

แนะนำแนวคิดเชิงวัตถุ

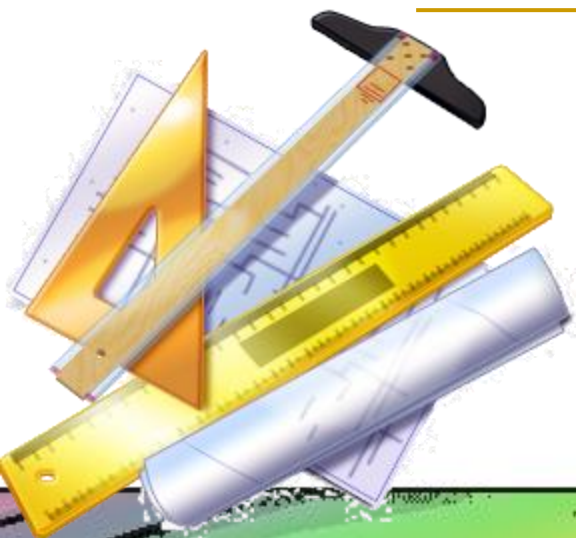
Introduction to Object-Oriented Concepts

Lecture 1

รศ.ดร.เยาวดี เต็มธนาภักดิ์

รศ.ปกรณ์ เสริมสุข

ปรับปรุงโดย อ.ดร.กฤตคม ศรีจิรานนท์



จุดมุ่งหมายของบทนี้

- อธิบายแนวคิดและวิวัฒนาการของการโปรแกรม
- อธิบายหลักการเชิงวัตถุ
- รู้จักกับภาษาจาวาและการโปรแกรมจาวาอย่างง่าย
- สามารถเขียนและใช้ comments ในโปรแกรมได้อย่างเหมาะสม
- สร้างความคุ้นเคยในการใช้ตัวแปลภาษาและสั่งให้โปรแกรมจาวาทำงาน
- สังเกตและแก้ไขความผิดพลาดในเรื่องของวากยสัมพันธ์ (Syntax) และตรรกะการทำงานอย่างง่ายได้

การโปรแกรม

■ การแก้ปัญหาคอมพิวเตอร์

- การกำหนดขั้นตอนวิธีในการแก้ปัญหาในภาษา/รูปแบบที่คอมพิวเตอร์เข้าใจ

■ โปรแกรมคอมพิวเตอร์ (Computer Program):

- code ที่อธิบายลำดับขั้นตอนให้คอมพิวเตอร์ทำงาน

■ การโปรแกรม (Programming):

- ศิลปะในการออกแบบและเขียนโปรแกรมคอมพิวเตอร์

วิวัฒนาการของการโปรแกรม

```
15 28
15 32
16 28 32 28
18 28 36
```

Machine Instructions

```
main:
:
    ld1 $1,28($15)
    ld1 $2,32($15)
    addq $1,$2,$1
    st1 $1,36($15)
```

Assembly

```
int main() {
    int x, y, z;
    :
    z = x + y;
}
```

High-level language (Unstructured)

```
int main() {
    int x, y, z;
    :
    z = add(x, y);
}
```

```
int add(int a, int b) {
    return a + b;
}
```

High-level language (Structured)

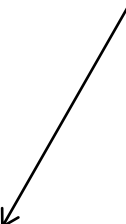
คุณภาพที่ต้องการของซอฟต์แวร์

- **ความถูกต้อง (Correctness)** – ถูกต้องสอดคล้องกับความต้องการใช้งาน
- **การทันต่อเวลา (In Time)** – สามารถส่งมอบได้ตามกำหนด
- **ความน่าเชื่อถือ (Reliability)** – ทำงานได้แม้ในภาวะไม่ปกติ (ในระดับหนึ่ง)
- **ความง่ายต่อการใช้งาน (Ease of use)** – โดยกลุ่มผู้ใช้เป้าหมาย
- **ความมีประสิทธิภาพ (Efficiency)** – ทำงานได้อย่างมีประสิทธิภาพที่ดี
- **การบำรุงรักษาได้ (Maintainability)** – ยืดหยุ่น, ง่าย, อ่านง่าย
 - กว่า 40% ของการบำรุงรักษาเกิดขึ้นจากการเปลี่ยนแปลงความต้องการของผู้ใช้
- **การนำกลับมาใช้และขยายต่อได้ (Reusability & Extendibility)** – ถูกนำกลับมาใช้ซ้ำและนำมาขยายต่อเพิ่มได้ง่าย
- **การย้าย (Portability)** สามารถย้ายไปใช้ในสิ่งแวดล้อมใหม่ได้ง่าย

วิวัฒนาการของการโปรแกรม (ต่อ)

High-level language (Object-Oriented)

```
class Calculator {  
    public static void main(String[] args) {  
        int x = Integer.parseInt(args[0]);  
        int y = Integer.parseInt(args[1]);  
  
        SimpleMath m = new SimpleMath();  
        int z = m.add(x,y);  
    }  
}
```

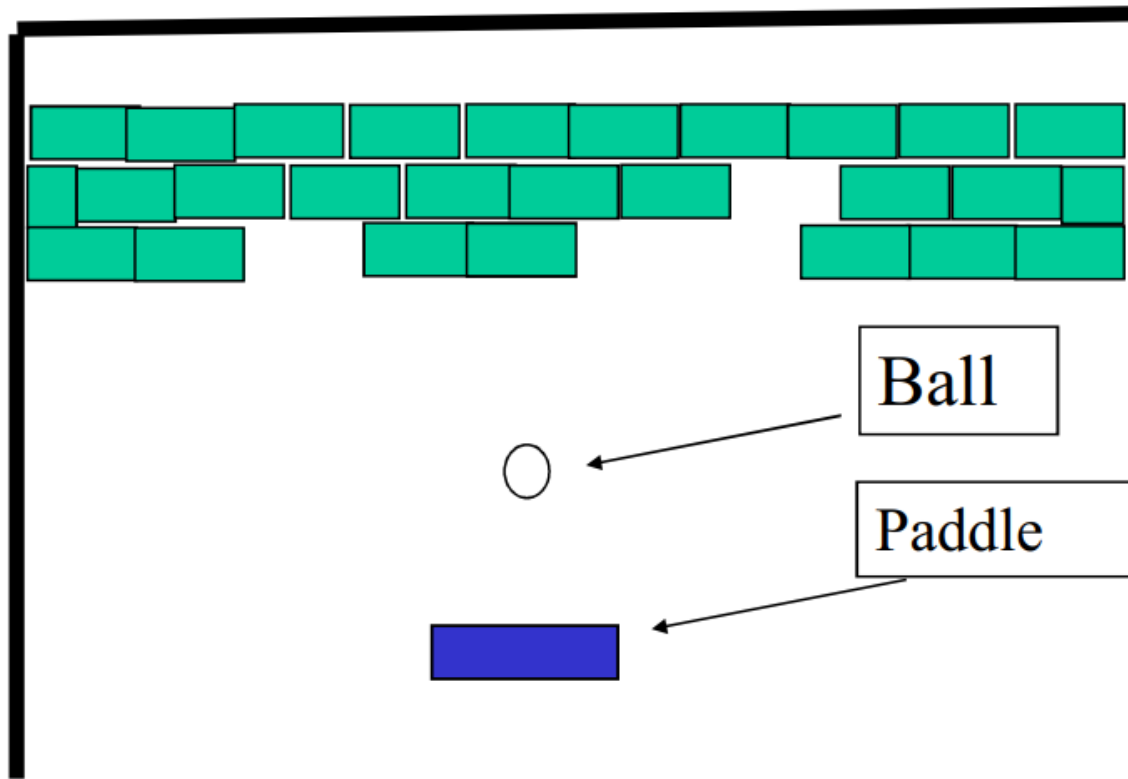


```
class SimpleMath {  
    int add(int a, int b)  
    {  
        return a+b;  
    }  
}
```

Grady Booch

“Abstraction and Information Hiding are software engineering's best hopes to date for managing the complexity of our solutions to problems.”

แนวทางของ Traditional versus OO

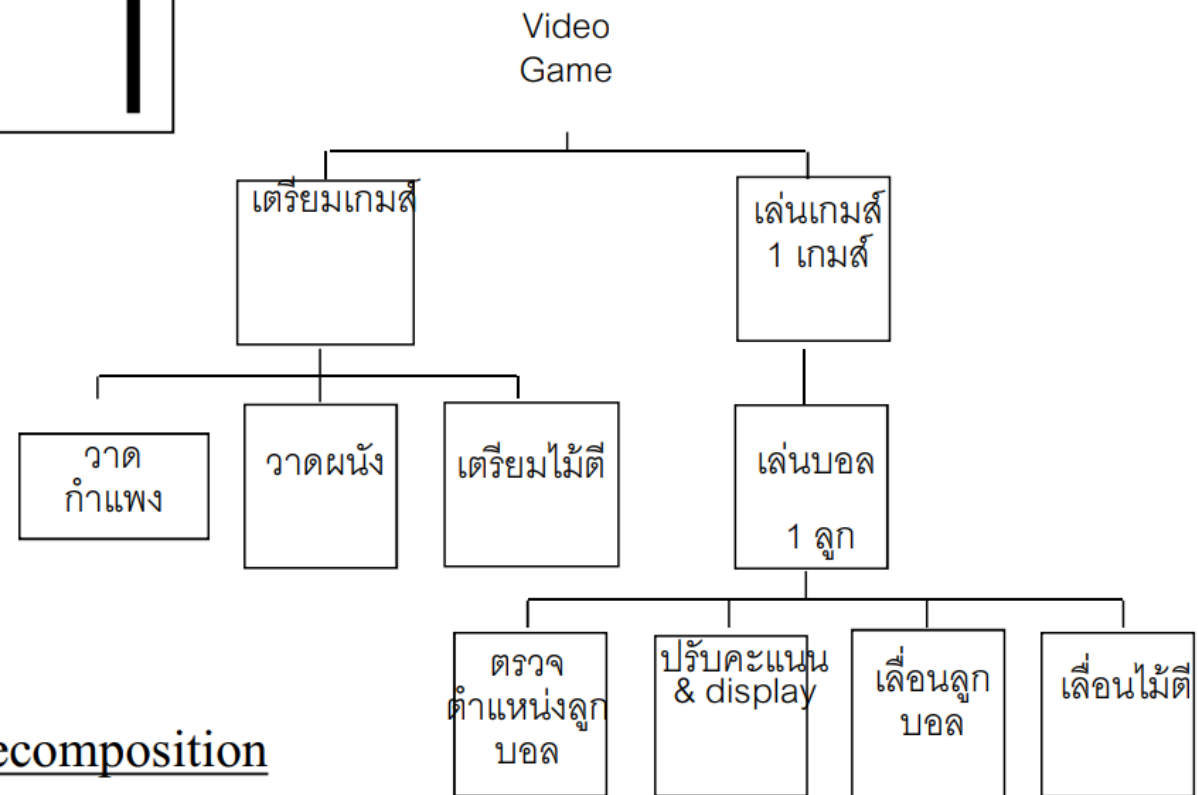
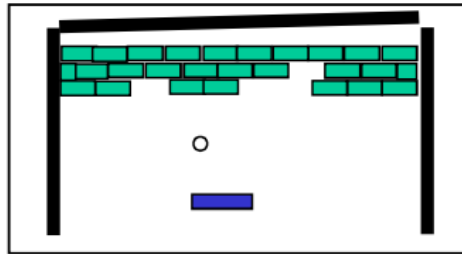


Simple video game

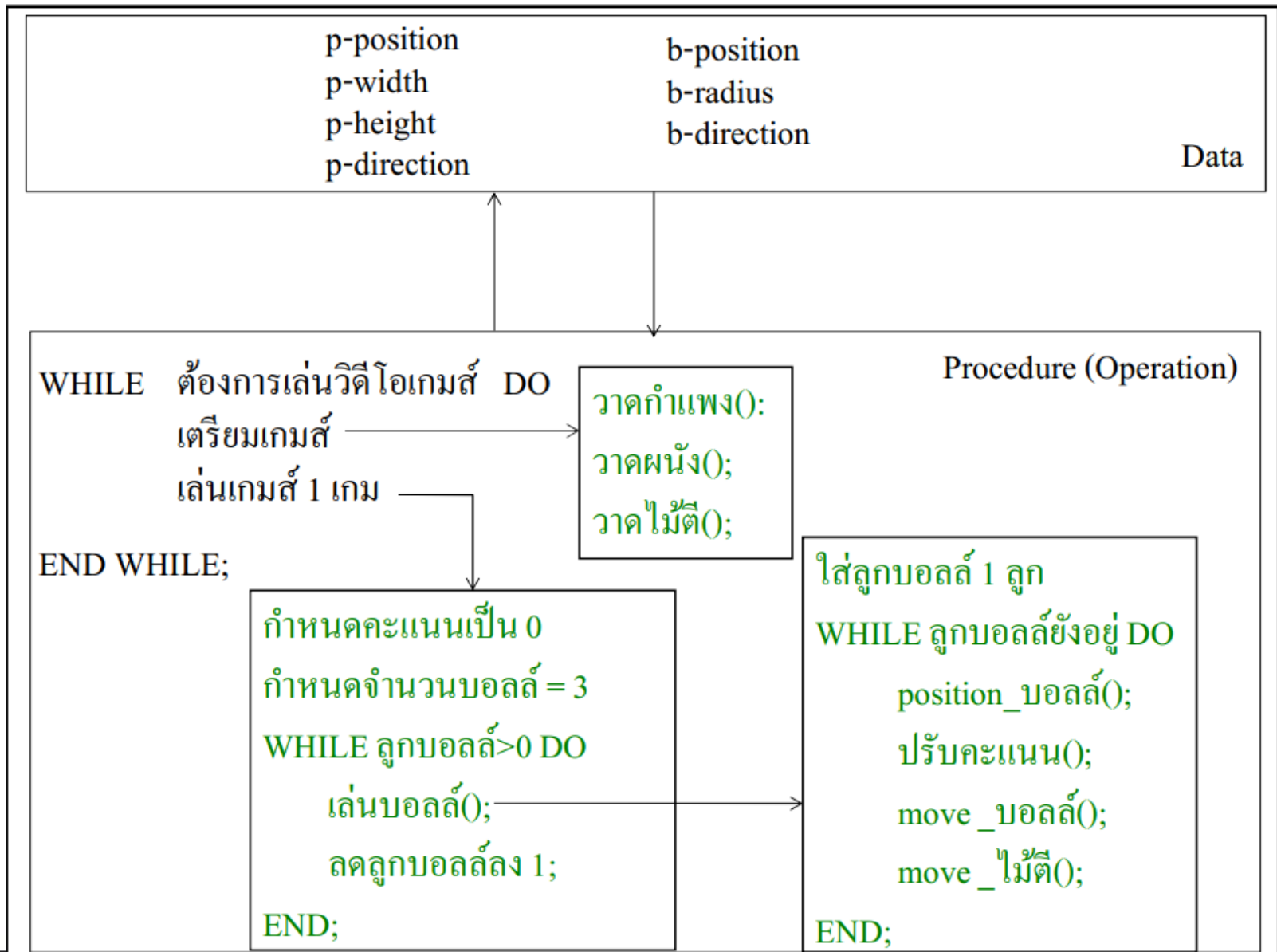
Traditional Approach (Structured Design)

- หาวิธีการทำงาน (Algorithm)
- แแตกงานใหญ่ออกเป็นงานย่อย
- แแตกงานย่อยลงไปในระดับที่ย่อยลงไป จนกระทั่งสามารถเขียนวิธีการทำงานย่อยนั้นได้
- การทำงานเริ่มจากงานหลัก (main) แล้วเรียกใช้งานย่อย (procedure) เพื่อให้ได้ผลลัพธ์ตามที่ต้องการ

Traditional Approach (Structured Design)



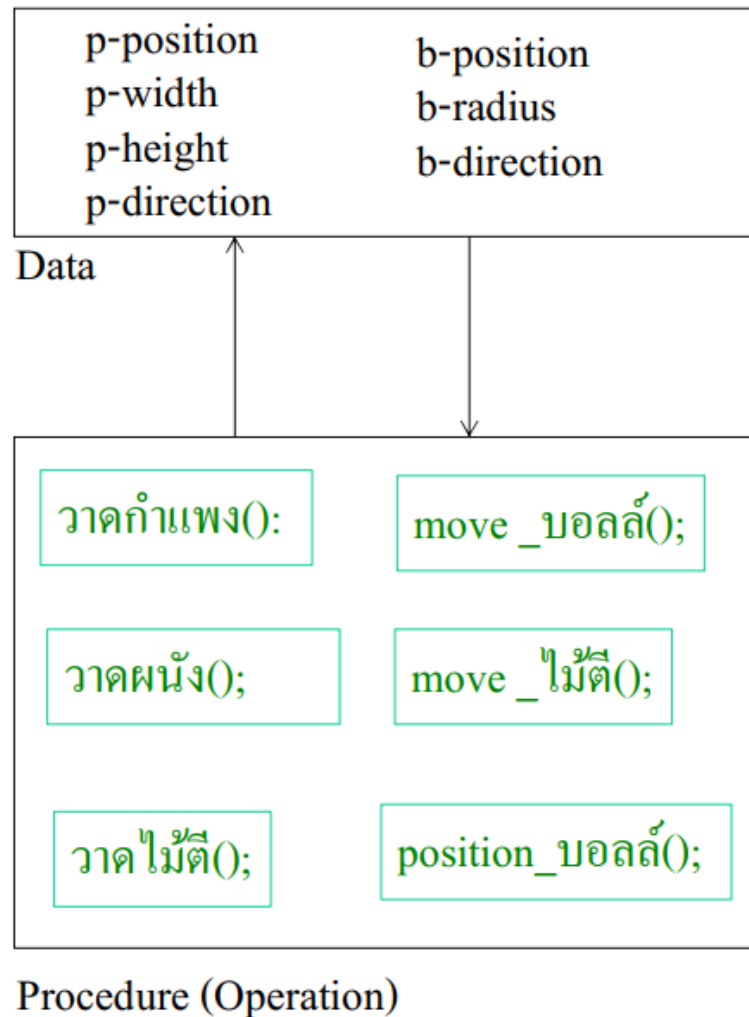
Algorithmic decomposition



จุดด้อยของ Structured approach

- การมองระบบเป็นงาน และข้อมูลแยกจากกัน ไม่สอดคล้องกับโลกทัศน์แห่งความจริง
- ข้อมูลถูกแยกจากกระบวนการ การธำรงรักษาข้อมูลให้ถูกต้องทำได้ลำบาก
- การปรับปรุงส่วนใดส่วนหนึ่งกระทบกับส่วนอื่นที่สัมพันธ์กัน
- การนำซอฟต์แวร์บางส่วนกลับมาใช้ใหม่ทำได้ยาก
- โครงสร้างงานคือโครงสร้างซอฟต์แวร์ทำให้การแก้ไขกรรมวิธีงานกระทบกับซอฟต์แวร์
- การกระจายการประมวลผลทำได้ยาก

แนวคิดใหม่ – รวม data และ procedure เข้าด้วยกันเป็น Object



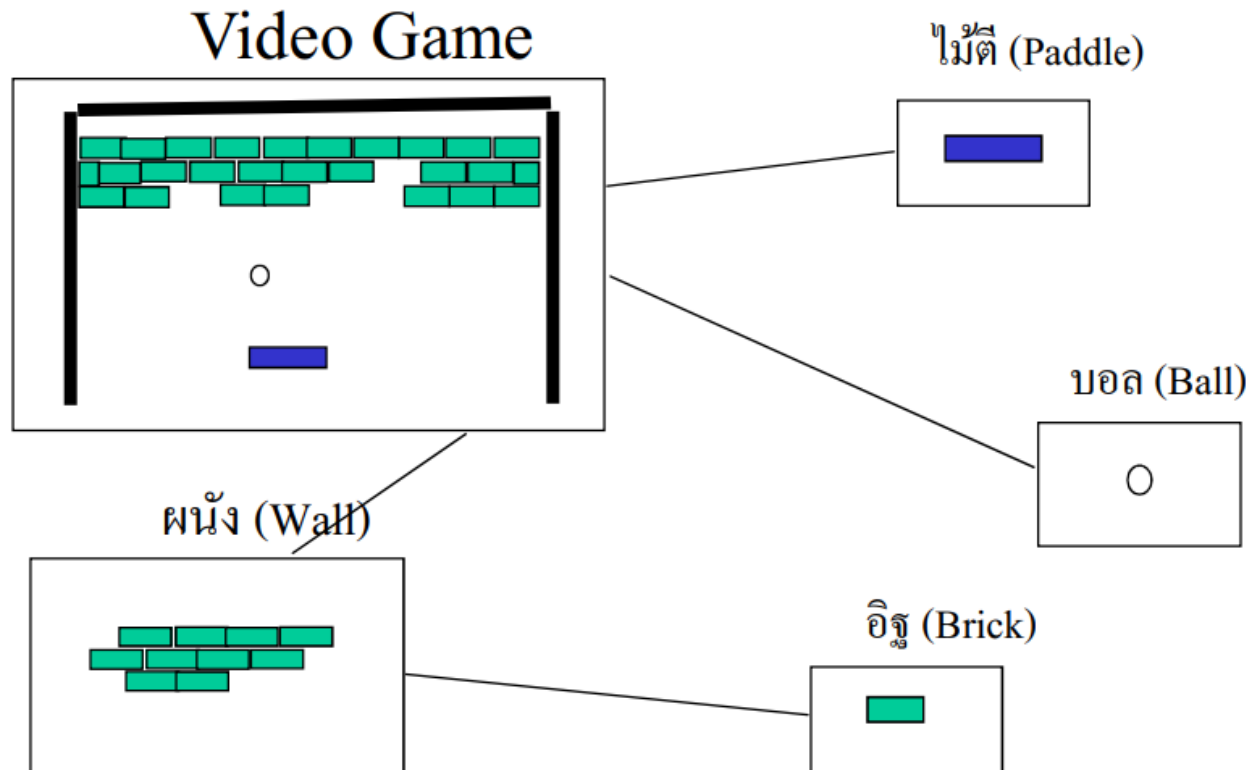
Ball
- position - radius -direction
+position() +direction() +modifyPosition() +behindPaddle()

Paddle
- position - width - height -direction
+move() + direction() + ..()

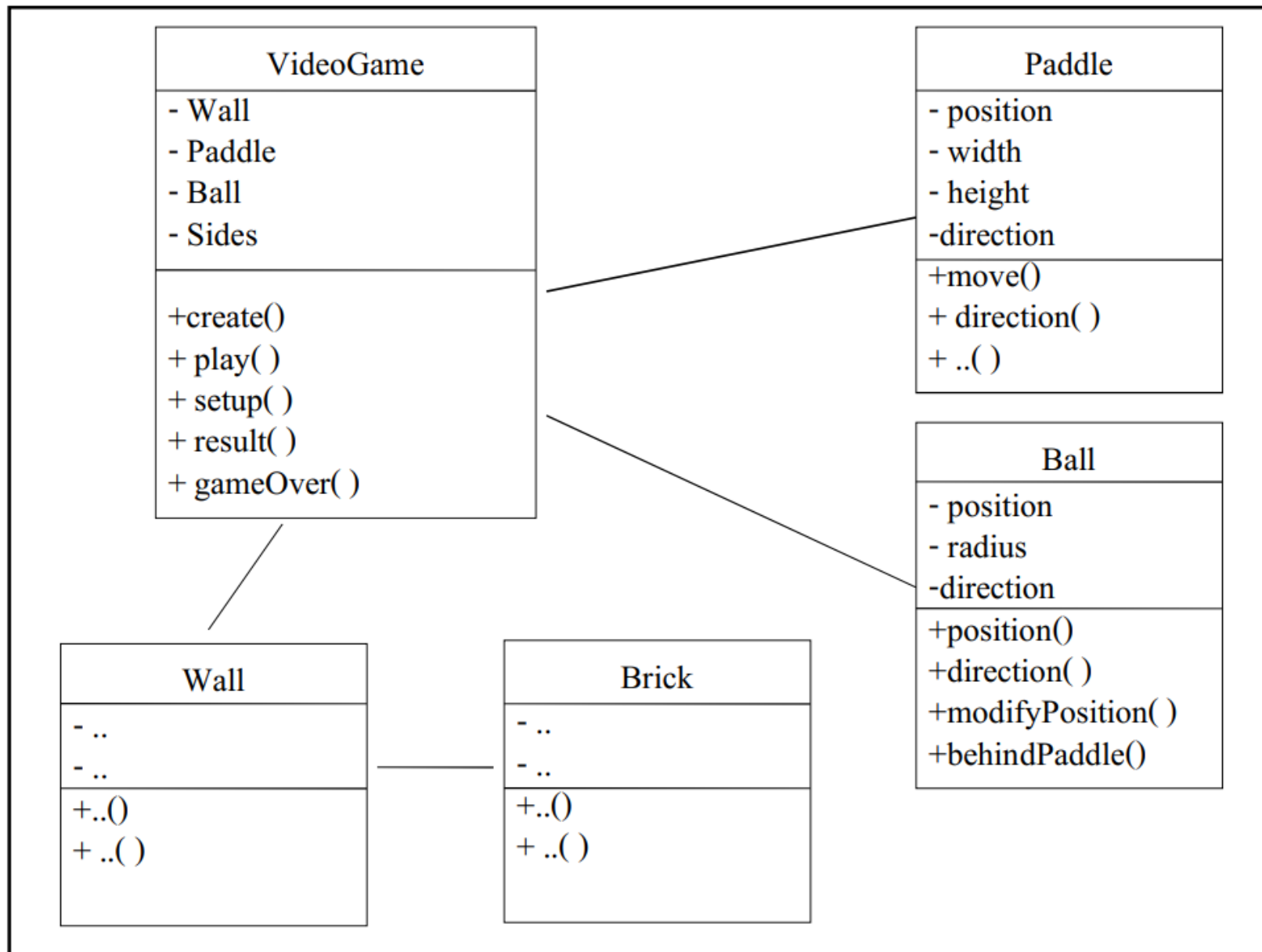
An Object-Oriented Approach

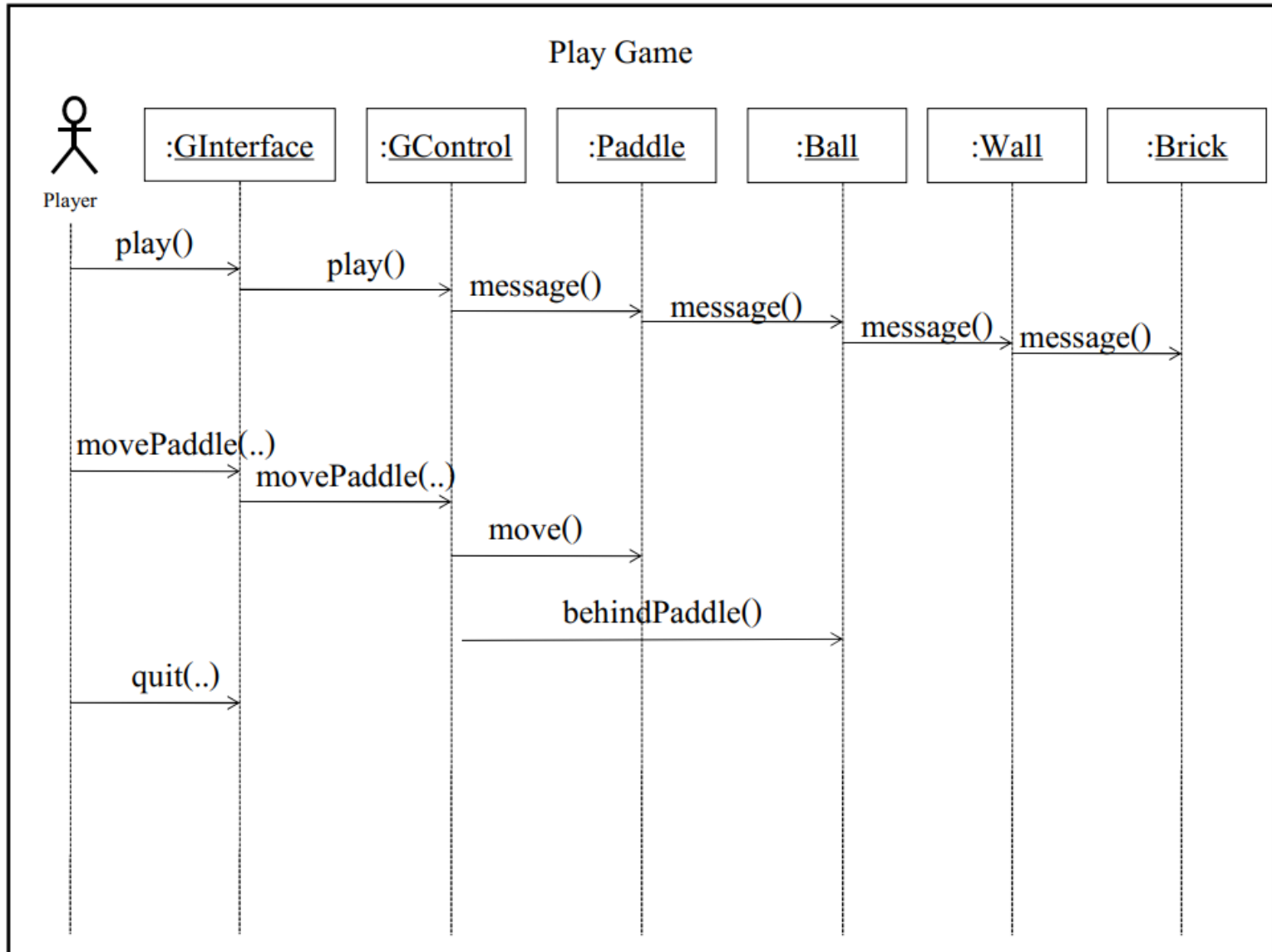
- หาสิ่งที่จำเป็นต้องใช้ในการทำงาน -> วัตถุ (Objects) ที่จำเป็นต้องใช้ในการทำงาน
- หาคุณสมบัติ (attributes) และความสามารถ (operations) ที่ต้องมีของแต่ละวัตถุ หาความสัมพันธ์ (Relationships) ระหว่างวัตถุเหล่านั้น
- แสดงการทำงานร่วมกันระหว่างวัตถุ เพื่อให้ได้ผลลัพธ์ตามที่คาดหวัง

An Object-Oriented Approach



Object-Oriented decomposition





จุดเด่นของ Object-Oriented approach

- การนำบางส่วน of SW (code และ architecture) มาใช้ใหม่ทำได้ง่าย
- การโมเดลระบบ SW ทำได้ตรงกับโลกทัศน์ของความเป็นจริง
 - More accurately describes corporate entities
 - decomposed based on natural partitioning
 - Easier to understand and maintain
- SW มีความมั่นคง (Stability)
 - A small change in requirements does not mean massive changes in the system under development
- รองรับกับการเปลี่ยนแปลง (Adaptive to change)
- Object technology เป็น single paradigm
 - A single language used by users, analysts, designers, and implementers

หลักการเชิงวัตถุ

■ การกำหนดสาระสำคัญ (*Abstraction*)

- การกำหนดสิ่งต่าง ๆ โดยเน้นไปที่สาระสำคัญและละเว้นรายละเอียดปลีกย่อย

■ การประกอบ (*Composition*)

- การสร้างซอฟต์แวร์โดยการใช้องค์ประกอบย่อยรวมกันเข้าเพื่อแก้ปัญหาที่ต้องการ

■ การห่อหุ้มและการซ่อนข้อมูล (*Encapsulation and Information Hiding*)

- การแยกส่วนของวัตถุในแง่มุมมองภายนอกกับมุมมองภายใน เพื่อห่อหุ้มและซ่อนสิ่งที่ไม่จำเป็นต้องรู้จากมุมมองภายนอก

■ ลำดับชั้น (*Hierarchy*)

- การจัดลำดับชั้นของวัตถุเพื่อให้ง่ายในการเข้าใจและขยายต่อได้

การโปรแกรมเชิงวัตถุ

- **Object-Oriented:** แนวคิดที่พิจารณาวัตถุและปฏิสัมพันธ์ระหว่างวัตถุ
- **การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming):**
 - การเขียนโปรแกรมในลักษณะ object-oriented style คือการนำ objects ที่สัมพันธ์กันหรือมีการทำงานร่วมกันมาประกอบกัน
 - **วัตถุ (Object)** เป็นหัวใจสำคัญในการเข้าใจและเขียนโปรแกรมเชิงวัตถุ

วัตถุ (Object)

- วัตถุ (Object) ใช้แสดงสิ่ง (entity) ซึ่งอาจเป็นวัตถุที่จับต้องได้ (physical) เป็นเพียงวัตถุในระดับแนวคิด conceptual

- ☐ Physical entity



- ☐ Conceptual entity



วัตถุ (ต่อ)

A concrete manifestation of an abstraction; an entity with a well-defined boundary and identity that encapsulates state and behavior; an instance of a class. (Booch, 1999)

■ วัตถุ ประกอบด้วย:

- ❑ **ลักษณะ (Properties):** ถูก implemented โดยกลุ่มของลักษณะ (attributes) ซึ่งเก็บค่าของวัตถุและความสัมพันธ์ของวัตถุกับวัตถุอื่น
- ❑ **ความสามารถ (Capabilities):** วิธีการที่วัตถุปฏิสัมพันธ์กับวัตถุอื่น (ผ่าน public interface)
- ❑ **อัตลักษณ์ (Identity):** ID ซึ่งแตกต่างกันออกไปในแต่ละวัตถุ ถึงแม้วัตถุอาจมีลักษณะและความสามารถเหมือนกันก็ตาม (ทั่วไปถูกกำหนดโดยระบบและไม่เห็นจากภายนอก)

ลักษณะของวัตถุ

- เราอธิบายคุณลักษณะของวัตถุ (ในมุมมองที่เราใส่ใจ) โดยอ้างอิงไปยังคุณสมบัติหรือ แอททริบิวต์ (Attribute) ของวัตถุนั้น
 - ตย. อธิบายจน. ด้วยรูปร่าง สีผม หน้าตา ความสัมพันธ์ และอื่น ๆ
- แอททริบิวต์มีค่าได้ (ค่าแอททริบิวต์คือข้อมูล (Data) อยู่ภายในวัตถุ)
 - ตย. รูปร่าง – ผอมสูง, สีผม – ดำแดง, หน้าตา – สวย, ความสัมพันธ์ - เป็นเพื่อน กับจน.
- ค่าของแอททริบิวต์เปลี่ยนแปลงได้
 - ตย. รูปร่าง – อ้วน, สีผม – แดง
- ค่าของแอททริบิวต์บ่งชี้สภาพ (State) ตัวตนของวัตถุเวลานั้น ๆ
- วัตถุในระบบคอมพิวเตอร์
 - วัตถุชั่วคราว (Transient Object) – สลายเมื่อสิ้นสุดโปรแกรม ไม่มีการเก็บข้อมูลของวัตถุ
 - วัตถุยืนยาว (Persistent Object) – มีชีวิตยาวนาน ต้องเก็บข้อมูลในฐานข้อมูล/แฟ้ม เพื่อให้วัตถุมีสภาพล่าสุดเมื่อ ใช้โปรแกรมครั้งถัดไป

BankAccount
-accNo -owner -balance -close
+deposit(float amt) +withdraw(float amt) +float getBalance()

PakornAccount	
accNo	1001100
owner	Pakorn Sermsuk
balance	250,000.00
close	no

สภาวะ PakornAccount เมื่อ 1 มค. 61

PakornAccount	
accNo	1001100
owner	Pakorn Sermsuk
balance	100,000.00
close	no

สภาวะ PakornAccount เมื่อ 9 มค. 61

ความสามารถของวัตถุ

- วัตถุมีความสามารถในการทำพฤติกรรม (Behavior)
- พฤติกรรมของวัตถุคือ การทำออปอเรชัน (Operation) ที่วัตถุมีให้บริการ
 - ตย. ออปอเรชัน โทรศัพท์มือถือ รับสาย โทรออก โอน อื่น ๆ
 - ตย. ออปอเรชันของวินโดว์ -> open(), close(), resize(), move()
- ออปอเรชันที่วัตถุมีบริการแสดงไว้ที่ส่วนอินเตอร์เฟส (Interface) ของวัตถุ
- ผู้ใช้บริการส่งแอสเซส (Message Passing) เพื่อเรียกใช้ออปอเรชัน ที่ต้องการรับบริการ
- เมื่อได้รับแอสเซส วัตถุผู้ให้บริการจะดำเนิน การตามวิธีปฏิบัติงาน (Method) ที่กำหนดไว้ภายในวัตถุ

อัตลักษณ์ (Identity) ของวัตถุ

- วัตถุมีอัตลักษณ์ (Identity) แสดงตัวตนที่ต่างกันของวัตถุ
- อัตลักษณ์ช่วยแยกแยะวัตถุหนึ่งจากวัตถุอื่น ทำให้วัตถุสื่อสารกันได้อย่างถูกต้อง
- แอททริบิวต์หรือกลุ่มแอททริบิวต์ ใช้เป็นตัวบ่งชี้อัตลักษณ์ของวัตถุได้
 - ตย. อัตลักษณ์ที่บ่งชี้นักศึกษาคือ ID
- บางกรณีวัตถุในระบบคอมพิวเตอร์ใช้ตำแหน่งในหน่วยความจำเป็นอัตลักษณ์บ่งชี้วัตถุ
 - ตย. `int a, b;`

มุมมองของคลาสและวัตถุ

■ ในชีวิตประจำวัน

- **วัตถุ (Object):** สิ่งที่เราสนใจในขอบเขตของปัญหา ซึ่งสามารถระบุแยกออกมาได้อย่างชัดเจน
- **คลาส (Class):** กลุ่มของวัตถุที่มีลักษณะและความสามารถอย่างเดียวกัน
 - วัตถุเป็นตัวอย่าง (instance) ของคลาส

■ ในโมเดลเชิงวัตถุ

- **วัตถุ:** วัตถุมี identity เฉพาะไม่ซ้ำกัน มีลักษณะและความสามารถ
- **คลาส:** โครงสร้างลักษณะและความสามารถร่วมของทุกวัตถุ (instances) ในคลาสนั้น

ตัวอย่างวัตถุในระบบซอฟต์แวร์

UI objects	Attributes	Methods
Button	size, shape, color, location, caption	click, enable, disable, hide, show
Form	width, height, border, style, bg color	change size, minimize, maximize, appear, disappear
Label	size, shape, color, location, text	see text, get text, hide, show

Problem Domain objects	Attributes	Methods
Customer	name, address, phone number	set name, set address, add new order for customer
Order	order number, date, amount	set order date, calculate order amount, add product to order, schedule order, shipment
Product	product number, description, price	add to order, set description, get price

ข้อดีของการ โปรแกรมเชิงวัตถุ

- ความสามารถในการ โมเดลสิ่งสำคัญขององค์ประกอบในระบบได้ดี
- มีลักษณะการ implementation ที่เป็น modular
- แยกรายละเอียดวิธีการทำงานออกจากการติดต่อ (interface)
- การสนับสนุนการนำกลับมาใช้ (Software Reuse)

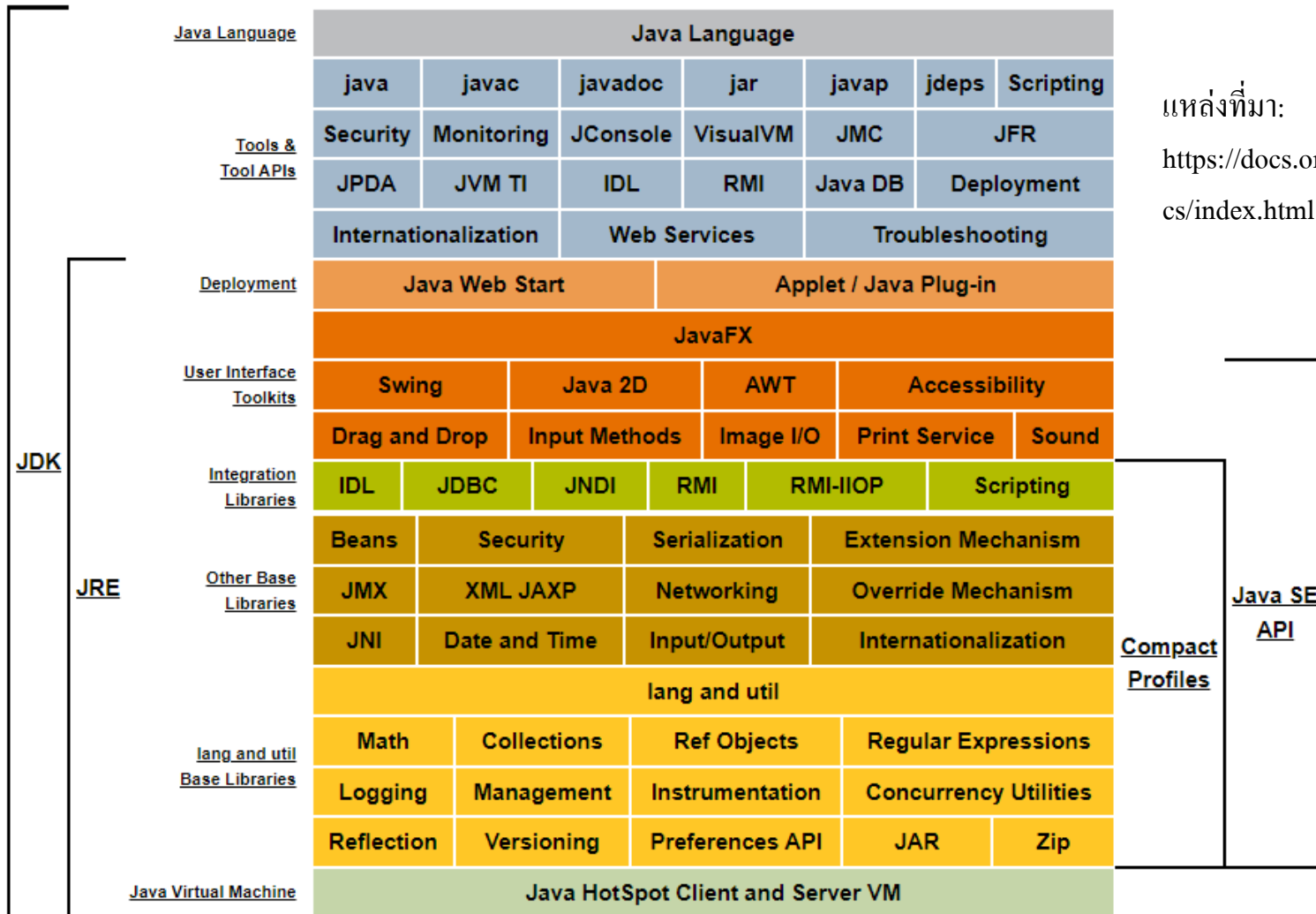
ข้อดีของการโปรแกรมเชิงวัตถุ

- ต้องออกแบบก่อนพัฒนา
- การเลือก**วัตถุให้ถูกต้อง**เป็นเรื่องยาก
- ปฏิกริยาสัมพันธ์ระหว่างวัตถุอาจซับซ้อน
- มีลักษณะการทำงานแบบคู่ขนานซ้อน โดยนัย (Inherent) ซึ่งทำให้ยากในการเข้าใจและอาจนำไปสู่ความผิดพลาดในด้านการประสานกัน (synchronization errors)
- อาจต้องไล่ตามข้อความ (message) ที่ส่งผ่านระหว่างวัตถุจำนวนมาก

เปรียบเทียบการ โปรแกรมในลักษณะ โครงสร้างและเชิงวัตถุ

- ในโปรแกรมแบบ โครงสร้าง (Structured Programming)
 - กำหนดสิ่งที่ต้องการจะทำ
 - กำหนดโครงสร้างของข้อมูลที่เกี่ยวข้องที่ใช้ในการแก้ปัญหา
- ในโปรแกรมเชิงวัตถุ (Object-Oriented Programming)
 - กำหนดวัตถุที่จะใช้ในการแก้ปัญหา
 - จัดกลุ่มของวัตถุที่มีลักษณะเหมือนกันเข้าด้วยกัน
 - กำหนดสิ่งที่วัตถุกระทำได้

องค์ประกอบของ Java SE Technology



Java SE
API

Compact
Profiles

จาวา (Java)

■ Java Programming Language เป็นภาษาโปรแกรมเชิงวัตถุ

- ❑ ง่าย: วากยสัมพันธ์ (Java Syntax) - คล้าย C++ แต่ความหมาย (Semantic) คล้าย Smalltalk
- ❑ ถูกออกแบบโดยคำนึงถึงความปลอดภัยและ portability
- ❑ มี libraries (API) จำนวนมาก
- ❑ ไม่ขึ้นกับ Platform
- ❑ สามารถใช้ในการพัฒนาแอปพลิเคชันได้ในหลากหลายระดับ เช่น แอปพลิเคชันบนมือถือ จนถึงระดับองค์กร (Enterprise)

องค์ประกอบของภาษาจาวา

- **Class Loader:** โหลดโค้ดเพื่อเริ่มการทำงานและการตรวจสอบความปลอดภัยของโค้ด
 - ❑ *Loading Code* (แยก namespaces ของ local classes & remote classes, วาง memory layout ของ file)
 - *Verifying Code* (ตรวจ code ว่าไม่ละเมิดสิทธิ ไม่เปลี่ยนชนิดของวัตถุ ไม่ก่อให้เกิด overflow, underflow)
 - ❑ *Linking:* เชื่อมคลาสต่าง ๆ สำหรับ execution
 - ❑ *Initializing:* จองและ setup ค่าเริ่มต้นในหน่วยความจำ
- **Java Virtual Machine (JVM)**
 - ❑ *เครื่องสมมติ (Imaginary machine)* ที่ถูก implemented โดยการทำให้ S/W emulate บนเครื่องคอมพิวเตอร์จริง ทำงานโดยอ่าน byte codes แล้ว interpret เป็น machine code
- **ตัวกำจัดขยะ (Garbage Collection)**
 - ❑ ทำหน้าที่คืนหน่วยความจำให้ระบบเมื่อไม่ได้มีการอ้างถึงแล้ว การทำงานเป็นไปโดยอัตโนมัติ และวิธีการทำงานของตัวกำจัดขยะแตกต่างกันไปขึ้นกับ JVM

ตัวอย่างโปรแกรม MyHelloWorldTest และ Class Greeter

```
1. /* Greeter class to greet by saying a message from parameter */
2. public class Greeter {
3.     public void say(String str) {
4.         System.out.println(str);
5.     }
6. }
```

```
1. /* Program MyHelloWorldTest
2. This program is my first program.*/
3. public class MyHelloWorldTest {
4.     public static void main(String[] args) {
5.         Greeter greeter = new Greeter();
6.         greeter.say("Hello World");
7.     }
8. }
```

เมื่อแปลและสั่งให้ทำงาน โปรแกรมนี้จะพิมพ์คำว่า Hello World ออกทางหน้าจอ

องค์ประกอบพื้นฐานอย่างง่ายของคลาสในภาษาจาวา

- การอธิบาย (Comments)
- การประกาศคลาส (Class Declaration)
- ประกาศองค์ประกอบของคลาส (Attributes และ Methods)

Comment

- Comment ใช้ช่วยในการอธิบายสิ่งต่างๆในโปรแกรม เพื่อช่วยให้ง่ายในการอ่านและเข้าใจโปรแกรม อาทิ ใช้บอกจุดมุ่งหมายของโปรแกรม, อธิบายตัวแปร, อธิบาย codes
- Comment คือข้อความซึ่งอยู่

- ❑ ระหว่าง comment marker: `/* This is a comment example*/`

- ❑ อยู่หลัง `//`: `// This is a one-line comment`

- ❑ ระหว่าง javadoc comment marker: `/** javadoc sample comment */`

ถึงแม้ว่า comment จะไม่ใช่ส่วนบังคับในโปรแกรม แต่ช่วยให้การอ่านและเข้าใจโปรแกรมเป็นไปได้ง่ายขึ้น (self-explanatory program code)

3 รูปแบบของการ Comments ในจาวา

```
// This is C++-like style comment
```

```
// This is a one-line comment
```

```
/*
```

```
    This is a C-like style comment
```

```
    We can have multiple line here
```

```
*/
```

```
/**
```

```
    This is java comment for documenting
```

```
*/
```

การจับคู่ marker ของ Comment

```
/* This is a comment on 1 line*/
```

```
/*  
    This is another comment  
*/
```

```
/*
```

```
/* ←  
    Comment is here
```

Marker ตัวนี้เป็นส่วนใน comment

```
*/
```

```
* / ←
```

Error สำหรับ Marker ตัวนี้: ไม่ match กับตัวใด

หลักทั่วไปในการ comment โค้ด

- ข้อเสนอแนะส่วนของโค้ดที่ควรเขียน comment:
 - ส่วนบนของโปรแกรม (program file) หรือ File Header Comment
 - ชื่อผู้เขียน วันที่เขียนและแก้ไข จุดมุ่งหมายของโค้ดและหน้าที่ของโค้ด
 - เหนือทุกเมทอด หรือ Method Header Comment
 - จุดประสงค์ของเมทอด หรือ sub-component นั้นในโปรแกรม
 - ถ้ามีเมทอดเดียวอาจไม่เขียนส่วนนี้ แต่ไปรวมไว้กับ File Header Comment เลย
 - บรรทัดที่สำคัญ หรือ In line
 - ส่วนของโค้ดที่มีความซับซ้อน ("tricky") ที่ยากต่อความเข้าใจ
- ส่วนโค้ดที่ ไม่ จำเป็นต้อง comment:
 - ไม่ต้องเขียน Comments ในส่วนโค้ดที่อ่านแล้วเข้าใจได้ชัดเจน
 - ตัวแปร ควรใช้ชื่อที่สื่อความแทนการ comment

การประกาศ คลาส (Class Declaration)

- คลาสเป็นแนวคิดพื้นฐานและหัวใจสำคัญในการทำงานของจาวา
- จุดมุ่งหมายของคลาส เพื่อเป็นตัวสร้างวัตถุ (factories for objects)
- โปรแกรมจาวา ประกอบด้วยตั้งแต่ 1 คลาส: เป็นคลาสที่สร้างขึ้นเองหรือคลาสที่กำหนดไว้ก่อนล่วงหน้า (pre-defined)
- Syntax: ในการประกาศคลาสใหม่

```
public class <class name> {  
    class member declaration  
}
```

□ class member declaration = กลุ่มลำดับของ attribute หรือ method

ตัวอย่างการประกาศคลาส (Class Declaration)

■ ตัวอย่าง:

```
public class Greeter {  
    :  
    :  
    :  
}
```

■ หมายเหตุ: คำว่า public และ class เป็นคำสงวน (reserved word)

การประกาศตัวแปร (Variable Declaration)

■ Syntax:

`modifiers dataType variableName;`

- ❑ `modifiers` = กลุ่มตัวขยายที่ใช้บอกลักษณะของตัวแปร เช่น `public`, `final`
- ❑ `dataType` = ชนิดของตัวแปร
- ❑ `variableName` = ชื่อของตัวแปร

■ ตัวอย่างเช่น

- ❑ `private String name;`

การประกาศ Method (Method Declaration)

■ Syntax:

```
modifiers returnType methodName(parameters) {  
    methodBody  
}
```

- ❑ `modifiers` = กลุ่มตัวขยายที่ใช้บอกลักษณะของเมทอดเช่น `public`, `static`
- ❑ `return type` = ชนิดข้อมูล (data value) ที่คืนให้กับเมทอดตัวเรียก (caller)
- ❑ `method name` = ชื่อของเมทอด
- ❑ `parameters` = กลุ่มลำดับของตัวแปรที่ส่งมายังเมทอด
- ❑ `method body` = กลุ่มลำดับของชุดคำสั่ง (instructions)

ตัวอย่างของ method

Modifier

Return Type

Method Name

Parameter

```
public void say (String str) {
```

```
    System.out.println(str);
```

```
}
```

Method Body เป็นส่วนที่เราใส่
statements ของการทำงานของโปรแกรม

การเขียนโปรแกรมสำหรับการทดสอบ

- ในการ execute, JVM จะเรียก *main method* ในคลาสทำงานเป็นอันดับแรก
- สร้าง *Main Class* สำหรับเป็นจุดเริ่มต้นการทำงานของโปรแกรม ซึ่งทำได้โดย
 - สร้างคลาสใหม่ ที่ใช้เป็นโปรแกรมหลักสำหรับการทดสอบ
 - กำหนด main method ให้กับคลาสทดสอบในลักษณะดังนี้

```
public static void main(String[] args) {  
    methodBody  
}
```
 - กำหนดคำสั่งการทำงานใน main method

ขั้นตอนการทำงานพื้นฐานของคลาสทดสอบ

- คลาสทดสอบ โดยทั่วไปบรรจุคำสั่งในการทดสอบอีกคลาสหนึ่งไว้
 - คลาสทดสอบ หรือ main class หมายถึงคลาสที่มี main method ตามข้อกำหนดของจาวา
- ทั่วไป main method จะประกอบด้วยขั้นตอนดังนี้
 - สร้างตั้งแต่ 1 วัตถุที่ต้องการทดสอบ
 - เรียกอย่างน้อย 1 method ของวัตถุนั้นให้ทำงาน
 - พิมพ์ผลลัพธ์การทำงาน

โครงสร้างตัวอย่างสำหรับโปรแกรมทดสอบอย่างง่าย

comment

```
public class className
{
    public static void main(String[] args)
    {
        method body
    }
}
```

ตัวอย่างโปรแกรม MyHelloWorldTest และ Class Greeter (ต่อ)

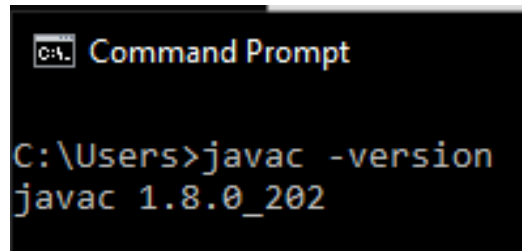
```
1. /* Greeter class to greet by saying a message from parameter */
2. public class Greeter {
3.     public void say(String str) {
4.         System.out.println(str);
5.     }
6. }
```

```
1. /* Program MyHelloWorldTest
2. This program is my first program.*/
3. public class MyHelloWorldTest {
4.     public static void main(String[] args) {
5.         Greeter greeter = new Greeter();
6.         greeter.say("Hello World");
7.     }
8. }
```

เมื่อแปลและสั่งให้ทำงาน โปรแกรมนี้จะพิมพ์คำว่า Hello World ออกทางหน้าจอ

การติดตั้ง JDK

- ดาวน์โหลด Java SE เวอร์ชัน 8 (ปัจจุบันเวอร์ชัน 17)
 - <https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html>
 - เลือกเวอร์ชันที่เหมาะสมกับแพลตฟอร์ม (i.e., Windows, Mac, Linux)
- ทำการติดตั้งตามขั้นตอนและ set path
- ทดลองใช้งานผ่าน console
 - `javac -version`



```
C:\Users>javac -version
javac 1.8.0_202
```

ขั้นตอนการสร้างและสั่งให้ทำงาน: MyHelloWorldTest

- การสร้างใช้ Text Editor เช่น Notepad, RealJ, อื่น ๆ

- การคอมไพล์ (Compile)

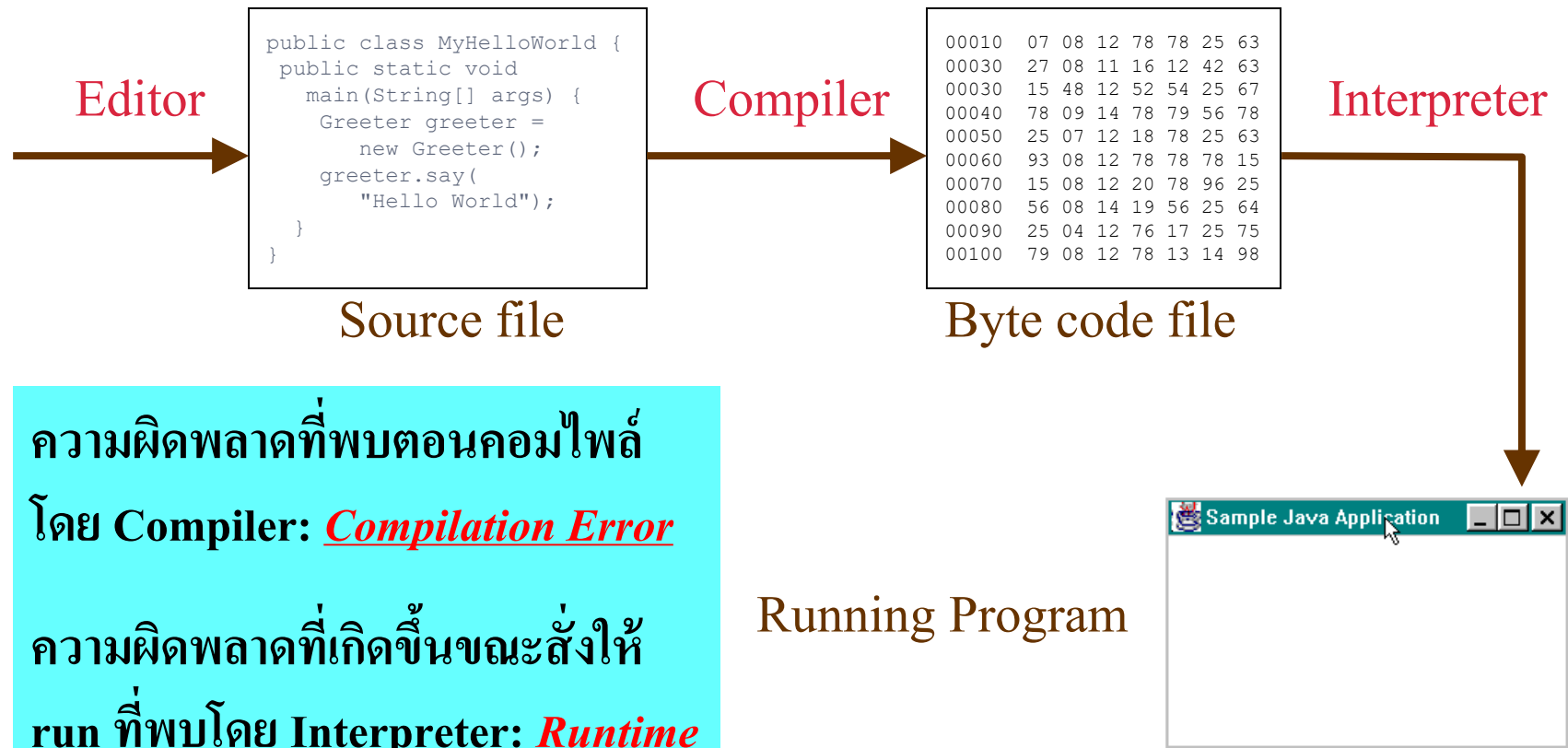
```
javac MyHelloWorldTest.java
```

- การ Run

```
java MyHelloWorldTest
```

- Compile error และ Runtime error

วงจรการเขียน-การคอมไพล์-และการรันโปรแกรม



ความผิดพลาดที่พบตอนคอมไพล์
โดย Compiler: **Compilation Error**

ความผิดพลาดที่เกิดขึ้นขณะสั่งให้
run ที่พบโดย Interpreter: **Runtime Error** หรือ **Execution Error**

ตัวอย่างโค้ดที่คอมไพล์และพบข้อผิดพลาด

```
C:\Users\[redacted]>javac MyHelloWorldTest.java
MyHelloWorldTest.java:5: error: cannot find symbol
    Greeter greeter = new Greeter();
    ^
  symbol:   class Greeter
  location: class MyHelloWorldTest
MyHelloWorldTest.java:5: error: cannot find symbol
    Greeter greeter = new Greeter();
    ^
  symbol:   class Greeter
  location: class MyHelloWorldTest
2 errors
```

สรุปการเรียนรู้วันนี้

- แนวคิดเกี่ยวกับการโปรแกรมโดยภาษาระดับสูง
- แนวคิดเกี่ยวกับเชิงวัตถุ
- การใช้ตัวแปลภาษาจาวา
- ขั้นตอนการสร้างและการสั่งให้โปรแกรมจาวาทำงาน (Creating and Running Java Program)
- ประโยชน์ของการใช้และเขียน comments ในโปรแกรม
- สังเกตและแก้ไขความผิดพลาดในเรื่องของวากยสัมพันธ์ (syntax) และตรรกะการทำงานอย่างง่ายได้