Software Development

FRA 162: Scientific Process
FRA 163: Science in Motion Project

Rattanachai Ramaithitima, PhD
Institute of Field Robotics (FIBO)
March 28, 2025

Final Simulation Requirements (M3)

องค์ประกอบของโปรแกรม Simulation

- สามารถใส่ค่าพารามิเตอร์ต่างๆ เพื่อใช้ในการคำนวณปรับแต่งระบบในการตีลูกควอชให้ลงในตำแหน่งที่ต้องการได้
- Simulation จะทำงาน หลังจากกดปุ่ม Start ใน GUI
- Simulation จะต้องมีปุ่ม Reset เพื่อเปลี่ยนระบบให้กลับสู่สถานะเริ่มต้น
- มีการประยุกต์ใช้ OOP ในการออกแบบ Simulation
- มีการออกแบบ Back-end ผ่าน UML Diagram
- มีการออกแบบ Front-end ที่ง่ายต่อการใช้
- สามารถรองรับการเปลี่ยนแปลงข้อมูลของสนามทดสอบในรูปแบบอื่นๆได้

M2 (Programming requirements)

Software Requirements • Functional Requirements • Non-functional Requirements	Presentation: key requirements Report: full requirements
Front-end Design • GUI of your system	Presentation and report: show full front-end design of the system
Back-end Design • UML Diagram	Presentation and report: show full back-end design of the system, and explanation of every class, variables, and functions in the diagram.

Software Requirements

Functional Requirements

What are they?

- describe the specific functionalities or behaviors required for satisfying **user needs, business requirements, and stakeholder expectations**.
- detail **what the software should do** and how it should behave under certain conditions
- Defined at component level

Ref: https://www.geeksforgeeks.org/what-are-functional-requirements-in-system-design-examples-definition/

Functional Requirements

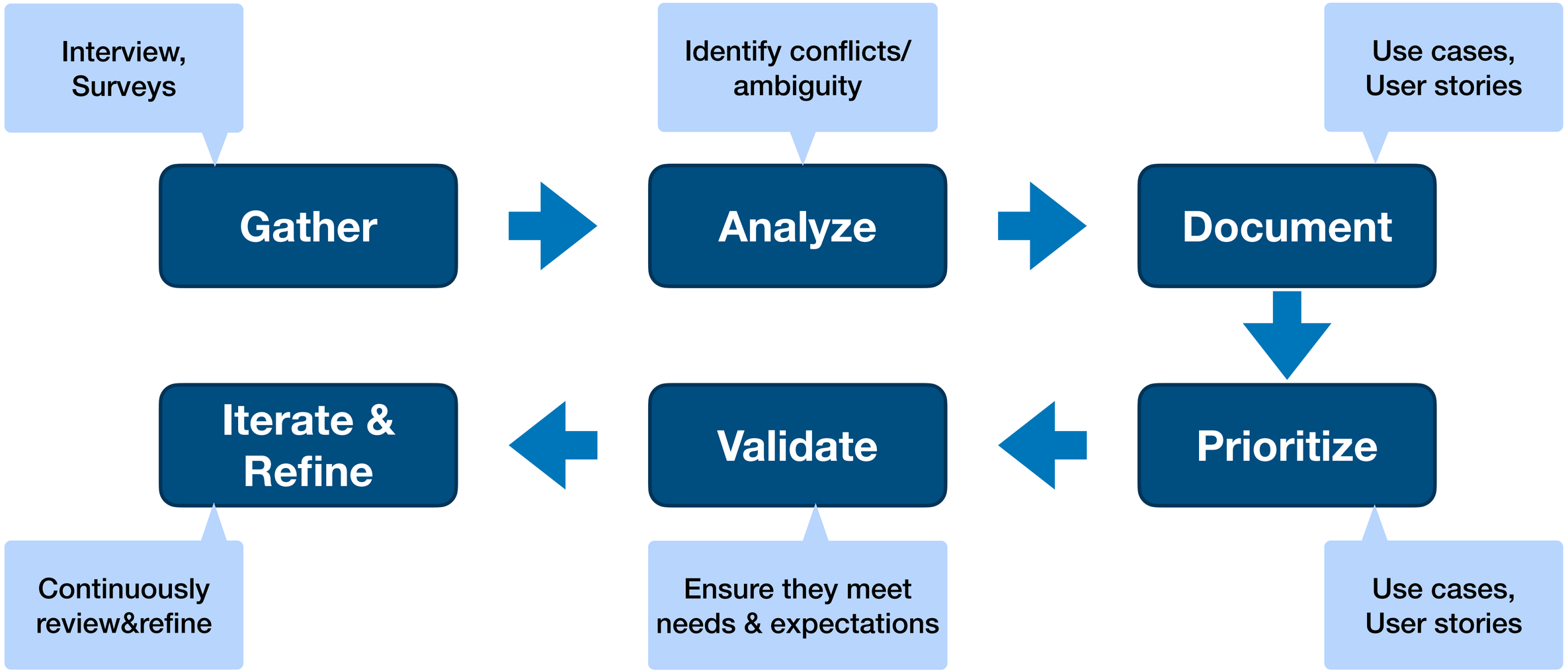
Examples

- Users should be able to register for a new account by providing their name, email address, and password
- Users should be able to proceed to checkout, enter billing and shipping information, and complete the purchase.
- Administrators should be able to manage orders, view order details, and update order statuses as needed.

Ref: https://www.geeksforgeeks.org/what-are-functional-requirements-in-system-design-examples-definition/

Functional Requirements

How to identify ones?



Ref: https://www.geeksforgeeks.org/how-to-identify-functional-requirements/

Non-functional Requirements

What are they?

- define the **characteristics/quality attributes** of a software system
- describe **how the system should perform**
- deal with issues like scalability, maintainability, performance, portability, security, reliability, etc.

Ref: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

Non-functional Requirements

Examples

- Performance: the expected performance such as response time, throughput, resource utilization
  - The online game server must be able to handle 50,000 concurrent users with latency less than 100ms.
  - The trading platform must be able to handle more than 20000 concurrent transactions within 1 second.
  - The system must be able to process 5TB of data within 30 minutes.

Ref: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

Non-functional Requirements

Examples

- Reliability: ability of the software system to perform its functions consistently and accurately over time
  - (Bad example): Google share went down 8 percent as Bard AI gives inaccurate info during the advertisement
  - (Bad example): Robinhood (trading app) blocked trading of certain stocks during meme stock frenzy in January 2021

Ref: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/
https://www.aljazeera.com/economy/2023/2/8/google-shares-tank-8-as-ai-chatbot-bard-flubs-answer-in-ad
https://www.cnet.com/personal-finance/investing/robinhood-backlash-what-you-should-know-about-the-gamestop-stock-controversy/

Non-functional Requirements

Examples

- Security: measures taken to protect the software system from unauthorized access, attacks, or data breaches
  - Logging into apple account requires authentication from other apple device
  - Signing into gmail on new devices will send confirmation email to users
- Usability: ease of use and user-friendliness
- Maintainability: ease with which the software system can be modified, updated, and maintained over time.

Ref: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/

Comparison

Functional VS Non-Functional Requirements

Functional Requirements	Non-Functional Requirements
Defines a system/its components	Define the quality attribute
What should the software system do?	How should the software system fulfill functional requirements?
Specified by users	Specified by engineers
Defined at component level	Applied to system as a whole
Captured in use case	Captured as quality attribute
Testing: system test, integration test, end-to-end test, API test	Testing: benchmarking, stress test, usability test, security test

Ref: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

Examples

Online-shopping App

- Functional Requirements: (user)
  - User should be able to create a new account by providing name/e-mail/password
  - Registered user should be able to log in using account/password
  - User should be able to search for the item they want
  - User should be able to add items to cart
  - User should be able to proceed to checkout, input shipping & billing address, add payment info, and complete the purchase

Examples

Online-shopping App

- Functional Requirements: (admin)
  - Admin should be able to check/track/modify order details as needed
  - Admin should be able to view/response to inquiry from customers
  - Admin should be able to manage the list of merchants or inventory of merchandises

Examples

Online-shopping App

- Non-Functional Requirements:
  - Login time should take less than 5 seconds
  - Logging in to new device should either send email to users/require two factor authentication or both
  - Search function should provide accurate/relevant items and take less than 10 seconds to return the results
  - Inventory status should be updated in real time to properly reflect their availability
  - Shopping cart should be easy to review/modify
  - Password and other sensitive user info such as payment info should be encrypted and not viewable by anyone (even admin)

Worksheet V (งานคู่)

Part1: Functional & Non-functional Requirements

- ให้เลือกโปรแกรมที่สนใจมา 1 โปรแกรมและเขียนลิสต์ Functional และ Non-functional requirements ของโปรแกรมนั้นๆ มาอย่างน้อยประเภทละ 5 รายการ

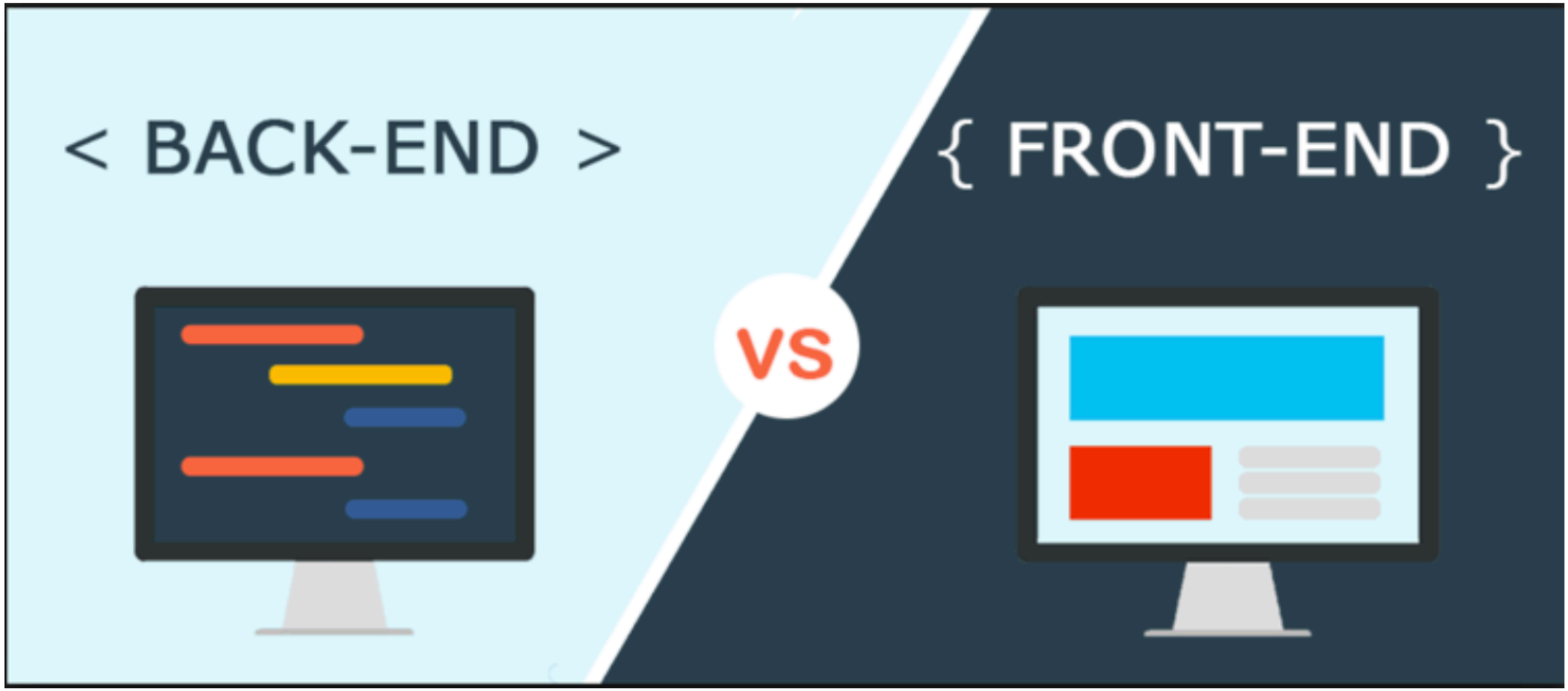
Software Design

Software Design

Intro

- A process to create a blueprint of a software. Usually it refers to all activities following requirements specification and before programming.
  - Interface Design
  - Architectural Design
  - Detailed Design

Software Design



Front-end and Back-end Development

Overview

- Front End
  - What we see
  - UI, images, buttons, text, information,…
- Back End
  - What we can’t see
  - Codes behind each button, interactions between objects,…

Front-end Development

Let’s try

https://userinyerface.com/

Front-end Development

Overview

- Developing a GUI (Graphical User Interface) that can communicate software's functionalities to users.
- **Goal:** Design whatever so that the user can use your software clearly and easily

Front-end Development

Overview

- **Golden Rules:**
  - Place users in control
  - Reduce users’ memory loads
  - Make the Interface Consistent

Ref: https://theomandel.com/resources/golden-rules-of-user-interface-design/

Front-end Development

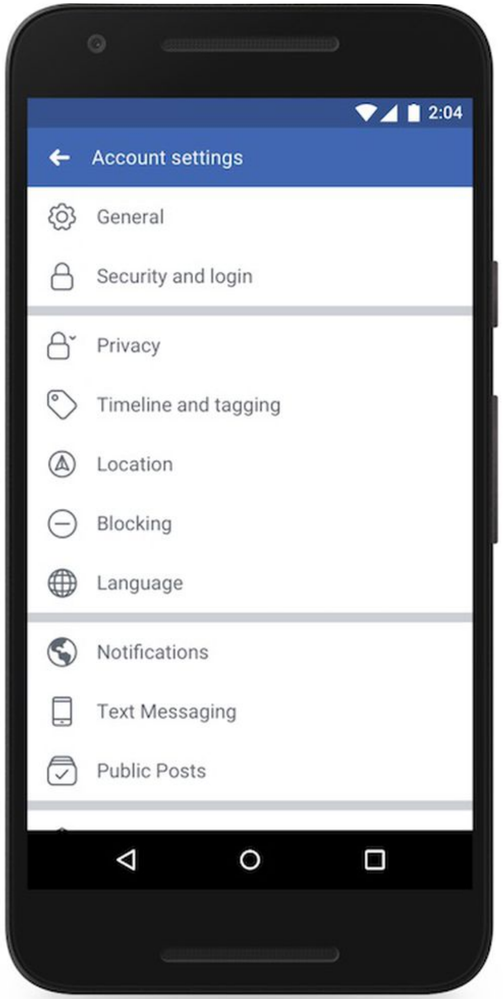
Place Users in Control

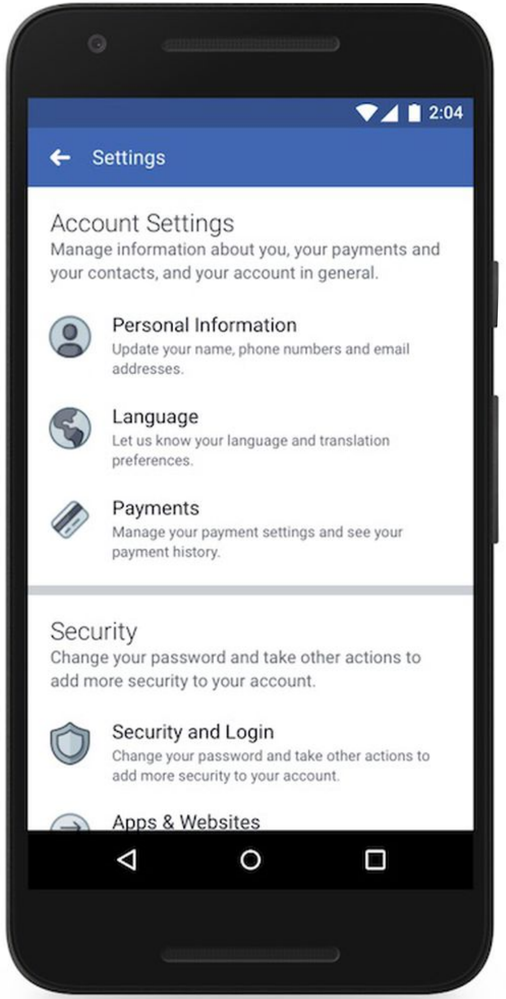
1. Use modes judiciously (modeless)
2. Allow users to use either the keyboard or mouse (flexible)
3. Allow users to change focus (interruptible)
4. Display descriptive messages and text(Helpful)
5. Provide immediate and reversible actions, and feedback (forgiving)
6. Provide meaningful paths and exits (navigable)
7. Accommodate users with different skill levels (accessible)
8. Make the user interface transparent (facilitative)
9. Allow users to customize the interface (preferences)
10. Allow users to directly manipulate interface objects (interactive)

Ref: https://theomandel.com/resources/golden-rules-of-user-interface-design/

Front-end Development

Place Users in Control





Ref: https://www.macrumors.com/2018/03/28/facebook-details-revamped-privacy/

Front-end Development

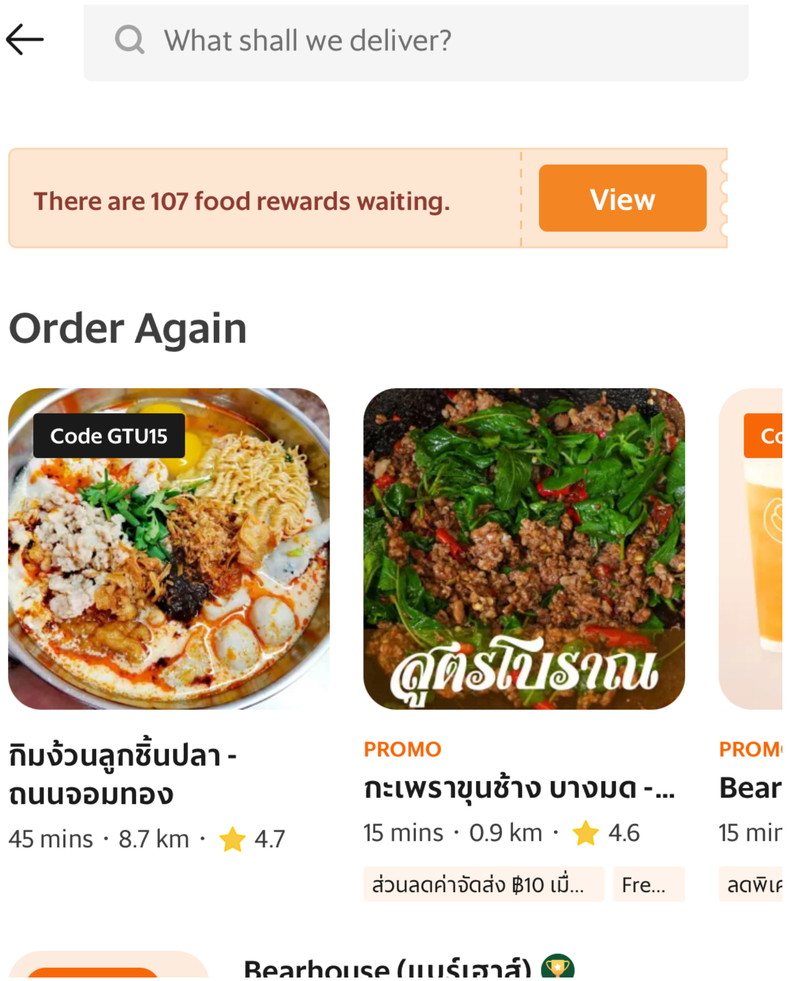
Reduce Users' Memory Load

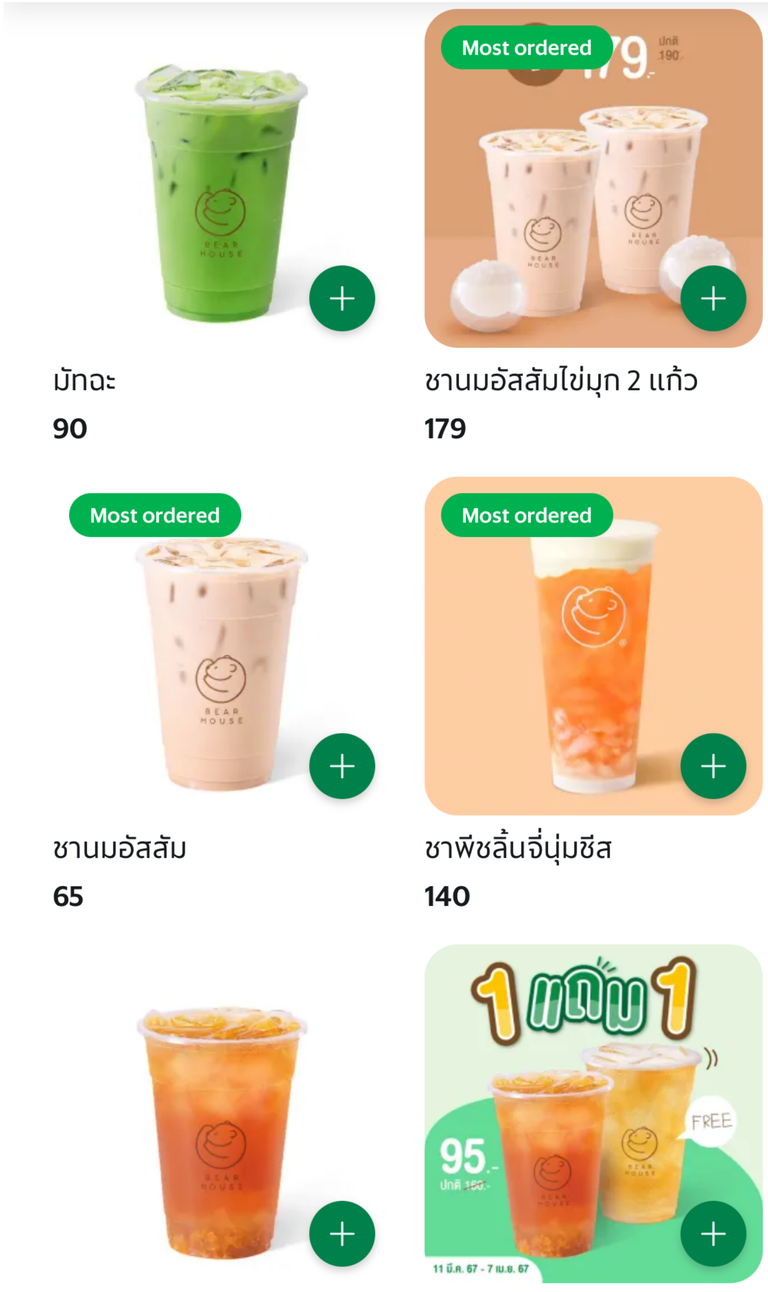
1. Relieve short-term memory (remember)
2. Rely on recognition, not recall (recognition)
3. Provide visual cues (inform)
4. Provide defaults, undo, and redo (forgiving)
5. Provide interface shortcuts (frequency)
6. Promote an object-action syntax (intuitive)
7. Use real-world metaphors (transfer)
8. User progressive disclosure (context)
9. Promote visual clarity (organize)

Ref: https://theomandel.com/resources/golden-rules-of-user-interface-design/

Front-end Development

Reduce Users' Memory Load





Front-end Development

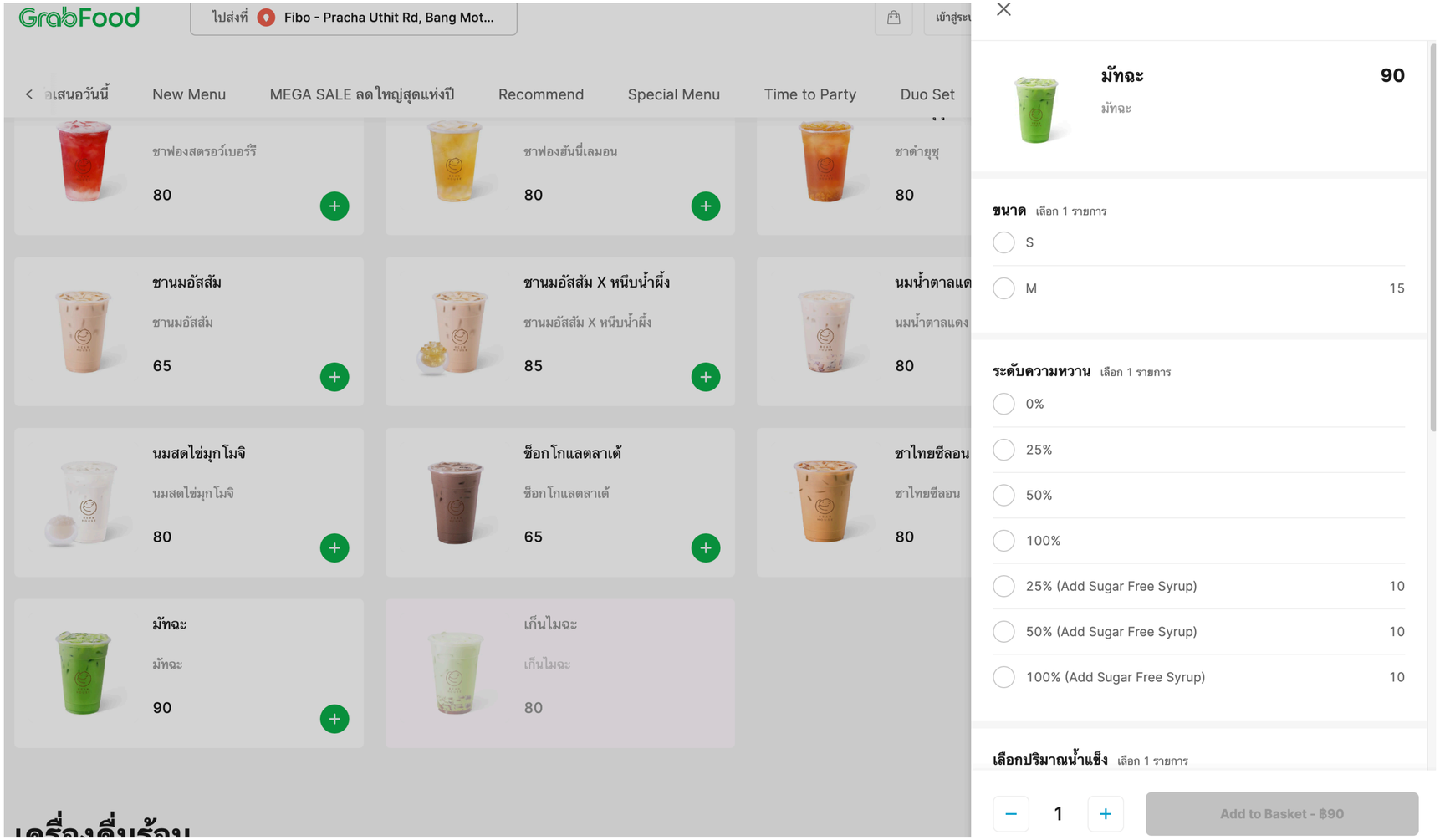
Make the Interface Consistent

1. Sustain the context of users' tasks (continuity)
2. Maintain consistency within and across products (experience)
3. Keep interaction results the same (expectations)
4. Provide aesthetic appeal and integrity (attitude)
5. Encourage exploration (predictable)

Ref: https://theomandel.com/resources/golden-rules-of-user-interface-design/

Front-end Development

Make the Interface Consistent



Front-end Development

Other techniques

- Colors matter.
  - Colors themselves are meaningful.
  - Green can represent approval. Red can represent denial.
  - Black-out/ Grey-out box can represent impassable/unpressable button.

COLOR EMOTION GUIDE

Ref: https://clairification.com/2020/08/31/hue-color-can-mean-nonprofit-marketing-strategy/

Back-end Development

Software Architecture

- Just like when we want to build anything, we need to have a plan to build it.
  - When you build a robot, you need to have a solid structure design first.
  - Same as building a software, you need to design a system such that it is not a mess.
  - This planning is called software architecture.

Back-end Development

Software Architecture

- High level structures of a software
  - Overview
  - Big picture

- It describes
  - Software elements
  - Relationships
  - Properties of both relationships and elements

Back-end Development

OOP (FRA 142)

- We create an object, and manipulate that object.
  - To create an object, we need to create a class, and functions for that object.

- What if we have two objects interacting with each other?
  - We have a relationship.

- What if we have many objects interacting with each other?

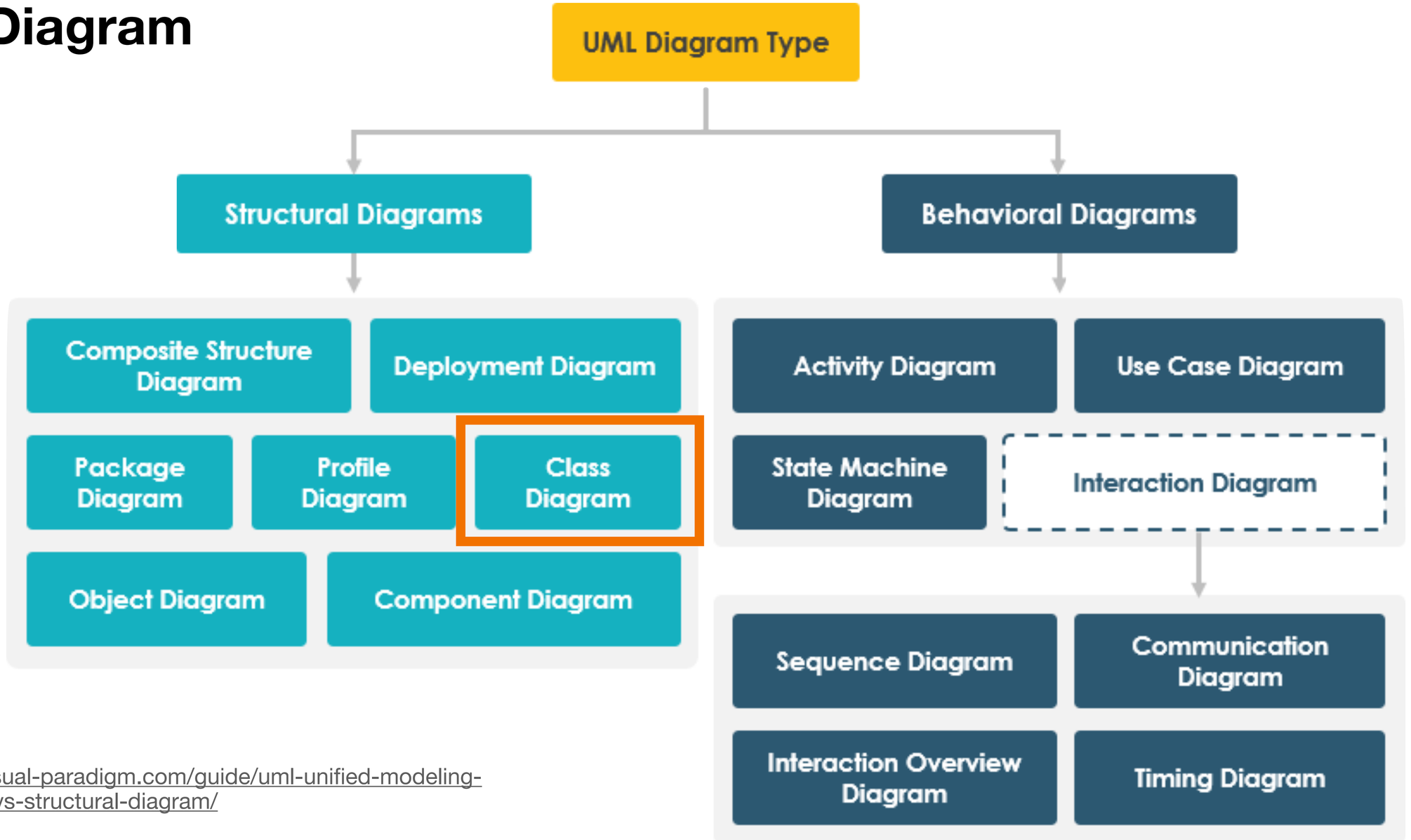
Back-end Development

UML Diagram

- We have classes
  - Superclass
  - Subclass
  - Interface (if any)
- We put these together and labels their relationships.
  - Unified Modeling Language (UML) Diagram:
    - A graphical way to describe software systems.

Back-end Development

UML Diagram

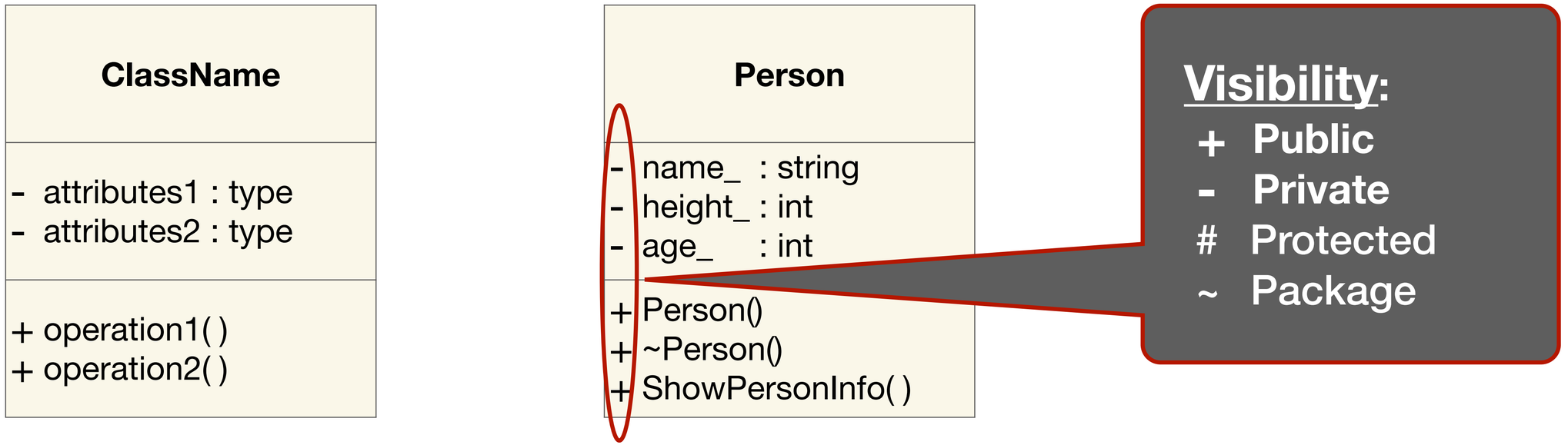


Ref: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/

Back-end Development

Class Diagram

- Has 3 sections: class name, attributes, and operations



Back-end Development

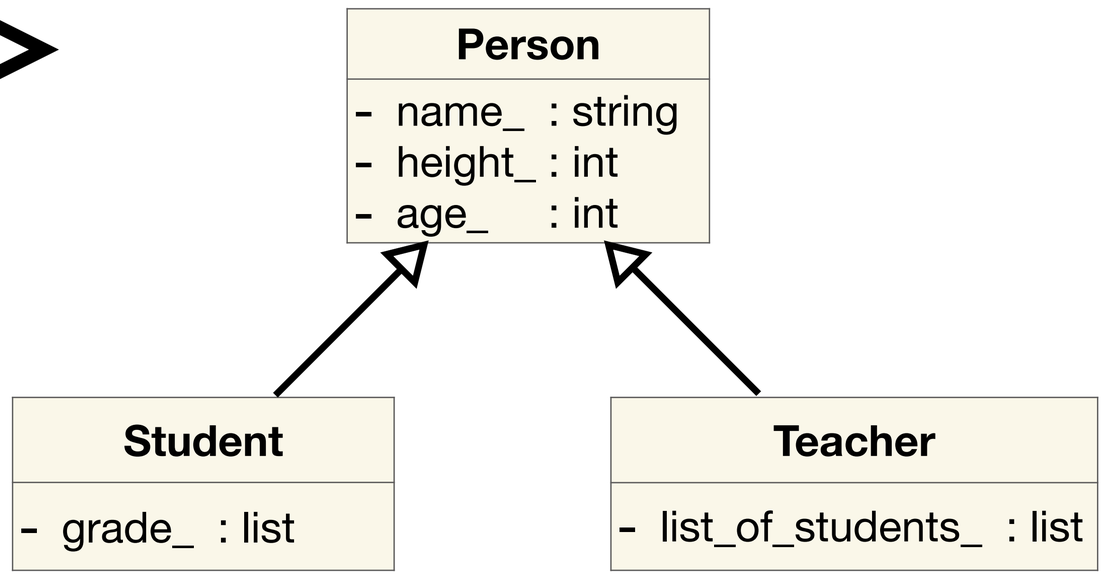
Class Diagram

- Relationships: define connection between multiple classes
  - Inheritance 
  - Association 
    - Aggregation 
    - Composition 

Back-end Development

Class Diagram

- Relationships: define connection between multiple classes
  - Inheritance
    - Defines *subclass* and *superclass*
    - Person and student/teacher



Back-end Development

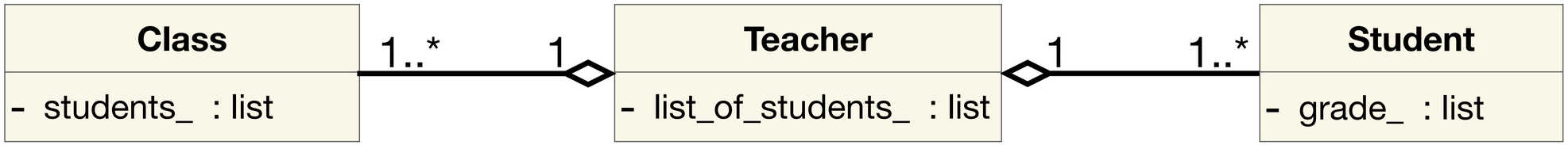
Class Diagram

- Relationships: define connection between multiple classes
  - Association: ——————
    - Has general version to describe associate relationship
    - General form is bidirectional relationship but can have more specific ones where it is uni-directional

Back-end Development

Class Diagram

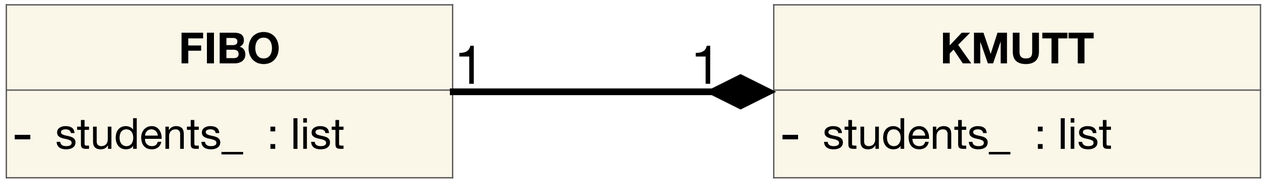
- Relationships: define connection between multiple classes
  - Aggregation 
    - A variant of specific association relationship
    - A part-whole or part-of relationship: containers and contents
    - If container is destroyed, the contents are usually not destroyed
      - Teachers has a class/students to teach



Back-end Development

Class Diagram

- Relationships: define connection between multiple classes
    - Composition 
        - A variant of specific association relationship
        - A whole-part relationship: containing and contained classes
        - If container is destroyed, the contained class are also destroyed
            - University and its department

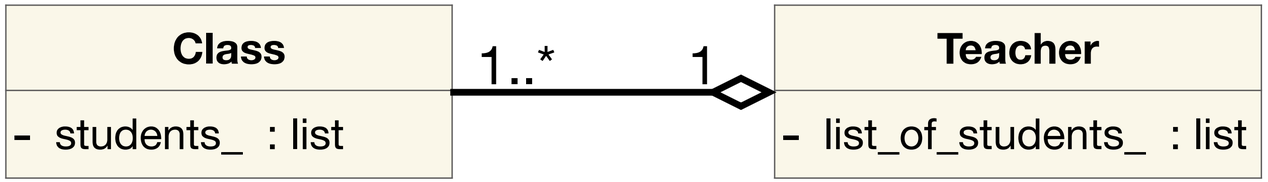


Back-end Development

Class Diagram

- Multiplicity
  - the range of number of objects that participate in the association from the perspective of the other end

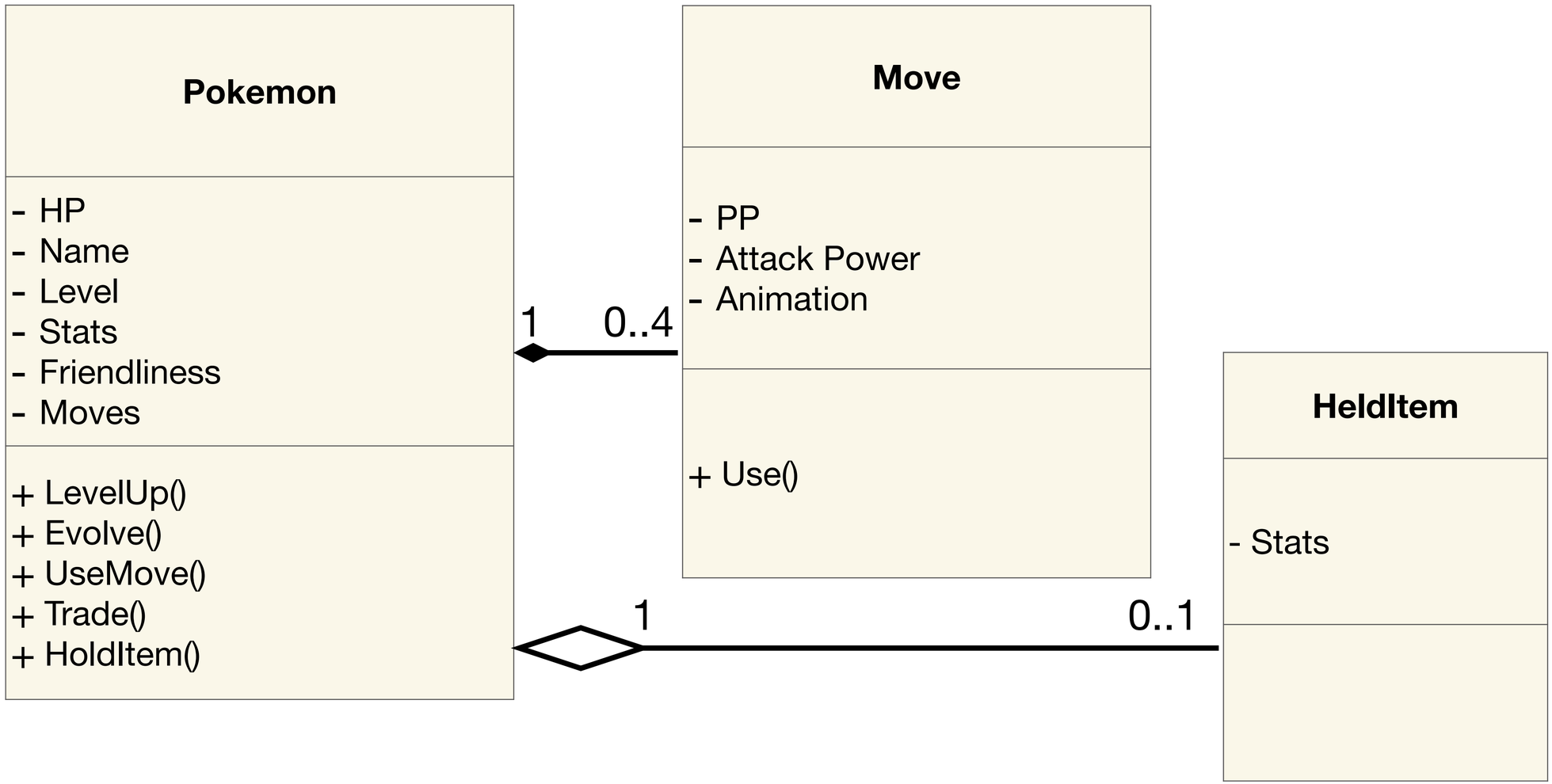
0	No instance (rare)
0..1	No instance or one instance
1	Exactly one instance
0..*	Zero or more instances
*	Zero or more instances
1..*	One or more instances



- A professor has 1 or more classes to teach.
- A class has 1 professor

Back-end Development

Class Diagram - Example



Back-end Development

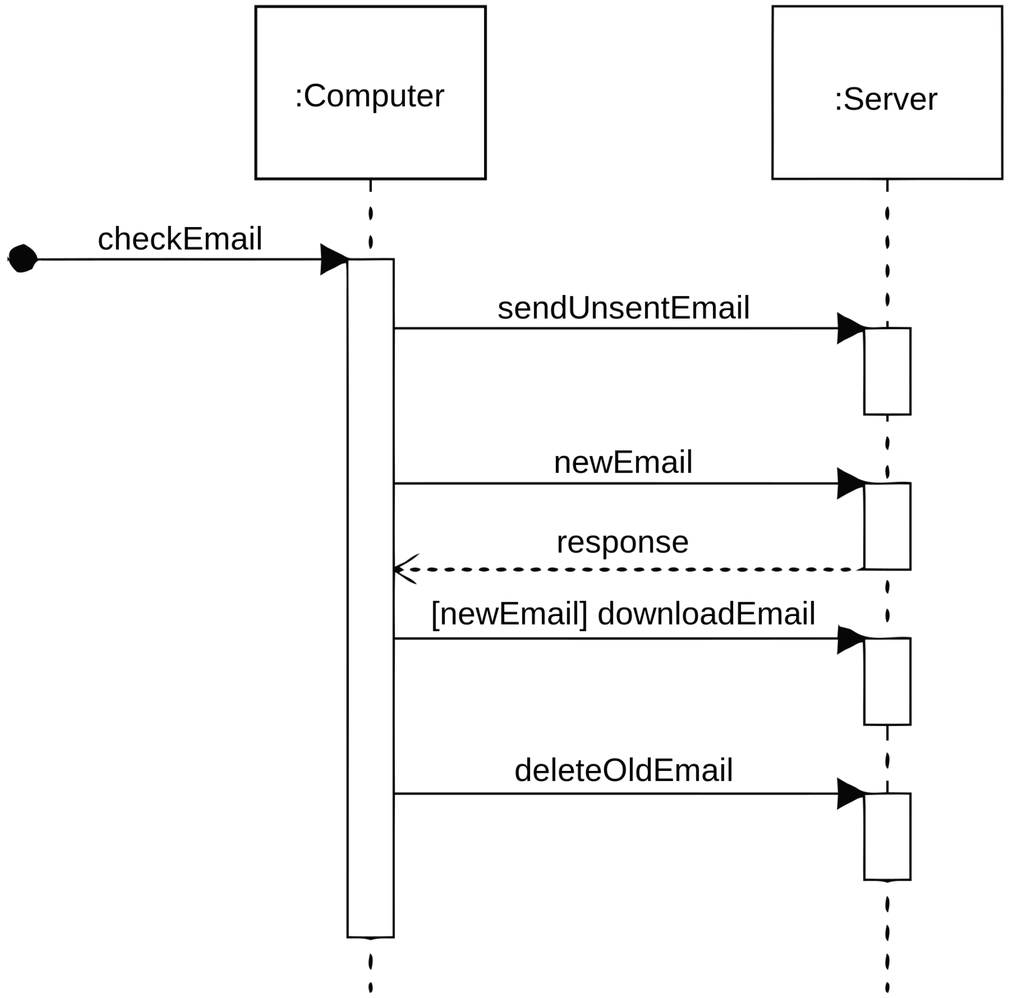
Other Design Patterns

- a framework/guideline to follow when encountering a specific problem.
  - In ODE, you have many techniques when you encounter different kinds of differential equations.
  - In software design, those techniques are design patterns.
  - There are a lot of design patterns.
  - See examples in https://sourcemaking.com/design_patterns

Back-end Development

Interaction Diagram

- describes interactions between two or more classes when an event occurs
- helps developers to set up proper functions to implement



Ref: By Coupling_loss_graph.svg - File:CheckEmail.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20544977

Worksheet V (งานคู่)

Part 2: UML-Class Diagram

- ให้เลือกเกมส์ที่สนใจมา 1 เกมส์เพื่อเขียน Class Diagram โดยต้องมีส่วนประกอบอย่างน้อย 4 classes ที่แตกต่างกัน และต้องลากเส้นแสดงความสัมพันธ์ระหว่างคลาสต่างๆ ให้เหมาะสมด้วย