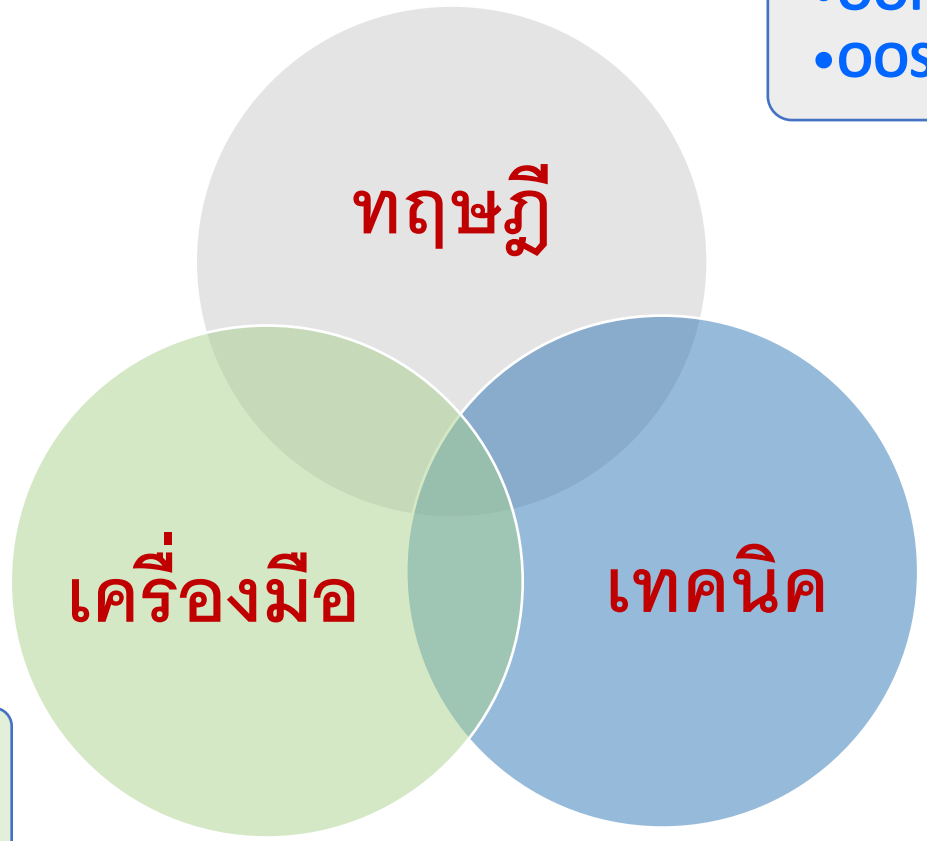
หน่วยที่ 1 การเขียนโปรแกรมเชิงวัตถุ

การเขียนโปรแกรมเชิงวัตถุ

- การพัฒนาโปรแกรม OOP
- ยุคต่างๆ ในการพัฒนาภาษา OOP
- วัตถุในโลกแห่งความเป็นจริง
- ภาษาโปรแกรมเชิงวัตถุ

มีอะไรให้เรียนรู้บ้าง

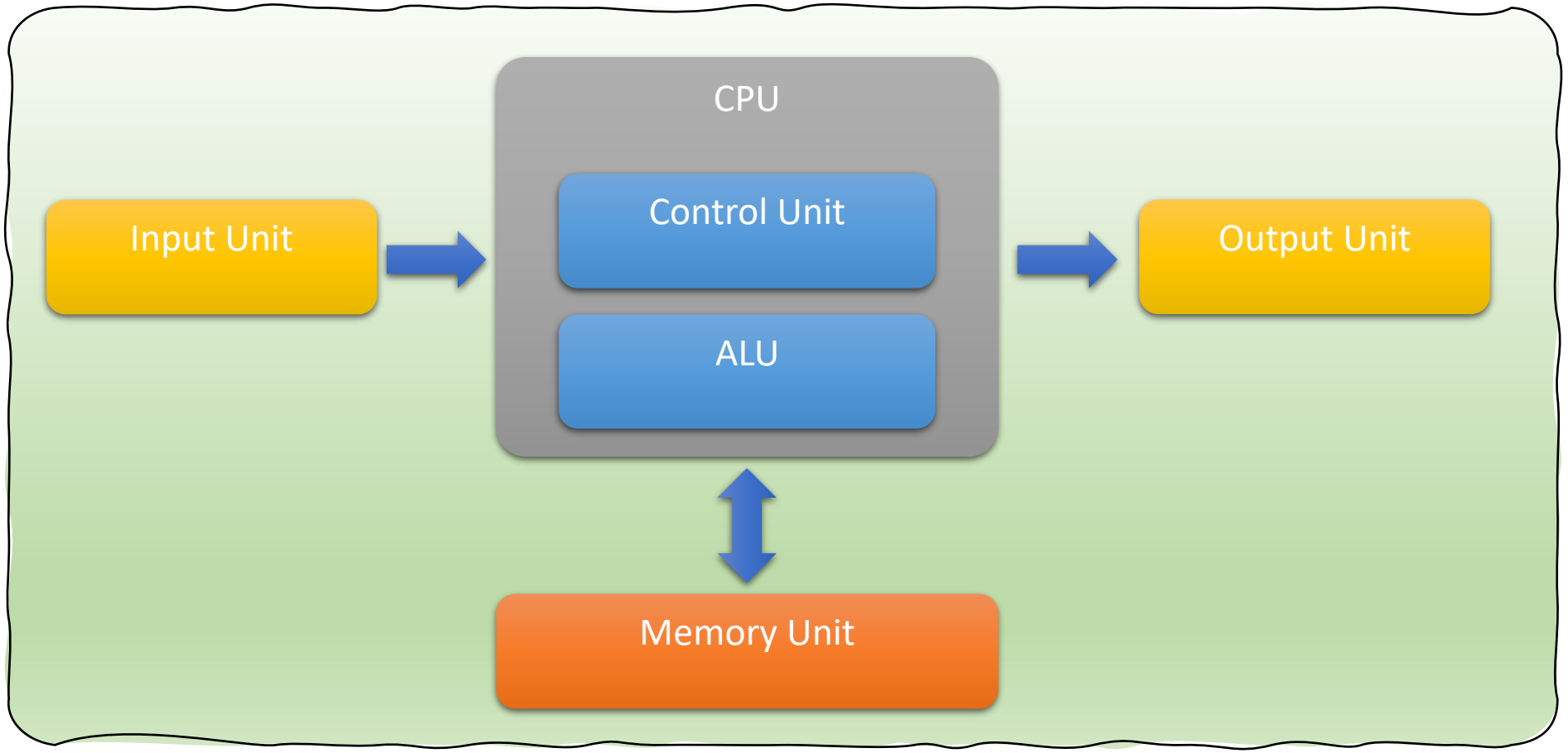
- OOAD (Analysis & Design)
- OOP (Programming)
- OOSE (Software Eng.)



- Graphics mode tools
- Text Mode tools

- Static diagram
- Dynamic Diagram

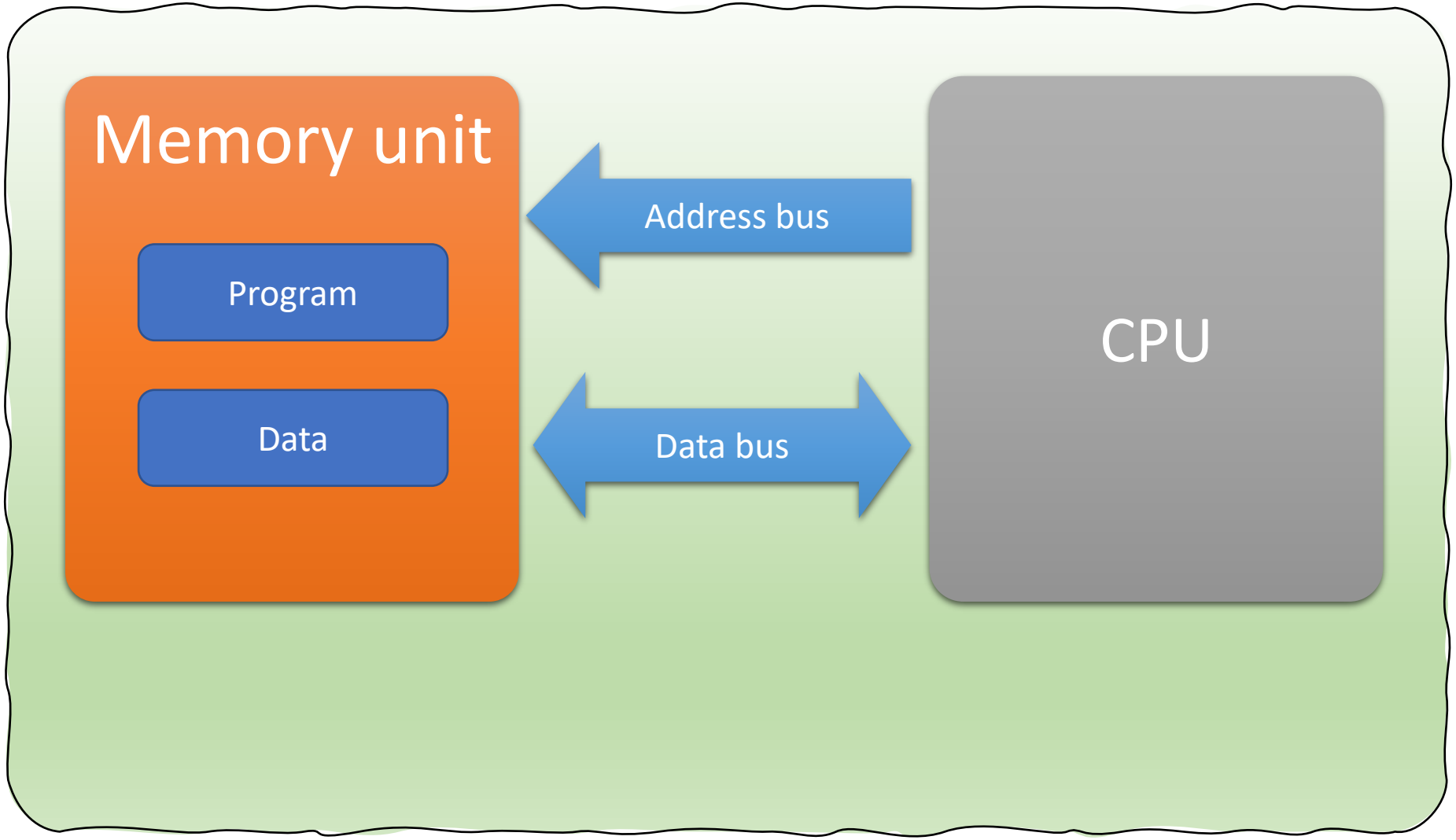
Computer Architecture (Review)



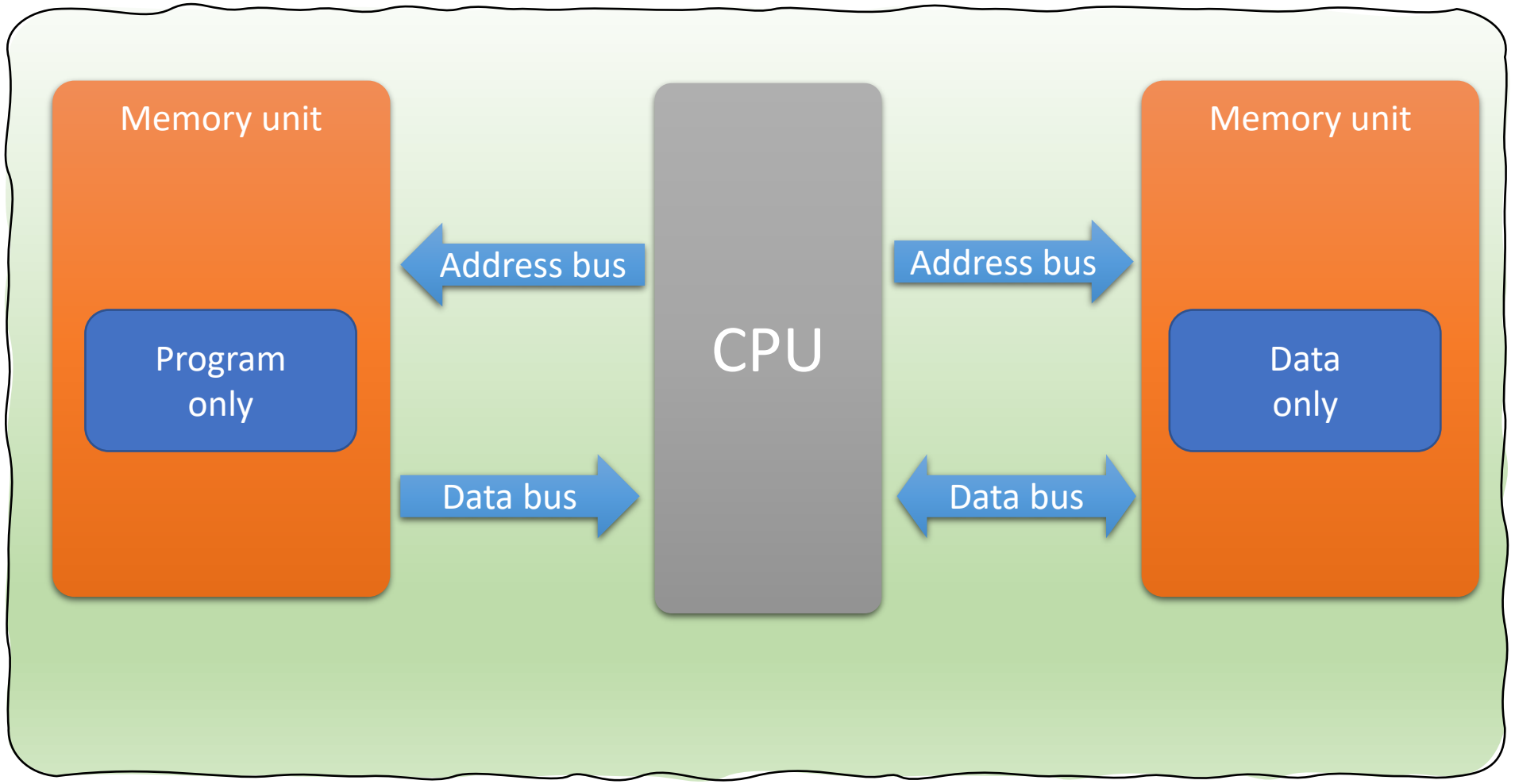
Von Neumann architecture

Harvard architecture

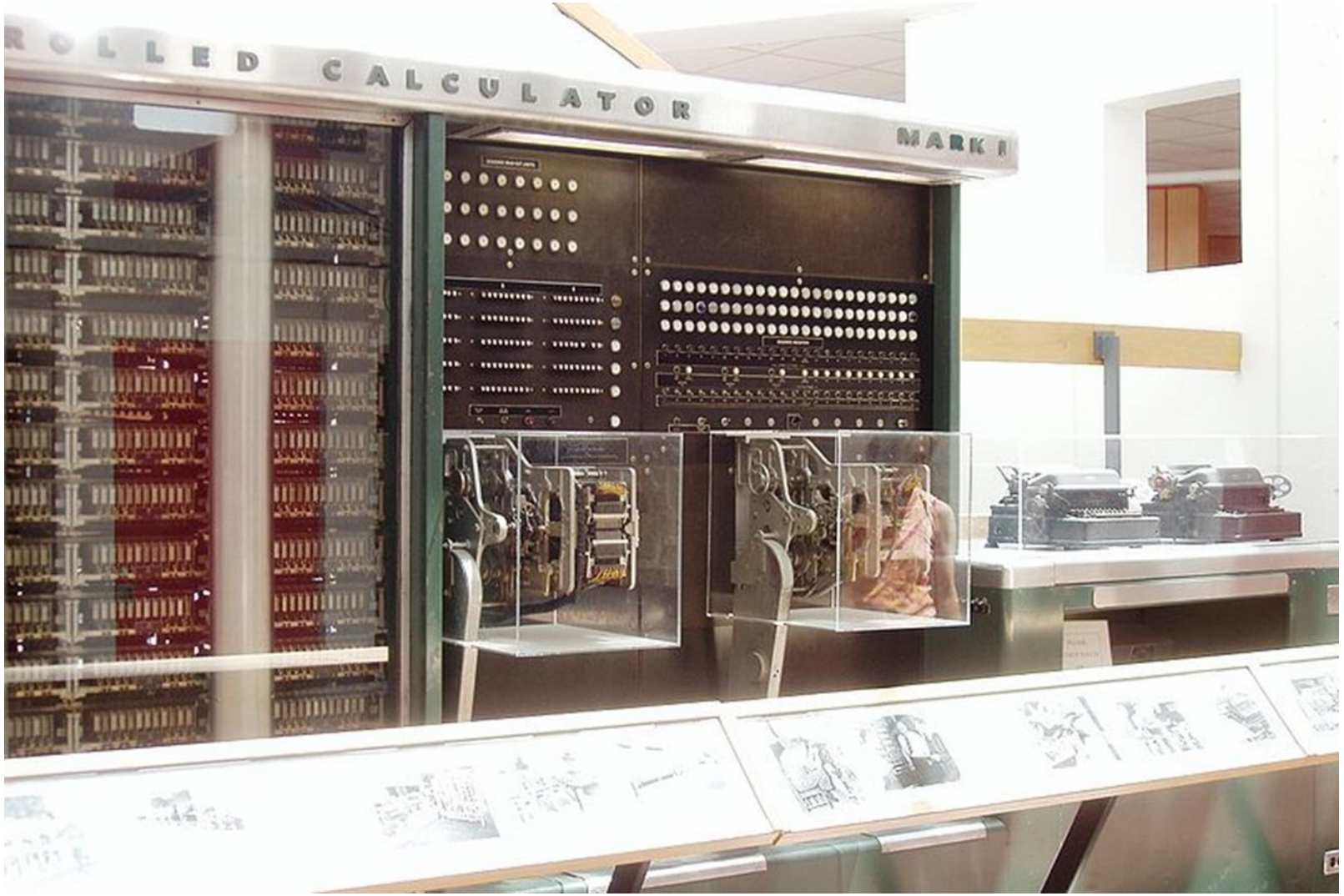
Von Neumann architecture



Harvard architecture



Harvard Mark I Computer



ยุคต่างๆ ในการพัฒนา software (1)

- ภาษาเครื่อง (Machine code)
  เขียนโดยใช้เลขฐาน 2 และ 16

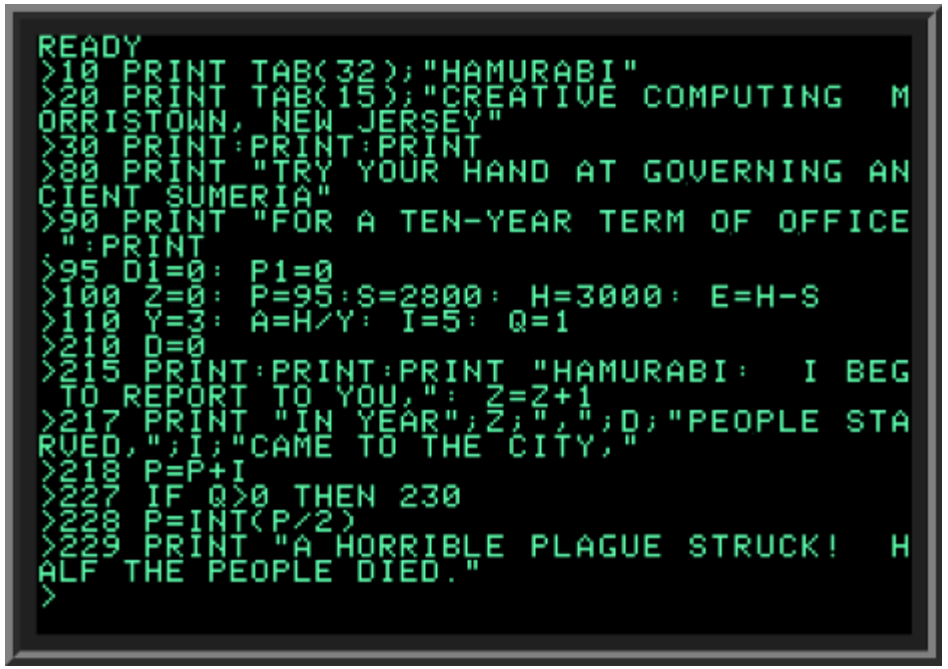
Flowchart --> psudocode -->
assembly code --> translate -->
input to mamory --> run

ยุคต่างๆ ในการพัฒนา software (2)

- ภาษา Assembly ใช้ opcode, mnemonic และสัญลักษณ์ต่างๆ แล้ว assemble เป็นภาษาเครื่อง

ยุคต่างๆ ในการพัฒนา software (3)

- ภาษาระดับสูง (High-level Language) เช่น Fortran, COBOL, BASIC เขียนด้วยภาษาที่เข้าใจง่าย
- เช่นคำสั่ง loop, if-else, function, for, do-while แล้วจึงใช้ compiler แปลเป็นภาษาเครื่อง



ยุคต่างๆ ในการพัฒนา software (4)

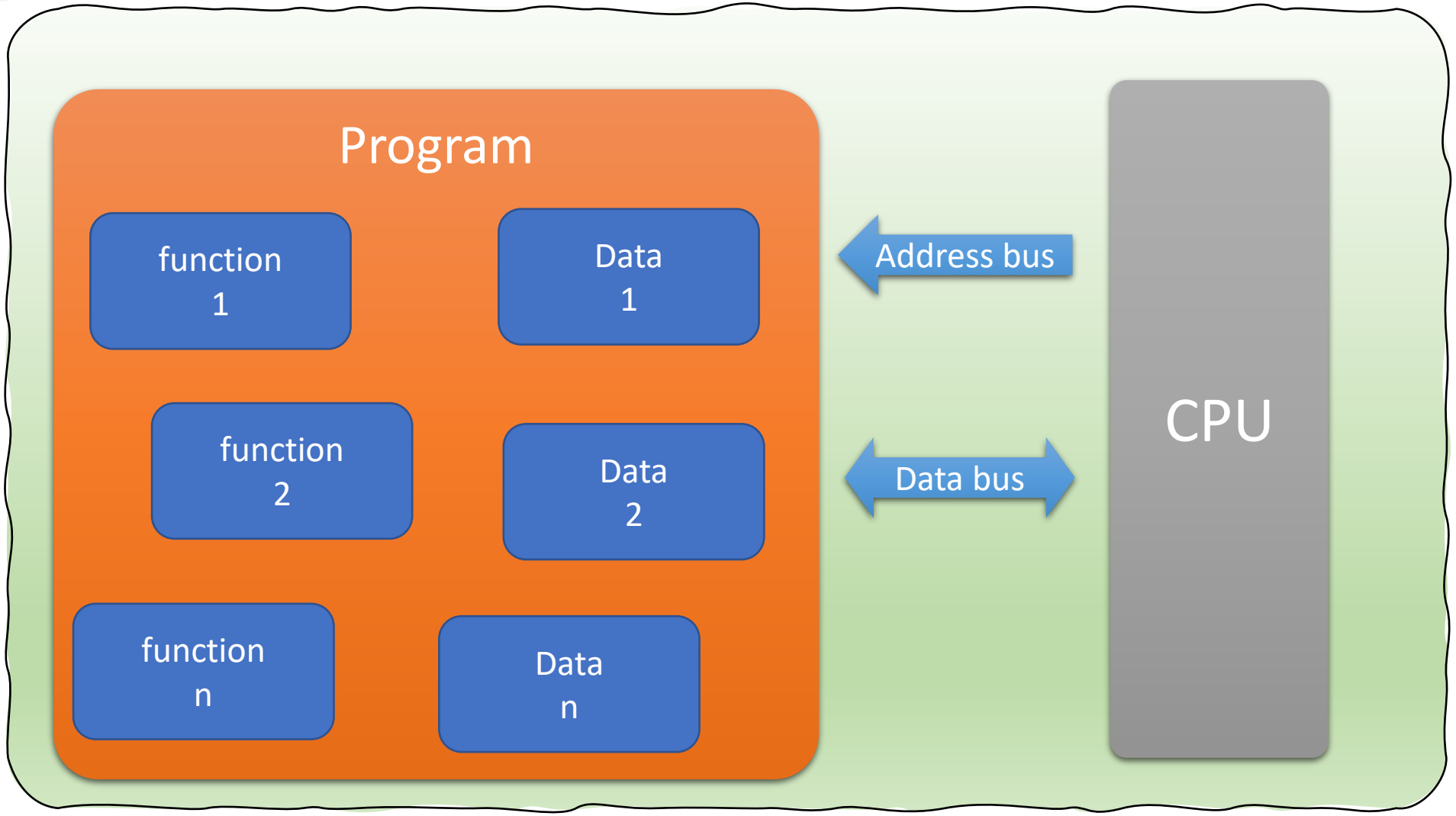
- ภาษาเชิงโครงสร้าง (Structured programming) เป็นภาษาระดับสูงที่เข้าใจได้ง่ายขึ้น เช่น Pascal, Ada ซึ่งจะแบ่งการทำงานยาวๆ ที่ซ้ำซ้อนออกเป็นบล็อกย่อยๆ เรียกว่า sub-task หรือ sub-routine
- ภาษาเชิงวัตถุ (Object-oriented programming) เป็นภาษาที่สร้างวัตถุที่อิสระต่อกันขึ้นมาในหน่วยความจำของคอมพิวเตอร์แล้วโปรแกรมให้ทำงานประสานสอดคล้องกัน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

ปัญหาที่พบในภาษาเชิงโครงสร้าง

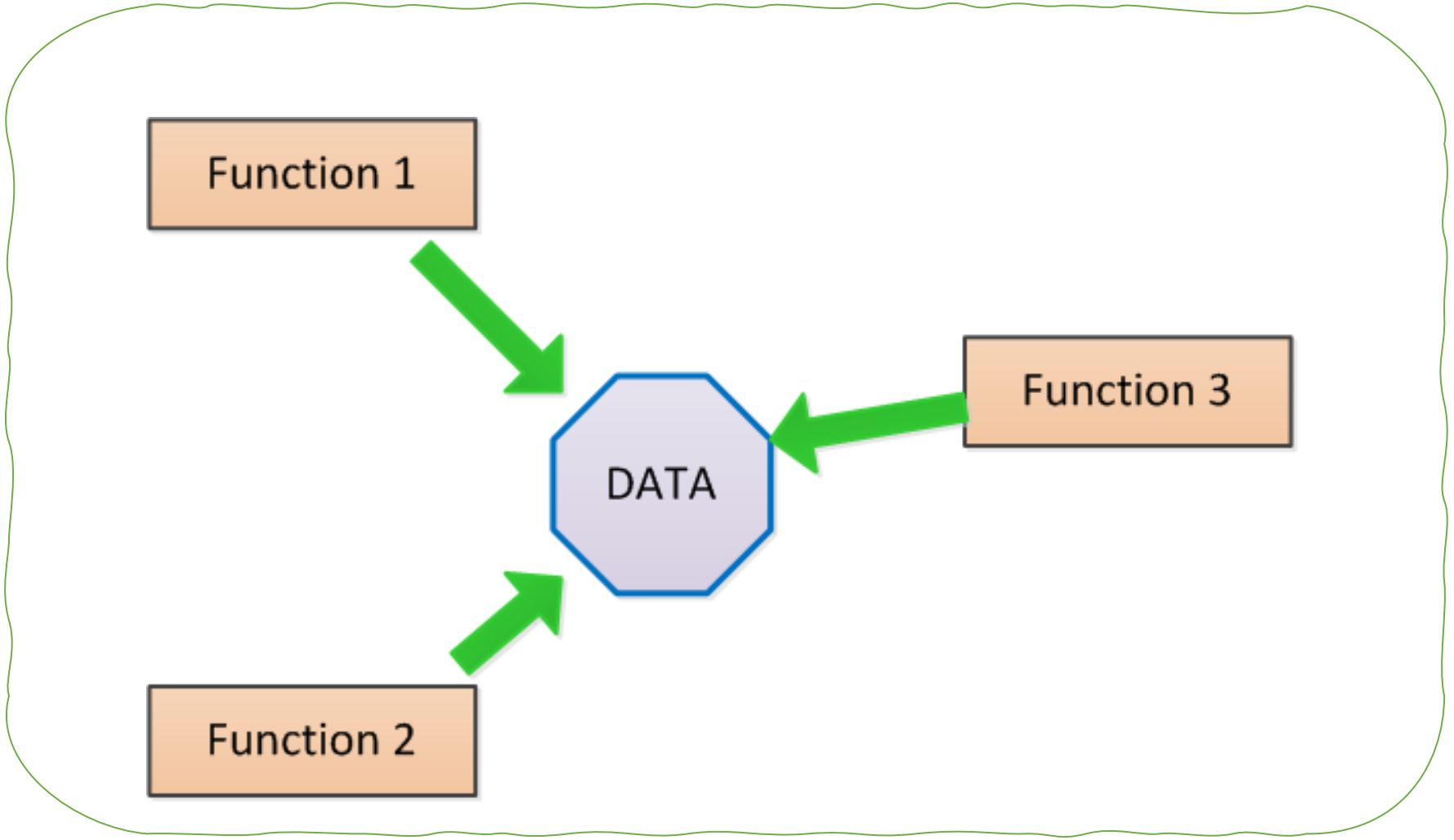
ในด้านความปลอดภัยของข้อมูล

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

ปัญหาที่พบในภาษาเชิงโครงสร้าง



ปัญหาที่พบในภาษาเชิงโครงสร้าง

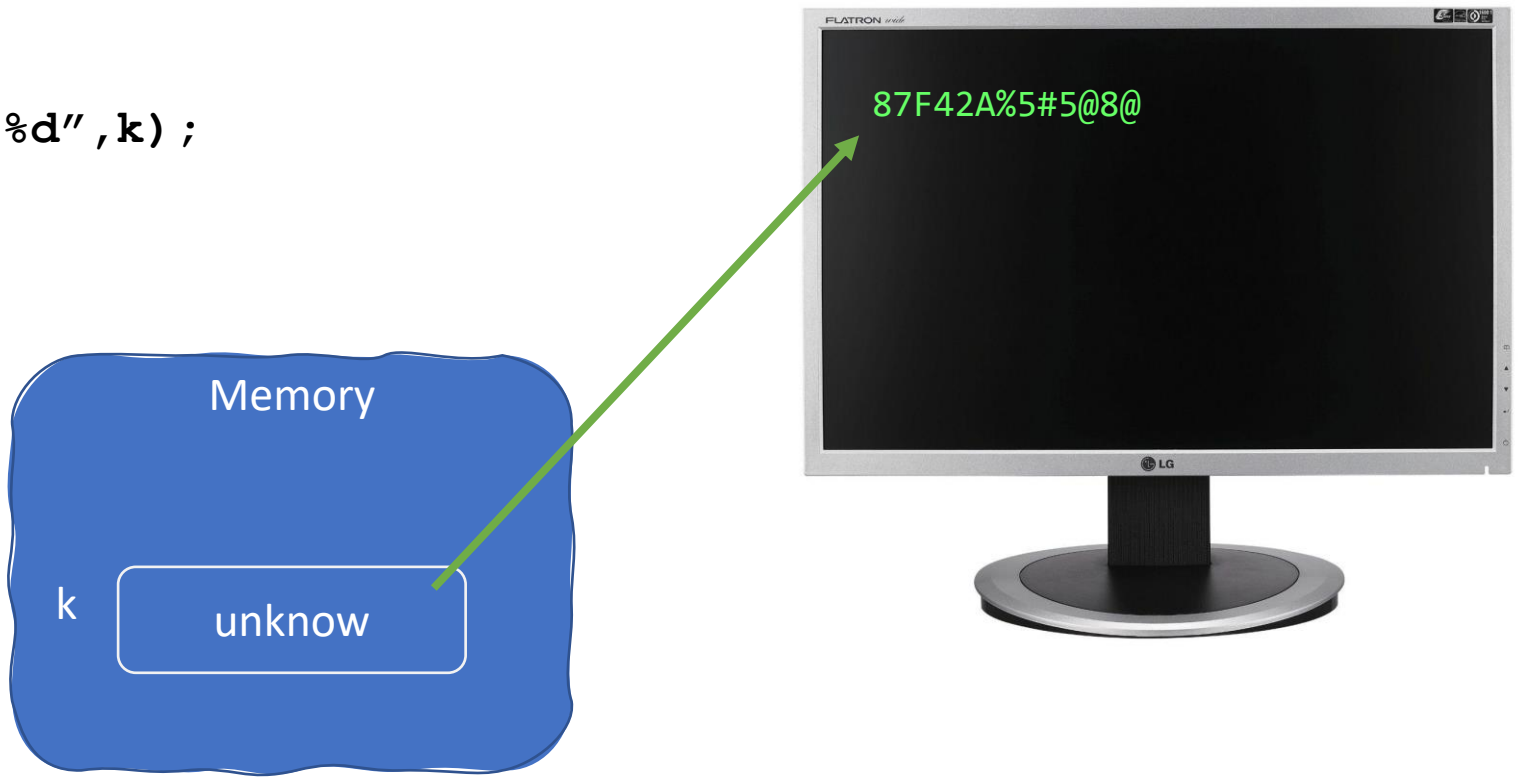


ปัญหาที่พบในภาษาเชิงโครงสร้าง

ข้อมูลที่ไม่ได้กำหนดค่าเริ่มต้น

```
int k;

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



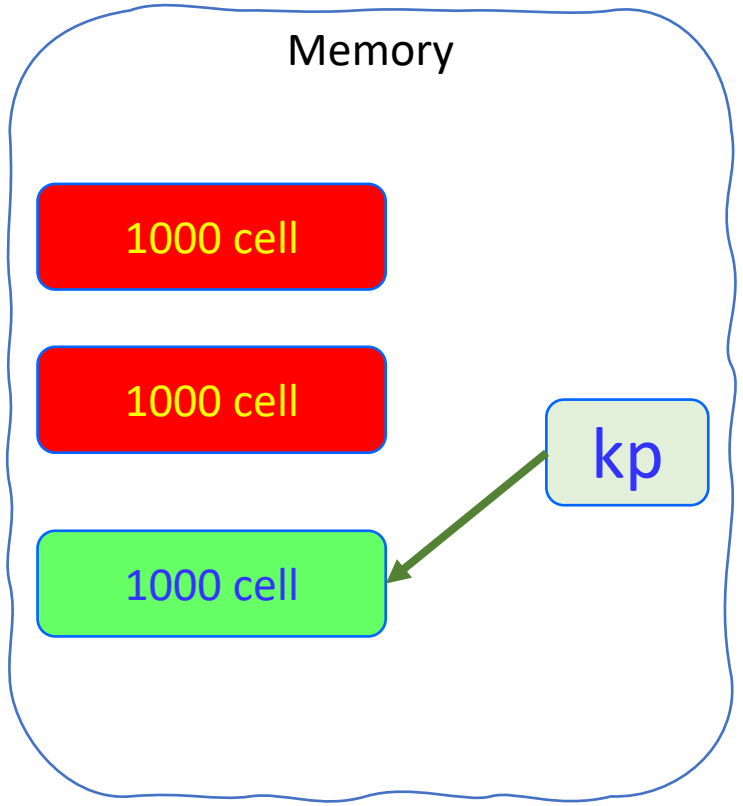
ปัญหาที่พบในภาษาเชิงโครงสร้าง

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: อาจจะเป็น ตัวแปรเดี่ยวหรือ Structure

Functions: Code ที่ใช้ในการจัดการข้อมูล

Structure programming

ใครเปิดปลา

กระป๋อง?

Tools

Data

Object-Oriented Programming

Data + Tools

“วัตถุ” มีเครื่องมือที่จะจัดการกับ data ในตัว

ยุคต่างๆ ในการพัฒนา ภาษา OOP

- SIMULA 1 (1962)
- SIMULA 67 (1967)

ภาษาซิมูลา เริ่มพัฒนาขึ้นประมาณปี พ.ศ. 2503 ที่ "ศูนย์คอมพิวเตอร์นอร์เวย์" (Norwegian Computing Center) ที่นครออสโล

โดย โอล-โยฮาน ดาห์ล (Ole-John Dahl) และ คริสเตน ไนกาอาร์ด (Kristen Nygaard)

ภาษา SIMULA อย่างง่าย

```
Begin
End;
```

```
Begin
OutText ("Hello World!");
Outimage;
End;
```

ศึกษาเพิ่มเติมได้จาก INTRODUCTION TO OOP IN SIMULA

http://staff.um.edu.mt/jskl1/talk.html

ยุคต่างๆ ในการพัฒนาภาษา OOP (ต่อ)

- Smalltalk (1970s)

ภาษาสมอลล์ทอล์ก (Smalltalk) เป็นภาษาโปรแกรมเชิงวัตถุที่ได้ออกแบบในปี ค.ศ. 1970 ที่ Xerox PARC โดย อลันด์ เคย์ (ผู้ริเริ่มใช้คำเรียกว่า Object-oriented) Dan Ingalls, Ted Kaehler, Adele Goldberg และคนอื่น ๆ

ภาษาสมอลทอล์คยังคงมีการพัฒนาอย่างต่อเนื่อง และมีชุมชนผู้ใช้ที่เหนียวแน่น ภาษาสมอลล์ทอล์กเป็นภาษาที่มีกระบวนการจำแนกชนิดแบบยืดหยุ่น (dynamic)

ตัวอย่าง ภาษา Smalltalk

```
(x < y) ifTrue: [
 max := y. i := j
]
ifFalse: [
max := x. i := k
]
```

ยุคต่างๆ ในการพัฒนาภาษา OOP (ต่อ)

- C++ (1983)

ภาษาซีพลัสพลัส (อังกฤษ: C++) เป็นภาษาโปรแกรมคอมพิวเตอร์อเนกประสงค์ มีโครงสร้างภาษาที่มีการจัดชนิดข้อมูลแบบสแตติก (statically typed) และสนับสนุนรูปแบบการเขียนโปรแกรมที่หลากหลาย (multi-paradigm language) ได้แก่ การโปรแกรมเชิงกระบวนคำสั่ง, การนิยามข้อมูล, การโปรแกรมเชิงวัตถุ, และการโปรแกรมแบบเจเนริก (generic programming) ภาษาซีพลัสพลัสเป็นภาษาโปรแกรมเชิงพาณิชย์ที่นิยมมากภาษาหนึ่งนับตั้งแต่ช่วงทศวรรษ 1990

C++ (ต่อ)

- เบียเนอ สเดราสดร็อบ (Bjarne Stroustrup) จากเบลล์แล็บส์ (Bell Labs) เป็นผู้พัฒนาภาษาซีพลัสพลัส ในปี ค.ศ. 1983 เดิมใช้ชื่อ “C with classes”
- สิ่งที่พัฒนาขึ้นเพิ่มเติม ได้แก่ เวอร์ชวลฟังก์ชัน การโอเวอร์โหลดโอเปอเรเตอร์ การสืบทอดหลายสาย เทมเพลต และการจัดการเอกเซพชัน
- มาตรฐานของภาษาซีพลัสพลัสได้รับการรับรองในปี ค.ศ. 1998 เป็นมาตรฐาน ISO/IEC 14882:1998

ยุคต่างๆ ในการพัฒนาภาษา OOP (ต่อ)

- Java programming language (January 1991)
- ภาษาจาวา (อังกฤษ: Java programming language) เป็นภาษาโปรแกรมเชิงวัตถุ (อังกฤษ: Object Oriented Programming)
- พัฒนาโดย เจมส์ กอสลิง และวิศวกรคนอื่นๆ ที่ ซัน ไมโครซิสเต็มส์
- ภาษาจาวาถูกพัฒนาขึ้นในปี พ.ศ. 2534 (ค.ศ. 1991) โดยเป็นส่วนหนึ่งของ โครงการกรีน (the Green Project) และสำเร็จออกสู่สาธารณะในปี พ.ศ. 2538 (ค.ศ. 1995)
- ภาษานี้มีจุดประสงค์เพื่อใช้แทนภาษาซีพลัสพลัส (C++) เพื่อใช้กับ WWW
- รูปแบบที่เพิ่มเติมคล้ายกับภาษาอ็อบเจกต์ทีฟซี (Objective-C)
- เดิมภาษานี้เรียกว่า ภาษาโอ๊ก (Oak) ซึ่งตั้งชื่อตามต้นโอ๊กใกล้ที่ทำงานของ เจมส์ กอสลิง แต่ว่ามีปัญหาทางลิขสิทธิ์ จึงเปลี่ยนไปใช้ชื่อ "จาวา" ซึ่งเป็นชื่อกาแฟแทน

ยุคต่างๆ ในการพัฒนาภาษา OOP (ต่อ)

- ภาษาซีชาร์ป (C# Programming Language) (January 1999)
  - เป็นภาษาโปรแกรมแบบหลายโมเดล
  - ใช้ระบบชนิดข้อมูลแบบรัดกุม (strong typing)
  - สนับสนุนการเขียนโปรแกรมเชิงคำสั่ง
  - การเขียนโปรแกรมเชิงประกาศ
  - การเขียนโปรแกรมเชิงฟังก์ชัน
  - การเขียนโปรแกรมเชิงกระบวนการ
  - การเขียนโปรแกรมเชิงวัตถุ (แบบคลาส)
  - การเขียนโปรแกรมเชิงส่วนประกอบ

ภาษาซีชาร์ป (C#)

- พัฒนาเริ่มแรกโดยบริษัทไมโครซอฟท์เพื่อทำงานบนดอตเน็ตเฟรมเวิร์ก
- มีแอนเดอร์ เฮลส์เบิร์ก (Anders Hejlsberg) เป็นหัวหน้าโครงการ
- มีรากฐานมาจากภาษาซีพลัสพลัสและภาษาอื่นๆ (โดยเฉพาะภาษาเดลไฟและจาวา)
- โดยมีจุดมุ่งหมายให้เป็นภาษาสมัยใหม่ที่ไม่ซับซ้อน ใช้งานได้ทั่วไป (general-purpose) และเป็นเชิงวัตถุเป็นหลัก

ยุคต่างๆ ในการพัฒนาภาษา OOP

- ภาษาโปรแกรมอื่นๆ ที่รองรับ OOP
  - ฯลฯ

ข้อดีของภาษา OOP

- Flexible : มีความยืดหยุ่นสูง ภาษาที่รองรับ OOP มักจะสามารถสร้าง Library ให้ทำงานบนระบบต่างๆ ได้โดยง่าย สามารถทำงานข้ามเครื่อง หรือข้ามภาษาได้ เช่นใน Java หรือ .NET
- Powerful : เนื่องจากทำงานในเชิงวัตถุ หลายๆ ภาษา มักมีการเข้าถึงสมาชิกของวัตถุคล้ายๆ กัน ทำให้เรียนรู้ได้เร็ว สร้างงานได้มากกว่า
- Easier to use : โดยทั่วไป ภาษา OOP จะมีรายงานข้อผิดพลาดในขณะคอมไพล์และมีการตรวจจับข้อผิดพลาดขณะทำงานได้ดีกว่า ทำให้เสียเวลาในการหาข้อผิดพลาดน้อยกว่า

ข้อดีของภาษา OOP

- Visually oriented : เนื่องจากเป็นภาษายุคใหม่ ที่เน้น design by reuse ทำให้นักพัฒนาภาษา มีเวลามากพอที่จะสร้าง IDE ที่ฉลาดและใช้งานง่าย ดังนั้น การออกแบบ UI ทั้งหลาย จึงอยู่ในแบบเสมือนจริงมากกว่า
- Internet-friendly : ภาษา OOP ที่นิยมใช้ มักมีการพัฒนา Library ที่รองรับ internet ทำให้สามารถใช้ความสามารถดังกล่าวได้อย่างง่ายดายและรวดเร็ว

ข้อเสียของ OOP

- ใช้เนื้อที่ในการทำงานมาก เนื่องจากมีชื่อฟังก์ชันที่ซ้ำๆ กันจากหลายคลาส (โดยเฉพาะการสืบทอดจากคลาสสู่คลาส) ทำให้ระบบต้องมีการจัดเก็บตาราง virtual functions
- ในบางภาษาจะมีความรัดกุมในการเขียนมาก เมื่อต้องทำงานขนาดใหญ่อาจพัฒนาได้ช้า เนื่องจากขาดแคลน programmer ที่เชี่ยวชาญภาษานั้นๆ
- ในบางภาษา อาจใช้ virtual machine ทำให้ต้องใช้ประสิทธิภาพของเครื่องสูงกว่าความเป็นจริง (มากกว่า native)

วัตถุ (object) ในโลกแห่งความเป็นจริง

- *Object Orientation* คืออะไร?
  - ในโลกของเรามี Objects ต่างๆ มากมาย

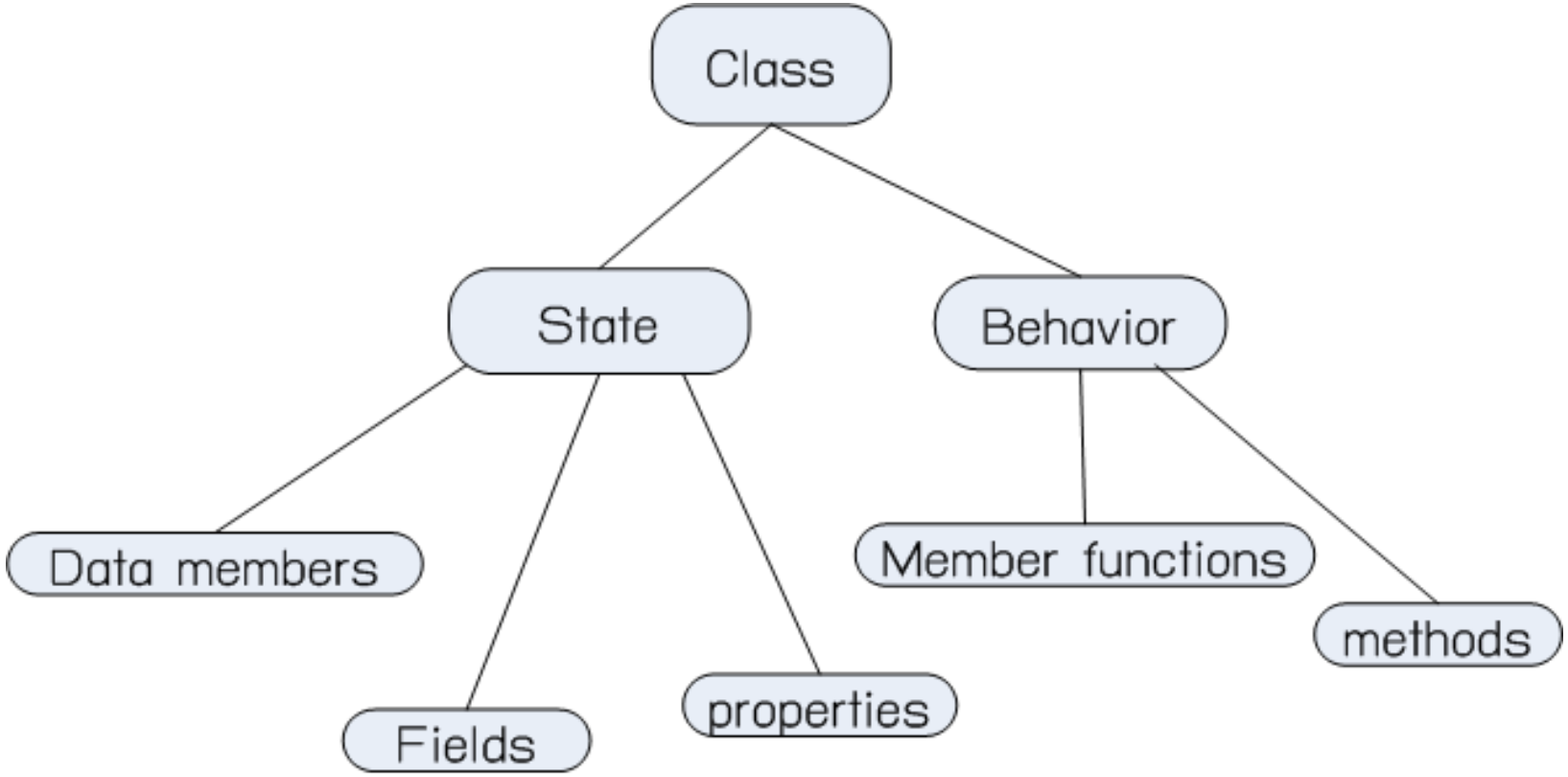
  สิ่งที่เกิดขึ้นจาก Objects ต่างๆ ได้แก่
    - กิจกรรม (Activities)
    - ความเคลื่อนไหว (Movement)
    - การกระทำ (Actions)

    เช่น ผู้หญิงคนหนึ่งรับประทานอาหาร เด็กเล่นกับแมว เป็นต้น

- ในความเป็นจริง รอบๆ ตัวเรามีวัตถุ (Object) ต่างๆ มากมาย
- วัตถุที่สามารถมองเห็นได้และจับต้องได้ (Tangible Objects)
  - อาทิเช่น โต๊ะ รถยนต์ คอมพิวเตอร์ คน สัตว์ ฯลฯ
- วัตถุที่มีอยู่จริงแต่ไม่สามารถจับต้องได้ (Intangible Objects)
  - อาทิเช่น กฎหมาย เวลา หรือความรู้ วิชาการต่างๆ ฯลฯ
- วัตถุต่างๆ จะมีความสัมพันธ์ (Relationship : static) และปฏิสัมพันธ์กัน (Interaction : Dynamic) กันในโดเมน (Domain) ที่เราสนใจ

วัตถุ (Objects) ในการเขียนโปรแกรมเชิงวัตถุ

เราจะไม่สร้าง object โดยตรง แต่จะทำผ่านกระบวนการสร้างคลาส แล้วนำคลาสมาสร้าง object



การสร้างแบบจำลอง

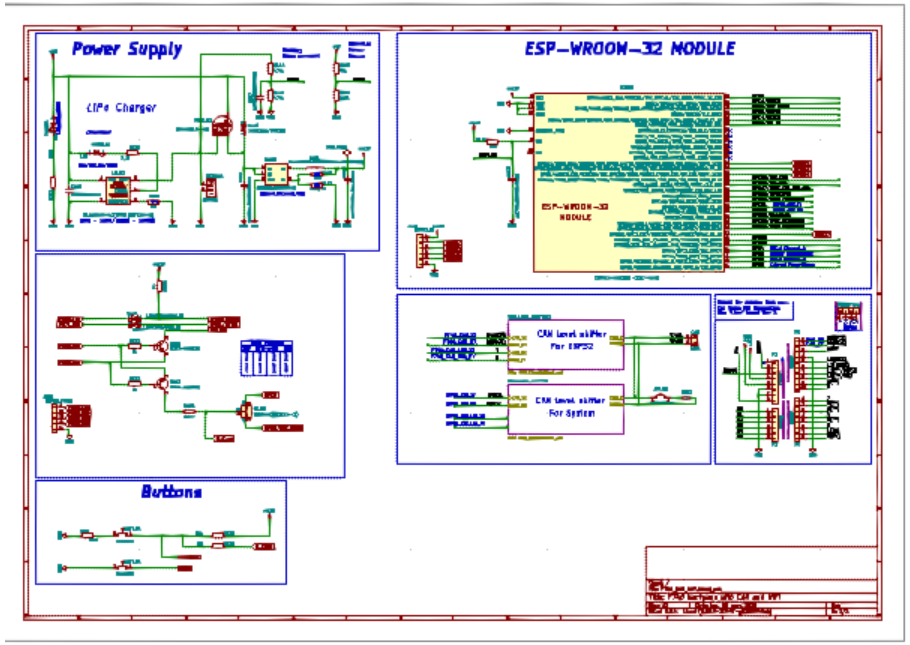
Real world object

Abstractions

Model

การสร้างวัตถุจากแบบจำลอง

Model

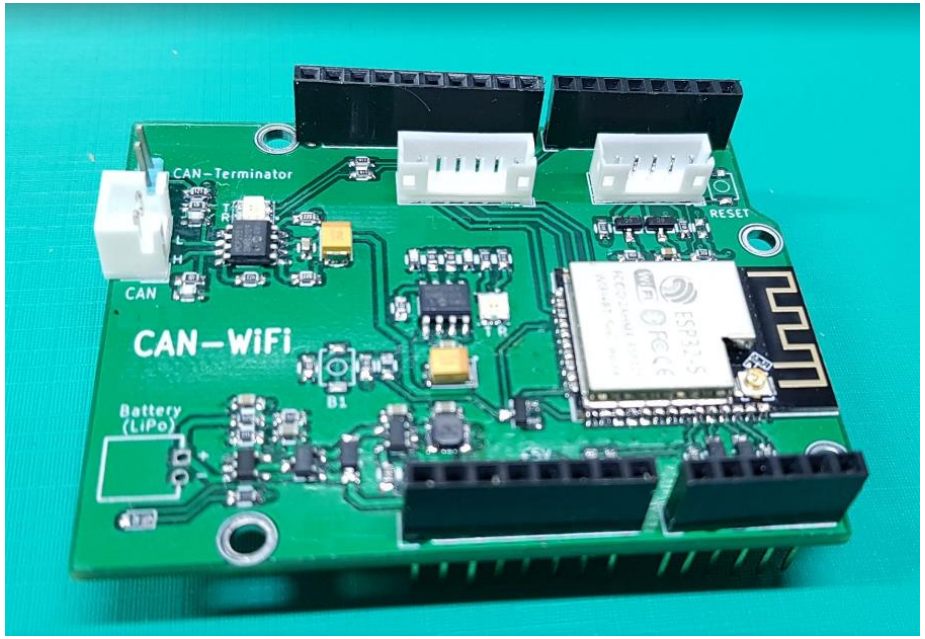


Realization

Or

Instantiation

Real world object



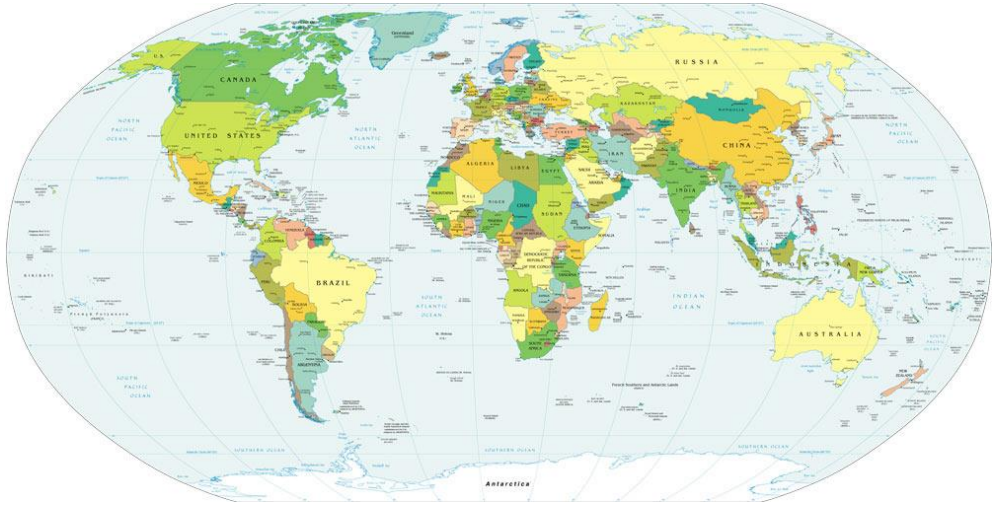
แบบจำลองวัตถุ (Object Models)

- การให้นิยามแบบจำลองควรเรียบง่าย ไม่ซับซ้อนเท่าวัตถุในโลกแห่งความจริง
- แต่ควรมีความเที่ยงตรงมากพอ ที่จะสะท้อนพฤติกรรมของวัตถุในโลกแห่งความจริง
- แบบจำลองที่ออกแบบมาดี จะสามารถใช้ทำนายพฤติกรรมของโลกจริงได้อย่างแม่นยำ

แบบจำลองในลักษณะต่างๆ

1. Procedure-oriented
   - Algorithms
2. Object-oriented
   - Classes and objects
3. Logic-oriented
   - Goals, often expressed in a predicate calculus
4. Rule-oriented
   - If–then rules
5. Constraint-oriented
   - Invariant relationships

แบบจำลองยอดนิยมของโลก (earth)



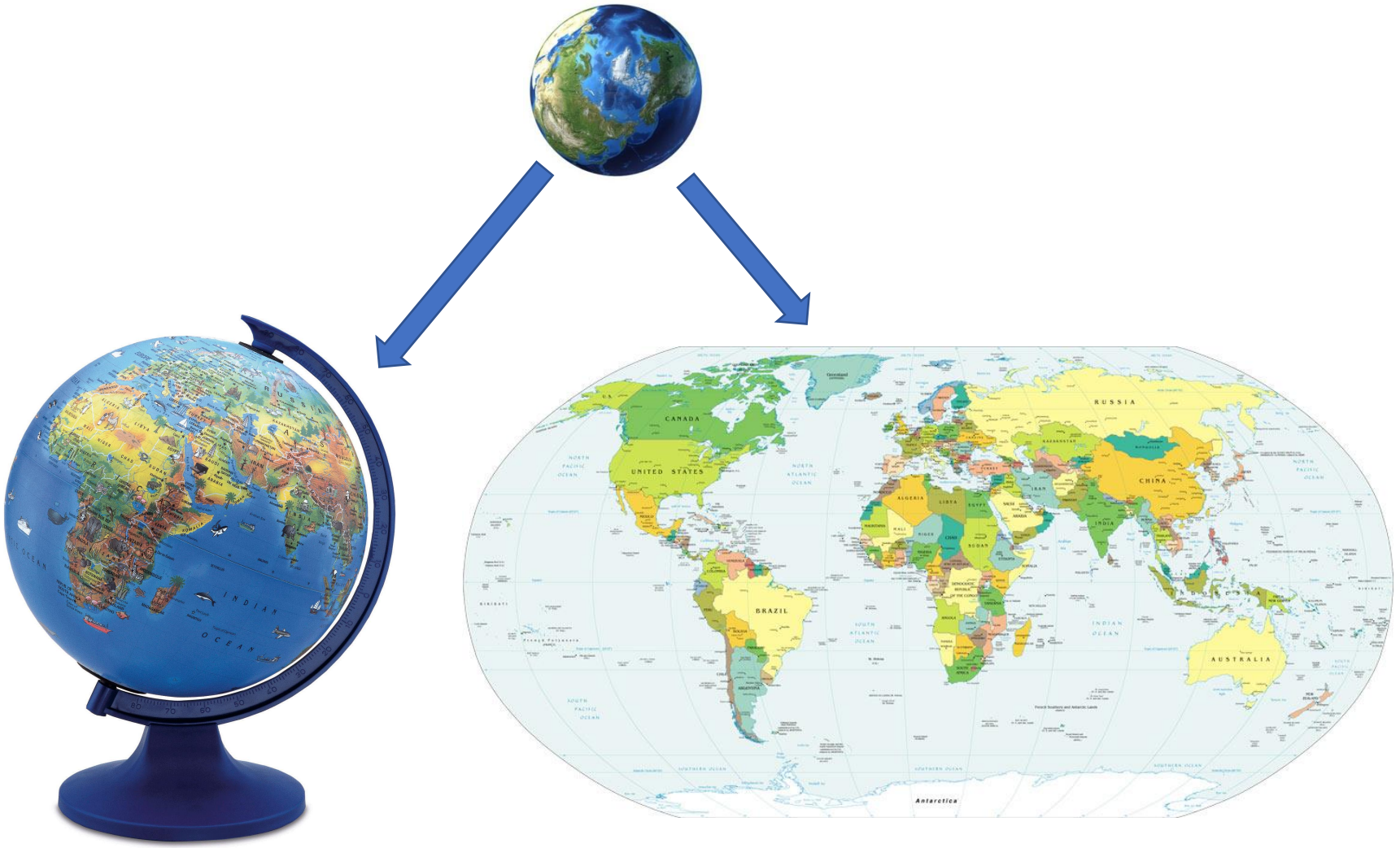
สถานะ (State) และพฤติกรรม (Behaviors) ของโลก

- ฝนตก แดดออก
- น้ำท่วม น้ำบ่าไหลหลาก
- แผ่นดินไหว ดินถล่ม
- ฟ้าร้อง ฟ้าผ่า
- ทอร์นาโด ไต้ฝุ่น ไซโคลน
- กลางวัน กลางคืน
- พระอาทิตย์/พระจันทร์ ขึ้น--ตก
- ฯลฯ

สถานะ (State) และพฤติกรรม (Behaviors) ของลูกโลก

- คำนวณระยะทาง
- จำลองกลางวัน กลางคืน
- ระบุตำแหน่งแห่งหนบนพื้นโลก

เราสามารถสร้างแบบจำลองของสิ่งเดียวกันที่ใช้งานแตกต่างกัน



- วัตถุในโลกแห่งความเป็นจริง มีความซับซ้อนแตกต่างกันไป

Domain ของ object

- ใน Domain หนึ่งๆ จะมี Objects ได้ตั้งแต่ 2 ตัวขึ้นไป จนถึงจำนวนนับไม่ถ้วน
- Object ตัวหนึ่งสามารถอยู่ในหลายๆ Domains ได้
- ขึ้นอยู่กับว่าเราจะกำหนด Domain ที่เราสนใจ (Domain of interest) อย่างไร

หลักสำคัญของ OOP

- Class and Subclass
- Encapsulate
- Inheritance
- Polymorphism
- Abstract Data Type (ADT)

Class and Subclass

- Class คือกลุ่ม (category) ของ objects ที่มีคุณสมบัติและพฤติกรรมที่เหมือนกัน
- ได้มาจากการทำ abstraction แบบ classification
- class จะต้องประกอบไปด้วย
  - data
  - behavior
  - interface
- เป็นต้นแบบ (prototype) หรือพิมพ์เขียว ที่จะกำหนดตัวแปรและวิธีการให้กับทุก object ที่สร้างขึ้นจาก class นั้น

Objects

- Objects ต้องมีคุณลักษณะ (State) ที่บ่งบอกถึงความเป็นตัวของมันเองในขณะนั้น
- Objects ต้องสามารถแสดงพฤติกรรม (Behavior) ของตัวเองออกมาได้

เช่น รถยนต์สีน้ำเงินคันหนึ่งวิ่งออกไปและหยุดลงตรงทางม้าลาย มีความหมายคือ วัตถุ ประเภท รถยนต์ มีคุณลักษณะของสีเป็นสีน้ำเงิน และมีพฤติกรรมที่แสดงถึงการเคลื่อนที่และหยุดได้

objects

- objects ต้องประกอบด้วย
  - ชื่อ (Identity)
  - สถานะ (State) คุณสมบัติ หรือค่าของข้อมูล (value)
  - พฤติกรรม (Behavior) ที่ระบุว่าสามารถทำอะไรได้บ้าง (method)

Encapsulate

- Encapsulate คือการปิดบัง หรือจำกัดการเข้าถึงข้อมูลบางอย่าง (Information hiding) ที่ไม่จำเป็นต้องให้ส่วนอื่นรับรู้
  - เช่น เราจะไม่ให้ผู้ใช้สนใจ (หรือมองเห็นได้) ว่า smartphone จะแปลงไฟล์ MP3 ออกมาเป็นเพลงได้อย่างไร (ผ่านไอซีตัวไหนบ้าง)
  - เขาควรสนใจแค่ใช้และติดต่อกับเครื่อง โดยการควบคุมผ่านแผงควบคุม เช่น เปิด-ปิด เล่น เร่งเสียง เปลี่ยนเพลงไปข้างหน้า ย้อนกลับ เป็นต้น
  - โดยเราต้องออกแบบควบคุมกฎเกณฑ์ต่างๆ ของซอฟท์แวร์ให้ สอดคล้องกับความเป็นจริง

Inheritance

- Inheritance เป็นการถ่ายทอดข้อมูล (state และ behavior) จาก class ลำดับที่สูงกว่า (base class) ไปยังลำดับที่ต่ำกว่า (derived class)
- โดยที่ derived class นั้นสามารถเปลี่ยนแปลง หรือแทนที่ข้อมูล (override) ซึ่งได้รับการถ่ายทอดมานั้นได้

Polymorphism

- Polymorphism มาจากภาษา Greek แปลว่า "having multiple forms“
- ใน method ใดๆ ใน derived class สามารถมีพฤติกรรมที่ต่างไปจาก method ชื่อเดียวกันใน base class ได้

Abstract Data Type (ADT)

- เป็นรูปแบบชนิดของข้อมูลที่ผู้พัฒนาเป็นผู้กำหนดขึ้นมาเอง

เช่นอะไรบ้าง???

Object-Oriented Programming

Data + Tools

มีเครื่องมือที่จะจัดการกับ data ในตัว

ห่อหุ้ม ทำให้ไม่รู้ว่าในนี้มีอะไร?

คำถาม