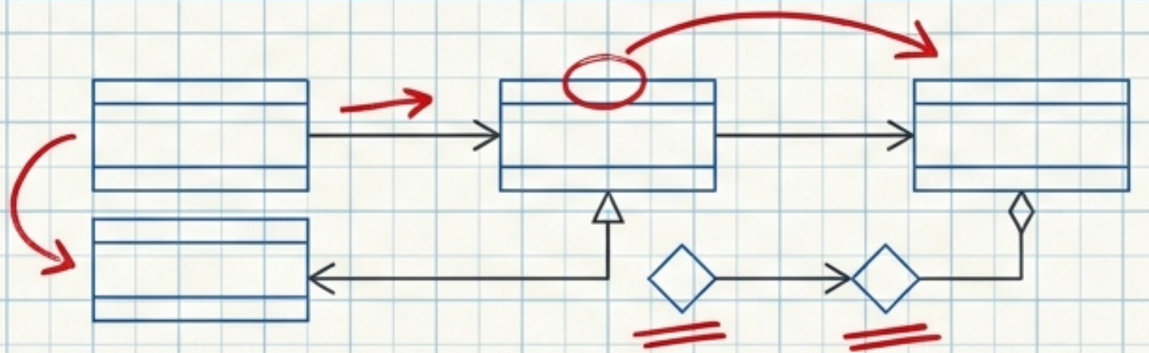
Object-Oriented Technology (ITP4909)

Professor's Edition:
Memorize the Red Ink!
[100 Marks Total]

The Ultimate Exam Masterclass｜從零到滿分：物件導向技術終極考試指南



Designed for self-study. Bilingual concepts. Exam rubrics embedded.

NotebookLM

The Exam Blueprint (考試解構)

Section A: 40 Marks

Structural & Implementation (結構與實作)

- Class Diagrams (12-13 marks)
- Java Implementation (12-15 marks)
- Tracing Code

Heavy logic! Master 1-to-many relationships in Java to secure these marks.

Section B: 60 Marks

Requirements & Dynamic Modeling (需求與動態建模)

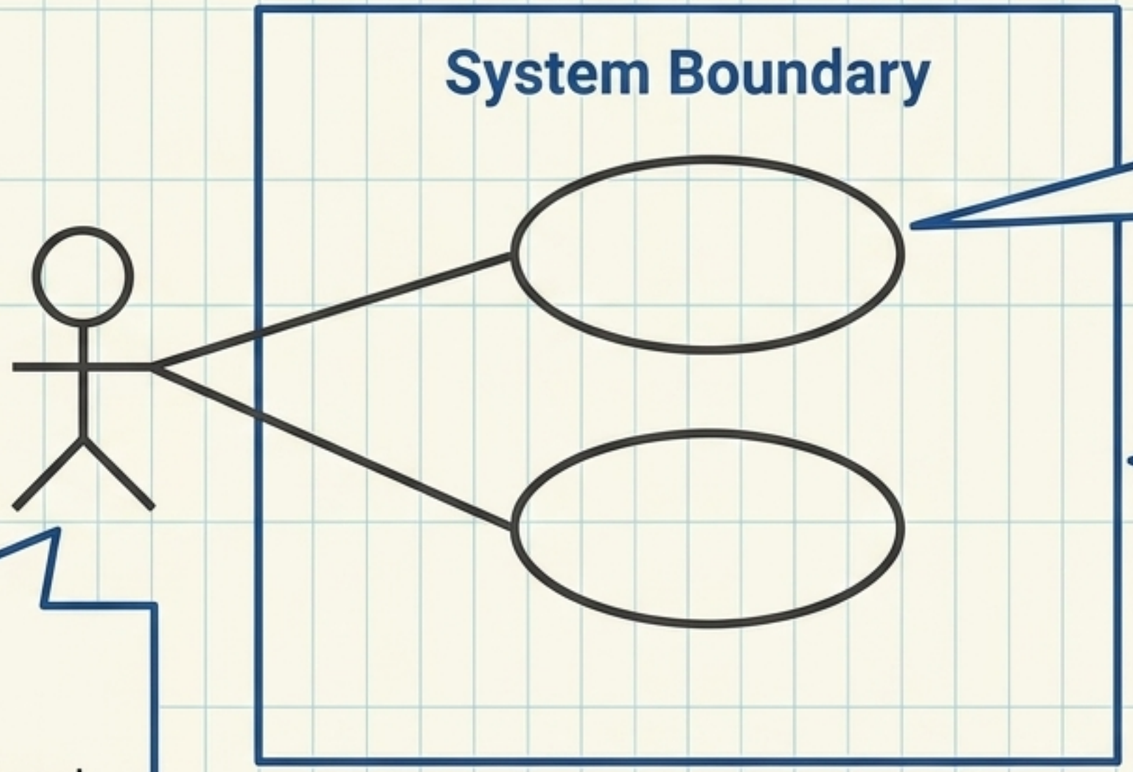
- Use Case Diagrams (13-20 marks)
- Textual Analysis (7-9 marks)
- Sequence/Activity/State Diagrams (15-28 marks)

The absolute core! The 3-Tier MVC Sequence Diagram is the final boss.

NotebookLM

Phase 1 - Use Case Modeling (用例建模)

用例建模是整個系統的起點，描述系統「能做什麼」而不是「怎麼做」。



“Use Case” (用例):
A sequence of actions yielding an observable result of value.

3. **“System Boundary”** (系統邊界):
Defines what is inside the system vs. outside.

1. **“Actor”** (參與者):
External entities (People, Hardware, External Systems) that interact with the system. They play a role.

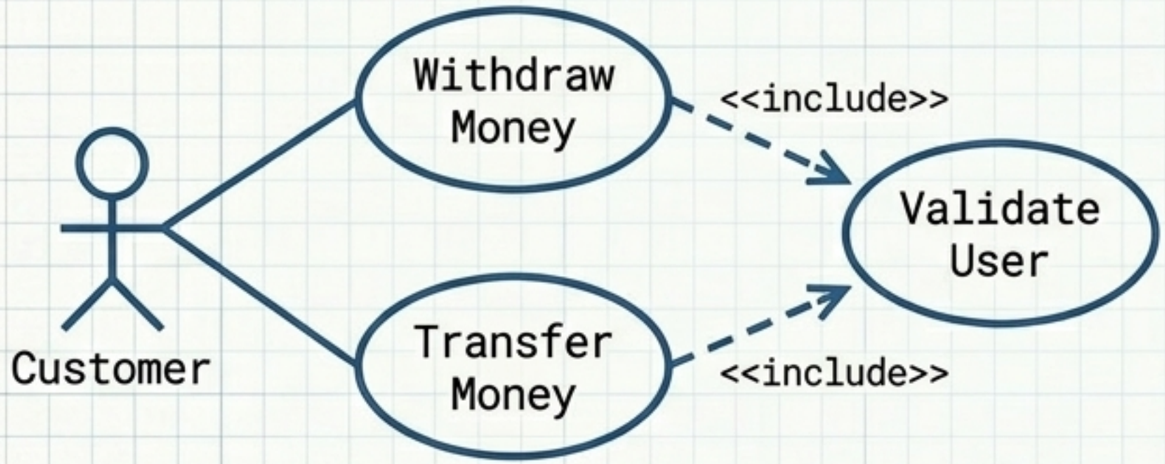
EXAM TIP: Actors are ALWAYS **outside the boundary box. Never draw them inside!**

NotebookLM

The Ultimate Trap: <<include>> vs <<extend>>

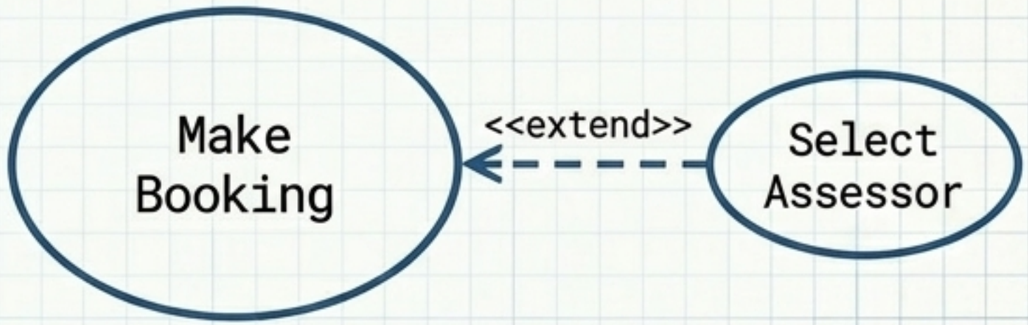
<<include>>

Mandatory behavior factored out.
(必須執行的共用功能).



<<extend>>

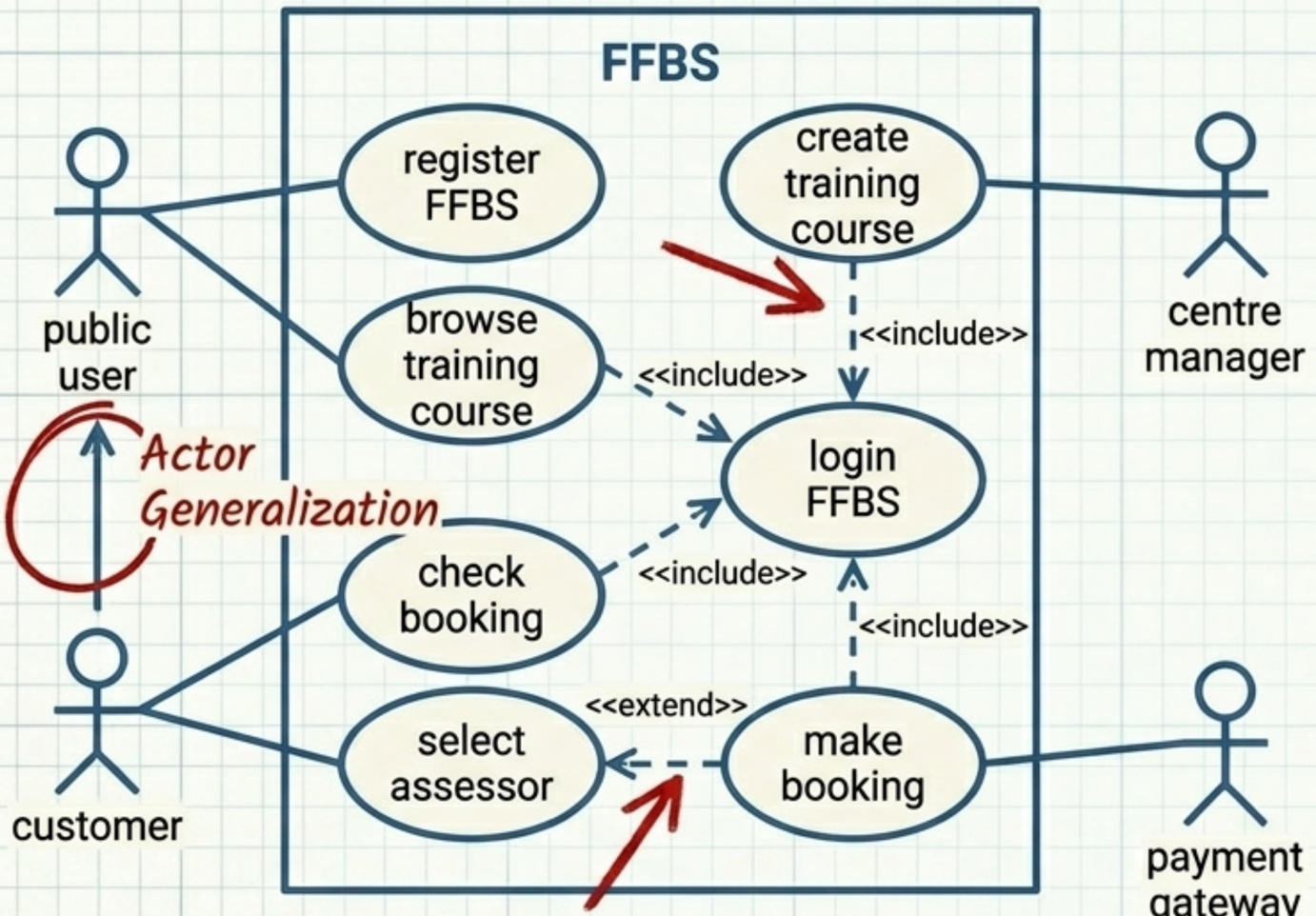
Optional or exceptional behavior.
(額外或例外擴展的功能).



WARNING: Arrow Direction! Base --> Included. Extending --> Base.
Get this wrong, lose the marks.

NotebookLM

Exam Focus: Drawing the Use Case Diagram



Marking Scheme Secrets

- [X] Usually worth 19-20 marks.
- [X] System Name at top
- [X] Correct Actors
- [X] Communication Links (solid lines, NO arrows!)
- [X] Includes/Extends

RED PEN ALERT: Directly quoting the rubric: **"Deduction for excess irrelevant use cases/communication links!"** Don't invent requirements.

NotebookLM

Phase 2 - Flow of Events (事件流)

Use case name	XXXXXXXXX
Use case ID	XXXXXXXXX
Actor(s)	XXXXXXXXX
Brief description	XXXXXXXXX
Flow of events	1. Xxxxxxxxxxx 2. Xxxxxxxxxxx 3. If XXXXXXXX, execute use case UC-KKKK 4. Xxxxxxxxxxx

Flow of events (事件流):
Describing the behavior step-by-step.

Base use case flow (基本事件流):
The "happy path" where everything goes right.

Alternative / Exception flow (替代/例外流):
What happens when conditions fail (e.g., incorrect PIN).

Exam Strategy Guide

Step 1: Start with user action (e.g., "The sequence starts with clicking Create My Case").	**Step 2:** Alternate between System Response and Actor Input.	**Step 3:** Keep it UI-independent (say "select an option" not "click the blue radio button").

NotebookLM

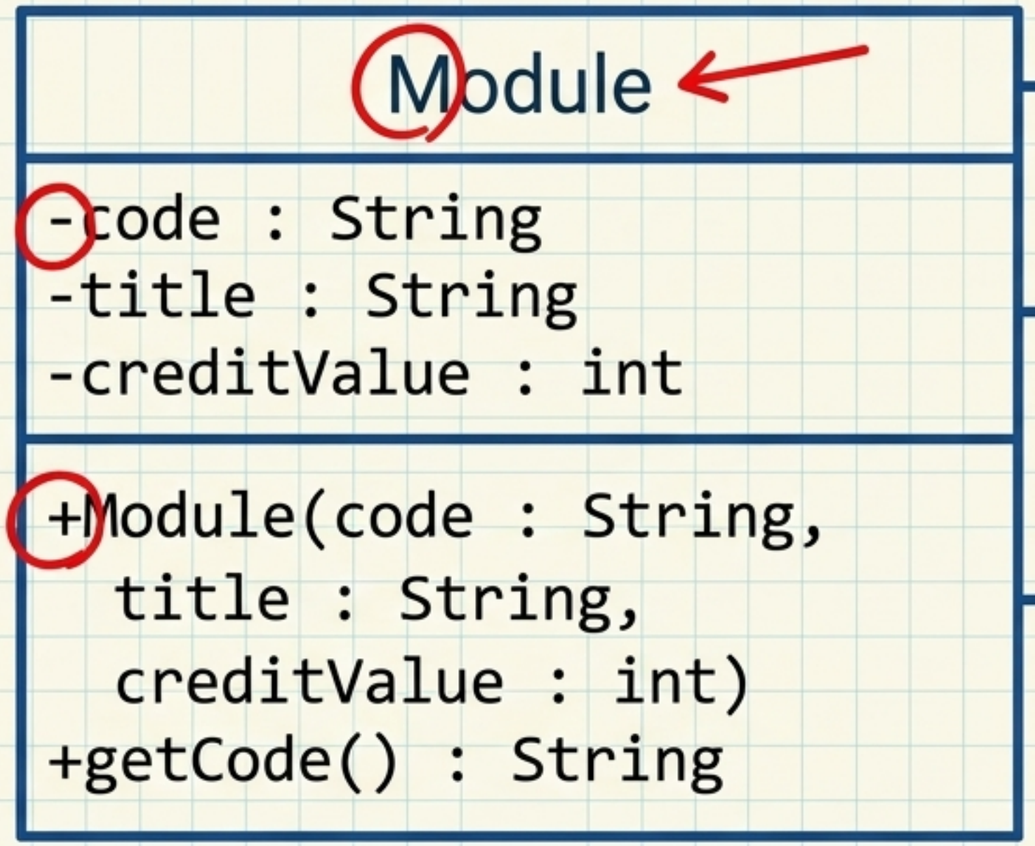
Textual Analysis (文本分析) -> Candidate Classes

Noun (from problem statement)	Candidate Class?	Reason / Category
Customer, Manager, Patient	YES	Role Play (角色扮演)
Subscription, Account	YES	Conceptual (概念)
Coffee, Vaccine, Sensor	YES	Tangible Thing (實體物件)
Booking, Delivery, Payment	YES	Event (事件)
Payment Gateway, SMS Gateway	YES	External System (外部系統)

PROFESSOR's TIP: *List the candidate class, state "**YES**", and name exactly **ONE** of these 5 English reasons to secure your 7-9 marks.*

NotebookLM

Phase 3 - Structural Modeling: Class Diagrams (類別圖)



Class Name (Capitalized)

Attributes (屬性).
Format: visibility name : type

Methods / Operations (方法).
Format: visibility name() : return_type

Visibility Cheat Sheet

+ = Public
- = Private
= Protected

Read the prompt! If it says 'based on given information', don't invent extra attributes!

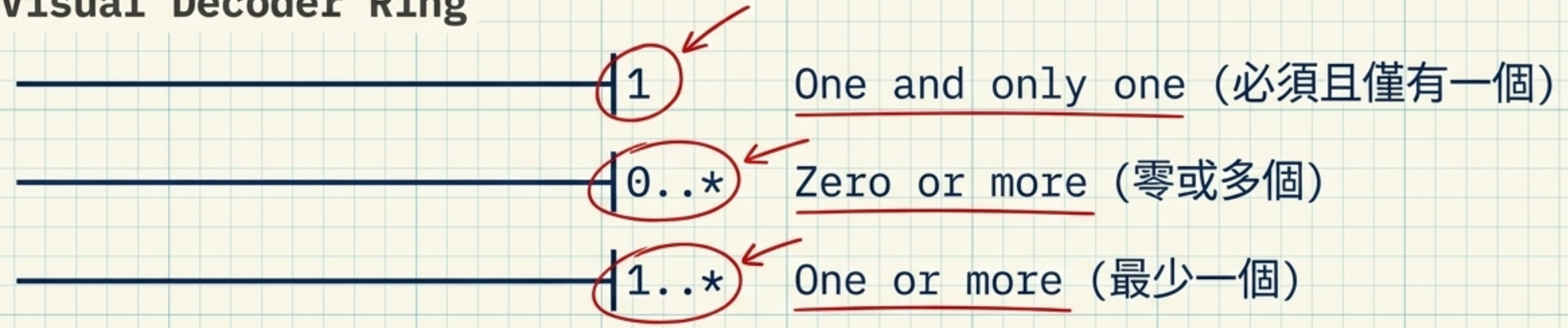
NotebookLM

Relationships & Multiplicity (關聯與多重性)

Association (關聯): A structural relationship connecting classes. Show the name and multiplicity.

Multiplicity (多重性): How many instances are linked?

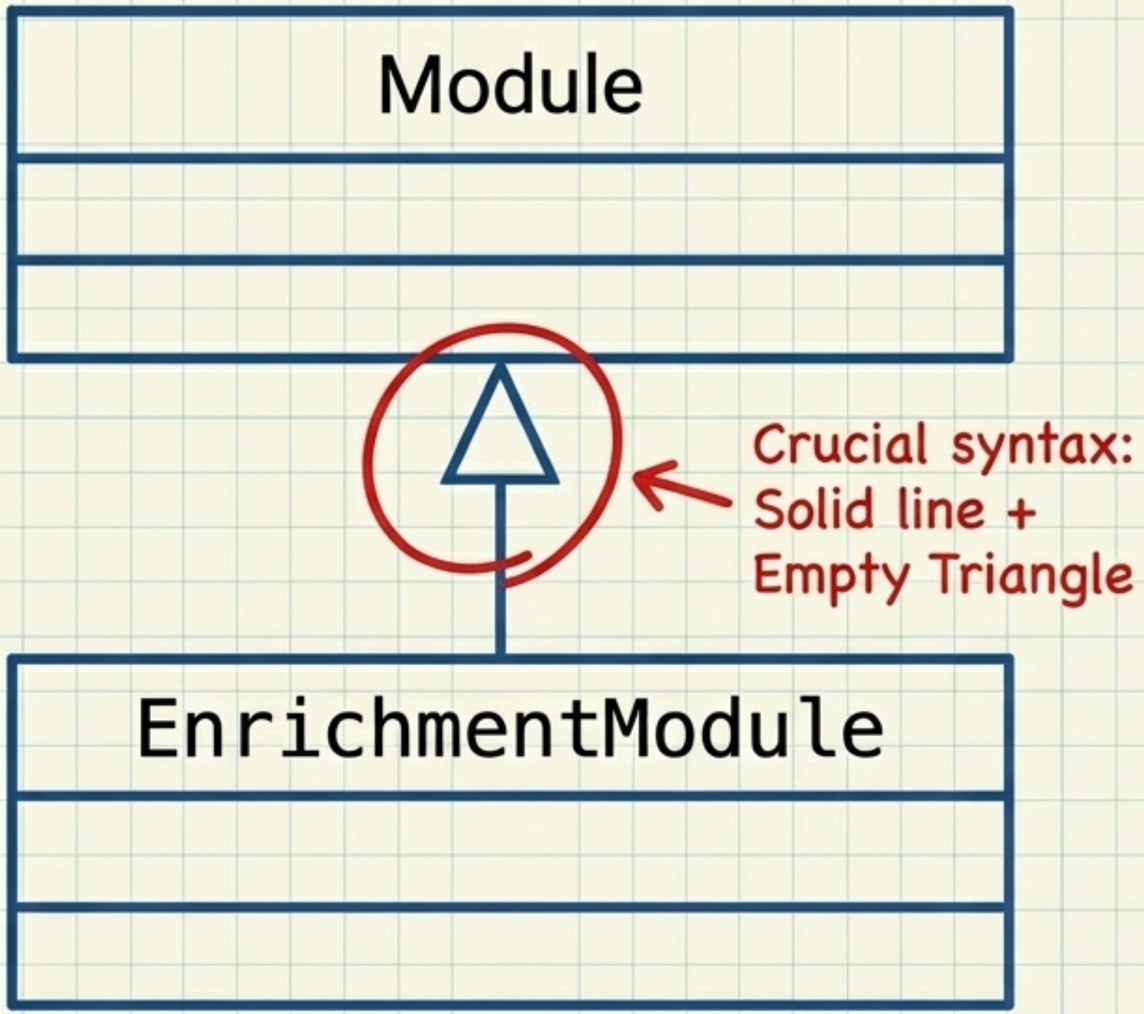
Visual Decoder Ring



RED PEN TRAP: The rubric explicitly states: "You do not need to add association classes for this question." Don't waste time drawing dashed-line association classes in Section A!

NotebookLM

Generalization (繼承/泛化)



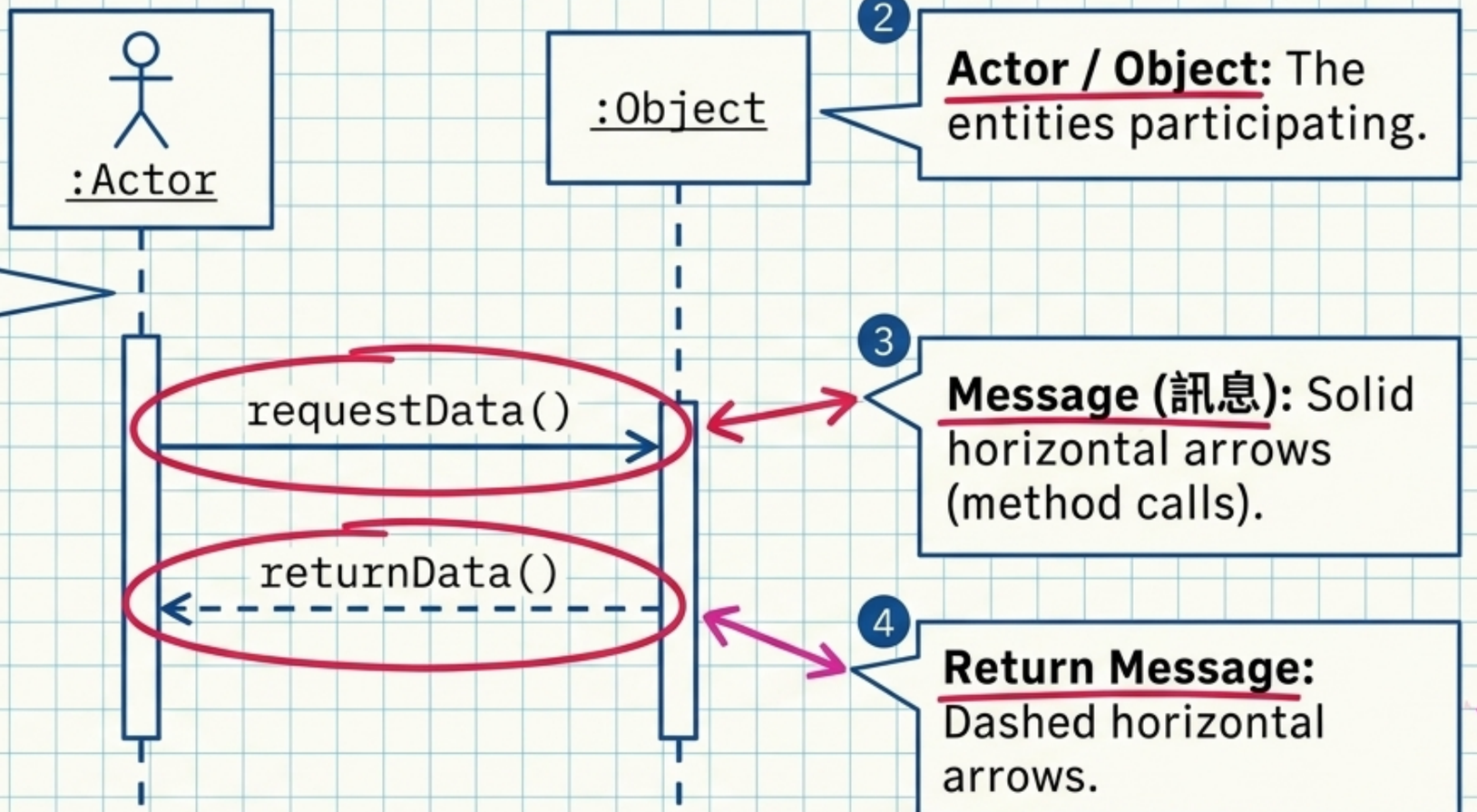
Concept: Generalization.
Extracting common attributes and methods into a parent class.

Key Rule:
Subclasses **inherit** all attributes and operations from the **Superclass** but can have their own specific ones.

Always structure with inheritance if possible to get full marks (e.g., NormalCourse vs SignatureCourse).

NotebookLM

Phase 4 - Dynamic Modeling: Sequence Diagrams (循序圖)



1 **Lifeline (生命線):** The vertical dashed line representing the object's life during interaction.

2 **Actor / Object:** The entities participating.

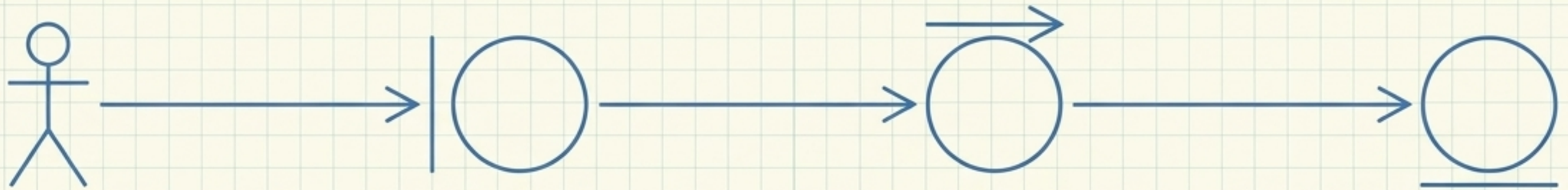
3 **Message (訊息):** Solid horizontal arrows (method calls).

4 **Return Message:** Dashed horizontal arrows.

Context: Section B asks for either a 'System-level' sequence diagram (Actor to System) (Actor to System) OR a '3-Tier' sequence diagram.

NotebookLM

The 3-Tier MVC Sequence Diagram Master Pattern



Actor
(e.g., Customer)

Boundary Object
(e.g., :SystemUI)
- Handles user input/display.

Control Object
(e.g., :SystemController)
– The brain, handles logic and delegates.

Entity Objects
(e.g., :Coffee, :Packing)
– The data objects.

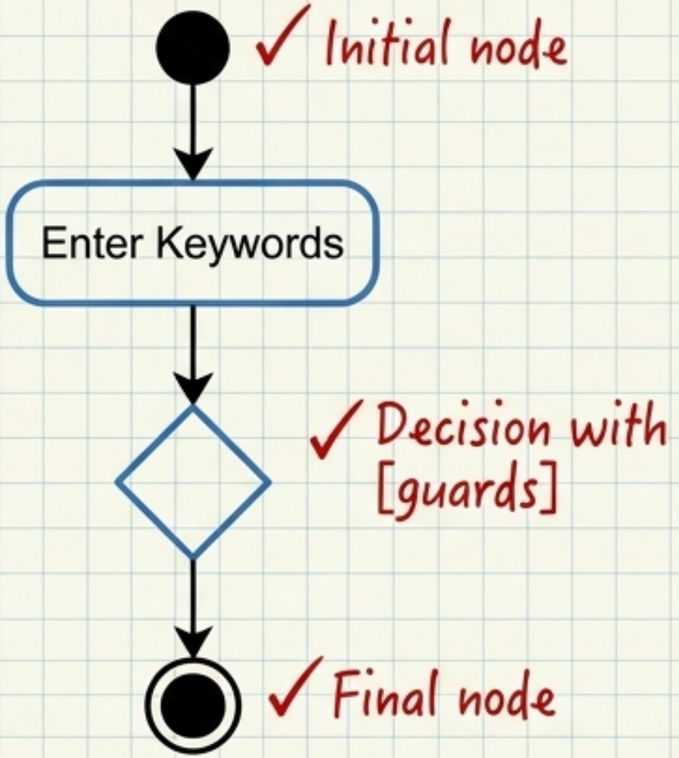
ABSOLUTE RULE: For simplicity, use ONLY ONE user interface object and ONLY ONE controller object. Marks will be deducted if you draw multiple UI screens as separate lifelines!

NotebookLM

Activity & State Machine Diagrams (活動圖與狀態圖)

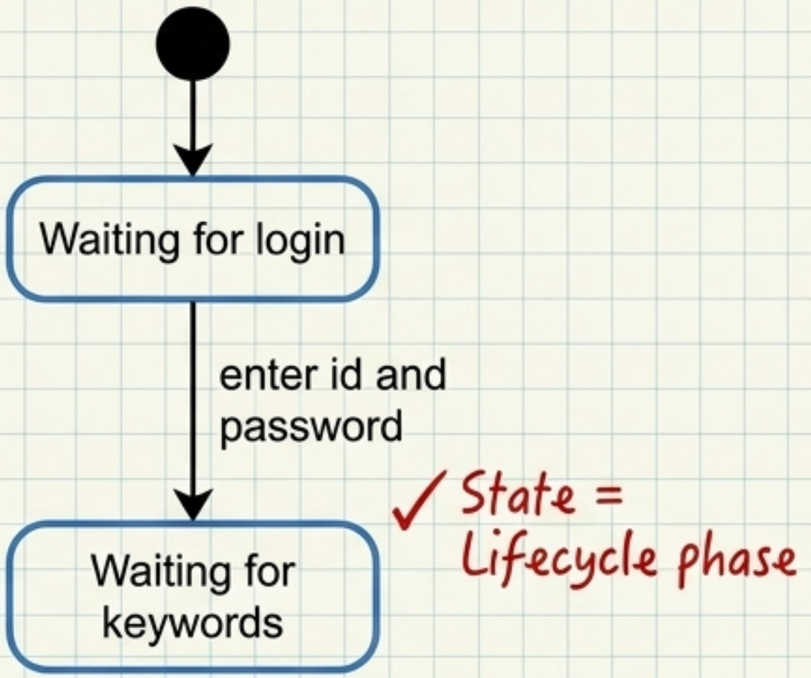
Activity Diagram (活動圖)

Models the flow of actions (like a flowchart).



State Machine Diagram (狀態圖)

Models the states of a single object.

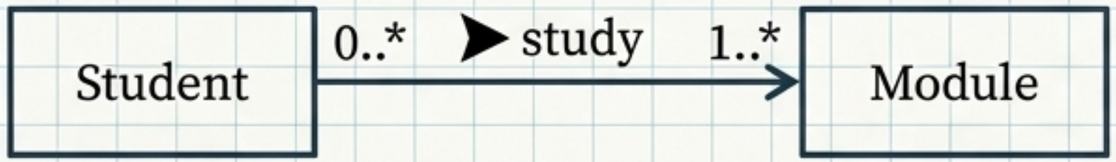


Activity = Flow of the whole process. State = Lifecycle of ONE thing.

NotebookLM

Phase 5 – Java Implementation (Java 代碼實現)

UML Snippet



Java Code Translation

```
// 1-to-Many Relationship
private Vector _modules;
// or private ArrayList<Module> _modules;

public void addModule(Module m) {
    _modules.add(m);
}

// Inheritance
public class EnrichmentModule extends Module {
    public EnrichmentModule(...) {
        super(code, title, creditValue);
        // super() MUST be the first line
    }
}
```

Don’t forget to initialize the Vector/ArrayList in the constructor!

NotebookLM

The Professor’s Final Checklist

- [ ] Know the 5 Textual Analysis categories in English.
- [ ] Verify Use Case arrows: `Base --> Include`, `Extend --> Base`.
- [ ] Use only ONE Boundary and ONE Controller in 3-Tier Sequence Diagrams.
- [ ] Memorize the exact syntax for `Vector/ArrayList` and `super()` in Java.
- [ ] Do NOT draw Association Classes in Section A.

掌握這些核心規律，不要在無關的細節上失分。祝考試順利！

NotebookLM