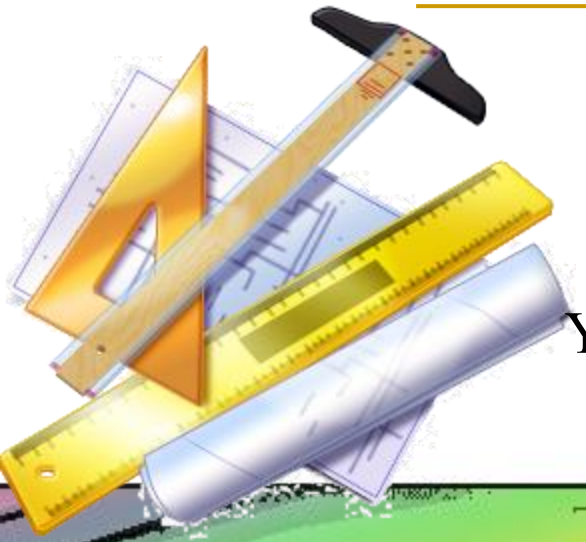
Object-Oriented Design and UML



Lecture 11

เยาวดี เต็มธนาภัทร์ และ สุกญัญา รัตโนทยานนท์

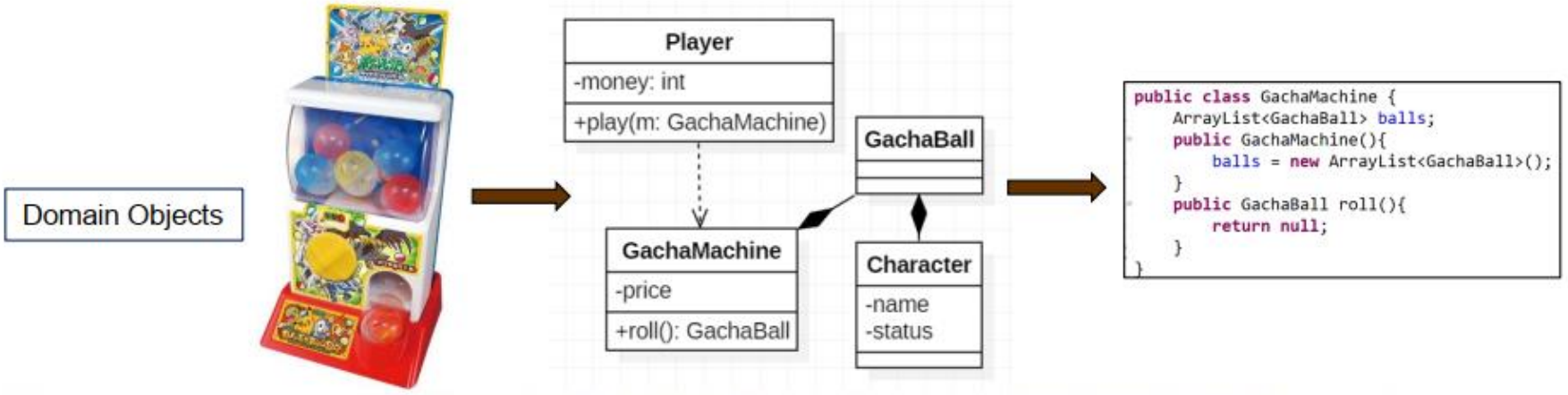
Yaowadee Temtanapat & Sukanya Ratanotayanon

วัตถุประสงค์ของการเรียนวันนี้

- เข้าใจแนวคิดเกี่ยวกับการออกแบบเชิงวัตถุ
- เข้าใจการสร้างและแปลความหมายของแผนภาพ UML เบื้องต้น
  - Use Case Diagram
  - Sequence Diagram
  - Class Diagram

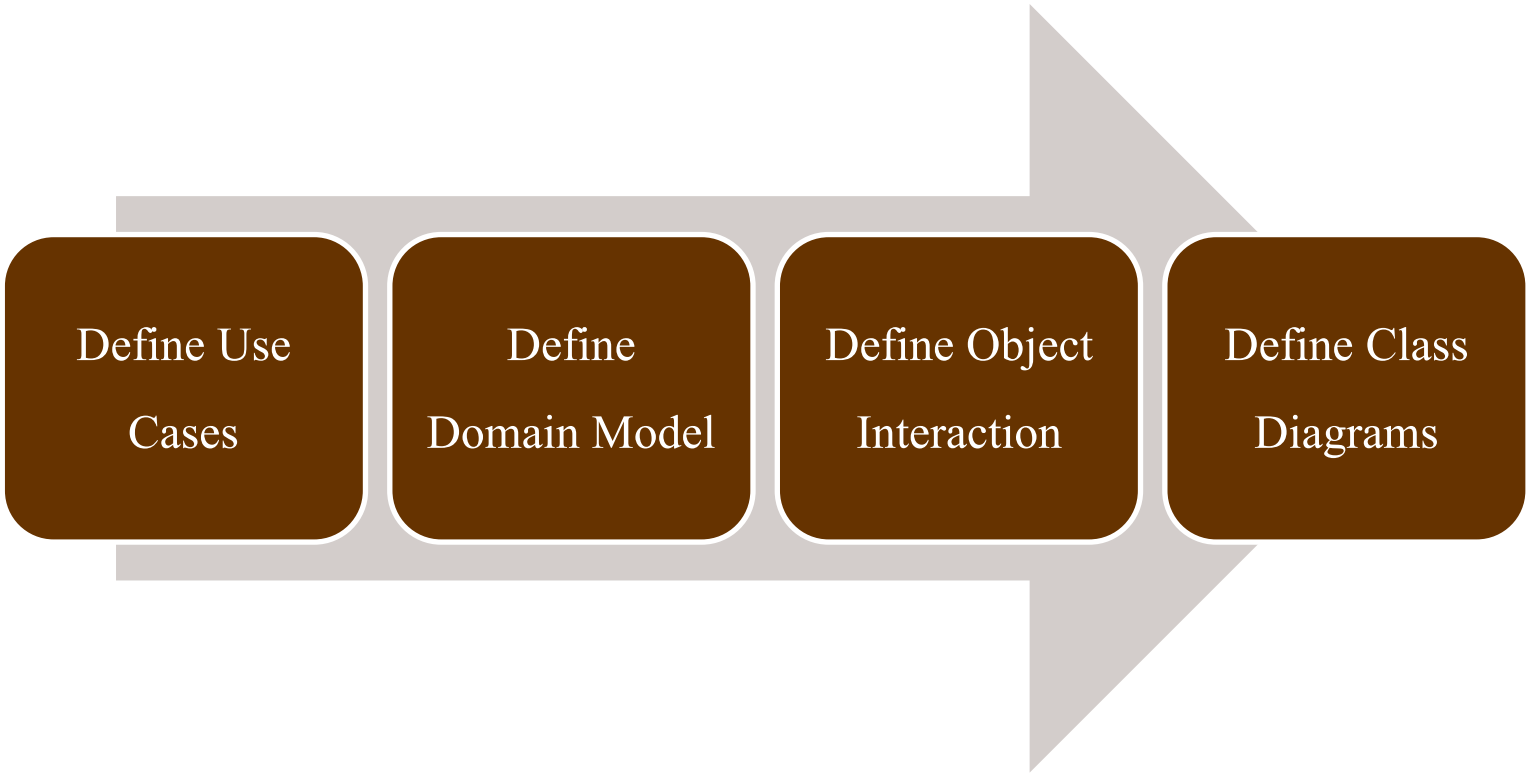
การวิเคราะห์และออกแบบเชิงวัตถุ

- การวิเคราะห์เชิงวัตถุ (Object-Oriented Analysis) เน้นการหา และอธิบายวัตถุ หรือแนวคิดในโดเมนของปัญหา (Domain Object) เช่น สำหรับเกม หมุนกาชาเพื่อให้ได้ตัวละคร วัตถุที่เป็นไปได้คือ ตู้กาชา ผู้เล่น เงิน/เหรียญ ของรางวัล
- การออกแบบเชิงวัตถุ (Object-Oriented Design) เน้นการระบุวัตถุในโปรแกรม (Software Object) รวมถึงคุณสมบัติของวัตถุ และการทำงานร่วมกันของวัตถุเหล่านั้น เช่น ผู้เล่น หยอดเงิน ตู้กาชาสุ่มของรางวัลให้



ขั้นตอนในการวิเคราะห์และออกแบบอย่างง่าย

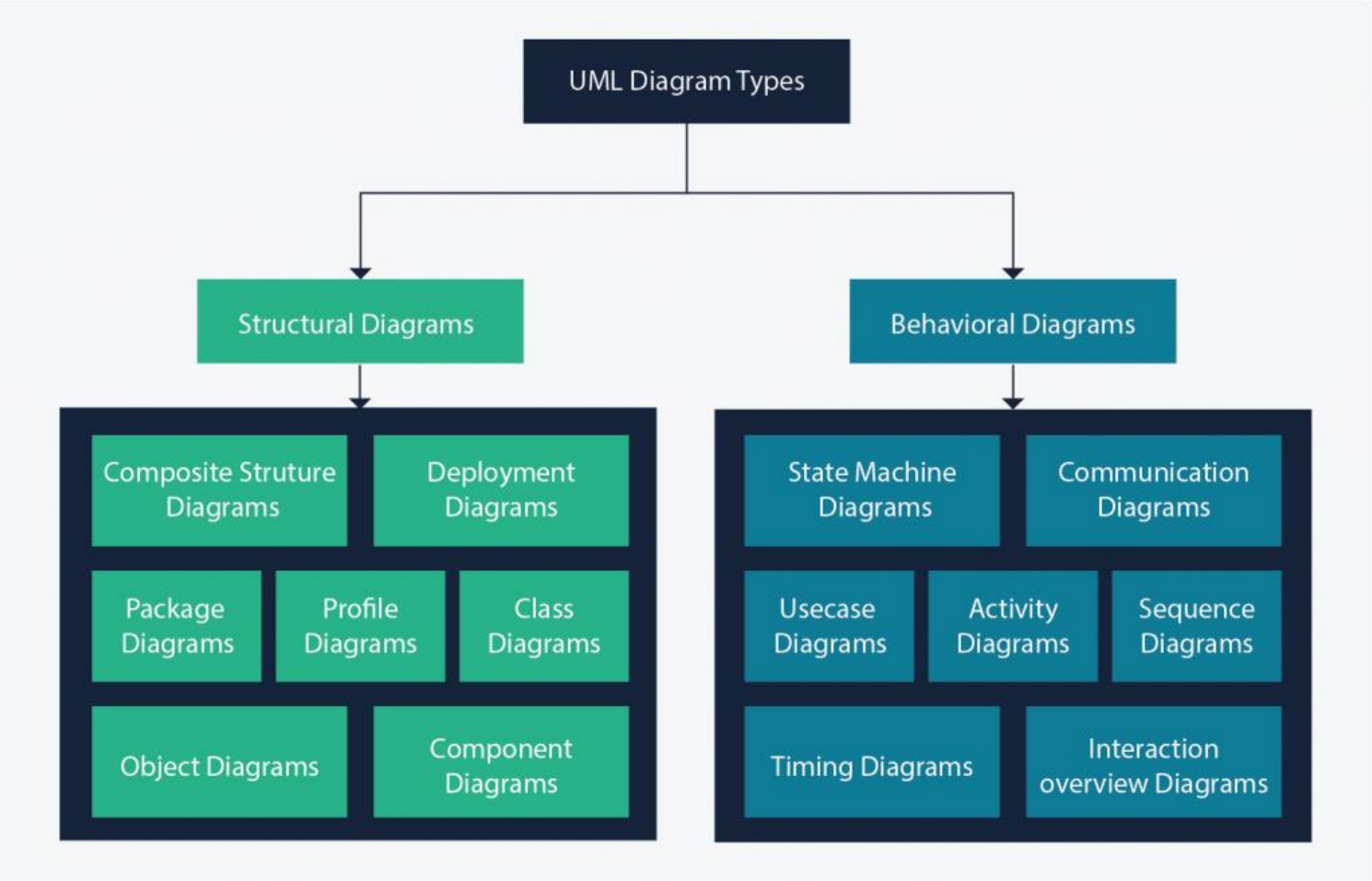
- ในแต่ละขั้นตอนสามารถใช้แผนภาพ UML เพื่อช่วยในการอธิบาย



Unified Modeling Language - UML

- **"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system."**
- เป็นภาษากึ่งรูปนัย (semi-formal)
  - ส่วนประกอบของภาษามีความหมายที่ถูกกำหนดไว้อย่างชัดเจน
- เป็นมาตรฐานสำหรับอธิบายการออกแบบด้วยหลักการของวิธีการเชิงวัตถุ
  - ปัจจุบันถูกจัดการโดย Object Management Group (OMG)
- เป็นภาษารูปภาพที่ใช้ในการทำแบบจำลองของระบบซอฟต์แวร์ แบ่งเป็น
  - แผนภาพแสดงโครงสร้าง (Structure Diagrams)
  - แผนภาพแสดงพฤติกรรม (Behavior Diagrams)

Types of UML Diagrams



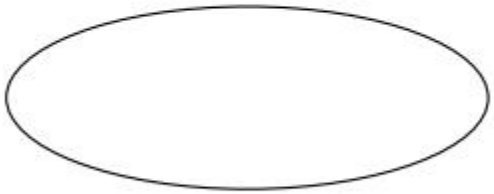
https://creately.com/blog/diagrams/uml-diagram-types-examples/

Define Use Cases

- Use case อธิบายเรื่องหรือสถานการณ์ที่ผู้ใช้จะใช้โปรแกรม
- เขียนอยู่ในรูปรายละเอียด Use Case และ แผนภาพ Use Case Diagram
- รายละเอียด Use Case (Use Case Description)
  - อธิบายเป้าหมายการทำงานของผู้ใช้ และฟังก์ชันของระบบ
  - มีหลายรูปแบบซึ่งมีรายละเอียดต่างกัน แต่ควรระบุ ชื่อ ประเภทผู้ใช้ (Actor) และรายละเอียดการทำงาน
  - เช่น Use Case สุ่มกาชา : ผู้เล่นจ่ายเงินเพื่อกดสุ่มกาชาจากตู้กาชา และเปิดลูกบอลที่ได้เพื่อเอาตัวละครข้างใน
- Use Case Diagram เป็น UML Diagram ที่แสดงภาพรวมของ Use Case ในระบบ และความสัมพันธ์กับผู้ใช้

Use Case Diagram

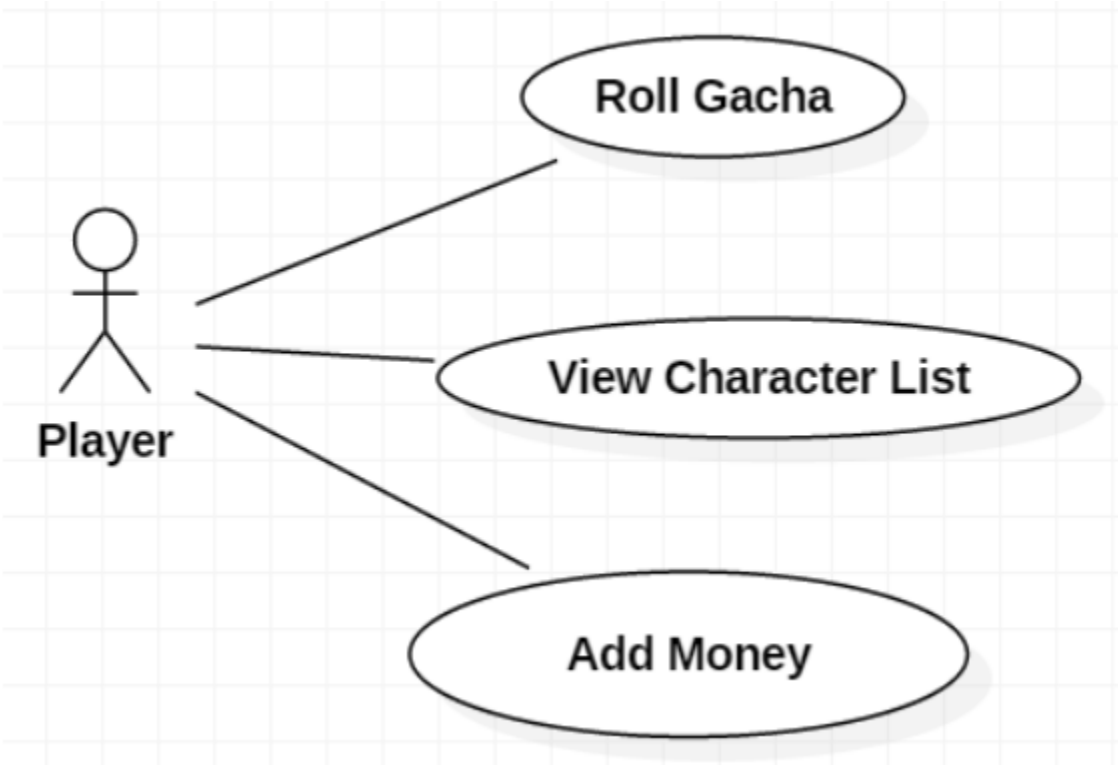
Actor



Use Case

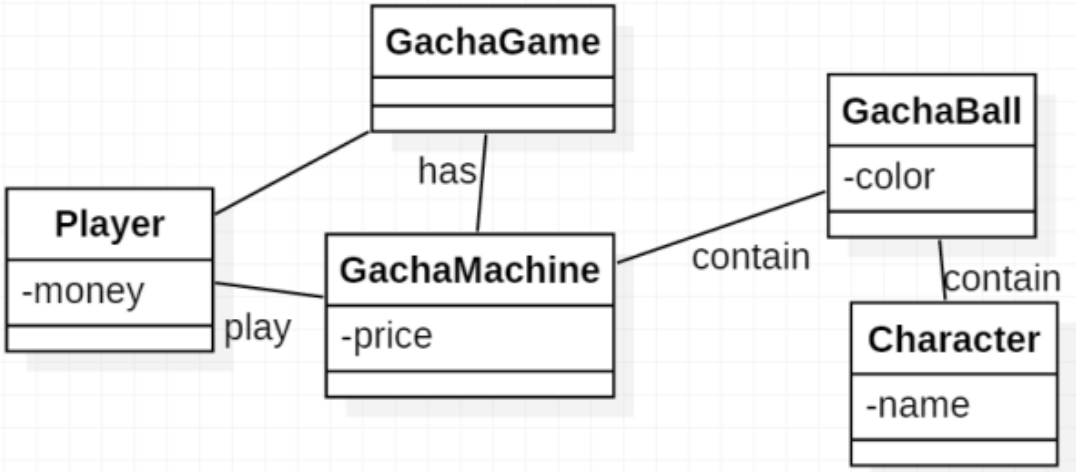
Use Relationship

Example



Define Domain Model

- อธิบายสิ่งที่จะสร้างในมุมมองของวัตถุ ระบุ แนวคิดและ คุณสมบัติที่สำคัญของวัตถุ รวมถึงความเกี่ยวข้องคร่าว ๆ
- สามารถใช้เทคนิคในการหาวัตถุที่กล่าวถึงในสัปดาห์ที่สองเพื่อช่วยในการหาวัตถุและความสัมพันธ์ได้
- ใช้ Class Diagram แบบไม่ลงรายละเอียด ซึ่งได้ผลลัพธ์เป็น Domain Model



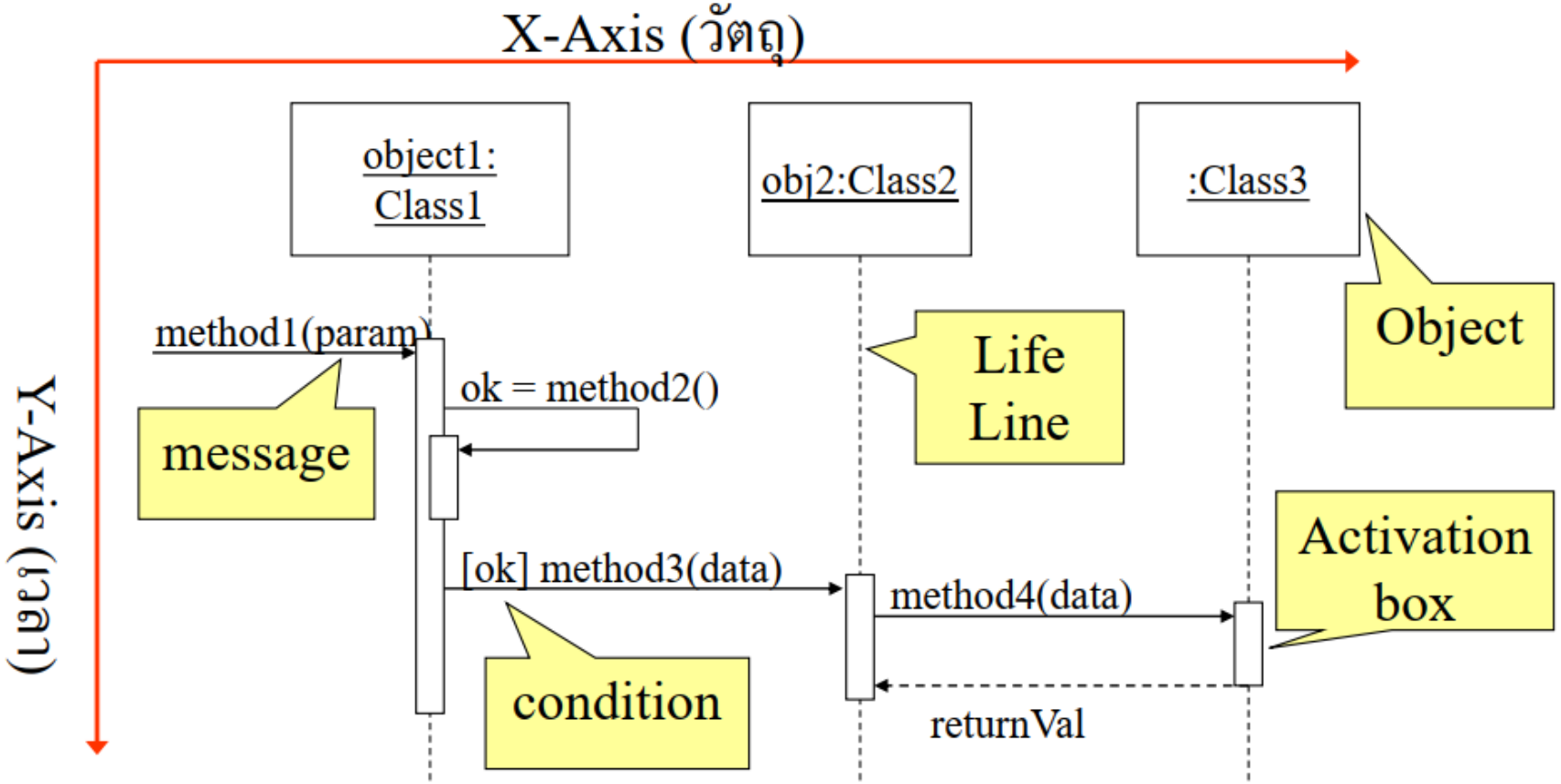
Define Object Interaction

- กำหนดหน้าที่ให้แต่ละวัตถุใน Domain Model
  - กำหนด ผู้เรียกใช้ และขั้นตอน ที่เรียกใช้หน้าที่นั้น
  - เพื่อเพิ่มรายละเอียดคุณสมบัติและหน้าที่ให้กับวัตถุ เพื่อนำไปใช้เป็นวัตถุในโปรแกรม
- แสดงแต่ละกรณีของ Use Case ด้วย Sequence Diagram แสดงลำดับการเรียกใช้หน้าที่ (ส่ง Message) ระหว่าวัตถุในโปรแกรมสำหรับแต่ละกรณีของ Use Case
  - อาจเขียน Sequence Diagram รวมทุกกรณี หรือแยกกรณีก็ได้ แล้วแต่ความซับซ้อนของ Use Case

Sequence Diagram

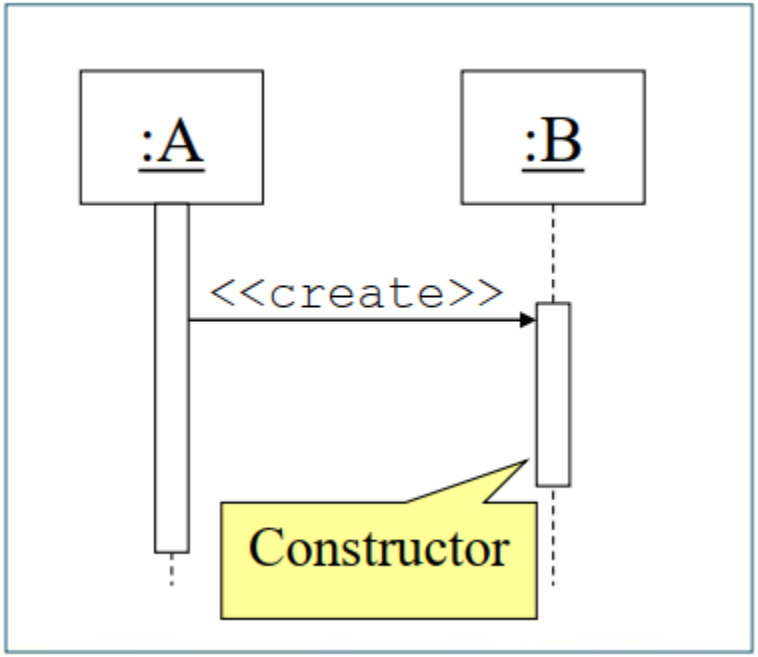
- เป็นหนึ่งในแผนภาพที่ใช้แสดงการปฏิสัมพันธ์ระหว่างวัตถุของ UML
- เน้นเวลาและลำดับของข้อความที่ส่งหากันของวัตถุ
- ประกอบด้วย
  - วัตถุที่มีส่วนร่วมในการทำงาน
  - เส้นแสดงลำดับเวลา
  - เส้นแสดงข้อความที่ส่งหากันระหว่างวัตถุ

Example


X-Axis (วัตถุ)
Y-Axis (เวลา)
object1:
Class1
obj2:Class2
:Class3
Object
method1(param)
message
ok = method2()
Life
Line
[ok] method3(data)
condition
method4(data)
Activation
box
returnVal

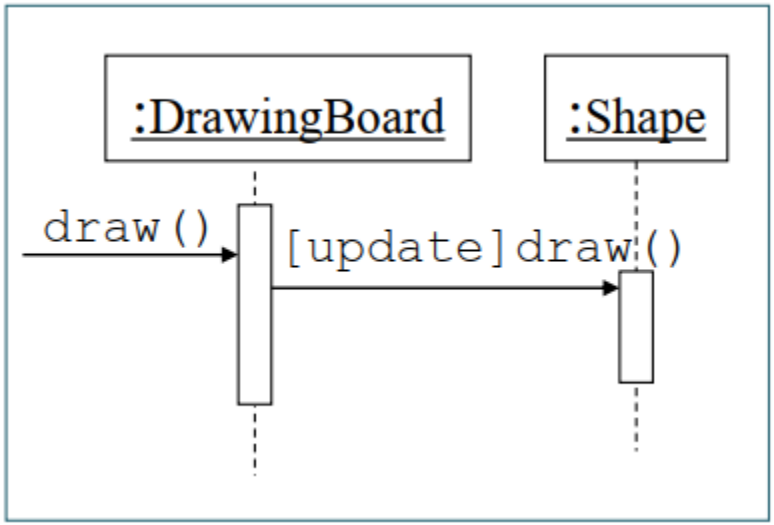
Object Creation Message

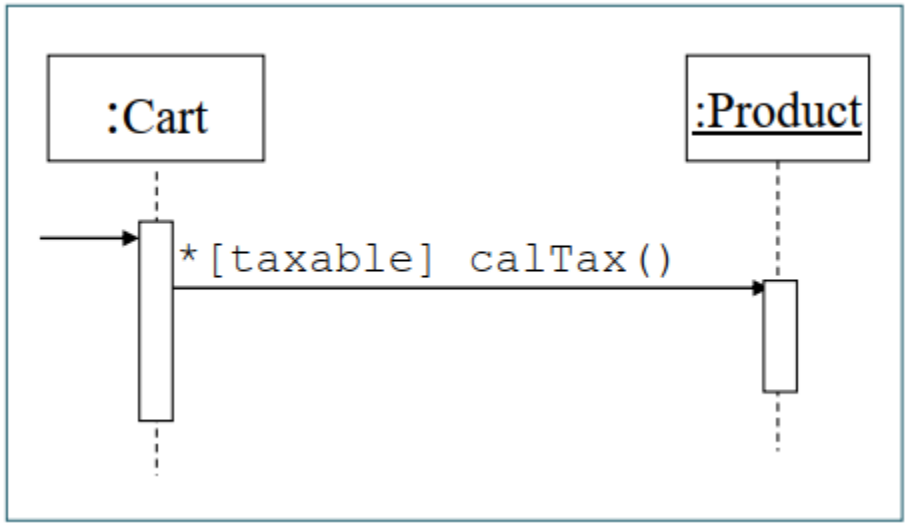
- แสดงเวลาที่วัตถุถูกสร้าง



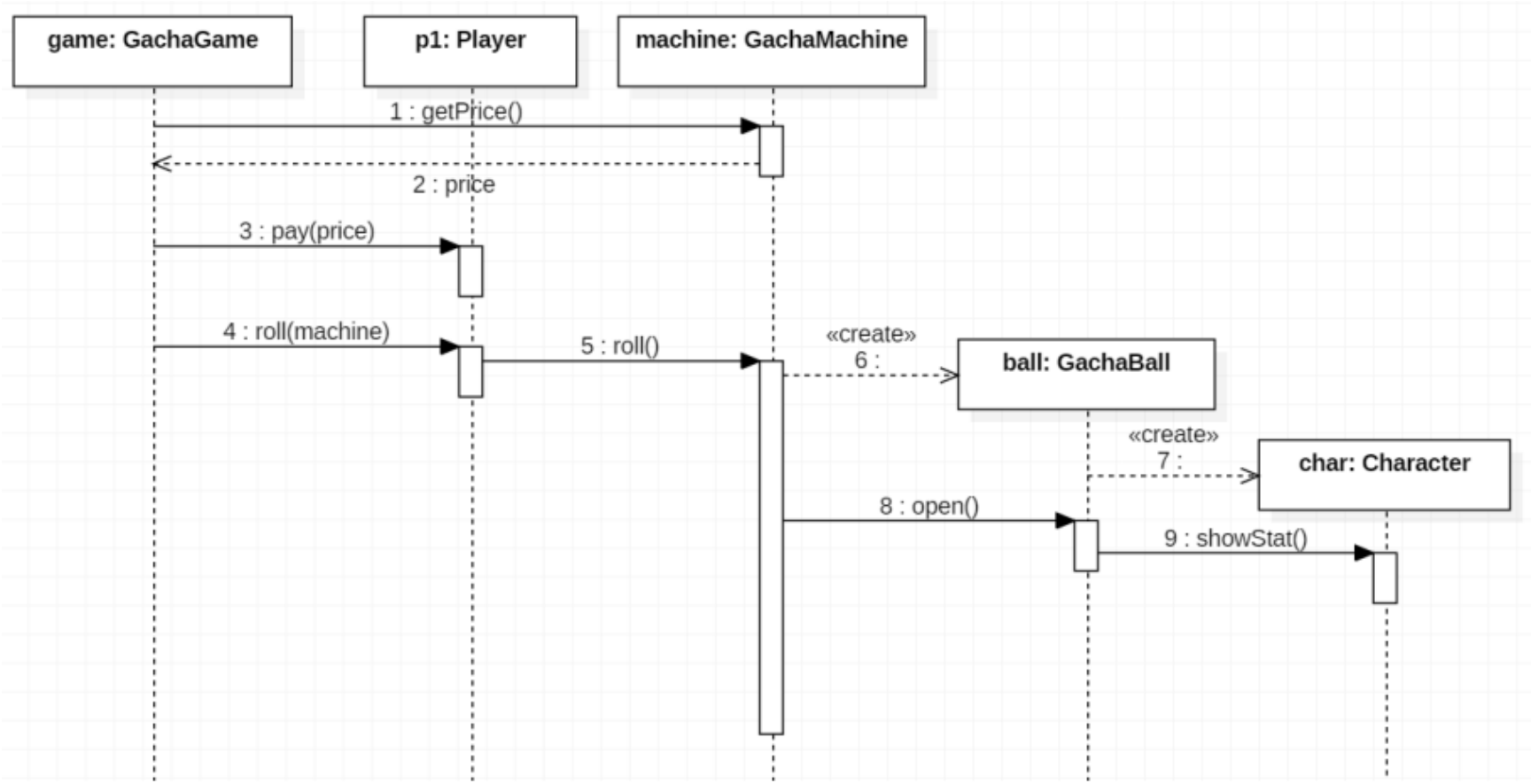
Control Flow Information

- Condition – ใช้ [เงื่อนไข] วางไว้บน message โดย เงื่อนไขจะต้องเป็นจริง จึงจะส่ง Message นั้น
- Loop – ใช้ *[เงื่อนไข] วางไว้บน message โดย message จะถูกส่งตราบเท่าที่เงื่อนไขเป็นจริง





Example



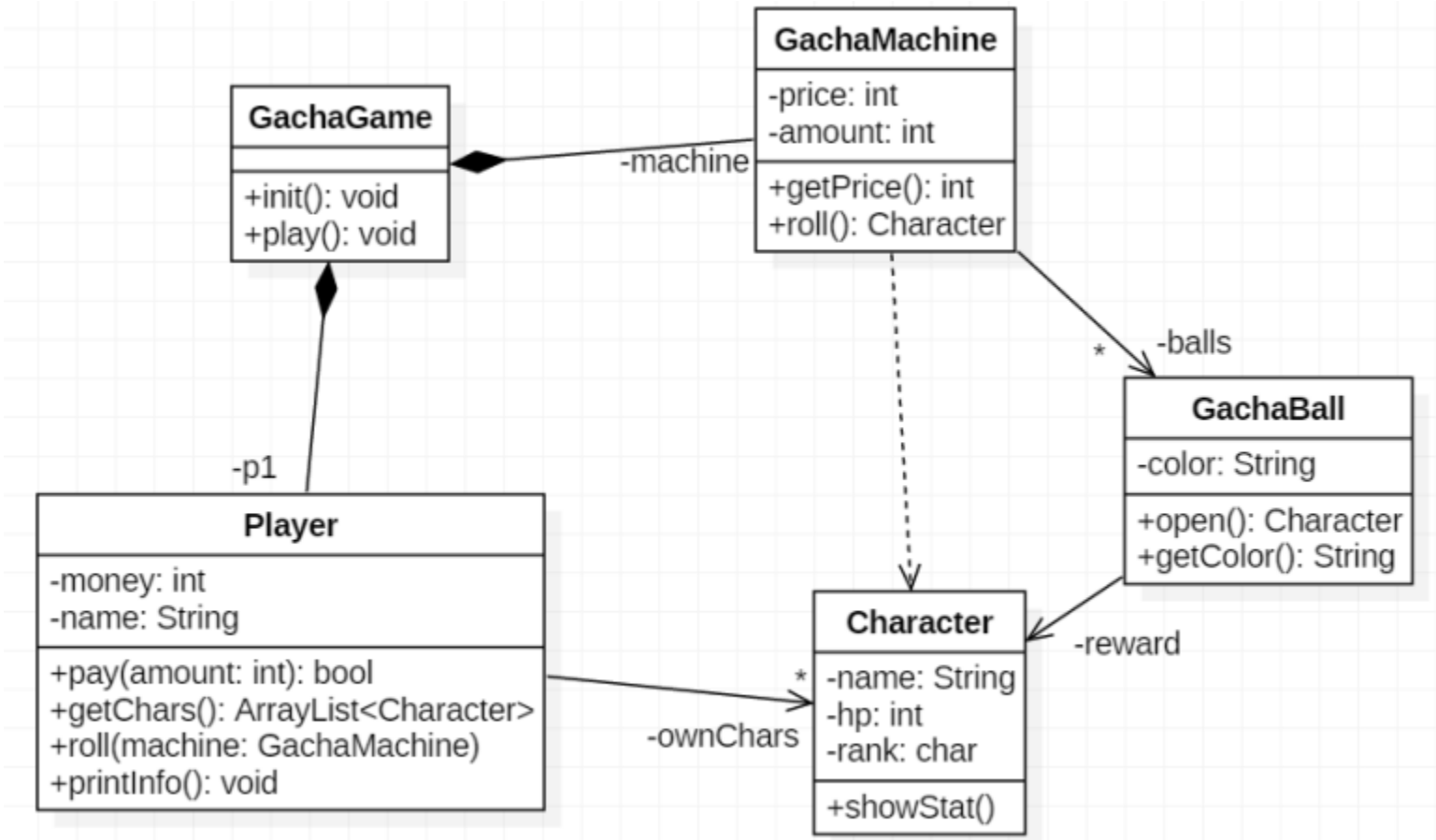
Defining Sequence Diagram

- สร้าง Sequence Diagram โดยพิจารณา Use Case ที่มี ทีละ Use Case โดยไม่ต้องคิดถึงความเกี่ยวข้องของแต่ละ Use Case
- ดูวัตถุที่มีใน Domain Model และระบุวัตถุที่ต้องร่วมทำกิจกรรมใน Use Case นั้น
  - อาจมีการเพิ่มวัตถุใน Domain Model เมื่อพบว่าขาดวัตถุที่เหมาะสมจะทำหน้าที่ที่จำเป็นในการทำงานตาม Use Case
- วางวัตถุเรียงต่อกันในแนวนอน โดยพยายามนำวัตถุที่เป็นจุดเริ่มต้นของการทำงานไว้ด้านซ้ายสุด แล้วเรียงลำดับต่อ ๆกันไป
- ลาก Life Line และ Message ที่จะต้องส่งต่อกัน

Define Class Diagram

- ใช้ข้อมูลที่ได้เพิ่มเติมระหว่างการสร้าง Sequence Diagram เพื่อกำหนดคุณสมบัติ และ เมท็อดของคลาสใช้เป็นแผนผังในการสร้างโปรแกรม
  - Message ที่ส่งกลายเป็นเมท็อด หรือหน้าที่ของคลาสที่ถูกเรียก
  - ข้อมูลที่ส่งระหว่างกันกลายเป็นคุณสมบัติที่ต้องมี
- คลาสที่ได้จะมีข้อมูลคุณสมบัติและหน้าที่อย่างละเอียด
- แสดงด้วย Class Diagram
  - ไม่เหมือน Domain Model ตรงที่ Class Diagram นี้เน้นแสดงคุณสมบัติของวัตถุ (Software Object) ในโปรแกรมแทนที่จะเป็นวัตถุจริงในโดเมน

Example


GachaMachine
-price: int
-amount: int
+getPrice(): int
+roll(): Character
GachaGame
+init(): void
+play(): void
-machine
-balls
*
GachaBall
-color: String
+open(): Character
+getColor(): String
-p1
Player
-money: int
-name: String
+pay(amount: int): bool
+getChars(): ArrayList<Character>
+roll(machine: GachaMachine)
+printInfo(): void
Character
-name: String
-hp: int
-rank: char
+showStat()
-reward
*
-ownChars

กรณีศึกษา

- โปรแกรมเกมผจญภัยแบบ Text-Based โดยมีกฎของเกมดังต่อไปนี้
  - โลกของเกมประกอบไปด้วยห้อง 4 ห้อง แต่ละห้องจะมีจุดเคลื่อนย้ายไปห้องอื่น (ที่ warp) โดยจุดเคลื่อนย้ายนี้จะทำงานแบบสุ่ม คือจะย้ายผู้เล่นจากห้องไหนไปห้องไหนก็ได้ (**อาจอยู่ห้องเดิมก็ได้**)
  - แต่ละห้องจะมีสัตว์ประหลาดเฝ้าห้องอยู่หนึ่งตัว โดยเมื่อผู้เล่นเข้ามาอยู่ในห้องสัตวประหลาดตัวนี้อาจจะนอนหลับอยู่หรือตื่นอยู่ โดยเปอร์เซ็นต์ที่จะตื่นเป็น 40% ถ้าสัตว์ประหลาดหลับอยู่จะไม่ทำอะไรผู้เล่นแต่ถ้าสัตว์ประหลาดตื่นจะกินผู้เล่น ถ้าสัตว์ประหลาดในห้องหนึ่งเคยโดนผู้เล่นฆ่าไป ในรอบถัดไปสัตว์ประหลาดตัวนั้นจะมีโอกาสฟื้นชีวิตเป็น 30% โดยไม่สนใจว่าผู้เล่นจะเข้ามาในห้องในรอบนั้นหรือไม่
  - เมื่อเริ่มเล่นผู้เล่นมีคะแนนเริ่มต้นเป็น 0 มีเนื้อ 2 ชิ้นในมือ และ มีปืน 1 กระบอก ปืนสามารถใช้ฆ่าสัตว์ประหลาดได้ และถ้าฆ่าสำเร็จจะได้เนื้อมา 1 ชิ้น โดยเมื่อเริ่มต้นปืนมีประสิทธิภาพ 100% และประสิทธิภาพของปืนจะเสื่อมไป 10% ทุกครั้งที่ยิง (อีกนัยหนึ่งคือโอกาสที่ยิงแล้วยิง ไม่ออกจะเพิ่มทีละ 10% หลังจากยิง)

กรณีศึกษา (ต่อ)

- ในแต่ละรอบผู้เล่นจะทอยลูกเต๋า 1 ลูก และจะมีโอกาสเคลื่อนย้ายห้องไปเป็นจำนวนเท่ากับหน้าลูกเต๋าที่ทอยได้ ในระหว่างเคลื่อนย้ายผู้เล่นสามารถจะเลือกว่าจะหยุด ในแต่ละห้องที่ไปถึงหรือไม่
  - ถ้าไม่หยุด ได้ 10 คะแนนต่อการเคลื่อนย้าย 1 ครั้ง
  - ถ้าหยุด ได้ 50 คะแนนและถือว่าผู้เล่นเข้ามาอยู่ในห้อง ครั้งสุดท้ายของการเคลื่อนย้ายผู้เล่นต้องหยุดเสมอ
  - เมื่อเริ่มต้นในแต่ละรอบเกมจะตั้ง ค่าสถานะของสัตว์ประหลาดในทุกห้องว่าหลับหรือตื่น และตรวจว่าสัตว์ประหลาดที่ตายในรอบที่แล้ว จะฟื้นหรือไม่
- เมื่อผู้เล่นเข้ามาอยู่ในห้อง
  - ถ้าสัตว์ประหลาดตายไปแล้วหรือหลับอยู่ผู้เล่นจะปลอดภัย และเลือกได้ว่าจะไปยังห้องต่อไป หรือ ถ้าสัตว์ประหลาดหลับจะสามารถยิงเพื่อฆ่าสัตว์ประหลาด ถ้ายิงแล้วสัตว์ประหลาดตายผู้เล่นจะได้เนื้อ 1 ชิ้น แต่ถ้าสัตว์ประหลาดไม่ตายจะตื่นขึ้นมา
  - ถ้าสัตว์ประหลาดตื่น (ไม่ว่าจะตื่นอยู่แล้วหรือโดนยิงตื่น) จะโจมตีผู้เล่น ผู้เล่นสามารถเอาชีวิตรอดได้สองวิธีคือ a) โยนเนื้อให้ 1 ชิ้นถ้าผู้เล่นมีเนื้ออยู่ในมือ สัตว์ประหลาดจะกินเนื้อแล้วหลับไป และเข้ากรณีที่ i) ใหม่ หรือ b) ยิงสัตว์ประหลาด (ถ้าไม่มีเนื้อในมือจะต้องยิงสัตว์ประหลาดที่ตื่นเสมอ) ถ้ายิงแล้วสัตว์ประหลาดตายผู้เล่นจะได้เนื้อ 1 ชิ้น ถ้ายิงไม่ออกสัตว์ประหลาดจะกินผู้เล่นและจบเกม

สรุปการเรียนในวันนี้

- เข้าใจแนวคิดเกี่ยวกับการวิเคราะห์และออกแบบเชิงวัตถุ
- เข้าใจการสร้างและแปลความหมายของแผนภาพ UML เบื้องต้น
  - Use Case Diagram
  - Sequence Diagram
  - Class Diagram