ภาพรวมของการวิเคราะห์และออกแบบเชิงวัตถุ

ครั้งที่ 1

การวิเคราะห์และออกแบบเชิงวัตถุ

- ในรายวิชานี้ จะศึกษาการวิเคราะห์และออกแบบโปรแกรมเชิงวัตถุ โดยใช้ภาษา UML เป็นเครื่องมือในการวิเคราะห์ และออกแบบเชิงวัตถุ
- UML ย่อมาจาก Unified Modeling Language ซึ่งสามารถทำความเข้าใจได้ง่ายและมีขอบเขตการใช้งานกว้างมาก

คำถาม

ให้บอกชื่อ ภาษาโปรแกรมที่รู้จัก

ยุคต่างๆ ในการพัฒนา software (1)

- ภาษาเครื่อง (Machine code) เขียนโดยใช้เลขฐาน 2 และ 16
- ภาษา Assembly ใช้ opcode, mnemonic และสัญลักษณ์ต่างๆ แล้ว assemble เป็นภาษาเครื่อง
- ภาษาระดับสูง (High-level Language) เช่น Fortran, COBOL, BASIC เขียนด้วยภาษาที่เข้าใจง่าย เช่นคำสั่ง loop, if-else, function, for, do-while แล้วจึงใช้ compiler แปลเป็นภาษาเครื่อง

ยุคต่างๆ ในการพัฒนา software (2)

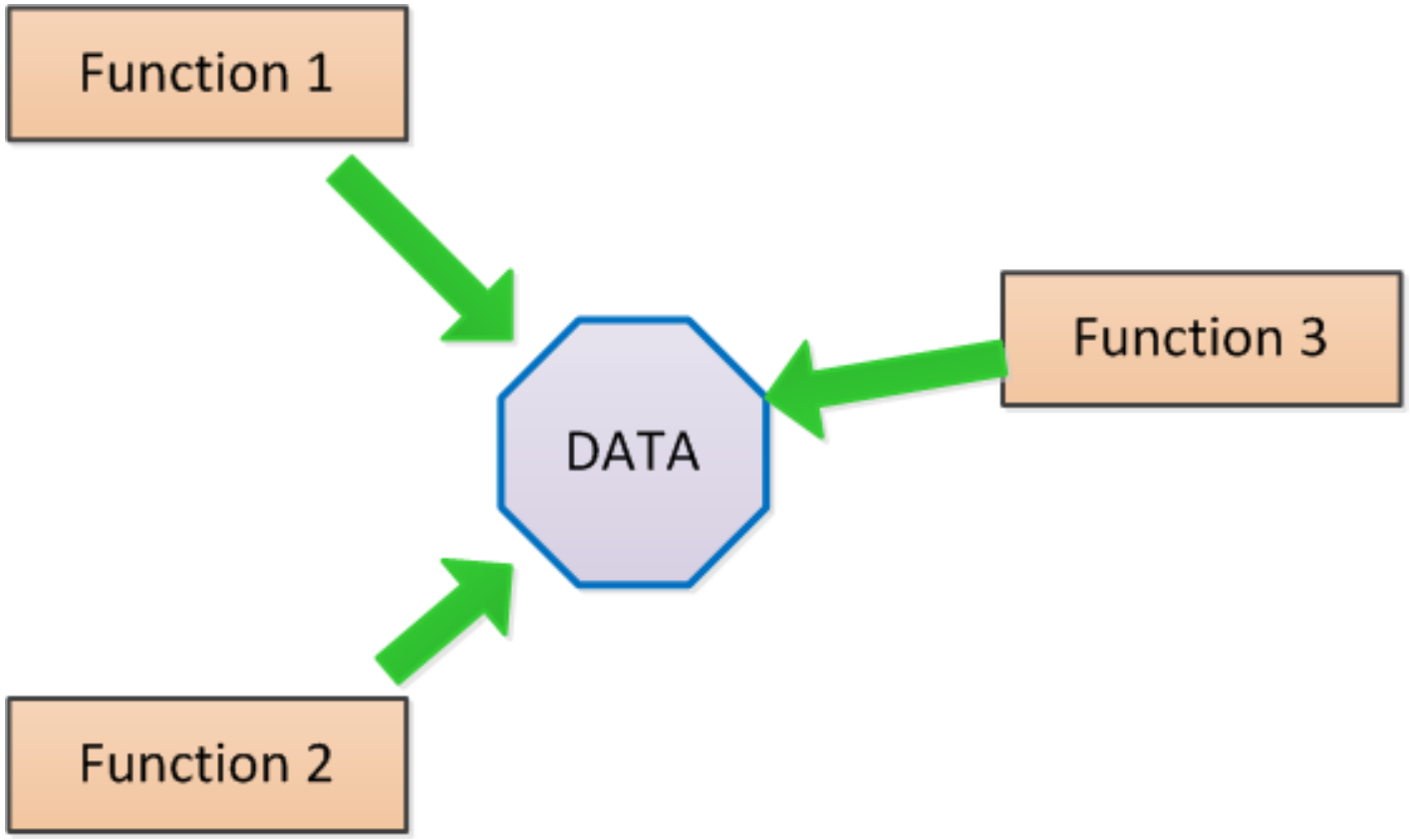
- ภาษาเชิงโครงสร้าง (Structured programming) เป็นภาษาระดับสูงที่เข้าใจได้ง่ายขึ้น เช่น Pascal, Ada ซึ่งจะแบ่งการทำงานยาวๆ ที่ซ้ำซ้อนออกเป็นบล็อกย่อยๆ เรียกว่า sub-task หรือ sub-routine
- ภาษาเชิงวัตถุ (Object-oriented programming) เป็นภาษาที่สร้างวัตถุที่อิสระต่อกันขึ้นมาในหน่วยความจำของคอมพิวเตอร์แล้วโปรแกรมให้ทำงานประสานสอดคล้องกัน การเขียนโปรแกรมลักษณะนี้ มีความใกล้เคียงกับโลกแห่งความจริงมากที่สุด

ปัญหาที่พบในภาษาเชิงโครงสร้าง

ในด้านความปลอดภัยของข้อมูล

- ไม่มี และ/หรือ ไม่รู้ว่าใครเป็นเจ้าของข้อมูล
- ควบคุมความปลอดภัยของข้อมูลไม่ได้
- ข้อมูลสามารถเข้าถึงและแก้ไขได้จากหลายๆ ฟังก์ชัน
- ถ้าเกิดความผิดพลาดกับข้อมูล จะไม่สามารถบ่งบอกฟังก์ชันที่ทำให้ข้อมูลมีปัญหานั้นได้

ปัญหาที่พบในภาษาเชิงโครงสร้าง



ปัญหาที่พบในภาษาเชิงโครงสร้าง

ข้อมูลที่ไม่ได้กำหนดค่าเริ่มต้น

```
int k;

printf("%d",k);
```

ปัญหาที่พบในภาษาเชิงโครงสร้าง

การจองและคืนทรัพยากร

- Memory leak
- Open files
- Open Database connections
- Open network connection

ตัวอย่าง memory leak

- `int* kp;`
- `kp = (int*) malloc(1000);`
- `kp = (int*) malloc(1000);`
- `kp = (int*) malloc(1000);`

ปัญหาที่พบในภาษาเชิงโครงสร้าง

ไม่รองรับ Abstract Data Type (ADT)

ADT = Data + Functions

Data: modeled as structure

Functions: global, ไม่เกี่ยวกับ structure (บางทีเรียกว่า operation)

Structure programming

ใครเปิดปลากระป๋อง?

Tools

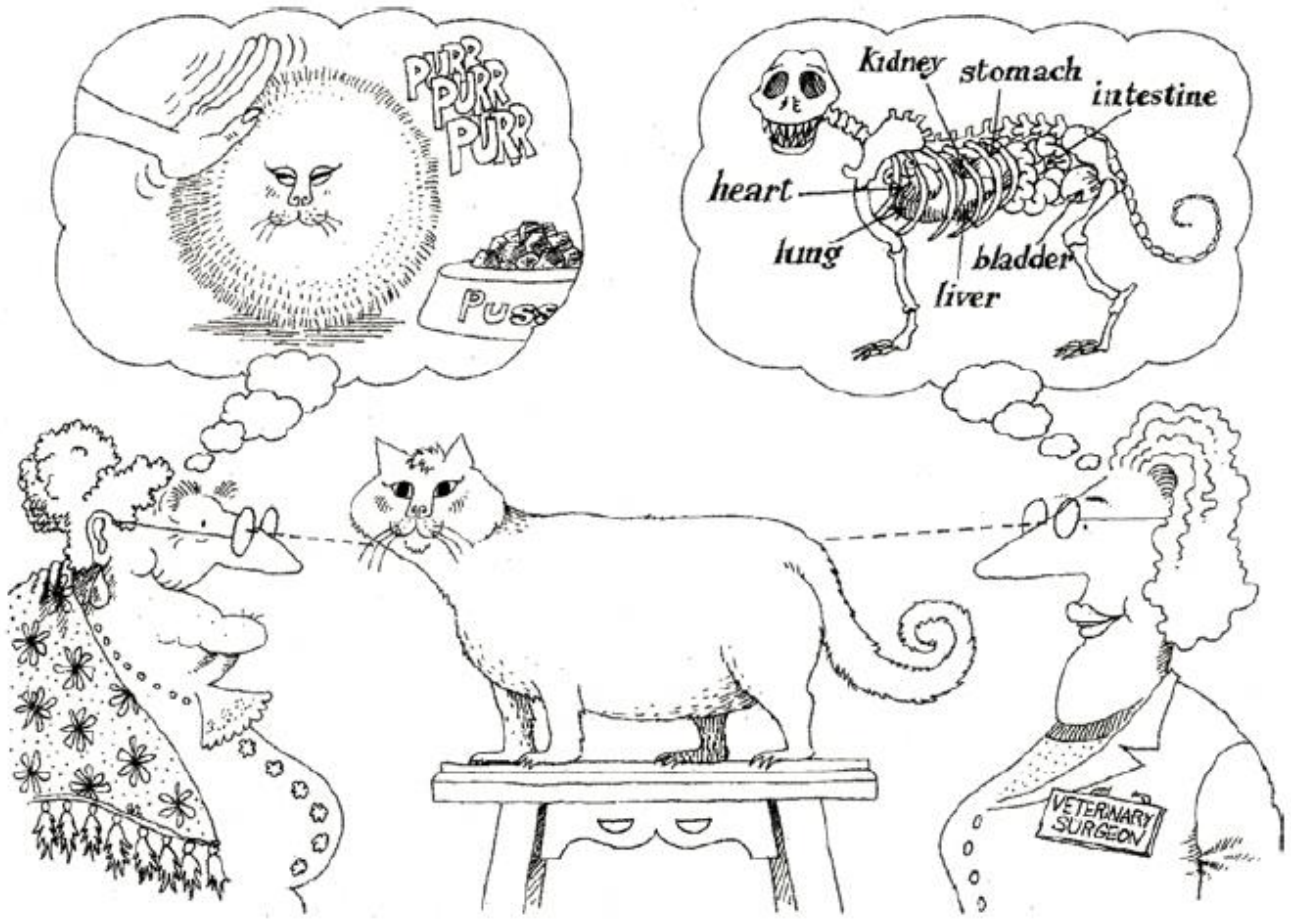
Data

Object-Oriented Programming

Data + Tools

มีเครื่องมือที่จะจัดการกับ data ในตัว

Abstraction ขึ้นอยู่กับมุมมองของแต่ละคน



Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer.

ภาพจาก Object-Oriented Analysis and Design with Applications (Grady Booch และคณะ)

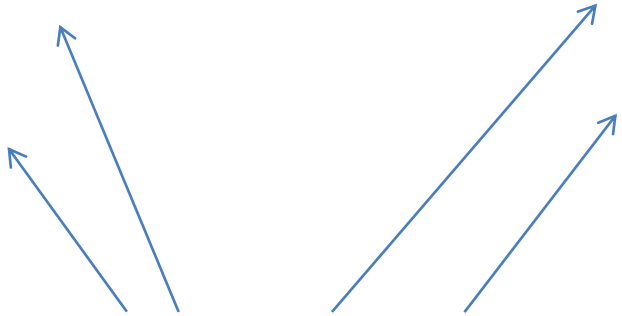
ปัญหาที่พบในภาษาเชิงโครงสร้าง

```
struct stack
{
  int top;
  char * store;
};
```

```
struct queue
{
  int front, rear;
  char * store;
};
```

```
push();

pop();
```

```
insert();

delete();
```



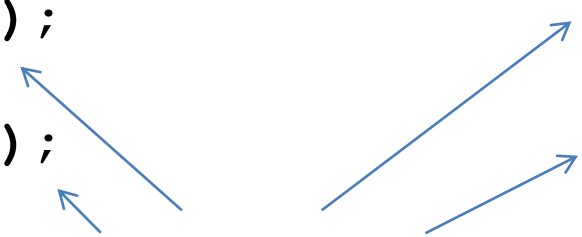
All functions are global !!

ทุก functions ใช้ได้กับทุก data structure

ปัญหาที่พบในภาษาเชิงโครงสร้าง

```
struct horse
{
  weight,
  color,
  age;
};

gallop();

canter();
```

```
struct eagle
{
  weight,
  age,
  wingspan;
};

fly();

hunt();
```



All functions are global !!

ทุก functions ใช้ได้กับทุก data structure

compiler ไม่สามารถรู้ได้ว่า function ใด ใช้กับ structure ใด

OO ช่วยแก้ปัญหาได้ โดยใช้ ADT

```
class stack
{
  int top;
  char * store;
};
```

```
   push();

   pop();
```

```
class queue
{
  int front, rear;
  char * store;
};
```

```
   insert();

   delete();
```

Class = State + Behavior

Complier รู้ว่าฟังก์ชันใดทำงานกับ structure ใด

OO ช่วยแก้ปัญหาได้ โดยใช้ ADT

class horse

```
{ weight, color,
  age;
};
```

```
gallop();

canter();
```

class eagle

```
{ weight, age,
  wingspan;
};
```

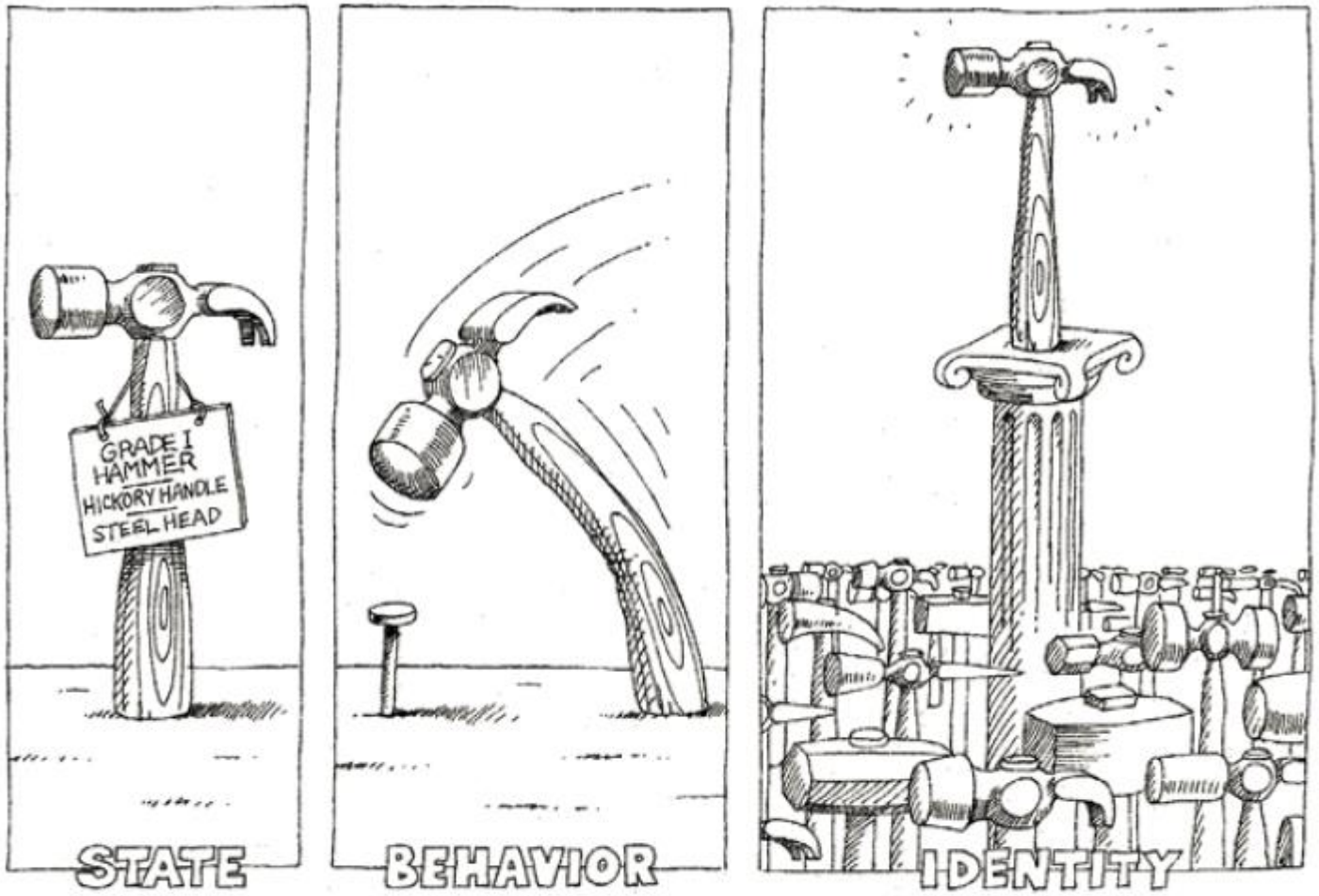
```
fly();

hunt();
```

Class = State + Behavior

- State
  - data members
  - fields
  - properties
- Behavior
  - member functions
  - methods

Objects

- An object has state, Exhibits some well defined behavior, and has a unique identity.

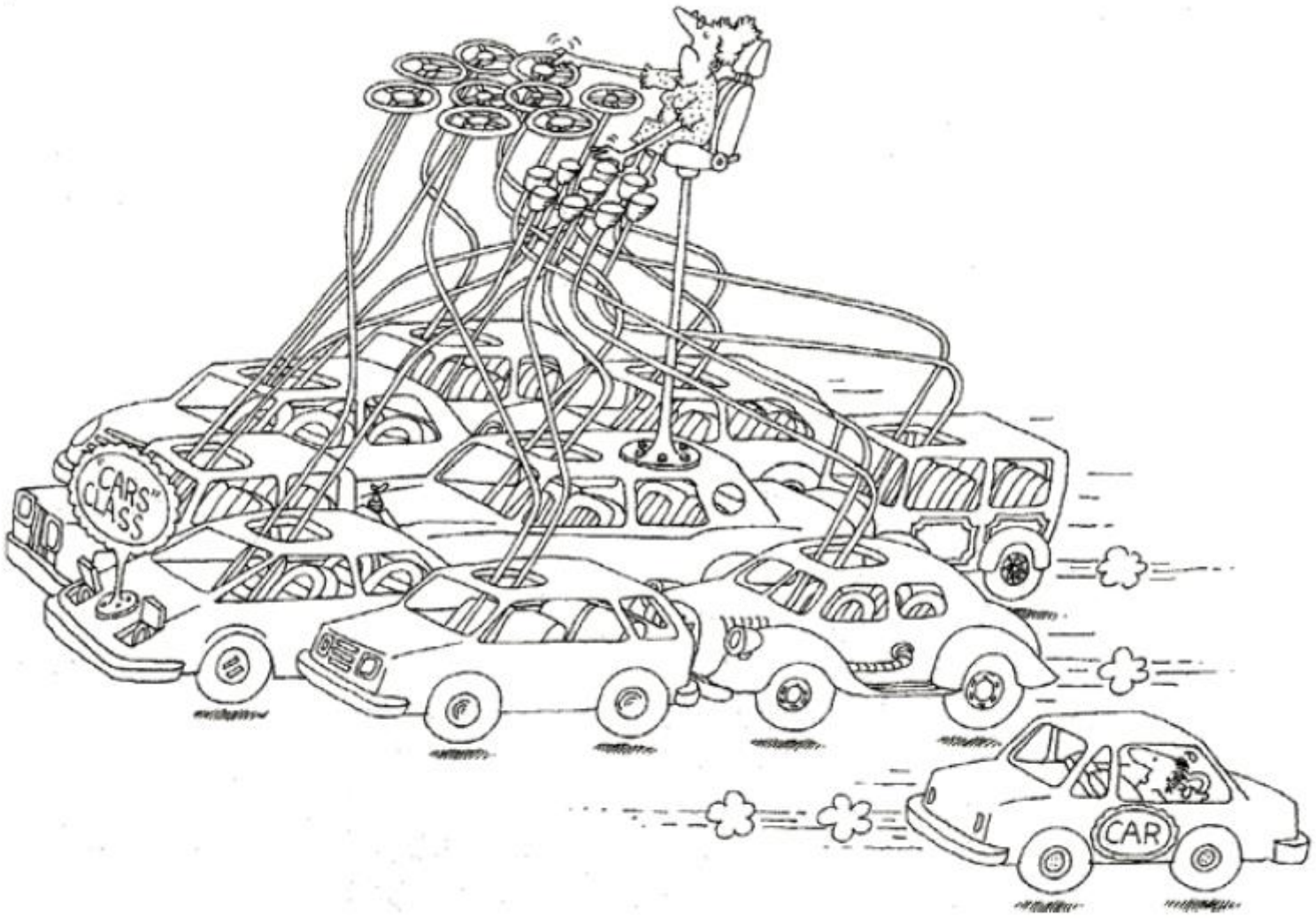


An object has state, exhibits some well-defined behavior, and has a unique identity.

ภาพจาก Object-Oriented Analysis and Design with Applications (Grady Booch และคณะ)

Class

- A class represents a set of objects that share a common structure and a common behavior.



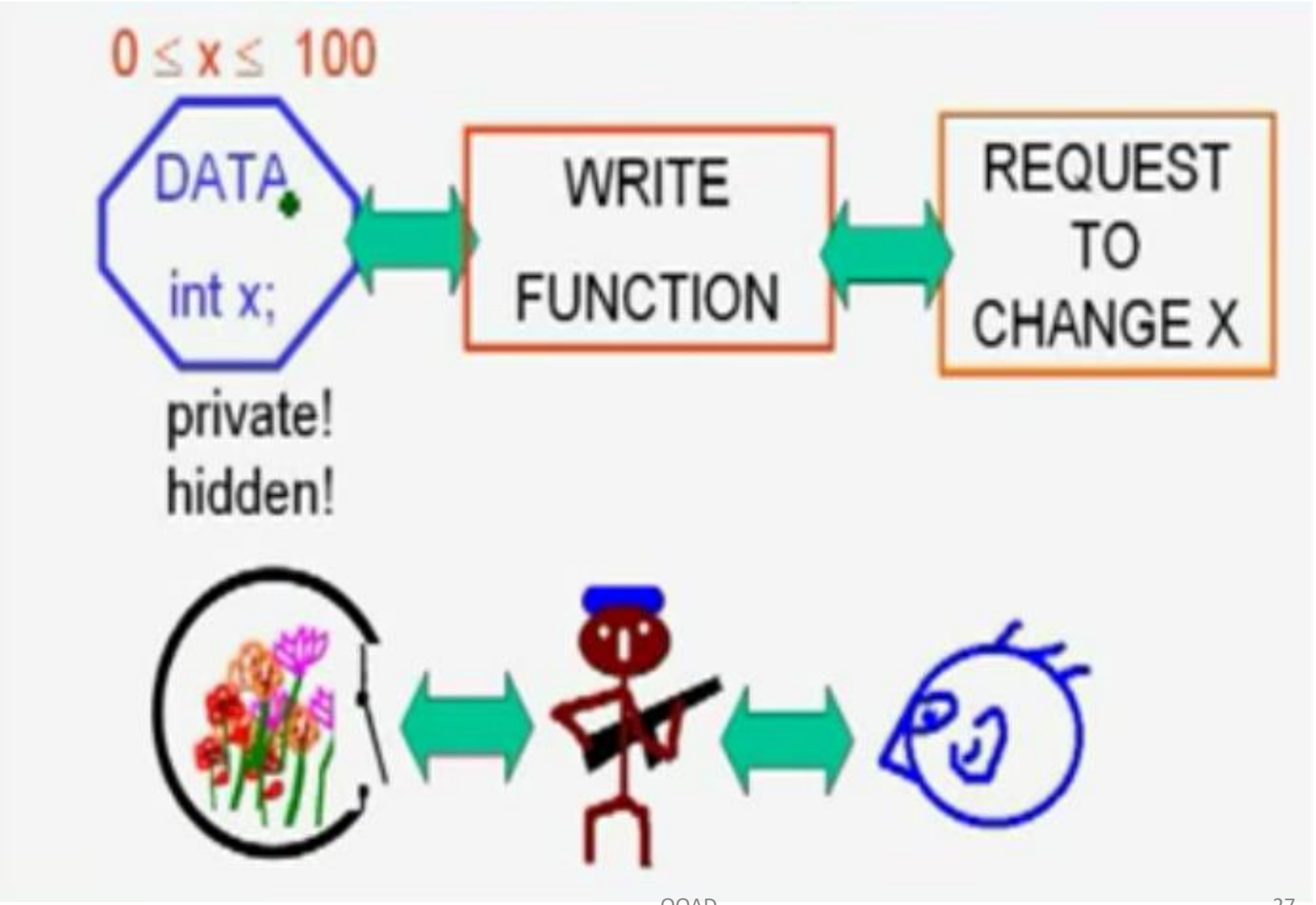
A class represents a set of objects that share a common structure and a common behavior.

ตัวอย่าง my Public Garden

ทุกคนมีสิทธิ์
เพราะสวนดอกไม้ถูกกำหนดเป็นสาธารณะสมบัติ

Solution 1: สร้างกำแพงสูงล้อมสวน

Solution 2: สร้างกำแพงแต่จ้างยามเฝ้าประตู


0 ≤ x ≤ 100
DATA
int x;
WRITE
FUNCTION
REQUEST
TO
CHANGE X
private!
hidden!

Accessing data via guardian function.

Write function

```
void modify_x(int newval)
{
 if( newval > 100 || newval < 0)
     {PRINT ERROR AND EXIT}
else
     x = newval;
}
```

Accessing data via guardian function.

Read function

```
int read_x()
{
   return x;
}
```

Encapsulation

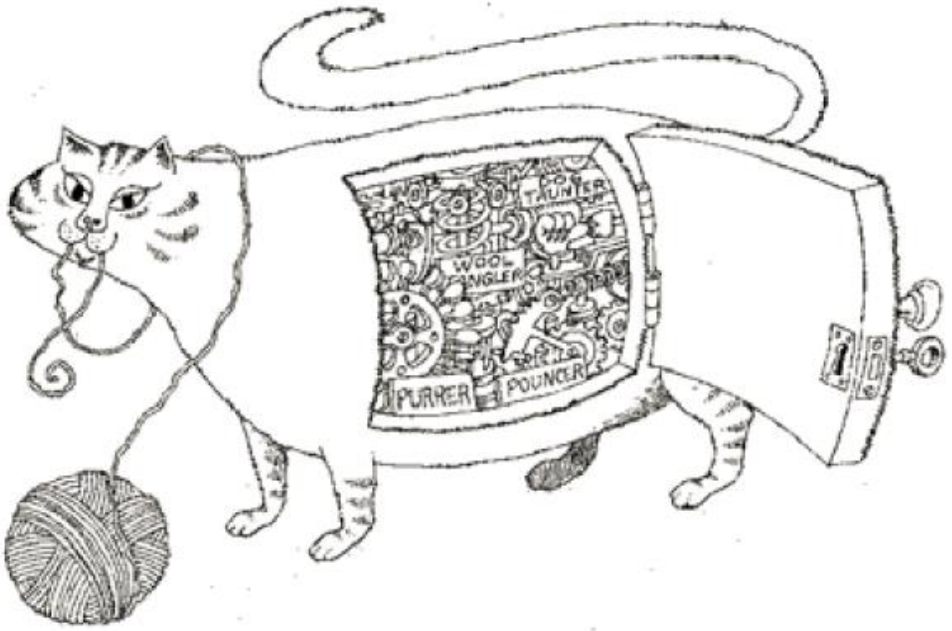
- FIRST LAW OF OOP: Data must be hidden (PRIVATE).
- Read access through read functions.
- Write access through write functions.

Encapsulation

- 4 possibilities to access data
  - read and write allowed
  - read only
  - write only
  - no access

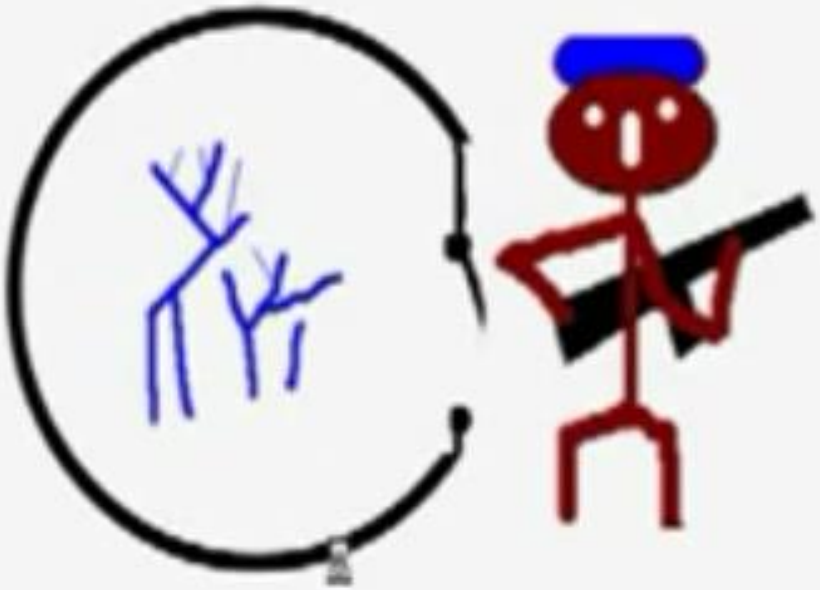
Encapsulation

- Encapsulation hides the details of the implementation of an object.



Objects initialization

จะทำอย่างไร ถ้าสวนมีแต่วัชชพืช



Improper Initialization

Initialization of Objects

- ฟังก์ชันพิเศษ เรียกว่า Constructor จะถูกเรียกโดยอัตโนมัติทุกครั้งที่เริ่มใช้งาน object
- เป็นจุดที่เหมาะสม ที่จะทำการ initialization

Resource Deallocation

- เมื่อจะเลิกใช้งาน Objects บางครั้งต้องมีการคืนทรัพยากรให้ระบบ จะมีฟังก์ชันที่ถูกเรียกใช้งานทุกครั้งคือ DESTRUCTOR
- ในบางภาษาจะไม่มี Destructor

Lifecycle of Object

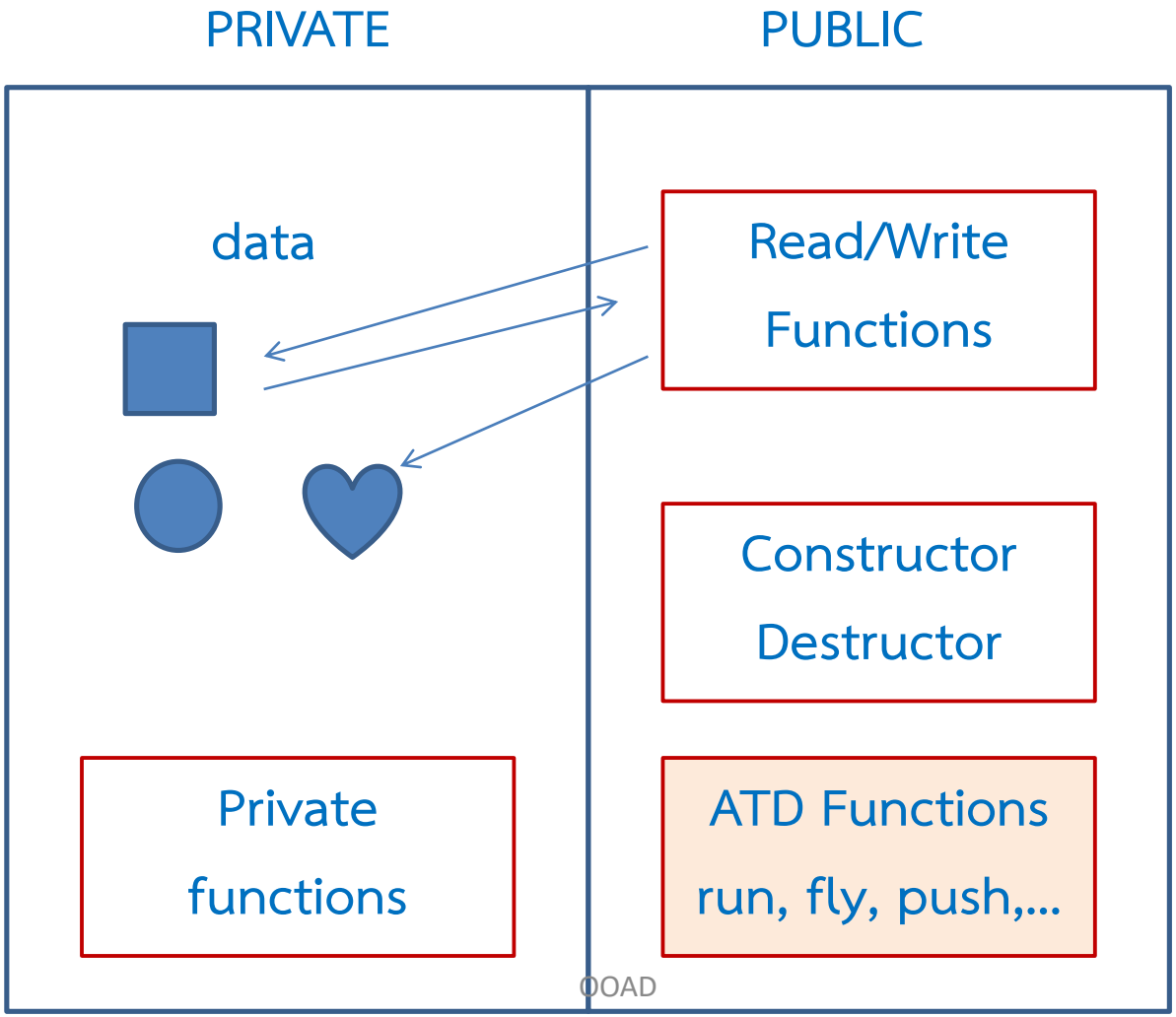
- Born healthy
  - ใช้ Constructors
- Lives safety
  - ใช้ read/write functions
- Dies cleanly
  - ใช้ Destructors

เกิดขึ้น

ตั้งอยู่

ดับไป

โครงสร้างของ class



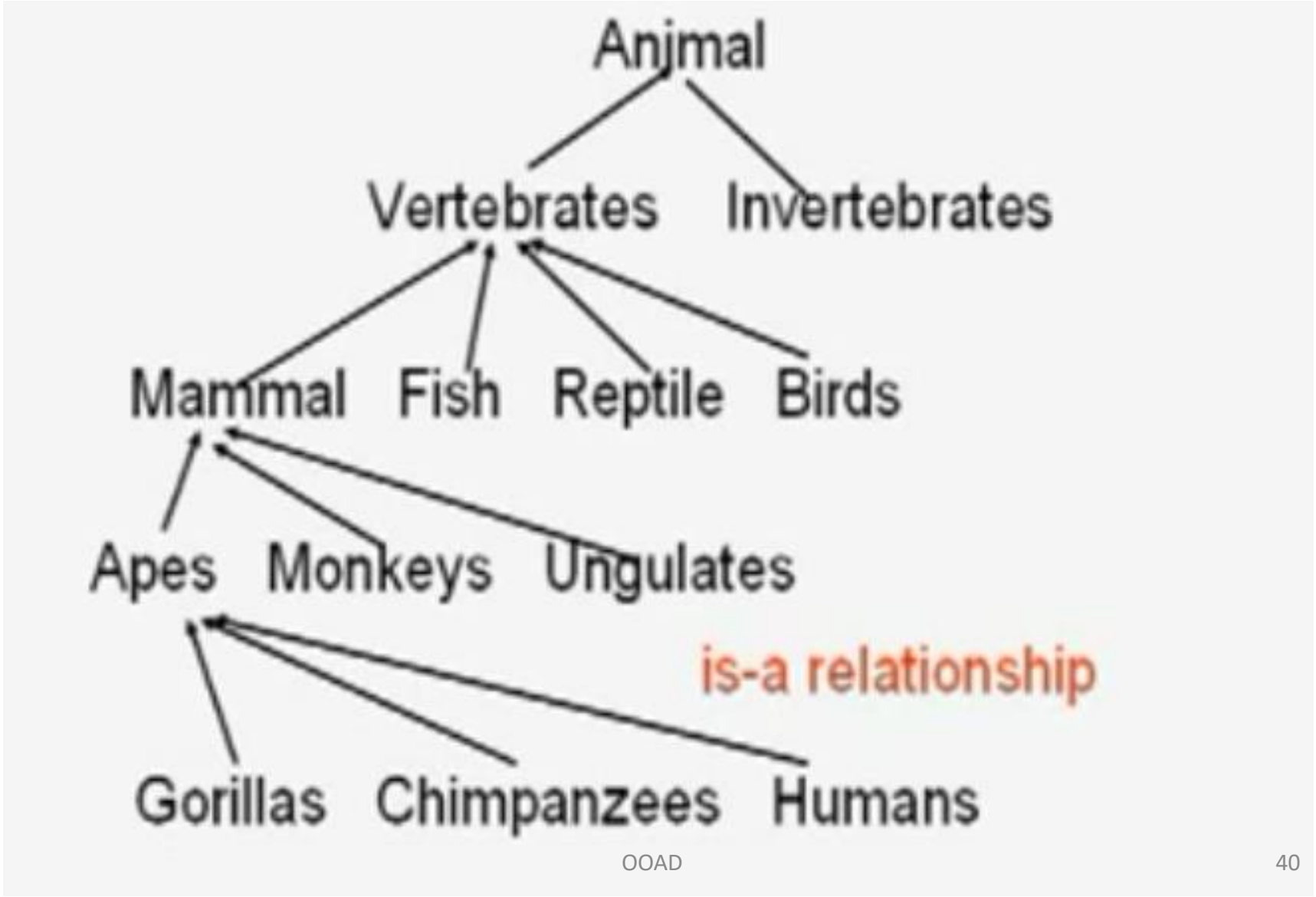
Code Reuse - Inheritance

สมชาย เขียน code ทำงานกับ stack ได้อย่างดีเยี่ยม โดยมีฟังก์ชัน push, pop อยู่มาวันหนึ่งสมชายลาออก และสมศรีต้องการเพิ่มฟังก์ชัน peek สมศรีจะทำอย่างไร

Subclass - Inheritance

- A subclass may inherit the structure and behavior of its superclass

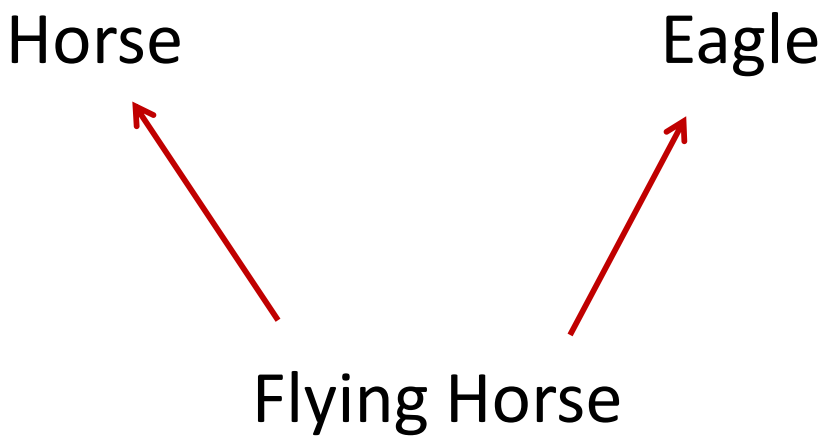
Inheritance Trees



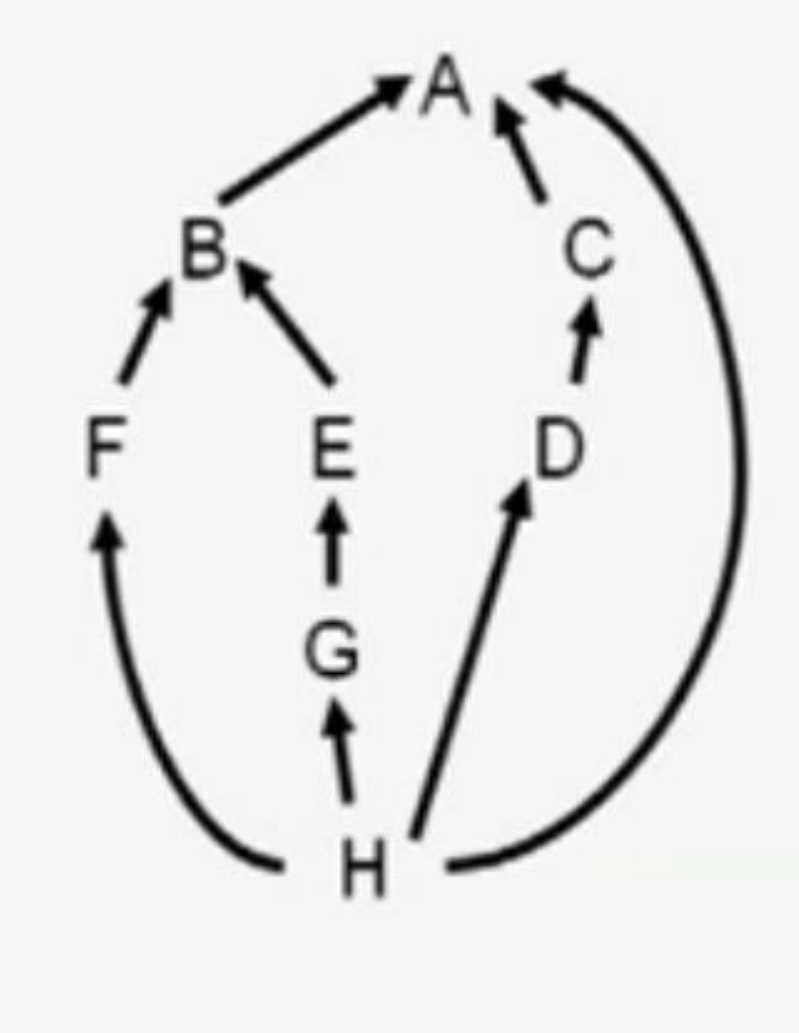
ศัพท์เฉพาะ

Not recommended		BEST
Superclass	Parent Class	Base Class
Subclass	Child Class	Derived Class

Multiple Inheritance



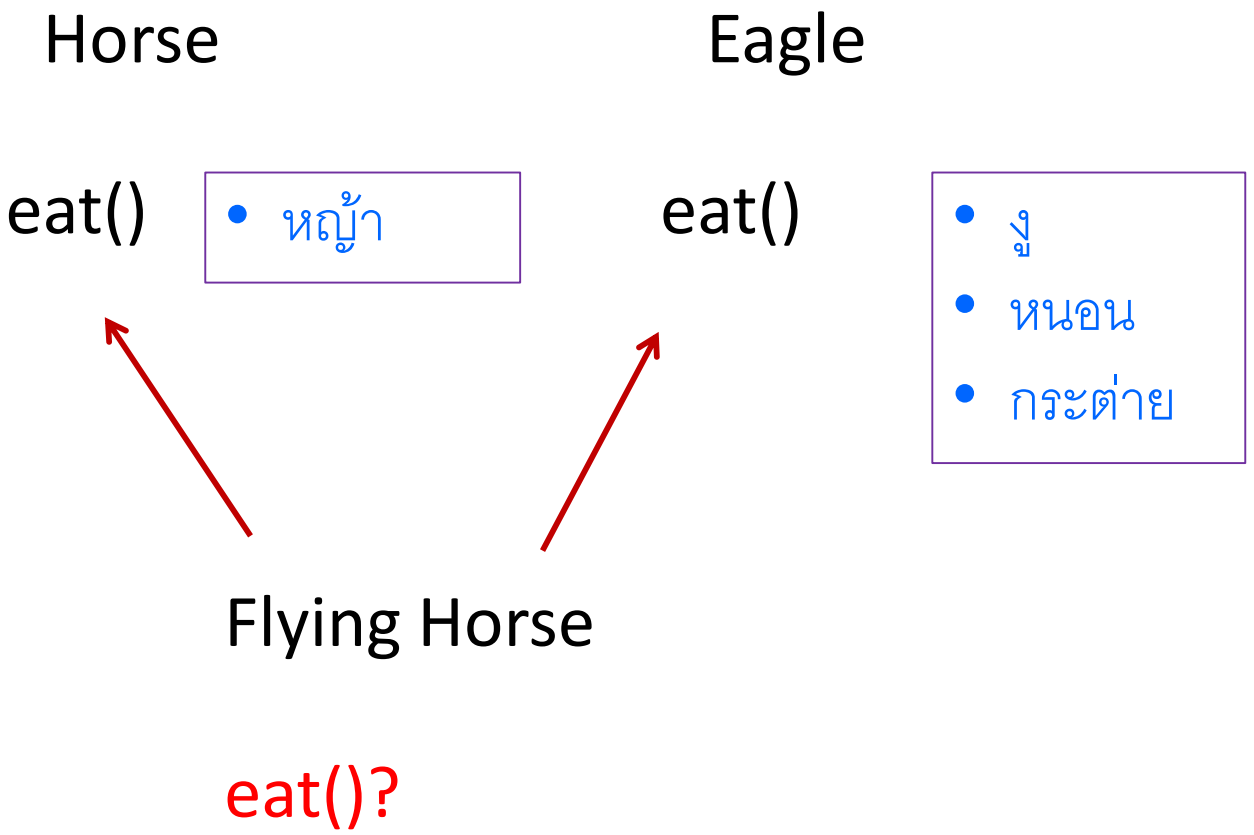
Graph Structure of Multiple Inheritance



- graphs มีรูปแบบที่ซับซ้อนกว่า trees
- บำรุงรักษาและแก้ bug ยาก
- เกิดความสับสนในการเรียกใช้ functions

ถ้าพบ bug เมื่อใช้งานคลาส H ???

Ambiguity in Multiple Inheritance

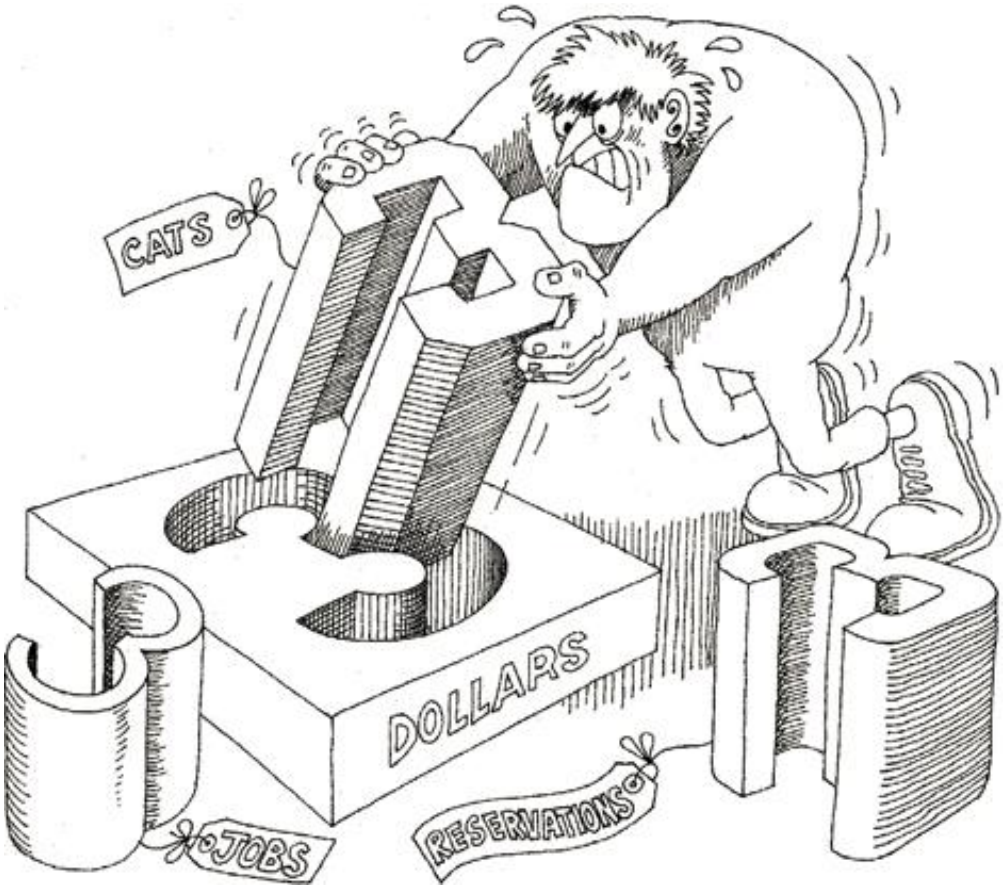


Modularity

- Modularity packages abstractions into discrete units.
  - Classes
  - Packages
  - Domains

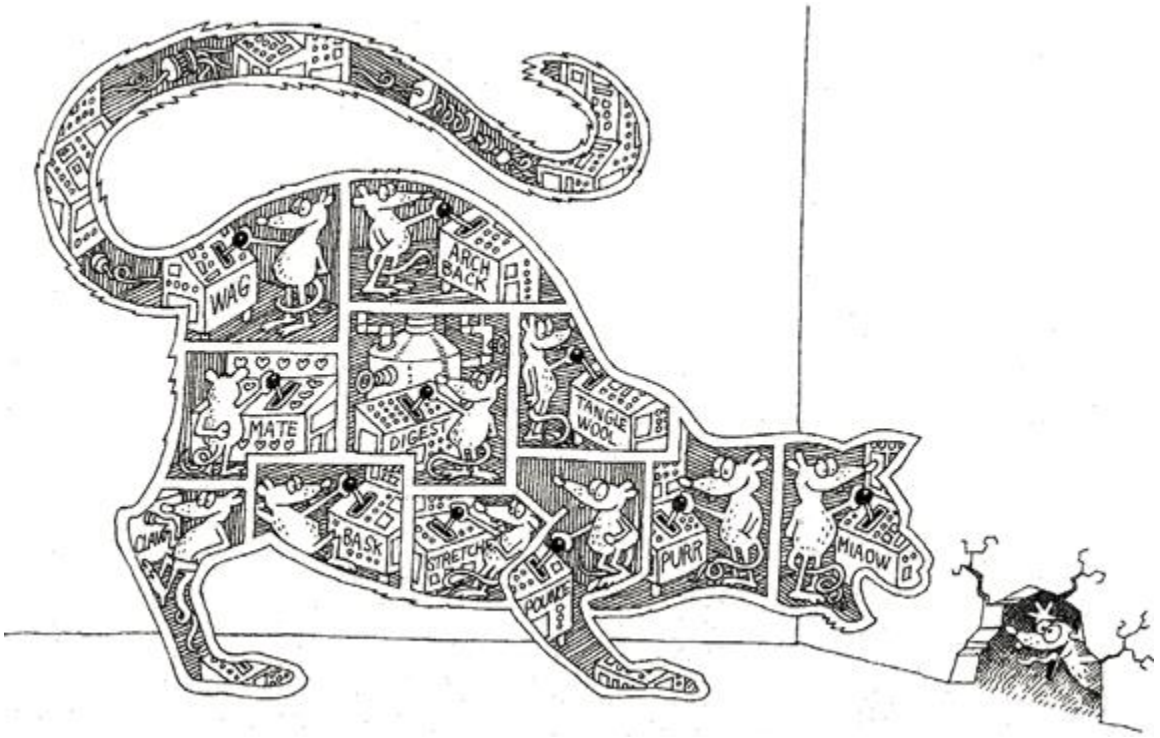
Strong Typing

- Strong typing prevents mixing of abstractions



Concurrency

- Concurrency allows different objects to act at the same time.



Strong typing

X = Y

จะยอมให้ใช้ operator = ได้ก็ต่อเมื่อ X และ Y เป็น object จาก class เดียวกันเท่านั้น

- Weak Typing ยอมให้ละเมิดกฎได้บางครั้ง เช่นภาษา C
- Untyped หรือ Dynamic Typing เช่น LISP, javascript

Persistence

- Persistence ช่วยให้เราสามารถเก็บ Object ทั้งก้อนไว้ใน media ต่างๆ เช่น hard disk เพื่อนำมาใช้ในได้อนาคต
- ถ้า base class มีความสามารถนี้แล้ว จะทำให้ derived class มีความสามารถตามไปด้วย
- เรียกอีกอย่างว่า Object serialization

ทบทวน

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Concurrency
- Strong Typing
- Persistence

Question???