บทที่ 2

ทฤษฎีพื้นฐาน

2.1 หลักการเชิงวัตถุ

ระบบที่สร้างด้วยหลักการเชิงวัตถุคือระบบที่ประกอบด้วยส่วนประกอบ (Component) ต่าง ๆ ที่ครอบคลุม (Encapsulate) กลุ่มของข้อมูลและฟังก์ชันไว้ภายในตัว ส่วนประกอบเหล่านี้สามารถสืบทอด (Inherit) คุณสมบัติ (Attribute) และพฤติกรรม (Behavior) จากส่วนประกอบอื่น และส่วนประกอบที่ประกอบกันขึ้นเป็นระบบนั้นสามารถติดต่อผ่านกันโดยใช้ข่าวสาร (Message)

2.1.1 คำนิยามของวัตถุและคลาส

มีหลายคนนิยามไว้คำนิยามของ "วัตถุ" และ "คลาส" ในที่นี้จะยกคำนิยามของนาย ปีเตอร์ คอร์ด (Mr.Peter Coad) และนายเอดเวิร์ด ยัวร์ดอน (Mr.Edward Yourdon) [3] ดังนี้

วัตถุ (Object) คือ สิ่งที่ใช้แทนบางสิ่งบางอย่างในปัญหาที่กำลังพิจารณา ซึ่งครอบคลุมข้อมูลและฟังก์ชัน

คลาส (Class) คือ กลุ่มของวัตถุตั้งแต่หนึ่งวัตถุขึ้นไป ซึ่งมีคุณสมบัติและพฤติกรรมเหมือนกัน รวมถึงวิธีการสร้างวัตถุของคลาสนั้นด้วย

2.1.2 คุณสมบัติสำคัญของหลักการเชิงวัตถุ

Abstraction คือ กลไกอย่างใดอย่างหนึ่งที่ทำให้เราสามารถแสดงความจริงบางสิ่งบางอย่างที่สลับซับซ้อนในรูปแบบของแบบจำลองที่ง่าย

Encapsulation คือ กลไกอย่างใดอย่างหนึ่งที่ทำให้เราสามารถซ่อนรายละเอียดที่ประกอบกันเป็นวัตถุ เพื่อให้ส่วนประกอบหรือวัตถุอื่น ๆ ในระบบไม่ต้องสนใจรายละเอียดที่เก็บอยู่ภายในวัตถุนั้น หมายความว่า ข้อมูลหรือคุณสมบัติของวัตถุ และฟังก์ชันที่จัดการกับข้อมูลนั้น (ซึ่งเรียกว่า "Method", "Operation" หรือ "Service") ถูกรวมไว้ด้วยกัน ซึ่ง

เป็นสิ่งสำคัญในการเขียนโปรแกรม, การออกแบบ และการวิเคราะห์ระบบ ระบบเชิงวัตถุ (Object-Oriented System) เป็นระบบที่เกิดจากการติดต่อกันระหว่างวัตถุ โดยใช้ข่าวสารซึ่งจะส่ง "วิธี" (Method) หรือ ฟังก์ชันที่อยู่ภายในวัตถุที่รับข่าวสารนั้น

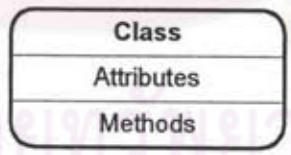
Inheritance คือ กลไกอย่างใดอย่างหนึ่งที่ทำให้วัตถุสามารถรวมเอาการนิยาม (Definition) ทั้งหมดหรือเพียงบางส่วนของวัตถุอื่น มาเป็นส่วนหนึ่งของการนิยามของวัตถุนั้นเอง ทำให้มีลำดับชั้นของวัตถุ (Hierarchies of Objects) เกิดขึ้นในระบบเชิงวัตถุ ซึ่งวัตถุที่อยู่ถัดลงมา (Childs หรือ Subordinate Objects) จะสืบทอดการนิยามทั้งหมดหรือเพียงบางส่วนของวัตถุที่อยู่ลำดับบน (Parents หรือ Superclasses)

Polymorphism คือ การตอบสนองที่แตกต่างกันของวัตถุต่างชนิดกันต่อข่าวสารชนิดเดียวกัน

2.2 สัญลักษณ์ที่ใช้ในการวิเคราะห์และออกแบบเชิงวัตถุ

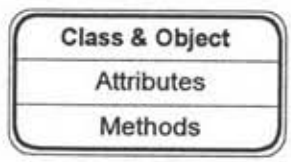
สัญลักษณ์ที่จะอธิบายต่อไปนี้ จะใช้ในการอธิบายการวิเคราะห์และออกแบบเชิงวัตถุของโปรแกรมจำลองการทำงานวงจรไฟฟ้า ซึ่งได้นำมาจาก [3] ซึ่งสรุปได้ดังนี้

2.2.1 สัญลักษณ์ของคลาส



รูปที่ 2.1 สัญลักษณ์ของคลาส

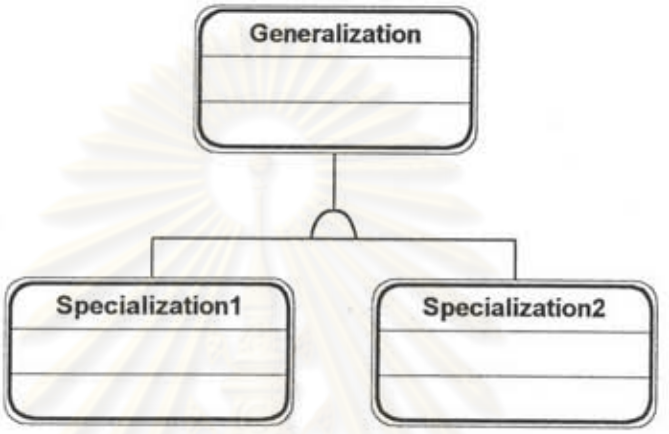
2.2.2 สัญลักษณ์ของคลาสและวัตถุ



รูปที่ 2.2 สัญลักษณ์ของคลาสและวัตถุ

สัญลักษณ์รูปที่ 2.2 บอกเป็นนัยว่าในระบบที่ออกแบบนั้นจะมีวัตถุของคลาสถูกสร้างขึ้นมาในระบบ

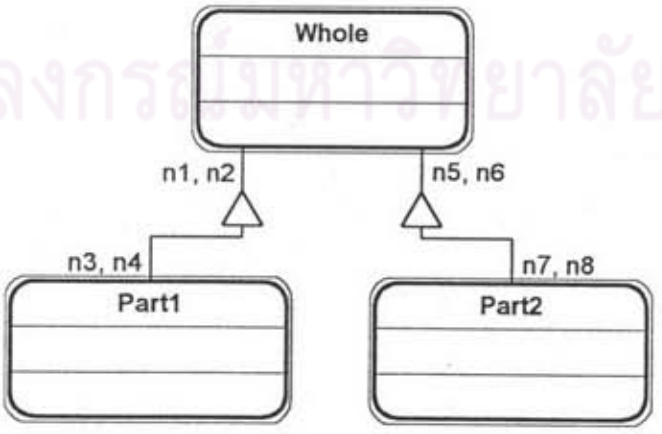
2.2.3 สัญลักษณ์ของลำดับชั้นแบบ Gen-Spec Structure



รูปที่ 2.3 สัญลักษณ์ของลำดับชั้นแบบ Gen-Spec Structure

รูปที่ 2.3 แสดงลำดับชั้นของคลาสและวัตถุซึ่งสืบทอดกันเป็นลำดับชั้น หมายความว่าคลาสและวัตถุ Specialization1 และ Specialization2 สืบทอดคุณสมบัติและพฤติกรรมจากคลาสและวัตถุ Generalization

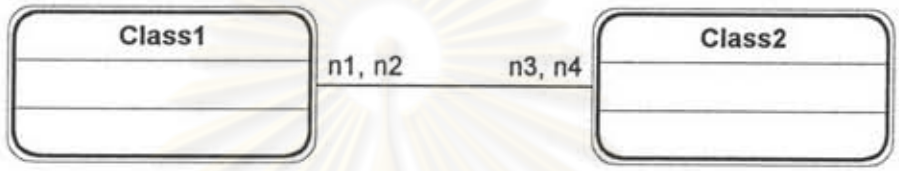
2.2.4 สัญลักษณ์ของลำดับชั้นแบบ Whole-Part Structure



รูปที่ 2.4 สัญลักษณ์ของลำดับชั้นแบบ Whole-Part Structure

สัญลักษณ์ในรูปที่ 2.4 อธิบายว่า แต่ละวัตถุของคลาส Whole ต้องประกอบด้วยวัตถุของคลาส Part1 ตั้งแต่ n1 ถึง n2 ชิ้น และวัตถุของคลาส Part2 ตั้งแต่ n5 ถึง n6 ชิ้น ในทางกลับกันแต่ละวัตถุของคลาส Part1 และ Part2 อาจจะเป็นส่วนประกอบของวัตถุของคลาส Whole ตั้งแต่ n3 ถึง n4 ชิ้น และ n7 ถึง n8 ชิ้นตามลำดับ

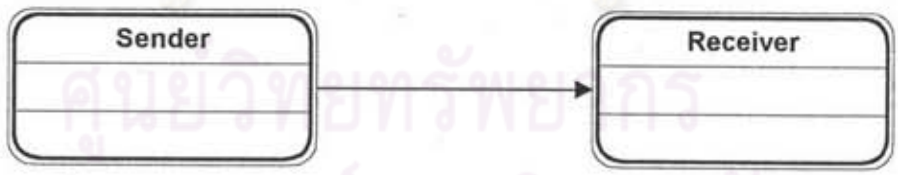
2.2.5 สัญลักษณ์ของความสัมพันธ์ระหว่างวัตถุ



รูปที่ 2.5 สัญลักษณ์ของความสัมพันธ์ระหว่างวัตถุ

สัญลักษณ์ในรูป 2.5 อธิบายว่า วัตถุของคลาส Class1 มีความสัมพันธ์กับวัตถุของคลาส Class2 มีจำนวนตั้งแต่ n1 ถึง n2 ชิ้น และวัตถุของคลาส Class2 มีความสัมพันธ์กับวัตถุของคลาส Class1 มีจำนวนตั้งแต่ n3 ถึง n4 ชิ้น

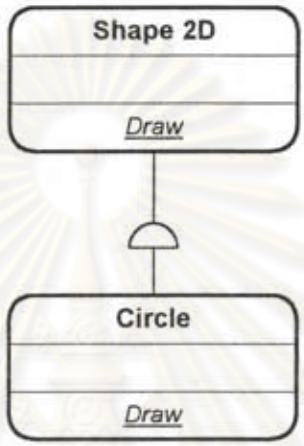
2.2.6 สัญลักษณ์ของการติดต่อระหว่างวัตถุ



รูปที่ 2.6 สัญลักษณ์ของการติดต่อระหว่างวัตถุ

สัญลักษณ์ในรูปที่ 2.6 แสดงการติดต่อกันระหว่างวัตถุของคลาส Sender กับวัตถุของคลาส Receiver โดยการส่งข่าวสาร ซึ่งเป็นการติดต่อกันระหว่างวัตถุเป็นแบบ 2 ทาง (2-way Communication) โดยที่วัตถุของคลาส Sender จะเป็นตัวเริ่มส่งข่าวสารก่อน ส่วนวัตถุของคลาส Receiver จะเป็นตัวรับข่าวสารแล้วตอบสนองต่อข่าวสารนั้น

* ***ข้อกำหนดเพิ่มเติม*** สำหรับสัญลักษณ์ของคลาสที่ใช้ในวิทยานิพนธ์นี้ มีดังนี้ คือ สำหรับในส่วนของวิธีในสัญลักษณ์ของคลาส ถ้าวิธีใดถูกเขียนด้วยตัวเอียงและขีดเส้นใต้ แสดงว่าวิธีนั้นเป็นฟังก์ชันเสมือน (Virtual Function) และถ้าถูกแสดงซ้ำอีกในคลาสที่สืบทอดอีกครั้ง หมายความว่ามีการกำหนดวิธีนั้นใหม่ในคลาสที่สืบทอดไป ดังตัวอย่าง



รูปที่ 2.7 ตัวอย่างสัญลักษณ์ของคลาสที่มีวิธีเป็นฟังก์ชันเสมือน

จากรูปที่ 2.7 เป็นลำดับชั้นของคลาส Shape 2D ซึ่งเป็นคลาสที่ครอบคลุมข้อมูลและวิธีการวาดรูป 2 มิติ จากรูปนี้แสดงให้เห็นว่า คลาส Shape 2D ได้กำหนดวิธีการวาด Draw ไว้แล้ว และคลาสที่สืบทอดจากคลาส Shape 2D คือ คลาส Circle ซึ่งได้กำหนดวิธีในการวาดใหม่ให้วาดรูปเป็นรูปวงกลมแทนวิธีการวาดที่ถูกกำหนดโดยคลาส Shape 2D

2.3 การวิเคราะห์และออกแบบเชิงวัตถุ [4]

2.3.1 การค้นหาวัตถุในระบบ ดังจะกล่าวเป็นข้อ ๆ ดังนี้

1. โดยมองจากข้อมูล สามารถหาวัตถุได้จากที่เหล่านี้

1.1 มองที่ตัวระบบที่กำลังจะออกแบบว่ามีข้อมูลอะไรบ้าง โดยพิจารณาร่วมกับแผนผัง, รูปภาพ และข้อมูลที่เป็นตัวอักษรจากผู้ใช้

1.2 มองที่ระบบอื่นที่ต้องติดต่อกับระบบที่กำลังสร้างแบบจำลอง

1.3 มองที่อุปกรณ์ทางกายภาพที่จะมีในสภาวะแวดล้อม (Environment) และการติดต่อกับระบบ โดยไม่คำนึงถึงเทคโนโลยีที่จะนำมาใช้สร้างระบบ

1.4 มองที่เหตุการณ์ (Event) ที่ระบบจะต้องจำและเก็บไว้

1.5 มองที่ผู้ใช้งานระบบนั้นหลาย ๆ คน ซึ่งมีวิธีการติดต่อกับวัตถุที่แตกต่างกัน

1.6 มองที่สถานที่และบริเวณที่เกี่ยวข้องกับระบบ

2. โดยมองจากหน้าที่ โดยพิจารณาว่า "วัตถุอะไรที่จะทำหน้าที่นี้ในระบบ" และสมควรหรือไม่ที่จะมีวัตถุนี้ในแบบจำลองของระบบ

3. โดยพิจารณาจากพฤติกรรมของวัตถุว่า วัตถุมีการติดต่อกับวัตถุตัวใดและตอบสนองต่อข่าวสารอย่างไร

เมื่อหาวัตถุเหล่านั้นได้แล้วจะต้องพิจารณาว่าวัตถุเหล่านั้นควรจะมีในแบบจำลองของระบบหรือไม่ แล้วเขียนความสัมพันธ์ระหว่างคลาสและวัตถุ ในรูปของลำดับชั้นของคลาส

2.3.2 สมบัติของวัตถุ (Object Attribute)

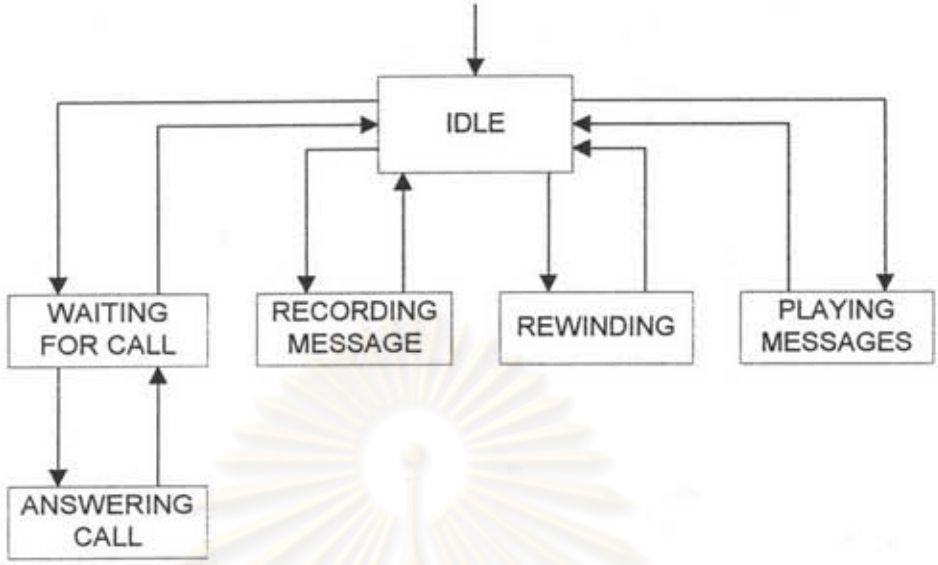
เป็นข้อมูลที่ซ่อนอยู่ภายในวัตถุ ซึ่งมองไม่เห็นจากภายนอก และคุณสมบัติของวัตถุจะบอกถึงสถานะ (State) ของวัตถุ ในขณะที่ วิธี (Method) จะแสดงถึงพฤติกรรมของวัตถุ (Object Behavior) สมบัติของวัตถุสามารถหาได้โดยศึกษาระบบที่กำลังจะออกแบบ, ถามผู้ใช้ระบบ และพยายามเรียนรู้ลักษณะของแต่ละคลาสและวัตถุในแบบจำลอง ใส่คุณสมบัติเหล่านั้นลงในสัญลักษณ์ของคลาสแต่ละคลาส

2.3.3 พฤติกรรมของวัตถุ (Object Behavior)

แสดงให้เห็นถึงการส่งข่าวสารให้กับวัตถุตัวอื่นทำให้มีการเปลี่ยนสถานะของวัตถุ และสามารถแสดงพฤติกรรมของวัตถุโดยใช้แผนผัง Object Life-History ซึ่งส่วนประกอบหลักคือ กรอบสี่เหลี่ยมแทน *สถานะ* และลูกศรแทน *การเปลี่ยนสถานะ* ดังตัวอย่างในรูปที่ 2.8 เป็นแผนผัง Object Life-History ของเครื่องตอบรับโทรศัพท์

2.3.4 การบริการของวัตถุ (Object Service) หรือ วิธีของวัตถุ (Object Method)

ซึ่งคำว่า "Service" จะสื่อความหมายได้ดีสำหรับผู้ใช้ที่ไม่มีความรู้ทางเทคนิค ส่วนคำว่า "Method" นั้นจะใช้ในแง่ของการโปรแกรมเชิงวัตถุ การบริการของวัตถุเป็นกระบวนการทำอะไรบางอย่างเมื่อวัตถุได้รับข่าวสาร



รูปที่ 2.8 แผนผัง Object Life-History ของเครื่องตอบรับโทรศัพท์

2.3.5 การออกแบบคลาส

เมื่อหาคลาสและวัตถุต่าง ๆ ในระบบ, ลำดับชั้นของคลาส และความสัมพันธ์ของคลาสและวัตถุแล้วจะต้องนำคลาสและวัตถุเหล่านั้นมาปรับปรุงดัดแปลง เพื่อใช้ในงานร่วมกับคลาสไลบรารี่ที่มีอยู่แล้ว

2.4 การเขียนโปรแกรมเชิงวัตถุ [5]

การเขียนโปรแกรมแนวเชิงวัตถุเป็นวิธีการเขียนโปรแกรมอีกแนวหนึ่ง ซึ่งการเขียนโปรแกรมแนวนี้จะมองสภาพแวดล้อมของคอมพิวเตอร์ (Computer Environment) ว่าเกิดจากการรวมกันของวัตถุต่าง ๆ ที่ติดต่อกันโดยผ่านทางข่าวสารต่าง ๆ

ดังนั้นโปรแกรมที่เขียนโดยวิธีการเขียนโปรแกรมแนวนี้ จะต้องประกอบไปด้วยวัตถุต่าง ๆ ซึ่งวัตถุเหล่านั้นจะมีคุณสมบัติและพฤติกรรมเป็นของตัวเอง โดยที่คุณสมบัติของวัตถุนั้นจะไม่มีวัตถุตัวอื่นสามารถจัดการได้โดยตรง แต่คุณสมบัติของวัตถุจะถูกจัดการโดยพฤติกรรมของวัตถุนั้นเอง ซึ่งพฤติกรรมนั้นจะเกิดขึ้นเมื่อวัตถุนั้นได้รับข่าวสาร

ในโลกแห่งความเป็นจริง ทุกสิ่งทุกอย่างจะมีคุณสมบัติและพฤติกรรมเป็นของตัวเอง เช่น ลูกบาสเก็ตบอลส่วนใหญ่มีสีส้มเป็นคุณสมบัติของมัน และมันสามารถกระดอนได้ ซึ่งเป็นพฤติกรรมของมัน ในโลกแห่งการเขียนโปรแกรมก็เช่นกัน วัตถุก็มีคุณสมบัติและพฤติกรรมเช่น

เดียวกัน อย่างเช่น ในโปรแกรมวาดรูป วงกลมที่ถูกวาดขึ้นจะมีคุณสมบัติ คือ จุดศูนย์กลาง, สี, รัศมี, รูปแบบที่ระบายในวงกลม เป็นต้น และพฤติกรรม เช่น วาด, เคลื่อนที่, ลบ เป็นต้น ดังนั้นจึงถือว่าวงกลมที่วาดนี้เป็นวัตถุ

ภาษาโปรแกรมเชิงวัตถุ (Object-Oriented Programming Language) เป็นภาษาที่เกิดขึ้นมาเพื่อสนับสนุนการสร้างระบบที่ได้ออกแบบโดยใช้หลักการเชิงวัตถุ ตัวอย่างภาษาโปรแกรมเชิงวัตถุเช่น C++, SmallTalk เป็นต้น

2.5 ภาษาโปรแกรมเชิงวัตถุ C++

ภาษาโปรแกรมเชิงวัตถุ C++ เป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุซึ่งมีคุณสมบัติสำคัญต่าง ๆ ดังที่กล่าวไว้ในหัวข้อ "หลักการเชิงวัตถุ" ในที่นี้จะไม่กล่าวถึงรายละเอียดการเขียนโปรแกรมภาษา C++ แต่จะขอกล่าวถึงคุณสมบัติพิเศษของตัวภาษาในแง่ของการนำกลับมาใช้ใหม่ (Reusability) นั่นคือ คลาสเท็มเพลท (Class Template)

คลาสเท็มเพลท (Class Template) [6]

คลาสในภาษา C++ ถูกออกแบบไว้เพื่อที่ใช้จัดการกับข้อมูลชนิดใดชนิดหนึ่ง แต่ว่ามีบ่อยครั้งที่หน้าที่ของคลาสในการจัดการกับข้อมูลชนิดหนึ่งยังสามารถนำไปจัดการกับข้อมูลชนิดอื่นได้ด้วย เปลี่ยนเพียงชนิดของข้อมูลในคลาสเท่านั้น ดังนั้นจึงเกิดคลาสเท็มเพลทขึ้นเพื่อใช้แยกชนิดของข้อมูลออกจากคลาส

พิจารณาตัวอย่างข้างล่างนี้เป็นการประกาศคลาสที่ใช้จัดการรายการแบบเชื่อมโยง (Linked List) การจัดการต่าง ๆ ต่อรายการแบบเชื่อมโยงนั้นไม่ขึ้นกับชนิดของข้อมูลที่เก็บอยู่ในรายการแบบเชื่อมโยง ซึ่งคลาสเท็มเพลทเพียงหนึ่งคลาสสามารถใช้จัดการรายการแบบเชื่อมโยงของข้อมูลทั่วไป (ตัวแปร T ในตัวอย่างการประกาศคลาสเท็มเพลท) ได้ทุกชนิดไม่ว่าจะเป็นตัวแปรชนิดจำนวนนับ หรือตัวแปรชนิดที่ผู้เขียนโปรแกรมกำหนดขึ้นเอง

```
// Declaring a template class.
template<class T> class List {
public:
        List();
        void add(T& t);
        void remove(T& t);
      void detach(T& t) {remove(t)};
      ~List();
};
template<class T> List<T>::List()
{
        // .....
}
template<class T> void List<T>::add(T& t)
{
        // .....
}
template<class T> void List<T>::remove(T& t)
{
        // .....
}
template<class T> List<T>::~List()
{
        // .....
}
```

การประกาศคลาสเริ่มต้นด้วยคำสำคัญ (Keyword) "template" ตามด้วยนิพจน์ (Expression) "<class T>" เพื่อที่จะประกาศข้อมูลทั่วไปชนิด T

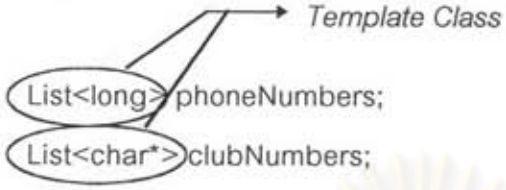
เมื่อใดก็ตามที่กำหนดชนิดของตัวแปรให้กับคลาสเท็มเพลทจะได้คลาสที่จัดการกับตัวแปรชนิดนั้นและเรียกคลาสที่ได้นี้ว่า เท็มเพลทคลาส (Template Class)

ตัวอย่างการใช้เท็มเพลทคลาสของคลาสเท็มเพลทรายการแบบเชื่อมโยงที่กล่าวข้างต้น

```
void main()
{
        List<long> phoneNumbers;
        List<char*> clubNumbers;

        static long number = 5551000;
        phoneNumber.add(number);

        static char* name = "Michael";
        clubNumbers.add(name);
}
```



ตัวแปลภาษา (Compiler) จะสร้างรหัส (Code) เพื่อจัดการกับข้อมูลชนิด long int และ char* จะได้คลาสหนึ่งคลาสสำหรับข้อมูลทั้งสองชนิด ซึ่งถ้าในคลาสเท็มเพลทประกาศข้อมูลชนิด *static* หรือ *virtual* function คลาสที่ได้จากการสร้างของตัวแปลภาษาจะมีข้อมูลชนิด *static* และ *virtual* function เป็นของแต่ละคลาส

2.6 คลาสไลบรารี่ MFC รุ่นที่ 4.0 [7]

คลาสไลบรารี่ MFC (Microsoft Foundation Class Library) เป็นคลาสไลบรารี่ของบริษัท Microsoft Corporation ที่มีให้มาพร้อมกับตัวแปรภาษา Microsoft Visual C++ ซึ่งคลาสไลบรารี่นี้ได้จัดเตรียมคลาสต่าง ๆ ที่เป็นโครงร่างงานของแอพลิเคชัน (Application Framework) ให้นักโปรแกรมภาษา C++ สามารถเขียนแอพลิเคชัน (Application) บนไมโครซอฟต์วินโดวส์ที่มีการติดต่อกับผู้ใช้และส่วนทำงานหลักของแอพลิเคชันได้ง่ายและรวดเร็ว

2.6.1 หลักการสำคัญ

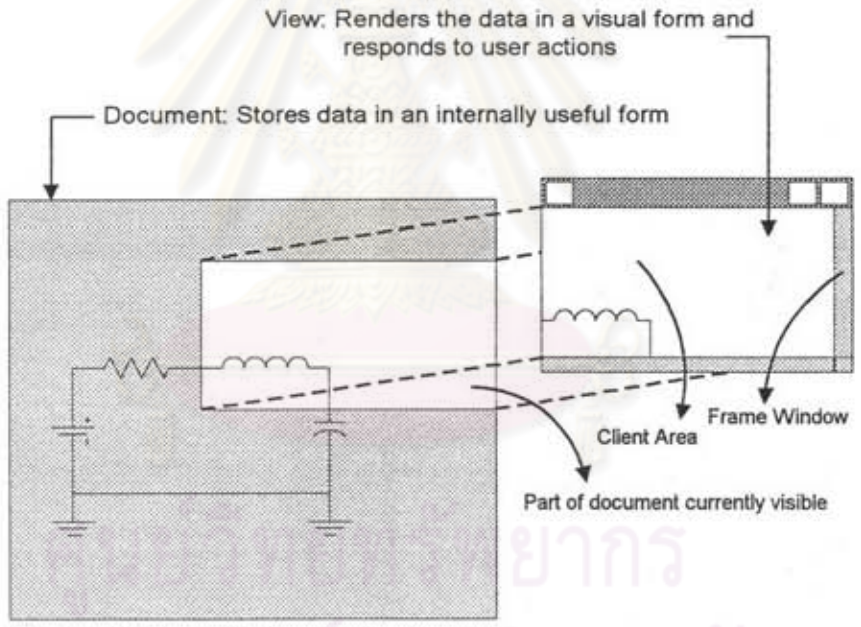
1. หัวใจสำคัญของแอพลิเคชันอยู่ที่ "Application Object" ทำหน้าที่จัดการ Document และดึงคำสั่ง (Dispatch Command) ให้กับวัตถุอื่น ๆ ในโปรแกรม

2. ข้อมูลที่ผู้ใช้กำลังใช้หรือจัดการอยู่เรียกว่า "Document"

3. ผู้ใช้ติดต่อกับข้อมูลได้โดยผ่านทาง "View" ซึ่ง View นี้คือหน้าต่างที่อยู่บน Client Area ของกรอบหน้าต่าง (Frame Window) ทำหน้าที่รับคำสั่งเพื่อจัดการกับข้อมูลของ Document

4. วัตถุอื่น ๆ ในส่วนติดต่อกับผู้ใช้ เช่น เมนู (Menu) และปุ่ม (Button) จะส่งคำสั่งไปให้ Document, View และวัตถุอื่น ๆ ในแอพพลิเคชันทำตามคำสั่งที่ได้รับ

ความสัมพันธ์ระหว่าง Document และ View แสดงไว้ในรูปที่ 2.9



รูปที่ 2.9 ความสัมพันธ์ระหว่าง Document และ View

2.6.2 การใช้งานโครงร่างงาน

1. นิยาม (Define) ข้อมูลของแอพลิเคชันไว้ในคลาส Document

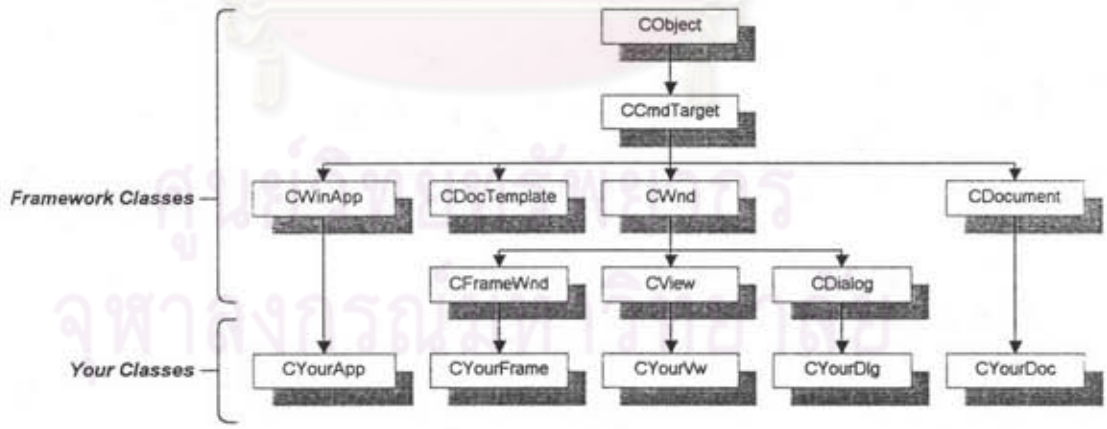
2. นิยามวิธีที่ผู้ใช้มองหรือติดต่อจัดการกับข้อมูลภายในหน้าต่าง

3. เชื่อมการติดต่อระหว่างเมนู (Menu), ปุ่ม (Button) และวัตถุที่ติดต่อกับผู้ใช้ (User-Interface Object) อื่น ๆ เข้ากับคำสั่ง (Command) แล้วเขียนฟังก์ชันที่จัดการกับคำสั่งนั้น

2.6.3 ส่วนโปรแกรมที่ต้องเขียนเพิ่ม

1. ประกาศ (Declare) โครงสร้างข้อมูล (Data Structure) ของ Document
2. Serialize ข้อมูลของ Document เพื่อที่สามารถอ่านและเขียนจากไฟล์ได้
3. แสดงข้อมูลของ Document บน View
4. จัดการกับข่าวสารของคีย์บอร์ด (Keyboard Message) และข่าวสารของเมาส์ (Mouse Message)
5. จัดการคำสั่งจากเมนูและปุ่มของทูลบาร์
6. จัดการเรื่องการพิมพ์, การเลื่อนหน้าจอ (Scrolling) และส่วนอื่นที่สามารถหาได้จากโครงร่างงาน

รูปที่ 2.10 แสดงตำแหน่งในลำดับชั้นของคลาสที่นักเขียนโปรแกรมต้องสืบทอดคลาสจากคลาสในคลาสไลบรารี่ MFC



รูปที่ 2.10 คลาสที่นักเขียนโปรแกรมต้องสืบทอดต่อจากคลาสในคลาสไลบรารี่ MFC

2.7 การจำลองการทำงานวงจรไฟฟ้าของโปรแกรมจำลองการทำงานวงจรไฟฟ้า "เล็ก" [1]

การจำลองการทำงานวงจรไฟฟ้าของโปรแกรมจำลองการทำงานวงจรไฟฟ้า "เล็ก" ใช้วิธี Modified Nodal ในการสร้างสมการวงจรไฟฟ้า (Circuit Equation) ในรูปของสมการเมตริกซ์

$$Ax = b$$

โดยที่ A เป็นเมตริกซ์ของสัมประสิทธิ์ของตัวแปรวงจรไฟฟ้า ขนาด n x n, n = จำนวนตัวแปรวงจรไฟฟ้า (Circuit Variable)

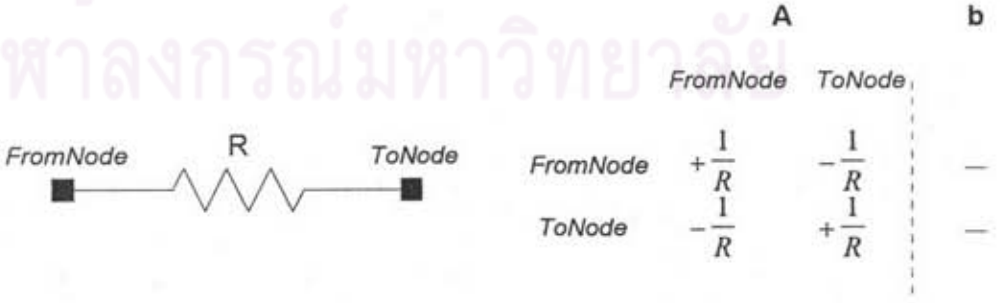
x เป็นเวกเตอร์หลัก (Column Vector) ของตัวแปรวงจรไฟฟ้า ขนาด n

b เป็นเวกเตอร์หลัก (Column Vector) ของค่าคงที่ ขนาด n

ซึ่งตัวแปรวงจรไฟฟ้าได้แก่ แรงดันที่ปม (Node Voltage), กระแสที่ไหลผ่านแหล่งกำเนิดแรงดัน และกระแสที่ไหลผ่านตัวเหนี่ยวนำ ส่วนการแก้สมการวงจรไฟฟ้าใช้วิธี LU Decomposition with Complete Pivoting

การสร้างสมการเมตริกซ์ของวงจรไฟฟ้าใช้วิธี Modified Nodal ค่าประจำอุปกรณ์ไฟฟ้าแต่ละชนิดในวงจรไฟฟ้าที่นำมาใช้สร้างสมการเมตริกซ์นั้นมีตำแหน่งเป็นรูปแบบที่แน่นอนในสมการเมตริกซ์ของวงจรไฟฟ้า เพียงแต่เปลี่ยนตำแหน่งในเมตริกซ์ A หรือเวกเตอร์ b เท่านั้น ซึ่งเรียกรูปแบบของการใส่ค่าประจำอุปกรณ์ลงไปในสมการเมตริกซ์ของวงจรไฟฟ้าว่า "ตราประจำอุปกรณ์" ("Element Stamp") ของอุปกรณ์ และอุปกรณ์แต่ละตัวก็จะมีตราประจำอุปกรณ์เป็นของตัวเอง ทำให้การสร้างสมการเมตริกซ์ไม่จำเป็นต้องสร้างมาจากวิธี Modified Nodal โดยตรง

ตัวอย่างตราประจำอุปกรณ์ของตัวต้านทานแสดงในรูปที่ 2.11



รูปที่ 2.11 ตราประจำอุปกรณ์ของตัวต้านทาน

ขั้นตอนการจำลองการทำงานวงจรไฟฟ้ามี 3 แบบ คือ

1. ขั้นตอนการจำลองการทำงานแบบ DC แสดงไว้ในรูปที่ 2.12 มีรายละเอียดแต่ละส่วนดังนี้

1.1 Load Matrix Circuit Equation คือการสร้างสมการวงจรไฟฟ้าที่ผู้ใช้ป้อนโดยใช้วิธีการใส่ตราประจำอุปกรณ์ของแต่ละอุปกรณ์ลงในสมการเมตริกซ์

1.2 ใช้วิธี LU Decomposition เพื่อหาคำตอบของสมการเมตริกซ์ของวงจรไฟฟ้า ซึ่งแบ่งเป็น 2 ส่วน

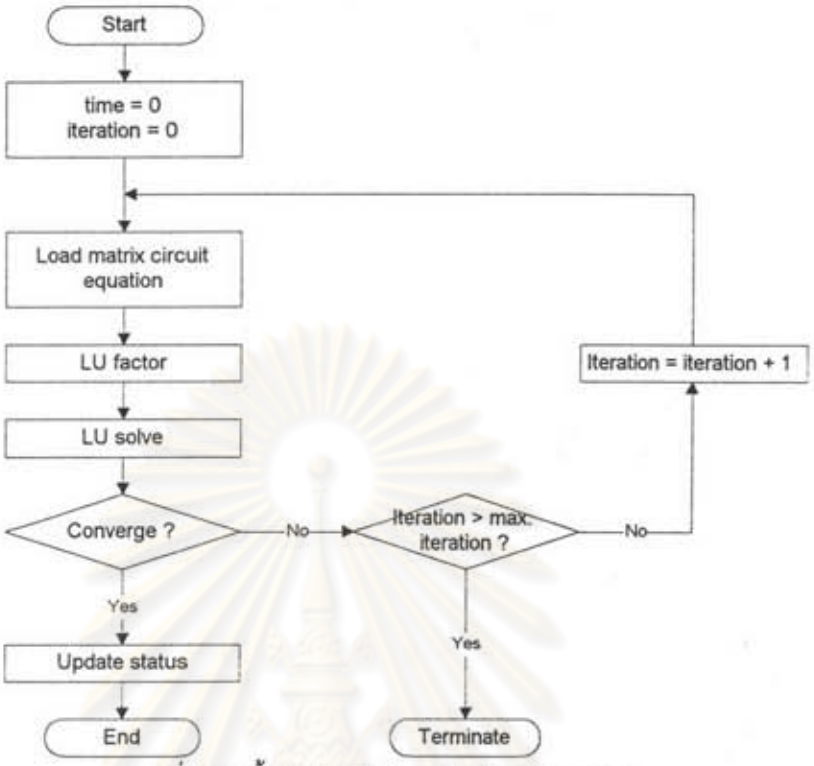
1.2.1 LU factor เป็นการแยกให้เมตริกซ์ A ในสมการเมตริกซ์อยู่ในรูปผลคูณของเมตริกซ์ L และ เมตริกซ์ U โดยที่เมตริกซ์ Lคือ เมตริกซ์สามเหลี่ยมข้างล่างและเมตริกซ์ U เป็นเมตริกซ์สามเหลี่ยมข้างบนและมีค่าในแนวเส้นทแยงมุมเป็น 1 ทั้งหมด ซึ่งจะได้สมการในรูป LUx = b

1.2.2 LU solve เป็นการแก้สมการเมตริกซ์ที่ได้จากการทำ LU factor คำตอบที่ได้จะอยู่ในเวกเตอร์ x

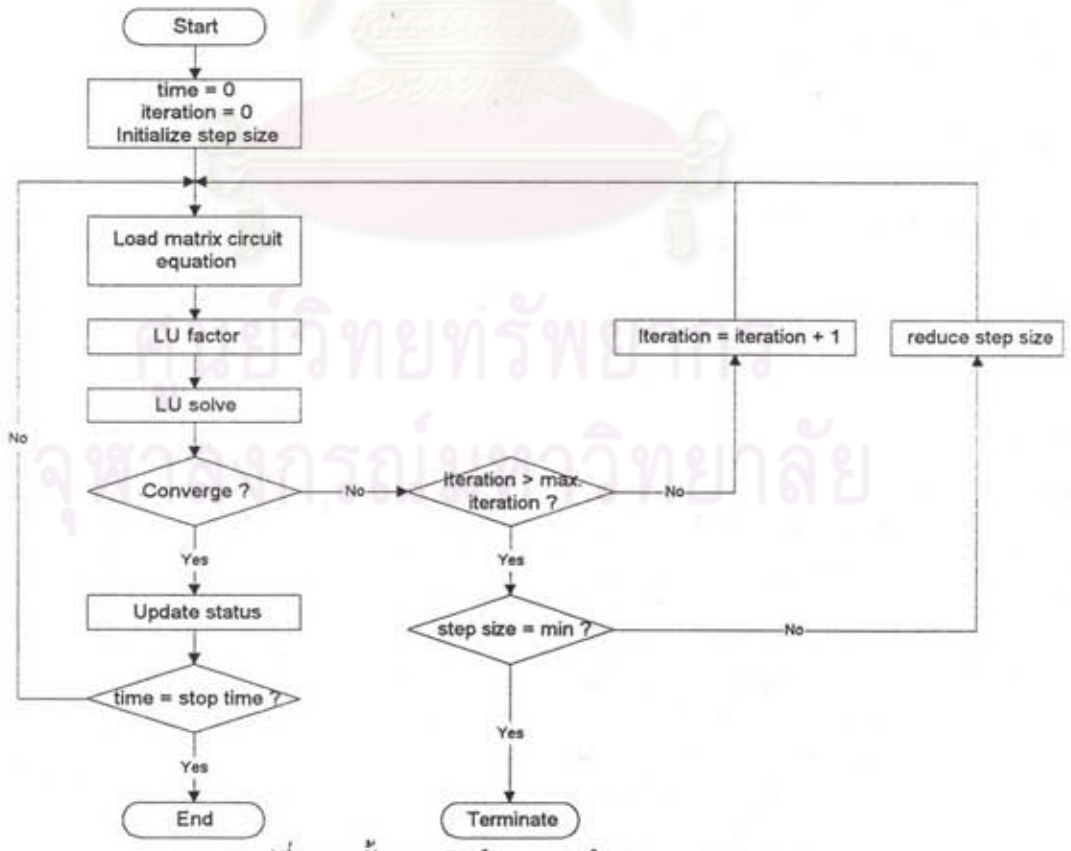
1.3 Check Converge เป็นการตรวจสอบการลู่เข้าของวงจรว่าคำตอบที่ได้นั้นลู่เข้าหรือไม่ ถ้าคำตอบนั้นลู่เข้าก็จะได้ผลลัพธ์ของการจำลองการทำงานวงจรไฟฟ้าแบบ DC นั้น แต่ไม่ลู่เข้าก็จะทำการวนซ้ำ (Iteration) หาคำตอบจนคำตอบนั้นลู่เข้า แต่ถ้าการวนซ้ำมากกว่าค่าการวนซ้ำสูงสุดที่กำหนดไว้ (Maximun Iteration) ก็จะหยุดการจำลองการทำงานวงจรไฟฟ้า

2. ขั้นตอนการจำลองการทำงานแบบ Transient แสดงไว้ในรูปที่ 2.13

จากแผนผังสายงานในรูปที่ 2.13 การจำลองการทำงานวงจรไฟฟ้าแบบ Transient เหมือนกับแบบ DC แต่ต่างตรงที่การจำลองการทำงานวงจรไฟฟ้าแบบ Transient มีการคำนวณที่หลาย ๆ จุดเวลา และมีการตรวจสอบว่าคำตอบของสมการเมตริกซ์ลู่เข้าหรือไม่ ถ้าลู่เข้าก็เก็บคำตอบแล้วแก้คำตอบของสมการที่จุดเวลาถัดไป แต่ถ้าคำตอบยังไม่ลู่เข้าก็จะตรวจสอบการวนซ้ำว่ามากกว่าค่าการวนซ้ำมากสุดหรือไม่ ถ้ามากกว่าก็จะตรวจสอบค่า Step Size ว่าเท่ากับค่า Minimum Step Size หรือไม่ ถ้าค่า Step Size เท่ากับค่า Minimum Step Size ก็จะเลิกการจำลองการทำงานวงจรไฟฟ้า แต่ถ้าไม่เท่ากับก็จะลดค่า Step Size ลง แต่ถ้าค่าการวนซ้ำไม่มากกว่าค่าการวนซ้ำมากที่สุดก็ให้เพิ่มค่าเริ่มต้นการวนซ้ำใหม่ จนกว่าคำตอบจะลู่เข้าแล้วเริ่มหาคำตอบที่จุดเวลาถัดไป

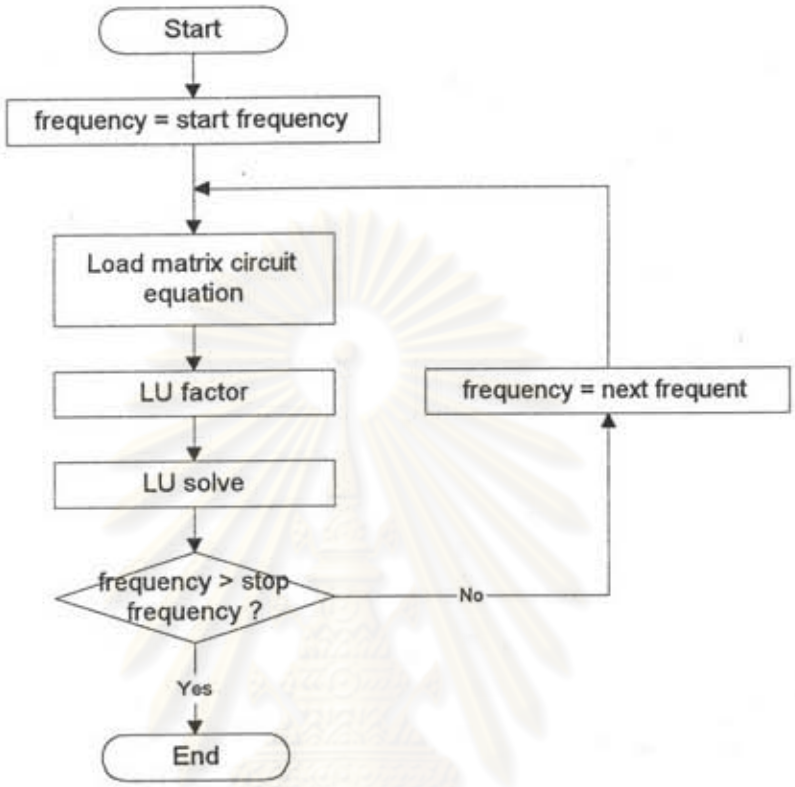


รูปที่ 2.12 ขั้นตอนการจำลองการทำงานแบบ DC



รูปที่ 2.13 ขั้นตอนการจำลองการทำงานแบบ Transient

3. ขั้นตอนการจำลองการทำงานแบบ AC แสดงไว้ในรูปที่ 2.14



รูปที่ 2.14 ขั้นตอนการจำลองการทำงานแบบ AC

จากแผนสายงานรูปที่ 2.14 ในขั้นตอนการทำงานวงจรไฟฟ้าในขั้นตอน Load matrix circuit equation, LU factor และ LU solve เหมือนกับการจำลองการทำงานวงจรไฟฟ้าแบบ DC แต่ค่าในเมตริกซ์เป็นจำนวนเชิงซ้อน และไม่มีการตรวจสอบการลู่เข้าของคำตอบ เพราะคำตอบที่ได้จะอยู่ในรูปของจำนวนเชิงซ้อนที่ความถี่ต่าง ๆ