การโปรแกรมเชิงวัตถุ

สมพันธุ์ ชาญศิลป์[1]

เราเคยตื่นเต้นกับการเขียนโปรแกรมด้วยภาษา FORTRAN ต่อมาตื่นเต้นกับภาษา BASIC ที่มีคำสั่งง่ายๆ จากนั้นก็ตื่นเต้นกับภาษา PASCAL และภาษา C ที่เป็นภาษาโครงสร้าง ซึ่งจะขอใช้คำย่อว่า PP (Procedural Programming) จนในปัจจุบัน OOP (Object-oriented Programming) กำลังเป็นที่กล่าวขานกันมากในวงการโปรแกรมมิ่ง ประหนึ่งว่าต่อไปอีกไม่นาน ทุกคนจะเขียนโปรแกรมแบบ OOP กันทั้งหมด และหวังว่าวงการซอฟท์แวร์ (software) จะพัฒนาตนเองอย่างรวดเร็วเพราะคุณสมบัติพิเศษของ OOP นี้

เรื่องของ OOP ถูกตีพิมพ์อย่างแพร่หลายในวารสารทางคอมพิวเตอร์ในช่วง 1 - 2 ปีมานี้ แต่ผู้ที่รู้เรื่องและเข้าใจมีสักกี่คน ไม่มีใครทราบว่าผู้ที่กำลังฝึกหัด การเขียนโปรแกรมแบบ OOP มีมากน้อยเท่าใด แต่ผู้ที่ใหม่ต่อการเขียนโปรแกรมก็ดี หรือที่มีประสบการณ์ชำชองก็ดีมีสิทธิ์ที่จะรู้จัก OOP กันทั่วหน้า และถ้าหลายๆ คนหันมาสนใจ OOP, มาเขียนโปรแกรมแบบ OOP, มาพูดมาคุยเรื่อง OOP กันมากๆ เชื่อแน่ว่า ในอีกไม่นานวงการซอฟท์แวร์บ้านเรา จะเจริญรุดหน้าอย่างรวดเร็วกว่าที่เป็นอยู่

ในบทความนี้ จะนำเสนอเรื่องราวของ OOP ด้วย คำพูดง่ายๆ ตรงไปตรงมา พร้อมตัวอย่างและคำอธิบายอย่างชัดเจน ด้วยความหวังว่าส่วนที่ยังสงสัยหรือข้องใจ อยากรู้เกี่ยวกับ OOP จะได้ถูกขจัดออกไปในที่สุด โดย ได้แบ่งเนื้อหาเป็นลำดับ ตั้งแต่ความเป็นมาของ OOP, ลักษณะของ OOP, เปรียบเทียบ OOP กับ PP, ตัวอย่างโปรแกรมพร้อมคำอธิบาย ไปจนถึงคำแนะนำสำหรับ ผู้ที่สนใจ OOP หวังว่าผู้ที่อ่านจนจบจะกลายมาเป็น OOP ฟีเวอร์ในเร็ววัน

ความเป็นมาของ OOP

ย้อนกลับไปสู่อดีตเมื่อประมาณ 15 - 20 ปีที่ผ่านมา ผู้ที่ไปศึกษา ณ ต่างประเทศจะมีโอกาสเรียนการเขียนโปรแกรมทางคอมพิวเตอร์ ซึ่งในสมัยนั้นใช้ภาษา FORTRAN นักศึกษาจะวุ่นอยู่กับการเขียนโปรแกรมตอกบัตร ป้อนบัตรเข้าเครื่องอ่าน รอผลลัพธ์และเมื่อมีข้อผิดพลาดก็ไปแก้ไขโดยการตอกบัตรใหม่ ป้อนบัตรเข้าเครื่องอ่านและรอผลอีก นักศึกษาคอมพิวเตอร์ส่วนมากจะวนเวียนอยู่บริเวณห้องตอกบัตร ที่เครื่องอ่านและที่เครื่องพิมพ์ (Line Printer) เป็นประจำ

นักศึกษาบางคนที่สนใจด้านคอมพิวเตอร์ เมื่อศึกษาเพิ่มเติมจะได้ศึกษาทางด้านฮาร์ดแวร์ (Hardware) โดยอาจเริ่มต้นจากการศึกษาภาษาแอสเซมบลี (Assembly Language) ไปพร้อมๆ กับภาษาเครื่อง (Machine Language) ในระยะต่อมาเมื่อ บริษัท คอมพิวเตอร์ IBM ผลิต PC (IBM Personal Computer) ออกมาขาย หลายคนอาจจะไม่มีเงินซื้อเพราะช่วงแรก ราคาแพงมาก รอมาอีกสักระยะก็ได้มี Home Computer ออกมาขาย ภายใต้ชื่อ Timex Sinclair ที่มีราคา 99 เหรียญ ในช่วงนี้หลายๆ คน ได้มีโอกาสซื้อมาเล่นและได้ฝึกการเขียนโปรแกรมภาษา BASIC ควบคู่ไปกับการใช้ภาษา

---

[1] *อาจารย์ สาขาวิชาเทคโนโลยีคอมพิวเตอร์ สำนักวิชาเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง นครราชสีมา 30000*

เครื่อง ในระยะต่อมาเมื่อราคาเครื่อง APPLE และเครื่อง IBM PC ลดต่ำลง หลายๆ คนก็ได้มีโอกาสได้ใช้ฝึกฝน การเขียนโปรแกรมด้วยภาษา BASIC ที่ติดมากับ ROM ใช้คอมไพเลอร์ภาษา FORTARN, ภาษา PASCAL และภาษา C ตามลำดับ

เมื่อเราได้ศึกษาและได้สัมผัสกับการเขียนโปรแกรมด้วยภาษาต่างๆ ด้วยตนเองมาโดยตลอดแล้ว เราก็จะมองเห็นขั้นตอน การเกิดขึ้นและวิธีการแก้ปัญหาของแต่ละภาษาดังกล่าวมาได้อย่างชัดเจน เริ่มจากสมัยแรกๆ ช่วงนั้นกำลังตื่นเต้นกับความสามารถของ IBM Mainframe ในการใช้แก้ปัญหาทางการคำนวณด้านต่างๆ ภาษา FORTRAN จึงถือกำเนิดขึ้น เพื่อใช้เป็นภาษาที่จัดการกับการคำนวณโดยเฉพาะ ในขณะที่ FORTRAN เป็นภาษาที่ใช้แพร่หลายนั้น คนหลายระดับก็เกิดมีการอยากเรียนรู้ภาษาสั่งงานคอมพิวเตอร์กันมากขึ้นแต่ปัญหาก็คือ ภาษา FORTRAN มีรายละเอียดของภาษามาก คือเรียนรู้ได้ยากและดูจะไม่เหมาะสมสำหรับบุคคลทั่วๆ ไปที่ไม่ได้ศึกษาทางวิทยาศาสตร์ นี่เองภาษา BASIC จึงถือกำเนิดขึ้นมาโดยมีพื้นฐานของภาษาที่มีคำสั่งง่ายๆ เหมาะสำหรับผู้เริ่มต้นใหม่ทั่วไป และในขณะเดียวกันที่มี IBM PC และ Home Computer ยี่ห้ออื่นๆ ออกสู่ท้องตลาดภาษา BASIC ก็กลายเป็นภาษาที่ติดมากับเครื่องเพราะเป็นภาษาที่ทุกคนสามารถใช้สั่งงานได้ง่ายที่สุด ถ้าใครเคยเขียนโปรแกรมทั้งของภาษา FORTRAN และภาษา BASIC ในระยะต้นๆ จะเห็นว่า เราไม่ต้องคำนึงถึงหลักเกณฑ์ในการเขียนโปรแกรมมากนักเพราะมีเพียง 3 ขั้นตอนเท่านั้น คืออ่านข้อมูลเข้ามาคำนวณแล้วก็แสดงผลออกไป ในสมัยแรกๆ โปรแกรมที่เขียนขึ้นมักสั้น มีขนาดเล็ก เราจึงเขียนโดยไม่คำนึงถึงรูปแบบเราใช้คำสั่ง GOTO กันอย่างฟุ่มเฟือย จนในระยะต่อมาเมื่อโปรแกรมที่เขียนเริ่มโตขึ้น การเขียนโปรแกรมที่ไม่เป็นระเบียบ เริ่มมีปัญหา นั่นคือเวลามีข้อผิดพลาดจะต้องใช้เวลามากในการค้นหาคำสั่งที่ผิด หรือส่วนที่จะต้องแก้ไขเพิ่มเติม ยิ่งกว่านั้น เมื่อโปรแกรมมีคำสั่งที่ไม่เป็นระเบียบก็ย่อมยากที่บุคคลอื่นจะมาศึกษาการทำงานของโปรแกรมที่เขียนไว้แล้ว ในยุคนี้เอง ภาษา PASCAL เริ่มเข้ามาสู่วงการการศึกษา โดยได้เสนอตัวว่าเป็นภาษาโครงสร้าง (Structural Language) หรือบางครั้งเรียกว่า ภาษาแบบ โพรซีเจอร์ (Procecural Language) นั่นคือเป็นภาษาที่มีการแก้ปัญหาแบบบนลงล่าง (top-down design) คือ จากกว้างไปหาแคบ จากโพรซีเจอร์ใหญ่ไปหาโพรซีเจอร์ย่อย มีการแบ่งแต่ละขั้นตอน การแก้ปัญหาเป็นโพรซีเจอร์หรือฟังก์ชัน (function) ซึ่งจะทำให้การหาข้อผิดพลาดทำได้ง่ายขึ้น โดยการตรวจสอบแต่ละฟังก์ชัน จากฟังก์ชันใหญ่ไปหาฟังก์ชันเล็ก และการทำความเข้าใจการทำงานของโปรแกรม ก็เป็นไปได้ง่ายเช่นกัน ด้วยวิธีดูกว้างๆ ก่อน แล้วจึงไปดูรายละเอียดในแต่ละส่วนทีหลัง ซึ่งวิธีนี้เราเรียกว่าแบบบนลงล่างนั่นเอง PASCAL จึงเป็นภาษาที่หลายๆ มหาวิทยาลัยทั้งในและต่างประเทศให้นักศึกษาเรียนแทนภาษา FORTRAN แต่อย่างไรก็ตาม ต่อมาภาษา FORTRAN ก็ดี BASIC ก็ดีได้เริ่มถูกปรับปรุงให้เป็นภาษาที่มีการสั่งงานเป็นแบบโพรซีเจอร์เหมือนกัน จึงทำให้ FORTRAN ยังใช้อยู่ในบางมหาวิทยาลัยและ BASIC ยังเป็นที่นิยมของคนทั่วๆ ไปมาจนทุกวันนี้

PASCAL เป็นที่นิยมมาไม่กี่ปี ระบบปฏิบัติการ UNIX เริ่มขยับจาก Minicomputer (VAX) มาสู่ Workstation และมาสู่ PC ในที่สุด ในช่วงนี้เองภาษา C (C Language) เริ่มเข้ามาเป็นที่รู้จัก และเป็นที่สนใจของนักเขียนโปรแกรมทั้งหลาย ทั้งนี้เพราะระบบปฏิบัติการ UNIX เขียนขึ้นด้วยภาษา C ภาษา C จึงเป็นภาษาคอมพิวเตอร์ที่เกิดขึ้นมาทีหลังสุด และเป็นภาษาแบบโพรซีเจอร์เหมือน PASCAL แต่ที่ดีกว่า คือเป็นภาษาที่สั่งงานได้เร็วกว่า สามารถติดต่อกับส่วนต่างๆ ของคอมพิวเตอร์ในระดับฮาร์ดแวร์ (Hardware) ได้ดีกว่า ในระยะ 2 - 3 ปีมานี้ C จึงเป็นที่นิยมกันมากทั่วโลก บางมหาวิทยาลัย ทั้งในสหรัฐอเมริกาและประเทศไทยเราด้วย ถึงกับให้เรียนในวิชา Computer Programming for Engineers & Scientists กันแล้ว ถ้าเราหันมาดูในแง่ของผู้เขียนโปรแกรมด้วยภาษาต่างๆ เหล่านี้ เราจะพบว่าในระยะแรกๆ ที่ใช้ภาษา FORTRAN หรือภาษา BASIC เราจะคำนึงเพียงว่า เราจะอ่านข้อมูลเข้ามาอย่างไร เราจะทำการคำนวณอย่างไร และจะแสดงผลลัพธ์ในรูปของตัวเลขอย่างไรเท่านั้น (เราไม่ต้องคิดมาก) แต่ต่อมาเมื่อเราใช้ภาษา PASCAL หรือ C เขียนโปรแกรม เราเริ่มต้องคำนึงถึงหลายสิ่งหลายอย่าง คือต้องมีการออกแบบและวางแผน มีการแบ่งขั้นตอนการสั่งงานออกเป็นส่วนๆ การป้อนข้อมูลเข้ามา อาจอยู่ในรูปของตัวเลข

อาจเกิดจากการเลื่อนแถบสว่าง อาจเกิดจากการใช้เม้าส์ (Mouse) ส่วนการแสดงผลอาจอยู่ในรูปลักษณะเป็นตัวเลข เป็นเสียง รวมไปถึงรูปภาพที่มีสีสันต่างๆ โปรแกรมที่ถูกผลิตขึ้นมาในระยะหลังๆ เริ่มเป็นลักษณะ GI (Graphic Interface) คือเป็นลักษณะของรูปภาพเสียส่วนมาก เมื่อเป็นดังนี้ โปรแกรมของเราเริ่มโตขึ้นเรื่อยๆ ความสลับซับซ้อนก็มีมากขึ้นเป็นเงาตามตัว เราเริ่มมีปัญหาในการแก้ไขข้อผิดพลาดในการเรียนรู้การทำงานของโปรแกรม ในการเพิ่มเติมและเปลี่ยนแปลงโปรแกรม รวมไปถึงการนำมาใช้ใหม่ (Reusability) ของโปรแกรม เหตุผลเหล่านี้เองจึงเป็นที่มาของภาษาโปรแกรมในแนวใหม่ที่เรียกว่า OOP (Object-oriented programming) ขอเน้นในที่นี้ว่า OOP ไม่ใช่ภาษาใหม่ หากแต่เป็นภาษาเก่าภาษาไหนก็ได้ เช่น ภาษา BASIC, PASCAL หรือ C ก็สามารถนำมา ดัดแปลงเพิ่มเติมให้มันมีคุณสมบัติไปในเชิงวัตถุ (object-oriented) ได้เหมือนกัน ตัวอย่างเช่น ภาษา BASIC เกิดเป็น Object BASIC, PASCAL เป็น Object-oriented PASCAL และภาษา C เป็น C++ และ Objective C เป็นต้น

ภาษาที่มีรูปแบบของการแก้ปัญหาในแนว OOP นี้สามารถแบ่งได้เป็น 2 สาย คือสายที่เป็น OOP แท้ (pure OOP group language) เช่นภาษา Simula, Ada, Modula-2, *Smalltalk, Eiffel และ Actor เป็นต้น* สายที่สองเป็น ไฮบริด OOP (hybrid OOP group language) เช่นภาษา C++, Objective C, Object - BASIC, Object-COBOL, Common Lisp Object System (CLOS) และ Object - oriented PASCAL เป็นต้น โดยกลุ่มแรกโครงสร้างของภาษาเป็นออบเจกต์ (Object) ทั้งหมด การสั่งงานจะทำโดยการส่งข้อความ (message) ไปยังออบเจกต์ที่ต้องการ ซึ่งส่วนใหญ่อยู่ในไลบรารีของภาษาหมดแล้ว ส่วนกลุ่มที่สองเป็นการผสมผสานระหว่างภาษาแบบโพรซีเจอร์เดิมและภาษาในแนว OOP ใหม่ด้วย

ลักษณะของ OOP

ตามความรู้สึกของคนทั่วๆ ไปจะเห็นว่า OOP เป็นเรื่องแปลกใหม่ไม่รู้ว่าคืออะไร สงสัยอยากรู้จนคิดเลยไปว่ามันคงเป็นเรื่องยากเกินกว่าที่จะรู้ได้ หรือมันคงต้องใช้เวลานานมากเพื่อที่จะเรียนรู้ได้ ในสภาพเป็นจริงแล้วหลักการของเชิงวัตถุ (Object - oriented) ได้ถูกนำมาใช้แพร่หลาย และเราได้สัมผัสกันอยู่เกือบทุกวันเป็นส่วนมากนั่นคือ ระบบการติดต่อกับผู้ใช้แบบกราฟิกหรือ GI (Graphic Interface) ที่มีอยู่ใน WINDOWS, OS/2 และระบบปฏิบัติการของเครื่อง Macintosh ล้วนใช่ทั้งสิ้นโลกที่เราอยู่ก็เป็นระบบเชิงวัตถุโดยธรรมชาติอยู่แล้ว ยกตัวอย่างเช่น เราเดินเข้าไปหาแม่ค้าแล้วบอกแม่ค้าว่า "แม่ค้าขอข้าวผัด 1 จาน" จะเห็นว่าตัวเราและแม่ค้าต่างก็เป็นออบเจกต์หนึ่งๆ การเคลื่อนไหวพูดจาโต้ตอบก็คือ วิธีการสื่อสารระหว่างออบเจกต์ เพื่อทำให้ระบบดำเนินไป

ขอยกอีกสักตัวอย่างหนึ่งทางการสั่งงานคอมพิวเตอร์เพื่อให้เห็นภาพชัดเจนยิ่งขึ้น เช่น ถ้าเราต้องการจะก๊อปปี้ไฟล์ X.DOC จากไดรว์ C ไปไว้ที่ไดรว์ B เราจะใช้คำสั่งของ DOS ดังนี้

C : > COPY X.DOC B : < กด Enter>

ลักษณะของคำสั่งนี้เป็นการเรียกใช้โพรซีเจอร์ COPY ให้ทำการก๊อปปี้ไฟล์

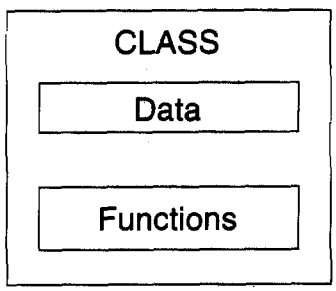
แต่ถ้าจะทำในลักษณะเดียวกันใน WINDOWS ซึ่งมีการสั่งงานในรูปแบบของเชิงวัตถุนั้น เมื่อเราเข้าไปใน File Manager เราสามารถใช้เม้าส์ติ๊กที่ไฟล์ X.DOC ของไดรว์ C แล้วลากเม้าส์ (ขณะกดปุ่มซ้ายค้างอยู่) *ไปยังเครื่องหมายของไดรว์ B แล้วปล่อยนิ้ว* ที่กดปุ่ม ไฟล์ X.DOC ก็จะถูกก๊อปปี้ไปยังไดรว์ B ตามต้องการจะเห็นว่าวิธีการหลังนี้มีการเลือกออบเจกต์ก่อน (โดยการใช้เม้าส์ไปติ๊กที่ไฟล์ X.DOC ของไดรว์ C) การลากเม้าส์เป็นการร้องขอให้มีการก๊อปปี้ออบเจกต์ที่ถูกเลือก ไปไว้ยังออบเจกต์คือไดรว์ B และก็มีการตอบสนองจนสำเร็จตามประสงค์

อย่างไรก็ตาม ที่กล่าวมาทั้งหมดเป็นเพียงตัวอย่างคร่าวๆ เท่านั้น แท้ที่จริงลักษณะของ OOP ต้องมีส่วนประกอบที่สำคัญ 3 ประการคือ

ก. Encapsulation คือ การกำหนดออบเจกต์ให้มีทั้งคุณสมบัติ (attribute) และหน้าที่เรียกว่า เมตธอด (method) หรือทั่วๆ ไปมักกล่าวว่าให้มีทั้งข้อมูล (data) และฟังก์ชันอยู่ด้วยกัน พร้อมทั้งสามารถที่จะซ่อน (hide) ข้อมูลหรือฟังก์ชันในระดับที่ต้องการได้ ซึ่งทั้งหมดนี้สำเร็จขึ้นได้ด้วยการกำหนดโครงสร้างของข้อมูลแบบใหม่ (สำหรับภาษา C++) ที่เรียกว่าคลาส (class) ดัง

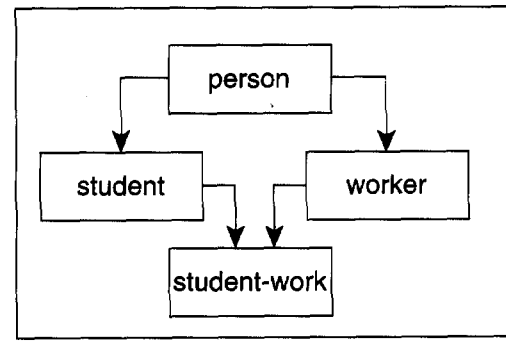
รูปที่ 1 ในระบบ OOP ออบเจกต์ต่างๆ สื่อสารกันเพื่อการร้องขอและการตอบสนองด้วยการส่งข่าวสารที่เรียกว่า เมสเสจ (message) หากัน ตัวอย่างเช่น "ในขณะที่คุณกำลังใช้งานไมโครซอฟท์วินโดวส์อยู่ คุณคือออบเจกต์หนึ่งในระบบ ในขณะที่วินโดวส์ก็เป็นอีกออบเจกต์หนึ่งเมื่อเริ่มคลิกเม้าส์.....นั่นคือ เรากำลังส่งเมสเสจให้กับวินโดวส์เพื่อรอรับผลอะไรบางอย่าง และแน่นอนที่สุด เมื่อวินโดวส์ตอบสนองนั่นหมายความ ว่ามันกำลังตอบสนองต่อเมสเสจของคุณด้วยเมตธอด ที่สร้างขึ้นเพื่อรองรับเมสเสจนั้นและเมื่อมีการตอบสนองเมสเสจครั้งหนึ่ง ก็จะมีผลให้สถานภาพของวินโดวส์เปลี่ยนแปลงไป ["Object - oriented เทคโนโลยี" ปฏิวัติวงการ : คอมพิวเตอร์รีวิว ฉบับ 96 หน้า 170)]

คลาส คือ รูปแบบ (type) ของข้อมูลและฟังก์ชันที่กำหนดขึ้น ส่วนออบเจกต์ ก็คือ ข้อมูลและฟังก์ชันจำเพาะที่มีตัวตน (instance) ที่จะปฏิบัติงานได้จริง ดูตัวอย่างที่ชัดเจนได้ในหัวข้อตัวอย่างโปรแกรม



รูปที่ 1. ส่วนประกอบของคลาส.

ข. Inheritance คือ ความสามารถกำหนดคลาสขึ้นมาใหม่ ที่สามารถสืบทอดคุณสมบัติและหน้าที่มาจากคลาสต้นแบบได้ (ขอเรียกคลาสต้นแบบว่าคลาสแม่และคลาสที่สร้างใหม่ว่าคลาสลูก) จะมีมากกว่าหนึ่งคลาสก็ได้ (ในกรณีเช่นนี้เราเรียก multiple inheritance) จากรูปที่ 2 จะเห็นว่าคลาส person เป็นคลาสแม่ของคลาสstudent และคลาส worker ในขณะเดียวกันทั้งสองคลาส หลังก็เป็นคลาสแม่ของคลาส student_work ไปพร้อมกัน วิธีการสืบทอดคุณสมบัตินี้ ทำให้เกิดระบบระดับขั้นของคลาส (class hierachy) ขึ้น ในลักษณะคลาสแม่คลาสลูก



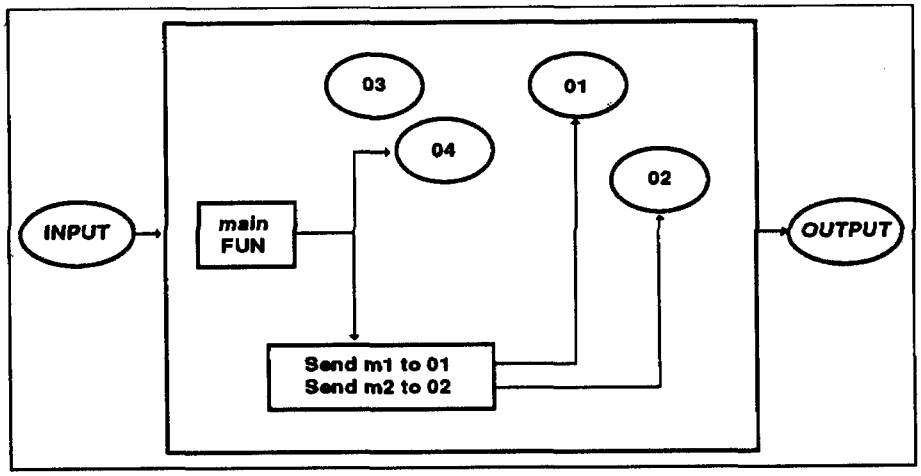
รูปที่ 2 Multiple Inheritance.

ค. Polymorphism คือความสามารถของคลาสลูกในการกำหนดฟังก์ชันที่มีในคลาสระดับสูงกว่าได้ใหม่โดยใช้ชื่อเดิม รวมไปถึงการกำหนดฟังก์ชันที่มีชื่อเดียวกันในระดับเดียวกันด้วยตัวโปรแกรมจะเป็นตัวจัดการเรียกใช้ฟังก์ชันที่เหมาะสมกับระดับต่างๆ ได้เอง

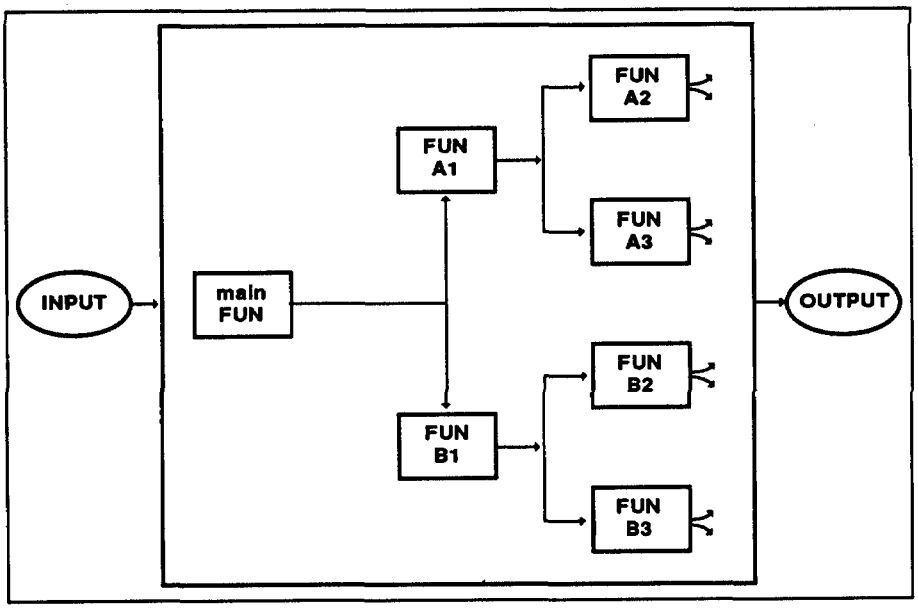
จะเห็นว่าจากคุณสมบัติของ OOP ทั้ง 3 ประการทำให้เราสรุปได้ว่า เราสามารถสร้างออบเจกต์ขึ้นมาได้ใหม่ ด้วยความสามารถที่จะนำทั้งข้อมูล และฟังก์ชันมาไว้ด้วยกัน สามารถกำหนดระดับของการเรียกใช้ทั้งข้อมูลและฟังก์ชันร่วมกันได้ (ตามคุณสมบัติข้อ ก.) และคลาสลูกสามารถสืบทอดคุณสมบัติ และหน้าที่ต่างๆ (ทั้งข้อมูลและฟังก์ชัน) มาจากคลาสแม่ได้หลายตัว (ตามคุณสมบัติข้อ ข.) พร้อมทั้งสามารถเปลี่ยนแปลงแก้ไข และเพิ่มเติม ฟังก์ชันของคลาสแม่ และคลาสที่อยู่ในระดับขั้นสูงกว่าที่สืบทอดคุณสมบัติมาได้ (ตามคุณสมบัติข้อ ค.)

เปรียบเทียบ OOP กับ PP

1. โดยธรรมชาติแล้ว OOP และ PP มีความแตกต่างกันดังนี้ คือ OOP มีข้อมูลในลักษณะคงที่ ในขณะที่ฟังก์ชันจะเป็นแบบอิสระ การติดต่อระหว่างออบเจกต์ จะเป็นไปในรูปของข่าวสาร (message) การทำงานคล้ายรูปที่ 3 ผลคือ ข้อผิดพลาดที่เกิดขึ้นจะเป็นของแต่ละออบเจกต์โดยเฉพาะ ส่วน PP มีฟังก์ชันหรือโพรซีเจอร์ ในลักษณะคงที่ (fixed) ในขณะที่ข้อมูล (data) เป็นอิสระ (mobile) การเรียกใช้ฟังก์ชันจะใช้การส่งข้อมูลเข้าไป หลังจากฟังก์ชันจัดการตามขั้นตอนที่กำหนดเรียบร้อยแล้ว ก็จะได้ผลลัพธ์ออกมา ผลลัพธ์



รูปที่ 3. การเรียกใช้ออบเจกต์ของ OOP (Dillon 1993).



รูปที่ 4. การเรียกใช้ฟังก์ชันของ PP (Dillon 1993).

อาจถูกส่งต่อไป ยังฟังก์ชันอื่นๆ อีก การทำงานคล้าย รูปที่ 4 ผลคือ ข้อผิดพลาดที่เกิดขึ้นในระหว่างขั้นตอนการเรียกใช้ฟังก์ชัน อาจทำให้ระบบต้องหยุดทำงาน

2. ผลจากความสามารถในการสืบคุณสมบัติ จากคลาสแม่ไปยังคลาสลูกของ OOP ทำให้เราสามารถนำข้อมูลหรือฟังก์ชันเดิมมาใช้ใหม่ได้ โดยไม่ต้องกำหนดขึ้นอีก โดยทั้งข้อมูลและฟังก์ชันจะกลายมาเป็นส่วนหนึ่งของคลาสลูก ถ้าเป็น PP เราจะต้องก๊อบปี้ทั้งข้อมูล และฟังก์ชันมาไว้ในที่ใหม่เพื่อที่จะได้ผลแบบเดียวกัน

3. สำหรับ OOP เมื่อมีบางสิ่งที่ต้องการเพิ่มเติมหรือแก้ไขเราก็ใช้คุณสมบัติข้อ ค. (คือ Polymorphism) ทำการเพิ่มเติมหรือแก้ไขบางฟังก์ชันเพื่อให้ทำงานตามสภาพที่ต้องการในคลาสนั้นๆ ได้ ถ้าแบบ PP เราต้องเขียนฟังก์ชันขึ้นมาใหม่ ใช้ชื่อใหม่ ชื่อจะซ้ำกับชื่อเดิมไม่ได้

4. รูปแบบของ OOP เหมาะที่จะทำให้บริษัทซอฟท์แวร์ ผลิตซอฟท์แวร์ออกมาขายในรูปของซอฟท์แวร์ ไอซี (software - IC) ซึ่งเมื่อเป็นดังนี้วงการ

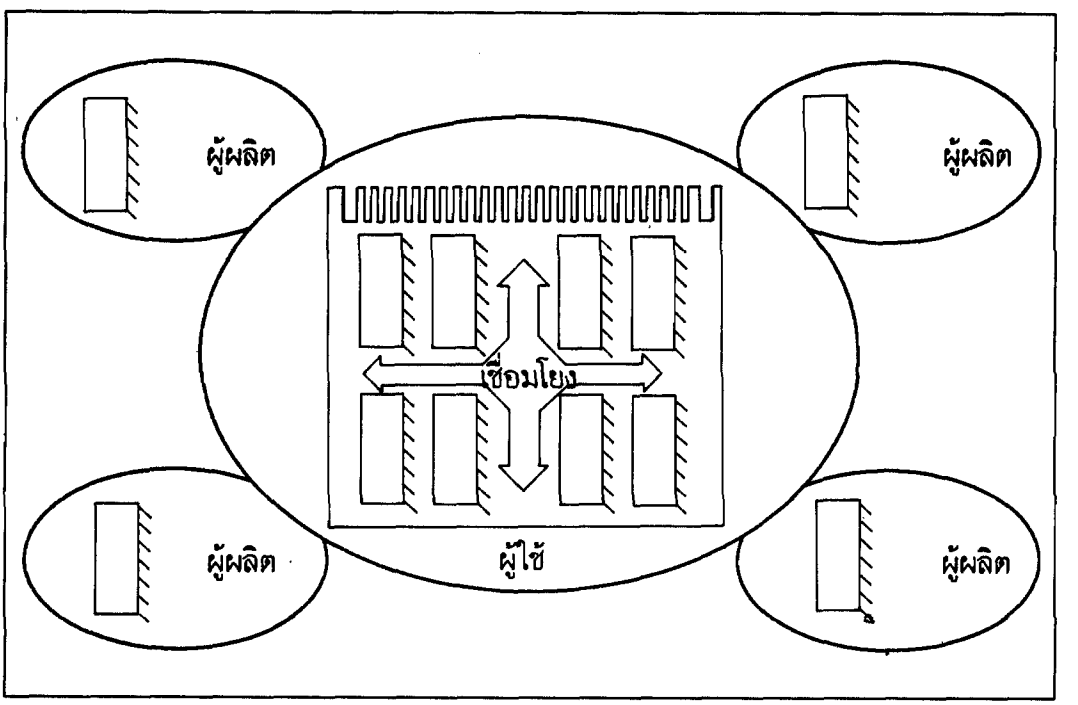
ซอฟท์แวร์ ก็จะมีการขยายตัวไปได้อย่างรวดเร็ว ดูตามรูปที่ 5 จะเห็นว่า ผู้ใช้เพียงแต่เชื่อมซอฟท์แวร์ส่วนที่เหมาะสมสำหรับงานของตนเอง และทำการรวบรวมคุณสมบัติต่างๆ เข้าด้วยกัน ปรับแต่งเล็กน้อยก็จะได้โปรแกรมสำเร็จที่บรรลุจุดประสงค์ได้ในเวลาอันรวดเร็วกว่าแบบ PP ที่ต้องมารวบรวมฟังก์ชันแล้วมาทำการปรับแต่ง ทั้งนี้เพราะการปรับแต่งในรูปของ OOP นั้นจะทำในระดับสูง ส่วนของ PP จะต้องทำในระดับคำสั่งภายในฟังก์ชันที่นำมาใช้

5. ในระยะแรกๆ ที่โปรแกรมไม่โตนัก ซอฟท์แวร์ ในรูปแบบ PP ยังไม่พบปัญหาจนปัจจุบันเมื่อโปรแกรมโตขึ้นมาก เราจะพบปัญหาในเรื่องการแก้ไขข้อผิดพลาด (bugs) ที่มีอยู่ในโปรแกรม ถึงแม้ว่าแต่ละส่วนของโปรแกรมจะถูกแบ่งเป็นฟังก์ชันแล้วก็ตาม แต่การเรียกใช้ฟังก์ชันเพื่อจัดการกับข้อมูลก็สืบโยงกันไปอาจจะหลายทอด ดังนั้นการแก้ไขที่ระดับใดระดับหนึ่งก็สามารถยังผลให้เกิดข้อผิดพลาดสำหรับระดับอื่นต่อไปอีก สำหรับ OOP ข้อผิดพลาดแต่ละตัวสามารถแก้ไขได้ที่ออบเจกต์นั้นๆ โดยตรงและจะไม่ส่งผลไปยังออบเจกต์อื่นๆ ที่ไม่มีการสืบทอดต่อไปอีก หรือถ้ามีการสืบทอดต่อไป ก็สามารถแก้ไขเพิ่มเติมเฉพาะส่วนที่จำเป็นในคลาสนั้นๆ

6. การตั้งชื่อฟังก์ชันสำหรับ OOP สามารถให้เหมือนกันได้ โดยตัวโปรแกรมจะทำการแยกแยะเองว่าควรจะเรียกฟังก์ชันไหนมาใช้งานให้เหมาะสมตามที่กำหนดไว้ โดยลักษณะของโปรแกรมจะเป็นชนิดเชื่อมโยงกันก่อน (early binding) หรือเชื่อมโยงกันภายหลัง (late binding) ก็ได้ ส่วนของ PP นั้นจะต้องเชื่อมโยงกันก่อนเสมอ

7. OOP ใช้วิธี Encapsulation ดังนั้นแต่ละออบเจกต์จะมีข้อมูลของมันอยู่แล้ว การติดต่อระหว่างออบเจกต์ จึงมีการส่งข้อมูลไปมาน้อยมากเมื่อเปรียบเทียบกับ PP ซึ่งจะมีทั้งข้อมูลที่ส่งผ่านไปหากัน และข้อมูลชนิดที่เป็น Global อยู่มาก

8. ในปัจจุบันนี้พบว่าผู้ที่นิยมใช้ PC ส่วนมากจะรัน WINDOWS และตามที่ได้ทราบแล้วว่า WINDOWS ก็เป็นโปรแกรมเชิงวัตถุอยู่แล้ว ดังนั้นการที่จะเขียนโปรแกรมเพื่อรันภายใต้ WINDOWS น่าจะเป็นโปรแกรมประเภท OOP ด้วยเช่นกัน เมื่อเป็นเช่นนี้ผู้เขียนโปรแกรมสำหรับ WINDOWS ทั้งหลายควรต้องเรียนรู้ OOP และคิดว่าในอีกไม่กี่ปีข้างหน้า OOP จะเข้ามาแทนที่ PP อย่างแน่นอน



รูปที่ 5. Software - IC (จาก Cox. (1986))

ตัวอย่างโปรแกรม

ต่อไปนี้จะเป็นตัวอย่าง OOP ของภาษา C++ เป็นโปรแกรมที่แสดงให้เห็นชัดถึงคุณสมบัติทั้ง 3 ประการของ OOP อันมี Encapsulation, Inheritance และ Polymorphism จากตัวอย่างนี้จะมีคลาสทั้งหมด 4 คลาส คือ คลาส person, student, worker และ student_work โดยมีรายละเอียดดังนี้

- คลาส person เป็นคลาสหลัก หรือคลาสแม่คลาส นี้เก็บข้อมูลอยู่ 4 ตัว คือ
  - name ใช้เก็บชื่อของบุคคล
  - lastname ใช้เก็บนามสกุล
  - age ใช้เก็บอายุ
  - address ใช้เก็บที่อยู่

มีอยู่ 2 ฟังก์ชัน คือ
  - ask ( ) ใช้ในการป้อนข้อมูลทั้งหมดของคลาส person
  - show ( ) ใช้ในการแสดงข้อมูลทั้งหมดของคลาส person

- คลาส student เป็นคลาสสำหรับนักศึกษา คลาสนี้ จะดึงคุณสมบัติของคลาสแม่ คือ person มาใช้ด้วย และต้องเพิ่มเติมทั้งข้อมูลและฟังก์ชันขึ้นอีกเพื่อความเหมาะสม ข้อมูลที่เพิ่มเติมมี 3 ตัว คือ
  - level ใช้เก็บระดับชั้นนักศึกษากำลังเรียนอยู่
  - grade ใช้เก็บค่าคะแนนเฉลี่ยของการเรียนล่าสุด
  - schoolname ใช้เก็บชื่อของสถานศึกษา

มี 2 ฟังก์ชันคือ
  - ask ( ) ใช้ในการป้อนข้อมูลทั้งหมดของคลาส student
  - show ( ) ใช้ในการแสดงข้อมูลทั้งหมดของคลาส student

- คลาส worker เป็นคลาสสำหรับคนทำงาน คลาสนี้จะดึงคุณสมบัติของคลาสแม่ คือ person มาใช้ด้วย และต้องเพิ่มเติมทั้งข้อมูลและฟังก์ชันขึ้นอีกเพื่อความเหมาะสม ข้อมูลที่เพิ่มเติมมี 3 ตัว คือ
  - position ใช้เก็บตำแหน่งหน้าที่ที่ทำอยู่
  - workplace ใช้เก็บชื่อสถานที่ทำงานเป็นเงินเดือน

มี 2 ฟังก์ชัน คือ
  - ask ( ) ใช้ในการป้อนข้อมูลทั้งหมดของคลาส worker
  - show ( ) ใช้ในการแสดงข้อมูลทั้งหมดของคลาส worker

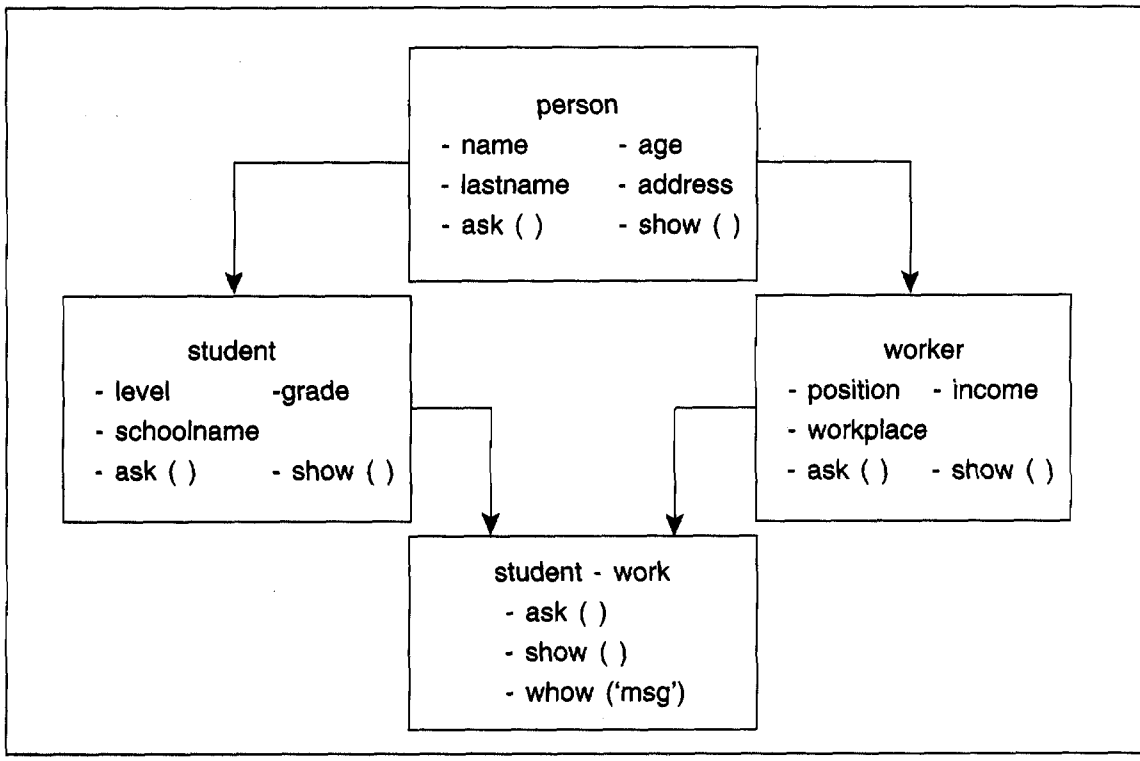
- คลาส student_work เป็นคลาสสำหรับนักศึกษาที่ทำงานด้วย คลาสนี้จะดึงคุณสมบัติของคลาสแม่ คือ student และ worker มาใช้พร้อมกัน (Multiple - inheritance) คลาสนี้ไม่มีข้อมูลเพิ่มเติม ส่วนฟังก์ชันที่ต้องกำหนดขึ้นใหม่มี 3 ฟังก์ชัน คือ
  - ask ( ) ใช้ในการป้อนข้อมูลทั้งหมดของคลาส student_work
  - show ( ) ใช้ในการแสดงข้อมูลทั้งหมดของลาส student_work
  - show (msg) ใช้ในการแสดงข้อความคลาส student_work

ทั้งหมดสามารถแสดงเป็นโฟลว์ชาร์ตได้ดังรูปที่ 6

ตัวโปรแกรมจะประกอบด้วย 3 ไฟล์ คือ ไฟล์ examhead.h, examdef.cpp และ exammain.cpp ซึ่งมีรายละเอียดดังนี้

ไฟล์ examhead.h

```
1     class person        {
2         private:
3             char name [50];
4             char lastname [50];
5             int age;
6             char address [100];
```



รูปที่ 6. โฟลว์ชาร์ตส่วนประกอบของคลาสต่างๆ ในโปรแกรมตัวอย่าง

```
7               public:
8                   person(void);
9                   void ask(void);
10                  void show(void);
11              };
12              class student : public person       {
13                  private:
14                  char level[50];
15                  float grade;
16                  char schoolname[50];
17              public:
18                  student (void);
19                  void ask (void);
20                  void show (void);
21          };
22          class worker : public person {
23              public:
24                  char position [50];
25                  char workplace [50];
```

```
26                  int income;
27              public:
28                  worker (void);
29                  void ask (void);
30                  void show (void);
31          };
32          class student_work : public student,worker  {
33              public:
34                  student_work (void);
35                  void ask (void);
36                  void show (void);
37                  void show (char *msg);
38          ;
```

ไฟล์ examdef. cpp

```
39      #include "examhead.h"
40      #include <iostream.h>
41      #include <iomanip.h>
42      #include <string.h>
43      // ส่วนของการกำหนดฟังก์ชันต่างๆ สำหรับคลาส person
44          person::person (void) { // กำหนด constructor ของคลาส person
45              strcpy (name,"\0");
46              strcpy (lastname,"\0");
47              age = 0;
48              strcpy (address,"\0");
49          };
50          void person::ask(void) {//กำหนด ฟังก์ชัน ask() ของคลาส person
51              char ch;
52              cout << "ป้อนชื่อ: "; cin.get (name,sizeof(name),'\n'); cin.get(ch);
53              cout << "ป้อนนามสกุล: "; cin.get (lastname,sizeof (lastname),'\n');cin. get(ch);
54              cout << "ป้อนอายุ: "; cin >> age; cin.get(ch);
55              cout << "ป้อนที่อยู่: "; cin.get(address,sizeof(address),'\n') ;cin.get(ch);
56          };
57          void person::show (void) { //กำหนดฟังก์ชัน show () ของคลาส person
58              cout << "ชื่อ:" << name;
59              cout << "นามสกุล:" << lastname << "\n";
60              cout << "อายุ:" << age << "\n";
61              cout << "ที่อยู่:" << address << "\n";
62          };
```

```
63    // ส่วนของการกำหนดฟังก์ชันต่างๆ สำหรับคลาส student
64        student::student (void):person () { // กำหนด constructor ของคลาส student
65            strcpy(level,"\0");
66            grade = 0;
67            strcpy (schoolname,"\0");
68        };
69        void student::ask(void) //กำหนดฟังก์ชัน ask() ของคลาส student
70            char ch;
71            person::ask(); // เรียกใช้ฟังก์ชัน ask() ของคลาส person
72            cout << "กำลังเรียนอยู่ชั้น: "; cin.get(level,sizeof (level),'\n'); cin.get (ch);
73            cout << "ได้เกรดเฉลี่ย: "; cin >> grade; cin.get (ch);
74            cout << "ป้อนชื่อสถานที่ศึกษา: "; cin.get(schoolname,sizeof (schoolname),'\n');
                  cin.get(ch);
75        };
76        void student::show(void) { //กำหนดฟังก์ชัน show() ของคลาส student
77            person::show(); // เรียกใช้ฟังก์ชัน show() ของคลาส person
78            cout << "กำลังเรียนอยู่ชั้น: " << level << "\n";
79            cout << "ได้เกรดเฉลี่ย: " << grade << "\n";
80            cout << "กำลังศึกษาอยู่ที่: " << schoolname << "\n";
81        };
82    // ส่วนของการกำหนดฟังก์ชันต่างๆ สำหรับคลาส worker
83        worker::worker(void):person() { // กำหนด constructor ของคลาส worker
84            strcpy(position,"\0");
85            strcpy(workplace,"\0");
86            income=0;
87        };
88        void worker::ask(void) { //กำหนดฟังก์ชัน ask() ของคลาส worker
89            char ch;
90            person::ask(); // เรียกใช้ฟังก์ชัน ask() ของคลาส person
91            cout << "กำลังทำงานอยู่ในตำแหน่ง: ";cin.get (position,sizeof (position),'\n');
                  cin.get(ch);
92            cout << "ป้อนชื่อสถานที่ทำงาน: "; cin.get(workplace,sizeof (workplace),'\n');
                  cin.get(ch);
93            cout << "ป้อนเงินเดือน: ";cin >> income; cin.get(ch);
94        };
95        void worker::show (void) { //กำหนดฟังก์ชัน show () ของคลาส worker
96            person::show(); // เรียกใช้ฟังก์ชัน show() ของคลาส person
97            cout << "กำลังทำงานอยู่ในตำแหน่ง: " << position << "\n";
98            cout << "กำลังทำงานอยู่ที่: " << workplace << "\n";
```

```
99              cout << "มีรายได้เดือนละ: " << income << " บาท\n";
100         };
101     // ส่วนของการกำหนดฟังก์ชันต่างๆ สำหรับคลาส student_work
102         student_work::student_work(void):student(),worker() {
103         };   // กำหนด constructor ของคลาส student_work
104         void student_work::ask(void) { //กำหนดฟังก์ชัน ask() ของคลาส student_work
105             char ch;
106             student::ask(); // เรียกใช้ฟังก์ชัน ask() ของคลาส student
107             cout << "กำลังทำงานอยู่ในตำแหน่ง: ";cin.get (position,sizeof (position),'\n');
                    cin.get (ch);
108             cout << "ป้อนชื่อสถานที่ทำงาน: "; cin.get (workplace,sizeof (workplace),'\n');
                    cin.get(ch);
109         cout << "ป้อนเงินเดือน: ";cin >> income; cin.get(ch);
110         };
111     void student_work::show(void) { //กำหนดฟังก์ชัน show() ของคลาส student_work
112         student::show(); // เรียกใช้ฟังก์ชัน show() ของคลาส student
113             cout << "กำลังทำงานอยู่ในตำแหน่ง: " << position << "\n";
114             cout << "กำลังทำงานอยู่ที่: " << workplace << "\n";
115             cout << "มีรายได้เดือนละ: " << income << " บาท\n";
116         };
117     void student_work::show(char *msg) { //กำหนดฟังก์ชัน show ("ข้อความ") ของคลาส student_work
118         cout << msg;
119         };
```

ไฟล์ exammain. cpp

```
120     #include "examhead.h"
121     #include <iostream.h>
122     main ()    {
123     person s1;
124     student s2;
125     worker s3;
126     student_work s4;
127     cout << "ป้อนข้อมูลบุคคลทั่วไป.....\n";
128     s1.ask();
129     cout << "แสดงข้อมูลของบุคคลทั่วไป.....\n";
130     s1.show();
131     cout << "ป้อนข้อมูลสำหรับนักศึกษา.....\n";
132     s2.ask();
```

```
133     cout << "แสดงข้อมูลของนักศึกษา.....\n";
134     s2.show();
135     cout << "ป้อนข้อมูลสำหรับคนทำงาน.....\n";
136     s3.ask();
137     cout << "แสดงข้อมูลของคนทำงาน.....\n";
138     s3.show();
139     cout << "ป้อนข้อมูลสำหรับนักศึกษาที่ทำงาน.....\n";
140     s4.ask();
141     cout << "แสดงข้อมูลของนักศึกษาที่ทำงาน.....\n";
142     s4.show();
143     s4.show("<<<<<<<<<< จบโปรแกรมตัวอย่าง >>>>>>>>>>");
144     return 0;
145     };
```

ตัวอย่างเมื่อรันโปรแกรม

```
146     ป้อนข้อมูลบุคคลทั่วไป.....
147     ป้อนชื่อ : สมพันธุ์
148     ป้อนนามสกุล : ชาญศิลป์
149     ป้อนอายุ: 35
150     ป้อนที่อยู่ : 77 ถ.สุรนารี อ. เมือง จ. นครราชสีมา 30000
151     แสดงข้อมูลของบุคคลทั่วไป.....
152     ชื่อ : สมพันธุ์ นามสกุล: ชาญศิลป์
153     อายุ : 35
154     ที่อยู่ : 77 ถ.สุรนารี อ.เมือง จ.นครราชสีมา 30000
155     ป้อนข้อมูลสำหรับนักศึกษา.....
156     ป้อนชื่อ : บุญมี
157     ป้อนนามสกุล : จันดา
158     ป้อนอายุ : 18
159     ป้อนที่อยู่ : 137 ถ.โพธิ์กลาง อ. เมือง จ. นครราชสีมา 30000
160     กำลังเรียนอยู่ชั้น : ปีที่ 2
161     ได้เกรดเฉลี่ย : 3.48
162     ป้อนชื่อสถานที่ศึกษา : มหาวิทยาลัยเทคโนโลยีสุรานารี อ. เมือง จ. นครราชสีมา
163     แสดงข้อมูลของนักศึกษา.....
164     ชื่อ : บุญมี นามสกุล : จันดา
165     อายุ : 18
166     ที่อยู่ : 137 ถ.โพธิ์กลาง อ. เมือง จ. นครราชสีมา 30000
167     กำลังเรียนอยู่ชั้น : ปีที่ 2
168     ได้เกรดเฉลี่ย : 3.48
```

```
169 กำลังศึกษาอยู่ที่ : มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง จ. นครราชสีมา
170 ป้อนข้อมูลสำหรับคนทำงาน.....
171 ป้อนชื่อ : บดินทร์
172 ป้อนนามสกุล : สินพัฒน์
173 ป้อนอายุ : 45
174 ป้อนที่อยู่ : 12/8 ถ. อุดม อ. เมือง จ. บุรีรัมย์
175 กำลังทำงานอยู่ในตำแหน่ง : ผู้ควบคุมการก่อสร้าง
176 ป้อนชื่อสถานที่ทำงาน : สมานก่อสร้างจำกัด บุรีรัมย์
177 ป้อนเงินเดือน : 30000
178 แสดงข้อมูลของคนทำงาน.....
179 ชื่อ : บดินทร์ นามสกุล : สินพัฒน์
180 อายุ : 45
181 ที่อยู่ : 12/8 ถ.อุดม อ. เมือง จ. บุรีรัมย์
182 กำลังทำงานอยู่ในตำแหน่ง : ผู้ควบคุมการก่อสร้าง
183 กำลังทำงานอยู่ที่ : สมานก่อสร้างจำกัด บุรีรัมย์
184 มีรายได้เดือนละ : 30000 บาท
185 ป้อนข้อมูลสำหรับนักศึกษาที่ทำงาน.....
186 ป้อนชื่อ : เอกภาพ
187 ป้อนนามสกุล : ปานมี
188 ป้อนอายุ : 19
189 ป้อนที่อยู่ : 111 ถ. สุรนารี อ. เมือง จ. นครราชสีมา 30000
190 กำลังเรียนอยู่ชั้น : ปีที่ 3
191 ได้เกรดเฉลี่ย: 3.25
192 ป้อนชื่อสถานที่ศึกษา : มหาวิทยาลัยเทคโนโลยีสุรนารี
193 กำลังทำงานอยู่ในตำแหน่ง : ส่งหนังสือพิมพ์รายวัน
194 ป้อนชื่อสถานที่ทำงาน : งานส่วนตัว
195 เงินเดือน : 1500
196 แสดงข้อมูลของนักศึกษาที่ทำงาน.....
197 ชื่อ : เอกภาพ นามสกุล: ปานมี
198 อายุ : 19
199 ที่อยู่ : 111 ถ. สุรนารี อ. เมือง จ. นครราชสีมา 30000
200 กำลังเรียนอยู่ชั้น : ปีที่ 3
201 ได้เกรดเฉลี่ย : 3.25
202 กำลังศึกษาอยู่ที่: มหาวิทยาลัยเทคโนโลยีสุรนารี
203 กำลังทำงานอยู่ในตำแหน่ง: ส่งหนังสือพิมพ์รายวัน
204 กำลังทำงานอยู่ที่: งานส่วนตัว
205 มีรายได้เดือนละ: 1500 บาท
206 <<<<<<<<<< จบโปรแกรมตัวอย่าง >>>>>>>>>>
```

อธิบายโปรแกรมตัวอย่าง

จุดประสงค์ของโปรแกรมนี้ เพื่อใช้ประกอบการอธิบายเรื่องของ OOP ดังนั้นบางคำสั่งอาจจะไม่จำเป็น แต่ก็มีไว้เพื่อแสดงให้เห็นเด่นชัด บางคำสั่งอาจสามารถเพิ่มเติมปรับแต่งให้ดีขึ้นอีกได้ รวมทั้งรูปแบบก็กำหนดเพื่อให้เข้าใจได้ง่าย

ตั้งแต่ บ. 1 (บรรทัดที่ 1) ถึง บ. 38 (บรรทัดที่ 38) เป็นส่วนของการกำหนดคลาสต่างๆ ซึ่งมีความสัมพันธ์กันตามรูปที่ 6. จากส่วนแรกนี้ สามารถใช้อธิบายการทำงานของ OOP ดังนี้

- Encapsulation คือ คุณสมบัติที่รวมทั้งข้อมูล (data) และฟังก์ชัน (functions) เข้าไว้ด้วยกัน จากโปรแกรมจะเห็นว่าทุกคลาสต่างก็มีทั้งข้อมูลและฟังก์ชันอยู่ด้วยกัน ตัวอย่างเช่น คลาส person ได้มีทั้งข้อมูล (บ. 3 ถึง บ. 6) และฟังก์ชัน (บ. 8 ถึง บ. 10) อยู่ด้วยกัน ส่วนที่อยู่ภายใต้คำว่า private นั้นคือส่วนที่คลาสอื่น ถึงแม้ว่าจะเป็นคลาสลูกก็ตาม ไม่สามารถเรียกใช้ได้ เป็นหัวข้อเฉพาะของคลาสนี้เท่านั้น ส่วนที่อยู่ใต้คำว่า public นั้น คือ ส่วนที่คลาสลูกสามารถเรียกใช้ร่วมกันได้การกำหนดการทำงานของแต่ละฟังก์ชันในแต่ละคลาสมีรายละเอียดดังปรากฏอยู่ในไฟล์ examdef. cpp

- Inheritance คือ คุณสมบัติในการสืบทอดลักษณะของคลาสแม่ รวมทั้งความสามารถในการเรียกใช้ทั้งข้อมูล และฟังก์ชันที่คลาสแม่กำหนดอนุญาตไว้ได้ด้วย เช่น บ. 12 ถึง บ. 21 คือ คลาส student จะมีข้อมูลทั้งหมด 7 ตัว มี 4 ตัว สืบทอดคุณสมบัติมาจากคลาสแม่ แต่คลาส student เรียกใช้ไม่ได้เพราะเป็น private สำหรับคลาสแม่ เท่านั้น อีก 3 ตัวที่กำหนดเพิ่ม คือ บ. 14 ถึง บ. 16 ส่วน ฟังก์ชันจะมีทั้งหมด 6 ฟังก์ชัน มี 3 ฟังก์ชันที่เรียกใช้ได้ กับอีก 3 ฟังก์ชันที่กำหนดเพิ่มขึ้นอีก คือ บ. 18 ถึง บ. 20

จากโปรแกรม บ. 32 ถึง บ. 38 จะเห็นว่าคลาส student_work เป็นคลาสลูกที่มีคลาสแม่สองคลาส คือ คลาส student และคลาส worker ในกรณีนี้เราเรียก multiple inheritance และที่น่าสังเกต คือ ข้อมูลของคลาส worker ถูกประกาศเป็น public ดังนั้นฟังก์ชันของคลาสลูก สามารถเรียกใช้ข้อมูลเหล่านี้ได้ ซึ่งคลาส student _work ก็ได้ใช้คุณสมบัติข้อนี้ โดยมีฟังก์ชัน ask ( ) และ show ( ) ดู บ. 111 ถึง บ. 116 ที่สามารถเรียกใช้ข้อมูล position, workplace และ income สืบทอดมาจากคลาส worker ได้

- Polymorphism คือ คุณสมบัติที่ยอมให้หลายฟังก์ชัน ที่ทำงานต่างกัน มีชื่อเดียวกันได้ จากตัวอย่างโปรแกรม บ. 9, บ. 10, บ. 19, บ. 20, บ. 29, บ. 30, บ. 35, บ. 36, และ บ. 37 จะเห็นว่าแต่ละคลาสต่างก็มีฟังก์ชัน ask ( ) และ show ( ) ที่มีชื่อเดียวกันแต่มีการทำงานต่างกันเป็นของตนเอง ลักษณะเช่นนี้เป็น Polymorphism ที่เรียกว่า early binding แต่จะมีอีกกรณีหนึ่ง คือ late binding ที่ทำได้โดยกำหนดฟังก์ชันให้เป็นแบบ virtual ซึ่งจะไม่ขอกล่าวรายละเอียดในที่นี้

ผลของการรันโปรแกรม เกิดจากคำสั่งในส่วนของฟังก์ชัน main ( ) ตั้งแต่ บ. 127 ถึง บ. 143 โดยมี

- คำสั่ง บ.123 ถึง บ.126 เป็นการกำหนดออบเจกต์ s1, s2, s3, และ s4 ตามลำดับ
- คำสั่ง บ. 127 ให้ผลเป็นข้อความ บ. 146
- คำสั่ง บ. 128 เป็นการขอให้ออบเจกต์ s1 ทำการถาม ผลที่ได้คือ บ.147 ถึง 150
- คำสั่ง บ. 129 ให้ผลเป็นข้อความ บ. 151
- คำสั่ง บ. 130 ให้ผลเป็นการแสดงข้อมูล บ. 152 ถึง บ. 154 เป็นต้น

จากฟังก์ชัน main ( ) นี้ person, student, worker และ student_work เปรียบเสมือน IC ตัวหนึ่งๆ ผู้ใช้เพียงทำหน้าที่รวบรวม IC เหล่านี้เข้าด้วยกันเพื่อให้การสั่งงานเป็นไปตามความต้องการ จะดูได้จากลักษณะคำสั่งตั้งแต่ บ. 127 ถึง บ. 143 จะเห็นว่าเป็นคำสั่งสั้นๆ ง่ายๆ ทั้งสิ้นก็หวังว่าเมื่อ OOP แพร่หลายเป็นที่รู้จักโดยทั่วไปแล้ว ลักษณะของ Software - IC จะเกิดขึ้นอย่างหลากหลายเพื่อผู้ใช้จะได้ซื้อหาและเรียกใช้ เมื่อนั้นการพัฒนาโปรแกรมจะเป็นไปด้วยความรวดเร็ว และมีประสิทธิภาพยิ่งกว่าปัจจุบันนี้มาก

สำหรับผู้ที่สนใจจะศึกษา OOP

เราจะเห็นว่า ในปัจจุบันภาษาสั่งงานคอมพิวเตอร์เกือบทุกภาษาได้ผนวกเอาคุณสมบัติของ OOP เข้าไว้เรียบร้อยแล้วไม่ว่าจะเป็นภาษา BASIC, PASCAL, C หรือ COBOL (อาจจะเกิดมี Object - FORTRAN ขึ้น

ในเร็ววันนี้ก็ได้) ดังนั้นถ้าท่านเป็นคนหนึ่งที่สนใจจะศึกษาเรื่องของ OOP อย่างจริงจัง ถ้าถามว่าจะควรเริ่มจากไหน ? คำตอบก็คือ ถ้าท่านถนัดภาษา PASCAL ก็ขอให้เริ่มที่ PASCAL อาจจะใช้คอมไพเลอร์ เป็น Borland PASCAL V7.0 ก็ได้ แต่สำหรับท่านถนัดภาษา C ก็เริ่มที่ภาษา C อาจใช้ Borland C++ V3.1 ก็ได้ แต่สำหรับผู้ที่ยังไม่คุ้นเคยกับภาษา PASCAL หรือ C เลย ก็ขอให้เรียนรู้ภาษา C แล้วก็ศึกษา OOP โดยใช้คอมไพเลอร์ตัวเดียวกันคือ C++ ที่แนะนำภาษา C แทนที่จะเป็นภาษา PASCAL ก็เพราะ OOP ของภาษา PASCAL มีข้อจำกัดบางประการ OOP ของภาษา C ทำงานได้เร็วและกว้างขวางกว่า

สรุป

จากคุณสมบัติทั้ง 3 ประการของ OOP ดังกล่าวมาแล้วจะเห็นได้ชัดเจนว่าการนำมาใช้ได้ใหม่ (Reuasbility) ของโปรแกรมเป็นไปได้สูง และเป็นไปด้วยความสะดวกยิ่งกว่าเดิม และมีความหวังเป็นอย่างยิ่งในวงการซอฟท์แวร์ว่าในเร็ววันนี้จะมีผู้สนใจเรียนรู้เรื่องของ OOP มากขึ้นและบริษัทซอฟท์แวร์จะเริ่มผลิต Software - IC ออกมาจำหน่าย

ถ้าหันมาดูวิวัฒนาการของการเขียนโปรแกรม ดังได้กล่าวมาแล้ว จะเห็นว่าในเวลาอันไม่นานข้างหน้านี้ การเขียนโปรแกรมจะเป็นลักษณะของ OOP เกือบทั้งสิ้น ปัจจุบันเราอยู่ในขั้นตอนของการเคลื่อนย้าย สำหรับท่านที่กำลังลังเล ก็ขอให้ท่านมั่นใจได้เลยว่าท่านหลีกเลี่ยงไม่ได้ เหมือนสมัยก่อนเราหลีกเลี่ยงไม่ได้ที่จะผละจากการเขียนโปรแกรมอย่างธรรมดา ที่ไม่มีกฎเกณฑ์มากนักมาเป็นแบบโพรซีเจอร์ที่เต็มไปด้วยกฎเกณฑ์และในปัจจุบัน จากแบบโพรซีเจอร์เรากำลังผละเข้าไปสู่ OOP หรือการโปรแกรมเชิงวัตถุ ที่เต็มไปด้วยการวางแผนและการจัดการอย่างแน่นอน

คำขอบคุณ

บทความนี้ สำเร็จได้ด้วยความช่วยเหลือของบุคคลหลายฝ่าย อาทิเช่น ผศ.ดร. สราวุฒิ สุจิตจร อาจารย์พิชโยทัย มหัทธนาภิวัฒน์ ได้ปริทรรศน์บทความ พร้อมกับคำแนะนำอันเป็นประโยชน์ คุณมณฑาทิพย์ พันธุ์ภูวงศ์ และ คุณภัทรวรรณ แซ่ไต้ ช่วยในด้านการพิมพ์ทางศูนย์คอมพิวเตอร์รวมทั้งเจ้าหน้าที่ต่างก็อำนวยความสะดวกให้เต็มความสามารถรวม ทั้งกองบรรณาธิการของวารสาร อันประกอบด้วยบุคคลหลายฝ่ายต่างก็ช่วยเหลือเป็นอันดี ผมขอแสดงความขอบคุณไว้ ณ ที่นี้ด้วย

เอกสารอ้างอิง

Cox B.J. (1986) Object-oriented Programming. Addison Wesley Publishing Company.

Dillon T.S. (1993) Object-oriented Conceptual Modelling. Prentice Hall, Inc., New Jersey.

Object-oriented เทคโนโลยีปฏิวัติวงการ (2535) วารสารคอมพิวเตอร์รีวิว 96

บรรณานุกรม

Anon. (1990) Turbo C++ Getting Start. Borland International, Inc., California.

Bar-David T. (1993) Object-oriented Design for C++. Prentice Hall, Inc., New Jersey.