CH9.Moving On To Design

School of Computer Science
Prof. Euijong Lee

Evolving to Design Models

Design Concepts

- **Abstraction** 추상화
- **Stepwise refinement** 분할 개발.
- **Modularity** 기본 단위 = module, module → component → package → ...
- **Information hiding** 정보은닉
- **Separation of concerns** SOC = 영역, 관심.
- **Top-down versus Bottom-up** ↓ vs. ↑

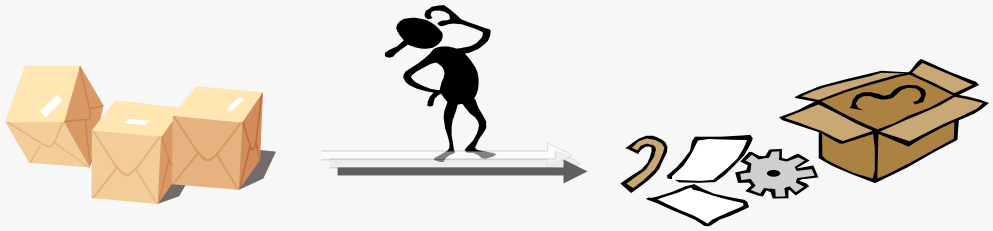
Avoid Classic Design Mistakes

- **Reducing design time** 늘려 맹신 → 시간 부족 실패
  - use timeboxing to eliminate functionality
  - move it to future version
- **Feature creep** SW 완성 과한 기능 추가
  - aware of the impact on cost and time to the user
  - try to move **proposed changes** into future versions
- **Silver bullet syndrome** 이렇게는 되겠지라고 안일
  - just say "no" for claiming omnipotent design tools
- **Switching tools in mid-project** 중간에 도구 변경
  - Do not switch or upgrade unless compulsion

(source: time boxing / www.lucidchart.com)

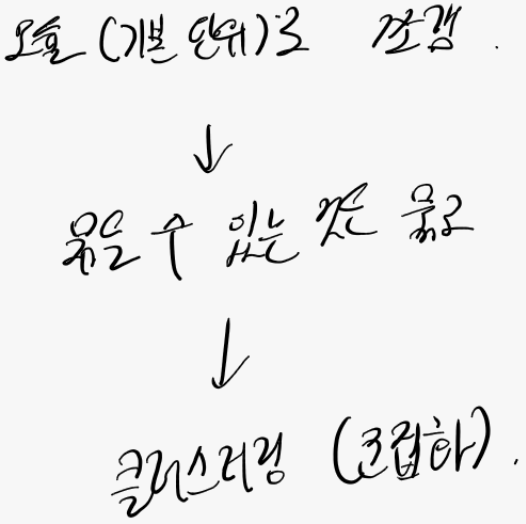
From Analysis to Design

- **Way to evolve problem domain-oriented analysis models into optimal solution domain-oriented design models**
  - Factoring,
  - Partitions and Collaborations, and
  - Layers



Factoring

- **Process of separating out a module into a standalone module in and of itself**
- **Creating modules that account for similarities and differences between units of interest**
- **Relationships to create New classes**
  - Generalization
  - Aggregation
- **Way of factoring**
  - Abstracting
    - abstract classes, super classes
  - Refinement
    - concrete classes, sub classes



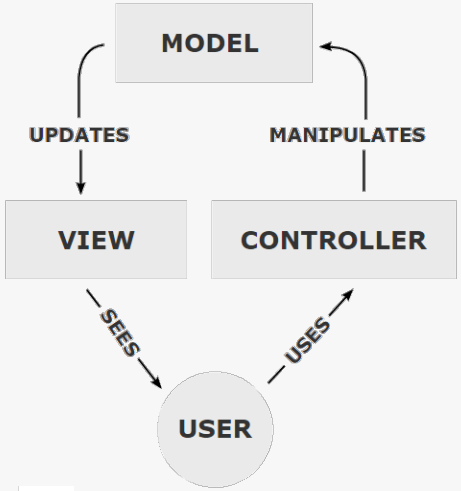
Partitions and Collaborations

- **Partition: create a sub-system of closely collaborating classes**
  - Base partitions on patterns of activity
    (e.g., collaborations found in a communication diagram)
  - Greater coupling among classes may identify partitions
    (e.g., more messages passes between objects suggests that they belong in the same partition)
- **Creating "subsystems" or larger units**
- **Grouping units that collaborate**
  - May have collaboration among units or partitions
- **The more messages or contracts between objects, the more likely they are in the same partition**
- **From communication diagram, in general**
- **Identifying partitions and collaborations determines which classes should be grouped together**

Layers 계층

- Separating different elements of the system and its environment
- Model-view-controller (MVC) architecture
  - Models implement application logic (problem domain)
  - Views and controllers do user interfaces
    - view : handle the output
    - controller : handle the input

보여주는 것.
입력하는 것.



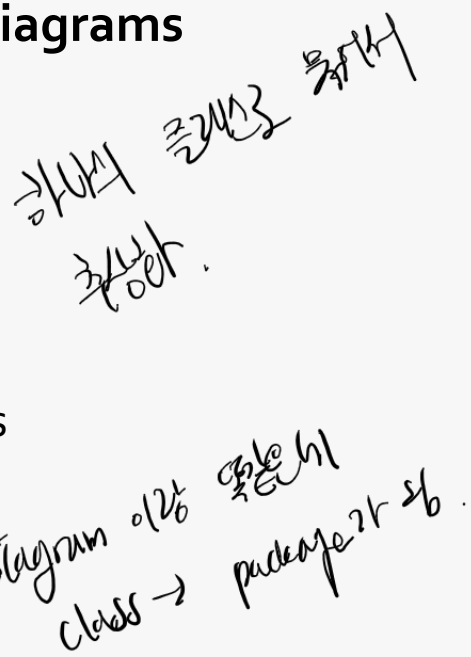
- Separating application logic from user interface logic
- Layers

Layers	Examples
Foundation	Date, Enumeration
Problem Domain	Employee, Customer
Data Management	DataInputStream, FileInputStream
Human–Computer Interaction	Button, Panel
Physical Architecture	ServerSocket, URLConnection

Package & Package Diagram

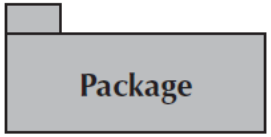
Packages

- **General construct that can be applied to any of the elements in UML models**
- **High-level logical construct to simplifies UML diagrams**
  - groups related elements into a package
- **Package relationship**
  - **dependency relationships** ---------→
    - Shows a dependency between packages
  - **aggregation / association relationships**
    - when packages represent grouping of classes
- **Package diagram**
  - a class diagram that only shows packages
  - can be considered as a high-level or logical architecture
  - If one package is modified, others that depend on it may also require modification



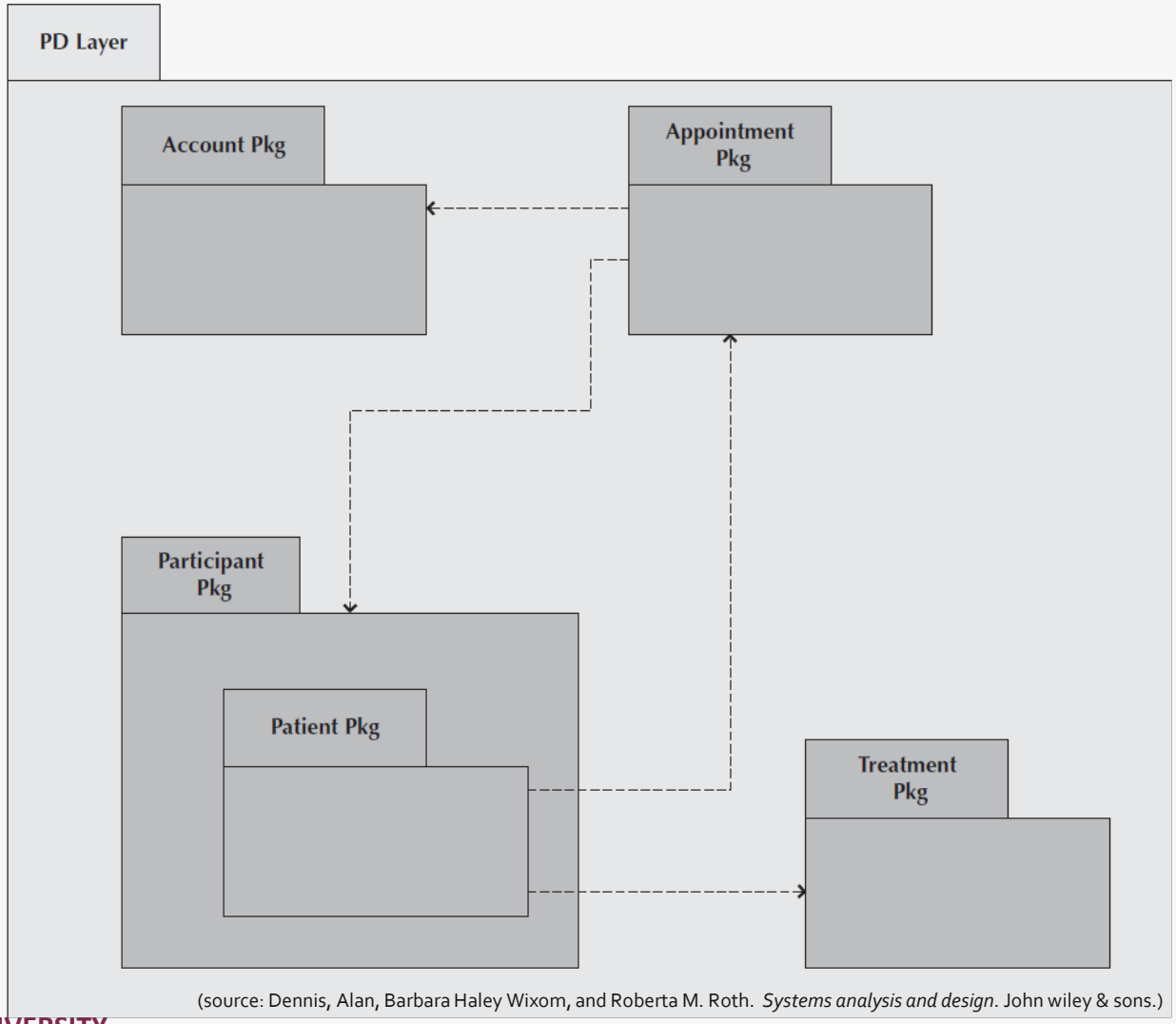
Syntax for Package Diagram

- **Name, Meanings and its Symbol**

A package: ■ Is a logical grouping of UML elements. ■ Is used to simplify UML diagrams by grouping related elements into a single higher-level element.	
A dependency relationship: ■ Represents a dependency between packages: If a package is changed, the dependent package also could have to be modified. ■ Has an arrow drawn from the dependent package toward the package on which it is dependent.	

Package Diagram Example : Appointment System

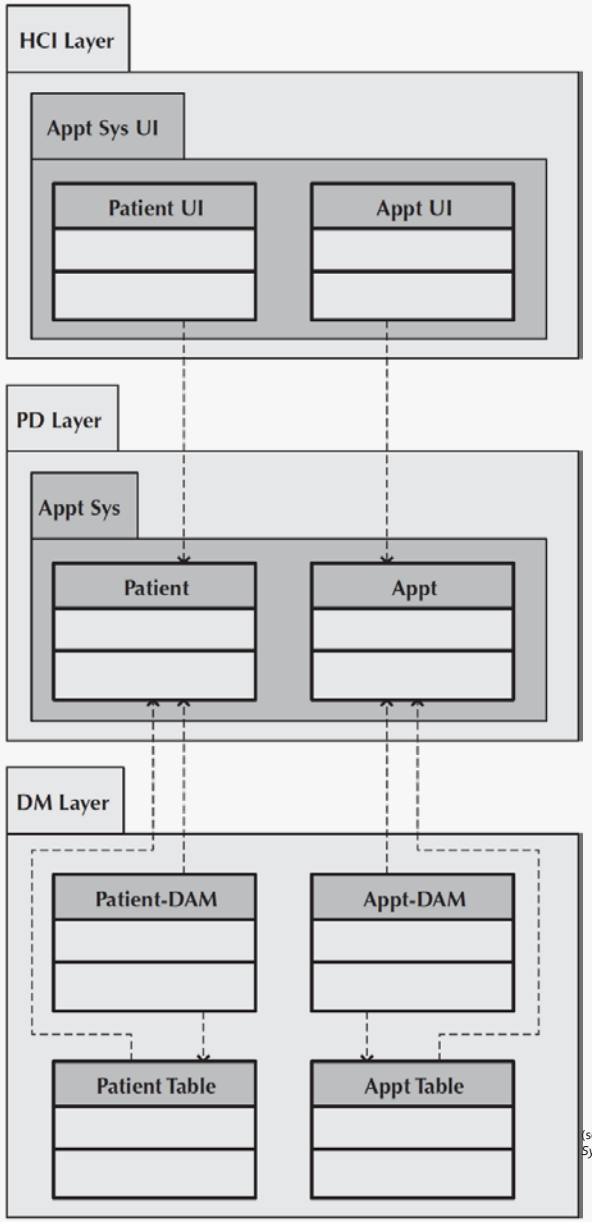
- Among the different Layers.



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Package Diagram Example : Appointment System

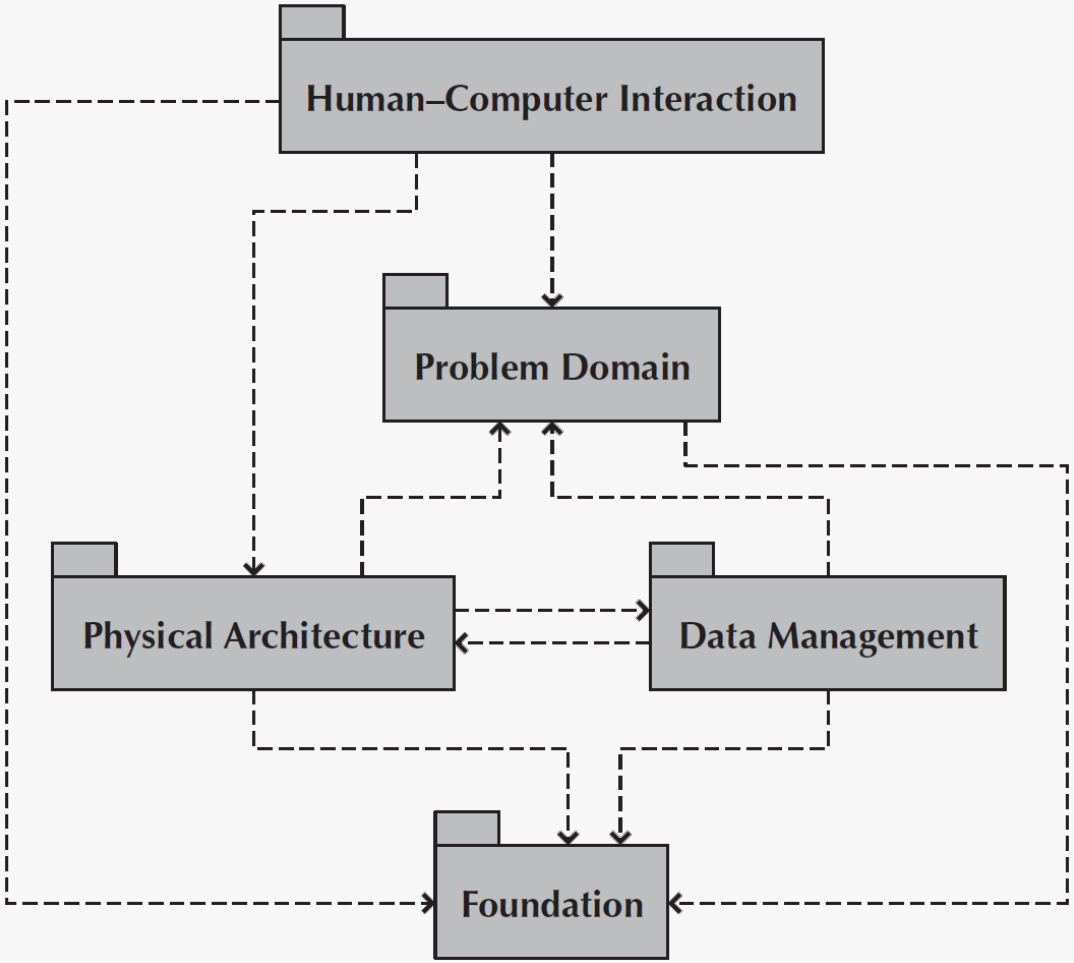
- Layered Package Diagram



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Package Diagram of Dependency Relationships

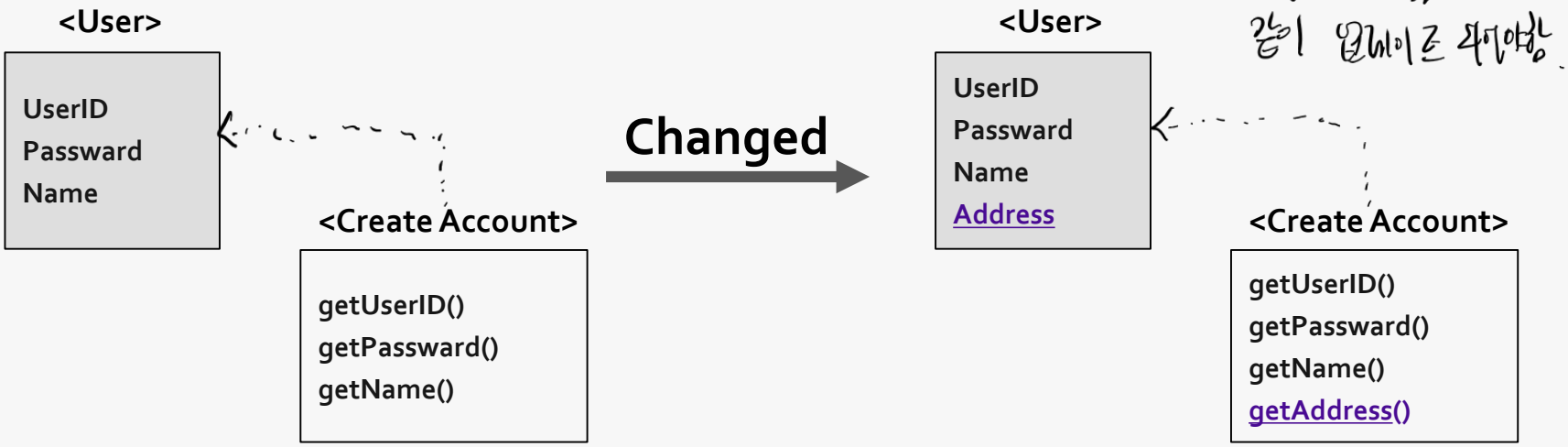
- Among the different Layers.



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Modification Dependency

- **Indicates that a change in one package could cause a change to be required in another package.**
- **Example:**
  - A change in one method will cause the changes of the interface for all objects of this class to change.
  - Therefore, all classes that have objects that send messages to the instances of the modified class could have to be modified.



Creating Package Diagrams

Step 1: Set the context

Step 2: Cluster classes together based on shared relationships

- Any classes in a generalization hierarchy should be kept together in a single partition

Step 3: Model clustered classes as a package

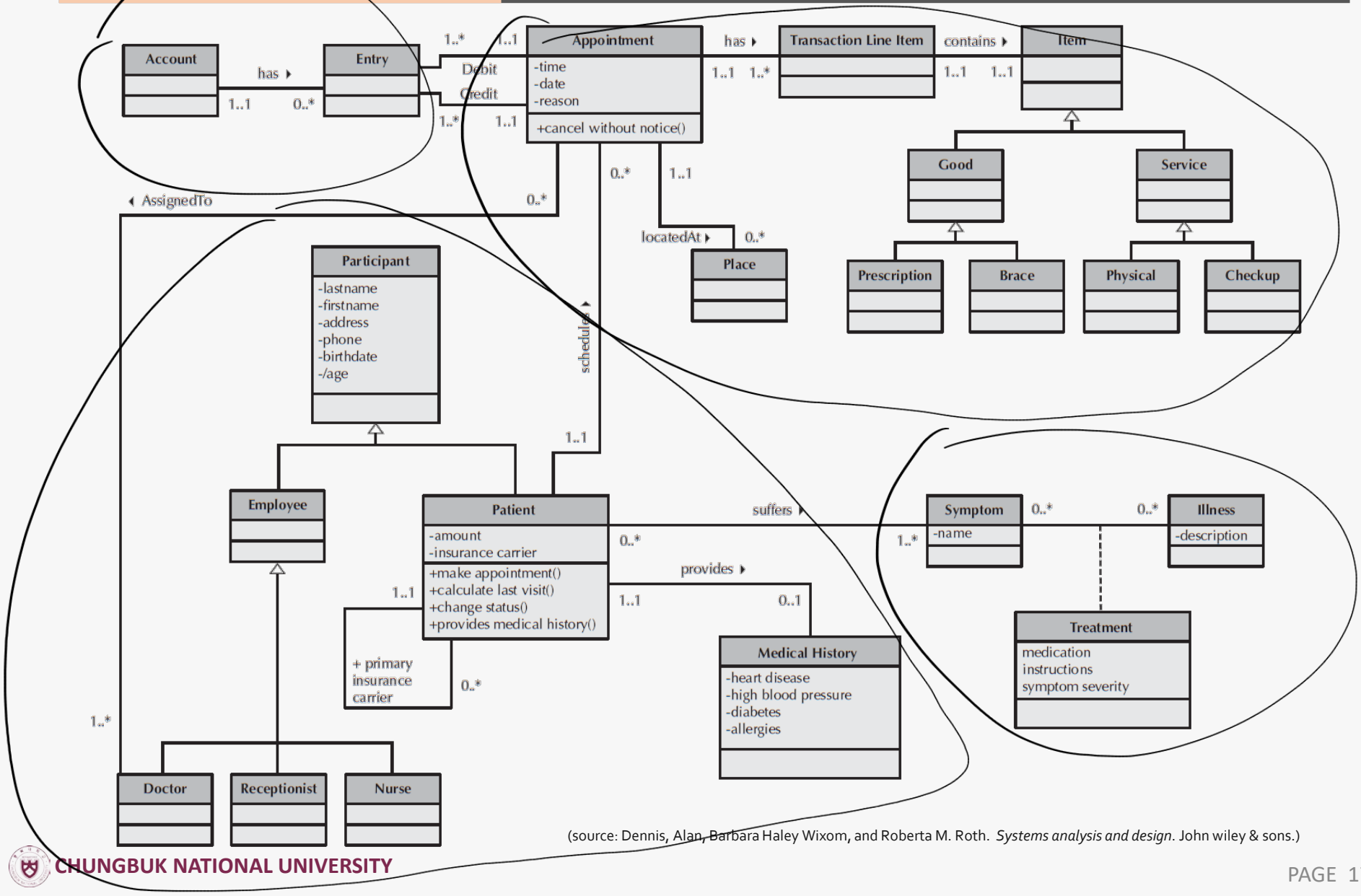
Step 4: Identify dependency relationships among packages

- review the relationship that cross the boundaries of the package

Step 5: Place dependency relationships between packages

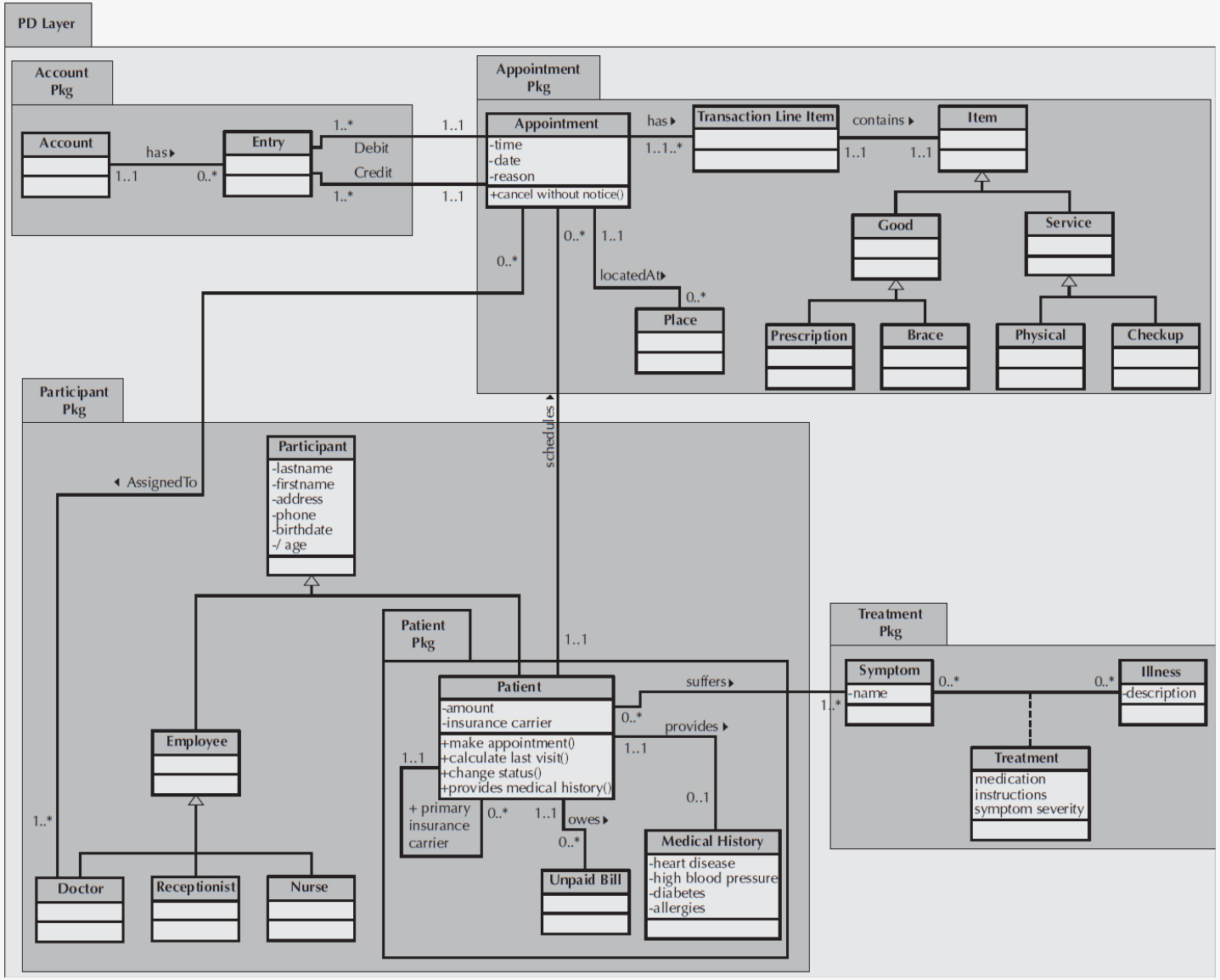
Step 6: Check the dependency relationship using a specific scenario

Class Diagram : the Appointment System



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

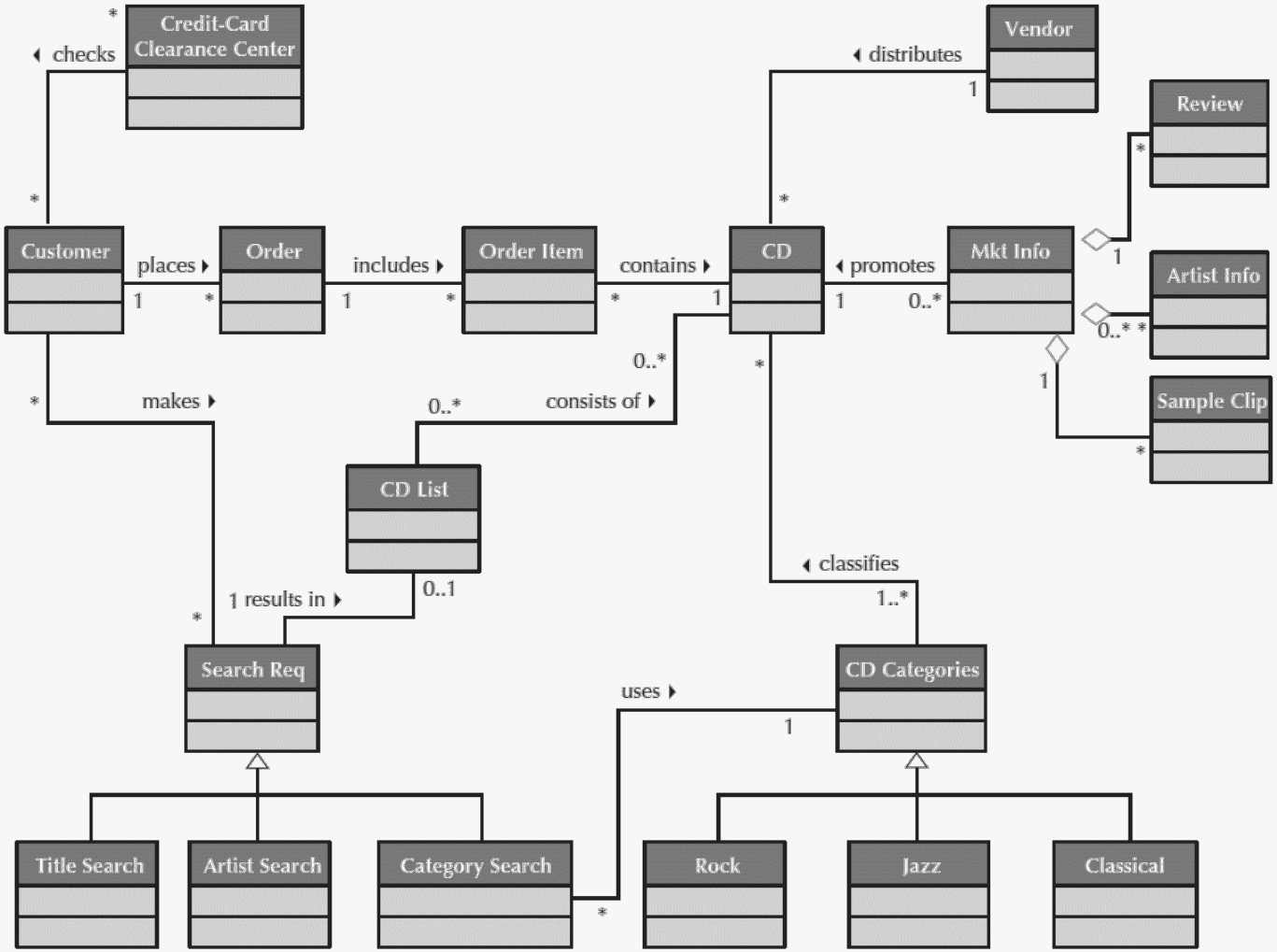
Package Diagram : the Appointment System



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Your Turn – Activity

- Identify PACKAGES from the Class Diagram for CD Selection company

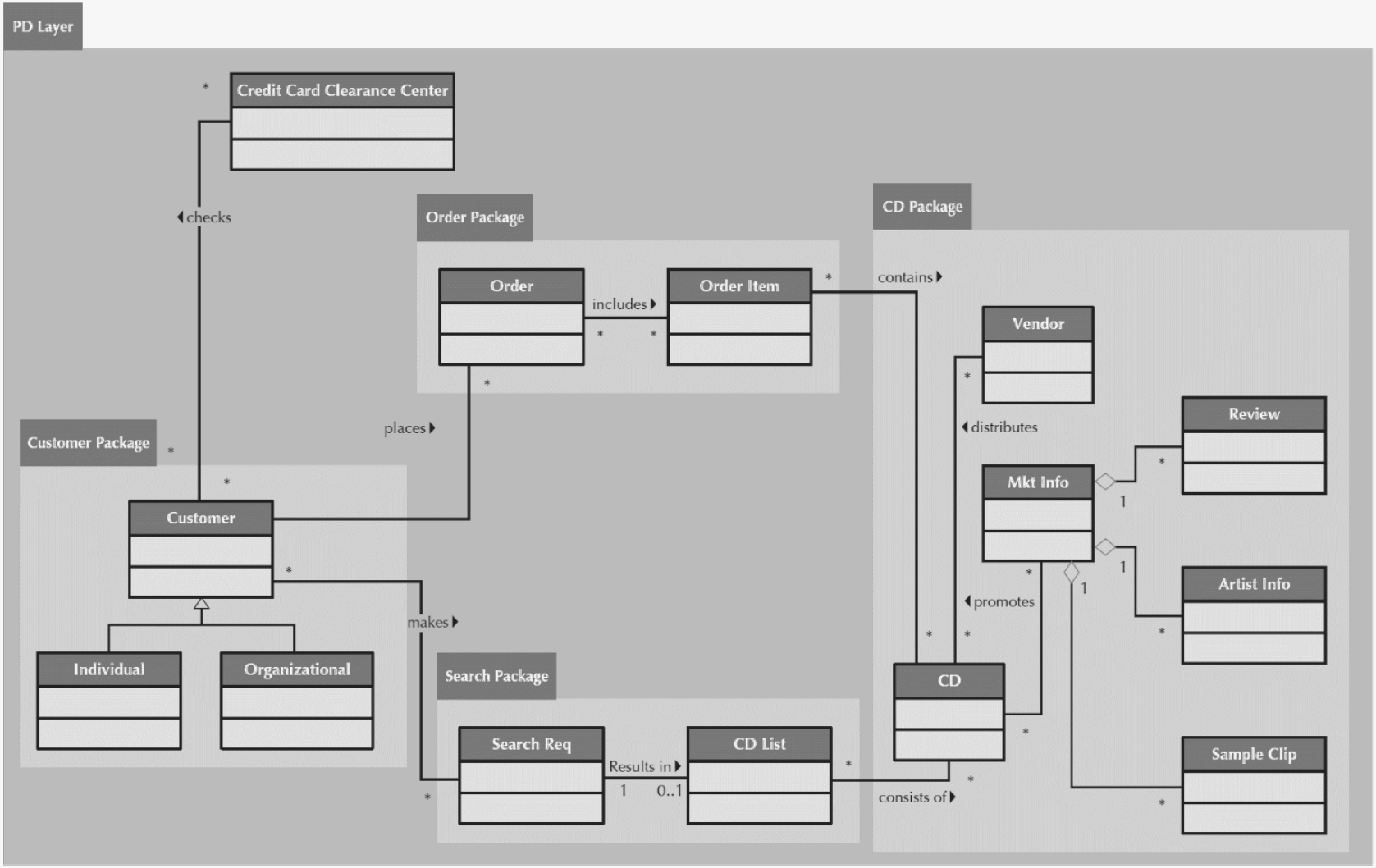


How many packages do you have ?

(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

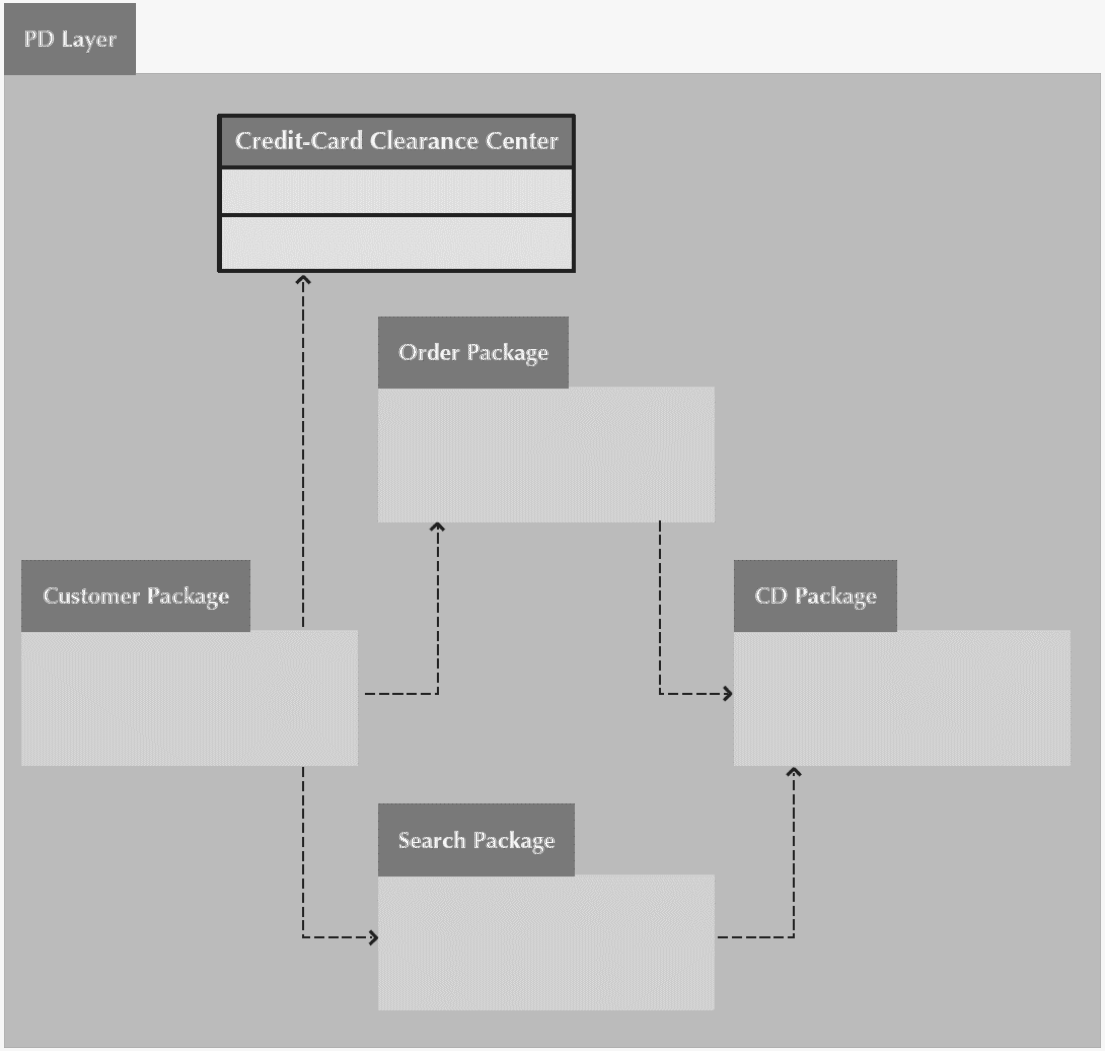
❖ Package Diagram of the PD Layer



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Case Study : CD Selection Company

- Overview Package Diagram of the PD Layer



(source: Dennis, Alan, Barbara Haley Wixom, and Roberta M. Roth. *Systems analysis and design*. John wiley & sons.)

Summary and Discussion

- **When evolving analysis into design models,**
  - it is important to review the analysis models , then add system environment information.
- **From analysis to design**
  - Factoring,
  - Partitions and Collaborations, and
  - Layers

- **Packages and package diagrams**
  - provide structure and less complex views of the new system.
- **What is the benefits of Layered Architecture ?**