Java OOP Programming

การเขียนโปรแกรมเชิงวัตถุด้วยภาษาจาวา

เนื้อหาประกอบด้วย

- ความหมายและแนวคิดการเขียนโปรแกรมเชิงวัตถุ
- ภาษายูเอ็มแอล
- ความรู้ทั่วไปเกี่ยวกับภาษาจาวา
- การจำลองสรรพสิ่งด้วยแนวคิดเชิงวัตถุ
- คลาสและเมธอด
- ความสัมพันธ์ของคลาส
- การรับทอด
- อินเทอร์เฟสและแอบสแทรกคลาส
- ภาวะพหุสัณฐาน
- ดีไซน์แพทเทิร์นและเฟรมเวิร์ก

โดย เชาวลิต ขันคำ

คำนำ

การเขียนโปรแกรมคอมพิวเตอร์อุปมาเหมือนกับการเขียนเรียงความหรืองานเขียนหนึ่งเรื่อง หมายความว่า การเขียนโปรแกรมคอมพิวเตอร์ ต้องวางโครงเรื่อง เรียนรู้และสรรหาคำสั่งเรียบเรียงคำสั่งต่าง ๆ เข้าด้วยกัน เพื่อให้ได้งานที่ดีและตรงตามวัตถุประสงค์ที่ต้องการ

การเขียนโปรแกรมยุคใหม่ เรียกว่า 4GL (4^{th} Generation Language) เป็นการเขียนโปรแกรมเชิงวัตถุ ที่ต้องเริ่มคิดแก้ปัญหาโดยกำหนดสิ่งที่เกี่ยวข้องออกมาเป็นวัตถุ และขับเคลื่อนโปรแกรมโดยพฤติกรรมของวัตถุที่สร้างขึ้น การเริ่มเขียนโปรแกรมหลังจากคิดให้เป็นวัตถุแล้วจะเริ่มเขียนโปรแกรมโดยสร้างแม่แบบของวัตถุที่เรียกว่าคลาส (Class) ในแนวคิดเชิงวัตถุถือว่าเป็นแม่แบบของวัตถุ หลังจากนั้นต้องวางโครงโปรแกรมตามแนวการแก้ปัญหาต่าง และเขียนให้ได้ดังที่กำหนดวัตถุประสงค์ไว้

เอกสารประกอบการสอนเล่มนี้มุ่งเน้นให้ผู้ที่เริ่มศึกษาแนวการคิดเชิงวัตถุ การเขียนโปรแกรมเชิงวัตถุโดยใช้ภาษาจาวา ตามแนวทางของการเขียนโปรแกรมแบบเชิงวัตถุ โดยกำหนดให้คิดและออกแบบให้สิ่งต่างๆ เป็นวัตถุ การสร้างแม่แบบของวัตถุ การสร้างความสัมพันธ์ของคลาสแบบต่างๆ การสร้างโปรแกรมเพื่อให้เกิดความยืดหยุ่นตามแนวทางของพหุสัณฐาน การกำหนดการแก้ปัญหาโดยใช้ดีไซน์แพทเทิร์นและเฟรมเวิร์กเบื้องต้น

ผู้ที่ควรจะศึกษาจากหนังสือเล่มนี้เป็นอย่างยิ่งคือ นักเรียน นักศึกษา ที่เป็นผู้เริ่มศึกษาการเขียนโปรแกรม และ ครู อาจารย์ ที่ต้องการสอนการเขียนโปรแกรมเชิงวัตถุ เนื่องจากเนื้อหาในหนังสือ จะแสดงแนวคิดเชิงทฤษฎีให้กระชับพอเข้าใจ และเน้นตัวอย่างที่ตัวอย่างรหัสโปรแกรมและผลการรันโปรแกรม ที่สามารถศึกษาได้โดยง่าย

ท้ายนี้ผู้เขียนขอขอบคุณทุกท่านที่มีส่วนเกี่ยวข้องที่ทำให้เอกสารเล่มนี้สำเร็จลงด้วยดี และผู้เขียนหวังเป็นอย่างยิ่งว่า ผู้ที่ได้ศึกษาจากเอกสารประกอบการสอนเล่มนี้ จะมีทัศนคติที่ดีต่อการเขียนโปรแกรม จนพัฒนาตนเองให้เป็นโปรแกรมเมอร์ที่เก่ง สามารถคิดและวางแผนการเขียนโปรแกรมอย่างเป็นระบบ เพื่อพัฒนาประเทศไปสู่ความเป็นผู้นำทางด้านการพัฒนาซอฟต์แวร์ในอนาคต

เชาวลิต ขันคำ

9 พฤษภาคม 2554

สารบัญ

หน้า

บทที่ 1
โปรแกรมคอมพิวเตอร์

คอมพิวเตอร์(Computer) คือ อุปกรณ์อิเล็กทรอนิกส์ที่สามารถประมวลผลข้อมูลให้ได้โดยอัตโนมัติตามโปรแกรมที่มนุษย์ได้ป้อนเข้าไปเพื่อสั่งให้ทำงาน (สมโภชน์ ชื่นเอี่ยม, 2553, หน้า 11) องค์ประกอบที่ทำให้คอมพิวเตอร์ประมวลผลดังกล่าวได้นั้น ได้แก่ ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์(Software) ฮาร์ดแวร์ คือ ตัวเครื่องคอมพิวเตอร์และอุปกรณ์รายรอบ(Peripheral Devices)ซึ่งสามารถสัมผัสได้จับต้องได้ ส่วนซอฟต์แวร์ คือ โปรแกรมที่ใช้สั่งงานเครื่อง (วัชราภรณ์ สุริยาภิวัฒน์, 2553, หน้า 57) เรียกว่า โปรแกรมคอมพิวเตอร์ซึ่งไม่สามารถสัมผัสได้จับต้องได้ การพัฒนาซอฟต์แวร์ต้องอาศัยโปรแกรมภาษาคอมพิวเตอร์เป็นเครื่องมือ ในขณะที่รูปแบบการพัฒนาโปรแกรมเป็นส่วนสำคัญในการกำหนดคุณภาพและความรวดเร็วของโปรแกรม แนวทางที่นิยมใช้ในปัจจุบัน คือ การพัฒนาโปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งมีความสามารถสูงมากและการพัฒนาก็แตกต่างจากการเขียนโปรแกรมโครงสร้างแบบเดิม บทนี้นำเสนอความรู้เบื้องต้นเกี่ยวกับ ภาษาคอมพิวเตอร์ การพัฒนาโปรแกรมโดยใช้โปรแกรมภาษาคอมพิวเตอร์ยุคต่างๆ ขั้นตอนการพัฒนาโปรแกรม นอกจากนั้นยังเปรียบเทียบลักษณะการทำงานของโปรแกรมแบบโครงสร้างและโปรแกรมแบบเชิงวัตถุ รวมถึงให้ความรู้เกี่ยวเครื่องมือสำหรับออกแบบและเขียนโปรแกรมเชิงวัตถุ พอสังเขป

ภาษาคอมพิวเตอร์

โปรแกรมคอมพิวเตอร์แบ่งออกเป็น 2 ประเภท คือ โปรแกรมระบบและโปรแกรมประยุกต์ (สมโภชน์ ชื่นเอี่ยม, 2553, หน้า 187) โปรแกรมระบบทำหน้าที่ควบคุมการทำงานของคอมพิวเตอร์รวมถึงอุปกรณ์ โปรแกรมเหล่านี้ได้แก่ ระบบปฏิบัติการ (Operating system) โปรแกรมอรรถประโยชน์ (Utility program) โปรแกรมภาษาคอมพิวเตอร์ (Programming languages) เป็นต้น ส่วนโปรแกรมประยุกต์ คือ โปรแกรมที่เขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงานเฉพาะอย่าง (วัชราภรณ์ สุริยาภิวัฒน์, 2553, หน้า 143) เช่น โปรแกรมที่ใช้ในการจัดการกราฟิก โปรแกรมเกม โปรแกรมสำหรับสำนักงาน เป็นต้น ผู้ที่มีบทบาทสำคัญที่สุดต่อการเขียนโปรแกรม คือ โปรแกรมเมอร์ (Programmer) ซึ่งเป็นผู้เขียนชุดคำสั่งต่างๆ เพื่อทำให้คอมพิวเตอร์ทำงานได้

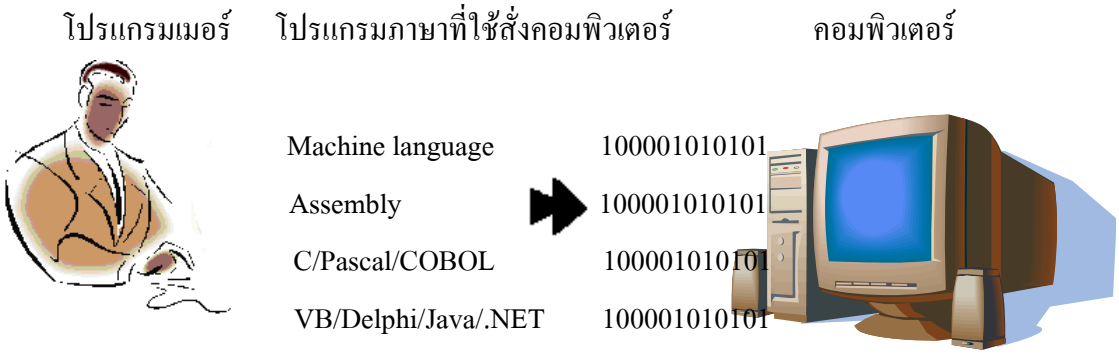
ตามที่ผู้ใช้ต้องการ ซึ่งผู้เขียนโปรแกรมต้องอาศัยโปรแกรมภาษาคอมพิวเตอร์ในการเขียนชุดคำสั่งเหล่านั้น

ความหมายของโปรแกรมภาษาคอมพิวเตอร์

ธีรวัฒน์ ประกอบผล (2545, หน้า 3) กล่าวว่า ภาษาคอมพิวเตอร์ คือ ภาษาที่ใช้สื่อสารกันระหว่างคอมพิวเตอร์กับมนุษย์ ส่วน เพ็ญศรี ปักกะสีนัง (2546, หน้า 97) กล่าวว่า ภาษาคอมพิวเตอร์เป็นการนำชุดคำสั่งแต่ละคำสั่งมาต่อกันให้คอมพิวเตอร์ทำงาน การเขียนชุดคำสั่งนี้ไม่ว่าจะเขียนด้วยภาษาอะไรจะเรียกว่าโปรแกรมต้นฉบับ (Source program) หรือ รหัสต้นฉบับ (Source code)

ขณะที่ พนิดา พานิชกุล (2554, หน้า 2) กล่าวว่า ภาษาการโปรแกรมคอมพิวเตอร์ (Computer Programming Language) เป็นภาษาที่โปรแกรมเมอร์ใช้ในการเขียนชุดคำสั่งหรือเขียนโปรแกรมคอมพิวเตอร์เพื่อสั่งให้คอมพิวเตอร์ทำงาน

ดังนั้น โปรแกรมภาษาคอมพิวเตอร์ คือ โปรแกรมที่ใช้เพื่อการพัฒนาโปรแกรมอื่นๆ แบ่งออกเป็นหลายระดับ เช่น ระดับภาษาระดับต่ำ ภาษาระดับสูง และภาษาธรรมชาติ เป็นต้น ผู้ใช้ภาษาคอมพิวเตอร์ คือ โปรแกรมเมอร์ ซึ่งเป็นผู้ออกแบบและเขียนโปรแกรมให้ผู้อื่นใช้งาน โดยโปรแกรมเมอร์จะเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ตามความถนัดและเหมาะสม หลังจากนั้นจะแปลคำสั่งของโปรแกรมที่เขียนขึ้นด้วยคอมไพเลอร์ (Compiler) หรืออินเทอร์-พรีเตอร์ (Interpreter) ให้เป็นคำสั่งที่ติดต่อกับคอมพิวเตอร์ได้ ซึ่งอยู่ในรูปแบบของภาษาเครื่องเพื่อควบคุมให้คอมพิวเตอร์ทำงานได้ตามที่ต้องการ แนวคิดการพัฒนาโปรแกรมแสดงดังภาพที่ 1.1



ภาพที่ 1.1 แสดงความสัมพันธ์ระหว่างโปรแกรมภาษากับการเขียนโปรแกรม

จากภาพที่ 1.1 อธิบายว่าได้เมื่อโปรแกรมเรียนรู้รูปแบบก็นำสิ่งที่ต้องการแก้ไขปัญหามาเขียนด้วยคำสั่งตามรูปแบบของภาษานั้นๆ หลังจากนั้นตัวแปลภาษาซึ่งเป็นโปรแกรมที่เขียนขึ้นมา

โดยบริษัทผู้ผลิตคอมพิวเตอร์หรือบริษัทผู้ผลิตซอฟต์แวร์ (พนิดา พานิชกุล, 2554, หน้า 2) มาแปลให้เป็นภาษาเครื่องโดยรูปแบบการแปลนั้นเป็นแบบอินเทอร์พรีเตอร์หรือคอมไพเลอร์ ซึ่งจะตรวจสอบคำสั่งแล้วแสดงผลหรือดำเนินการให้ได้ดังที่ผู้เขียนต้องการให้เป็น

ยุคของภาษาคอมพิวเตอร์

พนิดา พานิชกุล (2554, หน้า 3-5) ได้แบ่งยุคของภาษาคอมพิวเตอร์ไว้ 5 ยุค คือ ยุคที่หนึ่งเป็นภาษาระดับล่างหรือภาษาเลขฐานสองเท่านั้นสำหรับสั่งให้คอมพิวเตอร์ทำงาน ยุคที่สองเป็นยุคของภาษาสัญลักษณ์ที่เขียนคำสั่งให้เข้าใจง่าย และสิ้นลงด้วยภาษาอังกฤษ ภาษายุคนี้ได้แก่ภาษา Assembly ยุคที่สาม เป็นภาษาระดับสูงซึ่งใกล้เคียงภาษามนุษย์มากขึ้น ภาษายุคนี้ได้แก่ C, Pascal, COBOL เป็นต้น ภาษายุคที่สี่ เป็นภาษาระดับสูงแต่มีความสะดวกรวดเร็วในการพัฒนาโปรแกรมและมีส่วนการติดต่อที่สวยงาม ตัวอย่างของภาษายุคนี้ได้แก่ ภาษาจาวา ภาษาวิชวลเบสิค เป็นต้น ส่วนภาษายุคที่ห้า คือภาษาที่ใช้พัฒนาซอฟต์แวร์สำหรับระบบผู้เชี่ยวชายเน้นการใช้งานแบบปัญญาประดิษฐ์

ส่วน วัชราภรณ์ สุริยาภิวัฒน์ (2553, หน้า 149-153) แบ่งกลุ่มภาษาออกเป็นกลุ่มภาษาเชิงโพรซีเดอรัล (Procedural Programming Language) และโปรแกรมภาษาเชิงวัตถุ (OOP) โดยที่โปรแกรมเชิงโครงสร้างถูกแบ่งออกเป็นภาษาเครื่องจักร ภาษาสัญลักษณ์ และภาษาระดับสูง

วิวัฒนาการของภาษาคอมพิวเตอร์แบ่งออกตามยุคของการพัฒนาหรือแบ่งตามระดับของความใกล้เคียงกับภาษาของมนุษย์ ได้แก่ ภาษาเครื่อง ภาษาแอสเซมบลี ภาษาระดับสูง ภาษาระดับสูงมาก และภาษาธรรมชาติ สำหรับในเอกสารประกอบการสอนเล่มนี้จะแบ่งโดยเน้นด้านการพัฒนาโปรแกรมภาษาตามการเขียนโปรแกรมโดยแบ่งดังเป็นยุคที่ 1 ถึงยุคที่ 5 โดยเน้นยุคที่ 4 (Fourth Generation Language : 4GL) ที่มีบทบาทในปัจจุบันและยุคที่ 5 (Fifth Generation Language : 5GL) ซึ่งจะมีความสำคัญมากในอนาคต และคาดว่าโปรแกรมคอมพิวเตอร์ในยุคที่ N ซึ่งการเขียนโปรแกรมอาจมิใช่เพียงโปรแกรมเมอร์เท่านั้น แต่มนุษย์ทุกคนจำเป็นต้องเขียนโปรแกรมด้วยภาษาธรรมชาติอย่างสมบูรณ์แบบและการเขียนโปรแกรมสามารถทำได้อย่างง่ายดายและฝังเอาไว้กับอุปกรณ์และทุกสิ่งทุกอย่างในโลกนี้ จากสิ่งที่กล่าวมาข้างต้นสามารถแบ่งยุคต่างๆ ของคอมพิวเตอร์ออกได้ 5 ยุคดังนี้

ยุคที่ 1: ภาษาเครื่อง

การพัฒนาโปรแกรมคอมพิวเตอร์ในยุคแรกใช้ภาษาเครื่อง หรือรหัสเครื่อง (Machine language or Machine code) การเขียนโปรแกรมต้องป้อนชุดคำสั่งโดยใช้รหัสเลขฐานสอง คือ 0 กับ 1 ให้คอมพิวเตอร์โดยตรง ทำให้การเขียนโปรแกรมเป็นภาระหนักของโปรแกรมเมอร์มาก เนื่องจากไม่มีเครื่องมือที่ช่วยในการพัฒนา

ข้อดีของภาษาเครื่อง คือ เครื่องคอมพิวเตอร์ไม่ต้องแปลรหัสต้นฉบับ คอมพิวเตอร์สามารถนำไปใช้งานได้ทันที ทำให้การทำงานของโปรแกรมเร็วมาก แต่ก็มีข้อเสียคือ การเขียนโปรแกรมทำได้ยาก เพราะต้องจดจำคำสั่งและข้อมูลต่างๆ เป็นรหัสเลขฐานสอง จึงเป็นแรงผลักดันให้สร้างโปรแกรมภาษาในระดับที่สูงขึ้น

ยุคที่ 2: ภาษาแอสเซมบลี

ภาษานี้อยู่ในยุคที่ 2 เริ่มเมื่อประมาณต้นของช่วง ค.ศ. 1950 ยุคนี้มีแนวคิดที่จะทำให้ภาษาคอมพิวเตอร์มีความใกล้เคียงภาษามนุษย์มากขึ้น โดยนำเอาภาษาอังกฤษมาสร้างเป็นคำสั่งภาษาแอสเซมบลี (Assembly language) มีคำสั่งเป็นคำภาษาอังกฤษง่าย ๆ เช่น ADD, SUB, MUL, DIV แทนการบวก ลบ คูณ และหาร เป็นต้น การเขียนโปรแกรมในยุคนี้ตัวแปลภาษาจะแปลจากภาษาอังกฤษมาเป็นภาษาเครื่องโดยใช้ตัวแปลภาษาที่เรียกว่า แอสเซมเบลอร์ (Assembler) โปรแกรมเมอร์ต้องมีความรู้ทั้งรูปแบบการใช้คำสั่งและระบบของฮาร์ดแวร์ของคอมพิวเตอร์เป็นอย่างดี จึงจะสามารถเขียนโปรแกรมได้ดี เครื่องมือที่ใช้ในการพัฒนาโปรแกรมภาษาในยุคนี้มีให้เลือกมากขึ้น เช่นโปรแกรม Turbo Assembly และ Macro Assembly เป็นต้น

ข้อดี คือ สามารถพัฒนาโปรแกรมได้ง่ายขึ้น โปรแกรมทำงานเร็ว แต่ข้อเสีย คือ โปรแกรมเมอร์ต้องเรียนรู้สถาปัตยกรรมของคอมพิวเตอร์แต่ละยี่ห้อ แต่ละรุ่น เพื่อจะควบคุมการเขียนโปรแกรมให้ควบคุมฮาร์ดแวร์ให้ได้ ซึ่งคอมพิวเตอร์ต่างก็มีสถาปัตยกรรมที่แตกต่างกันไป ดังนั้นแทนที่โปรแกรมเมอร์จะทุ่มเทให้การเขียนโปรแกรมอย่างเดียว แต่กลับต้องศึกษาสถาปัตยกรรมคอมพิวเตอร์อีก จึงจำเป็นต้องพัฒนาโปรแกรมภาษาให้ง่ายต่อการเขียนโปรแกรมให้มากขึ้น

ยุคที่ 3: ภาษาระดับสูง

ภาษากลุ่มนี้เรียกว่าภาษายุคที่ 3 เริ่มเมื่อประมาณกลางช่วง ค.ศ. 1950 ยุคนี้ถือว่าเป็นยุคแรกของภาษาระดับสูง (High level language) เพราะยุคที่ 1 และ 2 นั้นถือว่าเป็นยุคของภาษาระดับต่ำ (Low level language) ที่ยังไม่เหมือนภาษามนุษย์เท่าใดนักแต่ยุคนี้ภาษาคอมพิวเตอร์จะใกล้เคียงกับภาษาของมนุษย์มากขึ้นอีก โดยใช้ประโยคหรือวลีในภาษาอังกฤษเพื่อเขียนโปรแกรม ข้อดีของภาษาในยุคนี้คือ

1) การเขียนโปรแกรมใช้เวลาน้อยและต้นทุนต่ำลง

2) โปรแกรมเมอร์ไม่จำเป็นต้องเรียนรู้สถาปัตยกรรมของคอมพิวเตอร์ ทำให้การเขียนโปรแกรมทำได้กว้างขวางมากขึ้น

แนวการเขียนโปรแกรมมีชื่อเรียกหลากหลาย เช่นโปรแกรมแบบโครงสร้าง (Structure program) โปรแกรมแบบโมดูลหรือแบบโพรซีเดอรอล (Modular program or procedural program)

มีหลายภาษาที่เกิดขึ้นในยุคนี้ คำสั่งต่าง ๆ จะถูกกำหนดโดยไวยากรณ์ของแต่ละภาษา โปรแกรมเมอร์ต้องเขียนรหัสต้นฉบับของคำสั่งทั้งหมดด้วยตนเอง

เครื่องมือช่วยพัฒนาโปรแกรม ส่วนมากจะเป็นอิดิเตอร์ (Editor) ที่ติดมากับตัวภาษาแต่ละภาษา โปรแกรมเมอร์จะเขียนรหัสต้นฉบับทั้งหมด จากนั้นใช้ตัวแปลภาษาซึ่งอาจเป็น คอมไพเลอร์ หรืออินเตอร์พรีเตอร์แปลรหัสต้นฉบับให้เป็นภาษาเครื่องสั่งให้คอมพิวเตอร์ทำงาน ภาษาคอมพิวเตอร์ยุคนี้มีหลากหลายเหมาะสมกับงานที่แตกต่างกัน เช่น ภาษา FORTRAN เหมาะกับงานคำนวณทางด้านวิทยาศาตร์ ภาษา C , Pascal , BASIC เหมาะกับงานทั่วไป ส่วนภาษา COBOL เหมาะกับงานทางด้านธุรกิจ เป็นต้น

แม้ว่าภาษายุคนี้จะใกล้เคียงกับภาษามนุษย์มากก็ตาม แต่ปัจจัยทางด้านธุรกิจและความต้องการให้โปรแกรมทำงานยืดหยุ่นรองรับการทำงานได้กว้างขวางและมีความเข้ากันได้กับโปรแกรมอื่นๆ ได้ดีมากขึ้น จำเป็นต้องพัฒนาโปรแกรมภาษาให้ง่ายขึ้นไปอีก ยุคที่ 4 จึงเกิดขึ้นในเวลาต่อมา

ยุคที่ 4: ภาษาระดับสูงมาก

ยุคที่ 4 เป็นโปรแกรมภาษาแนวใหม่ที่เพื่อให้สนองความต้องการด้านความรวดเร็ว ยืดหยุ่นและนำกลับมาใช้ใหม่ได้ ต้นยุคเริ่มเมื่อประมาณช่วง ค.ศ. 1970 ยุคนี้อาจกล่าวได้ว่าเป็นยุค (นอน-โปรซีเดอรอล) ที่เน้นให้ได้ผลลัพธ์ตามความต้องการอย่างรวดเร็วมากกว่าที่จะเน้นว่าจะต้องเขียนอย่างไร ในยุคก่อนหน้านี้โปรแกรมเมอร์ต้องรู้คำสั่งและวิธีการเขียนทุกสิ่งทุกอย่างด้วยตนเอง แต่ยุคนี้มีเครื่องมือช่วยพัฒนาได้รวดเร็วขึ้นกว่าเดิมมาก หากเปรียบการเขียนโปรแกรมเป็นการประกอบรถยนต์ 1 คัน ในยุคที่ผ่านมาผู้ประกอบต้องสร้างชิ้นส่วนทุกอย่างด้วยตนเองก่อนจึงประกอบเป็นรถยนต์ได้ แต่ยุคนี้การประกอบรถยนต์ทำได้ในเวลาอันรวดเร็ว เนื่องจากมีผู้ผลิตชิ้นส่วนทั้งหมดไว้แล้ว ผู้ประกอบ (โปรแกรมเมอร์) จึงเพียงแต่เลือกมาประกอบให้เหมาะสมเท่านั้น

การเขียนโปรแกรมยุคนี้เรียกว่าการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming :OOP) ซึ่งมองการเขียนโปรแกรมเหมือนการสร้างวัตถุเป็นก้อนๆ หรือ เป็นชิ้นๆ มีเครื่องมือช่วยเขียนโปรแกรม เรียกว่า ตัวสร้าง (Generators) ทำหน้าที่ติดต่อกับโปรแกรมเมอร์แบบกราฟิกอินเตอร์เฟส (Graphic interface) บางทีอาจเรียกว่าเป็นการเขียนโปรแกรมแบบวิสชวล (Visual programming) โปรแกรมเมอร์ใช้วิธีแดร็กแอนด์ดร็อพ (Drag and Drop) คือ การคลิกเอาวัตถุที่ใช้เป็นส่วนประกอบต่าง ๆ ของโปรแกรม เช่น ปุ่ม ฟอร์ม ไอคอน มาวางในพื้นที่เขียนโปรแกรม หลังจากนั้นกำหนดคุณสมบัติและหน้าที่การทำงานที่ควบคุมด้วยเหตุการณ์ที่เข้ามากระทำต่อวัตถุที่สร้างขึ้น หลังจากนั้นตัวสร้างจะแปลงให้เป็นรหัสต้นฉบับแล้วแปลด้วยคอมไพเลอร์หรืออินเตอร์

พรีเตอร์ที่ติดมาด้วยเสมอ ตัวอย่างภาษายุคนี้ เช่น Visual Basic, Visual C++, Borland Delphi, Borland C++ Builder, JBuilder,VisualStudio.NET, Java เป็นต้น สำหรับเอกสารเล่มนี้ได้ ใช้โปรแกรมภาษาจาวา ซึ่งออกแบบและเขียนโปรแกรมตามแนวทางนี้ และการนำเสนอเนื้อหาตลอดเล่มเน้นให้เรียนรู้ในแนวคิดของการเขียนโปรแกรมเชิงวัตถุ ซึ่งเป็นการออกแบบเชิงแนวคิดและนำไปประยุกต์ใช้ โดยเครื่องมือช่วยพัฒนาคือโปรแกรมเน็ตบีน สำหรับการสร้างส่วน แดร็กแอนดร็อพในส่วนการประยุกต์ใช้งาน

ข้อดีของโปรแกรมภาษาในยุคนี้คือ การเขียนโปรแกรมทำได้รวดเร็วมาก ส่วนการติดต่อกับผู้ใช้และโปรแกรมเมอร์สวยงามน่าใช้มากขึ้น ใช้งานง่ายกว่าภาษายุคที่ 3 มาก แต่ข้อเสียคือโปรแกรมเมอร์ไม่สามารถรู้รายละเอียดบางอย่างที่ซ่อนเอาไว้โดยเครื่องมือช่วยพัฒนาเหล่านั้น โปรแกรมจะมีขนาดใหญ่มากขึ้น ฮาร์ดแวร์ที่ใช้จะต้องมีความเร็วสูง

ยุคที่ 5: ภาษาธรรมชาติ

ปัจจุบันเริ่มเข้าสู่ต้นยุคของภาษาธรรมชาติ ซึ่งถือว่าเป็นยุคที่ 5 ยุคนี้เน้นการนำปัญญาประดิษฐ์ (Artificial Intelligent :AI) มาใช้เป็นแนวคิดหลักของการเขียนโปรแกรม โดยเริ่มใช้ภาษาธรรมชาติของมนุษย์ (Natural language) ในแนวการเขียนโปรแกรมแบบตรรกะและโมเดลภายใต้สภาวะแวดล้อมของโลกจริง ๆ สนับสนุนและยืดหยุ่นต่อพฤติกรรมที่เปลี่ยนไปภายใต้การจัดการของคอมพิวเตอร์ ปัจจุบันเริ่มมีหุ่นยนต์และซอฟต์แวร์ที่สามารถโต้ตอบกับมนุษย์ได้ ตัวอย่างภาษาช่วงเริ่มต้นของยุคนี้เช่น ภาษาโปรลอก 2 (Prolog 2) เป็นโปรแกรมที่สามารถสอบถามข้อมูลด้วยประโยคที่ใช้สื่อสารกันตามปกติ ระบบผู้เชี่ยวชาญ (Expert System) ที่สามารถวินิจฉัยในเรื่องที่เชี่ยวชาญได้ ระบบฐานความรู้ (Knowledge Base System) ที่เก็บรวบรวมและประมวลผลข้อมูลอย่างชาญฉลาด เมื่อถึงยุคปลายของยุคนี้ทั้งฮาร์ดแวร์และซอฟต์แวร์จะเป็นยุคเอไอ (AI) อย่างสมบูรณ์ การเขียนโปรแกรมจะใช้ภาษาธรรมชาติ มนุษย์สามารถเขียนโปรแกรมคอมพิวเตอร์โดยภาษาที่เราใช้สื่อสารกัน

ข้อดีของภาษายุคนี้คือ การเขียนโปรแกรมทำได้โดยไม่ต้องเรียนรู้ภาษาคอมพิวเตอร์ให้ยุ่งยาก แต่ข้อเสียคือโปรแกรมเมอร์อาจถูกจำกัดความสามารถ เนื่องจากมีเครื่องมือช่วยในการเขียนโปรแกรมที่ฉลาดมากเกินไป ทำให้ไม่สามารถจัดการบางสิ่งบางอย่างตามที่ต้องการได้ กล่าวคือในระบบต่าง ๆ เหล่านั้นจะต้องทำการแปลคำสั่งเช่นเดียวกับยุคที่ผ่าน ๆ มา แต่ผู้เขียนโปรแกรมไม่รู้สึกว่าต้องทำเองเหมือนที่ผ่านมาเพราะระบบเหล่านั้นมีความฉลาดเปรียบเสมือนรวมการแปลและการสั่งการจากภาษาธรรมชาติแล้วแปลจนกระทั่งสามารถเป็นภาษาเครื่องได้

ในยุคต่อจากนี้ถือว่าเป็นยุคที่ N ที่ จะเป็นการนำเอาภาษาธรรมชาติมาเขียนสั่งให้คอมพิวเตอร์ที่ไม่ใช่เป็นเพียงอุปกรณ์ที่เราเห็นปัจจุบันเท่านั้นแต่จะเป็นทั้งหุ่นยนต์ อุปกรณ์ต่างๆ

ในชีวิตประจำวัน รวมไปถึงวิศวพันธุกรรม และการเขียนโปรแกรมสามารถกลมกลืนกับชีวิตประจำได้อย่างง่ายดาย

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

ธีรวัฒน์ ประกอบผล (2553, หน้า 11-14) และ วัชราภรณ์ สุริยาภิวัฒน์ (2553, หน้า 195) ต่างกล่าวตรงกันว่าขั้นตอนการเขียนโปรแกรมมีดังนี้

(1) การกำหนดและวิเคราะห์ปัญหา (Problem definition and problem analysis) ด้วยการกำหนดขอบเขตของปัญหา

(2) การเขียนผังงาน (Flow chart) และซูโดโค้ด (pseudo coding) เป็นการแทนขั้นตอนต่างๆ ด้วยสัญลักษณ์

(3) การเขียนโปรแกรม(Programming) ด้วยการเขียนโปรแกรมตามรูปแบบของไวยากรณ์ตามรูปแบบของภาษาที่เลือกใช้

(4) การทดสอบและแก้ไขโปรแกรม(Program testing and debugging) เป็นการทดสอบว่าโปรแกรมที่เขียนขึ้นมานั้นถูกต้องตามรูปแบบที่เขียนหรือไม่ และทดสอบว่าโปรแกรมที่เขียนขึ้นเป็นไปตามที่วิเคราะห์ไว้หรือไม่

(5) การทำเอกสารและการบำรุงรักษาเป็นขั้นตอนของการทำเอกสารคู่มือ ทั้งคู่มือการใช้งานและคู่มือสำหรับโปรแกรมเมอร์

ขณะที่ กระทรวงศึกษาธิการ ได้กล่าวไว้ใน หลักการเขียนโปรแกรม คพ. ทป. 014 (2533, หน้า 46) ระบุขั้นตอนการเขียนโปรแกรมดังนี้

(1) การวิเคราะห์ปัญหา

(2) การออกแบบขั้นตอนการทำงานของโปรแกรม

(3) การลงรหัสโปรแกรม

(4) การทดสอบและแก้ไขโปรแกรม

(5) การทำเอกสารประกอบโปรแกรม

พานิชกุล (2554, หน้า 9-10) ระบุว่าขั้นตอนการพัฒนาโปรแกรมคอมพิวเตอร์มีดังนี้

(1) การวิเคราะห์ปัญหา

(2) การออกโปรแกรม กำหนดส่วนนำเข้า ประมวลผล และส่วนการแสดงผล

(3) เขียนโปรแกรมโดยใช้ภาษาคอมพิวเตอร์

(4) การทดสอบและแก้ไขโปรแกรมเพื่อความถูกต้องก่อนนำไปใช้งานจริง

(5) การทำเอกสารและดูแลรักษาโปรแกรม

จากที่กล่าวมาข้างต้น ดังนั้นขั้นตอนของการเขียนโปรแกรมมีดังนี้

1. ศึกษาปัญหาและทำความเข้าใจสิ่งที่ต้องการของผู้ใช้

ขั้นตอนนี้เป็นขั้นตอนแรกสุด เพราะหากมองปัญหาออกก็ไม่สามารถที่จะเขียนโปรแกรมหรือสร้างงานออกมาได้อย่างถูกต้องตามความต้องการ ในขั้นตอนนี้สิ่งที่ต้องพิจารณาและทำความเข้าใจมีดังนี้

1.1 สิ่งที่โจทย์หรือผู้ใช้ต้องการ (Output) ของงาน ผู้เขียนจะต้องรู้อย่างชัดแจ้งว่าผู้ใช้ต้องการผลการทำงานอย่างไร แนวปฏิบัติของขั้นตอนนี้คือศึกษาจากโจทย์ที่ได้รับ หรือสอบถามรวบรวมข้อมูลจากผู้ปฏิบัติงานของผู้ใช้งานจริงให้ละเอียดที่สุด

1.2 สิ่งที่โจทย์ให้มาหรือปัจจัยต่าง ๆ ที่ทำให้เกิดผลลัพธ์ ซึ่งก็คือข้อมูลนำเข้า (Input) ดังนั้นผู้เขียนโปรแกรมจะต้องพิจารณาประเด็นนี้เป็นสิ่งสำคัญอันดับที่สอง โดยดูจากโจทย์กำหนดให้ว่ามีปัจจัยใดประกอบอยู่แล้วบ้าง และสิ่งใดจะต้องกำหนดหรือหาเพิ่มเติมเพื่อที่จะช่วยในการแก้ไขปัญหาดังกล่าวให้ได้ผลลัพธ์ตามต้องการ

1.3 วิธีการดำเนินงาน (Process) คือ ลำดับขั้นตอนการทำงาน เป็นกระบวนการเพื่อนำข้อมูลนำเข้ามาดำเนินการเพื่อให้ได้ผลลัพธ์

2. การออกแบบและกำหนดขั้นตอนการทำงานของโปรแกรม

การกำหนดวิธีการทำงานนั้นสำหรับผู้ที่เพิ่งจะเริ่มเขียนโปรแกรมคอมพิวเตอร์นั้นจำเป็นอย่างยิ่งต้องมีแนวคิดเหมือนการแก้โจทย์หรือการแก้ปัญหาทางคณิตศาสตร์ กล่าวคือ ต้องคิดเป็นขั้นตอนและกำหนดลำดับการแก้ปัญหาให้ชัดเจนดังนี้คือ

2.1 ให้เขียนลำดับขั้นตอนการแก้ปัญหาเป็นขั้นตอนให้ชัดเจนที่สุดเท่าที่สามารถทำได้

2.2 แทนลำดับขั้นตอนการแก้ปัญหาดังกล่าวด้วยแผนภาพหรือผังงาน (Flowchart) คลาสไดอะแกรม (Class Diagram)

2.3 ลงรหัสโปรแกรม (Pseudo Code) หรือรหัสเทียม ของโปรแกรม โดยรหัสเทียมดังกล่าวจะต้องดูจากผังการทำงานเป็นหลัก การลงรหัสเทียมนี้จะเขียนไว้เป็นกลาง ๆ อาจจะอิงโปรแกรมภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่งก็ได้หรือเขียนเป็นภาษาอังกฤษล้วน ๆ ก็ได้ หรือหากเป็นภาษาไทยก็ยังอนุโลมให้ใช้ได้เช่นกัน

3. ลงมือเขียนโปรแกรม

ในการเริ่มลงมือเขียนโปรแกรมคอมพิวเตอร์นั้น ผู้เขียนจะต้องดำเนินการดังต่อไปนี้

3.1 ศึกษารูปแบบไวยากรณ์ (Syntax) ของโปรแกรมภาษานั้น ๆ ซึ่งจะเป็นกฎเกณฑ์ข้อบังคับ หรือรูปแบบบังคับต่าง ๆ ของโปรแกรมภาษาที่ใช้ในการเขียน โดยโปรแกรมเมอร์ต้องศึกษาให้เข้าใจและแม่นยำ จึงจะสามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ

3.2 ศึกษา คำสั่งปฏิบัติการ ตัวดำเนินการ ตัวแปร หรือฟังก์ชัน (Functions) ต่างๆ ที่มีในภาษาที่เลือก ซึ่งในขั้นตอนนี้ผู้ใช้จะต้องดำเนินการเขียนโปรแกรมจริง ๆ จึงจะสามารถเข้าใจและใช้งานได้อย่างถูกต้อง

4. ทดสอบโปรแกรม

เมื่อเขียนโปรแกรมแล้วการทดสอบการทำงานของโปรแกรมเพื่อตรวจสอบความถูกต้องของโปรแกรมมีขั้นตอนดังนี้

4.1 แปลรหัสต้นฉบับของโปรแกรมภาษาที่เลือกมาใช้เขียนอาจมีความแตกต่างกันและใช้คำสั่งที่ไม่เหมือนกัน แต่โดยทั่วไปจะขึ้นอยู่กับตัวแปลภาษาซึ่งถ้าเป็นคอมไพเลอร์ จะทำการแปลรหัสดังกล่าวเป็นภาษาเครื่องและอาจบันทึกเอาไว้ เพื่อให้สามารถเรียกใช้งานได้ในภายหลังโดยไม่ต้องทำการแปลคำสั่งใหม่ แต่ถ้าเป็นตัวแปลภาษาที่เป็นอินเตอร์พรีเตอร์จะต้องแปลทุกครั้งเมื่อต้องการรันโปรแกรมเพื่อใช้งาน

4.2 ทดสอบการรันของโปรแกรม หรือการเรียกใช้งานโปรแกรมแต่ละโปรแกรมภาษาจะมีการเรียกใช้ต่างกันบ้างแต่โดยทั่วไปจะทำได้หลังจากที่คอมไพล์แล้ว และไม่มีข้อผิดพลาดสามารถเรียกใช้งานตามที่ต้องการผ่านระบบปฏิบัติการที่นำโปรแกรมไปติดตั้ง

5. จัดทำเอกสารและการบำรุงรักษา

5.2 การจัดทำเอกสารที่เกี่ยวข้อง เช่นคู่ผู้ใช้ คู่มือการพัฒนาโปรแกรมและออกแบบระบบ และคู่มือการแก้ปัญหาสำหรับผู้ดูแลระบบหรือโปรแกรมเมอร์

5.2 บำรุงรักษาโปรแกรมด้วยการตรวจสอบความถูกต้องปรับปรุงให้ทำงานเร็วขึ้น หรือการปรับปรุงดัชนีของฐานข้อมูล หรือปรับปรุงรุ่นของซอฟต์แวร์ เป็นต้น

ความหมายและแนวคิดการเขียนโปรแกรมเชิงวัตถุ

ไซมอล เคนดัล (Simol Kendal, 2009, p.13) ได้กล่าวว่า ในช่วงปี ค.ศ. 1970 โปรแกรมแบบโครงสร้างได้เป็นที่นิยมและยอมรับว่าเป็นมาตรฐานการเขียนโปรแกรมคอมพิวเตอร์ที่มีขนาดใหญ่และซับซ้อน ซึ่งหลักการของโปรแกรมแบบนี้มุ่งเน้นไปที่กระบวนการเชิงตรรกะของโครงสร้างโปรแกรมจะถูกพัฒนาเป็นส่วนส่วนๆ เพื่อสะดวกต่อการทำความเข้าใจและบริหารจัดการคอมโพเนนต์ อย่างไรก็ตามปัญหาที่สำคัญของการเขียนโปรแกรมก็คือ การทำความเข้าใจเกี่ยวกับระบบที่ต้องการจะสร้างและการเปลี่ยนแปลงซอฟต์แวร์ที่มีอยู่ตามความต้องการของผู้ใช้ที่

เปลี่ยนไป จนกระทั่งในปี ค.ศ. 1990 หลักการเขียนโปรแกรมแบบเชิงวัตถุและการเขียนบนพื้นฐานของคอมโพเนนต์ได้ถูกสร้างขึ้นและภาษาเชิงวัตถุได้กลายมาเป็นสิ่งเป็นการเขียนโปรแกรมกันโดยทั่วไปในปี ค.ศ. 2000 รายละเอียดต่อไปนี้จะนำเสนอความหมายของการเขียนโปรแกรมเชิงวัตถุที่มีผู้ให้ความหมายเอาไว้ นอกจากนั้นยังนำเสนอแนวคิดเกี่ยวกับแนวคิดที่แตกต่างกันระหว่างแนวคิดการเขียนโปรแกรมเชิงโครงสร้างและเชิงวัตถุ

ความหมายของการเขียนโปรแกรมเชิงวัตถุ

ไซมอล เคนดัล (Simol Kendal, 2009, p.13) กล่าวว่า หลักการเขียนโปรแกรมเชิงวัตถุ คือ หลักการที่อยู่บนพื้นฐานการพัฒนาหลายๆ แนวคิดในช่วง 30 ปีที่ผ่านมา โดยแนวคิดดังกล่าวได้แก่ แนวคิดเชิงนามธรรม (Abstraction) การห่อหุ้ม (Encapsulation) การสืบทอด (Generalization) และพหุสัณฐาน(Polymorphism) โดยการอนุญาตให้สามารถนำข้อมูลและตัวดำเนินการรวมเข้าด้วยกัน

วาย. เดเนียล เหลียง (Y. Deniel Liang, 2007, p. 215) กล่าวว่าหลักการเขียนโปรแกรมเชิงวัตถุ เกี่ยวข้องกับการเขียนโปรแกรมที่มีการใช้วัตถุ โดยใช้วัตถุแทนสิ่งต่างๆ ที่มีอยู่จริงในโลก ซึ่งสามารถเป็นจุดหมายปลายทางของการถูกกำหนดให้วัตถุมีความเป็นปัจจเจกทางตัวตน สถานะและพฤติกรรม

วีกีพีเดีย (Wikipedia, 2007) กล่าวว่า การเขียนโปรแกรมเชิงวัตถุ คือ หลักการของการเขียนโปรแกรมซึ่งใช้วัตถุ และการกระทำต่างๆ ของวัตถุ ในการออกแบบแอพพลิเคชัน และโปรแกรมคอมพิวเตอร์ ซึ่งใช้เทคนิคหลายประการ เช่น การสืบทอด การนำความสามารถต่างๆ ย่อยๆ มาประกอบกัน การพ้องรูป และการห่อหุ้ม ซึ่งใช้ในการออกแบบ และพัฒนาแอพพลิเคชันมาตั้งแต่ต้น ค.ศ. 1990 โปรแกรมภาษายุคใหม่สนับสนุนการเขียนโปรแกรมแบบนี้แล้ว

การเขียนโปรแกรมเชิงวัตถุหรือเรียกว่าการเขียนโปรแกรมแบบโอโอพี คือ การเขียนโปรแกรมการโดยแทนสิ่งต่างๆ ที่มีอยู่ในโลกด้วยการมองเป็นวัตถุหนึ่งชนิด ซึ่งวัตถุดังกล่าวประกอบด้วยคุณสมบัติหรือแอตทริบิวต์ (Properties or Attribute) และหน้าที่หรือฟังก์ชันงานหรือเมธอด (Operation or Method) แนวคิดการเขียนโปรแกรมแบบนี้จะเน้นความยืดหยุ่นในเรื่องการรับทอดคุณสมบัติ (Inheritance) ความปลอดภัยของข้อมูล (Encapsulation) การมีพหุสัณฐาน (Polymorphism) เป็นต้น

ข้อแตกต่างระหว่างการเขียนโปรแกรมเชิงโครงสร้างกับเชิงวัตถุ

การเขียนโปรแกรมคอมพิวเตอร์ของโปรแกรมเมอร์ต้องเลือกโปรแกรมภาษาก่อนเป็นลำดับแรก หลังจากนั้นต้องศึกษารูปแบบการใช้งานของโปรแกรมภาษา แต่ละภาษาจะมีรูปแบบการเขียนโปรแกรมที่แตกต่างกันในแต่ละคำสั่ง แต่อย่างไรก็ตามรูปแบบการเขียนจะเป็นไปในแนวเดียวกันที่พอจำแนกได้ คือ การเขียนโปรแกรมภาษาเครื่อง การเขียนโปรแกรมแบบภาษา

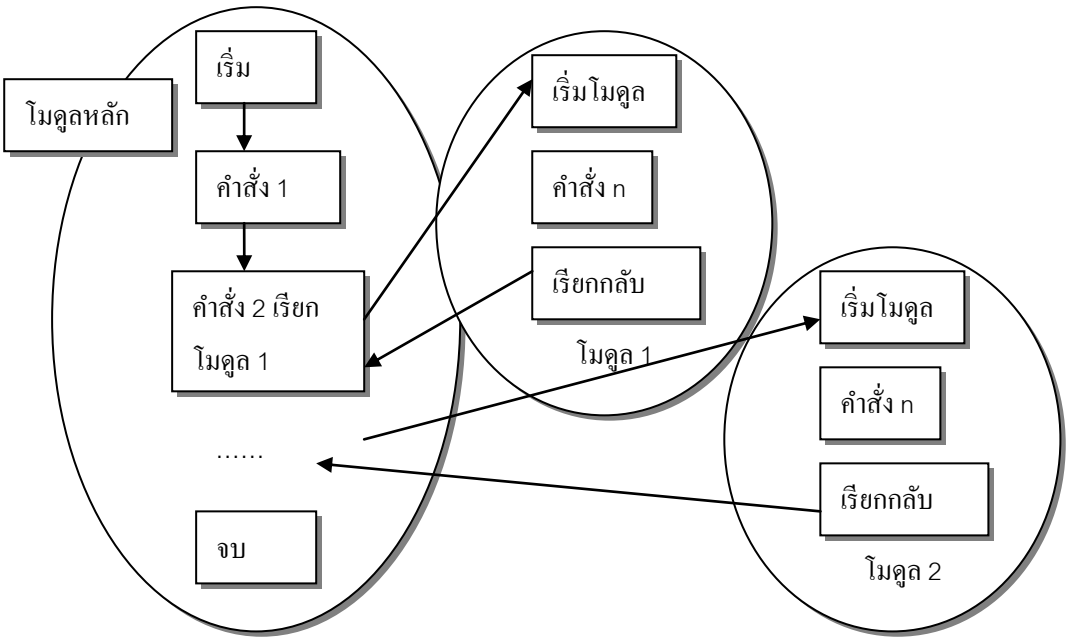
โครงสร้าง การเขียนโปรแกรมแบบภาษาเชิงวัตถุ และการเขียนโปรแกรมด้วยภาษาธรรมชาติ ในเอกสารประกอบการสอนนี้ใช้การเขียนโปรแกรมแบบเชิงวัตถุ ดังนั้นต่อไปนี้จะนำเสนอแนวคิดเกี่ยวกับการเขียนโปรแกรมด้วยภาษาทั้งเชิงโครงสร้างและเชิงวัตถุ เพื่อเปรียบเทียบให้เห็นข้อแตกต่าง ระเบียบวิธี และแนวคิดของของการเขียนโปรแกรมทั้งสองแบบนี้

วาย. เดเนียล เหลียง (Y. Danial Liang, 2007, p. 214) กล่าวว่า การเขียนโปรแกรมลักษณะโพรซีเดอรัลเช่น ภาษาซี ปาสคาล เบสิค เอดา และโคบอล เกี่ยวกับการเลือกโครงสร้างข้อมูลและการออกแบบอัลกอริทึมและการแปลงอัลกอริทึมไปสู่โค้ด ส่วนโปรแกรมเชิงวัตถุ เช่น จาวา สมอลทอล์ค รวมความสามารถของโปรแกรมแบบโพรซีเดอรัลด้วยการเพิ่มมิติซึ่งเตรียมไว้สำหรับความซับซ้อนที่มากขึ้น โมดูล ความชัดเจนเรื่องการออกแบบ การนำกลับมาใช้ใหม่ โดยผ่านแนวคิดเชิงนามธรรม การห่อหุ้ม การรับทอด และพาวะพหุสัณฐาน

พนิดา พานิชกุล (2548, หน้า 4) กล่าวว่า ในอดีตการเขียนโปรแกรมแบบโพรซีเดอรัลนั้น ภายในโครงสร้างของโปรแกรมจะแบ่งออกเป็น 2 ส่วนได้แก่ โปรแกรมหลัก (Main program) ที่มีข้อมูล (Data) เป็นส่วนประกอบและส่วนโปรแกรมย่อย (Procedure หรือ Function) โดยข้อมูลที่ประกาศใช้อยู่ภายในโปรแกรมหลักนั้นจะถูกเรียกใช้โดยโปรแกรมย่อยต่างๆ ที่อยู่ภายในโปรแกรม ลักษณะของข้อมูลที่ประกาศทั่วไปทั้งในโปรแกรมนั้น เรียกว่าการประกาศใช้แบบใช้งานได้ทั่วทั้งโปรแกรม (Global) แต่สำหรับการเขียนโปรแกรมเชิงวัตถุแล้วข้อมูลจะถูกประกาศใช้โดยเฉพาะภายในแต่ละวัตถุเท่านั้น

ไซมอล เคนดัล (2009, p. 13-15) กล่าวว่า หลักการเขียนโปรแกรมเชิงโครงสร้างนั้นจะมุ่งเน้นให้บล็อกขนาดย่อยของโปรแกรมที่เขียนขึ้นมาแล้วสามารถเปลี่ยนแปลง ซึ่งทำให้ง่ายต่อการเข้าใจและการเปลี่ยนแปลง แต่โปรแกรมเชิงวัตถุนั้นจะสร้างโมเดลคำสั่งในโปรแกรมคอมพิวเตอร์ด้วยข้อมูล ซึ่งคำสั่งและข้อมูลควรผูกติดไปด้วยกัน และอีกประการคือคอมโพเนนต์ของซอฟต์แวร์ต้องนำกลับมาใช้ใหม่ได้ ในขณะเดียวกันการสร้างหลักการนี้ยังขยายแนวคิดของโปรแกรมเชิงโครงสร้างที่ใช้กันมาตั้งแต่ปี ค.ศ. 1970 อีกด้วย

ดังนั้น การพัฒนาโปรแกรมด้วยภาษาเชิงโครงสร้างถือเป็นยุคที่สามของโปรแกรมภาษาคอมพิวเตอร์ และได้รับความนิยมเป็นอย่างมาก ในปัจจุบันเองก็ยังนิยมใช้กันอย่างแพร่หลายอยู่ในทุกๆ วงการ ลักษณะของการพัฒนาโปรแกรมก็คือ จะพัฒนาเป็นโมดูลย่อยๆ และให้โปรแกรมหลักได้เรียกใช้งาน โดยการเรียกใช้งานจากโปรแกรมหลักนั้น จะทำงานจากบนลงล่างคือเริ่มต้นไปจนสิ้นสุด ในระหว่างที่ทำงานจากบนลงล่างโดยทั่วไปจะมีการเรียกใช้โมดูลย่อย ซึ่งโปรแกรมจะไปทำคำสั่งโมดูลนั้นจนหมดแล้วจึงกลับมาดำเนินการตามโปรแกรมหลักต่อไป

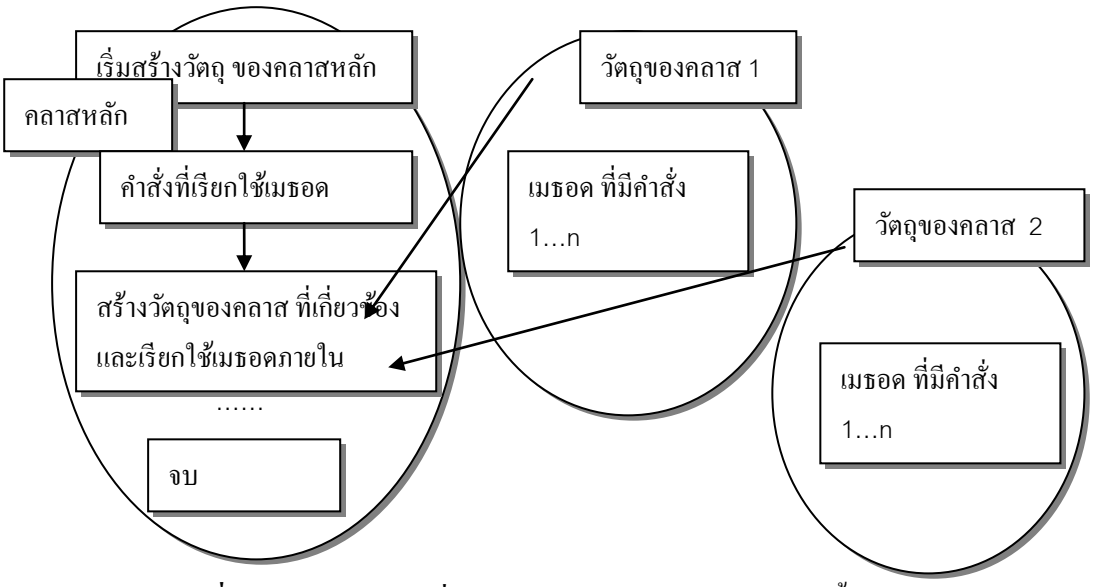


ภาพที่ 1.2 แสดงการใช้งานแบบเชิงโครงสร้าง

จากแนวคิดนี้จึงเป็นพื้นฐานการพัฒนาโปรแกรมในรูปแบบอื่นๆ ด้วย เช่นรูปแบบเชิงวัตถุเองก็มีส่วนที่เป็นโครงสร้างนี้ไปฝังเป็นส่วนหนึ่งของเนื้อโปรแกรมอยู่เสมอ ในการเขียนโปรแกรมใดๆ ของโปรแกรมแบบโครงสร้างนั้น มิได้เป็นข้อบังคับว่าจะต้องมีโมดูลมากกว่าหนึ่งเสมอไป ซึ่งอาจมีเพียงโมดูลหลักเพียงโมดูลเดียวก็สามารถทำงานได้ และลักษณะของโมดูลเองก็ไม่จำเป็นต้องอยู่ภายนอกโมดูลหลักก็ได้ อาจจะอยู่เป็นโมดูลย่อยๆ ภายในโมดูลหลักก็เป็นได้

การเขียนโปรแกรมเชิงวัตถุนั้น เริ่มต้นด้วยการพิจารณางานต่างๆ เป็นวัตถุที่ต้องมีฟังก์ชันการทำงานหรือเรียกว่าเมธอดและตัวแปรสำหรับขับเคลื่อนโปรแกรม หลังจากนั้นการเขียนโปรแกรมจริงเริ่มต้นด้วยการสร้างคลาส ที่ถือว่าเป็นแม่ของวัตถุ โดยมีคลาสหลักเพียงหนึ่งคลาสและคลาสที่เกี่ยวข้องอีกหลาย ๆ คลาส และแต่ละคลาสอาจมีความสัมพันธ์กันตามแนวทางของวัตถุ หลังจากนั้นการขับเคลื่อนโปรแกรมจะเริ่มจากคลาสหลักสร้างวัตถุของตนเองขึ้นมา แล้วสร้างวัตถุที่เป็นองค์ประกอบของการทำงานทั้งหมด และสั่งทำงานโดยผ่านเมธอดต่างๆ ของแต่ละคลาสที่เกี่ยวข้อง ภาพด้านบนนี้แสดงแนวคิดเบื้องต้นของการเขียนโปรแกรมเชิงวัตถุเท่านั้น ยังมีแนวคิดอื่นๆ ที่สามารถแทนได้ด้วยภาพอื่นๆ อีก

แนวคิดเกี่ยวกับความแตกต่างระหว่างการเขียนโปรแกรมแบบโครงสร้างกับแนวคิดการเขียนโปรแกรมเชิงวัตถุนั้น สามารถเปรียบเทียบกันได้โดยใช้ตัวอย่างต่อไปนี้

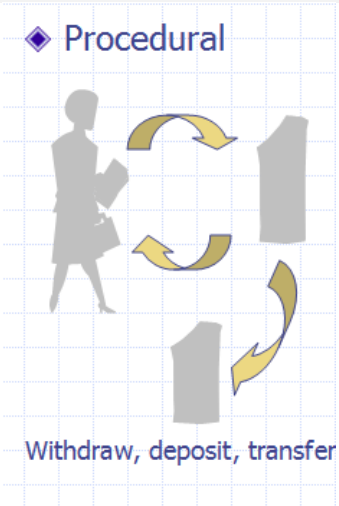


ภาพที่ 1.3 แสดงแนวคิดเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุเบื้องต้น

จากภาพอธิบายได้ว่า การเขียนโปรแกรมเชิงวัตถุนั้น เริ่มต้นด้วยการพิจารณางานต่างๆ เป็นวัตถุที่ต้องมีฟังก์ชันการทำงานหรือเรียกว่าเมธอดและตัวแปรสำหรับขับเคลื่อนโปรแกรม หลังจากนั้นการเขียนโปรแกรมจริงเริ่มต้นด้วยการสร้างคลาส ที่ถือว่าเป็นแม่ของวัตถุ โดยมีคลาสหลักเพียงหนึ่งคลาส และคลาสที่เกี่ยวข้องอีกหลายๆ คลาส และแต่ละคลาสอาจมีความสัมพันธ์กันตามแนวทางของวัตถุ หลังจากนั้นการขับเคลื่อนโปรแกรมจะเริ่มจากคลาสหลักสร้างวัตถุของตนเองขึ้นมา แล้วสร้างวัตถุที่เป็นองค์ประกอบของการทำงานทั้งหมด และสั่งทำงานโดยผ่านเมธอดต่าง ๆ ของแต่ละคลาสที่เกี่ยวข้อง ภาพด้านบนนี้แสดงแนวคิดเบื้องต้นของการเขียนโปรแกรมเชิงวัตถุเท่านั้น ยังมีแนวคิดอื่น ๆ ที่สามารถแทนได้ด้วยภาพอื่น ๆ อีก

ดังนั้นผู้ที่จะเขียนโปรแกรมเชิงวัตถุได้ต้องฝึกมองทุกอย่างให้เป็นเชิงวัตถุก่อน ภาพที่ 1.3 และ 1.4 แสดงการเปรียบเทียบแนวคิดหากคิดเชิงโครงสร้างและคิดเชิงวัตถุสำหรับการเขียนโปรแกรม ให้สังเกตแนวคิดจาก ตัวอย่างที่ 1.1 และตัวอย่างที่ 1.2

ตัวอย่างที่ 1.1 แนวคิดเปรียบเทียบการทำงานแบบโครงสร้าง กับการทำงานแบบเชิงวัตถุ



ภาพที่ 1.4 แสดงแนวคิดการทำงานแบบโครงสร้าง

(ที่มา : จากสไลด์เรื่อง Object Oriented Programming, Tal Pasternak, ISIS)

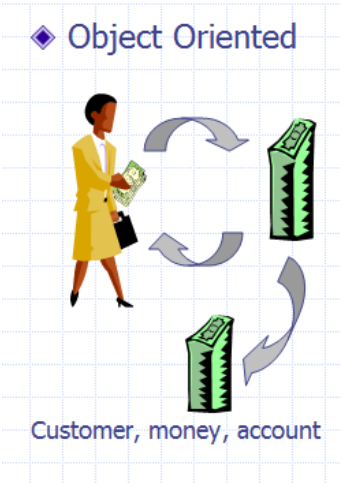
<u>แบบโครงสร้าง</u>

พิจารณาที่ฟังก์ชันการทำงานเป็นประกอบด้วย

1. ฟังก์ชันการฝาก
2. ฟังก์ชันการถอน
3. ฟังก์ชันการโอน

ข้อมูลที่ผ่านค่าพารามิเตอร์ ให้กับฟังก์ชันดังกล่าวเหล่านั้นคือ ข้อมูลลูกค้า ข้อมูลเกี่ยวกับจำนวนเงินที่ดำเนินการ การเรียกใช้รายการใดรายการหนึ่งเพื่อเรียกใช้ฟังก์ชันดังกล่าว

<u>แบบเชิงวัตถุ</u>



ภาพที่ 1.5 แสดงแนวคิดการทำงานเชิงวัตถุ

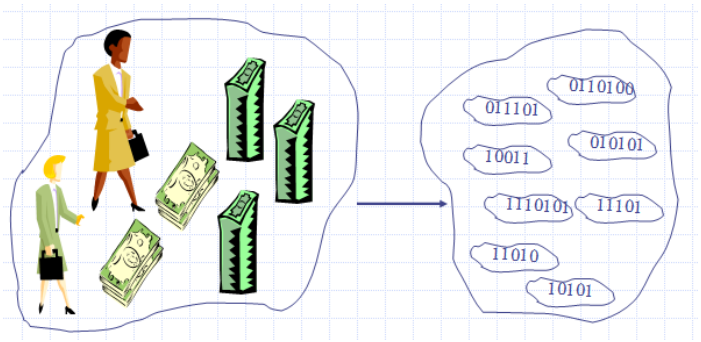
(ที่มา : จากสไลด์เรื่อง Object Oriented Programming, Tal Pasternak, ISIS)

พิจารณาการทำงานของโปรแกรมเชิงวัตถุ ประกอบด้วย

1. ลูกค้า
2. ตัวเงิน
3. บัญชีธนาคาร

ข้อมูลและตัวฟังก์ชันงานต่างๆ จะอยู่ภายใต้วัตถุเหล่านี้ เช่นวัตถุลูกค้าประกอบด้วย ชื่อลูกค้า หมายเลขบัญชี และจะมีเมธอดการสอบถามชื่อลูกค้า เมธอดการเรียกหมายเลขบัญชีธนาคาร วัตถุบัญชีธนาคารประกอบด้วยเมธอด การฝาก ถอน และโอน เป็นต้น

หลังจากนั้นพิจารณาว่าจะนำแปลงจากแนวคิดเชิงวัตถุเป็นโปรแกรมที่เปลี่ยนการทำงานทุกอย่างอยู่ภายใต้การทำงานของโปรแกรมที่เขียนขึ้น



ภาพที่ 1.6 แสดงการทำงานของโปรแกรมแบบเชิงวัตถุในการเขียนโปรแกรม

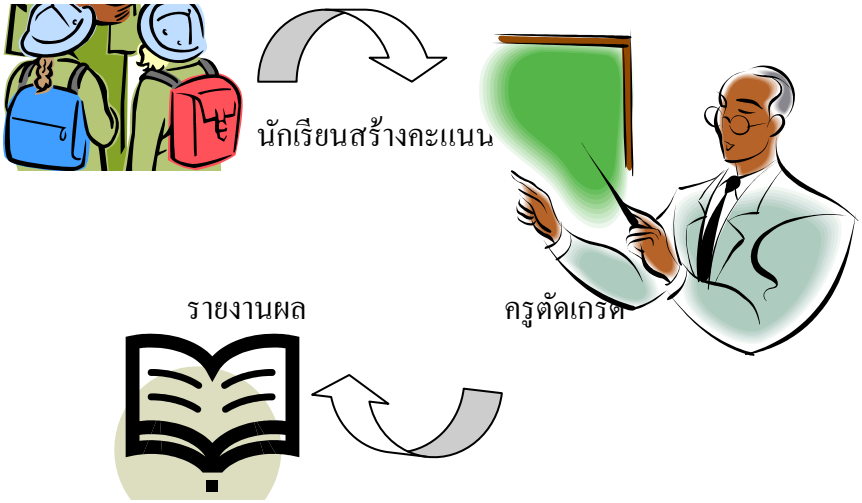
(ที่มา : จากสไลด์เรื่อง Object Oriented Programming, Tal Pasternak, ISIS)

จากลักษณะการทำงานด้านบนพบว่าหากพิจารณาวัตถุเพียง Account (บัญชีธนาคาร) เป็นการรวมระหว่างวัตถุ หรือคุณสมบัติต่างๆ ของวัตถุที่เกี่ยวข้อง และฟังก์ชันการทำงานอยู่ด้วยกัน

ภาพที่ 1.7 แสดงองค์ประกอบภายในของการฝาก-ถอน-โอนเงิน

(ที่มา : จากสไลด์เรื่อง Object Oriented Programming, Tal Pasternak, ISIS)

ตัวอย่างที่ 1.2 จงเปรียบเทียบแนวคิดการออกแบบการตัดเกรดของนักเรียน



ภาพที่ 1.8 ภาพประกอบระบบการตัดเกรด

จากภาพอธิบายได้ว่า โครงสร้างประกอบด้วย รับค่าคะแนน คำนวณเกรด และรายงาน หากพิจารณาเป็นเชิงวัตถุจะประกอบด้วย นักเรียน เกรด วิชาที่เรียน ครู

ข้อดีของการเขียนโปรแกรมเชิงวัตถุ

1. การออกแบบโปรแกรมสามารถดำเนินการได้ในเชิงนามธรรม ซึ่งสามารถรวมทั้งข้อมูลและพฤติกรรมหรือเมธอดได้
2. มีผลดีต่อการบริหารจัดการ เช่น การเพิ่มขยายส่วนต่างๆ ของโปรแกรม ยิ่งในระหว่างที่มีการรันโปรแกรมอยู่
3. สามารถนำกลับมาใช้ใหม่ได้ ทั้งการออกแบบและส่วนของโปรแกรมที่ออกแบบเชื่อมโยงกันไว้ สามารถนำมาประกอบกันเป็นซอฟต์แวร์ขนาดใหญ่ได้โดยง่าย
4. มีการปกป้องข้อมูลที่จำกัดให้ผู้ใช้ที่มีสิทธิ์ใช้เท่านั้น

เครื่องมือที่ใช้เขียนโปรแกรมเชิงวัตถุ

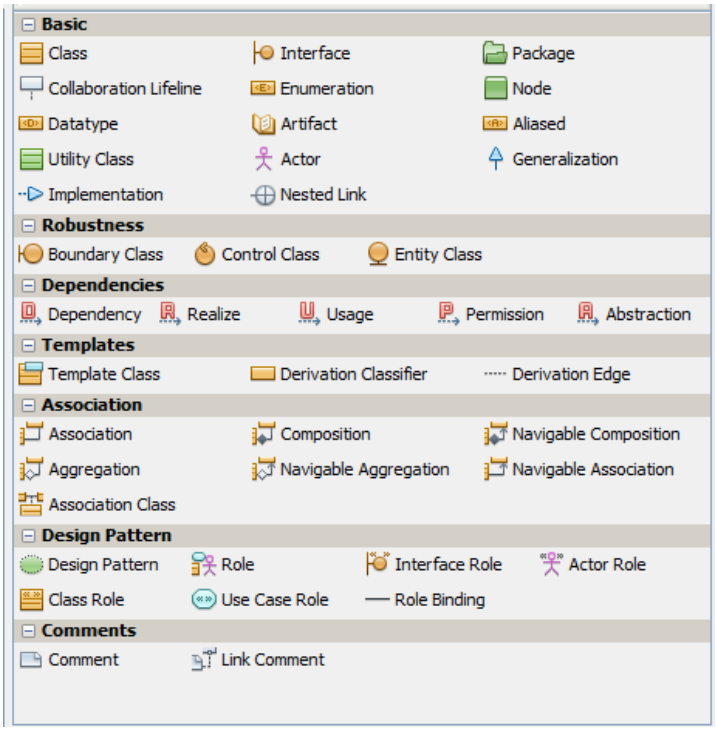
ในการเขียนโปรแกรมเชิงวัตถุนั้น จำเป็นอย่างยิ่งต้องมีความรู้เรื่องของเครื่องมือสำหรับเขียนโปรแกรมแบบนี้ ซึ่งประกอบด้วยสองส่วนคือ เครื่องมือสำหรับออกแบบโปรแกรมซึ่งเป็นภาษาภาพ (Graphic Language) ที่ใช้สำหรับออกแบบอีกทั้งยังมีเครื่องโปรแกรมคอมพิวเตอร์ที่สามารถแปลงภาษาภาพดังกล่าวไปสู่โค้ดโปรแกรมได้อีกด้วย นอกจากนั้นผู้เขียนโปรแกรมต้องเรียนรู้โปรแกรมภาษาคอมพิวเตอร์ที่สามารถเขียนโปรแกรมตามแนวคิดเชิงวัตถุได้อีกด้วย

รายละเอียดต่อไปนี้นำเสนอเครื่องมือสำหรับออกแบบโปรแกรมที่ใช้ในเอกสารประกอบการสอนนี้ และส่วนสุดท้ายได้นำเสนอโปรแกรมภาษาคอมพิวเตอร์ที่สนับสนุนการเขียนโปรแกรมแบบเชิงวัตถุพอสังเขป

เครื่องมือสำหรับออกแบบโปรแกรม

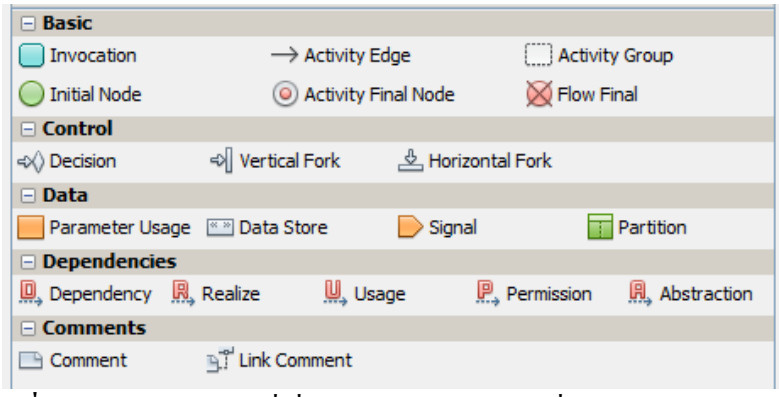
เครื่องมือสำหรับการออกโปรแกรมเชิงวัตถุ คือ ยูเอ็มแอล (UML: Unified Modeling Language) ที่ใช้สัญลักษณ์ภาพที่แตกต่างจากสัญลักษณ์ของงานที่ใช้ในการเขียนโปรแกรมแบบโครงสร้าง สำหรับการออกแบบขั้นตอนการทำงานภายในย่อย ๆ นั้น ยังคงใช้หลักการตามลำดับเงื่อนไข และการวนซ้ำ เช่นเดียวกันกับการเขียนโปรแกรมแบบโครงสร้าง รายละเอียดของการออกแบบดังกล่าวนั้นควรศึกษาจากการออกแบบอัลกอริทึม (Algorithm) สำหรับรายละเอียดของเอกสารประกอบการสอนนี้เน้นการนำเสนอยูเอ็มแอลเกิดจากโปรแกรมเน็ตบีน 6.0 ซึ่งจะนำมาใช้งานตลอดเอกสารนี้ ตัวอย่างของเครื่องมือที่ใช้สำหรับการออกแบบโปรแกรมแสดงดังภาพที่ 1.7

- กลุ่มของคลาสไดอะแกรม



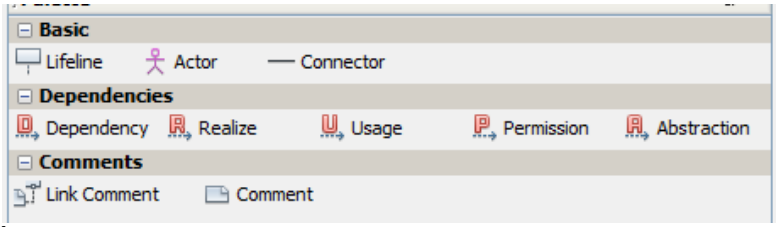
ภาพที่ 1.9 ยูเอ็มแอลกลุ่มที่เกี่ยวคลาสไดอะแกรมที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Activity Diagram



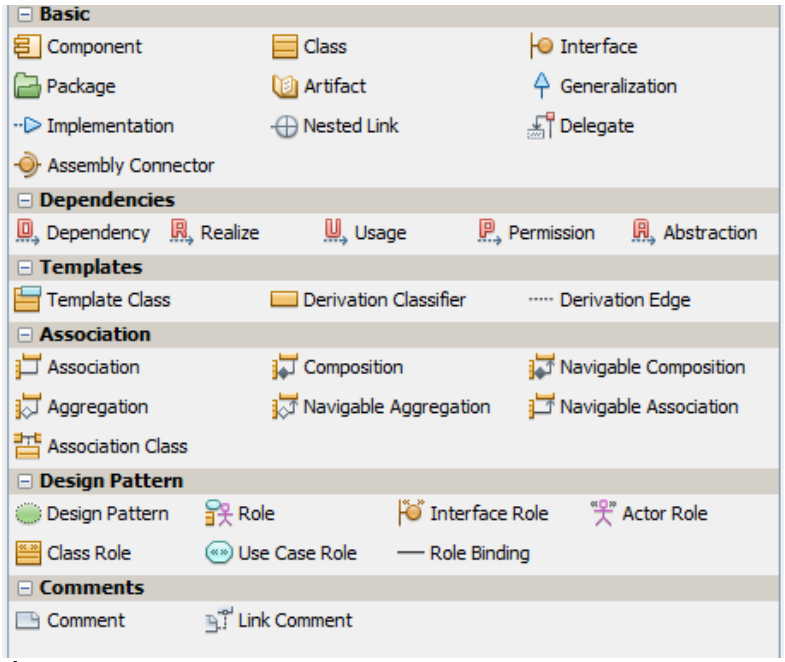
ภาพที่ 1.10 ยูเอ็มแอลกลุ่มที่เกี่ยว Activity Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Collaboration Diagram



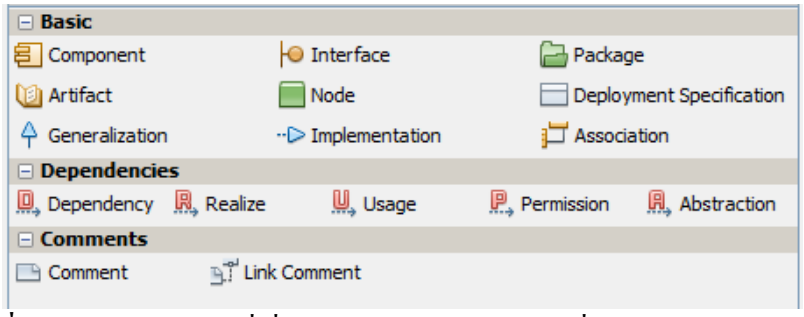
ภาพที่ 1.11 ยูเอ็มแอลกลุ่มที่เกี่ยว Collaboration Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Component Diagram



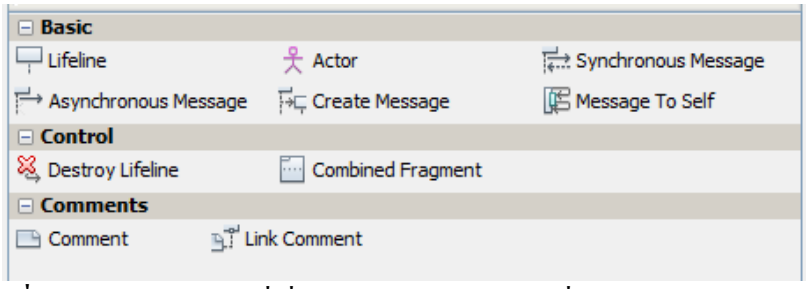
ภาพที่ 1.12 ยูเอ็มแอลกลุ่มที่เกี่ยว Component Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Deployment Diagram



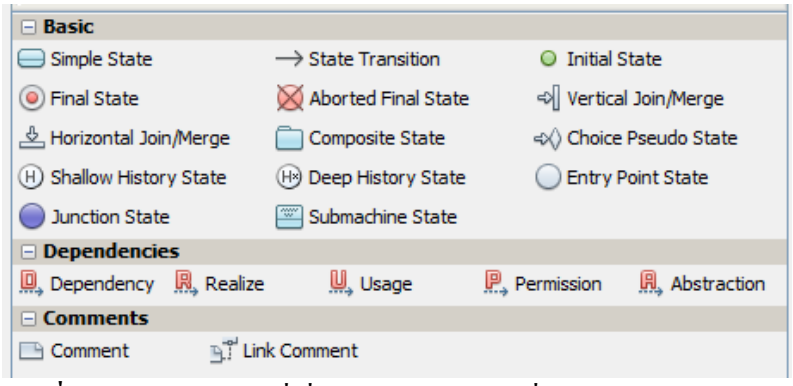
ภาพที่ 1.13 ยูเอ็มแอลกลุ่มที่เกี่ยว Deployment Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Sequence Diagram



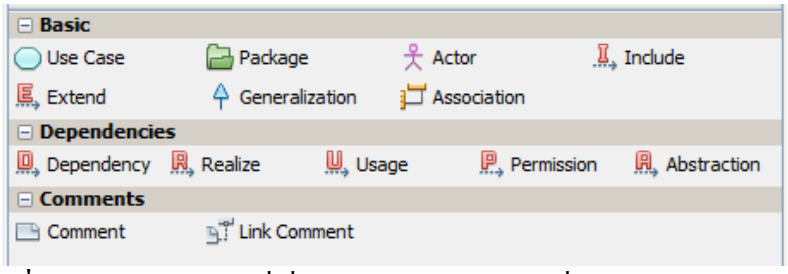
ภาพที่ 1.14 ยูเอ็มแอลกลุ่มที่เกี่ยว Sequence Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ State Diagram



ภาพที่ 1.5 ยูเอ็มแอลกลุ่มที่เกี่ยว State Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

- กลุ่มของ Use case Diagram



ภาพที่ 1.16 ยูเอ็มแอลกลุ่มที่เกี่ยว Use Case Diagram ที่มีในโปรแกรมเน็ตบีน 6.0

สำหรับรายละเอียดของการใช้งานภาษายูเอ็มแอลที่จำเป็นสำหรับการใช้งานเอกสารประกอบการสอนเล่มนี้ นำเสนอในบทที่ 2

โปรแกรมภาษาที่ใช้เขียนโปรแกรมเชิงวัตถุ

โปรแกรมภาษาในปัจจุบันส่วนมากแล้วจะสนับสนุนการเขียนโปรแกรมแบบเชิงวัตถุเกือบทั้งหมด อีกทั้งโปรแกรมภาษาในยุคที่สามของภาษาคอมพิวเตอร์ได้มีการพัฒนาให้สามารถเขียนโปรแกรมตามแนวทางแบบเชิงวัตถุได้อีกด้วย โปรแกรมภาษาคอมพิวเตอร์ต่อไปนี้ได้ถูกเลือกมาใช้สำหรับการเขียนโปรแกรมวัตถุกันอย่างแพร่หลาย

1) โปรแกรมภาษาจาวา เป็นโปรแกรมภาษาที่สามารถเขียนโปรแกรมเชิงวัตถุได้หลากหลายแพลตฟอร์ม ทั้งส่วนที่เป็นโปรแกรมประยุกต์ โปรแกรมสำหรับจัดการบนเว็บ โปรแกรมสำหรับจัดการบนอุปกรณ์ต่างๆ รวมถึงบนมือถือได้อีกด้วย

2) โปรแกรมภาษาซีพลัสพลัส (C++) เป็นโปรแกรมภาษาที่สร้างขึ้นบนพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ ซึ่งโปรแกรมนี้มีความสามารถสูงในการจัดการในเชิงของระบบต่างๆ ของฮาร์ดแวร์

3) โปรแกรมตระกูลวิชวลดอทเน็ต เช่น VB.NET, C#.NET ซึ่งสามารถพัฒนาในรูปแบบวิชวลตามแนวทางไมโครซอฟต์ที่มุ่งเน้นการเขียนโปรแกรมแบบรวดเร็วแต่มีการพัฒนาได้อย่างมีประสิทธิภาพสูง

4) ภาษาสคริปต์ยุคใหม่ เช่น php, jsp, ASP.NET ซึ่งสามารถออกแบบและเขียนโปรแกรมเชิงวัตถุเอาไว้ให้ส่วนของภาษาสคริปต์ได้เรียกใช้งานโดยการนำไปสร้างเป็นวัตถุ แล้วใช้งานในรูปแบบของเว็บแอพพลิเคชัน เป็นต้น

สรุป

โปรแกรมภาษาคอมพิวเตอร์ คือ โปรแกรมที่ใช้เพื่อการเขียนโปรแกรม ซึ่งแตกต่างกันไปตามยุคและระดับของภาษาที่ใช้ แนวทางการเขียนโปรแกรมที่นิยมใช้ คือ การเขียนโปรแกรมแบบโครงสร้าง การเขียนโปรแกรมเชิงวัตถุ การเขียนโปรแกรมคอมพิวเตอร์จะต้องออกแบบโปรแกรมโดยการวิเคราะห์ความต้องการ ออกแบบและวางแผนการเขียนโปรแกรมก่อน จึงลงมือเขียนตามความต้องการที่วิเคราะห์ไว้ หลังจากนั้นคอมไพล์และทดลองรันโปรแกรมเพื่อปรับปรุงที่ผิด และแก้ไขให้ถูกต้องก่อนการนำไปใช้งานจริง การเขียนโปรแกรมเชิงวัตถุ คือ การเขียนโปรแกรมการโดยแทนสิ่งต่างๆ ที่มีอยู่ในโลกเช่นวัตถุหนึ่งชนิด ซึ่งวัตถุดังกล่าวประกอบด้วยแอตทริบิวต์ และเมธอด แนวคิดการเขียนโปรแกรมแบบนี้จะเน้นความยืดหยุ่นทั้งการรับทอด การห่อหุ้มเพื่อปกป้องข้อมูล และการภาวะพหุสัณฐาน

แบบฝึกหัดท้ายบท

1. จงอธิบายความหมายของคำต่อไปนี้พอสังเขป
   - คอมพิวเตอร์
   - โปรแกรมคอมพิวเตอร์
   - โปรแกรมภาษาคอมพิวเตอร์
   - การเขียนโปรแกรมคอมพิวเตอร์
2. ยุคของโปรแกรมภาษาคอมพิวเตอร์แบ่งได้อย่างไรบ้าง จงอธิบาย
3. จงอธิบายและยกตัวอย่างระดับของโปรแกรมภาษาคอมพิวเตอร์
4. จงยกตัวอย่างโปรแกรมภาษาคอมพิวเตอร์ในยุคต่าง ๆ ตั้งแต่ยุคที่ 1-5
5. จงวิเคราะห์ว่าทำไมภาษาคอมพิวเตอร์จึงใกล้เคียงกับภาษาของมนุษย์มากขึ้นทุกขณะ
6. จงคาดคะเนเกี่ยวกับแนวทางการเขียนโปรแกรมคอมพิวเตอร์ในอนาคตอีก 50 ข้างหน้า พร้อมอธิบายเหตุผล
7. จงอธิบายว่าการเขียนโปรแกรมแบบโครงสร้างมีข้อดีข้อด้อยอย่างไร
8. จงอธิบายขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์
9. จงอธิบายขั้นตอนว่าหากท่านได้รับภารกิจให้เขียนโปรแกรมระบบบุคลากรให้กับหน่วยงานท่านจะเริ่มต้น และดำเนินการอย่างไร
10. ท่านคิดว่าในอนาคตที่ก้าวผ่านขั้นยุคที่ N ไปแล้วการเขียนโปรแกรมคอมพิวเตอร์จะเป็นอย่างไร
11. จงบอกความหมายของการเขียนโปรแกรมเชิงวัตถุ
12. จงสรุปและเปรียบเทียบแนวคิดการเขียนโปรแกรมแบบโครงสร้างกับแบบเชิงวัตถุ
13. จงอธิบายการแนวคิดการออกแบบโปรแกรมควบคุมการขึ้นลงของลิฟต์
14. จงอธิบายเครื่องมือการออกแบบโปรแกรมเชิงวัตถุ
15. จงระบุชื่อโปรแกรมภาษาที่ใช้สำหรับพัฒนาโปรแกรมเชิงวัตถุได้ พร้อมทั้งอธิบายหลักการใช้งานเบื้องต้น

เอกสารอ้างอิง

กระทรวงศึกษาธิการ. (2533). **หลักการเขียนโปรแกรม คพ.ทป. 014 ระดับม.ปลาย**. กรุงเทพฯ: องค์การค้าคุรุสภา.

กิตติ ภักดีวัฒนะกุล และกิตติพงษ์ กลมกล่อม. (2548). **คัมภีร์ การวิเคราห์และออกแบบเชิงวัตถุด้วย UML**. กรุเทพฯ: เคทีพี แอนด์ คอนซัลท์.

กิตติ ภักดีวัฒนะกุล และ ศิริวรรณ อัมพรดนัย. (2544). **Oject-Oriented ฉบับพื้นฐาน**. กรุงเทพฯ: KTP Comp&Consult.

กิตติพงษ์ กลมกล่อม. (2552). **การวิเคราะห์และออกแบบระบบเชิงวัตถุด้วย UML**. กรุเทพฯ: เคทีพี แอนด์ คอนซัลท์.

ครรชิต มาลัยวงศ์ และ วิชิต ปุณวัตร์. (2536). **เทคนิคการออกแบบโปรแกรม**. กรุงเทพฯ: ซีเอ็ดยูเคชัน.

ธีรวัฒน์ ประกอบผล. (2552). **คู่มือการเขียนโปรแกรมภาษา Java**. กรุงเทพฯ: ซัคเซส มีเดีย.

_____. (2545). **การโปรแกรมภาษาซีสำหรับงานวิทยาศาสตร์**. กรุงเทพฯ: ส.ส.ท.

_____. (2553). **คู่มือการเขียนโปรแกรมภาษา Java**. กรุงเทพฯ: ซิมพลิฟลาย.

นุกูล กระจาย. (2540). **การเขียนโปรแกรมในดอสและวินโดว์ด้วยบอร์แลนด์ C++ 5.0**. กรุงเทพฯ: ซีเอ็ดยูเคชัน.

บุญเลิศ เอี่ยมทัศนาและคณะ. (2534). **โปรแกรมคอมพิวเตอร์ภาษาซี.**กรุงเทพฯ: ซีเอ็ดยูเคชัน.

พนิดา พานิชกุล. (2548). **Object-Oriented ฉบับพื้นฐาน**. กรุงเทพฯ: เคทีพี แอนด์ คอนซัลท์.

_____. (2554). **การเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นด้วยภาษาจาวา**. พิมพ์ครั้งที่ 5. กรุงเทพฯ: เคทีพี แอนด์ คอนซัลท์.

เพ็ญศรี ปักกะสีนัง. (2546). **เทคโนโลยีสารสนเทศเพื่อชีวิต**. กรุงเทพฯ: ทริปเพิ้ล เซเว่น มัลติเทค.

วัชราภรณ์ สุริยาภิวัฒน์. (2553). **คอมพิวเตอร์เบื้องต้นและเทคนิคการเขียนโปรแกรม**. กรุงเทพฯ: ศูนย์หนังสือแห่งจุฬาลงกรณ์มหาวิทยาลัย.

สมโภชน์ ชื่นเอี่ยม. (2553). **วิทยาการคอมพิวเตอร์เบื้องต้น**. กรุงเทพฯ: ซีเอ็ดยูเคชัน.

Brett Spell. (2000). **Professional Java Programming**. USA: Wrox Press Ltd. Arden House, 1102 Warwick Road, Acocks Green, Birmingham, B27 6BH, UK.

Bruce Eckel. (2000). **Thinking in Java**. 2nd . USA: Prentice Hall PTR Prentice-Hall.

Cay Horstmann. (2002). **Object-Oriented Design & Patterns**. USA: John Wiley & Sons.

David Etheridge. (2009). **Java:The Fundamentals of Objects and Classes-An Introduction to Java Programming**. Ventus Publishing Aps. [Online]. Available: http://www.BookBoon.com

Simol Kendal. (2009). **Object Orientted Programming using Java**. Ventus Publishing Aps. [Online]. Available: http://www.BookBoon.com

Wikipedia. (2007). "**Object-oriented programming**". [Online] Available: http://en.wikipedia.org/wiki/Object-oriented_programming.

Y. Daniel Liang. (2007). **Introduction to Java Programming**. USA: Pearson Prentice Hall Pearson Education, Inc.