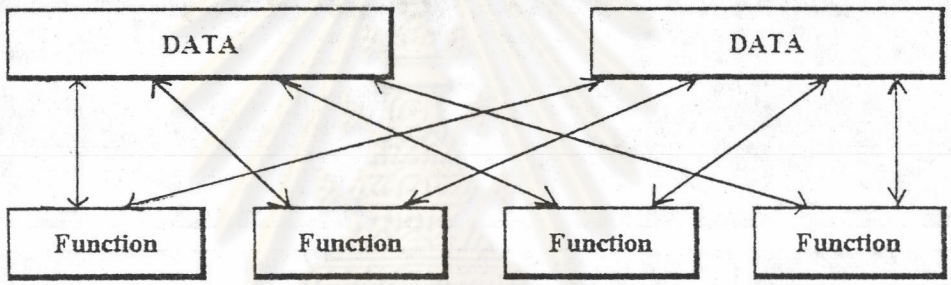
บทที่ 2
หลักการของการโปรแกรมเชิงวัตถุ

การพัฒนาโปรแกรมในปัจจุบัน จะยึดแนวทางการเขียนโปรแกรมแบบเป็นโครงสร้าง (Structure Programming) ซึ่งจะสนใจการจัดคำสั่ง (Instruction) ต่าง ๆ เข้าด้วยกันเป็นฟังก์ชัน เพื่อแก้ปัญหาของงานต่าง ๆ นั่นคือจะสนใจการกระทำ (Action) มากกว่าข้อมูล (Data) โดยในการเขียนโปรแกรมจะแยกส่วนที่เป็นฟังก์ชัน และข้อมูลออกจากกัน โดยฟังก์ชันแต่ละฟังก์ชันจะถูกออกแบบมาเพื่อใช้กับโครงสร้างข้อมูลที่กำหนดไว้เท่านั้น จากวิธีการนี้จะทำให้เกิดปัญหาในการพัฒนาโปรแกรม โดยเฉพาะโปรแกรมที่มีขนาดใหญ่ โดยโปรแกรมอาจจะมีข้อมูลที่ต้องใช้ร่วมกันระหว่างหลายฟังก์ชัน ก็จะกำหนดให้ข้อมูลตัวนั้นถูกเรียกใช้จากฟังก์ชันหลาย ๆ ฟังก์ชัน ดังรูปที่ 2.1

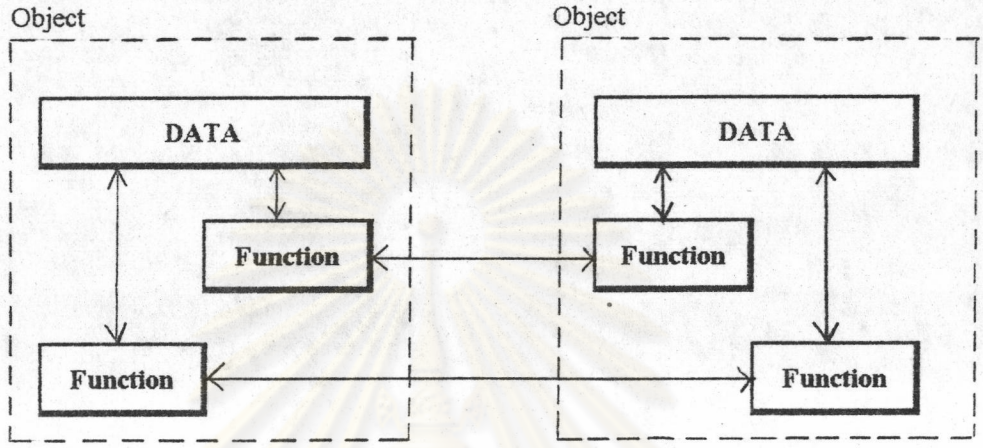


รูปที่ 2.1 การใช้ข้อมูลร่วมกันระหว่างหลายฟังก์ชัน ในการเขียนโปรแกรมแบบเป็นโครงสร้าง

ปัญหาที่เกิดขึ้นก็คือข้อมูลจะถูกเปลี่ยนแปลงโดยฟังก์ชันต่างๆ ทำให้การทดสอบและแก้ไขโปรแกรมทำได้โดยลำบาก และถ้ามีการแก้ไขโครงสร้างข้อมูล ก็อาจจะต้องแก้ไขโปรแกรมเกือบทั้งหมด ซึ่งการเขียนโปรแกรมโดยใช้หลักการของการโปรแกรมเชิงวัตถุจะช่วยแก้ปัญหานี้ได้

การโปรแกรมเชิงวัตถุไม่ใช่ภาษาโปรแกรม (Programming Language) แต่เป็นแนวคิดที่เกี่ยวข้องกับการพิจารณาข้อมูล ฟังก์ชัน และความสัมพันธ์ระหว่างข้อมูลกับฟังก์ชันนั้น ซึ่งแนวความคิดนี้สามารถนำไปใช้งานได้กับทุกภาษาโปรแกรม โดยหลักการของการโปรแกรมเชิงวัตถุจะมีประโยชน์ในการพัฒนาโปรแกรมที่มีขนาดใหญ่ ซึ่งอาจจะมีฟังก์ชันเป็นร้อย ๆ ฟังก์ชัน การเขียนโปรแกรมโดยใช้หลักการของการโปรแกรมเชิงวัตถุจะช่วยให้การเขียนโปรแกรม การแก้ไขโปรแกรม และการบำรุงรักษาโปรแกรม ทำได้สะดวกขึ้น

การเขียนโปรแกรมโดยใช้หลักการของการโปรแกรมเชิงวัตถุ จะช่วยแก้ปัญหาของการเขียนโปรแกรมแบบเป็นโครงสร้าง โดยหลักการที่สำคัญของการโปรแกรมเชิงวัตถุก็คือการผนึกข้อมูล และฟังก์ชันเข้าไว้ด้วยกันให้กลายเป็นวัตถุ (Object) ซึ่งการผนึกข้อมูล และฟังก์ชันเข้าด้วยกันนี้เรียกว่า เอนแคป ซูเลชั่น (Encapsulation) โดยข้อมูลที่อยู่ในวัตถุจะถูกเรียกใช้จากฟังก์ชันที่อยู่ในวัตถุเท่านั้น ถ้าฟังก์ชันที่อยู่ในวัตถุหนึ่งต้องการจะใช้ข้อมูลที่อยู่ในอีกวัตถุหนึ่ง จะต้องเรียกผ่านฟังก์ชันที่อยู่ในวัตถุนั้นเท่า นั้นดังรูป ที่ 2.2



รูปที่ 2.2 การเรียกใช้ข้อมูลระหว่างวัตถุ 2 วัตถุ

จะเห็นได้ว่าหลักการของการโปรแกรมเชิงวัตถุ จะช่วยแก้ปัญหาในการเขียนโปรแกรมแบบเป็นโครงสร้าง เนื่องจากข้อมูลจะถูกเปลี่ยนแปลงโดยฟังก์ชันที่อยู่ในวัตถุนั้นเท่านั้น และเมื่อมีการแก้ไขโครงสร้างข้อมูลใหม่ ผู้พัฒนาโปรแกรมก็เพียงแต่แก้ไขเฉพาะฟังก์ชันที่เกี่ยวข้องที่อยู่ในวัตถุนั้นเท่านั้น

ในการศึกษาหลักการของการโปรแกรมเชิงวัตถุ ควรจะรู้จักคำศัพท์ที่เกี่ยวข้องกับการโปรแกรมเชิงวัตถุ ดังต่อไปนี้

1) วิธีการ (Method) คือฟังก์ชันที่อยู่ประจำในวัตถุนั้น

2) การซ่อนข้อมูล (Data Hiding) คือการกำหนดให้ข้อมูลที่อยู่ในวัตถุหนึ่ง ไม่สามารถเรียกใช้โดยตรงจากวัตถุอื่น

3) ข้อความ (Message) จะเป็นสิ่งที่วัตถุใช้ติดต่อกัน โดยข้อความที่วัตถุหนึ่งส่งไปให้อีกวัตถุหนึ่งจะประกอบด้วยตัวเลือก (Selector) และอาร์กิวเมนต์ (Argument) ต่าง ๆ (ถ้ามี) โดยข้อความจะกระตุ้นให้วัตถุทำงานตอบสนอง (Response) ต่อข้อความนั้น

4) โพลีมอร์ฟิซึ่ม (Polymorphism) คือการที่วัตถุต่างชนิดกันตอบสนองต่อข้อความเดียวกันด้วย วิธีการที่แตกต่างกัน

5) คลาส คือกลุ่มของวัตถุที่มีคุณสมบัติพื้นฐานร่วมกัน

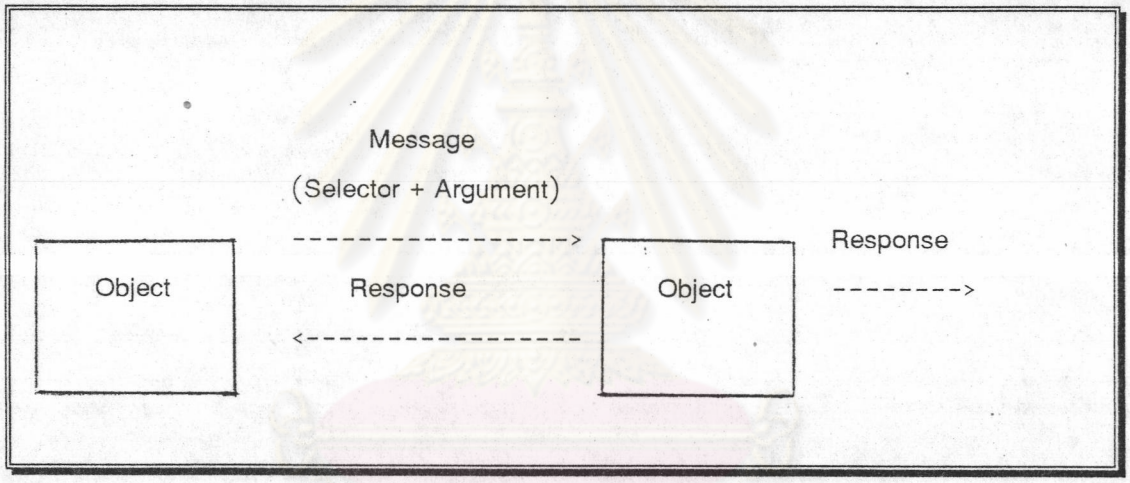
6) ดีไรฟ์คลาส (Derived Class) คือคลาสที่เกิดจากการจัดกลุ่มของวัตถุที่อยู่ในคลาสเดิม โดยคลาสที่เกิดขึ้นใหม่นี้ก็จะมีคุณสมบัติพื้นฐานของคลาสเดิม และจะมีคุณสมบัติที่เป็นคุณสมบัติเฉพาะของวัตถุที่อยู่ในดีไรฟ์คลาสนี้

7) เบสคลาส (Base Class) คือคำเรียกชื่อคลาสที่เป็นคลาสพื้นฐานของดีไรฟ์คลาส

8)อินเฮอริแทนซ์ (Inheritance) คือการถ่ายทอดคุณลักษณะร่วมจากเบสคลาสลงมายังดีไรฟ์คลาส

สำหรับคำอธิบายเพิ่มเติมของคำศัพท์ต่าง ๆ ที่ได้กล่าวมาแล้ว จะแสดงได้ดังต่อไปนี้

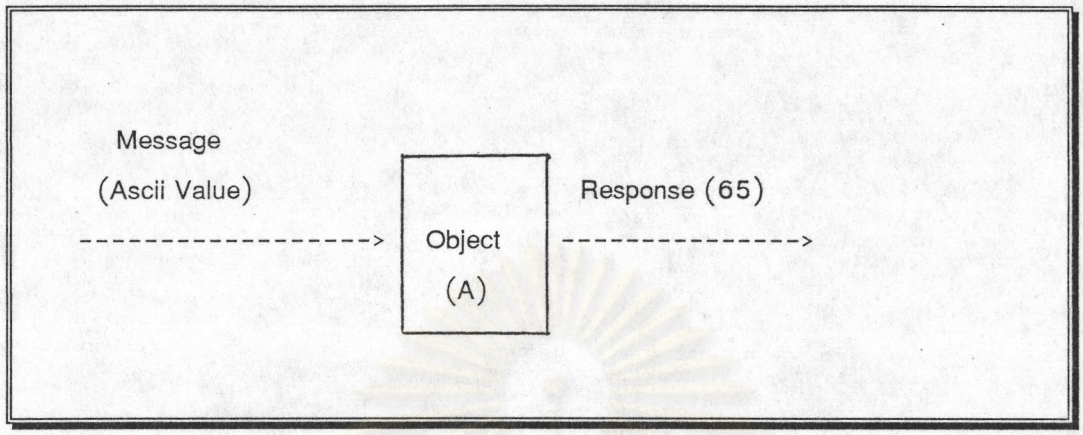
การส่งข้อความระหว่างวัตถุสามารถแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 การติดต่อกันระหว่างวัตถุ

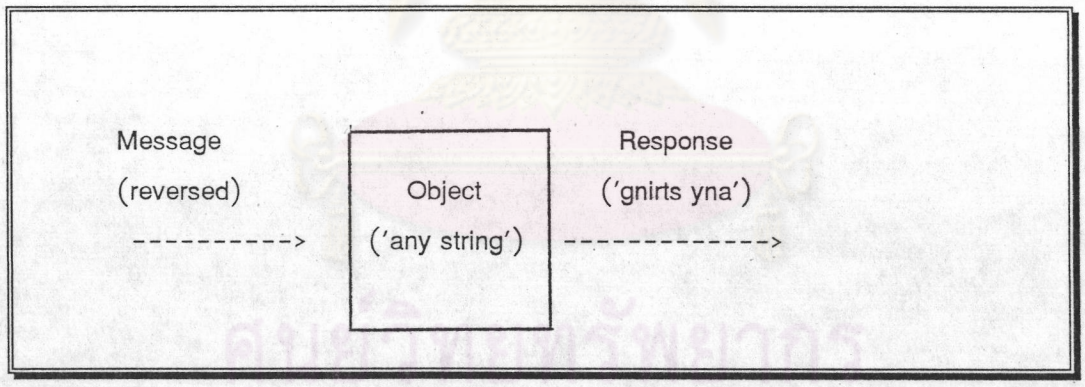
ตัวอย่างของวัตถุ ข้อความ และการตอบสนอง สามารถแสดงได้ดังตัวอย่าง ต่อไปนี้

ตัวอย่างที่ 1



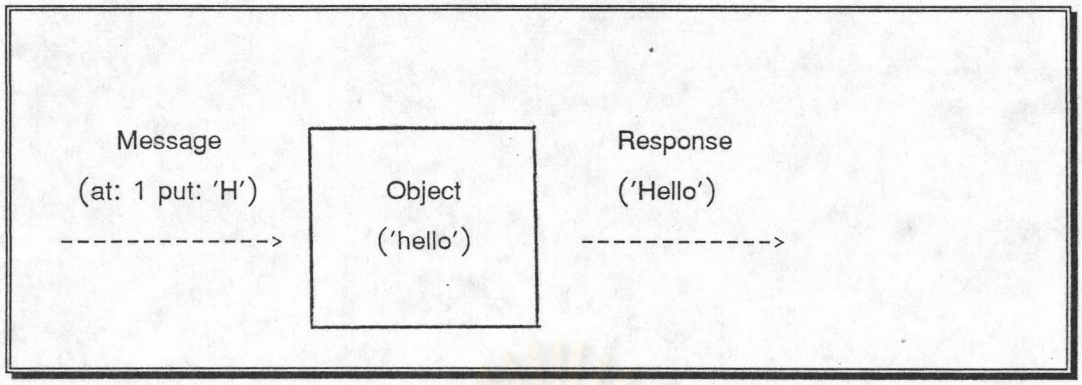
จากตัวอย่างนี้ข้อความจะประกอบด้วยตัวเลือกคือ Ascii Value ซึ่งหมายถึง ให้วัตถุส่งค่ารหัสแอสกีของข้อมูลในวัตถุนั้นออกมา ในตัวอย่างนี้ค่าที่อยู่ในวัตถุคือ A ดังนั้นวัตถุจะตอบสนองต่อข้อความโดยให้ค่า 65 ออกมา

ตัวอย่างที่ 2



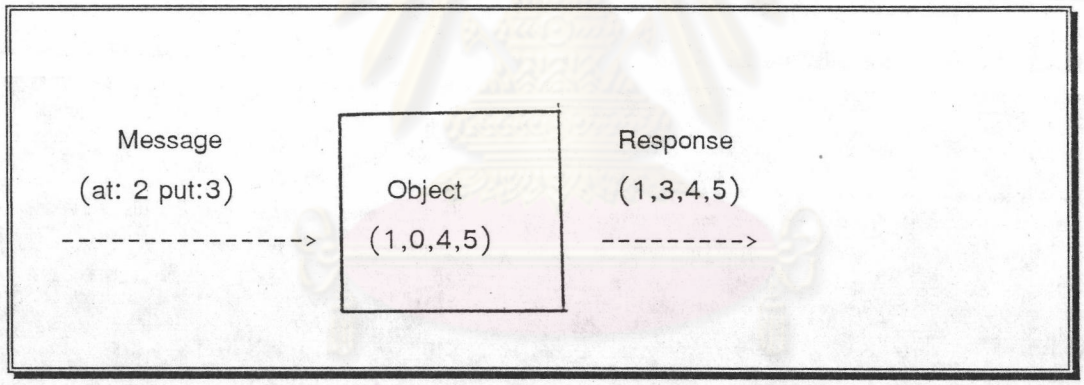
จากตัวอย่างนี้ข้อความประกอบด้วยตัวเลือกคือ reversed ซึ่งหมายถึงให้เรียงลำดับของข้อมูลที่อยู่ในวัตถุนั้นแบบย้อนกลับ ในที่นี้ข้อมูลในวัตถุคือคำว่า 'any string' ดังนั้นวัตถุจะตอบสนองต่อข้อความโดยจัดเรียงคำว่า 'any string' ให้เป็น 'gnirts yna'

ตัวอย่างที่ 3



ในตัวอย่างนี้ข้อความจะประกอบด้วยตัวเลือกคือ at: put: และอาร์กิวเมนต์คือ 1 และ 'H' ซึ่งข้อความนี้จะมีความหมายว่า ให้แทนที่ตำแหน่งที่ 1 ของข้อมูลในวัตถุด้วย 'H' ดังนั้นวัตถุจะตอบสนองต่อข้อความนี้ ด้วยการแทนค่าตำแหน่งแรกของคำว่า 'hello' ด้วย 'H' ซึ่งจะได้ผลลัพธ์เป็น 'Hello'

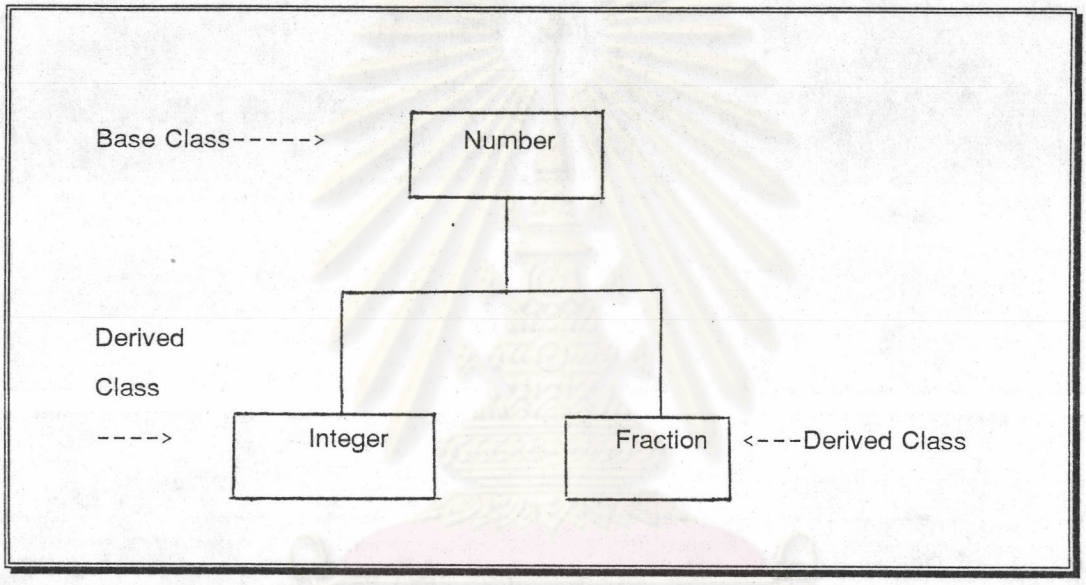
ตัวอย่างที่ 4



ในตัวอย่างนี้ข้อความจะประกอบด้วยตัวเลือกคือ at: put: และอาร์กิวเมนต์คือ 2 และ 3 ซึ่งข้อความนี้ จะมีความหมายว่า ให้แทนที่ตำแหน่งที่ 2 ของข้อมูลในวัตถุด้วย 3 ดังนั้นวัตถุจะตอบสนองต่อข้อความนี้ ด้วยการแทนค่าสมาชิกตัวที่ 2 ของอะเรย์ (Array) ด้วย 3 ซึ่งจะได้ผลลัพธ์เป็น (1,3,4,5)

โพลีมอร์ฟิซึ่มจะดูได้จากตัวอย่างที่ 3 และตัวอย่างที่ 4 ที่กล่าวมาแล้ว ซึ่งจะเห็นว่ามีการส่งข้อความเดียวกันคือ at: put: ไปยังวัตถุในตัวอย่างทั้งสอง แต่วัตถุในตัวอย่างที่ 3 จะรู้ว่าข้อมูลที่อยู่ในตัวมันเป็นคำในภาษาอังกฤษ และจะแปลข้อความว่าต้องนำ 'H' ไปแทนที่ตัวอักษรตัวแรกในคำว่า 'hello' ส่วนวัตถุในตัวอย่างที่ 4 จะรู้ว่าข้อมูลที่อยู่ในตัวมันเป็นอะเรย์ และจะแปลข้อความว่า จะต้องแทนที่สมาชิกตัวที่ 2 ของอะเรย์ด้วย 3

ในหลักการของการโปรแกรมเชิงวัตถุนั้น จะสามารถจัดวัตถุต่างๆ ให้อยู่รวมกันเป็นคลาสได้ การจะจัดให้วัตถุใดอยู่ในคลาสใด ก็ขึ้นอยู่กับว่าวัตถุนั้นมีคุณสมบัติร่วมของคลาสนั้นหรือไม่ นอกจากนี้ยังสามารถจัดกลุ่มของวัตถุที่อยู่ในคลาสให้เป็นคลาสอีกคลาสหนึ่งได โดยคลาสที่เกิดจากกลุ่มของวัตถุที่อยู่ในคลาสเดิมนั้น จะเรียกว่าดีไรฟ์คลาส ส่วนคลาสเดิมจะเรียกว่าเบสคลาส สำหรับความสัมพันธ์ ระหว่างเบสคลาสและ ดีไรฟ์คลาสสามารถแสดงได้ดังรูปที่ 2.4

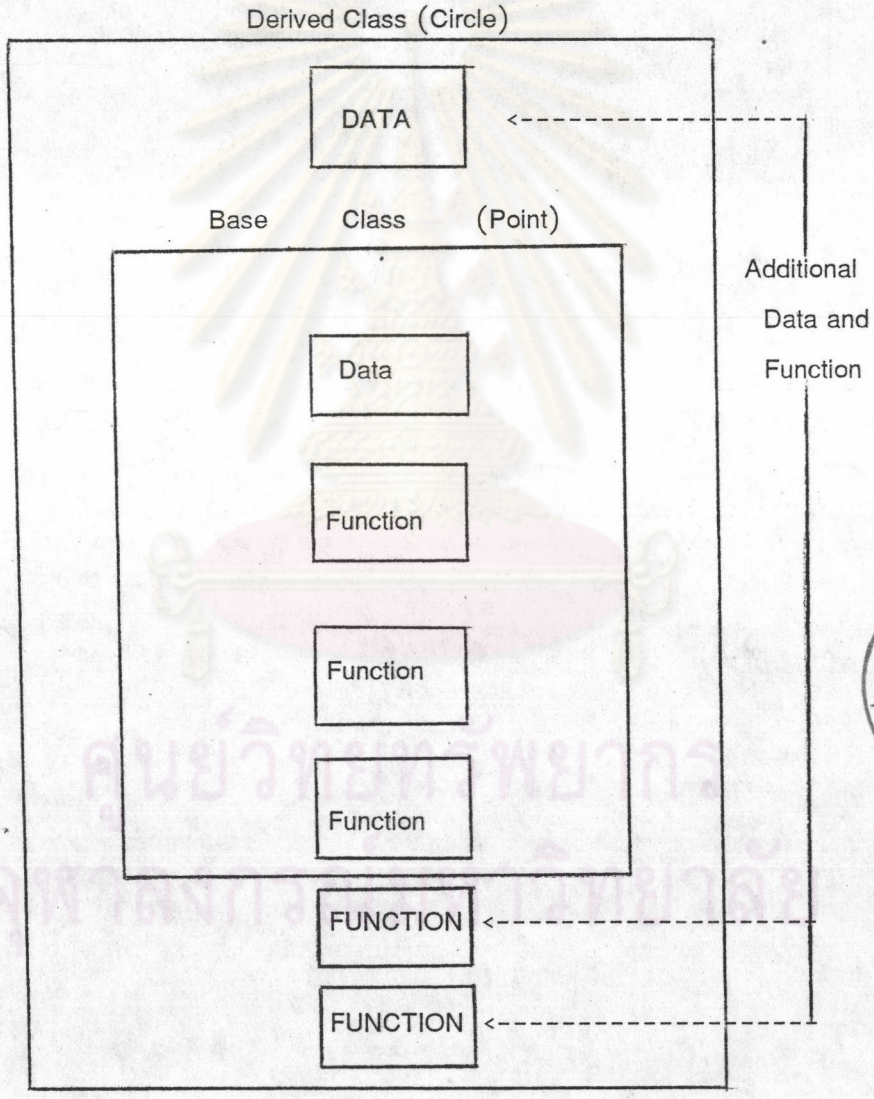


รูปที่ 2.4 ความสัมพันธ์ระหว่างเบสคลาส และดีไรฟ์คลาส

จากรูปจะเห็นว่าเบสคลาส "Number" จะมีดีไรฟ์คลาสคือ "Integer" และ "Fraction"

คุณสมบัติที่สำคัญอีกประการหนึ่งของการโปรแกรมเชิงวัตถุคือ อินเฮอร์ริแตนซ์ ซึ่งก็คือการถ่ายทอดคุณลักษณะร่วมจากเบสคลาสลงมายังดีไรฟ์คลาสนั้น เช่นสมมติว่าในคลาส "Number" มีวิธีการ ที่ตอบสนองต่อข้อความ "Plus" ซึ่งมีความหมายว่าให้เพิ่มค่าของข้อมูลที่อยู่ในวัตถุ แต่คลาส "Integer" และ คลาส "Fraction" ไม่มีวิธีการที่ตอบสนองต่อข้อความนี้ สมมติว่าวัตถุวัตถุหนึ่งในคลาส "Integer" มีข้อมูล เป็น 2 และได้รับข้อความ "Plus 3" การทำงานจะเริ่มจาก คอมพิวเตอร์จะหาวิธีการสำหรับข้อความ "Plus" ในคลาส "Integer" เมื่อหาไม่พบจึงไปค้นดูในเบสคลาส "Number" ซึ่งก็จะพบวิธีการสำหรับข้อความ "Plus" และทำการประมวลผล โดยเพิ่มข้อมูลที่อยู่ในวัตถุให้เป็น 5 เราจะกล่าวว่า วิธีการสำหรับ ข้อความ "Plus" นั้น ถูกอินเฮอร์ริทจากคลาส "Number" ไปยังคลาส "Integer" และในทำนองเดียวกัน วิธีการ สำหรับข้อความ "Plus" ก็จะถูกอินเฮอริทจากคลาส "Number" ไปยังคลาส "Fraction" ด้วย

จากคุณสมบัติของการอินเฮอร์ริแตนซ์นี้ จะทำให้สามารถสร้างคลาสใหม่จากคลาสที่มีอยู่แล้ว ซึ่งจะเป็นการลดความซ้ำซ้อนในการเขียนโปรแกรมขึ้นมาใหม่ เช่นถ้ามีคลาส "Point" ซึ่งเป็นคลาสสำหรับจุดแต่ละจุดที่แสดงบนจอภาพ โดยจะมีโครงสร้างข้อมูลคือ ตำแหน่งของจุด สีของจุด และมีวิธีการคือ การกำหนดตำแหน่งของจุด การกำหนดสีของจุด และการวาดจุดลงบนจอภาพ ต่อมาถ้าต้องการสร้างคลาสสำหรับการวาดวงกลมบนจอภาพ โดยการวาดวงกลมจะต้องใช้จุดศูนย์กลาง และรัศมี ซึ่งจะเห็นว่าสำหรับการกำหนดจุดศูนย์กลางนั้น สามารถใช้โครงสร้างของข้อมูล และวิธีการจากคลาส "Point" ได้ สิ่งที่ต้องทำก็คือ กำหนดให้คลาสสำหรับการวาดภาพวงกลม สมมติว่าชื่อคลาส "Circle" เป็นดีไรฟ์คลาสของคลาส "Point" และเพิ่มโครงสร้างข้อมูลคือ รัศมีของวงกลม และเพิ่มวิธีการคือ การวาดภาพวงกลมลงบนจอภาพ สำหรับความสัมพันธ์ระหว่างคลาส "Point" และคลาส "Circle" จะแสดงดังรูปที่ 2.5 และตัวอย่างของการนิยามคลาสทั้งสองโดยใช้ภาษา C++ จะแสดงในรูปที่ 2.6



รูปที่ 2.5 ความสัมพันธ์ระหว่างคลาส "Point" และคลาส "Circle"

```
class Point
{
    private:
        int loc_x, loc_y;       // location of point
        int color;              // color of point
    public:
        void set_loc( int x, int y );      // set location
        void set_color( int c );           // set color
        void draw( void );           // draw point
};
class Circle:public Point
{
    private:
        int radius;              // radius of circle
    public:
         void set_radius( int r );
         void draw( void );          // draw circle
};
```

รูปที่ 2.6 การนิยามคลาส "Point" และคลาส "Circle" โดยใช้ภาษา C++