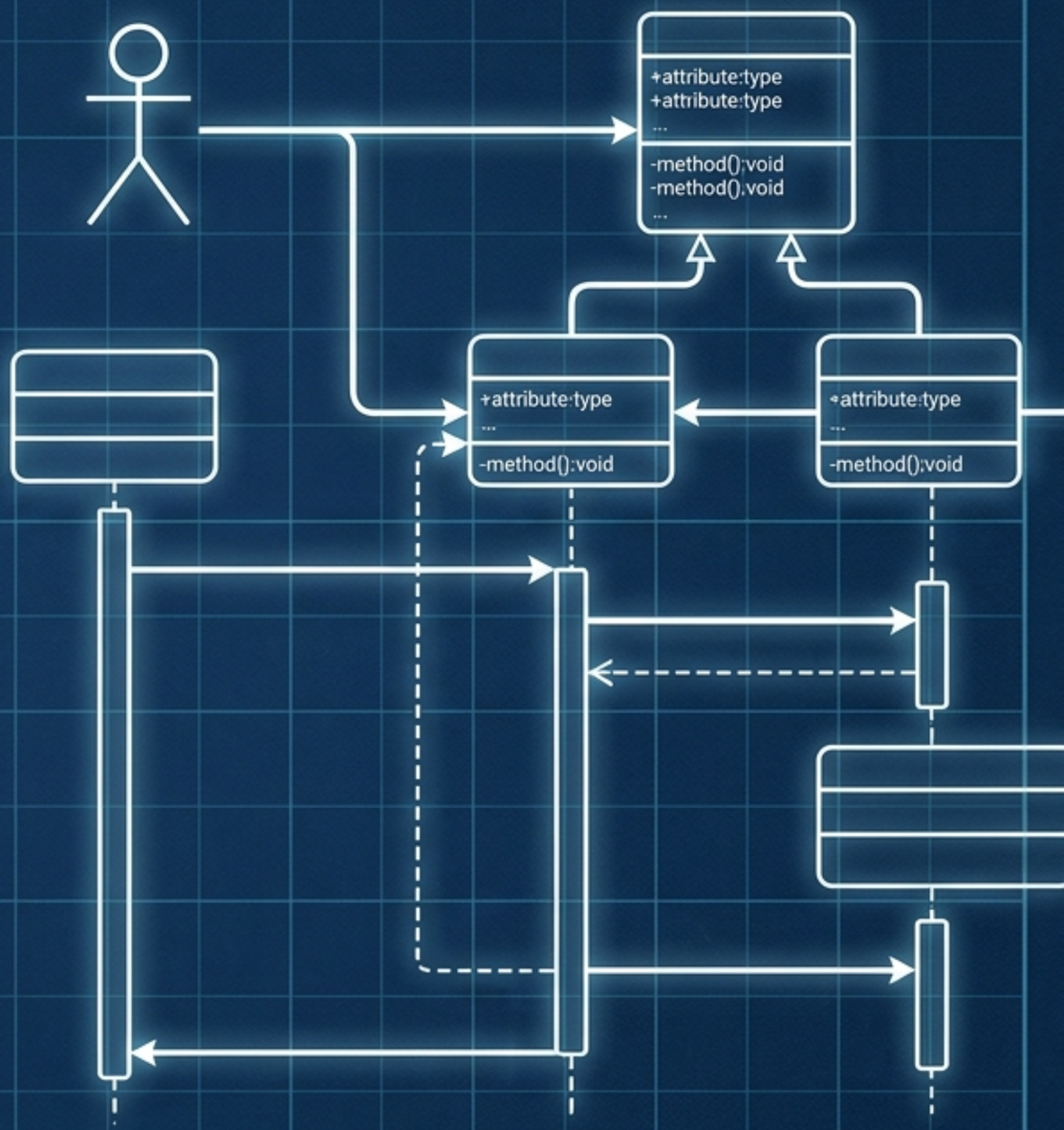
Object-Oriented Technology: The Masterclass

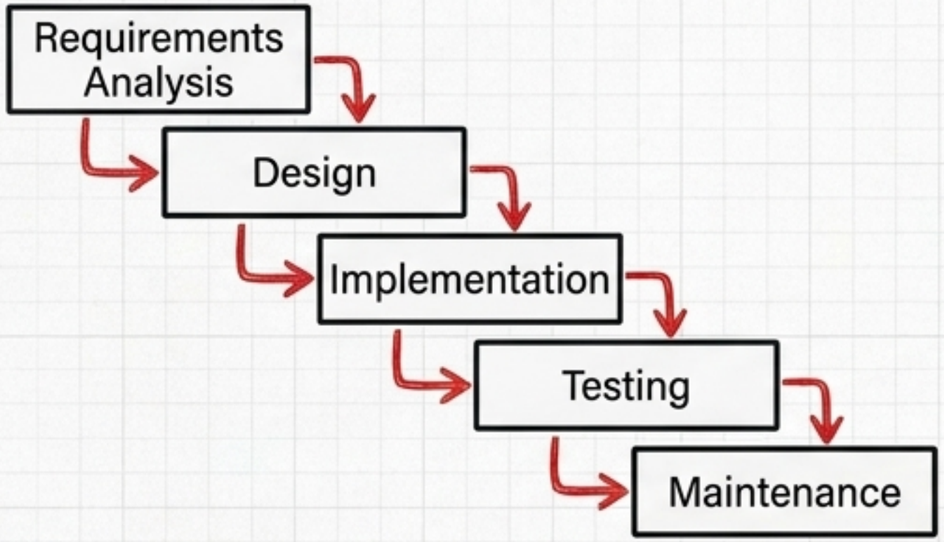
從零開始的 UML 與 OOAD 終極考試指南 (From Zero to Exam Mastery)

Professor's Guide to Conquering ITP4909



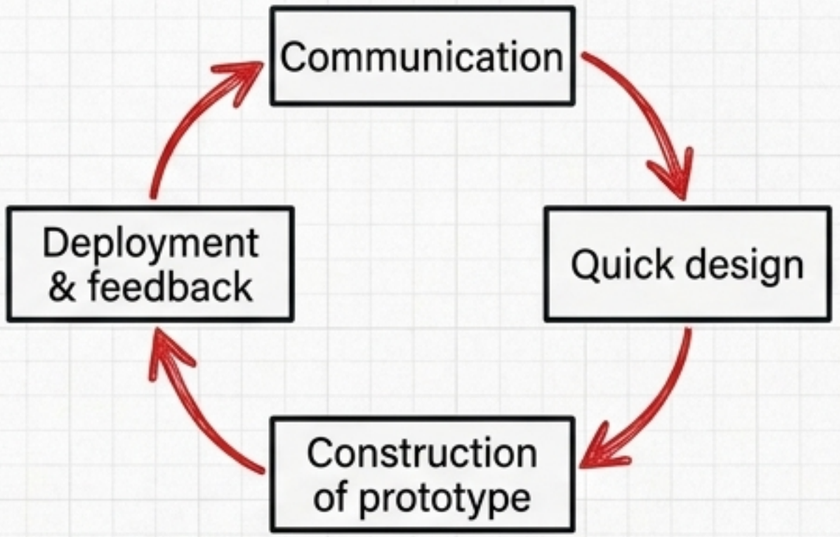
The Foundation: 軟體開發不僅僅是寫程式
(Software Development Requires Disciplined Processes)

Waterfall Model (瀑布模型)



- Sequential, rigid. Progresses only when verified.
- Pros: Accountable.
- Cons: Inflexible.

Iterative Models (迭代模型)



- Prototyping & Spiral.
- Feedback-driven, loop-based.
- Partial systems built quickly for user evaluation.

Professor's Callout
軟體開發流程 (Software Process) 包含 Requirements Specification, Design & Implementation, 及 Verification & Validation. 這是所有開發的基礎。

Unified Process (UP): 物件導向的核心開發框架

OOAD is modeling a system as a collection of collaborating objects. (將系統視為一組互相協作的物件)

Sticky Note 1
1. Inception (初始階段): Define scope & goals.

Sticky Note 2
2. Elaboration (細化階段): Analyze requirements & architecture.

Sticky Note 3
3. Construction (建構階段): Build the product iteratively.

Sticky Note 4
4. Transition (移交階段): Deploy final software to customers.

🏆 Exam Radar (考試重點)

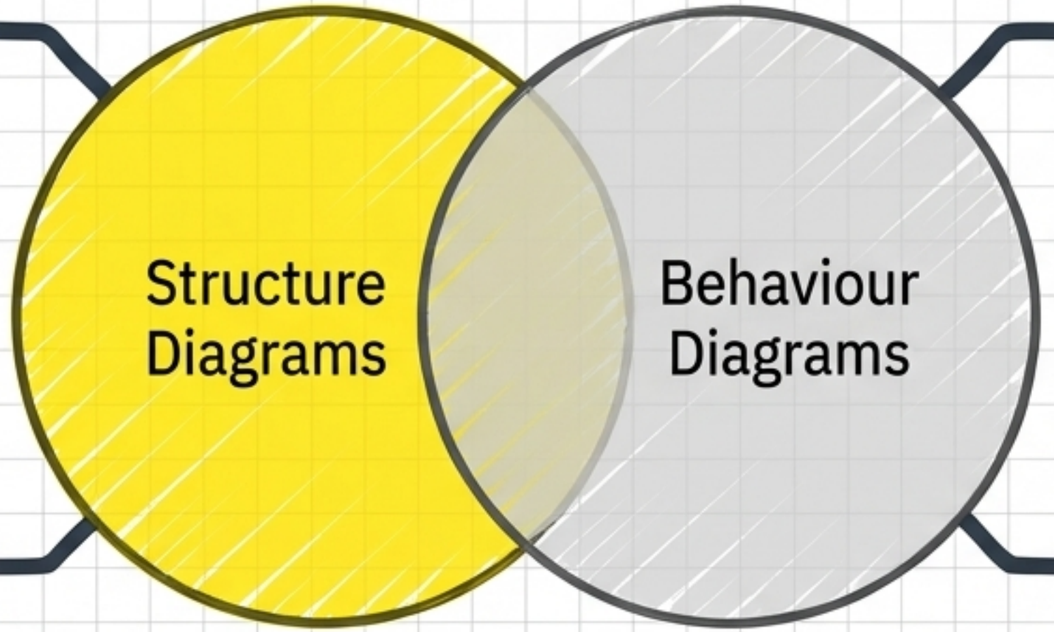
選擇題常考題型！You may be asked to identify which UP phase a specific task (like deploying software or defining initial scope) belongs to. Memorize these four phases and their exact English names!

The UML Universe: 軟體設計的「世界語」

UML is a modeling language, NOT a programming language. (UML 不是一種程式語言)

Structure (Static / 靜態結構)
Shows what the system IS.

- Class Diagram
- Package Diagram
- Deployment Diagram



Behaviour (Dynamic / 動態行為)
Shows what the system DOES.

- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- State Machine Diagram

Professor's Callout : 這是整份考卷的藍圖！Section A 專攻 Structure (Class Diagram 類別圖) , 而 Section B 專攻 Behavisour (各種動態流程圖)。

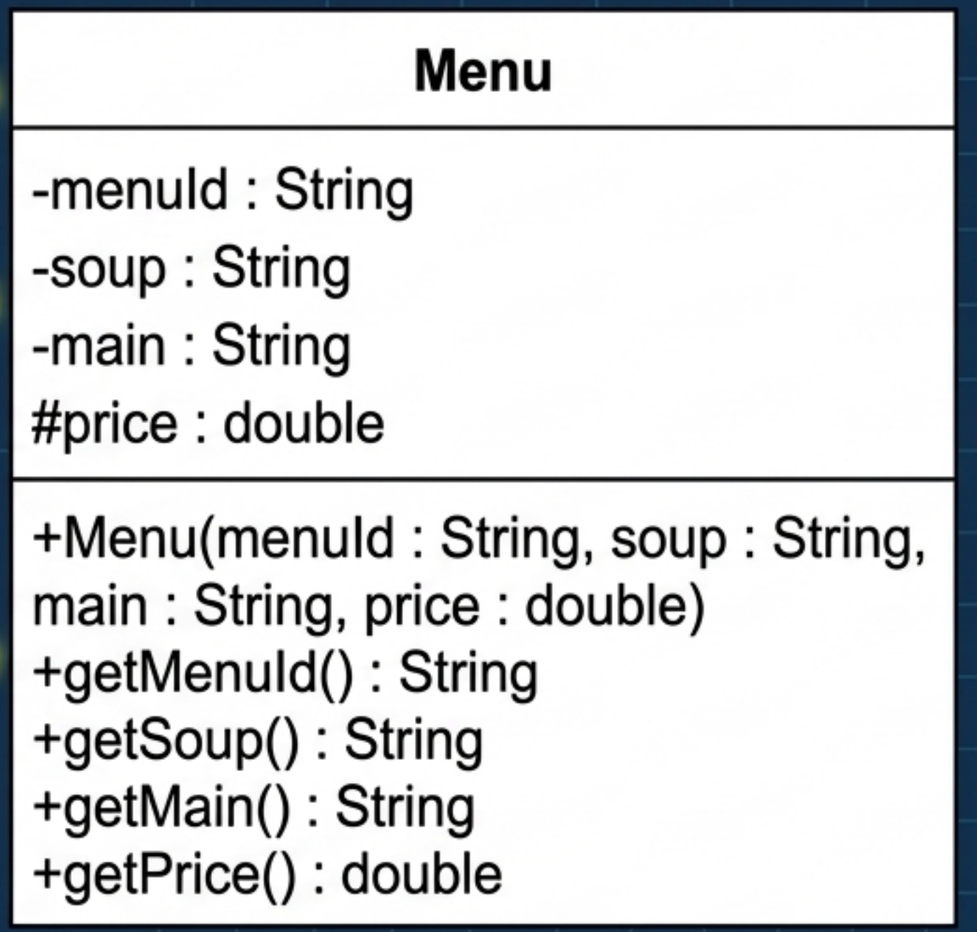
Exam Section A: 剖析 Class Diagram (類別圖)

類別圖展示系統的靜態藍圖，即「誰參與了系統，他們有什麼屬性和動作」。

Top Tier: Class Name (類別名稱)

Middle Tier: Attributes (屬性 - Data/Variables)

Bottom Tier: Methods/Operations (方法 - Actions/Functions)



Visibility Indicators

- + (Public 公開)
- - (Private 私有)
- # (Protected 保護)

🏆 **Exam Radar (13 Marks)**

這是 Section A 的必考繪圖題！仔細閱讀題目提供的資訊畫出完整的類別圖。
請注意：**不需要加入關聯類別** (Do not add association classes) 除非題目特別要求！

Class Relationships: 物件之間如何互動是高分關鍵

Association (關聯)

Basic relationship.

Driver [drives] Car

Inheritance / Generalization (繼承)

Is-a relationship.

Medical Robot is a sub-class of Robot

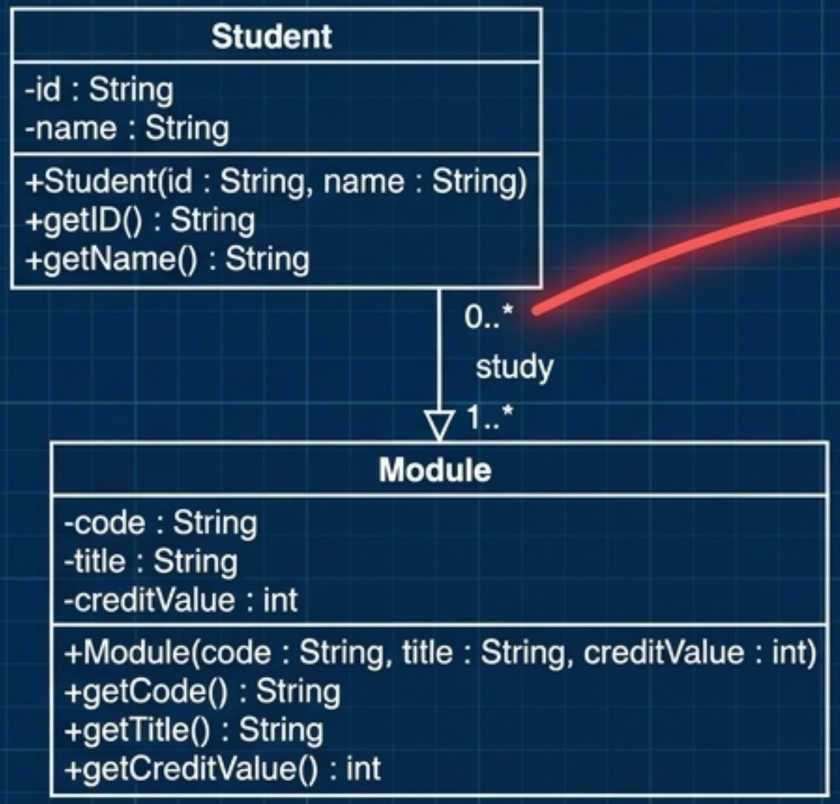
Multiplicity (多重性) Notation

1 (Exactly one / 剛好一個)
0..* (Zero or more / 零或多個)
1..* (One or more / 至少一個)

Professor's Callout: 忘記標示 Multiplicity 或關聯名稱 (Association name) 會被扣大量分數！永遠問自己：「這是一對一，還是一對多？」

Translating UML to Java (類別圖轉程式碼)

考試會要求你將畫出的 UML 轉換為真正的 Java 程式碼.




```java
import java.util.*;

public class Student {
    private String id, name;
    private Vector _modules;

    public Student(String id, String name) {
        this.id = id;
        this.name = name;
        _modules = new Vector();
    }

    public String getID() { return id; }
    public String getName() { return name; }

    public void addModule(Module module) {
        _modules.add(module);
    }
    public Enumeration getModules() {
        return _modules.elements();
    }
}
```

Implementation Secrets:

- **One-to-Many**: 必須使用 **Vector** 或 **ArrayList** 實作.
- **Inheritance**: 使用 **extends** 關鍵字.
- **Abstract Methods**: Must be implemented in child classes.

🏆 **Exam Radar (12-15 Marks)**

注意屬性的封裝 (**private**) 以及建構子 (**Constructor**) 的正確寫法。這是 Section A 的壓軸題！

Exam Section B: Textual Analysis (文本分析)

從一段冗長的需求描述中找出系統的候選類別 (Extracting candidate classes from problem statements using Noun Extraction).

The Customer orders a Subscription for weekly Coffee deliveries. The payment is processed via an external Payment gateway which must be integrated. The system tracks the status of each Delivery and ensures the Packing is completed by the staff. The Customer can view their Account details online.

Candidate Class / Reason	
Customer	Role Play
Subscription	Conceptual
Coffee	Tangible Thing
Payment gateway	External System

🏆 **Exam Radar (8-9 Marks)**

必須以表格格式作答！明確列出候選類別 (Candidate Class) 及其對應的分類原因 (Reason/Category)。

Categorizing Classes (分類法則):

- Role Play (角色): Customer, Manager, Nurse
- Tangible Thing (實體): Coffee, Vaccine, Hardware
- Event (事件): Booking, Payment, Delivery
- Conceptual (概念): Subscription, Account

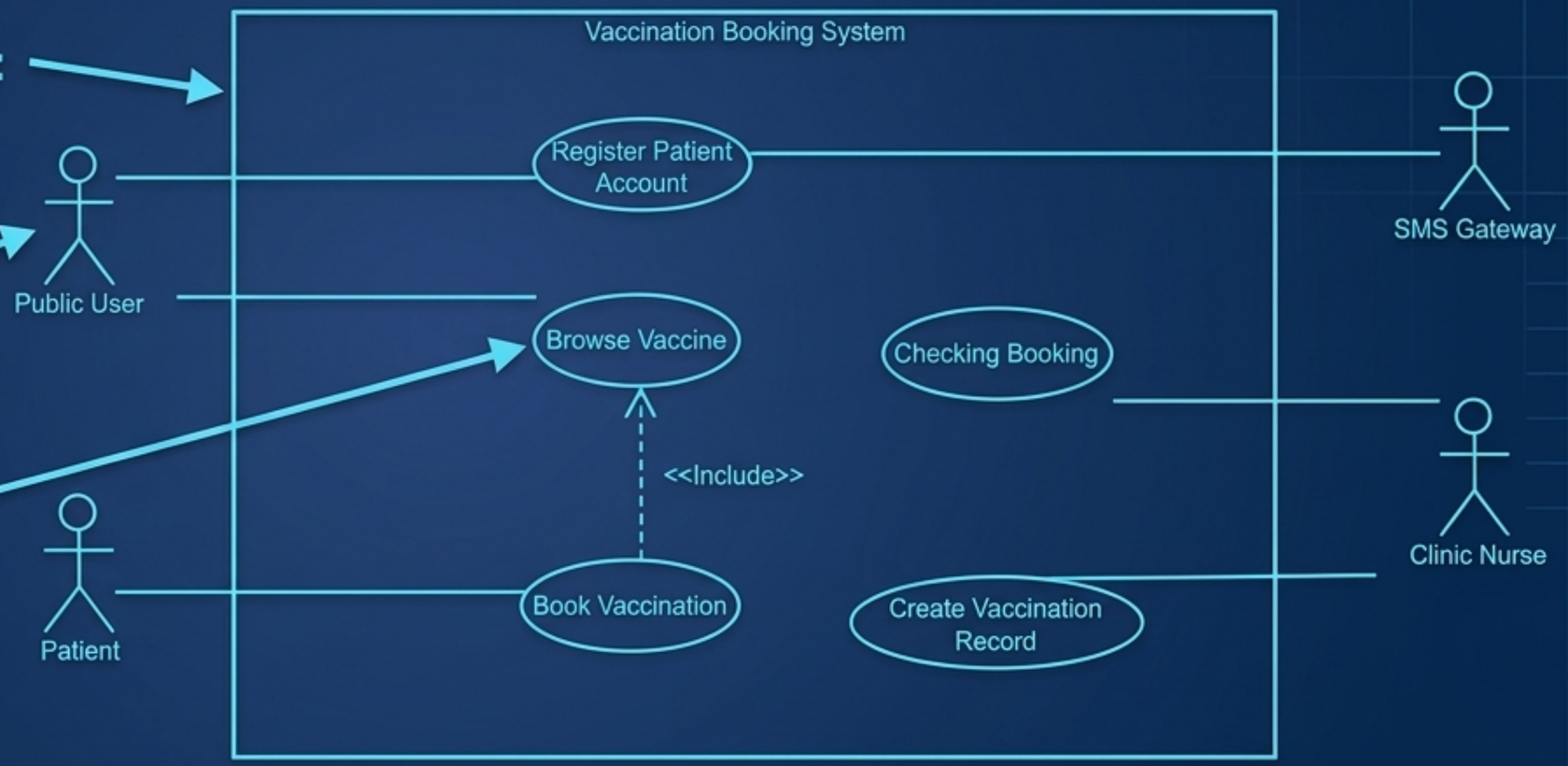
Use Case Diagrams: 定義系統邊界與功能

展示「誰」在使用系統，以及系統提供了「什麼」對外功能
(Shows WHO interacts with the system and WHAT it achieves).

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

Actors (參與者):
External users or other systems.

Use Cases (使用案例):
Action-oriented goals.

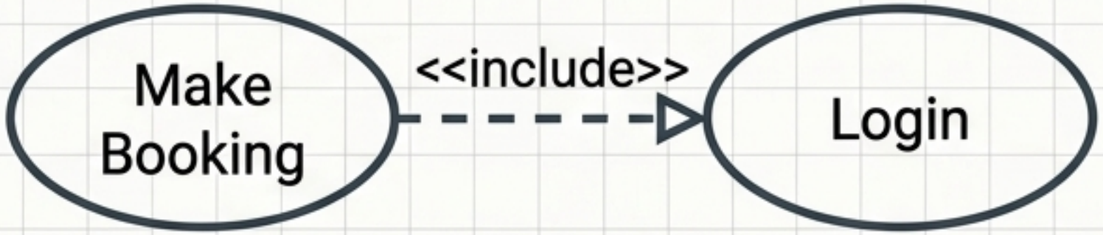


Professor's Callout

最大的錯誤是把內部系統邏輯畫成 Use Case！ Use Case 必須是對 Actor 有價值的完整目標
(e.g., "Process Payment", NOT "Save to database").

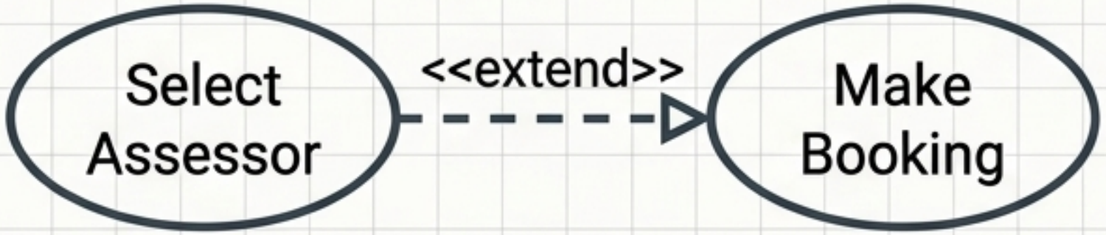
Use Case Traps: <<include>> vs <<extend>>

<<include>> (包含關係)



- **Logic:** Mandatory (必定發生). The base use case needs this to finish.
- **Example:** Make Booking <<include>> Login.
- **Rule:** Arrow points **TO** the included case.

<<extend>> (擴充關係)



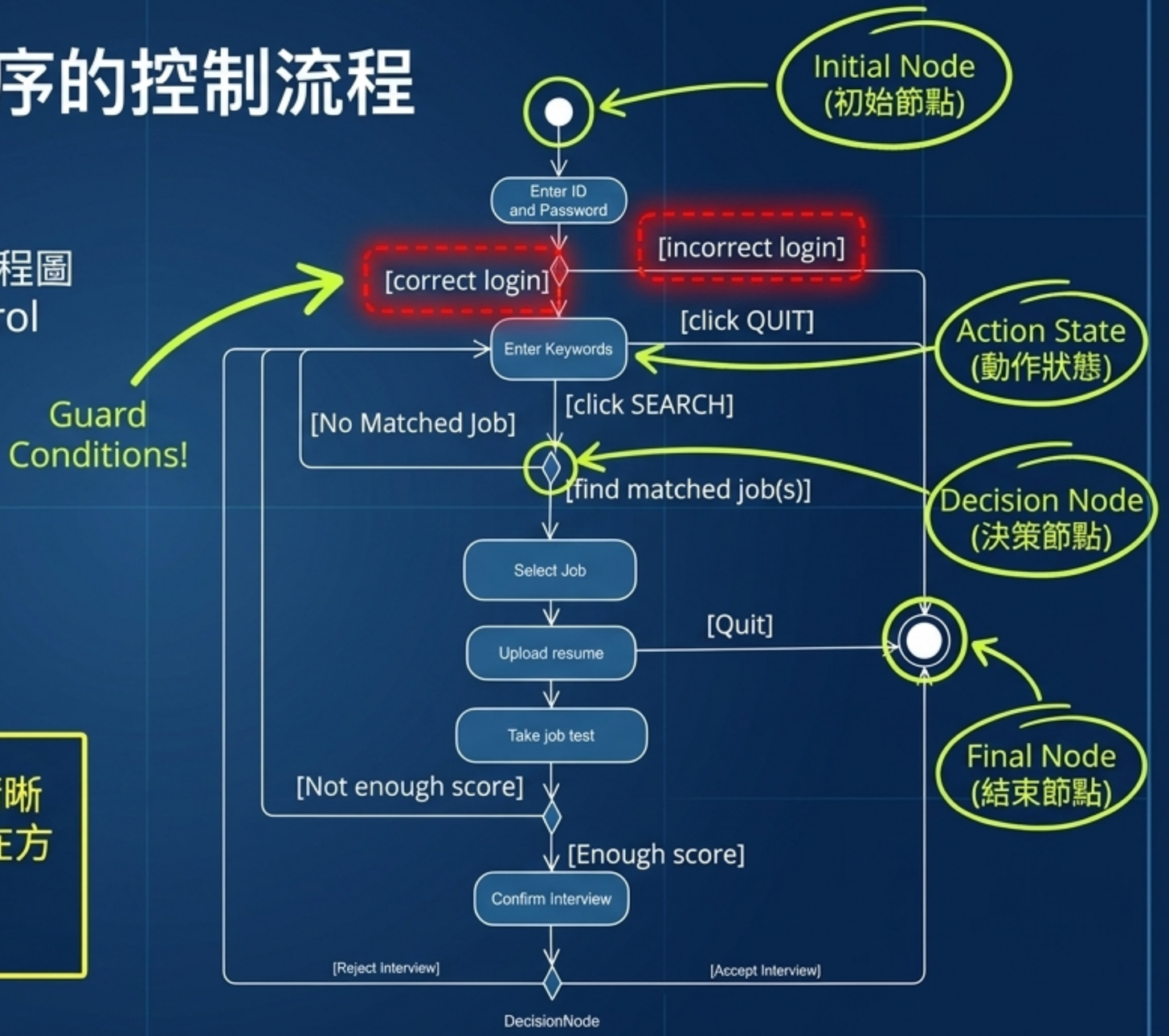
- **Logic:** Optional/Conditional (有條件發生). Exceptional flow.
- **Example:** Make Booking <<extend>> Select Assessor.
- **Rule:** Arrow points **TO** the base case.

🏆 Exam Radar

箭頭方向畫反直接扣分！ (Arrow directions matter!) Memorize the dependencies.

Activity Diagrams: 程序的控制流程 (Procedural Flow)

專注於程序的控制與決策流程，類似高階流程圖 (Focuses on the procedural flow of control between actions).



🏆 Exam Radar (6-8 Marks)

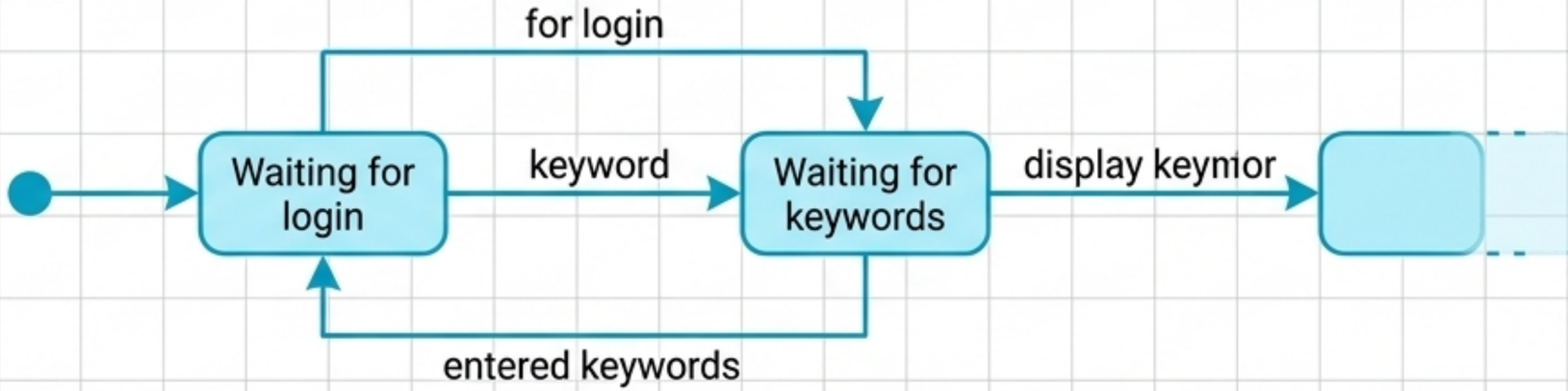
每個 Decision node 射出的箭頭都必須有清晰的 Guard Conditions (守衛條件)！必須寫在方括號內，例如 [correct login] 或 [incorrect login]。漏寫不予計分！

State Machine Diagrams: 物件的生命週期 (Object Lifecycle)

描述單一物件在其生命週期中的狀態變化，而非系統流程 (Models the states an object can be in and its transitions).

Key Notations

- State (狀態)
- Transition (轉換)
- Event (觸發事件)
- Guard Condition (條件)



The Golden Syntax

Transitions must be labeled as: event [guard] / action
Example: entered keywords [matched jobs] / display joblist

Professor's Callout

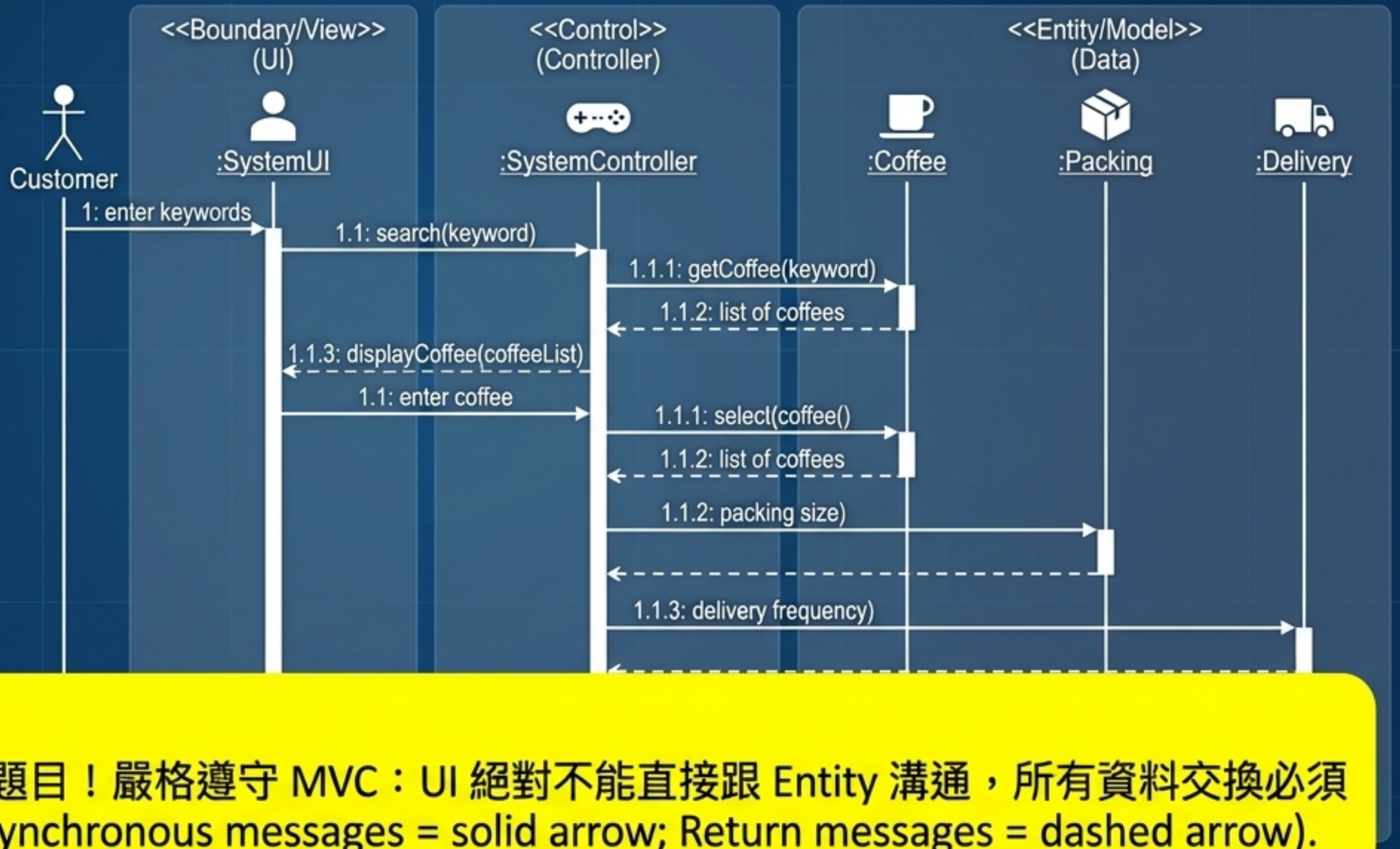
Activity 圖看的是「動作怎麼走」，State Machine 圖看的是「系統正處於什麼等待狀態」 (e.g., 'Waiting for resume').

MVC Sequence Diagrams: The Final Boss (15-28 Marks)

展示物件如何隨著時間依序互動以完成特定任務 (Shows message flow between objects over time).

The 3-Tier Rule (MVC 原則):

1. Boundary: SystemUI (使用者介面).
2. Control: SystemController (系統邏輯控制).
3. Entity: Objects like Customer, Booking (資料實體).



Exam Radar

這是整份試卷佔分最重的題目！嚴格遵守 MVC：UI 絕對不能直接跟 Entity 溝通，所有資料交換必須透過 **Controller** 傳遞！ (Synchronous messages = solid arrow; Return messages = dashed arrow).

The Professor’s Exam Cheat Sheet (題型判斷指南)

Need to see overall system functions & users?	⇨	**Use Case Diagram (使用案例圖)**
Need system data structure & object links?	⇨	**Class Diagram (類別圖)**
Need exact chronological steps of a specific scenario?	⇨	**Sequence Diagram (循序圖)**
Need workflow, procedural steps & logic loops?	⇨	**Activity Diagram (活動圖)**
Need to track a specific object’s status changes?	⇨	**State Machine Diagram (狀態機圖)**

Memorize this mapping. Understanding intent prevents drawing the wrong diagram.

Final Blueprint: Strategy for Success

"The only way to learn the UML is by writing models in it." – Booch, Rumbaugh, Jacobson.

Final Exam Tips (最後衝刺策略):

1. **仔細閱讀題目**: Nouns and verbs in the prompt are your literal cheat codes for classes and operations.
2. **不要留白**: Partial marks are generously given for correct shapes and notations. Draw something!
3. **檢查箭頭方向**: Double-check arrow directions in Use Case (include/extend) and Class diagrams (Inheritance).

Good luck on your ITP4909 Examination!
(祝考試順利，高分通過！)