เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน

AN OBJECT ORIENTED EXPERT SYSTEM SHELL WITH INDUCTIVE LEARNING

โดย

นายสุบัณฑิตย์ แก้วชูเสน

นางสาวอติพร อุ่นชู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน

AN OBJECT ORIENTED EXPERT SYSTEM SHELL WITH INDUCTIVE LEARNING

โดย

นายสุบัณฑิตย์	แก้วชูเสน	37014514
นางสาวอติพร	อุ่นชู	37014549

อาจารย์ที่ปรึกษา

รศ. ดร. ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน

ผู้จัดทำ

1.นายสุบัณฑิตย์ แก้วชูเสน รหัส 37014514

2.นางสาวอติพร อุ่นชู รหัส 37014549

อาจารย์ที่ปรึกษา

(รศ. ดร.ศุภมิตร จิตตะยโศธร)

เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน

นายสุบัณฑิตย์ แก้วชูเสน
นางสาวอติพร อุ่นชู

รศ. ดร.ศุภมิตร จิตตะยโศธร
ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอ "เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน " โดยได้ทำการศึกษาถึงคุณสมบัติ ลักษณะเด่น ข้อดีและข้อเสีย ของการเขียนโปรแกรมโดยใช้หลักการของ ออบเจกต์ โอเรียลเต็ด เทคนิค

โดยในการศึกษาครั้งนี้ได้เลือกใช้ภาษา Smalltalk อันเป็นภาษาที่ได้รับการยกย่องว่าเป็นภาษาทางออบเจกต์ โอเรียลเต็ด ที่สมบูรณ์แบบที่สุดภาษาหนึ่ง มาเป็นตัวอย่างเพื่อศึกษาแนวความคิดของออบเจกต์ โอเรียลเต็ด โดยได้นำภาษา Smalltalk มาใช้ represent งานทางได้ Expert System ซึ่งนำมาพัฒนาเป็นเปลือกระบบผู้เชี่ยวชาญ (Expert System Shell) โดยทำงานอ้างอิงกับสถาปัตยกรรม LEX Shell และได้เลือกใช้เทคนิควิธีการของ ID3 ในการหา root node ของต้นไม้สำหรับการตัดสินใจ (decision tree) ในส่วน Acquisition Module ของสถาปัตยกรรม LEX Shell

พร้อมทั้งทำการศึกษาเปรียบเทียบลักษณะการเขียนโปรแกรมแบบออบเจกต์ โอเรียลเต็ด กับการเขียนโปรแกรมแบบ Procedural Structure เป็นอย่างไร มีข้อดี - ข้อเสีย และข้อแตกต่างกันอย่างไร

AN OBJECT ORIENTED EXPERT SYSTEM SHELL WITH INDUCTIVE LEARNING

Subandit Kaeochoosan

Atiporn Unchoo

Suphamit Chittayasothorn

1997

Abstract

This thesis presents " An Object Oriented Expert System Shell with Inductive Learning ". We study the propoties, advantage and disadvantage of Object Oriented Technique

This research choose Smalltalk Language (It is the perfect Object Oriented Language) for study object oriented concept and use Smalltalk to represent the expert system shell with reference by LEX Shell architecture. We use ID3 technique for fine root node of decision tree in acquisition module of LEX Shell architecture.

Last we compare between " Object Oriented Programming " and " Procedural Structure Programming ".

สารบัญ

สารบัญตาราง

หน้า

สารบัญภาพ

บทที่1

บทนำ

ด้วยพัฒนาการและวิวัฒนาการที่กำลังเจริญก้าวหน้าไปอย่างไม่หยุดยั้งของระบบคอมพิวเตอร์ ทำให้คอมพิวเตอร์ได้เข้ามามีบทบาทสำคัญเป็นอย่างยิ่งต่อการพัฒนาภายในหน่วยงานและองค์กรต่างๆ ยกตัวอย่างเช่น ในวงการธุรกิจ คอมพิวเตอร์ได้เข้ามามีบทบาทเป็นอย่างสูง ทั้งในลักษณะการเป็นอุปกรณ์สำนักงานพื้นฐาน ไปจนถึงการเป็นแหล่งข้อมูลตลอดจนกระบวนการวิเคราะห์ข้อมูล เพื่อใช้เป็นส่วนหนึ่งของข้อมูลที่บุคคลในระดับของผู้บริหารใช้ในการตัดสินใจ เช่น ระบบ MIS (Management Information System) ก็ล้วนแต่อาศัยคอมพิวเตอร์เข้ามาช่วยทั้งสิ้น จึงอาจกล่าวได้ว่านับวันคอมพิวเตอร์ได้เข้ามามีบทบาทสำคัญต่อชีวิตคนเรามากขึ้นทุกทีจนไม่อาจหลีกเลี่ยงได้

1.1แนวคิดในการนำเสนอปริญญานิพนธ์

งานระบบผู้เชี่ยวชาญ (Expert System) ก็เป็นส่วนหนึ่งที่เป็นส่วนสำคัญอย่างยิ่ง สำหรับการนำคอมพิวเตอร์เข้ามาช่วยในการทำงานด้านต่าง ๆ ไม่ว่าจะเป็นทางด้านการแพทย์ การเกษตรกรรม การบริหารธุรกิจต่าง ๆ เพราะการใช้ระบบคอมพิวเตอร์เข้ามาช่วยในงานด้าน ต่าง ๆ จะช่วยให้งานมีประสิทธิภาพดีขึ้น ดังนั้นหากหน่วยงานใดมีระบบผู้เชี่ยวชาญที่มีประสิทธิภาพ มีระบบคอมพิวเตอร์ที่ดี ย่อมเป็นประโยชน์และเป็นความได้เปรียบของหน่วยงานนั้น ๆ ในการที่จะดำเนินงานในด้านต่าง ๆ ได้อย่างมีประสิทธิภาพและคงไว้ซึ่งความได้เปรียบเหนือหน่วยงานหรือองค์กรอื่น ๆ ด้วยมูลเหตุดังกล่าวนี้เอง ทำให้ปัจจุบันนี้จึงมีการศึกษาและพัฒนาระบบผู้เชี่ยวชาญกันอย่างกว้างขวาง เพื่อให้มีประสิทธิภาพเหมาะสมและเป็นประโยชน์ที่สุดในการนำไปใช้งาน

ปัจจุบันได้มีการพัฒนาแนวความคิดใหม่ในการเขียนโปรแกรมขึ้นมาใหม่ โดยแนวความคิดนั้นก็คือ การเขียนโปรแกรมแบบ ออบเจกต์ โอเรียลเต็ด ซึ่งเป็นการพัฒนารูปแบบการเขียนโปรแกรมให้มีประสิทธิภาพดียิ่งขึ้น จึงเป็นเรื่องที่น่าสนใจมากหากจะได้มีการนำแนวความคิดใหม่ของ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง มาใช้ในการพัฒนาระบบผู้เชี่ยวชาญให้มีประสิทธิภาพมากขึ้น ด้วยเหตุนี้จึงเป็นสาเหตุสำคัญที่ทำให้มีการทำจัดทำปริญญานิพนธ์ชิ้นนี้ขึ้นมา เพื่อทำการศึกษาหาแนวทางและความเป็นไปได้ของการเขียนโปรแกรม แบบ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง ที่ใช้ในงานระบบผู้เชี่ยวชาญ ว่าในการพัฒนาระบบผู้เชี่ยวชาญด้วย ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง นั้นจะเหมาะสมหรือไม่อย่างไร และส่วนที่สำคัญยิ่งของชิ้นงานนี้ก็คือจะได้ทำการศึกษาเพื่อหาข้อแตกต่าง ข้อได้เปรียบ ข้อเสียเปรียบ ของการพัฒนาโปรแกรมแบบ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง กับวิธีการแบบ โพรซีเดอรอล โปรแกรมมิ่ง (Procedural Structure Programming) แบบเก่าว่าแตกต่างกันอย่างไรและเหมาะสมหรือไม่หากจะมีการนำไปพัฒนาเพื่อใช้ในอนาคต

1.2วัตถุประสงค์ของปริญญานิพนธ์

1.เพื่อศึกษาถึงการออกแบบและเขียนโปรแกรมในรูปแบบออบเจกต์ โอเรียลเต็ด เทคนิค

2.เพื่อศึกษาเปรียบเทียบข้อดี-ข้อเสียระหว่างการออกแบบและการเขียนโปรแกรมแบบออบเจกต์ โอเรียลเต็ด และการเขียนโปรแกรมแบบโพรซีเดอร์

3.เพื่อศึกษาถึงความเหมาะสมในการใช้เทคนิควิธีการแบบ ออบเจกต์ โอเรียลเต็ด กับการทำแอพลิเคชั่นในเรื่องเกี่ยวกับ เปลือกของระบบผู้เชี่ยวชาญว่ามีความเหมาะสมเพียงใด

1.3 วิธีการดำเนินงาน

งานวิจัยนี้จะเริ่มต้นด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ 4 เรื่อง คือ เทคนิควิธีการออกแบบและเขียนโปรแกรมแบบออบเจกต์ โอเรียลเต็ด , ภาษาสมอลทอกค์, ระบบผู้เชี่ยวชาญ (Expert System) และวิธีการ ไอดี3

จากนั้นก็จะนำเอาความรู้ที่ได้ศึกษามาทั้งหมดมาทำการออกแบบโปรแกรมเพื่อพัฒนา เปลือกของระบบผู้เชี่ยวชาญ และก็จะเริ่มเข้าสู่ขั้นตอนของการพัฒนาโปรแกรม การทดสอบโปรแกรม และสรุปผลการพัฒนาโปรแกรม ที่ได้เลือกใช้ภาษาสมอลทอกค์ซึ่งเป็นภาษาทางออบเจกต์ โอเรียลเต็ดที่สมบูรณ์แบบที่สุดภาษาหนึ่งในการพัฒนา

1.4 ขอบเขตของปริญญานิพนธ์

ซึ่งปริญญานิพนธ์ฉบับนี้ ทำการศึกษาเกี่ยวกับเรื่อง "เปลือกระบบผู้เชี่ยวชาญเชิงวัตถุที่มีความสามารถในการเรียนรู้เชิงอุปมาน (An Object Oriented Expert System Shell With Inductive Learning) " และในการศึกษาในครั้งนี้ได้ทดลองนำภาษาสมอลทอกค์ มาใช้สำหรับงานในทางระบบผู้เชี่ยวชาญ และระบบฐานความรู้ (Knowledge base)ของระบบผู้เชี่ยวชาญนั้น ตลอดจนนำภาษา สมอลทอกค์ มาใช้เพื่อแทน Knowledge Acquisition เพื่อสร้างต้นไม้สำหรับการตัดสินใจ (decision tree)ที่เหมาะสมที่สุด

ซึ่งปริญญานิพนธ์ฉบับนี้ได้ทำการนำเสนอรายละเอียดต่าง ๆ ไว้ดังนี้

บทที่2 ทฤษฎีพื้นฐาน

: จะกล่าวถึงใน 3 ส่วน คือ หลักการของ ออบเจกต์ โอเรียลเต็ด (Object Oriented Concept) , ระบบผู้เชี่ยวชาญ (Expert System) , ทฤษฎี ไอดี3 (ID3)

บทที่3 ภาษาสมอลทอกค์ (Smalltalk)

: จะกล่าวถึงโครงสร้างการเขียนโปรแกรมในภาษาสมอลทอกค์ รวมถึงรูปแบบคำสั่ง และโครงสร้างการควบคุม (control structure) ต่าง ๆ

บทที่4 สถาปัตยกรรมเล็กซ์เชล (LEX Shell Architecture) และการออกแบบโปรแกรม

: จะกล่าวถึงรูปแบบของสถาปัตยกรรมเล็กซ์เชล และการออกแบบคลาส (Class) เมทอด (mother) ต่าง ๆ ที่ใช้ในโปรแกรม

บทที่5 โปรแกรมเปลือกระบบผู้เชี่ยวชาญ (Source Code)

: โปรแกรมเปลือกระบบผู้เชี่ยวชาญที่เขียนด้วยภาษาสมอลทอกค์

บทที่6 สรุปผลการทดลอง

: จะทำการเปรียบเทียบข้อดี,ข้อเสีย ของการเขียนโปรแกรมแบบ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง และแบบโพรซีเดอร์ (Procedural Structure Programming)

บทที่2
ทฤษฎีพื้นฐาน

2.1 เทคนิควิธีการแบบออบเจกต์ โอเรียลเต็ด (OBJECT ORIENTED TECHNIQUE)

2.1.1 หลักการของ ออบเจกต์ โอเรียลเต็ด (OBJECT-ORIENTED CONCEPT)

คือ ผู้ใช้ (user) ไม่จำเป็นต้องรู้โครงสร้างของการจัดการภายในคอมพิวเตอร์ (computer) แต่จะต้องรู้เรื่องเกี่ยวกับ ออบเจกต์ (object) และ โอเปอเรชั่น (operation) มีคำศัพท์ที่เกี่ยวข้องกับ ออบเจกต์ โอเรียลเต็ด อยู่ 5 คำ คือ

1 ออบเจกต์ (Object)

ออบเจกต์ ใน ออบเจกต์ โอเรียลเต็ด จะตรงกับคำว่า "เอ็นติตี้" (Entity) ในไนแอม ออบเจกต์ จะอยู่ภายใต้คลาส (class) ซึ่งเหมือนกับ เอ็นติตี้ อยู่ภายใต้ เอ็นติตี้ ไทป์ (Entity type)

ออบเจกต์ ประกอบด้วย 2 ส่วน คือ

- ดาต้า (Data) : เป็นสถานะ (state) ของ ออบเจกต์ นั้น
- โอเปอเรชั่น(Operation): เป็นวิธีการเข้าถึงข้อมูลและจัดการ ปรับปรุง เปลี่ยนแปลงสถานะ ของออบเจกต์ นั้น

2 คลาส (Class)

คลาส ใน ออบเจกต์ ตรงกับคำว่า เอ็นติตี้ ไทป์ ในไนแอม ใช้จัดกลุ่มของ ออบเจกต์ ตามคุณสมบัติ และ ผู้ใช้สามารถติดต่อ (interface) กับ คลาส โดยผ่านทาง เมทอด

3เมทอด (Method)

เป็นโอเปอเรชั่นที่สำคัญหรือเป็นฟังก์ชั่น (function) ที่สามารถทำงานกับ ออบเจกต์ ได้ แต่ละคลาส จะมีเมทอดของตัวเอง กลุ่มของเมทอดที่นำไปใช้กับคลาสที่กำหนด จะต้องคำนึงถึง ความหมาย (definition) ที่เก็บในคลาสนั้น

4 แมสเซส (Message)

ออบเจกต์ จะติดต่อกับ ออบเจกต์ อื่น โดยผ่านทาง แมสเซส การส่ง แมสเซส คือ การไปเรียก เมทอด มาทำงาน แล้วส่งผลลัพธ์ไปยังผู้ส่งแมสเซส (sender)

5 คลาส ไฮเออราคี (Class Hierarchy)

คลาส ไฮเออราคี ใน ออบเจกต์ โอเรียลเต็ด ตรงกับคำว่า ซับไทป์ (Subtype) ในไนแอม และเหมือนกับโครงสร้างลำดับชั้น (Hierarchy Structure) เช่น ถ้า คลาส B เป็น ซับคลาส (subclass) ของ คลาส A แล้ว ทุก ๆ อินสแตนส์ (instance) ของ B จะเป็น instance ของ A โดยอัตโนมัติ และ B ก็จะได้รับการถ่ายทอดคุณสมบัติจาก A ด้วย

การถ่ายทอดคุณสมบัติมี 2 ชนิด คือ

1.Structure Inheritance เช่น B ได้รับการถ่ายทอดคุณสมบัติ ตัวแปรอินเฮอริแตนส์ จาก A

2.Behavioral Inheritance เช่น B ได้รับการถ่ายทอดเมทอดจาก A

Object - Oriented Term	Programming Term	NIAM Term
Object	Variable	Entity
Class	Type	Entity Type
เมทอด	Function	-
Message	Call	-
Class Hierarchy	Type Hierarchy	Subtype

ตารางที่ 2-1 แสดงการเปรียบเทียบคำศัพท์ระหว่าง Object - Oriented Term กับ Programming Term และ NIAM Term

2.1.2 ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง (Object Oriented Programming)

ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง เป็นวิธีการใหม่ในการพัฒนาซอฟแวร์ (software) หลักการสำคัญของ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง คือ abstract data type, คลาส, การจัดลำดับของคลาส , อินเฮอริแตนส์ (inheritance) และ โพลิมอฟิซึ่ม (polymorphism)

abstract data type เป็นแก่นของ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง abstract data type จะรวม ดาต้าไทป์ (data type) และชุดของการทำงาน (operation) เข้าไว้ด้วยกัน ซึ่งการทำงานจะทำการกำหนดคุณสมบัติของดาต้าไทป์นั้น

การนิยามคลาส (class definition) ซึ่งจะกำหนดการทำงานของ abstract data type สามารถทำได้ด้วยการนิยามว่า จะเรียกใช้การทำงานของ data type นั้นอย่างไร การนิยามคลาส ยังกำหนดโครงสร้างข้อมูลของคลาสด้วย

การทำงานกับ ดาต้าไทป์ จะแบ่งออกเป็น 2 ชนิด คือ

1.การทำงานแบบ Public คือ สามารถเรียกใช้ได้จากภายนอกคลาส

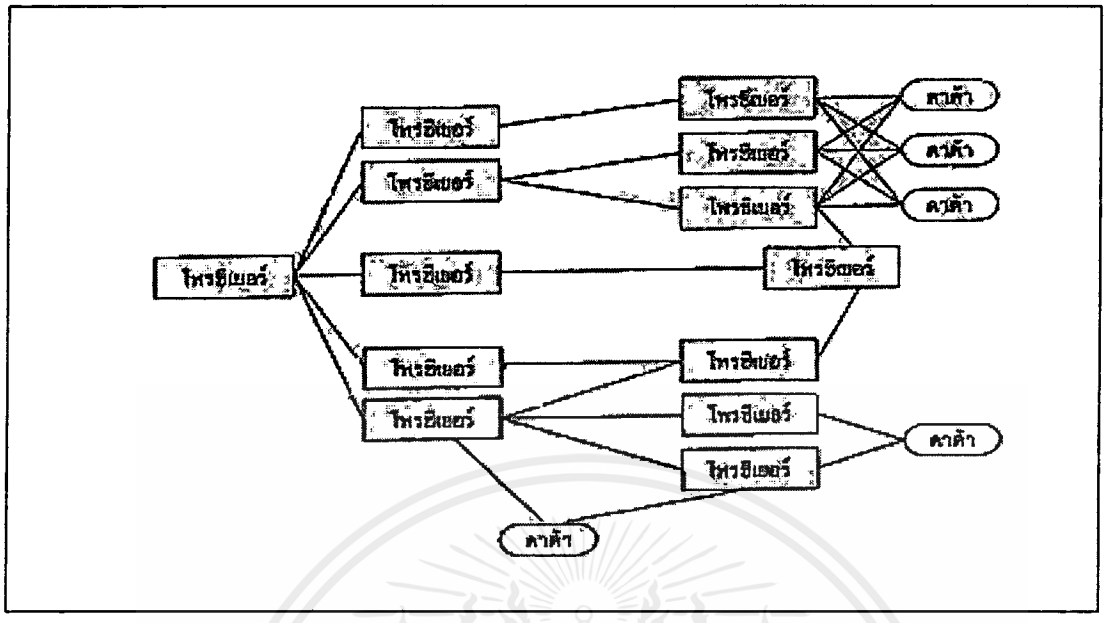
2..การทำงานแบบ Private คือ สามารถเรียกใช้ได้เฉพาะภายในคลาส

ทำไมต้องเป็นออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง (Why Object Oriented Programming ?

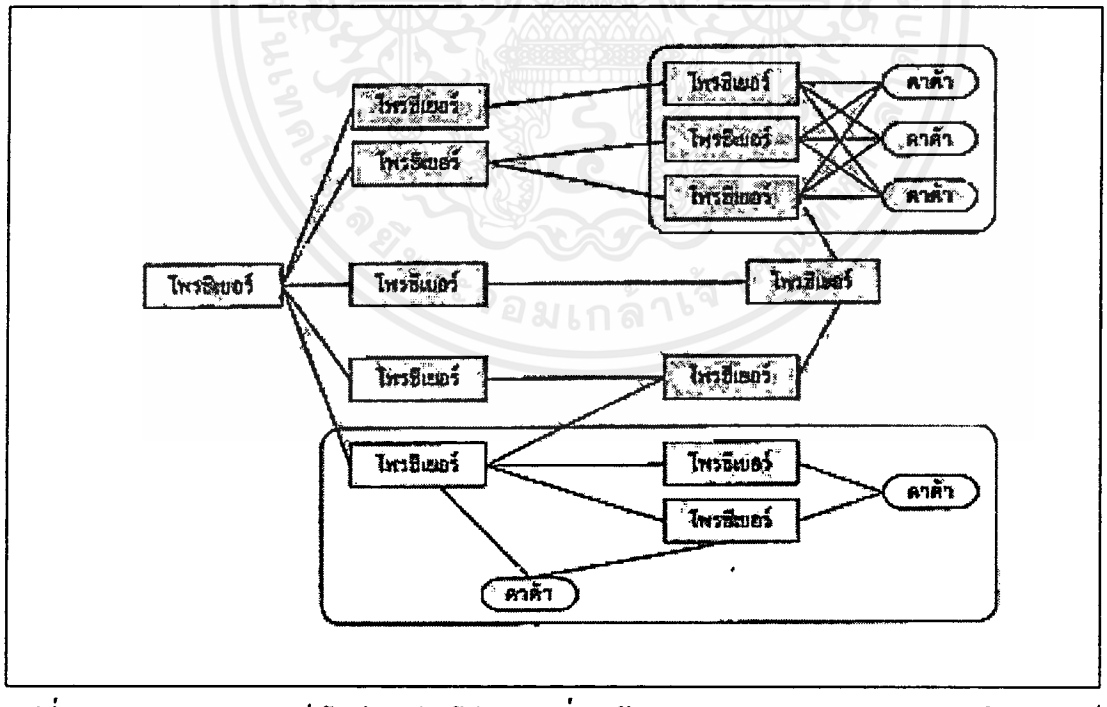
เดิมลักษณะการเขียนโปรแกรมเป็นแบบ Procedure คือ การสั่งให้คอมพิวเตอร์ ทำงานอย่างหนึ่ง โดยความหมายของแต่ละคำสั่ง (instruction) ซึ่งเป็นลำดับในโปรแกรม ถ้าโปรแกรมเล็ก ๆ จะไม่มีปัญหาหรือผลกระทบที่เห็นได้เด่นชัด ถ้าโปรแกรมมีขนาดใหญ่จะแก้ไขหาข้อผิดพลาดได้ยาก แต่มีข้อดี ที่สามารถแบ่งโปรแกรมเป็น ซับโปรแกรม (Subprogram) ได้ แต่ถ้าโปรแกรมมีความซับซ้อนมากเกินไปจะทำให้เกิดปัญหาขึ้นได้อีก

ถ้าเป็น ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง จะรวมส่วนที่เป็นข้อมูลและฟังก์ชั่นที่เรียกใช้ข้อมูลนั้นไว้ด้วยกัน เรียกว่า ออบเจกต์ เมื่อต้องการเข้าถึงข้อมูลในออบเจกต์ เมทอดจะเป็นตัวเข้าไปจัดการข้อมูล หลังจากนั้นจะทำการส่งข้อมูลที่ต้องการกลับมาตามที่ต้องการ เพื่อป้องกันการเข้าถึงข้อมูลของฟังก์ชั่นอื่น ๆ ซึ่งการ

เขียนโปรแกรมแบบนี้จะช่วยลดความยุ่งยากสำหรับโปรแกรมที่ซับซ้อน และยังช่วยให้ข้อมูลมีความปลอดภัยเพิ่มขึ้นด้วย



รูปที่ 2-1 ลักษณะการทำงานของภาษาในลักษณะ Procedural language จะเห็นว่า แต่ละฟังก์ชั่นจะทำงานกับข้อมูลในลักษณะที่เป็นอิสระต่อกัน



รูปที่ 2-2 สำหรับ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง แล้วจะพยายามมองการทำงานเป็นกลุ่มของฟังก์ชั่น ที่จำแนกออกตามวัตถุประสงค์ (class) รวมทั้งข้อมูลก็จะถูกซ่อนเก็บไว้ในคลาส

เป้าหมายหลักในการพัฒนาซอฟแวร์แบบออบเจกต์ โอเรียลเต็ด

1.ทำให้การพัฒนาซอฟแวร์ใช้เวลาสั้นลง ต้นทุนต่ำ โดยการใช้ คลาส ที่สามารถนำกลับมาใช้ได้อีก และสร้าง ซับคลาส (Subclass) ขึ้นมาเพื่อช่วยในการแก้ปัญหา

2.ทำให้ต้นทุนในการบำรุงรักษาซอฟแวร์ต่ำลง เพราะสามารถหาสิ่งที่ต้องการเปลี่ยนแปลงใน ซอฟแวร์ ได้ง่าย และการเปลี่ยนแปลงไม่มีผลกระทบต่อภายนอกคลาส ได้อย่างแน่นอน

3.ทำให้ประสิทธิภาพในการผลิตซอฟแวร์ดีขึ้น วิธีนี้ทำให้การออกแบบและการเขียนซอฟแวร์ มีความใกล้ชิดกันมากขึ้น

ความท้าทายของ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง

1. แบ่งย่อย ซอฟแวร์ ออกเป็น คลาส หลาย ๆ คลาส
2. เพิ่มเติมฟังก์ชั่นให้กับระบบ ซอฟแวร์ ที่มีอยู่
3. มีการจัดลำดับชั้นของ คลาส และ ซับคลาส

คุณสมบัติที่สำคัญของ ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง

1. ข้อมูลที่เก็บในออบเจกต์จะถูกคืนพื้นที่ในหน่วยความจำโดยอัตโนมัติ ถ้าไม่มีการใช้ข้อมูล
2. การโพรเซส (Process) ถูกทำโดยการ ส่งแมสเซส (Sent Message) ไปที่ ออบเจกต์
3. Object Behavior จะระบุไว้ใน คลาส เท่านั้น

2.1.3 คุณสมบัติของออบเจกต์ โอเรียลเต็ด (OBJECT-ORIENTED PROTOTIES)

คุณสมบัติที่สำคัญของ ออบเจกต์ มี 4 ประการ คือ

1.แอ็บสแตรกชั่น (Abstraction)

2.เอ็นแคปซูเลชั่น (Encapsulation)

3.อินเฮอริแตนส์ (Inheritance)

4.โพลิมอฟิซึ่ม (Polymorphism)

1. แอ็บสแตรกชั่น (ABSTRACTION)

คือ การแทนความคิดที่ยุ่งยากซับซ้อน ให้สั้น กระทัดรัด รัดกุมขึ้น โดยใช้ ออบเจกต์ ซึ่งใน ออบเจกต์ จะรวมทั้ง คาต้า (data) และ โอเปอเรชั่น (operation) ไว้ในตัวเลย หรืออาจกล่าวได้ว่า ออบเจกต์ เป็น encapsulation of abstractions

การกำหนด แอ็บสแตรกชั่นใหม่ ในวิธีการแก้ปัญหาแบบ ออบเจกต์ โอเรียลเต็ด จะต้องการออบเจกต์ใหม่ และสำหรับแต่ละ ออบเจกต์ใหม่ ต้องกำหนด ลักษณะของคุณสมบัติ (properties feature) และ แมสเซส ที่จะติดต่อจัดการเกี่ยวกับ ออบเจกต์ นั้นได้

วิธีการแก้ปัญหาหนึ่ง อาจประกอบด้วยหลาย ออบเจกต์ บาง ออบเจกต์ ก็มีลักษณะ คล้ายกัน ซึ่งเรียกกลุ่ม ออบเจกต์ ที่มีลักษณะคล้ายกันนี้ว่า คลาส คุณสมบัติพื้นฐานของ ออบเจกต์ ในคลาสหนึ่ง ๆ จะถูก

กำหนดๆไว้ในคลาส แต่ละออบเจกต์ เป็นอินสแตนส์ของคลาส และ behavior ของ ออบเจกต์ ถูกกำหนดโดยกลุ่มของคุณสมบัติ (properties)

2.เอ็นแคปซูเลชั่น (ENCAPSULATION)

คือ การกันของโครงสร้างข้อมูล (data structure) กับฟังก์ชั่นที่เกี่ยวข้อง หรือ ฟังก์ชั่น ที่เรียกใช้ข้อมูลนั้น (method) เกิดเป็นวัตถุตัวใหม่ที่สามารถซ่อนข้อมูลจากภายนอกได้

ข้อดีของ encapsulation คือ

1.กำหนดขอบเขต (scope) ของ ซอฟแวร์ ภายในตัวมันอย่างชัดเจน

2.กำหนดการเชื่อมต่อที่ดีคือระบุได้ว่า software component แต่ละตัวกระทำกับตัวอื่นอย่างไร

3.มีการป้องกัน internal implementation โดยให้รายละเอียดของ functionality

ความหมายของ encapsulation คือ แต่ละ instance ใน คลาส (Object) คือ 1 encapsulation

ออบเจกต์ (encapsulation) *มีลักษณะดังนี้*

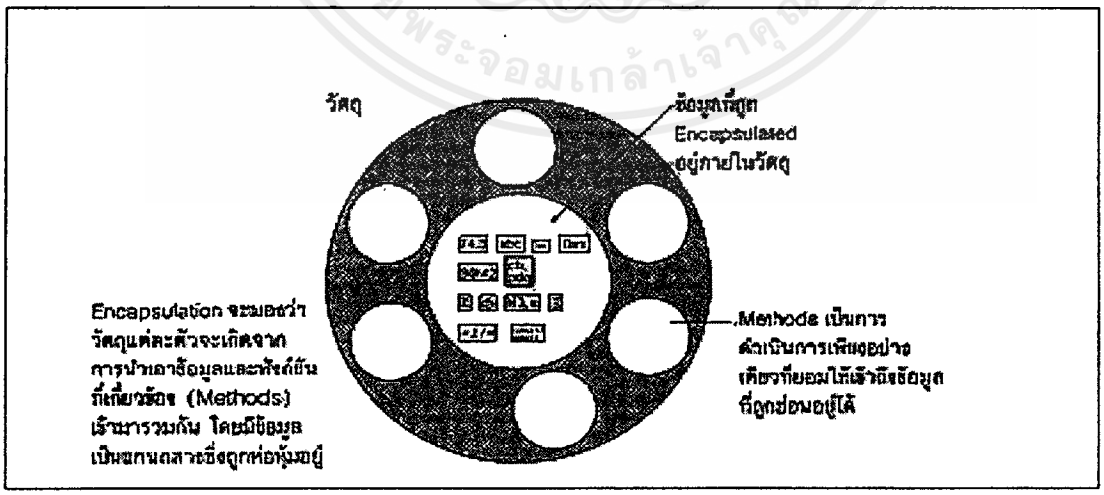
1.Private Data : แต่ละออบเจกต์จะกำหนดพารามิเตอร์ (parameter) ของแต่ละอินสแตนส์ ใน คลาส ซึ่งออบเจกต์ทั้งหมดที่เป็นอินสแตนส์ในคลาส จะมีพารามิเตอร์นี้ แต่จะมีค่าต่างกัน ออบเจกต์ หนึ่ง ๆ จะสามารถเข้าถึง (access) private data ของตัวมันเองได้เท่านั้น

2.Shared Data : พารามิเตอร์นี้จะมีค่าเหมือนกันสำหรับออบเจกต์ทั้งหมดที่เป็นอินสแตนส์ใน คลาส อินสแตนส์ทั้งหมดใน คลาส สามารถเข้าถึง shared data นี้ได้

3.Global -Shared Data : เป็นคาต้าที่อินสแตนส์ต่างคลาสสามารถใช้ได้ Global - Shared data จะมีค่าเหมือนกันสำหรับทุกออบเจกต์

4.แมสเซส : กลุ่มของแมสเซส คือ สิ่งที่ออบเจกต์จะทำการตอบสนอง(response) ออบเจกต์ทั้งหมดที่อยู่ใน คลาส เดียวกัน จะตอบสนองกลุ่มของแมสเซส เดียวกัน

ตัวอย่างของ เอ็นแคปซูเลชั่น เช่น ยาเม็ดแคปซูล ซึ่งมองไม่เห็นตัวยาภายใน รู้แต่ว่ายานี้รักษาโรคอะไร



รูปที่ 2-3 แสดงออบเจกต์กับเอ็นแคปซูเลชั่น

3. อินเฮอริแตนส์ (INHERITANCE)

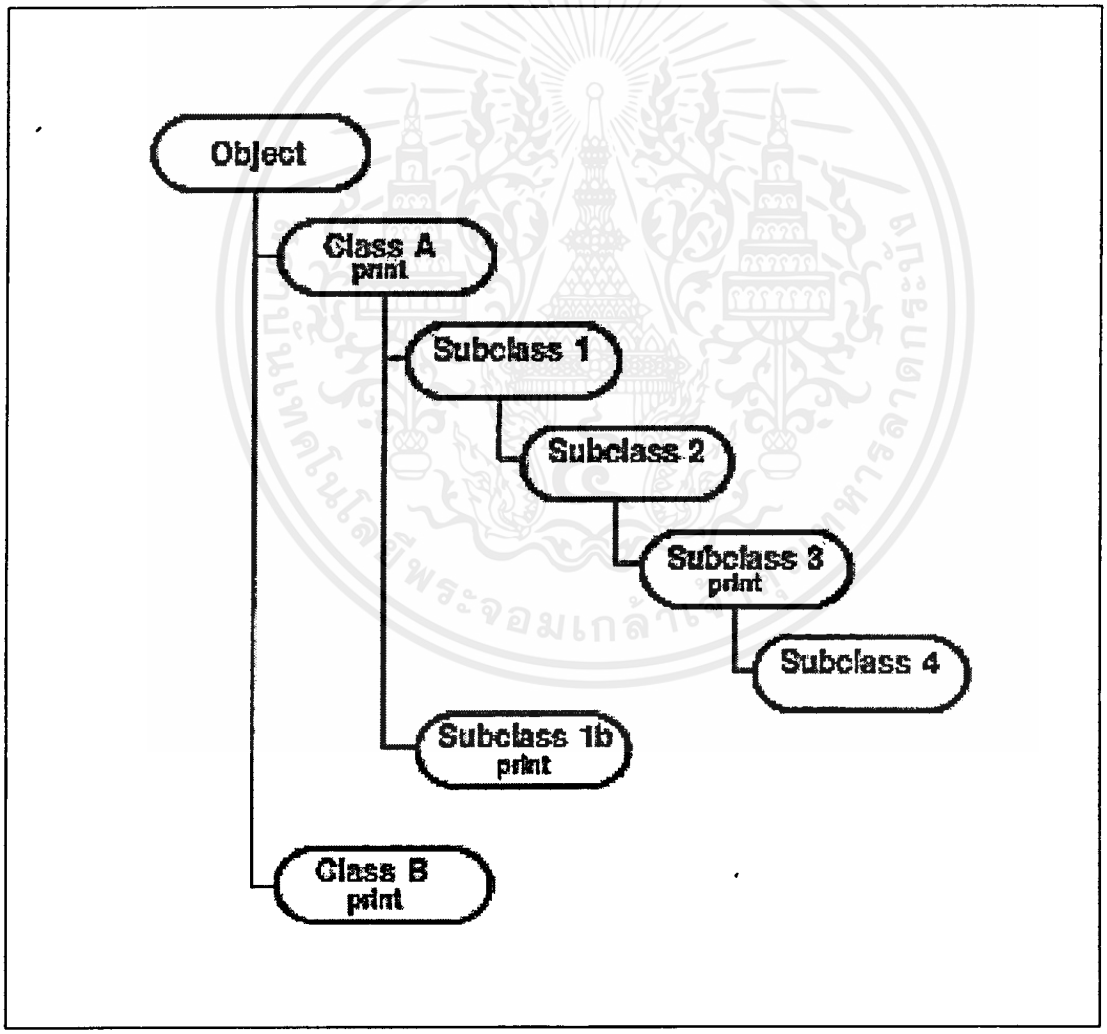
คือ การสร้าง คลาสใหม่ขึ้นมา โดยมีการสืบทอดคุณสมบัติพื้นฐานที่มาจาก คลาสเดิม แต่จะมีดาต้า หรือ เมทอด ที่พิเศษเป็นของตัวเองเพิ่มขึ้นมาจาก คลาส เดิม

ถ้า ซับคลาส B เป็น ซับคลาสของ คลาส A ออบเจกต์ ที่เป็น instance ของซับคลาส B จะมีคุณสมบัติพิเศษกว่า อินสแตนส์ของคลาส A แต่อย่างน้อยที่สุดต้องมีคุณสมบัติเหมือนอินสแตนส์ของ คลาส A

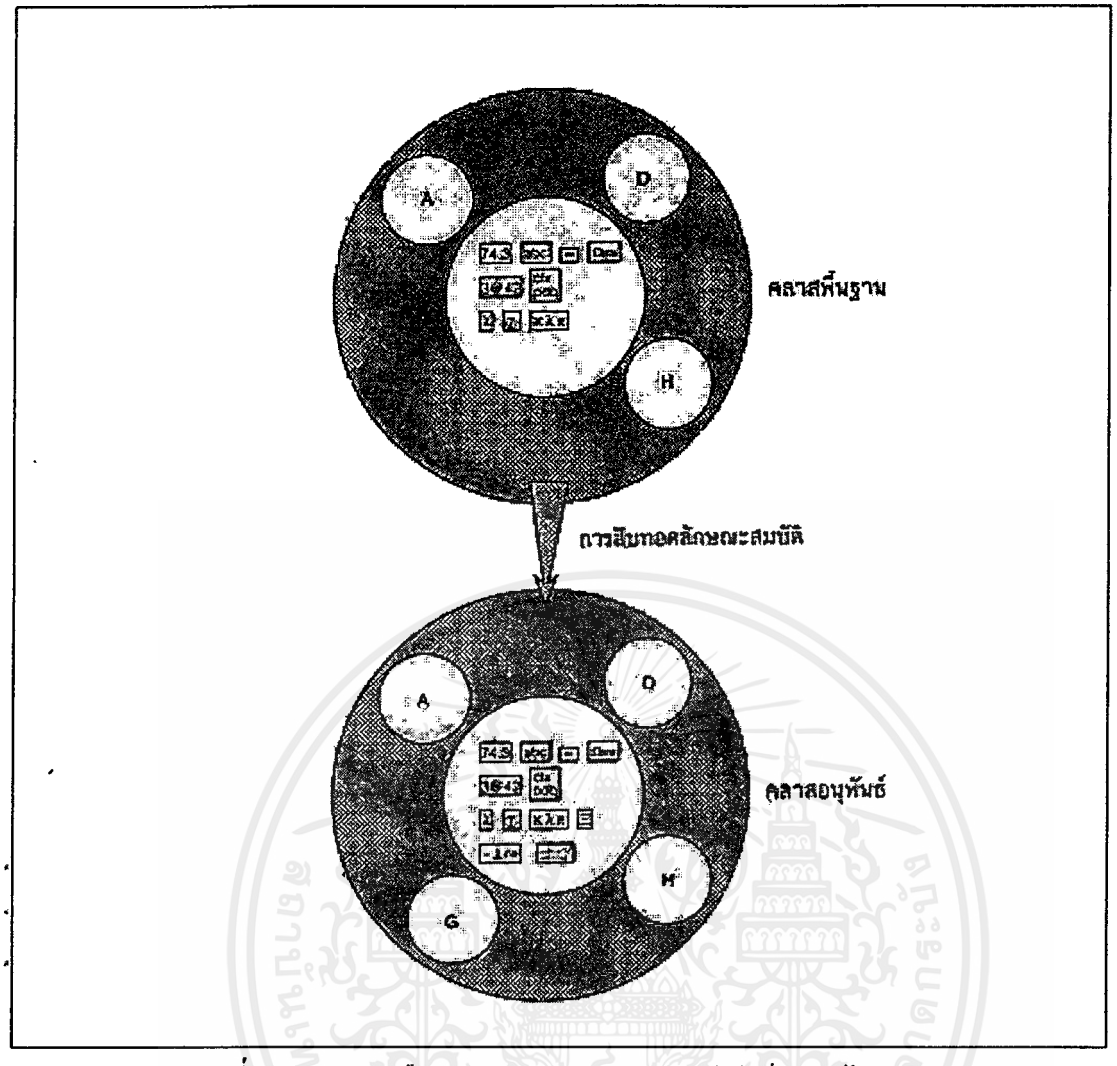
คุณสมบัติอินเฮอริแตนส์นี้จะรวมทั้ง private data , shared data และแมสเซส คลาส A จะถูกเรียกว่าเป็น ซุปเปอร์คลาส (superclass) ของ คลาส B

อินเฮอริแตนส์ เป็นคุณสมบัติของออบเจกต์ที่เป็นอินสแตนส์ของคลาส จะถ่ายทอดคุณสมบัติจาก ซุปเปอร์คลาสของมัน ซับคลาสมีหลายระดับ (level) คลาส ที่ระดับต่ำสุด ก็จะมีคุณสมบัติของ ซุปเปอร์คลาสหลายคลาส

ซับคลาสสามารถเข้าถึง private data , shared data , แมสเซส ของซุปเปอร์คลาสของมันได้ทุกคลาส

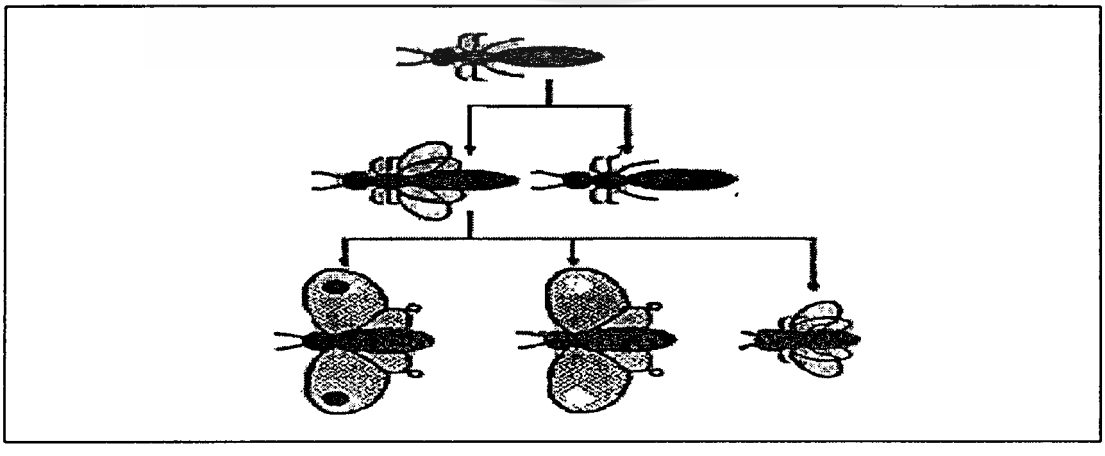


รูปที่ 2-4 Inheritance และ Class Hierarchy



รูปที่ 2-5 แสดงการสืบทอดลักษณะสมบัติจาก ซุปเปอร์คลาส ไปยัง ซับคลาส

ตัวอย่างของอินเฮอริแตนส์ เช่น คลาสของแมลง แต่ละประเภทของแมลงจะถูกแยกออกเป็นชั้น ๆ ที่ยอดบนสุด เป็น คลาส พื้นฐาน คือแมลงธรรมดา มี 6 ขา 3 ปล้อง ใน ไฟลัม (Phylum) จะแบ่งเป็น 2 Division คือ แมลงมีปีก กับไม่มีปีก (ซึ่งทั้ง 2 ประเภท ก็มี 6 ขา 3ปล้อง) และแมลงมีปีกก็ยังแบ่งได้อีก เป็นผีเสื้อ , ผึ้ง , แมลงปอ ฯลฯ ซึ่งแมลงเหล่านี้ก็มี 6 ขา 3 ปล้อง และมีปีกด้วย

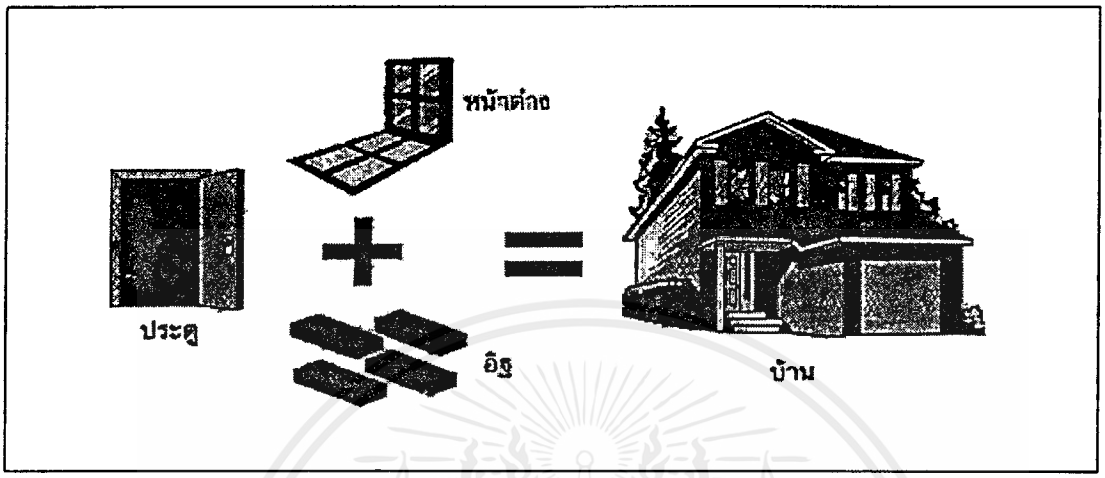


รูปที่ 2-6 แสดงการสืบทอดลักษณะสมบัติจาก ซุปเปอร์คลาส ไปยัง ซับคลาสของแมลง

มัลติเปิ้ล อินเฮอริแตนส์ (Multiple Inheritance)

ซับคลาสอาจสร้างจากซุปเปอร์คลาสที่มากกว่า 1 ตัวก็ได้ ซึ่งเรียกว่า มัลติเปิ้ล อินเฮอริแตนส์ ทำให้ภาษาออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง มีความยืดหยุ่นในการทำงานเพิ่มขึ้น

เช่น วัตถุบ้านถูกสร้างขึ้นมาจากการผสานกันระหว่างวัตถุหลายชนิด ถ้ามององค์ประกอบหลัก ๆ ของบ้าน แล้วจะพบว่า บ้านยังมีคุณสมบัติของประตู อิฐ หน้าต่างอยู่เหมือนเดิม



รูปที่ 2-7 แสดง Multiple Inheritance

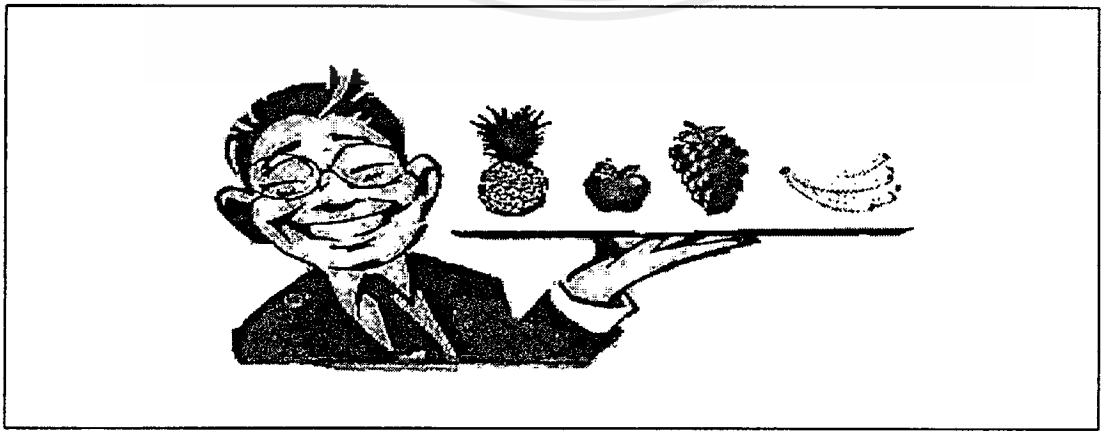
4.โพลิมอฟิซึ่ม (POLYMORPHISM)

คือ การที่เราส่งแมสเซสที่เหมือนกันไปในออบเจกต์ที่ต่างกัน แต่ละออบเจกต์จะตอบสนอง ออกมาไม่เหมือนกัน ตามแต่ชนิดและหน้าที่ของออบเจกต์

ความสามารถที่ใช้แมสเซสเหมือนกัน สำหรับโอเปอเรชั่นที่เหมือนกันไปยังออบเจกต์ต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหาหนึ่ง ๆ

ตัวอย่างของโพลิมอฟิซึ่ม

เช่น การกินเรากิน ส้ม แอปเปิ้ล ซึ่งขั้นตอนการกินผลไม้ทั้ง 2 ชนิดนี้ต่างกันแน่นอน แต่ผลที่ได้คือ อร่อย เหมือนกันและเวลาเราเรียกว่า กิน เราก็ไม่จำเป็นต้องบอกว่า กินอะไร เพราะรายละเอียดของการกินจะแตกต่างไปตามชนิดของอาหาร



รูปที่ 2-8 แสดง Polymorphism ขั้นตอนหรือวิธีการจะแตกต่างกันไปตามวัตถุหรือ แมสเซส

2.1.4 ความแตกต่างระหว่างการแก้ไขปัญหา โดย Object - Oriented Programming กับ Structure Programming

1.วิธีการเข้าถึงข้อมูล

OOP จะส่ง แมสเซส (action) ไปยัง ออบเจกต์ (data) และ ออบเจกต์ จะตอบสนองส่วนที่เป็นผลลัพธ์ไปยังผู้ส่ง

Structure พารามิเตอร์ ถูกส่งไปที่ โพรซีเดอร์ และโพรซีเดอร์จะทำโอเปอเรชั่นกับดาต้า นั้น

2.Data Abstraction

OOP ดาต้า แอ็บสแตรกชั่น (data abstraction) ถูกแทนที่โดย คลาส ของ ออบเจกต์

Structure ดาต้า แอ็บสแตรกชั่น ถูกแทนที่โดย ดาต้าไทป์ ของแต่ละภาษา

3.Functional Abstraction

OOP ฟังก์ชั่น แอ็บสแตรกชั่น (function abstraction) ถูกแทนที่โดย แมสเซส

Structure ฟังก์ชั่น แอ็บสแตรกชั่น ถูกแทนที่โดย โพรซีเดอร์และโอเปอเรชั่น

4.Encapsulation

OOP หน่วยของเอ็นแคปซูเลชั่น คือ ออบเจกต์

Structure การเอ็นแคปซูเลชั่นอยู่ในรูป Library Module ซึ่งคอมโพเน้น (Component) หนึ่งอาจบรรจุมากกว่า หนึ่งดาต้า แอ็บสแตรกชั่น

5.Inheritance and Polymorphism

OOP สนับสนุน (support) ทั้ง 2 อย่าง ทำให้การแสดงความสัมพันธ์ระหว่างซับคลาส ไม่ขึ้นต่อกันและยังช่วยลดความซ้ำซ้อน

Structure ไม่สนับสนุน ทั้ง 2 อย่าง โดย ซอฟแวร์ และ ดาต้า แอ็บสแตรกชั่น แทนด้วย Hierarchical Level ที่เท่ากัน และยังยุ่งยากในการตั้งชื่อโพรซีเดอร์

6.Extensibility

คือ คุณสมบัติของภาษาคอมพิวเตอร์ (Computer Language) ที่อนุญาตให้ผู้ใช้กำหนดโครงสร้างใหม่ได้

OOP ออบเจกต์ ทั้งหมดที่มีสถานะเท่ากัน ทั้งออบเจกต์ที่ถูกผู้ใช้สร้างและออบเจกต์ ที่เป็นส่วนของ System Kernel จะมีความสำคัญและประสิทธิภาพเท่า กัน

Structure โครงสร้างที่ถูกสร้างขึ้นมาใหม่จะมีความสำคัญและประสิทธิภาพน้อยกว่าโครงสร้างที่ถูกสร้างจากระบบ (system kernel)

7.Iteration

OOP จัดหาโครงสร้างที่เหมาะสมสำหรับการทำการวนซ้ำ (Loop)

Structure มีการทำการวนซ้ำมากเกินไป

2.1.5 สรุปลักษณะการเขียนโปรแกรมแบบ ออบเจกต์ โอเรียลเต็ด

1.ออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง ประกอบด้วย การส่งแมสเซส ไปยัง ออบเจกต์

2.ผลที่ได้จากการส่งแมสเซสไปยัง ออบเจกต์ คือ ออบเจกต์

3.วิธีการแก้ไขปัญหาประกอบด้วย

-กำหนด ออบเจกต์ ในรูปของปัญหา (Problem Statement)

-กำหนดแมสเซสที่เกี่ยวข้องกับแต่ละ ออบเจกต์

-พัฒนาลำดับของแมสเซสที่มีไปยังออบเจกต์ เพื่อทำการแก้ปัญหา

4.ภาษาทางออบเจกต์ โอเรียลเต็ด โปรแกรมมิ่ง จะต้องมีคุณสมบัติ เอ็นแคปซูเลชั่น, แอ็บสแตรกชั่น , อินเฮอริแตนส์, โพลิมอฟีซึ่ม

5.หน่วยหนึ่งของเอ็นแคปซูเลชั่น คือ ออบเจกต์ ซึ่งประกอบด้วย Private data , Shared data และ แมสเซส

6.แอ็บสแตรกชั่น ถูกสนับสนุนโดยโครงสร้างลำดับชั้นของคลาส (hierarchy of class) ซึ่งใช้แทนแต่ละ ออบเจกต์ ที่ต่างชนิดกัน ; คลาสโปรโตคอล จะกำหนดคุณสมบัติของแต่ละแอ็บสแตรกชั่น; แอ็บสแตรกชั่นใหม่จะถูกเพิ่มโดยการเพิ่มซับคลาสใหม่

7.คุณสมบัติของซับคลาสที่อินเฮอริทซุปเปอร์คลาส ประกอบด้วย Private data , shared data และ แมสเซส

8. อินเฮอริท แมสเซส เป็นโพลิมอฟีซึ่มโดยซับคลาส

9. โพลิมอฟีซึ่ม หมายถึง การส่งแมสเซสที่เหมือนกัน ไปยัง คลาส ต่างกันได้

10.ดาต้า คือ ข้อมูลที่ถูกเก็บใน ออบเจกต์ และถูกลบโดยอัตโนมัติเมื่อไม่ใช้

11.สนับสนุน การเพิ่มขึ้นของการแก้ไขปัญหา

12.ไม่ว่าออบเจกต์ที่เกิดจากระบบหรือเกิดจากการที่ ผู้ใช้ กำหนดจะ มีสถานะเท่ากัน

13.ขยายการสนับสนุนในเรื่องต่าง ๆ ได้เต็มที่

14.การใช้โครงสร้างการทำซ้ำจะถูกทำให้สั้น โดยการแทนรายละเอียดในรูปของแมสเซสง่าย ๆ

2.2 ระบบผู้เชี่ยวชาญ (EXPERT SYSTEM)

ระบบผู้เชี่ยวชาญ คือ กลุ่มของโปรแกรมที่ประกอบขึ้นจากการรวบรวมกลุ่มของความรู้ประเภทต่างๆ เพื่อทำการแก้ปัญหาเฉพาะด้านนั้นๆ โดยอาศัยการตัดสินใจจากฐานความรู้ที่ระบบมีอยู่ ซึ่งความรู้นี้อาจจะได้มาจากผู้มีความชำนาญหรือมีความเชี่ยวชาญในสาขานั้นๆ เช่น ระบบผู้เชี่ยวชาญการรักษาโรคอาจได้มาจากหมอ ซึ่งก็เป็นผู้เชี่ยวชาญทางด้านการรักษาโรคนั่นเอง

ความรู้ที่ระบบได้รับมานั้นจะถูกนำมาผ่านขั้นตอนต่างๆ เพื่อทำการแปรสภาพ จัดระเบียบ จัดหมวดหมู่ จัดลำดับความสำคัญ รวมถึงศึกษาความสัมพันธ์ระหว่างข้อมูลนั้นๆ กับสิ่งอื่นๆ สุดท้ายแล้วข้อมูลที่ได้ก็จะถูกจัดเก็บไว้ในรูปของฐานความรู้ของระบบผู้เชี่ยวชาญที่เหมาะสมที่สุดสำหรับการนำมาเรียกใช้ในภายหลัง ทั้งในด้านการตอบคำถามหรือในด้านการให้เหตุผลข้อมูล โดยรูปแบบการจัดเก็บความรู้นั้นจะมีอยู่ด้วยกันหลายรูปแบบ เช่น เป็นข้อความ (TEXT) , NETWORK ฐานข้อมูล (DATABASE) ทั้งนี้ขึ้นอยู่กับความเหมาะสมกับวิธีการ จำนวนข้อมูล และข้อกำหนดอื่นๆ อันเป็นส่วนสำคัญสำหรับระบบนั้นๆ

ชนิดของระบบผู้เชี่ยวชาญที่มักจะพบอยู่อย่างสม่ำเสมอ เช่น ระบบผู้เชี่ยวชาญทางภูมิศาสตร์ ระบบผู้เชี่ยวชาญทางการแพทย์ ระบบผู้เชี่ยวชาญการใช้ยา ระบบผู้เชี่ยวชาญการออกแบบต่างๆ ทางด้านวิศวกรรม

คุณลักษณะที่สำคัญของระบบผู้เชี่ยวชาญมี 5 ข้อดังนี้

1. ระบบผู้เชี่ยวชาญจะมีการใช้ความรู้ที่ระบบได้รับเข้ามาทำการวิเคราะห์ตอบปัญหาหรือแก้ปัญหาต่างๆมากกว่าจะใช้ข้อมูลซึ่งได้รับมาโดยตรง ข้อแตกต่างข้อหนึ่งซึ่งเห็นได้ชัดระหว่างความรู้กับข้อมูลก็คือ ข้อมูลคือเรื่องราวข่าวสารต่างๆ ซึ่งยังไม่ได้ผ่านกระบวนการสรุปวิเคราะห์ ว่ามีข้อเท็จจริงอย่างไร แต่ความรู้คือการสรุปเรื่องราวข้อเท็จจริงต่างๆของข้อมูล ตลอดจนการนำข้อมูลมาผ่านกรรมวิธีต่างๆ เพื่อจัดระเบียบ จัดลำดับ และเก็บไว้ในรูปแบบที่เหมาะสมแก่การนำไปใช้งานระบบ ซึ่งอาจถูกเรียกว่าฐานความรู้ ซึ่งฐานความรู้นี่เองคือสิ่งที่ระบบผู้เชี่ยวชาญ จะนำไปใช้เป็นข้อมูลสำหรับการตัดสินใจตอบปัญหาต่างๆ
2. ในระบบผู้เชี่ยวชาญความรู้ที่ถูกจัดเก็บไว้ จะถูกเปลี่ยนแปลงให้เหมาะสมเพื่อนำไปใช้งานและทำการเก็บไว้โดยแยกเป็นอีกส่วนหนึ่งต่างหาก ออกมาจาก โปรแกรมควบคุม (CONTROL PROGRAM) โดยทั้งสองส่วนจะไม่ถูกนำมาคอมไพล์พร้อมกัน แต่ในการทำงานจริงๆ จะเป็นในลักษณะที่ โปรแกรมควบคุม จะตัดสินใจจากฐานความรู้ที่มันมีอยู่ และปัจจัยที่มาการะทบจากภายนอก ซึ่งนั้นก็คือ ข้อมูลที่ผู้ใช้ระบบป้อนเข้ามา ด้วยการตอบคำถามของระบบนั่นเอง
3. ระบบผู้เชี่ยวชาญจะต้องสามารถที่จะระบุเหตุผล ในการตอบคำถามต่างๆนั้นแก่ผู้ใช้ระบบ โดยตัวของระบบจะต้องให้เหตุผลกับผู้ใช้ได้ว่า ทำไมจากข้อมูลที่ได้รับจากผู้ใช้เข้าสู่ระบบจึงทำให้ได้ข้อสรุปเช่นนี้ ตลอดจนถึงจะต้องสามารถให้รายละเอียดเพิ่มเติมเท่าที่ตัวระบบทราบแก่ผู้ใช้ ซึ่งทั้งสองสิ่งนี้เป็นสิ่งสำคัญมากเพราะเป็นการทำให้ผู้ใช้สามารถตรวจสอบการทำงานของระบบตรวจสอบความผิดพลาดของข้อมูลที่ป้อนให้กับระบบ และยังช่วยในการพัฒนาโปรแกรมต่อไปในอนาคตอีกด้วย

4. ในระบบผู้เชี่ยวชาญจะใช้สัญลักษณ์ เพื่อใช้แทนหรือใช้อธิบายความรู้เช่นการสรุปเป็นกฎ เป็นรูปแบบของต้นไม้เพื่อการตัดสินใจ หรือในรูปแบบอื่นๆแล้วแต่กรณี เพื่อความเข้าใจและง่ายต่อการนำไปใช้งานของระบบเอง
5. ระบบผู้เชี่ยวชาญจะให้ผลสรุปหรือคำตอบ แก่ผู้ใช้ตามความรู้ที่ตัวระบบมีอยู่ด้วยตัวของมันเองและมากที่สุด เท่าที่ความรู้ที่เก็บอยู่ในระบบ จะสามารถวิเคราะห์และสรุปออกมาได้

2.3 ไอดี3 (ID3)

วิธีการอนุมานโดยดูจากตัวอย่าง หรือกล่าวอีกอย่างหนึ่งว่าวิธีการสร้าง ต้นไม้สำหรับการตัดสินใจจากชุดฝึกหัด โดยวิธีการไอดี3 นั้น คิดค้นโดย J.R.Quinlan โดยพัฒนามาจาก CLS (Concept Learning System) ของ E.B.Hunt ซึ่งเป็นวิธีการเรียนรู้สมัยต้น ๆ ประมาณปี ค.ศ. 1963

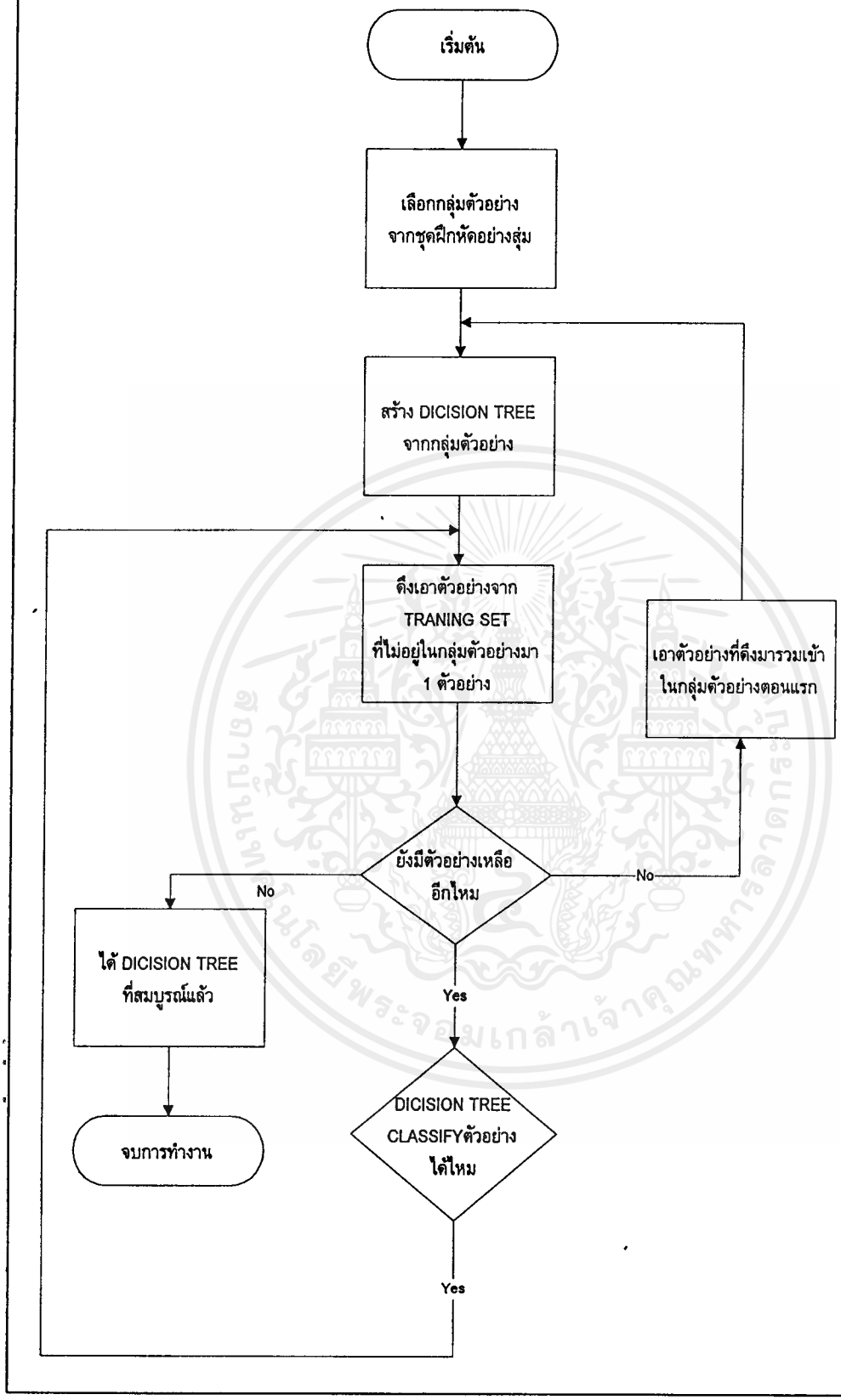
วิธีการ ไอดี3 นั้นมีขั้นตอนในการสร้างดังต่อไปนี้

1. ทำการเลือกกลุ่มตัวอย่าง จากชุดตัวอย่างโดยสุ่มโดยเรียกกลุ่มตัวอย่างนี้ว่า วินโดว์ (window)

2. สร้างต้นไม้สำหรับการตัดสินใจจากวินโดว์นั้น ซึ่งต้นไม้สำหรับการตัดสินใจที่ได้ จะสามารถแยกประเภทตัวอย่างได้ครอบคลุมวินโดว์ทั้งหมด

3. นำต้นไม้สำหรับการตัดสินใจจากข้อ 2 ไปลองทดสอบแยกประเภท ตัวอย่างในชุดฝึกอื่นๆ ที่ไม่ได้อยู่ในวินโดว์ทีละตัวอย่าง การใช้ต้นไม้สำหรับการตัดสินใจทดสอบแยกประเภทตัวอย่างก็โดยพิจารณาดูว่าตัวอย่างที่ กำลังทดสอบนั้นมีประเภทเช่นเดียวกันกับการใช้ต้นไม้สำหรับการตัดสินใจ ในการวินิจฉัยปัญหาหรือไม่ ถ้าได้ประเภทเดียวกันแล้วแสดงว่า ต้นไม้สำหรับการตัดสินใจ นั้นสามารถแยกแยะประเภทของตัวอย่างนั้นได้อย่างได้อย่างถูกต้อง มิฉะนั้นก็ถือว่าแยกประเภทไม่ได้ ถ้าทดสอบแล้วสามารถแยกประเภทได้อย่างถูกต้อง ก็จะทยอยเอาตัวอย่างที่เหลือ ไปทดสอบแยกประเภทอีกจนกว่าจะหมด เมื่อลองทดสอบจนไม่เหลือตัวอย่างที่จะให้ทดสอบอีกแล้ว ก็จะได้ต้นไม้ที่ใช้ทดสอบนั้นเป็นต้นไม้ที่ได้จากการอุปมานจาก ชุดฝึกหัดทั้งหมด ซึ่งก็จะสิ้นสุดขบวนการของการสร้างต้นไม้ มิฉะนั้นแล้วจะ นำเอาตัวอย่างที่แยกประเภทแล้วย้อนไปทำขั้นตอนที่ 2 ใหม่อีก

ขบวนการต่าง ๆ ในการสร้างต้นไม้จากชุดฝึกหัด แสดงได้ดังโฟล์วชาร์ต รูปที่ 2-9



รูปที่ 2-9 แสดง Flowchart ขั้นตอนหรือวิธีการสร้างต้นไม้สำหรับการตัดสินใจ

การสร้างต้นไม้ สำหรับการตัดสินใจโดยวิธีของ ไอดี3

ไอดี3 ได้ประยุกต์ใช้ทฤษฎีข่าวสาร (Information Theory) ในการเลือก แอทริบิวท์ เพื่อสร้างต้นไม้ ซึ่งต้นไม้สำหรับการตัดสินใจที่ได้จะมีขนาดเล็ก จากที่ได้กล่าวถึงลักษณะของต้นไม้สำหรับการตัดสินใจ ที่ใช้ในการวินิจฉัยประเภทของตัวอย่างแล้วว่า แต่ละ node จะแทนด้วยค่าของ แอทริบิวท์ และกิ่งแทนด้วย ค่าของ แอทริบิวท์ ส่วนปลายกิ่งสุดท้าย (leaf) ก็จะแทนประเภท ของตัวอย่าง

ถ้าพิจารณาโดยอาศัยทฤษฎีข่าวสารแล้วจะเห็นว่า โครงสร้างของต้นไม้สำหรับการตัดสินใจมีลักษณะคล้ายกันกับการเก็บข้อมูลสำหรับถาม คำถามเพื่อค้นหาข่าวสารนั่นเองโดยที่แอทริบิวท์ต่าง ๆ ที่แทนโนด (node) ของต้นไม้ จะเป็นเสมือนคำถามที่ถามเพื่อรับเอาข่าวสาร ซึ่งคำตอบที่ได้รับมาก็เป็นค่าของ แอทริบิวท์ที่จะนำไปเลือกคำถามต่อไป ซึ่งเป็นการถามเพื่อหาค่าของแอทริบิวท์ถัดไป จนกว่าจะพบข่าวสาร ซึ่งก็คือประเภทของตัวอย่างนั่นเอง

การสร้างต้นไม้สำหรับการตัดสินใจให้มีขนาดเล็กนั้นขึ้นอยู่กับการเลือกแอทริบิวท์เพื่อมาเป็น โนดของต้นไม้เป็นสำคัญ การเลือกแอทริบิวท์ที่ไม่ดีจะทำให้เกิดมีแอทริบิวท์ที่ซ้ำกันเกิดขึ้นเป็นโนด ในต้นไม้หลายแห่ง ต้นไม้ที่ได้จะมีขนาดใหญ่ การเลือกแอทริบิวท์ที่เหมาะสมจะทำให้จำนวนคำถามน้อยลง ลดการซ้ำซ้อนของแอทริบิวท์ในต้นไม้ ซึ่งทำให้ต้นไม้สำหรับการตัดสินใจมีขนาดเล็ก

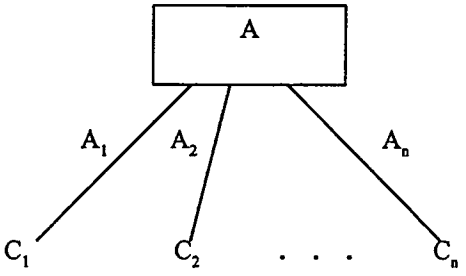
จากทฤษฎีข่าวสารจะเห็นว่าขนาดของต้นไม้สำหรับการตัดสินใจ ขึ้นอยู่กับขนาดของข่าวสาร ข้อมูลที่ต้นไม้สำหรับการตัดสินใจนั้นมีอยู่ถ้าข่าวสารมีจำนวนมากนั้น หมายความว่าต้องใช้จำนวนคำถามมาก จึงสามารถค้นพบข่าวสารที่ต้องการได้

ซึ่งส่งผลให้ต้นไม้สำหรับการตัดสินใจมีขนาดใหญ่ และในทางตรงกันข้ามถ้าข่าวสารที่ ต้นไม้สำหรับการตัดสินใจนั้นเก็บอยู่มีจำนวนน้อย ต้นไม้สำหรับการตัดสินใจจะมีขนาดเล็ก ซึ่งหมายความว่าถามคำถามจำนวนน้อยคำถาม ก็จะได้คำตอบที่ต้องการ เพราะฉะนั้นชุดฝึกหัดซึ่งใช้เป็นตัวอย่างชุดเดียวกันเราจึงพยายามทำให้ได้ต้นไม้สำหรับการตัดสินใจ ต้นที่เตี้ยที่สุด

วิธีการสร้าง ต้นไม้สำหรับการตัดสินใจ ของ ไอดี3 จะสมมุติว่า ประเภทของตัวอย่างในชุดฝึกหัด มีเพียง 2 ประเภท เท่านั้นคือ ประเภท "P"กับประเภท "N" โดยที่ P และ N แทนจำนวนตัวอย่างทั้งหมดในชุดฝึกหัด ที่มีประเภทเป็น "P" และ "N" ตามลำดับ ดังนั้นต้นไม้สำหรับการตัดสินใจนี้จะให้ข่าวสารที่เป็น "P" หรือ "N" เท่านั้นซึ่งค่าข่าวสารที่คาดว่าจะใช้ในการให้ข่าวสาร (expected Information) "P" หรือ "N" จากต้นไม้สำหรับการตัดสินใจ จะมีค่าดังต่อไปนี้

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

ถ้าให้ค่า แอทริบิวท์ A ซึ่งมีค่า (value) ต่างๆคือ [A1 A2 A3 .....A_n ขณะเดียวกันก็ให้ แอทริบิวท์เป็น root node ของ ต้นไม้สำหรับการตัดสินใจ ค่าของแอทริบิวท์ A จะแยกต้นไม้สำหรับการตัดสินใจ ออกเป็น ต้นไม้สำหรับการตัดสินใจย่อย (Subtree) คือ [C1 C2 C3.....C_n] โดยที่ Ci หมายถึง ต้นไม้สำหรับการตัดสินใจย่อยใดๆ ที่ตัวอย่างที่มีค่าของ แอทริบิวท์ A เป็น Ai ดังรูปที่ 3.6



รูปที่ 2-10 แสดงการแยกของต้นไม้ย่อย โดยค่าของ แอทริบิวท์ A

สมมุติว่า ต้นไม้สำหรับการตัดสินใจย่อย Ci นั้น มีตัวอย่างที่มี ประเภทเป็น "P" อยู่ Pi ตัวอย่าง และมีตัวอย่างที่มี ประเภทเป็น "N" อยู่ Ni ตัวอย่าง ดังนั้นค่าข่าวสาร ที่คาดว่าจะได้รับจาก ต้นไม้สำหรับการตัดสินใจย่อย Ci จะเป็น I[Pi,Ni] จากนั้นการคำนวณค่าข่าวสารที่ได้รับจากต้นไม้สำหรับการตัดสินใจทั้งหมด โดยมี แอทริบิวท์ A เป็นรูทโนด (Root node) จะสามารถหาได้จากการเฉลี่ยค่า ข่าวสาร ที่คาดว่าจะได้รับจากทุกต้นไม้สำหรับการตัดสินใจย่อยดังสมการต่อไปนี้

$$E(A)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I(p_i,n_i)$$

จากความจริงที่ว่า ต้นไม้สำหรับการตัดสินใจใดมีค่าข่าวสารน้อยก็แสดงว่าต้นไม้สำหรับการตัดสินใจนั้นมีขนาดเล็ก ดังนั้นการเลือกแอทริบิวท์ที่เป็นรูทโนดของต้นไม้สำหรับการตัดสินใจ จึงอาศัยความจริงในข้อนี้ โดยจะทำการเลือก แอทริบิวท์ใดๆ ก็ตามที่เป็นรูทโนดที่ทำให้ค่าข่าวสารที่คาดว่าจะได้รับจากต้นไม้สำหรับการตัดสินใจ มีขนาด ของ E(A) มีค่าน้อยที่สุดในบรรดา E(A) ด้วยกัน ทั้งนี้เพื่อเป็นการรับประกันว่าต้นไม้สำหรับการตัดสินใจที่ได้จะ เป็นต้นที่เตี้ยที่สุด

สำหรับขั้นตอนการสร้างต้นไม้สำหรับการตัดสินใจของไอดี3 จะเริ่มการจากการคำนวณค่าของ E(A) แต่ละตัวของแต่ละแอทริบิวท์ในชุดฝึกหัดแล้วเลือกเอาเฉพาะแอทริบิวท์ที่ให้ค่าของ E(A) น้อยที่สุดมาเป็นรูทโนด ของต้นไม้สำหรับการตัดสินใจ จากนั้นแบ่งชุดฝึกหัดตามกลุ่มของตัวอย่างที่มีค่าตามค่าของ แอทริบิวท์ที่เลือกมานั้นออกเป็นชุดฝึกหัดย่อยๆ ที่แยกโดยใช้ A1 A2 A3.....An ก็จะเป็น ต้นไม้สำหรับการตัดสินใจย่อยของ C1 C2 C3.....Cn ตามลำดับ เช่น ชุดฝึกหัดของ C1 ก็จะเป็นกลุ่มตัวอย่างทั้งหมดของชุดฝึกหัดเดิม ที่มีค่าของ แอทริบิวท์เป็น A1 เป็นต้น

จากนั้นทำการสร้างต้นไม้สำหรับการตัดสินใจย่อย C1 C2 C3....Cn โดยใช้ชุดฝึกหัดย่อยของแต่ละต้นไม้สำหรับการตัดสินใจย่อย ด้วยวิธีการเดียวกัน แต่ตอนนี้การเลือกแอทริบิวท์ จะไม่พิจารณาแอทริบิวท์ A อีกต่อไปเมื่อเลือกแอทริบิวท์ที่จะมาเป็นรูทโนดของต้นไม้สำหรับการตัดสินใจย่อย ได้แล้วก็สามารถจะแยกออกเป็นต้นไม้สำหรับการตัดสินใจย่อยต่อไปได้อีก ทำเช่นนี้ไปเรื่อยจนกว่าชุดฝึกหัดของต้นไม้ย่อยใดๆเป็น

ประเภทเดียวกันทั้งหมด ตอนนี้จึงสามารถแทนที่ [illegible] การตัดสินใจย่อย นั้นด้วยประเภทของ คลาส ได้เลย นั่นก็คือเราได้ปลายสุดของต้นไม้สำหรับการตัดสินใจ (leaf node) แล้วนั่นเอง

ตัวอย่างการสร้างต้นไม้สำหรับการตัดสินใจโดยวิธีการของไอดี3 จากตารางชุดฝึกหัดในตารางที่1-2 เป็นกลุ่มตัวอย่างอย่างง่ายที่ให้โดยผู้เชี่ยวชาญเพื่อต้องการการอุปมาน ต้นไม้สำหรับการตัดสินใจที่ใช้ในการวินิจฉัยสภาพอากาศแต่ละตัวอย่างจะอธิบายด้วยแอทริบิวท์ Outlook, Temperature, Humidity และ Windy มีทั้งหมด 14 ตัวอย่างส่วนประเภท คลาส ของตัวอย่างนั้นสมมติให้มีเพียง 2 ประเภท คือ "P", "N"

EX No.	ATRIBUTES				CLASS
	outlook	temperature	humidity	windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

ตารางที่ 2-2 แสดงชุดฝึกหัดของตัวอย่างที่ใช้ในการวินิจฉัยสภาพอากาศ

การสร้างต้นไม้สำหรับการตัดสินใจ ก็เริ่มจากการเลือกแอทริบิวท์เพื่อมาเป็นรูทโนดของต้นไม้สำหรับการตัดสินใจ โดยคำนวณหาค่า E(A) ของแต่ละแอทริบิวท์จะได้ว่า

สำหรับแอทริบิวท์ Outlook ซึ่งมีค่าเป็น [Sunny, Overcast, Rain]

ตัวอย่างที่มีค่าของแอทริบิวท์ "Outlook" เป็น "Sunny" มีประเภทเป็น"P" และ "N"เป็นจำนวน 2 และ 3 ตัวอย่างดังนั้น

P1 = 2, n1 = 3 I(P1,N1) = 0.971

P2 = 4, n2 = 0 I(P2,N2) = 0

P3 = 3, n3 = 2 I(P3,N3) = 0.971

จะสามารถหาค่าของ E(Outlook)ได้

โดย E(Outlook) = 5/14*I(P1,N1) + 4/14*I(P2,N2) + 5/14*I(P3,N3) = 0.694 bits

สำหรับ E(A) ของค่าอื่นมีดังนี้

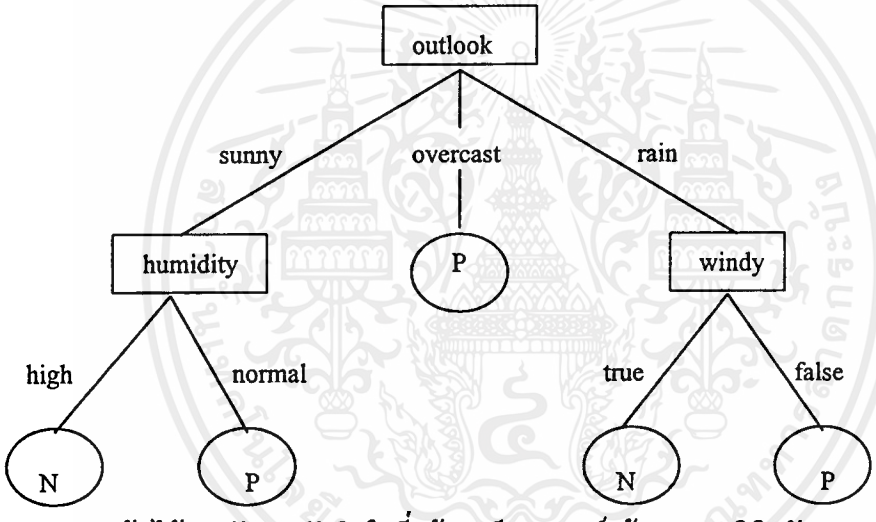
E(temperature) = 0.911bits

E(humidity) = 0.789bits

E(windy) = 0.892bits

โดยการพิจารณาค่า E(A) ของแต่ละแอทริบิวท์แล้วจะเลือกแอทริบิวท์ Outlook มาเป็น รูทโนด ของต้นไม้สำหรับการตัดสินใจ เนื่องจากให้ค่า E(A) น้อยที่สุด

จากนั้นชุดฝึกหัดจะถูกแบ่งเป็นชุดฝึกหัดย่อยๆ ตามแต่ละค่าของ Outlook แล้ว แต่ละชุดฝึกหัดย่อยๆ จะก็จะถูกใช้สร้างต้นไม้สำหรับการตัดสินใจย่อย โดยวิธีการที่ได้แสดงไปแล้ว ต้นไม้สำหรับการตัดสินใจ ที่สมบูรณ์แสดงดังรูปที่ 1.11



รูปที่ 2-11 แสดงต้นไม้สำหรับการตัดสินใจที่สร้างเสร็จสมบูรณ์แล้วของการวินิจฉัยสภาพอากาศ

2.3.1 การปรับปรุง ไอดี3

เนื่องจากชุดฝึกหัดบางชุดไม่สามารถใช้วิธีการของไอดี3มาใช้ในการอนุมานได้ ในกรณีที่การแบ่งชุดฝึกหัดนั้นเป็นชุดย่อยๆแล้วมีบางค่าของแอทริบิวท์ ไม่ปรากฏในชุดฝึกหัดย่อยๆนั้นๆ ดังเช่นตัวอย่างการสร้างต้นไม้สำหรับการตัดสินใจ จากชุดฝึกหัดต่อไปนี้

HUMAN	EYE_LAYER	SKIN	HAIR	HIGH	EYE_COLOR	NOSE	CLASS
1	1_layer	yellow	black	middle	black	sharp	asian
2	1_layer	white	black	short	black	sharp	asian
3	1_layer	white	black	middle	black	sharp	asian
4	2_layer	yellow	black	middle	black	sharp	asian
5	2_layer	yellow	black	middle	brown	sharp	asian
6	2_layer	yellow	black	middle	black	flat	asian
7	2_layer	white	brown	tall	blue	sharp	asian
8	2_layer	black	black	tall	black	sharp	western
9	2_layer	white	black	tall	blue	sharp	western
10	2_layer	yellow	black	tall	blue	sharp	western
11	2_layer	black	black	tall	black	flat	african
12	2_layer	black	black	short	black	flat	african

ตารางที่ 2-3 แสดงชุดฝึกหัดของตัวอย่างที่ใช้ในการวินิจฉัยลักษณะมนุษย์ในทวีปต่างๆ

จากชุดฝึกหัดนี้ เป็นตัวอย่างของการแบ่งประเภทของชาวเอเชีย ชาวตะวันตก และชาวแอฟริกา โดยอาศัยลักษณะทางกายภาพ เป็นแอทริบิวท์ในการแบ่งประเภทดังนี้

EYE_LAYER : บอกว่ามีตาชั้นเดียวหรือสองชั้น(1_layer,2_layer)

SKIN : บอกสีผิวในที่นี้คือ ผิวเหลือง ผิวขาว ผิวดำ

HAIR : บอกสีผม ซึ่งมีสีดำ สีน้ำตาล และ สีบรอนซ์

HIGHT : มี 3ระดับคือเตี้ย ปานกลาง สูง

EYE_COLOR : มีสีฟ้า สีดำ สีน้ำตาล

NOSE : มีค่าเป็นจมูกโด่งและจมูกแบน

การสร้างต้นไม้สำหรับการตัดสินใจ ก็ทำได้เช่นเดียวกันกับวิธีการที่ได้กล่าวไว้ข้างต้น โดยเริ่มจากการเลือก แอทริบิวท์มาเป็นรูทโนดของต้นไม้สำหรับการตัดสินใจเสียก่อน ซึ่งต้องทำการหาค่า E(A) ของแต่ละแอทริบิวท์ ในที่นี้ค่า E(A) หาได้จากสูตรดังต่อไปนี้

$$E(A) = \sum_{i=1}^{v} \frac{a_i + w_i + af_i}{a + w + af} I(a_i, w_i, af_i)$$

โดยที่

$$I(a,w,af) = -\left(\frac{a}{a+w+af}\log_2\frac{a}{a+w+af} + \frac{w}{a+w+af}\log_2\frac{w}{a+w+af} + \frac{af}{a+w+af}\log_2\frac{af}{a+w+af}\right.$$

a,w และ af คือจำนวนตัวอย่างที่มีประเภทเป็น asian, western, african ตามลำดับ

พิจารณาที่แอทริบิวท์ EYE_LAYER ซึ่งมีค่าเป็น {1_LAYER,2_LAYER}

a1=3 , w1=0 , af1=0 I(a1,w1,af1) = 0

a2=3 , w2=0 , af2=0 I(a2,w2,af2) = 1.53

E(EYE_LAYER) = 3/12 * I(a1,w1,af1) + 9/12 * I(a2,w2,af2)

=1.148 bits

E(SKIN) = 0.864 bits

E(HAIR) = 1.142 bits

E(HIGHT) = 0.468 bits

E(EYE_COLOR) = 0.866 bits

E(NOSE) = 0.980 bits

เนื่องจาก E(HIGHT) ต่ำที่สุดจึงเลือกแอทริบิวท์ HIGHT เป็นรูทโนดของต้นไม้สำหรับการตัดสินใจ จะพบว่าจากขบวนการที่ผ่านมายังสามารถใช้วิธีการของไอดี3ได้ แต่หลังจากนี้จะพบข้อจำกัดของไอดี3

เมื่อได้รูทโนดคือแอทริบิวท์ HIGHT แล้วใช้ค่า short,middle,tall แบ่งชุดฝึกหัดย่อยตามค่าเหล่านี้ จะพบว่าตัวอย่างที่แยกโดยใช้ชุดฝึกหัดย่อย middle มีค่าเป็น asian ทั้งหมด ดังนั้นเราแทนปลายกิ่งสุดท้ายได้ด้วยประเภทของ asian ได้เลย

ชุดฝึกหัดย่อย short แยกออกมาเป็นดังนี้

HUMAN	EYE_LAYER	SKIN	HAIR	HIGH	EYE_COLOR	NOSE	CLASS
2	1_layer	white	black	short	black	sharp	asian
12	2_layer	black	black	short	black	flat	african

ตารางที่ 2-4 แสดงชุดฝึกหัดย่อยของแอทริบิวท์ High ที่เป็น short

จากชุดฝึกหัดย่อยสามารถสร้างต้นไม้สำหรับการตัดสินใจย่อย เชื่อมต่อกับ short ได้ดังนี้

เลือกแอทริบิวท์รูทโนดคำนวณค่า E(A) ของแอทริบิวท์ที่เหลือ เริ่มพิจารณาที่แอทริบิวท์ EYE_LAYER (1_layer,2_layer)

a1 = 1 , w1 = 0 , af1 = 0 I(a1 , w1, af1) = 0

a2 = 0 , w1 = 0 , af2 = 1 I(a2 , w2, af2) = 0

E(EYE_LAYER) = 0 bits

สำหรับแอทริบิวท์ SKIN มีค่าเป็น (yellow, white,black)

a1 = 0 , w1 = 0 , af1 = 0 I(a1 , w1, af1) = หาค่าไม่ได้

a2 = 1 , w1 = 0 , af2 = 1 I(a2 , w2, af2) = 0

a3 = 0 , w3 = 0 , af3 = 1 I (a3 , w3, af3) = 0

สำหรับกรณีที่หาค่าได้เกิดขึ้นเนื่องจากชุดฝึกหัดย่อยนั้นไม่มีตัวอย่างใดที่มีค่าแอทริบิวท์ SKIN เป็น yellow อยู่เลย ไอดี3 ตรงส่วนนี้จึงไม่สามารถสร้างต้นไม้สำหรับการตัดสินใจย่อยได้อีกต่อไป การแก้ไขก็ทำได้โดยการตัดค่าที่เป็น yellow ออกจากค่าที่เป็นไปได้ของ แอทริบิวท์ SKIN ก่อนที่จะใช้วิธีเดิมของไอดี3 โดยให้ไอดี3 มองเห็นว่าค่าที่เป็นไปได้ของ แอทริบิวท์ SKIN ในขณะนี้มีเพียง white และ black เท่านั้น ซึ่งขบวนการสร้างต้นไม้โดยไอดี3 ก็จะสามารถดำเนินการได้ต่อไป ซึ่งตอนนี้จะมีเพียง I (a2 , w2, af2) และ I (a3 , w3, af3) ซึ่งเป็น 0 ทั้งคู่ ดังนั้นจะได้

E(SKIN) = 0 bits

สำหรับแอทริบิวท์ HAIR ก็เช่นเดียวกัน คือ brown และ bronzeจะถูกตัดไปทำให้ค่าของแอทริบิวท์ HAIR เป็น black เท่านั้น ซึ่งจะได้

a1=1 , w1=0 , af1=0 I(a1,w1,af1) = 1

นั่นคือจะได้

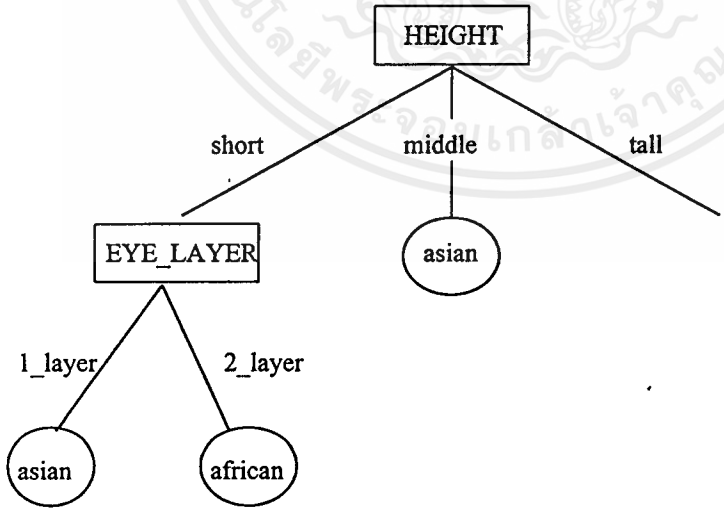
E(HAIR) = 1 bits

และด้วยวิธีเดียวกันจะได้

E(NOSE) = 0 bits

ทำให้ตอนนี้เรามี E(A) ที่น้อยที่สุด 3 ค่าคือ EYE_LAYER SKIN NOSE แต่โดยวิธีการของไอดี3 ไม่ได้มีการกำหนดว่าในกรณีจะมีวิธีในการพิจารณาเลือกรูทโนด อย่างไรจึงขอใช้ค่าแรกที่ระบบได้รับเป็นรูทโนดไปก่อน (ในทางปฏิบัติจริง อาจใช้การถ่วงน้ำหนักที่ค่าแอทริบิวท์แต่ละตัวตามแต่ cost ในการได้มาซึ่งแอทริบิวท์ตัวนั้น ซึ่งไม่ขอกล่าวถึงในที่นี้)

จากนั้นจึงดำเนินการสร้าง tree ตามวิธีการเดิม ก็จะได้ ต้นไม้สำหรับการตัดสินใจ ดังรูป



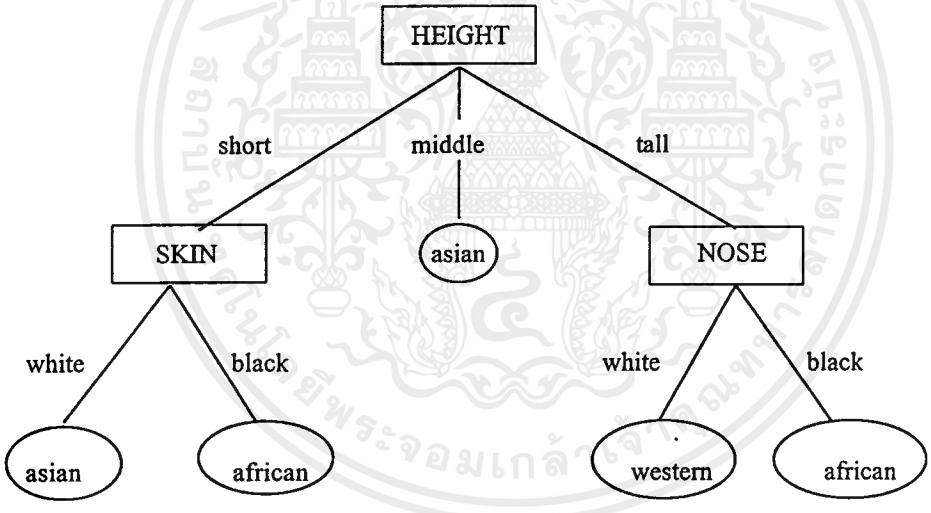
รูปที่ 2-12 แสดงต้นไม้สำหรับการตัดสินใจการวินิจฉัยลักษณะมนุษย์ที่แอทริบิวท์ High เป็น Short

หลังจากสร้างต้นไม้สำหรับการตัดสินใจย่อย ที่กิ่ง middle และ short แล้ว ต่อไปก็จะเป็นการสร้างต้นไม้สำหรับการตัดสินใจย่อยที่กิ่ง tall ซึ่งข้อมูลที่ได้นี้จะมาจากค่าของ tall ในแอทริบิวท์ HIGHT นั่นเอง แบ่งชุดฝึกหัดย่อยจากชุดฝึกหัดเดิม จะได้ชุดฝึกหัดย่อยดังนี้

HUMAN	EYE_LAYER	SKIN	HAIR	HIGH	EYE_COLOR	NOSE	CLASS
7	2_layer	white	brown	tall	blue	sharp	asian
8	2_layer	black	black	tall	black	sharp	western
9	2_layer	white	black	tall	blue	sharp	western
10	2_layer	yellow	black	tall	blue	sharp	western
11	2_layer	black	black	tall	black	flat	african

ตารางที่ 2-5 แสดงชุดฝึกหัดย่อยของแอทริบิวท์ High ที่เป็น tall

และโดยวิธีการเดียวกันจะสามารถสร้าง tree ที่สมบูรณ์ได้ดังรูป



รูปที่ 2-13 แสดงต้นไม้สำหรับการตัดสินใจที่สร้างเสร็จสมบูรณ์แล้วของการวินิจฉัยลักษณะมนุษย์

<u>ตัวอย่าง การคำนวณหารูทโนดของไอดี3</u>

การพัฒนาต้นไม้สำหรับการตัดสินใจ (decision tree) จากตัวอย่างข้อมูล จากตัวอย่าง aircraft identification ใช้วิธีของการวัด เอนโทรปี (entropy) ของแต่ละแอทริบิวท์ยิ่งเอนโทรปีของแอทริบิวท์ สูง ก็จะยิ่งมีความไม่แน่นอนของค่ามากขึ้น ดังนั้นเราจะเลือกแอทริบิวท์ที่มาเป็นรูทโนดโดยเลือกจาก แอทริบิวท์ที่มีเอนโทรปีน้อยที่สุด

โดยจะมีสูตรการหาเอนโทรปีดังนี้

$$H(C|A_k) = \sum_{j=1}^{M_k} p(a_{k,j}).\left[- \sum_{i=1}^{N} p(c_i|a_{k,j}).\log_2 p(c_i|a_{k,j}) \right]$$

EX. Investment Data Set

Mutual fund - type	Interested rates	Cash available	Tension	Fund value (Class)
Blue chip stock	High	High	Medium	Medium
Blue chip stock	Low	High	Medium	High
Blue chip stock	Medium	Low	High	Low
Gold stock	High	High	Medium	High
Gold stock	Low	High	Medium	Medium
Gold stock	Medium	Low	High	Medium
Mortgage related	High	High	Medium	Low
Mortgage related	Low	High	Medium	High
Mortgage related	Medium	Low	High	Low

ตารางที่ 2-6 แสดงชุดฝึกหัดของตัวอย่างที่ใช้ในการวินิจฉัย Investment Data Set

โดยที่

$H(C|A_k)$ = Entropy of the classification property of attribute A_k

$p(a_{k,j})$ = Probability of attribute k being at value j

$p(c_i|a_{k,j})$ = Probability that the class value is c_i when attribute k is at its jth value

M_k = Total number of values for attribute A_k ; $j = 1, 2, \ldots, M_k$

N = Total number of different (or outcomes) ; $j = 1, 2, \ldots, N$

K = Total number of attribute ; $k = 1, 2, \ldots, K$

- มี 4 แอทริบิวท์ (mutual fund-type , interest , cash , tension); ดังนั้น $K = 4$
- มี 3 คลาส (Fund value มี High , Medium , Low); ดังนั้น $N = 3$
- ที่ แอทริบิวท์ *mutual fund - type* มีค่า 3 ค่า (blue chip , gold , mortgage); ดังนั้น $M_1 = 3$
- ที่ แอทริบิวท์ *interest* มีค่า 3 ค่า (High , Medium , Low); ดังนั้น $M_2 = 3$
- ที่ แอทริบิวท์ *cash* มีค่า 2 ค่า (High , Low); ดังนั้น $M_3 = 2$
- ที่ แอทริบิวท์ *tension* มีค่า 2 ค่า (High , Medium); ดังนั้น $M_4 = 2$

ตัวอย่างการหาเอนโทรปีของแอทริบิวท์ *cash*

$p(a_{3,1})$ = probability that cash is high = 6/9

$p(a_{3,2})$ = probability that cash is low = 3/9

$p(c_1|a_{3,1})$= probability that fund value is high when cash is high =3/6

$p(c_2|a_{3,1})$= probability that fund value is medium when cash is high =2/6

$p(c_3|a_{3,1})$= probability that fund value is low when cash is high =1/6

$p(c_1|a_{3,2})$= probability that fund value is high when cash is high =0/3

$p(c_2|a_{3,2})$= probability that fund value is medium when cash is high =1/3

$p(c_1|a_{3,2})$= probability that fund value is low when cash is high =2/3

แทนในสมการได้

$$H(C|cash) = (6/9) . [-3/6.\log_2(3/6) -2/6.\log_2(2/6) -1/6.\log_2(1/6)] + (3/9) . [-0/3.\log_2(0/3) -1/3.\log_2(1/3) -2/3.\log_2(2/3)] = 1.2787$$

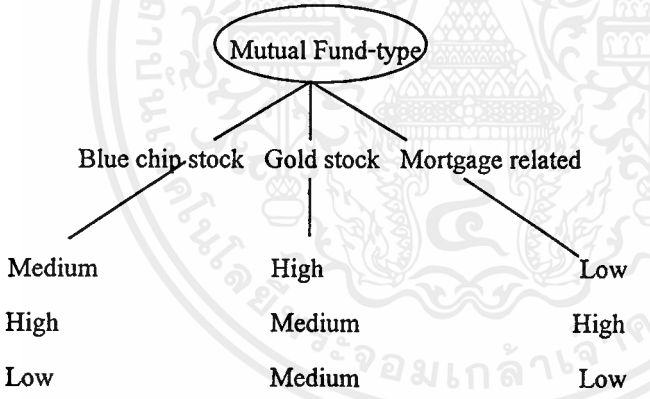
$$H(C|interest) = 1.140333$$

$$H(C|tension) = 1.2787$$

$$H(C| \text{ mutual fun-type}) = 1.140333$$

ซึ่งจะได้ค่าต่ำสุด 2 ค่า ระหว่าง *interest* กับ *mutual fun-type*

ถ้าแตกโครงสร้างของต้นไม้สำหรับการตัดสินใจ โดยใช้แอทริบิวท์ Mutual Fund-type เป็นรูทโนดจะได้ค่าของ คลาส ต่าง ๆ ดังรูปต้นไม้สำหรับการตัดสินใจ รูปที่ 1-13



รูปที่ 2-14 แสดงต้นไม้สำหรับการตัดสินใจของการวินิจฉัย Investment Data Set ที่แอทริบิวท์ Mutuall fund-type เป็น รูทโนด

ซึ่งจากตัวอย่าง ถ้า Mutual Fund-type เท่ากับ Blue chip stock จะได้ คลาส เป็น Medium, High, Low และค่าของ แอทริบิวท์อื่น ๆ เป็นดังตาราง

Interested rates	Cash	Tension	Fund value
High	High	Medium	Medium
Low	High	Medium	High
Medium	Low	High	Low

ตารางที่ 2-7 แสดงชุดฝึกหัดของตัวอย่างที่ Classifications เมื่อ Mutual fund-type เป็น Blue chip stock

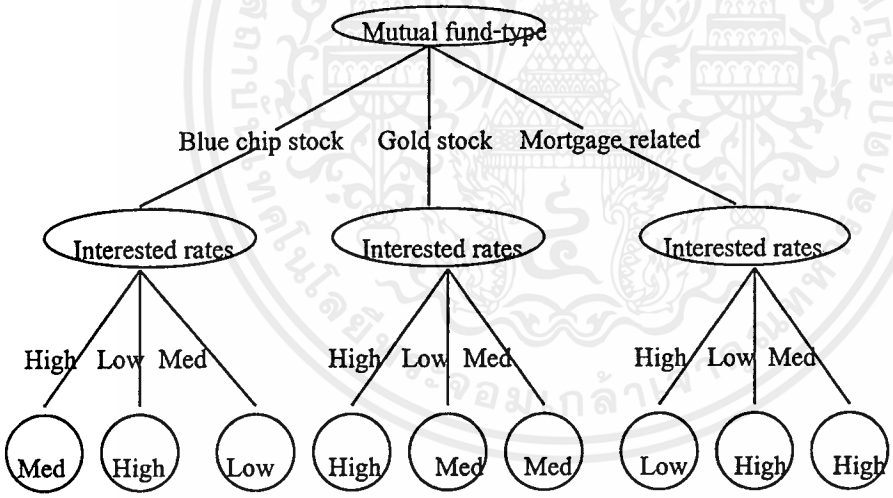
Interested rates	Cash	Tension	Fund value
High	High	Medium	Medium
Low	High	Medium	High
Medium	Low	High	Low

ตารางที่ 2-8 แสดงชุดฝึกหัดของตัวอย่างที่ Classifications เมื่อ Mutual fund-type เป็น Gold stock

Interested rates	Cash	Tension	Fund value
High	High	Medium	Medium
Low	High	Medium	High
Medium	Low	High	Low

ตารางที่ 2-9 แสดงชุดฝึกหัดของตัวอย่างที่ Classifications เมื่อ Mutual fund-type เป็น Mortgage related

ดังนั้น จากตารางจะเห็นว่า หาค่าเพียงแค่ 2 แอทริบิวท์ คือ Mutual fund-type กับ Interested rates ก็สามารถหาค่าของคลาสได้ ดังนั้นจะได้ ต้นไม้สำหรับการตัดสินใจดังรูปที่ 1-14



รูปที่ 2-15 แสดงต้นไม้สำหรับการตัดสินใจของการวินิจฉัย Investment Data Set ที่เสร็จสมบูรณ์แล้ว

บทที่3

ภาษาสมอลทอกค์ (Smalltalk)

ภาษาสมอลทอกค์ เกิดขึ้นในราวปี 1970 เป็นภาษาพื้นฐานของการเขียนโปรแกรมแบบ ออบเจกต์ โอเรียลเต็ด ซึ่งภาษาสมอลทอกค์นับว่าเป็น ภาษาออบเจกต์ โอเรียลเต็ด ที่สมบูรณ์แบบที่สุดจนถึงกับกล่าวได้ว่าเป็น

" The purest of pure Object - Orienţed language "

เนื่องจากภาษาสมอลทอกค์เป็นภาษาที่ไม่มีนิยามของคำว่า 'ดาต้า' มีแต่ 'ออบเจกต์' จะกระทำการใดๆ กับออบเจกต์ในโปรแกรมได้โดยผ่านแมสเซสเท่านั้น ทุกอย่างของภาษาสมอลทอกค์ไม่ว่าจะเป็น คำสั่งของภาษา,บล็อกของโปรแกรม (block of code program) หรือแม้แต่เอดิเตอร์ (editor) ในคอมไพเลอร์ ก็ล้วนแต่เป็นออบเจกต์ทั้งสิ้น

เมื่อเรียกโปรแกรมของภาษาสมอลทอกค์ ขึ้นมาทำงานก็จะพบกับหลักการของออบเจกต์ โอเรียลเต็ดในทุก ๆ ส่วนของโปรแกรม ตั้งแต่การติดต่อกับผู้ใช้แบบกราฟิกที่เรียกว่าการติดต่อกับผู้ใช้ (interface) แบบออบเจกต์ โอเรียลเต็ด, วินโดว์ต่างๆก็เป็นออบเจกต์ในระบบ, คลาสไลบรารี่ (library) ซึ่งเป็นที่เก็บของคลาสต่าง ๆ ของระบบรวม, เอดิเตอร์ที่ใช้ในการเขียนโปรแกรมก็เป็น ออบเจกต์ หนึ่งของ คลาส editor ในระบบ หรือแม้กระทั่งคอมไพเลอร์ที่ใช้คอมไพล์โปรแกรมภาษาสมอลทอกค์ ก็เป็นออบเจกต์หนึ่งในคลาสคอมไพเลอร์ของภาษาสมอลทอกค์ นั่นเอง

คลาสในภาษาสมอลทอกค์ เทียบได้กับออบเจกต์ในโปรแกรม เราสามารถส่งแมสเซสไปยัง คลาสได้เช่นเดียวกับที่ส่งไปยังออบเจกต์ ทั้งนี้เนื่องจากในความเป็นจริงแล้วคลาสในภาษาสมอลทอกค์ ก็ เป็นออบเจกต์ของคลาสที่ชื่อว่า mataclass อีกต่อหนึ่ง

ภาษาสมอลทอกค์เป็นภาษาที่ถูกกำหนดให้ผูกติดกับ GUI มาตั้งแต่เกิด เพราะความเชื่อที่ว่าภาษาแบบออบเจกต์ โอเรียลเต็ด ในอุดมคตินั้น ต้องมีการติดต่อกับผู้ใช้ ในแบบออบเจกต์ โอเรียลเต็ด ด้วย ภาษาสมอลทอกค์จึงเป็นภาษาแรกๆ ที่ใช้ GUI เป็นหลัก และด้วยเหตุนี้เองทำให้ภาษาสมอลทอกค์ สามารถแสดงลักษณะของคลาส library ได้อย่างชัดเจน โปรแกรมเมอร์สามารถดัดแปลง แก้ไข หรือเพิ่ม / ลด คลาสไลบรารี่ได้ โดยเพียงแค่สวิทช์การทำงานมายังวินโดว์ของคลาสไลบรารี่เท่านั้น เมื่อโปรแกรมเมอร์ต้องการเท็กซ์ เอดิเตอร์ (text editor) ก็สามารถดึงมาจากคลาสเอดิเตอร์ในไลบรารี่ ซึ่งจะได้เอดิเตอร์ที่มีความสามรถ เหมือนกับที่ใช้งานอยู่

ภาษาสมอลทอกค์ เป็นภาษาที่ง่ายสำหรับการเรียนรู้และการใช้เพราะมี syntax และมี semantic ที่เข้าใจง่าย มีหลักการเพียงเล็กน้อย โดยหลักการพื้นฐานของภาษาสมอลทอกค์ ประกอบด้วย

1. ออบเจกต์ (Object)
2. คลาส (Class)
3. แมสเซส (Message)
4. เมทอด (เมทอด)

3.1 ออบเจกต์ (Object)

ออบเจกต์ :เป็นบล็อกพื้นฐานของภาษาสมอลทอกค์ที่แสดงโครงสร้างข้อมูล (data structure) ทุก ออบเจกต์เป็นอินสแตนส์ของคลาสออบเจกต์ จะมีการป้องกันโครงสร้างข้อมูลโดยข้อมูลจะถูกเก็บอยู่ใน ออบเจกต์ ซึ่งจะถูกเข้าถึงได้ โดยผ่านทางแมสเซสเท่านั้น

เช่น 120 เป็นอินสแตนส์ของคลาส integer

$A เป็นอินแตนส์ของคลาส character

ตัวแปร (Variable): เป็นที่เก็บดาต้าของออบเจกต์ โดยจะเก็บพอยเตอร์(pointer) ของออบเจกต์ เดียว ตัวแปรหนึ่ง ๆ อาจเก็บพอยเตอร์ที่ต่างกัน ณ.เวลาต่างๆกัน

ตัวแปรแบ่งเป็น 2 ชนิด คือ

- private variable จะถูกเข้าถึงได้โดย ออบเจกต์ ตัวมันเองเพียง ออบเจกต์เดียว และ private variable จะขึ้นต้นด้วยอักษรตัวเล็ก

- share variable จะถูกเข้าถึงได้โดยหลายๆ ออบเจกต์ และ share variable จะขึ้นต้นด้วยอักษรตัวใหญ่

3.2 คลาส (Classes)

คลาสเป็นตัวอธิบายโครงสร้างข้อมูลของออบเจกต์ ,algorithms(เมทอด) และ external interface(แมสเซส)

ทุก ๆ ออบเจกต์เป็นอินสแตนส์ของคลาส ออบเจกต์ที่เป็นอินสแตนส์ของคลาสเดียวกันจะคล้ายกันเพราะมันมีโครงสร้างเหมือนกัน(เช่น มี instance variable เหมือนกัน) ,มีแมสเซสซึ่งมันจะตอบสนองเหมือนกันและมีเมทอดที่ใช้เหมือนกัน

เราถือว่า คลาสเป็นออบเจกต์ ซึ่งอยู่ใน global variables ดังนั้นชื่อของคลาสขึ้นต้นด้วยอักษรตัวใหญ่เสมอ

Class hierarchy

ประกอบด้วยรูทคลาส ที่ชื่อออบเจกต์พร้อมกับซับคลาสอีกมากมาย แต่ละคลาสจะอินเฮอริท หน้าที่ทั้งหมดของซุปเปอร์คลาส ของมันในโครงสร้างลำดับชั้น สำหรับทุกๆ ออบเจกต์ แต่ละซับคลาส จะสร้างอยู่บนซุปเปอร์คลาสของมัน โดยการเพิ่มเมทอดและ instance variable ของคลาสของมันเอง โดยสมอลทอคก์ได้จัดเตรียมคลาสต่าง ๆ ไว้อย่างมากมาย

Inheritance

คลาสหนึ่ง ๆ จะอินเฮอริแตนส์ทุกอย่างในซุปเปอร์คลาสของมัน เช่น instance variable, class variable และ method inheritance ของ คลาส variable ทำให้เมทอดของคลาสอ้างถึงคลาส variable ที่ กำหนดไว้ใน ซุปเปอร์คลาส ของมันได้

Class massage

แมสเซสที่ใช้สำหรับคลาสใหม่โดยทั่วไปนิยมใช้ new ,new: แต่บางคลาสอาจกำหนดแมสเซส ใช้สำหรับสร้างอินสแตนส์เองก็ได้

เช่นเดียวกันกับ ออบเจกต์ คลาสก็รู้ว่ามีแมสเซสใดบ้างที่สามารถตอบสนองได้ โดย metaclass จะเป็นตัวกำหนดว่า แมสเซสใดบ้างที่มันตอบสนองได้

3.3 แมสเซส (Message)

ทุก ๆ โปรเซสในภาษาสมอลทอกค์ จะเกี่ยวข้องกับการส่งแมสเซสไปให้กับออบเจกต์ แมสเซสเป็นภาษาที่ใช้เพื่อแสดงความต้องการที่จะทำอะไรกับออบเจกต์ แมสเซสจะ request service จากออบเจกต์ โดยอยู่ในรูปของการติดต่อจากภายนอก

แมสเซส : สิ่งที่ส่งไปยังออบเจกต์ เพื่อเรียกใช้เมทอดในออบเจกต์ ให้ทำงานได้ตามต้องการ แมสเซสคล้ายกับการเรียกใช้ฟังก์ชั่นใน procedural language

แมสเซส ประกอบด้วย

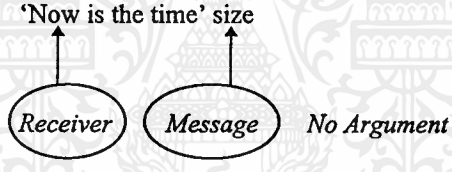
1.*Receiver object* : การกำหนดออบเจกต์ที่จะส่งแมสเซส

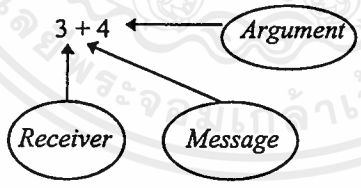
2.*Message selector* : การระบุโอเปอเรชั่นที่ต้องการ

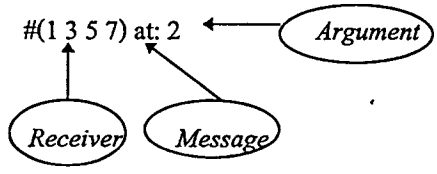
3.*Argument* (จะมีหรือไม่มีก็ได้) : การกำหนดออบเจกต์ เพิ่มเติมที่จะรวมอยู่ใน แมสเซส

4. การยอมรับ *single object* ที่จะถูก return ในฐานะ แมสเซส answer

เช่น







'20' factorial

จะเกิด error เพราะ '20' ไม่รับ แมสเซส factorial

* Object always know the แมสเซส that are appropriate for them *

<u>แมสเซส</u> แบ่งเป็น

3.3.1 Unary Message คือ แมสเซส ที่ไม่มี argument

เช่น	<u>Message</u>	<u>Result</u>
	'Hello' reversed	'olleH'
	#('a','of','string') size	3
	$A asciiValue	65
	'Now is the time' asUpcase	'NOW IS THE TIME'

3.3.2 Keyword Message คือ แมสเซส ที่ มี 1 หรือ มากกว่า 1 argument

เช่น	<u>Message</u>	<u>Result</u>
	'Hello' includes: $e	True
	'Now is the string' at: 6	$s
	#(1 0 4 5) at: 2 put: #(2 3)	(1 (2 3) 4 5)

3.3.3 Arithmetic Message คือ แมสเซส ที่เกี่ยวข้องกับการคำนวณ

เช่น	<u>Message</u>	<u>Result</u>
	5 * 7	35
	5 // 2	2 (DIV)
	5 \\ 2	1 (MOD)
	2 / 6	1/3 (Rational division)
	3 + 4 * 2	14
	3 + (4 * 2)	11

ภาษาสมอลทอกค์ จะทำ expression จาก ซ้ายไปขวา ถ้าไม่มีวงเล็บ (parentheses) แต่ถ้ามีให้ทำในวงเล็บก่อน

3.3.4 Binary Message คือ แมสเซส ที่มี 1 argument และมีตัวอักษรพิเศษ (special character)

เช่น	<u>Message</u>	<u>Result</u>
	'Hello','there'	'Hellothere'
	#(1 2 3),#(4 5 6)	(1 2 3 4 5 6)

ในที่นี้ special character คือ ,(Comma) ใช้ concatenate ระหว่าง argument กับ receiver

3.3.5 Message inside of message คือ การใช้แมสเซสที่ return ค่าเป็นออบเจกต์แทน ออบเจกต์

เช่น	<u>Message</u>	<u>Result</u>

'Hello' at: (#(5 3 1) at: 2) 1

'Hello' size + 4 9

Expression Series

Turtle black.

Turtle home.

Turtle go: 100.

Turtle turn:120.

Turtle go: 100.

Turtle turn:120.

Turtle go: 100.

Turtle turn:120.

-แต่ละแมสเซสจะแยกกันด้วย . (Period)

Cascade Message

-ถ้าแมสเซสส่งถึงออบเจกต์เดียวกันสามารถเขียนย่อได้ เป็น Cascadeแมสเซสคั่นด้วย ; (semi-colon)

Turtle

black;

home;

go:100;

turn:120;

go:100;

turn:120;

go:100;

turn:120;

Simple Loop

ใช้ block of code มาช่วย โดย series of message ที่จะทำวน loopจะอยู่ใน square brackets [ ] และใช้ แมสเซส **timeRepeat:**

เช่น Turtle

black;

home.

3 timeRepeat: [

Turtle

go:100;

turn:120]

Object and Message Are Safe

ออบเจกต์ มี state ซึ่ง ออบเจกต์ สามารถจดจำสิ่งต่าง ๆ ได้ แมสเซสเปลี่ยน state ของ ออบเจกต์ ได้

เช่น จากตัวอย่าง ออบเจกต์ Turtle สามารถจำตำแหน่ง heading และ สี ของมันได้

message black เปลี่ยนสี

message go: และ home: เปลี่ยนตำแหน่ง

message turn: เปลี่ยน heading

'Hello' at :1 put:23

เกิดข้อผิดพลาด (error) เพราะ 23 ไม่ใช่ character เราสามารถเปลี่ยน state ของ string ได้โดยการส่งแมสเซส และ string ก็สามารถเช็คได้ว่า argument ถูกต้องหรือเปล่า

*สิ่งนี้ทำให้ภาษาสมอลทอกค์ เป็น very safe language

Assignment Expressions

ในภาษาสมอลทอกค์กำหนดค่าให้ตัวแปรโดยใช้เครื่องหมาย := (assignment)

เช่น temp := 10.

Index := index + 1.

Return Expression

ใช้ ^ (caret) เป็นตัวกำหนดค่าที่จะ return จาก expression series

เช่น ^temp จะ return ค่า ในตัวแปร temp ออกมา

ซึ่งจะเรียก statement นี้ว่า *return expression*

Comment

ในภาษาสมอลทอกค์ เขียน comment โดยเขียนอยู่ใน Double quote marks " "

3.4 เมทอด (Method)

เมทอด เป็นอัลกอริทึ่ม (algorithms) ที่ถูกกระทำโดยออบเจกต์เพื่อตอบสนองต่อ แมสเซส ที่มันได้รับ เมทอดจะแสดงรายละเอียดภายในของการสร้างออบเจกต์

การกำหนดโปรโตคอลของ คลาส หนึ่งๆจะมี 2 ส่วน คือ class method และ instance method ซึ่งมีรายละเอียดดังนี้

- *class method*

จะสร้างแมสเซสที่ส่งให้แก่ คลาส รีซีพเวอร์ของ คลาส แมสเซส คือ คลาส ออบเจกต์ ไม่ใช่อินสแตนส์ของคลาสทุกคลาส เป็น global variables และสามารถถูกอ้างถึงโดยชื่อของคลาส

-*instance method*

จะสร้างแมสเซส ที่ส่งให้แก่อินสแตนส์ของ คลาส รีซีพเวอร์ของอินสแตนส์แมสเซส คือ ออบเจกต์

เมทอด เป็นลำดับ (sequence) ของ expression ซึ่งมี 4 ชนิดคือ

1.*Literals* เช่น #aSymbol #(1 2 4 16) 'magic'

2.*Variable name* เช่น สมอลทอกค์ x replacementCollection

3. *mdssage expression* เช่น 100 factorial

bad add : stream next

array at: index+ 10 put: Bag new

4.*Block of code* เช่น [: x : y| xname< yname]

การเริ่มต้นเมทอดเริ่มด้วยการกำหนดชื่อของเมทอด , argument ต่างๆ และ temporary variables ที่เมทอดใช้

3.5 Control Structure In Smalltalk

Variable แบ่งเป็น 2 ชนิด คือ

1.Temporary Variable

ตัวแปรแบบชั่วคราวจะกำหนดอยู่ภายในเครื่องหมาย "|" (vertical bar) ที่บรรทัดแรกของ expression series ชื่อของตัวแปรชั่วคราวต้องขึ้นต้นด้วยตัวอักษรตัวเล็ก ส่วนตัวถัดไปจะเป็นตัวเล็กหรือใหญ่ก็ได้

2.Global variable

ชื่อของตัวแปรแบบ Global ต้องขึ้นต้นด้วยตัวพิมพ์ใหญ่ ส่วนตัวถัดไปจะเป็นตัวเล็กหรือใหญ่ก็ได้ ประกาศใน expression ได้เลย

Comparing Object

ในภาษาสมอลทอกค์ จะเปรียบเทียบออบเจกต์โดยการส่งแมสเซส ซึ่ง comparisons ก็มี < , > , <= , >= , = , ~= ซึ่งจะถูกแทนด้วย binary แมสเซส

เช่น

Message	Result
3 < 4	true
#(1 2 3 4) = #(1 2 3 4)	true
'Hello' >= 'Goodbye'	false

* Comparison จะ return ค่า true กับ false *

Testing Object

บาง ออบเจกต์ จะเข้าใจ แมสเซส ก็ต่อเมื่อมีการทดสอบบางสิ่งเกี่ยวกับสถานะหรือเงื่อนไข

เช่น

Message	Result
$a isUpcase	false
('Hello' at: 1) isVowel	false
7 odd	true

แมสเซส นี้จะ return ค่า true , false เท่านั้น *

Condition Execution

จะ execute series of statement เมื่อ condition เป็นจริง โดยจะใช้ comparison แมสเซส และ แมสเซส **ifTrue:** และ **ifFalse**

เช่น

```
| max  a  b |
a:= 5 squared.
b:= 4 factorial.
a<b
        ifTrue: [max := b]
        ifFalse: [max := a]
^max
```

Boolean Expression

เป็น expression ที่ใช้ test condition ซึ่งสามารถกระทำ compound test ได้ โดยใช้ แมสเซส **or:** และ **and:**

เช่น

```
( c < 0 or: [c>9] )
( c >= $A and: [c <= $F] )
```

Looping Message

นอกจาก แมสเซส **timeRepeat:** แล้ว ยังใช้ แมสเซส **whileTrue:** และ **whileFalse:**

เช่น

```
"copy disk file"
| input  output |
input := File pathName: 'go'.
output := File pathName: 'junk'.
[input atEnd]
        whileFalse: [ output  nextPut: input next].
input close.
Output close.
```

* atEnd เป็น แมสเซส ที่จะ return ค่า true เมื่อยังมี character ที่อ่านจาก input file stream อยู่

* nextPut: เป็น แมสเซส ที่จะเขียน character ตัวถัดไปลงใน output file stream

Simple Iteration

Iteration แมสเซส คือ **to: do:** ซึ่งต้องมี 2 argument

เช่น

```
| temp |
temp:=10.
1 to: 10 do: [ :a | temp := temp + a].
^temp
```

* แมสเซส **to: do:** จะเพิ่มค่าทีละ 1 แต่เรากำหนดให้เพิ่มค่าได้เท่าไรก็ได้โดยใช้ แมสเซส **to: by: do:** ซึ่งต้องมี 3 argument

เช่น
```
| temp |
temp:=10.
1 to: 10 by: 2 do: [ :a| temp := temp + a].
^temp
```
(35)

Block Argument

เป็น temporary variable ชนิดหนึ่งแต่ไม่ต้องกำหนดไว้ที่บรรทัดแรกของชุดคำสั่ง Block Argument เป็นตัวแปรที่ใช้ใน block ซึ่งหน้าตัวแปรจะมีเครื่องหมาย " : "(colon) และตัวแปรเขียนแยกกับคำสั่งภายใน block ด้วยเครื่องหมาย "|" (vertical bar)

Block Argument ทำให้ภาษาสมอลทอกค์มีคำสั่งการทำซ้ำหลายแมสเซส เช่น **do: , select: , reject: , collect:**

Generalized Iterators

The do: Iterator

เป็นการส่งตัวอักษรทีละตัวเข้าไปใน block

เช่น
```
| vowel |
vowel:=0.
'Now is the time' do: [:char | char isVowel
                              ifTrue: [vowel := vowel + 1]
                    ]
^vowel
```

The select: Iterator

Receiver 0bject จะส่งค่ากลับไปเป็นสมาชิกทั้งหมดในบล็อคและจะ return ค่าเป็นจำนวนสมาชิกทั้งหมดที่ถูกเลือก

เช่น

Message	Result
'Now is the time' select: [:c | c isVowel]	'oieie'
('Now is the time' select: [:c | c isVowel]) size	5

The reject: Iterator

การทำงานของ reject: คล้ายกับ select: แต่จะ return ค่ากลับเฉพาะค่าที่ให้ผลลัพธ์จากบล็อค เป็นเท็จ ในขณะที่ select: จะ return ค่ากลับเมื่อให้ผลลัพธ์จาก block เป็นจริง

เช่น

Message	Result
#(1 2 3 4 5 6 7) reject: [: i | i ood]	(2 4 6)

The collect: Iterator

การทำงานของ collect: มีผลทำให้บล็อคของโค้ด ทำงานสำหรับแต่ละสมาชิกของ รีซีพเวอร์ทำให้ return ผลลัพธ์แต่ละตัวรวมกันเป็นบล็อค

เช่น	Message	Result
	#(1 4 7 10) collect: [:i | i * i]	(1 16 49 100)

เปรียบเทียบ 3 แมสเซส

	Message	Result
#(1 2 3 4 5 6 7)	select: [:c | c odd]	(1 3 5 7)
#(1 2 3 4 5 6 7)	reject: [:c | c odd]	(2 4 6)
#(1 2 3 4 5 6 7)	collect: [:c | c odd]	(true false true false true false true)

บทที่4

สถาปัตยกรรมเล็กซ์เชล (LEX Shell Architecture) และการออกแบบโปรแกรม

4.1สถาปัตยกรรมของระบบ LEX Shell

ประกอบด้วย 4 ส่วน หลักคือ

1.Acquisition Module เป็นส่วนที่ทำหน้าที่ ดึงเอาความรู้จากผู้เชี่ยวชาญ โดยอาศัยขบวนการเรียนรู้จากตัวอย่าง โดยรับเอากลุ่มตัวอย่าง สำหรับการวินิจฉัยปัญหา มาจากผู้เชี่ยวชาญ แล้วทำการอนุมานตามกฎ ซึ่งเป็นการจำแนกประเภทของความรู้ที่ใช้ในการวินิจฉัย โดยแทนกฎให้อยู่ในรูปของ ต้นไม้สำหรับการตัดสินใจ

2. Knowledge Base เป็นส่วนที่ใช้จัดเก็บความรู้ หรือกฎที่ได้จากการอนุมาน จาก Acquisition ซึ่งอยู่ในรูปของต้นไม้สำหรับการตัดสินใจ

3. Inference Engine เป็นส่วนที่ใช้ความรู้ซึ่งอยู่ในรูปของต้นไม้สำหรับการตัดสินใจใน knowledge base โดยใช้การ Data-driven decision tree search และสามารถจัดการกับความรู้ที่แน่นอนได้เท่านั้น

4. Consulting Session เป็นส่วนที่จะให้คำอธิบายการวินิจฉัยของส่วนซึ่งติดต่อกับผู้ใช้ โดยผู้ใช้สามารถที่จะถามหาเหตุผลว่าทำไมระบบจึงถามคำถามเช่นนี้ หรือทำไมถึงได้ขอสรุปเช่นนี้ และบางครั้งเมื่อได้ข้อสรุปแล้วระบบอาจจะต้องการการแสดง คำอธิบายเพิ่มเติมเกี่ยวกับข้อสรุป ซึ่งนั่นก็เป็นหน้าที่ของส่วนนี้เช่นเดียวกัน เมื่อพิจารณาให้ดีแล้วจะเห็นว่า ส่วน consulting session เป็นส่วนที่ สำคัญมากส่วนหนึ่ง เนื่องจากถึงแม้ระบบจะมีความสามารถในการวินิจฉัยเพียงใดก็ตาม ถ้าไม่สามารถให้เหตุผลของการวินิจฉัยได้ ก็ถือว่าเป็นระบบที่ยังใช้งานไม่ได้ เนื่องจากเป็นการใช้ความรู้โดยขาดซึ่งเหตุผลรองรับย่อมไม่สามารถยอมรับได้

4.2 การออกแบบโปรแกรม

เราได้ออกแบบ program ให้อยู่ในรูปแบบของ Object Oriented โดยกำหนดให้แต่ละส่วนเป็น Object

1. Acquisition Module: ออกแบบเป็น Object ในรูปแบบของ table โดยใช้ array 2 มิติในภาษา Smalltalk กำหนดเป็น table โดยจะทำการสร้าง method ในการรับตัวอย่าง Example เข้ามาจัดเก็บไว้ใน table และสร้าง method ในการคำนวณหาค่า entropy ที่ใช้ใน ID3 ในการ generate decision tree ที่ใช้แทนกฎ ดังรูป Data Structure ของส่วน Acquisition Module

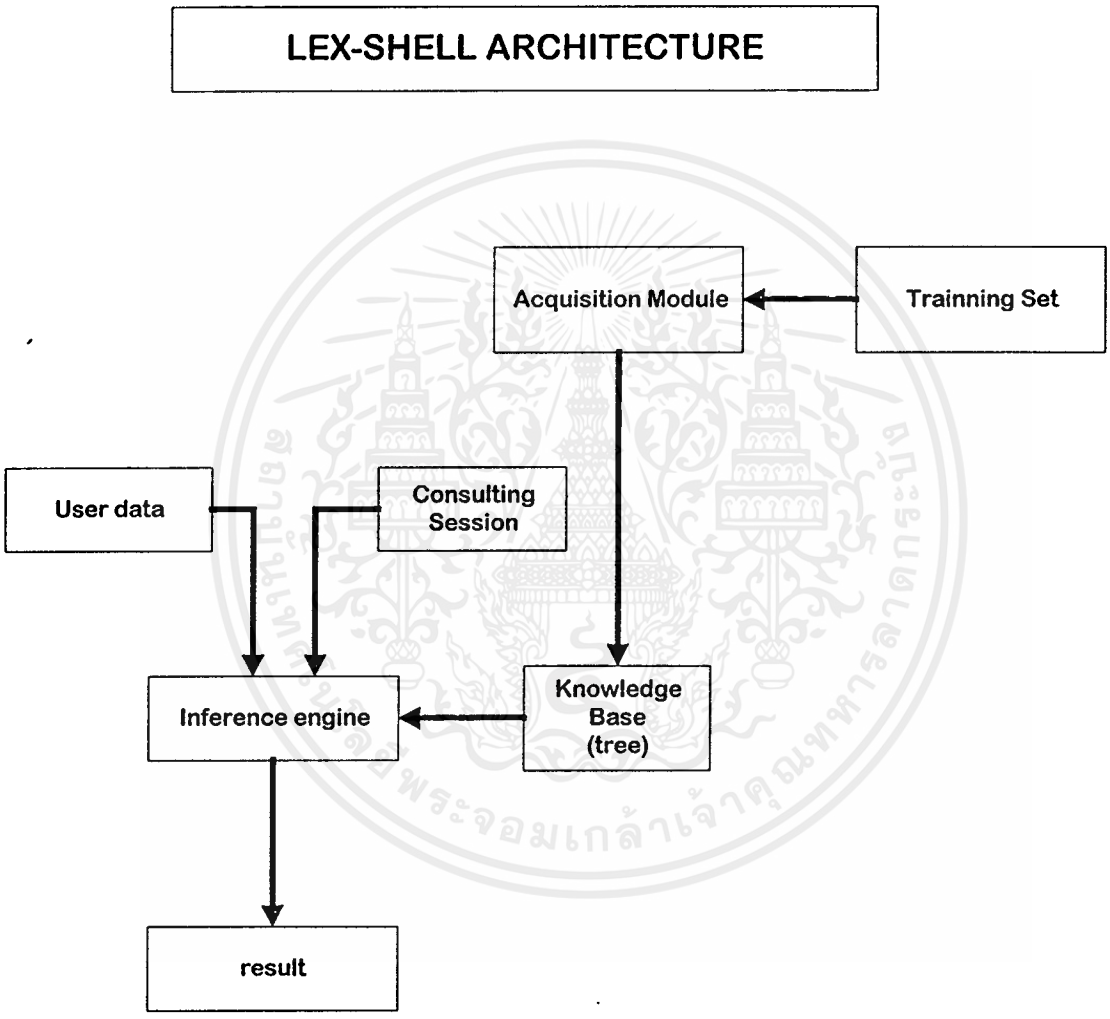
2. Knowledge Base: ออกแบบเป็น Object ในรูปแบบของ node ของต้นไม้ โดยทำการสร้าง method เพื่อใช้ในการเก็บค่าเข้า node และการ link node ต่าง ๆ เข้าด้วยกันเป็น tree ดังรูป Data Structure ของส่วน Knowledge Base

3. Inference Engine: ทำการสร้าง method ที่ใช้ในการ search ไปตาม link ของ tree เพื่อใช้หาคำตอบให้ user ในการเข้ามาใช้ระบบผู้เชี่ยวชาญได้

4. Consulting Session: สำหรับในส่วนนี้ทางผู้จัดทำยังมิได้มีการพัฒนาจัดทำขึ้น

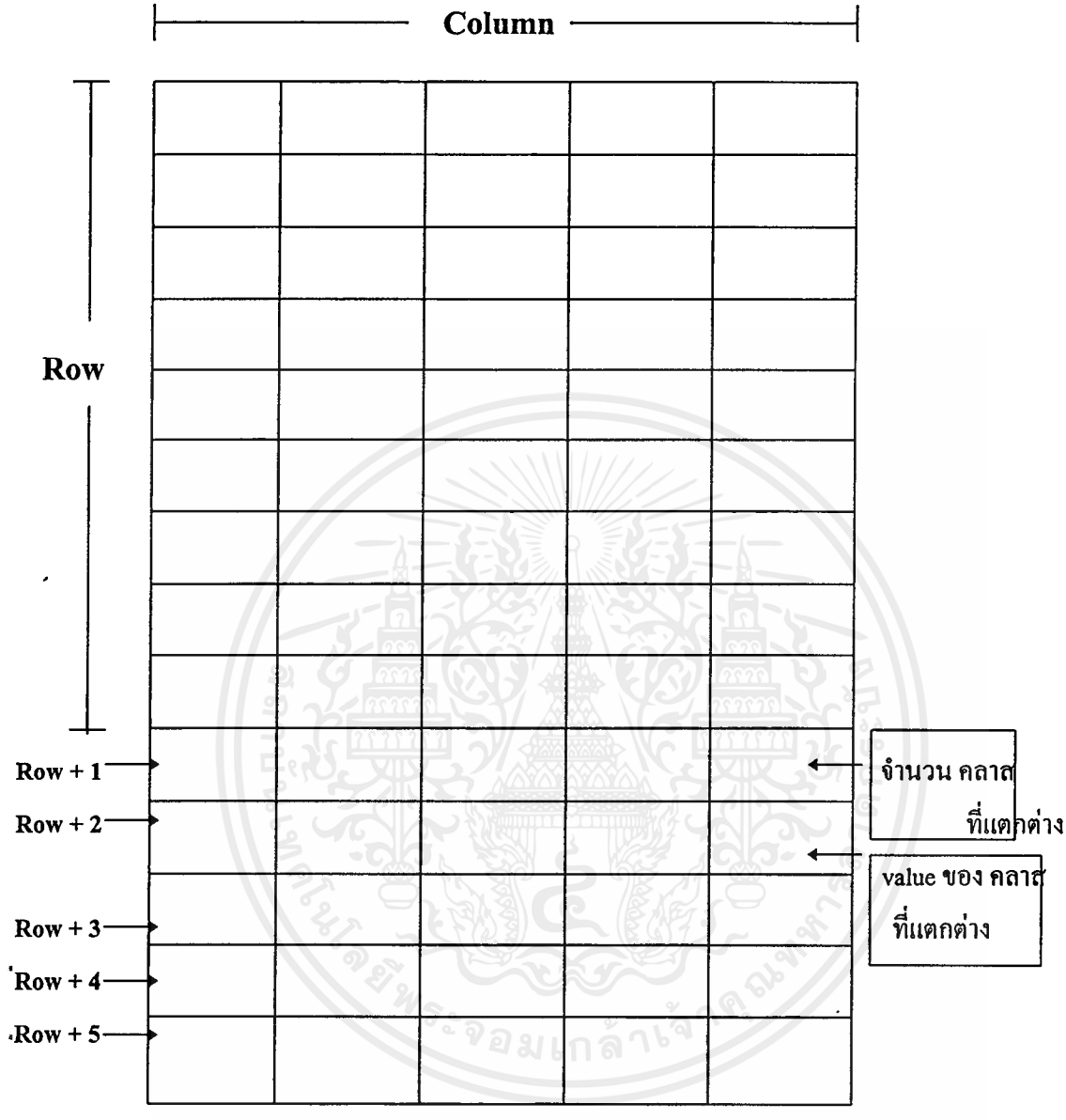
และนอกจากจะทำการสรุปกฎและอนุมานให้อยู่ในรูปของ decision tree แล้ว เรายังได้ทำการพัฒนาให้ระบบสามารถที่จะ

1. generate pseudo code: เพื่อให้ผู้อื่นสามารถนำไปใช้ในการเขียนโปรแกรมภาษาอื่น หรือช่วยให้ผู้ใช้เข้าใจกฎได้ง่ายขึ้นอีกด้วย
2. generate code ภาษา C: เนื่องจากภาษา C เป็นภาษาที่ใช้กันอย่างกว้างขวาง ดังนั้นเราจึงทำการพัฒนาระบบให้สามารถ generate code ภาษา C เพื่อให้ผู้ใช้นำไปใช้งานในรูปของภาษา C ได้อีกด้วย

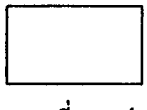


รูปที่ 4-1 แสดง Flowchart ของสถาปัตยกรรม LEX Shell

Data Structure ของส่วน Acquisition Module

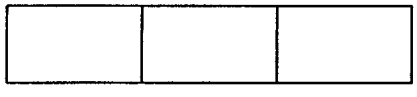


Row + 1



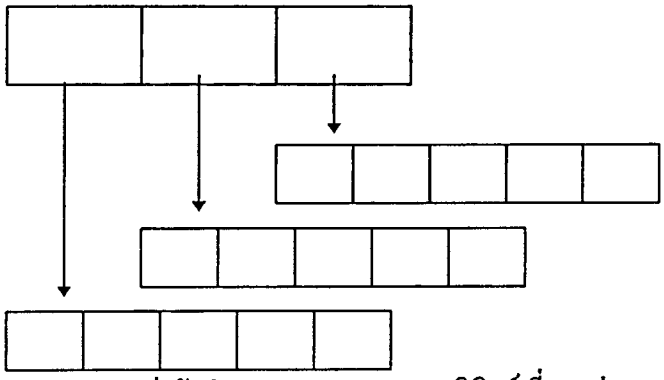
- เก็บค่าจำนวน value ที่แตกต่างกันของแต่ละ แอทริบิวท์

Row + 2



- var array ในช่อง (row + 2) เท่ากับจำนวน value ของ แอทริบิวท์ ที่แตกต่าง (row + 1) เพื่อใช้เก็บค่า value ของ แอทริบิวท์ ที่แตกต่าง

Row + 3



-var array ในช่อง (row + 3) เท่ากับจำนวน value ของ แอทริบิวท์ ที่แตกต่าง (row + 1) และในแต่ละช่อง var array เท่ากับจำนวน คลาส ที่แตกต่างกัน + 2 เพื่อใช้เก็บ

- ค่าจำนวน value ของแต่ละ value เก็บไว้ในช่องรองสุดท้าย
- ค่า เอนโทรปี ย่อย ของแต่ละ value เก็บไว้ในช่องสุดท้าย
- ค่า $p(c_i|a_{kj})$ เก็บแต่ละช่องช่วงแรก

Row + 4

- เก็บค่า เอนโทรปี ของแต่ละ แอทริบิวท์

Row + 5

- เก็บชื่อ แอทริบิวท์ ของแต่ละ Column

<u>ตัวอย่าง Data Structure ของ Investment data set</u>

<u>หมายเหตุ</u>

BC : Blue chip stock

GS : Gold stock

MR : Mortgage related

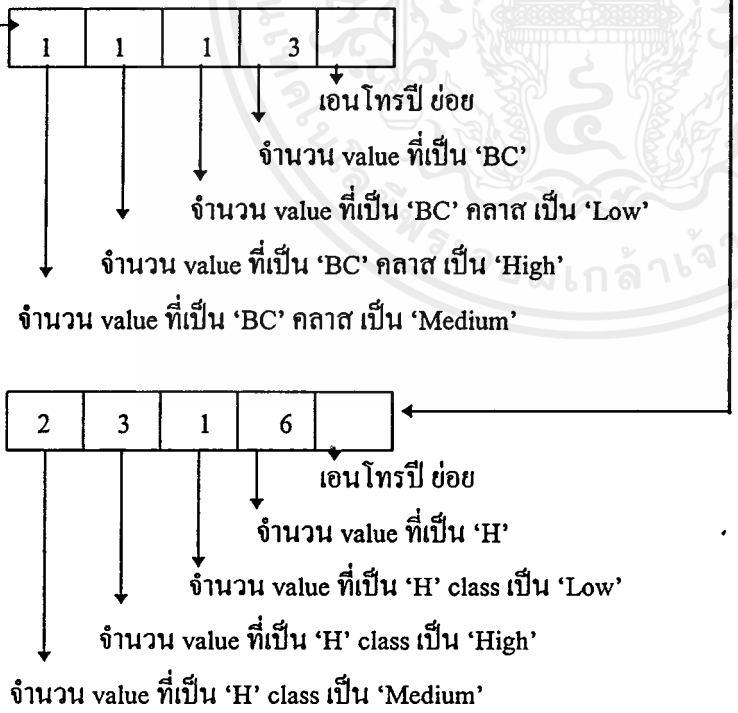
H : High

M : Medium

L : low

Blue chip stock	High	High	Medium	Medium
Blue chip stock	Low	High	Medium	High
Blue chip stock	Medium	Low	High	Low
Gold stock	High	High	Medium	High
Gold stock	Low	High	Medium	Medium
Gold stock	Medium	Low	High	Medium
Mortgage related	High	High	Medium	low
Mortgage related	Low	High	Medium	High
Mortgage related	Medium	Low	High	Low

3			3			2		2		3			Row+1
BC	**GS**	**MR**	**H**	**L**	**M**	**H**	**L**	**M**	**H**	**M**	**H**	**L**	Row+2
													Row+3
1.140333			1.140333			1.2787		1.2787					Row+4
Mutual fund - type			Interested rates			Cash available		Tension		Fund value (class)			Row+5

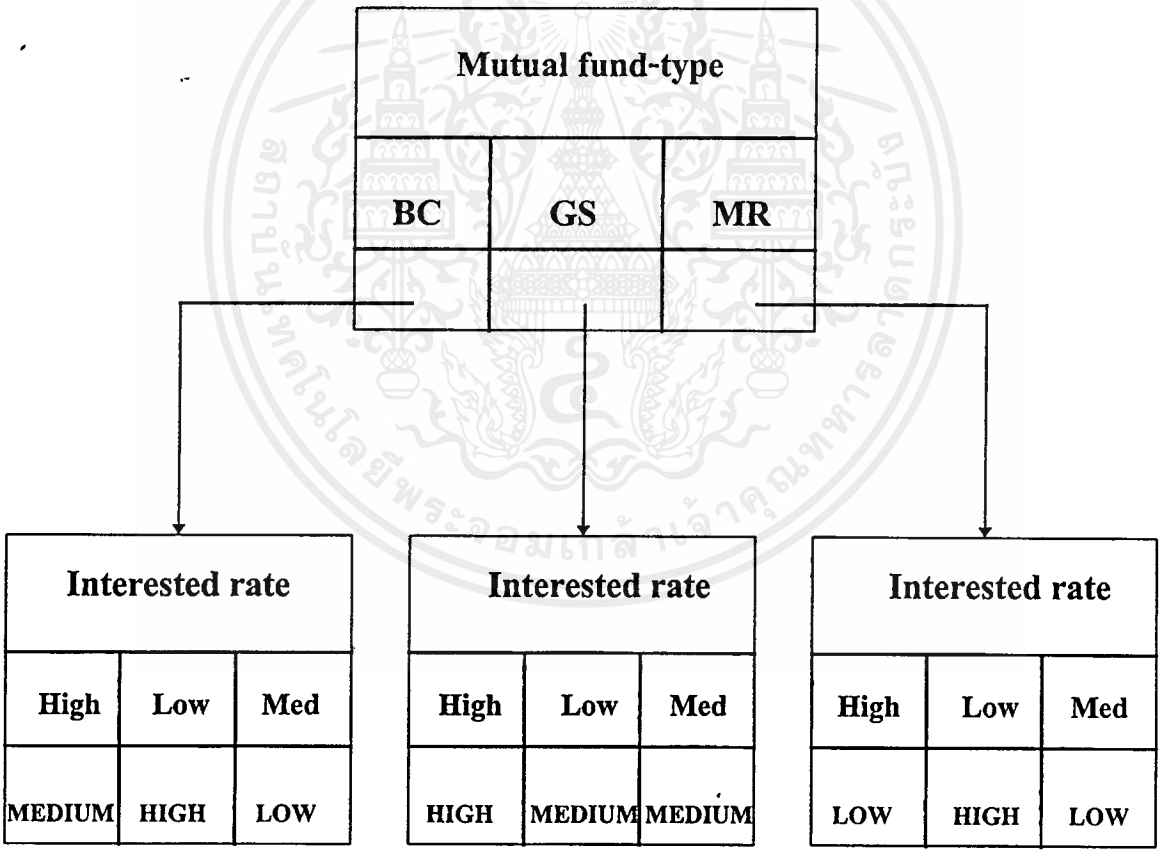


Data Structure ของส่วน Knowledge Base

Node

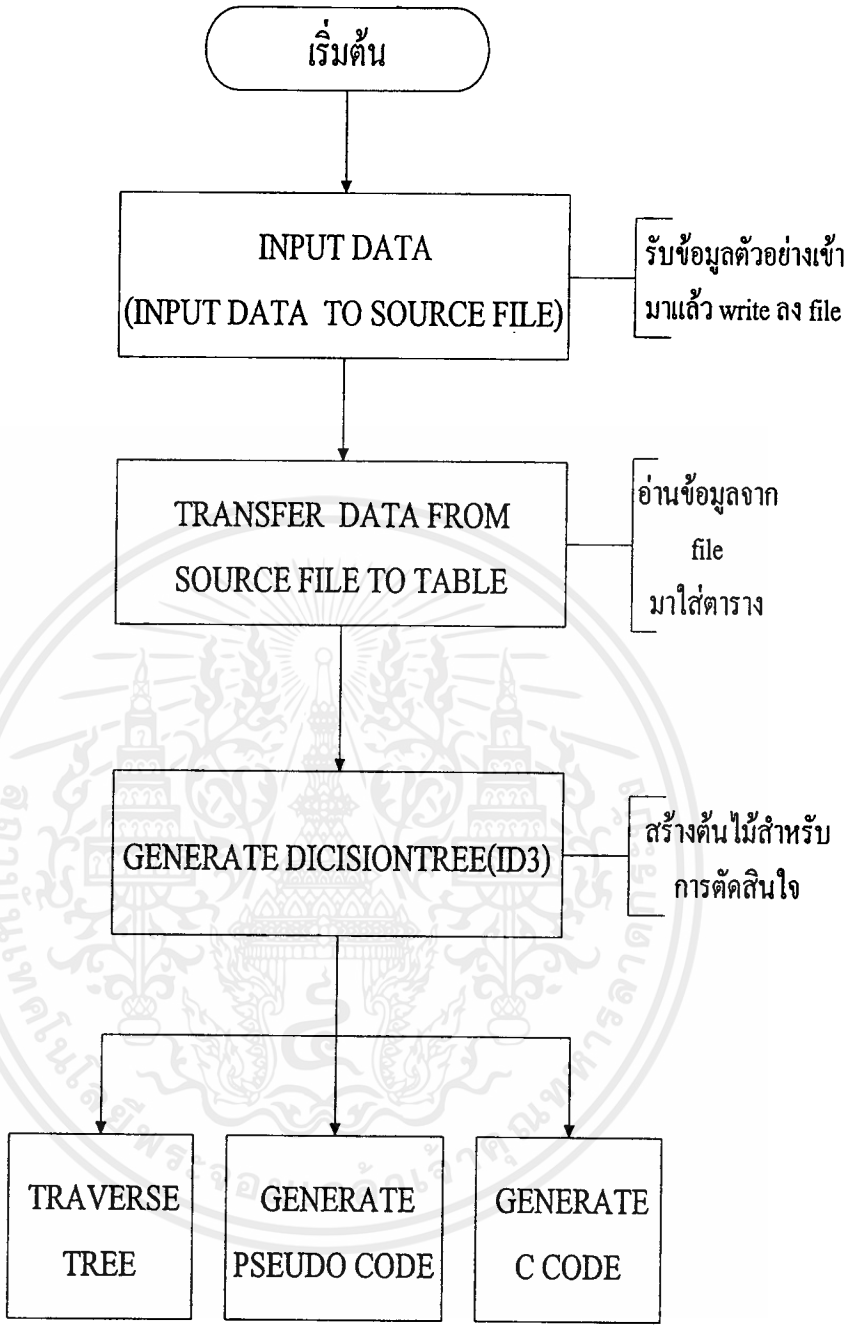
NAME (Attribute)		
VALUE	VALUE	VALUE
POINTER	POINTER	POINTER

ตัวอย่าง Data Structure ของ Investment data set

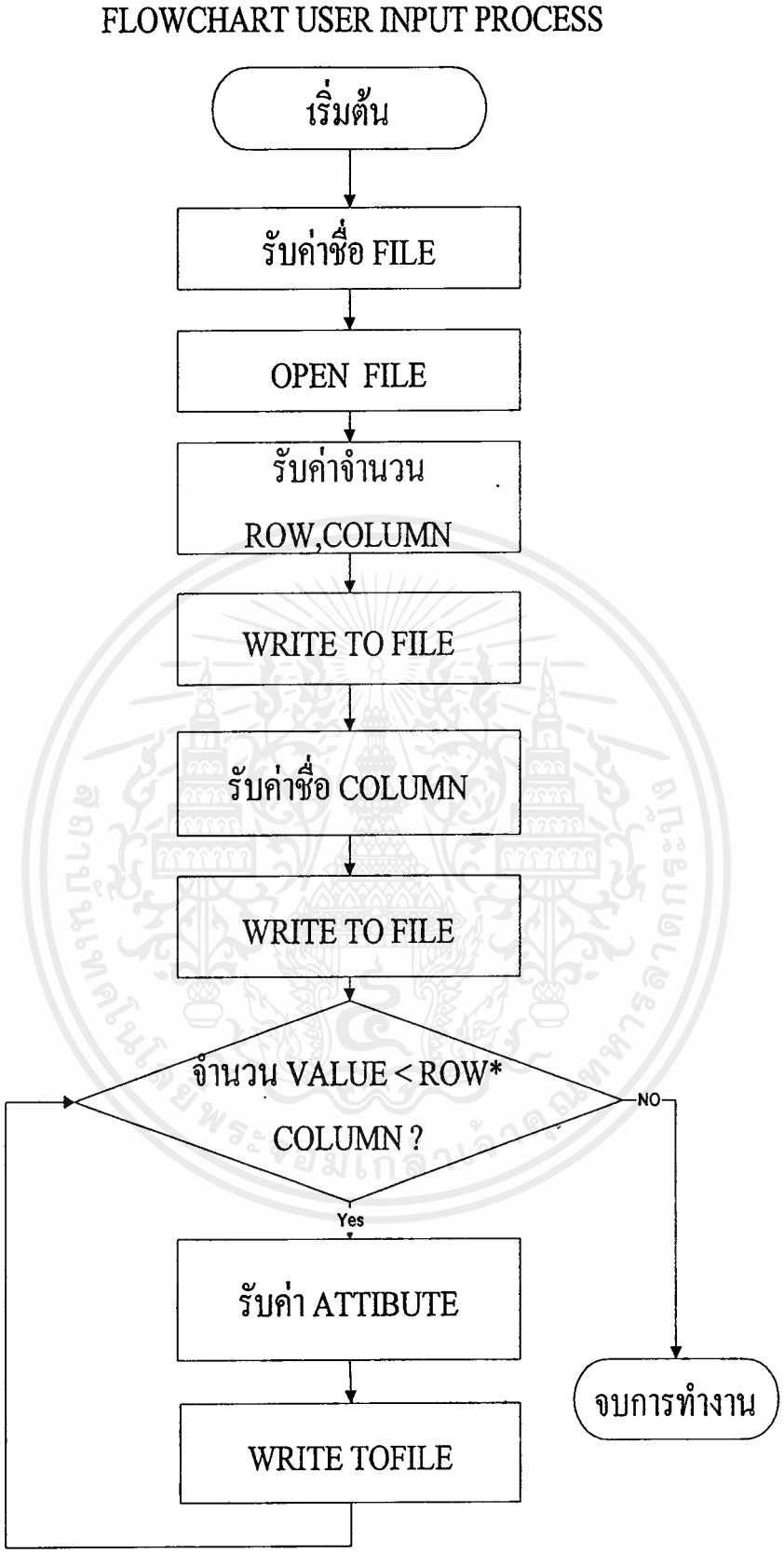


**** ช่อง pointer ของ leaf node จะเก็บค่าของ Class ****

FLOWCHART การทำงานทั้งหมดของระบบ

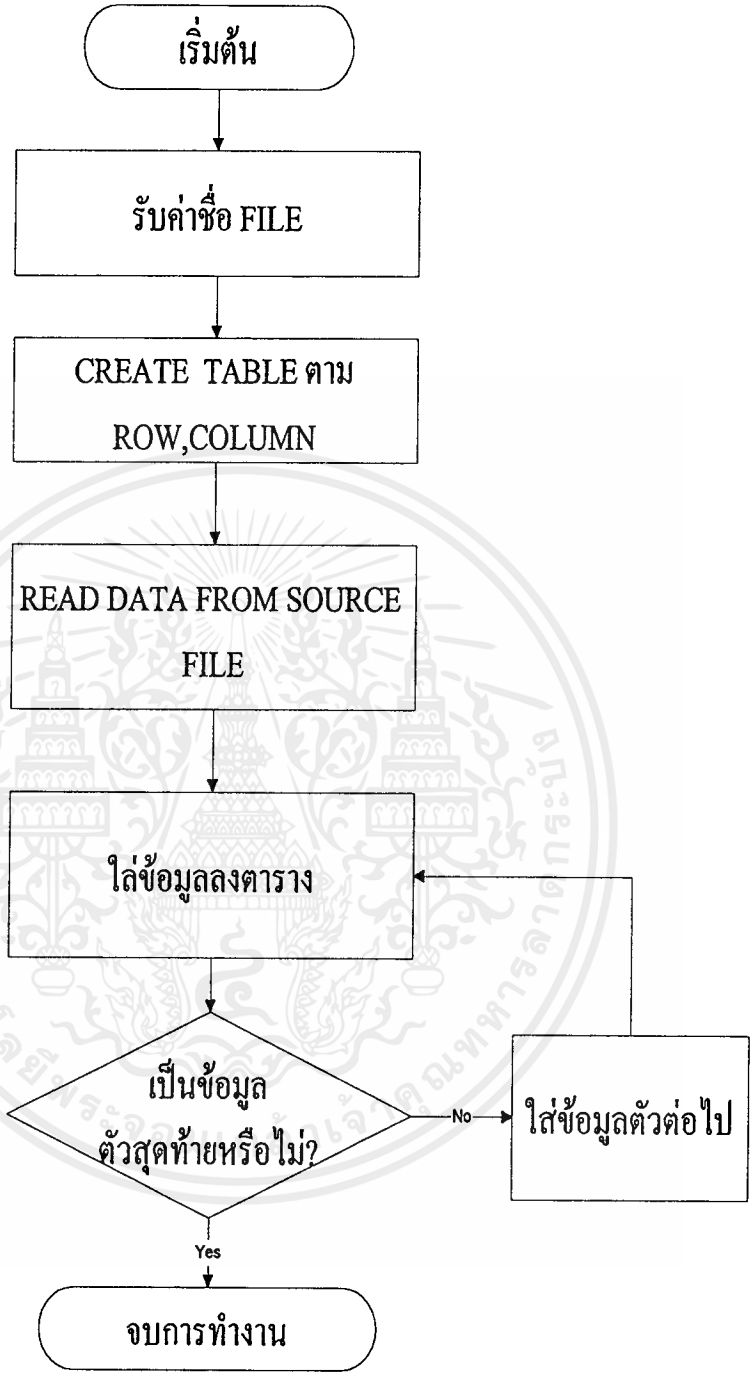


รูปที่ 4-2 แสดง Flowchart การทำงานทั้งหมดของระบบ



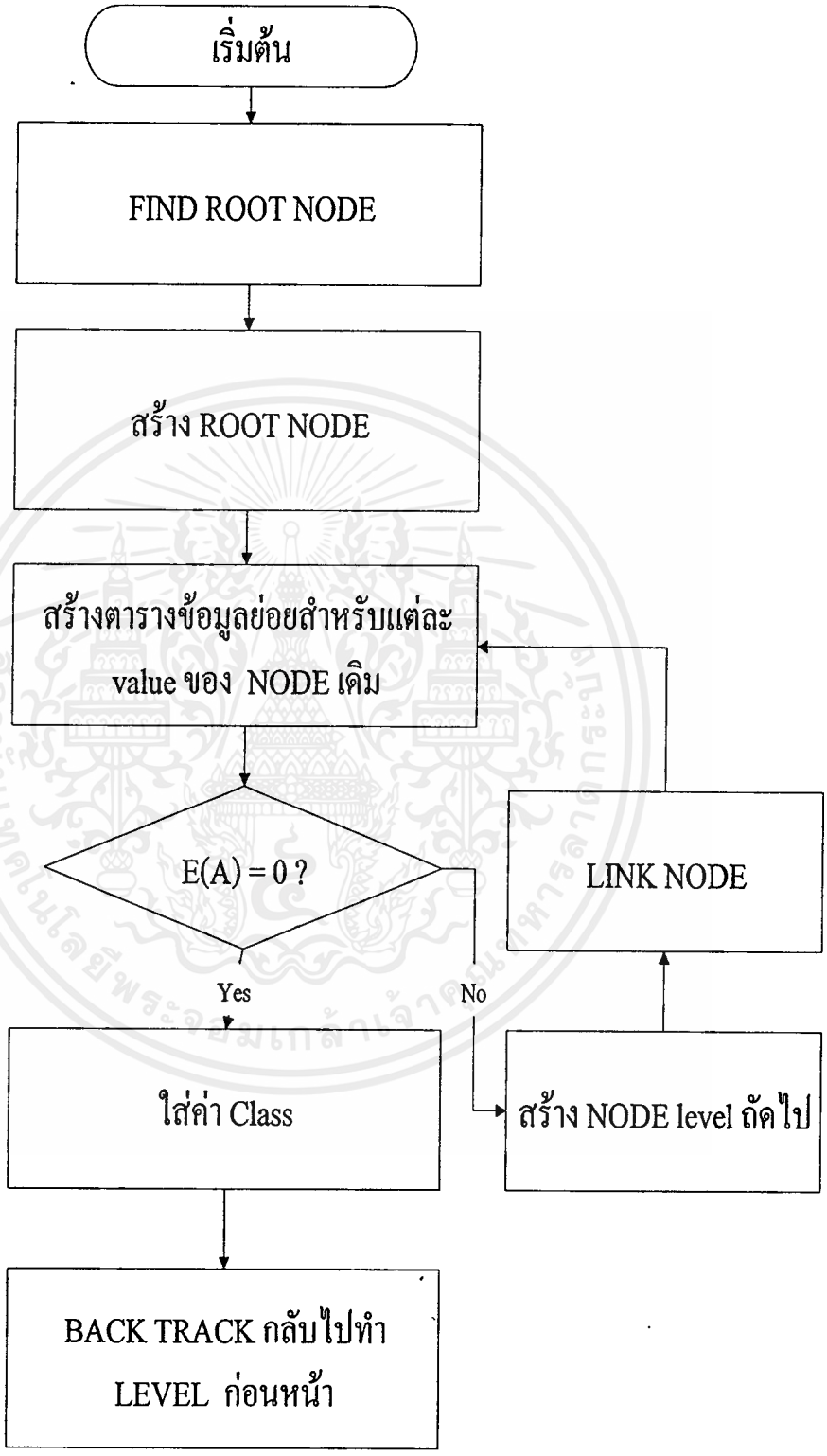
รูปที่ 4-3 แสดง Flowchart ของ User Input Process

FLOWCHART TRANSFER DATA FROM SOURCE FILE TO TABLE



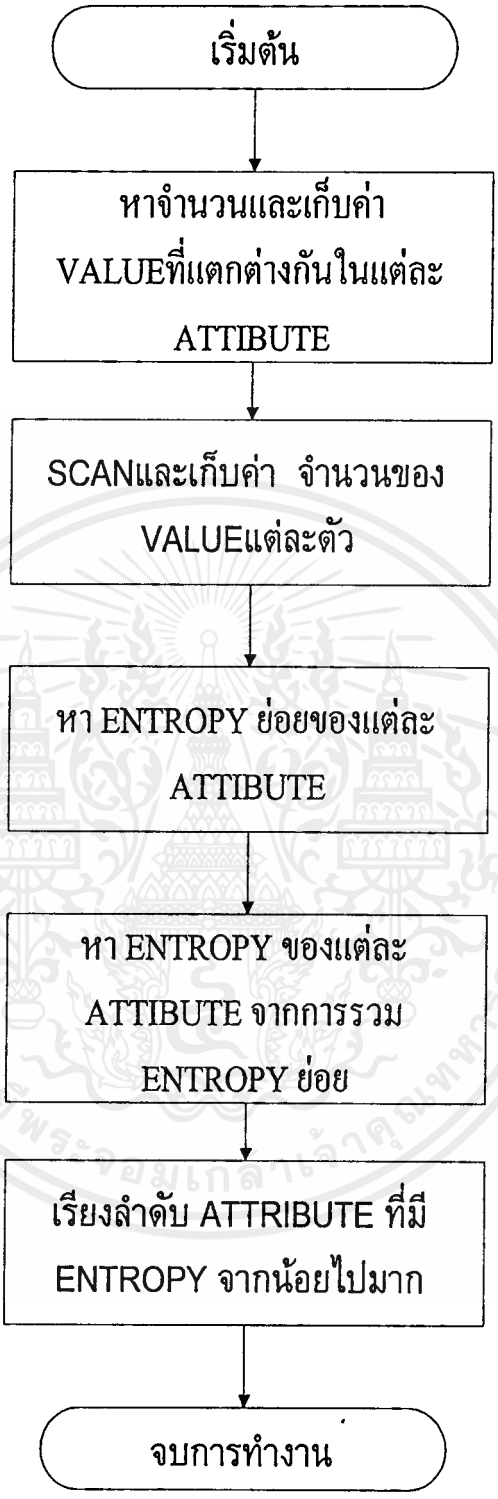
รูปที่ 4-4 แสดง Flowchart ของการ transfer data from source to table

FLOWCHART GENERATE TREE



รูปที่ 4-5 แสดง Flowchart ของการ generate tree

FLOWCHART FIND ROOT NODE

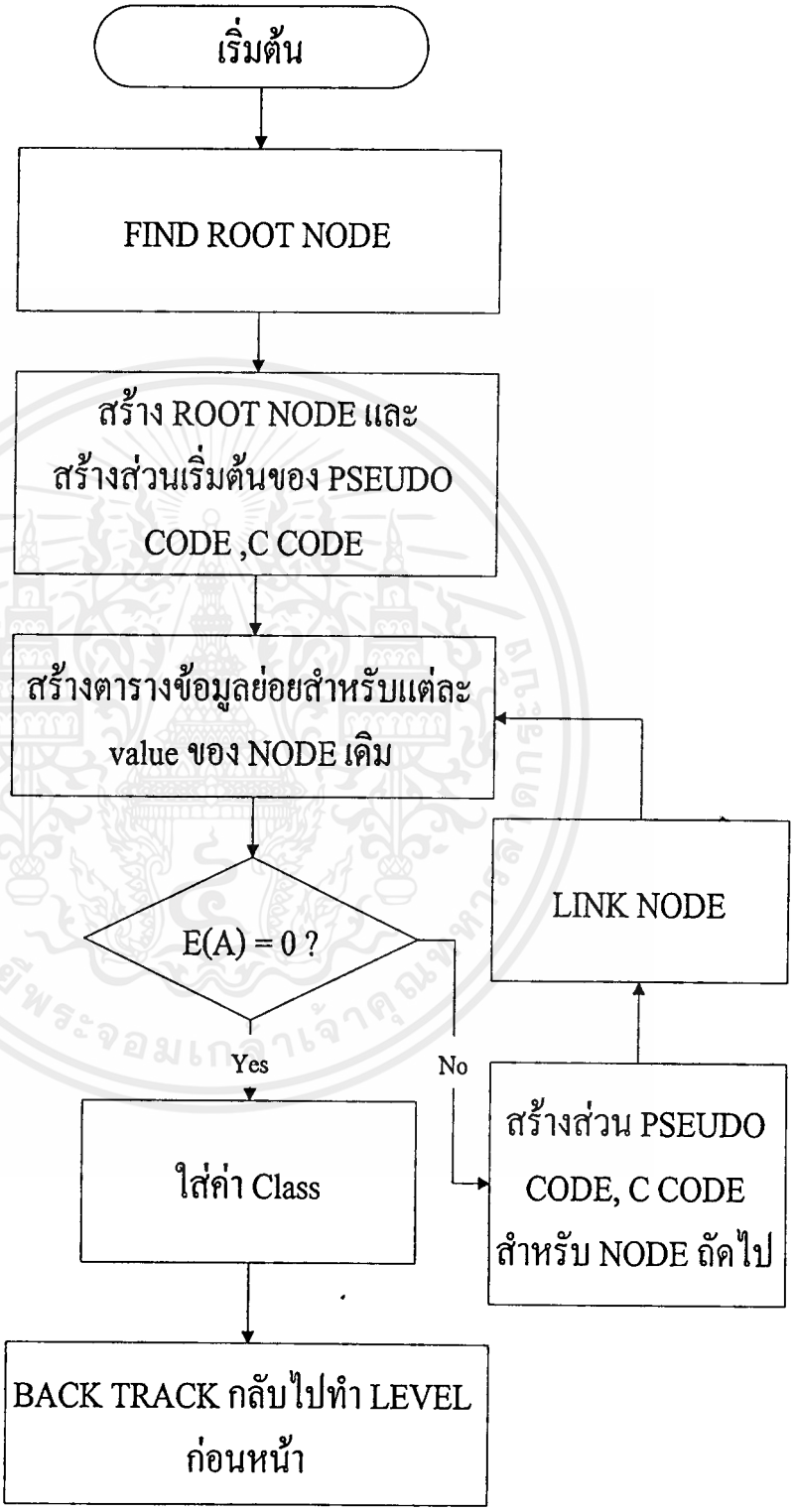


รูปที่ 4-6 แสดง Flowchart การหา root node

FLOWCHART TRAVERSE TREE

รูปที่ 4-7 แสดง Flowchart การ Traverse Tree

FLOWCHART GENERATE PSUDOCODE , C



รูปที่ 4-8 แสดง Flowchart การ generate Pseudo code และ code ภาษา C

บทที่5

โปรแกรมเปลือกระบบผู้เชี่ยวชาญ

SOURCE CODE

```
Object subclass: #LEX
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''
```

" ทำการสร้างคลาส LEX ขึ้นเป็นซับคลาสของคลาส Object ซึ่งเป็นคลาสใหญ่ที่สุดในภาษาสมอลทอกค์ "

"LEX methods"

```
menu
        |p q mes ans|
            p:= Prompter prompt:'Please select 1.Input Training set  2.Use System'
                default:' '.
            p:= p asInteger.
            (p=1)
               ifTrue:[ C := TrainingSet new.
                        q := Prompter prompt:'Please enter file name for write:'
                             default:' '.
                        C writeFile: q.
                        ans:= 'Write file ',q,' complete.'.
                        ^ans
                       ]
               ifFalse:[   (p=2)
                           ifTrue:[ C := TrainingSet new.
                                    q := Prompter prompt:'Please enter file name for read:'
                                         default:' '.
                                    C readFile: q.
                                    ans := A ID3.
```

^ans]

ifFalse:[mes:='error type mismatch'.

^mes].

].

"เป็นเมทอดที่ทำการแสดงเมนูให้ผู้ใช้เลือกว่าจะใช้ระบบผู้เชี่ยวชาญนี้แบบไหน โดยมีให้เลือกในหมวด

1.การ input file โดย

1.1. เลือก input จาก file เก่า

1.2. input เข้าไปใหม่โดยการ key เข้าไป

2.การใช้ระบบผู้เชี่ยวชาญ

2.1. เลือกที่จะปรึกษาหาคำตอบจากการวินิจฉัยของระบบ

2.2. เลือกการ Generate code ภาษา C

2.3. เลือกการ Generate Pseudo code"

LEX subclass: #Acquisition

instanceVariableNames:

'table row col root temp temp1 temp2 temp3 '

classVariableNames: ''

poolDictionaries: ''

" ทำการสร้างคลาส Acquisition ขึ้นเป็นซับคลาสของคลาส LEX โดยมี instance variable ที่มีหน้าที่ต่าง ๆ ดังนี้

table : เป็น array 2 มิติที่ทำหน้าที่เป็นตารางที่ใช้เก็บ training set ซึ่งตารางมีขนาดเท่ากับค่าในตัวแปร col * (row + 5)

row : เป็นจำนวน row ของตาราง Example ไม่รวมหัวตาราง (แอทริบิวท์)

col : เป็นจำนวน column ของตาราง Example

root : เป็น array ขนาด col - 1 ใช้เก็บ แอทริบิวท์ name ที่มี เอนโทรปี เรียงจากน้อยไปมาก

temp,temp1,temp2,temp3 : เป็นตัวแปรช่วยในการเขียนโปรแกรม"

"Acquisition methods "

addc: anInteger1 **r:** anInteger2 **put:** aString

"Input value in training set."

(table at: anInteger1) at: anInteger2 put: aString.

" เป็นเมทอดที่ทำการใส่ค่า aString ลงในตัวแปร table ซึ่งเป็น array 2 มิติ (มองเป็นตาราง) ในตำแหน่ง column คือ anInteger1 และ row คือ anInteger2 "

---

addCol: anInteger

col := anInteger.

" เป็นเมทอดที่ทำการใส่ค่า anInteger ลงในตัวแปร col ของออบเจกต์ Train"

---

addRoot: anIndex **put:** aString

"Input attribute name(column)"

root at: anIndex put: aString.

" เป็นเมทอดที่ทำการใส่ค่า aString ลงในตัวแปร root ของ ออบเจกต์ Train ที่ตำแหน่ง anIndex ซึ่ง root จะเป็น array ที่ใช้เก็บค่าของ แอทริบิวท์ เรียงจาก แอทริบิวท์ ที่มีค่า เอนโทรปี น้อยไปมาก เพราะฉะนั้นค่าในช่องแรกของ array root จะเป็นค่าของ แอทริบิวท์ ที่จะเป็นรูทโนดเสมอ "

---

addRow:anInteger

row:=anInteger.

" เป็นเมทอดที่ทำหน้าที่ในการใส่ค่า anInteger ลงในตัวแปร row ของออบเจกต์ Train ซึ่งตัวแปร row จะเก็บค่าจำนวนแถวของ table "

---

asChar: aNum

```
|ans mod div num tem|
num := aNum.
ans := ''.
[num >= 10]
        whileTrue: [  mod:=num\\10.
                      div  :=num//10.
                      num :=div .
                      mod = 0
                          ifTrue: [ tem:= '0'].
                      mod = 1
                          ifTrue: [ tem:= '1'].
                      mod = 2
                          ifTrue: [ tem:= '2'].
```

```
mod = 3
    ifTrue: [ tem:= '3'].

mod = 4
    ifTrue: [ tem:= '4'].
mod = 5
    ifTrue: [ tem:= '5'].
mod = 6
    ifTrue: [ tem:= '6'].
mod = 7
    ifTrue: [ tem:= '7'].
mod = 8
    ifTrue: [ tem:= '8'].
mod = 9
    ifTrue: [ tem:= '9'].

ans:= tem,ans   ].
num = 0
    ifTrue: [ tem:= '0'].
num = 1
    ifTrue: [ tem:= '1'].
num = 2
    ifTrue: [ tem:= '2'].
num = 3
    ifTrue: [ tem:= '3'].
num = 4
    ifTrue: [ tem:= '4'].
num = 5
    ifTrue: [ tem:= '5'].
num = 6
    ifTrue: [ tem:= '6'].
num = 7
    ifTrue: [ tem:= '7'].
num = 8
```

```
                ifTrue: [ tem:= '8'].
            num = 9
                ifTrue: [ tem:= '9'].
            ans:= tem,ans.
        ^ans
```

" เป็นเมทอดที่ทำหน้าที่ในการแปลงตัวเลขให้เป็นตัวอักษร"

```
createc: anInteger1 r: anInteger2
    "Create table for trainning set."
      table := Array new:anInteger1.
      1 to: anInteger1 do:
      [:i| table at: i put: (Array new: (anInteger2 + 5))].
      col := anInteger1.
      row := anInteger2.
```

" เป็นเมทอดที่ทำการ create table (สร้างตัวแปร table ตารางที่ใช้เก็บ trainning set)ในออบเจกต์ Train ให้เป็น array 2 มิติ ที่มีขนาด column เท่ากับ anInteger1 ขนาด row เท่ากับ anInteger2 + 5 "

```
createRoot
    "Create array for attribute (sort ascending)"
    root:=Array new:(col - 1).
```

" เป็นเมทอดที่ทำการสร้าง array root ในออบเจกต์ Train ซึ่งมีขนาดเท่ากับจำนวน column - 1 เพื่อใช้ในการเก็บค่าแอทริบิวท์ของตารางที่มีค่า เอนโทรปีเรียงจากน้อยไปมาก "

```
difValue: aColumn
  "Find different value for each attribute (aColumn)."
    (table at: aColumn) at:(row + 2) put: (Array new:((table at: aColumn) at: (row + 1))).
    ((table at: aColumn) at:(row + 2)) at:1 put: ((table at: aColumn) at: 1).
    temp1 := 1 .
    temp:=Set new.
    temp add: ((table at: aColumn) at:1).
    2 to: row do:
      [:i|
        (temp includes: ((table at: aColumn) at: i) )
              ifFalse:[ temp add: ((table at: aColumn) at: i).
```

```
                temp1 := temp1 + 1.
                ((table at: aColumn) at:(row + 2)) at: temp1 put: ((table at: aColumn) at:i).
              ]
        ].
^(table at: aColumn) at:(row+ 2)
```

" เป็นเมทอดที่ทำหน้าที่ในการ create array ในช่อง (row + 2) ให้มีขนาดเท่ากับ จำนวน value ที่แตกต่างกัน (อยู่ในช่อง row + 1) แล้วหาค่า value ที่แตกต่างกันของแต่ละ column ในตาราง แล้วเก็บไว้ในตัวแปร table ใน array ที่ตำแหน่ง (row +2) ของแต่ละ column "

---

findColRoot

```
"For find a column that it is root."
A scanColumn: (A readRoot:1).
```

" เป็น เมทอด ที่ทำหน้าที่ในการหาค่าตำแหน่ง column ของแอทริบิวท์ที่เป็นรูทโนด "

---

findPc: aColumn

```
"Find p(ci/akj)  keep it in (row + 3)."
"Create subarray1 in (row+3) =number of value."
 (table at: aColumn) at: (row + 3) put: (Array new:((table at: aColumn) at:(row + 1))).

"Add subarray Valve2"
  1 to: ((table at: aColumn) at:(row + 1)) do:
    [ :i|
     ((table at: aColumn) at: (row + 3)) at: i put: (Array new: (((table at: col) at: (row +1)) + 2))].

"Initial subarray Value2."
  1 to: ((table at: aColumn) at:(row + 1)) do:
    [:j|
    1 to: ((table at: col) at: (row +1)) + 2 do:
      [:i| (((table at: aColumn) at: (row + 3)) at:j) at:i put: 0].
    ].

"Find number of each value "
  1 to:((table at: aColumn) at:(row + 1)) do:
    [:j|
```

```
        1 to:row do:
          [:i|
            ((table at:aColumn) at:i) = (((table at: aColumn) at: (row + 2)) at:j)
              ifTrue: [((((table at: aColumn) at: (row + 3)) at:j) at: ((table at: col) at: (row +1))+1 put:
((((table at: aColumn) at: (row + 3)) at:j) at: ((table at: col) at: (row +1)) +1) + 1]
          ].
      ].

  "Find number value in class"
    1 to:((table at: aColumn) at:(row + 1)) do:
      [:j|
        1 to: ((table at: col) at: (row +1)) do:
          [:k|
            1 to: row do:
              [:i|
                (((table at:aColumn) at:i) = (((table at: aColumn) at: (row + 2)) at:j))
                  ifTrue:[ (((table at:col) at:i) = (((table at: col) at: (row+2)) at:k))
                    ifTrue:[((((table at: aColumn) at: (row + 3)) at:j) at:k put:(((((table
at: aColumn) at: (row + 3)) at:j) at:k) + 1)]]
              ].
          ].
      ].
```

" เป็นเมทอดที่ทำหน้าที่ในการหาค่า $p(c_i|a_{kj})$ ของแต่ละ Column เพื่อใช้ในการคำนวณหาค่า เอนโทรปี โดยมีขั้นตอนดังนี้

1.ในช่อง (row + 3) ให้ create array ที่มีขนาดเท่ากับ ค่าในตำแหน่งที่ (row + 1)และเก็บไว้ในตัวแปร table ที่ตำแหน่ง row + 3 ของแต่ละ column

2.ในแต่ละช่อง sub array ที่ตำแหน่ง (row + 3) ให้ create array ที่มีขนาดเท่ากับค่าจำนวน class ที่แตกต่างกัน + 2

3.scan หาค่า value แต่ละ value ว่ามีกี่ตัว แล้วเก็บไว้ในช่องรองสุดท้าย

4.scan หาค่า value แต่ละ value ที่อยู่ในแต่ละ class ($p(c_i|a_{kj})$) "

ID3

```
    |att menu ans|
    att:= ' '.
```

```
Filename:='Pseudo.txt'.
Filename1:='CC.txt'.
menu := Prompter prompt:'Please enter choise:  1: Traverse 2: Pseudocode 3:C '
        default:' '.
menu := menu asInteger.
A scanValue:A.
A sortRoot.
1 to: (col - 1) do:                    " set write variable for write C code. "
   [:i| (i=1)
        ifTrue:[att:=att ,(A readValuec:i r:(row + 5)),'[30]' ]
        ifFalse:[att:=att ,',',(A readValuec:i r:(row + 5)),'[30]']
   ].

X := KnowledgeBase new.

(menu = 2)
   ifTrue:[ Filename := Prompter prompt:'Please enter file name for Pseudocode: '
                default:' '].

(menu = 3)
   ifTrue:[ Filename1 := Prompter prompt:'Please enter file name for C code: '
                default:' '].
X addValueNode:A.

Output:= Disk newFile: Filename.                    "Pseudo code"
Output nextPutAll: 'case ',X readName,' of ';cr.

Output1:= Disk newFile: Filename1.                   "C code"
Output1 nextPutAll: '#include <stdio.h>';cr.
Output1 nextPutAll: 'main()';cr.
Output1 nextPutAll: '{';cr.
Output1 nextPutAll: '    char  ',att,';';cr.
Output1 nextPutAll: '    printf("Please Enter ',X readName,'  [',X concat,'] : ");';cr.
Output1 nextPutAll: '    scanf( "%s",',X readName,');';cr.
```

```
A linkID3:A node:X.
Output close.
Output1 nextPutAll: '   printf("Undefine!!!!! \n");';cr.
Output1 nextPutAll: '   getch();';cr.
Output1 nextPutAll: '   return(0);';cr.
Output1 nextPutAll: '}';cr.
Output1 close.
(menu = 1)
    ifTrue:[ Z := InferenceEngine new.
        ans := Z travelTree:A node:X.
        ^ans].
(menu = 2)
    ifTrue:[ ans:= 'Pseudo code is in file ', Filename.
        ^ans].
(menu = 3)
    ifTrue:[ ans:= 'C language code is in file ', Filename1.
        ^ans].
```

" main ของส่วนที่ทำการเช็ค input จาก menu

1.หารูทโนด

2.ทำการเริ่มต้นสร้าง node แรกของ tree (Acquisition)

3.ทำการเริ่มต้นการ generate pseudo code โดยทำการเตรียม file สำหรับเก็บ

4.ทำการเริ่มต้นการ generate code ภาษา C โดยทำการเตรียม file สำหรับเก็บ"

```
inAttFrom: aTable1 to: aTable2
  |n|
  n := 1.
  1 to: (aTable1 readCol) do:
    [:i| (aTable1 readValuec: i r: (aTable1 readRow + 5)) = (aTable1 readRoot:1)
      ifFalse:[ aTable2 addc: n r: (aTable2 readRow + 5) put:(aTable1 readValuec: i r: (aTable1
readRow + 5)).
        (n < ((aTable1 readCol) - 1))
          ifTrue:[ aTable2 addRoot:n put: (aTable1 readValuec: i r: (aTable1 readRow + 5))].
        n := n + 1.
```

```
            ].
        ].
```

" เป็นเมทอดที่ทำหน้าที่นำค่าแอทริบิวท์จาก aTable1 ไปใส่ aTable2"

```
linkID3: aTable node: aNode
        "Recursion"
      |class boo boo1 boo2|

1 to:(aNode readTem) do:
    [:i|
            D := Acquisition new.
            Y := KnowledgeBase new.
            boo:=1.
            boo1:=1.
            boo2:=1.

       ((aTable readTemp2:1)=0)
          ifFalse:[
             D tableNode:aTable to:D va:(aNode readValue2:i) node:Y.
             ((D nValue1:(D readCol)) = 1)
                ifTrue:[
                   1 to: (aTable readRow) do:
                [:j|((aNode readValue2:i) = (aTable readValuec:(aTable scanColumn:(aNode
readName)) r: j))
                    ifTrue: [class:= aTable readValuec:(aTable readCol) r:j.
                            (boo = 1)
                            ifTrue:[
                            Output nextPutAll: '    ',(aNode readValue2:i),' : Class := ',class;cr .
                            Output1 nextPutAll: '       if ( strcmp( ',(aNode readName),',"',(aNode
readValue2:i),'")==0)   printf ("Class is  ',class,' \n ");';cr.
                            Output1 nextPutAll: '      else';cr.
                            boo := 2.
                                ].
                           ].
```

```
        ].
        aNode putPointer:i value: class.
                ]
            ifFalse:[

        aNode addLink:Y i:i.
        (boo1=1)
        ifTrue:[
        Output nextPutAll: '    ',(aNode readValue2:i),' : case ',Y readName,' of ';cr.

         Output1 nextPutAll: '        if ( strcmp( ',(aNode readName),',"',(aNode
readValue2:i),'")==0)';cr.
        Output1 nextPutAll: '    {';cr.
        Output1 nextPutAll: '        printf("Please Enter ',Y readName,' [',Y concat,'] : ");';cr.
        Output1 nextPutAll: '        scanf( "%s",',Y readName,');';cr.
         boo1:=2.
            ].
        aTable linkID3:D node:Y.
        Output1 nextPutAll: '        printf("Undefine!!!! \n");';cr.
        Output1 nextPutAll: '    }';cr.
        Output1 nextPutAll: '    else';cr.

          ].
         ]
      ifTrue:[
        1 to: (aTable readRow) do:
          [:j|((aNode readValue2:i) = (aTable readValuec:(aTable scanColumn:(aNode
readName)) r: j))
            ifTrue: [class:= aTable readValuec:(aTable readCol) r:j.
                (boo2 = 1)
                ifTrue:[
                Output nextPutAll: '        ',(aNode readValue2:i),' : Class := ',class;cr .
                Output1 nextPutAll: '        if ( strcmp( ',(aNode readName),',"',(aNode
readValue2:i),'")==0)    printf ("Class is  ',class,' \n ");';cr.
```

```
                    Output1 nextPutAll: '          else';cr.
                    boo2:=2.
                        ].
                    ].
                ].
            aNode putPointer:i value: class.
                ].
        ].
```

" เป็นเมทอดที่ทำหน้าที่

1.สร้าง decision tree

2.generate pseudo code

3.generate code ภาษา C

โดยใช้วิธีการ recursion โดยทำการกำหนดจุดหยุดที่ ค่า เอนโทรปี = 0 หมายความว่า node นั้นเป็น leaf node ของ decision tree แล้ว ตามทฤษฎีของ ID3 "

```
makeTablec: anInteger1 r: anInteger2
    D createc: anInteger1 r: anInteger2.
    D createRoot.
    D addCol:anInteger1.
    D addRow:anInteger2.
```

" เป็นเมทอดที่ทำหน้าที่เป็น main ในการเรียกใช้เมทอดที่ทำการสร้าง table สร้าง root เก็บค่า column และ row "

```
numRo: aString train: aTable
  "Find number of aTable row that attribute is root and value aString"
    |co ro|
    co := aTable scanColumn: (aTable readRoot:1).
    1 to: (aTable readValuec:co r:(aTable readRow + 1)) do:
      [:i| (((aTable readValuec:co r:(aTable readRow + 2)) at:i) = aString)
          ifTrue: [ ^((aTable readValuec:co r:(aTable readRow + 3)) at:i) at:(((aTable readValuec:(aTable readCol) r:(aTable readRow + 1)) + 1))
                ].
    ].
```

" เป็น เมทอด ที่ทำหน้าที่ในการหาจำนวน row ของ ออบเจกต์ aTrain สำหรับแอทริบิวท์ที่เป็น root และ value มีค่าเป็น aString "

```
numVa: aString train: aTable
  "Find number of aTable row that attribute is root and value aString"
    |co ro|
    ro:=1.
    co := aTable scanColumn: (aTable readRoot:1).

    1 to:((table at: co) at:(row + 1)) do:
      [:i| ((((table at: co) at: row+2) at:i) = aString)
          ifTrue: [ ^(((table at: co) at: row + 3) at:i) at:(((table at: col) at: row + 1) +1) ].
      ].
```

" เป็น เมทอด ที่ทำหน้าที่ในการหาจำนวน value ใน table"

```
nValue: aColumn
  "Find number of different value for each attribute (aColumn)."
    temp := Set new.
    temp add: ((table at: aColumn) at:1).
    2 to: row do:
      [:i|
        (temp includes: ((table at: aColumn) at: i) )
            ifFalse:[ temp add: ((table at: aColumn) at: i) ].
      ].
    (table at: aColumn) at: (row + 1) put: temp size.
```

" เป็นเมทอดที่ทำหน้าที่ในการหาจำนวน value ที่แตกต่างกันของแต่ละ แอทริบิวท์ ที่อยู่ใน aColumn แล้วเก็บใน table ตำแหน่งที่ row + 1 ของแต่ละ column "

```
putValuec: anInteger1 r: anInteger2 va: aString
    ((table at: anInteger1) at: anInteger2 put: aString).
```

"เป็นเมทอดที่ทำหน้าที่ในการใส่ค่า aString ลงใน table ที่ตำแหน่ง column anInteger1 และ row anInteger2 "

```
readCol
```

```
    ^col
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่า column ของ table ที่เก็บไว้ในตัวแปร col ออกมา "

```
readRoot: aNum
    "For read value in array root."
    ^(root at: aNum).
```

" เป็นเมทอดที่ทำหน้าที่ในการอ่านค่าจาก array ที่ตำแหน่ง aNum "

```
readRow
    "Return row value."
    ^row
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่า row ของ table ที่เก็บไว้ในตัวแปร row ออกมา "

```
readTemp2:i
    ^temp2 at:i.
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าจากตัวแปร temp2 ซึ่งเป็น array ตำแหน่งที่ i ออกมา "

```
readValuec: aColumn r: aRow
    "For read value from table."
    ^(table at: aColumn) at: aRow.
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าจาก table ที่ตำแหน่ง column ที่ aColumn และตำแหน่ง row ที่ aRow "

```
scanColumn: aString
    "For find a column that aString."
        temp1:=1.
        [((table at: temp1) at: (row+5)) = aString]
            whileFalse: [(temp1 := temp1+1)].
        ^temp1
```

" เป็นเมทอดที่ทำหน้าที่ในการหาค่าตำแหน่ง column ของ แอทริบิวท์ ที่ชื่อ aString "

```
scanValue:aTable
    "Scan value for find root node ..Input table."
    1 to: col do:
```

```
[:i| aTable nValue: i.
    aTable difValue: i ].
1 to: (col - 1) do:
[:i| aTable findPc:i.
    aTable subEntropy:i ].
```

" เป็นเมทอดที่ทำหน้าที่เป็น main ในส่วนของการหาค่าที่จะต้องใช้ในการคำนวณหาค่าเอนโทรปี เพื่อหา รูทโนด ด้วยวิธีการ ID3 "

```
sortRoot
  "Sort Entropy  Ascending order."
  temp2:=Array new:(col - 1).
  1 to: (col - 1) do:
  [:i| temp2 at: i put: ((table at: i) at: (row + 4))].

 1 to:( col - 2 ) do:
  [:j|
  1 to:( col - 2 )  do:
  [:i| (temp2 at:i) > (temp2 at:(i+1))
       ifTrue: [ temp:= (temp2 at:i).
                 temp2 at: i put: (temp2 at: (i+1)).
                 temp2 at: (i+1) put: temp.
                temp:=(root at:i).
                root at: i put: (root at:(i+1)).
                root at: (i+1) put: temp.

               ].
  ].
  ].
```

" เป็นเมทอดที่ทำหน้าที่ในการเรียงลำดับค่าเอนโทรปีของแอทริบิวท์จากน้อยไปมาก เก็บไว้ใน array root ซึ่ง แอทริบิวท์ ที่มีค่า เอนโทรปี น้อยที่สุดจะเป็น รูทโนด "

```
subEntropy: aColumn
 "Find subentropy"
    1to: ((table at: aColumn) at:(row + 1))  do:
```

```
[:j|
  Suben:=0.
  1 to: ((table at: col) at: (row +1)) do:
    [:i|
      ((((table at: aColumn) at: (row + 3)) at: j) at: i) = 0
          ifTrue: [Suben:=Suben+0]
          ifFalse: [
        Suben := Suben +
        (0 - (((((table at: aColumn) at: (row + 3)) at:j) at:i) /((((table at: aColumn) at: (row + 3))
at:j) at:(((table at: col) at: (row +1)) + 1))) *
        ((((((table at: aColumn) at: (row + 3)) at:j) at:i) / ((((table at: aColumn) at: (row + 3)) at:j)
at:(((table at: col) at: (row +1))+1))) log: 2))]
      ].
    Suben:=Suben * (((((table at: aColumn) at: (row + 3)) at:j) at:(((table at: col) at: (row +1)) +
1))/row).
    (((table at: aColumn) at: (row + 3))at:j) at:(((table at: col) at: (row +1)) + 2) put: Suben.
  ].

  "Calculate total entropy for each attribute."
    Sum:=0.
    1 to: ((table at: aColumn) at:(row + 1)) do:
      [:i|
        Sum:=Sum + ((((table at: aColumn) at: (row + 3)) at:i) at:(((table at: col) at: (row +1)) +
2)).
      ].
    (table at: aColumn) at: (row + 4)  put:Sum.
```

" เป็นเมทอดที่ทำหน้าที่ในการคำนวณหาค่าเอนโทรปีย่อยสำหรับแต่ละ value ในแต่ละแอทริบิวท์ และหาค่าเอนโทรปีรวมสำหรับแต่ละแอทริบิวท์"

```
tableNode:aTable1 to:aTable2 va:aString node:aNode
  "make table from A to D and make Node."
    aTable1 makeTablec: (aTable1 readCol - 1) r: (aTable1 numVa:aString train:aTable1).
    aTable1 inAttFrom: aTable1 to:aTable2.
    aTable1 transferCol:(aTable1 scanColumn:(aTable1 readRoot:1)) va:aString.
```

```
aTable2 scanValue:aTable2.
aTable2 sortRoot.
aNode addValueNode:aTable2.
```

" เป็นเมทอดที่ทำหน้าที่เป็น main ในส่วนของการ generate subtable เพื่อหา node ต่อไปที่เหมาะสมที่สุดสำหรับ decision tree "

```
transferCol: anInteger va: aString
  "Col : column to cut, va: value of attribute (to cut)."
  " Transfer data from aTrain1 to aTrain2."
    |n|
    n:=1.
    1 to: (A readRow) do:
      [:i| ((A readValuec: anInteger r:i) = aString)
          ifTrue: [ 1 to: (A readCol) do:
                    [:j| 1 to: (D readCol) do:
                        [:k| (D readValuec:k r:(D readRow) + 5) = (A readValuec:j r: (A readRow)
+ 5)
                            ifTrue:[D putValuec:k r:n va: (A readValuec:j r:i)
                                ].
                    ].
                ].
          n := n+1.          ].
      ].
```

" เป็นเมทอดที่ทำหน้าที่ในการ transfer data จากตารางใหญ่ไปตารางเล็ก เพื่อใช้ในการคำนวณหา node ที่เหมาะสมต่อไปของ decision tree "

```
Acquisition subclass: #TrainingSet
 instanceVariableNames:
  'c r input output temp4 temp5 '
 classVariableNames: "
 poolDictionaries: "
```

" ทำการสร้างคลาส TrainingSet ขึ้นเป็นซับคลาสของคลาส Acquisition โดยมี instance variable ที่มีหน้าที่ต่าง ๆ ดังนี้

c : เป็นตัวแปรที่ใช้เก็บค่าจำนวน column ของ Example table

r : เป็นตัวแปรที่ใช้เก็บค่าจำนวน row ของ Example table

input : เป็นตัวแปรที่ใช้เก็บค่า path name ของ file ที่จะ input (read) เข้ามา

output : เป็นตัวแปรที่ใช้เก็บค่า path name ของ file ที่จะ output (write) ออกไป

temp4,temp5 : เป็นตัวแปรที่ใช้ช่วยในการเขียน program "

"TrainingSet methods"

readFile: fileName

```
"input file from fileName"
        input:= Disk file: fileName.
        A:= Acquisition new.
        col:= (input nextLine) asInteger.
        row:= (input nextLine) asInteger.
        A createc:col r:row.
        A createRoot.
        1 to: (col - 1) do:
          [:i| temp4:= input nextLine.
              A addc: i r: (row + 5) put: temp4.
              A addRoot:i put: temp4 ].
          temp4:= input nextLine.
          A addc: col r: (row + 5) put: temp4.

"input data from fileName to table"
        1 to: (col) do:
          [:i|
              1 to:row do:
               [:j|   temp4 :=  input nextLine.
                    A addc: i r:j put: temp4 ].
          ].
        input close.
```

" เป็นเมทอดที่ทำหน้าที่ในการ input file Example table เข้ามาใช้เป็น Training set "

```
writeFile: fileName
        |p x tmp att nameAtt|
         output:= Disk newFile: fileName.
         p:=Prompter prompt:'Please enter number of column.'
            default:' '.
         output nextPutAll: p;cr.
         col:= p asInteger.

         p:=Prompter prompt:'Please enter number of row.'
            default:' '.
         output nextPutAll: p;cr.
         row:= p asInteger.
  "output attibute to file fileName"
         nameAtt := Array new: col.
         1 to: col  do:

         [:i|   tmp:= A asChar: i.
                tmp:= 'Please enter attribute at column: ' ,tmp.
                p:=Prompter prompt:tmp
                   default:' '.
                output nextPutAll: p;cr.
                nameAtt at: i put:p ].

  "output data to file fileName"
        1 to: col do:
        [:i|
           1 to:row do:
           [:j|   tmp:= A asChar: j.
                  att:= nameAtt at: i.
                  tmp:= 'Please enter data at column: ' , att, ' row:  ', tmp.
                  p:=Prompter prompt:tmp
                  default:' '.
                  output nextPutAll: p;cr].
```

```
].
output close.
```

" เป็นเมทอดที่ทำหน้าที่ในการรับข้อมูล Example table เข้ามาเก็บไว้เป็น file"

```
TrainingSet subclass: #InferenceEngine
 instanceVariableNames: "
 classVariableNames: "
 poolDictionaries: "
```

" ทำการสร้างคลาส InferenceEngine ขึ้นเป็นซับคลาสของคลาส TrainingSet"

"InferenceEngine methods "

```
travelTree:aTable node:aNode
        |co ro set nextn i y boo j er|
        co := aTable readCol.
        ro := aTable readRow.
        set := Set new.
        1 to: (aTable readValuec: co r:( ro + 1))  do:
            [:i|set add: ((aTable readValuec: co r:( ro + 2 )) at: i)].

        aNode question: aNode.

        ( Ans='000' )
        ifFalse:[
            i:=(aNode scanPointer: Ans).
            (set includes: (aNode readPointer:i))
                ifTrue: [^ (aNode readPointer:i)].

            nextn := aNode search: aNode.
            nextn question: nextn.
            ( Ans='000' )
            ifFalse:[
               [set includes: (nextn readPointer:(nextn scanPointer: Ans))]
                   whileFalse: [ nextn := nextn search: nextn.
```

```
                    nextn question: nextn].

            ^ (nextn readPointer:(nextn scanPointer: Ans))
                ]
        ifTrue:[ ^y:='Input Error!!!!!!!!!!!!'].
            ]
    ifTrue:[ ^y:='Input Error!!!!!!!!!!!!'].
```

" เป็นเมทอดที่ทำหน้าที่ในการ traverse tree เพื่อหาค่า class ของผลที่ทำการวิเคราะห์จาก decision tree "

```
TrainingSet subclass: #KnowledgeBase
 instanceVariableNames:
  'name value pointer tem tem1 tem2 tem3 '
 classVariableNames: ''
 poolDictionaries: ''
```

" ทำการสร้างคลาส Knowledge ขึ้นเป็นซับคลาสของคลาส TrainingSet"

"KnowledgeBase methods"

```
addLink: nextNode i: anInteger
    pointer at:anInteger put: nextNode.
```

" เป็นเมทอดที่ทำหน้าที่ในการ link nextNode เข้ากับออบเจกต์ KnowledgeBase ที่ pointer ที่ i "

```
addTem2: anInteger
    tem2:=anInteger.
```

"เป็นเมทอดที่ทำหน้าที่ในการใส่ค่าเข้าตัวแปร tem2 "

```
addTem3: anInteger
    tem3:=anInteger.
```

"เป็นเมทอดที่ทำหน้าที่ในการใส่ค่าเข้าตัวแปร tem3 "

```
addValueNode: aTable
    |t|
    name := aTable readRoot:1.
```

```
tem1:= aTable readRow.
tem  := aTable readValuec: (aTable scanColumn: name) r:(tem1 + 1).
value := Array new: tem.
1 to: tem do:
  [:i| value at:i put: ((aTable readValuec: (aTable scanColumn:name) r:(tem1 + 2)) at:i)].
pointer := Array new: tem.
```

" เป็นเมทอดที่ทำการใส่ค่าใน ตัวแปรต่าง ๆ ของออบเจกต์ KnowledgeBase เพื่อทำหน้าที่เป็น node ของ decision tree "

```
concat
        |va|
      va:=' '.
      1 to: tem do:
        [:i| va:= va,(value at: i),' '].
^va
```

" เป็นเมทอดที่ทำหน้าที่ในการเชื่อมตัวอักษรเพื่อใช้ในการสร้าง code ภาษา C "

```
linkTree: aNode
    1 to: (aNode readTem ) do:
        [:i|
    ((aNode readPointer: i ) = 'Undifind')
      ifFalse:[ N1 := Node2 new.
                N1 addValueNode: aNode readTem3.
                aNode addLink: N1 i: i.
                N1 addTem3: (aNode readTem3 + 1).
                aNode linkTree: N1.
              ].
        ].
      aNode addTem3: (aNode readTem3 - 1).
```

" เป็นเมทอดที่ทำหน้าที่ในการ link node ต่าง ๆ เข้าด้วยกันเป็น decision tree "

```
putPointer: anInteger value: aString
    pointer at: anInteger put:aString.
```

" เป็นเมทอดที่ทำหน้าที่ในการใส่ค่า class (aString) ลงในตัวแปร pointer ที่ตำแหน่งที่ anInteger สำหรับ node ที่เป็น leaf node "

---

question: aNode

```
|p x a l boo j er|
a:=aNode readName.
l:=aNode readValue: aNode.
x:='HOW ABOUT ',a,' ? ','( ',l,')' .
p:=Prompter prompt: x
    default:' '.
Ans:= p.
er:=2.
j:=1.
boo:=2.
[boo = 1]
   whileFalse:[( j<= (aNode readTem))
                  ifTrue:[ (Ans =  (aNode readValue2:j))
                        ifTrue:[ boo:=1.]
                        ifFalse:[ j:=j+1.].
                     ]
                  ifFalse:[boo:=1.
                        er:=1].
               ].
(er=2)
   ifTrue:[^Ans]
   ifFalse:[^Ans:='000'].
```

" เป็นเมทอดที่ทำหน้าที่ในการแสดงหน้าจอ เพื่อสอบถามข้อมูลกับ user และ return ค่าเป็นคำตอบที่ user ตอบมา "

---

readName

```
^name
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร name "

---

readPointer: anInteger

```
^(pointer at: anInteger)
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร pointer ที่ตำแหน่ง anInteger "

---

readTem

```
^tem
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร tem "

---

readTem2

```
^tem2
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร tem2 "

---

readTem3

```
^tem3
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร tem3 "

---

readValue2: anInteger

```
^(value at: anInteger)
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร value ที่ตำแหน่ง anInteger "

---

readValue: aNode

```
|q j|
j:= ''.
1 to: tem do:
  [:i| q:= aNode readValue2: i .
     j:= j,q,' , ' ].
^j
```

" เป็นเมทอดที่ทำหน้าที่ในการ return ค่าของตัวแปร value ทุกค่าที่แตกต่าง "

---

scanPointer: aString

```
1 to: tem do:
  [:i| (aString = (value at: i))
     ifTrue:[ ^i ]. ].
```

"เป็นเมทอดที่ทำหน้าที่ในการ scan หาตำแหน่ง pointer ที่ตรงกับตำแหน่งของ value aString "

```
search: aNode
    |k j|
    1 to: tem do:
        [:i|  k:= aNode readValue2: i.
            (Ans = k)
                ifTrue:[
                    j:=aNode readPointer:i].
        ].
    ^j
```

" เป็นเมทอดที่ทำหน้าที่ในการหาตำแหน่ง pointer ที่จะทำการ link node ถัดไป "

บทที่6

บทวิจารณ์ และสรุปผลการทดลอง

6.1เปรียบเทียบข้อแตกต่างระหว่าง Object Oriented Programming กับ Procedural Structure Programming

1 วิธีการจัดการกับข้อมูล (Method for accomplishing actions with data)

- OOP: จะส่ง message (action) ไปที่ Object (data) และ Object ที่ได้รับ message ก็จะทำการตอบสนองต่อ message นั้นตาม method ที่เหมาะสมกับ message ที่ได้รับ

- Procedural: ก่อนที่ Procedure (action) จะทำการจัดการกับ data นั้น ๆ จะมีการส่งค่า parameter (data) ไปที่ Procedure ก่อน

2.Abstraction and Encapsultion

2.1 Abstraction

- Data Abstraction

OOP: Data Abstraction จะถูกแทนโดย class ของ Object

Procedural: Data Abstraction จะถูกแทนโดย data type ของแต่ละภาษา

- Function Abstraction

OOP: Function Abstraction จะถูกแทนโดย message

Procedural: Function Abstraction จะถูกแทนโดยการทำงานของ procedural

2.2 Encapsulation

OOP: 1 encapsulation คือ 1 object โดยภายใน encapsulation จะประกอบไปด้วย method และ private data ซึ่งเป็น instance ของ Class หนึ่ง ๆ เท่านั้น ซึ่งหนึ่ง class จะแทน object เพียงชนิดเดียวเท่านั้น แต่หลาย object อาจเป็น instance ของ class เดียวกันได้ ถ้าเป็น object ชนิดเดียวกัน

Procedural: การ encapsulation จะอยู่ในรูปของ library software component ซึ่งแต่ละ component อาจจะมีมากกว่า 1 data abstraction อยู่ด้วยกันได้

3.Inheritance and Polymorphism

OOP: จะ support inheritance and polymorphism

Procedural: จะไม่ support inheritance and polymorphism

4. Extensibility and relative status of new protocol

Extensibility คือ คุณสมบัติของภาษาทาง computer ที่จะอนุญาตให้ผู้ใช้กำหนดโครงสร้างใหม่ ๆ ได้

OOP: อนุญาตให้ผู้ใช้สามารถกำหนดโครงสร้างขึ้นมาใหม่ได้ โดยที่โครงสร้างที่สร้างขึ้นมาใหม่จะมีสถานะเทียบเท่ากับที่ system เป็นผู้กำหนด

Procedural: ถึงแม้ว่าใน Procedural จะอนุญาตให้ผู้ใช้สามารถกำหนดโครงสร้างขึ้นมาใหม่ได้เช่นเดียวกันกับ OOP แต่โครงสร้างที่สร้างขึ้นมาใหม่จะมีความสำคัญและประสิทธิภาพน้อยกว่าโครงสร้างที่สร้างจาก system

5. Class hierarchy

OOP: ใน OOP จะ support inheritance และ polymorphism จึงทำให้มีมีความสามารถในการ support hierarchy ของ class ได้ จึงมีประโยชน์ในกรณีที่ต้องการเพิ่มความสามารถให้กับ Class hierarchy ผ่านไปยัง subclass

Procedural: ไม่ support ทั้ง inheritance และ polymorphism แต่ละ component ของ software จะมีสถานะเท่ากันหมด จึงทำให้เกิดความซ้ำซ้อนขึ้นมาได้

6. Passing Parameter

OOP: การ pass parameter เข้า - ออก จะเป็นการ pass ของ object โดยสามารถทำได้ทั้ง private data และ share data สำหรับแต่ละ object

Procedural: การ pass parameter จะเป็นการ pass ของ data type เท่านั้น

6.2 ข้อดีของ Object Oriented Programming

1.Reusability: ภาษา OOP สามารถ reuse module ได้ เช่น OOP จะอนุญาตให้ program ที่ถูกสร้างขึ้นมาใหม่ สามารถเรียกใช้ method จากโปรแกรมเดิมได้

2.Portability: เพราะ OOP support คุณสมบัติ Polymorphism ซึ่งแตกต่างจากภาษา Procedural ซึ่งไม่ support คุณสมบัติ Polymorphism จึงทำให้เกิดปัญหายุ่งยากในการตั้งชื่อ Procedure Operation ที่เหมือนกันแต่กระทำต่าง component library กันจะต้องตั้งชื่อต่างกัน แต่ใน OOP สามารถตั้งชื่อ method ซ้ำกันได้ เพราะว่า Object สามารถเลือกตอบสนอง message โดยใช้ method ที่เหมาะสม

3.Reliability and Integrity: เนื่องจากภาษา OOP มีคุณสมบัติ Encapsulation หรือมีการทำ Information hiding คือ OOP จะเก็บ data ไว้ใน Object ซึ่งจะถูกจัดการได้โดย method ที่เหมาะสมเท่านั้น ดังนั้นจึงทำให้ data มีความปลอดภัยยิ่งกว่า แบบ Procedural

4.Flexibility: เนื่องจากภาษา OOP มีคุณสมบัติ Inheritance จึงทำให้มีความยืดหยุ่นมาก เช่น ในเราต้องการเพิ่มความสามารถให้กับ ทุก ๆ subclass เราก็เพียงแต่เพิ่มใน superclass เท่านั้น ทุก ๆ subclass ก็จะถูกเพิ่มความสามารถนั้น ๆ ไปด้วย

5. Understandability: ภาษา OOP มีลักษณะที่ตรงกับความเข้าใจของมนุษย์ได้ดีกว่า

6.Maintainability: ภาษา OOP ง่ายต่อการเพิ่มเติมเมื่อมี requirment เพิ่มขึ้น

7.Easy to Development : ในการพัฒนาโปรแกรมเราไม่จำเป็นต้องสนใจถึงระดับ code ว่า code เป็นอย่างไร แค่รู้ว่า ส่ง message เข้าไปใน object นี้แล้วได้ผลลัพธ์ออกมาเป็นอย่างไรเท่านั้นพอ จึงทำให้สามารถพัฒนาโปรแกรมแบบขนานกันไปได้

8.Virifiability: ภาษา OOP สามารถตรวจสอบหาข้อผิดพลาดได้ง่าย

6.3 ข้อเสียของ Object Oriented Programming

1. เนื่องจาก OOP เป็นเทคนิคแบบใหม่ ดังนั้น tool สำหรับ OOP จึงยังไม่ค่อยพัฒนามากนัก โดยเฉพาะ tool ที่เป็น pure Object Oriented Programming จริงๆ
2. ไม่มีสัญลักษณ์ที่เป็นมาตรฐานในการออกแบบ แบบ Object Oriented จริงๆ ดังนั้นจึงควรจะมีการกำหนดสัญลักษณ์ที่เป็น มาตรฐานอันเดียวกันขึ้นมา เพื่อผู้ใช้สามารถที่จะทำความเข้าใจได้ง่ายๆ
3. การพัฒนาทางด้าน Programming language ยังพัฒนาไปได้ไม่มากนักทั้งๆที่จริงๆแล้วอาจสามารถพัฒนาไปได้ไกลกว่านี้
4. จากคุณสมบัติ inheritance อาจทำให้เกิดปัญหาเรื่องความสัมพันธ์ระหว่างคลาสได้
5. จากคุณสมบัติของ polymorphism จะทำให้การ ทดสอบโปรแกรมมีความยุ่งยากซับซ้อนขึ้น

จะเห็นได้ว่าข้อเสียส่วนใหญ่ของการพัฒนาโปรแกรมในรูปแบบของ Object Oriented เทคนิค คือยังมีการพัฒนาไปได้ไม่มากนักและจากข้อเสียที่กล่าวมา ถ้าผู้ใช้ได้ทำการศึกษามาเป็นอย่างดีแล้วก็จะสามารถพัฒนาโปรแกรมไปได้อย่างดี

ภาคผนวก

USER MANUAL FOR LEX EXPERT SYSTEM SHELL (LESS)

ในคู่มือนี้จะแบ่งเนื้อหาออกเป็นส่วนต่างๆ ดังต่อไปนี้

1. การเตรียมเครื่องมือและอุปกรณ์ต่างที่จำเป็นสำหรับการใช้งานโปรแกรม LESS
2. ขั้นตอนการติดตั้งโปรแกรม LESS
3. การเตรียมข้อมูลหรือกลุ่มตัวอย่าง (TRAINNING SET) ก่อนนำเข้าสู่โปรแกรม LESS
4. การใส่ข้อมูลหรือกลุ่มตัวอย่าง เข้าสู่โปรแกรม LESS ซึ่งประกอบด้วย 3 วิธีการดังต่อไปนี้
    - 4.1 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทางส่วนการรับข้อมูลของโปรแกรม LESS
    - 4.2 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทาง TEXT FILE ธรรมดา
    - 4.3 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยการดึงมาจาก แฟ้มข้อมูลเก่า
5. วิธีการใช้โปรแกรม LESS ในการหาข้อสรุปและตอบคำถาม
6. วิธีการใช้โปรแกรม LESS ในการสร้าง PSEUDO CODE
7. วิธีการใช้โปรแกรม LESS ในการสร้าง CODE ภาษา C

1.การเตรียมเครื่องมือและอุปกรณ์ต่างที่จำเป็นสำหรับการใช้งานโปรแกรม LESS

1.1 การเตรียมความพร้อมทางด้าน HARDWARE

1.1.1 เครื่อง PC รุ่น 486 ขึ้นไป

1.1.2 หน่วยความจำภายใน (RAM) ขนาด 8 เมกะไบต์ขึ้นไป

1.1.3 หน่วยความจำภายนอก (HARDDISK) ขนาด 5 เมกะไบต์ขึ้นไป

1.2 การเตรียมความพร้อมทางด้าน SOFTWARE

1.2.1 โปรแกรมระบบปฏิบัติการ WINDOW 95

1.2.2 โปรแกรม SMALLTALK FOR WINDOW

1.2.3 โปรแกรม LESS

2. ขั้นตอนการติดตั้งโปรแกรม LESS

2.1 ติดตั้งระบบปฏิบัติการ WINDOW 95

2.2 ติดตั้งโปรแกรม SMALLTALK FOR WINDOW

2.3 ติดตั้งโปรแกรม LESS โดยมีรายละเอียดการทำงานดังนี้

2.3.1 เข้าสู่การทำงาน ของโปรแกรม SMALLTALK FOR WINDOW

2.3.2 ที่หน้าจอโหมดการทำงาน SMALLTALK \V TRANSCRIPT ใช้คำสั่ง

(File pathName: 'c:\vwin\less.cls')

fileIn

* เพื่อเป็นการ Install โปรแกรม LESS ลงใน SMALLTALK FOR WINDOW *

2.3.3 เมื่อสิ้นสุดขั้นตอน 2.3.2 จะส่งผลให้เกิด CLASS ใหม่ขึ้นใน SMALLTALK FOR WINDOW คือ CLASS LEX

2.3.4 ทำการใช้งานได้ต่อไป

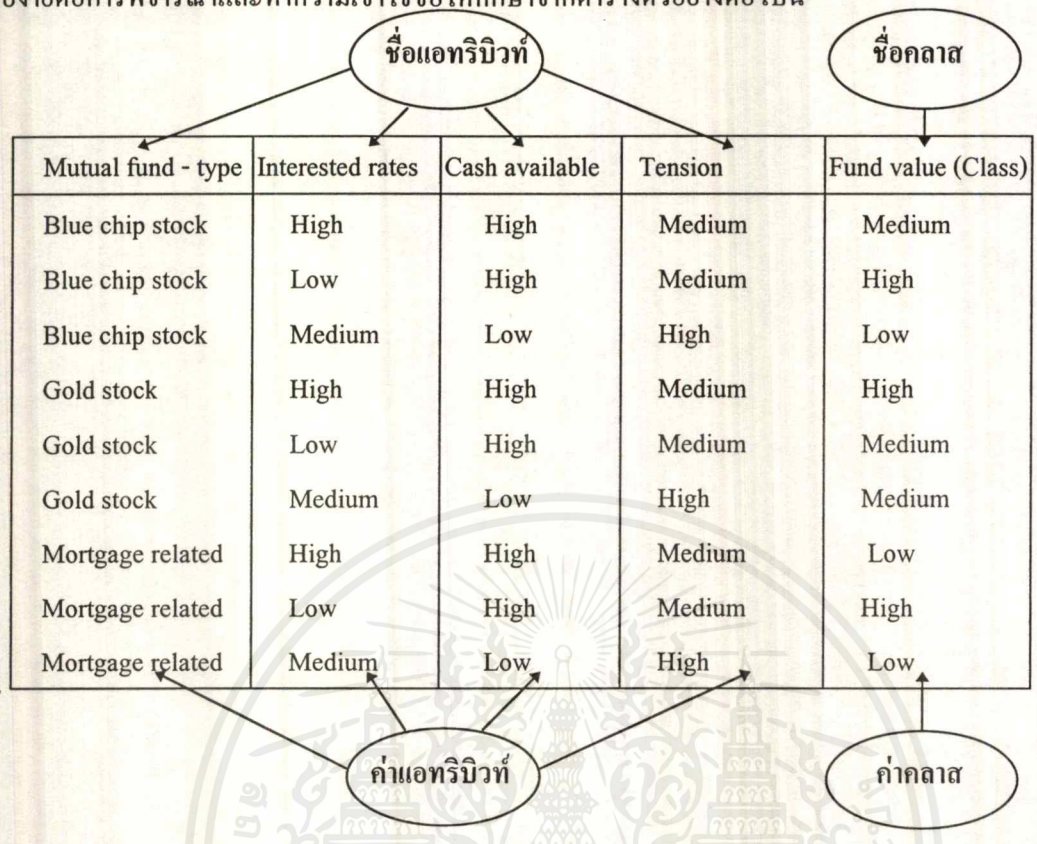
3. การเตรียมข้อมูลหรือกลุ่มตัวอย่าง (TRAINNING SET) ก่อนนำเข้าสู่โปรแกรม LESS

ผู้ใช้ระบบจะต้องทำการหาข้อมูลหรือกลุ่มตัวอย่างที่ต้องการให้ได้เป็นจำนวนมากพอและเหมาะสม ก่อนที่จะนำมาใช้งานโปรแกรม LESS กลุ่มตัวอย่างเหล่านั้นจะต้องถูกนำมาจัดระเบียบเสียใหม่เพื่อทำการกำหนดรูปแบบและลำดับก่อนหลังในการนำเข้าสู่ระบบ โดยจัดเก็บในรูปแบบของตาราง

ในตัวอย่างแต่ละตัวอย่างที่จะใช้ในโปรแกรมนี้ จะต้องประกอบด้วยชนิดหรือประเภทของตัวอย่าง (Class) ที่เราต้องการจะวินิจฉัย และแอทริบิวท์ (Attributes) ซึ่งเป็นค่าที่เราจะใช้ในการวินิจฉัยหาคลาส โดยในแต่ละคอลัมน์จะใช้สำหรับการเก็บค่าของแอทริบิวท์ต่าง ๆ อันจะต้องมีคุณสมบัติสำคัญที่ต้องมีผลต่อการแยกประเภท เช่น ลักษณะ อาการ รูปร่าง อายุ ความสามารถต่างๆ เป็นต้น

โดยในแถวแรกของแต่ละคอลัมน์ จะต้องทำการเก็บชื่อแอทริบิวท์นั้นๆไว้ด้วย และแถวต่อๆไปจึงจะทำการเก็บค่าแอทริบิวท์จริงๆ ของชื่อแอทริบิวท์นั้นๆ ยกเว้นในคอลัมน์สุดท้ายซึ่งจะเป็นคอลัมน์ของ คลาส

ซึ่งแถวแรกของคอลัมน์นี้จะถูกใช้เพื่อเก็บชื่อคลาส ส่วนในแถวต่อๆมาจะใช้เก็บค่าประเภทต่าง ๆ ของคลาส เพื่อง่ายต่อการพิจารณาและทำความเข้าใจขอให้ศึกษาจากตารางตัวอย่างต่อไปนี้



Mutual fund - type	Interested rates	Cash available	Tension	Fund value (Class)
Blue chip stock	High	High	Medium	Medium
Blue chip stock	Low	High	Medium	High
Blue chip stock	Medium	Low	High	Low
Gold stock	High	High	Medium	High
Gold stock	Low	High	Medium	Medium
Gold stock	Medium	Low	High	Medium
Mortgage related	High	High	Medium	Low
Mortgage related	Low	High	Medium	High
Mortgage related	Medium	Low	High	Low

จากตารางจะเห็นได้ว่า ทุกคอลัมน์จะต้องมีชื่อของคอลัมน์อยู่ที่แถวแรกสุดเสมอ แต่ต่อจากนี้ไปในโปรแกรม LESS จะเริ่มนับแถวที่ 1 ที่แถวของค่าแอทริบิวท์ตัวแรก (ในที่นี้คือแถวที่สองนั่นเอง) และในคอลัมน์สุดท้าย ก็จะประกอบไปด้วยชื่อของคลาส และค่าของคลาสในแถวถัดมา วิธีการอ้างถึงข้อมูลภายในตารางจะไม่มีการคำนึงว่า ข้อมูลตัวนั้นเป็น แอทริบิวท์หรือเป็นคลาส แต่คำนึงถึงข้อมูลนั้นเป็นเพียงข้อมูลตัวหนึ่งเท่านั้น การอ้างถึงจึงกระทำโดยการอ้างถึงตำแหน่งของคอลัมน์และแถวของมัน ยกตัวอย่างเช่น ข้อมูลในคอลัมน์ Tension แถวที่3 คือค่า HIGHT ซึ่งเป็นค่าของแอทริบิวท์ ขณะเดียวกัน ข้อมูลในคอลัมน์ Fund value แถวที่ 4 คือค่า HIGHT ซึ่งค่า HIGHT ตัวนี้กลับเป็นค่าของคลาส ซึ่งจะเห็นได้ว่า ไม่มีความแตกต่างใดในการอ้างถึงตำแหน่งของข้อมูลแต่ละตัว ไม่ว่ามันจะเป็นค่าแอทริบิวท์หรือเป็นค่าคลาส โดยการอ้างจะอาศัยความสัมพันธ์ระหว่างคอลัมน์และแถวที่ข้อมูลนั้นอยู่ แต่สิ่งสำคัญอยู่ที่ว่า ภายในตารางไม่สามารถที่จะสลับตำแหน่งของคอลัมน์ระหว่าง คอลัมน์ของแอทริบิวท์กับ คอลัมน์ของคลาสได้ เช่นหากภายในตารางนำค่าของแอทริบิวท์ทั้งคอลัมน์มาใส่ไว้ในตารางตรงคอลัมน์สุดท้าย โปรแกรมก็จะเข้าใจในทันทีว่า ค่าของแอทริบิวท์นั้นเป็นค่าของคลาสไป ดังนั้นก่อนทำการนำข้อมูลเขาสู่โปรแกรม ต้องมีการจัดทำให้อยู่ในรูปแบบที่เหมาะสมและถูกต้องเสียก่อน ค่าที่ได้รับจากโปรแกรมจึงจะเป็นค่าที่ถูกต้อง

4. การใส่ข้อมูลหรือกลุ่มตัวอย่าง เข้าสู่โปรแกรม LESS

ซึ่งประกอบด้วย 3 วิธีการดังต่อไปนี้

4.1 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทางส่วนการรับข้อมูลของโปรแกรม LESS

โปรแกรมLESSได้จัดเตรียมส่วนของการรับข้อมูลจากผู้ใช้ระบบไว้โดยตรงเพื่อความสะดวกผ่านทางหน้าจอ โดยผู้ใช้สามารถเรียกใช้โปรแกรมส่วนนี้ได้โดยการรียกใช้โดยตรงผ่านหน้าจอที่โหมดการทำงานหน้าจอ SMALLTALK \V TRANSCRIPT โดยการกระทำดังขั้นตอนต่อไปนี้

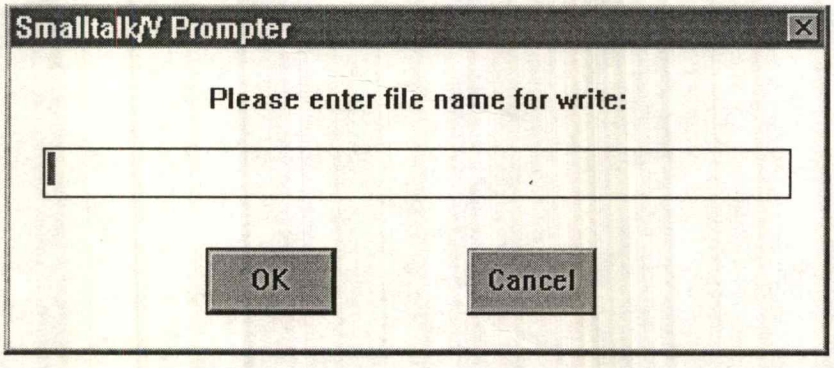
4.1.1 พิมพ์คำสั่ง **G := LEX new .**

โดย G คือตัวแปรอะไรก็ได้ที่ถูกกำหนดขึ้นมาเพื่อใช้ในโปรแกรม

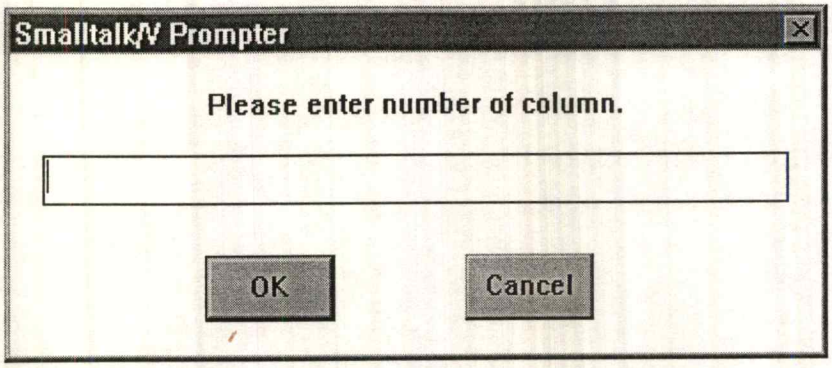
จากนั้นจึงทำการ RUN คำสั่งโดยการลากแถบสว่างให้คลุมทั้งคำสั่งกดเมาส์ขวาแล้วเลือกที่เมนู **DO IT**

4.1.2 พิมพ์คำสั่ง **G menu.** เพื่อที่จะทำการเรียกใช้เมนูของระบบที่สร้างไว้ ทำการ RUN คำสั่งนี้อีกครั้ง ซึ่งโปรแกรมจะแสดงหน้าจอดังรูป

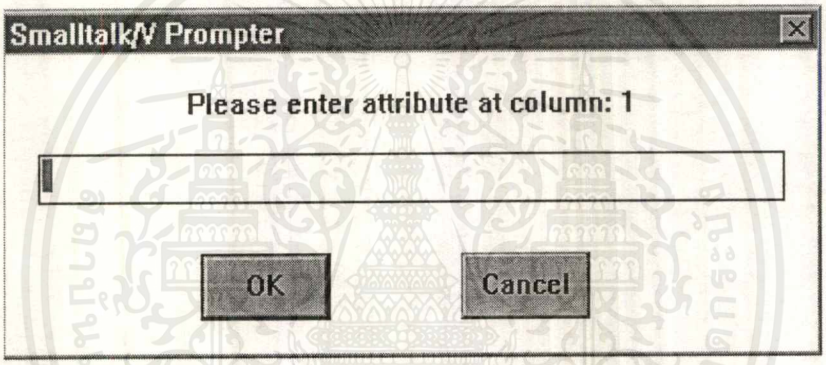
4.1.3พิมพ์เลข 1 เพื่อเลือกการ ใส่ข้อมูลผ่านส่วน Input training set กด enter หรือคลิกทีปุ่ม OK โปรแกรมจะปรากฏหน้าจอดังรูป



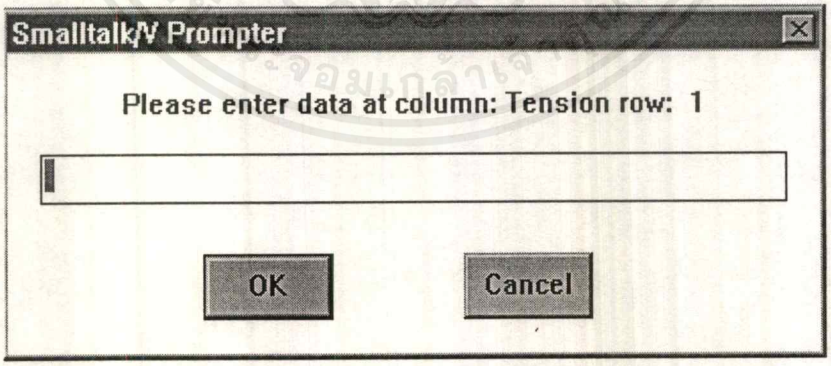
4.1.4 ใส่ชื่อไฟล์ที่ต้องการสำหรับบันทึกข้อมูล กด enter หรือคลิกที่ปุ่ม OK โปรแกรมจะปรากฏหน้าจอ ดังรูป



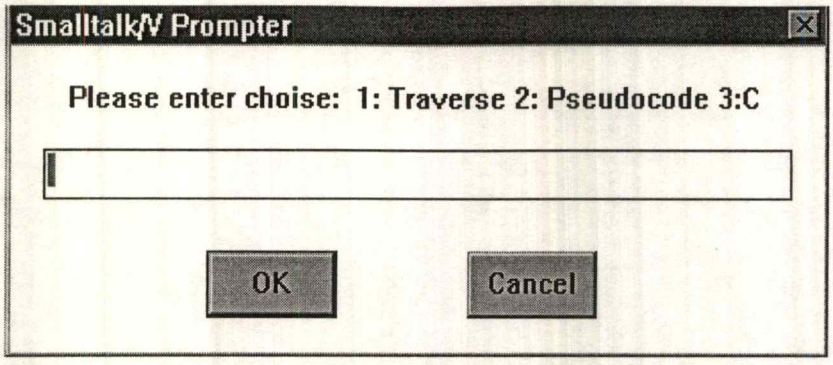
4.1.5 ทำการใส่ค่าคอลัมน์ของตาราง TRAINING SET กด enter หรือคลิกที่ปุ่ม OK ทำเช่นเดียวกันนี้กับค่าของแถวเมื่อใส่จำนวนแถวแล้วโปรแกรมจะแสดงหน้าจอดังรูป



4.1.6 ทำการใส่ค่าของชื่อของแอทริบิวท์แต่ละตัวจนครบทุกตัวแล้วโปรแกรมจะแสดงหน้าจอดังนี้



4.1.7 ใส่ค่าของแอทริบิวท์ที่ตำแหน่งต่างๆตาม คอลัมน์และแถวที่กำหนดให้ ในที่นี้เช่นใส่ในคอลัมน์ Tension แถวที่ 1 ทำเช่นนี้จนครบแอทริบิวท์กับการใส่ค่าแอทริบิวท์ทุกๆ ตัวจนครบ และโปรแกรมจะแสดงหน้าจอดังต่อไปนี้



4.1.8 เลือก 1,2 หรือ 3 เพื่อทำงานตามที่ต้องการกับข้อมูลชุดนี้ต่อไป (รายละเอียดการใช้งานในส่วนอื่นๆ อยู่ในหัวข้อถัดไป)

4.2 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทาง TEXT FILE ธรรมดา

เพื่อความสะดวกของผู้ใช้ระบบในการใส่กลุ่มตัวอย่างหรือข้อมูลเข้าสู่ระบบ โปรแกรม LESS จึงเอื้ออำนวยและเปิดโอกาสให้ผู้ใช้โปรแกรมสามารถ ใส่ข้อมูลเข้าโดยผ่าน TEXT FILE โดยตรง แต่ต้องมีรูปแบบตรงตามที่กำหนดไว้ เพื่อที่โปรแกรมจะสามารถนำข้อมูลนั้นไปใช้ได้อย่างถูกต้อง โดยมีรูปแบบการเขียนข้อมูลลงใน TEXT FILE เป็นค่าต่างๆที่ตรงตามรูปแบบ และต้องมีการกด enter เพื่อทำการขึ้นบรรทัดใหม่ทุกครั้งที่พิมพ์ข้อมูลแต่ละตัวเสมอ โดยแต่ละบรรทัดจะต้องพิมพ์สิ่งต่อไปนี้

บรรทัดที่ 1 จะเป็นค่าของจำนวน Column ของตารางของกลุ่มตัวอย่าง

บรรทัดที่ 2 จะเป็นค่าของจำนวน Row ของตารางของกลุ่มตัวอย่าง

บรรทัดที่ 3 เป็นต้นไปจะเป็นค่าของชื่อแอทริบิวท์แต่ละตัวหรือคือค่าชื่อคลาสแต่ละคลาส บรรทัดละ 1 ชื่อจนหมด

บรรทัดถัดจากนั้นไป จะเป็นค่าของแอทริบิวท์บรรทัดละ 1 ค่า โดยเริ่มเรียงจาก แอทริบิวท์ ในคอลัมน์ที่ 1 แถวที่ 1 บรรทัดถัดมาก็จะเป็นค่าของแอทริบิวท์ในคอลัมน์ที่ 1 แถวที่ 2 เรียงเช่นนี้ไปเรื่อยๆจนครบทุกค่า ดังเช่นตัวอย่าง TEXT FILE ต่อไปนี้

5 ↵
9 ↵
Mutual fund - type ↵
Interested rates ↵
Cash available ↵
Tension ↵
Fund value (Class) ↵
Blue chip stock ↵
Blue chip stock ↵
Blue chip stock ↵
....

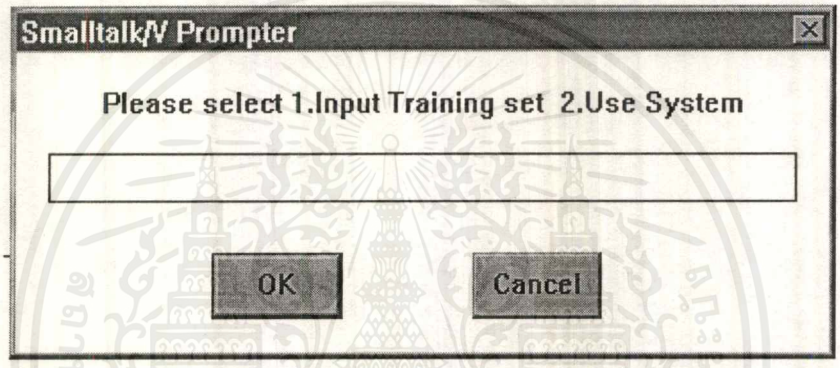
....

แล้วทำการบันทึกไฟล์ข้อมูลเก็บไว้เพื่อนำมาใช้ในโอกาสต่อๆไป

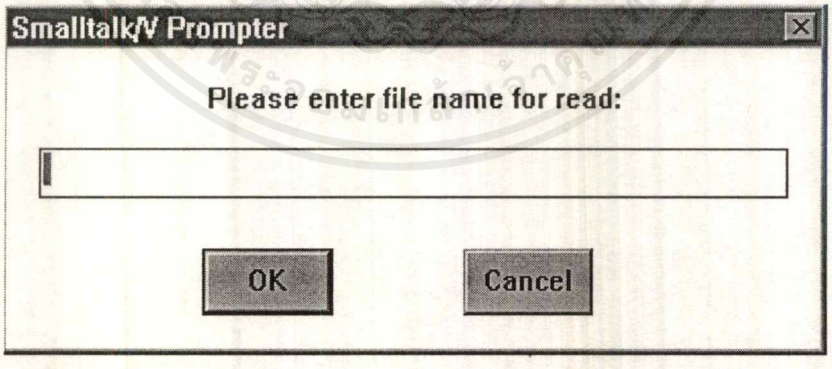
4.3 การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยการดึงมาจาก แฟ้มข้อมูลเก่า

ในกรณีที่ผู้ใช้ต้องการใช้ชุดฝึกหัดชุดเดิมที่มีอยู่แล้ว หรือในกรณีที่ผู้ใช้ ใช้การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทาง TEXT FILE ธรรมดา การจะใช้ข้อมูลนั้นกับโปรแกรม LESS ก็สามารถทำได้อย่างง่ายดาย โดยวิธีการดังต่อไปนี้

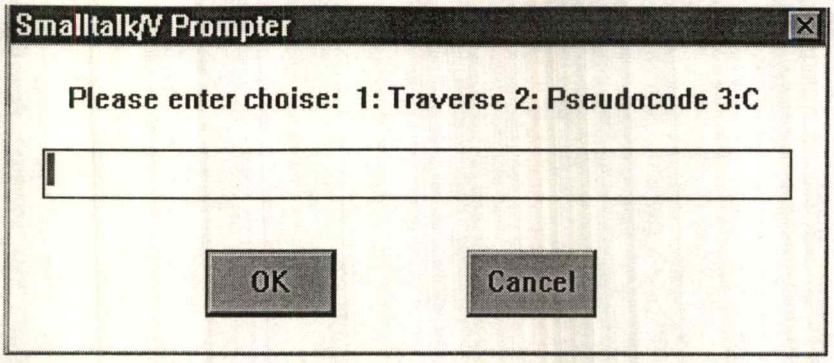
4.3.1 เรียกใช้โหมดหน้าจอเมนูที่หน้าจอการทำงานของ SMALLTALK/V TRANSCRIPT เช่น เดียวกันกับในหัวข้อ 4.1ในเรื่องของ การใส่ข้อมูลเข้าสู่โปรแกรม LESS โดยผ่านทางส่วนการรับข้อมูลของโปรแกรม LESSโดยกระทำตามขั้นตอนต่างเช่นเดียวกันกับข้างต้น จนโปรแกรมแสดงหน้าจอดังต่อไปนี้



4.3.2 พิมพ์เลข 2 เพื่อเลือกการใช้ระบบโดยการใช้ค่าจากไฟล์ข้อมูลที่มีอยู่แล้วกด enter หรือคลิกที่ปุ่ม OK โปรแกรมจะปรากฏหน้าจอดังรูป



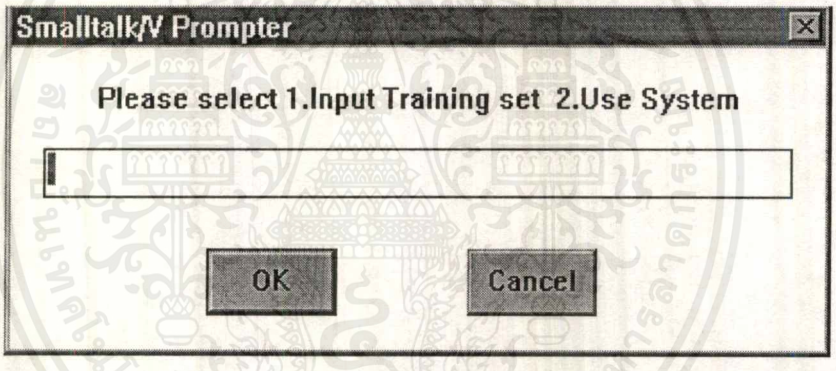
4.3.3ใส่ชื่อไฟล์ที่ต้องการสำหรับการอ่านข้อมูลที่ได้บันทึกไว้ขึ้นมา กด enter หรือคลิกทีปุ่ม OK โปรแกรมจะปรากฏหน้าจอ ดังรูป



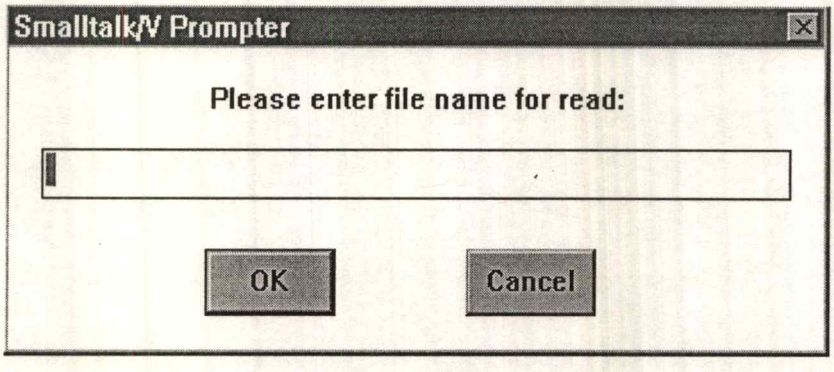
4.3.4 เลือก 1,2 หรือ 3 เพื่อทำงานตามที่ต้องการกับข้อมูลชุดนี้ต่อไป(รายละเอียดการใช้งานในส่วนอื่นๆ อยู่ในหัวข้อด้านหลัง)

5.วิธีการใช้โปรแกรม LESS ในการหาข้อสรุปและตอบคำถาม

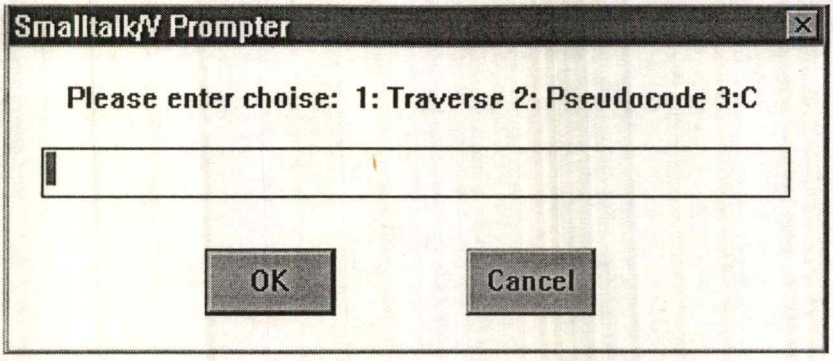
5.1 เข้าสู่การทำงานของโปรแกรม LESS ที่ SMALLTALK /V TRAINSCRIPT แล้ว ทำการเรียกใช้เมนูของโปรแกรมดังข้างต้น โปรแกรมจะแสดงหน้าจอดังรูป



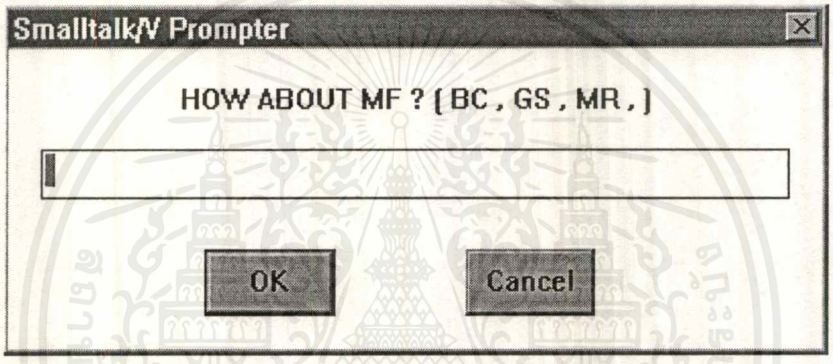
5.2 พิมพ์ 2 เพื่อทำการเข้าสู่ โหมดการใช้งานระบบ กด enter หรือคลิกที่ปุ่ม OK โปรแกรม จะปรากฏหน้าจอดังรูป



5.3 ใส่ชื่อไฟล์ข้อมูลที่เก็บชุดตัวอย่างพร้อมทั้งใส่ path ให้ถูกต้อง กด enter หรือ คลิกปุ่ม OK โปรแกรมจะปรากฏหน้าจอดังรูป



5.4 พิมพ์ 1 เพื่อทำการเข้าสู่ โหมดการใช้งาน Traverse tree กด enter หรือคลิกที่ปุ่ม OK โปรแกรมจะปรากฏหน้าจอดังรูป

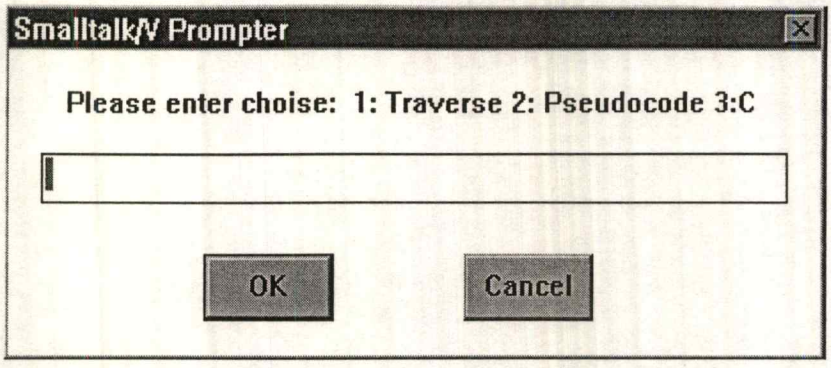


5.5 ตอบคำถามแก่ระบบ เพื่อระบบจะได้นำไปใช้ในการหาข้อสรุป กด enter หรือคลิกที่ปุ่ม OK โปรแกรม โปรแกรมจะขึ้นคำถามใหม่มาเพื่อถามผู้ใช้ไปเรื่อย ๆ จนมีข้อมูลเพียงพอที่จะสามารถหาข้อสรุปได้

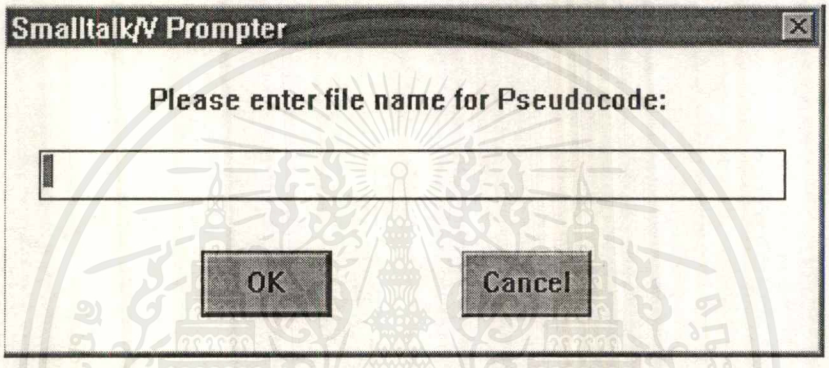
5.6 โปรแกรมจะกลับเข้าสู่ โหมดการทำงานที่หน้าจอ SMALLTALK /V TRAINSCRIPT เพื่อตอบคำถามโดยการระบุ คลาสนั้นๆ

6.วิธีการใช้โปรแกรม LESS ในการสร้าง PSUDOCODE

6.1 เข้าสู่การทำงานของโปรแกรม LESS เรียกใช้การทำงานของเมนูและกระทำตามขั้น ตอนต่างๆ เหมือนกับ วิธีการใช้โปรแกรม LESS ในการหาข้อสรุปและตอบคำถามในแบบ ข้อ5 จนโปรแกรมแสดงหน้าจอดังรูป



6.2 พิมพ์ 2 เพื่อทำการเข้าสู่ โหมดการใช้งานการสร้าง Pseudo code กด enter หรือคลิกที่ปุ่ม OK โปรแกรม จะปรากฏหน้าจอดังรูป

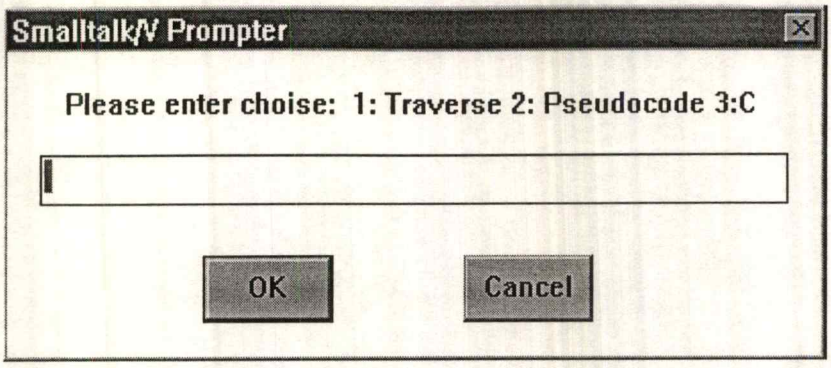


6.3 ใส่ชื่อไฟล์ข้อมูลที่จะใช้เก็บ Pseudo code พร้อมทั้งใส่ path ให้ถูกต้อง กด enter หรือ คลิกปุ่ม OK

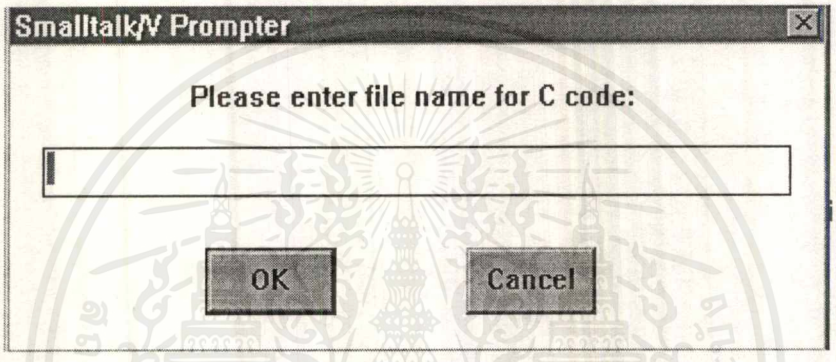
6.4 โปรแกรมจะกลับเข้าสู่ โหมดการทำงานที่หน้าจอ SMALLTALK /V TRAINSCRIPT และแสดงข้อความ 'Pseudo code is in file ชื่อไฟล์' อันแสดงให้เห็นว่า Pseudo code ได้ถูกสร้างและเก็บไว้ในไฟล์ชื่อนั้นๆเรียบร้อยแล้ว

7.วิธีการใช้โปรแกรม LESS ในการสร้าง CODE ภาษา C

7.1 เข้าสู่การทำงานของโปรแกรม LESS เรียกใช้การทำงานของเมนูและกระทำตามขั้น ตอนต่างๆ เหมือนกับ วิธีการใช้โปรแกรม LESS ในการหาข้อสรุปและตอบคำถามในแบบ ข้อ5 จนโปรแกรมแสดงหน้าจอดังรูป



7.2 พิมพ์ 3 เพื่อทำการเข้าสู่ โหมดการใช้งานการสร้าง code ภาษา C กด enter หรือคลิกที่ปุ่ม OK โปรแกรมจะปรากฏหน้าจอดังรูป



7.3 ใส่ชื่อไฟล์ข้อมูลที่จะใช้เก็บ code ภาษา C พร้อมทั้งใส่ path ให้ถูกต้อง กด enter หรือ คลิกปุ่ม OK

7.4 โปรแกรมจะกลับเข้าสู่ โหมดการทำงานที่หน้าจอ SMALLTALK /V TRAINSCRIPT และแสดงข้อความ 'C language code is in file ชื่อไฟล์' อันแสดงให้เห็นว่า code ภาษา C ได้ถูกสร้างและเก็บไว้ในไฟล์ชื่อนั้นๆเรียบร้อยแล้ว

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ก็เพราะได้รับความช่วยเหลือและการอนุเคราะห์จากหลาย ๆ ท่านด้วยกัน ซึ่งผู้จัดทำรู้สึกทราบซึ้ง และขอขอบพระคุณมา ณ.ที่นี้

ขอขอบพระคุณ รศ.ดร. ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาที่ได้ดูแลเอาใจใส่ ตลอดระยะเวลาการทำปริญญานิพนธ์นี้ ให้คำแนะนำในการพัฒนาโปรแกรม ชี้แนะแนวทางการแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการทำปริญญานิพนธ์ พร้อมทั้งยังให้คำชี้แนะในการทำปริญญานิพนธ์ฉบับนี้มาโดยตลอด

ขอขอบพระคุณ อาจารย์พิทักษ์ ธรรมวาริน อาจารย์ภาควิชาเทคนิคอุตสาหกรรม ที่ให้ความอนุเคราะห์ในการให้คำปรึกษาถึงปัญหาต่าง ๆ ช่วยให้คำชี้แนะในการทำปริญญานิพนธ์ฉบับนี้ ช่วยจัดหาโปรแกรมและหนังสือที่ใช้ประกอบการทำปริญญานิพนธ์ รวมทั้งยังให้ความอนุเคราะห์ในเรื่องอุปกรณ์การทำปริญญานิพนธ์ฉบับนี้อีกด้วย

ขอขอบพระคุณ อาจารย์วิศรุต ศรีรัตนะ อาจารย์ภาควิชาเทคโนโลยีการวัดคุม ที่ให้ความอนุเคราะห์อุปกรณ์การทำปริญญานิพนธ์ ให้คำปรึกษา และอำนวยความสะดวกในการทำปริญญานิพนธ์ฉบับนี้มาโดยตลอด

ขอขอบคุณ พี่ธีรวัฒน์ ที่ให้คำปรึกษาในการออกแบบและพัฒนาโปรแกรม รวมทั้งยังให้ความอนุเคราะห์หนังสือที่ใช้ประกอบการทำปริญญานิพนธ์ฉบับนี้อีกด้วย

ขอขอบคุณ เพื่อน ๆ พี่ ๆ และน้อง ๆ ที่คอยเอื้อเฟื้อ คอยช่วยเหลือในด้านต่าง ๆ และคอยเป็นกำลังใจให้กับผู้จัดทำมาโดยตลอด

สุดท้ายนี้ ขอขอบพระคุณทุกท่านที่มีส่วนร่วมในการทำปริญญานิพนธ์ฉบับนี้ ทั้งที่เอ่ยชื่อและไม่ได้เอ่ยชื่อก็ตาม ขอขอบพระคุณอีกครั้ง

ผู้จัดทำ

บรรณานุกรม

[1] Lewis J.Pinson, Richard S. Wiener, "Smalltalk/V Tutorial and Programming Handbook" digitalk inc.

[2] University of Colorado at Colorado Springs, "An Introduction to Object Oriented Programming and Smalltalk " , Addison-Wesley Publishing Company

[3] Dan W. Patterson ,University of Texas, El Paso, "Introduction to Aritificial Intelligence and Expert Systems", Prentice-Hall International,Inc.

[4] เกษมสันต์ พานิชการ, "C^{++} และหลักการของ OOP ", SE-EDUCATION Public Company Limited