CLASS DIAGRAM & USE CASE DIAGRAM

Danang Wahyu Utomo
danang.wu@dsn.dinus.ac.id
+6285 740 955 623

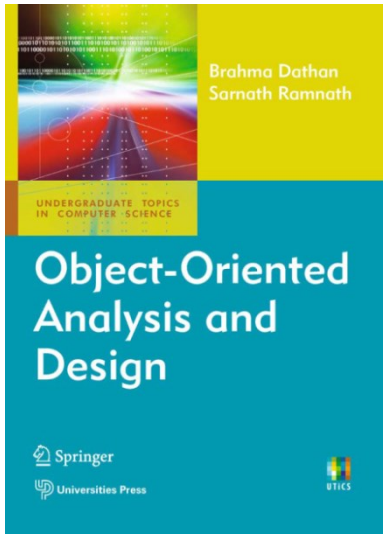
RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	Introduction Object Oriented Analysis & Design
2	Introduction UML
3	**Class Diagram and Use Case**
4	Analysis
5	Software Architecture
6	Software Architecture Modeling
7	
8	**Ujian Tengah Semester**

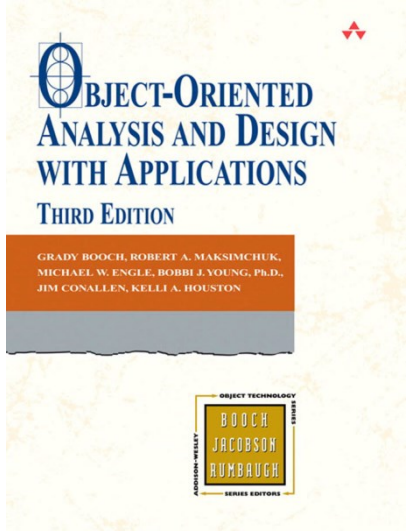
W	Pokok Bahasan
9	Design Pattern
10	Design
11	Implementation
12	
13	Testing
14	
15	Review Materi
16	**Ujian Akhir Semester**

References

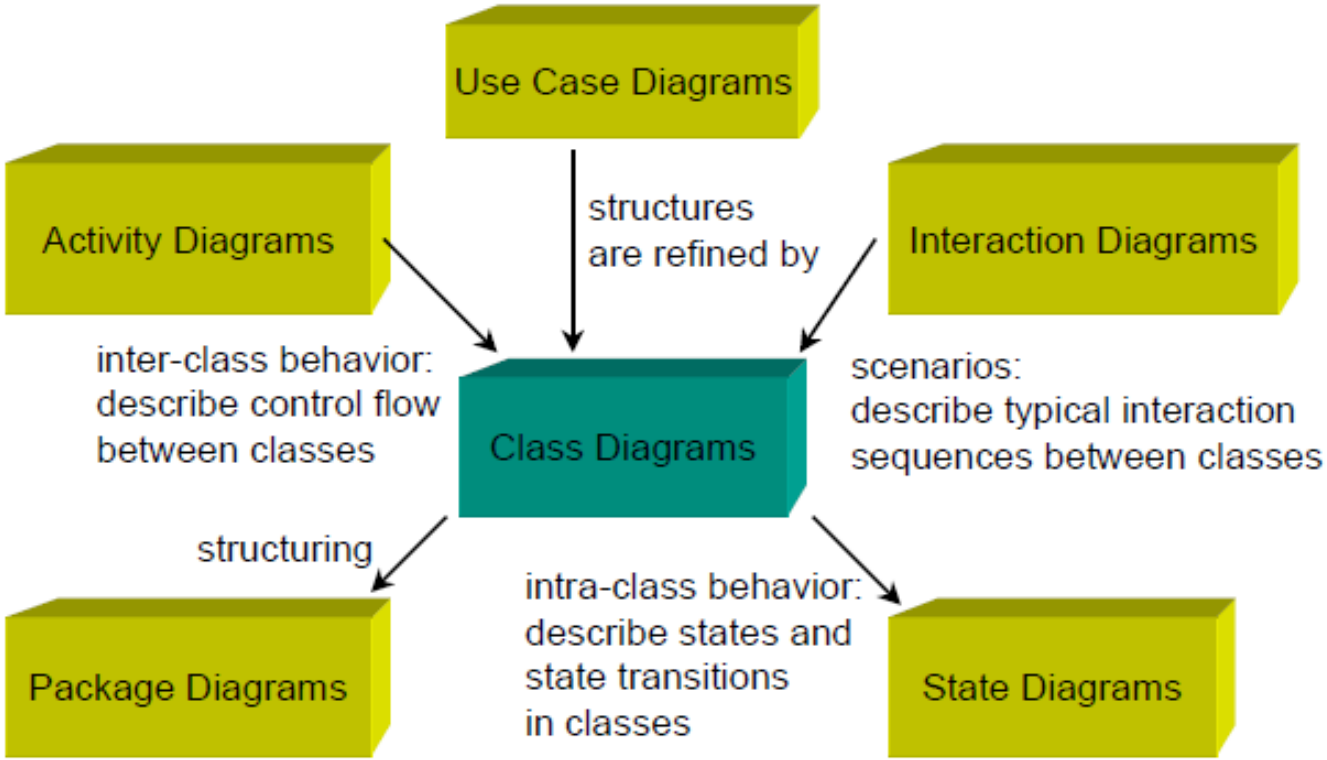
- Brahma Dathan, Sarnath Ramnath – **Object-Oriented Analysis and Design** (2011)



- Grady Brooch, Robert A Maksimchuk, Michael W. Engle, Robbi J. Young, Jim Conallen, Kelli A. Houston – **Object-Oriented Analysis and Design With Applications** Third Edition (2007)



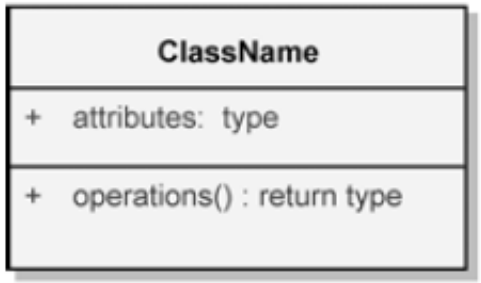
Role of Class Diagrams



Class Diagrams are central for analysis, design and implementation.
Class Diagram are the richest notation in UML

Class Diagram

- Show the classes, methods, and fields
- A class diagram is used to show the existence of classes and their relationship in the logical view of a system
- Class diagram is represented by a box, which is divided into three categories :
  1. The name of the class is given in the top of rectangle
  2. The attributes are shown in the second box
  3. The methods with their return types and parameter are shown in the third box



Attributes

- Visibility of Attributes
  - **+** Public : not hidden from any object
  - **#** Protected : hidden from all but immediate subclasses
  - **-** Private : hidden from all other classes
  - Default is Private

Operations

- Constructor : create object
- Query : se class state
- Update : change attribute values
- Operation can also be public, protected or private
  - Default for operations is public (+)

Example

```
                    Student
-----------------------------------------------------
 -name : String
 -address : String
 -gpa : double
-----------------------------------------------------
+Student(studentName : String, studentAdress: String)
+Student(studentName : String)
+Student()
+setName(studentName : String): void
+setAdress(studentAdress : String): void
+getName(): String
+getGpa(): double
+getAddress(): String
+computeGpa(course : Course,grade:char) : void
```

Relationships

- Classes collaborate with other classes in a variety of ways
- The essential connections among classes :
  - Association
  - Generalization
  - Aggregation
  - Composition

Associations

- Associations represent relationships between instances of classes
  - Relationships that don't fit "Is-A" or "Has-A"
  - Often a weaker form of "Has-A"
  - Relationships between classes

Customer may order several product

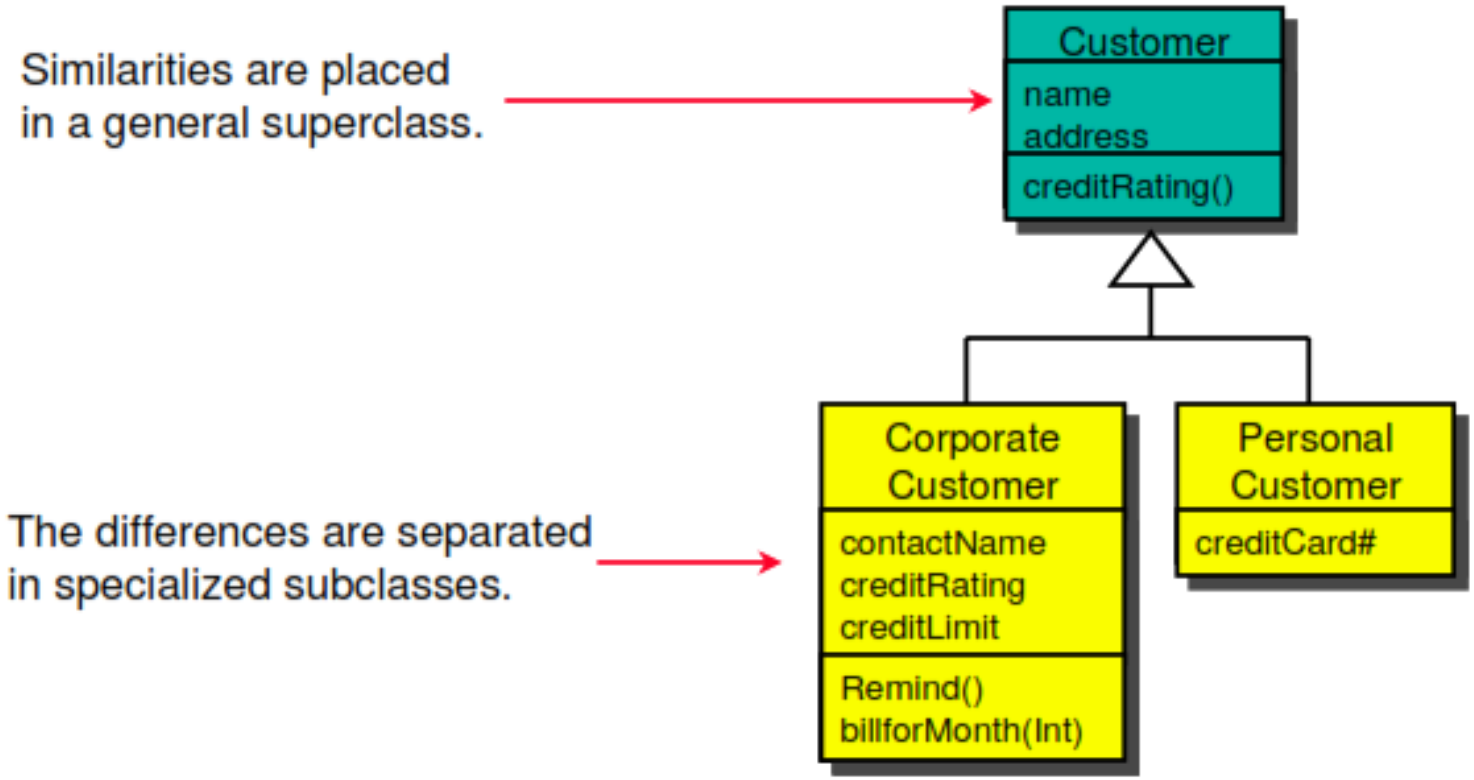
Relationship Multiplicity

1	**Exactly One**
*	Unlimited Number
0..*	Zero or More
1..*	One or More
0..1	Zero or One
3..7	Specified Range

Generalization

- “Is-A” relationship
- Enable inheritance of attributes & operations
- Subsclasses and superclasses
- Subclass be subtituted for superclass

Example: Generalization



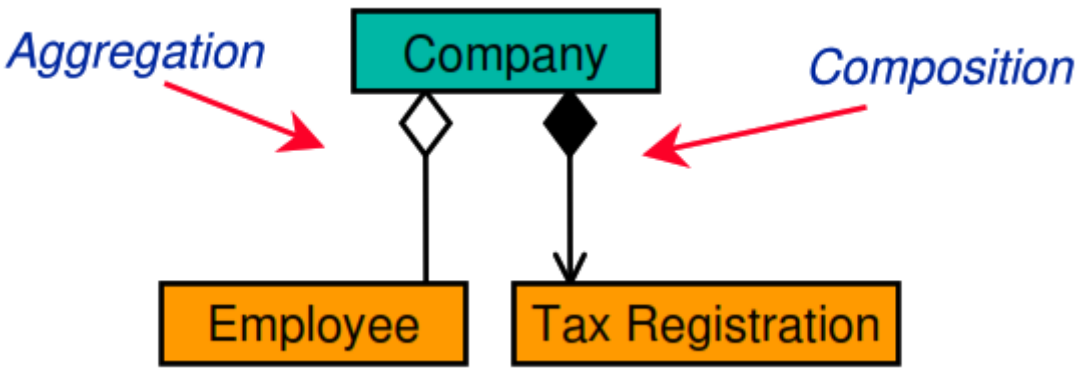
Aggregation

- Special type of Association
- Whole/part relationship
- Has – a relationship
- The part can exist separately from the whole

Composition

- The part can’t exist separately from the whole
- An object can be part of only one composite at a time

Example Aggregation vs Composition



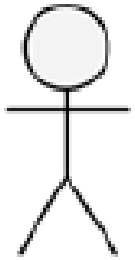
- **Aggregation**: A company has employess. The Employess may change the company
- **Composition**: The company has a tax registration. The Tax registration is tied to the company and dies with it

Use Case Diagram

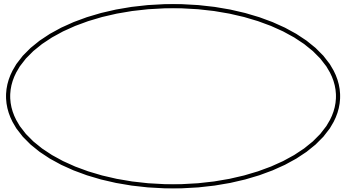
- Use case diagram are used to depict the context of the system to be built and the functionality provided by that system
- Who/ what interact with the system
- What the system to do ?
- Use case involves a user and the system
- Example : student registration for a course

Use Case Diagram

- Actor
  - Actor are entities that interface with the system
  - The can be people or other system



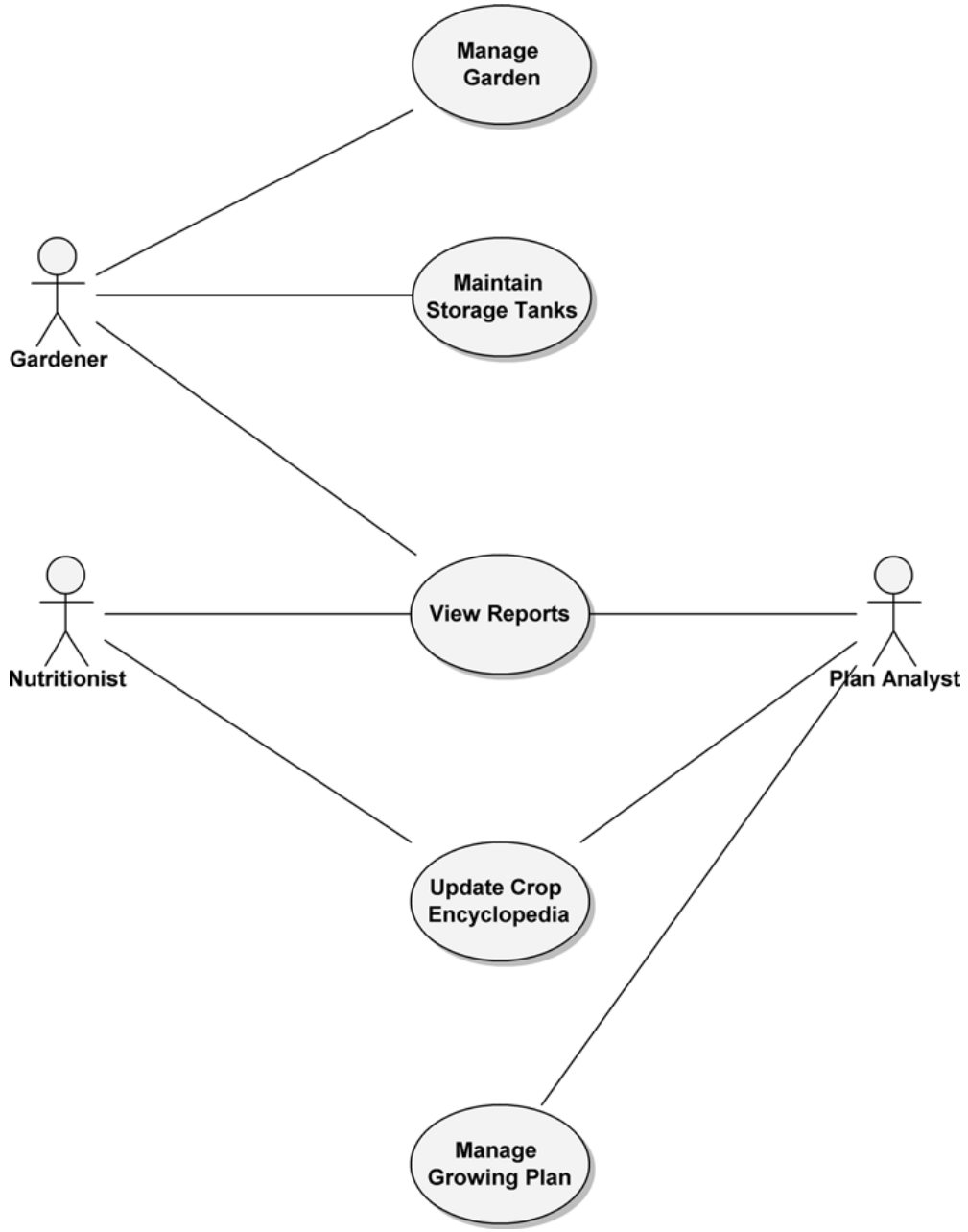
- Use Cases
  - Use cases represent what the actor want your system to do for them



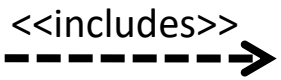
- System Boundary
  - Includes the name of the system inside or on top
  - Actor are outside the scope of the system

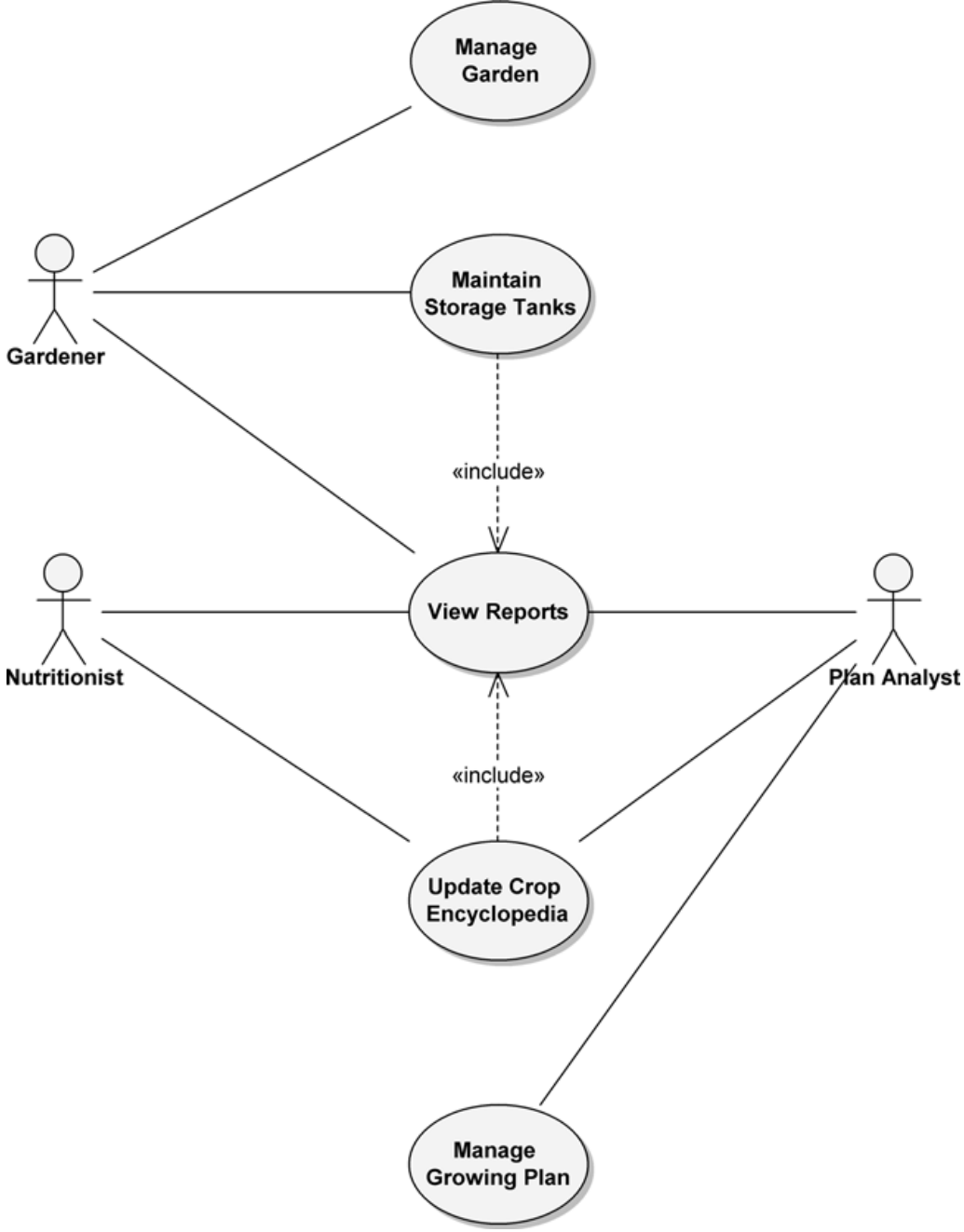
Association Relationship

- Links actor and the Use Cases
- Shows two way communication
  - If one way, arrows are used

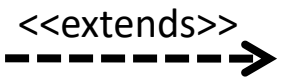

Manage Garden
Maintain Storage Tanks
Gardener
View Reports
Nutritionist
Plan Analyst
Update Crop Encyclopedia
Manage Growing Plan

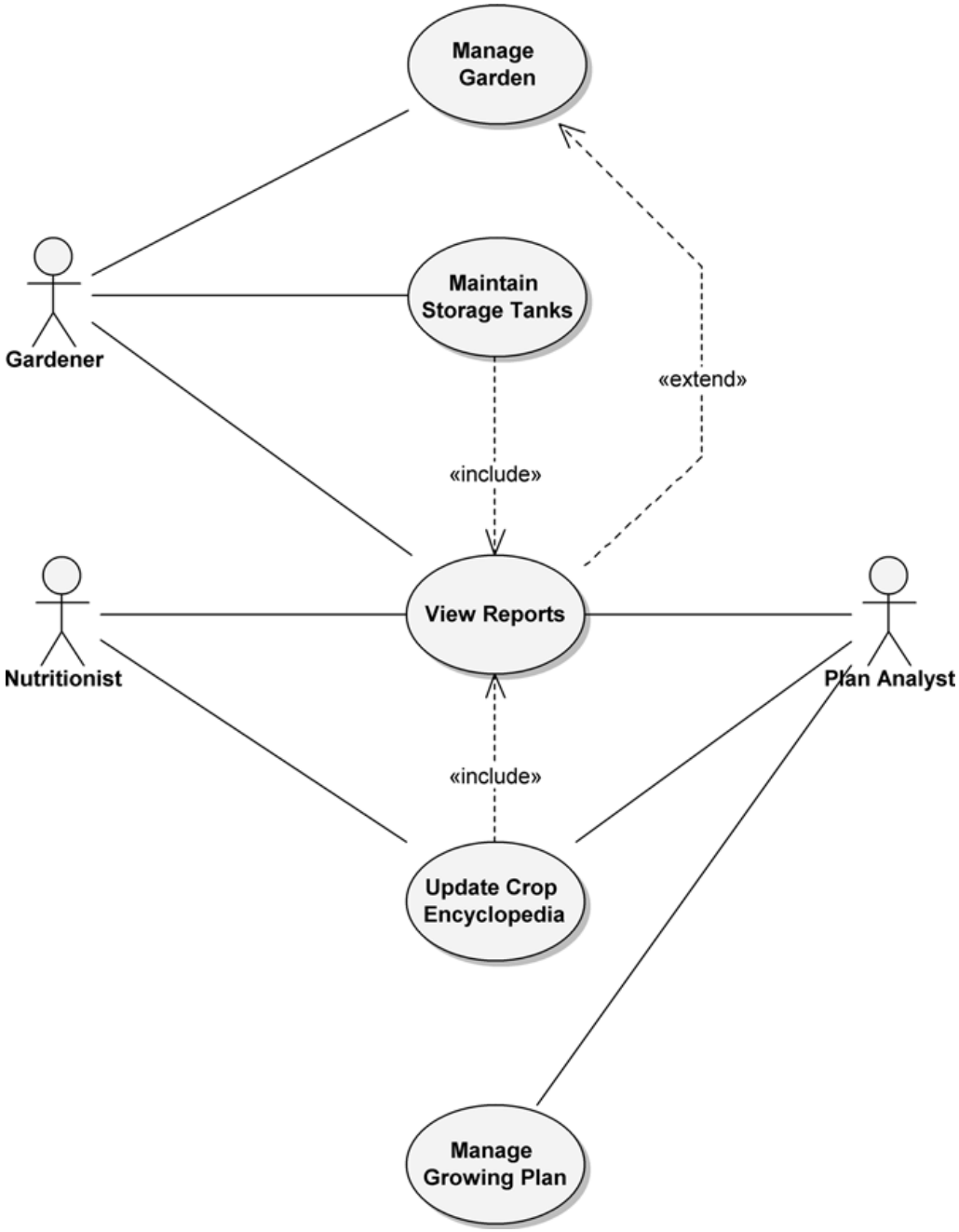
Include Relationship

- <<include>> relationship represents behavior that is factored out of the use case
- <<include>> behavior is factored out for reuse, not because it is an exception
- <<include>> relationship is **to** the using use case
- Notation 


Manage Garden
Maintain Storage Tanks
Gardener
«include»
View Reports
Nutritionist
Plan Analyst
«include»
Update Crop Encyclopedia
Manage Growing Plan

Extend Relationship

- <<extend>> relationships represent exceptional or seldom invoked cases
- The exceptional event flows are separated from the main event flow for clarity
- A use case representing exceptional event flows may extend one or more use cases
- The direction of a <<extend>> relationships is to the extended use case
- Notation 


Manage Garden
Maintain Storage Tanks
Gardener
«extend»
«include»
View Reports
Nutritionist
Plan Analyst
«include»
Update Crop Encyclopedia
Manage Growing Plan

Element of Use-Case Description

Use Case Name:	ID:	Importance Level:
Primary Actor:	Use Case Type:	
Stakeholders and Interests:		
Brief Description:		
Trigger:		
Relationships: (Association, Include, Extend, Generalization)		
Normal Flow of Events:		
Subflows:		
Alternate/Exceptional Flows:		

Use Case Name: Make appointment	ID: 2	Importance Level: High
Primary Actor: Patient	Use Case Type: Detail, essential	

Stakeholders and Interests:
Patient - wants to make, change, or cancel an appointment
Doctor - wants to ensure patient's needs are met in a timely manner

Brief Description: This use case describes how we make an appointment as well as changing or canceling an appointment.

Trigger: Patient calls and asks for a new appointment or asks to cancel or change an existing appointment.

Type: External

Relationships:

Association: Patient
Include: Make Payment Arrangements
Extend: Create New Patient
Generalization:

Normal Flow of Events:

1. The Patient contacts the office regarding an appointment.
2. The Patient provides the Receptionist with their name and address.
3. The Receptionist validates that the Patient exists in the Patient database.
4. The Receptionist executes the Make Payment Arrangements use case.
5. The Receptionist asks Patient if he or she would like to make a new appointment, cancel an existing appointment, or change an existing appointment.
   If the patient wants to make a new appointment,
   the S-1: new appointment subflow is performed.
   If the patient wants to cancel an existing appointment,
   the S-2: cancel appointment subflow is performed.
   If the patient wants to change an existing appointment,
   the S-3: change appointment subflow is performed.
6. The Receptionist provides the results of the transaction to the Patient.

Subflows:

S-1: New Appointment
1. The Receptionist asks the Patient for possible appointment times.
2. The Receptionist matches the Patient's desired appointment times with available dates and times and schedules the new appointment.

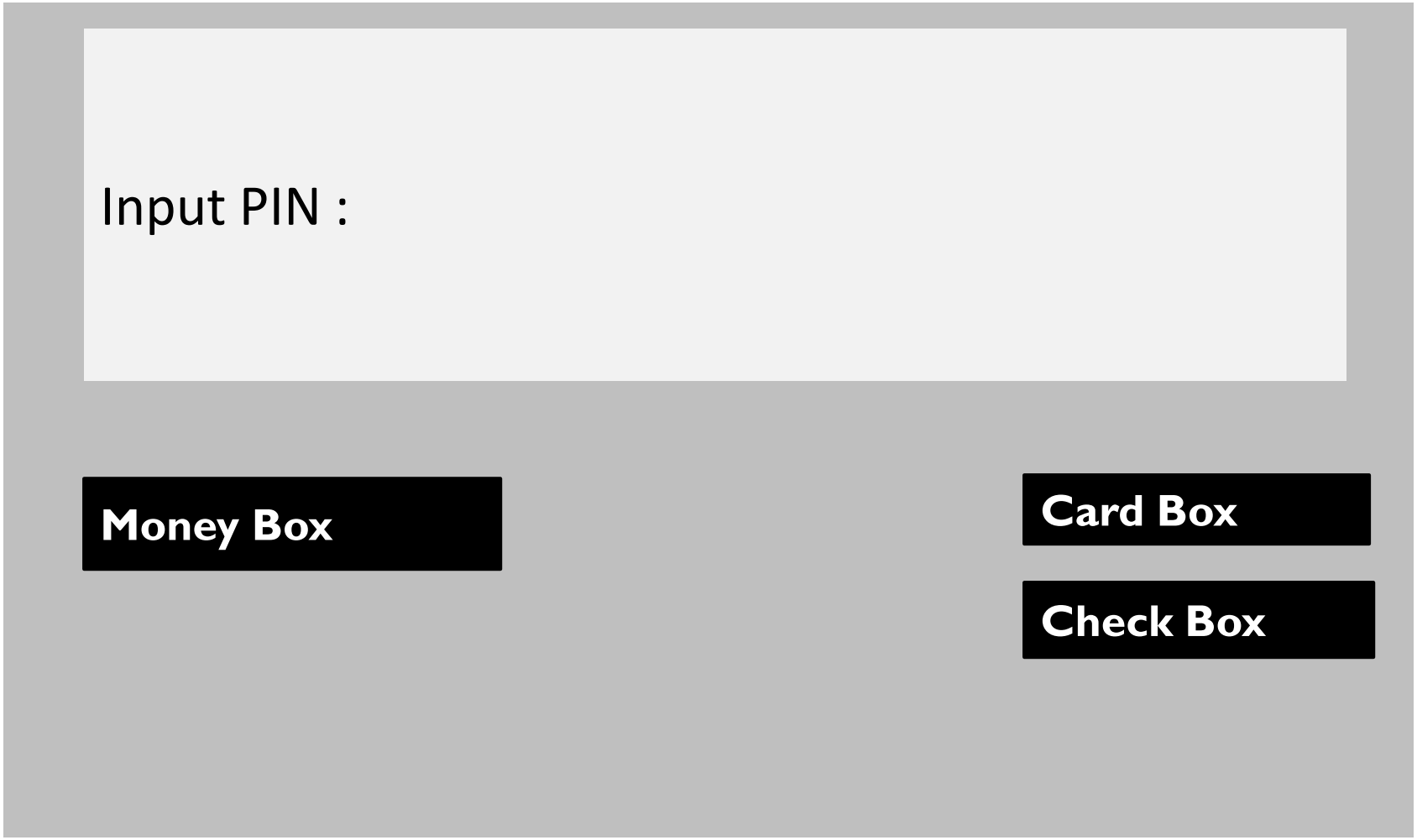
S-2: Cancel Appointment
1. The Receptionist asks the Patient for the old appointment time.
2. The Receptionist finds the current appointment in the appointment file and cancels it.

S-3: Change Appointment
1. The Receptionist performs the S-2: cancel appointment subflow.
2. The Receptionist performs the S-1: new appointment subflow.

Alternate/Exceptional Flows:

3a: The Receptionist executes the Create New Patient use case.
S-1, 2a1: The Receptionist proposes some alternative appointment times based on what is available in the appointment schedule.
S-1, 2a2: The Patient chooses one of the proposed times or decides not to make an appointment.

ATM Sistem



ATM

Main Menu

1. View Balance
2. Transfer
3. Withdrawal
4. Logout

Money Box

Card Box

Check Box

Exercise

- Create a set of use cases for a university library borrowing system. The system will record the books owned by the library and will record who has borrowed what books.
- Before someone can borrow a book, he or she must show a valid ID card that is checked to ensure that it is still valid against the student database maintained by the registrar's office (for student borrowers), the faculty/staff database maintained by the personnel office (for faculty/staff borrowers), or against the library's own guest database (for individuals issued a "guest" card by the library).
- The system must also check to ensure that the borrower does not have any overdue books or unpaid fines before he or she can borrow another book. Every Monday, the library prints and mails postcards to those people with overdue books.
- If a book is overdue by more than two weeks, a fine will be imposed and a librarian will telephone the borrower to remind him or her to return the book(s). Sometimes books are lost or are returned in damaged condition. The manager must then remove them from the database and will sometimes impose a fine on the borrower.

Exercise

- Create a set of use cases for an online university registration system. The system should enable the staff of each academic department to examine the courses offered by their department, add and remove courses, and change the information about them (e.g., the maximum number of students permitted). It should permit students to examine currently available courses, add and drop courses to and from their schedules, and examine the courses for which they are enrolled. Department staff should be able to print a variety of reports about the courses and the students enrolled in them. The system should ensure that no student takes too many courses and that students who have any unpaid fees are not permitted to register. (Assume that a fees data store is maintained by the university's financial office, which the registration system accesses but does not change).

TERIMA KASIH