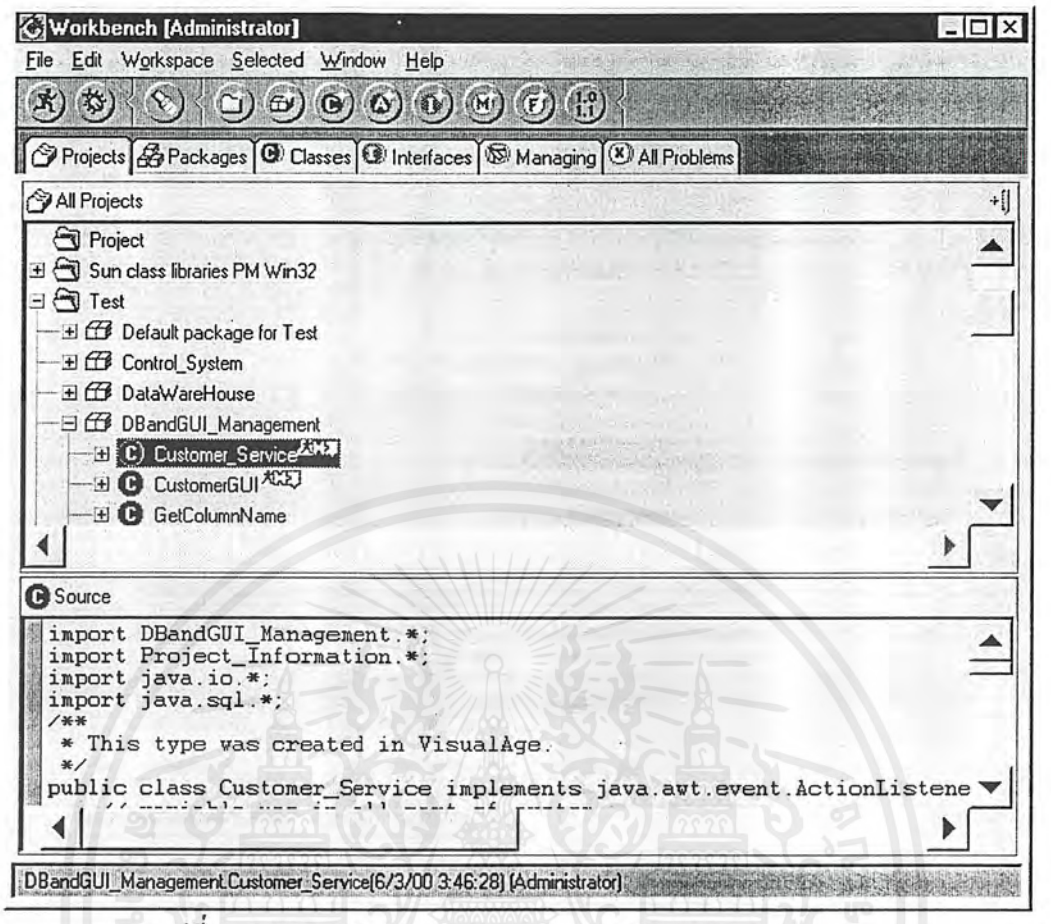
- Import : เป็นการรับไฟล์ .class หรือ java จากภายนอกเข้าสู่ระบบ
- Export : เป็นการ Export ไฟล์ .class หรือ java ออกไปยังไดเรกทอรี่ที่เราต้องการ
- Debugger : วินโดว์ที่ใช้ในการแสดง จุดที่เกิดผิดพลาดขึ้นว่ามีข้อผิดพลาดตรงส่วนไหนของ โปรแกรม

ซึ่งมีรายละเอียดและตัวอย่างหน้าจอต่างๆ แสดงให้ดูดังต่อไปนี้



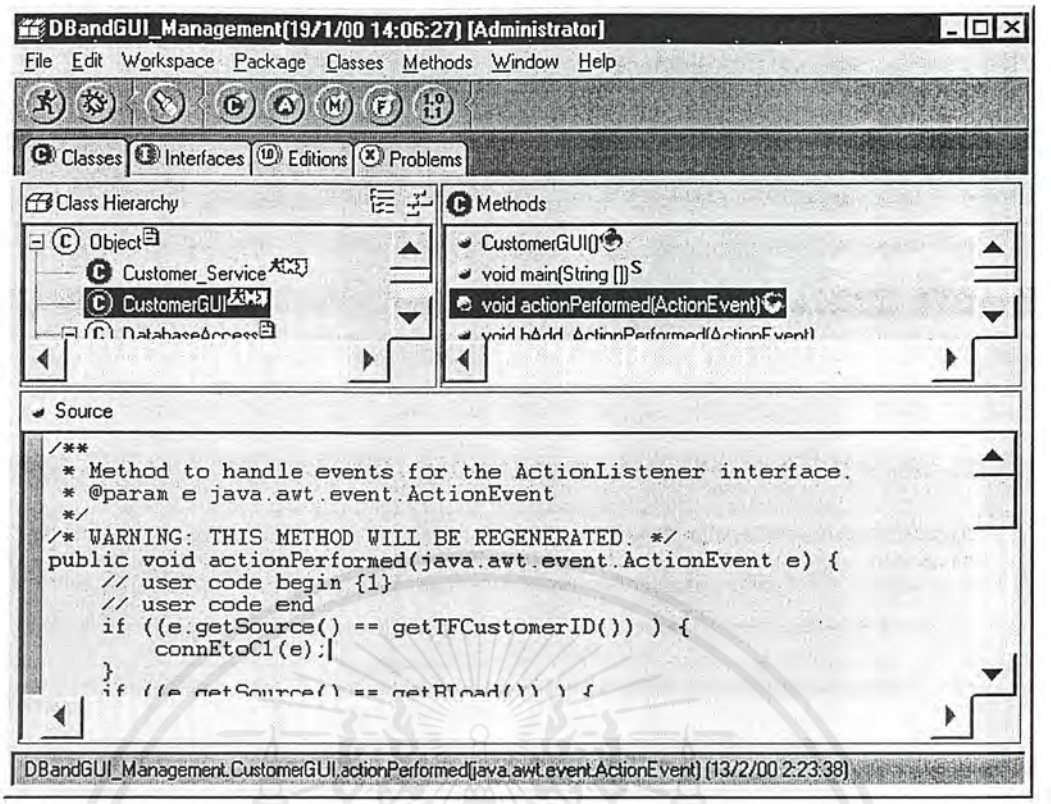
รูปที่ 2.3-2 แสดง Log Windows

จากรูปที่ 2.3-2 แสดง Log Windows ที่ใช้ในการช่วยเซพข้อมูลให้อัตโนมัติ และปิดระบบได้ถูกต้อง เพราะการจะใช้ VisualAge for Java นั้นจะมีหน้าจอถูกเปิดขึ้นมาหลายหน้าจอมาก ฉะนั้นเราจะทราบได้ว่า ระบบถูกปิดทั้งหมดแล้ว และได้ทำการเซพข้อมูลทั้งหมดลง Workspace แล้วก็โดยการเปิด Log Windows ขึ้นมาเราจะต้อง ปิดวินโดว์ต่างๆให้หมดก่อนแล้วจึงจะปิด Log Windows ได้ และออกจาก VisualAgeได้ พร้อมกับเซพข้อมูลทั้งหมดลง Workspace



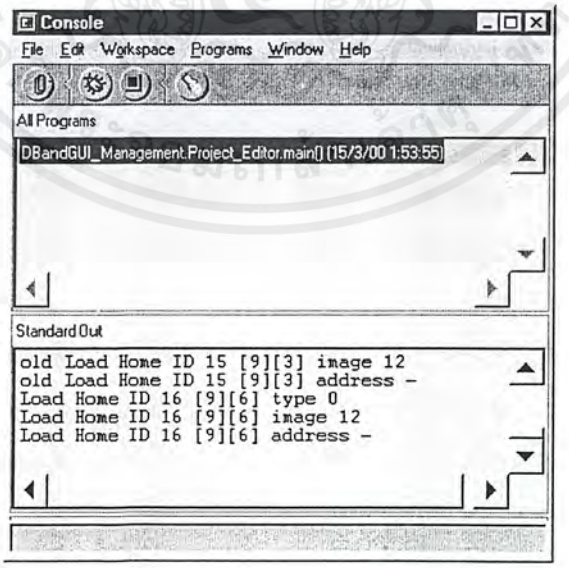
รูปที่ 2.3-3 แสดง Workbench Windows ของ VisualAge for Java

จากรูปที่ 2.3-3 แสดง Workbench Windows ของ VisualAge for Java ที่ใช้เป็นหน้าจอหลักในการทำงาน ไม่ว่าจะเป็นการเพิ่ม Project, Package, Class, Method หรือว่า Field ต่างๆ เราสามารถเห็นโครงสร้างของระบบทั้งหมดได้ที่หน้าจอนี้ และถ้าต้องการที่จะแก้ไขโปรแกรมในคลาสต่างๆก็สามารถที่จะทำได้โดยดับเบิ้ล คลิ้กที่คลาสที่เราต้องการจะแก้ไข เราก็จะสามารถเข้าไปแก้ไขในคลาสนั้นได้โดยอิสระอีกวินโดว์หนึ่งได้เลย แสดงดังรูปที่ 2.3-4



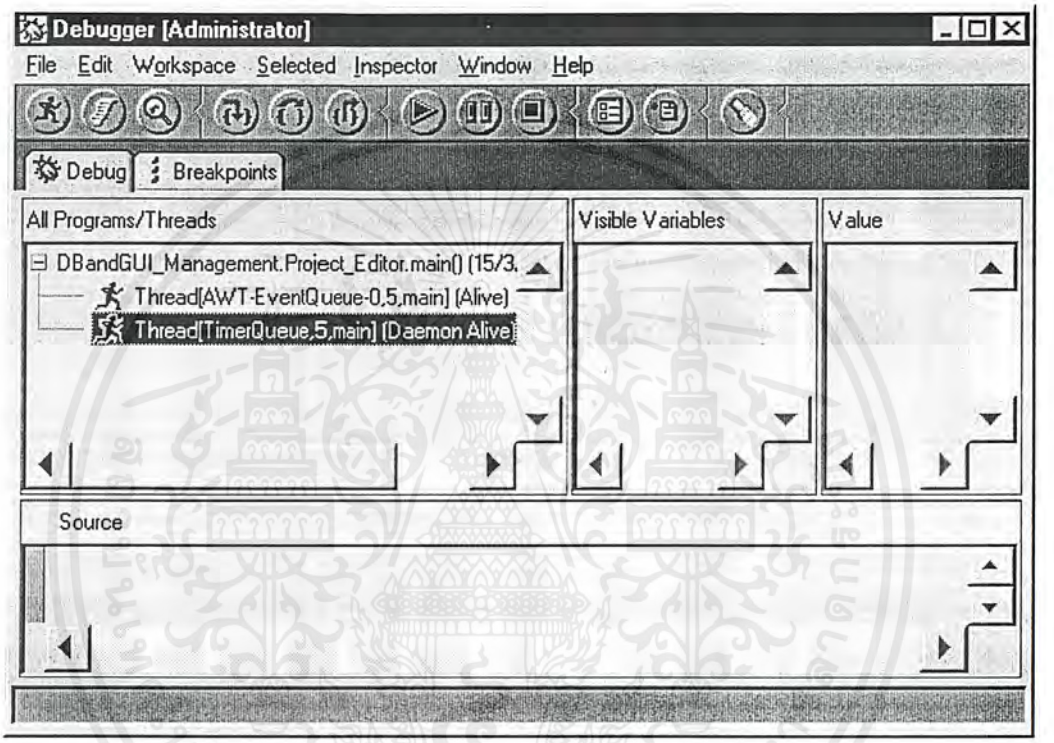
รูปที่ 2.3-4 แสดงการแก้ไขโปรแกรมโดยใช้วินโดว์คนละตัวกับ Workbench

จากรูปที่ 2.3-4 แสดงให้เห็นการแก้ไขโค้ดของโปรแกรมโดยใช้หน้าจอแยกออกมาจาก Workbench ซึ่ง Workbench นั้นเราจะใช้ในการแก้ไขระบบเช่น Create Class , Create Project เป็นต้น ซึ่งจะทำให้การแก้ไขระบบในส่วนต่างๆของเรานั้นสามารถทำได้ง่ายขึ้น ไม่เกิดความสับสน



รูปที่ 2.3-5 แสดง Console Windows

จากรูปที่ 2.3-5 แสดง Console Windows ที่ใช้ร่วมกับการทดสอบโปรแกรมที่ใช้อินพุทและเอาท์พุทของ ระบบ เอาท์พุทของระบบจะแสดงผลลัพธ์ที่เกิดจากการใช้คำสั่ง System.out.println ส่วนอินพุทของระบบ ก็เช่นการรับอินพุทจากคีย์บอร์ดและแสดงสภาวะของโปรแกรมให้เราดูด้วย เราสามารถที่จะยกเลิกโปรแกรมที่มาเรียกใช้ I/O ของระบบจาก Console Windows ได้ อีกทั้งอาจจะใช้ Console Windows เป็นส่วนที่ใช้ในการทดสอบโปรแกรมเพื่อพิมพ์ผลลัพธ์ที่ต้องการทดสอบออกทางหน้าจอ Console Windows นอกจากหน้าจอ Console Windows แล้วก็ยังมีหน้าจอที่ใช้ดีบัก (debug) ตัวโปรแกรมอีกคือหน้าจอ debugger แสดงดังรูปที่ 2.3-6

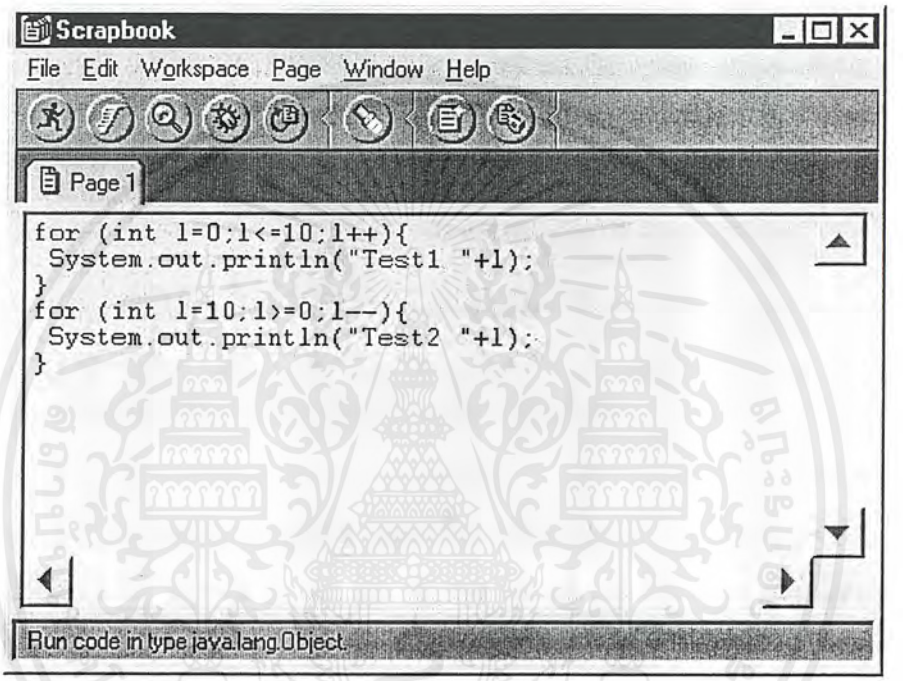


รูปที่ 2.3-6 แสดง Debugger Windows

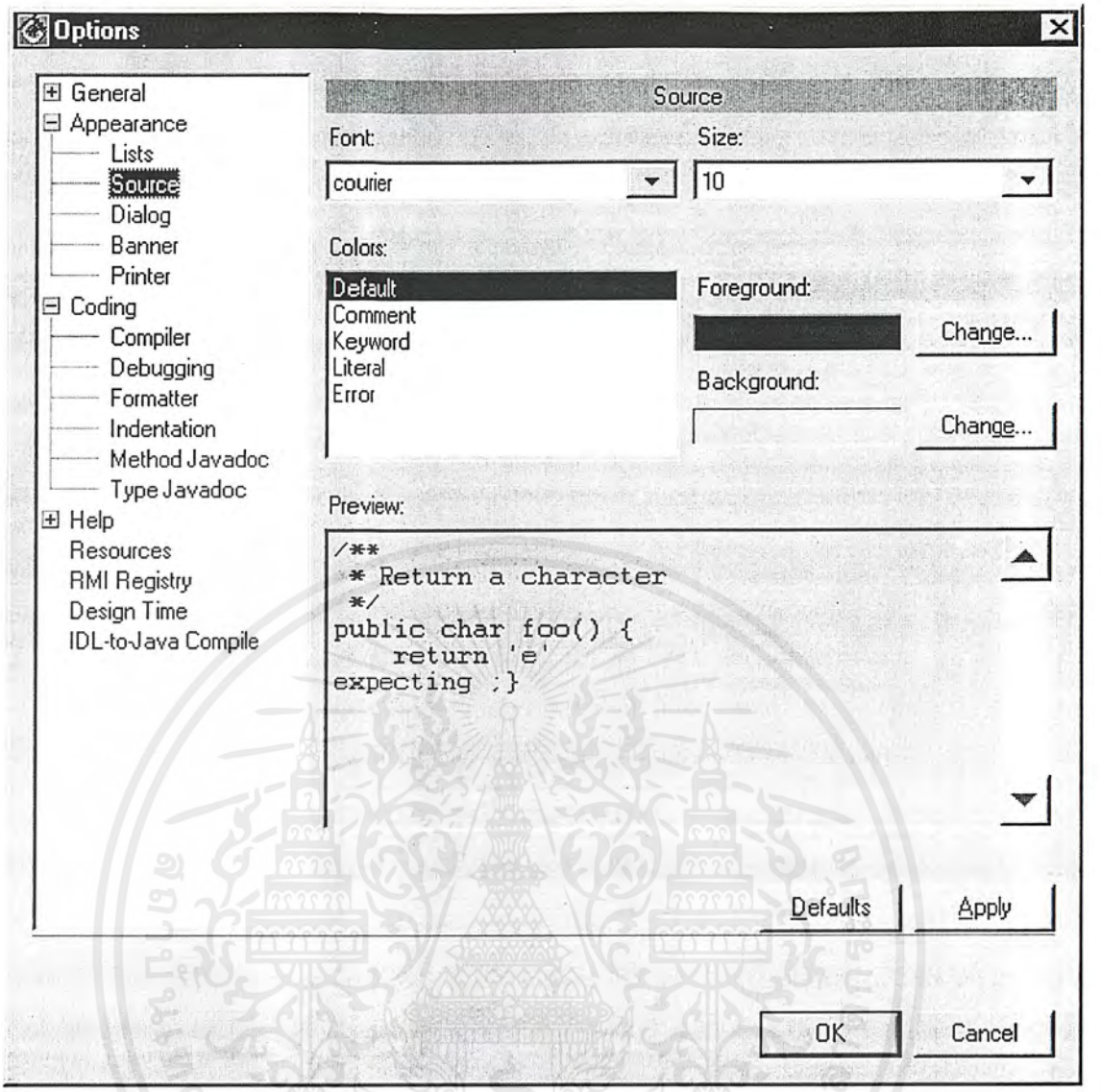
จากรูปที่ 2.3-6 แสดง Debugger Windows ที่ใช้ในการหาข้อผิดพลาดของโปรแกรมโดยเมื่อเราทำการคอมไพล์และรันโปรแกรมตัวใดขึ้นมาตัวดีบักเกอร์ (Debugger) ก็จะแสดงสภาวะการทำงานของโปรแกรมให้เราดู ถ้าเกิดข้อผิดพลาดขึ้นกับระบบหรือโปรแกรม ตัวดีบักเกอร์เองก็จะฟ้องและแสดงข้อผิดพลาดให้เราเห็น และสามารถไปแก้ไขข้อผิดพลาด ดังกล่าวนั้นได้ถูกต้อง นอกจากดีบักเกอร์แล้ว ถ้ากรณีที่เราทำการเขียน โปรแกรม ขึ้นมาโปรแกรมหนึ่ง หรือ สร้างคลาสขึ้นมา 1 คลาสหรือเราต้องการที่จะทดสอบตัวคลาสทั้งหมดของระบบว่าเมื่อนำมารวมกันแล้ว จะเกิดปัญหาอะไรหรือไม่ เราสามารถจะทำได้โดย ใช้ Scrapbook Windows ซึ่งเป็นตัวที่จะช่วยให้เราสามารถจะทดสอบตัวโปรแกรม หรือคลาสต่างๆ ได้ โดยไม่จำเป็นต้องไปเขียน โปรแกรม เพื่อทดสอบคลาสดังกล่าวได้โดยง่าย โดยใช้ Scrapbook Windows การใช้งานง่ายๆของ Scrapbook Windows มีดังนี้

- ทำการเขียนโค้ดที่เราต้องการทดสอบลงไปยังหน้าจอ Scrapbook Windows
- ลากแทบสว่างลงไปยังบริเวณโค้ดที่เราต้องการจะทดสอบและทำการรัน
- ตัว Scrapbook จะทำการรันและแสดงผลลัพธ์ออกมา ถ้าเกิดข้อผิดพลาดขึ้นบริเวณใด หรือ บรรทัดใด ตัว Scrapbook จะทำการคอมเมนท์เอาไว้ให้เรา
- เราก็จะทำการแก้ไขโค้ดที่ผิดพลาดนั้น และรันใหม่เพื่อให้ได้ผลลัพธ์ที่ถูกต้องตามที่ต้องการ

รูปของ Scrapbook Windows แสดงดังรูปที่ 2.3-7 เป็นตัวอย่างของการทดสอบโปรแกรมพิมพ์ตัวเลขจาก 1 ไป 10 และ 10 ไป 1

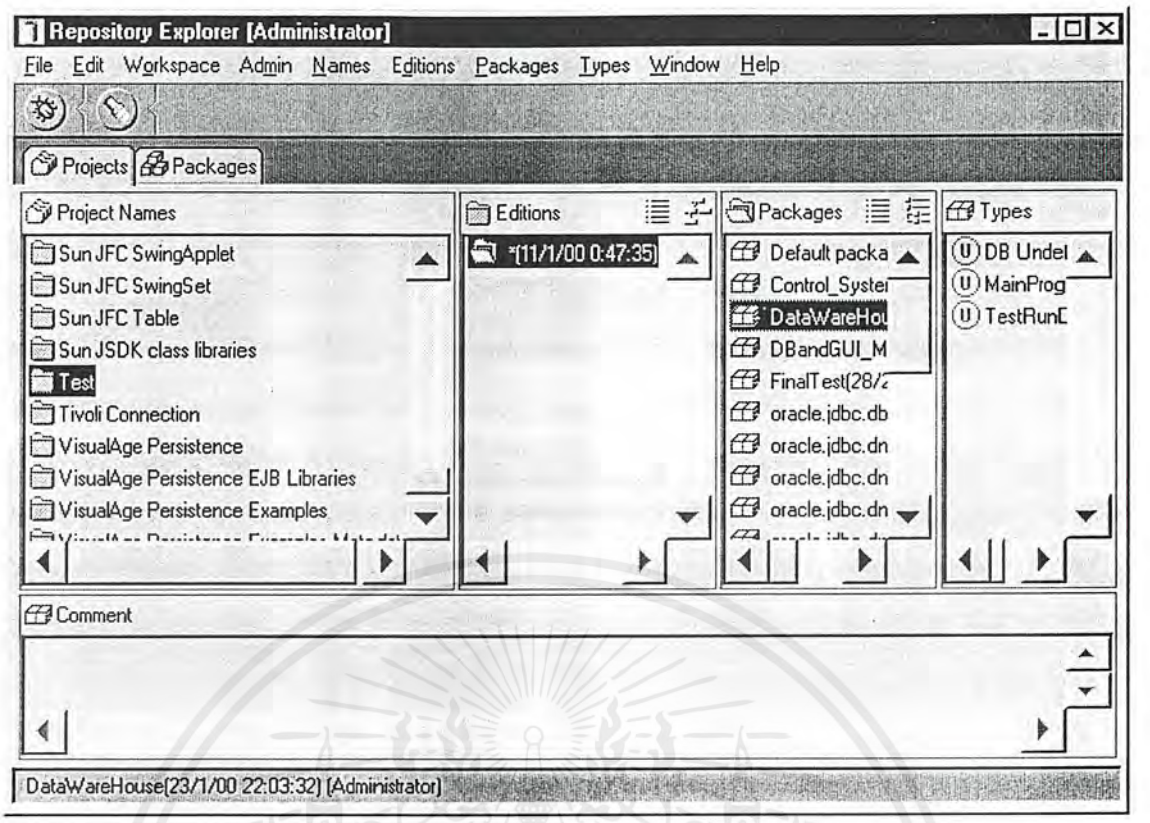


รูปที่ 2.3-7 แสดง Scrapbook Windows



รูปที่ 2.3-8 แสดง Option Windows

จากรูปที่ 2.3-8 แสดง Option Windows โดย เป็นวินโดว์ที่ใช้ในการเซตส่วนต่างๆของ VisualAge for Java เช่น เซตฟอนต์ของเมนู, โปรแกรม หรือว่าเซตค่าสีต่างๆ และรายละเอียดอื่นๆอีกมากมาย เป็นต้น



รูปที่ 2.3-9 แสดง Repository Explorer

จากรูปที่ 2.3-9 แสดง Repository Explorer ที่ใช้ในการจัดการ Repository ของระบบ โดยในการทำงานกับ VisualAge for Java นั้น ตัว VisualAge จะไม่ทำการแปลงโค้ดต่างๆที่เราเขียนออกมาเป็น .java ให้กับเรา แต่ตัว VisualAge จะทำการเก็บรวบรวม file เหล่านี้ไว้ใน Workspace ของระบบคือไฟล์ ivj.dat เมื่อเราทำการลบ file โปรแกรม ต่างๆที่เราเขียนขึ้นมาทิ้งไป โปรแกรม เหล่านั้นจะไม่ถูกลบทิ้ง แต่จะถูกนำไปเก็บไว้ใน Repository ซึ่งเราสามารถจะดึงไฟล์เหล่านี้กับมาใช้งานได้อีกโดยทำการสร้างคลาส หรือเมธอดแต่แทนที่เราจะสร้างมันขึ้นมาใหม่ เราจะไปทำการดึงมาจาก Repository แทน ซึ่งจะมีปุ่มให้เลือกอยู่ตอนที่เราสร้างคลาส

2.3.1 คุณสมบัติของVisualAge for Java Enterprise Edittion

เป็นเครื่องมือที่ใช้ในการพัฒนาภาษาจาวา ให้ใช้ในการออกแบบการเชื่อมต่อเครื่องไคลเอ็นต์ (Client)กับข้อมูลบนเครื่องเซิร์ฟเวอร์ (Server), ใช้ในการติดต่อสื่อสาร, และ พัฒนาโปรแกรมประยุกต์ทั่วไป

- ประกอบไปด้วยความสามารถทั้งหมดของ Professional Edition และคุณสมบัติที่สูงกว่า
- สนับสนุนการเขียนโปรแกรมจาวาแบบเป็นทีม
- เพิ่มเติมส่วนเครื่องมือที่ช่วยในการออกแบบ , คอมไพล์เลอร์ที่มีประสิทธิภาพสูง และตัวรีโมต ดีบักเกอร์ (Remote Debugger)

- เครื่องมือที่ช่วยในการติดต่อฐานข้อมูล
- เพิ่มส่วน Servlet Builder
- เครื่องมือในการช่วยแปลงจากออบเจ็กต์ เป็น รีเลชันนัล
- สนับสนุน SanFrancisco, Tivoli, Lotus และ Component Broker
- สนับสนุน AIX Development Environment

2.3.2 การติดต่อกับฐานข้อมูลรีเลชันนัลด้วย VisualAge for Java

VisualAge for Java นั้นรองรับการติดต่อกับฐานข้อมูลรีเลชันนัลผ่าน JDBC โดยเราสามารถติดต่อกับข้อมูลเหล่านั้นได้ทั้งที่อยู่ในแอพเพล็ต (Applet) และแอลพลิเคชัน (Application) โดยใช้ Data Access beans บน Visual Composition Editor beans paletteและถ้าเป็น VisualAge for Java Enterprise Edition เราก็สามารถติดต่อกับฐานข้อมูลรีเลชันนัลได้โดยผ่าน Data Access Builder Data Access beans นั้นเป็นข้อพิเศษอีกอย่างหนึ่งของ Visual Age for Java ซึ่งประกอบไปด้วย Select bean และ DBNavigator bean

2.3.3 Select bean

มันเป็น non – visual bean การใช้ Select bean เราสามารถที่จะ Query ฐานข้อมูลแบบรีเลชันนัลได้ โดยเพิ่ม,แก้ไข,ลบข้อมูลใน Result Set และคอมมิท (Commit) การเปลี่ยนแปลงที่เกิดขึ้นในฐานข้อมูลได้ เวลาที่เราใช้ Select bean เราจะต้องระบุคุณสมบัติในการติดต่อกับฐานข้อมูลรีเลชันนัลให้ถูกต้อง เช่นเวลาที่เราต้องการข้อมูลแถวใดๆจากตารางนั้นต้องระบุ Query Property ที่ใช้ในส่วนของการติดต่อ (Connect) กับเซิร์ฟเวอร์และกำหนด SQL ด้วย

- ในการติดต่อนั้นเราจะต้องระบุคุณสมบัติพิเศษต่างๆสำหรับ Select bean เช่น ค่า URL สำหรับการติดต่อและ User ID กับรหัสผ่าน (Password) ที่ส่งไปเพื่อร้องขอการติดต่อ
- การกำหนด SQL statement สำหรับ Select bean เราสามารถที่จะใส่ SQL statement ลงไปด้วยตนเองหรือว่าเราสามารถที่จะใช้ SQL Assist SmartGuide เพื่อช่วยเราในการมองและแปลงตัว SQL statement ออกมาให้เราได้

การติดต่อกับฐานข้อมูล Relational

Select bean ได้จัดเตรียมกลุ่มของ method สำหรับการติดต่อกับฐานข้อมูล Relational เอาไว้ เช่น Execute Method ก็ไว้สำหรับทำงาน SQL statement , Update Row Method ก็ใช้ในการ Update Row ในฐานข้อมูลซึ่งเป็นแถวปัจจุบัน (Current Row) ของ Result set สำหรับการติดต่อข้อมูลรีเลชันนัลที่ใช้ Select Bean นั้น เราสามารถที่จะสร้างการเชื่อมต่อกับ Select bean ได้ เช่น เราสามารถที่จะนำ event-to-method มาเชื่อมต่อระหว่าง AtionPerfromed Eent จากปุ่มกดกับ Execute Method ของ Select bean ได้ เวลาที่ปุ่มนั้นถูกกด SQL statement ที่เป็นของ Select bean นั้นจะถูกทำงานขึ้นทันที

เวลาที่เราเรียก SQL statement ทำงานโดยใช้ Select bean มันจะคืนค่า Result set กลับมา แต่ว่าจำนวนที่แท้จริงของ Row ที่ถูกเก็บเข้าไปในหน่วยความจำแคช (Cache Memory) นั้นก็ถูกควบคุมโดย Property ต่างๆของ Select bean เราสามารถที่จะเซตค่าของ Property เหล่านั้นและควบคุมจำนวนของ Row ที่จะถูกเก็บเข้าไปในหน่วยความจำแคชได้ Property ที่เราจะต้องกำหนดมีดังนี้

- จำนวนของแถวสูงสุดที่สามารถเฟ็ต (Fetch) เข้าไปในแคชได้
- ขนาดของแพ็กเกจซึ่งก็คือ Set Of Row
- จำนวนของแพ็กเกจสูงสุดที่ยอมรับเข้าไปในแคชได้

โดย Row ดังกล่าวนั้นคือ "subset of the result set" หมายถึงจำนวนของ Row ที่เราต้องการและต้องใช้เท่านั้น เช่นถ้า Result Set คือ 100 Row แต่ต้องการแสดงผลแค่ 10 Row เท่านั้น 10 Row นั้นจึงจะถูกเฟ็ตเข้าไปไว้ในแคชเราสามารถที่จะทำการแสดงข้อมูลใน Result Set โดยการเชื่อมต่อ property-to-property ระหว่าง property ต้นทางของ Select bean กับ property ปลายทางของอินเทอร์เฟซคอมโพแนนซ์ (Interface Component) เช่น Text Field เพื่อแสดงผลออกไปได้

เมธอดของ Select bean ต่างๆส่วนใหญ่จะถูกออกแบบให้กระทำโอเปอเรชันต่างๆกับแถวปัจจุบันของ Result Set เวลาที่ SQL statement execute โดยใช้ Select bean นั้น แถวแรกของ result set ก็คือแถวปัจจุบันโดย โอเปอเรชันต่างๆนั้นก็คือเมธอดที่ Select bean นั้นเตรียมไว้ให้เราสามารถที่จะทำการเพิ่ม,แก้ไข,ลบข้อมูล

2.3.4 DBNavigator bean

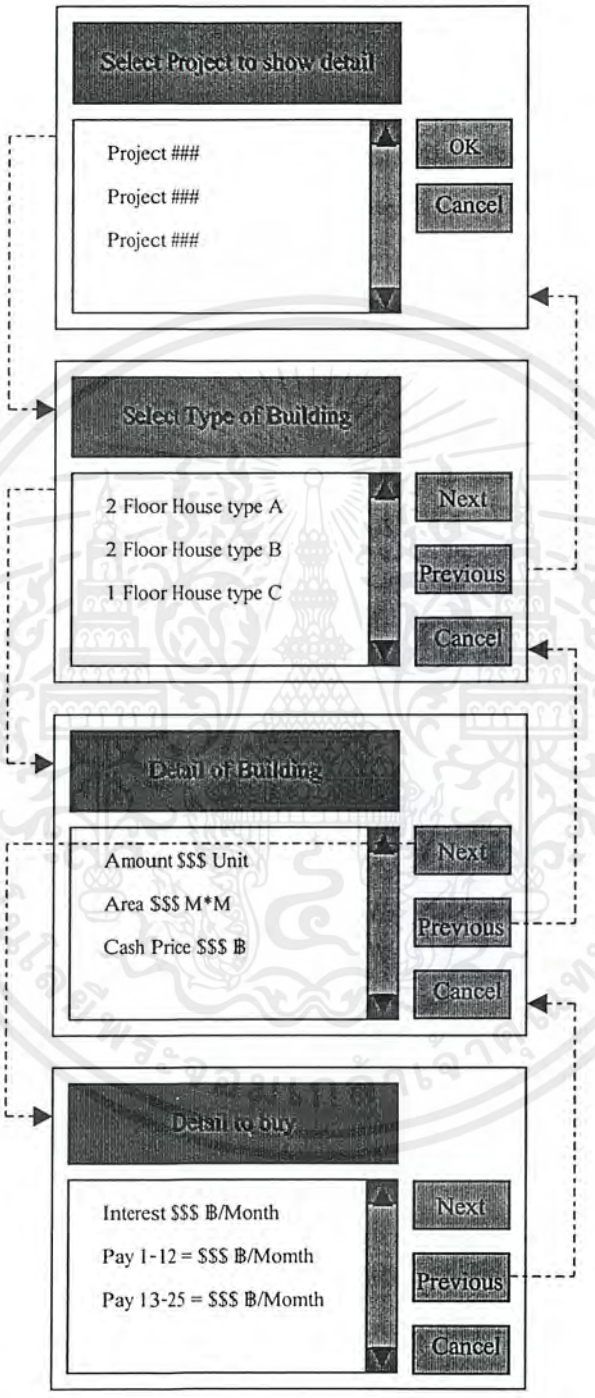
DBNavigator bean เป็น visual bean ที่ใช้ร่วมกับ Select bean โดยที่ DBNavigator bean ได้จัดเตรียมปุ่มต่างๆไว้ในการทำ SQL statement เพื่อมาทำงานต่างๆกับฐานข้อมูลรีเลชันนัลการใช้ DBNavigator bean เราสามารถสร้าง property-to-property เชื่อมต่อระหว่าง *this* property ของ Select Bean และ model property ของ DBNavigator bean *this* นั้นอ้างถึงออบเจ็กต์ ของ Select bean ส่วน model property จะระบุ DBNavigator ที่จะมาเกี่ยวเนื่องกับ Select bean

2.4 ตัวอย่างโปรแกรมประยุกต์ที่ใช้ในการออกแบบ

โปรแกรมประยุกต์นี้จะ พัฒนามาเพื่อให้ความสะดวกกับลูกค้าในการเลือกซื้อหรือจองอาคาร ที่อยู่ในโครงการทั้งหมด ทำให้ลูกค้าสามารถที่จะทราบรายละเอียดของอาคารที่สนใจเช่น มีขนาดพื้นที่เท่าไร , ราคาเท่าไร , จำนวนห้องนอน , จำนวนห้องน้ำ เป็นต้น ซึ่งช่วยให้ลูกค้าสามารถที่จะตัดสินใจ เลือกอาคารที่ต้องการ และเหมาะสมกับรายได้ มีการทำงานหลักๆดังนี้

- แอพพลิเคชั่นสามารถแสดงรายละเอียดของโครงการได้
- แอพพลิเคชั่นสามารถแสดงรายละเอียดของอาคารได้
- แอพพลิเคชั่นสามารถเก็บข้อมูลและรายละเอียดของลูกค้าได้
- ลูกค้าสามารถขอดูรายละเอียดของโครงการ, สั่งจอง, ซื้อ หรือผ่อนชำระได้

- เจ้าหน้าที่ที่จะทำการแก้ไข, เพิ่ม และลบโครงการ จะต้องมีชื่อและรหัสผ่านเพื่อความปลอดภัย



รูปที่ 2.4-1 แสดง ตัวอย่างการออกแบบ User-Interface ในส่วนของการขอดูรายละเอียดต่างๆของโครงการที่ออกแบบไว้

2.5 เทคนิคการแมป (Mapping) จาก ออบเจ็กต์ เป็นตาราง

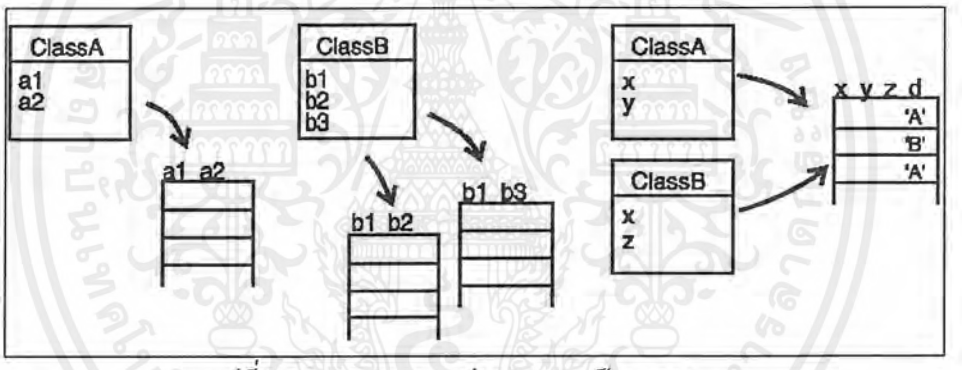
2.5.1 การแมปจากออบจ็กต์ เป็นตาราง (Mapping framework)

Mapping framework ทำให้เราสามารถแมปข้อมูลที่มีความสัมพันธ์เป็นลำดับชั้นแบบ ออบเจ็กต์ ได้ เราสามารถจะพิจารณาคลาสออบเจ็กต์ เป็นฐานข้อมูล ที่ประกอบไปด้วยฟิลด์ต่างๆ ซึ่งเราจะต้องแตกความสัมพันธ์ของแอ็บสแทรกต์คลาส (abstract class) ออกมาเป็น สับคลาส (subclass) ย่อยต่างๆ ในส่วนที่เพิ่มเติม ในการแปลงระหว่างข้อมูลกับ ออบเจ็กต์ นั้นเราสามารถจะทำได้โดยการแมปข้อมูลสกีมา (schema) element จากแอตทริบิวต์ของคลาส ความสัมพันธ์ของ ออบเจ็กต์ (Object relationships) สามารถเปลี่ยนมาเป็นความสัมพันธ์ของข้อมูล (Data relationships คือ foreign keys)

2.5.2 กลยุทธ์ในการแมปประเภทต่างๆ

2.5.2.1 แอตทริบิวต์ (Attribute mapping)

แอตทริบิวต์ต่างๆของคลาส สามารถแมปไปเป็น 1 หรือหลายคอลัมน์ได้ใน 1 ตารางหรือ หลายตารางด้วยก็ได้



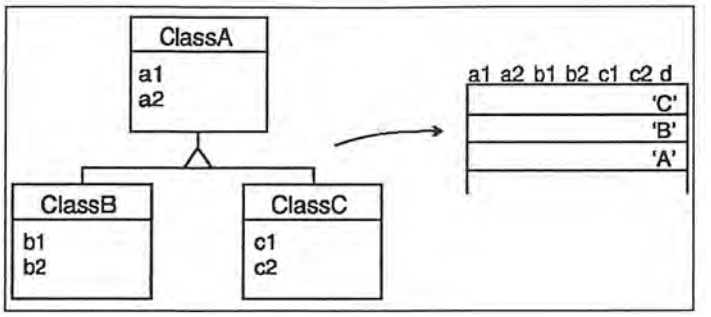
รูปที่ 2.5-1 แสดง การแมปจากคลาส เป็นตาราง

จากรูปซ้ายไปขวาเป็นการแสดงให้เห็นการแมปจาก คลาส A ที่ประกอบไปด้วยแอตทริบิวต์ a1 และ a2 เป็น 1 ตารางส่วนรูปถัดมาก็เป็นการแมปจาก คลาส B คลาส เดียวมี 3 แอตทริบิวต์ b1, b2 และ b3 แยกออกมาเป็น 2 ตาราง b1 กับ b2 และ b1 กับ b3 ส่วนรูปสุดท้ายจาก คลาส A และ คลาส B ที่มี แอตทริบิวต์เหมือนกันคือ x สามารถรวมกันเป็นตารางเดียวได้ รูปแบบของการแมป แอตทริบิวต์ นั้นเราจะต้องพิจารณาความสัมพันธ์ของ แอตทริบิวต์ ด้วยว่าควรแมปออกมาเป็นแบบใด

2.5.2.2 การแมปโดยมองที่การสืบทอดของคลาส (Inheritance mapping)

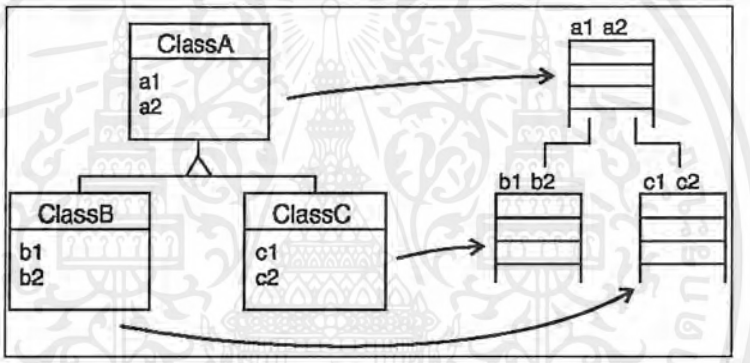
ความสัมพันธ์ของ คลาส เราสามารถแมปออกมาเป็นตารางเดียวหรือหลายตารางก็ได้แบ่งออกเป็น 2 ประเภท

- Typed partitioning เป็นความสัมพันธ์แบบลำดับชั้นของคลาส (Class hierarchy) ที่แมปออกมาเป็น ฐานข้อมูล 1 ตาราง



รูปที่ 2.5-2 แสดง การแมปจาก คลาส ทั้ง 3 ที่สืบทอดกันเป็น 1 ตาราง

- Vertical partitioning เป็นความสัมพันธ์แบบลำดับชั้นของคลาส ที่แมปออกมาเป็นตาราง Root และ ตาราง Leaf



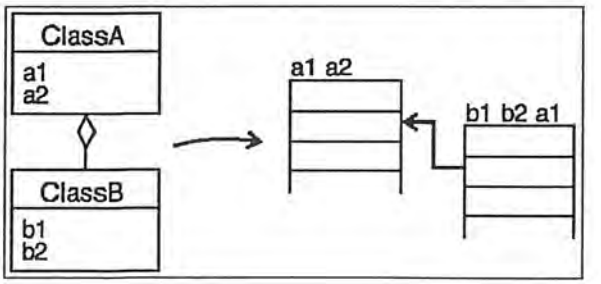
รูปที่ 2.5-3 แสดง การแมปจาก คลาส ทั้ง 3 ที่สืบทอดกันเป็น ตาราง Root และLeaf

2.5.2.3 การแมปโดยการมองที่ความสัมพันธ์ของคลาส (Relationship mapping)

ออบเจ็กต์ สามารถแมปเป็น one-to-one หรือ one-to-many relationships ได้

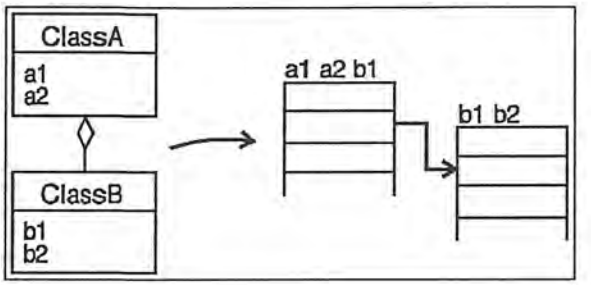
- One-to-one relationships สามารถแมปได้เป็น 2 ทางคือ

1. มองจากล่างขึ้นบน (Backward pointing)



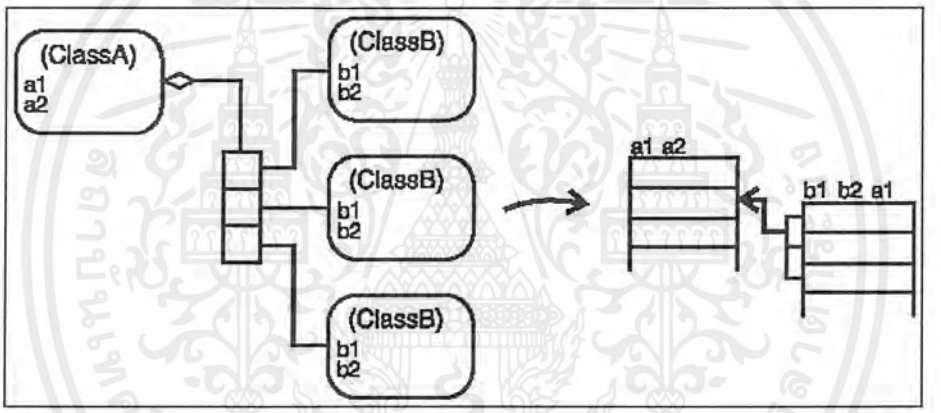
รูปที่ 2.5-4 แสดงการแมปคลาส ที่มีความสัมพันธ์กัน มาเป็นตารางโดยการมองจากล่างขึ้นบน

2. มองจากบนลงล่าง (Forward pointing)



รูปที่ 2.5-5 แสดงการแมปคลาส ที่มีความสัมพันธ์กัน มาเป็นตารางโดยการมองจากบนลงล่าง

- One-to-many relationships



รูปที่ 2.5-6 แสดง ความสัมพันธ์แบบ 1 to many ที่แมป ออกมาเป็นตาราง

2.6 การติดต่อกับฐานข้อมูลออราเคิลโดยผ่าน JDBC และ SQLJ

2.6.1 ติดต่อกับฐานข้อมูลออราเคิลโดยภาษาจาวา

ถ้าเราเป็น โปรแกรมเมอร์ ที่ใช้ภาษา จาวา และต้องการติดต่อกับ ฐานข้อมูล เราจะเลือกใช้อะไร ระหว่าง JDBC ร่วมกับ SQLJ หรือเลือกใช้อย่างใดอย่างหนึ่ง ซึ่งทั้งคู่ก็เป็น API ที่ใช้ในการติดต่อกับ ฐานข้อมูล ทำไมเราจะต้องมี API ที่ให้ใช้งานอยู่มากกว่าหนึ่งตัว ซึ่งจริงๆแล้วเราอาจจะจำเป็นใช้เฉพาะ JDBC เท่านั้นแต่ถ้ามองย้อนกลับไปที่ตัวเอกสารของเราให้ดีๆเรายังต้องการที่จะใช้ SQLJ เช่นกัน

JDBC เป็นมาตรฐานที่พัฒนาขึ้นมาโดยจาวาซอฟต์ (Javasoft) ซึ่งมันถูกออกแบบออกมาหลัง ODBC ที่พัฒนาโดย Microsoft JDBC มันกลายมาเป็นมาตรฐานสำหรับการเชื่อมต่อกับ ฐานข้อมูล ด้วย

ภาษา จาวา ไปแล้วหรือเรียกง่ายๆว่า JDBC ในทุกวันนี้ก็คือ "*de facto method*" ที่ใช้สำหรับติดต่อกับฐานข้อมูลด้วยภาษาจาวา

SQLJ เป็นมาตรฐานตัวใหม่ในการวาง SQL statement ลงไปในตัว Java โปรแกรม มันเป็นโปรดักต์ที่เกิดจากการร่วมมือกันของ Oracle, IBM, Sybase, Tandem และ Informix SQLJ มันง่ายต่อการเขียน ปรับปรุง และหาจุดบกพร่องได้ง่าย แต่มีขั้นตอนมากกว่า

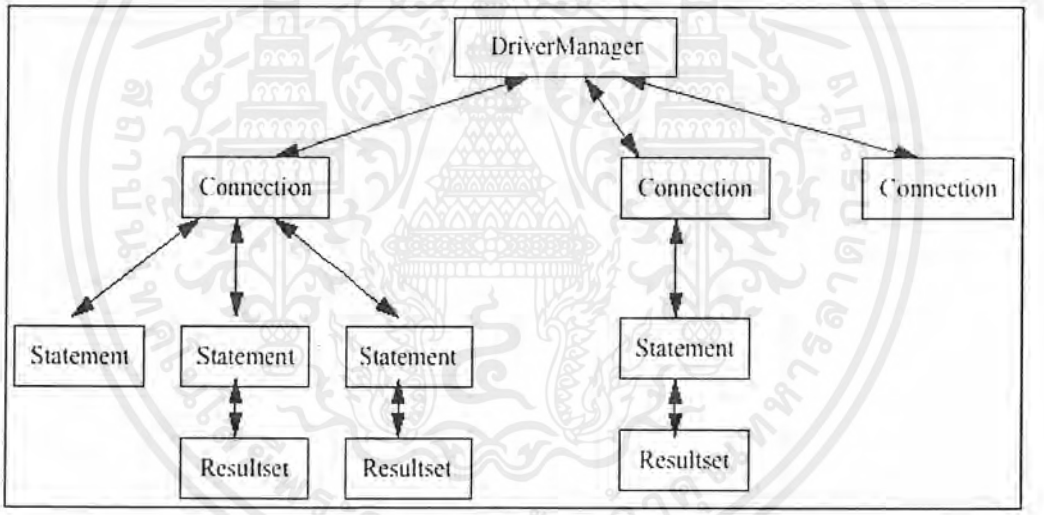
2.6.2 JDBC

2.6.2.1 ภาพรวมของการติดต่อกับฐานข้อมูลด้วย JDBC

สามารถแบ่งเป็นกลุ่มใหญ่ๆได้ 2 กลุ่มคือ กลุ่มแรกได้แก่ JDBC API สำหรับผู้ที่เขียนแอพพลิเคชั่น กลุ่มที่สองได้แก่ระดับต่ำ (Low Level) คือ JDBC Driver API

2.6.2.2 JDBC API

JDBC API เป็นจาวาอินเทอร์เฟซที่มีเพิ่มเข้ามาให้โปรแกรมเมอร์ที่เขียนแอพพลิเคชั่นได้เปิดการเชื่อมต่อกับฐานข้อมูลโดยเฉพาะ การทำ execute SQL statement และ จัดการกับผลลัพธ์ที่ได้



รูปที่ 2.6-1 แสดงเส้นทางของการติดต่อกับฐานข้อมูล

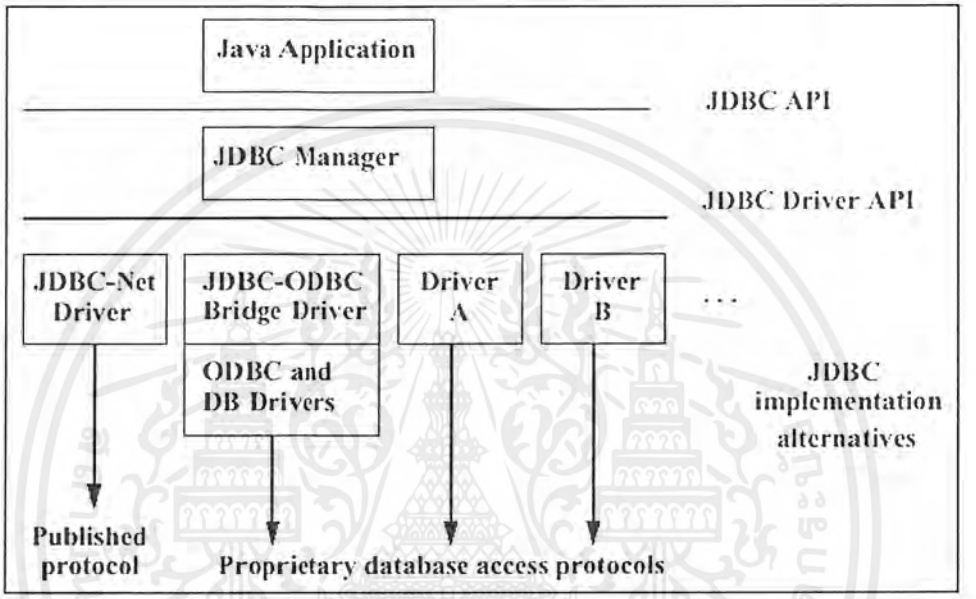
ส่วนที่สำคัญของการติดต่อกับฐานข้อมูลมี

- java.sql.DriverManager ซึ่งมันเป็นเครื่องมือในการโหลดไดรเวอร์ (Driver) และ จัดเตรียมการสนับสนุนสำหรับ การสร้างการเชื่อมต่อกับฐานข้อมูลขึ้นมาใหม่
- java.sql.Connection เป็นส่วนที่ทำการเชื่อมต่อเข้ากับฐานข้อมูลรีเลชันนัล
- java.sql.Statement เป็นส่วนที่เก็บ SQL statement ที่จะส่งผ่านการเชื่อมต่อออกไปยังฐานข้อมูล
- java.sql.ResultSet เป็นส่วนที่ควบคุมการติดต่อกับกับ Row ผลลัพธ์ที่ได้มาจาก SQL statement ที่ส่งไป

2.6.2.3 JDBC Driver API

เป็นส่วนหลักที่จะทำการจัดเตรียมการใช้งานของแอปแทรกต์คลาสที่เตรียมไว้โดย JDBC API โดยไดรเวอร์แต่ละตัวก็จะทำการจัดเตรียมการใช้ของ java.sql.Connection, java.sql.Statement, java.sql.PreparedStatement, java.sql.CallableStatement และ java.sql.ResultSet ในแต่ละไดรเวอร์จำเป็นจะต้องจัดเตรียม คลาส ซึ่งสนับสนุนการใช้ java.sql.Driver ติดต่อโดยใช้ generic java.sql.DriverManager คลาส ซึ่งมันต้องการตำแหน่งของไดรเวอร์เป็น URL ของฐานข้อมูล

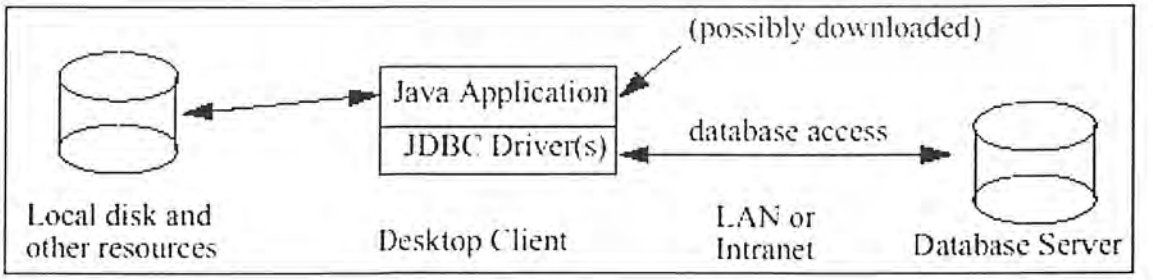
จาวาซอฟต์จะเตรียมการใช้ JDBC อยู่บน ODBC แสดงดังรูปซึ่ง JDBC มันเป็นรูปแบบที่มาที่หลัง ODBC ในส่วนการใช้งานมันจะเล็กและมีประสิทธิภาพ



รูปที่ 2.6-2 แสดง ลำดับชั้นการทำงานของ JDBC

2.6.2.4 จาวา แอพพลิเคชั่น (JAVA Applications)

จาวาสามารถที่จะใช้ในการสร้างแอพพลิเคชั่นได้ซึ่งรันได้แบบ *Shrink-Wrapped* หรือแอพพลิเคชั่นทั่วๆไปบนเครื่องไคลเอ็นต์ จากรูปเป็น Scenario ของจาวาแอพพลิเคชั่นที่ทำการติดต่อกับฐานข้อมูล



รูปที่ 2.6-3 แสดง การติดต่อระหว่างฝั่งไคลเอ็นต์กับฐานข้อมูล ฝั่งเซิร์ฟเวอร์ ผ่าน JDBC ด้วย จาวา

ก่อนที่เราจะเริ่มพัฒนาตัว JDBC เราก็ควรจะมามองและทำความเข้าใจว่าส่วนของ JDBC นั้นมีขั้นตอนและการทำงานพื้นฐานอย่างไรบ้าง โดยเราจะมาแสดงขั้นตอนพื้นฐานในการติดต่อกับฐานข้อมูลรีเลชันนัลดังนี้

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

เป็นการบอกให้ JDBC รู้ว่าเราใช้ Oracle Driver ในการติดต่อ

Connection conn = DriverManager.getConnection

("jdbc:oracle:oci8:@server:1521:orcl","ofen","july1997");

เป็นการสร้างการเชื่อมต่อตัวแปรชื่อ conn เข้ากับฐานข้อมูล *"jdbc:oracle:oci8"* เป็นส่วนของ *connect string* ที่ใช้ไดรเวอร์ OCI-base JDBC ที่ใช้สร้างการเชื่อมต่อสื่อสารกับฐานข้อมูล *"@server:1521:orcl"* เป็นส่วนของสตริงที่ระบุการเชื่อมต่อเข้ากับฐานข้อมูล ORCL บนเครื่องชื่อ "server" ผ่านพอร์ต 1521 และสุดท้าย "ofen" กับ "july1997" ก็คือชื่อและรหัสผ่านของผู้ใช้นั้นเอง หลังจากที่เราได้ทำการเชื่อมต่อเป็นที่เรียบร้อยแล้ว เราก็จะทำการเตรียม SQL query ที่ใช้ติดต่อสู่ฐานข้อมูล โดยมีตัวอย่าง query ดังนี้

ResultSet rs = stmt.executeQuery("select ename, empno, sal from emp");

Query สตริงนี้ก็คือการขอดูข้อมูล columns ename, empno และ sal จากตาราง emp ผลลัพธ์ที่ได้จะถูกเก็บอยู่ใน ResultSet object โดยสามารถจะแสดงข้อมูลภายใน ResultSet ออกมาได้โดย

```
While(rs.next())
{
    string name = rs.getString(1);
    int number = rs.getInt(2);
    double salary = rs.getDouble(3);
    System.out.println(name+" "+number+" "+salary);
}
```

โดยใน while ลูปจะแสดง result ออกมาโดยจะมีตัวชี้ (Point) ที่จะทำการชี้ไปยังแต่ละแถว (row) ใน ResultSet คำสั่ง rs.next() จะทำการคืนค่าจริงหรือเท็จกลับมาว่ามีแถวถัดไปอยู่อีกหรือไม่และจะทำการเลื่อนตัวชี้ไปยังแถวถัดไปถ้ามี เมื่อถึงแถวสุดท้ายก็จะส่งเท็จออกมาเพื่อออกจากลูป

2.6.2.5 JDBC ยังไม่สมบูรณ์แบบ

ก่อนที่เราจะรู้ที่มา และ เข้าใจใน JDBC ก็อยากให้มองย้อนกลับไปในส่วนของการเกิดข้อผิดพลาดขึ้นใน SQL statement ขณะอยู่ในช่วงรันไทม์เพราะ จาวาคอมไพล์เลอร์จะคอมไพล์โค้ด JDBC ในจาวาโดยไม่สนใจ SQL syntax หรือ semantics ,อย่างที่สองก็คือ การเขียน SQL statement ใน JDBC มันยังยุ่งยากอยู่ ดังตัวอย่าง

```
String name = 'SALMAN";
Int id=37;
```

```
Float salary=2500;
PreparedStatement stmt = conn.prepareStatement
        ("insert into emp (ename, empno, sal) values (?, ?, ?)");
stmt.setString(1,name);
stmt.setInt(2,id);
stmt.setInt(2,id);
stmt.setFloat(3,salary);
stmt.execute();
stmt.close();
```

จึงมีคำถามเกิดขึ้นว่า

1) จะทำอย่างไรให้การทำ SQL query นั้นเป็นไปได้ง่ายทั้งการอ่านและเขียน
2) ควรที่จะสามารถดีบักโค้ด SQL ได้ก่อนที่จะรันไทม์คำตอบคือการใช้ SQLJ

2.6.3 SQLJ

ซึ่งถ้าเรามาพูดถึง SQLJ แล้วมันดูง่ายและดีกว่ามากเมื่อเทียบกับ JDBC โดยดูเทียบได้จากตัวอย่างที่ผ่านมาเมื่อนำมาเขียนด้วย SQLJ

```
String name="SALMAN";
Int id=37;
Float salary=20000;
#sql {insert into emp (ename, empno, sal) values (:name,:id,:salary)};
```

ซึ่งถ้าดูโดยรวมแล้วมันง่ายกว่า JDBC มากโดยภายใน SQL statement นั้นสามารถที่จะฝังตัวแปรในภาษาจาวา ลงไปได้โดยใช้เครื่องหมาย ":" นำหน้าตัวแปร โดยจากตัวอย่างนั้นจะเห็นได้ว่าตัวแปรภาษาจาวา ซึ่งได้แก่ name, id และ salary สามารถจะฝังไว้ใน SQL statement ได้ SQLJ นั้นจะดีที่การตรวจสอบ SQL statement ก่อนที่จะรันไทม์เพราะโค้ด SQLJ มันไม่ใช่จาวา syntax ที่แท้จริงมันจำเป็นที่จะต้องรันได้โดยต้องผ่านการทรานสเลเตอร์ (Translator) ก่อนที่จะคอมไพล์เป็นจาวาไบต์โค้ดขั้นตอนในการทรานสเลเตอร์ มันจะวิเคราะห์ SQL statement ทั้ง semantically และ syntactically มันจึงสามารถตรวจสอบข้อผิดพลาดของ SQL ได้มากกว่าตอนที่รันไทม์ในกรณีของ JDBC

ในจุดนี้ทำเราคิดว่า ในเมื่อมี SQLJ ที่สามารถจะแก้ปัญหาของ JDBC ได้ แล้วทำไมเราขังจำเป็นจะต้องรู้เกี่ยวกับ JDBC อีกหรือไม่ โดยส่วนใหญ่แล้วทำไมจะต้องทำให้เรารู้เกี่ยวกับทั้งคู่ เหตุผลแรกคือทั้งคู่เป็นองค์ประกอบที่จำเป็นของ APIs ในการติดต่อกับฐานข้อมูลรีเลชันนัลทั้ง SQLJ และ JDBC สามารถที่จะใช้ร่วมกันได้ โดย JDBC นั้นจะต้องใช้สำหรับไดนามิก (*dynamic)* SQL คือ SQL statement ที่เกิดขึ้นมาในขณะรันไทม์ ในขณะที่ SQLJ นั้นดีมากสำหรับการใช้งานสแตติก (*static)* SQL คือเป็นการสร้าง SQL query ที่เรารู้ในช่วงเวลาการพัฒนาตัวแอพพลิเคชั่น เหตุผลที่สองคือ SQLJ จะต้องทำงานอยู่

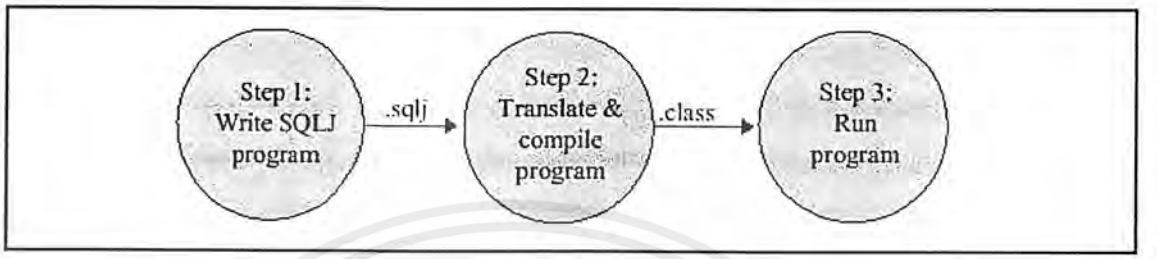
บน JDBC คือ SQLJ จะต้องแปรเปลี่ยนไปตาม JDBC คือมันหมายถึงว่าในการพัฒนา SQLJ นั้นจะต้องรู้เกี่ยวกับ JDBC driver ชนิดต่างๆ

ขั้นตอนในการสร้าง SQLJ แอพพลิเคชั่น

1 เขียนโปรแกรมด้วน SQLJ

2 ทำการทรานสเลเตอร์ตัวโปรแกรม SQLJ

3 รันตัวโปรแกรม SQLJ



รูปที่ 2.6-4 แสดง ขั้นตอนการสร้าง SQLJ แอพพลิเคชั่น

2.6.3.1 ขั้นตอนการเขียน SQLJ program

Oracle.connect("jdbc:oracle:thin:@oow11:5521:sol2","scott","tiger");

เป็นการกำหนดมาตรฐานการในเชื่อมต่อทั่วไป

#sql { insert into emp (ename, empno, sal) values {'SAlMAN',32,20000)};

เป็นการเพิ่มข้อมูล 1แถวเข้าไปในตารางซึ่งง่ายกว่า JDBC มาก

MyIter iter;

เป็นการกำหนดอินสแทนซ์ของ MyIter

#sql iter={select ename, empno, sal from emp};

มันเป็น SQL statement เกี่ยวกับ select query แล้วจะเกิดคำถามขึ้นว่าแล้ว result set ที่เก็บข้อมูลเข้ามามันอยู่ที่ไหน แล้วข้อมูลที่ได้จาก query มันจะถูกส่งไปตรงไหน จริงๆแล้ว SQLJ ไม่ต้องใช้ result set แต่เราจะใช้ *iterators* มันมีการใช้งานที่คล้ายๆกับ result set ส่วนความแตกต่างระหว่างทั้ง 2 ก็คือ iterator นั้นสามารถที่จะเก็บเอาท์พุตของ specific type of query ไว้ได้ในขณะที่ result set นั้นสามารถจะเก็บเอาท์พุตได้เฉพาะบาง query เท่านั้นเพราะ iterator type นั้นจำเป็นที่จะต้องกำหนดชนิดต่างๆของ query เอาไว้ก่อนที่จะทำ query นั้นดังตัวอย่าง

#sql iterator MyIter (String ename, int empno, float sal);

เราจะมีการกำหนดชนิดของมันเอาไว้จึงทำให้ iterator สามารถเก็บผลลัพธ์เอาไว้ได้ในตัวเองโดยคอลัมน์ 1, คอลัมน์ 2 และ คอลัมน์ 3 สามารถแมปออกมาเป็นชนิดของจาวา ได้เป็นจาวา String, จาวา int และ จาวา float ซึ่งจากประโยชน์ตรงนี้ทำให้การติดต่อถึงข้อมูลภายในตัวมันทำได้ง่ายและสั้นกว่า JDBC มากดังตัวอย่าง

```
while (iter.next()){
        system.out.println(iter.ename()+" "+iter.empno()+" "+iter.sal());
}
```

2.6.3.2 ขั้นตอนการแปลง (Translating) และคอมไพล์ลิ้ง (Compiling) ตัวโปรแกรม SQLJ

>sqlj MyExample.sqlj

เป็นการแปลง .sqlj เป็น .java file ซึ่งในขณะที่ทำการแปลงนั้น SQLJ ทรานสเลเตอร์ จะทำการตรวจสอบ SQL ที่สร้างขึ้นโดยมันจะวิเคราะห์จาก Syntactic และ Semantic มันสามารถที่จะตรวจสอบไปได้ทั้งคู่โดยมองจาก ตาราง, คอลัมน์และชนิดข้อมูลของจาวา ว่ามันสอดคล้องกันหรือไม่ หลังจากที่ได้ .java file มาแล้วต่อมาเราก็จะใช้ JDK javac คอมไพล์เลอร์มารันมันโดยจะทำการเจนเนอเรต (Generates) จาวาไบต์โค้ด หลังจากนั้นโปรแกรม SQLJ ก็จะสามารถรันได้ดังตัวอย่าง

>java MyExample

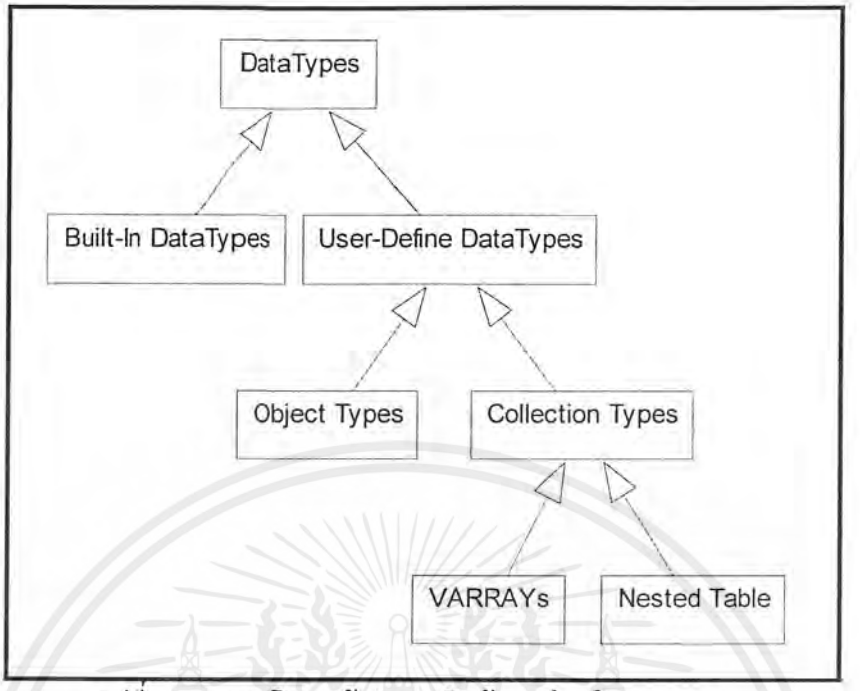
แต่ว่าในขณะนี้ในการใช้ SQLJ ยังไม่เป็นที่แพร่หลายมากนักในการติดต่อสื่อสารกับฐานข้อมูล แม้มันจะง่าย แต่ก็มีขั้นตอนที่มากขึ้นเช่นกัน

2.7 ORACLE 8

ดาต้าเบสเซิร์ฟเวอร์ (Database server) เป็นหัวใจสำคัญในการจัดการปัญหาที่เกี่ยวกับ การจัดการข้อมูลต่างๆ ซึ่งตัวเซิร์ฟเวอร์จะต้องมีความสามารถในการจัดการข้อมูลที่มีจำนวนมากแม้ในสภาวะที่มีผู้ใช้ ที่กำลังใช้ข้อมูลชุดเดียวกัน ก็ต้องจัดการให้ผู้ใช้แต่ละคนทำงานกับข้อมูลนั้นได้เหมือนไม่เห็นว่า ผู้ใช้คนอื่นได้ใช้ข้อมูลชุดนั้นอยู่ด้วย (Concurrency Control) และยังจะต้องมีความสามารถในการป้องกันการบุกรุกเข้ามาใช้ข้อมูลจากผู้ใช้ ที่ไม่มีสิทธิในระบบนั้นๆ รวมถึงการทำการกู้ข้อมูล (Recovery) กับคืนมาในกรณีที่ เกิดการเสีย (Failure) ของงานเกิดขึ้นได้

Oracle 8 เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์(object-relational database management system: ORDBMS) ซึ่งเป็นระบบฐานข้อมูลที่พัฒนาขึ้นมาจากระบบฐานข้อมูลแบบเดิมคือระบบฐานข้อมูลเชิงสัมพันธ์ (relational database management system: RDBMS) ซึ่งมีความสามารถที่เพิ่มขึ้นในการจัดเก็บข้อมูลที่มีความซับซ้อนมาก ๆ เช่น ข้อมูลมัลติมีเดีย, รูปภาพ, เสียง สามารถที่จะจัดการกับข้อมูลเหล่านี้ได้อย่างมีประสิทธิภาพ โดยมีการเพิ่มชนิดข้อมูล (DataType) ใหม่ขึ้นมาเพื่อรองรับการจัดเก็บข้อมูล และมีชนิดของข้อมูลที่ผู้ใช้สามารถที่จะกำหนดขึ้นมาเอง (User-Defined DataType) ข้อมูลชนิดนี้จะมีคุณสมบัติของออบเจ็กต์-โอเรียนเต็ด เหมือนกับ ออบเจ็กต์ที่เขียนขึ้นจากภาษาที่เป็นแบบโปรแกรมเชิงวัตถุแบบต่างๆ แต่ยังไม่เทียบเท่าภาษาเหล่านั้น

2.7.1 ชนิดของข้อมูลใน Oracle 8



รูปที่ 2.7-1 แสดงโครงสร้างของชนิดข้อมูลต่างๆใน oracle 8

2.7.1.1 Built-in DataTypes เป็นชนิดของข้อมูลที่มีให้อยู่ในตัวของออราเคิลประกอบด้วย

- Character Datatypes ประกอบด้วย

1. **CHAR** Datatype เป็นชนิดข้อมูลตัวอักษรที่มีขนาดความยาวคงที่ตามที่กำหนดไว้ในตอนต้น มีการกำหนดขนาดความยาวตั้งแต่ 1 ถึง 2,000 ไบต์ซึ่งถ้าหากมีการเพิ่มเติมหรือแก้ไขข้อมูลเกิดขึ้นแล้ว ขนาดของข้อมูลมีขนาดน้อยกว่าที่กำหนดไว้ ข้อมูลก็จะถูกเติมช่องว่างให้ได้ตามขนาดที่กำหนดไว้ ถ้าเกิดข้อมูลมีขนาดมากกว่าที่กำหนดก็จะตัดออกให้ได้ตามขนาดที่กำหนดไว้หรือถ้าเกิดข้อมูลนั้นยาวมากๆ ก็จะเกิดข้อผิดพลาด (Error) ขึ้น
2. **VARCHAR2** DataType เป็นชนิดข้อมูลตัวอักษรที่มีขนาดความยาวเปลี่ยนแปลงได้ มีการกำหนดขนาดความยาวได้ตั้งแต่ 1 ถึง 4,000 ไบต์ โดยในแต่ละคอลัมน์จะต้องมีการกำหนดขนาดความยาวเริ่มต้นไว้ ซึ่งถ้าข้อมูลที่เพิ่มเติมหรือแก้ไขมีขนาดน้อยกว่าที่กำหนดไว้จะเก็บเท่ากับจำนวนของข้อมูลจริง ไม่มีการใส่เพิ่มเติมเหมือน CHAR
3. **VARCHAR** DataType เป็นชนิดข้อมูลที่มีลักษณะเหมือนกับชนิดข้อมูลแบบ VARCHAR2 แต่ในเวอร์ชันใหม่ของออราเคิลจะสามารถที่จะเก็บขนาดของข้อมูลที่เปลี่ยนแปลงได้และเปรียบเทียบความแตกต่างของสตริงได้ด้วย
4. **NCHAR** และ **NVARCHAR2** เป็นชนิดข้อมูลแบบ NLS(National Language Support) ซึ่งจะเป็นรูปแบบที่ตัวอักษรที่เก็บนั้นจะเป็นลักษณะที่ว่า 1 ไบต์จะเก็บมากกว่า 1 ตัว

อักษร เช่น อักษรภาษาไทย ซึ่ง NCHAR จะเก็บข้อมูลที่มีความยาวคงที่ที่สอดคล้องกับข้อมูลที่มีความยาวคงที่ หรือข้อมูลที่มีความยาวเปลี่ยนแปลงได้ ส่วน NVARCHAR2 จะเก็บข้อมูลที่มีการเปลี่ยนแปลงได้ ซึ่งเมื่อกำหนดชนิดของข้อมูลเป็น NCHAR และ NVARCHAR2 สามารถที่จะมีขนาดไบต์สูงสุดของ NCHAR ที่ 2,000 ไบต์ และ NVARCHAR2 ที่ 4,000 ไบต์

5. **LONG** DataType เป็นชนิดของข้อมูลที่สามารถเก็บข้อมูลได้สูงสุดถึง 2 กิกะไบต์ เหมาะสำหรับการเก็บเท็กซ์ไฟล์

- **NUMBER** Datatype เป็นการเก็บข้อมูลตัวเลขที่เป็นจำนวนเต็ม (fixed numbers) และเลขจุดทศนิยม (floating-point numbers) ซึ่งสามารถเก็บได้สูงสุดถึง 32 หลัก
- **DATE** DataType ชนิดของข้อมูลที่เป็นวัน จะเก็บข้อมูลที่เป็นวันและเวลา จะกำหนดเป็น วัน-เดือน-ปี ส่วนเวลากำหนดเป็น ชั่วโมง-นาที-วินาที โดยแต่ละฟิลด์จะกำหนดไว้ 7 ไบต์
- **LOB** DataType เป็นการเก็บข้อมูลที่มีขนาดใหญ่ เช่น เท็กซ์ไฟล์ ภาพกราฟฟิก เสียง มีขนาดสูงสุดได้ถึง 4 กิกะไบต์ จะประกอบด้วย

1. **BLOB** DataType ชนิดของข้อมูลประเภทนี้จะเก็บข้อมูลที่เป็น ไบนารี่ ที่ไม่มีโครงสร้างในฐานข้อมูล
2. **CLOB** and **NCLOB** DataType ชนิดของข้อมูลที่เก็บเป็นตัวอักษร โดย CLOB จะเก็บข้อมูลประเภท Single-byte ส่วน NCLOB จะเก็บข้อมูลประเภท fixed-length
3. **BFILE** DataType ชนิดของข้อมูลประเภทนี้จะเก็บข้อมูลประเภทไบนารี่ที่เป็นแฟ้มข้อมูลนอกฐานข้อมูล โดยในคอลัมภ์ BFILE จะเก็บค่า file locator ที่ชี้ตำแหน่งของแฟ้มข้อมูล ชนิดของข้อมูลประเภทนี้สามารถเก็บได้สูงสุดถึง 4 กิกะไบต์ ซึ่งชนิดของข้อมูลประเภทนี้สามารถอ่านได้อย่างเดียวไม่สามารถเปลี่ยนแปลงแก้ไขข้อมูลได้

- **RAW** and **LONG RAW** DataType เป็นชนิดข้อมูลที่จะไม่มีการเปลี่ยนแปลงแม้ขณะที่มีการย้ายข้อมูลระหว่างระบบต่างๆ ซึ่งชนิดข้อมูลแบบนี้เหมาะกับข้อมูลที่เป็นแบบไบนารี่ และ ที่เป็นไบต์สตริง โดย RAW จะสามารถมีขนาดๆได้สูงสุดถึง 2,000ไบต์ ส่วน LONG RAW นั้นมีขนาดได้ถึง 2 กิกะไบต์ ซึ่งคอลัมน์ที่กำหนดเป็น LONG LAW ไม่สามารถที่จะนำไปสร้าง index ได้ส่วน RAW นั้นสามารถนำไปทำเป็น index ได้
- **ROWID** DataType เป็นชนิดข้อมูลที่บอกถึงตำแหน่งของแถวทุกๆ แถวในตารางซึ่งจะเก็บข้อมูลเป็นไบนารี่ โดย extened ROWID มีขนาด 10 ไบต์ และประเภท restricted ROWID จะมีขนาด 6 ไบต์

2.7.1.2 User-Define DataTypes เป็นคุณสมบัติที่เพิ่มขึ้นมาใน ORDBMS คือผู้ใช้สามารถที่จะกำหนดชนิดของข้อมูลขึ้นมาเอง ตลอดจนรายละเอียดของโครงสร้างข้อมูลและการทำงานของตัวมันเอง และสามารถที่จะใช้ชนิดของข้อมูลที่กำหนดขึ้นเหล่านี้ในรีเลชันนัลโมเดลได้ ซึ่งเพิ่มประสิทธิภาพในการที่จะพัฒนาโปรแกรมในรูปแบบการโปรแกรมเชิงออบเจ็กต์-โอเรียนเต็ด ซึ่งจะประกอบด้วย

- **Object Type** คือชนิดของข้อมูลที่มีความซับซ้อนและใกล้เคียงกับความเป็นจริงมากขึ้นซึ่งจะมีการนำเอาข้อมูล (data) และ การกระทำ (operation) มาไว้ด้วยกัน ผู้พัฒนาสามารถที่จะทำงานได้โดยไม่ต้องรู้รายละเอียดภายในของข้อมูลชนิดออบเจ็กต์นี้เลย และยังสามารถนำข้อมูลชนิดออบเจ็กต์นี้กลับมาใช้ใหม่ได้

การสร้าง Object Type

1. ตัวอย่างการสร้าง Object Type ที่ใช้เก็บข้อมูลที่อยู่

```
CREATE  TYPE  address_type AS OBJECT
(       home_address     VARCHAR2(20),
        tumbon           VARCHAR2(30),
        aumpor           VARCHAR2(30),
        city             VARCHAR2(30),
        zipcode          NUMBER(5) );
```

แสดงการสร้าง Object Type ซึ่งประกอบด้วย บ้านเลขที่ ตำบล อำเภอ จังหวัด และรหัสไปรษณีย์

2. ตัวอย่างการสร้างตารางความสัมพันธ์ที่นำเอา Object Type ที่สร้างจากตัวอย่างด้านบนมาใช้

```
CREATE TABLE  informationStudent
(       id#              NUMBER(8) PRIMARY KEY,
        first_name       VARCHAR2(30),
        last_name        VARCHAR2(30),
        major            VARCHAR2(30),
        address          ADDRESS_TYPE );
```

3. ตัวอย่างการใส่ข้อมูลในตาราง InformationStudent ที่มีชนิดข้อมูลเป็น Object Type ที่สร้างขึ้นมาเอง

```
INSERT INTO informationStudent
VALUES
```

```
(40013259,'Phairat','Phattanachaiyanan','Computer Engineering',ADDRESS_TYPE
('53 M.6','Ban Chat','Ban bung','Chonburi',20170));
```

- **Collection Type** มีลักษณะที่เป็นหน่วยของข้อมูล(Data Unit) ที่มีจำนวนสมาชิกไม่จำกัด และสมาชิกทั้งหมดมีชนิดข้อมูลเหมือนกัน ซึ่งจะประกอบด้วย Nested Table และ VARRAYs DataType

1. **Varrays** มีลักษณะเป็นเซ็ตของข้อมูลแบบมีลำดับ (Array) โดยสมาชิกทุกตัวจะเป็นข้อมูลชนิดเดียวกัน และสมาชิกแต่ละตัวจะมีเครื่องชี้ (Index) ซึ่งเป็นหมายเลขที่สอดคล้องกับตำแหน่งของสมาชิกในอาร์เรย์ ซึ่งจะไม่มีการกำหนดขนาดของอาร์เรย์ ที่แน่นอนแต่เราสามารถที่จะระบุขนาดที่ใหญ่ที่สุดเมื่อเรามีการกำหนดชนิดของข้อมูลเป็นแบบนี้ และสามารถนำชนิดข้อมูลแบบนี้ไปใช้เป็น
   - ชนิดของข้อมูลของคอลัมน์ ในตารางรีเลชันนัล (relational table)
   - แอตทริบิวต์ของชนิดออบเจ็กต์
   - ตัวแปร PL/SQL , พารามิเตอร์หรือชนิดของข้อมูลที่ส่งค่ากลับมาจากฟังก์ชัน

ตัวอย่างการกำหนดชนิดของข้อมูลแบบ VARRAYs

```
CREATE TYPE  address_types AS VARRAY(10) OF address_type;

CREATE TABLE  CLASS
(       name            VARCHAR2(30),
        address         ADDRESS_TYPES);
```

ตัวอย่างการใส่ข้อมูลลงในตาราง

```
INSERT INTO CLASS
VALUES
(       'John', ADDRESS_TYPES (ADDRESS_TYPE(''53 M.6',
        'Ban Chat','Ban bung','Chonburi',20170      )));
```

2. **Nested Tables** เป็นเซ็ตของข้อมูลแบบที่ไม่เรียงลำดับ ซึ่งสมาชิกของทุกๆ ตัว จะเป็นข้อมูลชนิดเดียวกันและ Nested Table จะมีเพียงคอลัมน์เดียวและ คอลัมน์นั้นจะเป็นข้อมูลชนิด Built-in DataTypes หรือออบเจ็กต์ซึ่งเราสามารถที่จะมองเหมือนเป็นตาราง ที่ภายในแต่ละแถว ที่คอลัมน์ที่เป็น Nested Table จะสามารถมีแอตทริบิวต์ได้หลายแถวย่อยๆ อีก

ตัวอย่างการกำหนดชนิดของข้อมูลแบบ Nested Table

```
CREATE TYPE test_score AS OBJECT
(student_id      NUMBER(8),
score            NUMBER);

CREATE TYPE test_score_table  AS TABLE OF test_score;
```

```
CREATE TABLE test_results
(       instructor_id    VARCHAR2(8),
        class_id         VARCHAR2(6),
        name             VARCHAR2(30),
        scores           TEST_SCORE_TABLE)
        NESTED TABLE  scores  STORE  AS  test_scores;
```

ตัวอย่างการใส่ข้อมูลเข้าไปในตาราง

```
INSERT INTO  test_results
VALUES
(       'E101','3P','Apiwat',
        TEST_SCORE_TABLE
        (  TEST_SCORE('40013259',90),
           TEST_SCORE('40013260',70),
           TEST_SCORE('40013261',80)));
```

บทที่ 3

การสร้างและออกแบบ

3.1 ความต้องการของระบบ (Requirement)

การออกแบบระบบเราจะต้องทำการกำหนดความต้องการโดยรวมของระบบออกมาเสียก่อน หลังจากนั้นจึงค่อยนำความต้องการเหล่านั้น มาออกแบบเป็นระบบตามที่เราต้องการได้ ในบทนี้จะกล่าวเริ่มต้นตั้งแต่การกำหนดความต้องการของระบบ การออกแบบระบบ คลาสไดอะแกรม ฐานข้อมูล การสร้างหน้าจอในการติดต่อกับผู้ใช้ เป็นต้น โดย ระบบที่นำมาใช้ในการออกแบบนั้น เป็นระบบการจัดการฐานข้อมูลโครงการขายบ้านจัดสรรในสำนักงาน โดยจะต้องออกแบบและทำขึ้นมาเป็นแอพพลิเคชั่นต้นแบบ (Prototype Application) โดยได้มีความต้องการของระบบสรุปออกมาได้ดังนี้

ระบบต้องการคือระบบที่ใช้ในการจัดเก็บข้อมูลต่างๆของโครงการ เริ่มจากระบบนั้นจะต้องสามารถทำการวางแผนผังของโครงการได้ สามารถเพิ่มลดจำนวนโครงการได้ แก้ไขแผนผังของโครงการได้ ใส่ข้อมูลให้กับบ้านแต่ละหลัง ทำการสั่งซื้อ จอง หรือว่าเช่าบ้านในโครงการได้ โดยจะเก็บข้อมูลการซื้อขายไว้ว่าใครเป็นคนซื้อ วันไหน เป็นต้น แก้ไขข้อมูลลูกค้าและพนักงานได้ ดูผลสรุปของแต่ละโครงการได้ โดยสามารถแยกรายละเอียดต่างๆออกมาได้ดังนี้

ระบบฐานข้อมูลโครงการเช่าซื้อบ้านจัดสรรในสำนักงานนั้นเป็นระบบที่ใช้ในบริษัทแห่งหนึ่ง โดยเป็นบริษัทที่ทำธุรกิจเกี่ยวกับการขายและให้เช่าบ้านจัดสรรในโครงการต่างๆ ซึ่งภายในระบบนั้นจะต้องแก้ไขโครงการหรือเพิ่มเติมโครงการเข้าไปได้ ตัวโครงการประกอบไปด้วยรายละเอียดต่างๆของโครงการเช่น ชื่อโครงการ, ขนาดของโครงการ, ที่อยู่, จำนวนบ้านในโครงการ, วันเริ่มต้นโครงการ, วันสิ้นสุดโครงการ ในแต่ละโครงการก็จะมีการเก็บแผนผังของโครงการเอาไว้ด้วย ประกอบไปด้วยบ้าน โดยจะเก็บตำแหน่งของบ้านและบ้านเลขที่ แล้วก็แผนที่ของโครงการ โดยบ้านจะประกอบไปด้วย ชนิดของบ้าน, สถานะของบ้าน (ให้เช่า, ขาย, จอง, อื่นๆ) โดยบ้านแต่ละหลังก็อาจจะมีชนิดที่เหมือนกันหรือต่างกัน แต่ละโครงการก็อาจมีบ้านชนิดเดียวกันก็ได้ โดยบ้านแต่ละชนิดก็จะมีรายละเอียดที่แตกต่างกันเช่น ชื่อชนิด, ความกว้างของบ้าน, พื้นที่โดยรวม, จำนวนห้องน้ำ, ห้องนอน, จำนวนชั้น, ราคา เป็นต้น

นอกจากตัวโครงการแล้ว ก็ยังมีรายละเอียดส่วนอื่นอีก บริษัทจะมีพนักงานอยู่ด้วยกัน 3 ตำแหน่ง คือ ผู้ดูแลฐานข้อมูล (DBA), ผู้จัดการ (Manager) และ พนักงาน (Official) โดย DBA นั้นจะเป็นคนที่ทำหน้าที่ในการแก้ไขเพิ่มเติมตัวโครงการ สร้างตัวโครงการ วางแผนผังโครงการ และสิทธิเพียงผู้เดียวในการทำงานด้านนี้ด้วย นอกจากนี้ DBA ก็ยังสามารถที่จะทำการแก้ไขข้อมูลของพนักงานคนอื่นๆหรือจัดการเกี่ยวกับพนักงาน การให้สิทธิในการเข้าไปจัดการกับโครงการใดๆ , ผู้จัดการนั้นจะมีสิทธิในการตรวจสอบโครงการหรือดูข้อมูลของแต่ละโครงการ ซึ่งขึ้นอยู่กับสิทธิด้วยว่า ผู้จัดการคนนั้นมีสิทธิที่จะเข้ามาจัดการเกี่ยวกับ โครงการใดได้บ้าง , พนักงานจะทำหน้าที่ในการจัดการเกี่ยวกับข้อมูลของลูกค้า การสั่งจอง,

ซื้อหรือเช่าของลูกค้า พนักงานจะจัดการในส่วนนี้ และในการเข้าไปจัดการข้อมูลของแต่ละโครงการ ก็จะต้องมีสิทธิกำหนดด้วยว่า พนักงานคนนั้นมีสิทธิเข้าไปจัดการในโครงการใดได้บ้าง โดยในการเข้าไปจัดการโครงการนั้นจะต้องผ่านการล็อกอินและใส่รหัสก่อนจึงจะเข้าสู่ระบบได้

ในส่วนของลูกค้านั้นจะมีการเก็บข้อมูลรายละเอียดต่างๆที่สำคัญๆของลูกค้าไว้เช่นชื่อ, นามสกุล, เชื้อชาติ, สัญชาติ, สถานะการแต่งงาน, ที่อยู่, หมายเลขโทรศัพท์, หมายเลขวิทยุติดตามตัว (pager), เลขที่บัตรประจำตัวประชาชน, เงินเดือน, อาชีพและบริษัท โดยถ้ามีการแก้ไขข้อมูลของลูกค้าคนใด ใครที่เป็นคนแก้ไข ก็จะถูกบันทึกรหัสเอาไว้ด้วย

ในส่วนของการซื้อขายนั้น ตัวสัญญาที่จะบันทึกไว้นั้นจะประกอบไปด้วย วันที่เริ่มทำสัญญา, วันหมดสัญญา, ทำสัญญากับบ้านหลังไหน โครงการอะไร และ ชนิดของสัญญา (เช่า, ซื้อ, จอง) โดยลูกค้านั้นสามารถที่จะทำสัญญาได้มากกว่า 1 สัญญา และทำได้กับทุกๆโครงการ

จากรายละเอียดของระบบที่กล่าวมาทั้งหมดนี้ จะนำความต้องการที่ได้มาหาว่าอะไรบ้างคือ ยูสเคส (usecase) ของระบบ มี ออบเจ็กต์ อะไรบ้าง โดยแต่ละ ออบเจ็กต์ ประกอบไปด้วยแอตทริบิวต์อะไรบ้าง มีโอเปอเรชันอะไรบ้าง แต่ละ ออบเจ็กต์ นั้นมีความสัมพันธ์กันอย่างไร

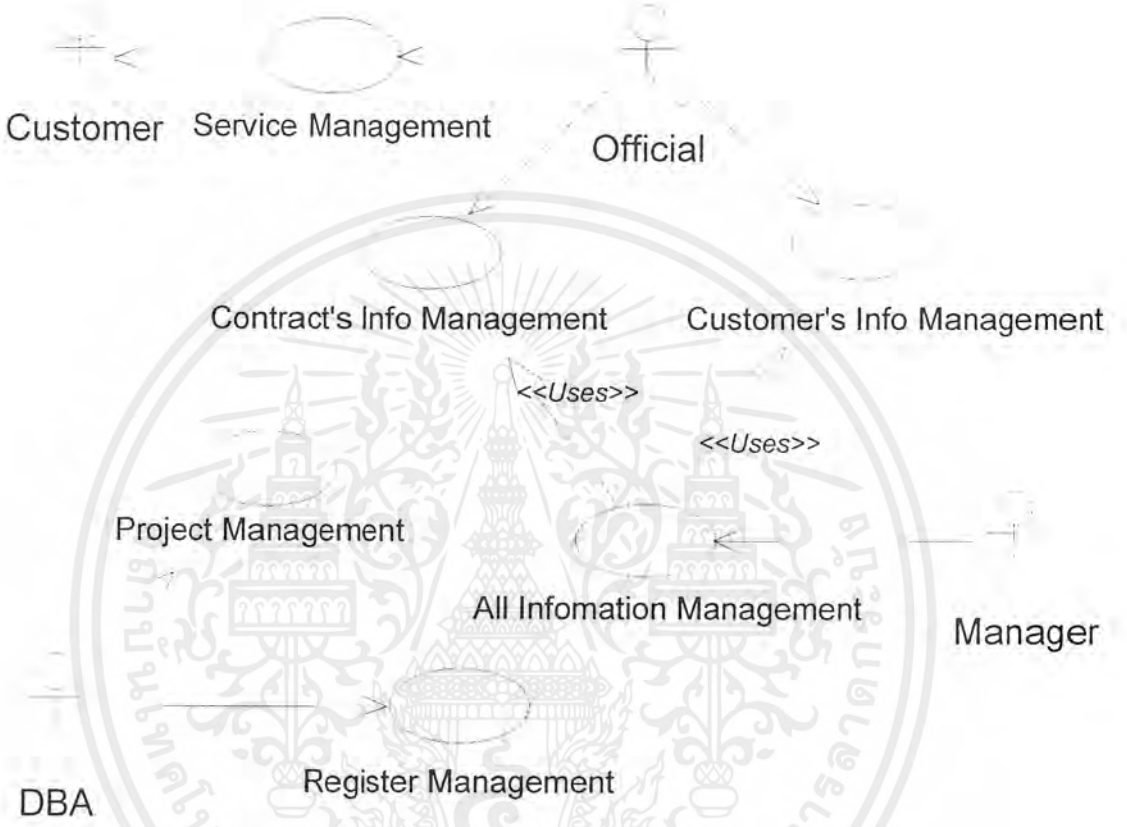
3.2 ยูสเคส (Use Case)

เราจะทำการหาความต้องการหลักๆของระบบว่ามีอะไรบ้าง โดยแสดงออกมาในรูปของยูสเคสไดอะแกรม เริ่มจากเราจะแสดงความต้องการของระบบแสดงดังรูปต่อไปนี้

รูปที่ 3.2-1 แสดงความต้องการของระบบ

จากรูปที่ 3.2-1 เป็นยูสเคสที่ใช้ในการแสดงความต้องการของระบบว่าประกอบไปด้วย ความต้องการอะไรบ้าง และใครที่เป็นคนเกี่ยวข้องกับความต้องการเหล่านั้น ดังนี้คือ Customer ต้องการที่จะซื้อ, เช่า หรือว่าจองบ้าน และต้องการดูรายละเอียดของบ้านในโครงการด้วยว่ามีรายละเอียดอะไรบ้างในการตัดสินใจ ซื้อ, เช่า หรือว่าจอง ส่วน Official จะคอยให้บริการ Customer โดยเป็นคนจัดการกับข้อมูลส่วนตัวต่างๆของ Customer เพื่อจัดเก็บไว้ในฐานข้อมูล และจัดเก็บตัวสัญญาที่ Customer จะทำกับบริษัทไว้ในฐานข้อมูล และจะอัพเดตข้อมูลลงเซิร์พเวอร์อยู่ตลอดเพื่อไม่ให้เกิดความซ้ำซ้อนเกิดขึ้นในการซื้อขาย ส่วน Manager ก็จะเป็นคนที่ควบคุมดูแลตัวโครงการอยู่ตลอด เพื่อดูความเป็นไปของโครงการต่างๆ เช่น มีบ้านขายไปแล้วกี่หลัง จองแล้วกี่หลังเป็นต้น

DBA เป็นคนที่มีสิทธิในการที่จะแก้ไขข้อมูลต่างๆของโครงการเช่น การแก้ไขแผนผังโครงการ การสร้างแผนผังให้โครงการใหม่ที่จะเกิดขึ้น รวมเป็นถึงจัดการกับข้อมูลของตัวบ้าน และมีสิทธิในการควบคุมการทำงานของ Employee ทุกๆคนเป็นต้น จากความต้องการของระบบระหว่างผู้ใช้กับระบบแล้ว เราก็สามารถแบ่งงานต่างๆออกมาได้เป็นยูสเคสที่จะแสดงดังต่อไปนี้

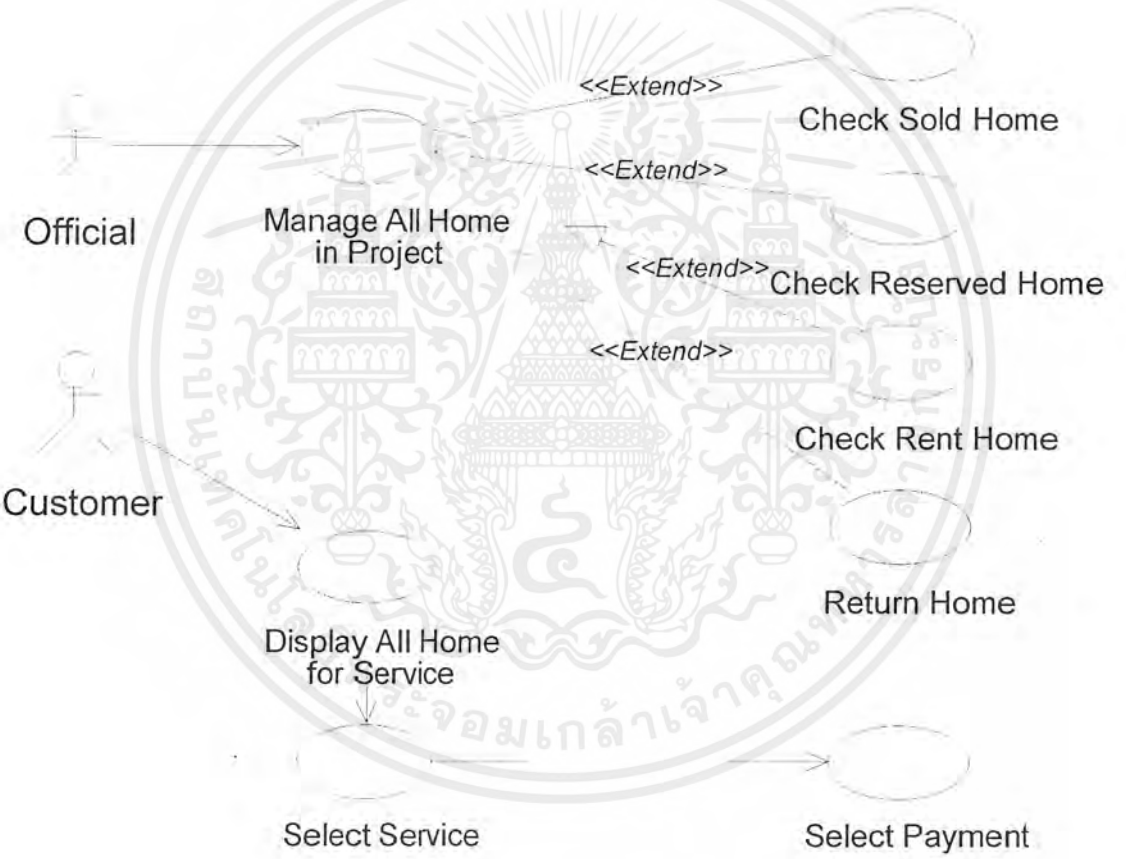


รูปที่ 3.2-2 แสดง use case หลักของระบบ

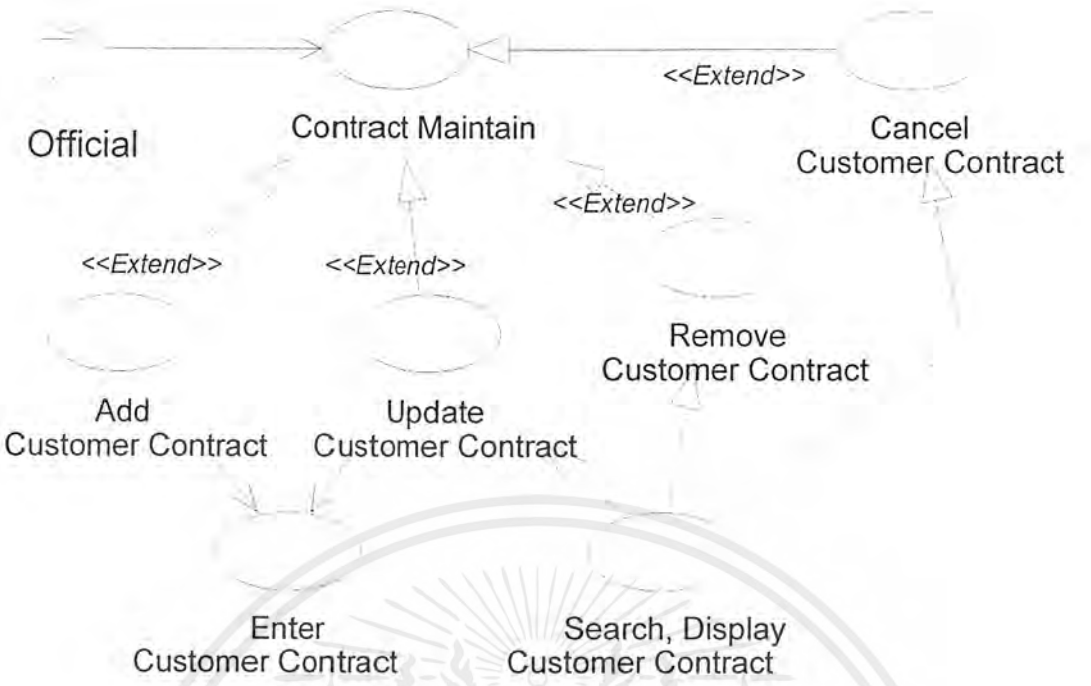
ดังที่กล่าวมาแล้วระบบประกอบไปด้วยผู้ใช้อยู่ด้วยกัน 4 ประเภทคือ ผู้ใช้ที่เป็น DBA ก็จะทำการจัดการกับตัวโครงการ และจัดการกับข้อมูลของ Employee รวมไปถึงสิทธิในการล็อกอินเข้าไปจัดการตัวโครงการของ Employee แต่ละคนด้วย Manager จะเป็นคนที่จัดการกับข้อมูลทั้งหมด โดยอาศัยข้อมูลจาก Customer's Info และ Contract's Info ส่วน Customer ก็เป็นคนที่เรียกใช้บริการจากระบบ Official ก็จะทำงานกับส่วนของการบริการจากระบบ และจัดการกับข้อมูลของ Contract และ Customerโดยเราสามารถที่จะแสดงยูสเคสหลักๆของระบบออกมาเป็น 6 ยูสเคสหลักๆดังต่อไปนี้

1	Service Management	คือ การให้บริการกับลูกค้าเมื่อต้องการ
2	Contract's Info Management	คือ การทำสัญญาระหว่างลูกค้ากับโครงการ
3	Customer's Info Management	คือ การจัดการข้อมูลส่วนตัวของลูกค้า
4	All Information Management	คือ การจัดการกับข้อมูลต่างๆของโครงการ
5	Project Management	คือ การแก้ไขหรือเพิ่มโครงการเข้าสู่ระบบ
6	Register Management	คือ การจัดการเกี่ยวกับข้อมูลของพนักงานในบริษัท

จากยูสเคสหลักๆของระบบเราสามารถนำยูสเคสเหล่านั้นมาแยกออกเป็นยูสเคสย่อยต่างๆ ได้แสดงดังรูปต่อไปนี้



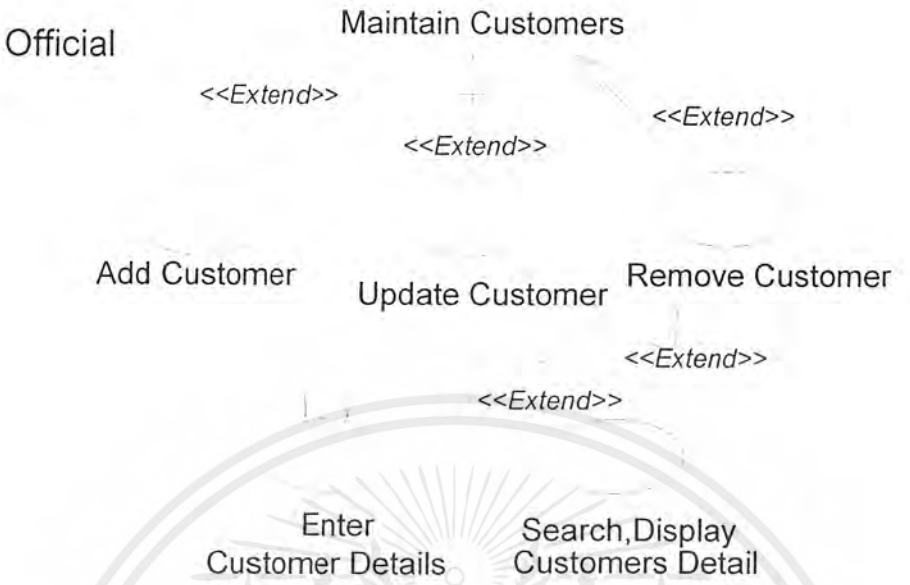
รูปที่ 3.2-3 แสดง Service Management use case



รูปที่ 3.2-4 แสดง Contract's Info Management use case

จากรูปที่ 3.2-3 แสดงยูสเคสของ Service Management คือจะแสดงถึงยูสเคสย่อยของ Service Management โดย Customer จะทำการดูข้อมูลบ้านในโครงการ และทำการเลือกบ้านที่ต้องการจะ เช่า ซื้อ หรือจอง หลังจากที่ทำการตกลงกันเรียบร้อยแล้ว Official ก็จะเป็นคนจัดการให้โดยจะทำการเลือกบ้านจากโครงการ และ สั่งจอง เช่า หรือ ซื้อจากโครงการ และถ้ากรณีของการจอง หรือ เช่า ถ้า Customer ต้องการยกเลิก Official ก็จะจัดการแก้ไขข้อมูลได้

รูปถัดมาคือรูปที่ 3.2-4 จะแสดงถึงสัญญาที่ทำกันระหว่าง Customer กับตัวโครงการสัญญาดังกล่าวจะถูกบันทึกข้อมูลเอาไว้ในเซิร์ฟเวอร์เช่นกันโดยที่ Official จะเป็นคนคอยจัดการประกอบไปด้วยการเพิ่ม, แก้ไข, ลบ, ยกเลิกตัวสัญญา โดยประกอบไปด้วยการ เพิ่มข้อมูล หรือว่า ค้นหาหาข้อมูลเป็นต้น

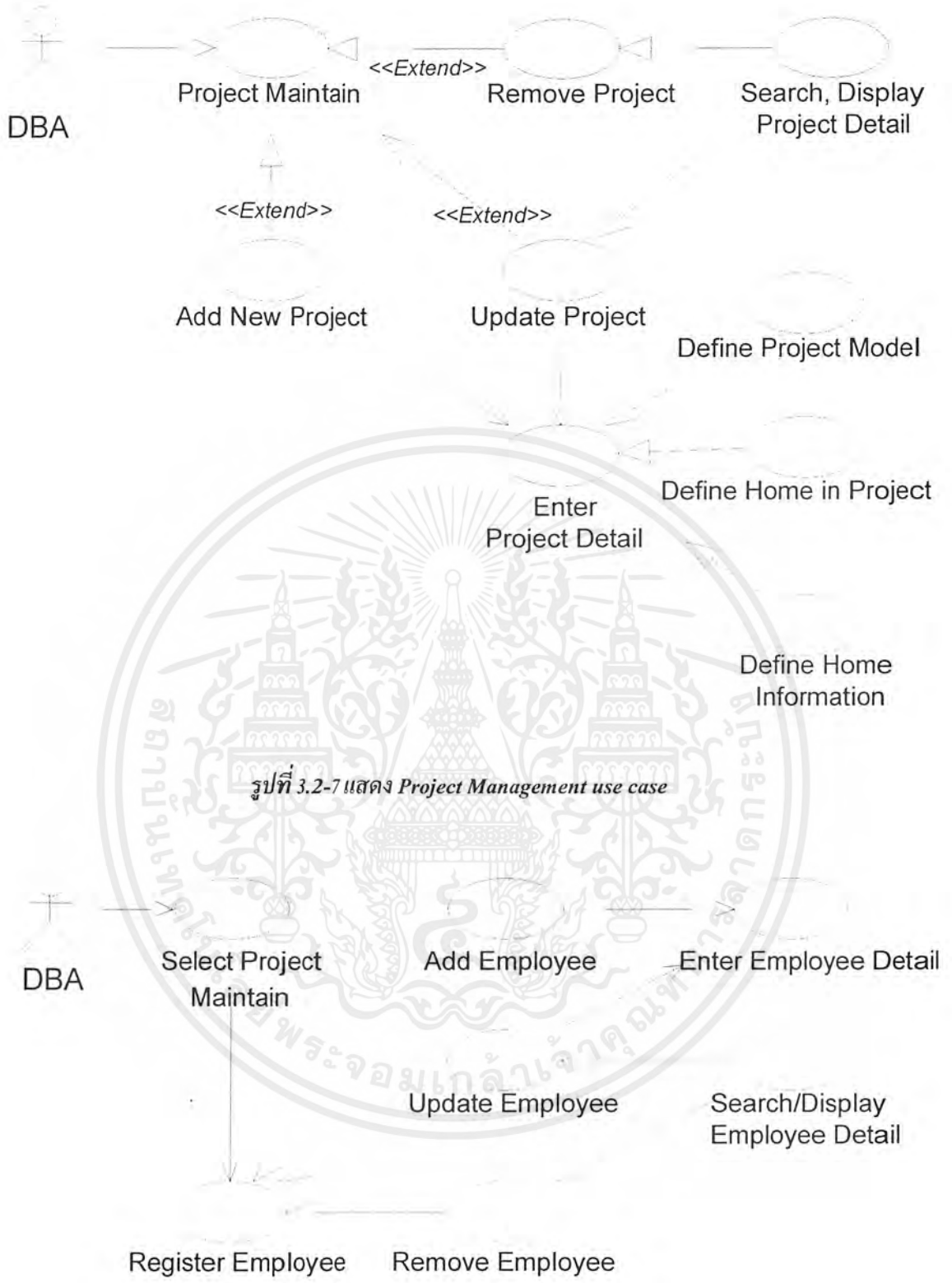


รูปที่ 3.2-5 แสดง Customer's Info Management use case

จากรูปที่ 3.2-5 แสดงการจัดการเกี่ยวกับข้อมูลของ Customer โดย Official ประกอบไปด้วยการเพิ่ม, แก้ไขและ ลบโดยมีการ เพิ่มข้อมูลส่วนตัวของ Customer ลงไป รวมไปถึงการ ค้นหาข้อมูลของ Customer ด้วย

รูปที่ 3.2-6 แสดง All Information Management use case

จ ากรูปที่ 3.2-6 แสดงถึงการจัดการกับข้อมูลของโครงการโดย Manager ซึ่งประกอบไปด้วยการค้นหาและ ดูข้อมูลต่างๆ



รูปที่ 3.2-7 แสดง Project Management use case

รูปที่ 3.2-8 แสดง Register Management use case

จากรูปที่ 3.2-7 และ 3.2-8 แสดงถึงการทำงานของ DBA ที่เป็นคนจัดการเกี่ยวกับตัวโครงการ โดยจะสามารถทำการ เพิ่ม, แก้ไขและลบโครงการได้ โดยใช้การเพิ่มรายละเอียดของโครงการที่ประกอบไปด้วยการกำหนดแผนผังของตัวโครงการ กำหนดบ้านให้โครงการ รวมไปถึงข้อมูลต่างๆของตัวบ้าน งาน อีกอย่างที่ DBA เป็นคนจัดการคือการจัดการกับสิทธิในการเข้าใช้ระบบของ Employee แต่ละ และข้อมูลต่างๆของ Employee

3.3 กำหนดออบเจ็กต์จากความต้องการของระบบ

จากความต้องการของระบบที่ได้มาจากความต้องการโดยรวมของระบบ และจากการวิเคราะห์แล้วแสดงออกมาในรูปของยูสเคสไดอะแกรม เราสามารถกำหนดออกเจ็กต์ออกมาได้เป็น 2 แพ็กเกจ (Package)

แพ็กเกจ 1 : ออบเจ็กต์ที่เกี่ยวข้องกับข้อมูลของตัวโครงการทั้งหมดในระบบ

แพ็กเกจ 2 : ออบเจ็กต์ที่เป็นส่วนที่ติดต่อกับผู้ใช้

มีรายละเอียดของออบเจ็กต์ภายในแพ็กเกจดังต่อไปนี้

1) ออบเจ็กต์ ที่เกี่ยวข้องกับข้อมูลของตัวโครงการทั้งหมดในระบบ ประกอบไปด้วยออบเจ็กต์ต่างๆดังต่อไปนี้

- Company เป็นบริษัทที่เป็นเจ้าของระบบนี้
- Project เป็นตัวโครงการของระบบ ซึ่งในระบบนั้นจะมีอยู่ได้หลายโครงการ
- Project Type เป็นประเภทของโครงการ ว่าเป็นประเภทใด เช่น เช่า/ซื้อ , จอง เป็นต้น
- Home ในแต่ละโครงการก็จะประกอบไปด้วยบ้านหลายหลัง
- Home Type ชนิดของบ้านเช่น บ้านเดี่ยว 2 ชั้นราคา 2,000,000 , ทาวน์เฮาส์ 3 ชั้นราคา 5,000,000 เป็นต้น
- Home Status สถานะของบ้านเพื่อใช้ในการตรวจสอบเช่น ขายไปแล้ว, ถูกจองอยู่ เป็นต้น
- Map Background เป็นพื้นที่สภาพแวดล้อมของโครงการ ว่าประกอบไปด้วยอะไรบ้าง อยู่ที่ไหนบ้าง
- Map Image รูปภาพที่ใช้แสดงในแผนผังของโครงการ
- Person บุคคลที่เกี่ยวข้องกับโครงการ
- Customer inherit มาจาก Person เป็นลูกค้าที่มาติดต่อกับโครงการ จะถูกเก็บข้อมูลเอาไว้
- Contract ข้อมูลสัญญาของลูกค้าที่ทำไว้กับโครงการ
- Contract Type ชนิดของสัญญาเช่น สัญญาเช่า, สัญญาจอง เป็นต้น
- Employee inherit มาจาก person เป็นพนักงานที่ใช้งานระบบทั้งหมด จะเก็บข้อมูลและสิทธิในการเข้าใช้ระบบเอาไว้
- Privilege ตัวสิทธิในการเข้าใช้ระบบของ Employee แต่ละคน
- Account ตัวรหัสผ่านของ Employee แต่ละคน

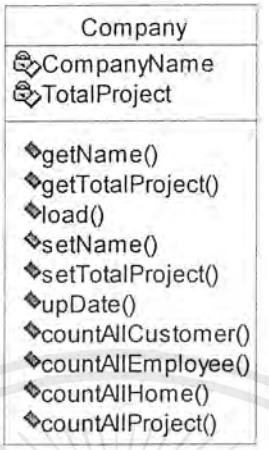
- Position ตำแหน่งของ Employee แต่ละคน เพื่อแยกแยะการเข้าใช้งานระบบแต่ละส่วน

2) ออบเจ็กต์ที่เป็นหน้าจอที่ใช้ในการติดต่อกับผู้ใช้

- Customer Service หน้าจอให้บริการลูกค้า
- Customer GUI ใช้ในการจัดการกับข้อมูลของลูกค้า
- Login GUI หน้าจอล็อกอินเข้าสู่ระบบ
- Project Editor ใช้จัดการกับตัวโครงการตั้งแต่เพิ่มโครงการ, ยกเลิกโครงการ หรือว่าแก้ไขโครงการ
- Project Management เป็นหน้าจอที่เป็นเมนูในการเข้าสู่การทำงานต่างๆที่เกี่ยวกับการบริหารโครงการ
- Service Management เป็นหน้าจอที่เป็นเมนูในการเข้าสู่การทำงานต่างๆที่เกี่ยวกับการบริการลูกค้า
- Home Type GUI แก้ไข เพิ่มเติม รายละเอียด ชนิดของบ้านที่มีอยู่ในโครงการต่างๆ
- Employee GUI ใช้ในการจัดการกับข้อมูลของพนักงาน
- Privilege GUI กำหนดสิทธิในการเข้าจัดการกับโครงการต่างๆ ว่ามีสิทธิเข้าทำโครงการใดบ้าง
- Project Summary บทสรุปของโครงการต่างๆเช่น ขายบ้านไปแล้วกี่หลัง เป็นต้น

3.4 กำหนดรายละเอียดและความสัมพันธ์ให้กับออบเจ็กต์

จาก object ต่างๆที่เราแยกออกมาได้ จาก requirement ต่อมาเราก็จะมาทำการกำหนดรายละเอียดให้กับ Class ของ object ต่างๆ คือการกำหนด Attribute และ Behavior ให้กับ Class จะแสดงดังต่อไปนี้

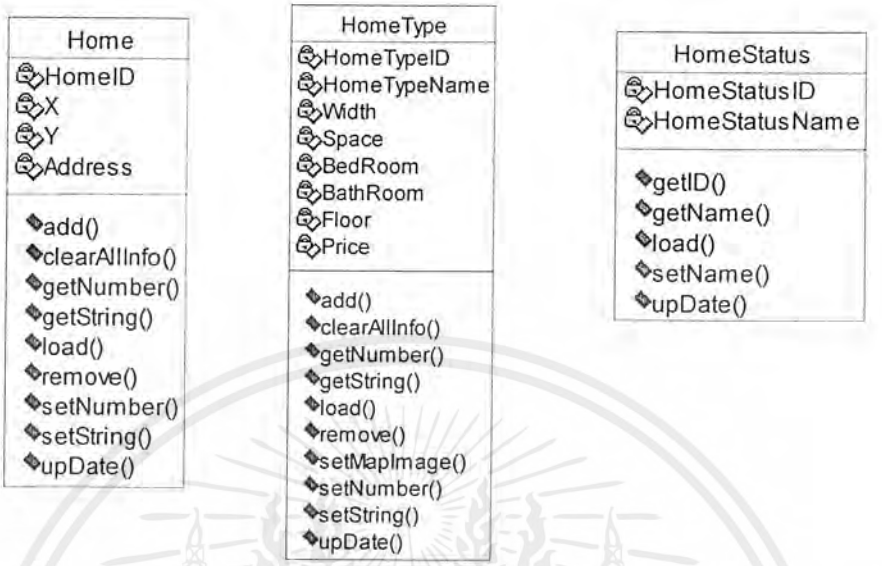


รูปที่ 3.4-1 แสดงคลาส Company

จากรูปที่ 3.4-1 แสดงคลาส Company ที่มีแอตทริบิวต์เป็น ชื่อ และ จำนวนโครงการทั้งหมด มีคุณสมบัติที่ให้เรียกใช้งานต่างๆ เช่น นับจำนวน Customer ทั้งหมด, นับจำนวนพนักงานทั้งหมด เป็นต้น

รูปที่ 3.4-2 แสดงคลาส Project และ Project Type

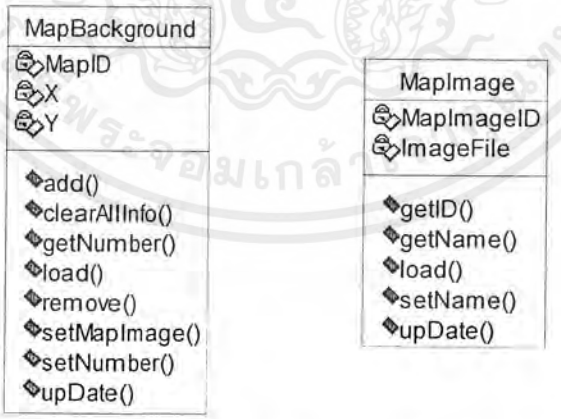
จากรูปที่ 3.4-2 เป็นการแสดงแอตทริบิวต์และคุณสมบัติของคลาส project และ project type โดยแอตทริบิวต์ที่สำคัญๆของ project เช่น ขนาดของโครงการ, ความกว้างยาวของโครงการ, ที่อยู่ของโครงการ และ วันเริ่มโครงการ เป็นต้น ส่วนคุณสมบัติของคลาสทั้งสองก็คือ คุณสมบัติที่เกี่ยวกับการ



รูปที่ 3.4-3 แสดงคลาสที่เกี่ยวกับ Home

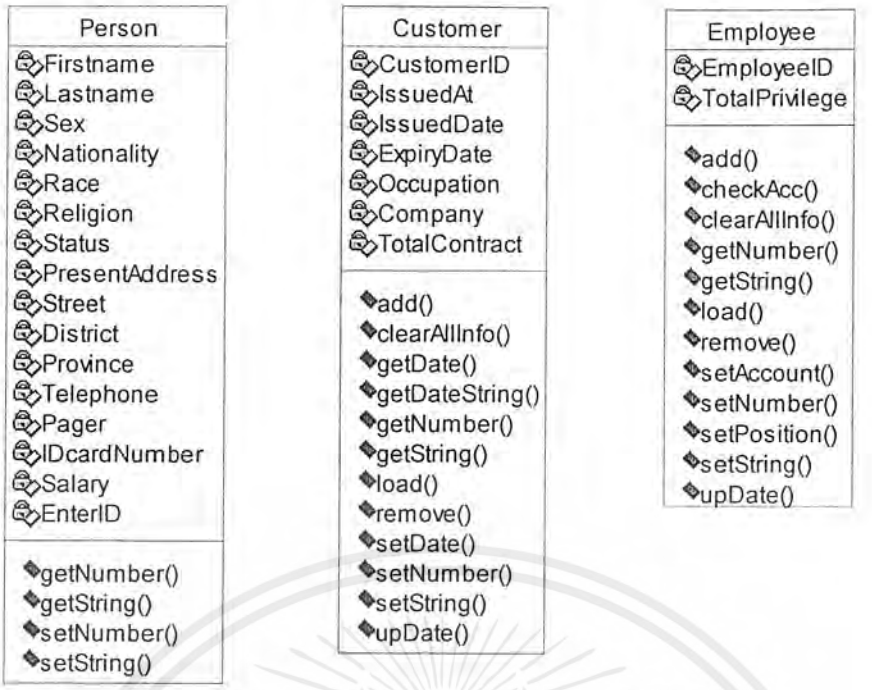
เพิ่ม ลบ และแก้ไขข้อมูลของโครงการ เป็นต้น

จากรูปที่ 3.4-3 ได้แสดงรายละเอียดของคลาสที่มีความเกี่ยวข้องกับ Home โดยคลาส Home จะประกอบไปด้วยแอตทริบิวต์ที่ แสดงตำแหน่งของบ้านและเลขที่บ้าน ชนิดของบ้านก็ประกอบไปด้วยแอตทริบิวต์หลักๆเช่น ขนาด ความกว้าง จำนวนห้องนอน ห้องน้ำ ราคา เป็นต้น



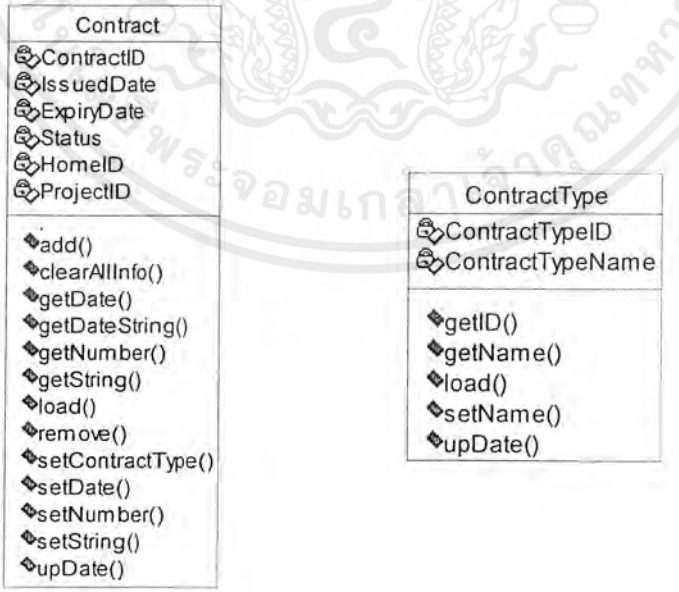
รูปที่ 3.4-4 แสดงคลาส MapBackground กับ MapImage

จากรูปที่ 3.4-4 เป็นคลาสที่ใช้ในการกำหนดแผนผัง สภาพแวดล้อมโดยรวมของโครงการประกอบไปด้วยตำแหน่งของ สิ่งต่างๆที่อยู่โดยรอบโครงการ และรูปภาพที่ไว้แสดงในแผนผัง โดยในคลาส HomeType ก็มีการอ้างถึงคลาส MapImage เช่นกัน เพื่อแสดงรูปบ้านที่เป็นโมเดลในแผนผังได้



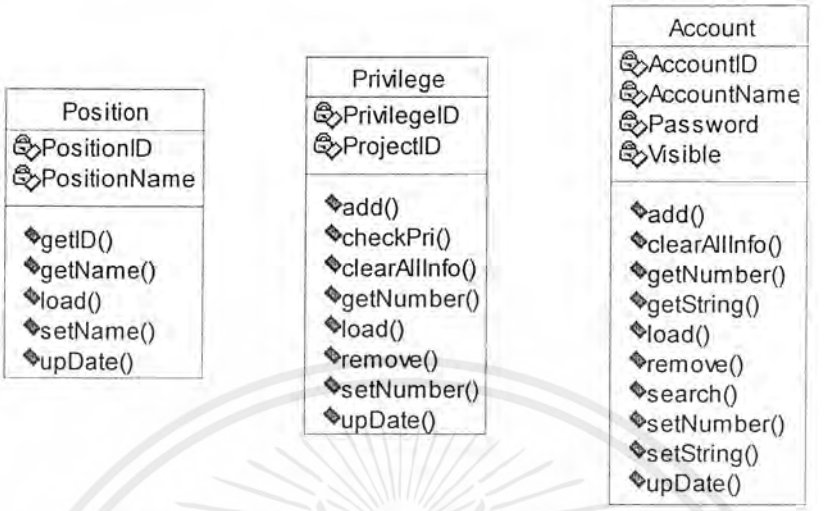
รูปที่ 3.4-5 แสดงคลาสที่แสดงถึงบุคคลที่เกี่ยวข้องกับระบบ

จากรูปที่ 3.4-5 แสดงคลาสที่แสดงถึงบุคคลที่เกี่ยวข้องกับระบบโดย Person คือคลาสที่แสดงถึงบุคคลในระบบ จะประกอบไปด้วยแอตทริบิวต์ที่ระบบต้องการเช่น ชื่อ, นามสกุล, เพศ, ที่อยู่ และ เลขบัตรประชาชน เป็นต้น ในส่วนของ Customer ก็คือลูกค้าที่มาติดต่อกับโครงการ โดย Customer จะสืบทอดมาจาก Person และจะเพิ่มเติมแอตทริบิวต์ที่ต้องการเพิ่มเติมเข้าไปอีกเช่น อาชีพ, ที่ทำงาน และจำนวนสัญญาที่ทำไว้กับโครงการเป็นต้น และในส่วนของ Employee ก็จะถูกสืบทอดมาจาก Person เช่นกัน และ มีแอตทริบิวต์ที่เพิ่มเติมเข้าไปคือ employeeid กับ totalprivilege



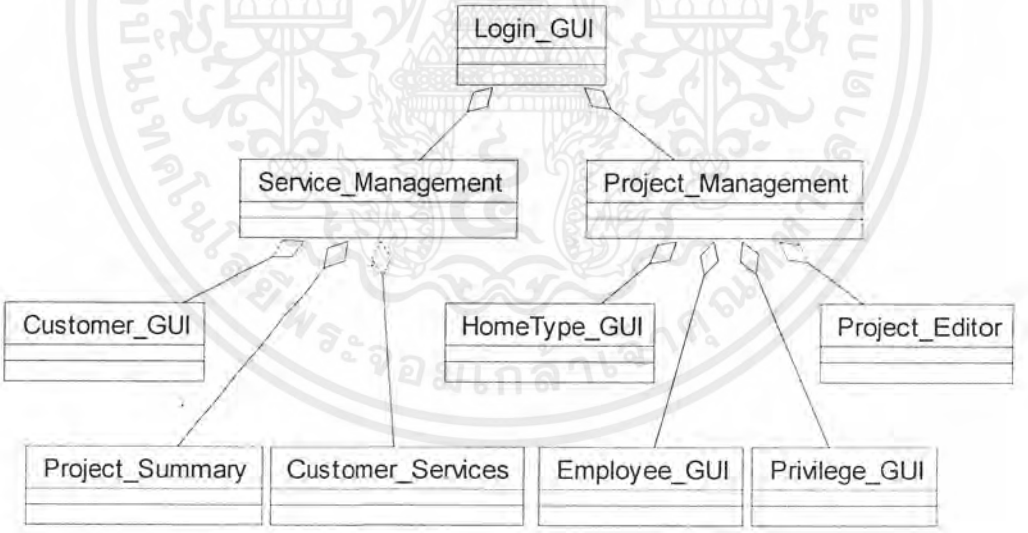
รูปที่ 3.4-6 แสดงคลาสที่เกี่ยวกับสัญญาของลูกค้า

จากรูปที่ 3.4-6 แสดงคลาสที่เกี่ยวข้องกับสัญญาของลูกค้าที่ทำไว้กับโครงการประกอบไปด้วย Contract และ Contract Type แอตทริบิวต์ของ Contract จะประกอบไปด้วย วันเริ่มสัญญา, เลขที่บ้านที่ทำสัญญาด้วยว่าอยู่ในโครงการใด เป็นต้น



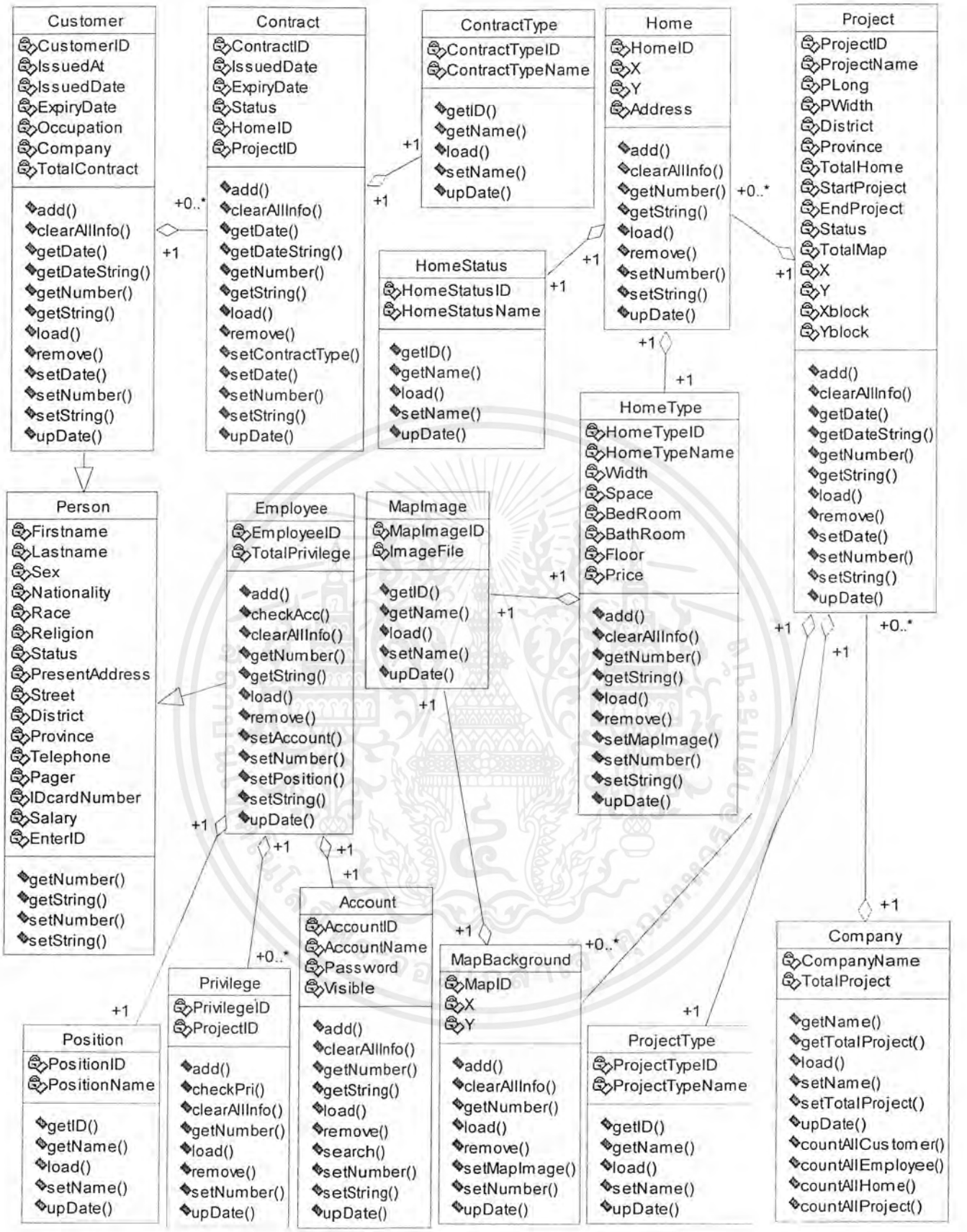
รูปที่ 3.4-7 แสดงคลาสที่เป็นองค์ประกอบของ Employee

จากรูปที่ 3.4-7 คลาสทั้ง 3 เป็นองค์ประกอบของ Employee ประกอบไปด้วยแอตทริบิวต์ต่างๆ แสดงดังรูป เช่น Account ประกอบไปด้วย ชื่อ, รหัสผ่าน เป็นต้น



รูปที่ 3.4-8 แสดงคลาสไดอะแกรมหน้าจอของระบบที่ใช้ติดต่อกับผู้ใช้

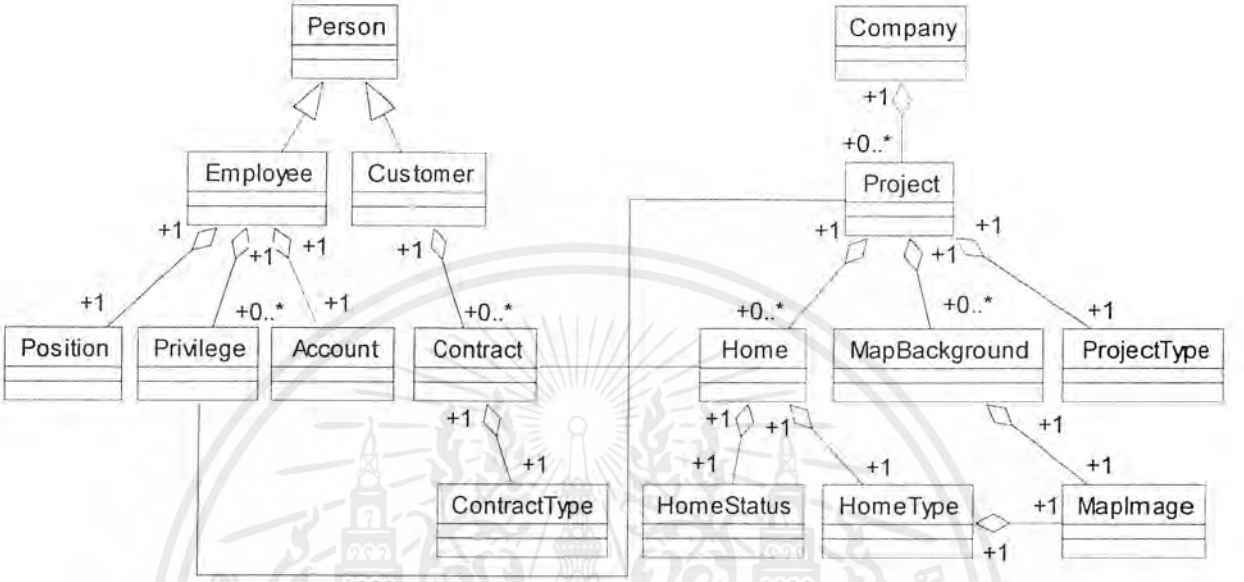
จากรูปที่ 3.4-8 (หมายเหตุ ไม่ได้แสดงรายละเอียด ภายในคลาส เพราะ เป็นคลาสที่เกิดจากการสร้างหน้าจออินเตอร์เฟซด้วยตัว VisualAge for Java) แสดงให้เห็นถึงคลาสในแพ็กเกจของ Graphic User Interface ของระบบ ที่ประกอบไปด้วยคลาสทั้งหมด 10 คลาสโดยการเรียกใช้งาน คลาสต่างๆจะมีลำดับชั้นแสดงดังไดอะแกรมโดยคลาส Login_GUI จะเป็นคลาสในแพ็กเกจ GraphicUser Interface ที่ใช้ในการเริ่มต้นรันแอพพลิเคชั่น (โดยรายละเอียดการทำงานของโปรแกรมจะอธิบายในภายหลัง) โดยเมื่อผ่านการล็อกอินแล้วก็จะทำการเรียก Service_management หรือ Project_Management ภายใน Service_Management ก็จะสามารถเรียก Customer_GUI, Project_Summary หรือ Customer_Service ส่วน Project_Management ก็จะสามารถเรียก HomeType_GUI, Project_Editor, Employee_GUI หรือ Privilege_GUI ได้



รูปที่ 3.4-9 แสดงคลาสไดอะแกรมทั้งหมดของระบบ

จาก รายละเอียดของคลาสที่เกี่ยวข้องกับข้อมูลของตัวโครงการทั้งหมดในระบบ เช่น customer, employee, home เป็นต้น เมื่อเรากำหนดรายละเอียดเรียบร้อยแล้ว เราก็จะมากำหนดความสัมพันธ์ให้กับคลาสต่างๆ แสดงโดยคลาสไดอะแกรมในรูปที่ 3.4-9

โดยจากรูปที่ 3.4-9 เป็นคลาสไดอะแกรมรวมของระบบ เพื่อให้มองได้ง่ายขึ้น จะแสดงเฉพาะความสัมพันธ์กันระหว่างคลาส ไม่แสดงรายละเอียดภายในคลาส ดังในรูปที่ 3.4-10

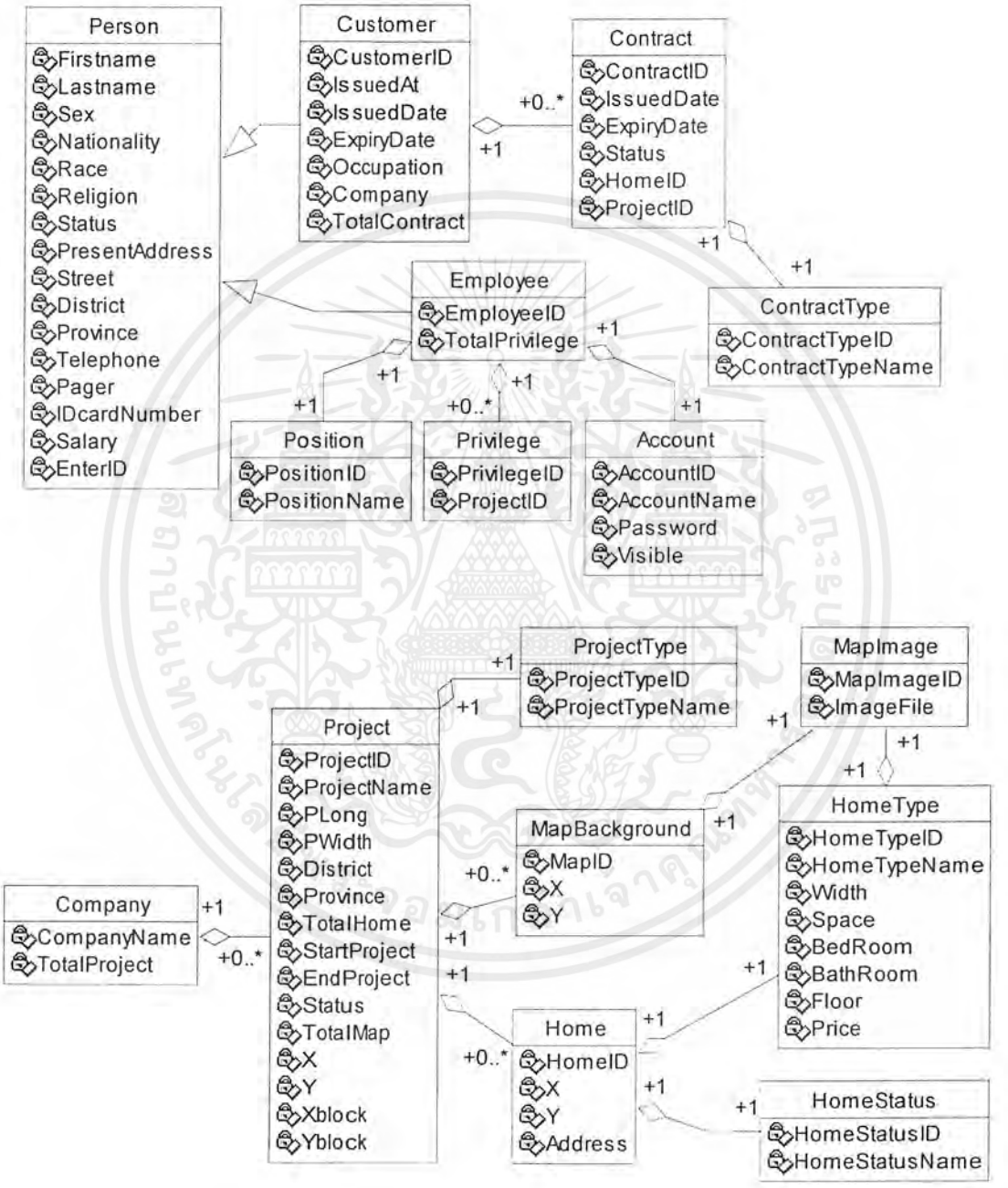


รูปที่ 3.4-10 แสดงความสัมพันธ์กันระหว่างคลาสโดยไม่แสดงแอตทริบิวต์กับคุณสมบัติ

จากรูปที่ 3.4-10 จะทำให้เราสามารถมองเห็นความสัมพันธ์ของ แต่ละคลาสได้ง่ายกว่าในรูปที่ 3.4-9โดยจะเห็นได้ว่า ระบบจะประกอบไปด้วยส่วนหลักๆ (ไม่รวมถึงส่วนยูสเซอร์อินเตอร์เฟซ) อยู่ด้วยกัน 2 ส่วนคือส่วนที่เป็นคน (Person) กับส่วนที่เป็นตัวโครงการ (Project) ความสัมพันธ์แบบแอสโซซิเอชัน ที่เกิดขึ้นระหว่าง 2 ส่วนนี้ก็คือ Contract กับ Home อีกส่วนก็คือ Privilege กับ Project โดยภายในส่วนอื่นๆก็มีความสัมพันธ์กันระหว่างคลาสดังแสดงให้เห็นในคลาสไดอะแกรม รูปที่ 3.4-10

3.5 การแมป (Mapping) ออบเจ็คเป็นฐานข้อมูลรีเลชันนัล

จากการออกแบบออบเจ็กต์ที่ผ่านมาเราก็จะได้คลาสไดอะแกรมออกมา ซึ่งคลาสไดอะแกรมที่เกี่ยวกับข้อมูลของระบบนั้น เราจะนำมาแมปเป็นฐานข้อมูลรีเลชันนัล โดยใช้หลักการของ การแบบโดยอาศัยความสัมพันธ์กันของคลาส (Relationship Mapping) ซึ่งสามารถแมปได้ทั้งแบบ one-to-one หรือ one-to-many ก็ได้ จากความสัมพันธ์ของคลาสทั้งหมด 16 คลาสที่แสดงดังรูปที่ 3.5-1



รูปที่ 3.5-1 แสดงความสัมพันธ์ระหว่างคลาสและแอตทริบิวต์ของแต่ละคลาส

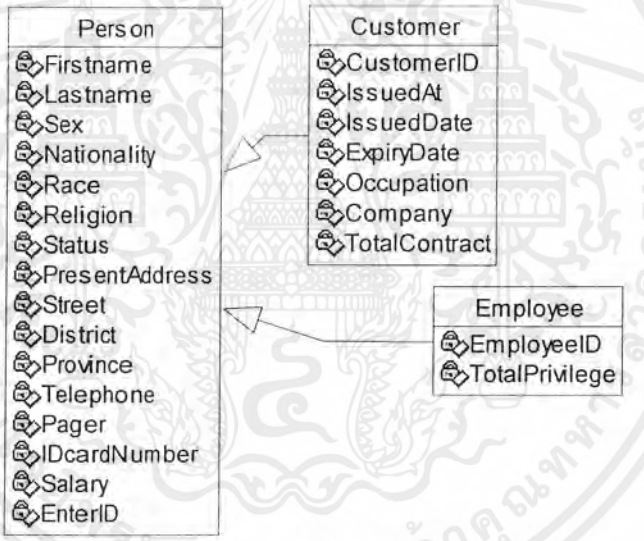
จากรูปที่ 3.5-1 เราสามารถนำความสัมพันธ์เหล่านี้มาทำการแมปเป็นฐานข้อมูลรีเลชันนัลได้โดยอาศัยหลักการดังต่อไปนี้

หลักการในการแมป ออบเจ็กต์เป็นฐานข้อมูลรีเลชันนัล จากคลาสไดอะแกรมในรูปที่ 3.5-1

1) จากคลาสไดอะแกรม ทั้งหมด 16 คลาสเราสามารถ Map ออกมาได้เป็น 15 ตารางโดยคลาสที่เราไม่ได้นำมา Map เป็นตารางด้วยก็คือคลาส Person เพราะคลาสดังกล่าวเป็น แอปสแตคคลาส (Abstract Class) เราจะนำแอตทริบิวต์ในแอปสแตคคลาสไปรวมกับแอตทริบิวต์จากคลาสที่สืบทอดจากตัวมันไปคือ คลาส Customer และ Employee

2) กำหนดตารางขึ้นมาได้เป็น 15 ตารางโดยกำหนดไพรมารี่คีย์ (Primary key) ของแต่ละตาราง โดยเลือกเอาแอตทริบิวต์ที่ไม่มีค่าเป็นนัล (Null Value) และ ค่าที่ซ้ำกัน

3) หลังจากที่กำหนดไพรมารี่ให้กับแต่ละตาราง แล้วเราก็จะมาพิจารณาที่ความสัมพันธ์ของแต่ละคลาส โดยคลาสที่มีความสัมพันธ์แบบอะกริเกชัน (Aggregation) กันแบบ 1 to 1 ก็จะนำไพรมารี่คีย์ของคลาสลูกไปใส่ไว้ในคลาสแม่ด้วย ส่วนคลาสที่มีความสัมพันธ์กันแบบ 1 to many ก็จะนำไพรมารี่คีย์ของคลาสแม่ไปใส่ไว้ในคลาสลูก โดยจะแสดงรายละเอียดการแมปดังต่อไปนี้



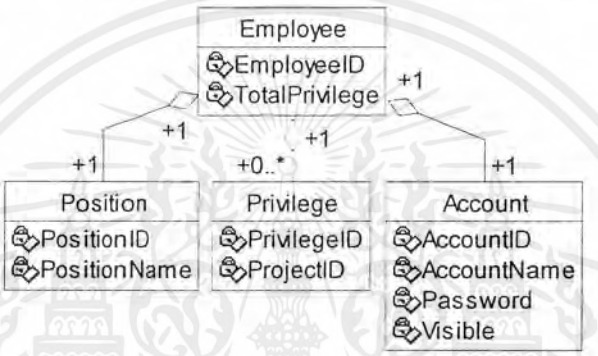
รูปที่ 3.5-2 แสดงความสัมพันธ์กันระหว่างคลาสของแอปสแกตคลาส Person ที่สืบทอดโดย Customer และ Employee

จากรูปที่ 3.52 ความสัมพันธ์ดังกล่าวตามหลักการในข้อที่ 1 เราจะแมปออกมาได้เป็น 2 ตาราง โดยตัดไป 1 ตารางคือ Person เราจะนำแอตทริบิวต์ใน Person ทั้งหมดมาเพิ่มลงไปใน Customer และ Employee จะออกมาได้เป็น

- ตาราง Customer ประกอบไปด้วย CustomerID, IssuedAt, IssuedDate, ExpiryDate, Occupation, TotalContract, Firstname, Lastname, Sex, Nationality, Race, Religion, Status, PresentAddress, Street, District, Province, Telephone, Pager, IDcardNumber, Salary, EnterID

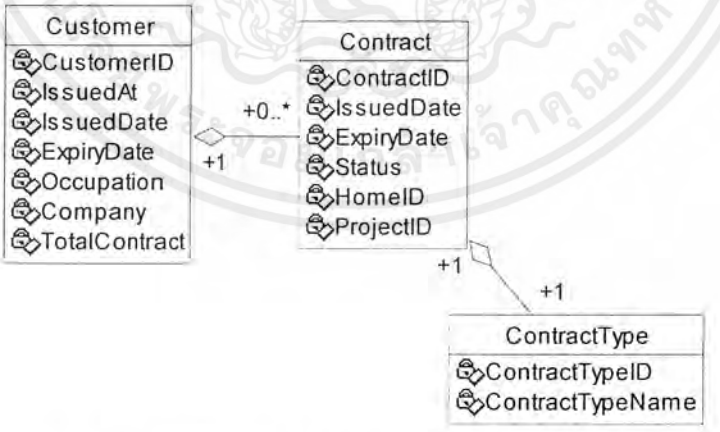
- ตาราง Employee ก็จะประกอบไปด้วย EmployeeID, TotalPrivilege, Firstname, Lastname, Sex, Nationality, Race, Religion, Status, PresentAddress, Street, District, Province, Telephone, Pager, IDcardNumber, Salary, EnterID

จาก ตารางที่ได้มาทั้ง 2 ตารางคือ Customer และ Employee นั้นยังไม่สมบูรณ์ หรือยังเอาไปใช้งานไม่ได้ เราจะต้องมาดูที่ความสัมพันธ์แบบอะกริเกชันอีกด้วย ว่าทั้ง 2 คลาสที่แมปออกมาเป็นตารางนั้นมีความสัมพันธ์อยู่กับคลาสใดหรือไม่ถ้ามี เราก็จะต้องทำตามหลักการในข้อที่ 2 ต่อไป โดยก่อนที่จะทำการแมปออบเจ็กต์เป็นฐานข้อมูลรีเลชันนัลในขั้นตอนต่อไป เราจะต้องทำการเลือกแอตทริบิวต์จากแต่ละคลาสว่าจะให้แอตทริบิวต์ตัวใดภายในคลาสนั้นเป็นไพรมารี่คีย์เพื่อที่จะได้นำไปเพิ่มในตาราง เพื่อทำการกำหนดความสัมพันธ์ให้ ระหว่างคลาสเมื่อกำหนดได้แล้ว (ผู้เขียนใช้ ID ของแต่ละคลาสเป็นไพรมารี่คีย์ จึงไม่ขอกล่าวถึงขั้นตอนนี้อีก) เราก็จะมาทำการแมปในขั้นตอนต่อไปดังต่อไปนี้



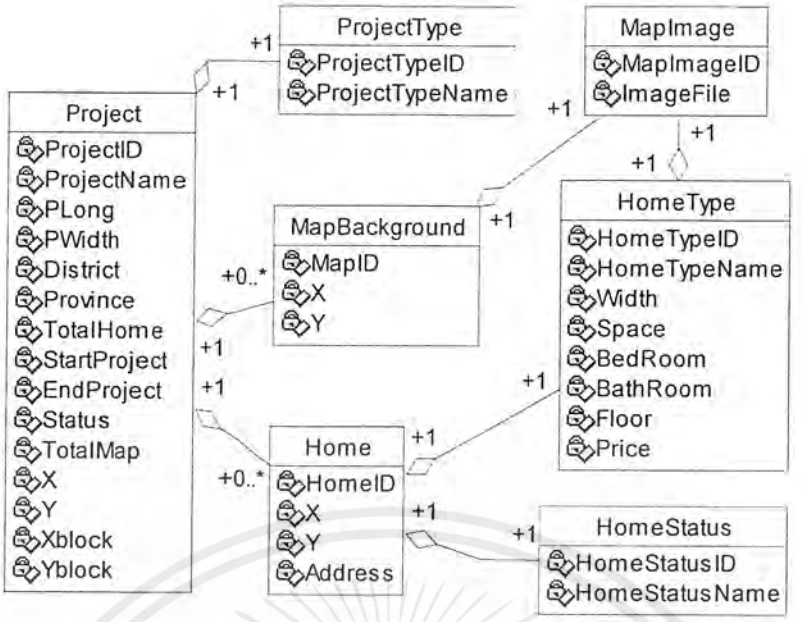
รูปที่ 3.5-3 แสดงความสัมพันธ์กันระหว่างคลาส Employee กับคลาสต่างๆ

จากรูปที่ 3.5-3 แสดงถึงความสัมพันธ์ระหว่างคลาส Employee และคลาสต่างๆ โดยเราจะนำมาพิจารณาที่ละคู่ เริ่มจาก ความสัมพันธ์ที่เป็น 1 to 1 เราก็จะนำ PositionID และ AccountID เพิ่มเข้าไปในตาราง Employee ส่วน 1 to many เราก็จะนำ EmployeeID เพิ่มเข้าไปใน ตาราง Privilege



รูปที่ 3.5-4 แสดงความสัมพันธ์ระหว่างคลาส Customer, Contract และ ContractType

จากรูปที่ 3.5-4 พิจารณาที่ความสัมพันธ์ 1 to 1 เราก็จะนำ ContractTypeID ไปใส่ไว้ในตาราง Contract และ 1 to many เราก็จะนำ CustomerID ไปใส่ไว้ในตาราง Contract



รูปที่ 3.5-5 แสดงความสัมพันธ์กันของคลาส Project กับคลาสต่างๆ

จากรูปที่ 3.5-5 สามารถแมปได้โดยมองที่ความสัมพันธ์แบบ 1 to 1 ก่อน ก็จะได้ว่านำ ProjectTypeID ไปใส่ไว้ใน ตาราง Project, MapImageID ใส่ไว้ใน 2 ตารางคือตาราง MapBackground และ ตาราง HomeType HomeTypeID, และ HomeStatusID ใส่ไว้ใน ตาราง Home ต่อมาก็มามองที่ 1 to many ก็จะนำ ProjectID จากตาราง Project มาใส่เพิ่มที่ตาราง MapBackground และ ตาราง Home

หลังจากที่ทำการแมปโดยมองความสัมพันธ์ทั้งหมดแล้วก็จะได้ตารางออก เราก็จะมาทำการกำหนดชนิดให้กับข้อมูลว่ามีชนิดอะไรบ้าง โดยทำการแปลงจากชนิดข้อมูลของแอตทริบิวต์ในออบเจ็กต์มาเป็นชนิดข้อมูลของออราเคิล แล้วเขียนออกมาเป็นตารางทั้งหมด 15 ตารางดังต่อไปนี้

3.6 แสดงตาราง (Table) ของฐานข้อมูลจากระบบทั้งหมด 15 ตาราง

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	CustomerID	Number		เลขที่ของลูกค้าเป็น P.K. NOT Null
2	FirstName	Varchar2	20	ชื่อ
3	LastName	Varchar2	20	นามสกุล
4	Sex	Varchar2	1	เพศ M / F
5	Nationality	Varchar2	20	สัญชาติ
6	Race	Varchar2	20	เชื้อชาติ
7	Religion	Varchar2	20	ศาสนา
8	Status	Varchar2	15	สถานะ
9	PresentAddress	Varchar2	10	บ้านเลขที่
10	Street	Varchar2	20	ถนน
11	District	Varchar2	20	ตำบล
12	Province	Varchar2	20	จังหวัด
13	Telephone	Varchar2	20	เบอร์โทรศัพท์
14	Pager	Varchar2	20	เบอร์เพจ
15	IdcardNumber	Varchar2	15	หมายเลขบัตรประจำตัวประชาชน
16	IssuedAt	Varchar2	20	ออก ณ. ที่
17	IssuedDate	Date		วันออกบัตร
18	ExpiryDate	Date		วันบัตรหมดอายุ
19	Occupation	Varchar2	20	อาชีพ
20	Company	Varchar2	20	ที่ทำงาน
21	Salary	Number		เงินเดือน
22	EnterID	Number		เลขที่ของผู้กรอกเป็น F.K.
23	TotalContract	Number		จำนวนรวมของสัญญาที่ได้ทำไว้

ตารางที่ 3-1 แสดงตารางที่ใช้เก็บข้อมูลของลูกค้าชื่อว่า Customer

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**EmployeeID**	Number		เลขที่ของลูกค้าเป็น P.K. NOT Null
2	FirstName	Varchar2	20	ชื่อ
3	LastName	Varchar2	20	นามสกุล
4	Sex	Varchar2	1	เพศ M / F
5	Nationality	Varchar2	20	สัญชาติ
6	Race	Varchar2	20	เชื้อชาติ
7	Religion	Varchar2	20	ศาสนา
8	Status	Varchar2	15	สถานะ
9	PresentAddress	Varchar2	10	บ้านเลขที่
10	Street	Varchar2	20	ถนน
11	District	Varchar2	20	ตำบล
12	Province	Varchar2	20	จังหวัด
13	Telephone	Varchar2	20	เบอร์โทรศัพท์
14	Pager	Varchar2	20	เบอร์เพจ
15	IdcardNumber	Varchar2	15	หมายเลขบัตรประจำตัวประชาชน
16	Salary	Number		เงินเดือน
17	EnterID	Number		เลขที่ของผู้กรอกเป็น F.K.
18	TotalPrivilege	Number		จำนวนรวมของ Project ที่มีสิทธิเข้าไปทำ
19	PositionID	Number		ID ของตำแหน่ง

ตารางที่ 3-2 แสดงตารางที่ใช้เก็บข้อมูลของพนักงานทั้งหมดชื่อว่า Employee

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	AccountID	Number		เลขที่ของ Account เป็น P.K. NOT Null
2	AccountName	Varchar2	20	ชื่อ user login
3	Password	Varchar2	20	Password ของ login
4	Visible	Varchar2	1	สามารถทำงานได้หรือไม่ T / F

ตารางที่ 3-3 แสดงตารางที่ใช้เก็บรหัสผ่านเข้าสู่ระบบชื่อว่า Account

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**CustomerID**	Number		P.K. NOT Null
2	**ContractID**	Number		P.K. NOT Null
3	IssuedDate	Date		วันเริ่มสัญญา
4	ExpiryDate	Date		วันหมดสัญญา
5	Status	Varchar2	1	สถานะของ Contract T / F
6	*HomeID*	Number		F.K.
7	*ContractType*	Number		F.K.
8	*ProjectID*	Number		F.K.

ตารางที่ 3-4 แสดงตารางที่ใช้เก็บสัญญาระหว่างลูกค้ากับโครงการชื่อว่า Contract

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**ContractTypeID**	Number		ID ของชนิดสัญญาเป็น P.K. NOT Null
2	ContractTypeName	Varchar2	20	ชื่อชนิดสัญญา

ตารางที่ 3-5 แสดงตารางที่ใช้เก็บชนิดของสัญญาชื่อว่า ContractType

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**EmployeeID**	Number		P.K. NOT Null
2	**PrivilegeID**	Number		P.K. NOT Null
3	*ProjectID*	Number		F.K.

ตารางที่ 3-6 แสดงตารางที่ใช้เก็บสิทธิในการเข้าสู่ระบบชื่อว่า Privilege

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**PositionID**	Number		ID ของตำแหน่งเป็น P.K. NOT Null
2	PositionName	Varchar2	20	ชื่อเรียกตำแหน่ง

ตารางที่ 3-7 แสดงตารางที่ใช้เก็บชื่อของตำแหน่งพนักงานชื่อว่า Position

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**CompanyName**	Varchar2	20	ชื่อบริษัทเป็น P.K. NOT Null
2	TotalProject	Number		จำนวนของโครงการทั้งหมด

ตารางที่ 3-8 แสดงตารางที่ใช้เก็บชื่อของบริษัทชื่อว่า Company

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**ProjectID**	Number		ID ของโครงการเป็น P.K. NOT Null
2	ProjectName	Varchar2	40	ชื่อของโครงการ
3	Long	Number		ความยาวของโครงการ
4	Width	Number		ความกว้างของโครงการ
5	District	Varchar2	20	อำเภอ
6	Province	Varchar2	20	จังหวัด
7	*ProjectTypeID*	Number		ID ของชนิดโครงการ F.K.
8	TotalHome	Number		จำนวนบ้านทั้งหมดของโครงการ
9	StartProject	Date		วันเริ่มโครงการ
10	EndProject	Date		วันปิดโครงการ
11	Status	Varchar2	1	สถานะของโครงการ T / F
12	TotalMap	Number		จำนวนของ Map ที่มีในแผนที่โครงการ
13	X	Number		จำนวน block ด้าน X
14	Y	Number		จำนวน block ด้าน Y
15	Xblock	Number		ขนาด block ด้าน x
16	Yblock	Number		ขนาด block ด้าน y

ตารางที่ 3-9 แสดงตารางที่ใช้เก็บข้อมูลของโครงการชื่อว่า Project

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**ProjectTypeID**	Number		ID ชนิดของโครงการ P.K. NOT Null
2	ProjectTypeName	Varchar2	20	ชื่อชนิดของโครงการ

ตารางที่ 3-10 แสดงตารางที่ใช้เก็บชนิดของโครงการชื่อว่า ProjectType

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**ProjectID**	Number		ID ของโครงการเป็น P.K. NOT Null
2	**MapID**	Number		ID ของ Map เป็น P.K. NOT Null
3	**X**	Number		ตำแหน่ง X บน Map
4	**Y**	Number		ตำแหน่ง Y บน Map
5	*MapImageID*	Number		F.K.

ตารางที่ 3-11 แสดงตารางที่ใช้เก็บแผนผังของโครงการชื่อว่า MapBackground

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**MapImageID**	Number		ID ของ MapImage เป็น P.K. NOT Null
2	ImageFile	Varchar2	40	ชื่อ File ของ Image

ตารางที่ 3-12 แสดงตารางที่ใช้เก็บชื่อของอิมเมจ (Image) ที่ใช้แสดงในแผนผังชื่อว่า MapImage

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**HomeTypeID**	Number		ID ของชนิดบ้านเป็น P.K. NOT Null
2	HomeTypeName	Varchar2	20	ชื่อชนิดของบ้าน
3	Width	Number		ความกว้างบ้าน
4	Space	Number		พื้นที่รวม
5	BedRoom	Number		จำนวนห้องนอน
6	BathRoom	Number		จำนวนห้องน้ำ
7	Floor	Number		จำนวนชั้น
8	Price	Number		ราคา
9	*MapImageID*	Number		F.K.

ตารางที่ 3--13 แสดงตารางที่ใช้เก็บชนิดของบ้านชื่อว่า HomeType

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**ProjectID**	Number		ID ของ โครงการ P.K. NOT Null
2	**HomeID**	Number		ID ของบ้าน P.K. NOT Null
3	**X**	Number		ตำแหน่ง X บ้านบน Map
4	**Y**	Number		ตำแหน่ง Y บ้านบน Map
5	Address	Varchar2	10	บ้านเลขที่
6	*HomeTypeID*	Number		F.K.
7	*HomeStatusID*	Number		F.K.

ตารางที่ 3—14 แสดงตารางที่ใช้เก็บบ้านในโครงการชื่อว่า Home

ลำดับ	ชื่อ	ชนิด	กว้าง	อธิบาย
1	**HomeStatusID**	Number		ID ของ HomeStatus เป็น P.K. NOT Null
2	HomeStatusName	Varchar2	10	ชื่อของ HomeStatus

ตารางที่ 3—15 แสดงตารางที่ใช้เก็บชื่อสถานะของบ้านชื่อว่า HomeStatus

จากการที่เราได้ทำการแมปจากออบเจ็กต์มาเป็นฐานข้อมูลรีเลชันนัลแล้ว ก็จะได้ตารางออกมาทั้งหมดรวม 15 ตารางดังที่แสดงผ่านมาแล้วในตารางที่ 3-1 ถึง 3-15 นั้น เราก็จะนำตารางเหล่านี้ไปสร้างโดยใช้คำสั่ง SQL ในการสร้างตาราง เพื่อใช้ในการเก็บฐานข้อมูลในออราเคิล โดยแสดงคำสั่ง SQL ในหัวข้อถัดไป

3.7 แสดงการใช้คำสั่ง SQL ในการสร้างตารางฐานข้อมูลในออราเคิล

3.7.1 สร้างตารางและกำหนดไพรมารี่คีย์ (Primary key) ให้กับตาราง

```
create table customer (
customerid          number          not null,
firstname           varchar2(20)    ,
lastname varchar2(20)    ,
sex                 varchar2(1)     ,
nationality         varchar2(20)    ,
race                varchar2(20)    ,
religion            varchar2(20)    ,
status              varchar2(15)    ,
presentaddress      varchar2(10)    ,
street              varchar2(20)    ,
district            varchar2(20)    ,
province            varchar2(20)    ,
telephone           varchar2(20)    ,
pager               varchar2(20)    ,
idcardnumber        varchar2(15)    ,
issuedat            varchar2(20)    ,
issueddate          date            ,
expirydate          date            ,
occupation          varchar2(20)    ,
company             varchar2(20)    ,
salary              number          ,
enterid             number          ,
totalcontract       number          ,
primary key (customerid));
```

รูปที่ 3.7-1 แสดงการสร้างตาราง Customer ด้วยคำสั่ง SQL

```
create table employee (
employeeid      number          not null,
firstname       varchar2(20)    ,
lastname varchar2(20)    ,
sex             varchar2(1)     ,
nationality     varchar2(20)    ,
race            varchar2(20)    ,
religion        varchar2(20)    ,
status          varchar2(15)    ,
presentaddress  varchar2(10)    ,
street          varchar2(20)    ,
district        varchar2(20)    ,
province        varchar2(20)    ,
telephone       varchar2(20)    ,
pager           varchar2(20)    ,
idcardnumber    varchar2(15)    ,
salary          number          ,
enterid         number          ,
totalprivilege  number          ,
positionid      number          ,
accountid       number          ,
primary key (employeeid));
```

รูปที่ 3.7-2 แสดงการสร้างตาราง Employee ด้วยคำสั่ง SQL

```
create table account (
accountid       number  not null,
accountname     varchar2(20)    ,
password        varchar2(20)    ,
visible         varchar2(1)     ,
primary key (accountid));
```

รูปที่ 3.7-3 แสดงการสร้างตาราง Account ด้วยคำสั่ง SQL

```
create table contract (
customerid      number          not null,
contractid      number          not null,
issueddate      date            ,
expirydate      date            ,
status          varchar2(1)     ,
homeid          number          ,
contracttypeid  number          ,
projectidnumber                 ,
primary key (customerid, contractid));
```

รูปที่ 3.7-4 แสดงการสร้างตาราง Contract ด้วยคำสั่ง SQL

```
create table contracttype (
contracttypeid      number          not null,
contracttypename    varchar2(20)    ,
primary key (contracttypeid));
```

รูปที่ 3.7-5 แสดงการสร้างตาราง ContractType ด้วยคำสั่ง SQL

```
create table privilege (
employeeid      number          not null,
privilegeid     number          not null,
projectidnumber                 ,
primary key (employeeid,privilegeid));
```

รูปที่ 3.7-6 แสดงการสร้างตาราง Privilege ด้วยคำสั่ง SQL

```
create table position (
positionid      number          not null,
positionname    varchar2(20)    ,
primary key (positionid));
```

รูปที่ 3.7-7 แสดงการสร้างตาราง Position ด้วยคำสั่ง SQL

```
create table company (
companyname     varchar2(20)     not null,
totalproject    number           ,
primary key (companyname));
```

รูปที่ 3.7-8 แสดงการสร้างตาราง Company ด้วยคำสั่ง SQL

```
create table project (
projectidnumber          not null,
projectname     varchar2(40)     ,
plong           number           ,
pwidth          number           ,
district        varchar2(20)     ,
province        varchar2(20)     ,
projecttypeid   number           ,
totalhome       number           ,
startproject    date             ,
endproject      date             ,
status          varchar2(1)      ,
totalmap        number           ,
x               number           ,
y               number           ,
xblock          number           ,
yblock          number           ,
primary key (projectid));
```

รูปที่ 3.7-9 แสดงการสร้างตาราง Project ด้วยคำสั่ง SQL

```
create table projecttype (
projecttypeid     number          not null,
projecttypename   varchar2(20)    ,
primary key (projecttypeid));
```

รูปที่ 3.7-10 แสดงการสร้างตาราง ProjectType ด้วยคำสั่ง SQL

```
create table mapbackground (
projectid number not null,
mapid number not null,
x number ,
y number ,
mapimageid number ,
primary key (projectid,mapid));
```

รูปที่ 3.7-11 แสดงการสร้างตาราง MapBackground ด้วยคำสั่ง SQL

```
create table mapimage (
mapimageid number not null,
imagefile varchar2(40) ,
primary key (mapimageid));
```

รูปที่ 3.7-12 แสดงการสร้างตาราง MapImage ด้วยคำสั่ง SQL

```
create table hometype (
hometypeid number not null,
hometypename varchar2(20) ,
width number ,
space number ,
bedroom number ,
bathroom number ,
floor number ,
price number ,
mapimageid number ,
primary key (hometypeid));
```

รูปที่ 3.7-13 แสดงการสร้างตาราง HomeType ด้วยคำสั่ง SQL

```
create table home (
projectidnumber          not null,
homeid          number          not null,
x               number          ,
y               number          ,
address         varchar2(10)    ,
hometypeid      number          ,
homestatusid    number          ,
primary key (projectid,homeid));
```

รูปที่ 3.7-14 แสดงการสร้างตาราง Home ด้วยคำสั่ง SQL

```
create table homestatus (
homestatusid      number          not null,
homestatusname    varchar2(10)    ,
primary key (homestatusid));
```

รูปที่ 3.7-15 แสดงการสร้างตาราง HomeStatus ด้วยคำสั่ง SQL

3.7.2 กำหนดฟอร์เรนคีย์ (Foreign key) ให้กับตาราง

```
Alter table customer Add constraint fk_customer_enterid
Foreign key (enterid) references employee (employeeid) ;
```

รูปที่ 3.7-16 แสดงการกำหนด Foreign Key ของตาราง Customer ด้วยคำสั่ง SQL

```
Alter table employee Add constraint fk_employee_enterid
Foreign key (enterid) references  employee (employeeid)  ;
Alter table employee Add constraint fk_employee_positionid
Foreign key (positionid) references position (positionid)  ;
Alter table employee Add constraint fk_employee_accountid
Foreign key (accountid) references account (accountid)  ;
```

รูปที่ 3.7-17 แสดงการกำหนด Foreign Key ของตาราง Employee ด้วยคำสั่ง SQL

```
Alter table contract Add constraint fk_contract_homeid_projectid
Foreign key (projectid,homeid) references  home (projectid,homeid)  ;
Alter table contract Add constraint fk_contract_contracttypeid
Foreign key (contracttypeid) references contracttype (contracttypeid) ;
```

รูปที่ 3.7-18 แสดงการกำหนด Foreign Key ของตาราง Contract ด้วยคำสั่ง SQL

```
Alter table privilege Add constraint fk_privilege_projectid
Foreign key (projectid) references project (projectid) ;
```

รูปที่ 3.7-19 แสดงการกำหนด Foreign Key ของตาราง Privilege ด้วยคำสั่ง SQL

```
Alter table project Add constraint fk_project_projecttypeid
Foreign key (projecttypeid) references projecttype (projecttypeid) ;
```

รูปที่ 3.7-20 แสดงการกำหนด Foreign Key ของตาราง Project ด้วยคำสั่ง SQL

```
Alter table mapbackground Add constraint fk_mapbackground_mapimageid
Foreign key (mapimageid) references mapimage (mapimageid) ;
```

รูปที่ 3.7-21 แสดงการกำหนด Foreign Key ของตาราง MapBackground ด้วยคำสั่ง SQL

```
Alter table hometype Add constraint fk_hometype_mapimageid
Foreign key (mapimageid) references mapimage (mapimageid) ;
```

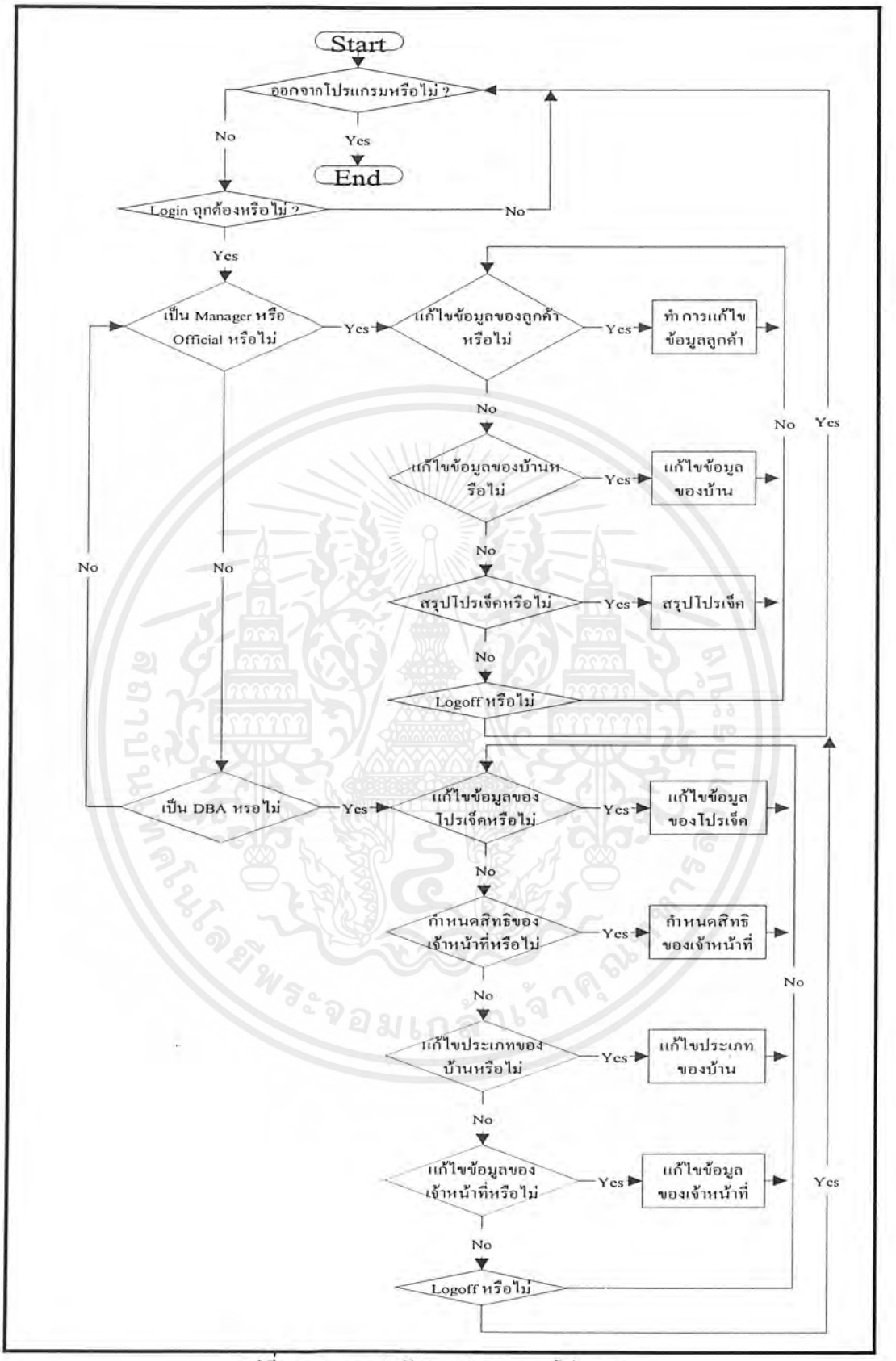
รูปที่ 3.7-22 แสดงการกำหนด Foreign Key ของตาราง HomeType ด้วยคำสั่ง SQL

```
Alter table home Add constraint fk_home_hometypeid
Foreign key (hometypeid) references hometype (hometypeid) ;
Alter table home Add constraint fk_home_homestatusid
Foreign key (homestatusid) references homestatus (homestatusid) ;
```

รูปที่ 3.7-23 แสดงการกำหนด Foreign Key ของตาราง Home ด้วยคำสั่ง SQL

จากตารางดังกล่าวหลังจากที่เราได้ทำการสร้างขึ้นมาเสร็จเรียบร้อยแล้ว เราก็จะทำการใส่ข้อมูลลงไปในตารางที่ใช้เป็นตารางอ้างอิงข้อมูล (reference table) คือ ContractType, HomeStatus, MapImage, ProjectType และ Position ซึ่งข้อมูลในตารางเหล่านี้จะถูกอ้างถึงในขณะที่โปรแกรมทำงาน เพื่อใช้ร่วมกับตารางอื่นๆ และไม่มีการเปลี่ยนแปลงบ่อยครั้งนัก

3.8 การออกแบบหน้าจอในการติดต่อกับผู้ใช้

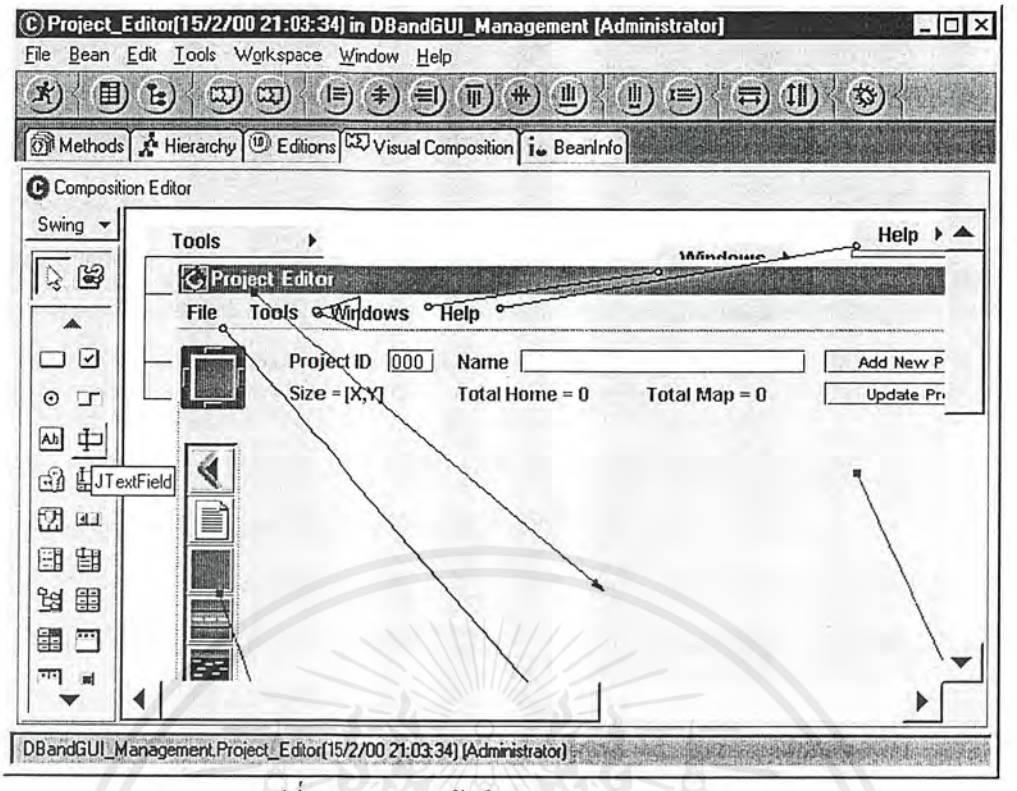


รูปที่ 3.8-1 แสดงลำดับการทำงานของโปรแกรม

จากการที่เราได้ทำการออกแบบในส่วนของคลาสไดอะแกรมและกำหนดเมดธอดให้กับคลาสแต่ละตัวแล้ว เราก็จะมาดูที่ลำดับการทำงานของระบบทั้งหมด ซึ่งแสดงในรูปที่ 3.8-1 การทำงานของระบบในแต่ละส่วนจะต้องอาศัยหน้าจอในการติดต่อกับผู้ใช้ (User Interface) ดังนั้นเราก็จะต้องออกแบบและทำการสร้างหน้าจอเหล่านั้นขึ้นมา เพื่อให้ระบบสามารถทำงานได้ เราก็จะมีขั้นตอนในการออกแบบหน้าจอในการติดต่อกับผู้ใช้โดยมีขั้นตอนดังต่อไปนี้

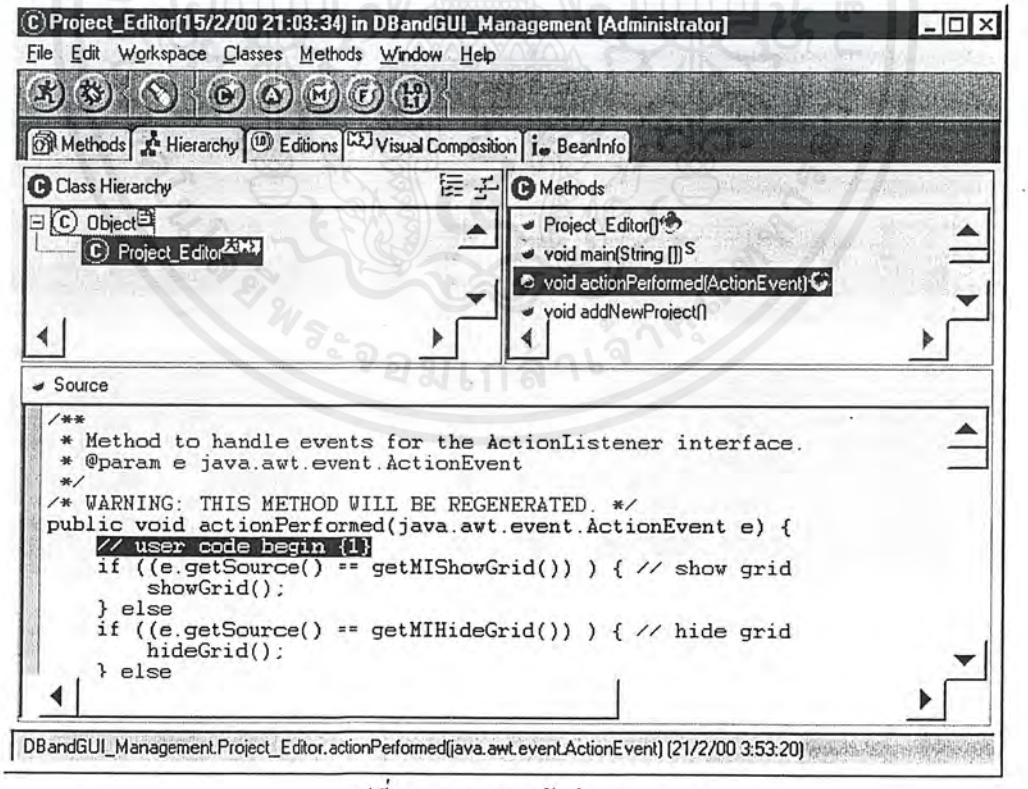
- ทำการกำหนดหน้าจอยูสเซอร์อินเตอร์เฟช ทั้งหมดที่เราจะต้องทำการออกแบบ ว่าประกอบไปด้วยหน้าจออะไรบ้าง
- ทำการกำหนดรายละเอียดบนหน้าจอของยูสเซอร์อินเตอร์เฟช ว่าประกอบไปด้วย คอมโพแนนซ์อะไรบ้าง
- หลังจากที่กำหนดรายละเอียดแล้ว ก็จะทำการกำหนดเหตุการณ์ (Event) ให้กับคอมโพแนนซ์ว่าแต่ละคอมโพแนซ์มีเหตุการณ์ อะไรกันบ้าง (ในที่นี้ผู้เขียนจะกำหนดเฉพาะเหตุการณ์ หลักๆเท่านั้น ส่วนที่เหลือจะทำการเพิ่มเติมโค้ดที่หลัง)
- ทำการสร้างยูสเซอร์อินเตอร์เฟช ด้วย VisualAge for java โดยวางคอมโพแนนซ์ลงไปตามที่ได้ออกแบบไว้ และลากเส้นเหตุการณ์ (Event) เชื่อมกันระหว่างคอมโพแนนซ์
- ทำการเพิ่มเติมโค้ดของโปรแกรมเข้าไปในส่วนที่เราต้องการเพิ่มเติม แก้ไข นอกจากที่ตัวเครื่องมือ (Visual) แปลงโค้ดออกมาให้เรา เนื่องจากที่เราสร้างยูสเซอร์อินเตอร์เฟชนั้นเป็นเพียงส่วนหน้าจอเท่านั้น ส่วนหลักๆ ไม่ว่าจะเป็นการควบคุมการติดต่อกับฐานข้อมูล โดยการเรียกคลาสที่เราได้สร้างและ ออกแบบเอาไว้นั้น เราจะต้องเป็นคนเขียนโค้ดส่วนนั้นเอง

จากขั้นตอนดังกล่าว ก็จะทำการแสดงตัวอย่าง หน้าจอของ VisualAge for Java ที่ใช้ในการสร้างหน้าจอติดต่อกับผู้ใช้ หรือที่เรียกว่า Visual Composition และหน้าจอที่ใช้ในการเพิ่มเติมโค้ดเข้าไปเพื่อทำการควบคุมระบบ หรือที่เรียกว่า หน้าจอ Hierarchy ซึ่งใช้ในการแก้ไขโค้ด กรณีที่คลาสนั้นเป็นคลาสที่ใช้ในการแสดงหน้าจอในการติดต่อกับผู้ใช้ ดังรูปที่ 3.8-2 และ 3.8-3 ตามลำดับ



รูปที่ 3.8-2 แสดงหน้าต่าง Visual Composition

จากรูปที่ 3.8-2 แสดงหน้าต่าง Visual Composition ที่ใช้ในการแก้ไขหน้าจอยูสเซอร์อินเตอร์เฟช โดยทางด้านซ้ายมือของหน้าต่าง จะเป็นส่วนของคอมโพแนนซ์ที่ให้เราเลือกใช้ในการนำมาวางลงในหน้า



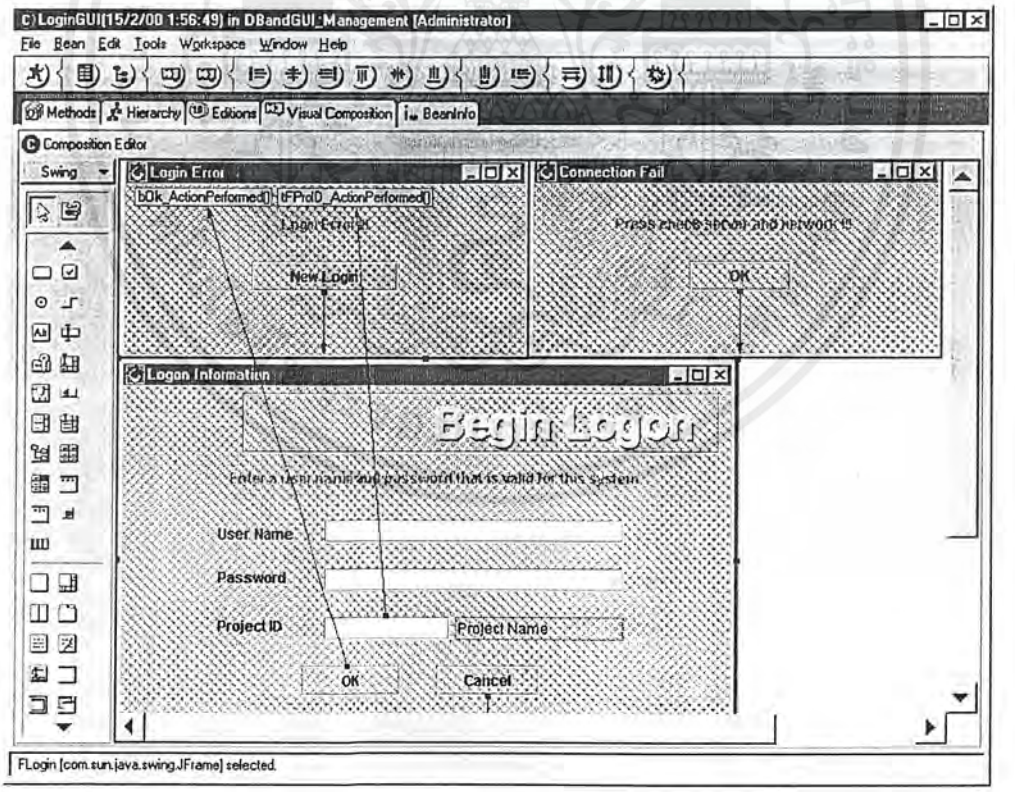
รูปที่ 3.8-3 แสดงหน้าต่าง Hierarchy

จอที่เราได้ออกแบบไว้ โดยมีคอมโพแนนซ์หลักๆก็เป็น AWT กับ SWING โดย SWING นั้นจะเป็นคอมโพแนนซ์ที่มีออบชันให้เลือกใช้มากกว่า AWT แต่ก็ทำงานช้ากว่าเช่นกัน หลังจากที่เราทำการสร้างยูสเซอร์อินเตอร์เฟซเรียบร้อยแล้ว เราก็จะทำการแก้ไขโค้ดต่างๆเพิ่มเติมเข้าไป แสดงดังรูปที่ 3.8-3

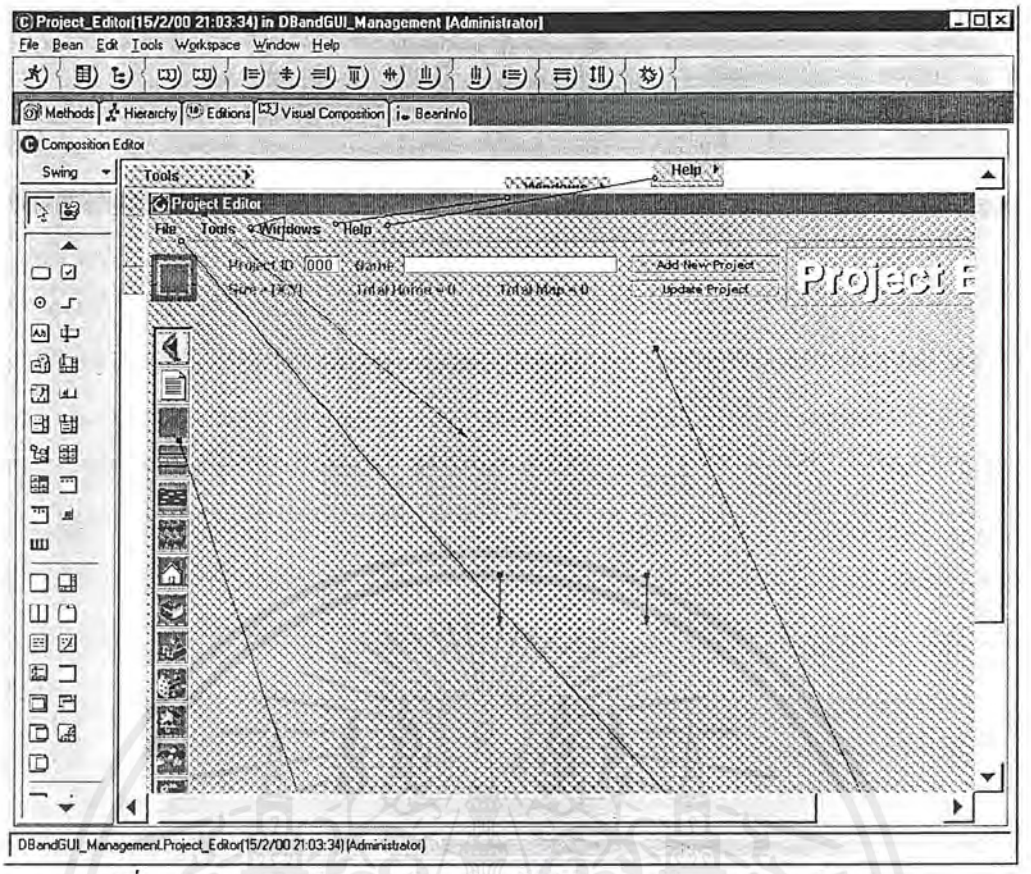
จากรูปที่ 3.8-3 แสดงหน้าจอ Hierarchy ของระบบนั้น นอกจากนี้เราก็สามารถที่จะแก้ไขโค้ดทั้งหมดที่หน้าต่างนี้ เพื่อที่จะเพิ่มเติมในส่วนที่จำเป็น เพื่อที่จะทำให้ยูสเซอร์อินเตอร์เฟซที่เราสร้างขึ้นมาสามารถทำงานได้จริง โดยเมื่อทำการคอมไพล์แล้ว ตัว VisualAge ก็จะทำการแปลงโค้ดจากหน้าจอยูสเซอร์อินเตอร์เฟซที่เราสร้างออกมาเป็นโค้ดโปรแกรมตามที่เราต้องการและจะเว้นพื้นที่ไว้ให้เราแทรกโค้ดเข้าไปเอง โดยจะใช้เป็นคำคอมเมนท์หัวท้ายเอาไว้ว่า "// user code begin {1}" และ "// user code end{1}" เราก็สามารถที่จะเติมโค้ดเข้าไปได้ในส่วนนี้

ต่อไปก็จะทำการแสดงรูปตัวอย่างในการสร้างหน้าจอและการลากเหตุการณ์ระหว่างคอมโพแนนซ์ต่างๆในภาพหน้าจอ เช่น

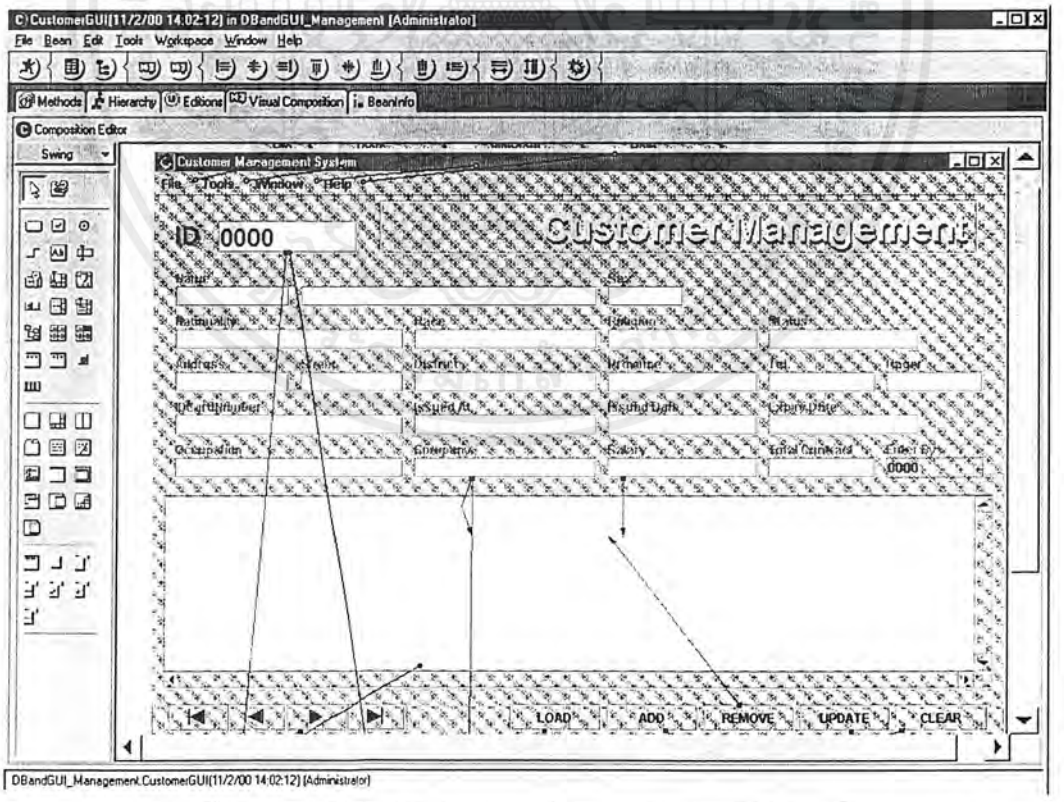
- หน้าจอล็อกอิน
- หน้าจอแก้ไขแผนผังของโครงการ
- หน้าจอแก้ไขข้อมูลลูกค้า
- หน้าจอที่ใช้บริการลูกค้า



รูปที่ 3.8-4 แสดงตัวอย่างการสร้างหน้าจอการล็อกอินเข้าสู่ระบบ



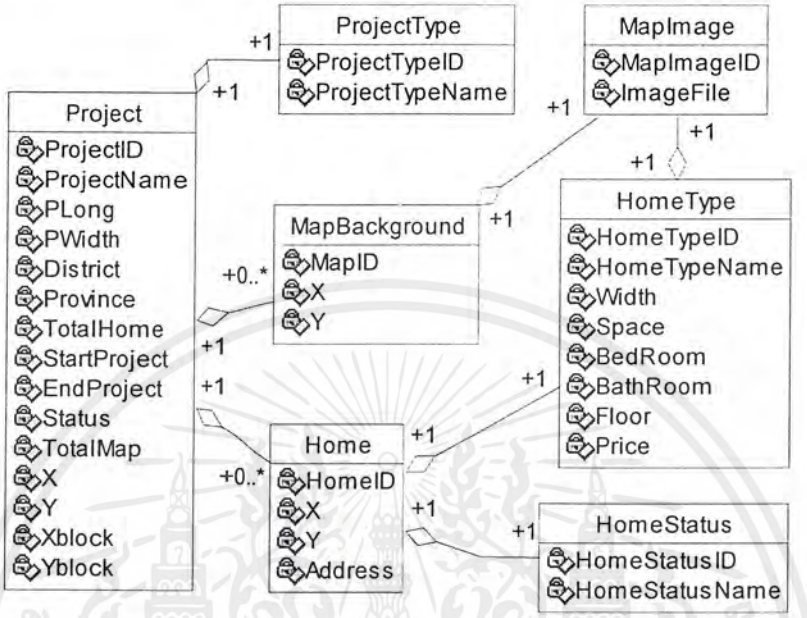
รูปที่ 3.8-5 แสดงตัวอย่างการสร้างหน้าจอไว้ใช้ในการสร้างแผนผังของโครงการ



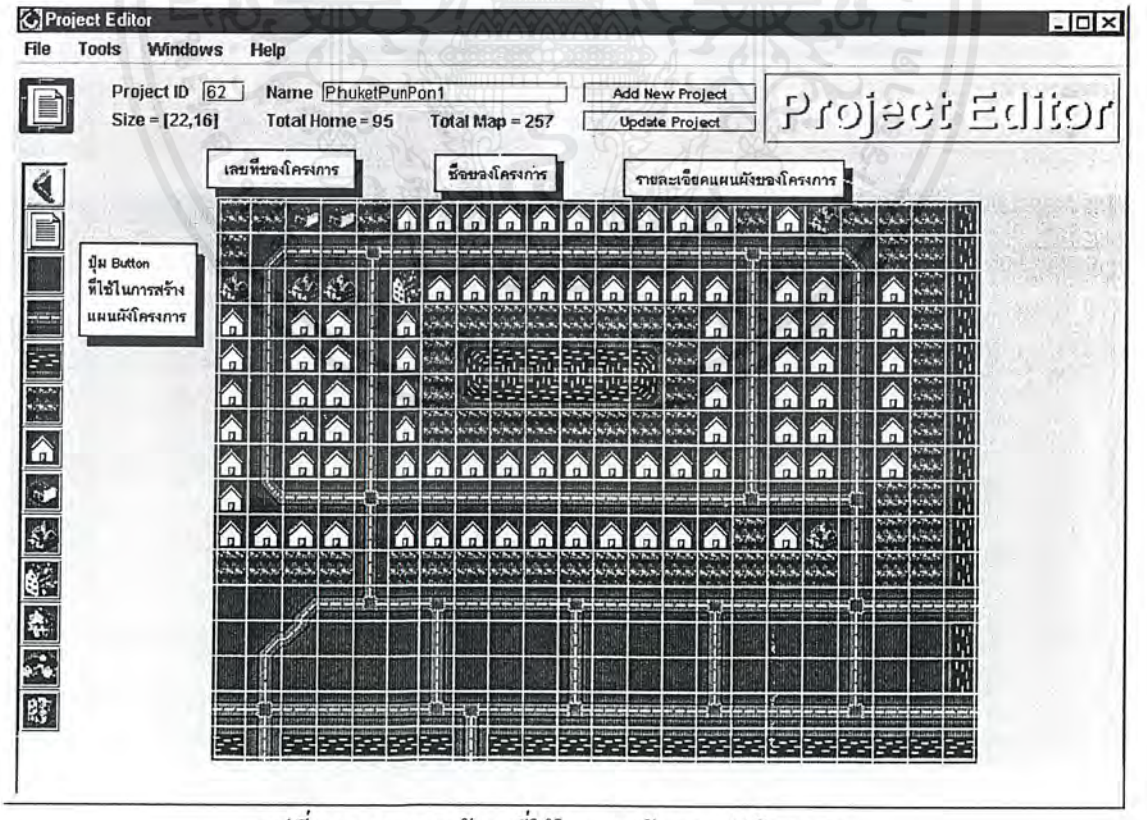
รูปที่ 3.8-6 แสดงตัวอย่างการสร้างหน้าจอของการเก็บข้อมูลลูกค้า

3.9 การจัดเก็บแผนผังของโครงการ

แผนผังของโครงการเป็นส่วนที่สำคัญของระบบ ประกอบไปด้วย ข้อมูลของโครงการ, บ้าน และสิ่งแวดล้อมต่างๆ ที่อยู่รอบๆโครงการ ที่เราได้ออกแบบมาแล้วทั้งคลาสไดอะแกรมและ ฐานข้อมูลในการจัดเก็บ ต่อไปก็จะขออธิบายถึงรายละเอียดในการจัดเก็บแผนผังโครงการเข้าสู่ระบบ



รูปที่ 3.9-1 แสดงคลาสไดอะแกรมของตัวโครงการ

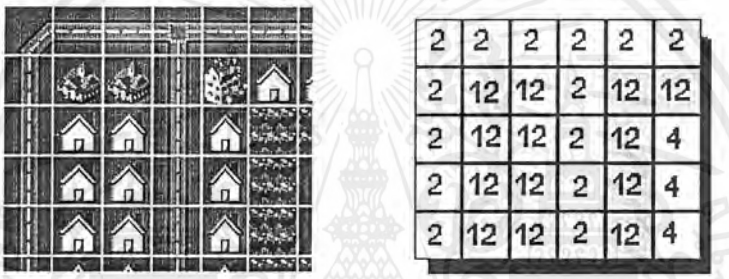


รูปที่ 3.9-2 แสดงหน้าจอที่ใช้ในการสร้างแผนผังโครงการ

จากรูปที่ 3.9-1 แสดงถึงคลาสไดอะแกรมของตัวโครงการที่ประกอบไปด้วยองค์ประกอบต่างๆ ไม่ว่าจะเป็นตัวบ้าน ตัวแผนผัง รวมไปถึงภาพที่ใช้ในการแสดงแผนผัง ในรูปที่ 3.9-2 เป็นรูปตัวอย่างของหน้าจอการสร้างแผนผัง ซึ่งปุ่มต่างๆในด้านซ้ายมือก็คือ คอมโพแนนซ์ต่างๆที่เราจะนำมาใส่ไว้ในแผนผัง ไม่ว่าจะเป็นบ้าน ถนน ต้นไม้ หรือว่าแม่น้ำ นอกจากคอมโพแนนซ์เหล่านี้แล้วที่นำมาใส่ไว้ในแผนผัง เราก็จะต้องทำการกำหนดรายละเอียดของโครงการไว้ที่ด้านบนของแผนผัง ตัวแผนผังเองก็ประกอบไปด้วยบล็อกสี่เหลี่ยมขนาดความกว้างสามารถปรับไป สูงสุดที่ 30*18 โดยเราจะมีการจัดเก็บตัวแผนผังของโครงการนี้ลงฐานข้อมูลโดยมีหลักการดังต่อไปนี้

แผนผังของโครงการจะประกอบไปด้วยส่วนหลักๆ 3 ส่วนคือ

- ข้อมูลรายละเอียดของโครงการ
- บ้านในโครงการ ประกอบไปด้วยรายละเอียดต่างๆของบ้าน
- แผนผังของโครงการ ในที่นี้หมายถึง สิ่งแวดล้อมต่างๆรอบๆโครงการ เช่น ถนน ต้นไม้ แม่น้ำ เป็นต้น



รูปที่ 3.9-3 แสดงส่วนหนึ่งของแผนผังและข้อมูลที่อยู่ในอาร์เรย์

จากรูปที่ 3.9-2 เป็นรูปที่แสดงแผนผังทั้งหมดของโครงการหนึ่ง ได้ตัดบางส่วนออกมาแสดงให้เห็นดังในรูปที่ 3.9-3 แสดงให้เห็นอย่างชัดเจน โดยแผนผังของโครงการในขณะที่ทำการแก้ไขนั้น จะถูกเก็บเป็นข้อมูลในตัวแปรอาร์เรย์ที่ใช้เก็บแผนผัง และจะมีอาร์เรย์อีกชุดที่ใช้เก็บรายละเอียดของบ้านแต่ละหลัง ในกรณีที่ในอาร์เรย์ช่องนั้นเป็นบ้าน เพราะบ้านไม่ใช่เก็บว่าตำแหน่งนั้นเป็นบ้านก็เพียงพอแล้ว แต่เราจะต้องทำการเก็บรายละเอียดของบ้านหลังนั้นเอาไว้ด้วย โดยอาร์เรย์ที่ใช้ในการเก็บข้อมูลรายละเอียดของบ้านก็มีชนิดเป็นออบเจ็กต์ Home

เมื่อผู้ใช้ต้องการที่จะจัดเก็บแผนผังที่สร้างขึ้นลงฐานข้อมูล เราก็จะแบ่งการจัดเก็บออกเป็น 2 ส่วนคือ

1) ส่วนที่เป็นแผนผัง ไม่ใช่บ้านก่อน จะถูกเก็บไว้ในตารางชื่อ MapBackground
2) ส่วนที่เป็นบ้านก็จะถูกเก็บไว้ในตารางที่ชื่อว่า Home

โดยเมื่อเราทำการสร้างแผนผังของโครงการเสร็จแล้วโปรแกรมก็จะทำการบันทึกข้อมูลของรายละเอียดโครงการลงไปยังตาราง Project ก่อนแล้วก็ตาราง Mapbackground และ Home ตามลำดับ จะขอแสดงรายละเอียด และขั้นตอนจัดเก็บดังต่อไปนี้

1) นำข้อมูลรายละเอียดของโครงการจัดเก็บลงฐานข้อมูล โดยจะทำการเซ็ตข้อมูลต่างๆ ของโครงการตามที่ได้ใส่ไว้ในหน้าจอดังรูปที่ 3.9-2 และทำการเพิ่มโครงการเข้าไปยังฐานข้อมูล แล้วก็เซ็ตชนิดของโครงการ แสดงลำดับขั้นตอนของโปรแกรมดังรูปที่ 3.9-4

```
project.setString("ProjectName",""+getTFProjectName().getText()); // Set ชื่อของโครงการ
project.setNumber("PLong",fNY*10);
project.setNumber("PWidth",fNX*10);
project.setString("District","-");
project.setString("Province","Bangkok");
project.setNumber("TotalHome",totalHome);
project.setDate2("StartProject","1/Jan/2000");
project.setDate2("EndProject","1/Jan/2999");
project.setString("Status","T");
project.setNumber("TotalMap",totalMap-totalHome);
project.setNumber("X",fNX);
project.setNumber("Y",fNY);
project.setNumber("Xblock",25);
project.setNumber("Yblock",25);
project.add2(conS);
project.setProjectType(conS,3);
```

รูปที่ 3.9-4 แสดงโปรแกรมส่วนที่ใช้ในการจัดเก็บรายละเอียดต่างๆของโครงการ

2) นำข้อมูลของแผนผังจัดเก็บลงฐานข้อมูล โดยจะใช้ For ลูปวนตรวจสอบค่าแต่ละจุดของแผนผังว่าเป็นอะไรบ้างถ้าเป็นบ้านคือ ค่าในอาร์เรย์จะมีค่าเป็น 12 ก็จะจัดเก็บลงตาราง Home โดยดึงรายละเอียดเข้าไปเก็บด้วย นอกจากนี้ก็จะจัดเก็บลงตาราง MapBackground แสดงโปรแกรมดังรูปที่ 3.9-5

```
int countMapID = 0;
int countHomeID = 0;
for(int x=1;x<=fNX;x++)  // For Loop ตามจำนวน block ในโครงการ
for(int y=1;y<=fNY;y++) {
  if (iMapArray[x][y] == 12) {
    project.home.setNumber("X",x);              // ค่า X,Y ของบ้านหลังนั้น
    project.home.setNumber("Y",y);  //
    project.home.setString("Address",home[x][y].getString("Address")); // เลขที่บ้าน
    project.home.add3(conS,++countHomeID);  // ลำดับของบ้านในโครงการ
    project.home.setHomeStatus(conS,home[x][y].status.getID());
    // สถานะเริ่มต้น ยังไม่มีใครเช่า
    project.home.setHomeType(conS,home[x][y].type.getNumber("HomeTypeID"));
    //  ชนิดของบ้าน
   } else {
        project.map.setNumber("X",x);  // ค่า X,Y ของ Map
        project.map.setNumber("Y",y);
        project.map.add3(conS,++countMapID);  // ลำดับของ Map
        project.map.setMapImage(conS,iMapArray[x][y]);  // ภาพของ Map
    }
  }
}
```

รูปที่ 3.9-5 แสดงโปรแกรมส่วนที่จัดเก็บรายละเอียดในแผนผังลงฐานข้อมูล

3.10 โปรแกรมส่วนของเจดีบีซี (JDBC)

เมื่อเราทำการสร้างหน้าจอในการติดต่อเสร็จเรียบร้อยแล้ว เราก็จะต้องนำหรือคลาส ที่เราได้เตรียมไว้มาใช้ หรือแก้ไขเพิ่มเติมเข้าไปตามที่ได้กล่าวมาแล้วนั้น ก็จะขอกล่าวถึงโค้ดในส่วนที่จำเป็นที่จะต้องใช้ในการติดต่อกับฐานข้อมูล โดยโปรแกรมในการติดต่อกับฐานข้อมูลดังกล่าวนี้ จะอยู่ในรูปของ เมธอด ในแต่ละคลาสเช่นคลาส Home, Project, Customer เป็นต้น ซึ่งจะประกอบไปด้วยส่วนสำคัญๆหลักๆ ในการติดต่อกับฐานข้อมูลก็คือ การ ADD, REMOVE, UPDATE และ LOAD ซึ่งจะแยกออกเป็นเมธอดๆ ทำงานกันคนละอย่าง โดยเมธอดทั้ง 4 นี้ไม่ว่าจะเป็น ADD, REMOVE, UPDATE หรือ LOAD นั้นก็จะมีโค้ดที่คล้ายๆกันในแต่ละคลาส ดังนั้นจึงขอยกตัวอย่างขึ้นมา 1 คลาส คือเมธอด ในคลาส Project จะอธิบายถึงการ ADD, REMOVE, UPDATE และ LOAD (ส่วนรายละเอียดในคลาสอื่นๆ หาดูได้จากซอสโค้ดแอพพลิเคชันของโครงงานนี้)

3.10.1 การสร้างการเชื่อมต่อกับฐานข้อมูล (Connect)

```
public Connection connectServer() throws SQLException, IOException {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver()); // กำหนดไดรเวอร์
        Connection conS = DriverManager.getConnection
        ("jdbc:oracle:thin:@161.246.6.121:1521:orcl","Project","july97");
        // กำหนด URL และคอนเน็คชันสตริง
        return conS; // คืนตัวแปรที่ใช้ในการเชื่อมต่อกลับไป เพื่อนำไปใช้
}
```

รูปที่ 3.10-1 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการเชื่อมต่อกับฐานข้อมูลออราเคิล

3.10.2 การโหลดข้อมูล (Load)

```
public boolean load(Connection conS,int id) throws SQLException, IOException {
```

จากโค้ดของโปรแกรมข้างบนนี้ เป็นการกำหนดส่วนหัวของเมธอดโดยรับ conS ที่ใช้ในการเชื่อมต่อกับฐานข้อมูลฝั่งเซิร์ฟเวอร์ , ส่วน id ก็คือการส่งค่าเข้ามาว่าต้องการโหลดข้อมูล project id ที่เท่าไหร และ ประกาศ throws SQLException และ IOException เอาไว้เพื่อเอาไว้กำหนดในภายหลัง ให้ข้ามไม่สนใจข้อผิดพลาดในส่วนนี้ไปก่อน

```
boolean b = false;
Statement stmt = conS.createStatement();
ResultSet rset = stmt.executeQuery ("select * from Project  where ProjectID = '"+id+"' and "+
                                    "StartProject is not null  and  EndProject is not null     ");
```

จากโค้ดโปรแกรมข้างบนนี้ b จะเป็นตัวแปร boolean ที่ใช้ในการคืนค่ากลับไปว่า ถ้าโหลดได้หรือว่ามีข้อมูลในฐานข้อมูลก็จะคืนค่าเป็นจริงนอกจากนี้เป็นเท็จแล้วก็ทำการกำหนดตัวแปรชนิด Statement และทำการสร้าง Query เพื่อ select ข้อมูลของโครงการ โดยมี projectid = id ที่พาสค่าเข้ามา โดยส่งผลลัพธ์กลับไปที่ Resultset ที่ชื่อว่า rset

```
while (rset.next()) {
b = true;
home = new Home(id);
map = new MapBackground(id);
type.load(conS,c.toInt(""+rset.getString(7)));
setNumber("ProjectID",c.toInt(""+rset.getString(1)));
setString("ProjectName",""+rset.getString(2));
setNumber("PLong",c.toInt(""+rset.getString(3)));
setNumber("PWidth",c.toInt(""+rset.getString(4)));
setString("District",""+rset.getString(5));
setString("Province",""+rset.getString(6));
setNumber("TotalHome",c.toInt(""+rset.getString(8)));
setDate("StartProject",c.toInt(""+rset.getString(9).substring(0,4)),
        m.toString(c.toInt(""+rset.getString(9).substring(5,7))),
        c.toInt(""+rset.getString(9).substring(8,10)));
setDate("EndProject",c.toInt(""+rset.getString(10).substring(0,4)),
        m.toString(c.toInt(""+rset.getString(10).substring(5,7))),
        c.toInt(""+rset.getString(10).substring(8,10)));
setString("Status",""+rset.getString(11));
setNumber("TotalMap",c.toInt(""+rset.getString(12)));
setNumber("X",c.toInt(""+rset.getString(13)));
setNumber("Y",c.toInt(""+rset.getString(14)));
setNumber("Xblock",c.toInt(""+rset.getString(15)));
setNumber("Yblock",c.toInt(""+rset.getString(16)));  }
```

จากโค้ดโปรแกรมตั้งแต่ while ลูปเป็นต้นมาจนจบลูป นั้นถ้า ResultSet ที่รับมานั้นมีข้อมูลอยู่ก็จะเข้าไปในลูป while ได้เป็นการเกจค่าเข้ามาเก็บไว้ในแอตทริบิวต์ต่างๆของออบเจ็กต์ project และเซ็ต b ที่ใช้ในการคืนค่ากลับเป็นจริงด้วย นอกจากนั้น เราจะมาพิจารณาโค้ดอีก 3 บรรทัดที่เพิ่มเติมเข้ามาก็คือ

```
home = new Home(id);
map = new MapBackground(id);
type.load(conS,c.toInt(""+rset.getString(7)));
```

โดย home, map ก็คือการกำหนดแอกกรีเกชัน (aggregation) แบบ 1 to 1 ให้กับ project ให้มี home และ map ได้ถูก new ขึ้นมา ส่วน type นั้นเป็นแอกกรีเกชันแบบ 1 to many type ก็จะทำการโหลดชนิดที่เป็นของ project ขึ้นมา โค้ดในส่วนนี้จะสัมพันธ์กันกับการออกแบบฐานข้อมูลและออกแบบคลาสในตอนแรก ที่ทำให้แต่ละออกเจ็กต์นั้นมีความสัมพันธ์กัน และเรียกใช้กันเมื่ออีกตัวถูกโหลดขึ้นมา

```
rset.close();
stmt.close();
return b; }
```

หลังจากที่เกจข้อมูลเข้ามาเก็บไว้ในแอตทริบิวต์เรียบร้อยแล้ว เราก็จะทำการปิด resultset ที่ชื่อว่า rset และปิด SQL สเตทเมนท์ที่ชื่อว่า stmt และคืนค่าบูลีนตัวแปร b กับไป ก็จะจบการทำงานของเมธอดในการโหลด

3.10.3 การลบข้อมูล (Remove)

```
public void remove(Connection conS,int id) throws SQLException, IOException    {
        Statement stmt = conS.createStatement();
        stmt.execute ("delete from project where projectid = '"+id+"'");
        stmt.close();
}
```

รูปที่ 3.10-2 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการลบข้อมูล

เนื่องจาก การลบไม่มีขั้นตอนที่ยุ่งยากและซับซ้อน ทำให้มีโค้ดที่สั้นกว่าการทำงานอื่นๆ ประกอบไปด้วย การกำหนดส่วนหัวให้กับเมธอด การกำหนดการสร้าง Statement และ ทำการ Execute query โดยไม่ต้องมีการรับ result set เพราะการลบไม่มีการคืนค่า result set กับมาให้ไคล์เอ็นท์ โดยจะทำการลบข้อมูลแถวที่มี project id = id ที่พาสค่าเข้ามา และทำการปิด Statement ก็จะจบการทำงานของเมธอดการลบ

3.10.4 การ แก้ไขอัพเดตข้อมูล (Update)

```
public void upDate(Connection conS,int id) throws SQLException, IOException {
        ProjectID = id;
        Statement stmtS = conS.createStatement();
        stmtS.execute("update Project set    "+
                    "    ProjectName    = '"+ProjectName+"'        ,"+
                    "    PLong          = '"+PLong+"'              ,"+
                    "    PWidth         = '"+PWidth+"'             ,"+
                    "    District       = '"+District+"'           ,"+
                    "    Province       = '"+Province+"'           ,"+
                    "    TotalHome      = '"+TotalHome+"'          ,"+
                    "    StartProject   = '"+getDateString("StartProject")+"',"+
                    "    EndProject     = '"+getDateString("EndProject")+"' ,"+
                    "    Status         = '"+Status+"'             ,"+
                    "    TotalMap       = '"+TotalMap+"'           ,"+
                    "    X              = '"+X+"'                  ,"+
                    "    Y              = '"+Y+"'                  ,"+
                    "    Xblock         = '"+Xblock+"'             ,"+
                    "    Yblock         = '"+Yblock+"'             "+
                    "where  ProjectID   = '"+ProjectID+"'  ");
        stmtS.close();
}
```

รูปที่ 3.10-3 แสดงโปรแกรมที่ใช้ JDBC ในส่วนของการอัพเดตข้อมูล

การอัพเดตก็ไม่มีขั้นตอนที่ยุ่งยากซับซ้อนอะไร เพียงแค่ทำการส่งคำสั่งอัพเดตไปยังเซิร์ฟเวอร์ โดยส่งค่าแอตทริบิวต์ต่างๆของออบเจ็กต์ไปให้และทำการอัพเดตแถวที่มี projectid = id ที่พาสค่าเข้ามา

3.10.5 การเพิ่มข้อมูล (Insert, Add)

รูปแบบของการเพิ่มข้อมูลคล้ายกับการอัพเดต เพียงแค่เปลี่ยนเป็นส่ง insert ไปยังเซิร์ฟเวอร์และกำหนด id ที่จะ insert เข้าไปให้เป็น id ที่ว่างอยู่และมีค่า id น้อยที่สุด

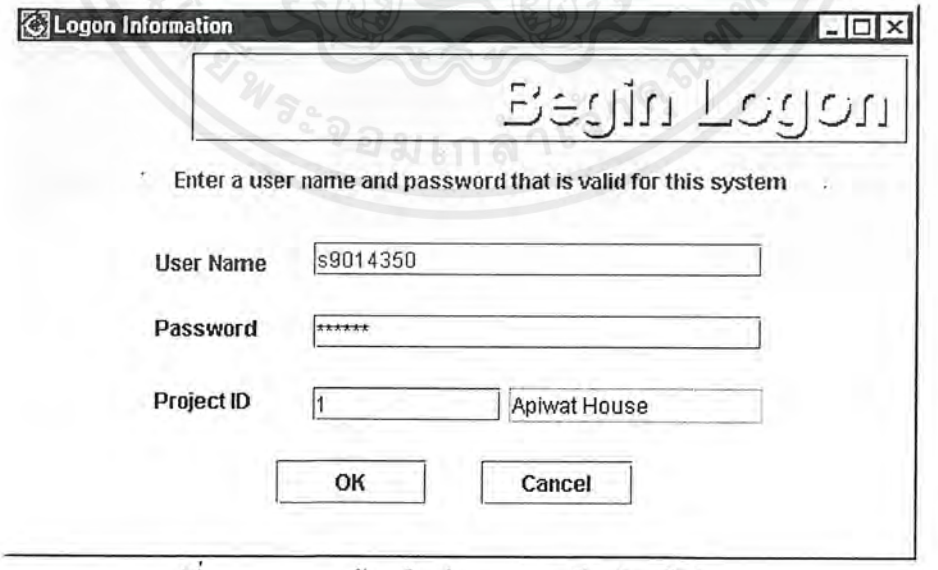
บทที่ 4

การทดสอบและผลการทดลอง

4.1 การทดสอบหน้าจอและแอพพลิเคชัน

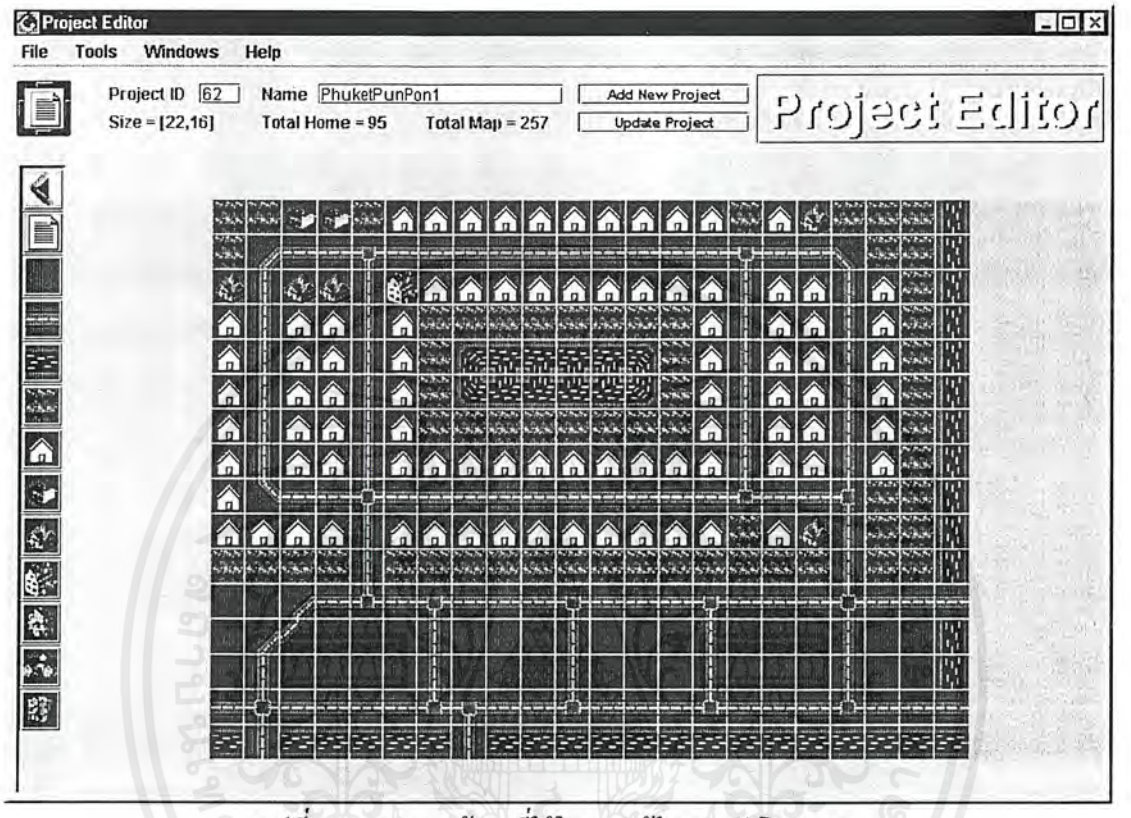
หลังจากที่ทำการออกแบบหน้าจอ ,แก้ไข เพิ่มเติมโค้ดและ สร้างคลาสที่ใช้ใน ระบบทั้งหมดเสร็จเรียบร้อยแล้ว เราก็จะได้แอพพลิเคชั่นออกมา โดยในแต่ละหน้าจอของการทำงาน เราได้มีการทดสอบการทำงานของ หน้าจอ หรือว่าคลาสนั้นๆเป็นที่เรียบร้อยและ ว่าสามารถทำงานได้อย่างถูกต้อง และไม่มีข้อผิดพลาด เราก็จะทำการทดสอบระบบรวม โดยนำคลาสต่างๆมาใช้งานรวมกัน โดยการเรียกผ่านเมนูที่เราสร้างขึ้นเพื่อเรียกใช้งานคลาสหรือ หน้าจอนั้นๆ เพื่อเป็นการรับรองว่า เมื่อนำคลาสต่างๆมารวมกันแล้วจะไม่เกิดข้อผิดพลาดขึ้น โดยทั่วไปแล้วการออกแบบโดยใช้ ออบเจ็กต์-โอเรียนเต็ด ทั้งการออกแบบ และการเขียนโปรแกรม เมื่อนำคลาสหรือ หน้าจอการทำงานต่างๆมารวมกัน หรือว่าใช้งานรวมกันแล้ว ก็มักจะไม่มีปัญหาเกิดขึ้น เพราะแต่ละคลาสนั้นมีแยกตัวแปรที่ชัดเจน ไม่มีการใช้ตัวแปรร่วมกัน ซึ่งเป็นข้อดีข้อหนึ่งของ ออบเจ็กต์-โอเรียนเต็ด ต่อไปก็จะแสดงหน้าจอตัวอย่างของการทำงานหลักๆของโปรแกรม ดังนี้

- หน้าจอของการล็อกอินเข้าสู่ระบบ
- หน้าจอของการแก้ไขแผนผังโครงการ
- หน้าจอของการแก้ไขข้อมูลของลูกค้า
- หน้าจอของการให้บริการลูกค้าในการซื้อขายสั่งจองบ้าน
- หน้าจอของการเพิ่มเติมข้อมูลของบ้านแต่ละหลัง เข้าสู่โครงการ



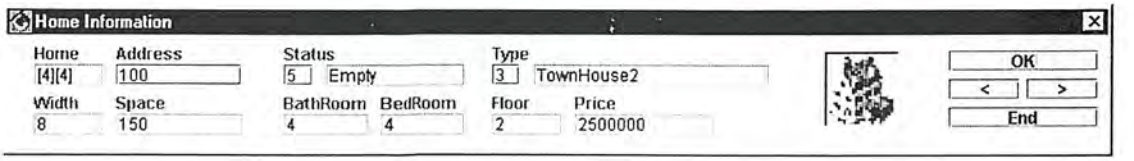
รูปที่ 4.1-1 แสดงหน้าจอในส่วนของการล็อกอินเข้าสู่ระบบ

จากรูปที่ 4.1-1 แสดงหน้าจอในส่วนของการล๊อกอินเข้าสู่ระบบ โดยผู้ใช้ที่เป็น Employee ไม่ว่าจะเป็น DBA, Manager หรือ Official จะต้องทำการล๊อกอินเข้าสู่ระบบก่อนที่จะเข้าไปทำงานกับระบบ โดยจะต้องป้อนชื่อ, รหัสและ เลขที่โครงการที่ต้องการจะเข้าไปจัดการ (ในกรณีของ DBA ไม่ต้องป้อนในส่วนนี้ เพราะมีสิทธิในการเข้าไปจัดการกับทุกๆโครงการ) โดย Employee ทุกคนจะต้องป้อนข้อมูลในส่วนนี้ให้ถูกต้อง จึงจะสามารถเข้าสู่ระบบได้



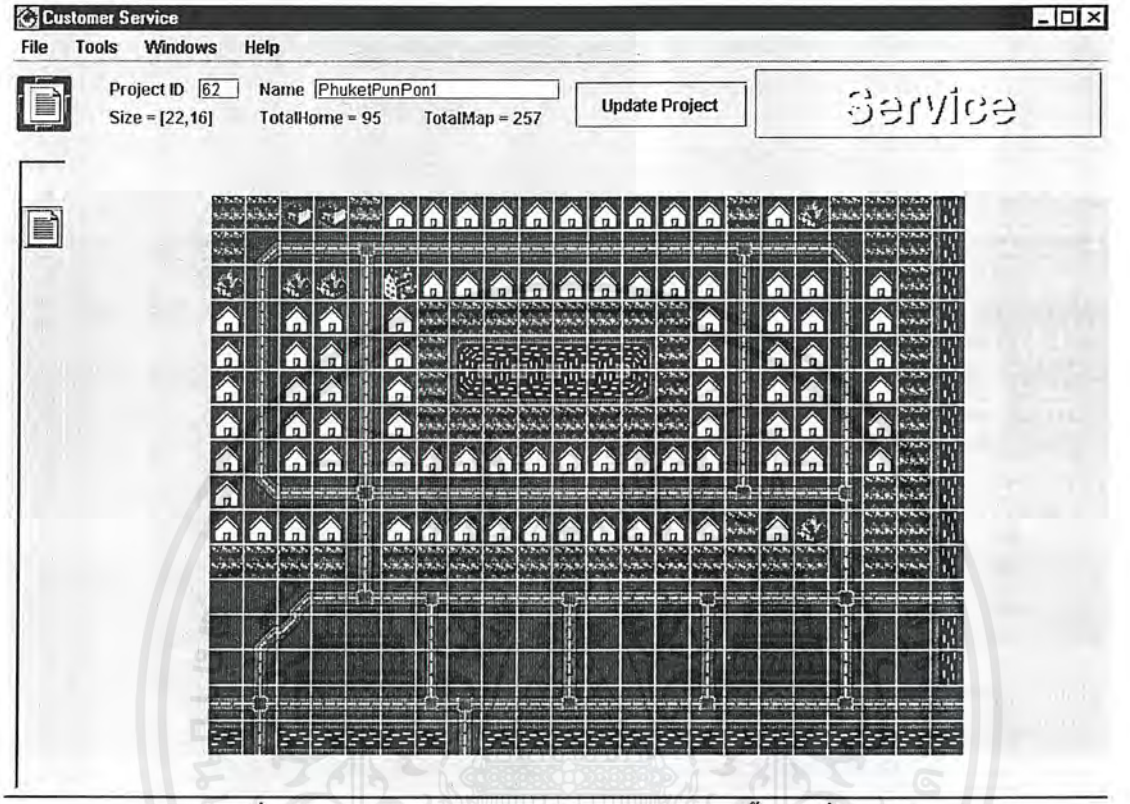
รูปที่ 4.1-2 แสดงหน้าจอที่ใช้ในการแก้ไขแผนผังโครงการ

จากรูปที่ 4.1-2 แสดงหน้าจอที่ใช้ในการแก้ไขแผนผังของโครงการ โดยตัวแผนผังจะประกอบไปด้วยแม่น้ำ ต้นไม้ ถนน ตัวบ้านต่างๆ โดยสามารถที่จะกำหนดขนาดของแผนผังได้ว่าจะให้มีขนาดเท่าไหร โดยสูงสุดที่ทำไว้อยู่ที่. 30 * 18 บล็อกเมื่อทำการวางบ้านลงไปบนโครงการ เราก็จะทำการใส่ข้อมูลต่างๆ ให้กับบ้านแต่ละหลัง โดยเลือกที่ปุ่มที่เป็นรูปเอกสาร แล้วไปคลิกที่ตัวบ้าน ก็จะมีหน้าจอให้เรากรอกข้อมูลรายละเอียดของบ้านหลังนั้นแสดงขึ้นมาดังรูปที่ 4.1-3



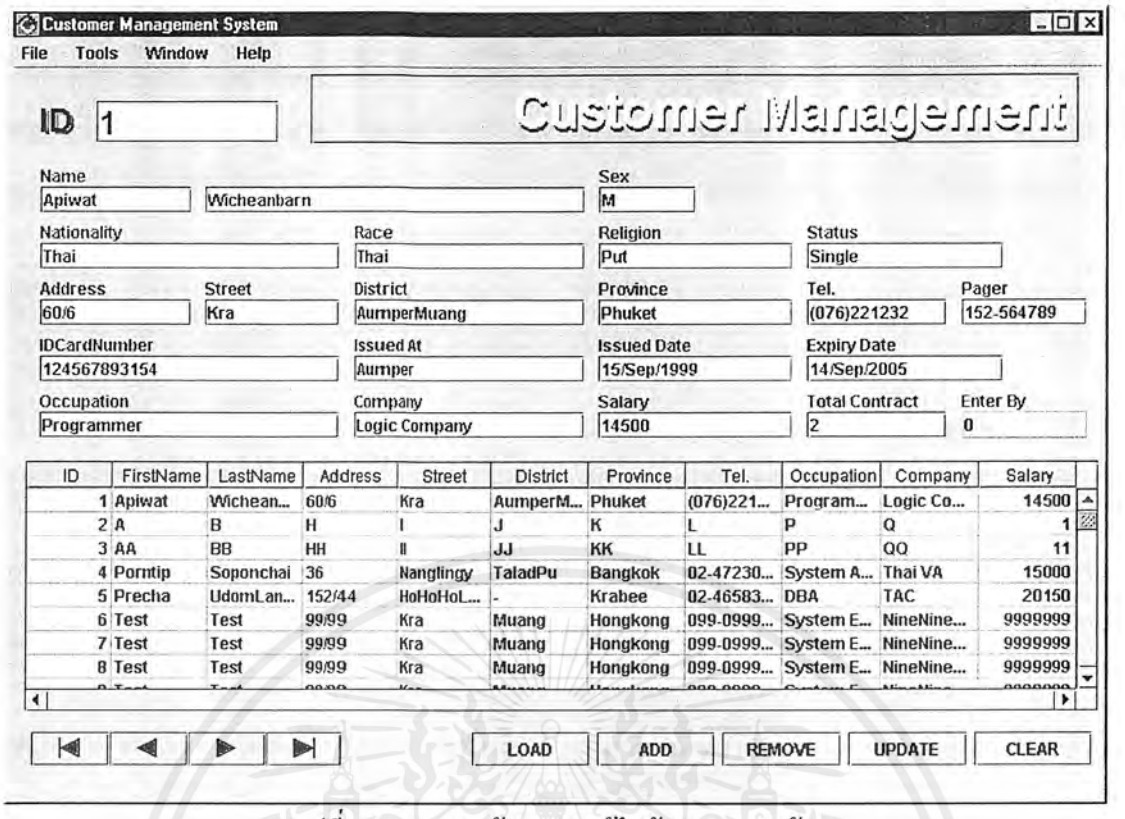
รูปที่ 4.1-3 แสดงหน้าจอที่ใช้ในการเพิ่มเติมข้อมูลรายละเอียดของบ้านแต่ละหลังในโครงการ

โดยจะมีส่วนที่ให้เราใส่ข้อมูล เพิ่มเติมให้กับบ้านก็คือ เลขที่บ้าน (address), สถานะ (status) และชนิด (type) เมื่อเราทำการป้อนข้อมูลดังกล่าวเรียบร้อยแล้วก็ทำการกดปุ่ม "End" หลังจากนั้นเมื่อทำการอัพเดตโครงการหรือ เพิ่มโครงการใหม่เข้าไป ข้อมูลของบ้านแต่ละหลังก็จะถูกเก็บไว้ในฐานข้อมูลในโครงการนั้นด้วย



รูปที่ 4.1-4 แสดงหน้าจอการให้บริการลูกค้า ในการซื้อขายสั่งจองบ้าน

จากรูปที่ 4.1-4 เป็นการแสดงหน้าจอของการให้บริการลูกค้า ในการซื้อขายสั่งจองบ้าน โดยหน้าตาของหน้าจอจะคล้ายกับหน้าจอการออกแบบแผนผัง เมื่อลูกค้าต้องการที่จะรับบริการไม่ว่าจะเป็น การจอง เช่า หรือ ซื้อบ้านจากโครงการใด โครงการหนึ่งแล้ว ก็จะสามารถเลือก ตำแหน่งที่ต้องการได้ พนักงานก็จะทำการ จัดเก็บข้อมูลตรงนี้เอาไว้ ว่า ลูกค้าทำการเลือกบ้านหลังใดไว้ ข้อมูลก็จะถูกส่งกับไปยังเซิร์ฟเวอร์ทันที เมื่อมีลูกค้าคนอื่นที่ต้องการจะรับบริการจากบ้านหลังนั้น ก็ไม่สามารถทำได้ เพราะบ้านได้ถูกลูกค้าคนก่อนเลือกไปแล้ว เพราะข้อมูลทั้งหมดจะถูกอัพเดตลงเซิร์ฟเวอร์อยู่เสมอ



รูปที่ 4.1-5 แสดงหน้าจอการแก้ไขข้อมูลของลูกค้า

จากรูปที่ 4.1-5 เป็นการแสดงหน้าจอการแก้ไขข้อมูลของลูกค้า ที่ประกอบไปด้วย รายละเอียดต่างๆ ดังที่แสดงให้เห็นในรูป

4.2 การทดสอบกลไกของระบบและการจัดเก็บลงฐานข้อมูล

จากการทำงานของโปรแกรมที่ได้แสดงไปในหัวข้อที่แล้ว เราสามารถที่จะทำการทดสอบระบบโดยแบ่งการทดสอบเป็นการทดสอบส่วนต่างๆดังต่อไปนี้

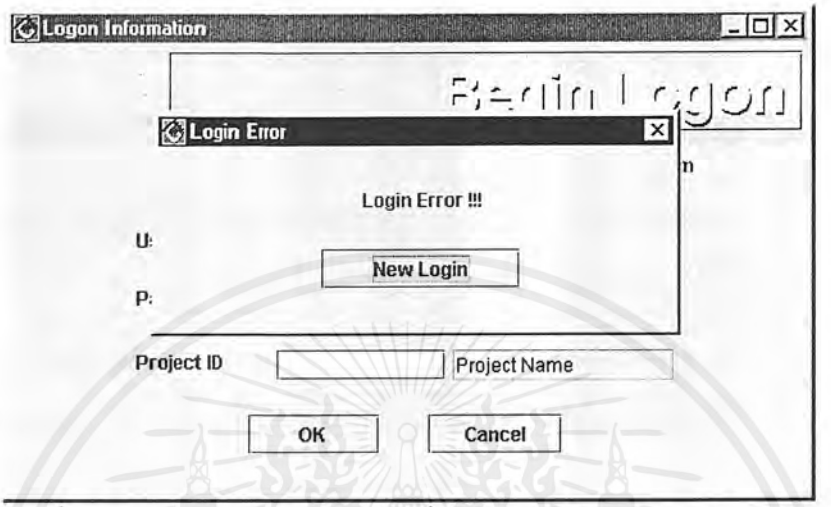
1 ทดสอบการล็อกอินเข้าสู่ระบบ

2 ทดสอบการแก้ไขข้อมูลของลูกค้า

3 ทดสอบการแก้ไขข้อมูลของโครงการ และดึงข้อมูลของโครงการ

4.2.1 ทดสอบการล๊อกอินเข้าสู่ระบบ

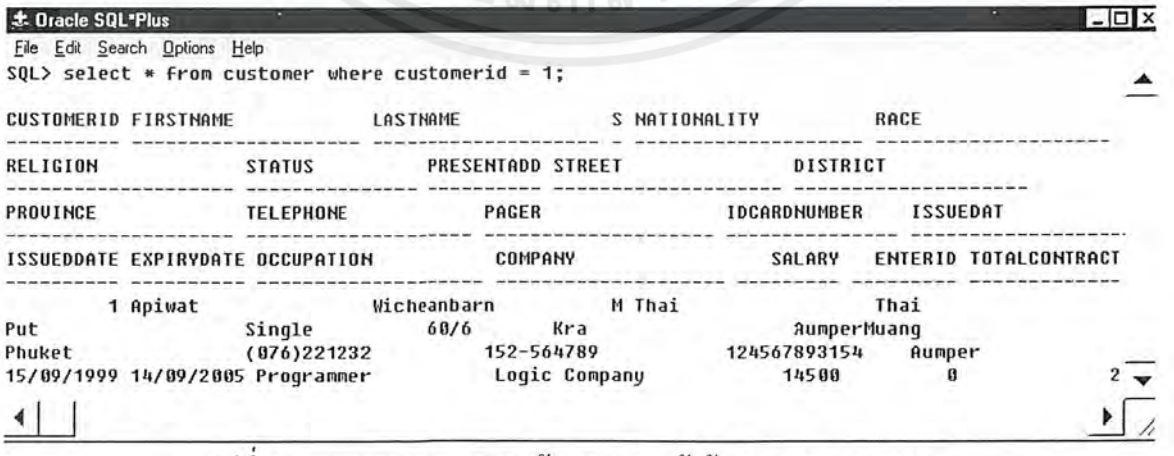
เป็นการทดสอบล๊อกอินของพนักงานว่าโปรแกรมสามารถทำการตรวจสอบได้ถูกต้องหรือไม่ โดยการทดสอบล๊อกอินโดยชื่อผู้ใช้ (User Name) หลายๆคนโดยทดลองล๊อกอินแบบที่ถูกและไม่ถูกต้องกับระบบ ซึ่งระบบก็ยังสามารถทำงานได้ถูกต้อง เมื่อล๊อกอินผิดพลาด ก็จะฟ้องข้อผิดพลาดให้แก่พนักงานว่า "Login Error !!!" แสดงดังรูปที่ 4.2-1



รูปที่ 4.2-1 แสดงหน้าจอฟ้องผิดพลาดเมื่อพนักงานใส่ข้อมูลในการล๊อกอินผิด

4.2.2 ทดสอบการแก้ไขข้อมูลลูกค้า

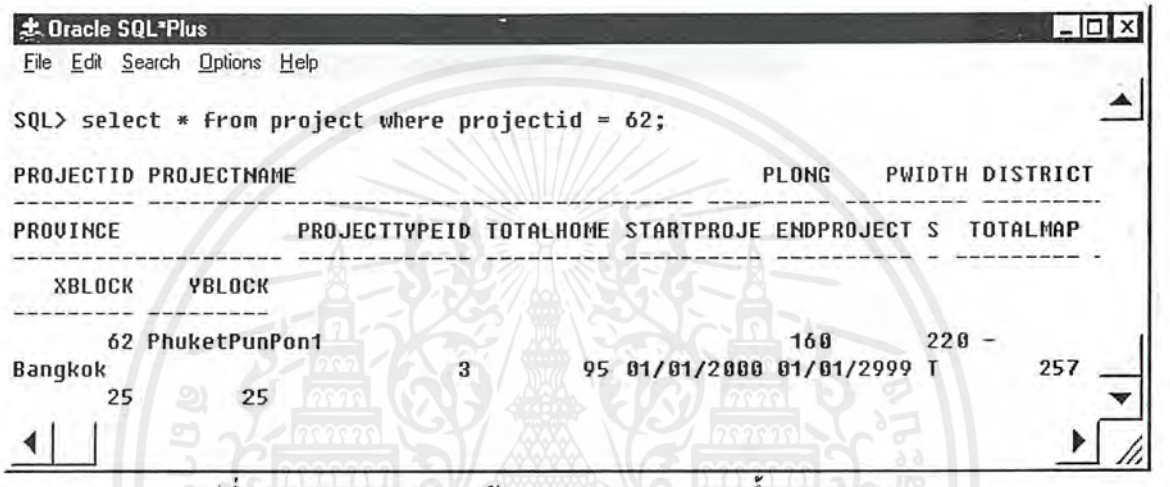
เป็นการทดสอบการแก้ไขข้อมูลส่วนตัวของลูกค้า โดยการทดสอบก็สามารถทำได้โดยการใช้งานจริง โดยทำการป้อนข้อมูลในลักษณะต่างๆกัน แล้วทำการตรวจสอบข้อมูลที่ป้อนเข้าไปนั้น เมื่อทำการโหลดขึ้นมาอีกครั้งหนึ่ง ข้อมูลที่ป้อนลงไปนั้นถูกต้องหรือไม่ และทำการตรวจสอบจากตารางฐานข้อมูลว่ามีการเก็บข้อมูลใน Row นั้นเข้าไปถูกต้องหรือไม่ ด้วย Oracle SQL*Plus ดังรูปที่ 4.2-2 ซึ่งแสดงการ Select ข้อมูลของลูกค้าจากตาราง Customer ว่ามีข้อมูล Customer ของ ID นั้นอยู่ในตารางหรือไม่ เป็นการตรวจสอบเพื่อให้แน่ใจว่าข้อมูลได้ถูกเก็บไว้ถูกต้องแล้ว



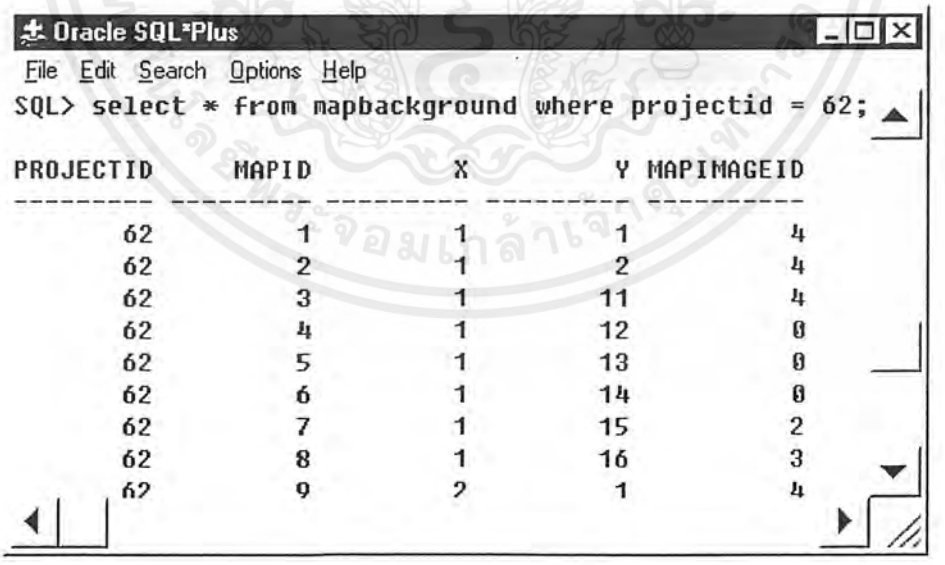
*รูปที่ 4.2-2 แสดงการตรวจสอบข้อมูลของลูกค้าด้วย Oracle SQL*Plus*

4.2.3 ทดสอบการแก้ไขข้อมูลของโครงการ และดึงข้อมูลของโครงการ

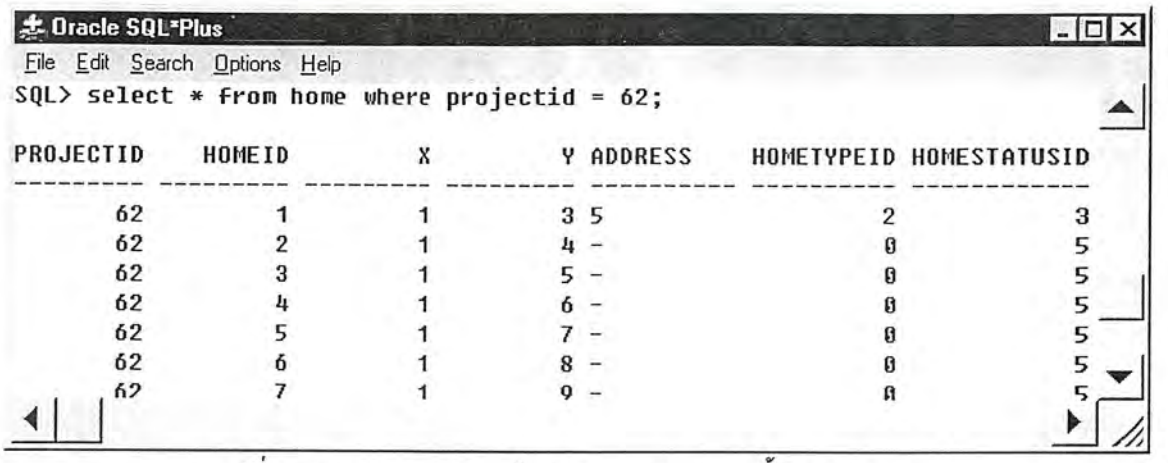
เป็นการทดสอบการแก้ไขหรือว่าสร้างแผนผังของโครงการว่าถูกต้องหรือไม่ ตรวจสอบระบบได้โดยทำการสร้างโครงการ หรือว่าสร้างแผนผังของโครงการขึ้นมา เมื่อสั่งจัดเก็บโครงการดังกล่าวลงฐานข้อมูลแล้ว เราสามารถจะดึงโครงการดังกล่าวขึ้นมาดู และทำการซื้อขายบ้านในโครงการดังกล่าวนั้นได้ถูกต้องหรือไม่ โดยทำการตรวจสอบข้อมูลในตารางอีกด้วยว่า ข้อมูลดังกล่าวตรงตามที่เรากำหนดไว้หรือไม่ ไม่ว่าจะเป็น ข้อมูลของโครงการ ข้อมูลของบ้านในโครงการแต่ละหลัง ข้อมูลของแผนผังโครงการว่าถูกต้องหรือไม่ รวมไปถึงสัญญาที่ลูกค้าทำกับโครงการอีกด้วย แสดงการตรวจสอบโดยใช้ Oracle SQL*Plus ดังรูปที่ 4.2-3



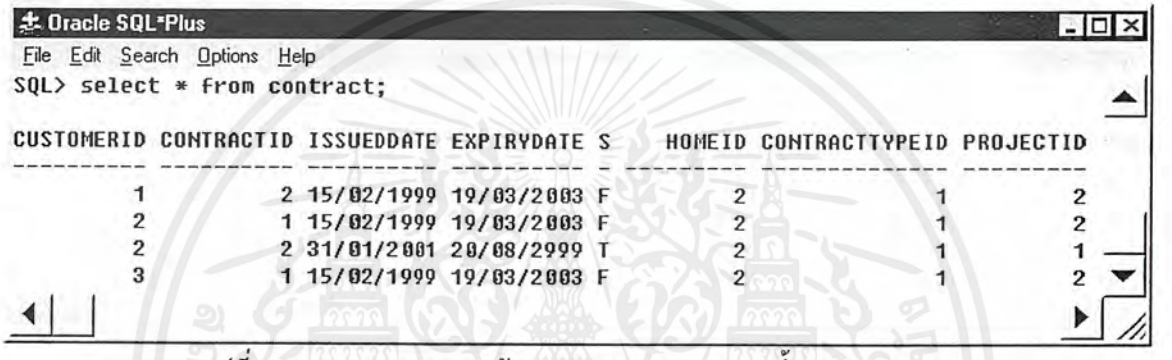
รูปที่ 4.2-3 แสดงการ Select ข้อมูลจากตาราง Project ขึ้นมาตรวจสอบ



รูปที่ 4.2-4 แสดงการ Select ข้อมูลจากตาราง MapBackground ขึ้นมาตรวจสอบ



รูปที่ 4.2-5 แสดงการ Select ข้อมูลจากตาราง Home ขึ้นมาตรวจสอบ



รูปที่ 4.2-6 แสดงการ Select ข้อมูลจากตาราง Contract ขึ้นมาตรวจสอบ

บทที่ 5

การวิจารณ์และสรุป

5.1 สรุป

การพัฒนาโปรแกรมประยุกต์เชิงวัตถุ หรือออบเจ็กต์-โอเรียนเต็ด แอพพลิเคชันนั้น ต้นแบบของแอพพลิเคชันที่เรานำมาพัฒนานั้นคือ ระบบการจัดการฐานข้อมูลโครงการขายบ้านจัดสรร ซึ่งมีการทำงานแบบคอนเคอเรน คอนโทรล (Concurrent Control) มีผู้ใช้ฝั่งไคล์เอ็นท์ที่เป็นพนักงานได้หลายคน พนักงานแต่ละคน สามารถที่จะสั่งซื้อ จอง หรือเช่า บ้านในโครงการต่างๆได้ไปยังเซิร์ฟเวอร์ ข้อมูลจากทุกๆเครื่องของพนักงานแต่ละคนก็จะถูกอัพเดตไปพร้อมๆกัน เพราะเราใช้การจัดเก็บลงฐานข้อมูลตัวเดียวกัน มีการตรวจสอบข้อมูล ไม่ให้เกิดการซื้อขายที่ซ้ำซ้อนกันเกิดขึ้น

ระบบมีความสามารถที่จะเพิ่มโครงการเข้าไปใหม่ได้เมื่อมีโครงการใหม่เกิดขึ้น โดยทั่วไปแล้วระบบสมัยก่อนเมื่อมีโครงการใหม่เกิดขึ้น ก็ต้องมาทำการแก้ไขระบบ เพื่อให้ใช้งานกับโครงการใหม่ได้ แต่สำหรับต้นแบบแอพพลิเคชันตัวนี้ สามารถที่จะรองรับกับโครงการใหม่ๆได้ แม้ขนาดหรือรายละเอียดของโครงการจะแตกต่างกันก็ตาม ระบบที่ออกแบบมาเป็นแบบออบเจ็กต์-โอเรียนเต็ด สามารถที่จะทำการขยายระบบได้ อีกทียังมีความสามารถในการปรับเปลี่ยนระบบได้ง่ายอีกด้วย จึงมีความยืดหยุ่นกว่าระบบแบบเก่า

5.2 บทวิจารณ์

สำหรับการออกแบบและพัฒนาโปรแกรมประยุกต์เชิงวัตถุนั้น ในที่นี้เราได้ใช้เครื่องมือที่ช่วยในการออกแบบแอพพลิเคชันคือ VisualAge for Java ซึ่งเป็นเครื่องมือที่ช่วยเราในการพัฒนาแอพพลิเคชัน ไม่ว่าจะเป็นการจัดระบบ สร้างคลาส เมธอด สร้างหน้าจอยูสเซอร์อินเตอร์เฟซ แต่เราก็จะต้องเข้าใจถึงโครงสร้างภาษา ไม่ว่าจะเป็นด้าน Syntax หรือ Semantic ผู้ที่จะใช้เครื่องมือในการพัฒนาเหล่านี้ ควรที่จะมีความรู้พื้นฐานของตัวภาษานั้นๆเป็นอย่างดี ซึ่งจะทำให้สามารถพัฒนาระบบได้อย่างมีประสิทธิภาพ

สำหรับเครื่องที่ใช้ในพัฒนานั้นเป็นเครื่องเพนเทียม MMX-166 หน่วยความจำ 64 เมก ซึ่งเป็นสเปกของเครื่องที่ต่ำมาก ในการพัฒนาระบบด้วย VisualAge for Java การพัฒนาสามารถทำได้ช้ามาก ไม่ว่าจะเป็นการคอมไพล์โปรแกรมเพื่อนำไปรัน การออกแบบหน้าจอในการติดต่อกับผู้ใช้ การรันโปรแกรม อีกทั้งภาษาจาวาที่มีความเร็วในการทำงานไม่สูงมากนัก จึงทำให้ระบบโดยรวมเมื่อทำออกมาแล้ว มีการทำงานที่ช้ามาก ซึ่งในการพัฒนาควรจะคำนึงถึงจุดเหล่านี้ด้วย ว่าถ้าเราจะต้องนำไปใช้งานจริงแล้ว สิ่งบกพร่องเหล่านี้จะไม่มีผลกระทบต่อผู้ใช้

5.3 ข้อเสนอแนะและแนวทางในการพัฒนาต่อ

จากที่เคยกล่าวมาแล้วว่าระบบนั้นได้ออกแบบขึ้นมาเพื่อเป็นต้นแบบของแอพพลิเคชัน ระบบจึงยังคงมีความสามารถไม่สมบูรณ์พอ ที่จะนำไปใช้งานจริงได้ ไม่ว่าจะเป็นในด้านของความเร็วในการทำงาน ความสมบูรณ์ของระบบ หน้าจอในการติดต่อกับผู้ใช้ ยังคงจะต้องมีพัฒนาเพิ่มเติมในส่วนเหล่านี้อีก ซึ่งสิ่งที่จะต้องพัฒนาเพิ่มเติม เพื่อทำให้ระบบมีความสมบูรณ์ สามารถนำไปใช้งานจริงได้สามารถแสดงออกมาได้ มีดังต่อไปนี้

1) ปรับปรุงการติดต่อกับฐานข้อมูล โดย การลดเวลาในการติดต่อกับฐานข้อมูล, การสร้างชนิดของข้อมูลขึ้นมาใช้งานเอง เป็นต้น
2) ปรับปรุงหน้าจอภาพในการติดต่อกับผู้ใช้
3) ปรับปรุงการออกแบบระบบให้ดีและสมบูรณ์ยิ่งขึ้น

นอกจากการนำระบบไปปรับปรุงให้ดียิ่งขึ้น เพื่อให้สามารถใช้งานได้จริงแล้ว ตัวระบบเองก็ยังสามารถนำไปพัฒนาเป็นระบบอื่นๆ ได้อีก เพราะระบบนั้นได้ถูกออกแบบมาเป็นออบเจ็กต์ เราสามารถนำคุณสมบัติที่ตัวระบบเดิมมีอยู่ ไปพัฒนาหรือแก้ไข เพิ่มเติมคุณสมบัติใหม่เข้าไปเพื่อให้เป็นระบบใหม่ที่เราต้องการได้

บรรณานุกรม

หนังสืออ้างอิง

[1] Bruce Powel Douglass : "Real-Time UML : Developing Efficient Objects for Embedded Systems", Addison-Westleg, 1998

[2] Peter Vander Linden : " Just JAVA and Beyond THIRD EDITION", Sun Microsystem Press A Pretice Hall Title, 1998

[3] Stephen Gilbert and Bill Mccarty : "Object-Oriented Programming in JAVA", The Waite Group, Inc. ®, 1997

[4] Bili Mccarty and Luke Cassady Dorion : "JAVA Distributed Objects", SAMS, 1999

[5] กิตติ ภักดีวัฒนะกุล : "JAVA ฉบับโปรแกรมเมอร์", หจก. ไทยเจริญการพิมพ์, 1999

[6] Marc Carrel-Billiard and John Akerley : "Programming with VisualAge for Java", International Business Machines Corporation, 1998

[7] Olaf Graf, Avril Kotzen, Osamu Takagiwa and Ueli Wahli : "VisualAge for Java Enterprise Version 2:Data Access Beans - Servlets - CICS Connector", International Business Machines Corporation, 1998

เวปไซต์อ้างอิง

[1] http://www.oracle.com

[2] http://www.ibm.com

[3] http://www.javasoft.com