SE-ED COMPUTER BOOK developer

การเขียนโปรแกรมเชิงวัตถุ ด้วย Visual Basic .NET

- แนวคิดเชิงวัตถุ
- การวิเคราะห์และออกแบบโปรแกรมเชิงวัตถุเบื้องต้น
- ตีความเชิงวัตถุ
- การกำหนดความรับผิดชอบของคลาสและพฤติกรรมของคลาส
- อีเวนต์และดีลีเกต
- การสืบทอดคุณสมบัติของคลาส
- การใช้งานอินเตอร์เฟซ

ธีระพล ลิ้มศรัทธา

ซีเอ็ด

คำนำ

ความตั้งใจที่ได้เรียบเรียงหนังสือการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming : OOP) ด้วย VB.NET เนื่องจากผมพบว่ามีการใช้โปรแกรม VB.NET จำนวนมาก และใช้กันแพร่หลาย ซึ่งในระยะเวลาที่ผมเรียนมา กลับได้เรียนการเขียนโปรแกรมเชิงวัตถุด้วยภาษาอื่นๆ เท่านั้น เช่น ภาษา Java, ภาษา C หรือภาษาอื่นๆ ที่ไม่นิยมใช้งานได้จริงเป็นจำนวนมาก เนื่องด้วยเหตุผลอะไรก็ตามแต่ที่เป็นไปเช่นนั้น เมื่อถึงคราวที่ต้องเรียนสิ่งที่ใช้งานจริง เพื่อนำไปใช้ได้รวดเร็ว ภาษาของ Visual Basic ตระกูล .NET ก็มีความสมบูรณ์ในเชิงวัตถุแล้วจึงต้องเลือกเรียน VB.NET เพื่อนำไปเขียนโปรแกรมเชิงวัตถุ ซึ่งการเรียนภาษานี้ก็ใช่ว่าจะไม่นำไปประยุกต์ใช้ไม่ได้กับภาษาอื่นๆ เหมือนเหตุผลที่เลือกเรียนภาษาอื่นๆ ก็ใช่ว่าจะใช้ไม่ได้กับภาษา VB.NET เพราะทุกภาษาก็ล้วนใช้แนวคิดเชิงวัตถุนำไปประยุกต์ใช้งานได้ใกล้เคียงกัน แต่เพราะเหตุผลอะไรนั้น ผมขอตอบเป็นเพียงทางเลือกว่า ควรเลือกภาษา VB.NET เพราะการเรียนรู้ภาษาตระกูล .NET รู้เพียงเฟรมเวิร์กเดียว ก็สามารถเขียนได้หลายภาษา บนภาษาที่สนับสนุน .NET ครับ

ความรู้ในเชิงวัตถุเรื่องใด ประเด็นใด ที่คิดว่ายังขาดไป หรือไม่ชัดเจนนั้น ต้องขออภัยมา ณ ที่นี้ด้วย และยินดีรับคำติชมผ่านทางอีเมล ซึ่งจะเป็นพระคุณอย่างสูง และผมขอขอบคุณบิดา มารดา ที่ได้ส่งเสริมด้านการศึกษาตลอดมา กำลังใจจาก อ้อย และอ๋อง และขอขอบคุณบริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) สำหรับโอกาสงานเขียนครั้งนี้ครับ

ธีระพล ลิ้มศรัทธา
theerapone_x@yahoo.com

สารบัญ

[บทที่]

1 แนวคิดเชิงวัตถุ

วัตถุประสงค์

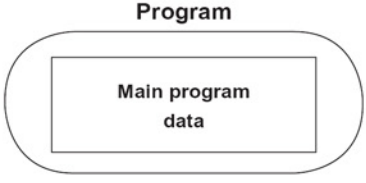
- อธิบายความเป็นมา และแนวคิดเชิงวัตถุได้
- อธิบายคำศัพท์พื้นฐานที่ใช้ในการเขียนโปรแกรมเชิงวัตถุได้
- อธิบายแนวคิดสำคัญในการเขียนโปรแกรมเชิงวัตถุได้
- อธิบายพื้นฐานสถาปัตยกรรม .NET Framework ได้

ในบทนี้อธิบายความเป็นมาของการเขียนโปรแกรมเพื่อนำไปพัฒนาโปรแกรม และแนวคิดเชิงวัตถุเพื่อนำไปสู่การพัฒนาโปรแกรมเชิงวัตถุ ซึ่งมีภาษาต่างๆ ที่สามารถพัฒนาเพื่อส่งเสริมแนวคิดเชิงวัตถุได้ เช่น ภาษา C++, Java แต่สำหรับการศึกษาในครั้งนี้จะใช้ภาษาเบสิก ที่มีการจัดสภาพแวดล้อมให้สามารถทำงานได้ง่ายขึ้น แล้วเรียกใหม่ว่า Visual Basic และการพัฒนาเพิ่มภายใต้กรอบเชิงวัตถุที่ทำงานได้ดีขึ้น ชื่อว่า Visual Basic Version .NET (วิชวลเบสิก เวอร์ชั่น .NET) เนื้อหาในบทนี้ประกอบด้วยหัวข้อที่ควรเข้าใจได้คือ

- ความเป็นมา และแนวคิดเชิงวัตถุ
- การพัฒนาโปรแกรมเชิงวัตถุ
- เครื่องมือที่ใช้พัฒนาเชิงวัตถุ
- Microsoft.NET Framework, Visual Basic.NET และ Visual Studio.NET

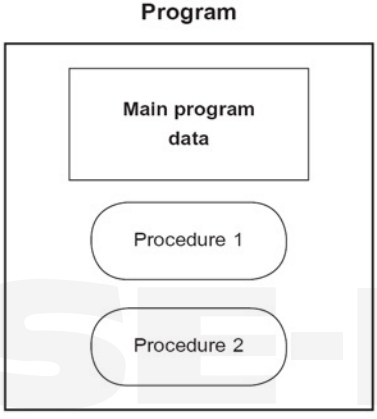
วิวัฒนาการของการเขียนโปรแกรม

ในปี 1960 มีการพัฒนาโปรแกรมด้วยภาษา Assembly, FORTRAN, COBOL มักเป็นโปรแกรมที่มีขนาดเล็ก ซึ่งคำสั่งหลายๆ คำสั่งที่โปรแกรมเมอร์เขียน จะเข้าใจได้เฉพาะโปรแกรมเมอร์คนนั้น เพราะคำสั่งมีการเขียนเป็นลำดับ และมีการกระโดดไปมา เช่น ใช้คำสั่ง GOTO หรือที่เรียกว่า **โปรแกรม Spaghetti** ซึ่งการเขียนโปรแกรมแบบนี้ถือว่าเป็นการเขียนโปรแกรมเชิงไม่มีโครงสร้าง (Unstructured Programming) ในการเขียนโปรแกรมต้องอาศัยความขยัน ความเข้าใจอย่างดี โดยเฉพาะโปรแกรมที่มีความซับซ้อน บางครั้งเป็นเรื่องยากที่จะเข้าใจเมื่อโปรแกรมเมอร์คนอื่นมาอ่านโค้ด อาจจำเป็นต้องเขียนโปรแกรมใหม่ซึ่งจะคุ้มค่ากว่า



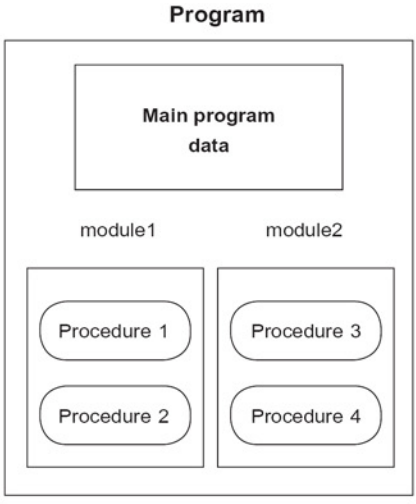
รูปที่ 1.1 *การเขียนโปรแกรมเชิงไม่มีโครงสร้าง*

ต่อมาในปี 1970 Larry LeRoy Constantine และ Edward Yourdon ช่วยกันคิดให้วิธีการเขียนโปรแกรมมีลักษณะเป็นกลุ่ม หรือบล็อก (Block) กรรมวิธีนี้รู้จักกันดีอีกชื่อหนึ่งว่า **การเขียนโปรแกรมเชิงโครงสร้าง (Structure Programming)** ที่จะช่วยให้การจัดการโปรแกรมมองดูง่ายขึ้น บางทีก็เรียกว่า **การเขียนโปรแกรมเชิงโพรซีเยอร์ (Procedural Programming)** เป็นการกำหนดหน้าที่การทำงานหรือฟังก์ชั่นที่สามารถใส่ค่าตัวแปรให้กับฟังก์ชั่น ค่าตัวแปรมีทั้งรับตัวแปรหรือส่งตัวแปรเข้า-ออก ซึ่งแล้วแต่การออกแบบโปรแกรม



รูปที่ 1.2 *การเขียนโปรแกรมเชิงโครงสร้าง*

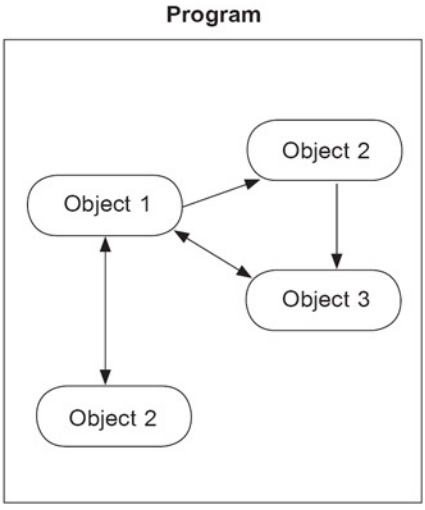
สำหรับโปรแกรมที่มีขนาดใหญ่ จะนิยมรวมโพรซีเยอร์ หรือฟังก์ชั่นไว้ด้วยกัน ซึ่งเรียกหน่วยนี้ว่า **โมดูล (Module)** การเขียนโปรแกรมแบบนี้จึงมีชื่อว่า **การเขียนโปรแกรมเชิงโมดูล (Modular Programming)** เพื่อเรียกใช้งานตามโพรซีเยอร์ หรือฟังก์ชั่น และรวมทุกโมดูลไว้ในโปรเจ็กต์ (Project) หรือบางครั้งก็เรียกว่า โซลูชั่น (Solution) วิธีการเหล่านี้ช่วยให้มองระบบได้ง่ายขึ้น และลดการใช้คำสั่ง GOTO ได้มาก นักพัฒนาโปรแกรมเพียงวิเคราะห์โครงสร้างและออกแบบด้วยการจัดการกับฟังก์ชั่นเหล่านี้ แต่สำหรับโปรแกรมที่มีความซับซ้อนและขนาดใหญ่ ก็ยังถือว่าช่วยได้น้อยอยู่ครับ



รูปที่ 1.3 *การเขียนโปรแกรมเชิงโมดูล*

ในปี 1980 Peter Chen ผู้พัฒนาแผนภาพ E-R (Entity Relationship Diagram) และ Dr. Edger Frank Codd ผู้พัฒนาฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ได้วางพื้นฐานในการพัฒนาโปรแกรมด้วยการจัดกลุ่มข้อมูลให้อยู่ในที่เดียวกัน เรียกว่า **เอ็นทิตี้ (Entity)** แต่ส่วนนี้เป็นเพียงกลุ่มข้อมูล หรือใช้สำหรับสร้างฐานข้อมูล ดังนั้นในการออกแบบฐานข้อมูลก็ใช้ลักษณะเอ็นทิตี้หรือเชิงวัตถุได้ แต่โปรแกรมที่ใช้งานยังคงเขียนแนวโครงสร้างหรือเชิงโมดูล การใช้ประโยชน์ที่แยกกันของข้อมูลและหน้าที่การใช้งานจากการเขียนโปรแกรมจึงยังไม่สมบูรณ์ จำเป็นต้องมีกรรมวิธีอื่นๆ ที่จะทำให้ทั้งข้อมูลและคำสั่งใช้งานทั้งสองอย่างรวมกันได้

การโปรแกรมเชิงวัตถุจะช่วยแก้ปัญหาข้างต้น ที่จะทำให้นักพัฒนาโปรแกรมมีแนวทางทั้งสองทัศนะที่รวมเป็นเรื่องเดียวกัน ผลที่ได้คือนักพัฒนาสามารถพัฒนาโปรแกรมที่มีความซับซ้อนและขนาดโครงการใหญ่ๆ ที่มีความยืดหยุ่นต่อการปรับแต่งและดูแลรักษาระบบง่ายขึ้นอีกด้วย



รูปที่ 1.4 *การเขียนโปรแกรมเชิงวัตถุ*

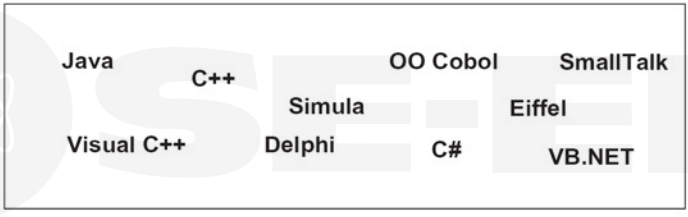
การเขียนโปรแกรมเชิงวัตถุ

ตามประวัติของภาษาเขียนโปรแกรมเชิงวัตถุ ภาษา SmallTalk เป็นภาษาแรกๆ ที่ถูกพัฒนาโดย Alan Kay และคณะผู้ร่วมงานของบริษัท Xerox Palo Alto Research Center ในต้นปี 1970 ภาษา SmallTalk ถูกใช้เขียนโปรแกรมเชิงวัตถุในลักษณะใช้งานทั่วไป ซึ่ง Kay ต้องการนำภาษานี้ไปใช้กับคอมพิวเตอร์ที่ใช้งานบนยานอวกาศ ซึ่งต้องการใช้งานกับคอมพิวเตอร์ขนาดเล็ก แต่ไม่เล็กเท่าคอมพิวเตอร์พกพาอย่างปัจจุบัน คอมพิวเตอร์ขนาดเล็กในยุคนั้นก็ยังมีขนาดใหญ่มาก และ Kay ได้พัฒนาภาษาเพื่อการเขียนโปรแกรม Small-Talk ที่กำหนดเป็นวัตถุเพื่อสื่อสารกับผู้ใช้ หรือที่เรียกว่า Graphic User Interface เรียกสั้นๆ ว่า GUI ซึ่งต่อมาได้พัฒนาใช้งานเชิงพาณิชย์เช่นเดียวกัน อีกสิบปีต่อมาคอมพิวเตอร์ส่วนบุคคลของ Apple Macintosh ก็ใช้หลักการ GUI จนมีชื่อเสียง

นอกจากนี้ยังมีภาษาเขียนโปรแกรมเชิงวัตถุอื่นๆ อีกหลายตัว เช่น Objective-C, Eiffel และตัวที่นิยมใช้กันมากคือ C++ เพราะสามารถพัฒนาจากภาษา C เดิมได้ รวมทั้งสามารถเขียนได้ทั้งแบบเชิงโครงสร้างและเชิงวัตถุได้พร้อมๆ กัน ต่อมาบริษัท ไมโครซอฟต์

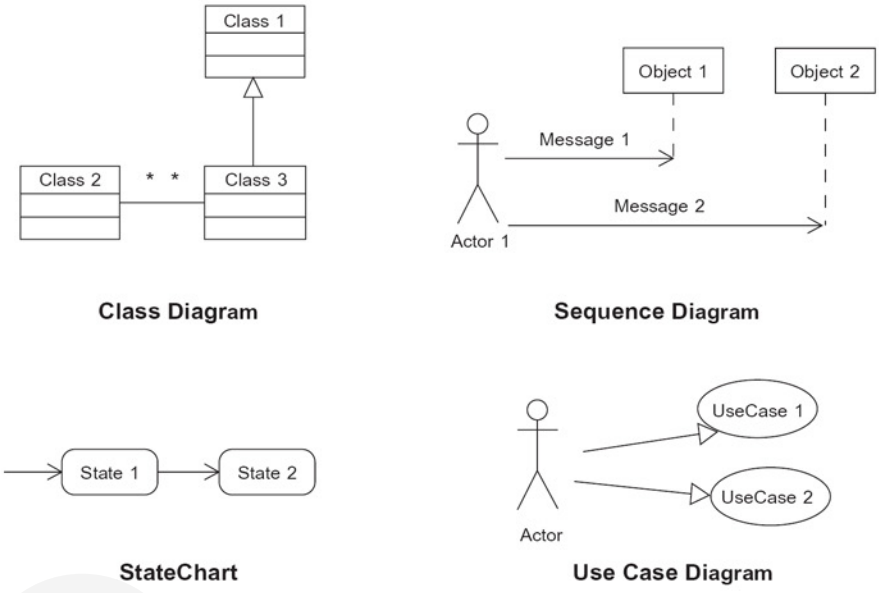
ได้พัฒนาให้มีการใช้งานได้สะดวกรวดเร็ว รู้จักกันในชื่อ Visual C++ ซึ่งอยู่ในชุด Visual Studio สำหรับ Object Oriented COBOL ก็นิยมใช้เขียนโปรแกรมเชิงธุรกิจ

ในปี 1995 บริษัท ซัน ไมโครซิสเต็มส์ ได้พัฒนาโปรแกรมภาษา Java เป็นภาษาเชิงวัตถุอย่างสมบูรณ์ ที่มีโครงสร้างภาษาคล้ายกับภาษา C++ เป็นภาษาที่เหมาะสำหรับใช้งานทางอินเทอร์เน็ต (ใช้ Applets) และสามารถทำงานได้หลายแพล็ตฟอร์ม (Platform) ที่สามารถทำงานได้กับสถาปัตยกรรมฮาร์ดแวร์ต่างๆ ได้หลากหลาย และใช้กับหลายระบบปฏิบัติการ บริษัท ไมโครซอฟต์ ได้พัฒนาภาษา Java บนสภาพแวดล้อมใหม่ แล้วให้ชื่อว่า J++ นอกจากนี้บริษัท ไมโครซอฟต์ ยังมีการพัฒนาภาษาเชิงวัตถุอีกหลายตัว คือ C# (อ่านว่า ซีชาร์ป) และ VB.NET ที่อยู่บนกรอบการทำงาน .NET (.NET Framework) โดยเฉพาะ VB.NET ถือเป็นภาษาเชิงวัตถุที่สมบูรณ์และสามารถนำไปพัฒนาโปรแกรมเชิงวัตถุได้รวดเร็ว



รูปที่ 1.5 *ภาษาเขียนโปรแกรมเชิงวัตถุ*

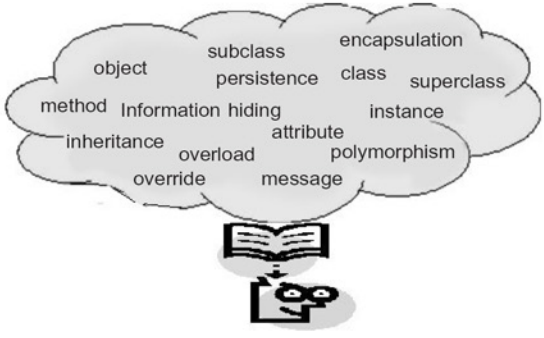
นอกจากภาษาเชิงวัตถุที่นำไปเขียนโปรแกรมเพื่อใช้งานแล้ว ต้องมีเครื่องมือในการวิเคราะห์ระบบเชิงวัตถุด้วยก่อนจะเขียนโปรแกรม เรียกว่า **ภาษา UML (Unified Modeling Language)** ซึ่งมีลักษณะเป็นภาษารูปภาพ ณ ปัจจุบันใช้กันแพร่หลายแทนเครื่องมือเดิมที่ใช้วิเคราะห์เชิงวัตถุ เพราะได้มีการปรับปรุง และกำหนดมาตรฐานจากผู้คิดค้นเครื่องมือวิเคราะห์เชิงวัตถุ รวมตัวกันออกมาตรฐานภาษา UML และปัจจุบันก็ยังมีการพัฒนาต่อไปเรื่อยๆ



รูปที่ 1.6 *UML Diagrams*

แนวคิดเชิงวัตถุ

ในการอธิบายแนวคิด 3 ประการของการเขียนโปรแกรมเชิงวัตถุนั้น มีคำศัพท์หลายตัวที่นิยมใช้กันอยู่ และบางครั้งก็สร้างความสับสนในการกล่าวถึง จึงควรทำความเข้าใจคำศัพท์สำคัญเหล่านี้ด้วย เพื่อความเข้าใจแนวคิดของการเขียนโปรแกรมเชิงวัตถุได้ดีขึ้น



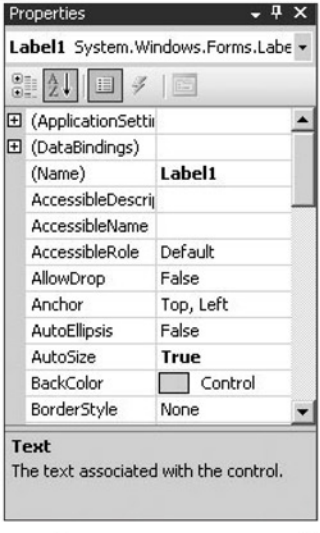
รูปที่ 1.7 *คำศัพท์หลักๆ ของภาษาเชิงวัตถุ*

ออบเจ็กต์ แอตตริบิวต์ และเมธอด

ออบเจ็กต์ ในทางคอมพิวเตอร์ก็เหมือนกับวัตถุในโลกความเป็นจริง ที่หมายถึงสิ่งของหรือสิ่งมีชีวิต ที่มีทั้งคุณสมบัติและพฤติกรรม เช่น รถยนต์ มีคุณสมบัติคือ มีสี, มีล้อ, มีประตู, มีพฤติกรรมหรือกิริยา คือ แล่นเดินหน้าได้, ถอยหลังได้, หยุดได้, ประตูเปิดได้ เป็นต้น ในทางคอมพิวเตอร์มีออบเจ็กต์มากมายหลายแบบ เช่น GUI เป็นออบเจ็กต์ที่ใช้งานเป็นสื่อในการติดต่อระหว่างผู้ใช้งานกับคอมพิวเตอร์ ซึ่ง GUI ก็มีหลายแบบ เช่น Label, Button, ComboBox, TextBox ฯลฯ GUI เหล่านี้ก็มีคุณสมบัติในทางภาษาเชิงวัตถุเรียกว่า **แอตตริบิวต์** เช่น Label มีแอตตริบิวต์ เป็นขนาด 16 มีสีดำ แอตตริบิวต์เหล่านี้สามารถกำหนดได้จากหน้าต่างพร็อปเพอร์ตี้ของ VB.NET

พร็อปเพอร์ตี้ มีพฤติกรรมคล้ายเมธอด และคล้ายแอตตริบิวต์ได้ด้วย ในบางครั้งจึงจัดให้พร็อปเพอร์ตี้เป็นเมธอดแบบพิเศษ

ออบเจ็กต์นอกจากจะมีแอตตริบิวต์แล้ว ยังมีพฤติกรรมในทางภาษาเชิงวัตถุเรียกว่า **เมธอด** ซึ่งเหมือนกับออบเจ็กต์ คือมีกิริยา, อาการ, พฤติกรรม เช่น Button มีเมธอด คือสามารถคลิกได้, Label มีเมธอด คือสามารถแสดงข้อความได้ เป็นต้น



รูปที่ 1.8 *หน้าต่างพร็อปเพอร์ตี้*

สำหรับออบเจ็กต์ที่ใช้สำหรับปัญหาใดปัญหาหนึ่ง หรือเรื่องใดเรื่องหนึ่ง โดยเฉพาะใช้สำหรับโปรแกรมประยุกต์ทางธุรกิจ เราจะเรียกว่า **Problem Domain Objects** ตัวอย่าง เช่น ในการวิเคราะห์ระบบการทำงานทางธุรกิจ เราจะพบว่า ต้องมีลูกค้า, ใบสั่งซื้อ, สินค้า หรืออื่นๆ ซึ่งขึ้นอยู่กับการวิเคราะห์ระบบ คล้ายๆ กับที่เราได้วิเคราะห์ E-R Diagram เพียงแต่ใน E-R มีเพียงข้อมูลหรือแอตตริบิวต์ แต่ไม่มีเมธอด ในการวิเคราะห์ปัญหาเชิงวัตถุโดยทั่วไปแล้ว ออบเจ็กต์มีทั้งแอตตริบิวต์และเมธอด เช่น ออบเจ็กต์ลูกค้า มีแอตตริบิวต์คือ ชื่อ, ที่อยู่, เบอร์โทรศัพท์ และมีเมธอดคือ เพิ่มชื่อได้, สร้างที่อยู่ได้, ออกคำสั่งซื้อได้ เป็นต้น

เมสเสจ (Message)

ออบเจ็กต์มีการกระทำต่อกัน (Object Interaction) การกระทำต่อกันก็คือการส่งข้อความระหว่างกัน ข้อความหรือเมสเสจนี้เป็นคำสั่ง หรือการร้องขอบริการ อาจเปรียบเป็นผู้ส่ง (Sender) (หรือไคลเอนต์ (Client)) และผู้รับข้อความอาจเปรียบได้เป็นผู้รับ (Receiver) หรือผู้ให้บริการ (Server)

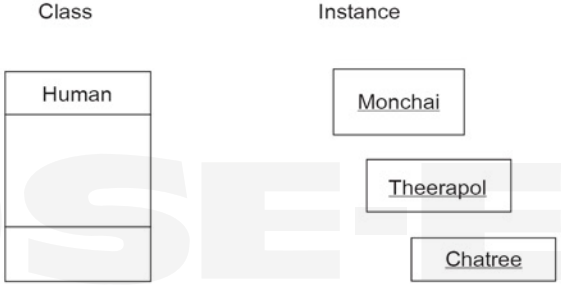
สำหรับการวิเคราะห์และออกแบบเชิงวัตถุ การส่งเมสเสจนั้น เป็นคุณสมบัติสำคัญอย่างหนึ่งที่จะให้ออบเจ็กต์ดำเนินการระหว่างกัน ความแตกต่างที่สำคัญกับการวิเคราะห์และออกแบบโดยใช้โพรซีเยอร์ คือ การส่งเมสเสจ ระหว่างโพรซีเยอร์หรือฟังก์ชั่น ส่วนการส่งเมสเสจแบบเชิงวัตถุเป็นการส่งภายในออบเจ็กต์ เพราะมีการนิยามเมธอดภายในคลาสแล้ว หรือจะเป็นการส่งเมสเสจระหว่างออบเจ็กต์ก็ได้

รูปที่ 1.9 *การส่งเมสเสจระหว่างกัน*

คลาสและอินสแตนซ์

ในกระบวนการทำงานเชิงวัตถุ ทุกออบเจ็กต์เกิดจากคลาส ดังนั้นคลาสจึงเปรียบเสมือนแม่แบบที่จะผลิตออบเจ็กต์ ออบเจ็กต์ที่ผลิตแล้วจะเรียกว่า **อินสแตนซ์** (หมายถึง ตัวแทนหรือตัวอย่าง) ใน 1 คลาส มีอินสแตนซ์ได้มากมาย เช่น ในคลาสของมนุษย์ สามารถสร้างอินสแตนซ์นายมนต์ชัย, นายธีระพล, นายชาตรี เป็นต้น ทุกอินสแตนซ์มาจากแม่แบบคลาสมนุษย์

คำว่า **ออบเจ็กต์** บางครั้งใช้แทนกันไปมาได้กับศัพท์ **อินสแตนซ์** แต่ออบเจ็กต์มักเป็นคำศัพท์กว้างๆ มากกว่าอินสแตนซ์

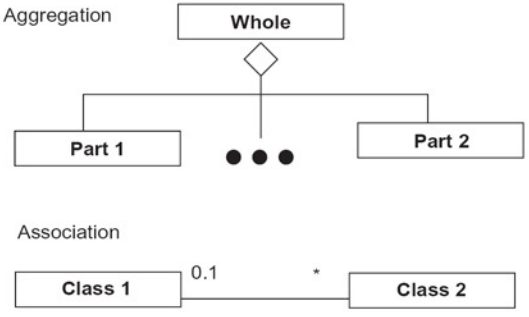


รูปที่ 1.10 *คลาสและอินสแตนซ์*

ความสัมพันธ์แบบ Association และ Aggregation

อินสแตนซ์แต่ละตัวที่ถูกสร้างขึ้นจะมีความสัมพันธ์กัน เช่น นายมนต์ชัยเป็นน้องชายนายธีระพล และนายธีระพลเป็นน้องชายนายชาตรี ก็เป็นความสัมพันธ์แบบหนึ่งต่อหนึ่ง เป็นต้น แต่เราจะเรียกว่า มี Associate กัน ซึ่งมีการแปลได้มากมาย เช่น แปลว่าเชื่อมโยงกัน, เกี่ยวพันกัน เป็นต้น จนกว่าจะมีบัญญัติอย่างเป็นทางการ ส่วน Aggregation หมายถึงการประกอบด้วย เช่น รถประกอบด้วย ประตู 4 ชิ้น, มีล้อ 4 ล้อ เป็นต้น

เราควรเข้าใจในความที่ใกล้เคียงกันนี้ ของความสัมพันธ์แบบ Association และ Aggregation เพราะสัญลักษณ์ที่เขียนประกอบการวิเคราะห์จะใช้สัญลักษณ์ที่แตกต่างกันไป และจะง่ายในการนำไปเขียนโปรแกรมเชิงวัตถุต่อไป



รูปที่ 1.11 *ภาษาสัญลักษณ์ UML ที่มีความสัมพันธ์แบบ Association และ Aggregation*

ตารางที่ 2.1 คำศัพท์ทางภาษาเชิงวัตถุ

คำศัพท์	ความหมาย
Object	การรวมกลุ่มของซอฟต์แวร์ ที่ประกอบด้วยข้อมูลและเมธอด
Class	แม่แบบที่ใช้สร้างเป็นวัตถุ
Attribute	ข้อมูลที่เป็นส่วนหนึ่งของคลาสหรือวัตถุ
Operation	เป็นชุดของคำสั่งหรือหน้าที่การทำงาน ที่กำหนดให้เป็นส่วนหนึ่งของคลาสหรือวัตถุ มักใช้คำนี้ในกรณีที่ประกาศใช้วัตถุใดๆ ที่เรียกใช้บริการ และกับขั้นตอนการนำไปใช้
Method	เป็นชุดของคำสั่งหรือหน้าที่การทำงาน ที่กำหนดให้เป็นส่วนหนึ่งของคลาสหรือวัตถุ มักใช้คำนี้ในกรณีอยู่ในขั้นตอนการออกแบบ
Message	เป็นการส่งหรือสื่อสารของวัตถุ ระหว่างเมธอด
Encapsulation	เป็นการรวมกันของข้อมูลและเมธอด แล้วสร้างส่วนที่เปิดเผยได้ และส่วนที่ไม่เปิดเผย

ตารางที่ 2.1 (ต่อ) คำศัพท์ทางภาษาเชิงวัตถุ

คำศัพท์	ความหมาย
Data Hiding	เป็นการสร้างการซ่อนข้อมูลไม่ให้วัตถุอื่นเข้าถึงได้
Inheritance	เป็นการสร้างคลาสใหม่จากพื้นฐานของคลาสเดิม ที่เคยได้สร้างไว้แล้ว
Polymorphism	เป็นความสามารถที่ซ่อนส่วนการเขียนรายละเอียดที่แตกต่างกันจากคลาสต่างๆ แต่สามารถตอบสนองการเขียน การใช้งานที่เหมือนกันได้

แนวคิดสำคัญในการเขียนโปรแกรมเชิงวัตถุ มีอยู่ 3 ประการคือ

1. **เอ็นแคปซูเลชั่น (Encapsulation)** ที่เปรียบเสมือนการรวมข้อมูลและหน้าที่การทำงานเข้าด้วยกัน เช่น ออบเจ็กต์รถยนต์ มีข้อมูลหรือคุณสมบัติเรียกว่า *แอตตริบิวต์* คือ รถสีขาว, มี 4 ล้อ, มีพวงมาลัย และมีหน้าที่การทำงาน หรือจะเรียกว่า *เมธอด* คือ วิ่งได้, เปิดไฟได้ เป็นต้น ซึ่งเป็นการรวมทั้งแอตตริบิวต์และเมธอดไว้ด้วยกัน จะมีวิธีป้องกันข้อมูลหรือเข้าถึงข้อมูล ด้วยการกำหนดการเข้าถึงเป็น Private ซึ่งหมายถึงข้อมูลนี้จะไม่สามารถเข้าถึงได้โดยตรง และโดยทั่วไปแล้วมักกำหนดให้แอตตริบิวต์มีลักษณะนี้ ซึ่งการเข้าถึงได้จะต้องเป็น Public ในส่วนที่สามารถเข้าถึงและใช้งานได้นี้ เปรียบได้ดังประตูที่เป็นทางเข้าสู่วัตถุ ถือเป็นการซ่อนข้อมูล (Information Hiding) สำหรับการสร้างประตูให้เข้าสู่ข้อมูลภายในวัตถุ ก็เพื่อป้องกันการแก้ไขข้อมูลของแอตตริบิวต์โดยตรง เพื่อความปลอดภัยของข้อมูล หรือให้การเข้าสู่ข้อมูลได้ต้องเป็นไปตามกฎเกณฑ์ของประตูที่จะเขียนกำหนดไว้
2. **อินเฮอริแทนซ์ (Inheritance)** เปรียบเสมือนการสืบทอดคุณสมบัติจากรุ่นพ่อมาสู่รุ่นลูก คุณสมบัติการสืบทอดจะช่วยให้การเขียนโปรแกรมเชิงวัตถุมีความสะดวกในการเขียนโปรแกรมที่ไม่ต้องเขียนใหม่ทั้งหมด และยังสามารถจัดการกับวัตถุได้ง่ายขึ้น ตัวอย่างเช่น คลาสมนุษย์ มีคุณสมบัติคือ มี 2 มือ 2 ขา เมื่อสร้างคลาสรุ่นลูกแล้ว ก็จะสืบทอดคุณสมบัติคือ มี 2 มือ 2 ขามาด้วย เราเรียกคลาสที่สืบทอดว่า **Derived Class** การสืบทอดของคลาสจะสืบทอดได้ทีละคลาส แต่จะมี Derived Class ที่เขียนเมธอดเหมือนคลาสฐาน โดยเหตุผลที่ต้องการเขียนเพื่อปรับปรุงให้เหมาะกับคลาสตนเอง การทำอย่างนี้เรียกว่า

โอเวอร์ไรด์ (Override) แต่ถ้าภายในคลาสเดียวกันนี้มีชื่อเมธอดเหมือนกัน จะเรียกว่าเป็นการทำ **โอเวอร์โหลด (Overload)**

3. **โพลีมอร์ฟิซึม (Polymorphism)** ตามศัพท์แปลว่า มีหลายรูปแบบ ในเชิงวัตถุหมายถึง การตอบสนองได้หลากหลาย เช่น การรับเมสเสจหรือข้อความใดๆ ในรูปแบบต่างๆ แต่สามารถตอบสนองการทำงานได้ ลักษณะนี้ในภาษาเชิงวัตถุจะใช้คุณสมบัติการสืบทอด รวมกับการเขียนทับเมธอดเดิมที่สืบทอด หรือการทำโอเวอร์ไรด์ทั้งคลาสเดิมและคลาสที่สืบทอดมีเมธอดเดียวกัน แต่มีการเขียนการทำงานภายในไม่เหมือนกัน การตอบสนองในรูปแบบที่ต่างกัน อยู่ที่การเรียกใช้เมธอดที่ถูกโอเวอร์ไรด์ของคลาสฐานที่ชี้ตำแหน่งไปยังคลาสลูก แล้วผลลัพธ์ก็จะมีผลตามที่ได้เขียนโอเวอร์ไรด์ในคลาสลูก

   การทำโอเวอร์ไรด์ทำได้ 3 กรณีคือ

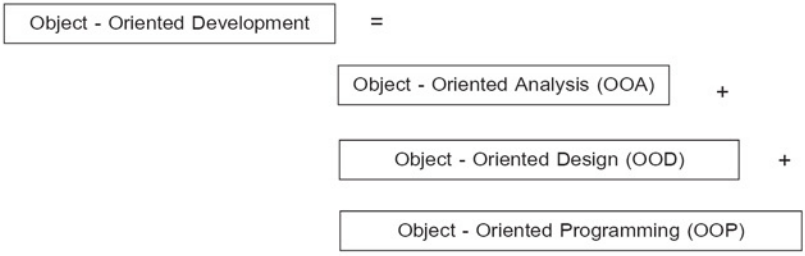
   - การสร้างคลาสฐานแบบทั่วไป แล้วทำโอเวอร์ไรด์ที่คลาสลูก
   - การสร้างคลาสฐานที่มีลักษณะเป็น Abstract คือบังคับให้ต้องเขียนโอเวอร์ไรด์ที่คลาสลูก
   - การเขียนคลาสที่สืบทอดจากอินเตอร์เฟซ (Interface) ซึ่งเป็นโครงสร้างที่บังคับให้ต้องเขียนขยายเพิ่มเติมในคลาสลูก

การใช้ลักษณะโพลีมอร์ฟิซึม มีประโยชน์คือ ผู้ใช้งานไม่จำเป็นต้องทราบชนิดของคลาสที่วัตถุนั้นสืบทอดมา ในบางครั้งการทำโอเวอร์โหลดก็ถือว่าเป็นโพลีมอร์ฟิซึมได้ แต่มักไม่ค่อยมีใครกล่าวถึงมากนัก เนื่องจากวิธีการทำโอเวอร์โหลดยังไม่ถือเป็นการทำโพลีมอร์ฟิซึมอย่างแท้จริง

การพัฒนาโปรแกรมเชิงวัตถุ

การจะพัฒนาโปรแกรมได้ เราต้องเข้าใจแนวคิด *การวิเคราะห์ระบบ, การออกแบบระบบ และการอิมพลีเมนต์ระบบ* ซึ่งการอิมพลีเมนต์ในที่นี้เน้นตีความเป็นการเขียนโปรแกรม ซึ่งแนวคิดทั้งสามอย่างนี้สามารถพัฒนาได้ทั้งแบบเชิงวัตถุหรือไม่เชิงวัตถุ

ก็ได้ (วิธีเก่า) หากจะพัฒนาเชิงวัตถุ มีรูปแบบ 3 ขั้นตอน คือ วิเคราะห์, ออกแบบ และอิมพลีเมนต์ ซึ่งระบบจะแตกต่างกันมากจากวิธีเก่า เราจะได้ศึกษาในบทต่อไป



รูปที่ 1.12 *การพัฒนาระบบเชิงวัตถุ*

การพัฒนาโปรแกรมเชิงวัตถุ ในการเรียนครั้งนี้จะเน้นไปที่การพัฒนาเพื่อพัฒนาระบบสารสนเทศ โดยใช้เครื่องมือที่นิยม อย่างเช่น VB.NET จึงมีหลักเกี่ยวพันด้วยกัน 3 ส่วนคือ

1. การตีความหมายคลาสจากปัญหาที่สนใจ (Problem Domain Class) เช่น คลาสลูกค้า, คลาสคำสั่งซื้อ, คลาสสินค้า เป็นต้น
2. การใช้ GUI Class เพื่อสร้างการเชื่อมต่อระบบกับผู้ใช้ระบบ เช่น ฟอร์ม, ปุ่มคำสั่ง, Label, ลิสต์บ็อกซ์ หรือในรูปเท็กซ์โหมด
3. การเข้าถึงข้อมูล โดยใช้ Data Access Class จะทำงานร่วมกับระบบจัดการฐานข้อมูล (Database Management System) คอยจัดเก็บจากคลาสทางธุรกิจให้สามารถเรียกใช้งานในเวลาต่อไป

ลักษณะการทำงานร่วมกันทั้ง 3 ประเภทข้างต้นนี้ ถือว่าเป็นระบบ 3 ส่วนที่เชื่อมโยงกัน เรียกศัพท์ในการออกแบบว่า Three-Tier Design เป็นสถาปัตยกรรมซอฟต์แวร์ที่นิยมใช้กันแพร่หลาย

The Microsoft.NET Framework, Visual Basic.NET (VB.NET)

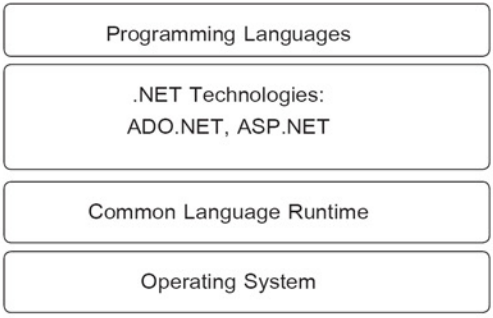
สถาปัตยกรรมของ .NET สร้างโดยบริษัท ไมโครซอฟต์ สร้างขึ้นมาเพื่ออำนวยความสะดวกในการเขียนโปรแกรมภาษาต่างๆ ที่มีสภาพแวดล้อมบน .NET ให้สามารถทำงานร่วมกันได้บนระบบปฏิบัติการ Windows (Microsoft Windows Operation System) ส่วนประกอบสำคัญ 4 ส่วนของโครงสร้าง .NET คือ

1. **Programming Languages** มีภาษาที่สนับสนุนการใช้งาน .NET อยู่หลายภาษา โดยเฉพาะภาษาที่อยู่ในเครื่องมือพัฒนาของ Visual Studio คือภาษา Basic, C#, J#, C++, JScript.NET นอกจากภาษาเหล่านี้ใน Visual Studio แล้ว ยังมีภาษาอื่นๆ เช่น ภาษา Pascal ของเครื่องมือชื่อ Delphi ของบริษัท Borland

2. **.NET Technologies** เป็นเทคโนโลยีที่ถูกจัดเป็นกลุ่มเพื่อใช้งานในด้านต่างๆ โดยนำมาจากคลาสไลบรารี เช่น กลุ่ม ADO.NET ซึ่งเป็นกลุ่มเทคโนโลยีเพื่อการใช้งานฐานข้อมูล, กลุ่ม, เว็บ, ฟอร์ม (ASP.NET) ซึ่งกลุ่ม Windows ฟอร์ม เป็นกลุ่มงานที่ใช้พัฒนาโปรแกรมประยุกต์บนเว็บ และบน Windows

3. **.NET Class Library** เป็นชุดของคลาสที่มีพร้อมให้ใช้งาน โดยไม่ต้องประดิษฐ์หรือเขียนขึ้นเองซึ่งเป็นข้อดี การได้ศึกษาคลาสไลบรารีชุดนี้ ทำให้สามารถสื่อสาร หรืออ่านภาษาใดๆ ที่ใช้คลาสไลบรารี ให้เข้าใจโดยง่าย

4. **Common Language Runtime (CLR)** คือกลไกการทำงานที่ .NET ทุกโปรแกรมต้องทำงานผ่าน CRL และมีบริการต่างๆ เพิ่มเติม เช่น การตรวจสอบความผิดพลาด การจัดการหน่วยความจำ ซึ่งทำให้ผู้พัฒนาระบบไม่ต้องกังวล หรือต้องเขียนโปรแกรมในส่วนเหล่านี้



รูปที่ 1.13 *.NET Framework*

หน้าที่ของ CLR คือ คอยจัดการคำสั่งให้โปรแกรมทำงานบนระบบปฏิบัติการ Windows หรือคอยเป็นตัวกลางในการจัดการกับระบบปฏิบัติการ เช่น การจัดการกับหน่วยความจำ, การจัดการโปรเซส, การคอมไพล์ หรือบริการอื่นๆ ซึ่งบริการเหล่านี้นักพัฒนาไม่จำเป็นต้องเสียเวลาปรับแต่งเอง ปล่อยให้ CLR คอยดำเนินการแทนให้ นักพัฒนาเพียงใช้โปรแกรมภาษา เช่น VB, C++, C#, J++ ที่ตนเองถนัดพัฒนาโปรแกรมในส่วนต่างๆ ดังนั้น โปรแกรมหนึ่งอาจมาจากหลายภาษาได้ เมื่อนำมารวมกันก็สามารถทำงานได้อย่างไม่มีปัญหา ซึ่ง CLR ใช้โค้ดสื่อกลางหรือโค้ดกลางจากการคอมไพล์ โดยคอมไพเลอร์ของภาษานั้นๆ ไม่ว่าจะเขียนด้วยอะไร จะถูกแปลงเป็นโค้ดกลางก่อนที่จะถูกคอมไพล์อีกครั้งเป็นภาษาเครื่อง ซึ่งเป็นภาษาสุดท้ายที่คอมพิวเตอร์ใช้ทำงานได้ ดังนั้นทุกครั้งที่จะทำงานได้จะต้องคอมไพล์เป็นภาษาเครื่องก่อน ลักษณะการคอมไพล์แบบนี้จะเรียกว่า just-in-time ถ้าเครื่องคอมพิวเตอร์มีสถาปัตยกรรม .NET ก็จะสามารถใช้งานโค้ดกลางนี้ได้ แต่น่าเสียดายว่าตอนนี้มีเพียงระบบปฏิบัติการ Windows เท่านั้นที่เรียกใช้งานได้ ซึ่งยังไม่มีลักษณะเป็นอิสระทางแพล็ตฟอร์ม (Platform Independence) อย่างแท้จริง

Visual Basic มีความเป็นมายาวนาน ตั้งแต่รุ่น VB1, VB2 ผู้ที่เคยใช้รุ่นเหล่านี้ถือว่าเป็นผู้ชำนาญการ, VB3 เป็นรุ่นที่มาพร้อมกับการทำงานกับฐานข้อมูล, VB4 เริ่มมีลักษณะเป็นเชิงวัตถุ, VB5, VB6 มีเพิ่ม ActiveX, ActiveX Data Objects (ADO) และ Distributed Component Object Model (DCOM) มีหลายอย่างเพิ่มขึ้นซึ่ง VB6 แม้จะสนับสนุนการเขียนโปรแกรมเชิงวัตถุ เช่น มีระดับการเข้าถึงวัตถุ ในระดับ Private, Friend, Public

และ Static แต่ไม่สมบูรณ์มากนัก เพราะยังไม่ครอบคลุมการเข้าถึงระดับ Protected ใน VB.NET เป็นรุ่นที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุได้มากขึ้น และถือว่ามีความสมบูรณ์สำหรับการเขียนโปรแกรมเชิงวัตถุ เพราะสามารถกำหนดระดับการเข้าถึงวัตถุในระดับ Protected และ Protected Friend ได้ รวมถึงความสามารถอื่นๆ เช่น การสร้างคอนสตรักเตอร์ด้วยคีย์เวิร์ด New ซึ่งไม่มีใน VB6 และเนื่องจาก VB.NET สนับสนุนเทคโนโลยี .NET จึงสามารถใช้คลาสไลบรารีจาก .NET และ .NET คลาสไลบรารีนี้มีคุณสมบัติเชิงวัตถุ ทำให้ VB.NET ใช้คุณสมบัติเชิงวัตถุจากคลาสไลบรารีได้ ด้วยการเรียนรู้การเขียนโปรแกรมเชิงวัตถุ การเข้าใจแนวคิดเชิงวัตถุจะเป็นส่วนสำคัญสำหรับการเข้าใจเทคโนโลยี .NET ของ VB ตัวใหม่นี้

สรุปท้ายบท

วิวัฒนาการทางการเขียนโปรแกรม ตั้งแต่การเขียนโปรแกรมไม่เป็นโครงสร้างมาสู่ยุคปัจจุบันที่เน้นกันมากคือ การเขียนโปรแกรมเชิงวัตถุ และก็มีหลายภาษาที่สนับสนุน เนื่องจากขนาดของโปรแกรมที่ใช้งานมีขนาดใหญ่มากขึ้น มีความซับซ้อนสูงขึ้น ภาษาเชิงวัตถุสนับสนุนโครงสร้างโปรแกรมขนาดใหญ่ได้เป็นอย่างดี ในการเริ่มต้นการศึกษาการเขียนโปรแกรมเชิงวัตถุ มีการใช้ศัพท์พื้นฐานทางการเขียนโปรแกรมเชิงวัตถุหลายตัว เราควรเข้าใจคำศัพท์พื้นฐาน อย่างเช่น คลาส, วัตถุ, แอตตริบิวต์, เมธอด หรือแม้กระทั่งแนวคิดสำคัญของการเขียนโปรแกรมเชิงวัตถุ 3 ประการ คือ เอ็นแคปซูเลชั่น, อินเฮอริแทนซ์ และโพลีมอร์ฟิซึม สามารถนำความรู้การเขียนโปรแกรมเชิงวัตถุไปพัฒนาเป็นโปรแกรมประยุกต์เชิงวัตถุได้ ด้วยเครื่องมือที่เหมาะสม อย่างในที่นี้แนะนำการเขียนโปรแกรม VB.NET ซึ่งมีคุณสมบัติเชิงวัตถุอย่างครบครัน และอยู่บนสถาปัตยกรรม .NET Framework ที่รองรับการทำงานได้หลายภาษา ซึ่งเป็นไปได้ว่า จะสื่อสารกับโปรแกรมเมอร์อื่นๆ ที่ใช้ภาษาต่างกัน แต่อยู่สถาปัตยกรรมเดียวกันได้อย่างมีความเข้าใจต่อกันและทำงานร่วมกันได้

แบบฝึกหัดท้ายบท

ตอนที่ 1 ตอบคำถามต่อไปนี้

1. จงอธิบายแนวคิด การเขียนโปรแกรมเชิงโครงสร้างและแนวคิดเชิงวัตถุ
2. ให้ยกตัวอย่างภาษาที่ไม่สนับสนุนการเขียนโปรแกรมเชิงวัตถุมา 5 ชื่อ
3. ภาษา UML เป็นภาษาเชิงวัตถุหรือไม่ และใช้สำหรับทำอะไร
4. แนวคิดเชิงวัตถุที่สำคัญ โดยทั่วไปมีอยู่ 3 ประการคืออะไร
5. ให้อธิบายคำศัพท์เชิงวัตถุเหล่านี้ ให้เป็นเรื่องราวที่ต่อเนื่องกันภายในหนึ่งย่อหน้า Object, Class, Attribute, Method, Message, Encapsulation, Data Hiding, Inheritance และ Polymorphism
6. แนวคิดการพัฒนาระบบเชิงวัตถุแบบ Three-Tier Design คืออะไร

ตอนที่ 2 คำถามวิเคราะห์

แนวคิดการเขียนโปรแกรมเชิงวัตถุกับแนวคิดการเขียนโปรแกรมเชิงโครงสร้าง และโมดูล จะยังคงนิยมใช้เขียนกันหรือไม่ และสามารถใช้ร่วมกับแนวคิดการเขียนโปรแกรมเชิงวัตถุได้หรือไม่ อย่างไร

ตอนที่ 3 คำถามค้นคว้าเพิ่มเติม

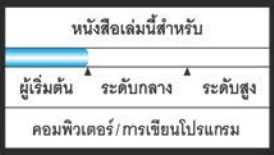
1. ให้ค้นคว้าข้อมูลเกี่ยวกับเทคโนโลยี .NET และเทคโนโลยี Java เพื่อเปรียบเทียบเทคโนโลยีทั้งสองแบบ ในแง่มุมต่างๆ
2. ให้หาความแตกต่าง ระหว่าง VB6 กับ VB.NET โดยใช้แง่มุมการเขียนโปรแกรมเชิงวัตถุ

■ ■ ■

การเขียน
โปรแกรมเชิงวัตถุ
ด้วย
Visual Basic.NET

เมื่อ Visual Basic พร้อมสมบูรณ์สำหรับการเขียนโปรแกรมเชิงวัตถุภายใต้ .NET Framework ดังนั้น Visual Basic จึงเป็นเทคนิคและเทคโนโลยีที่ใช้พัฒนาโปรแกรมประยุกต์ที่เร็วที่สุดตัวหนึ่งบนเครื่องมือพัฒนา MS Visual Studio ที่คุณควรเลือกใช้เป็นทางเลือกอันดับต้นๆ

เนื้อหาประกอบด้วย



ISBN 978-616-08-0160-2

149 บาท

www.se-ed.com