บทที่ 2
การเขียนโปรแกรมเชิงวัตถุและการเขียนโปรแกรมแบบ Procedural

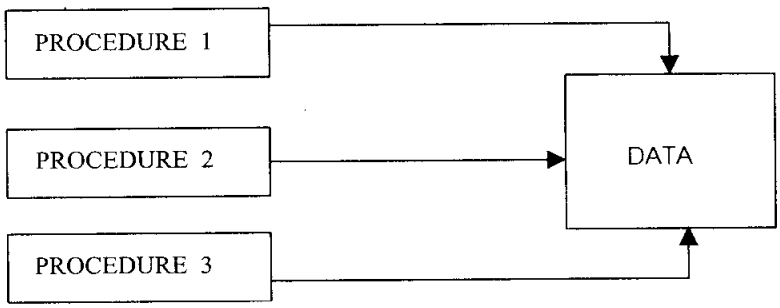
บทนำ แนวความคิดการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming)

การเขียนโปรแกรมเชิงวัตถุ เป็นแนวคิดของการมองสิ่งทุกสิ่งในโปรแกรมว่าเป็นวัตถุ ดังนั้นโปรแกรมก็จะประกอบด้วยวัตถุต่างๆนำมาประกอบกันเข้า โดยที่วัตถุ (Object) นั้นๆก็จะประกอบด้วย ขั้นตอนในการดำเนินงาน และ ข้อมูลที่จะนำไปดำเนินการ เราอาจจะเปรียบเทียบได้ว่าวัตถุนั้นๆก็เปรียบเสมือน ข้อมูล 1 Record แต่เป็น Record พิเศษที่ยอมให้สมาชิกบางตัวทำหน้าที่เป็น Procedure ได้ แต่ต้องมีข้อแม้ว่าวัตถุนั้นจะต้องมีหลักการในเรื่องของ Encapsulation และ Protection ด้วย

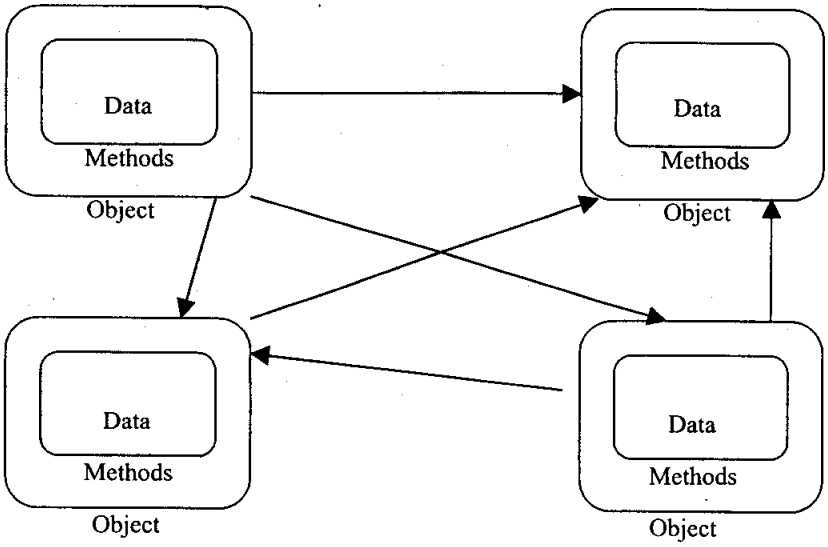
แนวความคิดของการเขียนโปรแกรมแบบดั้งเดิมนั้น เราจะแบ่งระบบโปรแกรมออกเป็นส่วนๆที่เรียกว่า Module (Procedure or Function) เพื่อดำเนินกิจกรรมแต่ละส่วนของระบบ โดยลักษณะการแบ่งที่นิยมใช้กันก็คือการออกแบบโปรแกรมแบบ Top Down Design ซึ่งการเขียนโปรแกรมแบบดั้งเดิมนั้นจะมีจุดอ่อนตรงที่ว่า โปรแกรมนั้นๆไม่สามารถควบคุมการไหลของข้อมูลและไม่สามารถปกป้องข้อมูลได้ นอกจากนี้ยังขาดลักษณะของการนำโปรแกรมนั้นกลับมาใช้ใหม่ (Reusable) ภายหลัง ดังนั้นจึงมีการพยายามในการจะแก้จุดอ่อน ดังกล่าวและทางหนึ่งที่ค้นพบก็คือการแก้ปัญหาด้วยวิธีของ OOP

แนวทางการแก้ปัญหาแบบ OOP ก็คือการนิยามว่าโปรแกรมจะเป็นเซทของ Object โดยที่ Object แต่ละ Object จะรวมเอาส่วนของข้อมูลและกรรมวิธีในการจัดการข้อมูลเข้าไว้ด้วยกัน โดยการนิยามเช่นนี้จึงทำให้แนวทางในการออกแบบโปรแกรมเปลี่ยนแปลงไปด้วย กล่าวคือจะเป็นการออกแบบความเกี่ยวกันระหว่าง Object และโครงสร้างของ Object นั้นๆ เราจะเปรียบเทียบแนวทางในการออกแบบโปรแกรมตามรูปแบบดั้งเดิมคือ Traditional Program และ Object Oriented ได้ดังนี้

ภาพที่ 2.1 รูปแบบการเขียนแบบ Traditional Program



ภาพที่ 2.2 แสดงวิธีการเขียนโปรแกรมเชิง วัตถุ (Object Oriented Program)

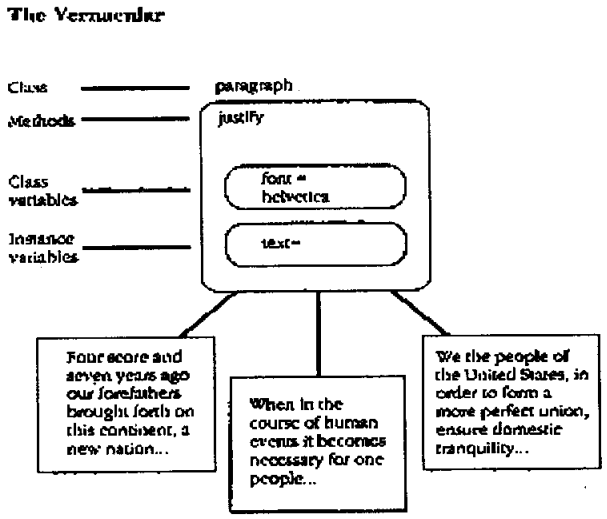


หลักการต่างๆที่สำคัญของ OOP

Class Class กับ Object จัดว่าเป็นของคู่กันเพราะ Class ก็คือสิ่งที่นำมาสร้าง Object (อาจจะเรียกว่าเป็นแม่พิมพ์ของ Object ก็ได้) เพราะ Class จะถ่ายทอดคุณสมบัติให้ติดไปกับ Object ที่มาจาก Class นั้นๆด้วย ตัวอย่างที่เราสามารถนำมาเปรียบเทียบได้ง่ายๆก็คือ ถ้ามนุษย์หมายถึง Class ดังนั้นมนุษย์แต่ละคนที่อยู่บนโลกนี้ก็คือ Object นั่นเอง ตัวอย่างเช่น นายแดง ก็ถือว่าเป็น Object หนึ่งซึ่งจะต้องมีคุณสมบัติมาจาก Class ของมนุษย์คือต้องพูดได้ กินได้ เดินได้ เป็นต้น แต่ถ้าเกิดนายแดงมีคุณสมบัติพิเศษที่เพิ่มเติมเข้ามาเช่น มีลักษณะเฉพาะตัวคือ กระดิกหูได้ ซึ่งไม่ปรากฏในคุณสมบัติของมนุษย์โดยทั่วไป เราจะเรียกคุณสมบัติพิเศษนี้ว่า Instance Variable ในขณะที่คุณสมบัติที่มีในมนุษย์ทั่วไปอันสืบเนื่องมาจากการสืบทอดคุณสมบัติมาจาก Class ที่เป็นต้นกำเนิดว่าเป็น Class Variable

ตัวอย่างของการออกแบบโปรแกรม Word Processor โดยใช้แนวทางของ OOP โดยการกำหนด Class ที่เป็น Paragraph Class ขึ้นมาหนึ่ง Class โดยที่ Paragraph Class จะมีคุณสมบัติดังต่อไปนี้คือ ประกอบด้วย Variable คือ Text และ Font โดยที่ Text จะเป็นตัวแปรที่ใช้เก็บขนาดความยาวของข้อความของ Paragraph ส่วน Font จะเก็บลักษณะ Font ของ เช่น Fixedfont จากรูปจะเห็นได้ว่า Paragraph แต่ละอันจะมีขนาดไม่เท่ากัน ซึ่งลักษณะเช่นนี้จะเป็น Instance Variable ในขณะที่ Font จะเป็น Class Variable ต่อไปนี้จะแสดงคุณสมบัติที่อธิบายให้เห็นชัดเจนยิ่งขึ้นการถ่ายทอด การสร้าง Subclass

ภาพที่ 2.3 แสดงโครงสร้างของ Class



โครงสร้างของคลาส Paragraph

Encapsulation เป็นหลักการซ่อนเร้นความลับของ Object กล่าวคือจะป้องกันการใช้หรือแก้ไขตัวแปรภายใน Object นั้นๆให้มีสถานะของตัวแปรเป็นแบบ Private หรือที่เรียกว่า Local Variable ซึ่งจะทำให้ไม่สามารถเข้าถึงข้อมูลได้โดยตรง การเข้าถึงข้อมูลนั้นจำเป็นจะต้องกระทำผ่าน Procedure ที่เป็น Method ของ Object เท่านั้น (Object หนึ่ง Object สามารถที่จะถูกเรียกได้จาก Method ต่างๆกันได้หลาย Method โดยที่แต่ละ Method ก็จะมี Message ที่ต่างกันหรือเหมือนกันก็ได้) ให้พิจารณาจาก โปรแกรมต่อไปนี้

```
class Point  { int x,y ;
public
        int GetX () ; return x;
        int GetY () ; return y;
};
```

จากตัวอย่างเป็น class Pointของ Object ที่เก็บจุดของข้อมูลจุดหนึ่งบนจอภาพ การจะขอดูข้อมูลค่า X หรือค่า Y ใน Object นี้ไม่สามารถกระทำได้โดยตรง ดังนี้

```
printf ("% d", X) ;
```

แต่จะต้องดำเนินการโดยผ่านการเรียกใช้ Method int GetX ดังนี้

```
printf ("% d", Point.GetX()} ;
```

Inheritance หมายถึงการถ่ายทอดคุณสมบัติของ Class โดยที่เราอาจจะจัดได้ว่า Class ก็คือข้อมูลประเภทหนึ่งนั่นเอง เราอาจจะสร้าง Class ขึ้นมาใหม่ใช้งานได้โดยการอาศัยจากรูป Class เดิมที่มีอยู่ โดยที่ Class ลูกที่เกิดมาใหม่จะสืบทอดคุณสมบัติของ class แม่มาไว้ที่และยังผนวกด้วยคุณสมบัติใหม่เพิ่มเข้าไปอีกด้วย โดยวิธีการเช่นนี้จึงทำให้ผู้ใช้สามารถสร้าง Class ใหม่โดยไม่ยุ่งยาก ลักษณะของการสืบทอดก่อให้เกิดประโยชน์ 3 ประการคือ

1. ช่วยลดเวลาในการพัฒนาระบบ
2. ช่วยลดค่าใช้จ่ายในการพัฒนาระบบ
3. ระบบที่สร้างขึ้นมานั้นจะมีมาตราฐานสามารถปรับเปลี่ยนได้ง่าย

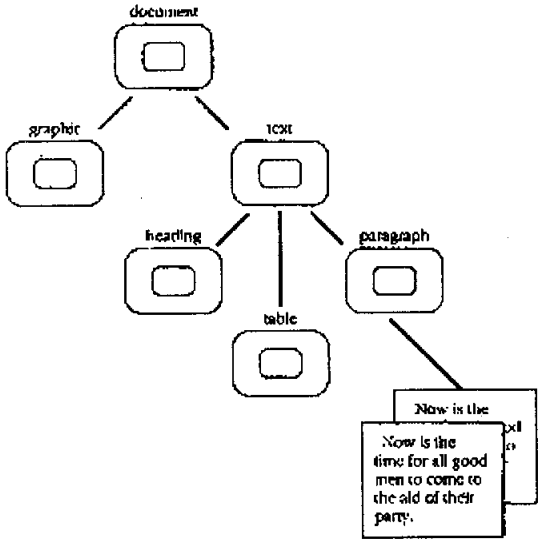
พิจารณาจากตัวอย่างลักษณะของการสืบทอดได้ดังนี้

```
Point = object
   X , Y :integer;
   procedure MakePoint (x,y :integer);
    procedure Show ;
    end;
   Circle = object(Point)
         Radius : integer ;
          procedure MakeCircle (r :integer);
          procedure Show ;
    end;
```

ตัวอย่างดังกล่าวนั้น Class Point และ Circle ที่ปรากฏนั้น Class Circle จะเป็นลูกของ Class Point โดยที่ Class ทั้งสองจะมี procedure Show เหมือนกัน แต่ผลจากการทำงานจะไม่เหมือนกันตรงที่ว่า Show ของ Class Point จะแสดงจุด ในขณะที่ Show ของ Class Circle จะแสดงผลออกมาเป็นวงงกลม Class Circle คือ Class ของลูกจะถูกถ่ายทอด procedure Show จาก Class Point ของแม่ แต่นำมาเปลี่ยนแปลง

การถ่ายทอดคุณสมบัติจากบรรพบุรุษไปยังลูกหลานนั้น ในบางภาษาสามารถดำเนินการได้ในลักษณะของการจัดการที่เรียกว่า Multiple Inheritance โดยการนำ Class หลายๆ Class มาสร้างเป็น Class ใหม่ขึ้นมาดังนั้น Class ใหม่จึงสืบเอาคุณสมบัติจากหลายๆ Class เข้ามาไว้ด้วยกัน ตัวอย่างเช่นการนำเอาคุณสมบัติของ Software Text Editor มาผนวกกับ Image Processor เข้าไว้ด้วยกัน เราจึงได้โปรแกรม Word ที่มีความสามารถในเชิงของ Text กับ Image เข้าไว้ด้วยกัน

Subclass หรือ Derived Class (Child Class) สืบเนื่องมาจากความสามารถในการสืบทอด ซึ่งทำให้มีการสร้าง Subclass ออกมามากมาย จึงมีการนำเอากลุ่มของ Subclass มารวมไว้อยู่ที่เดียวกัน เรียกว่า Class Library (Class Hierarchy) แนวคิดคล้ายๆกับเป็นที่รวมของบรรดา Procedure เข้าไว้ด้วยกัน



ภาพ 2.4 แสดง Class และ Subclass ของ Document

Polymorphism หมายถึงคุณลักษณะที่สามารถตอบสนองได้หลายกรณี อาจจะเรียกได้ว่า "One Interface Multiple Response" ลักษณะงานดังกล่าวเปรียบได้กับการส่ง Message แบบเดียวกันแต่ได้รับการตอบสนองไม่เหมือนกัน ถ้าจะเปรียบเทียบกับการเขียนโปรแกรมแบบดั้งเดิมก็หมายความว่าเรามีสิทธิที่จะเขียนโปรแกรมย่อยสองโปรแกรมที่มีชื่อเหมือนกัน โดยที่ทั้งสองโปรแกรมจะทำงานที่แตกต่างกัน ใน OOP จะมี Method แทนที่ฟังชัน โดยที่ Method นั้นจะมี

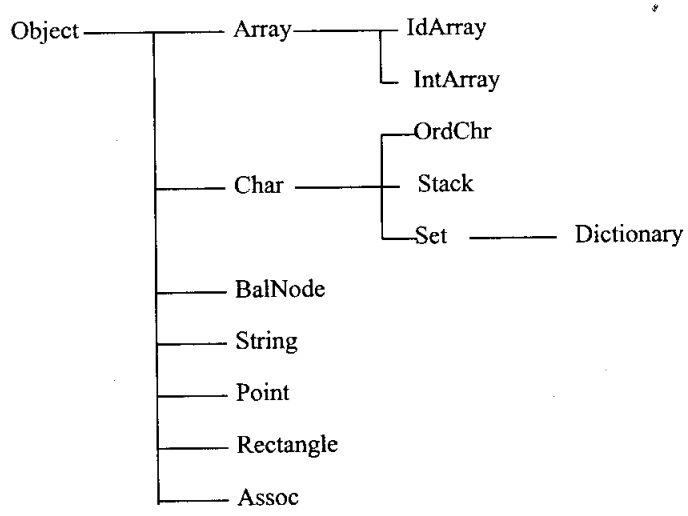
ลักษณะเป็น Module ที่อยู่ภายใต้สังกัดไม่ใช่เป็น Module ที่แยกเป็นเอกเทศเหมือนฟังก์ชันที่เราเขียนในโปรแกรมแบบดั้งเดิม Method จะอยู่ภายใต้ Object โดยที่ Object แต่ละ Object ถึงแม้จะได้รับ Message ที่เหมือนกัน ก็จะให้ Method ภายในที่ตอบสนองออกมาในลักษณะที่แตกต่างกัน ตัวอย่างเช่น Object X ส่ง Message ให้กับ Object Y ดังนั้น Method Draw () ที่อยู่ภายใน Object Y จะทำงานในการตอบสนองต่อ Message เช่นถ้า Object Y เป็น Object ที่มาจาก Class ของ Circle ก็จะวาดรูปวงกลม แต่ถ้าเป็น Object ใน Class ของ Line ก็จะลากเส้นให้ จะเห็นว่าเราใช้ Message เช่น On Mouse Click ตัวเดียวกันในการส่ง Message ให้กับ Method ชื่อเดียวกันก็คือ Draw () แต่เนื่องจาก Draw() นั้นอยู่ใน Object ที่มาจาก Class ที่ต่างกัน ผลจากการกระทำจึงออกมาต่างกันไป สรุปได้ว่าการส่ง Message แบบเดียวกันสามารถก่อให้เกิดการตอบสนองได้หลายอย่างขึ้นอยู่กับว่า Object ที่รับ Message นั้นเป็นอะไร ความหมายของ Polymorphism มักจะถูกรวมเข้ากับคุณสมบัติของการถ่ายทอด โดยถือว่า ถ้า Class ของ Child รับเอา Method ของ Parent มาก็สามารถสืบทอดคุณสมบัติของ Method นั้นต่อไป หรือมิฉะนั้นก็สามารถที่จะนำไปดัดแปลงแก้ไขหรือเพิ่มการกระทำได้

Dynamic Binding การ Binding ก็คือการเรียก Function (call routine) ในกรณีของการเขียนโปรแกรมในรูปแบบดั้งเดิม (Procedural Language)นั้น การ Binding จะเป็นการเชื่อม Address ของ Routine **ที่ถูกเรียก** ให้เข้ากับ Routine ที่เป็น**ผู้เรียกใช้** แต่ในกรณีของการเขียนโปรแกรมแบบ OOP นั้นการ Binding จะเชื่อมเอา Address ของ Method ของ Object ที่ได้รับ Message กับ Message ที่ส่ง โดยที่การ Binding นั้นอาจจะเกิดได้ในขณะที่เรา Compile Program หรือเกิดในขณะที่เรา Run โปรแกรม ก็ได้ ในกรณีของการ Binding ในช่วงของการ Compile นั้นเราจะเรียกว่า Static Binding (Early Binding or Strong Binding or Compile Time Binding) ละถ้าเกิดในขณะที่เรา Run โปรแกรมเราจะเรียกว่า Dynamic Binding (Late Binding or Weak Binding or Run Time Binding)

สำหรับการเชื่อมโยง Routine ในภาษาประเภท Procedural Language เช่น Pascal หรือภาษา FORTRAN นั้นจะเป็นกรณีของ Static Binding เท่านั้น ส่วนการเชื่อมโยงของ OOP จะมีทั้งแบบ Static Binding และ Dynamic Binding กรณีที่เป็น Dynamic Binding ก็เพราะว่า Compiler ไม่สามารถตรวจสอบได้ว่าจะเป็นการเชื่อมโยง Object ไหนที่ควรจะได้รับ Message ดังกล่าวเข้าด้วยกัน จนกว่าจะมีการ Run โปรแกรมนั้นๆจึงจะสามารถตรวจสอบว่า Message คืออะไร การมีลักษณะของ Dynamic Binding นั้นจะช่วยให้เราสามารถเขียนโปรแกรมได้สั้นเข้า แต่ก็จะทำให้เราเสียเวลาเพิ่มขึ้นในขณะทำงาน

Class Library ดังที่ได้กล่าวมาแล้วในเรื่องของการสร้าง Class ว่าจะมีการรวบรวมเอา Class ต่างๆที่สร้าง ขึ้นมาเก็บเอาไว้ใน Class Library (Class จะเปรียบเสมือนเบ้าที่ใช้หล่อ Object) การดำเนินการของ OOP นั้นจะถือว่า Class Library เป็นส่วนที่สำคัญมากๆ ทั้งนี้เพราะ Class Library จะเป็นแหล่งที่เก็บรวบรวมทรัพยากรที่ใช้ในการสร้างโปรแกรม (เปรียบได้กับที่เก็บบรรดา Built in Function ต่างๆเช่น RAND , ABS ,SIN ,COS etc ที่สามารถนำไปใช้งานได้ โดยที่เราไม่ต้องนำมาเขียนโปรแกรมขึ้นมาใหม่ ในภาษาที่เป็น Procedural Language)

ภาพ 2. 5 ตัวอย่างบางส่วนของ Class Library



ภาษาบางภาษาเช่น Smalltalk จะมี Class Library ที่แสดงผลเป็น graphic ให้ใช้งานปรากฏอยู่ โดยที่ผู้ใช้สามารถแก้ไขหรือดัดแปลงได้ง่ายๆ ความแตกต่างระหว่าง Function Library ในภาษา Procedural Language กับ Class Library ใน OOP ก็คือ เราสามารถเรียก Function Library มาใช้ได้ แต่เราไม่สามารถที่จะแก้ไข Function Library ดังกล่าวได้ ในขณะที่เราสามารถที่จะแก้ไข Class Library ได้ เช่นในการเพิ่ม - ลดตัวแปร หรือ Method ใน Class ต่างๆได้

แนวคิดเปรียบเทียบระหว่าง OOP กับ ภาษา Procedural Language

เราอาจจะเปรียบเทียบรูปแบบการดำเนินแบบ OOP กับ Procedural Language ได้ว่า Procedural Language เป็นรูปแบบของการจัดการแบบ Data Driven Program ในขณะที่ OOP ใช้รูปแบบการจัดการเป็นแบบ Event Driven Program ขอให้พิจารณาจากการดำเนินการบน ระบบปฏิบัติการ DOS กับระบบปฏิบัติการ Window ในการดำเนินกิจการบางอย่างเช่น การจัดการทำสำเนาแฟ้มข้อมูลนั้น เราจะใช้คำสั่ง C > COPY A:ONE.DAT B: ในขณะที่กิจกรรมเดียวกันนี้เราจะใช้ Mouse Click ไปที่ แฟ้มบน Drive A: กับ แฟ้ม ONE.DAT แล้วนำภาพของแฟ้มดังกล่าวไป

วางไว้บน Drive B: ผลจะเป็นการทำสำเนาแฟ้มเช่นเดียวกับคำสั่งบน DOS แต่ด้วยวิธีการเช่นนี้บนระบบปฏิบัติการบน Window โอกาสที่จะผิดพลาดจะน้อยกว่า

ภาษาแบบ Object Oriented

แนวความคิดของ OOP เกิดขึ้น ในช่วงปี 1950 – 1960 โดยต้นแบบคือภาษา LISP (list Processing) ซึ่งเขียนขึ้นมาเพื่อเจตนาใช้งานทางด้าน AI (Artificial Intelligence) สืบเนื่องมาจากภาษา LISP จะเป็นภาษาแรกที่มีเทคนิคของการทำ Dynamic Binding นอกเหนือจากนี้ยังนำไปใช้ประโยชน์ในด้านการออกแบบที่เป็น GUI (Graphic User Interface) ภายหลังจากการเกิดของภาษา LISP ก็มีการพัฒนาภาษาตามคือภาษา Ada และ Modula 2 และในปี ค.ศ. 1970 ก็เกิดภาษา Smalltalk โดยผู้ที่พัฒนาคือ Alan Key ภาษา Smalltalk นับเป็นภาษาที่นับว่าเป็น OOP สมบูรณ์ที่สุดและนับเป็นภาษาแรกของ OOP ที่สมบูรณ์ที่สุด ในช่วงของการพัฒนา Smalltalk ก็มีการพัฒนาภาษาตระกูล Conventional Programming เช่นภาษา BASIC , PASCAL เช่นกัน แต่เนื่องจากความนิยมในรูปแบบของ OOP ยังไม่แพร่หลายมาก ดังนั้นกระบวนการใช้ภาษา OOP เพื่อเขียนโปรแกรมจึงยังไม่นิยมแพร่หลายเท่าที่ควร โดยที่ยังคงนิยมวิธีการเขียนโปรแกรมแบบดั้งเดิม คือการเขียนโปรแกรมแบบ Data Driven Program จวบจนกระทั่งแนวความคิดแบบ OOP ได้ถูกยอมรับกันมากขึ้น จึงมีการพัฒนาภาษา รวมทั้ง Software ประเภท DBMS ในแนวทางของ Object Oriented เกิดขึ้นมากมาย เช่น ภาษา C++ , Object Oriented Pascal

ภาษา OOP นั้นจะมีการแบ่งกลุ่มออกเป็น 2 กลุ่มคือ

- Pure OOP
- Hybrid OOP

โดยที่ภาษาที่ถือว่าอยู่ในลักษณะ Pure OOP ก็คือ SmallTalk และ Simula ส่วนภาษาในกลุ่มของ Hybrid OOP เช่นภาษา C++ , Object Pascal เป็นต้น ภาษาในกลุ่มของ Pure OOP จะกำหนดให้ทุกหน่วยเป็น Object ไปหมด แม้กระทั่งการใช้คำสั่งควบคุมเช่น If … Then หรือ Do ….While ก็จะถูกกำหนดให้เป็น Object ส่วนภาษาในกลุ่มของ Hybrid OOP จะมีลักษณะการจัดการเป็นรูปแบบผสมระหว่างแนวคิดการเขียนโปรแกรมแบบเดิมกับการเขียนโปรแกรมเชิงวัตถุผสมกัน

ตัวอย่างการเขียนโปรแกรมแบบ OOP โดยการใช้ภาษา C++

```
class Vector  {
// …
public :
Vector(Vector& v);      // ผู้สร้าง…สร้างสำเนาของ v ขึ้นมาใหม่
Vector(int n) ;         // ผู้สร้าง…สร้างเวคเตอร์องค์ประกอบของ n
Vector(char *num) ;     // ผู้สร้าง… เปิด nm และอ่านในเวคเตอร์
```

```
-Vector() ; }            // ผู้ทำลาย
```

ภาษา C++ นับว่าเป็นภาษา OOP ที่นิยมใช้กันมาก ดังนั้นขอสรุปคุณลักษณะของ C++ มาเทียบกับภาษา C ที่เคยใช้กันอยู่ว่ามีความละม้ายคล้ายกันหรือแตกต่างกันอย่างไร

1. C++ นับว่า Super Set ของ C การที่มีลักษณะเช่นนี้แปลว่าเราสามารถนำโปรแกรม C มารันบนบน C++ ได้ ซึ่งกรรมวิธีเช่นนี้จะเรียกว่า Backward Compatible ซึ่งนับว่าเป็นจุดแข็งของ C++ เพราะทำให้ผู้ใช้ C เดิมสามารถใช้ C++ ได้
2. C++ เป็นรูปแบบของ Hybrid OOP
3. C++ มีคุณสมบัติให้เลือกมาก เช่นถ้าเป็นการแก้ปัญหาเชิงคณิตศาสตร์จะเลือกคุณลักษณะที่เน้นในการปฏิบัติงานทางคณิตศาสตร์เช่นเลือกใช้ Library ทางคณิตศาสตร์ แต่ถ้าเน้นในลักษณะของ OOP ก็เลือกคุณลักษณะของ Object
4. C++ มี Class เป็นโครงสร้างหลักภายใน Class จะมีองค์ประกอบของ Data และ Code จัดการ ซึ่งจะนับว่าเป็นสมาชิกของ Class (Class Member) รวมทั้ง Method ก็จะจัดว่าเป็นสมาชิกด้วย แต่จะเรียกว่า Member Function โดยที่สมาชิกภายใน Class สามารถจะกำหนดให้มีสถานะภาพเป็น Private หรือ Public หรือ Protected ก็ได้ การใช้งานสมาชิกที่เป็น Private นั้นจะกระทำผ่านสมาชิกที่เป็น Public ก่อนเสมอ ตัวอย่างเช่น ถ้ามีตัวแปร int a = 1 ราไม่สามารถจะกำหนดให้ a = 1 ได้โดยตรง แต่เราจะต้องกระทำผ่าน Method เป็น Public ก่อนเท่านั้น และในขณะเดียวกันถ้าสมาชิก-ของเป็น Class เป็นแบบ Public เราจะต้องดำเนินการผ่าน Method ที่เป็น Public อีกเช่นกัน ส่วนในกรณีของการกำหนดสมาชิกใดๆให้มีสภาพเป็น Protected หมายความว่า Method จะสามารถกระทำกับสมาชิกที่มีคุณลักษณะเช่นนี้ได้นั้น Method นั้นๆจะต้องเป็นลูกของClass แม่ที่มีสมาชิกดังกล่าวอยู่สืบทอดมาจาก Class ของแม่นั้น และในกรณีที่ไม่มีการระบุลักษณะว่าเป็น Public หรือ Private หรือ Protected จะถูกกำหนดให้ค่าปริยายเป็น Private
5. การเรียกใช้ ฟังก์ชันต่างๆบน C++ จะอยู่ในรูปของการส่ง Message

ตัวอย่างของโปรแกรม C++

```
# include <stdio.h>
class point {
        private :
                int X,Y : // private variable
                // private member function
                        void init (int a, int b) {
                                x = a;  y = b;
                                }
        public
                // public member function
                Point (int x, int y ) { // constructor no.1
                        X = x; Y = y ;
                        }
                        Point() { // constructor no.2
                                init(0,0) ;
                                }
                        ShowPont () {
                                Printf("\n X- Coordinate = %d",X);
                                Printf("\n Y- Coordinate = %d",Y);
                                }
                        };
                        main() {
                                Point data (3,4); // create object named 'data'
                                data.ShowPoint (); // ' X –ordinate =3'
                                                // ' Y –ordinate=4'
                                }
```

นอกจากนี้บนภาษา C++ ซึ่งมีคุณลักษณะของ OOP ได้กำหนดฟังก์ชันขึ้นมาใช้งานสองชนิดในการสร้างและทำลาย Object ที่เรียกว่า Constructor & Destructor โดยที่ Constructor จะเป็นตัวสร้าง Object ละ Destructor จะเป็นตัวทำลาย Object เมื่อเลิกใช้ Object นั้นๆ นอกจากนี้ C++ ยังมีความสามารถใช้กรรมวิธีของการ Overloading ได้ทั้ง ฟังก์ชันและ Operation นั่นคือเรา

สามารถเขียนฟังก์ชั่นหลายๆฟังก์ชันที่มีชื่อซ้ำกันแต่ทำงานคนละแบบได้ เช่นถ้าหากเป็นฟังก์ชันของสมาชิก จะใช้ชื่อ Object ส่ง Message มา แยกแยะว่าจะใช้ฟังก์ชันใดในการทำงาน และถ้าเป็นกรณีของไม่ใช่ฟังก์ชันของสมาชิก จะใช้ Argument ของฟังก์ชันในการแยกแยะแทน ส่วนลักษณะของ Overloading Operation ก็คือการเพิ่มรูปแบบพิเศษ เช่น การเพิ่มตัวแปรที่เป็น Complex Number นอกเหนือจากตัวแปรปกติคือ int, float , char ที่ใช้อยู่

การ Binding ของ C++ ใช้หลักของ Dynamic Binding ที่เรียกว่า Virtual Binding ซึ่งทำได้โดยการสร้างฟังก์ชันใน Parent Class ให้ใช้คำว่า virtual แล้วสร้าง ฟังก์ชันสมาชิกชื่อเดียวกันใน Class ลูก เมื่อมีการส่ง Message ชื่อเดียวกับกับชื่อเดียวกับชื่อของ ฟังก์ชันสมาชิกมายัง Parent Class ในขณะที่โปรแกรมกำลังปฏิบัติงาน โปรแกรมจะเลือกเอาฟังก์ชันสมาชิกของ Class ลูก Class มาใช้งานตามที่ต้องการ

คำถามท้ายบท

1. จงอธิบายถึงลักษณะการออกแบบโปรแกรมชนิด Traditional Program
2. จงอธิบายถึงลักษณะการออกแบบโปรแกรมชนิด Object Oriented Programming
3. ข้อดีของการโปรแกรมแบบ Object Oriented Programming มีอะไรบ้าง
4. จงอธิบายถึงความสัมพันธ์ระหว่าง Class และ Object
5. กำหนดให้รูปภาพ 1 รูปคือ Object แล้วให้อธิบายถึง Class Variable และ Instance Variable
6. หลักการของ Encapsulation คืออะไร และมีประโยชน์อย่างไรในเรื่องของการออกแบบโปรแกรม
7. ลักษณะการสั่งงานบน Window นั้นจะเป็นรูปแบบของ Event (Message) Driven Program ให้อธิบาย โดยการยกตัวอย่างการทำงานของ Software บน Window ประกอบ
8. ลักษณะของ Pure Hybrid นั้นจะเป็นอย่างไร
9. การ เชื่อมโยง (Binding) ในภาษา Third Generation Language นั้นเป็นลักษณะใดมีข้อหรือเสียอย่างไร
10. การใช้รูปแบบโดย Object Oriented Programming นั้นจะมีคุณลักษณะหนึ่งคือ การนำกลับมาใช้ใหม่ (Reuse) ลักษณะดังกล่าวนี้จะไปเกี่ยวกันกับ Class Library อย่างไร