888143
การสร้างแบบจำลองและการโปรแกรมเชิงวัตถุ

พีระศักดิ์ เพียรประสิทธิ์

Outline

- constructors และ destructors
- information hiding
- สมาชิกแบบอาร์เรย์ภายในคลาส
- อาร์เรย์ของวัตถุ (object)
- หลักการสำคัญของ OOP
  - Encapsulation
  - Inheritance
  - Polymorphism
- คุณสมบัติ composition
- การค้นหาคลาส แอตทริบิวต์ เมธอด

Constructors

- Constructors มีไว้สำหรับการกำหนดค่าเริ่มต้นของสมาชิกใน class
- Constructors มี 2 ประเภท
  - Constructors แบบมี parameters
  - Constructors แบบไม่มี parameters (default constructor)
- ชื่อของ constructor จะต้องเหมือนกันชื่อของ class
- constructor ไม่มีการคืนค่าข้อมูล

Constructors (ต่อ)

- คลาสใดๆ สามารถมีได้มากกว่า 1 constructor
  - แต่ละ constructor จะมี parameter ไม่เหมือนกัน
- Constructors ถูกเรียกใช้งานอัตโนมัติเมื่อมีการสร้าง object
- Constructors execute automatically when a class object enters its scope
  - They cannot be called like other functions
  - Which constructor executes depends on the types of values passed to the class object when the class object is declared

คลาส Rectangle

```
class Rectangle{
    private:
        double width;
        double length;
    public :
        double getArea(){
            return width * length;
        }
        void setArea(double w, double l){
            width = w;
            length = l;
        }
        void print(){
            cout << "Rectangle" << endl;
            cout << "width : " << width << endl;
            cout << "length : " << length << endl;
        }
};
```

โปรแกรมหลัก

```
int main()
{
    Rectangle myRectangle;
    myRectangle.print();
    myRectangle.setArea(3.0, 4.0);
    cout << "My rectangle have area : "
         <<  myRectangle.getArea() << endl;
}
```

ผลลัพธ์

```
Rectangle
width : 0
length : 0
My rectangle have area : 12
```

เพิ่ม Constructors แบบไม่มี parameters

```
class Rectangle{
    private:
        double width;
        double length;
    public :
        Rectangle(){
            width = 0;
            length = 0;
        }
        double getArea(){
            return width * length;
        }
        void setArea(double w, double l){
            width = w;
            length = l;
        }
        void print(){
            cout << "Rectangle" << endl;
            cout << "width : " << width << endl;
            cout << "length : " << length << endl;
```

Constructors แบบไม่มี parameters

เพิ่ม Constructors แบบมี parameters

```
class Rectangle{
    private:
        double width;
        double length;
    public :
        Rectangle(){
            width = 0;
            length = 0;
        }
        Rectangle(double w, double l){
            width = w;
            length = l;
        }
        double getArea(){
            return width * length;
        }
        void setArea(double w, double l){
            width = w;
            length = l;
```

Constructors แบบมี parameters

โปรแกรมหลัก

```
int main()
{
    Rectangle myRectangle;
    myRectangle.print();
    Rectangle yourRectangle(2,3);
    yourRectangle.print();
    return 0;
}
```

เรียกใช้งาน constructor แบบไม่มี parameter

เรียกใช้งาน constructor แบบมี parameter

ผลลัพธ์

```
Rectangle
width : 0
length : 0

Rectangle
width : 2
length : 3
```

การใช้งาน Constructor

- constructor จะทำงานอัตโนมัติเมื่อมีการประกาศตัวแปรอ็อบเจกต์
- ตัวอย่างการเรียกใช้ default constructor อัตโนมัติ

```
className classObjectName;
```

- เช่น Rectangle myRectangle;

การใช้งาน Constructor แบบมี Parameters

- รูปแบบ syntax

```
className classObjectName(argument1, argument2, ...);
```

- จำนวนของ argument และชนิดข้อมูล จะต้องตรงกับ formal parameters (ตามลำดับ)

ข้อควรระวังเกี่ยวกับ constructor

- ถ้า class ไม่มี constructor
  - C++ จะมี default constructor ให้อัตโนมัติ แต่ไม่มีการกำหนดค่าเริ่มต้นให้กับสมาชิก
- ถ้า class มี constructor แบบมี parameter แต่ไม่ได้สร้าง default constructor
  - C++ ไม่สร้าง default constructor

Destructors

- Destructors are functions without any type
- ชื่อของ destructor ขึ้นด้วย '~' และตามด้วยชื่อ class
- ตัวอย่าง

```
~Rectangle();
```

- หนึ่ง class มีได้เพียง 1 destructor
- destructor ไม่มี parameters
- Destructor ทำงานอัตโนมัติเมื่ออ็อบเจกต์นั้นๆ สิ้นสุดการทำงาน หรือออกนอกขอบเขตการทำงาน

ตัวอย่าง Destructor

```
class Rectangle{
    private:
        double width;
        double length;
    public :
        Rectangle(){
            width = 0;
            length = 0;
        }
        Rectangle(double w, double l){
            width = w;
            length = l;
        }
        ~Rectangle(){
            cout << "Destroy Rectangle" << endl;
        }
};
```

ตัวอย่างการเพิ่มเมธอด

- เนื่องจากตัวดำเนินการเปรียบเทียบ == ไม่สามารถใช้งานได้นั้น
- เราสามารถเพิ่มเมธอดดังนี้

```
 bool isEqual(Rectangle param){
     if(width == param.getWidth() && length ==
param.getLength())
         return true;
     else
         return false;
}
double getLength(){
   return length;
}
double getWidth(){
   return width;
}
```

ตัวอย่างการเรียกใช้งานในโปรแกรมหลัก

```
if(myRectangle.isEqual(yourRectangle)){
        cout << "myRectangle is equal yourRectangle"
             << endl;
    }else{
        cout << "myRectangle is not equal yourRectangle"
             << endl;
    }
```

คุณสมบัติการซ่อนข้อมูล

- Information hiding: การซ่อน/ปิดบัง รายละเอียดของวิธีการดำเนินการกับข้อมูล
- Interface (header) file: ข้อกำหนดของคลาส
- Implementation file: รายละเอียดการทำงาน
- ในไฟล์ header ประกอบด้วย function prototypes และคอมเม้นที่บรรยายถึงฟังก์ชันการทำงาน
- ให้เขียนอธิบายถึง preconditions และ postconditions

คุณสมบัติการซ่อนข้อมูล (ต่อ)

- ไฟล์ header จะมีนามสกุลเป็น .h
- ไฟล์ Implementation จะมีนามสกุลเป็น .cpp
- ไฟล์ Implementation จะต้องทำการ include header file ด้วยคำสั่ง include
- การใช้งานคำสั่ง include
  - หากเป็นไฟล์ header ที่ผู้ใช้งานสร้างเอง ให้ใช้เครื่องหมาย double quotes (“) ครอบชื่อไฟล์ เช่น “myclass.h”
  - หากเป็นไฟล์ header ของระบบ (ไลบรารีของระบบที่มีอยู่แล้ว) ให้ใช้เครื่องหมาย angular brackets (<>) ครอบชื่อไลบรารี เช่น <iostream>

Rectangle.h

```
#include <iostream>
using namespace std;
class Rectangle{
    private:
        double width;
        double length;
    public :
        Rectangle();
        Rectangle(double w, double l);
        ~Rectangle();
        double getArea();
        void setArea(double w, double l);
        void print();
};
```

Rectangle.cpp

```
#include "Rectangle.h"
Rectangle::Rectangle(){
    width = 0;
    length = 0;
}
Rectangle::Rectangle(double w, double l){
    width = w;
    length = l;
}
Rectangle::~Rectangle(){
    cout << "Destroy Rectangle" << endl;
}
```

Rectangle.cpp (ต่อ)

```
double Rectangle::getArea(){
    return width * length;
}
void Rectangle::setArea(double w, double l){
    width = w;
    length = l;
}
void Rectangle::print(){
    cout << "Rectangle" << endl;
    cout << "width : " << width << endl;
    cout << "length : " << length << endl;
}
```

การแยกไฟล์ .h และ .cpp

- การนิยามคลาสในไฟล์ .h
  - ทุก contructor และทุก method ต้องปิดท้ายด้วยเครื่องหมาย ;
- การ implement ไฟล์ .cpp
  - เรียกใช้งาน #include “ชื่อไฟล์.h”
- สำหรับ constructor
  - ชื่อคลาส :: ชื่อคอนสตักเตอร์ ()
- สำหรับ method
  - การคืนค่า ชื่อคลาส :: ชื่อเมธอด ()
- เครื่องหมาย :: เรียกว่า Scope resolution operator

สมาชิกแบบอาร์เรย์ภายในคลาส

- เราสามารถกำหนด attribute ของคลาสให้เป็นแบบอาร์เรย์ได้
- ในการเข้าถึงอาร์เรย์แต่ละตัวนั้นยังคงใช้เครื่องหมาย [] ในการระบุตำแหน่งที่ต้องการจะเข้าถึงข้อมูล
- ข้อควรระวัง
  - ควรกำหนดค่าเริ่มต้นให้กับทุกๆ สมาชิกในอาร์เรย์ผ่าน constructor มิฉะนั้นค่าของอาร์เรย์อาจเป็นค่าใดๆ ในหน่วยความจำขณะนั้น

ตัวอย่างคลาสนักเรียน

- นักเรียนแต่ละคน มี ชื่อ สกุล
- มีคะแนนสอบกลางภาค 40%
- มีคะแนนสอบปลายภาค 40%
- มีคะแนนเก็บ (การบ้าน) 20%
  - การบ้านมีทั้งหมด 5 ครั้งๆ ละ 10 คะแนน

- จงคำนวณเกรดและแสดงค่า

ตัวอย่างข้อมูล

- นักเรียนชื่อว่า Jame Wattson
- ส่งการบ้านทั้ง 5 ครั้ง มีคะแนนดังนี้ 10, 8, 7, 5, 10 = 40
- คะแนนการบ้าน = 20/50*40 = 16 คะแนน
- สอบกลางภาค ได้คะแนน 30 คะแนน
- สอบปลายภาค ได้คะแนน 20 คะแนน
- รวมคะแนน 66 คะแนน
- เกรด C

```cpp
#include <iostream>
#include <string>
using namespace std;


class Student{
    private:
        string firstname;
        string lastname;
        float mid_score;
        float final_score;
        float hw_score[5];
        float sumScore();
    public:
        Student();
        Student(string first, string last);
        void setMidScore(float mid);
        void setFinScore(float fin);
        void setHW(float score, int no);
        char calculateGrade();


};
```

```
#include "Student.h"
Student::Student(){
    firstname = "";
    lastname = "";
    for(int i = 0 ; i < 5; i++)
        hw_score[i] = 0;
}
Student::Student(string first, string last){
    firstname = first;
    lastname = last;
    for(int i = 0 ; i < 5; i++)
        hw_score[i] = 0;
}

void Student::setMidScore(float mid){
    mid_score = mid;
}
void Student::setFinScore(float fin){
    final_score = fin;
}
void Student::setHW(float score, int no){
    hw_score[no] = score;
}
```

```
char Student::calculateGrade(){
    float sum = sumScore();
    if(sum > 80){
        return 'A';
    }else if(sum > 70){
        return 'B';
    }else if(sum > 60){
        return 'C';
    }else if(sum > 50){
        return 'D';
    }else{
        return 'F';
    }
}
float Student::sumScore(){
    float sumhw = 0;
    for(int i = 0 ; i < 5; i++){
        sumhw += hw_score[i];
    }
    float sum = mid_score + final_score + (20.0/50.0*sumhw);
    return sum;
}
```

```
#include "Student.h"
int main(){
    Student jame("Jame","Wattson");
    // Set Homework score
    jame.setHW(10,0);
    jame.setHW(8,1);
    jame.setHW(7,2);
    jame.setHW(5,3);
    jame.setHW(10,4);
    jame.setMidScore(30);
    jame.setFinScore(20);
    cout << "Jame has grade : " <<
jame.calculateGrade() << endl;


    return 0;
}
```

```
  Jame has grade : C
```

อาร์เรย์ของวัตถุ

- ในการสร้างอาร์เรย์ของวัตถุ ก็คล้ายกับการสร้างตัวแปรอารเรย์ปกติ ก็คือ
  - dataType arrayName[intExp];
  - ชื่อคลาส ตัวแปร[intExp]
  - โดยที่ intExp เป็นตัวเลข (interger) จำนวนเต็มบวก
- เช่น Student students[10]; เป็นการสร้างวัตถุที่ชื่อว่า students จำนวน 10 ตัวโดยที่มีแม่แบบมาจากคลาส Student

อาร์เรย์ของวัตถุ

- ตัวแปรอาร์เรย์ของวัตถุ แต่ละตัวมีข้อมูล (attribute) เป็นของตนเอง
- ตัวแปรอาร์เรย์ของวัตถุ แต่ละตัวมีเมธอด (method) ที่สามารถเรียกใช้งานได้เหมือนเดิม
- วิธีการเข้าถึงข้อมูลหรือเรียกใช้เมธอดจะต้องอ้างอิงหมายเลขลำดับของวัตถุนั้น เช่น
  - students[2].firstName
  - students[2].getFirstName()

หลักการสำคัญของ OOP

- Information hiding คือ ซ่อนรายละเอียดการทำงานของเมธอดที่เกี่ยวข้องกับข้อมูล (attribute) และไม่ให้อ็อบเจกต์อื่นๆสามารถที่จะเข้าไปแก้ไขข้อมูลได้โดยตรง
- Encapsulation คือ การรวมข้อมูล (attribute) และพฤติกรรม (method) ทั้งหลายที่เกี่ยวข้องกัน และทำงานร่วมกันเอาไว้ใน object หนึ่งๆ (combine data and operations on data in a single unit)
- Inheritance คือ การสร้างแม่แบบใหม่ (Class) ใหม่จากแม่แบบที่มีอยู่แล้ว (create new objects from existing objects) ดังนั้น Object ที่สร้างจากแม่แบบใหม่ จะมี attribute และ method จากคลาสแม่แบบเดิม
- Polymorphism คือ การพ้องรูป การมีหลายรูปแบบ (the ability to use the same expression to denote different operations)

ความสัมพันธ์ระหว่างคลาส

- ความสัมพันธ์ระหว่าง class 2 class นั้น
  - Inheritance (“is-a” relationship)
  - Composition (“has-a” relationship)

ความสัมพันธ์ระหว่างคลาส

- ความสัมพันธ์ระหว่างคลาส 2 คลาสนั้น
  - ความสัมพันธ์แบบ has-a คือ การสร้างคลาสใหม่โดยที่มีคลาสอื่นเป็นส่วนประกอบ เช่น
    - รถยนต์มีเครื่องยนต์เป็นส่วนประกอบอยู่ภายใน
    - นกมีปีกเป็นส่วนประกอบ
  - ความสัมพันธ์แบบ is-a คือ การที่เราสร้างคลาสใหม่ขึ้นจากคลาสที่อยู่แล้ว (คลาสที่สร้างขึ้นมาใหม่นั้นจะมีคุณสมบัติจากคลาสเดิมทุกประการ) เช่น
    - พนักงานทุกคนเป็นมนุษย์
    - นกเป็นสัตว์

Inheritance

- Inheritance เป็นความสัมพันธ์แบบ “is-a”
  - เช่น พนักงานทุกคนเป็นมนุษย์
- Inheritance เป็นวิธีการหนึ่งที่ให้เราสร้าง Class ใหม่จาก Class ที่มีอยู่แล้วได้
  - คลาสที่สร้างใหม่เรียกว่า derived classes
  - คลาสที่มีอยู่แล้วเรียกว่า base classes
- Derived classes ถ่ายทอดคุณสมบัติจาก base classes

Inheritance (ต่อ)

- Inheritance คือการถ่ายทอดข้อมูล (ซึ่งก็คือ attribute และ method) จากคลาสลำดับที่สูงกว่า (base class) ไปยังคลาสลำดับที่ต่ำกว่า (Derived Class)
- โดยที่ Derived class นั้นสามารถเปลี่ยนแปลงหรือแทนที่ข้อมูล (override) ที่ได้รับการถ่ายทอดมานั้นได้ เช่น
  - คลาสพนักงาน จะประกอบด้วย attribute ชื่อ/รหัสประจำตัวพนักงาน
  - คลาสหมอ จะประกอบด้วย ชื่อ/รหัสประจำตัวพนักงาน/สาขาที่เชี่ยวชาญ

นิยามคำศัพท์

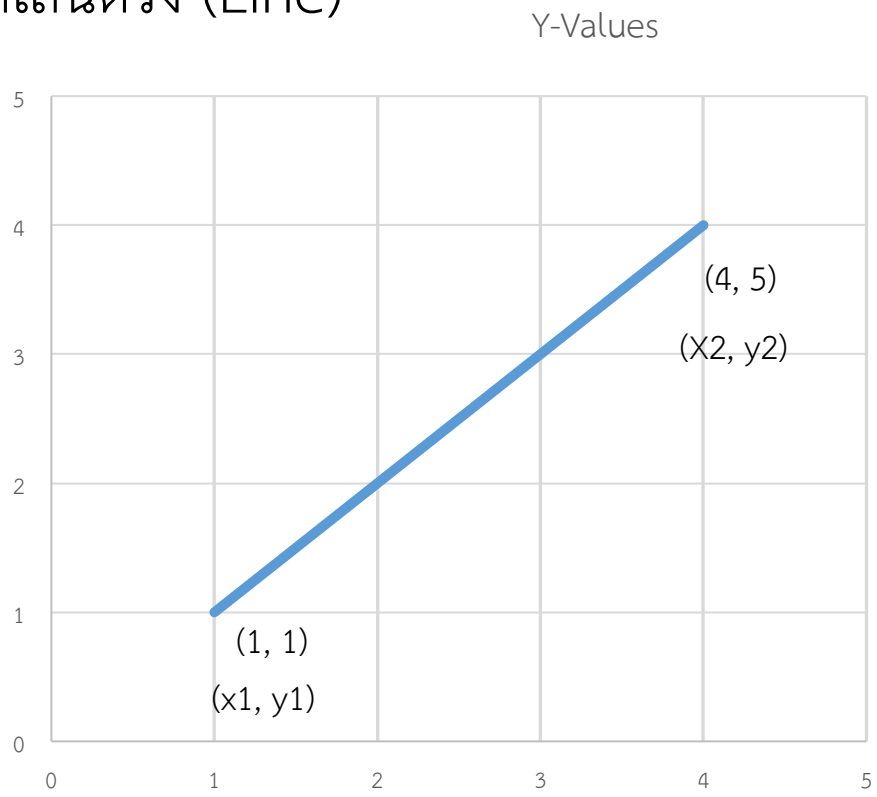
คำศัพท์	ความหมาย	คำเหมือน
base class	คลาสต้นแบบ	super class parent class คลาสแม่
Derived class	คลาสที่สืบทอดคุณลักษณะและพฤติกรรมมาจากคลาสแม่	Subclass คลาสลูก

คุณสมบัติประกอบ (Composition)

- คุณสมบัติประกอบ (Composition) เป็นการนำ Class ที่มีอยู่เดิมมาใช้งาน โดยกำหนดเป็น attribute ของ Class ที่จะทำการสร้างขึ้นมาใหม่
- Composition เป็นความสัมพันธ์แบบ “has-a”
  - เช่น คลาสรถยนต์ ประกอบด้วย คลาสเครื่องยนต์ คลาสตัวถัง คลาสล้อรถ
- ตัวแปรพารามิเตอร์ของวัตถุ (object) ที่ถูกส่งไปยังคอนสตรัคเตอร์ (constructor) ของนั้นจะถูกระบุในนิยามของคลาสที่สร้างขึ้นใหม่ด้วย

ตัวอย่าง

- คลาสเส้นตรง (Line)



ตัวอย่างคลาสเส้นตรง

- อาจจะประกอบด้วย x1, y1, x2, y2
- อาจจะประกอบด้วย จุดเริ่มต้น (x1, y1) และจุดสิ้นสุด (x2, y2)

ตัวอย่าง คลาสรถยนต์

- รถยนต์ประกอบด้วย
  - ล้อ
  - ประตู
  - เครื่องยนต์

ตัวอย่าง คลาสบุคคล

- ทุกๆ คนต่างมีวันเกิด
- คลาส วัน (DateType)
  - มีข้อมูล วัน เดือน ปี
- คลาสบุคคล (PersonType)
  - มีข้อมูล ชื่อ นามสกุล
- คลาสข้อมูลบุคคล PersonalInfo ประกอบด้วย
  - หมายเลข ID
  - ชื่อ นามสกุล
  - วันเกิด

Composition (Aggregation) (ต่อ)

```
private:
    int dMonth; //variable to store the month
    int dDay;   //variable to store the day
    int dYear;  //variable to store the year
};
```

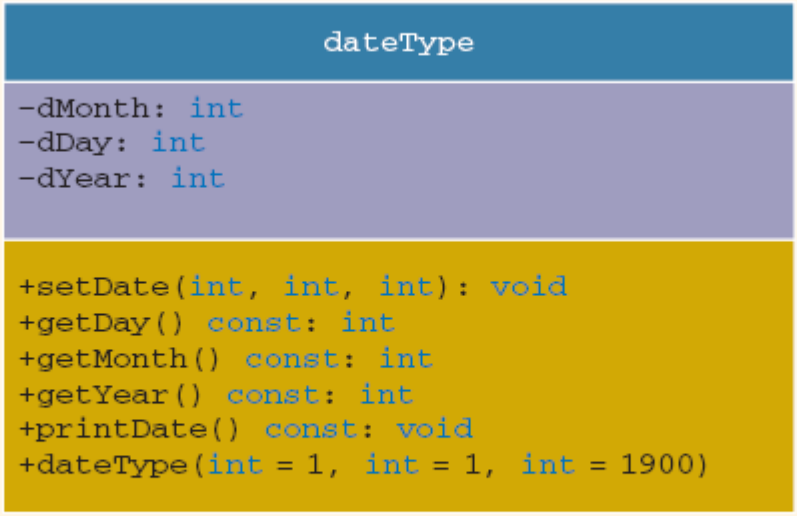


FIGURE 13-7 UML class diagram of the `class dateType`

Composition (Aggregation) (ต่อ)

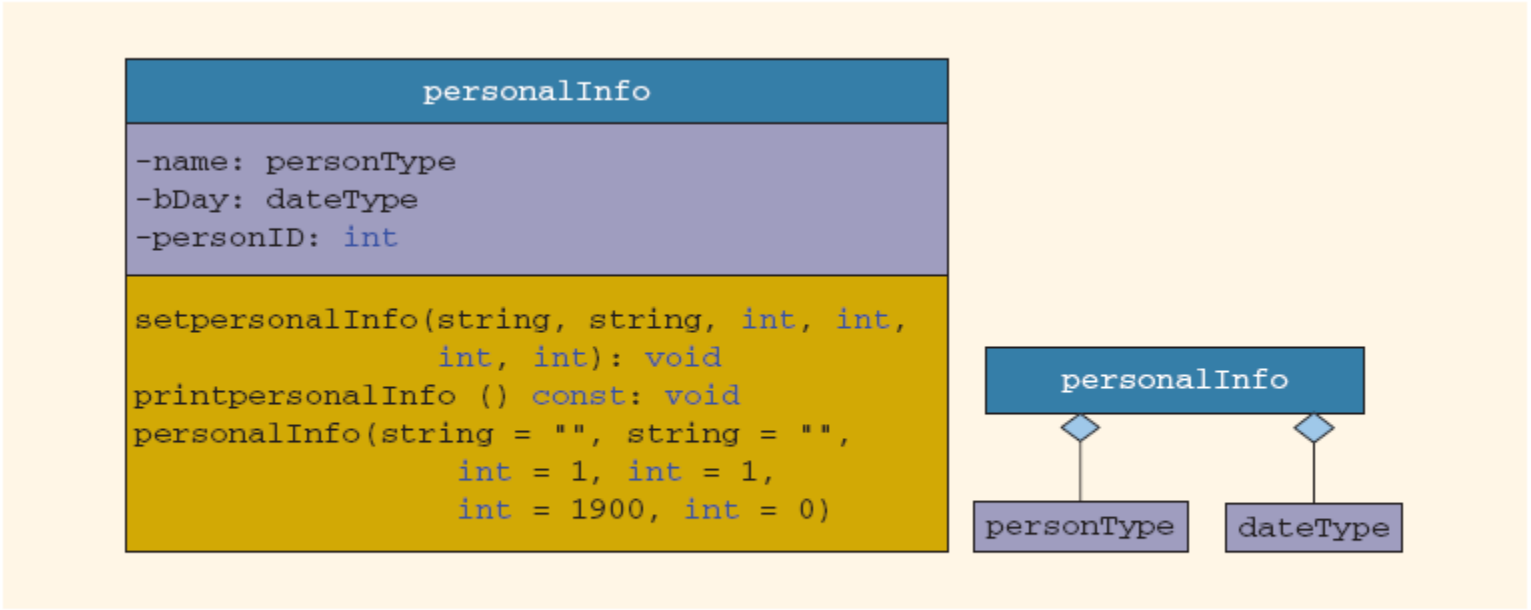


FIGURE 13-8 UML class diagram of the `class personalInfo` and composition (aggregation)

```
personalInfo::personalInfo(string first, string last, int month,
                           int day, int year, int ID)
          : name(first, last), bDay(month, day, year)
{
     .
     .
     .
}
```

Composition (Aggregation) (ต่อ)

```
void personalInfo::setpersonalInfo(string first, string last,
                           int month, int day, int year, int ID)
{
    name.setName(first,last);
    bDay.setDate(month,day,year);
    personID = ID;
}

void personalInfo::printpersonalInfo() const
{
    name.print();
    cout << "'s date of birth is ";
    bDay.printDate();
    cout << endl;
    cout << "and personal ID is " << personID;
}

personalInfo::personalInfo(string first, string last, int month,
                           int day, int year, int ID)
        : name(first, last), bDay(month, day, year)
{
    personID = ID;

}
```

Composition (Aggregation) (ต่อ)

- Object ของ Class จะถูกสร้างตามลำดับ
  - ตามลำดับที่ประกาศใน Class
    - ไม่ใช่ตามลำดับที่เรียงกันใน constructor

การออกแบบเชิงวัตถุและการโปรแกรมเชิงวัตถุ (Object-Oriented Design and Object-Oriented Programming)

- หลักการพื้นฐานของการออกแบบเชิงวัตถุมีดังนี้
  - Encapsulation การรวมข้อมูล (attribute) และพฤติกรรม (method) ทั้งหลายที่เกี่ยวข้องกันเข้าไว้ด้วยกัน
  - Inheritance การสร้างคลาสใหม่จากคลาสที่มีอยู่แล้ว
  - Polymorphism การพ้องรูป การมีหลายรูปแบบ (ความสามารถในการเรียกใช้ชื่อเดียวกันแต่ในการทำงานนั้นมีความแตกต่างกัน)
  - รายละเอียดของต่างๆ จะกล่าวในหัวข้อถัดไป

Identifying Classes, Objects, and Operations

- วิธีการค้นหาคลาส : เริ่มต้นทำการวิเคราะห์ปัญหาแล้วทำการค้นหาคำนาม (nouns) และคำกริยา (verbs)
  - คำนาม -> คลาส
  - คำกริยา -> เมธอด (พฤติกรรมของคลาส)
- สมมติว่าเราต้องการที่จะเขียนโปรแกรมที่คำนวณและพิมพ์ปริมาณและพื้นที่ผิวของรูปทรงกระบอก

Identifying Classes, Objects, and Operations (ต่อ)

- เราเขียนบรรยายเพื่อวิเคราะห์ปัญหาได้ดังนี้
  - เขียน**โปรแกรม**ที่*รับข้อมูล*ของ**ขนาดของรูปทรงกระบอก**และ*คำนวณ*แล้วทำการ*พิมพ์*ค่า**พื้นที่ผิว**และ**ปริมาตร**
  - *Write* a **program** to *input* the **dimensions** of a **cylinder** and *calculate* and *print* the **surface area** and **volume**
  - คำนาม คือ คำที่เป็นตัวพิมพ์หนา และ คำกริยา คือ คำที่ขีดเส้นใต้
  - จากรายการด้านบนเราจะได้รูปทรงกระบอก (cylinder) เป็นคลาส และสร้างวัตถุ (object) ได้หลายขนาด

Identifying Classes, Objects, and Operations (ต่อ)

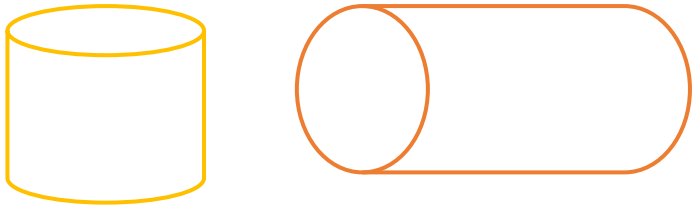
- คำนามที่มีลักษณะเฉพาะของรูปทรงกระบอก ให้กำหนดเป็น attribute
  - ขนาด (Dimensions)
  - พื้นที่ผิว (Surface area)
  - ปริมาตร (Volume)
- หลังจากระบุคลาสและระบุข้อมูล (attribute) ภายในคลาสแล้ว
  - ทำการกำหนดเมธอด (Operations) ที่วัตถุนั้นสามารถทำได้
  - ระบุว่าข้อมูล (attribute) ที่จำเป็นต้องมีในคลาส

Identifying Classes, Objects, and Operations (ต่อ)

- นำคำกริยาจากรายการมากำหนดเป็นเมธอด
- สำหรับคลาสรูปทรงกระบอก
  - ระบุเมธอด
    - นำเข้าข้อมูล (Input)
    - คำนวณ (Calculate)
    - พิมพ์ (Print)
  - ขนาด (Dimensions) เป็นส่วนจำเป็นของคลาสรูปทรงกระบอก แต่ พื้นที่ผิว (Surface area) และ ปริมาณ (Volume) สามารถคำนวณได้

Identifying Classes, Objects, and Operations (ต่อ)

- ขนาด (Dimensions) ที่เป็นส่วนจำเป็นของคลาสรูปทรงกระบอก



  - ศูนย์กลางของฐาน ความยาวรัศมี และความสูงของรูปทรงกระบอก เป็นคุณลักษณะเฉพาะของขนาด (dimensions)
  - The center of the base, radius of the base, and height of the cylinder are the characteristics of the dimensions
- เมธอดคำนวณ (Calculate)
  - เมธอด cylinderSurfaceArea สำหรับคำนวณหาพื้นที่ผิว (Surface area)
  - เมธอด cylinderVolume สำหรับคำนวณหาปริมาตร (Volume)
- เมธอด Print สำหรับแสดงค่าพื้นที่ผิวและปริมาตรออกทางจอภาพ

Identifying Classes, Objects, and Operations (ต่อ)

- วิธีการค้นหาคลาสและเมธอด จากคำนามและคำกริยา จากคำอธิบายปัญหา (descriptions to the problem) เป็นเพียงเทคนิคหนึ่งที่ใช้ในการค้นหาเท่านั้น ยังคงมีวิธีการอื่นๆ ที่ใช้ในการค้นหาคลาส (ในรายวิชาที่สูงขึ้น)

สรุป

- ความสัมพันธ์ระหว่างคลาส
  - Has-a
  - Is–a
- Composition เป็นความสัมพันธ์แบบ “has-a”
- คุณสมบัติ composition
  - สมาชิกของคลาส (attribute) เป็นอ็อบเจกต์ที่สร้างมาจากคลาสอื่น
  - อ็อบเจกต์ที่เป็นสมาชิกของคลาส จะถูกกำหนดการเรียกใช้งาน constructor ในนิยามของ constructor ของคลาสที่สร้างใหม่

สรุป

- หลักการพื้นฐานของการวิเคราะห์โปรแกรมเชิงวัตถุ
  - Encapsulation
  - Inheritance
  - Polymorphism
- การค้นหาคลาส
  - บรรยายปัญหา
  - ระบุคลาส จาก คำนาม
  - ระบบเมธอดจาก คำกริยา

เอกสารอ้างอิง

- C++ Programming: Program Design Including Data Structures, D.S. Malik