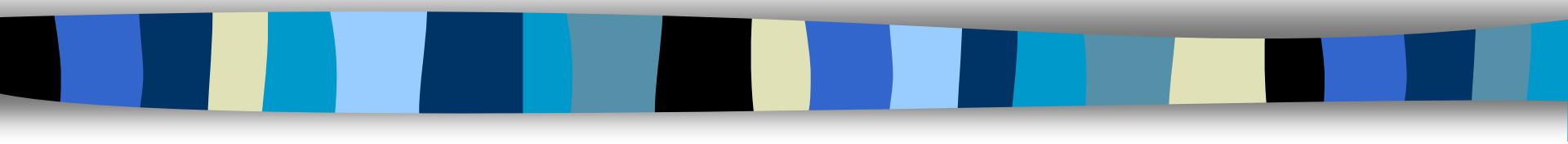
Object-Oriented Programming



Abstract Data Types (ADTs)

Paradigm Evolution

Characteristics of OOP

Object-oriented Concept

Imperative VS OOP

Abstract Data Types (ADTs)

- เป็นชนิดข้อมูลที่สร้างขึ้นโดยผู้เขียนโปรแกรม (user defined data type)
- สร้างโดยใช้ structure ที่มีในภาษา imperative
- เพื่อให้ชนิดข้อมูลนี้ใช้งานได้อย่างปลอดภัย ควรมีลักษณะ
  - รายละเอียดของการเก็บข้อมูลต้องไม่มีความสำคัญต่อการใช้งานชนิดข้อมูล
  - ข้อมูลภายในจะไม่ถูกอ้างถึงได้โดยตรง แต่ผ่านทาง operations ที่กำหนดไว้
- ภาษาที่สร้าง ADT ได้ต้องมีกลไก 2 อย่าง คือ
  - ***Encapsulation*** เพื่อนำข้อมูลกับ operations มาผูกติดกันเป็นหน่วยหนึ่ง
  - ***Information hiding*** เพื่อกำหนดขอบเขตในการอ้างถึงข้อมูลหรือ operations ของชนิดข้อมูลหนึ่ง ๆ
- ตัวอย่างเช่น ภาษา Modula-2, Ada --> object-based

Paradigm Evolution

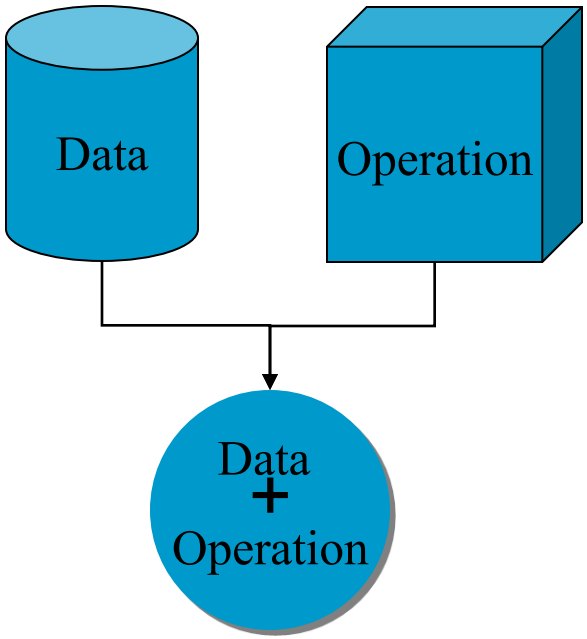
- Procedural 1950s -1970s (procedural abstraction)
- Data-Oriented early 1980s (data-oriented/ADTs)
- Object-Oriented late 1980s (Inheritance & dynamic binding)

Categories of OOP Language

- สนับสนุน OOP โดยเป็นส่วนเพิ่มเติมในภาษาที่อยู่แล้ว เช่น ภาษา C++, Ada95, CLOS, Scheme
- สนับสนุน OOP แต่ยังคงใช้โครงสร้างพื้นฐานแบบภาษา imperative เช่น ภาษา Eiffel, Java
- ภาษา OOP อย่างเดียว (Pure OOP) เช่น ภาษา Smalltalk

Characteristics of OOP

- Encapsulation
- Information hiding
- Inheritance
- Dynamic binding



OOP definitions

- Class
- Object, instance
- Subclass, derived class
- Superclass, parent class
- Method
- Message, behavior
- Inheritance
- Polymorphism
- Overriding
- Encapsulation

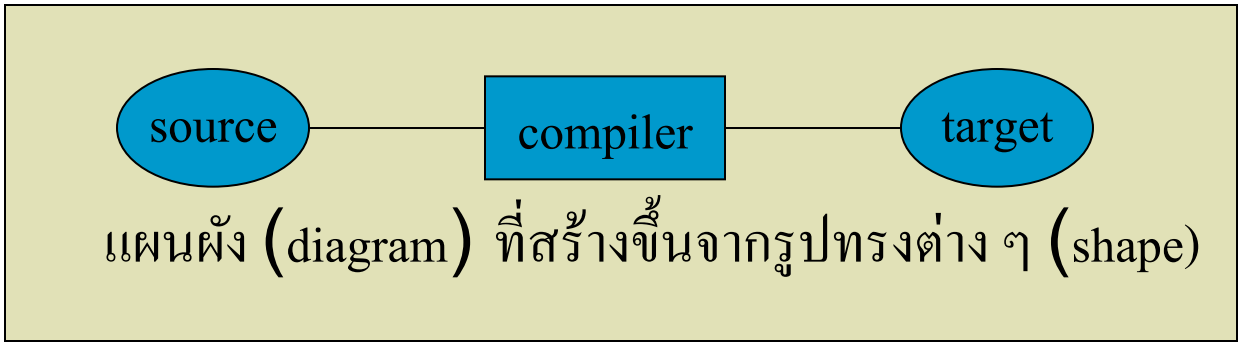
Object-oriented concept

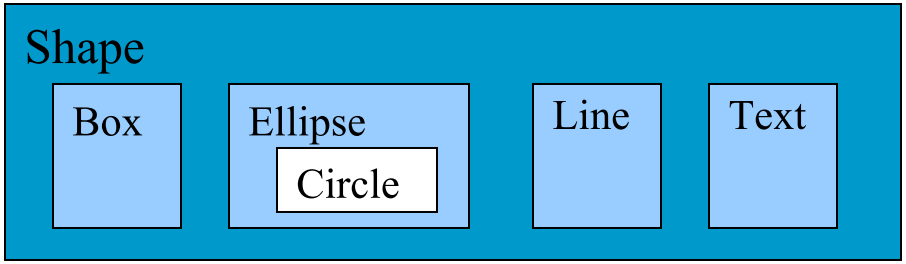
- วัตถุ (Object) ทั่วไปจะเก็บสถานะ (state) เป็นข้อมูลของตัวเองได้ และสามารถทำกิจกรรมบางอย่างได้ เช่น คำนวณ,เปลี่ยนข้อมูล, ติดต่อหรือโต้ตอบกับวัตถุอื่น
- นักเรียน --> วัตถุ
  - ข้อมูล : ชื่อ ที่อยู่ ผลการเรียน
  - กิจกรรม : ตอบคำถามต่าง ๆ เกี่ยวกับตัวเองได้, ติดต่อเพื่อดูผลสอบ
- ตัวแปรในภาษา Imperative ไม่สามารถจำลองพฤติกรรมและการทำงานของวัตถุเช่นนี้ได้

Object-oriented concept

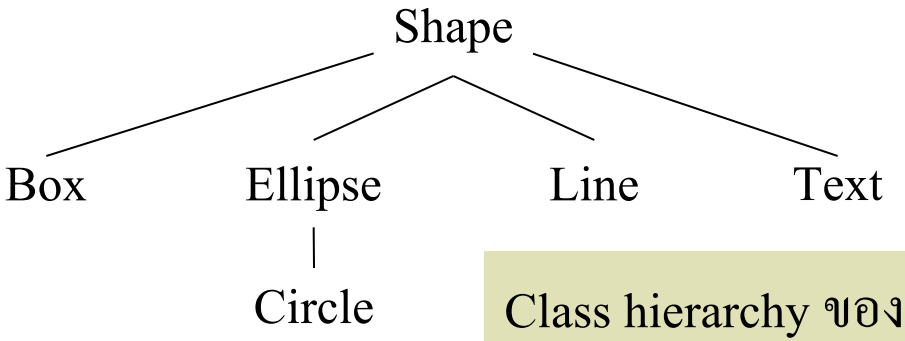
- *Class* เป็นกลไกสำหรับกำหนดโครงสร้างและพฤติกรรมของวัตถุ
- *Instance* จะถูกสร้างขึ้นเพื่อจำลองการทำงานของวัตถุใน class นั้น
  - มีพื้นที่หน่วยความจำสำหรับเก็บข้อมูลของวัตถุ
  - มี method สำหรับกระทำต่อข้อมูล หรือโต้ตอบกับ instance อื่น
  - สามารถสร้าง instance ของ class ได้หลาย instance โดยแต่ละตัวจะมีโครงสร้างเหมือนกัน แต่มีข้อมูลที่ต่างกันไป
  - เมื่อสร้าง instance แล้ว มันจะรออยู่เฉย ๆ จนกว่าจะมีใครเรียกใช้
- Class -> Type
- Instance -> Variable

OO Thinking





การจำแนกประเภทของรูปทรง



Class hierarchy ของ shape

Procedural thinking

Procedure draw(diagram *d*);

begin

for each shape *s* in diagram *d* **do**

begin

case *s* **of**

box : code to draw a box;

ellipse : code to draw an ellipse;

…..

arc : code to draw an arc;

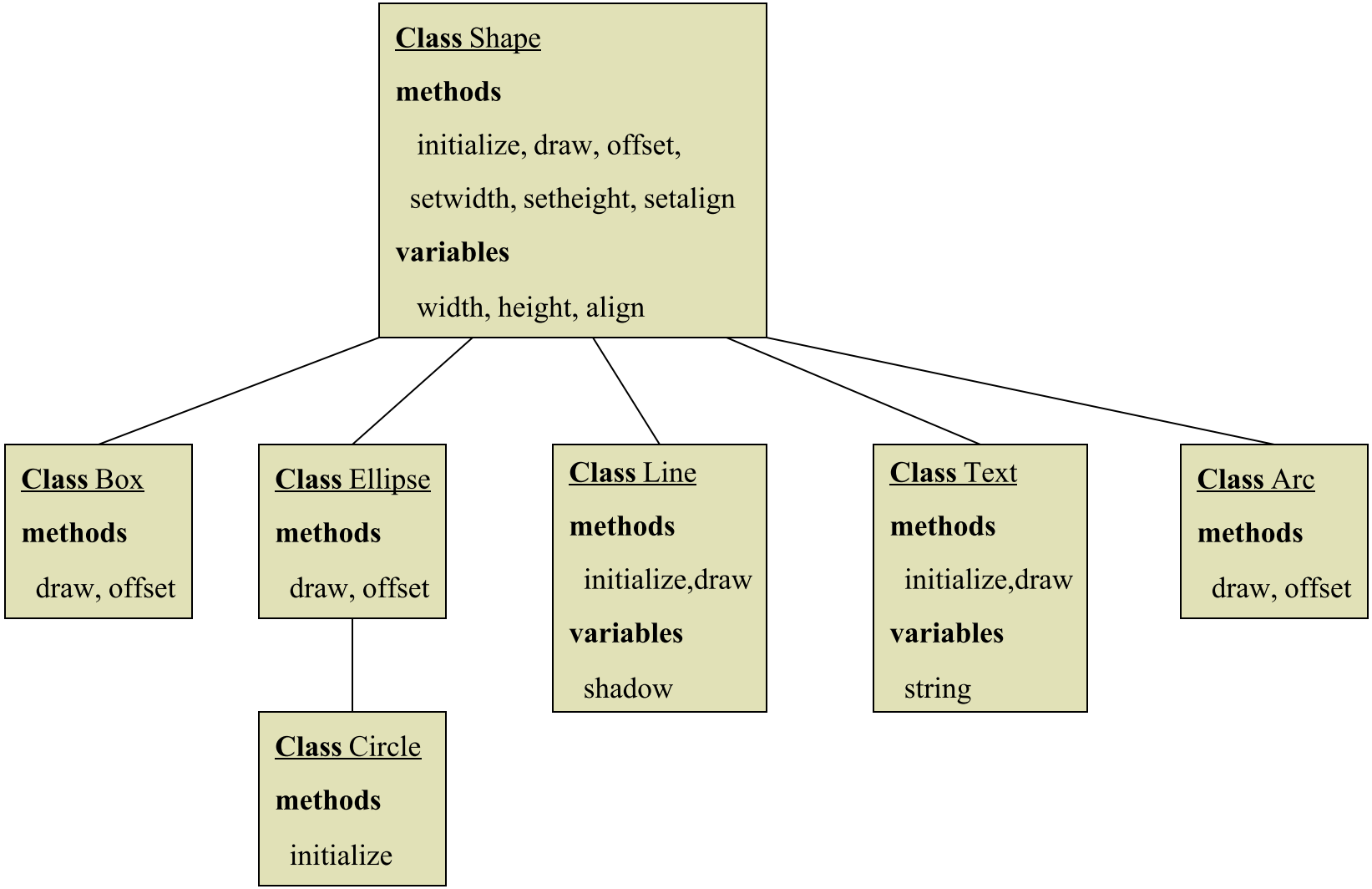
Object-oriented concept

- Class เป็นตัวระบุคุณสมบัติต่าง ๆ ของ object
- เมื่อมีการกำหนดพื้นที่สำหรับ object หนึ่งจะมีการสร้าง instance ขึ้น
- Object แต่ละตัวจะติดต่อกันโดยการรับส่ง *message*
- เมื่อ object ได้รับ message จะ execute method เพื่อตอบสนอง
- Class ประกอบด้วยสมาชิก 2 ประเภท คือ
  - data member คือสมาชิกที่เป็นข้อมูล อาจเป็นค่าคงที่ ตัวแปรของชนิดข้อมูลพื้นฐาน หรือตัวแปรที่เป็น object
  - method member คือสมาชิกที่เป็นฟังก์ชัน ซึ่งตอบสนองต่อ message เรียกสั้น ๆ ว่า *method*

Object-oriented concept

- เราสามารถสร้าง class ใหม่ได้ โดยนำ class ที่ถูกสร้างและใช้งานอยู่แล้วมาปรับปรุงเปลี่ยนแปลงให้เหมาะสมกับหน้าที่ใหม่ที่ต้องการ
- class เดิมจะยังคงใช้งานได้เช่นเดิม เรียกว่าเป็น *superclass* ของ class ใหม่
- class ใหม่จะได้สมาชิกของ class เดิมมาพร้อมกับมีการเพิ่มสมาชิกใหม่หรือเปลี่ยนแปลงสมาชิกที่มีอยู่เดิมให้ต่างไป จะเรียกว่าเป็น *subclass* ของ class เดิม
- กลไกการสร้าง class แบบนี้เรียกว่า การสืบทอดคุณสมบัติ หรือ *inheritance*

OO Thinking


Class Shape
methods
initialize, draw, offset,
setwidth, setheight, setalign
variables
width, height, align
Class Box
methods
draw, offset
Class Ellipse
methods
draw, offset
Class Circle
methods
initialize
Class Line
methods
initialize,draw
variables
shadow
Class Text
methods
initialize,draw
variables
string
Class Arc
methods
draw, offset

Object-oriented concept

- Method *overriding* เป็นการกำหนดชื่อของ method ใน subclass ให้มีชื่อเหมือนกัน method ใน parent class เพื่อเปลี่ยนแปลงการทำงาน method ใน parent class ให้ทำหน้าที่อื่นใน subclass
- การ binding ว่าชื่อ method ที่ instance ส่งมาเป็น method ตัวใด จะทำแบบ dynamic binding
- แนวคิดในการจัดการกับ instance ของ class ที่ต่างกันด้วย method เดียวกันเรียกว่า *polymorphism*
- polymorphism เป็นการสนับสนุนการออกแบบโปรแกรมเพื่อให้ใช้งานร่วมกัน หรือการนำกลับมาใช้ได้อีก

Object-oriented concept

- Method overloading เป็นการเรียก method หนึ่งด้วย argument ที่แตกต่างกันไปในการเรียกแต่ละครั้ง และจะต้องมีหลาย ๆ method สำหรับการเรียกแต่ละแบบ เช่น method ในการบวกอาจประกอบด้วย 3 method ซึ่งมี argument ต่างกัน คือ
  - function add(x:integer; y:integer)
  - function add(x:real; y:real);
  - function add(x:integer; y:real);
- Multiple Inheritance เป็นการสร้าง class ใหม่จาก class แม่มากกว่าหนึ่ง class

Example in C++

```
class A  {
public:
    int x;
    char f( );
        A( );
};
class D : public A {
    int x;
    int g( );
};
```

class D เป็น subclass ของ class A

Object ของ class D มีสมาชิก 4 ตัวคือ

- x เป็น data member ที่สืบทอด (inherit) มาจาก class A (A::x)
- f เป็น method ที่สืบทอดมาจาก class A
- x เป็น data member ที่ประกาศเพิ่มใน class D
- g เป็น method ที่ประกาศเพิ่มใน class D

Encapsulation & Inheritance

Example in C++

```
class Shape {
public : Shape * draw (Shape *)
  {  return this;  }
};
class Ellipse : public Shape {
public : Shape * draw (Shape *)
  {  return this;  }
};
class Circle : public Shape {
public : Shape * draw (Shape *)
  {  return this;  }
};
```

```
Shape * s;
s = new Ellipse;
Shape *p =  ...;
s->draw(p);          // Ellipse::draw(p)
s = new Circle;
s->draw(p);          // Circle::draw(p)
```

Polymorphism & Dynamic binding

Example in C++

```
class List {
    cell * rear;
public :   List( );
    int    empty( );
protected:
    void   add(int);
    void   push(int);
    int    get( );
};
class Queue : public List {
public :
           Queue( )   {}
    int    get( ) { return List::get(); }
    void   put(int x)  {add(x); }
};
```

สมาชิกของ object ใน class Queue

Public function

Queue	added(constructor)
get	added
put	added
List::empty	inherited

Protected function

add	inherited
push	inherited
List ::get	inherited

Information Hiding

ข้อดีของ OOP

- เข้าใจง่าย เพราะการทำงานเปรียบเสมือนการจำลองเหมือนในโลกจริง โดยอาศัยการมองทุกอย่างเป็น object ซึ่งแต่ละตัวมีหน้าที่และความหมายในตัว
- บำรุงรักษา และแก้ไขโปรแกรมได้ง่าย
- มีความปลอดภัยสูง เพราะจัดการกับ error ได้ดี
- นำกลับมาใช้ใหม่ได้ (reusability) ลดขั้นตอนในการเขียนโปรแกรม
- โปรแกรมมีคุณภาพสูง ใช้ได้หลาย platform

ข้อเสียของ OOP

- เข้าใจยาก สำหรับผู้เริ่มต้นเขียนโปรแกรม หรือถนัดเขียนโปรแกรมแบบ procedural
- ทำงานได้ช้ากว่าภาษาโปรแกรมอื่น
- ภาษามีความกำกวม ถ้ามีลักษณะ multiple inheritance

Imperative VS Object-oriented

Imperative

- Top-down design
- Procedure-oriented
- Algorithm
- Share global data
- Program-data separation
- Data transportation
- Single flow

Object-oriented

- Bottom-up
- Object-oriented
- Behavior
- Information hiding
- Encapsulation
- Communication with messages
- Multiple objects

Question

- ระบบตรวจคนเข้าเมือง
- ฉากคือสนามบิน
- ผู้โดยสารเข้าแถวเพื่อรอตรวจ
- เจ้าหน้าที่ตรวจคนเข้าเมืองทำหน้าที่ตรวจสอบวีซ่า
- ปัญหา คือ ต้องการศึกษาระบบการเข้าแถว
  - จัดคิวอย่างไร
  - ผู้โดยสารใช้เวลารอเท่าใด