# บทที่ 1
# หลักการเชิงวัตถุเบื้องต้น

## เกริ่นนำ

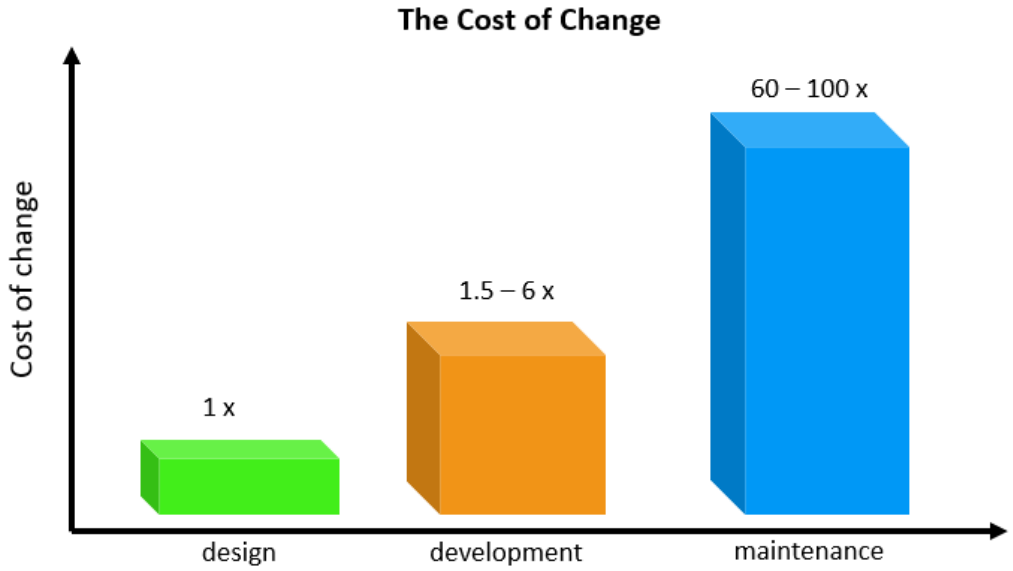
ในบทนี้จะอธิบายถึงความรู้เกี่ยวกับภาษาคอมพิวเตอร์ (Computer Language) และหลักการวิเคราะห์และออกแบบระบบเชิงโครงสร้าง ความสำคัญของการออกแบบและพัฒนาระบบ ความรู้เบื้องต้นเกี่ยวกับหลักการเชิงวัตถุ (OO: Object Orientation) เช่น ความหมายของคลาส (Class) ความหมายวัตถุ (Object) และหลักการที่สำคัญและกลไกที่มีประโยชน์ของหลักการเชิงวัตถุ เช่น การซ่อนรายละเอียด การห่อหุ้ม การรับทอดมรดก และพอลีมอร์ฟิซึม เป็นต้นโดยเมื่อผู้อ่านได้ศึกษาจบบทนี้แล้วจะทำให้ทราบถึงหมายและความสำคัญของหลักการเชิงวัตถุ สามารถอธิบายขั้นตอนการพัฒนาซอฟต์แวร์โดยใช้หลักการเชิงวัตถุและสามารถนำระบบคอมพิวเตอร์มาช่วยในการแก้ไขปัญหาต่าง ๆ โดยอาศัยหลักการเชิงวัตถุแทนการพัฒนาแบบดั้งเดิมที่เป็นแบบเชิงโครงสร้าง (Structural Programming) รวมถึงเหตุผลที่สำคัญในการสร้างหรือพัฒนาซอฟต์แวร์เชิงวัตถุขึ้นมานอกจากนี้ยังอธิบายถึงแนวคิดที่เป็นประโยชน์ของวิธีทางเชิงวัตถุอีกด้วย เพื่อจะเป็นความรู้พื้นฐานที่จะนำไปสู่เนื้อหาของบทต่อ ๆ ไปได้

## ความสำคัญและที่มาของการพัฒนาซอฟต์แวร์

ปัจจุบันเทคโนโลยีสมัยใหม่มีการเปลี่ยนแปลงไปอย่างรวดเร็วและมีแนวโน้มว่าราคาลดลงอย่างต่อเนื่องเมื่อเทียบกับอดีตในช่วงระยะเวลาหลายสิบปีที่ผ่านมาซึ่งทำให้จำนวนของผู้ใช้เทคโนโลยีสารสนเทศมีเพิ่มมากขึ้นทุกวันและได้กลายมาเป็นส่วนหนึ่งที่จำเป็นต่อการดำเนินธุรกิจ ตัวอย่างของเทคโนโลยีที่กล่าวถึง คือ เทคโนโลยีทางคอมพิวเตอร์ อินเทอร์เน็ต การประชุมวิดีโอทางไกล ระบบเครือข่ายคอมพิวเตอร์เทคโนโลยีการสื่อสารไร้สาย อุปกรณ์สื่อสารสมาร์ทโฟนและระบบสารสนเทศเพื่อการวิเคราะห์และการตัดสินใจในด้านต่าง ๆ เป็นต้น โดยเทคโนโลยีเหล่านี้ถือว่าเป็นปัจจัยที่มีความสำคัญต่อการพัฒนาองค์กรในอนาคตดังจะเห็นได้จาก เดฟอุลริช ปรมาจารย์ทางด้านการบริหารทรัพยากรมนุษย์กล่าวว่าเทคโนโลยีเป็นปัจจัยสำคัญอันหนึ่งที่จะมีบทบาทสำคัญต่อการแข่งขันธุรกิจในอนาคตเช่นเดียวกันกับคัมมิ่ง และเวอร์รี่ผู้เชี่ยวชาญในด้านการพัฒนาองค์กรและการบริหารการเปลี่ยนแปลงได้จัดให้เทคโนโลยีสารสนเทศเป็นสิ่งหนึ่งที่จะทำให้เกิดการเปลี่ยนแปลงในองค์กรเทคโนโลยีที่ใช้กันในปัจจุบันจึงถือว่าเป็นตัวขับเคลื่อนที่สำคัญที่จะช่วยให้ผู้ใช้สามารถเก็บรวบรวมข้อมูล การแก้ไขเปลี่ยนแปลง การเรียกดูข้อมูล การประมวลผลการใช้งานร่วมกันแบบหลายเครื่องพร้อม ๆ กัน ทำให้การวิเคราะห์ข้อมูลทำได้ง่ายขึ้นโดยที่มีค่าใช้จ่ายลดลง เพิ่มคุณค่าและประโยชน์ในการใช้งานข้อมูลและสารสนเทศที่ได้มาจะมีคุณภาพในการนำไปวิเคราะห์และใช้งานเพิ่มมากขึ้นในขณะเดียวกันเทคโนโลยียังสามารถช่วยให้เกิดการพัฒนาและปรับปรุงกระบวนการในการผลิตและการทำงานให้มีต้นทุนที่ต่ำลงโดยใช้เวลาในการทำงานที่

น้อยลงโดยได้สินค้าหรือผลลัพธ์ที่มีคุณภาพมากยิ่งขึ้น ดังนั้นเทคโนโลยีจึงมีความสำคัญต่อการพัฒนาองค์กรเป็นอย่างยิ่ง

นับตั้งแต่ช่วงยุคต้นจนถึงยุคกลางของการพัฒนาระบบงานโดยใช้คอมพิวเตอร์ (ยุคต้นที่ผู้แต่งกล่าวถึงนี้นับตั้งแต่ได้มีการพัฒนาซอฟต์แวร์ด้วยภาษาคอมพิวเตอร์ที่ทำงานอยู่บนระบบปฏิบัติการดอส ส่วนยุคกลางหมายถึงการพัฒนาระบบด้วยภาษาคอมพิวเตอร์ที่ทำงานบนระบบปฏิบัติการวินโดวส์) การพัฒนาระบบส่วนใหญ่จะมุ่งเน้นไปที่การมองปัญหาเป็นประเด็นหลัก (Problem Domain) แล้วพยายามแก้ปัญหาโดยใช้การออกแบบขั้นตอนวิธี (Algorithm) แล้วเลือกภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่งในการพัฒนาระบบ จากนั้นก็ลงมือเขียนคำสั่งตามขั้นตอนวิธีด้วยภาษาที่เลือกใช้นั้นจนได้ซอฟต์แวร์ที่ตรงกับความต้องการของผู้ใช้งานแล้วจึงนำไปติดตั้งและใช้งานต่อไปแต่เนื่องจากในสมัยก่อนระบบซอฟต์แวร์ต่าง ๆ ที่พัฒนานั้นส่วนใหญ่แล้วมักจะเป็นระบบงานเดี่ยว คือเป็นระบบที่มีขนาดเล็ก มีความซับซ้อนของปัญหาไม่มากนัก ดังนั้นวิธีการนี้จึงสามารถใช้งานได้ดีในระดับหนึ่งแต่ปัจจุบันเมื่อเทคโนโลยีสารสนเทศและคอมพิวเตอร์ได้พัฒนามาถึงยุคของอินเทอร์เน็ตข้อมูลและข่าวสารต่าง ๆ ก็สามารถเข้าถึงได้อย่างรวดเร็ว โดยปริมาณข้อมูลความต้องการของผู้ใช้ และจำนวนผู้ใช้งานที่มากมายมหาศาลทำให้ระบบงานจึงมีขนาดใหญ่และซับซ้อนมากขึ้นดังนั้นวิธีการพัฒนาระบบแบบดั้งเดิมจึงไม่เหมาะสมอีกต่อไป




**ภาพที่ 1.1** เปรียบเทียบค่าใช้จ่ายของการปรับเปลี่ยนระบบงาน

จากภาพที่ 1.1 แสดงการเปรียบเทียบค่าใช้จ่ายที่อาจจะเกิดจากการปรับเปลี่ยนระบบซอฟต์แวร์ในช่วงระยะเวลาต่าง ๆ ซึ่งจะสังเกตเห็นได้ว่าค่าใช้จ่ายนั้นจะแปรผันโดยตรงกับระยะเวลาของการพัฒนาระบบโดยเมื่อเวลาผ่านไปเรื่อย ๆ จะทำให้มีค่าใช้จ่ายเพิ่มสูงขึ้นตามไปด้วยนั่นเอง

## ความหมายของซอฟต์แวร์

ซอฟต์แวร์ (Software) หมายถึงชุดคำสั่งหรือโปรแกรมที่ใช้สั่งงานให้คอมพิวเตอร์ทำงานตามความต้องการของมนุษย์ ซึ่งภายในซอฟต์แวร์ประกอบไปด้วยลำดับหรือขั้นตอนการทำงานที่ถูกเขียนขึ้นด้วยคำสั่งของคอมพิวเตอร์ เนื่องจากการทำงานพื้นฐานเป็นเพียงการกระทำกับข้อมูลที่เป็นตัวเลขฐานสอง ซึ่งใช้แทนข้อมูลที่เป็นตัวเลข ตัวอักษร รูปภาพ หรือเสียง โปรแกรมคอมพิวเตอร์ที่ใช้สั่งงานคอมพิวเตอร์จึงเป็นซอฟต์แวร์ เพราะเป็นลำดับขั้นตอนการทำงานของคอมพิวเตอร์ คอมพิวเตอร์เครื่องหนึ่งทำงานแตกต่างกันได้มากมายด้วยซอฟต์แวร์ที่แตกต่างกัน ซอฟต์แวร์จึงหมายรวมถึงโปรแกรมคอมพิวเตอร์ทุกประเภทที่ทำให้คอมพิวเตอร์ทำงานได้การที่เห็นคอมพิวเตอร์ทำงานให้มนุษย์ได้มากมายเป็นเพราะว่ามีผู้พัฒนาโปรแกรมคอมพิวเตอร์มาให้สั่งงานคอมพิวเตอร์ ร้านค้าอาจใช้คอมพิวเตอร์ทำบัญชีที่ยุ่งยากซับซ้อน บริษัทขายตั๋วใช้คอมพิวเตอร์ช่วยในระบบการจองตั๋ว คอมพิวเตอร์ช่วยในเรื่องกิจการงานธนาคารที่มีข้อมูลต่าง ๆ มากมาย คอมพิวเตอร์ช่วยงานพิมพ์เอกสารให้สวยงาม เป็นต้น การที่คอมพิวเตอร์ดำเนินการให้ประโยชน์ได้มากมายมหาศาลจะอยู่ที่ซอฟต์แวร์ ซอฟต์แวร์จึงเป็นส่วนสำคัญของระบบคอมพิวเตอร์หากขาดซอฟต์แวร์คอมพิวเตอร์ก็ไม่สามารถทำงานได้ ดังนั้นซอฟต์แวร์จึงเป็นสิ่งที่จำเป็นและมีความสำคัญต่อคอมพิวเตอร์เป็นอย่างมาก (วิกิตำรา, 2563)

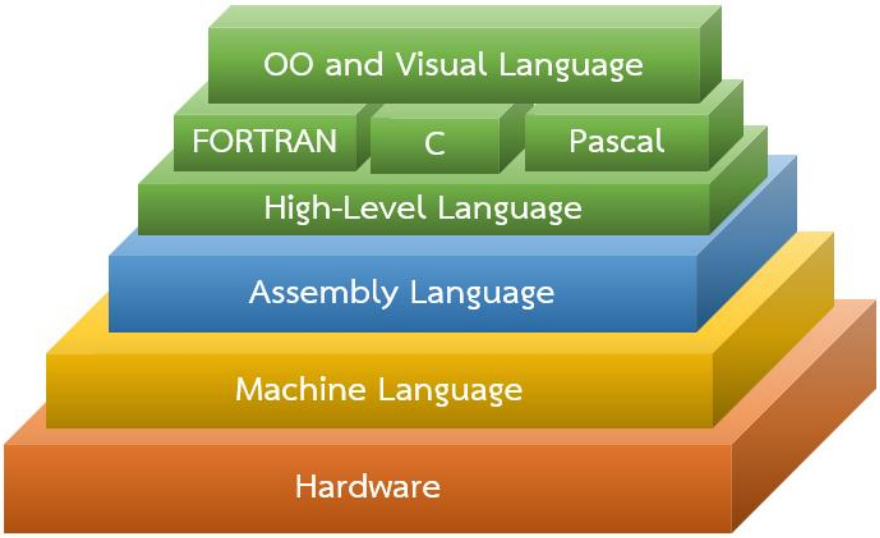
ดังนั้นความหมายของซอฟต์แวร์ตามเข้าใจของผู้แต่งคือชุดคำสั่ง (Instructions) ที่สั่งให้คอมพิวเตอร์ประมวลผลโดยสามารถกระทำงานตามที่ระบุไว้ให้สำเร็จ และมีประสิทธิภาพที่ต้องการได้โดยใช้โครงสร้างข้อมูล (Data Structure) ทำหน้าที่เก็บสารสนเทศที่ใช้งานหรือถูกเรียกใช้โดยโปรแกรม ซึ่งการพัฒนาซอฟต์แวร์ที่ดีควรมีเอกสารหรือคู่มือผู้ใช้งาน (User Manual) ที่อธิบายหรือบรรยายวิธีปฏิบัติการและวิธีการใช้งานโปรแกรมรวมอยู่ด้วยโดยตัวซอฟต์แวร์นั้นอาจถูกพัฒนาขึ้นมาด้วยภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง เช่น ภาษาซี ภาษาปาสคาล ภาษาจาวา ภาษาวิชวลเบสิก เป็นต้น


**ภาพที่ 1.2** ตัวอย่างซอฟต์แวร์ต่าง ๆ

## ภาษาคอมพิวเตอร์

### 1) ความหมายของภาษาคอมพิวเตอร์

ภาษาคอมพิวเตอร์ หมายถึงภาษาใด ๆ ที่ผู้ใช้งานใช้สื่อสารกับคอมพิวเตอร์ หรือคอมพิวเตอร์ด้วยกัน โดยคอมพิวเตอร์สามารถทำงานตามคำสั่งนั้นได้ คำนี้มักใช้เรียกแทนภาษาโปรแกรม แต่ความเป็นจริงภาษาโปรแกรมคือส่วนหนึ่งของภาษาคอมพิวเตอร์เท่านั้น และมีภาษาอื่น ๆ ที่เป็นภาษาคอมพิวเตอร์ด้วยเช่นกันตัวอย่างภาษาเอชทีเอ็มแอล (HTML) เป็นทั้งภาษามาร์กอัพและภาษาคอมพิวเตอร์ด้วย แม้ว่ามันจะไม่ใช่ภาษาโปรแกรมแต่ก็นับเป็นภาษาคอมพิวเตอร์ ซึ่งโดยทางเทคนิคสามารถใช้ในการเขียนโปรแกรมได้ ภาษาคอมพิวเตอร์นั้นสามารถแบ่งออกเป็นสองกลุ่มคือภาษาระดับสูง (High Level Language) และภาษาระดับต่ำ (Low Level Language) โดยภาษาระดับสูงถูกออกแบบมาเพื่อให้ใช้งานง่ายและสะดวกสบายมากกว่าภาษาระดับต่ำโปรแกรมหรือซอฟต์แวร์ที่เขียนถูกต้องตามกฎเกณฑ์และไวยากรณ์ของภาษาจะถูกแปล (Compile) ไปเป็นภาษาระดับต่ำเพื่อให้คอมพิวเตอร์สามารถนำไปใช้งานหรือปฏิบัติตามคำสั่งได้ต่อไป ซอฟต์แวร์สมัยใหม่ส่วนมากเขียนด้วยภาษาระดับสูงที่แปลไปเป็นรหัสคำสั่งวัตถุ (Object Code) แล้วเปลี่ยนให้เป็นชุดคำสั่งในภาษาเครื่อง (Object Management Group, 1997)




**ภาพที่ 1.3** แสดงตัวอย่างระดับของภาษาคอมพิวเตอร์

### 2) ระดับของภาษาคอมพิวเตอร์

ภาษาคอมพิวเตอร์นั้นแบ่งออกเป็นระดับต่าง ๆ ได้ดังต่อไปนี้

2.1 ภาษาเครื่อง

ภาษาเครื่อง(Machine Language) เป็นภาษาที่ใช้กันมาตั้งแต่เริ่มมีคอมพิวเตอร์ใหม่ๆ ลักษณะทั่วไปก็คือใช้รหัสเป็นเลขฐานสองทั้งหมดซึ่งนับว่ายุ่งยากกับผู้ใช้มากแต่ก็เป็นภาษาเดียวที่คอมพิวเตอร์จะเข้าใจได้ทันทีคำสั่งที่เขียนด้วยภาษานี้จะแบ่งเป็นสองส่วน โดยส่วนแรกเป็นคำสั่งที่จะสั่งให้

เครื่องคอมพิวเตอร์รู้ว่าจะต้องทำอะไรเรียกส่วนนี้ว่าออพโค้ด (Op-Code หรือที่ย่อมาจากคำว่า Operation Code) และส่วนที่สองจะบอกคอมพิวเตอร์ว่าให้ไปนำข้อมูลมาจากที่ใดเรียกส่วนนี้ว่าตัวถูกดำเนินการ (Operand) ในการเขียนด้วยคำสั่งภาษานี้ผู้พัฒนาโปรแกรมจะต้องจำที่อยู่ของข้อมูลหรือที่เก็บข้อมูลเหล่านั้นได้ซึ่งจะเป็นตัวเลขทั้งหมดอาจมีตั้งแต่ 1 ถึง 100,000 แล้วแต่ขนาดของเครื่องโดยปกติกว่าจะจำได้มักจะใช้เวลามากและแม้กระนั้นก็ยังผิดพลาดอยู่เสมอ เช่นถ้าจะสั่งให้นำค่าที่หน่วยความจำเลขที่ 0184 บวกกับค่าที่อยู่ในหน่วยความจำ 8672 จะเขียนดังนี้

```
00100000000000000000000000010111000
```

หรือแม้แต่เขียนเป็นเลขฐานสิบก็ยังยุ่งยาก คำสั่งของภาษาเครื่องนี้จะต่างกันไปตามชนิดของเครื่องคอมพิวเตอร์ที่ใช้โดยภาษาเครื่อง (Machine language) จึงจัดเป็นภาษาเดียวที่คอมพิวเตอร์เข้าใจและเป็นภาษาที่ประกอบด้วยตัวอักษรเพียงสองตัวคือ 0 กับ 1 เท่านั้น การใช้ภาษาเครื่องนั้นค่อนข้างยากมาก นอกจากจะต้องจำคำสั่งเป็นลำดับของเลข 0 กับ 1 แล้วยังจะต้องออกคำสั่งต่าง ๆ อย่างละเอียดมาก ๆ ด้วยตัวอย่างของคำสั่งภาษาเครื่องสำหรับบวกเลขสองจำนวนอาจมีลักษณะดังนี้

```
0110000000000110
0110110000010000
1010010000010001
```

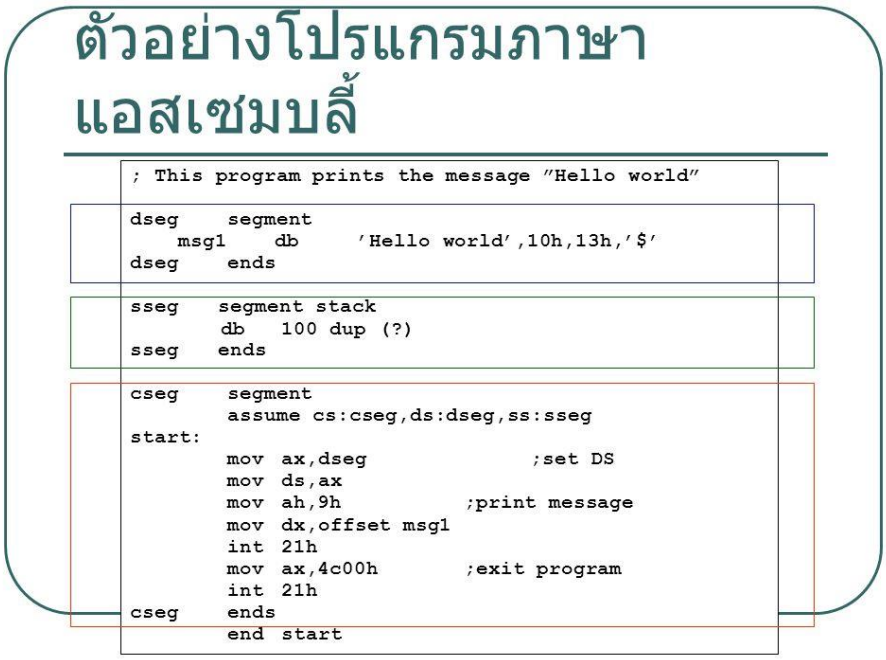
**ข้อสังเกต** ภาษาเครื่องเป็นภาษาเดียวที่คอมพิวเตอร์เข้าใจแต่เป็นการยากสำหรับมนุษย์ที่จะเขียนหรือพัฒนาโปรแกรมที่มีความซับซ้อนด้วยภาษาเครื่อง มนุษย์จำเป็นต้องสร้างภาษาที่ใกล้เคียงภาษามนุษย์ขึ้นมาเพื่อให้สามารถพัฒนาซอฟต์แวร์ได้ง่ายขึ้น

โดยข้อดีของภาษาเครื่องก็คือสามารถเขียนโปรแกรมควบคุมการทำงานคอมพิวเตอร์ได้โดยตรง และสั่งงานให้คอมพิวเตอร์ทำงานได้อย่างรวดเร็ว

2.2 ภาษาแอสเซมบลี

ภาษาแอสเซมบลี (Assembly) เป็นภาษาที่มีการใช้สัญลักษณ์ข้อความ (Mnemonic Codes) แทนกลุ่มของเลขฐานสองเพื่อให้ง่ายต่อการเขียนและการจดจำมากกว่าภาษาเครื่องแต่เนื่องจากคอมพิวเตอร์รู้จักหรือเข้าใจเฉพาะภาษาเครื่องเท่านั้นดังนั้นภาษาแอสเซมบลีจึงต้องใช้ตัวแปลภาษาที่เรียกว่า “แอสแซมเบลอร์ (Assembler)” เพื่อแปลคำสั่งภาษาแอสเซมบลีให้เป็นภาษาเครื่องนอกจากนี้ผู้

ที่จะเขียนโปรแกรมภาษาแอสเซมบลี้ได้จะต้องมีความรู้ความเข้าใจในเรื่องของฮาร์ดแวร์เป็นอย่างดี เนื่องจากต้องยุ่งเกี่ยวกับการใช้งานหน่วยความจำที่เป็นรีจิสเตอร์ภายในตลอดดังนั้นจึงเหมาะกับงานที่ต้องการความเร็วในการทำงานสูงถึงแม้ว่าภาษานี้จะง่ายกว่าการเขียนภาษาเครื่องแต่ก็ยังถือว่าเป็นภาษาชั้นต่ำที่ยังยากต่อการเขียนและการเรียนรู้ของผู้ที่ไม่มีความรู้ด้านฮาร์ดแวร์เป็นอย่างมาก (Tutorialspoint, 2014) ดังแสดงในภาพที่ 1.4




**ภาพที่ 1.4** ตัวอย่างคำสั่งของภาษาแอสเซมบลี

(**ที่มา:** https://slideplayer.in.th/slide/2144422/)

2.3 ภาษาชั้นสูง

ภาษาชั้นสูงหรือเรียกอีกอย่างว่าภาษารุ่นที่ 3 (3rd Generation Languages หรือ 3GLs) เป็นภาษาที่ถูกสร้างขึ้นมาเพื่อให้สามารถเขียนและอ่านโปรแกรมได้ง่ายขึ้นเนื่องจากมีลักษณะเหมือนภาษาอังกฤษทั่ว ๆ ไป และที่สำคัญคือผู้เขียนโปรแกรมไม่จำเป็นต้องมีความรู้เกี่ยวกับระบบฮาร์ดแวร์ตัวอย่างของภาษาประเภทนี้ เช่น ภาษาฟอร์แทน (FORTRAN) ภาษาโคบอล (COBOL) ภาษาเบสิก (BASIC) ภาษาปาสคาล (Pascal) ภาษาซี (C) ภาษาเอดา (Ada) เป็นต้น อย่างไรก็ตามโปรแกรมที่ถูกเขียนด้วยภาษาประเภทนี้จะทำงานได้ก็ต่อเมื่อมีการแปลงให้เป็นภาษาเครื่องเสียก่อนซึ่งวิธีการแปลงภาษาชั้นสูงให้เป็นภาษาเครื่องนั้นจะทำได้โดยใช้โปรแกรมที่เรียกว่า คอมไพล์เลอร์ (Compiler) หรืออินเตอร์พรีเตอร์ (Interpreter) อย่างใดอย่างหนึ่งโดยภาษาชั้นสูงแต่ละภาษาจะมีตัวแปลภาษาเฉพาะเป็นของตนเองใช้แทนกันไม่ได้โดยจะมี 2 ประเภทดังนี้

1) คอมไพเลอร์จะทำการแปลโปรแกรมทั้งโปรแกรมให้เป็นภาษาเครื่องทีเดียวการแปลนี้จะเป็นการตรวจสอบไวยากรณ์ของภาษา ถ้ามีข้อผิดพลาดทางไวยากรณ์ของภาษาเกิดขึ้น (Syntax Error) ก็จะแจ้งให้ทราบเป็นข้อความเตือน เพื่อให้ผู้เขียนโปรแกรมแก้ไขให้ถูกต้อง แล้วจึงค่อยแปลคำสั่งใหม่ โปรแกรมที่ยังไม่ผ่านการแปลจะเรียกว่าโปรแกรมต้นฉบับ (Source Program) แต่ถ้าผ่านการแปลเรียบร้อยแล้วและไม่มีข้อผิดพลาดใด ๆ จะเรียกโปรแกรมส่วนนี้ว่าโปรแกรมวัตถุ (Object Program) หรือวัตถุโมดูล (Object Module) วัตถุโปรแกรมนี้จะยังไม่สามารถทำงานได้ จะต้องผ่านลิงค์ (Link) หรือรวมเข้ากับไลบรารี (Library) ของระบบก่อนจึงจะเป็นโปรแกรมที่สามารถทำงานได้หรือเป็นภาษาเครื่องที่เรียกว่าโปรแกรมเอ็กซ์ซีคิวท์ (Execute Program) หรือโหลดโมดูล (Load Module) ซึ่งโดยทั่วไปแล้วจะเป็นไฟล์ที่มีนามสกุลเป็น .exe หรือ .com และสามารถนำโปรแกรมนี้ไปใช้งานได้ตลอดโดยไม่ต้องแปลใหม่อีกแต่ถ้ามีการแก้ไขโปรแกรมแม้เพียงเล็กน้อยก็จะต้องทำการแปลใหม่ตั้งแต่ต้น

2) อินเตอร์พรีเตอร์ คือ ตัวแปลภาษาที่จะทำการแปลโปรแกรมภาษาชั้นสูงทีละคำสั่งให้เป็นภาษาเครื่องและทำการประมวลผลหรือทำงานคำสั่งนั้นทันทีทันใด ก่อนที่จะทำการแปลในบรรทัดถัดไปถ้าในระหว่างการแปลเกิดพบข้อผิดพลาดที่บรรทัดใดก็จะฟ้องให้ทำการแก้ไขทีละบรรทัดนั้นทันที โดยอินเตอร์พรีเตอร์นี้เมื่อโปรแกรมเสร็จแล้วจะไม่สามารถเก็บเป็นโปรแกรมเอ็กซ์ซีคิวท์ได้ ซึ่งต่างกับคอมไพเลอร์ดังนั้นเมื่อจะเรียกใช้งานหรือรันโปรแกรมก็จะต้องทำการแปลโปรแกรมใหม่ทุกครั้ง ดังนั้นเมื่อจะเรียกใช้งานโปรแกรมเอ็กซ์ซีคิวท์คอมไพเลอร์ย่อมจะทำงานได้เร็วกว่าการเรียกใช้งานโปรแกรมที่ต้องผ่านการแปลด้วยอินเตอร์พรีเตอร์ แต่ประโยชน์ของภาษาที่ถูกแปลด้วยอินเตอร์พรีเตอร์คือโปรแกรมจะมีโครงสร้างที่ง่ายต่อการพัฒนา ตัวอย่างของภาษาโปรแกรมที่มีการใช้อินเตอร์พรีเตอร์เป็นตัวแปลภาษา เช่น ภาษาเบสิก (Basic) ภาษาเพิร์ล (Perl) เป็นต้น

การเขียนโปรแกรมด้วยภาษาชั้นสูงนอกจากจะให้ความสะดวกแล้วผู้เขียนแทบจะไม่ต้องมีความรู้เกี่ยวกับการทำงานของฮาร์ดแวร์ก็สามารถเขียนโปรแกรมสั่งให้เครื่องคอมพิวเตอร์ทำงานได้ข้อดีอีกอย่างคือสามารถนำโปรแกรมที่เขียนขึ้นไปใช้งานบนเครื่องใดก็ได้ โดยมีลักษณะที่ไม่ขึ้นอยู่กับเครื่อง (Hardware Independent) เพียงแต่ต้องทำการการแปลโปรแกรมใหม่เท่านั้นแต่ภาษาเครื่องที่ได้จากการแปลภาษาชั้นสูงนี้อาจมากเกินความจำเป็นและไม่มีประสิทธิภาพเท่ากับการเขียนด้วยภาษาเครื่องหรือแอสเซมบลีโดยตรงภาษารุ่นที่ 3 นี้ส่วนใหญ่จะจัดอยู่ในกลุ่มของภาษาที่มีแบบแผน (Procedural Language) เนื่องจากลักษณะการเขียนโปรแกรมจะมีโครงสร้างแบบแผนที่เป็นระเบียบ หมายถึงงานทุกอย่างผู้เขียนโปรแกรมต้องเขียนโปรแกรมควบคุมการทำงานเองทั้งหมดและต้องเขียนคำสั่งการทำงานที่เป็นขั้นตอนทุกอย่างไม่ว่าจะเป็นการสร้างแบบฟอร์มกรอกข้อมูล การประมวลผล หรือการสร้างรายงาน ซึ่งโปรแกรมที่เขียนจะค่อนข้างซับซ้อนและใช้เวลาในการพัฒนาค่อนข้างยาก

2.4 ภาษาชั้นสูงมาก

ภาษาชั้นสูงมากอาจเรียกอีกอย่างหนึ่งว่าภาษาในรุ่นที่ 4 (4GLs: Fourth Generation Languages) ภาษานี้เป็นภาษาที่อยู่ในระดับที่สูงกว่าภาษารุ่นที่ 3 มีลักษณะของภาษาในรุ่นที่เป็น

ธรรมชาติคล้าย ๆ กับภาษาพูดของมนุษย์จะช่วยในเรื่องของการสร้างแบบฟอร์มบนหน้าจอเพื่อจัดการเกี่ยวกับข้อมูลรวมไปถึงการออกรายงานซึ่งจะมีการจัดการที่ง่ายมากไม่ยุ่งยากเหมือนภาษารุ่นที่ 3 ตัวอย่างของภาษาในรุ่นที่ 4 เช่น SQL, Informix-4GL, Focus, Sybase, InGres เป็นต้น

2.5 ภาษาธรรมชาติ

ภาษาธรรมชาติ (Natural Language) เป็นภาษาในยุคที่ 5 ที่มีรูปแบบเป็นแบบไม่เป็นโครงสร้าง (Nonprocedural) เช่นเดียวกับภาษารุ่นที่ 4 ภาษาธรรมชาตินี้ถูกสร้างขึ้นมาจากเทคโนโลยีทางด้านระบบผู้เชี่ยวชาญ (ES: Expert System) ซึ่งเป็นงานที่อยู่ในสาขาปัญญาประดิษฐ์ (AI: Artificial Intelligence) ในการที่พยายามทำให้คอมพิวเตอร์เปรียบเสมือนกับเป็นผู้เชี่ยวชาญคนหนึ่งที่สามารถคิดและตัดสินใจได้เช่นเดียวกับมนุษย์ การที่เรียกว่าภาษาธรรมชาติเพราะมนุษย์สามารถใช้ภาษาพูดป้อนเข้าไปในคอมพิวเตอร์ได้โดยตรงซึ่งอาจมีรูปแบบที่ไม่แน่นอนตายตัวแล้วคอมพิวเตอร์ก็จะแปลคำสั่งเหล่านั้นให้อยู่ในรูปแบบที่คอมพิวเตอร์เข้าใจ ถ้าคำถามใดไม่กระจ่างก็จะมีการถามกลับเพื่อให้เข้าใจคำถามเมื่อเข้าใจคำถามแล้วคอมพิวเตอร์ก็จะสามารถตอบคำถามของมนุษย์ได้อย่างถูกต้องพร้อมทั้งมีข้อแนะนำต่าง ๆ เพื่อช่วยในการตัดสินใจของมนุษย์ได้อีกด้วย

**ตารางที่ 1.1** ตารางเปรียบเทียบความแตกต่างของภาษาระดับต่ำและภาษาระดับสูง

| ภาษาระดับต่ำ | ภาษาระดับสูง |
|---|---|
| ภาษาระดับต่ำ เป็นภาษาที่ยังใกล้เคียงกับภาษาเครื่องมาก ภาษานี้ยังใช้สัญลักษณ์ต่าง ๆ แทนตัวเลขฐานสองซึ่งยุ่งยากอย่างไรก็ตามภาษานี้มีเพียงภาษาเดียว คือ ภาษาแอสเซมบลี้ เมื่อคอมพิวเตอร์รับคำสั่งภาษาแอสเซมบลี้เข้าไปแล้ว ก็จะต้องส่งไปให้ตัวแปลที่มีชื่อว่า แอสเซมเบลอร์ (Assembler) ถอดรหัสให้เสียก่อนคอมพิวเตอร์ก็จะเข้าใจ โปรแกรมที่เขียนส่งเข้าไปให้ตอนแรก เรียกว่า โปรแกรมดิบ (Source Program) และโปรแกรมที่แปลเป็นภาษาเครื่องแล้ว เรียกว่า โปรแกรมผล (Object Program) การเขียนคำสั่งเป็นภาษาแอสเซมบลี้นั้น นับว่าง่ายขึ้นมากแล้ว การพัฒนาก้าวต่อไปถึงขั้นที่ให้ผู้ทำโปรแกรมกำหนดที่เก็บข้อมูลได้เองและไม่ต้องจำว่าทีเก็บอยู่ในส่วนใด เช่น ถ้าสั่งให้ตำแหน่งที่เก็บที่ชื่อ CREDIT เก็บค่านั้นค่านี้ไว้ ต้องการเรียกค่านั้นใช้ ก็เรียกด้วยคำ CREDIT โดยไม่ต้องสนใจจำว่าอยู่ส่วนไหนของ | ภาษาระดับสูงเป็นภาษาที่ใช้ง่ายขึ้นกว่าภาษาสัญลักษณ์ โดยผู้คิดค้นภาษาได้ออกแบบ คำสั่ง ไวยากรณ์ และกฎเกณฑ์ต่าง ๆ ออกมาให้รัดกุม และจำได้ง่าย ภาษาระดับสูงนี้ยังอาจจะแบ่งออกได้เป็นหลายประเภท ดังต่อไปนี้<br>1) ประเภทที่เหมาะกับงานวิทยาศาสตร์และวิศวกรรมศาสตร์ได้แก่ ภาษา Fortran ภาษา BASIC ภาษา PASCAL ภาษา C<br>2) ประเภทที่เหมาะกับงานธุรกิจได้แก่ ภาษา COBOL ภาษา RPG<br>3) ประเภทที่เหมาะกับงานควบคุมเครื่องคอมพิวเตอร์เองได้แก่ ภาษา C โปรแกรมที่จัดทำขึ้นโดยใช้ภาษาระดับนี้ก็เช่นเดียวกับโปรแกรมภาษาสัญลักษณ์คือ จะต้องใช้ตัวแปลภาษาแปลให้เป็นโปรแกรมภาษาเครื่องก่อน คอมพิวเตอร์จึงจะเข้าใจและทำงานให้ได้ |

| | |
|---|---|
| หน่วยความจำ การสั่งงานก็ง่ายขึ้น ความผิดพลาดก็น้อยลง คนหันมาให้ความสนใจ ใช้งานคอมพิวเตอร์มากขึ้นทุกที<br>ภาษาสัญลักษณ์ (Assembly Language) เป็นภาษาที่ใช้คำ หรือตัวอักษรที่มีความหมายแทนลำดับของเลข 0 กับ 1 ของภาษาเครื่องทำให้จำคำสั่งได้ง่ายขึ้น แต่การสั่งงานเครื่องยังคงต้องสั่งอย่างละเอียดเหมือนการใช้ภาษาเครื่อง อย่างไรก็ตามก่อนใช้งานจำเป็นจะต้องใช้คอมพิวเตอร์แปลโปรแกรมที่เขียนเป็นภาษาสัญลักษณ์นี้เป็นโปรแกรมภาษาเครื่องก่อน | ภาษาระดับสูง ได้แก่<br>1. ภาษาเบสิค(BASIC) ย่อมาจาก Beginnig's All Purpose Symbolic Instruction Code) เป็นภาษาที่นิยมมากที่สุดภาษาหนึ่ง ส่วนมากใช้กับมินิและไมโครคอมพิวเตอร์ เพราะสื่อสารโต้ตอบได้ทันที (Interactive Language) การเขียนค่อนข้างง่าย การแก้ไขโปรแกรมก็สะดวก<br>2. ภาษาฟอร์แทรน (FORTRAN) คำนี้ย่อมาจาก Formular Translator เป็นภาษาเพื่อใช้ในการคำนวณทางวิทยาศาสตร์ ภาษานี้เหมาะกับงานคำนวณมาก จึงเป็นที่นิยมในกลุ่มวิศวกร นักสถิติและนักวิจัย ในการคำนวณจะมีฟังก์ชันต่างๆ ไว้ให้เรียกใช้ได้เต็มที่ ไม่สามารถสั่งพิมพ์ผลหรือรายงานได้ดีเหมือนภาษาโคบอล<br>3. ภาษาโคบอล (COBOL) คำนี้ย่อมาจาก Common Business Oriented Language) ซึ่งผู้คิดเฉพาะเจาะจงให้ใช้กับงานประเภทธุรกิจ ปัจจุบันเป็นที่นิยมมากในงานธุรกิจใหญ่ๆ<br>4. ภาษาพีแอลวัน (PL/I) ย่อมาจาก (Programming Language/One) จัดได้ว่าใช้กับงานประเภทใดก็ได้ ผิดกับฟอร์แทรนและโคบอล ซึ่งเหมาะกับงานคนละอย่าง เป็นภาษาที่ค่อนข้างยากและซับซ้อน มีการใช้เครื่องหมาย ; (Semicolon) ตอนจบคำสั่งทุกคำสั่ง<br>5. ภาษาปาสคาล (Pascal Language) ได้รับการปรับปรุงมาจากภาษาอัลกอร์ (ALGO) เหมาะกับงานทางด้านวิทยาศาสตร์เช่นเดียวกันชื่อ ปาสคาล มาจาก เบลส ปาสคาล นักคณิตศาสตร์ชาวฝรั่งเศสที่เป็นผู้คิดไม้บรรทัดสไลด์รูล แต่ภาษานี้เกิดขึ้นหลังจากที่ปาสคาล ไม่ได้รับความนิยมไปหลายปีแล้ว |

## วัฏจักรของการพัฒนาซอฟต์แวร์

วัฏจักรของการพัฒนาซอฟต์แวร์ หรือ Software Development Life Cycle (SDLC) เป็นโครงร่างหรือแนวทางวิธีการเพื่อใช้ทำความเข้าใจและเพื่อใช้เป็นขั้นตอนการพัฒนาระบบสารสนเทศหรือซอฟต์แวร์ ให้สำเร็จ โดยการให้มาซึ่งซอฟต์แวร์อาจจะเป็นโดยการซื้อหรือการจ้างทำหรือการพัฒนาเองก็

ได้ ซึ่งระเบียบวิธีการพัฒนาซอฟต์แวร์มีอยู่หลายวิธีการ แต่ละวิธีการมีข้อดีและข้อเสียที่แตกต่างกัน ตัวอย่างระเบียบวิธีการพัฒนาซอฟต์แวร์ที่ได้รับความนิยม เช่น โครงสร้างแบบน้ำตก (Waterfall Model), โครงสร้างแบบก้นหอย (Spiral Model) วิธีการพัฒนาซอฟต์แวร์แบบคล่องแคล่วว่องไว (Agile Software Development) วัฏจักรของการพัฒนาซอฟต์แวร์สามารถแบ่งออกเป็น 7 ขั้นตอนดังนี้

**ขั้นตอนที่ 1.** ระบุข้อกำหนดของความต้องการ (Requirements Specification) คือขั้นตอนการกำหนดปัญหา(Define Problem Domain) ว่าต้องการพัฒนาระบบใด

**ขั้นตอนที่ 2.**การวิเคราะห์ (Analysis) คือขั้นตอนการรวบรวมข้อมูลเพื่อทำการวิเคราะห์ว่ามีปัญหาใดบ้าง ปัญหา 1 ปัญหา 2 ปัญหา 3 ... และปัญหา N ที่ต้องแก้

**ขั้นตอนที่ 3.** การออกแบบ (Design) ในขั้นตอนนี้เป็นการบอกว่าแต่ละปัญหาแก้อย่างไรที่ได้กำหนดไว้จะแก้ไขอย่างไร

**ขั้นตอนที่ 4.** พัฒนาระบบ (Implementation) เป็นการเลือกภาษาโปรแกรมแล้วก็ลงมือทำตามที่ได้ออกแบบไว้แล้ว

**ขั้นตอนที่ 5.** การทดสอบระบบ (Testing) เป็นขั้นตอนที่ต้องการให้แน่ใจว่าระบบที่สร้างนั้นสามารถใช้งานได้ตรงตามความต้องการของผู้ใช้

**ขั้นตอนที่ 6.** บำรุงรักษา (Maintenance) เมื่อทำการสร้างซอฟต์แวร์แล้วทดลองใช้งานจำเป็นต้องมีการบำรุงดูแลรักษาซอฟต์แวร์เพื่อให้มีการทำงานที่ทันสมัยอยู่อย่างต่อเนื่อง

**ขั้นตอนที่ 7.** ปรับเปลี่ยนระบบใหม่ (Retirement) เมื่อถึงกำหนดเวลาระบบก็จะถูกปรับเปลี่ยนหรือเลิกใช้งานไป

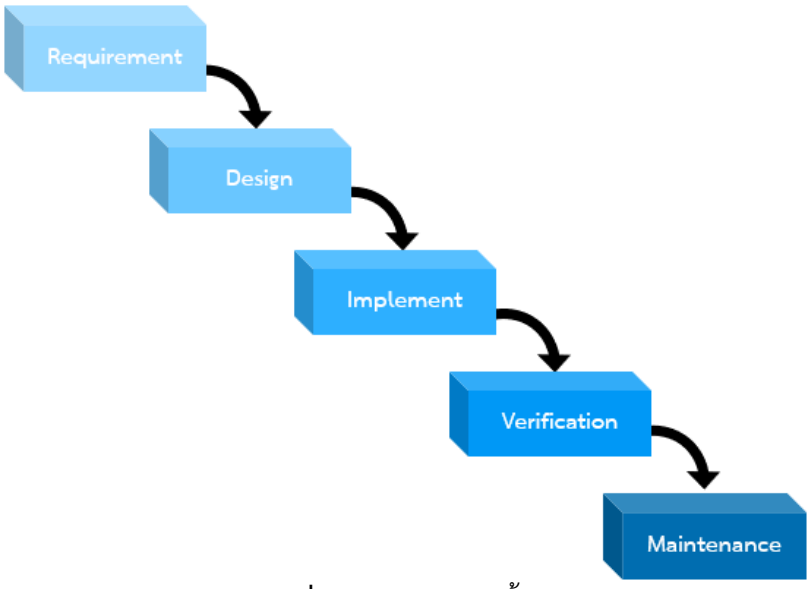
## แบบจำลองกระบวนการพัฒนาซอฟต์แวร์

แบบจำลองกระบวนการพัฒนาซอฟต์แวร์ (Software Process Model) หมายถึงการจำลองภาพของกระบวนการผลิตซอฟต์แวร์ เพื่อให้เห็นถึงการจัดโครงสร้างลำดับขั้นตอนของกระบวนการในรูปแบบที่แตกต่างกันออกไป โดยแบบจำลองกระบวนการพัฒนาซอฟต์แวร์ เป็นการนำเสนอกระบวนการผลิตซอฟต์แวร์ในแบบนามธรรม ดังนั้นรายละเอียดของกระบวนการผลิตในแบบจำลองกระบวนการพัฒนาซอฟต์แวร์จึงเป็นรายละเอียดเพียงส่วนหนึ่งเท่านั้น ซึ่งแบบจำลองการพัฒนาซอฟต์แวร์จะประกอบด้วยจำนวนชุดของขั้นตอนในกระบวนการพัฒนาซอฟต์แวร์แบบจำลองกระบวนการพัฒนาซอฟต์แวร์โดยทั่วไปมีพื้นฐานมาจากแบบจำลอง 2 รูปแบบ คือ Waterfall Model และ Evolution Model ซึ่งแต่ละตัวแบบจะแตกต่างกันในการพัฒนาโดยแต่ละตัวแบบมีจุดเด่นจุดด้อยหรือข้อดีข้อเสียแตกต่างกันไปดังนี้

1. Waterfall Model เป็นแบบจำลองที่ประกอบไปด้วยขั้นตอน 5 ขั้นตอน ได้แก่
    1) การกำหนดความต้องการ (Requirement Definition)
    2) การออกแบบซอฟต์แวร์และระบบ (System and Software Design)
    3) การลงมือทำและทดสอบระดับหน่วย (Implementation and Unit Testing)

4) การประสานระบบและทดสอบระบบ (Integration and System Testing)

5) การนำไปใช้และบำรุงรักษา (Operation and Maintenance)




**ภาพที่ 1.5** แบบจำลองน้ำตก

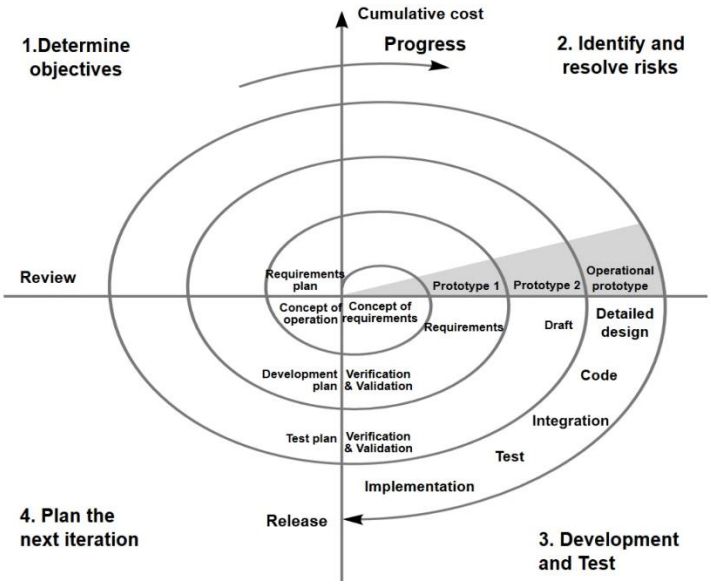
2. แบบจำลองสนซ้ำ (Evolution Model หรือ Iterative Model) เป็นแบบจำลองที่มีการทำกิจกรรมในลักษณะวนซ้ำ (Iterative) โดยเมื่อลงมือพัฒนาซอฟต์แวร์แล้วจะมีการนำเสนอต่อลูกค้า เพื่อนำคำแนะนำและติชมมาปรับปรุง และวนกลับมาแก้ไขใหม่ได้ในรอบต่อไป ประกอบไปด้วย 4 ขั้นตอน ได้แก่

1) การวิเคราะห์ความต้องการ (Requirement Analysis)

2) การออกแบบระบบ (System Design)

3) การเขียนและทดสอบโปรแกรม (Coding and Unit Testing)

4) การประเมินระบบ (Assessment)



**ภาพที่ 1.6** แบบจำลอง Evolution Model (Iterative)

**(ที่มา :** https://www.glurgeek.com/education/spiral-development-หรอ-รู้จักไหม/**)**

## แนวคิดและหลักการเชิงวัตถุเบื้องต้น

เมื่อมองไปรอบ ๆ ตัวจะสามารถสังเกตเห็นว่ามีสิ่งของต่าง ๆ มากมายในชีวิตประจำวัน ไม่ว่าจะเป็นสิ่งที่มองเห็นและจับต้องได้ (Tangible) และสิ่งที่มองไม่เห็น (Intangible) ตัวอย่างของสิ่งที่มองเห็นและจับต้องได้หรือเรียกอีกอย่างหนึ่งว่ารูปธรรมเช่น คอมพิวเตอร์ โต๊ะ เก้าอี้ โทรทัศน์ รถยนต์แมว สุนัข เป็ด ไก่เป็นต้น ส่วนตัวอย่างของสิ่งที่มองไม่เห็นหรือจับต้องไม่ได้หรือนามธรรมเช่น กฎเกณฑ์ หรือกฎหมาย (ที่ไม่ใช่รูปเล่ม) ราคาสินค้า อากาศเวลาความรู้ทฤษฎีต่าง ๆ ภาพยนตร์ (ในที่นี้หมายถึงตัวเนื้อหาของเรื่อง) เป็นต้น (กิตติพงษ์ กลมกล่อม, 2552)

แนวคิดเชิงวัตถุ (Object Oriented) คือ การใช้หรือยึดถือวัตถุเป็นตัวหลักสำหรับพิจารณาความเป็นจริงต่าง ๆ ที่เกิดขึ้นบนโลกนี้ โดยมองทุกสิ่งทุกอย่างในโลกให้เป็นวัตถุทั้งหมด และมองว่ากิจกรรมที่เกิดขึ้นในโลกนี้ล้วนเกิดจากความสัมพันธ์ (Relationship) และปฏิสัมพันธ์ (Interaction) ระหว่างวัตถุ

**1) แนวคิดเกี่ยวกับการโปรแกรมเชิงวัตถุ**

ในปัจจุบันการเขียนโปรแกรมเชิงวัตถุ ได้เข้ามาแทนที่รูปแบบการเขียนโปรแกรมแบบดั้งเดิมหรือแบบโครงสร้าง การเขียนโปรแกรมในรูปแบบเดิมนั้นมีจุดบกพร่องหรือข้อเสียหลายประการโดยการเขียนโปรแกรมแบบใหม่สามารถแก้ไขจุดบกพร่องเหล่านั้นได้ซึ่งต่อไปนี้จะเปรียบเทียบความแตกต่างระหว่างการเขียนโปรแกรมภาษารูปแบบเดิมกับรูปแบบใหม่

**2) ความหมายของวัตถุ**

วัตถุ (Object) คือสิ่งต่าง ๆ ที่พบได้ในชีวิตประจำวันอาจจะเป็น คน สัตว์ หรือสิ่งของ ซึ่งคนสามารถรับรู้ได้ 2 อย่างเกี่ยวกับวัตถุคือ 1. แอตทริบิวต์ (Attributes) และ 2. ฟังก์ชัน (Functions) ของวัตถุที่สามารถกระทำได้ เช่น ปากกา มีหน้าที่ไว้เขียนข้อความหรือตัวอักษร ตู้เย็น ไว้สำหรับทำความเย็นให้อาหาร เป็นต้นเมื่อพิจารณาดูภายในองค์กรหรือบริษัทต่าง ๆ จะพบว่ามีกิจกรรมต่าง ๆ เกิดขึ้นมากมาย เช่น การดำเนินการของบริษัทกิจกรรมการขายสินค้า การผลิตสินค้า การซื้อสินค้าหรือการดำเนินการของคน กิจกรรมที่เกี่ยวข้องกับเงินและการบัญชีตัวที่ทำให้เกิดเหตุการณ์ขึ้นสิ่งต่าง ๆ เหล่านี้เรียกว่า “วัตถุ” โดยคำว่าวัตถุจึงได้รับการใช้ในสถานะต่าง ๆ กัน และอาจมีความหมายที่คลาดเคลื่อนกันบ้าง อย่างไรก็ดีได้มีการกำหนดความหมายของคำว่า “วัตถุ” ไว้ว่า เป็นตัวที่ทำให้เกิดเหตุการณ์ที่มีข่าวสารและให้คุณลักษณะบางอย่าง (กิติ ภักดีวัฒนะกุล, และกิตติพงษ์ กลมกล่อม, 2544)

ให้พิจารณาตัวอย่างต่อไปนี้

1. บัญชีธนาคารเป็นวัตถุในธนาคาร สำนักงานก็เป็นวัตถุในธนาคาร และเช่นเดียวกันลูกค้าก็เป็นวัตถุ

2. นโยบายการประกันภัยก็เป็นวัตถุที่อยู่ในบริษัทประกันภัย และเช่นเดียวกันธนาคาร ตัวสำนักงาน และลูกค้าก็มีอยู่ด้วยและเป็นวัตถุด้วย

3. รถยนต์ก็เป็นวัตถุที่อยู่ในหน่วยทะเบียนรถยนต์กลาง รถยนต์ก็ยังเป็นวัตถุของระบบการผลิตในโรงงานรถยนต์

จากตัวอย่างที่ยกมาให้ดูนี้เห็นได้ชัดว่า ในการประยุกต์ใช้งานหรือในสิ่งแวดล้อมหนึ่งอาจมีหลาย ๆ วัตถุ และในอีกสถานการณ์หนึ่งก็มีวัตถุที่มีลักษณะคล้ายกัน

ลูกค้าของธนาคารกับลูกค้าของบริษัทประกันภัยจัดว่าเป็นวัตถุวัตถุลูกค้าในบริษัทประกันภัยหรือไม่ ทำนองเดียวกันจะเห็นว่ารถยนต์เป็นวัตถุของหน่วยขึ้นทะเบียนกลาง หรือกรมการขนส่งทางบกและก็เป็นวัตถุของบริษัทผลิตรถยนต์ด้วย วัตถุของรถยนต์ในสองสถานการณ์นี้เหมือนกันหรือไม่คำตอบที่เด่นชัดคือไม่เหมือนกัน ในโดเมนของสิ่งแวดล้อมต่างกัน การมองที่วัตถุจะต่างกัน ลูกค้าทั้งของธนาคารและบริษัทประกันภัยอาจมีชื่อ นามสกุลที่อยู่ที่ติดต่อเหมือนกัน แต่อาจจะมีบางส่วนที่ความต้องการของธนาคารแตกต่างจากบริษัทประกันภัย ลองพิจารณาที่รถยนต์จะเห็นเด่นชัด คือรถยนต์ที่เป็นวัตถุอยู่ในส่วนของโรงงานผลิตรถยนต์ บริษัทผู้ผลิตสนใจว่ารถยนต์นั้นจะขายให้ผู้ใด ใครคือผู้ซื้อ วันที่ผลิต สีรถ สิ่งที่ลูกค้าสั่งให้เพิ่มเติม เฉพาะในรถยนต์ตามความต้องการของลูกค้า ส่วนหน่วยทะเบียนของกรมการขนส่งทางบกมองวัตถุรถยนต์ว่าใครเป็นเจ้าของ วันที่ซื้อขาย เสียภาษีรถยนต์แล้วหรือยัง ตลอดจนข้อมูลที่เกี่ยวกับการจดทะเบียนรถยนต์

ตามที่ได้รู้มาบ้างแล้วว่าในโลกของมีสิ่งต่าง ๆ มากมายสิ่งที่เกิดขึ้นจากวัตถุต่าง ๆ ก็คือ กิจกรรม (Activities) ความเคลื่อน (Movement) หรือการกระทำ (Action ) เช่นคนรับประทานอาหารสุนัขเล่นกับแมวนักศึกษาเล่นเกมออนไลน์นักกีฬาทีมชาติไทยลงแข่งขันกีฬาชกมวยในการแข่งขันกีฬาโอลิมปิกหากพิจารณาโดยละเอียดแล้วจะพบว่า กิจกรรมต่าง ๆ ที่เกิดขึ้นในชีวิตประจำวันของนั้นล้วนแต่เกิดจากการมีความสัมพันธ์และการมีปฏิสัมพันธ์ระหว่างวัตถุสองตัวขึ้นไป ซึ่งจากข้อความตัวอย่างข้างต้นจะพบว่า

ตัวอย่างที่ 1.1 กิจกรรมคนรับประทานอาหาร เกิดจาก Interaction “รับประทาน”ระหว่างคนและอาหาร และเกิดจาก Relationship “เป็นเจ้าของ” ระหว่างคนและอาหาร(เพราะคนเป็นเจ้าของอาหาร จึงจะสามารถรับประทานได้)

ตัวอย่างที่ 1.2 นักมวยชกต่อยคู่ต่อสู้ เกิดจากปฏิสัมพันธ์ “ชกต่อย” ระหว่าง นักมวยและคู่ต่อสู้ และเกิดจากความสัมพันธ์ “คู่ชก” ระหว่างคนและคู่ต่อสู้

ตัวอย่างที่ 1.3 รถกำลังวิ่งไปบนถนนเกิดจากปฏิสัมพันธ์ “วิ่งไปบน” ระหว่าง รถและถนน และเกิดจาก ความสัมพันธ์ “การใช้งาน” ระหว่างรถและถนน

ตัวอย่างที่ 1.4 อาจารย์สอนหนังสือนักศึกษาเกิดจากปฏิสัมพันธ์ “สอน” ระหว่าง อาจารย์และนักศึกษา และเกิดจากความสัมพันธ์ “อาจารย์-ลูกศิษย์”

**3) การพัฒนาระบบโดยใช้หลักการเชิงวัตถุ**

การพัฒนาระบบเชิงวัตถุ จะประกอบด้วยกลุ่มของวัตถุต่าง ๆ ที่ทำงานร่วมกันโดยแบ่งบทบาทหน้าที่ความรับผิดชอบ ซึ่งใช้หลักการจัดแบ่งประเภทของวัตถุในลักษณะทางนามธรรม (Abstract) ออกเป็นกลุ่ม ๆ ที่เรียกว่าคลาส (Class) แต่ละคลาสก็จะมีสถานะ (State) รวมทั้งพฤติกรรม (Behavior) ตามบทบาทของตน โดยมีข้อมูลรายละเอียดหรือคุณสมบัติ (Characteristic) ที่เก็บซ่อน (Encapsulate) ในคลาสของตนโดยไม่มีการปะปนกับคลาสอื่น ๆ แต่ในการติดต่อสื่อสารหรือการร้องขอใช้บริการก็สามารถติดต่อสื่อสารกันได้ด้วยข่าวสาร (Message) แนวคิดเชิงโครงสร้างนั้นเป็นโครงสร้างที่โปรแกรมกับข้อมูลนั้นแยกออกจากกัน แต่แนวคิดเชิงวัตถุนั้นจะมองเป็นวัตถุหนึ่งที่เป็นแหล่งรวบรวมข้อมูลวิธีการโดยมีคลาสเป็นตัวกำหนดคุณสมบัติของวัตถุนั้น ซึ่งคุณสมบัติยังสามารถทำการรับทอด (Inheritance) ในลักษณะซับคลาส (Subclass) ต่าง ๆ ดังนั้นหากมีคลาสที่เป็นต้นแบบที่ดีอยู่แล้วผู้พัฒนาก็สามารถนำคุณสมบัติของคลาสต้นแบบนั้นมาใช้งานได้ทันที ซึ่งเป็นการนำกลับมาใช้ใหม่ (Reusable) ทำให้ช่วยลดเวลาในการพัฒนาและลดค่าใช้จ่ายและมีความมั่นใจในคลาสต้นแบบที่ใช้งานมานานจะบ่งบอกถึงความถูกต้องซึ่งก่อให้เกิดความผิดพลาดได้น้อย (ญาดา เขื่อนใจ, 2554)

จึงสามารถกล่าวโดยสรุปว่าการวิเคราะห์และออกแบบเชิงวัตถุนี้ เป็นแนวคิดที่พยายามจัดระบบกระบวนการพัฒนาระบบงานให้มีระเบียบ และสามารถนำโปรแกรมที่เคยเขียนมาก่อนให้สามารถกลับมาใช้งานใหม่ ซึ่งถ้าเปรียบเทียบกับการเขียนโปรแกรมเชิงโครงสร้าง ถึงแม้ระบบงานที่มีความใกล้เคียงกันแต่โมดูลที่จะนำมาใช้งานก็จะต้องมีการปรับเปลี่ยนมากมายแทบจะเริ่มต้นเขียนใหม่ทั้งหมด เป็นเพราะแนวทางการพัฒนาซอฟต์แวร์เชิงโครงสร้าง (Structural Programming) นั้นมีลักษณะเป็นนามธรรม ซึ่งเกิดจากการจินตนาการ ดังนั้นระบบงานที่พัฒนาตามแนวคิดเชิงโครงสร้างในแต่ละระบบก็จะเกิดจากการจินตนาการของแต่ละบุคคลจินตนาการของแต่ละบุคคลก็มีแนวคิดที่ต่างกัน ดังนั้นจึงเห็นซอฟต์แวร์จำนวนมากมาย ทั้งที่เป็นระบบเดียวกันก็อาจจะไม่สามารถนำกลับมาใช้ใหม่ได้ทั้งหมด

**4) เทคนิคที่ใช้ในการพัฒนาระบบตามหลักการเชิงวัตถุ**

ในการพัฒนาระบบตามหลักการเชิงวัตถุมีการนำเทคนิคหรือวิธีการต่าง ๆ มาใช้ ดังนี้

1.1 มุมมองใหม่ของซอฟต์แวร์ดิคอมโพสิชัน (Software Decomposition) คือการพิจารณาข้อมูลและฟังก์ชันหรือพฤติกรรมรวมกันเป็นวัตถุ

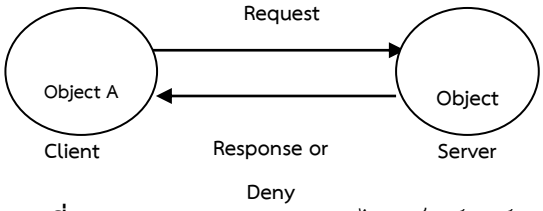
1.2 เสนอแนวทางใหม่ในการสังเคราะห์ (Synthesis) คือพิจารณาแนวทางการสังเคราะห์ วิธีการแก้ปัญหาโดยมุ่งเน้นที่ข้อมูลและฟังก์ชัน โดยรวมเป็นหน่วยเดียวกันคือวัตถุโดยแยกส่วนของข้อมูลและฟังก์ชันที่ไม่เกี่ยวข้องไว้ในรูปของการเรียกใช้โมดูล

1.3 พิจารณาขอบเขตของปัญหา (Problem Domain) และสร้างแบบจำลองการแก้ปัญหาตามสภาพความเป็นจริง

1.4 พิจารณาซอฟต์แวร์ในรูปแบบของไคลเอ็นท์ (Client) และเซิร์ฟเวอร์ (Server)

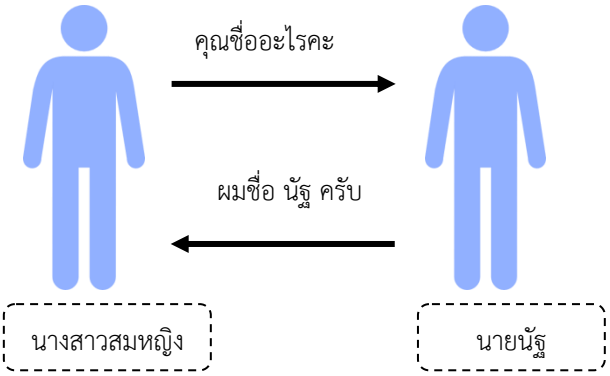
1.5 เซิร์ฟเวอร์ซ่อนรายละเอียดของการกระทำไว้ภายใต้อินเตอร์เฟส (Interface)

1.6 ไคลเอนต์เรียกใช้ได้เฉพาะอินเตอร์เฟสที่เซิร์ฟเวอร์เปิดให้บริการเท่านั้น




**ภาพที่ 1.7** แสดงการทำงานแบบไคลเอ็นท์-เซิร์ฟเวอร์

จากภาพที่ 1.7 เป็นการจำลองการทำงานร่วมกันของวัตถุในลักษณะของการทำงานแบบไคลเอ็นท์เซิร์ฟเวอร์ ซึ่งสามารถอธิบายได้ว่าวัตถุ A ร้องขอไปยังวัตถุ B เมื่อวัตถุ B ได้รับข่าวสารและมีทำงานเสร็จแล้วส่งผลลัพธ์ที่ได้กลับมายังวัตถุ Aตัวอย่างที่พบในชีวิตประจำวันได้แก่

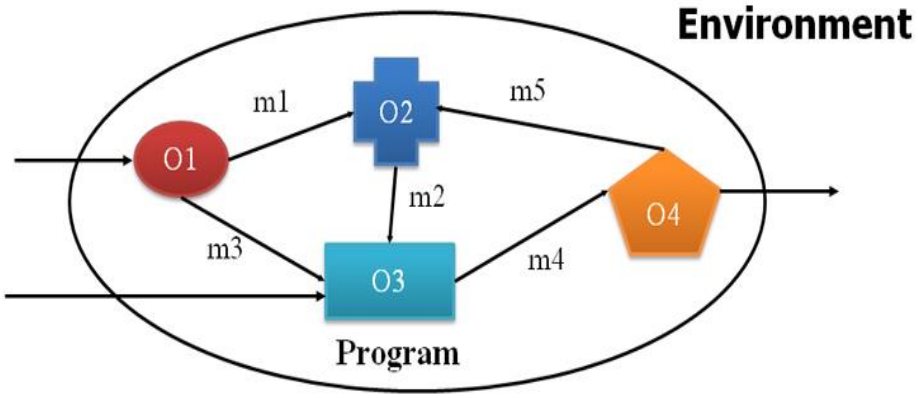



**ภาพที่ 1.8** แสดงการสื่อสารกันของวัตถุ

จากภาพที่ 1.8 เป็นการสื่อสารกันระหว่างวัตถุโดยนางสาวแป้งต้องการทราบชื่อของอีกคนก็เลยใช้เมธอดถามชื่อ ส่วนอีกวัตถุหนึ่งก็มีเมธอดสำหรับบอกชื่อเช่นเดียวกัน

**5) ซอฟต์แวร์เชิงวัตถุ**

ซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software) หมายถึง กลุ่มหรือชุด (Collection) ของวัตถุที่มีหน้าที่รับผิดชอบต่อการจัดการข้อมูลของตนเองและติดต่อสื่อสารกับวัตถุอื่น ๆ โดยทำการส่งข่าวสาร (Message) ให้แก่กันและกันเพื่อทำงานร่วมกันและเพื่อให้ได้ผลลัพธ์ที่ตรงตามที่ผู้ใช้ต้องการ (User Requirement) (Kung, D., and Lei, J., 2016) ดังแสดงในภาพที่ 1.9




**ภาพที่ 1.9** แสดงการทำงานภายในของซอฟต์แวร์เชิงวัตถุ

จากภาพที่ 1.9 คือการถึงแสดงการทำงานภายในซอฟต์แวร์เชิงวัตถุ ซึ่งจะประกอบไปด้วยวัตถุต่าง ๆ ในที่นี้หมายถึงวัตถุ O1 ถึง O4 โดยที่แต่ละวัตถุจะมีส่วนที่เก็บข้อมูลและวิธีจัดการข้อมูลของตัวเองและทำงานร่วมกันกับวัตถุอื่น ๆ ภายในระบบด้วยวิธีการส่งข่าวสารต่าง ๆ ตั้งแต่ข่าวสาร m1 ถึง m4 ให้แก่กัน

**6) ความหมายของหลักการเชิงวัตถุ**

หลักการเชิงวัตถุ คือ หลักการที่ใช้ในการพัฒนาระบบโดยการมองซอฟต์แวร์ให้เป็นชิ้นส่วนหรือวัตถุที่จะสามารถนำมาประกอบกันกลายเป็นซอฟต์แวร์ได้ หลักการเชิงวัตถุยังเป็นการจำแนกความสามารถหรือหน้าที่ของซอฟต์แวร์เพื่อจัดให้เป็นระเบียบหมวดหมู่เพื่อให้ง่ายต่อการเรียกใช้ นอกจากนี้หลักการเชิงวัตถุยังมีความสามารถใช้การติดต่อระหว่างซอฟต์แวร์ คลาสและเมธอดได้ (นัฐพงศ์ ส่งเนียม, 2563)

**7) ลักษณะที่สำคัญของวิธีการเชิงวัตถุ**

วิธีการเชิงวัตถุประกอบไปด้วยองค์ประกอบ 4 อย่าง ดังนี้คือ

1) มุมมองของวัตถุ (Abstraction)

2) การห่อหุ้มของวัตถุ (Encapsulation)

3) ลำดับชั้นของวัตถุ (Hierarchy)

4) การพ้องรูป (Polymorphism)

**8) องค์ประกอบของวัตถุ**

องค์ประกอบหลักของวัตถุมี 3 ส่วนด้วยกันดังนี้คือ 1. คุณลักษณะ (Attributes) 2. พฤติกรรมหรือความสามารถทำงาน (Behavior) และ 3. เอกลักษณ์ (Object Identity) โดยมีรายละเอียดดังต่อไปนี้

**องค์ประกอบที่ 1 คุณลักษณะ**

คุณลักษณะ (Attributes) อาจจะเรียกเป็นอย่างอื่น เช่น คุณสมบัติ (Properties) หรือข้อมูลของวัตถุ (Data) โดยจะทำหน้าที่เป็นข้อมูลที่ใช้ในคลาสเปรียบเสมือนตัวแปร (Variable) ของคลาส เช่น

คลาส Human ประกอบไปด้วยคุณลักษณะ หมายเลขบัตรประชาชน ชื่อ นามสกุล เพศ และที่อยู่ เป็นต้น

คลาส สุนัข ประกอบไปด้วยลักษณะ สี สายพันธ์ เป็นต้น

**องค์ประกอบที่ 2 พฤติกรรม**

พฤติกรรมหรือความสามารถทำงาน (Behavior) คือ การทำงานหรือความสามารถในการทำงานของคลาสต่าง ๆ เช่น

คลาส สุนัข สามารถ กินได้ เห่าได้ กัดได้

คลาส เครื่องคิดเลข สามารถเปิดได้ ปิดได้ คำนวณค่าต่าง ๆ ได้

**องค์ประกอบที่ 3 เอกลักษณ์ของวัตถุ**

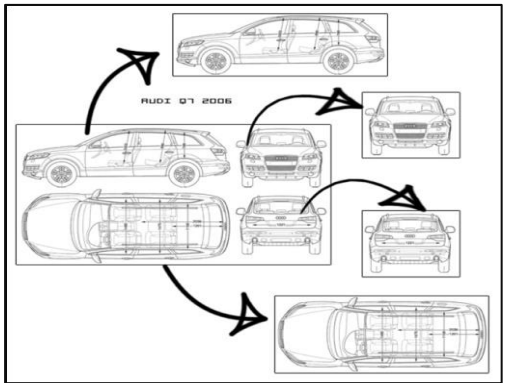
เอกลักษณ์ของวัตถุ (Object Identity) หมายถึง วัตถุแต่ละตัวที่ถูกสร้างขึ้นจากคลาสซึ่งจะมีความแตกต่างกันเสมอ เช่น แบบแปลนบ้าน สามารถนำมาสร้างบ้านได้หลายหลัง แต่บ้านแต่ละหลังไม่สามารถสร้างอยู่บนตำแหน่งเดียวกันได้ ทำให้บ้านแต่ละหลังมีความแตกต่างกันในเรื่องของตำแหน่ง เป็นต้น ซึ่งทำให้วัตถุแต่ละตัวที่ถูกสร้างขึ้นนั้นไม่มีความเกี่ยวข้องกัน ถือว่ามีเอกลักษณ์เฉพาะตัวซึ่งสามารถนำวัตถุนั้นไปปรับแต่งหรือทำงานให้แตกต่างกันหรือคนละช่วงเวลาได้

**9) ความหมายของคลาส**

คลาส หมายถึง ตัวต้นแบบที่เก็บแนวคิดทั้งหมดไว้อย่างเป็นหมวดหมู่ เป็นระเบียบ มีกฎและข้อจำกัดการเข้าถึง คลาสถือว่าเป็นนามธรรม ในขณะที่วัตถุนั้นเป็นสิ่งที่มีตัวตน (Concrete) คลาสก็คือแบบพิมพ์เขียวของวัตถุโดยที่คลาสไม่สามารถทำงานได้ ในขณะที่วัตถุสามารถทำงานได้ การทำงานของวัตถุจะเป็นไปตามคุณสมบัติที่กำหนดไว้ในคลาส และวัตถุทุกตัวก็จะต้องอยู่ในคลาส โดยสามารถดูคุณลักษณะของวัตถุได้ด้วยการดูที่คลาส ดังนั้นคลาสก็คือกลุ่มของวัตถุที่มีโครงสร้างพื้นฐานพฤติกรรมเดียวกัน โดยวัตถุที่มีคุณลักษณะเดียวกันก็จะรวมกลุ่มอยู่ในคลาสเดียวกัน จึงสรุปได้ว่าคลาสก็คือต้นแบบข้อมูลที่มีไว้เพื่อสร้างวัตถุ คลาสนอกจากจะมีชื่อคลาสกำกับแล้วยังมีแอตทริบิวต์ที่ใช้อธิบายคุณสมบัติและโอเปอเรชัน (Operation) ที่ใช้อธิบายถึงพฤติกรรมของคลาสว่ามีตัวปฏิบัติการอะไรบ้าง ซึ่งโอเปอเรชันซึ่งในภาษาเชิงวัตถุจะเรียกว่าเมธอด (Method) โดยทั้งคลาสแอตทริบิวต์ และโอเปอเรชันสามารถแสดงในลักษณะเทมเพลตได้

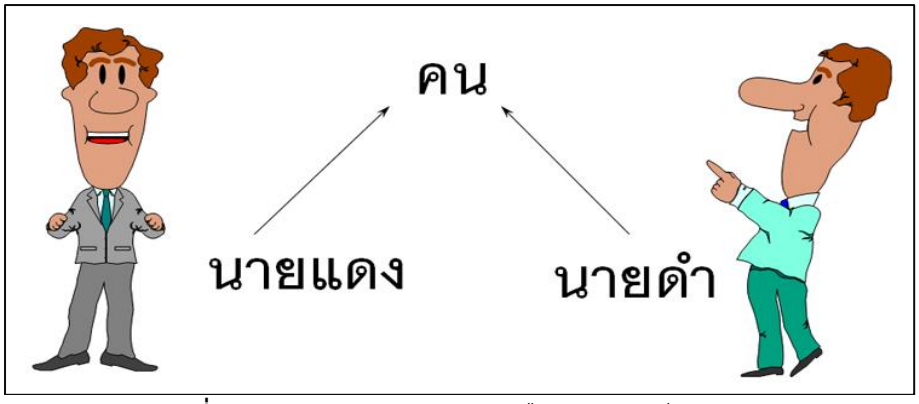
**10. ความสัมพันธ์ระหว่างคลาสและวัตถุ**

คลาสและวัตถุอาจเรียกได้ว่าคลาสเป็นพิมพ์เขียวหรือแบบแปลน (Blueprint) ของวัตถุและเรียกได้ว่าวัตถุเป็นอินสแตนซ์ของคลาสดังตัวอย่างในภาพที่ 1.10



**ภาพที่ 1.10** แสดงภาพของแม่พิมพ์หรือคลาสของรถยนต์

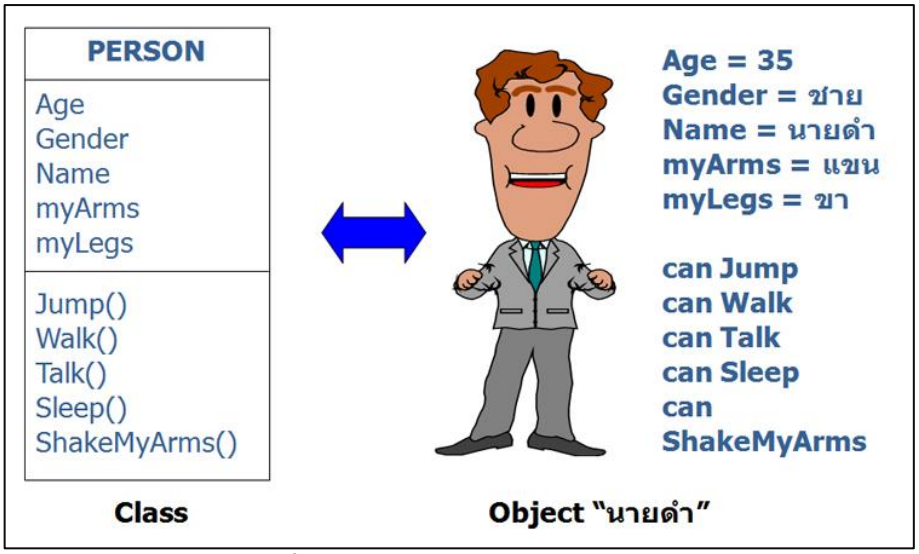
จากภาพที่ 1.10 แสดงภาพของแม่พิมพ์หรือคลาสของรถยนต์ ซึ่งเป็นการเปรียบเทียบคลาสเสมือนแม่พิมพ์นั่นเอง




**ภาพที่ 1.11** แสดงคลาสและวัตถุหรืออินสแตนซ์ของคลาส

จากภาพที่ 1.11 เมื่อพูดถึงคนก็หมายถึงทุก ๆ คนที่เป็นมนุษย์ คนจึงเป็นคลาส แต่หากระบุเจาะจงหรือกล่าวถึงชื่อบุคคลใดบุคคลหนึ่งในที่นี้ ได้แก่ นายแดง และนายดำ หมายถึง ตัวแทนของคลาสคนหรือที่เรียกว่าอินสแตนซ์ของคลาสคน หรือวัตถุนั่นเอง ดังนั้นวัตถุจึงเปรียบเสมือนตัวแทนของกลุ่มวัตถุ

วัตถุใด ๆ ที่ถูกสร้างขึ้นให้เหมือนกับคลาสแต่วัตถุจะสามารถทำงานได้เช่นเดียวกับความสามารถของคลาสเปรียบเสมือนบ้านที่ถูกสร้างขึ้นจากแบบแปลน สามารถเข้าไปอยู่อาศัยได้ จับต้องได้ เรียกการทำงานได้ดังนั้นเมื่อมองคลาสเป็นแบบแปลนบ้านก็สามารนำมาสร้างบ้านได้หลายหลัง




**ภาพที่ 1.12** แสดงแผนภาพคลาสและวัตถุ

จากภาพที่ 1.12 เป็นการอธิบายถึงรายละเอียดของส่วนประกอบของวัตถุ ที่ประกอบไปด้วย แอตทริบิวต์ พฤติกรรม และแอสโซซิเอชันต่าง ๆ ของวัตถุนายดำ

**11) คุณลักษณะที่สำคัญของวัตถุ**

อลัน คีย์(Alan key) ได้ให้นิยามที่สำคัญเกี่ยวกับวัตถุไว้ว่า สิ่งใดที่เป็นวัตถุได้จะต้องมีลักษณะที่สำคัญ ดังนี้ 1. เอกลักษณ์ (Identity) 2. การห่อหุ้ม (Encapsulation) 3. การซ่อนรายละเอียด (Information Hiding) 4. การรับทอดมรดก (Inheritance) และ 5. พอลิมอร์ฟิซึม (Polymorphism) โดยมีรายละเอียด ดังนี้

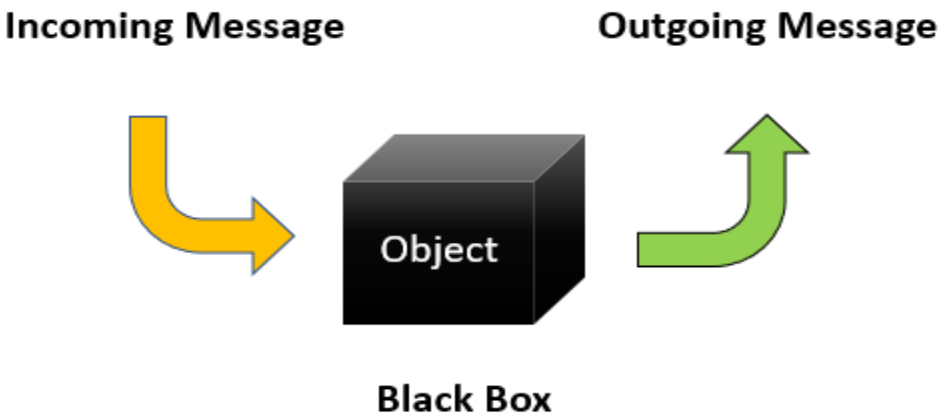
**1. เอกลักษณ์**

เอกลักษณ์ (Identity) ตามแนวคิดของอลัน เคีย์ วัตถุนั้นจะต้องมี ลักษณะเฉพาะประจำตัว (Identity) และสามารถมีคุณลักษณะเหมือน ๆ กันได้ เช่น วัตถุ “Tim” และ “Don” เป็นเพศชายและอายุ 25 ปี เหมือนกันวัตถุใด ๆ สามารถมีพฤติกรรมเหมือน ๆ กันได้ เช่น วัตถุ “Tim” และ “Don” สามารถกิน เดิน นอน นั่ง ได้เหมือนกันวัตถุใด ๆ สามารถมีคน วามสัมพันธ์เกี่ยวข้องกับวัตถุอื่น ๆ เหมือนกันได้ เช่น วัตถุ“Tim” และ“Don” ประกอบด้วย 2 แขนและ 2 ขาเหมือน ๆ กันวัตถุ “Tim” และ “Don” สร้างจากคลาสเดียวกันคือ “Person”แต่ถือว่า วัตถุ “Tim” และวัตถุ “Don” เป็นคนละวัตถุกัน

**2. การห่อหุ้ม**

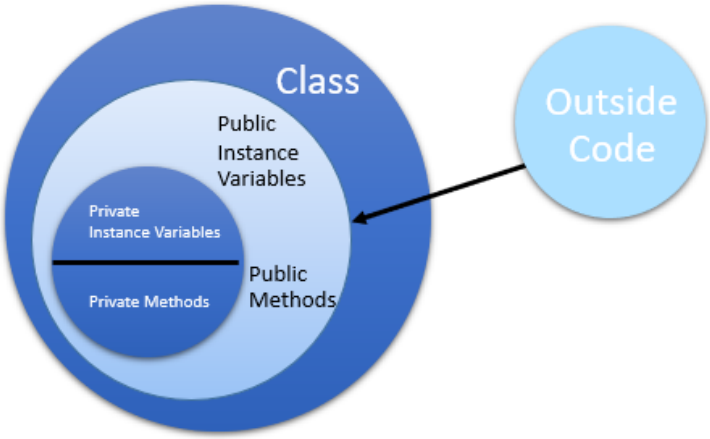
การห่อหุ้ม (Encapsulation) หมายถึง การที่วัตถุสามารถพิจารณาได้เหมือนกันกับกล่องดำ (Black Box) โดยวัตถุใด ๆ ต้องรู้วิธีหรือขั้นตอนการทำงานภายในของตนเองโดยที่วัตถุอื่นไม่จำเป็นต้องรู้

ว่าวิธีหรือขั้นตอนการทำงานภายในเป็นอย่างไรหากวัตถุอื่นต้องการเรียกใช้ฟังก์ชันของวัตถุนี้ได้ก็สามารถทำได้โดยส่งข้อความมายังวัตถุซึ่งข้อความที่ส่งมานั้นจะต้องอยู่ในรูปแบบที่ตกลงกันไว้ก่อน

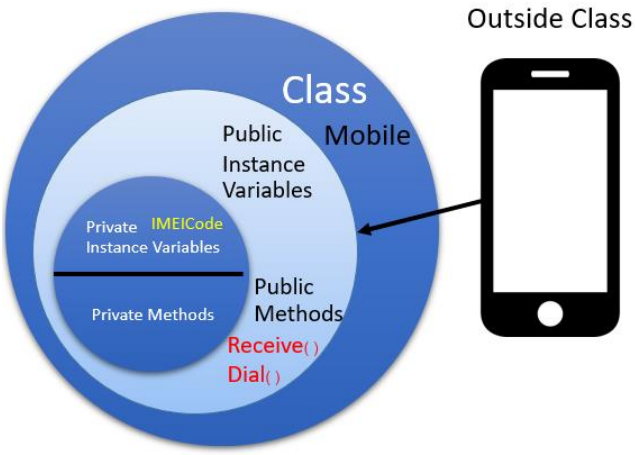



**ภาพที่ 1.13** แทนหลักการห้อหุ้มด้วยกล่องดำ

จากภาพที่ 1.13 เป็นการเปรียบเทียบการทำงานในลักษณะห่อหุ้มเหมือนกับการทำงานของกล่องดำ ซึ่งหมายถึงมีข้อมูลเข้าไปแล้วมีการทำงานบางอย่างภายในจนได้ข้อมูลออกมา ดังตัวอย่างของการห่อหุ้ม ในภาพที่ 1.14




**ภาพที่ 1.14** แสดงการห้อหุ้มและมุมมองจากภายนอก




**ภาพที่ 1.15** แสดงการห้อหุ้มและมุมมองจากภายนอกของคลาสโทรศัพท์มือถือ

จากภาพที่ 1.15 สามารถอธิบายได้ว่า โทรศัพท์มือถือมีการห่อหุ้มจากภายนอกโดยผู้ใช้ไม่จำเป็นต้องรู้การทำงานภายในของอุปกรณ์ เพียงแค่สามารถใช้งานในสิ่งที่จำเป็น เช่น โทรออก รับสาย หรือส่งข้อความสั้น ได้ก็เพียงพอแล้ว

**3. การซ่อนรายละเอียด**

การซ่อนรายละเอียด (Information hiding) เป็นหลักการพื้นฐานของการปกปิดข้อมูลภายในและวิธีการทำงานของวัตถุ โดยคำว่าเอ็นแคบซูเลต มีความหมายว่าผู้ใช้ไม่สามารถเห็นรายละเอียดได้แก่ข้อมูลและฟังก์ชันภายในได้เนื่องจากถูกซ่อนและบรรจุไว้ในแคปซูล แต่สามารถใช้งานวัตถุได้ด้วยเมธอด กล่าวคือ ในการเข้าถึงข้อมูลนั้นจะไม่สามารถเข้าถึงได้โดยตรง จะต้องมีการตอบรับจากเมธอดในวัตถุปลายทางนั้นว่าจะอนุญาตหรือไม่ที่จะให้วัตถุที่ส่งข่าวสารมาร้องขอเพื่อเข้าถึงข้อมูล (Shin, S., 2015)

ในยูเอ็มแอล (UML) จะมีการกำหนดให้การมองเห็น(Visibility) แอตทริบิวต์หรือเมธอดเป็น Public Protected หรือ Private ซึ่งจะใช้สัญลักษณ์ดังต่อไปนี้

**+** คือ สัญลักษณ์ Public Visibility

**-** คือ สัญลักษณ์ Private Visibility

**#** คือ สัญลักษณ์ Protected Visibility

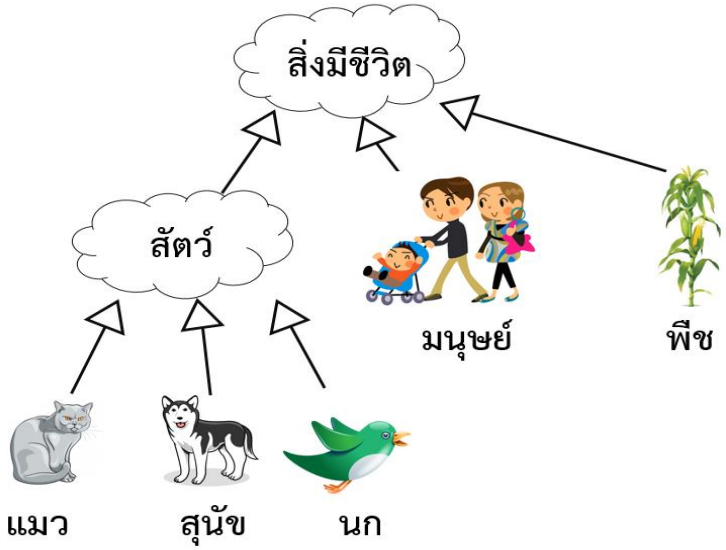
หมายถึง การพิจารณาว่าวัตถุนั้นสามารถกำหนดขอบเขตการเข้าถึงทั้งแอตทริบิวต์และเมธอดว่าคลาสใดสามารถมองเห็นหรือเข้าถึงแอตทริบิวต์และเมธอดของวัตถุนั้นได้บ้าง โดยประโยชน์ของหลักการซ่อนรายละเอียดและหลักการห่อหุ้มของวัตถุ มีดังนี้

1) Maintainability เป็นการพิจารณาโครงสร้างซอฟต์แวร์ในระดับนามธรรม (Abstract) หรือในระดับสูง โดยไม่คำนึงถึงรายละเอียดวิธีการหรือภาษาที่จะนำไปพัฒนา (Implementation)

2) มีความยืดหยุ่น (Flexibility) ทำให้การพัฒนาซอฟต์แวร์เชิงวัตถุมีความยืดหยุ่นโดยสามารถเปลี่ยนแปลงได้ง่าย (Ease of Change) ระบบซอฟต์แวร์ใด ๆ สามารถปฏิบัติงานร่วมกันได้อย่างสะดวกและมีประสิทธิภาพเนื่องจากไม่จำเป็นต้องรู้ถึงรายละเอียดการทำงานภายในของระบบ

**4. การรับทอดมรดก**

การรับทอดมรดก (Inheritance) คือ การกำหนดคุณสมบัติของวัตถุแต่ละตัวในระบบ จะใช้วิธีการรับทอดโดยอาศัยคุณสมบัติของวัตถุที่มีอยู่แล้วใส่ลงในวัตถุตัวใหม่ แนวความคิดเชิงวัตถุจะถือว่าการรับทอดเป็นกลไกที่สำคัญ เพราะว่าไม่มีสิ่งใดในโลกที่เกิดขึ้นเองจำเป็นจะต้องมีการรับทอด (Shin, S., 2016) โดยข้อดีของการรับทอดมรดกมีหลายประการด้วยกัน เช่น การทำให้มีโครงสร้างเป็นระบบระเบียบ สามารถปรับเปลี่ยนได้ง่าย ลดเวลาในการพัฒนาระบบ และลดค่าใช้จ่ายในการพัฒนาระบบอีกด้วย




**ภาพที่ 1.16** ตัวอย่างการรับทอดมรดก

จากภาพที่ 1.16 เป็นตัวอย่างการรับทอดมรดกจากคลาสพ่อแม่มายังคลาสลูก หรือกล่าวอีกนัยหนึ่งได้ว่า นก สุนัข แมว จัดว่าเป็นสัตว์ ในขณะที่ สัตว์ มนุษย์ และพืช ก็ถือว่าเป็นสิ่งมีชีวิตเหมือนกัน

**5. พอลิมอร์ฟิซึม**

พอลิมอร์ฟิซึม (Polymorphism) คือ ความสามารถในการติดต่อสื่อสารที่ทำให้มีการตอบสนองที่หลากหลายได้ หรือ อีกนัยหนึ่งเรียกว่าปรับรูปร่างได้

เช่น คลาส Human มี เมธอดชื่อว่า

Eat(“Meat”)

Eat(“Vegetable”)

Eat(“Water”)

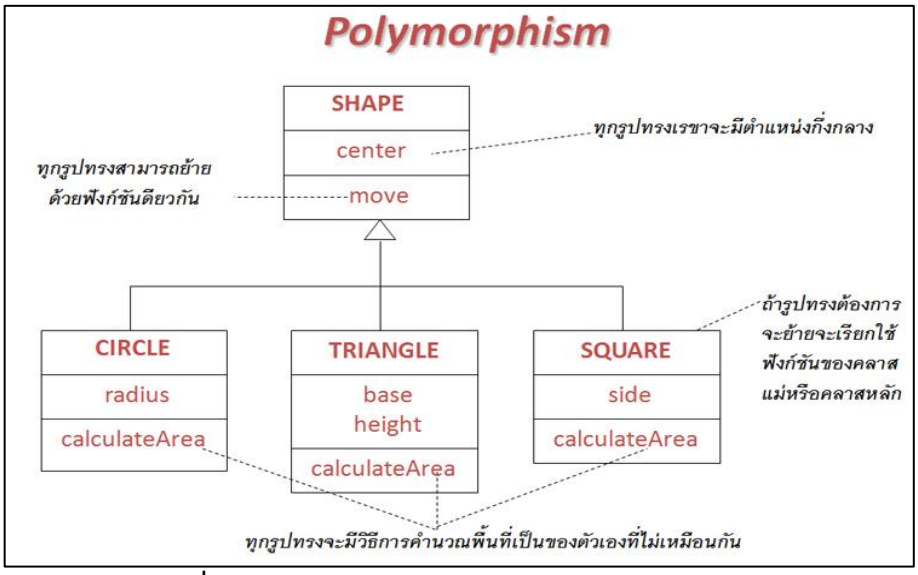
ดังนั้นเมื่อทานเนื้อก็จะไปสู่ระบบการย่อยอีกลักษณะ เมื่อทานผักก็จะไปสู่ระบบการย่อยและการดูซึมที่ต่างกับการทานเนื้อ เป็นต้นนอกจากนี้ วิธีการพอลิมอร์ฟิซึม (Polymorphism) ยังทำให้สามารถ

ปรับปรุงเมธอดในคลาสได้โดยที่ไม่ได้ทำการยุ่งกับคลาสต้นฉบับ ซึ่งจำเป็นต้องใช้วิธีการรับถอด (Inheritance) (อัษฎาพร ทรัพย์สมบูรณ์, 2554)

พอลิมอร์ฟิซึมหมายถึงวัตถุต่าง ๆ ที่มีพฤติกรรมแบบเดียวกันอยู่ในตัว แต่รายละเอียดวิธีการกระทำอาจแตกต่างกันได้ขึ้นอยู่กับว่าวัตถุนั้นถูกนำไปใช้แบบใด

เช่น วัตถุ “SQUARE” และวัตถุ “CIRCLE” ต่างก็สามารถถูก “calculateArea” ได้ แต่วิธีการ “calculateArea” ของทั้ง 2 วัตถุ อาจไม่เหมือนกันนอกจากนี้วัตถุต่าง ๆ จะมีคุณลักษณะ และรายละเอียดวิธีการกระทำที่แตกต่างกัน แต่มองให้เหมือนกันได้ จากคุณลักษณะและพฤติกรรมที่วัตถุเหล่านั้นมีร่วมกัน

เช่น วัตถุ “SQUARE” และวัตถุ “CIRCLE” มีวิธีการ “calculateArea” ต่างกัน แต่สามารถมองคุณสมบัติร่วมกันอันได้แก่ ทั้ง 2 วัตถุ ต่างก็เป็น วัตถุ “SHAPE” ที่สามารถคำนวณพื้นที่ได้ด้วยเมธอด “calculateArea”




**ภาพที่ 1.17** ตัวอย่างของพอลิมอร์ฟิซึม (Polymorphism)

จากภาพที่ 1.17 สามารถอธิบายได้ว่ารูปทรงเลขาคณิตสามารถแบ่งออกเป็น วงกลม วงรี หรือรูปสี่เหลี่ยมจัตุรัส โดยทั้งสามนั้นจะมีเมธอดเดียวกันคือการคำนวณพื้นที่แต่วิธีการที่แต่ละวัตถุกระทำกับเมธอดนั้นแตกต่างกันไปตามลักษณะของวัตถุนั้น

กล่าวโดยสรุป หลักการของพอลิมอร์ฟิซึมนั้นมีประโยชน์มากเนื่องจากการพัฒนาระบบซอฟต์แวร์ใด ๆ ก็ตามสามารถปฏิบัติงานร่วมกันได้อย่างสะดวก และมีประสิทธิภาพ โดยสามารถมองภาพรวมของวัตถุ ที่อยู่ภายในระบบเป็นรูปแบบที่มีลักษณะร่วมกัน โดยไม่จำเป็นต้องรู้รายละเอียดการทำงานภายในของวัตถุเลย

## เหตุผลที่หลักการเชิงวัตถุได้รับความนิยม

ทำไมต้องพัฒนาระบบโดยใช้หลักการเชิงวัตถุ สามารถนำเสนอมุมมองและแนวทางที่เป็นประโยชน์ในการจัดการกับปัญหาความซับซ้อนของซอฟต์แวร์ได้ดีกว่าวิธีการพัฒนาระบบแบบดั้งเดิม (Traditional Techniques) ทั้งสนับสนุนการติดต่อสื่อสารที่มีประสิทธิภาพในช่วงวงรอบการพัฒนาซอฟต์แวร์ (SDLC: Software Development Life Cycle) และยังตอบสนองต่อวิกฤติซอฟต์แวร์ (Software Crisis) ลดผลกระทบอันเกิดจากการเปลี่ยนแปลงซอฟต์แวร์ สนับสนุนการนำคอมโพเน้นท์ (Components) กลับมาใช้ใหม่ (High Level Reuse of Components) สนับสนุนการแก้ไขปัญหาของซอฟต์แวร์ เช่น Reusability, Extendibility, Interoperability เป็นต้น

## ข้อดีของหลักการเชิงวัตถุ

หลักการเชิงวัตถุมีข้อดีมากมายแต่ที่พอจะสรุปได้คร่าว ๆ มีดังนี้

1. ช่วยลดความซับซ้อนของการพัฒนาระบบ อีกทั้งยังทำให้การสร้างการดูแลเป็นไปได้ง่ายและรวดเร็ว

2. พัฒนาความสามารถในการสร้างและคุณภาพของโปรแกรมเมอร์ เนื่องจากเมื่อมีการวางโครงร่างการนำมาใช้งาน และมีการทดสอบสามารถที่จะนำระบบนี้ไปใช้กับระบบอื่น ๆ ได้อีก

3. ระบบที่มีการพัฒนาด้วยหลักการเชิงวัตถุจะมีความยืดหยุ่นสามารถแก้ไขและเพิ่มเติมได้ง่าย

4. หลักการเชิงวัตถุจะถูกนักวิเคราะห์ระบบมองในแง่ของระบบในโลกของความเป็นจริงไม่ใช่แค่เพียงระดับของโปรแกรมทางภาษาคือสามารถหาทางแก้ไขปัญหาที่เกิดขึ้นได้อย่างทันที

## สรุป

ในบทนี้ผู้ศึกษาได้เรียนรู้และเข้าใจความหมายของซอฟต์แวร์ ชนิดของภาษาในการพัฒนาซอฟต์แวร์ กระบวนการพัฒนาซอฟต์แวร์ทั้งแบบดั้งเดิมและแบบเชิงวัตถุ รวมถึงเข้าใจหลักการพื้นฐานต่าง ๆ เกี่ยวกับวัตถุ เช่นความหมายของวัตถุ ความหมายของคลาส ความสัมพันธ์ระหว่างคลาสและวัตถุ คุณลักษณะต่าง ๆ ของวัตถุ กลไกที่มีประโยชน์ของวัตถุ เช่น การห่อหุ้ม การซ่อนรายละเอียด การรับทอดมรดก และพอลิมอร์ฟิซึม เป็นต้นนอกจากนี้ยังอธิบายถึงข้อดีข้อเสียของหลักการเชิงวัตถุได้ เพื่อจะใช้เป็นความรู้พื้นฐานในการศึกษาบทต่อ ๆ ไปอีกด้วย

| ชื่อ-นามสกุล | รหัส | สาขาวิชา | รุ่น/หมู่ | คะแนน | ลายเซ็นต์อาจารย์ |
|---|---|---|---|---|---|
| | | | | | |

## แบบฝึกหัดท้ายบทที่ 1

1. จงตอบคำถามต่อไปนี้

1.1 คอมพิวเตอร์มีความเข้าใจภาษาอะไรมากที่สุด

.....

.....

.....

1.2 ทำไมหรือเหตุใดคอมพิวเตอร์จึงเข้าใจเพียงแค่ภาษาเครื่อง

.....

.....

.....

1.3 ภาษาเครื่องคืออะไร

.....

.....

.....

1.4 ภาษาปาสคาสคอมพิวเตอร์เข้าใจหรือไม่

.....

.....

.....

2. จงอธิบายหลักการเชิงวัตถุ (OO) มาพอสังเขป

.....

.....

.....

.....

3. จงอธิบายความหมายของวัตถุ (Object) และคลาส (Class)

.....

.....

.....

.....

4. จงยกตัวอย่างการสืบทอดตามหลักการสืบทอดคุณสมบัติ (Inheritance)

.....

.....

.....

.....

5. จงระบุวัตถุจากคลาสต่อไปนี้มาคลาสละ 5 อย่าง

| **คำนาม** | **วัตถุ/คลาส** |
|---|---|
| นก | ต.ย. 1. นกแก้ว 2. นกยูง 3. นกเขา 4. นกพิราบ 5. นกเค้าแมว |
| แมว | ต.ย. 1. แมวสีดำ 2. แมวสีขาว 3. jerry 4. แมวสีสวาท 5. แมวของสมหญิง |
| ลูกค้า | |
| ธนาคาร | |
| คน | |
| สินค้า | |
| โต๊ะ | |
| เก้าอี้ | |
| ห้องเรียน | |
| กฎหมาย | |
| มหาวิทยาลัย | |
| ภาพยนตร์ | |

6. จงอธิบายความหมายของซอฟต์แวร์เชิงวัตถุมาพอสังเขป

.....

.....

.....

.....

7. จงให้เหตุผลว่าทำไมจึงมีการขึ้นสร้างซอฟต์แวร์ใหม่อยู่เรื่อย ๆ

.....

.....

.....

.....

## เอกสารอ้างอิง


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

กิติ ภักดีวัฒนะกุล, และกิตติพงษ์ กลมกล่อม. (2544). *UML: วิเคราะห์และออกแบบระบบเชิงวัตถุ.* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

ญาดา เขื่อนใจ. (2554). การพัฒนาสื่อการสอนสำหรับการเรียนการสอน รายวิชา 5672501 การวิเคราะห์และออกแบบเชิงวัตถุ. *วารสารวิชาการคณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏลำปาง, 4*(1), 9-16.

ธนาวินท์ รักธรรมานนท์. (2557). โปรแกรมภาษาแอสเซมบลี้. สืบค้น 19 2 สิงหาคม 2563, จาก https://slideplayer.in.th/slide/2144422/

นัฐพงศ์ ส่งเนียม. (2563). *วิชาวิเคราะห์และออกแบบระบบเชิงวัตถุ (OOAD).* สืบค้น 18 กันยายน 2563, จาก http://www.siam2dev.net/siam2dev_4122506_OOAD.php

รสรินทร์ พาณิชย์วงศ์. (2561). Spiral Development หรอ.. รู้จักไหม?. สืบค้น 21 ตุลาคม 2562, จาก https://www.glurgeek.com/education/spiral-development-หรอ-รู้จักไหม/

วิกิตำรา. (2563). *ซอฟต์แวร์.* สืบค้น 19 กันยายน 2563, จาก https://th.wikipedia.org/wiki/ซอฟต์แวร์

อัษฎาพร ทรัพย์สมบูรณ์. (2554). *การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis And Design).* กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์.

Kung, D., and Lei, J. (2016). An Object-Oriented Analysis and Design Environment. *2016 IEEE 2 9 th International Conference on Software Engineering Education and Training (CSEET),* 91-100.

Object Management Group. (1997). *WHAT IS UML*. Retrieved 18 September 2020, from https://www.uml.org/

Shin, S. (2015). A Study on the Difficulties of Learning Phase Transition in Object-Oriented Analysis and Design From the Viewpoint of Semantic Distance. *IEEE Transactions on Education, 58*(2), 117-123.

Shin, S. (2016). Concept Maps as Instructional Tools for Improving Learning of Phase Transitions in Object-Oriented Analysis and Design. *IEEE Transactions on Education, 59*(1), 8-16.

Tutorialspoint. (2014). *Object Oriented Analysis & Design Tutorial.* Retrieved 18 September 2020, from https://www.tutorialspoint.com/object_oriented_analysis_design/index.htm