การพัฒนาโปรแกรมเชิงวัตถุโดยใช้ ภาษาจาวา
Object Oriented Software Development with JAVA

โดย

นายฤทัศน์ ภมลพละวัฒน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

การพัฒนาโปรแกรมเชิงวัตถุโดยใช้ ภาษาจาวา

Object Oriented Software Development with JAVA Programming

โดย

นาย สุทัศน์ กมลพละวัฒน์

อาจารย์ที่ปรึกษา

ดร. ชูติมนฑ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมเชิงวัตถุโดยใช้ ภาษาจาวา

Object Oriented Software Development with JAVA Programming

ผู้จัดทำ

1. นาย สุทัศน์ กมลพละวัฒน์ รหัสประจำตัว 40013276

ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา

(ดร. ชุติเมษฎ์ ศรีนิลทา)

การพัฒนาโปรแกรมเชิงวัตถุโดยใช้ ภาษาจาวา

นายสุทัศน์ กมลพละวัฒน์ 40013276
ดร. ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ในปัจจุบันเทคโนโลยีในการพัฒนาซอฟต์แวร์นั้นได้ก้าวหน้าไปมาก โดยจากเดิมนั้นจะเป็นการพัฒนาซอฟต์แวร์ในแบบโพรซิเดอรอล แต่ในปัจจุบันได้มีการนำเอาการพัฒนาแบบออบเจ็ค-โอเรียนเท็ดมาใช้ในการพัฒนาซอฟต์แวร์ ทำให้การพัฒนาซอฟต์แวร์เป็นไปในทางที่มีประสิทธิภาพเพิ่มมากขึ้น ทั้งยังง่ายต่อการพัฒนาอีกด้วย ซึ่งหนึ่งในภาษาที่เป็นภาษาทางด้านออบเจ็ค-โอเรียนเท็ด และเป็นที่นิยมเป็นอย่างมากก็คือภาษาจาวา

ในวิทยานิพนธ์ฉบับนี้เป็นอีกหนึ่งในการนำเอารูปแบบของออบเจ็ค-โอเรียนเท็ดมาใช้ในการพัฒนาซอฟต์แวร์ ซึ่งเป็นซอฟต์แวร์เกมที่สอดแทรกความรู้ในระดับหนึ่งไว้ โดยได้เลือกเอาภาษาจาวามาใช้งาน และด้วยข้อดีของภาษาจาวาในด้านการติดต่อผ่านเครือข่าย จึงพัฒนาให้เป็นซอฟต์แวร์ที่สามารถเล่นผ่านทางอินเตอร์เน็ตได้ และมีการเก็บข้อมูลที่จำเป็นต่อซอฟต์แวร์ไว้ในรูปของฐานข้อมูลอีกด้วย

Object Oriented Development by JAVA Programming

Sutas Kamolpalawat

Dr. Chutimet Srinilta Advicer

Abstract

Now a day, the software development technology is evolving from traditional procedural programming to object-oriented programming, the methodology makes the development of software much efficient than before and the developer can develop their application easier. One of the most favorite object-oriented language is Java.

This paper describes the use of object-oriented technology to the development of sofware. We constitute a game by Java which is provide fully of knowledge to the user who play it. By the benifits of Java in case of easy to communicate via computer network, this means that this game is ready to play via Internet and can store necessary data in built-in database.

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญภาพ VI
บทที่ 1 บทนำ 1
1.1 ความสำคัญและที่มา 1
1.2 ขอบเขตของงานวิจัย 1
1.3 ประโยชน์ที่คาดว่าจะได้รับ 1
1.4 วิธีการดำเนินงาน 2
บทที่ 2 ออบเจ็ค-โอเรียนเท็ด โปรแกรมมิ่ง 3
2.1 แนวความคิดของ Object - Oriented Programming 3
2.2 คุณสมบัติของ Object Oriented 4
2.2.1 Encapsulation/Information hiding 4
2.2.2 Inheritance 4
2.2.3 Polymorphism 4
2.2.4 Abstraction 4
2.3 ความสัมพันธ์ระหว่าง Objects 4
2.3.1 Association 4
2.3.2 Aggregation 5
2.3.3 Generalization 5
2.3.4 Depends On 5
2.4 หลักการพื้นฐานของ Object - Oriented Programming 5
2.5 ข้อแตกต่างของ Object Oriented กับ Procedural Structure Programming 7
2.6 ข้อดีของภาษาทางด้าน Object – Oriented 8
บทที่ 3 ทฤษฎีต่างๆที่ใช้ 9
3.1 UML 9
3.1.1 Use Case Diagram 9
3.1.2 Class Diagram 9
3.1.3 Sequence Diagram 10
3.2 Design Pattern 10

3.3 Network 11

3.3.1 โพรโตคอลคืออะไร 11

3.3.2 ความเป็นมาของ TCP/IP 11

3.3.3 OSI Model 12

3.3.4 โครงสร้างของ TCP/IP 15

3.4 โอเพนดาต้าเบสคอนเน็คติวิตี (โอดีบีซี) 18

บทที่ 4 ความรู้ทั่วไปเกี่ยวกับ จาวา 20

4.1 การทำงานของจาวา 20

4.2 Java's Virtual Machine 20

4.3 Java Applet 21

4.4 เจดีบีซี 21

4.4.1 การทำงานของเจดีบีซี 22

บทที่ 5 การพัฒนาซอร์ฟแวร์ 23

5.1 ขั้นตอนในการพัฒนาซอร์ฟแวร์ทั่วไป 23

5.1.1 Requirement Specification 23

5.1.2 Analysis 23

5.1.3 Design 23

5.1.4 Implementation 24

5.1.5 Testing 24

5.2 เป้าหมายของการพัฒนาซอร์ฟแวร์ 24

5.3 ข้อควรระวังในการพัฒนาซอร์ฟแวร์ 24

5.4 รูปแบบในการพัฒนาซอร์ฟแวร์เชิงวัตถุ 25

5.5 แพทเทิร์นที่สามารถนำมาใช้ในการพัฒนาโปรแกรมเกมที่เขียนขึ้นได้ 26

5.6 ขั้นตอนในการพัฒนาโปรแกรม 29

5.6.1 การทำ Requirement Specification 29

5.6.2 การวิเคราะห์ 31

5.6.3 การออกแบบ 35

5.6.4 การเขียนโปรแกรมและการทดสอบ 38

บทที่ 6 การทำงานของโปรแกรม 39

6.1 การทำงานของโปรแกรมทั้งหมด 39

6.2 สรุปการทำงาน 44

ภาคผนวก 46

ภาคผนวก ก. IIS 46

ภาคผนวก ข. การ Setup ODBC 48

สารบัญภาพ

หน้าที่

รูปที่ 3.1 โครงสร้างของ OSI 7-Layer Model 12
รูปที่ 3.2 การแบ่งกลุ่มของ OSI 7-Layer Model 13
รูปที่ 3.3 แสดง TCP/IP Stack เปรียบเทียบกับมาตรฐาน OSI 16
รูปที่ 3.4 โครงสร้างของโพรโตคอล TCP/IP 18
รูปที่ 3.5 แสดงลักษณะทั่วๆ ไปของโอดีบีซี 19
รูปที่ 4.1 รูปแบบที่ใช้ในเจดีบีซี 22
รูปที่ 5.1 แสดงวงจรชีวิตของออบเจ็ค-โอเรียนเท็ด 26
รูปที่ 5.2 โครงสร้างของ Prototype Pattern 27
รูปที่ 5.3 โครงสร้างของ Singleton Pattern 27
รูปที่ 5.4 โครงสร้างของ Iterator Pattern 28
รูปที่ 5.5 โครงสร้างของ Observer Pattern 29
รูปที่ 5.6 โครงสร้างของ State Pattern 29
รูปที่ 5.7 แสดงยูสเคสในส่วนของการทำงานเบื้องต้นทั้งหมด 30
รูปที่ 5.8 แสดงยูสเคสในส่วนของการติดต่อ เซิร์ฟเวอร์-ไคลเอนต์ 31
รูปที่ 5.9 แสดงยูสเคสในส่วนของการทำงานของผู้เล่นหนึ่งคน 31
รูปที่ 5.10 แสดง sequence diagram ในส่วนของการทำงานโดยรวม 32
รูปที่ 5.11 แสดง collaboration diagram ในส่วนของการทำงานโดยรวมทั้งหมด 33
รูปที่ 5.12 แสดง sequence diagram ในส่วนของการทำงานเมื่อเทียบกับผู้เล่นหนึ่งคน 34
รูปที่ 5.13 แสดง collaboration diagram ในส่วนของการทำงานเมื่อเทียบกับผู้เล่นหนึ่งคน 35
รูปที่ 5.14 แสดง class diagram ของความสัมพันธ์ระหว่าง class ทั้งหมด 36
รูปที่ 6.1 แสดงหน้าจอส่วนของ dialog 39
รูปที่ 6.2 แสดงหน้าหลังจอเชื่อมต่อแล้ว 40
รูปที่ 6.3 แสดงหน้าจอหลังกดปุ่มเริ่มเล่นเกม 41
รูปที่ 6.4 แสดงหน้าจอการตกในบล็อคคำถาม 42
รูปที่ 6.5 แสดงหน้าจอคำถาม 42
รูปที่ 6.6 แสดงหน้าจอลำดับคะแนนสูงสุด 43
รูปที่ 6.7 ชนิดของฐานข้อมูล 44

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากในการพัฒนาซอฟต์แวร์ในแบบโพรซิเดอรอลมักจะเกิดปัญหาขึ้นในหลายๆขั้นตอนของการพัฒนา ซึ่งทำให้ผู้พัฒนาต้องเสียค่าใช้จ่ายในการพัฒนาและค่าบำรุงรักษาระบบค่อนข้างสูงมากเมื่อเทียบกับผลตอบแทนที่ได้ เนื่องจากสามารถใช้ได้กับเพียงเฉพาะงานนั้นๆเท่านั้น (ไม่สามารถนำกลับมาใช้ใหม่ได้อีก) นอกจากนี้เมื่อ user ต้องการเปลี่ยนความต้องการเกี่ยวกับการทำงานของระบบที่ทำเสร็จแล้วก็ยังเป็น การยุ่งยากเพราะต้องไปเริ่มต้นทำใหม่หรือต้องทำการศึกษาระบบดังกล่าวทั้งหมดเพียงเพื่อทำการปรับปรุงแก้ไขบางส่วนเท่านั้น จึงมีผู้เชี่ยวชาญหลายท่านได้ทำการคิดค้นหาแนวทางที่จะแก้ไขปัญหาเหล่านี้ และพบว่า Object Technology เป็นแนวทางหนึ่งในการพัฒนาซอฟต์แวร์ที่มีประสิทธิภาพและในขณะนี้ได้เป็นที่นิยมและยอมรับในหมู่ผู้พัฒนาระบบในหลายๆองค์กร

ในปัจจุบัน โปรแกรมพัฒนาการศึกษาของเด็กได้มีออกมาอย่างมากมาย ทั้งในรูปแบบของเกมและโปรแกรมการศึกษา ผู้จัดทำโปรเจคนี้ได้มีความสนใจที่จะพัฒนาเกมเพื่อการพัฒนาความรู้สำหรับเด็กโดยมีการนำขั้นตอนในการพัฒนาโปรแกรมแบบ Object Oriented มาใช้ในการพัฒนา รวมถึงรูปแบบการพัฒนา การวางเป้าหมาย การแก้ปัญหาโดยการใช้แพทเทิร์นต่างๆ การใช้ Tools ในการช่วยเหลือในการออกแบบ

โดยโปรแกรมที่จะทำการพัฒนาขึ้นนี้อาจจะจัดอยู่ในประเภทของเกมออนไลน์ หรือเรียกได้อีกอย่างว่า เกมบนอินเตอร์เน็ท (internet game) ซึ่งในปัจจุบันนั้นได้มีออกมาอย่างแพร่หลาย โดยจะมีทั้งในแบบที่ต้องเล่นผ่านบราวเซอร์ และแบบที่เล่นโดยตรงจากเกมเลย โดยรูปแบบเกมนั้นจะมีทั้งเกมที่เป็นเกมที่ให้ความบันเทิงอย่างเดียว และแบบที่พัฒนาหรือเสริมความรู้และทักษะแก่ผู้เล่น

1.2 ขอบเขตของการศึกษา

ในปริญญานิพนธ์ฉบับนี้จะมุ่งเน้นที่จะศึกษาถึงคุณสมบัติ หลักการทำงาน การเขียนโปรแกรม และ แนวทางของการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ ของภาษาทางด้านออบเจ็ค โอเรียนเท็ด โดยในที่นี้จะเป็นการใช้โปรแกรมภาษาจาวาในการเขียนโปรแกรม และมีการใช้โปรแกรมจำพวก Tools ต่างๆทั้งในด้านการออกแบบ และ เขียนโปรแกรม เข้ามาช่วยเหลือในการพัฒนานี้

โดยโปรแกรมประยุกต์ที่จะทำการพัฒนาขึ้นนี้ จะอยู่ในรูปของเกมที่ใช้พัฒนาความรู้และทักษะสำหรับเด็ก และมีระบบรองรับการเล่นแบบผู้เล่นหลายคนโดยเชื่อมต่อกันทางอินเตอร์เน็ท ซึ่งจะต้องเล่นผ่านโปรแกรมบราวเซอร์ เช่น โปรแกรม Internet Explorer เป็นต้น

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจถึงทฤษฎีต่างๆเกี่ยวกับ Object Oriented
2. ได้รับความรู้และประสบการณ์ในการเขียนโปรแกรมด้วยภาษาจาวา

3. ทราบถึงวิธีการเขียนโปรแกรมติดต่อระหว่างเซิร์ฟเวอร์ กับ ไคลเอนต์
4. ได้ทราบถึงปัญหาที่เกิดขึ้นในการปฏิบัติงานจริง และแนวทางในการแก้ไขปัญหานั้นๆ

1.4 วิธีการดำเนินงาน

1. ทำการศึกษาถึงทฤษฎี และแนวความคิดของการเขียนโปรแกรมในเชิงวัตถุ
2. ทำการศึกษา และทำความเข้าใจในภาษาจาวา
3. ศึกษา และ ทดลองใช้ tools ต่างๆทั้งในการออกแบบ และ การเขียนโปรแกรม
4. กำหนดหัวข้อของโปรแกรมประยุกต์
5. กำหนดขอบเขตของการทำงานของโปรแกรมประยุกต์
6. ทำการวิเคราะห์และออกแบบโปรแกรม ทั้งในด้านการทำงานและการออกแบบหน้าจอ
7. ทำการเขียนโปรแกรมในส่วนต่างๆ
8. ทดลองใช้ เพื่อหาข้อบกพร่อง และ ทำการแก้ไขข้อบกพร่องนั้นๆ
9. รวบรวมข้อมูลและเอกสารต่างๆ เพื่อเรียบเรียงเป็นปริญญานิพนธ์

บทที่ 2
ออบเจ็ค-โอเรียนเท็ด โปรแกรมมิ่ง
(Object-Oriented Programming)

2.1 แนวความคิดของ Object - Oriented Programming

ในอดีตที่ผ่านมาการเขียนโปรแกรมจะเป็นแบบสั่งให้คอมพิวเตอร์ทำงานเป็นลำดับของคำสั่ง (instruction) ซึ่งในสมัยนั้นเป็นที่นิยมอย่างมาก เนื่องจากเขียนโปรแกรมได้ง่ายและรวดเร็ว และใช้วิธีการเก็บรวบรวมข้อมูลต่างๆ เป็นแฟ้มข้อมูล (File) หรือ ฐานข้อมูล (Database) แยกกันกับโปรแกรมในระบบงาน ถ้าโปรแกรมมีขนาดใหญ่จำเป็นต้องมีการแบ่งงานให้กับโปรแกรมเมอร์หลายคนทำการเขียนโปรแกรม และจะต้องมีสร้าง function ที่ซ้ำกันเกิดขึ้น ทำให้ขาดประสิทธิภาพ และเมื่อได้ทำการใช้งานไปนานๆ จะพบว่าการเขียนโปรแกรมในลักษณะนี้มีข้อจำกัดอยู่มาก เช่น ในเรื่องของการนำโปรแกรมมาพัฒนาให้ทันสมัยนั้นทำได้ยาก เพราะต้องทำการศึกษาโปรแกรมเก่าให้เข้าใจอย่างแท้จริงก่อนลงมือแก้ไข และไม่มีความปลอดภัยของข้อมูล เพราะทุกโพรเซส (Process) สามารถนำข้อมูลนั้นไปใช้งานได้ตามต้องการ

ทำให้เกิดแนวความคิดการเขียนโปรแกรมเชิงวัตถุ (Object - Oriented Programming) ขึ้นมา โดยเป็นวิธีใหม่ในการพัฒนาซอฟต์แวร์ (Software) เพื่อเพิ่มประสิทธิภาพในการผลิตซอฟต์แวร์ วิธีนี้ทำให้การออกแบบและการเขียนซอฟต์แวร์มีความใกล้ชิดกันมากขึ้น และจะกำหนดสิ่งต่างๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจ็ค (Object) ซึ่งในออบเจ็คจะประกอบด้วยข้อมูล (Data) และเมธธอด (Method) และการเข้าถึงข้อมูลนั้นได้จะต้องผ่านเมธธอดที่กำหนดไว้ (Encapsulation) เมธธอดเปรียบเทียบได้ทำนองเดียวกันกับโพรซีเยอร์ (Procedure) ในโปรแกรมระบบงานแบบเดิมนั่นเอง การเรียกใช้เมธธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธธอดทำได้โดยการส่งเมสเซจ (Message) ไปยังออบเจ็คนั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงหน้าที่การทำงาน (Function) ใดๆ ของออบเจ็คนั้น จะไม่มีผลกระทบกับเมสเซจที่ใช้ติดต่อมายังออบเจ็คนั้น ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธธอดสามารถใช้อธิบายถึงพฤติการณ์ของออบเจ็คแต่ละออบเจ็คได้ ออบเจ็คที่มีพฤติการณ์แบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่า คลาส (Class) ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ (Inheritance) จากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณสมบัติว่า ซุปเปอร์คลาส (Superclass) และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่า ซับคลาส (Subclass) ซึ่งทำให้การบำรุงรักษาระบบงานทำได้ง่ายและสามารถขจัดปัญหาต่างๆ ในการเขียนโปรแกรมแบบโพรซีเดอรอลได้อีกด้วย โดย Object - Oriented Language โปรแกรมแรกซึ่งออกมาในปี ค.ศ. 1967 คือ โปรแกรม Simula ภายในโปรแกรมมีการกำหนด Object , Class , inheritance และ อื่นๆ ต่อมาได้มีโปรแกรม SmallTalk ซึ่งเริ่มมีการใช้ graphic เข้ามาช่วยในโปรแกรมมากขึ้น จนถึงปัจจุบันได้มีโปรแกรมที่เข้ามาสนับสนุนภาษาทางด้าน Object Oriented มากขึ้น

2.2 คุณสมบัติของ Object Oriented

2.2.1 Encapsulation/Information hiding

ทฤษฎีนี้ถูกคิดค้นโดย James Rambaugh ซึ่งหมายถึง การแยกลักษณะภายนอก (Interface) ซึ่งสามารถติดต่อกับ Object อื่นๆ ออกจากส่วนที่ implement ภายในของ Object ซึ่งจะถูกใช้ได้เฉพาะตัว Object นั้นๆ มีข้อดีคือ เพิ่มความสามารถในการปกป้องข้อมูลโดยไม่ให้ Object อื่นเข้ามาทำการเปลี่ยนแปลงข้อมูลก่อนได้รับอนุญาต ทั้งนี้ตัว Object ควรรู้และทำการเปลี่ยนค่าด้วยตนเองซึ่งเป็นผลให้ maintain object ง่ายขึ้น

2.2.2 Inheritance

คือหลักการในการที่ Object ทุกๆตัวใน generalized collection ได้มีการใช้ข้อมูล และพฤติกรรม(behavior) ร่วมกันซึ่งจะมีความสัมพันธ์แบบ is-a relationship โดยเรียก class ที่อยู่เหนือกว่าว่า super class ซึ่งจะถ่ายทอดคุณสมบัติทั้ง Attribute และ method มายัง class ที่ต่ำกว่าเรียกว่า subclass ซึ่งจะมี Attribute และ method เพิ่มเติมจาก super class มีข้อดีคือ

- เพิ่ม consistency เพียงแค่เปลี่ยน Method หรือ Attribute ที่ Super class ซึ่งเป็นผลให้ค่าที่ Subclass เปลี่ยนไปด้วย
- เป็นการส่งเสริมการนำ Object กลับมาใช้ใหม่

2.2.3 Polymorphism

อาจหมายถึง "having many forms" โดยการส่ง message เดียวกันให้กับ Object ที่ต่างกันเป็นผลทำให้ Object แสดงพฤติกรรมที่แตกต่างกัน มีข้อดีคือ เป็นการสนับสนุนการนำโปรแกรมกลับมาใช้ใหม่และสามารถเปลี่ยนแปลงซอฟต์แวร์ได้ในระหว่างการพัฒนา

2.2.4 Abstraction

การแยกแยะเอกลักษณ์เป็นวิธีการหนึ่งที่มนุษย์ใช้ในการมองสิ่งใดๆเพื่อลดความซับซ้อนของสิ่งนั้นๆ โดยแยกเอาเฉพาะสิ่งที่ผู้มองสนใจออกมา ซึ่งแน่นอนว่าถ้าผู้มองระบบเป็นคนละคนกันก็จะแยกแยะออกมาได้ต่างกัน การแยกแยะเอกลักษณ์เป็นการแสดงคุณลักษณะที่สำคัญของออบเจ็กต์ ซึ่งเป็นคุณลักษณะที่ทำให้ออบเจ็กต์นั้นๆแตกต่างจากออบเจ็กต์อื่นๆ เช่น ออบเจ็กต์รถยนต์ ถ้าเป็นการแยกแยะเอกลักษณ์ของนักเดินทางก็จะมองว่ารถยนต์นั้นต้องพาผู้โดยสารไปยังปลายทางได้สะดวกรวดเร็ว แต่ถ้าเป็นการแยกแยะเอกลักษณ์ของช่างเครื่องก็จะมองว่ารถยนต์ประกอบด้วยเครื่องยนต์ต่างๆมากมายที่ทำงานร่วมกัน

2.3 ความสัมพันธ์ระหว่าง Objects

2.3.1 Association

เป็นการแสดงให้เห็นถึงความสัมพันธ์ระหว่างโครงสร้างของ Object Oriented ว่ามีความสัมพันธ์กันหรือไม่ โดยจะมีการแสดงการเชื่อมต่อระหว่าง Object Oriented ที่มีความสัมพันธ์แบบถาวร โดยมีโครงสร้าง ดังนี้

- constraint คือการแสดงข้อมูลที่เกี่ยวกับความสัมพันธ์ระหว่าง Objects

- Link คือ การเชื่อมต่อระหว่าง Objects ที่มีความสัมพันธ์แบบ ถาวร ซึ่งเป็น instance ของ Association

2.3.2 Aggregation

ในบางครั้งเราอาจถือได้ว่า Aggregation เป็นความสัมพันธ์แบบพิเศษของ Association โดยจะมีความสัมพันธ์ในแบบ Whole/Part หรือแบบ a-part-of หรืออาจเรียกอีกแบบหนึ่งว่ามีความสัมพันธ์แบบ has-a-relationship คือ class A จำเป็นต้องมีการเรียกใช้ class B ในการทำงาน ดังนั้น class A จะขาด class B ไม่ได้แต่ class B สามารถขาด class A ได้จึงเรียกได้ว่า class B มีความสัมพันธ์เป็นแบบ Aggregation กับ class A

2.3.3 Generalization

เป็นความสัมพันธ์ในการจัดกลุ่มสิ่งที่เหมือนกันระหว่าง Object หรือ Class ขึ้นมาเป็นอีก Object หนึ่งโดยพิจารณาจากข้อมูล (structure) หรือ พฤติกรรม (behavior) ในกลุ่มของ Object นั้น จะมีลักษณะเป็นลักษณะทั่วไปไม่ชี้เฉพาะเจาะจง

2.3.4 Depends On

เป็นความสัมพันธ์ที่ชี้ให้เห็นถึง source หรือ client ขึ้นอยู่กับ destination หรือ supplier เช่นใน Class หนึ่งสามารถมี Method ที่ขึ้นอยู่กับ Class อื่น เวลาทำงานจะมีการอ้างถึง Class นั้นด้วย

2.4 หลักการพื้นฐานของ Object - Oriented Programming

- **Object** : Object เป็นการเปรียบเสมือนกล่องๆ หนึ่ง ซึ่งข้างในกล่องมี Data (instance variable) และ Method รวมกันอยู่ภายใน และ Object นี้จะทำการรับและการส่ง message ระหว่าง Object เพื่อทำการใช้งาน Data และ Method ที่อยู่ภายใน โดยการรับและการส่ง message นี้จึงเปรียบเสมือน interface ของ Object นั้นๆ ด้วย โดย Object นี้ถือ ได้ว่าเป็นกุญแจสำคัญในการทำความเข้าใจการเขียนโปรแกรมแบบ Object Oriented
- **Class** : คลาสเป็นการจัดกลุ่มของ Object ตามคุณสมบัติที่เหมือนกัน นำมารวมกันเป็นคลาส และผู้ใช้สามารถติดต่อกับคลาสได้โดยผ่านทางเมธธอดภายในคลาส หรือเมธธอดในซุปเปอร์คลาส เท่านั้น (superclass คือ super class) โดยปกติคลาสจะถูกออกแบบให้มีความสัมพันธ์กับคลาสอื่นๆ ซึ่งคลาสหนึ่งคลาสสามารถเป็นต้นแบบให้แก่คลาสอื่นๆ ได้หลายคลาส (Inheritance)
- **Method** : เป็นการทำงาน (Operation)หรือเป็นวิธีการกระทำที่สามารถทำงานกับออบเจ็คได้ แต่ละคลาสจะมีเมธธอดของตัวเอง โดยการทำงาน จะเริ่มจากการส่ง message ไปยัง Object ที่ต้องการ (Receiver Object) เพื่อเรียกใช้เมธธอดที่อยู่ภายใน Object นั้น หรือรอรับ message ที่ส่งมาจาก Object อื่นๆ เพื่อมาเรียกใช้เมธธอดของตัวเอง
- **Message** : เป็นตัวกลางในการติดต่อสื่อสารระหว่างแต่ละออบเจ็ค โดยข้อมูลที่จะถูกส่งไปกับ message ด้วยคือ
  - ออบเจ็คที่ message ที่ระบุ
  - ชื่อของเมธธอดที่จะทำการติดต่อ

- พารามิเตอร์ต่างๆที่เมธธอดต้องการ

- **Encapsulation** : คือ การรวมกันของโครงสร้างข้อมูล (Data Structure) กับฟังก์ชั่น (Method , Action) เกิดเป็นวัตถุใหม่ที่มีความสามารถในการซ่อนข้อมูลจากระบบภายนอกได้ (มีลักษณะเหมือนเป็นยาเม็ดแคปซูล ซึ่งจะมองไม่เห็นตัวยาภายในรู้แต่เพียงว่ายาเม็ดนี้ใช้รักษาโรคอะไรเท่านั้น) ทำให้ข้อมูลมีความมั่นคงขึ้น ซึ่งวิธีในการป้องกันนั้นจะอยู่ที่ชนิดของการประกาศข้อมูลภายในคลาส ซึ่งจะมีอยู่ 3 ระดับ คือ
  - การประกาศข้อมูลแบบที่ใช้ภายในตัวเอง จะมีผลทำให้ไม่มีคำสั่งใดที่จะอ้างถึงข้อมูลภายในคลาสนั้นได้เลย นอกจากเมธธอด ภายในคลาสนั้นๆเอง ซึ่งตามปกติแล้วจะมีอย่างน้อยหนึ่งเมธธอดที่ไม่ได้ประกาศให้เป็นข้อมูลชนิดนี้ เพื่อที่จะได้ใช้ในการเข้าถึงข้อมูลภายในคลาสได้
  - การประกาศข้อมูลแบบสาธารณะ คือจะมีผลทำให้คลาสอื่นๆ หรือเมธธอดอื่นๆ สามารถอ้างอิงข้อมูลภายในคลาสนี้ได้อย่างอิสระ
  - การประกาศข้อมูลแบบสงวนไว้ให้คลาสลูก จะมีผลทำให้มีเฉพาะเมธธอดภายในของคลาสนั้นและคลาสที่สืบทอด (Inheritance) เท่านั้นที่สามารถอ้างอิงข้อมูลภายในคลาสได้

- **Inheritance** : คือ การสร้างคลาสใหม่ขึ้นมา โดยมีการสืบทอดคุณสมบัติพื้นฐานที่มาจากคลาสเดิม แต่จะมี ข้อมูล หรือ เมธธอด เพิ่มขึ้นมาจากคลาสเดิมด้วยก็สามารถทำได้ ถ้าคลาส B เป็นซับคลาสของคลาส A ออบเจ็คที่เป็นอินสแตนซ์ของคลาส B จะมีคุณสมบัติพิเศษกว่าอินสแตนซ์ของคลาส A แต่อย่างน้อยที่สุด จะต้องมีคุณสมบัติเหมือนอินสแตนซ์ของคลาส A การถ่ายทอดคุณสมบัติ นี้จะรวมถึง data และ method ของคลาส A และคลาส A จะถูกเรียกว่าเป็น ซุปเปอร์คลาสของคลาส B การสืบทอดแบบนี้นับได้ว่าเป็นคุณสมบัติเด่นของการเขียนโปรแกรมแบบ Object Oriented เพราะทำให้เราสามารถนำโค้ดเดิมมาใช้ได้อีกโดยไม่ต้องมีการเขียนโปรแกรมที่เขียนไว้แล้วขึ้นมาใหม่อีก

- **Polymorphism** : คือ การที่เราส่งเมสเซจที่เหมือนกัน ไปในออบเจ็คที่ต่างกัน แต่ละออบเจ็คจะตอบสนองออกมาไม่เหมือนกัน ตามแต่ชนิดและหน้าที่ของออบเจ็ค ความสามารถที่ใช้เมสเซจเหมือนกันสำหรับการกระทำที่เหมือนกัน ไปยังออบเจ็คต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหาหนึ่ง ๆ

  ตัวอย่าง การ polymorphism
  - AB + CD ได้ผลลัพธ์ABCD(concat)
  - 5 + 3ได้ผลลัพธ์ 8 (add)

  ทั้ง 2 แบบ เป็นการส่งเมสเซจ '+' เข้าไปยังออบเจ็ค ภายในคลาส String และ Integer ตามลำดับ

- **Dynamic Binding** : คือ การนำโปรแกรมย่อยๆ มาประกอบให้ใช้งานได้ในขณะ run time โดยในขณะ Compile time นั้นจะเก็บโปรแกรมในรูปแบบของ Class ต่างๆ ไว้ เพื่อไม่ให้เกิดความยุ่งยาก ซับซ้อน ต่อจากนั้น เมื่อนำมาใช้ในขณะ run time เมื่อมีการเรียกใช้ Class นั้นๆ จะทำการนำ Class นั้นมาไว้ในส่วนของโปรแกรม และเมื่อใช้งานเสร็จแล้วจะถูกลบออกจากหน่วยความจำ

- **Override** : คือ การที่เราสร้าง Subclass ขึ้นมาใหม่ และมีการสร้าง Method ที่ซ้ำกับ Method ที่เป็นของ Superclass
- **Overload** : คือ การที่เราสร้าง Method ชื่อเหมือนกัน แต่รับ Parameter ต่างกัน ภายใน Class เดียวกัน

2.5 ข้อแตกต่างของ Object Oriented Programming กับ Procedural Structure Programming

- วิธีการจัดเก็บข้อมูล

  OOP : เมื่อออบเจ็คหนึ่งต้องการทำงานกับเม็ทธอดของออบเจ็คอื่นๆ การทำงานกับเม็ทธอดของออบเจ็คนั้นจะส่งข้อความ (message) ไปที่ออบเจ็คที่ต้องการ เมื่อออบเจ็คนั้นได้รับข้อความก็จะตอบสนองต่อข้อความนั้น ตามแต่จะกำหนดไว้

  Procedural : ก่อนที่โพรซีเยอร์จะทำการกับข้อมูลนั้นๆ จะมีการส่งค่าพารามิเตอร์ไปที่โพรซีเยอร์นั้นก่อน โดยจะต้องกำหนดโพรซีเยอร์ที่ต้องการใช้งานเลย ซึ่งตัวโปรแกรมจะไม่สามารถเลือกทำงานได้เอง

- โครงสร้างโปรแกรม

  OOP : มีการรวมข้อมูลเข้ากับโพรซีเยอร์และฟังก์ชันเพื่อให้เป็นข้อมูลชนิดออบเจ็ค ซึ่งจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างข้อมูลภายในและเม็ทธอดของออบเจ็คด้วย โดยออบเจ็คหนึ่งจะมีคุณสมบัติเหมือนสิ่งของอย่างหนึ่ง คือ มีทั้งส่วยประกอบและการทำงานซึ่งสามารถใช้งานได้ทันที

  Procedural : จะมีอยู่ในรูปของไลบรารี ซอฟต์แวร์คอมโพเนนท์ (Library Software Component) ของแต่ละคอมโพเนนท์จะมีมากกว่าหนึ่งโครงสร้างของข้อมูลอยู่ด้วยกัน

- การสืบทอด

  OOP : สามารถรองรับการสืบทอด และ ความหลากหลายได้

  Procedural : ไม่สามารถรองรับการสืบทอด และ ความหลากหลายได้

- การกำหนดโครงสร้างใหม่

  OOP : อนุญาตให้ผู้ใช้สามารถกำหนดโครงสร้างขึ้นใหม่ได้ โดยที่โครงสร้างที่สร้างขึ้นมาใหม่จะมีสถานะเทียบเท่ากับที่ระบบเป็นผู้กำหนด

  Procedural : ถึงแม้ว่าจะยอมให้ผู้ใช้กำหนดโครงสร้างขึ้นมาใหม่ได้ แต่โครงสร้างที่สร้างขึ้นมาใหม่จะมีความสำคัญและประสิทธิภาพน้อยกว่าโครงสร้างที่สร้างจากระบบ

- ลำดับชั้นของคลาส

  OOP : สามารถรองรับลดับชั้นของคลาสได้ ดังนั้นจึงมีประโยชน์ในกรณีที่ต้องการเพิ่มความสามารถให้กับลำดับชั้นของคลาสหรือเพิ่มคลาสย่อย

  Procedural : แต่ละคอมโพเนนท์ของแต่ละซอฟต์แวร์จะมีสถานะเท่ากันหมด จึงทำให้เกิดความซ้ำซ้อนได้

- การส่งผ่านพารามิเตอร์

OOP : การส่งผ่านพารามิเตอร์เข้า-ออก จะเป็นการส่งผ่านจากออบเจ็คหนึ่งไปยังอีกออบเจ็คหนึ่ง
Procedural : การส่งผ่านพารามิเตอร์จะส่งผ่านชนิดของข้อมูลเท่านั้น

2.6 ข้อดีของภาษาทางด้าน Object – Oriented

- **Security** : ในภาษาทาง Object Oriented มีความสามารถในการทำ (Encapsulation / Information hiding) เนื่องจาก Object Oriented จะเก็บข้อมูลไว้ในออบเจ็คซึ่งจะถูกจัดการโดยเมธธอดที่เหมาะสมเท่านั้น ดังนั้นจึงทำให้ข้อมูลมีความปลอดภัยสูงกว่าภาษาแบบ procedural
- **Reusability** : Object Oriented สามารถนำคอมโพเนนท์ที่สร้างไว้แล้วกลับมาใช้งานได้อีก โดยไม่จำเป็นต้องทำการเขียนใหม่เลย
- **Portability** : ระบบที่ถูกออกแบบมาให้ใช้ได้บนสภาพแวดล้อมหนึ่งๆ สามารถที่จะปรับปรุงแก้ไขเพียงเล็กน้อยหรืออาจไม่ต้องปรับปรุงเลย ในการนำไปใช้กับอีกสภาพแวดล้อมอื่นๆ
- **Integrity** : เนื่องจาก Object Oriented มีคุณสมบัติในการทำ Encapsulation จึงทำให้เกิดความผิดพลาดของข้อมูลได้ยาก ทำให้มีความน่าเชื่อถือสูง
- **Flexibility** : ด้วยคุณสมบัติในด้านการสืบทอดความสัมพันธ์ (Inheritance) ของ Object Oriented จึงทำให้มีความยืดหยุ่นในการแก้ไขได้ง่าย คือ สามารถเพิ่มหรือลดความสามารถบางส่วนให้แต่ละคลาสย่อยได้ โดยอาจเพิ่ม หรือลดที่คลาสแม่ (Superclass) เท่านั้น ในทุกคลาสย่อยก็จะถูกเพิ่มเติมหรือ ลดความสามารถนั้นๆออกไปด้วย
- **Understandability** : ด้วยคุณลักษณะ Abstract ของ Object Oriented ทำให้ภาพรวมของตัวโปรแกรมตรงกับความเข้าใจของผู้ใช้ จึงทำให้ผู้ใช้สามารถเข้าใจการทำงานของระบบได้โดยง่าย
- **Maintainability** : เนื่องด้วยความสามารถทางด้านการสืบทอดความสัมพันธ์ (Inheritance) ของ Object Oriented จึงทำให้ง่ายต่อการปรับปรุง เมื่อมีความต้องการเพิ่มเติม specification ของระบบและด้วยข้อดีในด้าน Flexibility จึงทำให้เป็นการง่ายต่อการบำรุงรักษา
- **Easy to Development** : ในการพัฒนาทางด้าน Object Oriented นั้นเราไม่จำเป็นที่จะต้องสนใจโค้ดของโปรแกรมมากนัก เราเพียงแค่ทราบรูปแบบในการสื่อสาร (message) ของแต่ละออบเจ็ค และเอาท์พุท ที่ได้ในแต่ละแบบของการสื่อสารนั้นๆ ก็เพียงพอที่จะทำให้เราสามารถพัฒนาโปรแกรมในแบบขนานได้แล้ว

บทที่ 3
ทฤษฎีต่างๆที่ใช้

3.1 UML

UML (Unified Modeling Language) คือ รูปแบบหนึ่งของภาษาหรือ มาตรฐานที่ถูกกำหนดขึ้นเพื่อใช้ในการอธิบายถึงโครงสร้าง การทำงาน หรือแสดงถึงความสัมพันธ์ ของระบบที่มีความซับซ้อนนั่นเอง

3.1.1 Use Case Diagram

ยูสเคสไดอะแกรมเป็นการอธิบายขั้นตอนการทำงานของระบบโดยใช้ภาพประกอบถึง เซ็ตของการทำงาน (ยูสเคส) ของระบบ, ผู้กระทำ และความสัมพันธ์ระหว่างยูสเคสกับผู้กระทำ ซึ่งยูสเคสจะแสดงด้วยวงรี ส่วนผู้กระทำ (actor) จะแสดงโดยใช้รูปคน และเส้นตรงจะเป็นส่วนติดต่อระหว่างผู้กระทำกับยูสเคส จุดประสงค์ของไดอะแกรมนี้คือการแสดงถึงลักษณะของไดอะแกรมที่สามารถที่จะเข้าใจระบบได้ง่ายและรวดเร็วยิ่งขึ้น

ขั้นตอนในการเขียน use cases เบื้องต้น

1. ทำการกำหนดหน้าที่การทำงานของระบบ และทำการกำหนดขอบเขตของระบบ และกำหนดผู้กระทำและยูสเคส
2. เขียนผู้กระทำ (actor)
3. เขียนยูสเคส
4. เชื่อมโยงความสัมพันธ์ในยูสเคส
5. เชื่อมโยงความสัมพันธ์ระหว่างผู้กระทำ กับยูสเคส

3.1.2 Class Diagram

คลาสไดอะแกรม เป็นการแสดงถึงแนวความคิดของขอบเขตของปัญหา ใน UML การแสดงถึงกลุ่มของโครงสร้างของไดอะแกรม ซึ่งจะไม่มีการอธิบายถึงการกระทำต่างๆ แต่จะมีการแสดงถึงความสัมพันธ์กัน

การเชื่อมโยงความสัมพันธ์เป็นความสัมพันธ์ระหว่างแนวความคิดที่แสดงถึงความหมายและความสัมพันธ์กัน ใน UML การเชื่อมโยงความสัมพันธ์กันอาจหมายถึงโครงสร้างความสัมพันธ์ระหว่างออบเจคได้ด้วย

ขั้นตอนในการเขียนคลาสไดอะแกรมเบื้องต้น

1. กำหนดคลาสต่างๆที่ควรจะมีในระบบ
2. กำหนดคุณสมบัติและ หน้าที่การทำงานของแต่ละคลาส
3. ทำการวาดคลาสไดอะแกรมตามที่กำหนด
4. เชื่อมโยงความสัมพันธ์ระหว่างคลาสต่างๆ

3.1.3 Sequence Diagram

Sequence Diagram เป็นการใช้แผนภาพแสดงลำดับการทำงานของระบบ โดยจะแสดงรายละเอียดของยูสเคส, เหตุการณ์ภายนอกที่กระทำกับระบบ, การสั่งงาน, เหตุการณ์ต่างๆภายในระบบ และการทำงานของระบบที่มีการตอบสนองอีเวนท์ที่ทำกับระบบ

ขั้นตอนในการเขียน Sequence Diagram

1. วาดเส้นที่แสดงถึงระบบ
2. กำหนดผู้กระทำที่ทำงานโดยตรงกับระบบแล้ววาดเส้นไปยังแต่ละผู้กระทำ
3. กำหนดเหตุการณ์ที่ซึ่งผู้กระทำทำงานกับระบบแล้วทำการวาดลงในไดอะแกรม

3.2 Design Pattern

Christopher Alexander กล่าวไว้ว่า *"ในแต่ละแพทเทิร์นจะเป็นการอธิบายถึงปัญหาที่สามารถเกิดขึ้นและสามารถเกิดขึ้นได้อีกจากสภาวะแวดล้อมภายนอก และ แพทเทิร์นนี้ยังเป็นแกนสำคัญในการแก้ปัญหาต่างๆ เช่น คุณสามารถใช้แพทเทิร์นในการแก้ปัญหาที่เกิดขึ้นเป็นล้านครั้ง โดยที่คุณไม่จำเป็นที่จะต้องคอยแก้ปัญหานี้ซ้ำๆกัน"*

โดยทั่วไปแล้วแพทเทิร์นจะประกอบไปด้วยปัจจัย 4 อย่างดังนี้

- Pattern name คือ ชื่อที่เราสามารถไว้ใช้ในการเรียกรูปแบบที่เกิดปัญหา, วิธีแก้ปัญหา และ ลำดับการทำงาน หากเราตั้งชื่อได้ใจความที่เหมาะสมกับแพทเทิร์นนั้นๆจะทำให้สะดวกต่อการค้นหาว่าแพทเทิร์นใดที่เหมาะสมต่อการแก้ปัญหานั้นๆ
- Problem คือ รูปแบบที่แตกต่างกันของแพทเทิร์น โดยมันจะเป็นการอธิบายถึงรูปแบบของปัญหาที่เกิดขึ้นในแพทเทิร์นนั้นๆ บ่อยครั้งปัญหาที่เกิดขึ้นนั้นเราอาจไม่พบมันในรายชื่อของแพทเทิร์นต่างๆ ฉะนั้นก่อนทำการค้นหา หรือเปรียบเทียบปัญหากับแพทเทิร์นเราควรที่จะสามารถประยุกต์แพทเทิร์นต่างๆที่มีให้เข้ากับปัญหาที่เกิดขึ้นได้
- Solution จะเป็นการบอกถึงส่วนประกอบที่ถูกออกแบบมา, บอกความสัมพันธ์ต่างๆ, บอก responsibilities, และ collaborations โดย solution นั้นจะไม่ได้เป็นส่วนประกอบหรือการออกแบบขึ้นมาใหม่ หรือการ implementation เพราะแพทเทิร์นจะมีลักษณะคล้ายกับการนำรูปแบบหรือวิธีการเข้าไปแทนที่ โดยมันสามารถที่จะประยุกต์ใช้ได้กับสถานการณ์ต่างๆ
- Consequences คือ ผลลัพธ์และการ trade-off ของการประยุกต์ใช้แพทเทิร์น consequence จะเกิดขึ้นได้เมื่อมีการนำเอาแพทเทิร์นไปใช้ โดยที่ประสิทธิภาพสูงสุดที่จะเกิดขึ้นนั้นอยู่ที่เราสามารถนำมา ทำความเข้าใจ และออกแบบ แล้วนำไปประยุกต์ใช้ได้

3.3 Network

3.3.1 โพรโตคอลคืออะไร

โพรโตคอล (Protocol) คือระเบียบวิธีที่กำหนดขึ้นสำหรับการสื่อสารข้อมูล ให้สามารถส่งผ่านไปยังปลายทางได้อย่างถูกต้อง ในปัจจุบันโพรโตคอลในการสื่อสารข้อมูลก็มีอยู่หลายโพรโตคอล เช่น TCP/IP, IPX/SPX เป็นต้น

3.3.2 ความเป็นมาของ TCP/IP

TCP/IP เป็นมาตรฐานของการรับส่งข้อมูลระหว่างคอมพิวเตอร์สองระบบที่มีจุดเริ่มต้นราว 30 ปีมาแล้ว เมื่อกระทรวงกลาโหมสหรัฐฯ หรือ Department of Defense (DOD) ทำโครงการทดลองในปี ค.ศ. 1969 เชื่อมโยงคอมพิวเตอร์ทางทหารของแต่ละหน่วย ซึ่งเป็นคอมพิวเตอร์ต่างชนิดกันให้สามารถติดต่อรับส่งข้อมูลกันได้ (File Transfer) และสามารถใช้บริการอื่นๆ เช่น Remote Login รวมถึงการรับส่งจดหมายอิเล็คทรอนิคส์ (E-mail) ด้วย จุดประสงค์ของโครงการนี้ก็คือสร้างระบบเครือข่ายคอมพิวเตอร์ให้สามารถรับส่งข้อมูลกันได้ แม้ว่าสายส่งข้อมูลบางส่วนหรือคอมพิวเตอร์บางเครื่องในเครือข่ายจะถูกทำลายเสียหายไปก็ตาม ซึ่งเป็นคุณสมบัติที่สำคัญอย่างยิ่งเมื่อใช้งานยามเกิดสงคราม

ในขณะนั้นกองทัพบกเลือกใช้คอมพิวเตอร์และระบบเครือข่ายของ Digital Equipment Corporation (DEC), กองทัพเรือเลือกใช้คอมพิวเตอร์ของ Unisys ส่วนกองทัพอากาศเลือกใช้คอมพิวเตอร์ของ IBM เมื่อจะทำการรบกระทรวงกลาโหมสหรัฐก็พบว่าคอมพิวเตอร์ของทั้ง 3 กองทัพสื่อสารข้ามระบบกันไม่ได้ จึงได้ให้ทุนในการทำโครงการเชื่อมต่อคอมพิวเตอร์ของทั้ง 3 กองทัพเข้าด้วยกันเป็นระบบเครือข่าย โดยมีคุณสมบัติพิเศษแตกต่างจากระบบเครือข่ายที่ใช้งานกันทั่วๆไปคือ การรับส่งข้อมูลจะแบ่งข้อมูลออกเป็นส่วนย่อยๆเรียกว่า "แพ็คเก็ต" (packet) ข้อมูลแต่ละส่วนนี้จะถูกส่งไปให้คอมพิวเตอร์ผู้รับที่ปลายทางผ่านสายส่งข้อมูล โดยแต่ละส่วนอาจใช้เส้นทางสำหรับส่งข้อมูลคนละทางก็ได้ คอมพิวเตอร์ปลายทางจะนำข้อมูลที่ได้รับมาต่อรวมกันตามลำดับจนครบ โครงการนี้มีชื่อว่า Advanced Research Projects Agency Network หรือเป็นที่รู้จักกันดีในชื่อ ARPANET ในปี ค.ศ. 1975

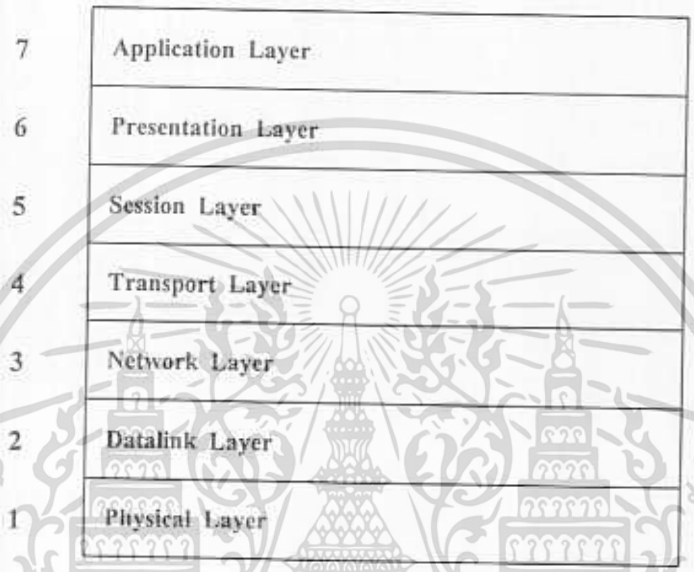
ซอฟต์แวร์ที่ใช้ควบคุมการรับส่งข้อมูลของ ARPANET ประกอบด้วยส่วนหลักๆ 2 ส่วนคือ Transmission control Protocol หรือ TCP และ Internet Protocol หรือ IP ซึ่ง TCP มีหน้าที่ตรวจสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ผู้รับและผู้ส่ง ให้ได้รับข้อมูลถูกต้องครบถ้วน หากข้อมูลสูญหายก็จะแจ้งให้ต้นทางส่งข้อมูลมาใหม่ ส่วน IP จะมีหน้าที่เลือกเส้นทางที่ใช้รับส่งข้อมูลผ่านระบบเครือข่ายและตรวจสอบ Address ของผู้รับโดยใช้ข้อมูลขนาด 4 ไบต์ หรือ 32 บิตเป็นตัวกำหนด Address ของผู้รับเรียกว่า IP Address ต่อมาในปี ค.ศ. 1983 TCP/IP ถูกกำหนดให้เป็นมาตรฐานการรับส่งข้อมูลของกระทรวงกลาโหมสหรัฐฯ เราจึงถือว่า TCP/IP มีต้นกำเนิดมาจากโครงการ ARPANET นั่นเอง และได้ถูกรวมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการ UNIX

หลังจากที่สถาบันการศึกษาและมหาวิทยาลัยต่างๆในสหรัฐฯเลือกใช้คอมพิวเตอร์ระบบปฏิบัติการ UNIX กันอย่างแพร่หลาย TCP/IP ก็ยิ่งมีบทบาทในการเชื่อมต่อคอมพิวเตอร์เข้าเป็นเครือข่ายมาก

ขึ้นเรื่องยๆ จนกระทั่งกลายเป็นอินเตอร์เน็ทในปัจจุบัน และได้มีการกำหนดมาตรฐานที่ใช้ในการรับส่งข้อมูลเครือข่ายอื่นๆเพิ่มเติมขึ้นมาในภายหลังรวมทั้ง OSI Model ในเวลาต่อมา

3.3.3 OSI Model

OSI กำหนดให้การสื่อสารข้อมูลจากระบบคอมพิวเตอร์หนึ่งไปยังอีกระบบหนึ่งแบ่งออกเป็น 7 ขั้นตอน ย่อยๆ ซึ่งคอมพิวเตอร์ทั้งสองระบบจะมีขั้นตอนทั้ง 7 นี้เหมือนกันทั้งสองฝั่ง เราเรียกการสื่อสารข้อมูลนี้ว่า OSI 7-Layer Reference Model ดังแสดงในรูป

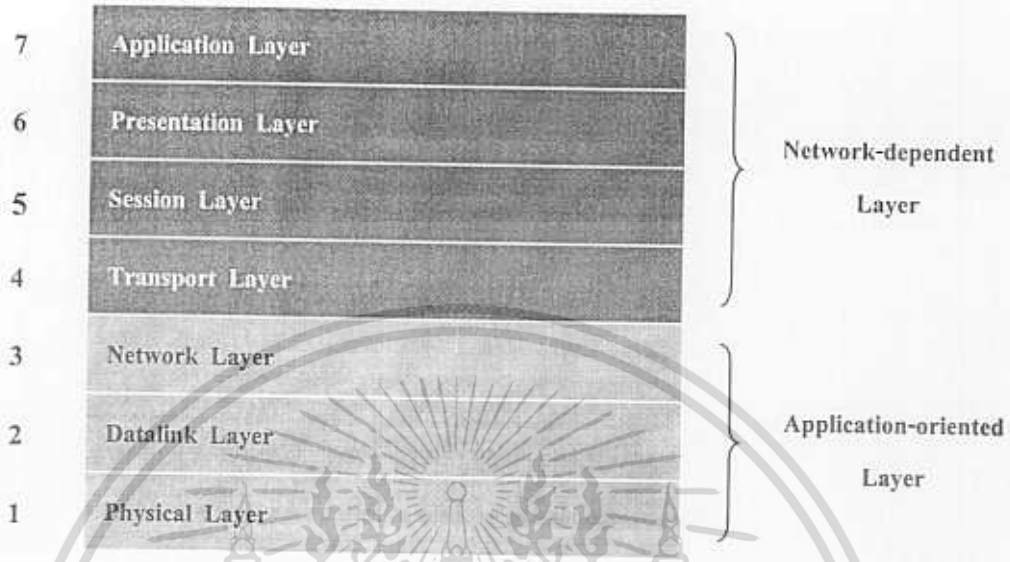


รูปที่ 3.1 โครงสร้างของ OSI 7-Layer Model

ในการสื่อสารข้อมูลจะประกอบด้วยชั้นย่อยๆ 7 ชั้น ในแต่ละชั้นหรือแต่ละ Layer จะเสมือนเชื่อมต่ออยู่กับชั้นที่เทียบเท่ากันของคอมพิวเตอร์อีกด้านหนึ่ง ส่วนการเชื่อมต่อกันจริงๆจะมีเพียงชั้นที่ 1 หรือ Layer 1 ซึ่งเป็นชั้นล่างสุดเท่านั้นที่มีการรับส่งข้อมูลเกิดขึ้นผ่านสายส่งข้อมูลระหว่างคอมพิวเตอร์ทั้งสองระบบ ส่วนชั้นอื่นๆจะไม่ได้เชื่อมต่อกันจริง เพียงแต่ทำงานเสมือนกับว่ามีการติดต่อรับส่งข้อมูลกับกลไกในชั้นเดียวกันของคอมพิวเตอร์อีกด้านหนึ่งเท่านั้น ในแต่ละชั้นจะมีการติดต่อรับส่งข้อมูลกับชั้นที่อยู่ติดกับตัวเองเท่านั้น จะติดต่อรับส่งข้อมูลข้ามกระโดดไปชั้นอื่นๆไม่ได้

ในทางปฏิบัตินั้น OSI 7-Layer Model จะแบ่งออกเป็น 2 กลุ่มใหญ่ๆ คือ กลุ่มแรกได้แก่ 4 ชั้นด้านบน คือชั้นที่ 7,6,5 และ 4 ทำหน้าที่เชื่อมต่อรับส่งข้อมูลระหว่างผู้ใช้กับซอฟต์แวร์โปรแกรมประยุกต์ให้รับส่งข้อมูลกับฮาร์ดแวร์ที่อยู่ชั้นล่างได้ถูกต้อง เรียกว่า Application-oriented layers ซึ่งจะเกี่ยวข้องกับซอฟต์แวร์เป็นหลัก โดย 4 ชั้นด้านบนนี้มักจะเป็นซอฟต์แวร์ของบริษัทใดบริษัทหนึ่งรวมอยู่อย่างเบ็ดเสร็จในโปรแกรมเดียว จะแยกออกจากกันเป็นชั้นๆ เพื่อใช้โปรแกรมของบริษัทอื่นๆได้ลำบาก หรือในบางกรณีก็อาจทำไม่ได้เลย

กลุ่มที่สองจะเป็นชั้นล่าง ได้แก่ชั้นที่ 3,2 และ 1 ทำหน้าที่เกี่ยวกับการรับส่งข้อมูลผ่านสายส่ง และควบคุมการรับส่งข้อมูล, ตรวจสอบข้อผิดพลาด รวมทั้งเลือกเส้นทางที่ใช้ในการรับส่งข้อมูล ซึ่งจะเกี่ยวข้องกับฮาร์ดแวร์เป็นหลักเรียกว่า Network-dependent layers ดังแสดงในรูป



รูปที่ 3.2 การแบ่งกลุ่มของ OSI 7-Layer Model

หน้าที่การทำงานของแต่ละชั้น คือ

ชั้นที่ 7 **Application Layer**

เป็นชั้นที่อยู่บนสุดของขบวนการรับส่งข้อมูล ทำหน้าที่เชื่อมต่อผู้ใช้เข้ากับระบบคอมพิวเตอร์ โดยรับคำสั่งต่างๆจากผู้ใช้ให้ระบบคอมพิวเตอร์แปลความหมาย และทำงานตามคำสั่งที่ได้รับในระดับโปรแกรมประยุกต์ ซึ่งการแปลคำสั่งจากผู้ใช้ส่งให้กับคอมพิวเตอร์รับไปทำงานนี้ จะต้องแปลออกมาถูกต้องตามกฎ (syntax) ที่ใช้ในระบบปฏิบัติการของคอมพิวเตอร์นั้นๆ ตัวอย่าง เช่น ถ้ามีการก็อปปี้ไฟล์เกิดขึ้นในระบบ คำสั่งที่ใช้จะต้องสร้างไฟล์ได้ถูกต้อง มีชื่อไฟล์ต้องประกอบด้วยตัวอักษรตามที่กำหนด ไม่มีตัวอักษรต้องห้ามมาตั้งเป็นชื่อไฟล์ เป็นต้น สิ่งต่างๆ เหล่านี้จะเกิดขึ้นในชั้นที่ 7 ของการสื่อสารข้อมูล รวมทั้งฟังก์ชันในการเชื่อมต่อรับส่งข้อมูลระหว่างชั้นที่ 7 กับชั้นที่ 6 ด้วย

ชั้นที่ 6 **Presentation Layer**

เป็นชั้นที่ทำหน้าที่ตกลงกับคอมพิวเตอร์อีกด้านหนึ่งว่า การรับส่งข้อมูลในระดับโปรแกรมประยุกต์จะมีขั้นตอนและข้อบังคับอย่างไร ข้อมูลที่ทำการรับส่งกันในชั้นที่ 6 นี้จะอยู่ในรูปแบบของข้อมูลชั้นสูง ซึ่งอยู่ในรูปแบบของคำสั่งที่มีกฎ (Syntax) บังคับอย่างแน่นอน เช่น ในการก็อปปี้ไฟล์ก็จะมีขั้นตอนย่อยประกอบกัน คือสร้างไฟล์ที่กำหนดขึ้นมาเสียก่อน จากนั้นจึงเปิดไฟล์ แล้วทำการรับข้อมูลจากปลายทางมาเก็บลงในไฟล์ที่สร้างขึ้นใหม่นี้โดยเนื้อหาของข้อมูลที่

ทำการรับส่งระหว่างกัน ก็คือคำสั่งของขั้นตอนย่อยๆข้างต้นนั่นเอง คำสั่งเหล่านี้จะต้องหมายถึงว่าจะให้ทำอะไรบ้างและถูกต้องตามกฎด้วย นอกจากนี้ในชั้นที่ 6 ยังทำหน้าที่แปลความหมายของคำสั่งที่ได้รับจากชั้นที่ 7 ให้เป็นคำสั่งระดับปฏิบัติการส่งให้ชั้นที่ 5 ต่อไปอีกด้วย

ชั้นที่ 5 **Session Layer**

ทำหน้าที่ควบคุม "จังหวะ" ในการรับส่งข้อมูลของคอมพิวเตอร์ทั้งสองด้านที่รับส่งแลกเปลี่ยนข้อมูลกันให้มีความสอดคล้องกัน (Synchronization) และกำหนดวิธีที่ใช้รับส่งข้อมูล เช่น อาจจะเป็นในลักษณะสลับกันส่ง (Half Duplex) หรือรับส่งข้อมูลพร้อมกันทั้งสองด้าน (Full Duplex) ซึ่งในชั้นที่ 5 นี้จะเป็นชั้นที่ควบคุมการรับส่งข้อมูลในลักษณะดังกล่าว ข้อมูลที่รับส่งกันในชั้นที่ 5 นี้จะอยู่ในรูปของ dialog หรือประโยคของข้อมูลที่สนทนาโต้ตอบกันระหว่างด้านรับและด้านส่งข้อมูล ไม่ได้มองเป็นคำสั่งอย่างในชั้นที่ 6

ชั้นที่ 4 **Transport Layer**

ทำหน้าที่เชื่อมต่อการรับส่งข้อมูลระดับสูงของชั้นที่ 5 (ซึ่งมองข้อมูลอยู่ในรูปที่เรียกว่า dialog หรือประโยคของข้อมูลที่โต้ตอบกัน) มาเป็นข้อมูลที่รับส่งในระดับฮาร์ดแวร์ เช่น แปลงค่าหรือชื่อของคอมพิวเตอร์ในเครือข่ายให้เป็น network address พร้อมทั้งเป็นชั้นที่ควบคุมการรับส่งข้อมูลจากปลายด้านส่งถึงปลายด้านรับข้อมูล ให้ข้อมูลมีการไหลลื่นตลอดเส้นทางตามจังหวะที่ควบคุมจากชั้นที่ 5 โดยในชั้นที่ 4 นี้จะเป็นรอบต่อระหว่างการรับส่งข้อมูลของซอฟต์แวร์กับฮาร์ดแวร์ การรับส่งข้อมูลของระดับสูงจะถูกแยกจากฮาร์ดแวร์ที่ใช้รับส่งข้อมูลที่ชั้นที่ 4 นี้ และจะไม่มีส่วนใดผูกติดกับฮาร์ดแวร์ที่ใช้รับส่งข้อมูลในระดับล่าง ดังนั้นฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ควบคุมการรับส่งข้อมูลในระดับล่างลงไปจากชั้นที่ 4 จึงสามารถสับเปลี่ยนและใช้ข้ามไปมากับซอฟต์แวร์รับส่งข้อมูลในระดับสูงที่อยู่ข้างบน (ตั้งแต่ชั้นที่ 4 ไปถึงชั้นที่ 7) ได้ง่าย หน้าที่อีกประการหนึ่งของชั้นที่ 4 คือ การควบคุมคุณภาพของการรับส่งข้อมูลให้มีมาตรฐานในระดับที่ตกลงกันของทั้งสองฝ่าย และการตัดข้อมูลออกเป็นส่วนย่อยๆให้เหมาะกับลักษณะการทำงานของฮาร์ดแวร์ที่ใช้ในเน็ตเวิร์ก

ชั้นที่ 3 **Network Layer**

ทำหน้าที่เชื่อมต่อคอมพิวเตอร์ของด้านรับและด้านส่งเข้าหากันผ่านระบบเครือข่าย พร้อมทั้งเลือกหรือกำหนดเส้นทางที่จะใช้ในการรับส่งข้อมูลระหว่างกัน และส่งผ่านข้อมูลที่ได้รับไปยังอุปกรณ์ในครือข่ายต่างๆจนกระทั่งถึงปลายทาง ในชั้นที่ 3 นี้ข้อมูลที่รับส่งกันจะอยู่ในรูปแบบของกลุ่มข้อมูลที่เรียกว่า Packet หรือ Frame ข้อมูลที่ชั้นที่ 4,5,6 และ 7 มองเห็นเป็นคำสั่งและ Dialog ต่างๆนั้น จะถูกแปลงและผนึกรวมอยู่ในรูปของ Packet และ Frame ที่มีเพียงแอดเดรสของผู้รับ, ผู้ส่ง, ลำดับการรับส่งและส่วนของข้อมูลเท่านั้น ตัวเนื้อหาของข้อมูลจะไม่มีผลใดๆในการรับส่งข้อมูลเลย หน้าที่อีกประการหนึ่งของชั้นที่ 3 นี้คือการทำ Call Setup หรือ เรียกติดต่อคอมพิวเตอร์ปลายทางก่อนการรับส่งข้อมูล และการทำ Call Clearing หรือยกเลิกการติดต่อเมื่อการรับส่งข้อมูลจบลงแล้ว ในกรณีที่มีการรับส่งข้อมูลนั้นต้องมีการติดต่อกันก่อน

ชั้นที่ 2 **Datalink Layer**

เป็นชั้นที่หน้าที่เชื่อมต่อการรับส่งข้อมูลในระดับฮาร์ดแวร์ โดยเมื่อมีการสั่งให้รับข้อมูลจากในชั้นที่ 3 ลงมา ชั้นที่ 2 จะทำหน้าที่แปลคำสั่งนั้นให้เป็นคำสั่งควบคุมฮาร์ดแวร์ที่ใช้รับส่งข้อมูล ทำการตรวจสอบข้อผิดพลาดในการรับส่งข้อมูลของฮาร์ดแวร์ และแก้ไขข้อผิดพลาดที่ตรวจพบนั้น ข้อมูลที่อยู่ในชั้นที่ 2 นี้จะอยู่ในรูปของ Frame คือกลุ่มของข้อมูลที่มีรูปร่างตามข้อบังคับของฮาร์ดแวร์ที่ใช้ในการรับส่งข้อมูล เช่น ถ้าฮาร์ดแวร์ที่ใช้เป็น Ethernet LAN ข้อมูลก็จะมีรูปร่างของ Frame ตามที่ระบุไว้ในมาตรฐานของ Ethernet หากว่าฮาร์ดแวร์ที่ใช้รับส่งข้อมูลเป็นชนิดอื่น เช่น Token Ring LAN หรือ Fiber Distributed Data Interface (FDDI) รูปร่างของ Frame ที่ใช้ในการรับส่งข้อมูลก็จะเปลี่ยนไปตามมาตรฐานนั้นๆ

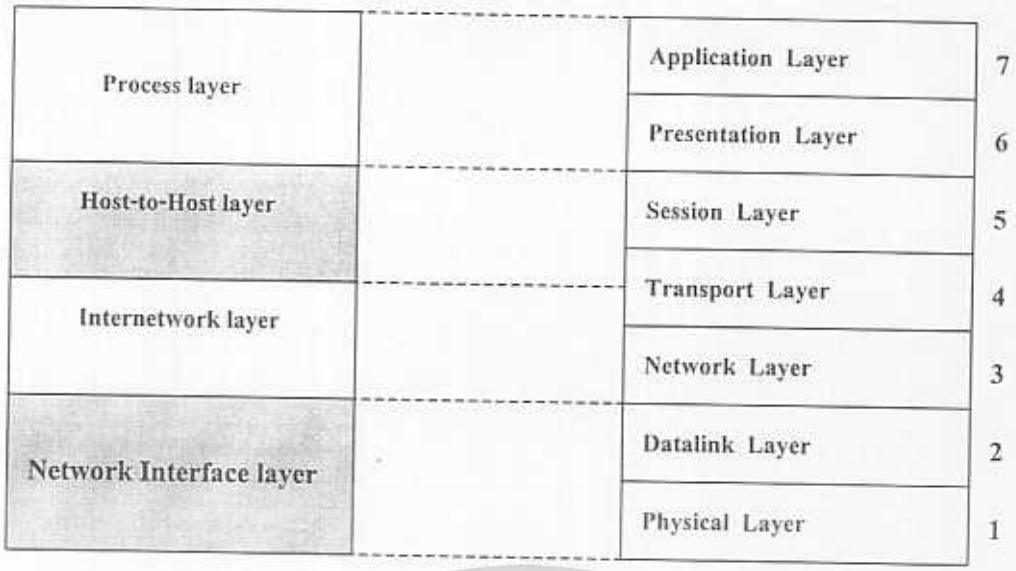
ชั้นที่ 1 Physical Layer

เป็นชั้นล่างสุดของขั้นตอนในการรับส่งข้อมูลของ OSI 7-Layer Reference Model ซึ่งเป็นชั้นเดียวที่มีการเชื่อมต่อกันทางกายภาพระหว่างคอมพิวเตอร์สองระบบที่ทำการรับส่งข้อมูลกัน ในชั้นที่ 1 นี้จะกำหนดคุณสมบัติทางกายภาพของฮาร์ดแวร์ที่ใช้เชื่อมต่อระหว่างคอมพิวเตอร์ทั้งสองระบบ เช่น สายที่ใช้รับส่งข้อมูลจะเป็นแบบไหน, ข้อต่อหรือปลั๊กที่ใช้ในการรับส่งข้อมูลมีมาตรฐานอย่างไร, ใช้ไฟกี่โวลต์, ความเร็วในการรับส่งข้อมูลเป็นเท่าใด ข้อมูลในชั้นที่ 1 นี้จะมองเห็นเป็นการรับส่งข้อมูลที่ละบิดเรียงต่อกันไป โดยไม่มีการพิจารณาเรื่องความหมายของข้อมูลเลยการรับส่งจะส่งข้อมูล "0" หรือ "1" ไปให้คอมพิวเตอร์ด้านรับข้อมูลในระดับฮาร์ดแวร์เท่านั้นหากการรับส่งข้อมูลมีปัญหาเนื่องจากฮาร์ดแวร์ เช่น สายสัญญาณที่ใช้รับส่งข้อมูลขาด, อุปกรณ์เสียหาย ก็จะเป็นหน้าที่ของชั้นที่ 1 นี้ที่จะตรวจสอบและแจ้งข้อผิดพลาดนั้นให้ชั้นอื่นๆ ที่อยู่เหนือขึ้นไปทราบ

3.3.4 โครงสร้างของ TCP/IP

TCP/IP มีการจัดกลไกการทำงานเป็นชั้นหรือ layer เรียงต่อกัน โดยในแต่ละ layer จะมีการทำงานเทียบได้กับ OSI model มาตรฐาน แต่บาง layer ของโพรโตคอล TCP/IP จะทำงานเทียบกับ OSI หลาย layer ปนกัน ซึ่งในแต่ละ layer ของโพรโตคอล TCP/IP จะประกอบด้วย

- Process layer
- Host-to-Host layer
- Internetwork layer
- Network Interface layer



รูปที่ 3.3 แสดง TCP/IP Stack เปรียบเทียบกับมาตรฐาน OSI

- **Process Layer**

จะอยู่ในชั้นบนสุดเรียกว่า Process layer ทำงาน 2 หน้าที่เทียบได้กับ Application layer และ Presentation layer ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชันต่างๆที่ทำงานเป็นโปรเซส อยู่ในเครื่องเซิร์ฟเวอร์ให้บริการและเครื่องที่ขอใช้บริการหรือ ไคลเอนต์ (client) ซึ่งจะติดต่อกันผ่านโพรโตคอลเฉพาะแอพพลิเคชันอีกทีหนึ่ง ตัวอย่างเช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการเรียกใช้งานคอมพิวเตอร์เครื่องที่อยู่ห่างไกลออกไปด้วยการใช้โปรแกรม telnet ที่เครื่องเซิร์ฟเวอร์ให้บริการ ตัวโปรเซส telnet ที่ทำงานอยู่ก็จะเรียกใช้โพรโตคอล Telnet เพื่อติดต่อกัน เป็นต้น

การทำงานของ แอพพลิเคชันต่างๆจะอยู่ที่ Process layer นี้ และมีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน จากการที่ Process layer ของ TCP/IP รองรับให้โพรโตคอลอื่นทำงานได้หลายโปรเซสและหลายโพรโตคอลได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้ปลายๆอย่างพร้อมกัน เช่น เปิดโปรแกรม IE เพื่อเรียกดูเว็บเพจ พร้อมกับใช้งานโปรแกรม Outlook Express เพื่อรับส่งอีเมล์ไปพร้อมกันได้โดยไม่ต้องรอให้ทำงานอย่างหนึ่งอย่างใดเสร็จก่อน

- **Host-to-Host Layer**

การทำงานที่ชั้นนี้จะมีบทบาทในการจัดการต่อจาก Process layer บางครั้งเรามักเรียกชั้น Host-to-Host ว่าเป็น Transport layer ซึ่งไม่ใช่ชั้นของ Transport layer ในมาตรฐาน OSI model การทำงานของ Host-to-Host นี้จะมีการสร้าง connection หรือการเชื่อมต่อกันระหว่างแอพลิเคชันกับ Host-to-Host layer โดยที่จุดที่เชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า port หรือ socket และในแต่ละแอพพลิเคชันก็จะสร้างการเชื่อมต่อผ่าน port ได้พร้อมกันหลายแอพพลิเคชัน ซึ่งการใช้งาน port ของแต่ละแอพพลิ

เคชันที่อยู่ในชั้น Process layer จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละโพรโตคอลจะมีการใช้งาน port หมายเลขต่างๆไม่ซ้ำกัน

เมื่อแอพพลิเคชันทำงานผ่านโพรโตคอลในชั้น Process layer จะมีการส่งผ่านข้อมูลไปยัง Host-to-Host layer ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน port ที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละโพรโตคอลทำได้ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซสที่แตกต่างกันก็ตาม หรือมีผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหลายแอพพลิเคชันในเวลาเดียวกัน ในชั้น Host-to-Host หรือ Transport layer ของ TCP/IP นี้ จะมีโพรโตคอลทำงานอยู่ 2 โพรโตคอลที่แตกต่างกัน คือ โพรโตคอล TCP และ โพรโตคอล UDP (User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปที่ชั้นถัดๆไป เราจะเห็นว่าโพรโตคอล TCP และ UDP จะถูกผนึกเข้าไปในโพรโตคอล IP อีกทีหนึ่งและส่งต่อไปยังเครือข่ายอินเตอร์เน็ตต่อไป

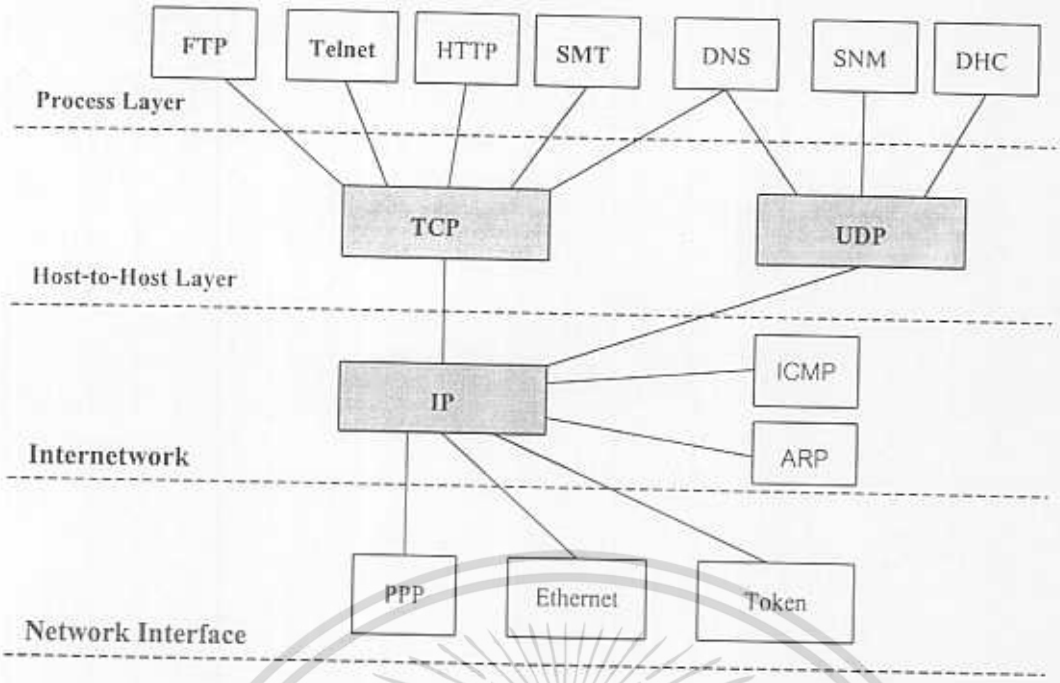
- **Internetwork Layer**

ในระดับล่างต่อมาในชั้น Internetwork layer มีหน้าที่เชื่อมต่อ และส่งผ่านข้อมูลในระหว่างเครือข่าย และจะทำหน้าที่เลือกเส้นทางการรับส่งข้อมูลผ่านอุปกรณ์เครือข่ายต่างๆจนไปถึงผู้รับข้อมูล โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านไปยังเครือข่ายใดๆบนอินเตอร์เน็ต คือ โพรโตคอล IP (Internet Protocol) นอกจากนี้ในชั้น Internetwork layer ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิดคือ โพรโตคอลโตคล Internet Control Message Protocol (ICMP) และโพรโตคอล Address Resolution Protocol (ARP)

- **Network Interface Layer**

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่าย แต่อย่างไรก็ตามในเครือข่ายอินเตอร์เน็ตนี้ ข้อมูลหรือ IP datagram จะถูกถ่ายทอดและส่งผ่านไปยังปลายทางโดยไม่คำนึงถึงรูปแบบการเชื่อมต่อทางกายภาพ ไม่ว่าจะเป็นการใช้เครือข่ายใยแก้วนำแสงหรือ เครือข่ายสาย Unshielded Twist Pair (UTP) เชื่อมต่อเป็นแบบเครือข่าย Ethernet ธรรมดาหรือ เครือข่าย Token Ring, ATM, ISDN ฯลฯ ก็ตาม

การทำงานระดับล่างสุดต่อจาก Internetwork layer จะเป็นการแปลงข้อมูล IP datagram ให้อยู่ในรูปที่เหมาะสม และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่ายต่อไปซึ่งในชั้น Network Interface layer นี้เมื่อเทียบกับมาตรฐาน OSI model แล้วจะเป็นการรวม 2 layer เข้าด้วยกัน คือ Data link layer และ Physical layer กล่าวโดยสรุปคือ การทำงานในชั้นต่างๆตามโครงสร้างของโพรโตคอล TCP/IP จะมีลักษณะดังรูป



รูป 3.4 โครงสร้างของโพรโตคอล TCP/IP ในแต่ละขั้นหรือ layer จะมีโพรโตคอลหลักทำหน้าที่ต่างๆ และส่งผ่านข้อมูลไปยังเครือข่ายและออกสู่อินเตอร์เน็ต

3.4 โอเพนดาต้าเบสคอนเน็กติวิตี (โอดีบีซี) (Open Database Connectivity, ODBC)

โอดีบีซี คือ วิธีการติดต่อและเข้าถึงจากแอพพลิเคชั่นสู่ระบบจัดการฐานข้อมูล (DBMS) โดยใช้ภาษาแอสคิวแอล เป็นมาตราฐานการเข้าถึงข้อมูลความสามารถในการเชื่อมต่อแบบนี้ทำให้แอพพลิเคชั่นสามารถเข้าถึงฐานข้อมูลได้หลายแบบ ซึ่งทำให้ผู้พัฒนาโปรแกรมสามารถพัฒนาโปรแกรมไปได้โดยไม่ต้องทำการระบุชนิดของระบบจัดการฐานข้อมูล

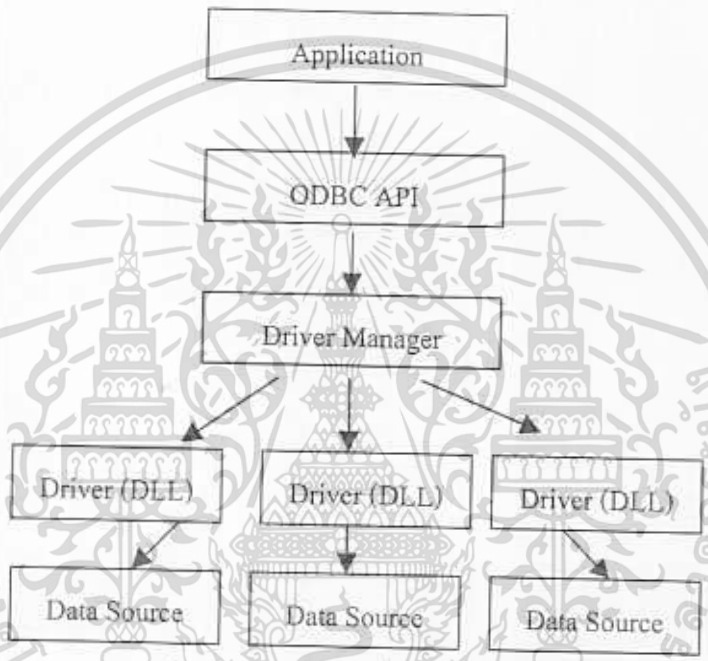
แต่เดิมนั้นการพัฒนาโปรแกรมประยุกต์ที่ใช้งานเกี่ยวกับฐานข้อมูล การเข้าใช้ฐานข้อมูลโปรแกรมเหล่านี้จะทำผ่านการเรียกใช้ เอ็มเบทเด็ดเอสคิวแอล (Embedded SQL) ซึ่งในขณะนั้นวิธีทางแบบนี้ก็ดูจะไปได้ทีเดียว เพราะว่าตัวแปรสามารถทำการเปลี่ยนรูปแบบของระบบไม่ว่าจะเป็นทางด้านฮาร์แวร์หรือ ซอฟท์แวร์ได้หลายแบบรวมทั้งระบบปฏิบัติการด้วย (โดยการคอมไพล์ใหม่ทุกครั้งที่มีการย้ายระบบ)

อย่างไรก็ตามในการพัฒนาโปรแกรมในระบบที่มีความแตกต่างกัน เช่น การเรียกใช้ข้อมูลของออราเคิลจากไมโครซอฟท์เอ็กเซล (Microsoft excel) วิธีการเข้าถึงข้อมูลแบบเดิมนั้นจะต้องทำการพรีคอมไพล์โค้ดของเอ็กซ์เซลและออราเคิลโดยใช้ ไอบีเอ็มพรีคอมไพเลอร์ (IBM Precompiler) และออราเคิลพรีคอมไพเลอร์ (Oracle Precompiler) ตามลำดับ ซึ่งจะเห็นว่าเป็นการยุ่งยากมากทีเดียว

วิธีการต่อเชื่อมแบบโอดีบีซีจะให้ความสะดวกในการติดต่อข้อมูลมากกว่าวิธีการดั้งเดิม โดยการกำหนดมาตรฐานการต่อเชื่อมของข้อมูล (Data Protocol, DBMS capability) และแนวทางนี้ทำให้เกิดความคิดที่จะสร้างไดร์ฟเวอร์การติดต่อกับงานข้อมูลขึ้นมา (DLL)

หมายเหตุ DLL (Dynamically Linked Libraries) เป็นไดรฟ์เวอร์ที่ทำหน้าที่แปลงชุดคำสั่งโอดีบีซีเอพีไอไปเป็นฟังก์ชันที่ดาต้าซอสสามารถเรียกใช้งานได้ โดยมากมักจะแปลงจากรูปแบบของเอสคิวแอลในโอดีบีซีไปเป็นรูปแบบที่ดาต้าซอสเข้าใจ

โอดีบีซีเป็นมาตรฐานสำหรับการสร้างดาต้าเบสไดร์ฟเวอร์ (Database Driver) ตามรูป โดยแอพพลิเคชั่นโปรแกรมจะใช้งานโอดีบีซีผ่านคำสั่งมาตรฐานคือ โอดีบีซีเอพีไอ (Open Database Connectivity Application Program Interface : ODBCAPI) โอดีบีซีเอพีไอไดร์ฟเวอร์ใช้ตามดาต้าซอสที่จะใช้แอพพลิเคชั่นโปรแกรม โดยใช้คำสั่งโอดีบีซี และสามารถใช้งานดาต้าซอสอื่นๆ ที่มีโอดีบีซีไดร์ฟเวอร์ได้ โดยไม่จำเป็นต้องเปลี่ยนแปลงโปรแกรม



รูปที่ 3.5 แสดงลักษณะทั่วๆ ไปของโอดีบีซี

ข้อดีของการติดต่อโดยใช้โอดีบีซี

1. ฟังก์ชั่นของโอดีบีซีอนุญาตให้แอพพลิเคชั่นติดต่อกับดีบีเอ็มเอสได้โดยสะดวก (การทำคำสั่งเอสคิวแอลและการรับผลลัพท์)
2. ใช้ภาษาแอสคิวแอลตามมาตราฐานเอสคิวแอล ซีเออี (SQL CAE), เอ็กซ์/ โอเพน (X/Open) และเอสคิวแอลแอกเซสกรุ๊ป (SQL Access Group, SAG)
3. มีการกำหนดการส่งกลับรหัสความผิดพลาด (Error Code) เป็นมาตราฐานเดียวกัน
4. เป็นวิธีการมาตราฐานในการติดต่อกับดีบีเอ็มเอส
5. มีการกำหนดชนิดของข้อมูล (Data Type) เป็นมาตราฐาน
6. ชุดคำสั่งเอสคิวแอล สามารถกำหนดได้แม้ในขณะรัน
7. สามารถเขียนโปรแกรมชุดเดียวแต่สามารถเข้าใช้ดีบีเอ็มเอสได้หลายตัว
8. ตัวโปรแกรมไม่ต้องรับผิดชอบในการดูแลการติดต่อข้อมูลกับดีบีเอ็มเอส
9. ค่าข้อมูลสามารถถูกส่งหรือรับได้ในรูปแบบที่สะดวกขึ้น

บทที่ 4
ความรู้ทั่วไปเกี่ยวกับ จาวา

ในหัวข้อนี้จะกล่าวถึงภาษาจาวา โดยทั่วๆ ไป

4.1 การทำงานของจาวา

เมื่อเราทำการสร้างไฟล์จาวาขึ้นมาได้แล้ว เราต้องทำให้ไฟล์นี้สามารถใช้งานได้ โดยการนำไฟล์จาวาที่สร้างขึ้นมานี้ไปทำการคอมไพล์ หลังจากคอมไพล์แล้วจะได้ไฟล์ที่เรียกว่า คลาสไฟล์ ซึ่งเป็นไฟล์ที่เก็บไบต์โค้ดไว้ (ไบต์โค้ดจะมีลักษณะใกล้เคียงคำสั่งเครื่องมากที่สุด) ไบต์โค้ดที่ได้นี้จะถูกรันบน Java Virtual Machine (JVM) ซึ่งจะทำการตรวจสอบไบต์โค้ดที่ได้ จากนั้นจะอ่านไบต์โค้ดและดำเนินการตามคำสั่ง

4.2 Java's Virtual Machine

ภาษา Java นำความคิดในการสร้างเครื่องจักรสมมติมาใช้ เพื่อให้ภาษามีคุณสมบัติ platform independent แต่แยกการแปลภาษาออกไปโดยใช้คอมไพเลอร์ แล้วนำโปรแกรมของเครื่องจักรสมมติที่ได้มาทำงานโดย interpreter วิธีการนี้จะทำให้โปรแกรมทำงานได้เร็วขึ้นกว่าการใช้ interpreter เพียงอย่างเดียว ไฟล์ของโปรแกรม source code ภาษา Java จะต้องมีสกุลเป็น .java คอมไพเลอร์ของภาษา Java จะทำหน้าที่คอมไพล์โปรแกรมภาษา Java ไปเป็นโปรแกรมที่ประกอบด้วยคำสั่งของ Java Virtual Machine (JVM) และจะถูกเก็บในไฟล์ที่มีสกุลเป็น .class เรียกว่าไฟล์ Java class

ปัจจุบันบริษัท Javasoft ซึ่งเป็นส่วนหนึ่งของบริษัท Sun Microsystems ได้กำหนดมาตรฐานของภาษา Java และเป็นผู้กำหนดชุดคำสั่งของ JVM รวมทั้งความหมายของแต่ละคำสั่ง ข้อกำหนดเหล่านี้เป็นมาตรฐานของภาษา Java ซึ่งเป็นที่เผยแพร่ให้แก่บุคคลทั่วไป ซึ่งใครก็อาจจะสร้าง JVM ของตนเองขึ้นได้ โดยอาจจะเป็น hardware หรือ software ก็ตาม แต่ต้องสามารถทำงานโปรแกรม Java class ให้ได้ผลลัพธ์ตามที่มาตรฐานกำหนดไว้

ภายใน JVM มีหน่วยประมวลผลสมมติที่เรียกว่า virtual processor ทำหน้าที่ประมวลผลคำสั่งของ JVM ปัจจุบัน JVM ที่เป็น hardware ยังอยู่ในระหว่างการพัฒนา ดังนั้น JVM เกือบทั้งหมดที่ ใช้กันอยู่ในตอนนี้จึงเป็นโปรแกรมที่จะลองการทำงานของ JVM บนเครื่องคอมพิวเตอร์ทั่วไป โดยทั่วไป virtual processor ของ JVM ที่จำลองขึ้นบนคอมพิวเตอร์เครื่องหนึ่ง จะแปลคำสั่งของ JVM เป็นคำสั่งของหน่วยประมวลผลในคอมพิวเตอร์เครื่องนั้นเรียกว่า native code แล้วให้หน่วยประมวลผลในคอมพิวเตอร์เครื่องนั้นทำงานคำสั่งนั้น คำสั่ง (opcode) ของ JVM มีขนาด 1 ไบต์ ทุกคำสั่ง (บางครั้งเราจึงเรียกโปรแกรม Java classว่าโปรแกรม byte code) จำนวนคำสั่งของ JVM จึงมีได้สูงสุดเพียง 256 คำสั่ง เปรียบกับหน่วยประมวลผลทั่วไปแล้วคล้ายกับว่าคำสั่งของ JVM มีมากมายอย่างยิ่ง แต่จริงๆแล้วคำสั่งของ JVM แบ่งออกได้เป็นไม่กี่ประเภท โดยที่และประเภทนั้นจะทำหน้าที่คล้ายๆกัน เพียงแต่ทำกับ operands ต่างชนิดข้อมูลกันเท่านั้น

ชุดคำสั่งของJVM ถูกออกแบบมาเพื่อสนับสนุนการทำงานของโปรแกรมเชิงวัตถุจึงมีคำสั่งเกี่ยวกับการสร้าง instance และการอ้างถึงสมาชิกใน instance ซึ่งไม่มีในหน่วยประมวลผลทั่วไป ภาษา Java เป็นภาษาที่เน้นความถูกต้องเกี่ยวกับชนิดข้อมูล จึงมีคำสั่งสำหรับคำนวณชนิดข้อมูลพื้นฐานแต่ละชนิด บางคำสั่ง

ของ JVM จะเหมือนกับคำสั่งที่มีในหน่วยประมวลผลทั่วไป แต่ก็ มีหลายคำสั่งที่ไม่มีในหน่วยประมวลผลที่ไป จึงต้องสร้างขึ้นเป็นโปรแกรมของคำสั่งนั้น

JVM ถูกออกแบบให้สามารถจำลองได้บนเครื่องทั่วไปไม่ว่าจะใช้หน่วยประมวลผลของบริษัทใด แต่หน่วยประมวลผลต่างรุ่นต่างบริษัทมีจำนวน registers ไม่เท่ากัน บางรุ่นมี registers หน้าที่พิเศษที่รุ่นอื่นไม่มี ผู้ออกแบบจึงตัดปัญหานี้โดยให้ JVM ไม่มี registers และทำการคำนวณทั้งหมดบน stacks ชุดคำสั่งของ JVM จึงเป็น stacked operations หรือกล่าวได้ว่า JVM เป็น stack machine Virtual processor ใน JVM จะทำการ mapping จากคำสั่ง byte code ในไฟล์ Java class ไปเป็นโปรแกรมของ native code ที่ทำหน้าที่ของคำสั่งนั้น แล้วส่ง native code นั้นให้หน่วยประมวลผลทำงาน สังเกตว่า native code นั้นอาจเป็น Application Program Interfaces (API) ของระบบปฏิบัติการที่ใช้หรืออาจจะเป็น standard classes ที่สร้างขึ้นสำหรับหน่วยประมวลผลนั้น ทำให้ JVM หนึ่งอาจใช้งานได้ในระบบต่างโดยเปลี่ยนแปลงแต่ standard classes เท่านั้น

4.3 Java Applet

Java เป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่มีพื้นฐานมาจากภาษา C++ ซึ่งเป็นภาษาเขียนโปรแกรมเชิงวัตถุ หรือ Object-Oriented โดยออกแบบมาให้ไม่ขึ้นอยู่กับเครื่องหรือฮาร์ดแวร์ใดๆ คือ สามารถทำงานได้บนเครื่องทุกรุ่นทุกแบบโดยไม่จำกัด การเอ็กซีคิวต์โปรแกรมภาษา Java นี้ไม่ได้ใช้วิธีการคอมไพล์เป็นภาษาเครื่องของคอมพิวเตอร์แต่ละแบบ แต่ใช้ "สภาพแวดล้อม" (Environment) ของ Java แทน โดยอ่านโปรแกรมต้นฉบับของ Java ขึ้นมา แล้วทำการแปลและเอ็กซีคิวต์ใน Java Virtual Machine (JVM) ทั้งนี้ในภาษา Java เองได้จัดเตรียม API ในแพลทฟอร์มต่างๆไว้ให้ใช้งาน ซึ่งประกอบด้วยฟังก์ชันพื้นฐานต่างๆ ที่จะเป็นในการพัฒนาโปรแกรม เช่น การจัดการวินโดว์ การจัดการหน่วยความจำ และการรับข้อมูลจากผู้ใช้ เป็นต้น

Java Applet เป็นรูปแบบหนึ่งของการใช้ภาษา Java ที่ไม่เอ็กซีคิวต์เป็นโปรแกรมเดี่ยวๆ แต่จะถูกฝังและทำงานภายใต้โปรแกรมอื่น เช่น โปรแกรมบราวเซอร์ เป็นต้น โดยจะเก็บ Java Applet เป็นไฟล์ไว้ที่เซิร์ฟเวอร์ และให้ไคลเอนต์โหลดมาทำงานด้วยโพรโตคอล HTTP เหมือนว่า Applet เป็นส่วนหนึ่งในคำสั่ง HTML ของเว็บเพจนั้น (โดยมี Tag Applet เป็นตัวกำหนดขอบเขตของ Java Applet ใน HTML) Java Applet นั้นจะไม่สามารถติดต่อกับระบบภายนอกโดยตรงได้เลย เช่น การติดต่อไปหาเว็บเซิร์ฟเวอร์ แต่จะต้องติดต่อผ่านบราวเซอร์อีกทีหนึ่ง การทำงานของ Java Applet ในลักษณะนี้จะช่วยให้บราวเซอร์สามารถทำงานแบบไดนามิค เช่น รับข้อมูลที่เป็นเสียง วิดีโอ หรือ ดาตาเบสต่างๆ รวมทั้งข้อมูลในประเภทอื่นๆได้

4.4 เจดีบีซี

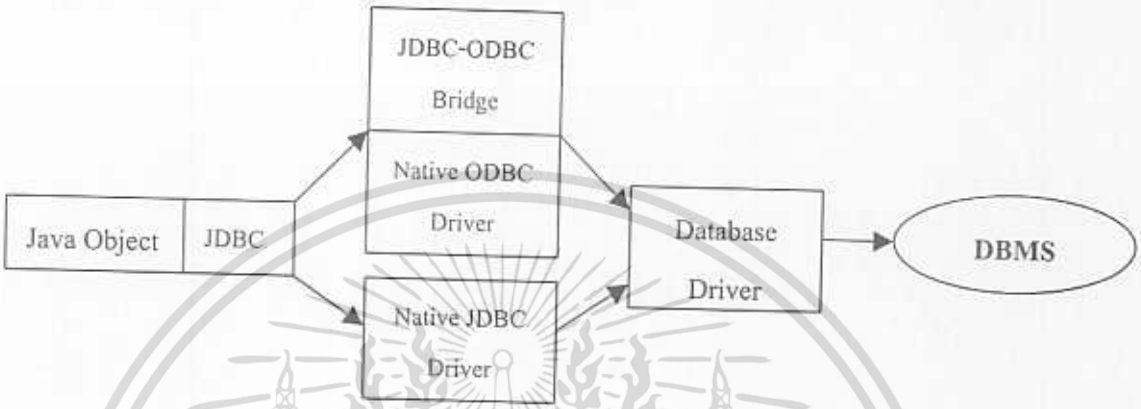
เจดีบีซี หรือ Java Database Connection เป็นมาตรฐานที่ช่วยในการติดต่อ ระหว่างตัวจาวากับระบบจัดการฐานข้อมูล (DBMS : Database Management System) โดยตัวที่ใช้ทำงานจริงๆระหว่างจาวากับระบบจัดการฐานข้อมูลก็คือ เจดีบีซีเอพีไอ (JDBC API : Java Database Connectivity Application Programming Interface) สำหรับใช้ในการปฏิบัติตามคำสั่งเอสคิวแอล ซึ่งประกอบด้วยคลาสและอินเตอร์เฟสที่เขียนขึ้นด้วยเจดีบีซี ซึ่งจะจัดหาเอพไอหามาตรฐานสำหรับผู้ทำการพัฒนา และทำให้เป็นไปได้ในการเขียนแอพพลิเคชันเกี่ยวกับฐานข้อมูลโดยใช้ภาษาจาวา

4.4.1 การทำงานของเจดีบีซี

เจดีบีซีจะทำหน้าที่หลักๆ 3 อย่างด้วยกันคือ

1. สร้างจะเชื่อมต่อกับฐานข้อมูลที่ต้องการติดต่อด้วย
2. ส่งคำสั่งเอสคิวแอล (SQL Statement) ไปยังฐานข้อมูลนั้นๆ
3. รับผลลัพธ์จากคำสั่งที่ส่งไปให้ในข้อ 2

รูปแบบที่ใช้ในเจดีบีซี จะเป็นดังนี้



รูปที่ 4.1 รูปแบบที่ใช้ในเจดีบีซี

ในรูปแบบนี้จะสังเกตเห็นได้ว่า จาวาออบเจ็ค (Java Object) จะร้องขอไปยังเจดีบีซีคลาส ซึ่งจะเป็นตัวเรียกไปยังตัว เจดีบีซีไดรฟ์เวอร์ หรือตัวเจดีบีซี-โอดีบีซีบริดจ์ ที่กำหนด จากนั้นตัวไดรฟ์เวอร์ จะเรียกตัวไดรฟ์เวอร์ของฐานข้อมูลจริงๆ เพื่อที่จะทำการเชื่อมต่อไปยังฐานข้อมูลให้สมบูรณ์

JDBC-ODBC Bridge

ทำหน้าที่เป็นตัวกลางในการเข้าถึงฐานข้อมูลโดยแปลงฟังก์ชันของเจดีบีซีให้อยู่ในรูปแบบฟังก์ชันของ โอดีบีซี ทำให้เจดีบีซีสามารถติดต่อกับฐานข้อมูลได้โดยผ่านการทำงานของโอดีบีซี

ข้อดี คือ ไม่ต้องติดตั้งและปรับเปลี่ยนไดรฟ์เวอร์ใหม่ทั้งหมด เพราะโอดีบีซีได้ถูกติดตั้งอยู่บนเครื่องคอมพิวเตอร์ส่วนมากที่ใช้งานในปัจจุบันอยู่แล้ว (Windows95/98/NT, Macintosh และ UNIX บางระบบ) ทำเพียงแค่นำเอาเจดีบีซี-โอดีบีซีบริดจ์มาต่อเชื่อมเพิ่มประสิทธิภาพให้กับโอดีบีซีเท่านั้น

ข้อเสีย คือ เจดีบีซี-โอดีบีซี บริดจ์ไดรฟ์เวอร์ถูกเขียนขึ้นมาจากภาษา C/C++ ในบางส่วนจึงทำให้ไม่สามารถดาวน์โหลดข้ามเครือข่ายอินเตอร์เน็ตและแปลบนเว็บบราวเซอร์ได้ เพราะภาษา C/C++ ต้องรันเฉพาะแต่ละระบบเท่านั้น

บทที่ 5
การพัฒนาซอร์ฟแวร์

5.1 ขั้นตอนในการพัฒนาซอร์ฟแวร์ทั่วไป

ในการพัฒนาซอร์ฟแวร์หนึ่งๆนั้นย่อมจะมีความซับซ้อนเกิดขึ้น เราจำเป็นที่จะต้องมีวิธีในการลดความซับซ้อนในการพัฒนาซอร์ฟแวร์ลง จึงได้มีการกำหนดรูปแบบในการพัฒนา หรือ ออกแบบซอร์ฟแวร์ขึ้นมาเพื่อให้การพัฒนาซอร์ฟแวร์นั้นเป็นไปได้ง่าย และมีระเบียบแบบแผน ในที่นี้จะขอใช้รูปแบบขั้นตอนในการพัฒนาซอร์ฟแวร์ ดังนี้

- Requirement Specification
- Analysis
- Design
- Implementation
- Testing

5.1.1 Requirement Specification

- จะเป็นการกำหนดขอบเขตของซอร์ฟแวร์ และเป็นการกำหนดพฤติกรรมต่างๆ ให้แก่ซอร์ฟแวร์
- เป็นการสื่อสารกันระหว่าง ผู้พัฒนาซอร์ฟแวร์, ผู้ออกแบบซอร์ฟแวร์ และ ผู้ใช้
- เป็นการกำหนดรูปแบบ วิธีการในการทดสอบซอร์ฟแวร์
- ประกอบด้วย
  - use case model
  - interfaces descriptions
  - a problem domain model

5.1.2 Analysis

โดยนิยามแล้วคำว่า analysis นั้นจะเป็นการกำหนดว่าซอร์ฟแวร์ควรจะทำ *อะไร* (what)

- เป็นการแสดงถึงโครงสร้างของซอร์ฟแวร์ โดยซอร์ฟแวร์นั้นๆจะต้องไม่ยึดติดกับการ Implement จริง
- เป็นการกำหนดขอบเขตบน logical implementation
- เป็นการกำหนดเพื่อให้มีโครงสร้าง เป็นแบบ stable, robust และ maintainable

5.1.3 Design

โดยนิยามแล้วคำว่า design นั้นจะเป็นการกำหนดว่าซอร์ฟแวร์ควรจะทำ *อย่างไร* (how)

- ต้องสามารถปรับเปลี่ยนขั้นตอนที่ได้จากการ analysis เพื่อให้สามารถนำไป implement ได้งานจริง

- มีรูปแบบในการแก้ไขจุดบอดที่เกิดจากการ analysis โดยที่เราจะต้องสามารถปรับเปลี่ยนข้อกำหนดที่ตายตัวเพื่อให้เข้ากับสภาพแวดล้อมที่นำไปใช้ในการ implement จริงได้
- มี 2 ระดับ คือ
  - Top-level design เป็นการออกแบบ module ต่างๆ และทำการเชื่อมโยงแต่ละ module เข้าด้วยกัน
  - Detail design เป็นการออกแบบในรายละเอียดของแต่ละ module เพื่อให้มีความสอดคล้องกับความต้องการ

5.1.4 Implementation

- Source code
- โดยในการเขียน code ที่ดีนั้น จะสามารถนำไปใช้ได้กับหลายๆภาษาที่มีโครงสร้างของภาษาเป็นแบบ Object-Oriented
- อย่างไรก็ตาม ภาษาทางด้าน Object Oriented programming จะช่วยทำให้คอนเซปท์พื้นฐานทั้งหมดง่ายดายขึ้น หากเราสามารถทำความเข้าใจกับโครงสร้างของภาษาได้เป็นอย่างดี

5.1.5 Testing

ในการตรวจสอบความผิดพลาดนั้นหากยิ่งเราสามารถพบข้อผิดพลาดนั้นๆได้เร็วเท่าไร ก็จะยิ่งเป็นผลดีต่อการพัฒนาซอร์ฟแวร์มากขึ้นเท่านั้น

- Test specification เป็นการตรวจสอบว่าซอร์ฟแวร์สามารถทำงานได้ตามข้อกำหนดที่กำหนดไว้ในขั้นตอนการ Requirement specification หรือไม่
- Test result เป็นการตรวจสอบว่าซอร์ฟแวร์สามารถทำงานได้ผลลัพธ์ตามที่ต้องการ และถูกต้องหรือไม่

5.2 เป้าหมายของการพัฒนาซอร์ฟแวร์

นอกจากขั้นตอนในการพัฒนาซอร์ฟแวร์แล้วนั้น การพัฒนาซอร์ฟแวร์ที่ดีนั้นควรที่จะต้องมีการกำหนดเป้าหมายของซอร์ฟแวร์ด้วย การกำหนดเป้าหมายของซอร์ฟแวร์ก็คือ การที่เมื่อซอร์ฟแวร์ของเราแล้วเสร็จตามขั้นตอนในการพัฒนาแล้วนั้นซอร์ฟแวร์ที่ดีควรจะ

- Stable : ความมีเสถียรภาพของซอร์ฟแวร์
- Robust : ความมีประสิทธิภาพในการทำงาน
- Maintainable : ความสามารถในการดูแลรักษาได้ง่าย

5.3 ข้อควรระวังในการพัฒนาซอร์ฟแวร์

- เราควรที่จะตั้งสมมติฐานไว้เสมอว่า ส่วนประกอบต่างๆภายในซอร์ฟแวร์ทั้งหมดนั้นสามารถที่จะถูกปรับเปลี่ยนได้อยู่ตลอดเวลา

- ส่วนที่จะถูกเปลี่ยนแปลงนั้นควรจะต้องอยู่ในรูปแบบของ local คือส่วนที่เปลี่ยนแปลงนี้ไม่ควรที่จะมีความเกี่ยวพันกับส่วนอื่นๆมากนัก เพราะอาจจะทำให้ส่วนที่เกี่ยวพันกันต้องถูกเปลี่ยนแปลงไปด้วย
- เรายังไม่ควรที่จะเชื่อว่าซอร์ฟแวร์ของเรานั้นเป็นซอร์ฟแวร์ที่ดีที่สุดแล้ว เราควรที่จะต้องคอยตรวจสอบและปรับปรุงแก้ไขซอร์ฟแวร์ของเราอยู่เสมอ
- จุดสำคัญที่สุดที่จะทำให้ซอร์ฟแวร์มีเสถียรภาพมากที่สุดนั้นอยู่ที่ขั้นตอนในการ analysis

5.4 รูปแบบในการพัฒนาซอร์ฟแวร์เชิงวัตถุ

ในวงจรชีวิตของออบเจ็ค โอเรียนเท็ด (Object Oriented life cycle) จะแบ่งเป็น 3 ขั้นตอนด้วยกัน คือ การวิเคราะห์ (Analysis), การออกแบบ (Design) และการนำไปใช้งาน โดยทั้ง 3 ขั้นตอนนั้นจะมีขอบเขตที่ครอบคลุมของการพัฒนาในรูปแบบทั่วๆ ไป

จุดแข็งของการรวมเอา วงจรชีวิตของออบเจ็คมาใช้ ก็คือเราสามารถใช้ออบเจ็คตัวเดิมที่มีอยู่แล้วในการแก้ไขปัญหาได้ มีการนำเอาเทคนิคต่างๆมาประยุกต์ใช้ในการวิเคราะห์ได้ และถูกนำมาใช้อีกในการออกแบบและการนำไปใช้งาน ข้อแตกต่างก็คือในการวิเคราะห์ได้ใช้เทคนิคต่างๆ ถูกนำมาประยุกต์ใช้ในระดับของ high-level Object ซึ่งหมายถึงโมเดลในการจัดการกับโพรซิเยอร์, บทบาทและข้อบังคับต่างๆ ภายในโครงสร้าง และในส่วนของการออกแบบนั้นออบเจ็คจะถูกกำหนดและทำให้ดีขึ้นโดยการเพิ่มออบเจ็คที่ใช้ในการจัดการทั้งในด้าน การติดต่อกับผู้ใช้, ความสัมพันธ์ต่างๆ, ความน่าเชื่อถือ เป็นต้น ซึ่งออบเจ็คเหล่านี้จะถูกนำไปใช้งานโดยตรงกับโปรแกรมภาษาออบเจ็คโอเรียนเท็ดได้เลย

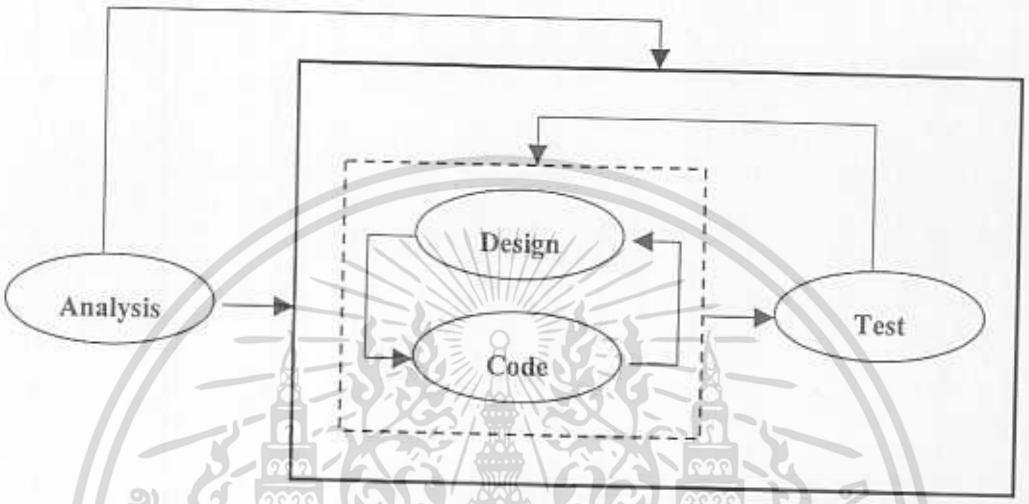
ในการวิเคราะห์ระบบที่เป็นออบเจ็ค-โอเรียนเท็ดนั้นจะทำให้เกิดมุมมองในการแก้ปัญหาที่แตกต่างไปจากการวิเคราะห์ในรูปแบบโครงสร้างทั่วๆไป ในทางลอจิคอล (logical) แล้ว คลาสจะถูกแทนที่ด้วย data และ function ซึ่งการแทนที่นี้ในความคิดแบบเก่าๆแล้วจะถูกกำหนดด้วย data เพียงอย่างเดียว คุณสมบัติการถ่ายทอด (Inheritance) จะถูกแทนที่ด้วยการแสดงความสัมพันธ์ของ subtype ในไดอะแกรมของการวิเคราะห์นี้ควรจะมีการแสดงให้เห็นถึงการส่งผ่านข้อความระหว่างอินสแตนท์ (Instance) ของคลาสต่างๆ แทนการไหลของข้อมูล และในแบบทั่วๆ ไปนั้นการไดอะแกรมที่แสดงความสัมพันธ์นั้นจะถูกแทนที่โดยการแสดงความสัมพันธ์ของคลาส และการสืบทอดของคลาสแทน

Henderson-Sellers, Booch and Bailin ได้พัฒนาเมธธอดที่ใช้สำหรับการพัฒนาระบบที่เป็นออบเจ็ค-โอเรียนเท็ดได้สำเร็จ นั่นคือการรวมเอาทั้ง top-down decomposition และ bottom-up system building เข้าด้วยกัน

การวิเคราะห์นั้นจะทำได้สำเร็จก็ด้วยวิธี top-down ซึ่งก็คือการกลุ่มของ คลาสแบบ high-level และการติดต่อระหว่างคลาสต่างๆ ในขั้นตอนการวิเคราะห์นั้นได้มีการนำมารวมเข้ากับการออกแบบแล้วทำให้ lower-level class diagram สามารถแสดงในส่วนของรายละเอียดที่ลึกลงไปของคลาสได้ คลาสไดอะแกรมจะถูกจะแบ่งได้เป็น n level ซึ่งในแต่ละเลเวลนั้นภายในจะแบ่งเป็นส่วนประกอบของคลาสและการแสดงการสืบทอด ในการออกแบบนั้นจะเป็นการรวมเอาคลาสที่จะนำไปใช้ไปจนถึงสภาพแวดล้อมที่เป็นทั้ง hardware และ software ที่ต้องการ และภาพรวมของลักษณะต่างๆที่จำเป็นประกอบด้วย generic abstract data / function type ซึ่งได้มาจากการสืบทอดของคลาส

ต่อมาจะเป็นการคำนึงในด้านของเทคนิค bottom-up ซึ่งจะมีลักษณะใกล้เคียงกับการนำชิ้นส่วนที่มีอยู่แล้วของคลาสมาเรียบเรียงให้เป็นประโยชน์นั่นเอง ข้อแตกต่างระหว่างวงจรชีวิตแบบโครงสร้างกับ ออบเจ็ค-โอเรียนเท็ด นั้นจะสามารถเห็นได้จากส่วนของการออกแบบ การออกแบบแบบออบเจ็ค-โอเรียนเท็ดนั้นจะแบ่งได้เป็น 2 ขั้นตอน ดังนี้

- Class design
- Application design



รูปที่ 5.1 แสดงวงจรชีวิตของออบเจ็ค-โอเรียนเท็ด

ในการออกแบบนั้นจะมีลักษณะคล้ายกับการรวมเอาแบบฟังก์ชันมาตามเดิม ลักษณะของคลาสและการพัฒนาการสืบทอดของคลาส ไปจนถึงการมองภาพรวมของข้อมูลและฟังก์ชัน การออกแบบคลาสนั้นจะประกอบไปด้วยหลักการของ bottom-up และ top-down โดยส่วนใหญ่แล้วการออกแบบแอพพลิเคชันในแบบ bottom-up จะกระทำภายหลังจากได้ทำการออกแบบคลาสเรียบร้อยแล้ว การเขียนโค้ดของคลาสที่เป็นอิสระนั้นมักจะมีการเขียนขึ้นก่อนการทำการออกแบบแอพพลิเคชันเสร็จ โดยในระหว่างการออกแบบแอพพลิเคชันและการเขียนโค้ดนั้นมักจะเป็นส่วนที่ไม่น้อยในการวงจรชีวิตทั่วๆไป

จากในรูปที่ 5.1 เป็นการแสดงให้เห็นถึงการพัฒนาวงจรชีวิตซึ่งเป็นหลักสำคัญสำหรับการวิเคราะห์และออกแบบออบเจ็ค-โอเรียนเท็ด

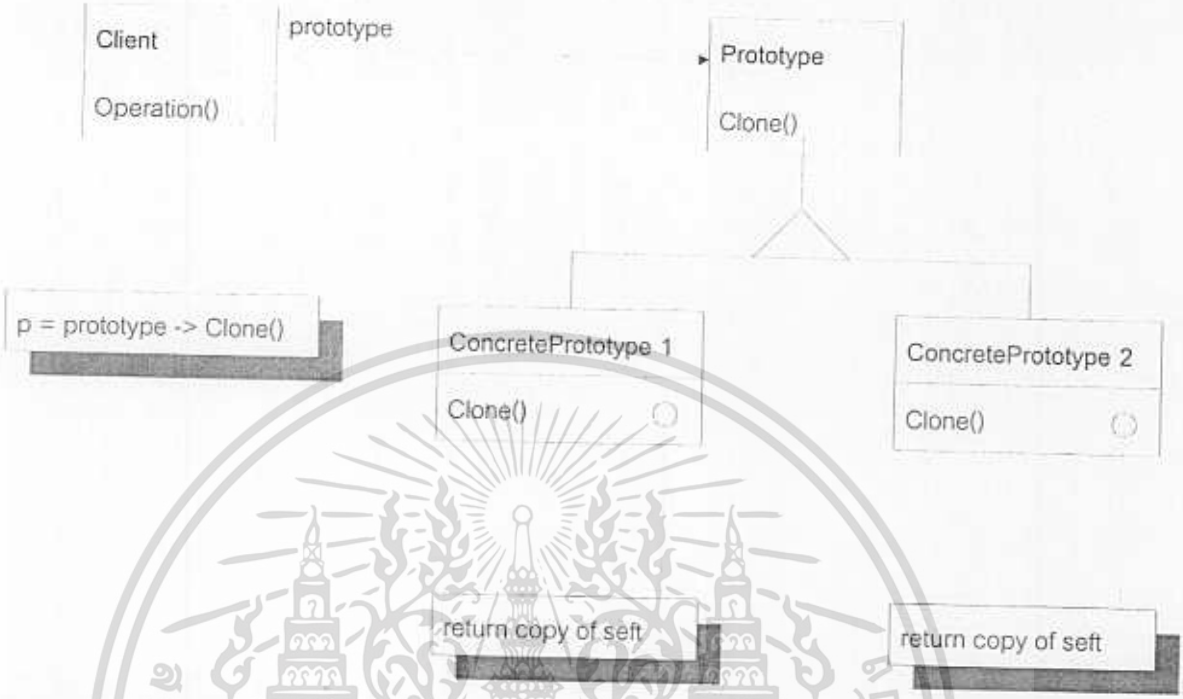
5.5 แพทเทิร์นที่สามารถนำมาใช้ในการพัฒนาโปรแกรมเกมที่เขียนขึ้นได้

โดยที่จะกล่าวถึงนี้เป็นตัวอย่างรูปแบบของแพทเทิร์นที่สามารถนำมาประยุกต์ใช้ในการพัฒนาซอร์ฟแวร์ครั้งนี้

- **Prototype Pattern**

ระบุชนิดของออบเจ็คที่จะสร้างโดยทำเป็นโพรโตไทพ์ไว้ และทำการสร้างออบเจ็คใหม่โดยการคัดลอกมาจากโพรโตไทพ์นี้

เราสามารถใช้โพรโตไทพ์แพทเทิร์นนี้ในการลดจำนวนของคลาสที่เกิดขึ้นได้ โดยแทนที่เราจะทำการสร้างคลาสใหม่ที่มีค่าต่างๆเหมือนกันขึ้นมา เราจะทำการกระจายคลาสเดิมออกให้อยู่ในรูปที่ง่ายที่สุดแล้วทำการสร้างคลาสใหม่จากส่วนนี้

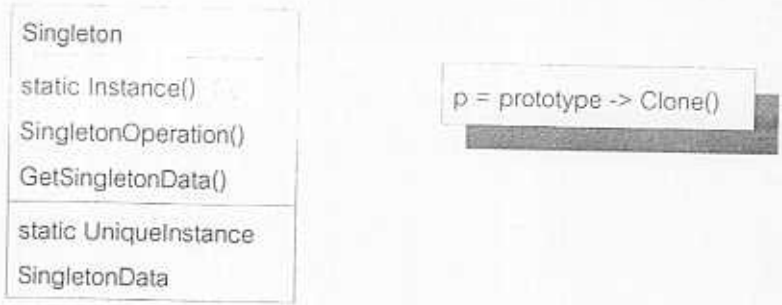


รูปที่ 5.2 โครงสร้างของ Prototype Pattern

- **Singleton Pattern**

เป็นการกำหนดให้คลาสสามารถมีได้เพียง instance เดียว และยินยอมให้สามารถเข้าถึงได้จากภายนอก

มันจะมีความสำคัญสำหรับบางคลาสที่จำเป็นต้องเฉพาะเจาะจงเพียงอินสแตนท์เดียว เช่นหากในซอร์ฟแวร์เรามีพริ้นเตอร์อยู่หลายเครื่อง จะมีพริ้นเตอร์เพียงตัวเดียวเท่านั้นที่จะได้งานไป วิธีที่ดีที่สุดในการทำคือ เราจะต้องมีช่องว่างในการเก็บค่าอินสแตนท์นั้น โดยคลาสนั้นจะต้องไม่สามารถสร้างอินสแตนท์ตัวนั้นขึ้นมาใหม่ได้อย่างแน่นอน และต้องมีการทำช่องทางให้สามารถเข้าถึงค่าอินสแตนท์นั้นได้

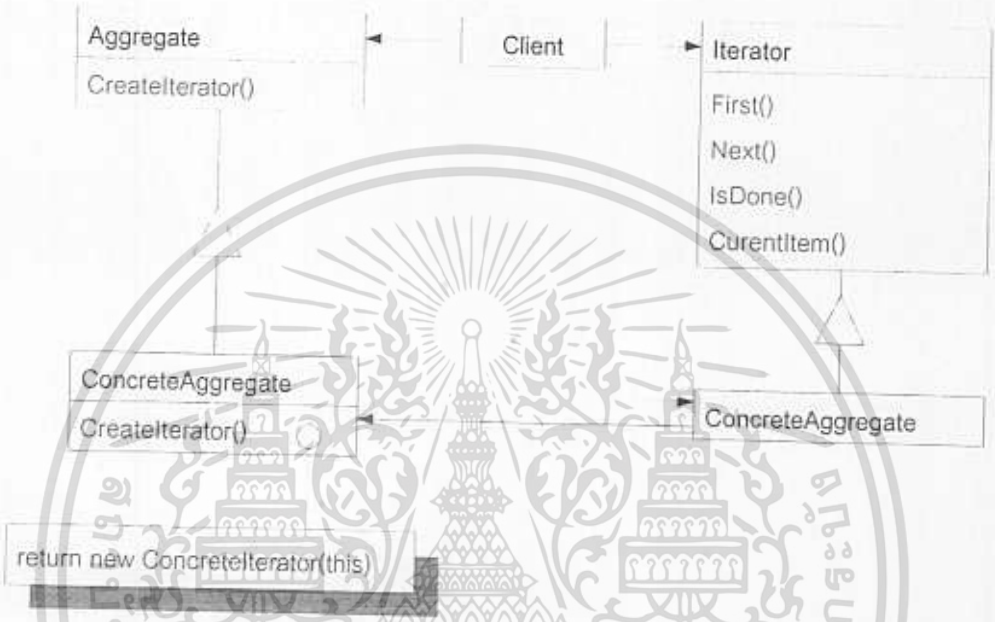


รูปที่ 5.3 โครงสร้างของ Singleton Pattern

- **Iterator Pattern**

เป็นการยินยอมให้มีการเข้าถึงส่วนย่อยต่างๆ ของออบเจ็คในแบบเรียงลำดับ

โดย Iterator pattern จะอนุญาตให้คุณสามารถทำทั้งหมดได้ ประเด็นหลักในการทำแพทเทิร์นนี้จะเป็นการนำเอาภาระหน้าที่สำหรับการเข้าถึงและการพิจารณาทั้งหมดของการรายการ และนำมาใส่ไว้ใน iterator ออบเจ็ค ใน iterator คลาสจะกำหนดให้มีการเข้าถึงได้ในทุกๆรายการ และจะมีการเก็บสถานะเพื่อใช้ในการบอกว่าได้เข้าถึงส่วนนั้นๆไปแล้ว

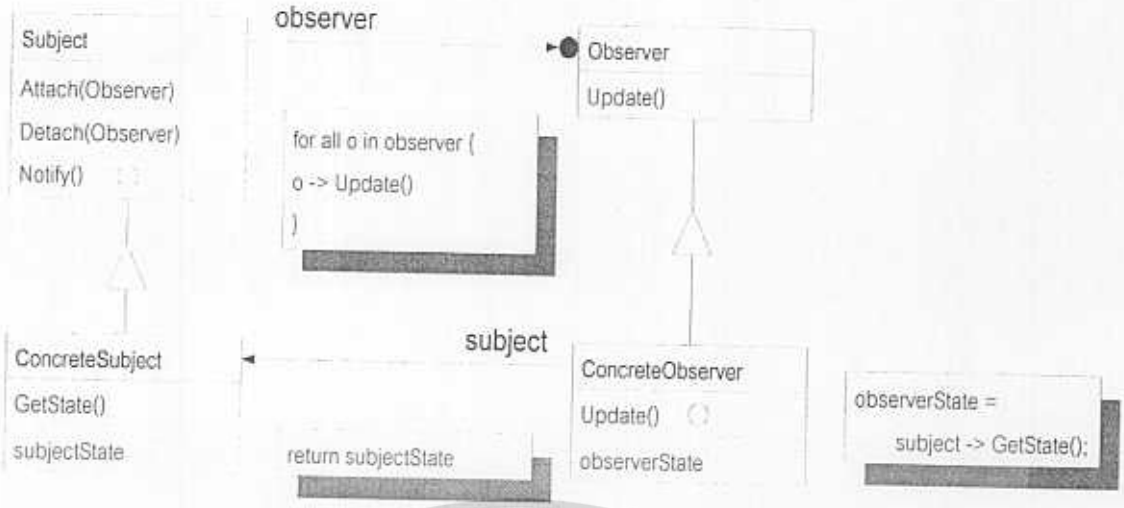


รูปที่ 5.4 โครงสร้างของ Iterator Pattern

- **Observer Pattern**

เป็นการกำหนด one-to-many ระหว่างออบเจ็ค เมื่อออบเจ็คหนึ่งๆมีการเปลี่ยนแปลงสถานะ ออบเจ็คอื่นที่มีความสัมพันธ์กันก็จะสามารถรับรู้และอัปเดทได้อย่างอัตโนมัติ

โดย Observer pattern นี้จะเป็นการกล่าวถึงความสัมพันธ์ที่มั่นคง ประเด็นสำคัญของออบเจ็คนี้คือ subject และ observer โดยในหนึ่ง subject นั้นจะประกอบไปด้วยหลาย observer ที่เป็นอิสระจากกัน โดย observer จะได้รับการแจ้งเมื่อ subject มีการเปลี่ยนสถานะเกิดขึ้น



รูปที่ 5.5 โครงสร้างของ Observer Pattern

- **State Pattern**

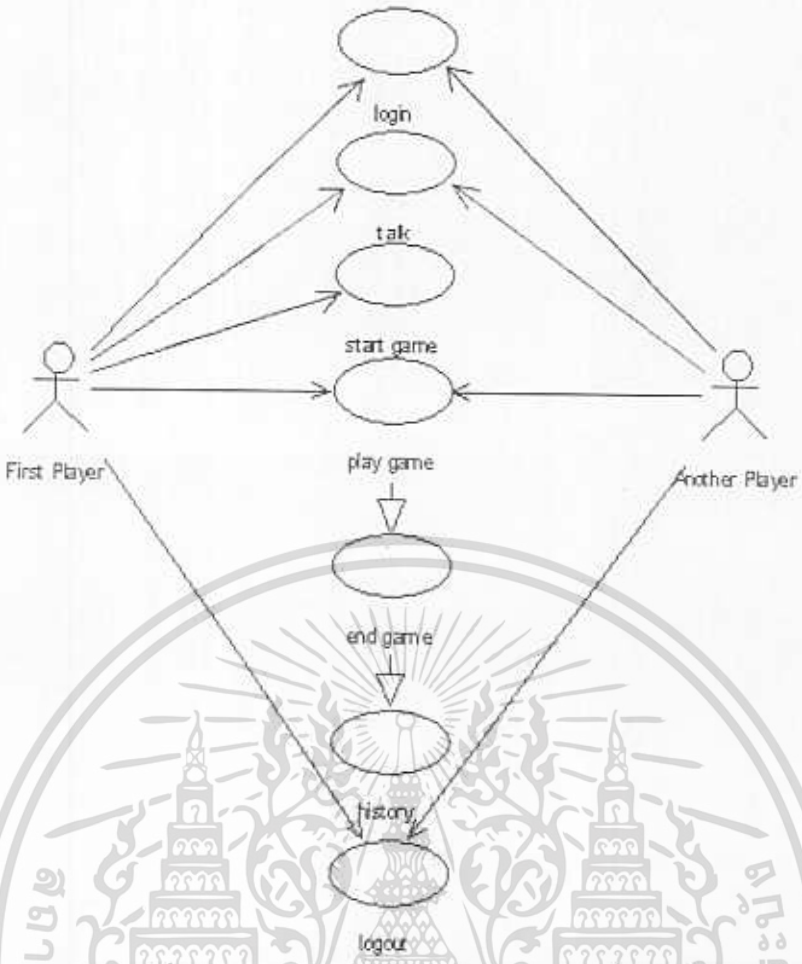
เป็นการเปลี่ยนแปลงพฤติกรรมของออบเจ็คเมื่อมีการเปลี่ยนแปลงสถานะของออบเจ็คนั้นๆ

รูปที่ 5.6 โครงสร้างของ State Pattern

5.6 ขั้นตอนในการพัฒนาโปรแกรม

5.6.1 การทำ Requirement Specification

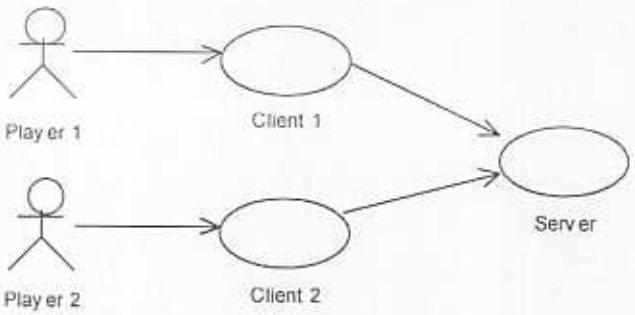
- เป็นเกมพัฒนาความรู้และทักษะสำหรับเด็ก
- เป็นเกมประเภท Board game
- ใช้ลูกเต๋าในการเล่น
- ผู้เล่นจะผลัดกันเล่นคนละรอบ
- มีการตอบคำถาม
- หาผู้ชนะจากคะแนนสะสมสูงสุด



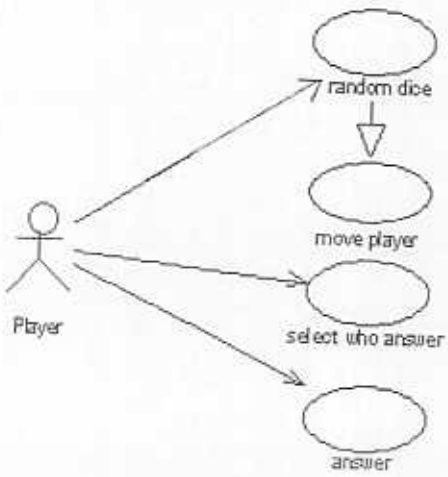
รูปที่ 5.7 แสดงยูสเคสในส่วนของการทำงานเบื้องต้นทั้งหมด

จากในรูปที่ 5.7 นี้จะเป็นการแสดงถึงลักษณะของการกำหนดความต้องการของโปรแกรม โดยจะเป็นการเสดงเพียงการรูปแบบการทำงานภายนอก หรืออาจเรียกได้อีกอย่างว่าเป็นเพียงส่วนที่ผู้ใช้สามารถสัมผัสได้

และจากในรูปที่ 5.8 นี้จะเป็นการแสดงถึงลักษณะของการกำหนดความต้องการของโปรแกรมในส่วนของการติดต่อผ่านอินเตอร์เน็ท โดยจะแบ่งการทำงานเป็นส่วนของตัวเซิร์ฟเวอร์ซึ่งจะเป็นตัวกลางการทำงานทั้งหมดของเกม และส่วนของไคลเอนต์ซึ่งจะใช้เป็นตัวเกมและตัวติดต่อกับผู้เล่นโดยตรง



รูปที่ 5.8 แสดงยูสเคสในส่วนของการติดต่อ เซิร์ฟเวอร์-ไคลเอนต์



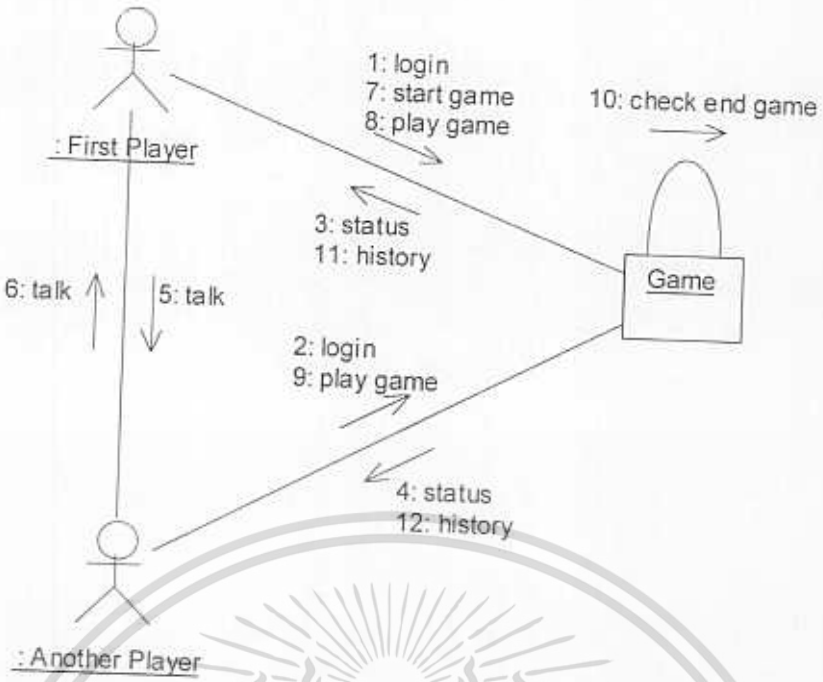
รูปที่ 5.9 แสดงยูสเคสในส่วนของการทำงานของผู้เล่นหนึ่งคน

จากในรูปที่ 5.9 นี้จะเป็นการแสดงถึงลักษณะของการกำหนดความต้องการของโปรแกรม โดยจะมีลักษณะคล้ายกันกับในรูปที่ 5.7 แต่จะแตกย่อยในมุมมองของผู้เล่นเพียงคนเดียว เพื่อเป็นตัวกำหนดการเริ่มต้นการเขียนโปรแกรม

5.6.2 การวิเคราะห์

เป็นการวิเคราะห์ว่าโครงงานจะต้องทำอะไร โดยในที่นี้จะใช้การเขียน sequence diagram ในการอธิบายถึงการทำงานของโครงงาน

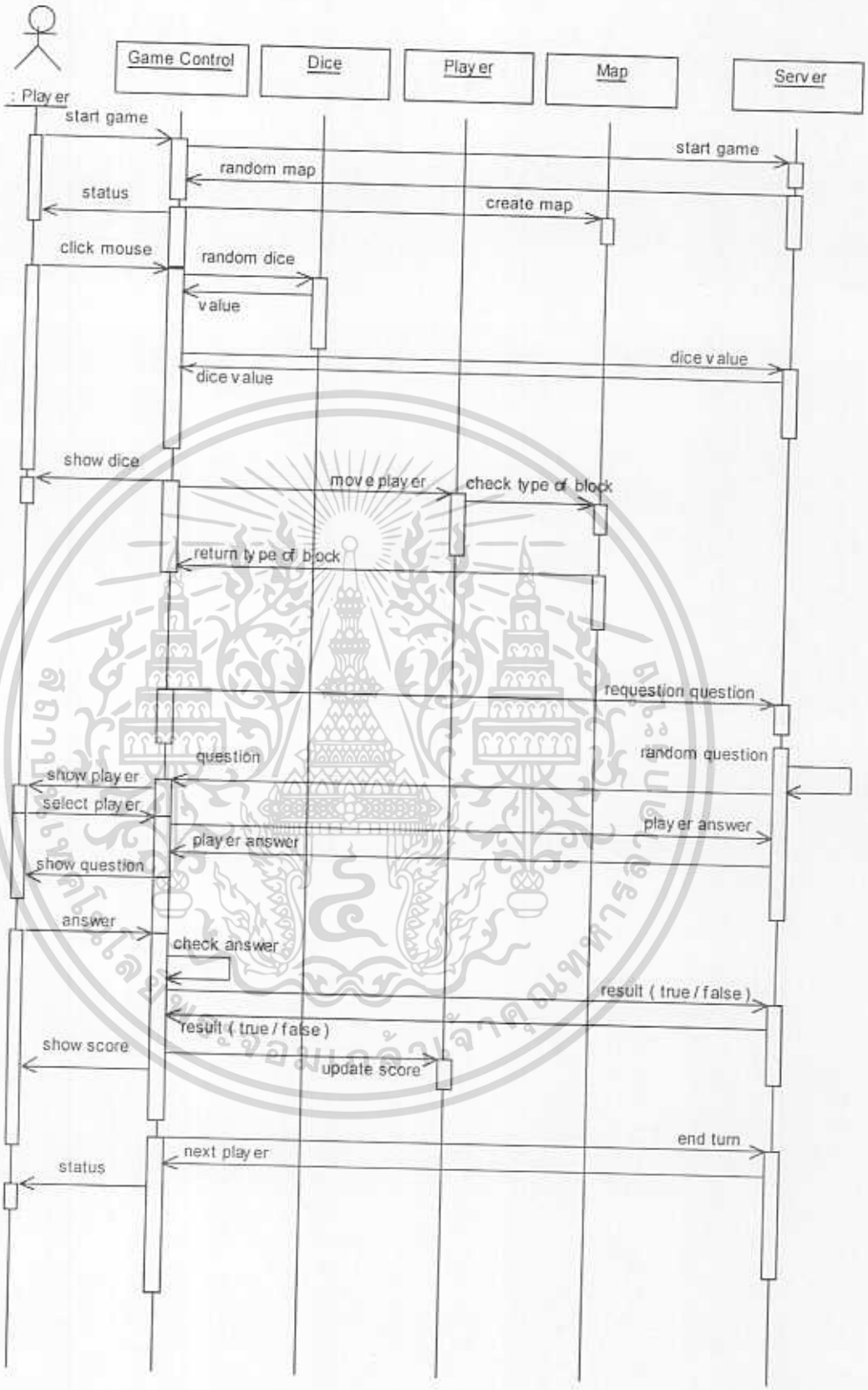
ในรูปที่ 5.10 เป็นการเขียน Sequence diagram แสดงถึงลำดับการทำงานของตัวเกม โดยจะแสดงลำดับการทำงานโดยเริ่มตั้งแต่การ login การเล่นเกม ไปจนจบเกม โดยในรูปที่ 5.11 นั้นจะเป็นการนำเอา diagram จากในรูปที่ 5.10 มาแปลงเป็น collaboration diagram ซึ่งได้มีการนำเอา Tools ที่ชื่อ Rational Rose 98 มาช่วยในการเขียน diagram โดยโปรแกรม Rational Rose 98 นี้จะช่วยในการแปลงจาก Sequence diagram เป็น Collaboration diagram ให้เราโดยอัตโนมัติ



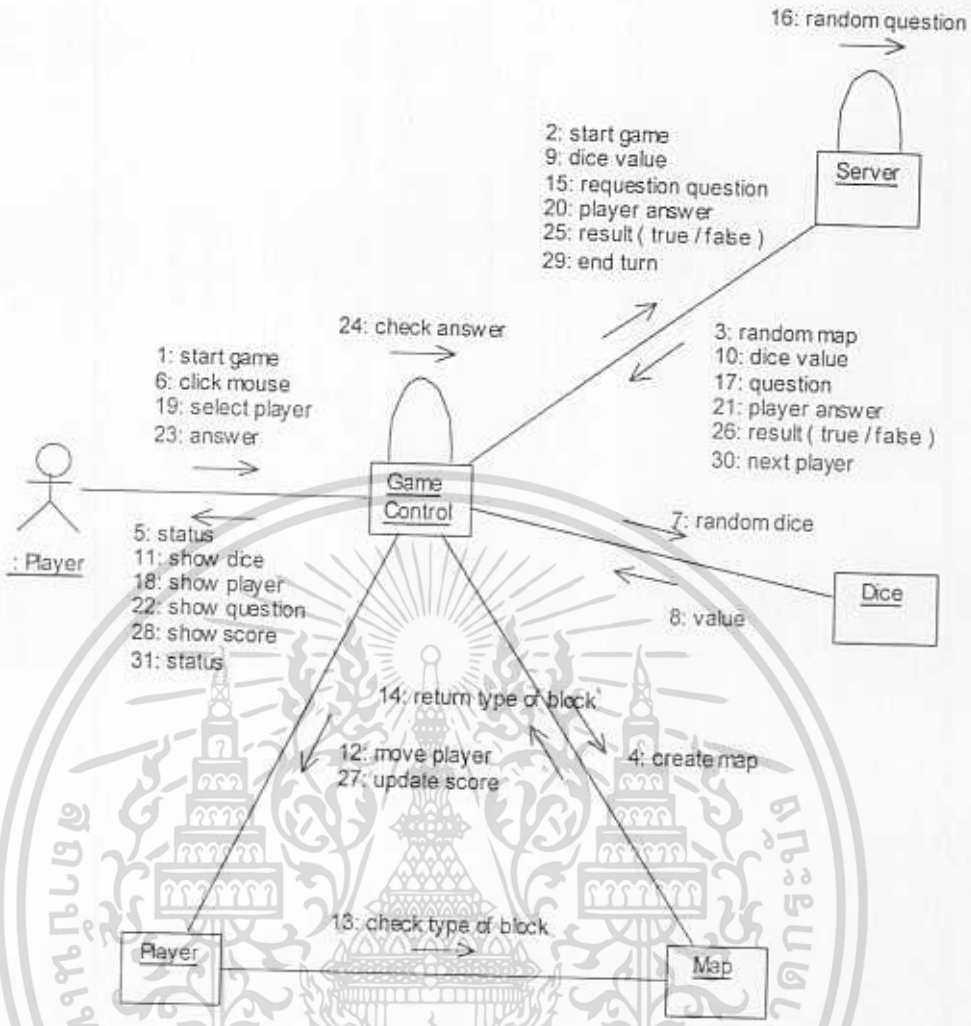
รูปที่ 5.11 แสดง collaboration diagram ในส่วนของการทำงานโดยรวมทั้งหมด

ในรูปที่ 5.12 นั้นจะเป็นการแสดงการทำงานในส่วนของการเล่นในส่วนของผู้ต่อหนึ่งคนเท่านั้น โดยจะเป็นการแสดงลำดับการทำงานที่มีขั้นตอนที่ละเอียดขึ้นเมื่อเทียบกับในรูปที่ 5.10 ซึ่งจะเป็นการแสดงลำดับการทำงานที่อยู่ภายในของโปรแกรม ทั้งในส่วนที่เป็นตัวที่ใช้ติดต่อกับผู้เล่น ส่วนที่เป็นตัวเกม และลำดับการติดต่อกับตัวเซิร์ฟเวอร์ รวมถึงแสดงให้เห็นถึงส่วนการทำงานของตัวเซิร์ฟเวอร์อีกด้วย

ในรูปที่ 5.13 นั้นจะเป็นการนำเอา sequence diagram จากในรูปที่ 5.12 มาแปลงเป็น collaboration diagram ซึ่งก็เป็นการนำเอาทูลที่ชื่อ Rational Rose 98 มาช่วยเหลืออีกเช่นเดียวกับในรูปที่ 5.11



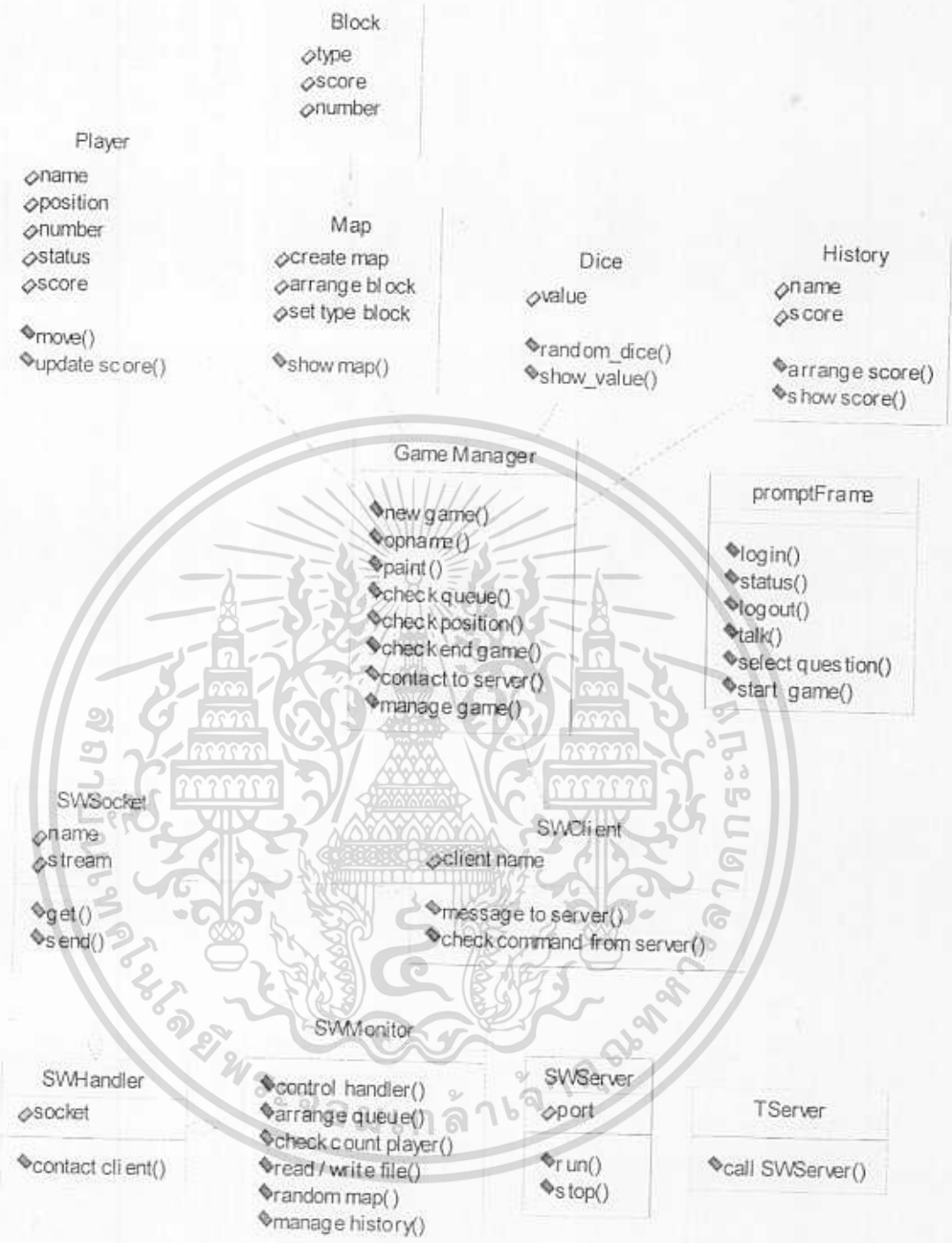
รูปที่ 5.12 แสดง sequence diagram ในส่วนของการทำงานเมื่อเทียบกับผู้เล่นหนึ่งคน



รูปที่ 5.13 แสดง collaboration diagram ในส่วนของการทำงานเมื่อเทียบกับผู้เล่นหนึ่งคน

5.6.3 การออกแบบ

จะเป็นการนำเอาส่วนของการ analysis มาใช้ในการออกแบบ เพื่อให้ง่ายในการทำในส่วนของการ implementation โดยจะทำการออกแบบให้อยู่ในรูปของ class diagram ต่างๆ ดังในรูปที่ 5.14 แสดงให้เห็นถึง class diagram ที่ได้ออกแบบไว้แล้ว



รูปที่ 5.14 แสดง class diagram ของความสัมพันธ์ระหว่าง class ทั้งหมด

และได้มีการนำเอาส่วนของ design pattern มาใช้ในส่วนของการออกแบบด้วย โดยจะสามารถนำมาแพทเทิร์นต่างๆที่ได้กล่าวไปแล้วในหัวข้อที่ 3.2 มาใช้งานในส่วนต่างๆของการออกแบบที่จะกล่าวต่อไป

- Prototype Pattern โดยในแพทเทิร์นนี้นั้นจะนำไปใช้ในส่วนของตัวเดินของผู้เล่นแต่ละคน คือ จะมีการสร้างตัวเล่นขึ้นมาเพียงตัวเดียว เพื่อเป็น prototype โดยกำหนดว่าใน prototype นั้นจะมีค่าอะไรบ้าง แล้วเราจึงจะทำการสร้างตัวเดินของผู้เล่นแต่ละตัวจาก prototype นี้ โดยอาจจะมีการเปลี่ยนแปลงค่าตัวแปรบางค่าไปในส่วนของรายละเอียดที่แตกต่างกัน
- Singleton Pattern เราจะทำแพทเทิร์นนี้มาใช้ในส่วนของตัวลูกเต๋า โดยค่าของลูกเต๋านี้จะเปรียบเหมือนกับว่ามีอยู่เพียงค่าเดียวในเกม สามารถใช้ได้ทีละคน คือ เมื่อถึงลำดับของผู้เล่นคนใดแล้ว ผู้เล่นคนนั้นๆจึงจะมีอภิสิทธิ์ในการทอดลูกเต๋าได้ และเดินตัวเดินตามค่าที่ได้จากการทอดลูกเต๋า โดยที่ผู้เล่นคนอื่นๆ ที่ยังไม่ถึงลำดับของตนนั้น จะไม่สามารถทำการทอดลูกเต๋าได้ และไม่สามารถนำค่าของลูกเต๋าที่ได้จากการทอดลูกเต๋าของผู้อื่นไปใช้ในการเดินตัวเดินของตนเองได้
- Iterator Pattern เราจะนำแพทเทิร์นนี้ไปใช้ในส่วนของลำดับการเล่นของผู้เล่น โดยจะเป็นการจัดลำดับว่าผู้เล่นแต่ละคนต้องเริ่มจากจุดเริ่มต้นของกระดาน ทำการทอดลูกเต๋า 1 ครั้ง เพื่อที่จะเดินตัวเล่นไปตามค่าที่ได้จากการทอดลูกเต๋า หากตกในช่องคำถามก็จะต้องตอบคำถาม ถ้าตอบถูกก็จะได้คะแนนเพิ่ม แล้วก็จะถึงลำดับของผู้เล่นคนต่อไป โดยที่จะทำเป็นลำดับตามขั้นตอนนี้ไปทุกครั้ง และต้องทำทุกคน
- Observer Pattern ในส่วนที่เราจะนำแพทเทิร์นนี้ไปใช้คือ ส่วนของการ update แต่ละหน้าจอของผู้เล่นให้มีการแสดงผลต่างๆเหมือนกัน คือหากผู้เล่นคนใดคนหนึ่งทำการทอดลูกเต๋า ผู้เล่นคนอื่นๆต้องสามารถเห็นได้ว่าค่าที่ได้จากลูกเต๋าเป็นอะไร หรือ หากผู้เล่นคนใดคนหนึ่งทำการเดินตัวเล่นไปตามบล็อคแล้ว ผู้เล่นคนอื่นๆก็ต้องสามารถเห็นได้ว่าตัวเดินของผู้เล่นคนนั้นๆได้มีการเดินไปที่ตำแหน่งใด และหากผู้เล่นคนใดสามารถตอบคำถามได้ถูก คะแนนของผู้เล่นคนนั้นก็จะต้องถูก update ในทุกๆหน้าจอของผู้เล่นแต่ละคนด้วย โดยที่เมื่อถึงลำดับการเล่นของผู้เล่นคนใดแล้วผู้เล่นคนนั้นก็จะเปรียบเสมือนตัวเองเป็น subject และผู้เล่นคนอื่นๆจะเปรียบเสมือนเป็น observer
- State Pattern จะมีการนำแพทเทิร์นนี้มาใช้ในหลายๆส่วนของซอร์ฟแวร์ คือ เมื่อถึงลำดับของผู้เล่นคนใดแล้วตัวเดินของผู้เล่นคนนั้นก็จะมีการเปลี่ยนสถานะเพื่อแสดงให้เห็นว่าถึงลำดับของผู้เล่นคนนั้นแล้ว เช่น อาจจะมีการกระพริบของตัวเล่น หรืออาจใช้การแสดงข้อความ เป็นต้น

การแสดงสถานะของการทอดลูกเต๋า เมื่อมีการทอดลูกเต๋าก็จะมีการแสดงด้วยการหมุนของตัวลูกเต๋าเพื่อเป็นการรับรู้ว่าได้ทำการทอดลูกเต๋าแล้ว ในส่วนต่อไปคือการเปลี่ยนสถานะจากการรอคอย เป็นการเดินเมื่อมีการทอดลูกเต๋าแล้ว และเมื่อเดินไปครบจนครบตามจำนวนค่าที่ได้จากลูกเต๋าแล้วก็จะกลับมายัง สถานะของการรอคอยอีกครั้งหนึ่ง

5.6.4 การเขียนโปรแกรมและการทดสอบ

ในหัวข้อนี้จะเป็นการทำงานในส่วนของการเขียนโปรแกรมทั้งหมด โดยจะขอกล่าวถึงขั้นตอนการทำงานของโปรแกรมในบทต่อไป

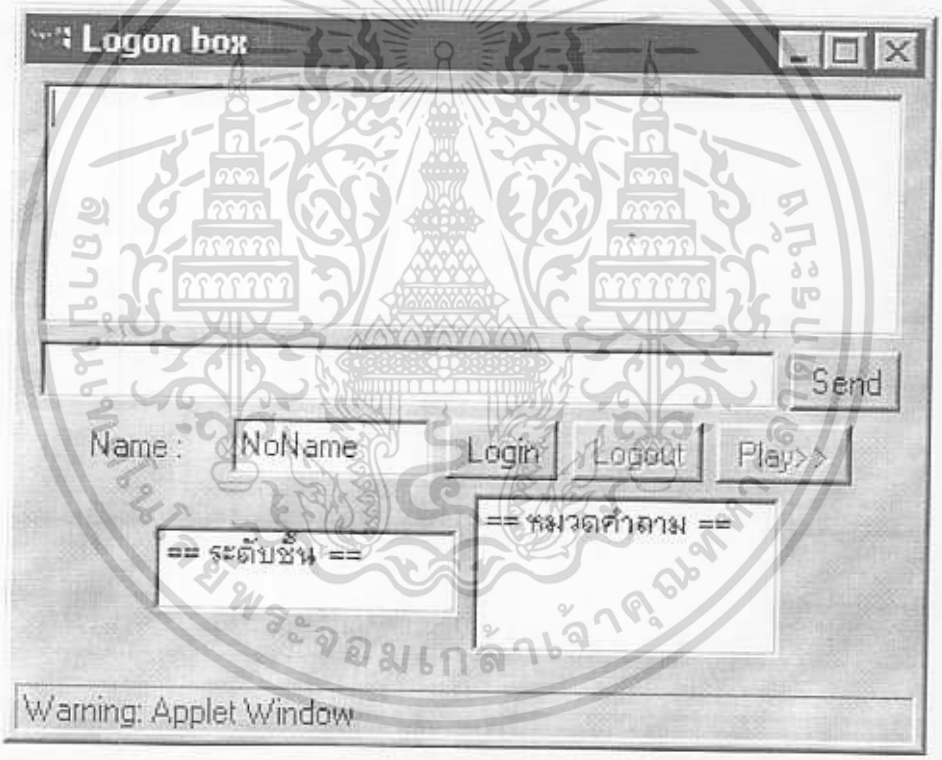
บทที่ 6
การทำงานของโปรแกรม

6.1 การทำงานของโปรแกรมทั้งหมด

ในส่วนของโปรแกรมทั้งหมดนั้น อาจจะแบ่งการทำงานออกได้เป็น 3 ส่วนหลักๆด้วยกันได้ดังนี้

- ส่วนของ dialog
- โปรแกรมในส่วนของไคลเอนต์
- โปรแกรมในส่วนของเซิร์ฟเวอร์

โดยเมื่อมีการเรียกแอพเพล็ต ของโปรแกรมขึ้น ตัวโปรแกรมจะทำการเรียกโปรแกรมย่อยในส่วนของ dialog ซึ่งไดอะล็อคนี้จะมีหน้าที่ในการควบคุมการติดต่อของผู้เล่นกับเซิร์ฟเวอร์ คือจะเป็นตัวที่ใช้ในการ login และ logout ทั้งยังมีหน้าที่ในการรับชื่อของผู้เล่น, เลือกหมวดคำถาม และการส่งข้อความคุยกันของตัวผู้เล่นเอง โดยจะมีหน้าตาดังนี้

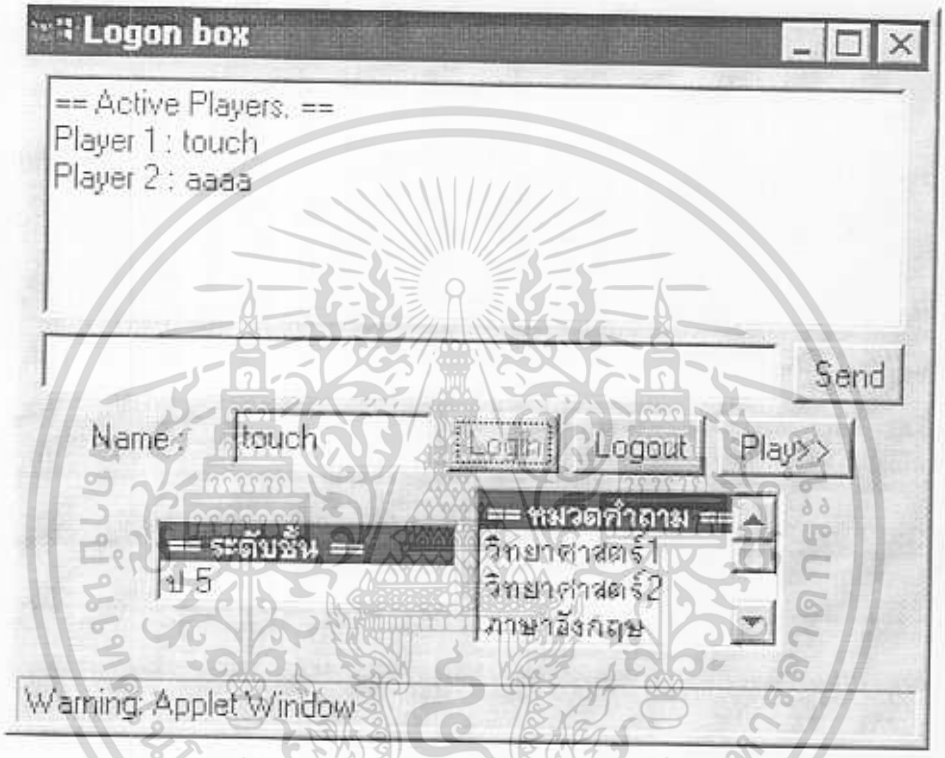


รูปที่ 6.1 แสดงหน้าจอส่วนของ dialog

โดยไดอะล็อคนี้จะรอให้ผู้เล่นกรอกชื่อในช่องใส่ชื่อ (Name:) ที่มีข้อความไว้ว่า 'NoName' และจะรอจนกว่าผู้เล่นจะกดปุ่ม 'Login' เมื่อผู้เล่นกดปุ่มล็อกอินแล้ว โปรแกรมก็จะทำการเชื่อมต่อไปยังตัวโปรแกรมในส่วนของเซิร์ฟเวอร์ โดยจะส่งชื่อของผู้เล่นที่กรอกไว้ไปเก็บไว้ที่ฝั่งของเซิร์ฟเวอร์ด้วย และก็จะส่งข้อความตอบรับการเชื่อมต่อกลับมาด้วยรายชื่อของผู้เล่นที่เชื่อมต่ออยู่ทั้งหมดรวมทั้งผู้เล่นที่เพิ่งทำการเชื่อมต่อนี้ด้วยไปให้แก่ผู้เล่นทั้งหมด ซึ่งในระหว่างนี้ถึงผู้เล่นจะยังไม่ได้เริ่มเล่นเกมก็จะสามารถส่งข้อความพูดคุย

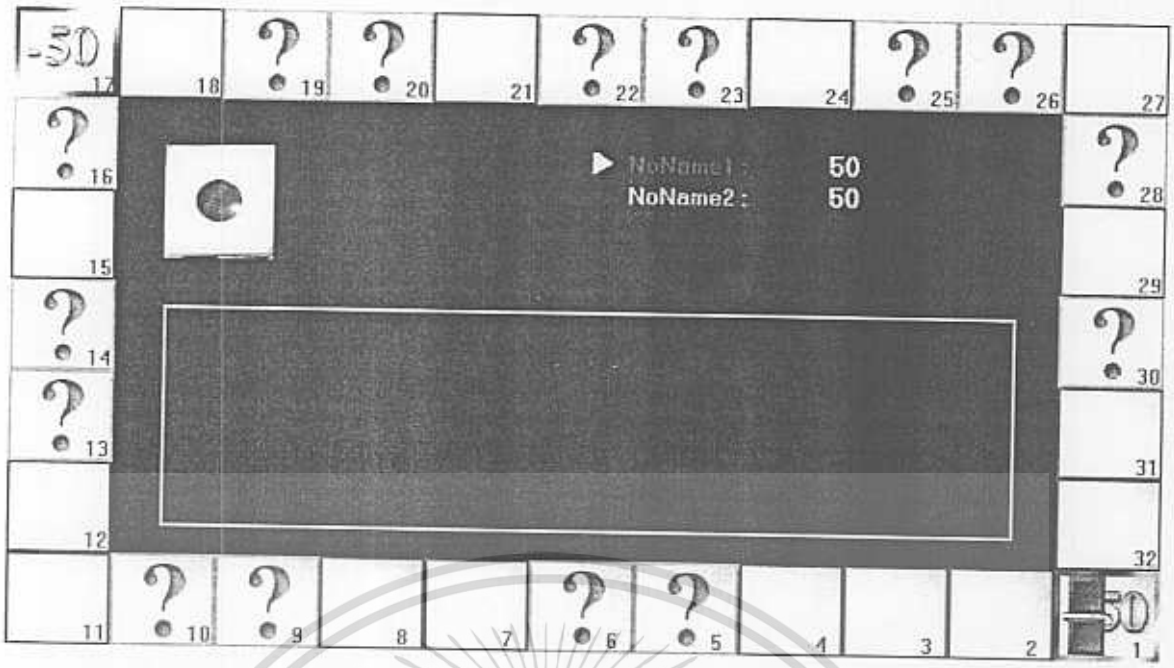
กันได้ โดยการกรอกข้อความที่จะพูดคุยลงในช่องที่กำหนดไว้ เมื่อเรียบร้อยแล้วให้กดปุ่ม 'Send' ข้อความที่กรอกไว้แล้วก็จะถูกส่งไปที่เซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์ก็จะทำการกระจายข้อความนี้กลับมาให้ทั้งตัวเราและผู้เล่นคนอื่นๆ โดยสามารถรองรับการใช้ภาษาไทยได้ด้วย

จากนั้นโปรแกรมจะอนุญาตให้ผู้เล่นที่ทำการเชื่อมต่อเป็นแรกเท่านั้นที่จะกดปุ่ม 'Play>>' เพื่อเริ่มเล่นเกมได้ และโปรแกรมก็จะเลือกรับหมวดคำถามจากผู้เล่นที่กดปุ่มนี้เท่านั้น ซึ่งในระหว่างนี้หากผู้เล่นคนแรกกดปุ่ม 'Logout' เพื่อออกจากการเชื่อมต่อไป เซิร์ฟเวอร์ก็จะทำการเลื่อนรายชื่อของผู้เล่นทั้งหมดขึ้นและจะส่งรายชื่อของผู้เล่นทางหมดให้แก่โปรแกรมในส่วนของไคลเอนต์ใหม่



รูปที่ 6.2 แสดงหน้าจอหลังเชื่อมต่อแล้ว

ภายหลังจากที่ผู้เล่นคนแรกเลือกหมวดคำถาม และกดปุ่มเพื่อเริ่มเล่นเกมแล้ว เซิร์ฟเวอร์ก็จะทำการตั้งค่าเริ่มต้น (initial) ของตัวแผนที่ โดยจะทำการสุ่มว่าจะให้บล็อคไหนเป็นบล็อคอะไร และจะทำการอ่านคำถามในหมวดที่เลือกทั้งหมดจากฐานข้อมูลมาเก็บไว้ในไฟล์ๆหนึ่ง เพื่อที่จะช่วยให้รวดเร็วขึ้นในการสุ่มคำถามขณะเล่นเกม จากนั้นเซิร์ฟเวอร์ก็จะส่งข้อความมาเพื่อบอกให้เริ่มเกมได้



รูปที่ 6.3 แสดงหน้าจอหลังกดปุ่มเริ่มเล่นเกม

จากนั้นก็จะเป็นการทำงานในส่วนของตัวเกม โดยจะเริ่มลำดับของผู้เล่นจากลำดับการเชื่อมต่อของผู้เล่นทั้งหมด และเนื่องจากรูปแบบของเกมเป็นแบบลำดับการเล่นทีละคน จึงอนุญาตให้มีเพียงผู้เล่นกระทำกับเกมเพียงคนเดียว เกมจะรอรับอีมพุดจากผู้เล่นที่ละคน โดยจะเริ่มจากผู้เล่นที่ทำการเชื่อมต่อเป็นคนแรกก่อน เมื่อผู้เล่นคนแรกทำการคลิกเมาส์ในบริเวณพื้นที่เป็นรูปลูกเต๋าแล้ว โปรแกรมในฝั่งไคลเอนต์ก็จะทำการสุ่มค่าลูกเต๋าขึ้นมาหนึ่งค่า แล้วจะส่งค่าของลูกเต๋าไปให้แก่เซิร์ฟเวอร์ เพื่อให้เซิร์ฟเวอร์กระจายค่านี้ไปให้แก่เครื่องไคลเอนต์ทุกๆเครื่องให้ได้รับค่าเดียวกัน จากนั้นไคลเอนต์ทุกตัวก็จะทำการเดินตัวเล่นของผู้เล่นคนแรกนี้ไปตามบล็อคตามค่าที่ได้จากลูกเต๋าที่เซิร์ฟเวอร์ส่งมาให้ เมื่อการเดินเสร็จสิ้นแล้วไคลเอนต์ทุกตัวก็จะทำการตรวจสอบว่าบล็อคสุดท้ายที่ไปตกนั้นเป็นบล็อคชนิดใด แต่จะมีเพียงไคลเอนต์เครื่องที่เป็นของผู้คนที่คนแรกนี้เท่านั้นที่สามารถจะส่งข้อความติดต่อกับเซิร์ฟเวอร์ได้ ซึ่งหากบล็อคที่ตกสุดท้ายนั้นเป็นบล็อคธรรมดานั้น ไคลเอนต์ก็จะส่งข้อความไปบอกแค่เซิร์ฟเวอร์ให้เซิร์ฟเวอร์ทำการจัดลำดับของผู้เล่นคนต่อไปมาแทน แต่หากบล็อคที่ตกเป็นบล็อคสุดท้ายนั้นเป็นบล็อคคำถาม ไคลเอนต์ก็จะส่งข้อความบอกเซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์ก็จะส่งข้อความรับรู้ด้วยรายชื่อของผู้เล่นที่เชื่อมต่ออยู่กลับมา เพื่อให้ผู้เล่นเลือกเอาว่าจะตอบคำถามเองหรือว่าให้ใครเป็นผู้ตอบคำถาม

รูปที่ 6.4 แสดงหน้าจอการตกในบล็อกคำถาม

เมื่อผู้เล่นคนแรกนี้ทำการกดคีย์เพื่อเลือกผู้ตอบคำถามได้แล้วไคลเอนต์ก็จะส่งข้อความไปบอกแก่เซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์ก็จะตอบรับโดยการส่งข้อความแสดงสถานะของผู้ตอบคำถามและสุ่มคำถามหนึ่งข้อพร้อมเฉลยจากไฟล์ไปให้เครื่องไคลเอนต์ทุกตัว เครื่องไคลเอนต์ก็จะรอรับการกดคีย์เพื่อตอบคำถามจากผู้เล่นที่ถูกเลือก

รูปที่ 6.5 แสดงหน้าจอคำถาม

เมื่อผู้เล่นตอบคำถามแล้วเครื่องไคลเอนต์ก็จะทำการตรวจคำตอบในตัวเลย แล้วส่งผลลัพธ์ที่ได้ไปให้แก่เซิร์ฟเวอร์เพื่อให้เซิร์ฟเวอร์กระจายผลลัพธ์นี้กลับมาให้แก่ไคลเอนต์ทุกตัว เมื่อเครื่องไคลเอนต์ได้รับผล

แล้วก็จะทำการอัพเดทคะแนนของผู้เล่นคนที่ตอบคำถามนั้นๆในเครื่องของไคลเอนต์เอง จากนั้นไคลเอนต์ก็จะส่งข้อความไปบอกแก่เซิร์ฟเวอร์ว่าได้เสร็จสิ้นการเดินของผู้เล่นคนแรกนี้แล้ว เซิร์ฟเวอร์ก็จะส่งทำการจัดลำดับของผู้เล่นใหม่คือเปลี่ยนไปให้สิทธิแก่ผู้เล่นที่เชื่อมต่อเป็นคนถัดไปได้เล่นแทน จากนั้นก็จะเข้าสู่การทำงานในรูปแบบเดิมไปจนกระทั่งจบเกม โดยในระหว่างนี้หากมีผู้เล่นคนไหนทำการกดปุ่ม 'Logout' เพื่อออกจากการเชื่อมต่อไปในขณะเล่นเกมอยู่ เซิร์ฟเวอร์ก็จะทำการตัดลำดับการเล่นของผู้เล่นคนนั้นออกไปได้เลยโดยไม่มีผลกระทบอื่นใด และจะส่งข้อความไปบอกแก่ไคลเอนต์ทุกตัวให้ทราบถึงการออกจากการเชื่อมต่อของผู้เล่นคนนั้นๆด้วย เมื่อไคลเอนต์ได้รับแล้วก็จะทำการตัดค่าตัวแปรที่ใช้เก็บสถานะต่างๆของผู้เล่นคนนั้นออกไปจะระบบได้โดยไม่มีผลกระทบกับผู้เล่นที่เหลือเลย

เมื่อจบเกมไคลเอนต์เครื่องที่เป็นของผู้เล่นคนสุดท้ายจะส่งข้อความแสดงรายชื่อและคะแนนของผู้เล่นทุกคนไปให้แก่เซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์ก็จะทำการอ่านรายชื่อและคะแนนเก่าๆจากไฟล์ 'history' มาแล้วจัดลำดับตามคะแนนใหม่ทั้งหมด โดยจะคิดเฉพาะผู้เล่นที่คะแนนไม่ติดลบเท่านั้นและจะตัดที่เพียง 10 คนเท่านั้น แล้วส่งข้อความไปบอกแกเครื่องไคลเอนต์ทุกเครื่อง เครื่องไคลเอนต์เมื่อได้รับข้อความนี้แล้วก็จะแสดงลำดับคะแนนสูงสุดตามที่ฝั่งเซิร์ฟเวอร์ได้เรียงลำดับไว้แล้วเหมือนกันทุกเครื่อง จากนั้นเครื่องไคลเอนต์ที่เป็นผู้เล่นคนแรกจะสามารถกดปุ่ม 'Play>>' เพื่อทำการเริ่มเกมใหม่ได้ต่อไป

รูปที่ 6.6 แสดงหน้าจอลำดับคะแนนสูงสุด

Command&&Information

รูปแบบการส่งข้อความที่ใช้ติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องไคลเอนต์ โดยในส่วนของ Information นั้นอาจจะซ่อนด้วยคำสั่งหรือเป็นข้อมูลที่อาจแตกย่อยลงไปเป็นคำสั่งย่อยอีกก็ได้ โดยเซิร์ฟเวอร์จะสามารถรับรู้ได้จาก command ว่าเป็นคำสั่งใด ซึ่งทั้งหมดนี้จะเป็นการรับส่งในรูปแบบของข้อความ

(String) แล้วนำมาตัดต่อออกตามเครื่องหมาย && ที่คั่นไว้ แล้วจึงแยกได้เป็น คำสั่ง และ ข้อมูล ดังที่ได้กล่าวแล้วนั้น เพื่อนำไปประมวลผลต่อไป

Index	ระดับชั้น	หมวดคำถาม

หมวดคำถาม	problem	Choice 1	Choice 2	Choice 3	Choice 4	answer

รูปที่ 6.7 ชนิดของฐานข้อมูล

ในส่วนของฐานข้อมูลนั้นจะแบ่งได้เป็น 2 ตารางใหญ่ๆ ดังยกัน คือ ตาราง Index กับตาราง หมวดคำถาม โดยในตารางของหมวดคำถามนั้นจะมีการนำเอา ข้อมูลในตาราง Index มาตั้งเป็นชื่อของตาราง คือจะนำเอาชื่อของระดับชั้น และหมวดคำถามมาต่อกันกลายเป็นชื่อของตารางหมวดคำถาม เพื่อให้ในระดับชั้นที่ต่างกันสามารถมีหมวดคำถามชื่อเดียวกันได้

6.2 สรุปการทำงาน

ในการที่เราจะเขียนเกมให้สามารถเล่นผ่านระบบเน็ทเวิร์คนั้น จำเป็นที่จะต้องรู้หรือมีการจัดลำดับขั้นตอนในการปฏิบัติ ได้ดังนี้

1. คือ เราควรที่จะต้องรู้ก่อนว่า เกมที่เราจะเขียนนั้นเป็นเกมประเภทไหน เพื่อที่เราจะได้สามารถนึกภาพออกได้ว่าตัวเกมจะมีลักษณะอย่างไร หรือมีรูปแบบวิธีการเล่นอย่างไร เหมาะกับผู้เล่นประเภทใด
2. เราจะต้องรู้ว่าเราจะใช้โปรแกรมภาษาอะไรในการพัฒนาเกมนั้น เนื่องจากว่าลำดับการคิดของการเขียนโปรแกรมในแบบทั่วไป กับในแบบ ออบเจ็ค-โอเรียนเท็ด จะไม่เหมือนกัน และเราควรที่จะต้องศึกษาการใช้โปรแกรมภาษานั้นมาก่อนแล้ว เพื่อให้ง่ายในข้อต่อไป
3. เมื่อเราสามารถกำหนดรูปแบบของเกมและเลือกภาษาที่จะใช้ได้แล้ว เราควรที่จะต้องลำดับหน้าที่การทำงานของเกม เช่น หากมีฉากหลังที่เลื่อนได้ เราจะต้องลำดับการวาดรูปอย่างไร เป็นต้น หรืออาจจะต้องเขียนออกมาเป็นไดอะแกรมเพื่อที่เราจะได้สามารถเห็นภาพได้ง่ายขึ้น และควรที่จะแบ่งหน้าที่การทำงานออกเป็นส่วนๆ เพื่อที่ว่าเราจะได้สามารถทำการเขียนโปรแกรมได้ง่าย ซึ่งในหัวข้อนี้มีความสำคัญมากเนื่องจากหากกระทำอย่างไม่รอบคอบแล้วจะเกิดปัญหาได้ในภายหลัง
4. การเขียนโปรแกรม โดยเราควรที่จะแบ่งออกเป็นส่วนๆ เพื่อให้สะดวกต่อการเขียน หากเป็นภาษาจำพวกออบเจ็ค-โอเรียนเท็ดแล้วจะทำให้สะดวกต่อการเขียนแยกเป็นส่วนๆ โดยเราควรจะเริ่มจากส่วนที่ง่ายไม่ซับซ้อนก่อน แต่ต้องเขียนให้รองรับในส่วนที่จะเพิ่มในภายหลังได้ ซึ่งจุดนี้จะเป็นส่วนที่กระทำได้ยากมาก และจะยากยิ่งขึ้นหากเราไม่ทำในหัวข้อที่ 3 ให้ดีเสียก่อน

5. ในระหว่างที่เราเขียนโปรแกรมอยู่นั้น หากเราเขียนโปรแกรมส่วนไหนเสร็จแล้วก็ควรที่เราจะต้องทำการทดสอบเพื่อหาข้อผิดพลาดในส่วนนั้นๆ เสียก่อน เพื่อที่เวลานำทุกส่วนมารวมกันแล้วจะทำให้มีเปอร์เซ็นต์ความผิดพลาดที่ลดน้อยลงไปอีก
6. เมื่อเกมเสร็จแล้ว หรืออยู่ในระหว่างการทำการทดสอบ เราควรที่จะมีการให้คนอื่นๆ เป็นผู้ร่วมทดสอบด้วยเพื่อที่เราจะได้เห็นจุดผิดพลาดที่เราอาจมองข้ามไปว่ามันน่าจะเกิด หรืออาจมีจุดไหนที่เราควรต้องเพิ่มเติม ก็จะทำให้เกมของเรานั้นมีความน่าเล่นเพิ่มขึ้นไปอีก

ภาคผนวก ก.

IIS

IIS (Internet Information Server)

Windows NT ได้จัดเตรียมการใช้งานโปรโตคอล TCP/IP ไว้ให้อย่างครบถ้วนได้ปรับปรุงให้โปรโตคอลพื้นฐานต่างๆของไมโครซอฟท์เอง เช่น NetBIOS หรือ NetBEUI สามารถใช้งานร่วมกับโปรโตคอล TCP/IP ได้อย่างกลมกลืน ดังนั้นเพื่อให้ Windows NT รองรับการทำงานของ อินเตอร์เน็ตได้อย่างสมบูรณ์แบบ Windows NT จึงได้จัดเตรียมซอฟต์แวร์ที่เรียกว่า IIS หรือ Internet Information Server ขึ้นมาเป็นส่วนหนึ่งของ Windows NT เพื่อใช้เป็นเว็บเซิร์ฟเวอร์ IIS เป็นซอฟต์แวร์ที่ให้มาคู่กันกับ Windows NT โดยจะต้องระบุในขณะที่ติดตั้ง Windows NT และเมื่อเปิดเครื่องเซิร์ฟเวอร์ครั้งถัดไปเซอร์วิสของ IIS ก็จะถูกเรียกขึ้นมาใช้งานได้โดยอัตโนมัติ ซึ่งสามารถเรียกตรวจสอบได้จากไอคอน Service ที่ Control Panel จะเห็นว่ามีเซอร์วิสของ IIS Admin Service และ FTP Publishing Service ซึ่งเป็นบริการหนึ่งของ IIS ทำงานอยู่

IIS ทำหน้าที่ให้บริการอินเตอร์เน็ตในรูปแบบต่างๆ เช่น เป็น FTP Server หรือ เว็บเซิร์ฟเวอร์ด้วย ในการกำหนดรายละเอียดต่างๆของ IIS จะทำได้โดยใช้ Microsoft Management Console หรือ MMC เพื่อกำหนดรายละเอียดต่างๆของ Windows NT ในหลายๆส่วน และ IIS ก็เป็นส่วนหนึ่งที่กำหนดโดยใช้ MMC ด้วย (MMC ใน Service Pack 4 จะอยู่ในเมนู Windows NT 4.0 Option Pack ที่ปุ่ม Start)

การกำหนดรายละเอียดที่สำคัญของ IIS คือการกำหนดคุณสมบัติต่างๆของบริการอินเตอร์เน็ตแต่ละประเภท โดยเลือกที่ Default FTP Site หรือ Default Web Site แล้วคลิกไอคอน Properties บนเมนูบาร์ หรือ คลิกเมาส์ขวาแล้วเลือกคำสั่ง Properties จากเมนูลัดก็ได้

การกำหนดคุณสมบัติต่างๆใน Default FTP Site Properties มีรายละเอียดเบื้องต้นดังต่อไปนี้

FTP Site	ใช้กำหนรายละเอียดทั่วไปของเซิร์ฟเวอร์ FTP เช่น หมายเลขพอร์ตที่ใช้งาน หรือจำนวนผู้ใช้สูงสุดที่ยอมให้ติดต่อเข้ามาพร้อมกัน เป็นต้น
Security Accounts	ใช้กำหนดรายละเอียดเกี่ยวกับผู้ใช้งาน ซึ่งสัมพันธ์กับรายละเอียดของผู้ใช้ใน Windows NT เช่น ยอมให้ผู้ใช้ติดต่อเข้ามาโดยไม่ใส่ชื่อและรหัสผ่าน (Allow Anonymous Connections) เป็นต้น
Message	ใช้กำหนดข้อความต่างๆที่ต้องการให้แสดงในขณะที่เริ่มติดต่อเข้ามาใช้งาน หรือข้อความที่แสดงเมื่อเลิกใช้งาน
Home Directory	ใช้กำหนดรายละเอียดเกี่ยวกับโฮมไดเร็คทอรี่เมื่อเริ่มติดต่อใช้งาน FTP, การกำหนดสิทธิการใช้งาน และการแสดงรายชื่อของไดเร็คทอรี่นั้นงานต้องการรูปแบบเป็น Unix หรือ MS-DOS

Directory Security	ใช้ระบุชื่อของไดเร็คทอรี่ย่อยที่อยู่ภายใต้โฮมไดเร็คทอรี่, การใช้สิทธิผู้ใช้งาน (Granted Access) หรือถอนสิทธิไม่ให้ผู้ใช้เข้ามาใช้งาน (Denied Access)

การกำหนดคุณสมบัติต่างๆ ใน Default Web Site Properties มีรายละเอียดเบื้องต้นดังต่อไปนี้

Web Site	ใช้กำหนดรายละเอียดทั่วไปของเว็บเซิร์ฟเวอร์ เช่น หมายเลขพอร์ตที่ใช้งาน หรือจำนวนผู้ใช้งานที่ยอมให้ติดต่อเข้ามาพร้อมกัน เป็นต้น
Operators	ใช้กำหนดรายละเอียดเกี่ยวกับผู้งาน Windows NT ที่ทำหน้าที่ควบคุมเว็บเซิร์ฟเวอร์
Performance	ใช้กำหนดและปรับประสิทธิภาพของเว็บเซิร์ฟเวอร์
ISAPI Filters	ใช้กำหนดรายละเอียดของไฟล์ DLL ที่เรียกใช้ตามกลไกของ ISAPI
Home Directory	ใช้กำหนดโฮมไดเร็คทอรี่ของเว็บเซิร์ฟเวอร์ ซึ่งจะเก็บไฟล์ HTML ของโฮมเพจแรก เมื่อผู้ใช้เรียกผ่านบราวเซอร์เข้ามา
Documents	ใช้ระบุชื่อไฟล์ HTML ที่เป็น default (เช่น index.html) ซึ่งจะถูกเรียกใช้งานกรณีที่ผู้เข้ามายังเว็บไซท์นั้นไม่ได้ระบุชื่อไฟล์ HTML มาด้วยใน URL
Directory Security	ใช้กำหนดระดับการรักษาความปลอดภัยของการติดต่อเข้ามาใช้งานเว็บเซิร์ฟเวอร์ เช่น การกำหนดสิทธิการใช้งานของผู้ที่ไม่ระบุชื่อ (Anonymous Access), การใช้งานระบบรักษาความปลอดภัย SSL และการจำกัดการใช้งานตามหมายเลข IP
HTTP Header	ใช้กำหนดส่วนเฮดเดอร์ของโปรโตคอล HTTP รวมทั้งกำหนดระดับของเนื้อหา หรือเรตติ้งของเว็บไซท์ (Content Rating – เช่น เหมาะกับเด็กหรือผู้ใหญ่, มีเรื่องที่ผู้ปกครองควรพิจารณาเป็นพิเศษหรือไม่ เป็นต้น) ที่ได้จัดทำขึ้น
Custom Errors	ใช้กำหนดรหัสและรายละเอียดของปัญหาหรือความผิดปกติต่างๆ

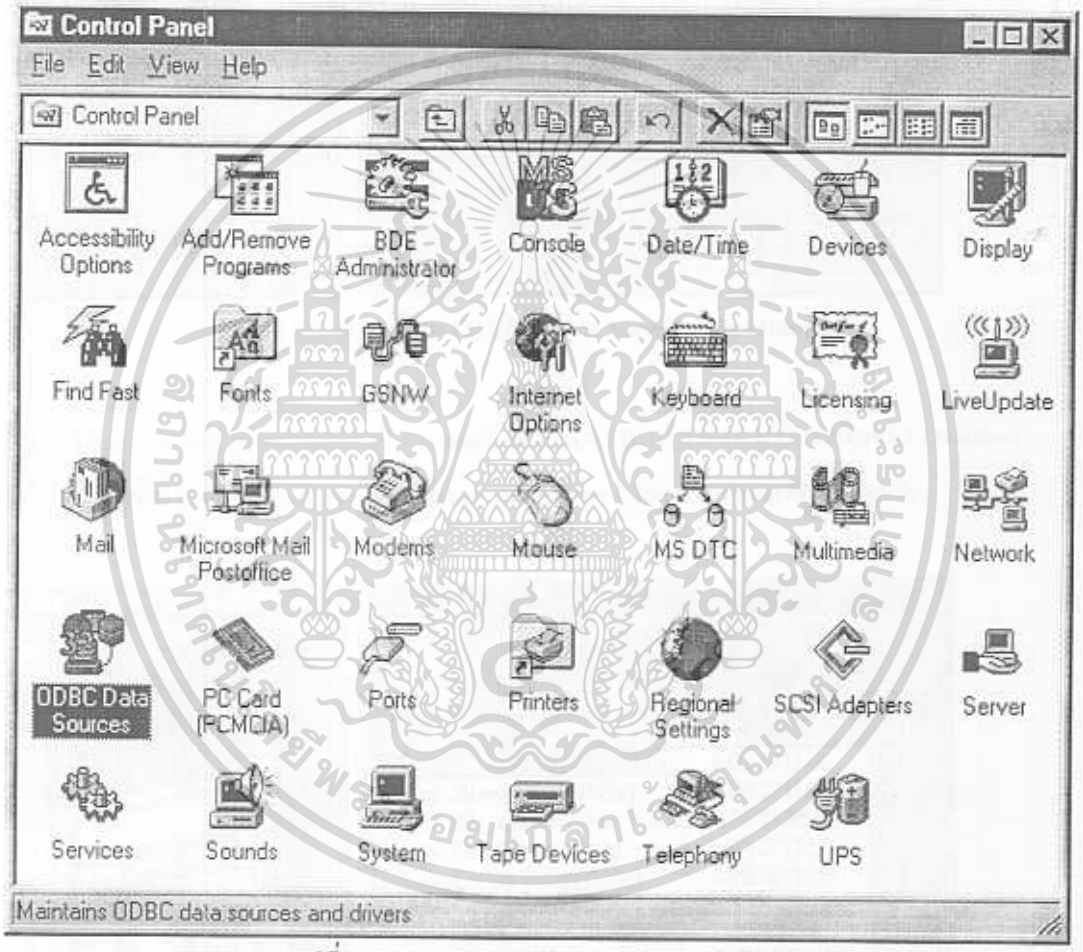
ภาคผนวก ข.

การ Setup ODBC

การ Setup ODBC บน Server ที่เป็นระบบ WindowNT

การติดต่อกับ Database ที่เราสร้างขึ้นมีหลายวิธีด้วยกันแต่วิธีที่ง่ายที่สุดก็คือการติดต่อโดยผ่าน ODBC ซึ่งมีขั้นตอนดังต่อไปนี้

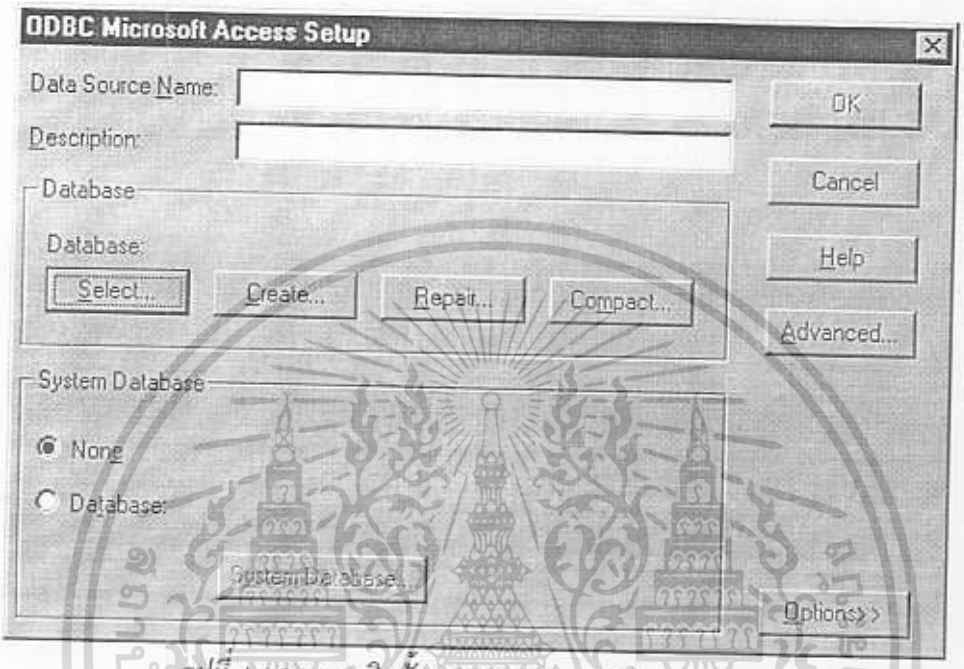
1. ทำการเปิด Folder ที่ชื่อ Control panel ขึ้นมาหลังจากนั้นดับเบิ้ลคลิ๊กไปที่ไอคอนของ ODBC ที่มีชื่อว่า ODBC Data Sources ดังรูป



รูปที่ 1 แสดง Folder ของ Control Panel ใน windowNT

2. หลังจากที่เราดับเบิ้ลคลิ๊กที่ไอคอนแล้วจะขึ้นไดอะล็อค ดังรูป เพื่อรอการตั้งค่าต่างๆ

4. หลังจากที่กดปุ่ม Finish แล้วให้ทำการ ใส่ชื่อของ Data Source ลงในช่องที่เขียนว่า Data Source Name ซึ่งชื่อของ Data Source จะใช้เป็นตัวกลางในการอ้างอิงเมื่อต้องการติดต่อกับ Database ส่วน Description นั้นจะป้อนหรือไม่ก็ได้ หลังจากนั้นให้กดปุ่ม Select จากนั้นให้ป้อนชื่อของ Database ที่ต้องการ Connect และ กดปุ่ม OK จะเป็นการเสร็จสิ้นการ Setup ODBC



รูปที่ 4 แสดงการติดตั้ง ODBC Microsoft Access Setup