# บทที่ 8

==========

# การทำโปรแกรมเชิงวัตถุ
# (Object-oriented programming)

## วัตถุประสงค์

1) เพื่อให้ผู้เรียนเข้าใจวัตถุประสงค์และประโยชน์ของการออกแบบและการทำโปรแกรมเชิงวัตถุ
2) เพื่อให้ผู้เรียนรู้จักหลักการพื้นฐานของการทำโปรแกรมเชิงวัตถุ ซึ่งประกอบด้วยการรวมข้อมูลเป็นหน่วยเดียว (data encapsulation), การถ่ายทอด (inheritance) และการมีหลายรูป (polymorphism)
3) เพื่อให้ผู้เรียนได้รู้ถึงประโยชน์ของการรวมข้อมูลเป็นหน่วยเดียวและรู้จักความแตกต่างระหว่างคลาสและอ็อบเจ็ค
4) เพื่อให้ผู้เรียนเข้าใจวิธีการถ่ายทอดคุณสมบัติระหว่างคลาสและได้ตระหนักถึงประโยชน์ของการถ่ายทอดที่มีต่อกระบวนการพัฒนาซอฟต์แวร์ในด้านการนำสิ่งที่มีอยู่มาใช้ประโยชน์ใหม่ (software reuse)
5) เพื่อให้ผู้เรียนได้รู้จักคุณสมบัติการมีหลายรูปและได้เรียนรู้วิธีการใช้ประโยชน์จากคุณสมบัตินี้

การโปรแกรมเชิงวัตถุเป็นการเปลี่ยนมุมมองการทำโปรแกรมจากเดิมที่ยึดข้อมูลเป็นหลัก มาเป็นการยึดวัตถุเป็นหลัก การทำโปรแกรมแบบเดิมที่เป็นการทำโปรแกรมเชิงคำสั่งใช้วิธีตั้งตัวแปรขึ้นมาเพื่อใช้เก็บข้อมูลแล้วใช้คำสั่งประเภทต่างๆ เปลี่ยนแปลงค่าของข้อมูลไปตามลำดับที่ต้องการ การทำโปรแกรมเชิงวัตถุต่างไปจากเดิมตรงที่ขยายมุมมองจากข้อมูลที่เป็นหน่วยย่อยให้เป็นวัตถุซึ่งเป็นหน่วยใหญ่ขึ้นโดยวัตถุจะมีทั้งข้อมูลและการกระทำต่างๆ ที่สามารถกระทำกับข้อมูลนั้นได้ ภาษาคอมพิวเตอร์ภาษาแรกที่ใช้แนวคิดเชิงวัตถุได้แก่ ภาษา Simula 67 และต่อมามีภาษาเชิงวัตถุอื่นๆ เกิดขึ้นอีกเป็นจำนวนมากได้แก่ ภาษา Smalltalk, Eiffel, C++, Java ภาษาเชิงวัตถุที่แพร่หลายมากที่สุดในปัจจุบันคือ ภาษา Java ซึ่งยังไม่ใช่ภาษาเชิงวัตถุที่แท้จริง (pure object-oriented languages) เช่น ภาษา Smalltalk จะพิจารณาทุกส่วนประกอบในโปรแกรมเป็นวัตถุ แม้แต่โครงสร้างการสั่งงานแบบมีเงื่อนไขและการสั่งวนรอบ

วิธีการทำโปรแกรมเชิงวัตถุอาจไม่จำเป็นต้องใช้ภาษาเชิงวัตถุ เช่น อาจจะทำโปรแกรมด้วยภาษา Ada แทนที่จะใช้ภาษา Java เป็นต้น แต่โปรแกรมนั้นจะต้องมีลักษณะของการทำโปรแกรมเชิงวัตถุ ซึ่งได้แก่ การรวมข้อมูลและปฏิบัติการกับข้อมูลเป็นหน่วยเดียว และในกรณีที่โปรแกรมมีขนาดใหญ่ และซับซ้อนขึ้นจะมีลักษณะอื่นเพิ่มเติม เช่น การถ่ายทอด, การมีหลายรูป

## 8.1 หลักการทำโปรแกรมเชิงวัตถุ

(Principles of object-oriented programming)

การทำโปรแกรมเชิงคำสั่งหรือเชิงกระบวนงาน จะออกแบบโปรแกรมโดยการพิจารณาจากงานที่โปรแกรมต้องกระทำ จากนั้นสร้างโพรซีเจอร์หรือฟังก์ชันขึ้นมาเพื่อให้ทำแต่ละงานย่อย ดังนั้นโปรแกรมจะประกอบด้วยโพรซีเจอร์และโครงสร้างข้อมูลต่างๆ แต่การทำโปรแกรมเชิงวัตถุจะมีมุมมองของโปรแกรมแตกต่างออกไปโดยพิจารณาว่าโปรแกรมประกอบด้วยวัตถุหรืออ็อบเจ็ค และโปรแกรมทำงานด้วยการส่งข้อความให้วัตถุนั้นสามารถกระทำได้ ตัวอย่างการสร้างโครงสร้างข้อมูลสแตกในภาษา C (รูปที่ 8.1) และภาษา C++ (รูปที่ 8.2) ต่อไปนี้เปรียบเทียบข้อแตกต่างระหว่างการทำโปรแกรมเชิงคำสั่ง และการทำโปรแกรมเชิงวัตถุ

```
/* file stack.h */
/* header file exporting declarations to clients*/

typedef struct stack {
        int elements[100];      /* stack of 100 integers */
        int top;                /* number of element */
};
void push(stack, int);
int pop(stack);
/*---------------- end of file -----------------*/


/* file stack.c */
/* implementation of stack operations */

#include "stack.h"
void push(stack s, int i) {
        s.element[s.top++]=i
};
int pop(stack s){
        return s.elements[--s.top];
};
/*---------------- end of file --------------------*/


/* file main.c */
/* a client of stack */

#include "stack.h"
void main() {
        stack s1, s2;           /* declare two stacks */
        s1.top=0; s2.top=0      /* initialize them */
        int i;
        push(s1, 5);            /* push 5 on first stack */
        push(s2, 6);            /* push 6 on second stack */
        ...
        i=pop(s1);              /* pop first stack */
        ...
}
/*---------------- end of file --------------------*/
```

รูปที่ 8.1 การสร้างและการเรียกใช้โครงสร้างข้อมูลสแตกในภาษา C

```
// file stack.h
// header file containing declarations exported to clients

class stack {
        public :
                stack();
                void push(int);
                int pop();
        private :
                int elements[100];      // stack represented as array
                int top=0;              // number of stored elements
};
// ------------------- end of file ---------------------

// file stack.cpp
// the implementation of stack member functions

#include "stack.h"
void stack::push(int i) {
        elements[top++]=I;
};
int stack::pop() {
        return elements[--top]
};
// ------------------ end of file -----------------------

// file main.cpp
// a client of stack

#include "stack.h"
main() {
        stack s1, s2;   // declare two stack
        int i;
        s1.push(5);     // push 5 on first stack
        s2.push(6);     // push 6 on second stack
        ...
        i=s1.pop();     // pop first stack
        ...
}
// ------------------- end of file ----------------------
```

รูปที่ 8.2 การสร้างและการเรียกใช้โครงสร้างข้อมูลสแตกในภาษา C++

จากตัวอย่างการสร้างสแตกด้วยภาษา C และภาษา C++ ดังกล่าวข้างต้นมีข้อแตกต่างที่ชัดเจนสองประการ คือ วิธีการใช้งานสแตก และการซ่อนรายละเอียดภายในสแตก

วิธีการใช้งานสแตกในการทำโปรแกรมเชิงคำสั่งจะเน้นที่ฟังก์ชันการทำงาน โดยให้วัตถุ (สแตก) และข้อมูล เป็นส่วนประกอบหนึ่งของฟังก์ชัน เช่น ถ้าต้องการพุชข้อมูลลงในสแตกคำสั่งที่ใช้จะเป็นดังนี้

*push(s1, 5);*

แต่ในการทำโปรแกรมเชิงวัตถุเน้นที่วัตถุก่อนแล้วจึงระบุการกระทำที่ต้องการกระทำกับวัตถุนั้น ดังนั้นการพุชข้อมูลสแตกจึงมีวิธีการเขียนคำสั่งได้ดังต่อไปนี้

*s1.push(5);*

การทำโปรแกรมเชิงคำสั่งตามตัวอย่างในรูปที่ 8.1 ไม่มีการซ่อนรายละเอียดภายในของโครงสร้างข้อมูลสแตก โครงสร้างภายในของสแตกประกอบด้วย elements ที่เป็นอาร์เรย์ขนาด 100 ทำหน้าที่เก็บข้อมูล และ top ที่เป็นตัวแปรเดี่ยวทำหน้าที่นับจำนวนข้อมูลที่เก็บอยู่ในสแตก ฟังก์ชัน main ที่เรียกใช้สแตกสามารถใช้คำสั่งกำหนดค่า top ได้ตามต้องการ เช่น

*s1.top=0;*

แต่ในการทำโปรแกรมเชิงวัตถุตัวแปร top และ elements จะถูกซ่อนไว้ด้วยข้อความ private ฟังก์ชัน main สามารถทำได้เพียงใช้คำสั่งสแตก, พุชข้อมูลลงสแตกและพ็อพข้อมูลออกจากสแตก

การซ่อนรายละเอียด (information hiding) เป็นวัตถุประสงค์หลักของการสร้างข้อมูลชนิดข้อมูลนามธรรม (abstract data type) เพื่อแยกส่วนการใช้งานข้อมูลออกจากส่วนรายละเอียดภายในของข้อมูล การทำโปรแกรมเชิงวัตถุขยายขอบเขตของข้อมูลให้เป็นมากกว่าค่าที่เก็บไว้เพื่อใช้งานโดยให้ข้อมูลมีทั้งส่วนที่เก็บค่า และส่วนที่เป็นการกระทำกับค่าที่เก็บไว้ ส่วนเก็บค่าและส่วนกระทำการกับค่าจะถูกรวมเป็นหน่วยเดียว เรียกว่า *การรวมข้อมูลเป็นหน่วยเดียว* (data encapsulation) นอกจากนี้ยังมีการเพิ่มเติมคุณสมบัติให้สามารถสร้างชนิดข้อมูลใหม่ต่อยอดจากชนิดข้อมูลเดิมที่มีอยู่ เรียกว่า *คุณสมบัติการถ่ายทอด* (inheritance)

คุณสมบัติการรวมข้อมูลเป็นหน่วยเดียว และคุณสมบัติการถ่ายทอดเป็นหลักการพื้นฐานของการทำโปรแกรมเชิงวัตถุ นอกจากคุณสมบัติหลักสองประการนี้แล้วภาษาสำหรับการทำโปรแกรมเชิงวัตถุยังต้องมีโครงสร้างที่สนับสนุน*การมีหลายรูป* (polymorphism) ของฟังก์ชันหรือเมธ็อด และ*การยึดเหนี่ยวแบบพลวัต* (dynamic binding) เพื่อเชื่อมโยงรูปแบบที่ถูกต้องให้กับฟังก์ชันขณะรันโปรแกรม

## 8.2 การรวมข้อมูลเป็นหน่วยเดียว
(Data encapsulation)

คุณสมบัติการรวมข้อมูลเป็นหน่วยเดียว หมายถึงการรวมส่วนโครงสร้างภายในของข้อมูลและส่วนปฏิบัติการกับข้อมูลไว้เป็นหน่วยเดียว โดยประกาศให้โปรแกรมภายนอกรู้เพียงว่าจะเรียกใช้ข้อมูลนี้ได้อย่างไรและจะปฏิบัติการใดกับข้อมูลนี้ได้บ้าง โดยไม่ต้องประกาศรายละเอียดว่าโครงสร้างภายในของข้อมูลประกอบด้วยอะไรบ้างและปฏิบัติการแต่ละอย่างที่กระทำกับข้อมูลมีขั้นตอนการทำงานอย่างไร การซ่อนรายละเอียดภายในไว้จะช่วยให้การเปลี่ยนแปลงใดๆ กับโครงสร้างภายในข้อมูลไม่เกิดผลกระทบกับโปรแกรมอื่นที่เรียกใช้ข้อมูลนั้น

ภาษาสำหรับการทำโปรแกรมเชิงวัตถุโดยทั่วไปแล้วจะใช้คลาสทำหน้าที่รวมข้อมูลเป็นหน่วยเดียว ดังตัวอย่างในรูปที่ 8.3 คลาสที่ถูกตั้งชื่อว่า stack ทำหน้าที่รวมส่วนโครงสร้างภายในของ stack ที่

ประกอบด้วยอาร์เรย์ชื่อว่า elements และตัวแปรชื่อว่า top ไว้กับส่วนปฏิบัติการ ซึ่งประกอบด้วยฟังก์ชัน push และ pop ทำหน้าที่เพิ่มและลบข้อมูลใน stack ตามลำดับ

```
class stack {
        public :
                void push(int v);          // operations; visible to users
                int pop();
        private :
                int elements[100];         // variables; invisible to users
                int top=0;
};
```

รูปที่ 8.3 การใช้คลาสทำหน้าที่รวมข้อมูลเป็นหน่วยเดียว

การใช้คำขยาย "private " ทำให้ส่วนโครงสร้างภายในที่ทำหน้าที่เก็บค่าต่างๆ ของข้อมูลถูกซ่อนไว้เฉพาะภายในคลาสเท่านั้น ส่วนที่ประกาศว่าเป็น "public" จะเป็นเฉพาะโพรโทไทพ์ของฟังก์ชัน เพื่อใช้สื่อสารกับภายนอกว่าสามารถปฏิบัติการ push และ pop กับโครงสร้างข้อมูลสแตกได้ โดยถ้ากระทำการพุชจะต้องให้ค่าตัวเลขที่ต้องการพุชลงสแตก ดังนั้นโปรแกรมภายนอกสามารถใช้งานสแตกได้ด้วยคำสั่งต่อไปนี้

```
stack s;
s.push(25);
```

จะเห็นได้ว่าในคำสั่งข้างต้น stack เป็นชื่อชนิดข้อมูลที่ถูกสร้างขึ้นใหม่ (user-defined data type) และ s เป็นชื่อตัวแปรที่มีชนิดข้อมูลเป็นชนิด stack และปฏิบัติการที่ s สามารถกระทำได้ก็คือ push และ pop ตามนิยามศัพท์ของการทำโปรแกรมเชิงวัตถุ ตัวแปร s จะถูกเรียกว่าวัตถุ หรือ อ็อบเจ็ค (object) โดยนิยามว่าวัตถุคือ สิ่งที่เกิดขึ้นจากคลาส (instance of class) วัตถุประกอบด้วยตัวแปร (instance variables) และวิธีกระทำการหรือเมธ็อด (methods) การจะทำให้วัตถุเกิดการกระทำใดจะใช้วิธีส่งข้อความ (message) ให้วัตถุนั้น เช่น s.push (25) หมายถึงการส่งข้อความ push(25) ให้กับวัตถุ s เมธ็อดคือ push และ (25) เป็นพารามิเตอร์ที่ส่งให้เมธ็อด

การซ่อนรายละเอียดของโครงสร้างข้อมูลภายในและส่วนรายละเอียดคำสั่งของแต่ละเมธ็อดทำให้การเปลี่ยนแปลงใดๆ ที่อาจเกิดขึ้นในภายหลังกับคลาสสแตกจะไม่มีผลกระทบกับโปรแกรมที่เรียกใช้สแตก ดังตัวอย่างในรูปที่ 8.4 ที่แสดงการเปลี่ยนแปลงคลาสสแตกให้ใช้ลิงค์ลิสต์เป็นโครงสร้างภายใน แทนที่จะใช้อาร์เรย์ โดยส่วนโพรโทไทพ์ของเมธ็อดยังเป็นเช่นเดิม ทำให้การเรียกใช้สแตกด้วยคำสั่ง stack s; s.push(25); ยังคงทำงานได้เหมือนเดิม

```
struct Node {
        int val;
        Node * next;
};
class stack {
        private :
                int count;
                Node * top;
        public :
                void push(int, v);
                int pop();
};
```

รูปที่ 8.4 การเปลี่ยนโครงสร้างภายในของสแตกจากอาร์เรย์เป็นลิงค์ลิสต์

คลาสทำหน้าที่กำหนดชนิดของข้อมูลขึ้นใหม่เพื่อให้ตรงกับความต้องการใช้งานของโปรแกรม ในการสร้างคลาสเราอาจไม่จำเป็นต้องกำหนดโครงสร้างตัวแปรและเมธ็อดใหม่ทั้งหมด ภาษาเชิงวัตถุอำนวยความสะดวกในการสร้างคลาสให้โดยอนุญาตให้ถ่ายทอดโครงสร้างและเมธ็อดพื้นฐานมาจากคลาสอื่น แล้วสร้างเพิ่มเติมเฉพาะบางโครงสร้างและเมธ็อดที่ยังขาดอยู่ ซึ่งจะช่วยให้การพัฒนาโปรแกรมทำได้รวดเร็วขึ้น

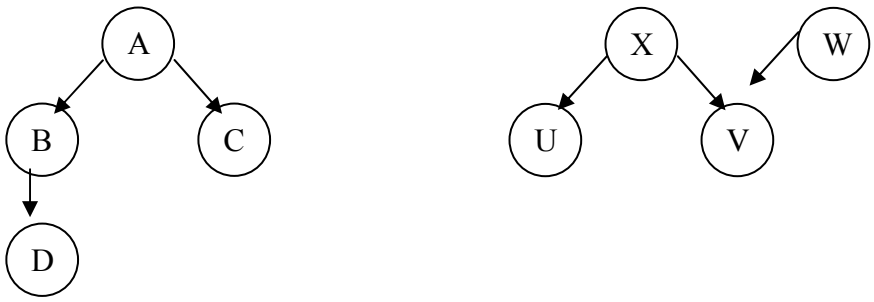
## 8.3 การถ่ายทอด

(Inheritance)

ในภาษา C++ อนุญาตให้สร้างคลาสใหม่โดยอาศัยพื้นฐานจากคลาสเดิมที่มีอยู่ด้วยรูปแบบคำสั่งดังต่อไปนี้

```
class NewClass : Existing Class{
        ...     // new member declarations
};
```

คลาสที่สร้างขึ้นใหม่จะเรียกว่าคลาสที่ได้รับการถ่ายทอด (derived class) และคลาสเดิมจะเรียกคลาสฐาน (base class) การถ่ายทอดสามารถทำเป็นลำดับขั้นดังแสดงในรูปที่ 8.5




(a) single inheritance (b) multiple inheritance for V

รูปที่ 8.5 ลำดับชั้นของการถ่ายทอดคลาส

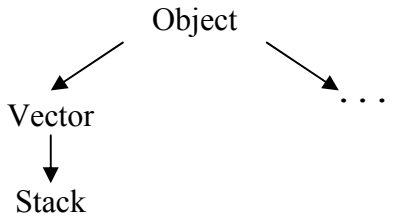
จากรูปที่ 8.5 (a) คลาส D ถ่ายทอดคุณสมบัติต่างๆ มาจากคลาส B แต่เนื่องจากจากคลาส B ถ่ายทอดมาจากคลาส A ดังนั้นคลาส D จึงได้รับคุณสมบัติมาจากทั้ง B และ A ในกรณีของการถ่ายทอดจากหลายคลาส (รูปที่ 8.5 (b)) เขียนรูปแบบคำสั่งได้ดังนี้

```
class V : X, W {
        ...         // new member declarations
}
```

ในภาษา Java คลาสที่ได้รับการถ่ายทอดจะเรียกว่าคลาสย่อย (subclass) ส่วนคลาสเดิมที่เป็นคลาสฐานจะเรียกว่า คลาสแม่ หรือ ซุปเปอร์คลาส (parent class or superclass) รูปแบบของคลาสจะเป็นดังนี้

```
class D extends B {
        ...         // new member declarations
};
```

การถ่ายทอดคลาสในภาษา Java จะเป็นแบบ single inheritance เท่านั้น เช่นถ้าคลาส Stack ถ่ายทอดมาจากคลาส Vector ซึ่งถ่ายทอดมาจากคลาส Object อีกต่อหนึ่ง (ตามรูปที่ 8.6) การประกาศคลาสจะเป็นดังในรูปที่ 8.7




รูปที่ 8.6 การถ่ายทอดคลาสในภาษา Java

```
class Vector extends Object {
        public void insertElementAt (Object obj, int index) {...}
        public Object elementAt     (int index) {...}
}
class Stack extends Vector {
        public Object pop() {...}
        public void push(Object obj) {...}
}
```

รูปที่ 8.7 การประกาศคลาส Vector และ Stack ในภาษา Java

คลาส Stack ถ่ายทอดมาจากคลาส Vector ดังนั้นถ้า s เป็นวัตถุของคลาส Stack นอกจากการเรียกใช้เมธ็อด s.push() และ s.pop() แล้วเรายังสามารถเรียกใช้เมธ็อด s.insertElementAt() และ s.elementAt ซึ่งเป็นเมธ็อดในคลาส Vector ที่เป็นคลาสแม่ของคลาส Stack

## 8.4 การมีหลายรูปและการยืดเหนี่ยวแบบพลวัต

(Polymorphism and dynamic binding)

แนวคิดของการออกแบบภาษาเชิงวัตถุ ที่ให้คลาสใหม่สามารถถ่ายทอดตัวแปรและเมธ็อดมาจากคลาสเดิมทำให้เกิดคุณสมบัติที่เรียกว่า การมีหลายรูป (polymorphism) ของเมธ็อดหรือฟังก์ชัน ดังตัวอย่างคำประกาศคลาสในรูปที่ 8.8 ที่มีคลาส shape เป็นคลาสฐาน ต่อจากนั้นคลาส circle และคลาส rectangle ถ่ายทอดคุณสมบัติต่างๆ ไปจากคลาส shape และสุดท้ายคลาส square ถ่ายทอดไปจากคลาส rectangle ทุกคลาสมีการสร้างฟังก์ชัน draw เพื่อทำหน้าที่วาดรูปทรงเฉพาะของคลาสนั้นแสดงบนจอภาพ เช่น คลาส circle วาดรูปทรงกลม, คลาส square วาดรูปทรงสี่เหลี่ยมจัตุรัส ดังนั้นฟังก์ชัน draw() จึงถูกเรียกว่าฟังก์ชันที่มีหลายรูป โดยฟังก์ชันจะมีรูปแบบเป็นอย่างไรขึ้นอยู่กับฟังก์ชันนั้นเป็นของคลาสใด

```
class shape {
        public :
                virtual void draw()=0;   // generic draw function
        ...
}
class circle : public shape {
        public :
                virtual void draw(){...}   // draw a circle
        ...
}
class rectangle : public shape {
        public :
                virtual void draw(){...}   // draw a rectangle
        ...
}

class square : public rectangle {
        public :
                virtual void draw(){...}   // draw a square
        ...
}
```

รูปที่ 8.8 การมีหลายรูปของฟังก์ชัน draw()

ฟังก์ชัน draw() ถูกประกาศไว้ที่คลาสฐานแล้วถูกนำมาเขียนใหม่ (override) ในแต่ละคลาสย่อยเพื่อให้สามารถวาดรูปทรงต่างๆ ตามลักษณะเฉพาะของคลาสนั้นๆ ได้ทำให้การปรับปรุงหรือเพิ่มเติมโปรแกรมทำได้รวดเร็วขึ้น เช่น ถ้าต้องการเพิ่มฟังก์ชันการวาดรูปสามเหลี่ยมก็สามารถทำได้โดยสร้างคลาส triangle ให้ถ่ายทอดมาจากคลาส shape แล้วเขียนฟังก์ชัน draw() ขึ้นใหม่คอมไพล์โปรแกรมแล้วลิงค์กับโปรแกรมเดิมก็จะได้โปรแกรมที่ทำงานกับรูปสามเหลี่ยมได้โดยไม่ต้องมีการแก้ไขใดๆ กับโปรแกรมเดิมที่มีอยู่

เมื่อมีการเรียกใช้ฟังก์ชัน draw() คอมไพเลอร์จะทำหน้าที่ตรวจสอบว่าการเรียกใช้นั้นกระทำผ่านอ็อบเจ็คใด เช่น

```
rectangle r;
r.draw();
```

อ็อบเจ็ค r เป็นชนิด rectangle ดังนั้นฟังก์ชัน draw() จะหมายถึงฟังก์ชันในคลาส rectangle การเชื่อมโยงฟังก์ชัน draw() ในคำสั่ง r.draw(); เข้ากับฟังก์ชันของคลาส rectangle เกิดขึ้นตั้งแต่ช่วงการคอมไพล์โปรแกรมดังนั้นการเชื่อมโยงแบบนี้จะเรียกว่า *การยึดเหนี่ยวแบบคงตัว* (static binding)

ถ้าการเรียกใช้ฟังก์ชัน draw() กระทำผ่านอ็อบเจ็คประเภทอ้างอิงดังตัวอย่างต่อไปนี้

```
square s;                    // s is a square shape
shape &ref_shape=s;          // a reference to s
ref_shape.draw();
```

ตัวแปร ref_shape เป็นตัวแปรหรือเป็นอ็อบเจ็คประเภทอ้างอิงที่ชี้ไปยัง shape ซึ่งเป็นคลาสฐาน แต่ถูกกำหนดให้ชี้ไปยังตำแหน่งเดียวกับ s ซึ่งเป็นอ็อบเจ็คของคลาส square ในกรณีเช่นนี้การพิจารณาว่าฟังก์ชัน draw() จะเป็น draw() ของคลาส shape หรือ draw() ของคลาส square จะไม่เกิดขึ้นในช่วงคอมไพล์โปรแกรมแต่จะไปเกิดขึ้นในขณะรันโปรแกรม ซึ่งระบบประมวลผลจะตัดสินว่าฟังก์ชัน draw() เป็นของคลาส square การชะลอการเชื่อมโยงฟังก์ชันเข้ากับคลาสที่ถูกต้องไปไว้ในช่วงรันโปรแกรมแบบนี้จะเรียกว่า *การยึดเหนี่ยวแบบพลวัต* (dynamic binding) หรือบางครั้งเรียกว่า การยึดเหนี่ยวที่ถูกชลอไว้ (late binding)

## 8.5 การทำโปรแกรมเชิงวัตถุในภาษาจาวา, ซีพลัสพลัส และสมอลทอล์ค

(Object-oriented programming in Java, C++, Smalltalk)

ตัวอย่างโปรแกรมต่อไปนี้แสดงการบวกค่าในอาร์เรย์ในลักษณะของการทำโปรแกรมเชิงวัตถุด้วยภาษา Java, C++ และ Smalltalk

*ตัวอย่างโปรแกรมในภาษา Java*

```
import java.io.*;
class DataConvert {
public int convert(byte ch) {return ch-'0';}};
class DataStore extends DataConvert {
public void initial (int a)
    {ci=0;
     size=a;};
void save(int a)
    {store[ci++]=a;};
 int setprint() { ci=0; return size;};
 int printval() { return store [ci++];};
 int sum()
      {int arrsum = 0;
       for(ci=0;ci<size;ci++)arrsum=arrsum+store[ci];
       return arrsum;};
 private static int maxsize = 9;
 int size; // Size of array
 int ci;   // Current index into array
 int[] store = new int[maxsize];};
 class sample {
 public static void main(String argv[])
      {int sz,j;
       byte[] Line = new byte [10];
       DataStore x = new DataStore ();
       try{
           while((sz= System.in.read(Line)) != 0)
               {int k = x.convert(Line[0]);
                x.initial(k);
               for(j=l;j<=k;j++)x.save(x.convert(Line[j]));
               for(j=x. setprint(); j>0; j—)
                       System.out.pr int (x. printval());
               System.out.print(" ; SUM=");
               System.out.sprintln(x.sum());}}
      catch(Exception e){System.out.println("File error.");}
     } // End main
   } // End class sample
```

*ตัวอย่างโปรแกรมในภาษา C++*

```
1   #include <stream.h>
2     // This is C++ IOstreams, stdio.h also works
3   class DataConvert {
4   protected:
5   int convert (char ch) {return ch-'0';}};

6   class DataStore: DataConvert{
7   public:
8       int initial(char a)
9           {ci=0;
10           return size = convert(a);};
11       void save(char a)
12           {store[ci++]=convert(a);};
13       int setprint() { ci=0; return size;};
14       int printval() { return store[ci++];};
15       int sum()
16           {int arrsum;
17           arrsuin=0;
18           for(ci=0;ci<size;ci++)arrsum=arrsum+store[ci];
19           return arrsum;}
20   private:
21       const int maxsize=9;
22       int size; // Size of array
23       int ci;   // Current index into array
24       int store[maxsize];};

25   main()
26       {int j,k;
27       DataStore x;
28       while((k=x.initial(cin.get()))!=0)
29           {for(j=0;j<k;j++)x.save(cin.get());
30           for(j=x.setprint();j>0;j--)cout << x.printval();
31           cout <<  "; SUM=" << x.sum() << endl;
32           while(cin.get()!='\n');}}
```

*ตัวอย่างโปรแกรมในภาษา Smalltalk*

```
1    Array variableSubclass: #Datastore
2          instanceVariableNames: ''
3          classVariableNames: 'DataFile ArrIndex Storage Size'
4          poolDictionaries: ''
5          category: nil !
6   !Datastore class methodsFor: 'instance creation'!
7   new
8         DataFile _ FileStream open:'data' mode: 'r'.
9         Storage _ Array new: 99.
10         Size _ 99.
11         self reset !!
12  !Datastore class methodsFor: 'basic' !
13  asgn: aValue
14        ArrIndex _ ArrIndex + 1.
15        Storage at: ArrIndex put: aValue !
16  getval
17        ArrIndex _ ArrIndex + 1.
18        ^Storage at: ArrIndex !
19  nextval
20        ^((DataFile next) digitValue - $0 digitValue)!
21  reset
22        ArrIndex _ 0 !!
23  |k j sum|
24  Datastore new.
25  "Initialize k"
26  [(k _ Datastore nextval) > 0]
27    while-True:[1 to: k do: [(j-Datastore nextval) print.
28                        Character space print.
29                        Datastore asgn: j].
30         Datastore reset.
31         sum _ 0.
32         ' SUM =' print.
33         1 to: k do: [ sum _ sum + Datastore getval].
34          sum printNl.
35          Datastore reset]!
```

## 8.6 สรุป

โปรแกรมคอมพิวเตอร์โดยทั่วไปจะประกอบด้วยส่วนที่เกี่ยวข้องกับการจัดเก็บข้อมูล เช่น ส่วนประกาศตัวแปรและชนิดของตัวแปร และส่วนที่เป็นชุดคำสั่งเพื่อควบคุมการเปลี่ยนแปลงค่าของข้อมูล วิธีการทำโปรแกรมเชิงวัตถุยังคงใช้ชุดคำสั่งเพื่อการควบคุมที่เหมือนกับวิธีการทำโปรแกรมเชิงคำสั่ง เช่น มีคำสั่ง if, for, while ประเด็นสำคัญที่ทำให้การทำโปรแกรมเชิงวัตถุแตกต่างจากการทำโปรแกรมเชิงคำสั่ง คือ ในส่วนที่เกี่ยวข้องกับข้อมูล โดยขยายมุมมองของข้อมูลที่หมายถึงตัวแปรที่ใช้เก็บค่าต่างๆ ให้ครอบคลุมกว้างขึ้นไปถึงฟังก์ชันหรือเมธ็อดต่างๆ ที่ทำงานกับข้อมูลเหล่านั้น แล้วเรียกข้อมูลที่ประกอบด้วยตัวแปรและฟังก์ชันบนตัวแปรว่าวัตถุ โดยมีคลาสทำหน้าที่บอกชนิดข้อมูลของวัตถุจึงเรียกคลาสว่าเป็นชนิดข้อมูลนามธรรม (abstract data type)

การออกแบบภาษาคอมพิวเตอร์สำหรับการทำโปรแกรมเชิงวัตถุจะต้องมีโครงสร้างของภาษาหรือมีคำสั่งเพื่อสนับสนุนหลักการสำคัญ 3 ประการของการทำโปรแกรมเชิงวัตถุ คือ การรวมข้อมูลเป็นหน่วยเดียว (data encapsulation), การถ่ายทอด (inheritance), การมีหลายรูปและการยืดเหนี่ยวแบบพลวัต (polymorphism and dynamic binding)

## แบบฝึกหัดท้ายบทที่ 8

**คำถามปรนัย:** ให้เลือกคำตอบที่ถูกต้องที่สุด

1. ภาษา C++ เป็นภาษาที่ได้รับการพัฒนาขึ้นมาจากภาษา C โดยเพิ่มคุณสมบัติใดเข้าไป ?

   ก. Procedure ข. Function

   ค. Object ง. Reference

2. ภาษาที่เป็นต้นกำเนิดของ Object และ class คือภาษาใด ?

   ก. Java ข. Smalltalk

   ค. Simula ง. J

3. ภาษา C++ แตกต่างจากภาษา C อย่างไร ?

   ก. การเขียนคำสั่ง ข. การตั้งชื่อตัวแปร

   ค. การกำหนดคลาส ง. การกำหนดฟังก์ชัน

4. ภาษา Java ใช้คลาสใดในการทำงานกับสตริง ?

   ก. char และ *char ข. String และ Snobol

   ค. String และ StringBuffer ง. String และ WordPat

5. ภาษา Java หลีกเลี่ยงปัญหาที่เกิดจากการใช้ pointer ด้วยวิธี ?

   ก. เปลี่ยนไปใช้ reference type ข. ให้ใช้เฉพาะ class ที่ผู้ใช้สร้างขึ้นเอง

   ค. ยกเลิกการใช้ค่า nil และ null ง. ให้ใช้คำสั่ง new

6. การใช้เครื่องหมาย + ให้ทำหน้าที่ทั้งการบวกและการเชื่อมต่อข้อความ เรียกการกระทำหลายหน้าที่นี้ว่าอะไร ?

   ก. mixed mode arithmetic ข. operator polymorphism

   ค. global side effect ง. operator overloading

7. ลักษณะในข้อใดเกี่ยวข้องกับ object-oriented programming มากที่สุด ?

   ก. การพัฒนาซอฟต์แวร์ที่เน้นข้อมูล และ การทำ data abstraction

   ข. การพัฒนาซอฟต์แวร์ที่เน้นกระบวนการ และ การทำ concurrency

   ค. การจัดสรรหน่วยความจำแบบคงที่ และ การทำ recursive

   ง. การใช้สถาปัตยกรรมแบบ von Neumann และ การทำ virtual machine

8. การสร้างสภาพแวดล้อมเพื่อเอื้อต่อการพัฒนาซอฟต์แวร์ (software development environment) ปรากฏในภาษาใด ?

   ก. UNIX shell programming ข. Script language

ค. Java Just-In-Time
ง. Microsoft visual C++

9. ความสามารถในข้อใดที่ภาษา Java ไม่มี ?

ก. การแปลงข้อมูลจาก int เป็น float
ข. การใช้คำสั่งบวกค่า pointer
ค. การใช้ตัวแปรที่ไม่ใช่ object
ง. การกำหนด inheritance

10. object ในภาษา Java เทียบเท่ากับตัวแปรประเภทใด ?

ก. static variable
ข. stack-dynamic variable
ค. explicit heap-dynamic variable
ง. implicit heap-dynamic variable

11. ข้อใดมีคุณสมบัติของ process abstraction ?

ก. subprogram
ข. primitive variable
ค. structure data type
ง. Java constant

12. การรวม subprogram เข้ากับข้อมูล เรียกว่าอะไร ?

ก. block
ข. meta-program unit
ค. encapsulation
ง. overriding

13. จุดมุ่งหมายของ abstract data type คืออะไร ?

ก. เพื่อป้องกันข้อมูลที่สำคัญจากบุคคลภายนอก
ข. เพื่อซ่อนรายละเอียดที่บุคคลภายนอกไม่จำเป็นต้องรู้
ค. เพื่อแยกการทำงานระหว่าง instance variable และ class method
ง. เพื่อแยกการคอมไพล์ class method ออกจาก instance variable

14. ภาษาแรกที่เริ่มใช้แนวทาง abstract data type คือภาษาใด ?

ก. Simula 67
ข. Smalltalk
ค. C++
ง. Java

15. เมื่อ subclass เปลี่ยนแปลงแก้ไข method ที่ถ่ายทอดมาจากคลาสอื่น การกระทำนี้เรียกว่าอะไร ?

ก. overwrite
ข. override
ค. overload
ง. overdrive

16. ถ้า subclass C ถ่ายทอดมาจากทั้ง class A และ class B จะเกิดเหตุการณ์ใดขึ้น ?

ก. multiple inheritance
ข. method overloading
ค. method polymorphism
ง. type-checking failure

17. ภาษาที่เป็น pure object-oriented จะทำคำสั่ง “ 21 + 3 “ อย่างไร ?

ก. กำหนดตัวแปรที่ 1 ให้มีค่า 21 และตัวแปรที่ 2 ให้มีค่า 3 จากนั้นเรียก method add()

ข. กำหนด int x = 21 และกำหนด int y = 3 จากนั้นสั่ง x = x + y

ค. กำหนด 21 ให้เป็น object ที่ 1 และ 3 ให้เป็น object ที่ 2 จากนั้นเรียก method add()

ง. จะเกิด compiler error เนื่องจากเครื่องหมาย + เป็น overload operator

18. ภาษาที่เรียกว่า ภาษาเชิงวัตถุ (object-oriented language) จะต้องประกอบด้วยลักษณะสำคัญใดบ้าง ?

| | | |
|---|---|---|
| ก. data abstraction, | garbage collection, | inheritance |
| ข. concurrency, | inheritance, | polymorphism |
| ค. data abstraction, | inheritance, | dynamic binding |
| ง. data abstraction, | interface, | abstract class |

19. ถ้าโปรแกรมเมอร์ ได้รับมอบหมายให้เขียนโปรแกรม device driver ควบคุมการทำงานของเครื่องถอนเงินอัตโนมัติ โปรแกรมเมอร์คนนี้ควรจะใช้ภาษาประเภทใดในการเขียนโปรแกรม ?

ก. system programming language ข. logic programming language

ค. functional programming language ง. object-oriented language

20. ข้อใดเป็นลักษณะสำคัญของภาษาในกลุ่ม imperative?

ก. การใช้คำสั่ง package, exception handling และ switch

ข. การใช้คำสั่ง assignment, selection และ iteration

ค. การใช้คำสั่ง assignment, exception handling และ garbage collection

ง. การใช้คำสั่ง assignment, recursive และ selection

21. ข้อใดเป็นลักษณะเด่นของภาษาคอมพิวเตอร์ในกลุ่ม procedural ?

ก. มีลักษณะ orthogonality สูง แต่มี reliability ต่ำ

ข. มีการใช้ตัวแปร การใช้คำสั่งกำหนดค่า และการทำงานซ้ำ

ค. มีการเรียกตัวเองซ้ำ (recursion) การทำ aliasing และการเรียกใช้ฟังก์ชัน

ง. มีคำสั่ง goto คำสั่ง switch และการใช้ pointer

22. ข้อใดกล่าวได้ถูกต้องเกี่ยวกับภาษา Java ?

ก. JDK บรรจุ runtime environment ที่ต้องใช้ในการ run โปรแกรม

ข. subclass สามารถเรียกใช้ตัวแปรที่ระบุว่าเป็น private ใน superclass

ค. สองเมธ็อดที่เป็น overload methods สามารถมี signature เหมือนกันได้

ง. สองเมธ็อดที่เป็น override methods สามารถมี signature เหมือนกันได้

23. ลักษณะ exception handling ปรากฏอยู่ในภาษาใด ?

ก. Ada, C++, Java
ข. FORTRAN 77, LISP, C
ค. Prolog, Scheme, ALGOL
ง. Perl, PL/I, C#

24. ความสามารถของโปรแกรมในการเผชิญกับ run-time error แล้วยังสามารถประมวลผลคำสั่งต่อไปได้ เรียกว่าอะไร ?

ก. run-time abstraction
ข. exception encapsulation
ค. exception handling
ง. error hiding

25. exception ทั้งหมดในภาษา Java ถือเป็น object ของคลาสที่ถ่ายทอดมาจากคลาสใด ?

ก. class Throwable
ข. class Exception
ค. class NullObject
ง. interface Exception

พิจารณาโปรแกรม Java ต่อไปนี้ แล้วตอบคำถามข้อ 26-27

```
/*  1  */ class A  extends  Exception { }
/*  2  */           class B  extends  Exception { }
/*  3  */ class Main {
/*  4  */           public static void main(String [] args) {
/*  5  */                try {
/*  6  */                         try {
/*  7  */                                if (args.length == 0) throw A(); else throw B();
/*  8  */                                }
/*  9  */                            catch(A a) { System.out.println("9"); }
/* 10 */                              finally   { System.out.println("11"); }
/* 11 */                          System.out.println("13");
/* 12 */                         }
/* 13 */                             catch(B b) { System.out.println("15"); }
/* 14 */                    System.out.println("17");
/* 15 */                }
/* 16 */          }
```

26. ถ้าคำสั่ง if ในบรรทัดที่ 7 throw exception A() โปรแกรม Java ข้างต้นจะพิมพ์ผลลัพธ์ใด ?

ก. 9 11 13 15 17
ข. 9 11 13 17
ค. 9 11 13 15
ง. 9 13 17

27. ถ้าคำสั่ง if ในบรรทัดที่ 7 throw exception B() โปรแกรม Java ข้างต้นจะพิมพ์ผลลัพธ์ใด ?

ก. 9 11 13 15 17
ข. 9 11 13 15
ค. 11 13 15 17
ง. 11 15 17

28. ถ้าปรากฏคำสั่ง Java ดังต่อไปนี้

```
throw new MyException("message");
```

คำกล่าวในข้อใดถูกต้องที่สุด ?

ก. โปรแกรมนี้ต้องสามารถเรียกใช้คลาสชื่อ MyException

ข. โปรแกรมนี้จะต้องตั้งชื่อตามชื่อคลาส คือ MyException

ค. โปรแกรมนี้มีข้อผิดพลาดเนื่องจากคำว่า throw ใช้ผิด

ง. โปรแกรมนี้มีข้อผิดพลาดเนื่องจากไม่มีคำว่า catch

29. คลาสในข้อใดต่อไปนี้ที่เป็น subclass ของ Throwable ?

ก. java.io และ java.lang

ข. Error และ Exception

ค. RuntimeException และ CompileTimeException

ง. NullException และ ObjectException