6. Structural Modelling using Class, Package Diagrams

IT 3106– Object Oriented Analysis and Design

Level II - Semester 3

Overview

In this section students will

- learn how the objects underlying the behavior modeled in the business process and functional models are organized and presented,
- be introduced to Class Diagrams with their benefits.
- learn CRC cards, Object Diagram, and steps involved in creating them
- learn to Verify and Validate the diagrams.

Intended Learning Outcomes

At the end of this lesson students will be able to

- understand the rules and style guidelines for creating CRC cards, class diagrams, and object diagrams,
- identify the UML concepts of Stereotypes,
- create Structural Models using CRC cards,
- draw Class Diagrams and Object Diagrams,
- illustrate the definitions of association, composition relationships between classes in the system,
- define reflexive relationships,
- illustrate the application of generalization and specialization principles to discover super class/subclass relationships,
- verify and validate the Structural Model.

List of Subtopics

6. Structural Modelling using Class Diagrams (6 hours)
    - 6.1 Introduction to structural modeling [Ref 1: Pg. 163-164]
    - 6.2 Basic elements of structural models [Ref 1: Pg. 164-166, Ref 4]
        - 6.2.1 Classes, Attributes, and Operations
        - 6.2.2 Relationships
            - 6.2.2.1 Association
            - 6.2.2.2 Aggregation, Composition
            - 6.2.2.3 Generalization
    - 6.3 Object Identification [Ref 1: Pg. 166-172]
        - 6.3.1 Textual Analysis
        - 6.3.2 Brainstorming
        - 6.3.3 Common Object Lists
        - 6.3.4 Patterns

List of Subtopics cont…

6.4 CRC Cards [Ref 1: Pg. 172-175]

- 6.4.1 Responsibilities and Collaborations
- 6.4.2 Elements of a CRC Card
- 6.4.3 Role-Playing CRC Cards with Use Cases

6.5 Class, Package Diagrams [Ref 1: Pg. 176-184, Pg. 263-265]

6.6 Creating Structural Models using CRC cards and Class Diagrams [Ref 1: Pg. 185-194]

6.7 Verifying and Validating the Structural Model [Ref 1: Pg. 194-197]

Ref 1: Alan Dennis, Barbara Haley, David Tegarden, Systems analysis design, An Object-Oriented Approach with UML: an object-oriented approach, 5th edition, John Wiley & Sons, 2015, ISBN 978-1-118-80467-4

6.1 Introduction to Structural Modeling

- During analysis, analysts create business process and functional models to represent how the business system will behave.
- At the same time, analysts need to understand the information that is used and created by the business system (e.g., customer information, order information).
- A structural model is a formal way of representing the objects that are used and created by a business system.
- It illustrates people, places, or things about which information is captured and how they are related to one another.
- The structural model is drawn using an iterative process in which the model becomes more detailed and less conceptual over time.

6.1 Introduction to Structural Modeling

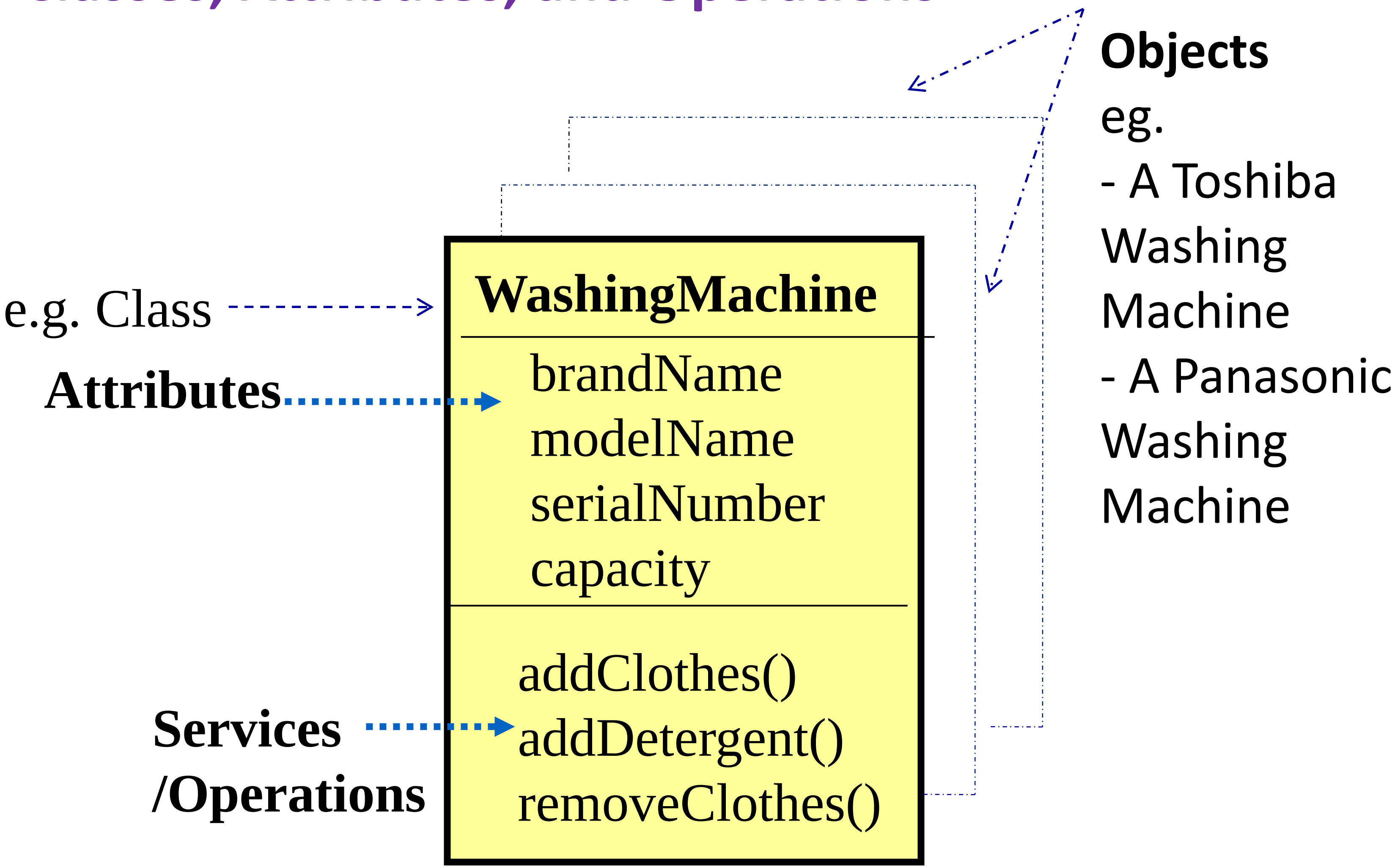
- In analysis, analysts draw a conceptual model, which shows the logical organization of the objects without indicating how the objects are stored, created, or manipulated.
- This model is free from any implementation or technical details, hence the analysts can focus more easily on matching the model to the real business requirements of the system.
- In design, analysts evolve the conceptual structural model into a design model that reflects how the objects will be organized in databases and software.
- At this point, the model is checked for redundancy, and the analysts investigate ways to make the objects easy to retrieve.

6.1 Introduction to Structural Modeling

- Structural models represent the things, ideas, or concepts contained in the domain of the problem.
- They also allow the representation of the relationships among the things, ideas, or concepts.
- Typically, structural models are shown using CRC cards, class diagrams, and, in some cases, object diagrams.

6.2 Basic Elements of Structural Models

Classes, Attributes, and Operations



Objects
eg.
- A Toshiba Washing Machine
- A Panasonic Washing Machine

e.g. Class

Attributes

Services /Operations

WashingMachine

brandName
modelName
serialNumber
capacity

addClothes()
addDetergent()
removeClothes()

6.2 Basic Elements of Structural Models

The Need for Relationships

- All systems are made up of objects and classes.
- Conceptually, objects do not exist in isolation.
- System behavior is achieved through the interactions of the objects in the system.

6.2 Basic Elements of Structural Models

The Need for Relationships

Example :

- When a member wants to borrow a book in a library system (*borrowing* use case), the system has to interact with the following objects:

  book, copy, borrower and borrowed copy

- For the *borrowing* use case following are some of the messages that these objects have to send and receive.

  checkBorrowerId , checkCopyBorrowable, checkOverdue, checkOverlimit etc.

6.2 Basic Elements of Structural Models

The Need for Relationships

- When there is an object interaction, it indicates there is a Relationships between the corresponding classes.
- A relationship is a semantic connection between classes.
- It allows one class to know about the attributes, operations of another class.

6.2 Basic Elements of Structural Models

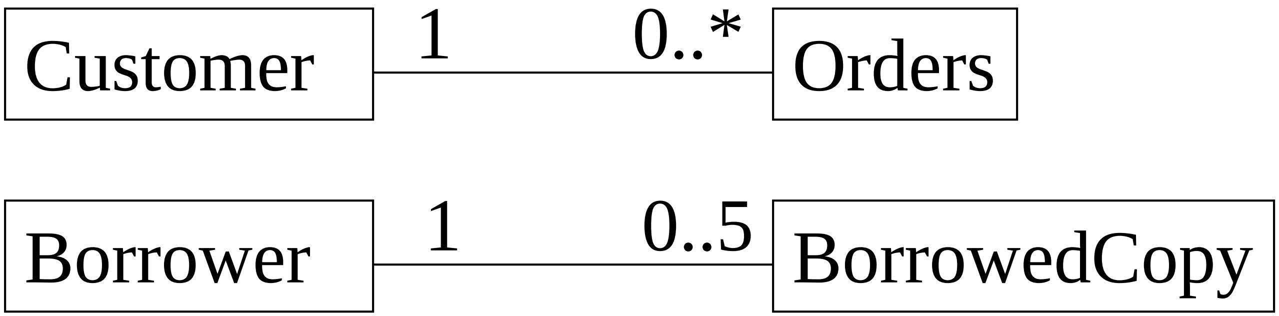
Relationships

- There are many different types of relationships that can be defined, but all can be classified into three basic categories of data abstraction mechanisms:
  - Association Relationships,
  - Aggregation/Composition Relationships, and
  - Generalization Relationships.

6.2 Basic Elements of Structural Models

Associations

- Indicate a connection (a link) between classes.
- Each class can send messages to the other.
- It can be bi-directional or unidirectional.
- In UML 2;
  - Bi-directional associations are drawn without arrowheads altogether.
  - Unidirectional associations have a single arrow.



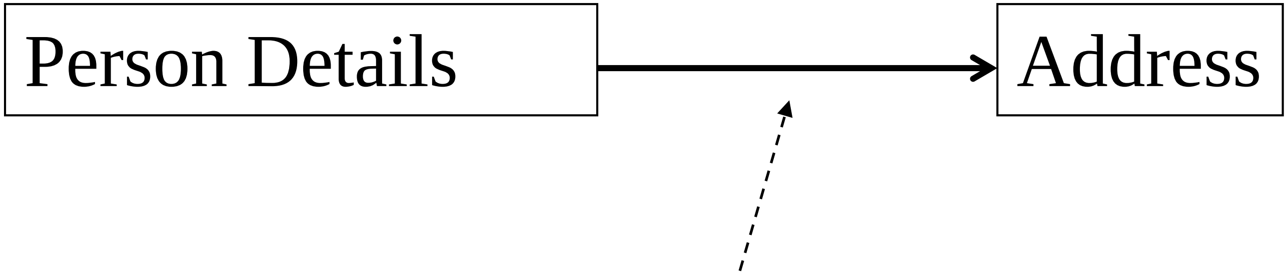
© e-Learning Centre, UCSC

6.2 Basic Elements of Structural Models

Associations

Example

- PersonDetails object has an object reference to the Address object, but not vice versa.
- Messages can only be sent from PersonDetails to Address.

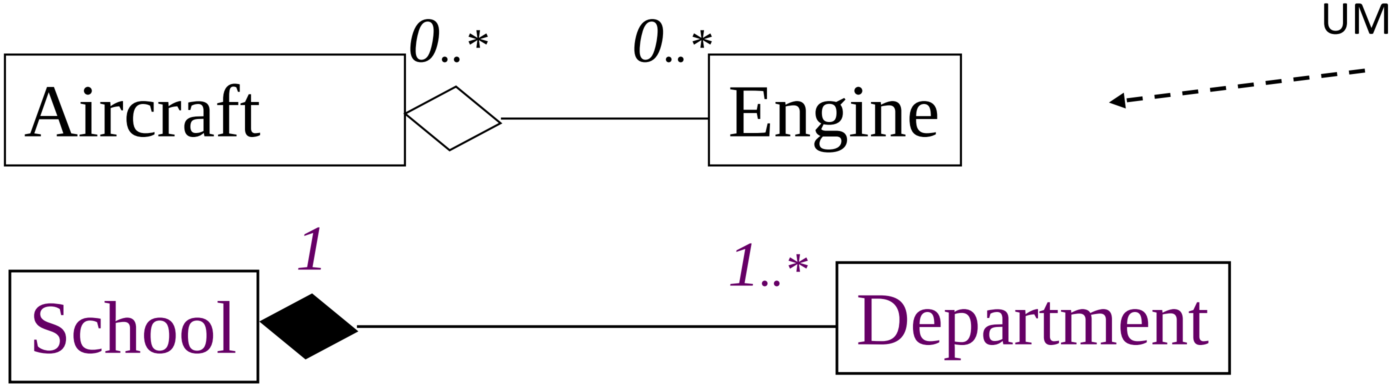


Unidirectional Link

6.2 Basic Elements of Structural Models

Associations

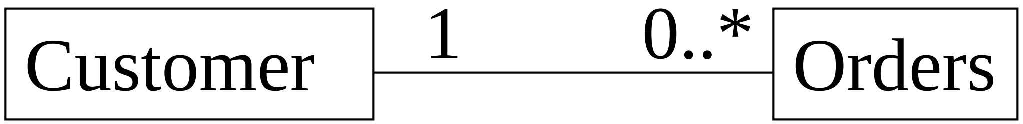
- An aggregation is a stronger form of association.
- It is a relationship between a whole and its parts or composition.



6.2 Basic Elements of Structural Models

Specifying Relationships

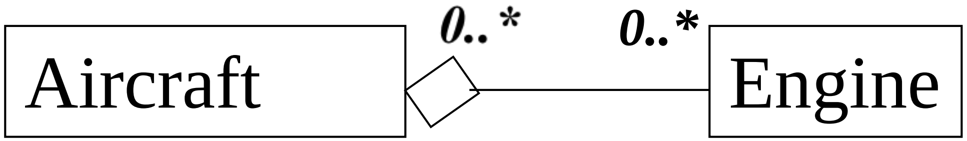
- Association Relationships



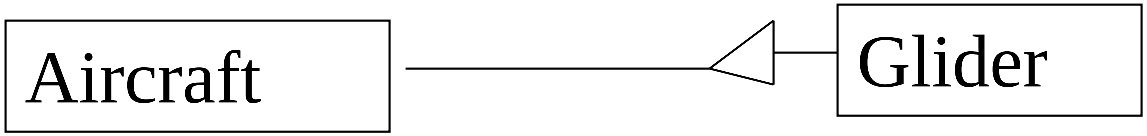
- Aggregation

  Stronger form of

  Association

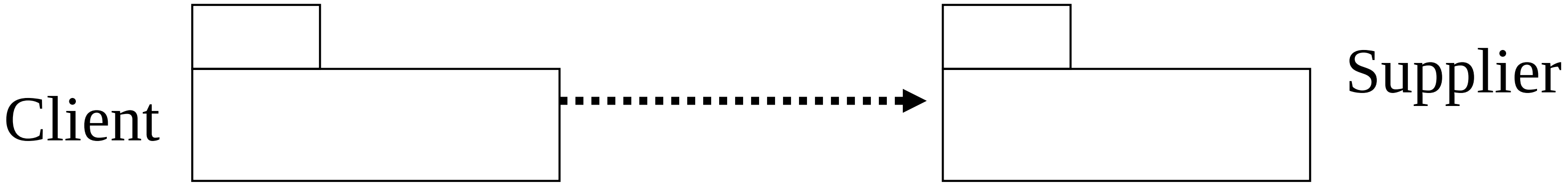


- Generalization



- Dependency Relationship

  Weaker form of relationship



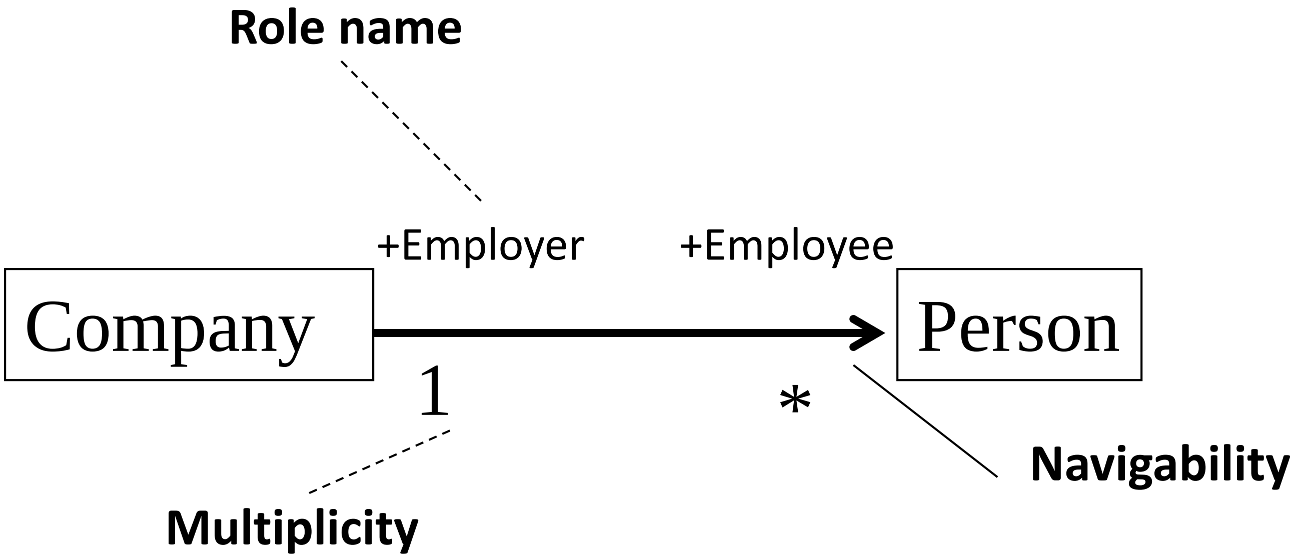
6.2 Basic Elements of Structural Models

Specifying Relationships

- Association Syntax

  Associations may have:

  - an association name , role names , multiplicity and navigability



6.2 Basic Elements of Structural Models

Specifying Relationships

- An Association may be named.

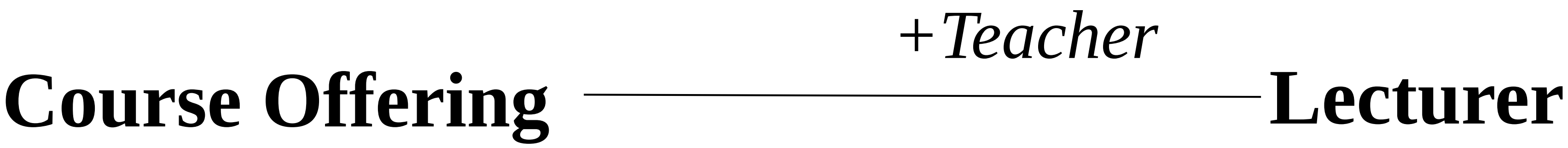
  eg. *employs*
- Usually the name of the association is an *active verb* or *verb phrase* that communicates the meaning of the relationship.

  eg. **a Lecturer** *teaches* **a Course**
- Association name is optional.
- Names are added to improve the clarity .
- Aggregation relationships typically are not named. They are read using the words “has”, “part of” or “contains”.

6.2 Basic Elements of Structural Models

Role Names

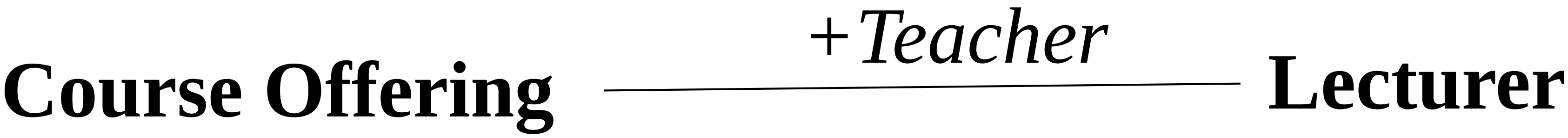
- The end of an association where it connects to a class is called an *association role*.
- *Role name* can be used instead of association names.
- It is a noun that describes the reason the relationship exists.
- The *role name* is placed on the association near the class it modifies.



6.2 Basic Elements of Structural Models

Role Names

- A *role name* may be placed on one or both ends of an association line.
- It is not necessary to have both a role name and an association name.



Course Offering

+Teacher

Lecturer

6.2 Basic Elements of Structural Models

Multiplicity Indicators

1	Exactly one
0..*	Zero or more
1..*	One or more
0..1	Zero or one
5..8	Specific Range (5,6,7 or 8)

6.2 Basic Elements of Structural Models

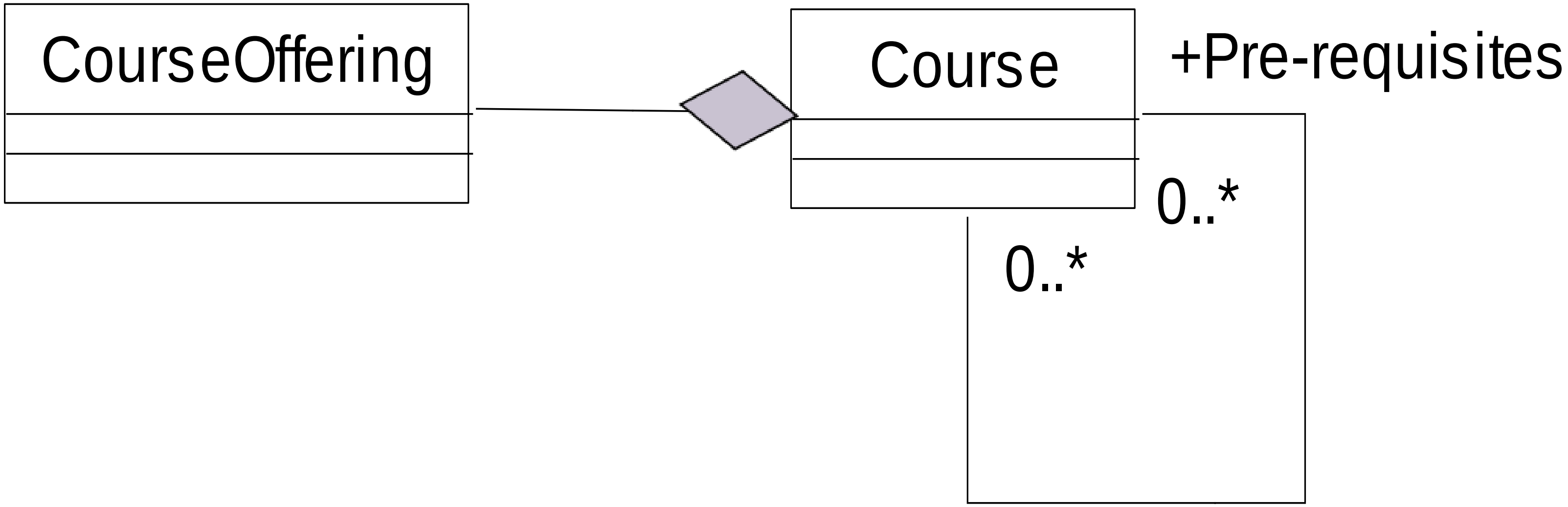
Reflexive Relationships

- Sometimes a class is in an association with itself.
- This can happen when a class has objects that can play a variety of roles.
- This is shown on the class diagram as a reflexive association or aggregation.
- Role names rather than association names are typically used for reflexive relationships.

6.2 Basic Elements of Structural Models

Reflexive Relationships cont…

- One Course object playing the role of Prerequisite is related to zero or more course objects.
- One Course object is related to zero or more course objects playing the role of Prerequisite.



© e-Learning Centre, UCSC

6.2 Basic Elements of Structural Models

Inheritance (Generalization)

- Provides the capability to create a hierarchy of classes.
- Common structure and behavior are shared among classes.
- The term super-class is the name given to the class holding the common information.
- The descendants are called subclasses.
- A subclass inherits all attributes, operations, and relationships that are defined for all of its super-classes.

6.2 Basic Elements of Structural Models

Inheritance (Generalization)

- An inheritance relationship:
  - is not a relationship between different objects.
  - is a relationship between different classes.
  - is never named.
  - *Role* names are not used.
  - Multiplicity does not apply.
- Inheritance is the key to reuse.
  - A class can be created for one application
  - A sub class may be created to add more information needed for a different application.

6.2 Basic Elements of Structural Models

Inheritance (Generalization)

- There are two ways to find inheritance in any system: *Generalization* and *Specialization*.
- *Generalization* provides the capability to create super-classes that encapsulate structure and behavior common to several classes.
- *Specialization* provides the ability to create subclasses that represent refinement to the super-class. Typically structure and behavior are added to the new subclass.

6.2 Basic Elements of Structural Models

Single Inheritance vs. Multiple Inheritance

- With Single inheritance, a class has one set of parents.
  - Savings A/C is a kind of Account.
- Multiple inheritance involves more than one chain of super-classes.
  - Interest cheque A/C is a kind of Savings A/C and also a kind of Current A/C.

6.3 Object Identification

- Different approaches have been suggested to aid the analyst in identifying a set of candidate objects for the structural model.
- The four most common approaches are
  - Textual analysis,
  - Brainstorming,
  - Common object lists, and
  - Patterns.
- Most analysts use a combination of these techniques to make sure that no important objects are missed out.

6.3 Object Identification

Textual analysis

- The analyst performs textual analysis by reviewing the use-case diagrams and examining the text in the use-case descriptions to identify potential objects, attributes, operations, and relationships.
- The **nouns** in the use case suggest possible classes, and the verbs suggest possible operations.

6.3 Object Identification

Brainstorming

- Brainstorming is a process that a set of individuals suggest potential classes that could be useful for the problem under consideration.
- Once a sufficient number of candidate objects have been identified, the participants should discuss and select which of the candidate objects should be considered further.
- Further brainstorming can take place to identify potential attributes, operations, and relationships for each of the identified objects.

6.3 Object Identification

Common Object Lists

- A common object list is simply a list of objects common to the business domain of the system.
- There are libraries of reusable objects that have been created for different business domains.
- These objects can be used as the initial list objects.

6.3 Object Identification

Patterns

- There are many definitions of a pattern.
- A pattern is simply a useful group of collaborating classes that provide a solution to a commonly occurring problem.
- Patterns provide a solution to commonly occurring problems, and hence they are reusable.
- We can look at the objects previously identified through textual analysis, brainstorming, and/or common object lists and see if it makes sense to map any of them into any predefined reusable patterns.
- If we are developing a business information system in one of the business domains where patters have been identified, then the patterns developed for that domain may be a very useful starting point in identifying needed classes and their attributes, operations, and relationships.

6.4 CRC Card

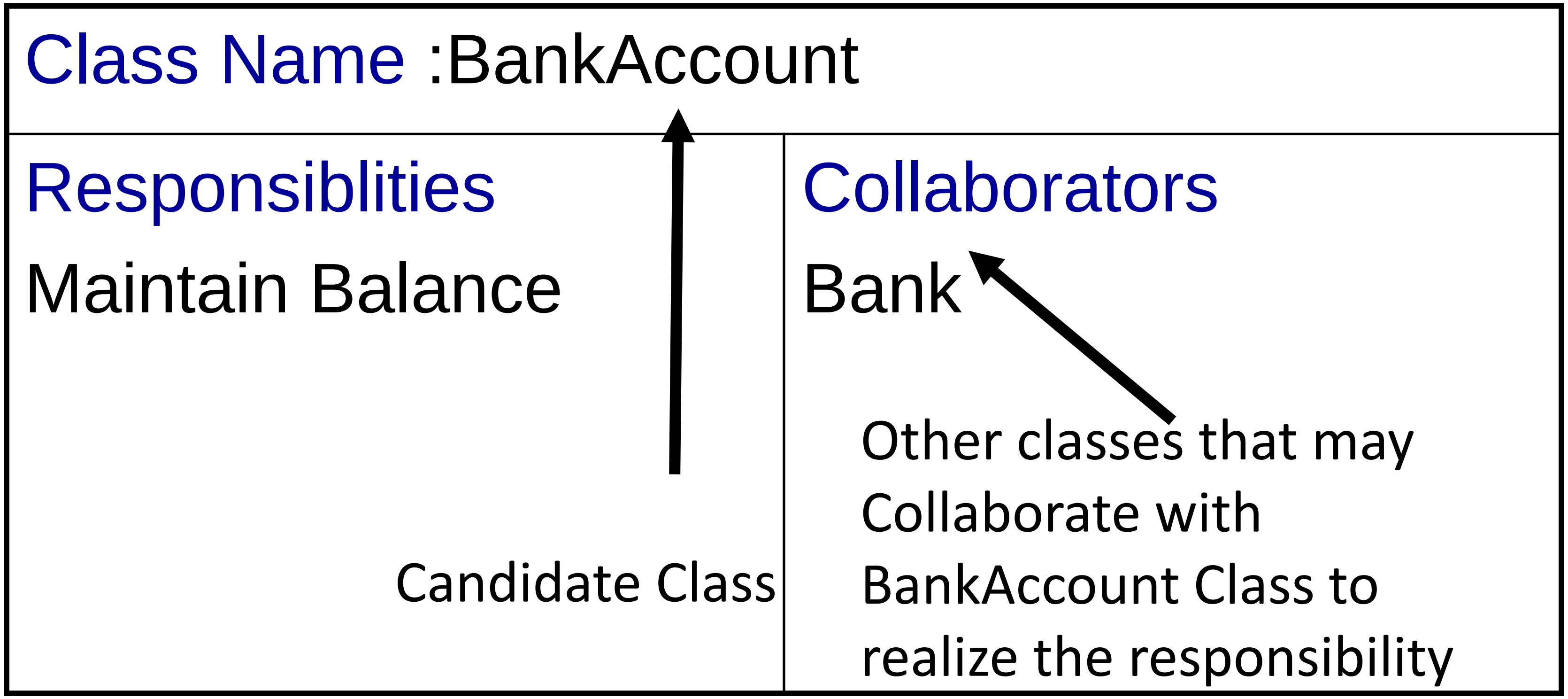
- CRC (**Class–Responsibility–Collaboration**) cards are used to document the responsibilities and collaborations of a class.

Responsibilities and Collaborations

- Responsibilities : Knowing , Doing
- Collaboration : Objects working together to service a request
- Object oriented systems are made up of objects working together to provide functionality.
- CRC cards can help to form a “bridge” from structural model to behavioral model.

6.4 CRC Card

CRC Card Template



Class Name :BankAccount

Responsiblities

Maintain Balance

Collaborators

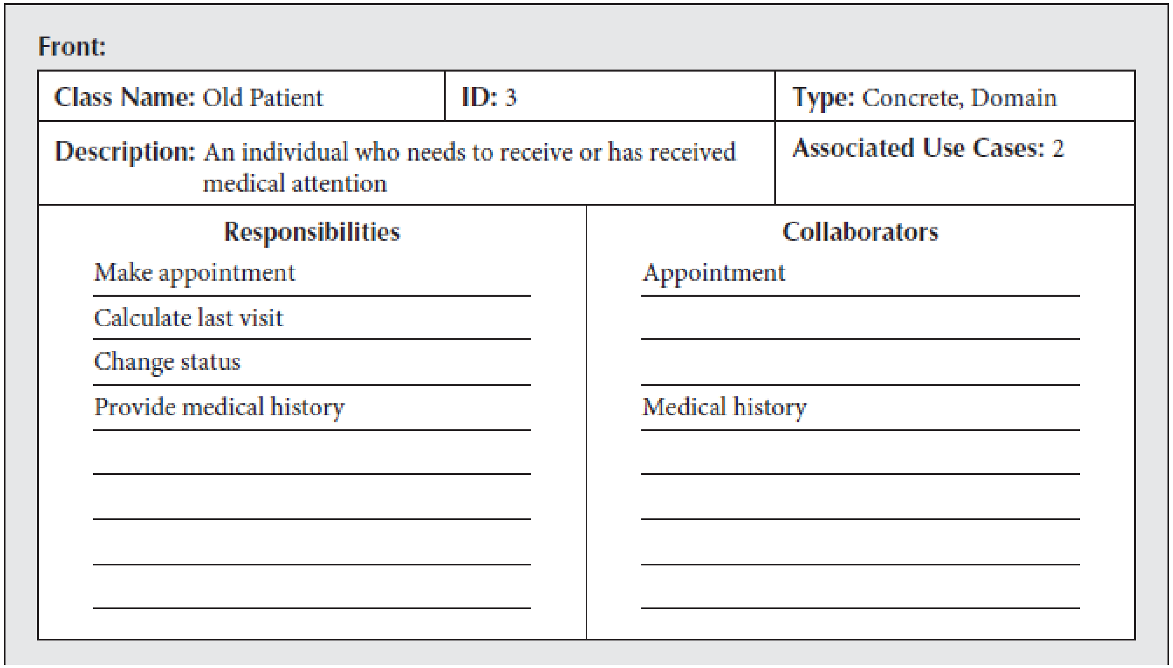
Bank

Candidate Class

Other classes that may Collaborate with BankAccount Class to realize the responsibility

Sample CRC Card

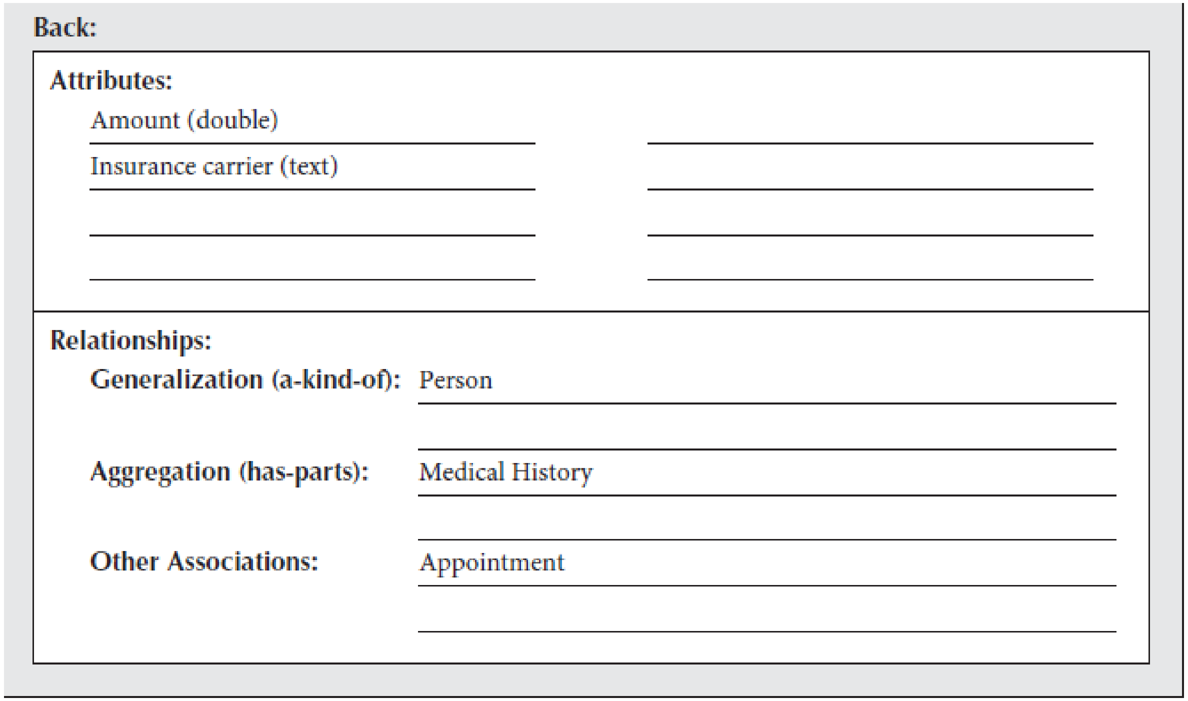
(No Standard Template)



Front:

Class Name: Old Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	Collaborators
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history



Back:

Attributes:

Amount (double)

Insurance carrier (text)

Relationships:

Generalization (a-kind-of):	Person
Aggregation (has-parts):	Medical History
Other Associations:	Appointment

© e-Learning Centre, UCSC

6.4 CRC Card

- Extremely simple technique
- Rather than using diagrams to develop models use 4 x 6 index cards
- Rather than indicating attributes and operations on the cards, they identify *responsibilities* and *collaborators*

6.4 CRC Card

Elements of a CRC Card

- **The name** of the class, at the top
- **The responsibilities** of the class, on the left-hand side
- **The collaborators** of the class, which help to carry out each responsibility, on the right-hand side of the card

Many templates exist

6.4 CRC Card

Role-Playing CRC Cards with Use Cases

- CRC cards are used to document the essential properties of a class.
- Once the cards are filled out, the analyst can use the cards in role-playing
- Each CRC card should be assigned to an individual who will perform the operations for the class on the CRC card.
- It is used to uncover missing properties by executing the different scenarios associated with the use cases.
- Role-playing also can be used as a basis to test the clarity and completeness of the evolving representation of the system.

Typical steps in , Role-Playing CRC Cards with Use Cases

1. ***Create CRC Cards :*** Create the CRC cards first and then transfer the information into a class diagram later.
2. ***Review CRC Cards :*** Review the CRC cards to determine if additional candidate objects, attributes, operations, and relationships are missing.
3. ***Role-Play the CRC Cards :*** Each CRC card should be assigned to an individual who will perform the operations for the class on the CRC card.
4. ***Create Class Diagram*** : Information contained on the CRC cards is transferred to the class diagrams.
5. **Review Class Diagram** : review the structural model for missing and/or unnecessary classes, attributes, operations, and relationships.
6. **Incorporate Patterns** : incorporate useful patterns into the evolving structural model. A useful pattern is one that would allow the analyst to more fully describe the underlying domain of the problem being investigated.
7. **Review the Model**: validate the structural model, including both the CRC cards and the class diagram.

6.5 Class, Package Diagrams

Stereotypes and Classes

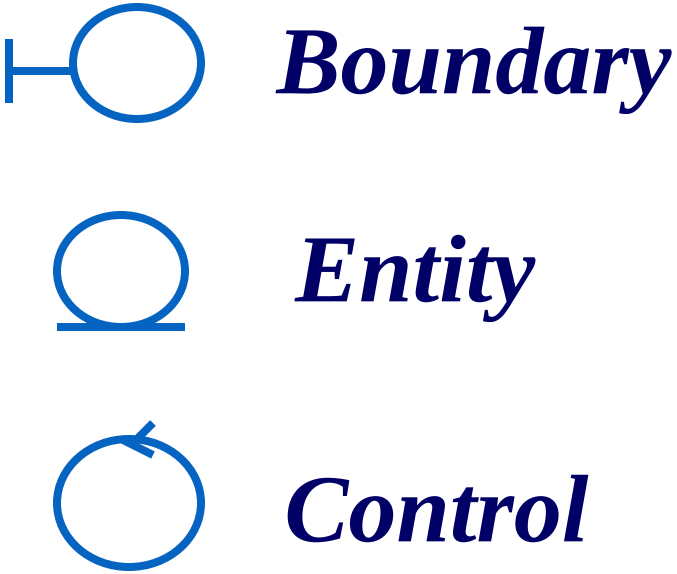
A Stereotype is a mechanism you can use to categorize your classes.

- Say you want to quickly find all of the forms in the model,
- You could create a stereotype called form, and assign all of your windows to this stereotype.
- To find your forms later, you would just need to look for the classes with that stereotype.

6.5 Class, Package Diagrams

Stereotypes and Classes

- There are *three* primary class stereotypes in UML.



Boundary

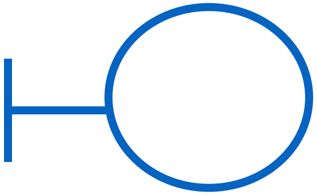
Entity

Control

6.5 Class, Package Diagrams

Stereotypes and Classes

***Boundary** Class:*

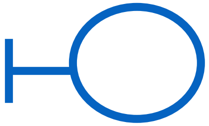


- They provide the interface to a user or another system. (ie. Interface to an actor).
- Handles communication between system surroundings and the inside of the system.
- To find the *Boundary* classes, you can examine your Use Case diagram.

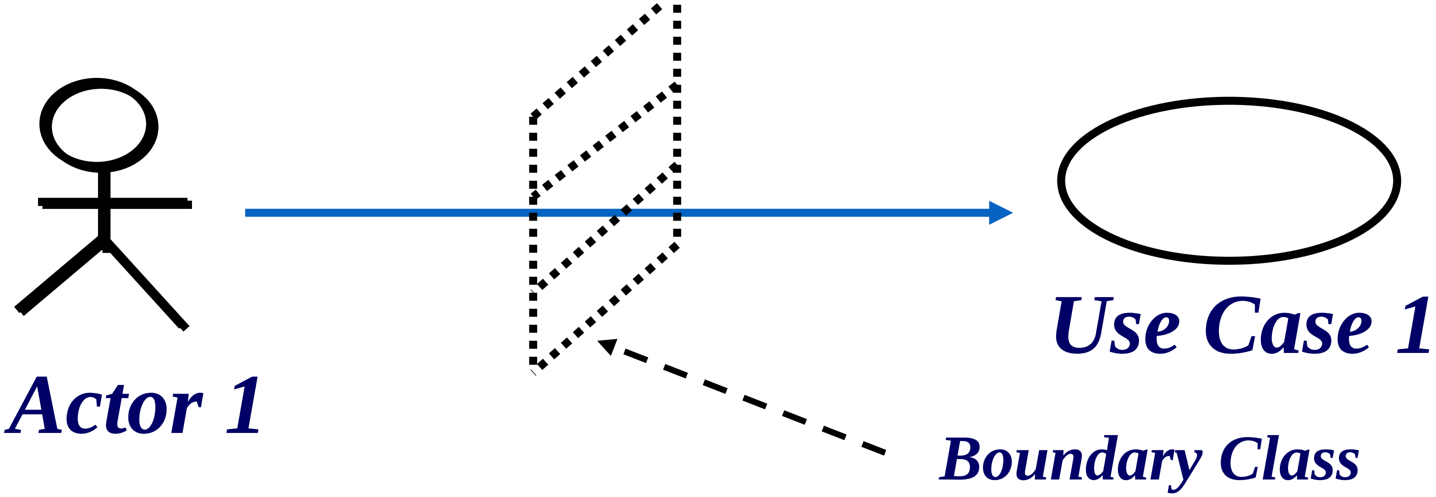
6.5 Class, Package Diagrams

Stereotypes and Classes

***Boundary** Class:*



- At a minimum there must be, one *Boundary* class for every actor-use case interaction.
- Boundary class allows actor to interact with the system.



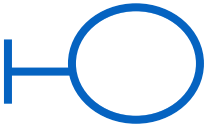
Actor 1

Use Case 1

Boundary Class

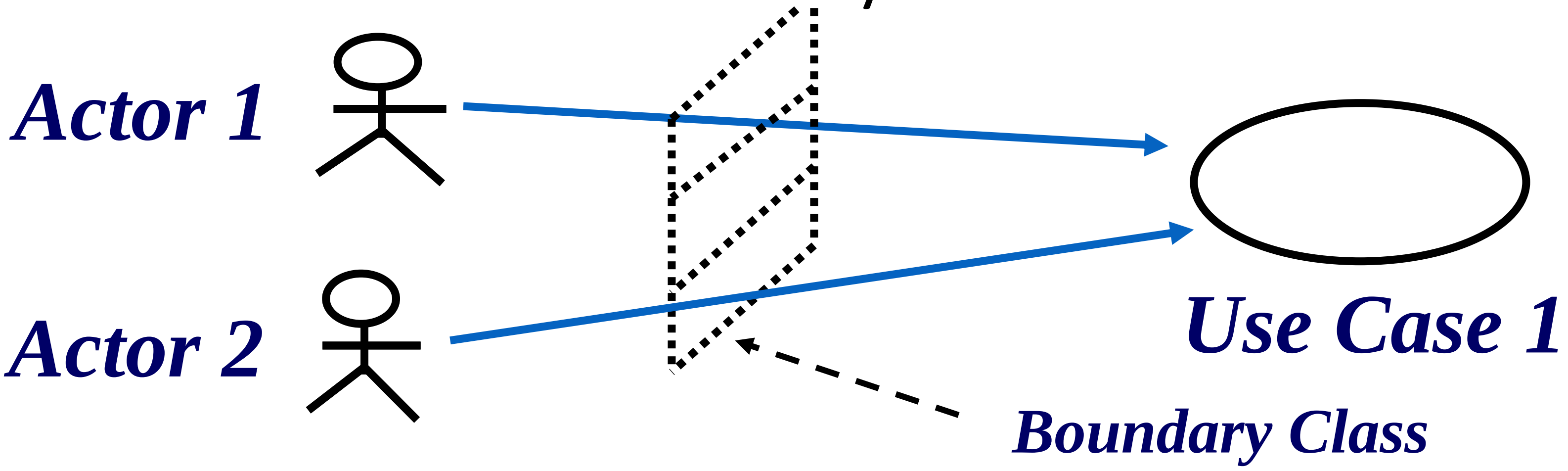
6.5 Class, Package Diagrams

Stereotypes and Classes



***Boundary** Class:*

- You do not necessarily have to create a unique *Boundary* class for every actor-use case pair.
- Two actors may initiate the same use case.
- They might both use the same Boundary class to communicate with the system.



6.5 Class, Package Diagrams

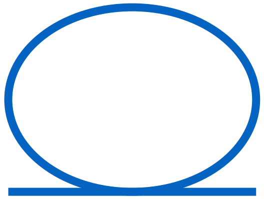
Stereotypes and Classes

These are classes that mediate between the subject (System boundary) and its environment.

- User Interface class – classes that interface between the system and humans;
- System Interface class – classes that interface with other systems;
- Device Interface class – classes that interface with external devices such as sensors;

6.5 Class, Package Diagrams

Stereotypes and Classes



Entity Class

- They are needed to perform task internal to the system. Reflect a real world entity.

Identifying Entity Classes

Noun/Verb Analysis

- Identify the nouns and noun phrases

6.5 Class, Package Diagrams

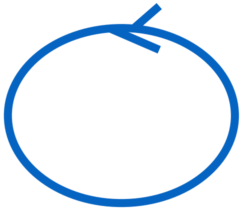
Stereotypes and Classes

Entity Class

- The initial list of nouns must be filtered because it could contain nouns that are ,
  - outside the problem domain.
  - just language expressions.
  - redundant.
  - attributes.

6.5 Class, Package Diagrams

Stereotypes and Classes



Control Class:

- Sequencing behaviour specific to one or more use cases.
- There is typically one *control* class per use case.
- Co-ordinates the events needed to realise the behaviour specified in the use case.

E.g. Running or *executing* the use case.

6.5 Class, Package Diagrams

Finding Controller Classes

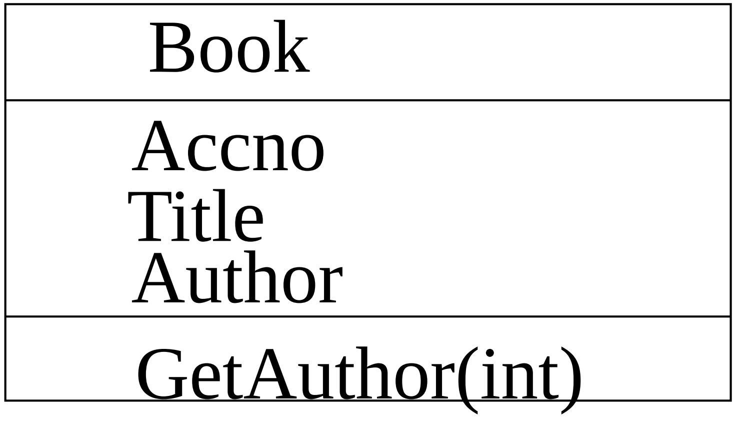
- Simple behavior can often be distributed between Boundary or Entity classes.
- Consider more complex behavior of the system as described by the use cases.
- Work out how these behavior should be partitioned among the analysis classes.
- Control classes process messages from an interface class and respond to them by sending and receiving messages from the entity classes.

6.5 Class, Package Diagrams

Class Diagrams

- They are backbone of nearly all OO Methods/Processes.
- A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them.
- It also shows the attributes and services of a class and the constraints that apply to the way objects are connected.

UML Notation for a Class



Book

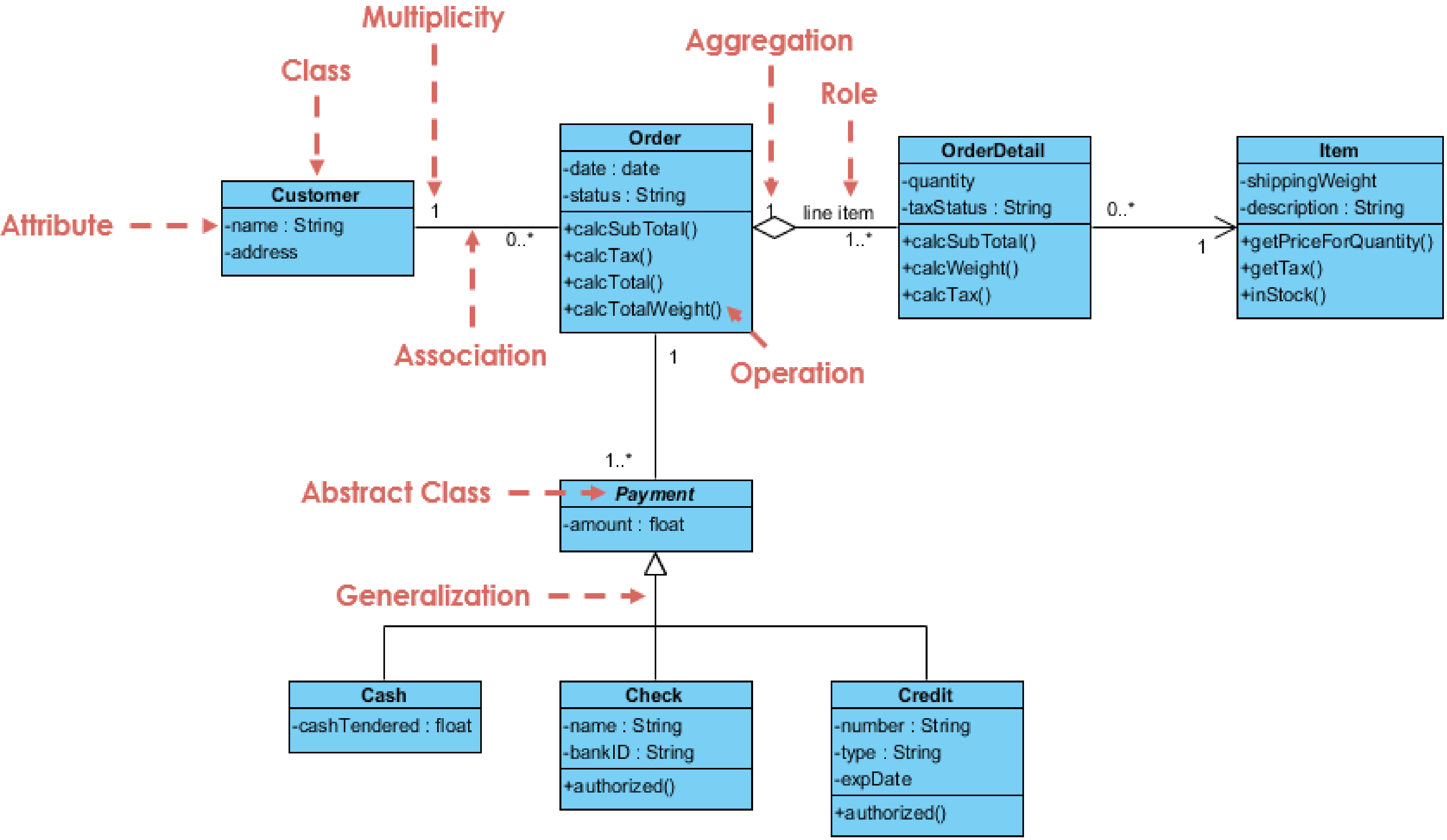
Accno
Title
Author

GetAuthor(int)

6.5 Class, Package Diagrams

Class Diagram Example

Can be a Composition-depends on the problem domain. This will be an association with latest UML versions

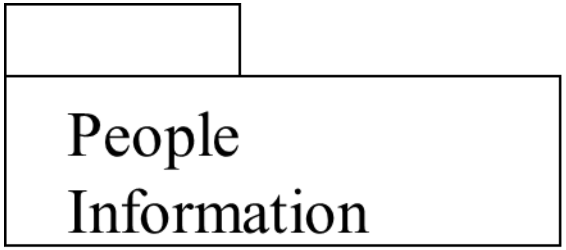


© e-Learning Centre, UCSC

6.5 Class, Package Diagrams

Package Diagrams

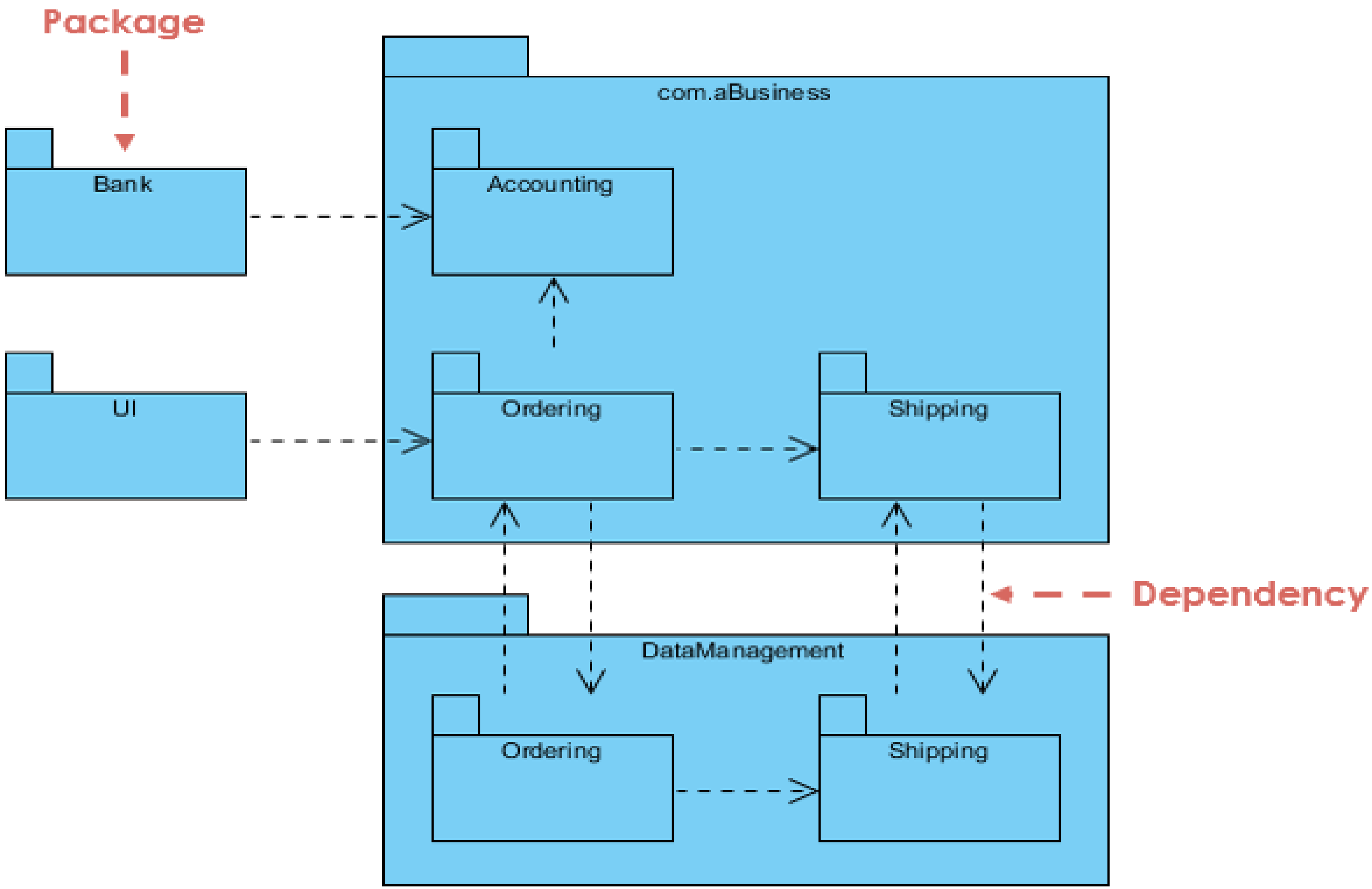
- A **Package** is a general construct that can be applied to any of the elements in UML models. A package is a collection of logically related UML elements.
- **Package diagram** is used to simplify complex class diagrams, you can group classes into packages.
- A package diagram is effectively a class diagram that only shows packages. They are used to simplify complex class diagrams,
- By grouping classes to packages we can look at the higher-level view of the model.
- Package diagrams help you to maintain control over a system's overall structure



UML Notation

6.5 Class, Package Diagrams

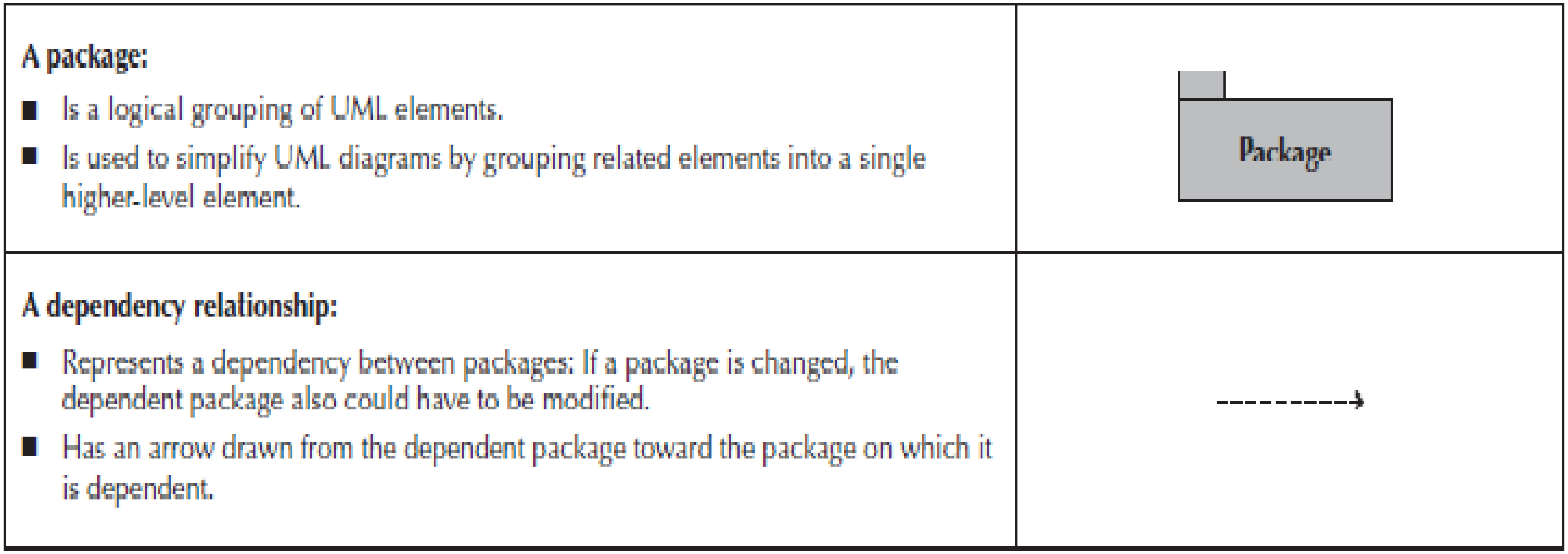
An Example of a Package Diagram



© e-Learning Centre, UCSC

6.5 Class, Package Diagrams

Package Diagram Syntax



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.	Package
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.	----------→

6.6 Creating Structural Models using CRC cards and Class Diagrams

- Creating a structural model is an incremental and iterative process whereby the analyst makes a rough cut of the model and then refines it over time.
- A use-case–driven process that can be used to create the structural model of a problem domain.
  - **Create CRC Cards :** Create the CRC cards first and then transfer the information into a class diagram later.
  - **Role-Play the CRC Cards :** Each CRC card should be assigned to an individual who will perform the operations for the class on the CRC card. Role-playing the CRC cards, requires to apply the three role-playing steps. Review Use Cases, Identify Relevant Actors and Objects, Role Play Scenarios
  - **Create Class Diagram :** Information contained on the CRC cards is transferred to the class diagrams.
  - **Review Class Diagram :** Review the structural model for missing and/or unnecessary classes, attributes, operations, and relationships.
  - **Review the Model :** Validate the structural model, including both the CRC cards and the class diagram.

6.7 Verifying and Validating the Structural Model

- Before creating behavioral models of the problem domain, we need to verify and validate the structural model.
- Walkthroughs with the power of role-playing can be used to more completely verify and validate the structural model that will underlie the business processes and functional models.
- First the analyst walks through the model, explaining each part of the model to developers and users.
- He needs to describe all the reasoning behind the decision to include each of the classes in the structural model.
- Each class should be linked back to at least one use case.
- Every CRC card should be associated with a class on the class diagram, and vice versa.

6.7 Verifying and Validating the Structural Model

- The responsibilities listed on the front of the CRC card must be included as operations in a class on a class diagram, and vice versa. Every responsibility and operation must be checked.
- Collaborators on the front of the CRC card imply some type of relationship on the back of the CRC card and some type of association that is connected to the associated class on the class diagram.
- Attributes listed on the back of the CRC card must be included as attributes in a class on a class diagram, and vice versa.
- The object type of the attributes listed on the back of the CRC card and with the attributes in the attribute list of the class on a class diagram implies an association from the class to the class of the object type.

6.7 Verifying and Validating the Structural Model

- The relationships included on the back of the CRC card must be portrayed using the appropriate notation on the class diagram.

  E.g. Generalization, composition, association, association class etc.

- Verify and validate the functional model will ensure the consistency of the structural representations such as CRC cards and class diagrams.

Summary

- During analysis, analysts create business processes and functional models to represent how the business system will behave.
- At the same time, analysts need to understand the information that is used and created by the business system such as customer information, order information in an order processing system.
- A structural model is a formal way of representing the objects that are used and created by a business system.
- Typically, structural models are shown using CRC cards, class diagrams, and, in some cases, object diagrams.
- All systems are made up of objects and classes. Conceptually, objects do not exist in isolation. System behavior is achieved through the interactions of the objects in the system.

Summary cont..

- A relationship is a semantic connection between classes. It allows one class to know about the attributes, operations of another class
- All can be classified into three basic categories of data abstraction mechanisms: Association relationships. Aggregation/Composition relationships (part of) , and Generalization relationships (kind of).
- Sometimes a class is in an association with itself. This is called reflexive relationship.
- The multiplicity is an indication of how many objects may participate in the given relationship or the allowable number of instances of the element.

Summary cont..

- Different approaches have been suggested to aid the analyst in identifying a set of candidate objects for the structural model.
- The four most common approaches to identify set of candidate objects for the structural model are Textual analysis, Brainstorming, Common object lists, and Patterns.
- The analyst performs textual analysis by reviewing the use-case diagrams and examining the text in the use-case descriptions to identify potential objects, attributes, operations, and relationships. The nouns in the use case suggest possible classes, and the verbs suggest possible operations.
- Brainstorming is a process that a set of individuals suggest potential classes that could be useful for the problem under consideration.

Summary cont..

- A common object list is simply a list of objects common to the business domain of the system.
- A pattern is simply a useful group of collaborating classes that provide a solution to a commonly occurring problem.
- CRC cards are used to document the responsibilities and collaborations of a class.
- In Role playing each CRC card should be assigned to an individual who will perform the operations for the class on the CRC card. It is used to uncover missing properties by executing the different scenarios associated with the use cases.

Summary cont..

- A Stereotype is a mechanism you can use to categorize your classes. Three stereotypes used for classes are Boundary, Entity and Control.
- Boundary class provide the interface to a user or another system.
- Entity classes are problem domain classes.
- Control Classes Co-ordinates the events needed to realize the behaviour specified in the use case.
- A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them.

Summary cont..

- Creating a structural model is an incremental and iterative process whereby the analyst makes a rough cut of the model and then refines it over time.
- Before we move on to creating behavioral models of the problem domain, we need to verify and validate the structural model.
- A use-case–driven process can be used to create the structural model of a problem domain. Steps involve , Create CRC Cards, Role-Play the CRC Cards, Create Class Diagram, Review Class Diagram , Validate the structural models, including both the CRC cards and the class diagram.
- Walkthroughs with the power of role-playing can be used as a way to more completely verify and validate the structural model that will underlie the business processes and functional models.
- Verify and validate the functional model will ensure the consistency of the structural representations such as CRC cards and class diagrams

***