

Cs 211: System Analysis and Design



LECTURE 6: STRUCTURING SYSTEM REQUIREMENTS: OOAD
MS MURO

OOAD



- Object-oriented (O-O) analysis and design is an approach that is intended to facilitate the development of systems that must change rapidly in response to dynamic business environments
- Object-oriented approaches use the industry standard for modeling object-oriented systems, called the unified modeling language (UML)

OOAD



- Object-oriented programming differs from traditional procedural programming by examining objects that are part of a system.
- Each object is a computer representation of some actual thing or event.
- Objects may be customers, items, orders, and so on
- Objects are represented by and grouped into classes that are optimal for reuse and maintainability.
- A class defines the set of shared attributes and behaviors found in each object in the class.

OOAD concepts



Objects

- Objects are persons, places, or things that are relevant to the system we are analyzing.
- Object-oriented systems describe entities as objects.
- Objects may also be GUI displays or text areas on the display.

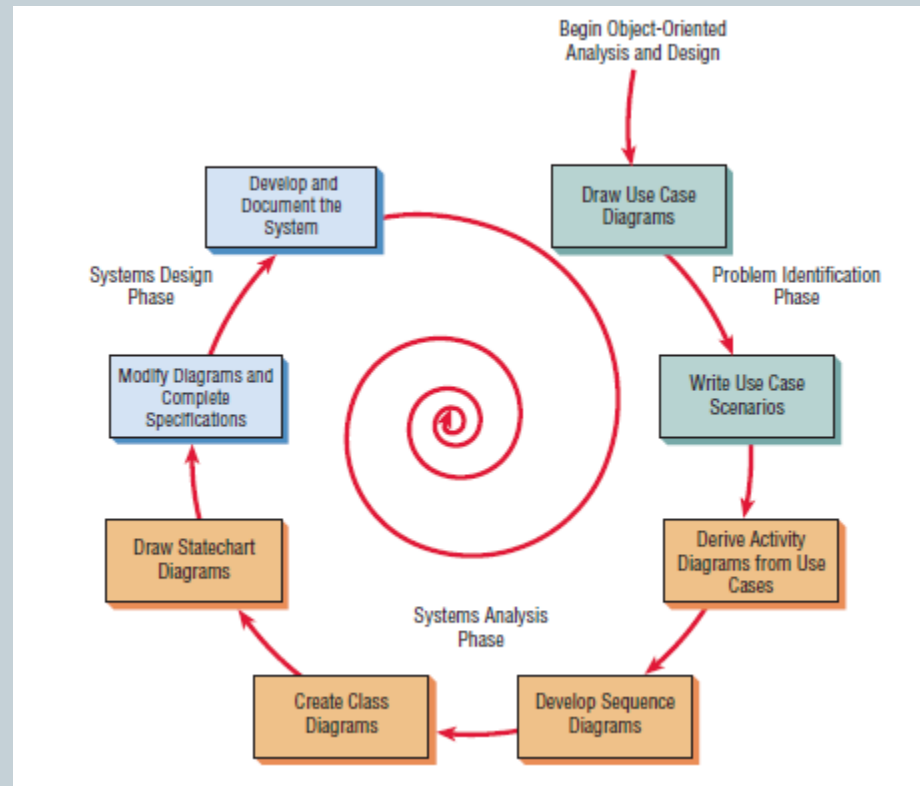
OOAD concepts



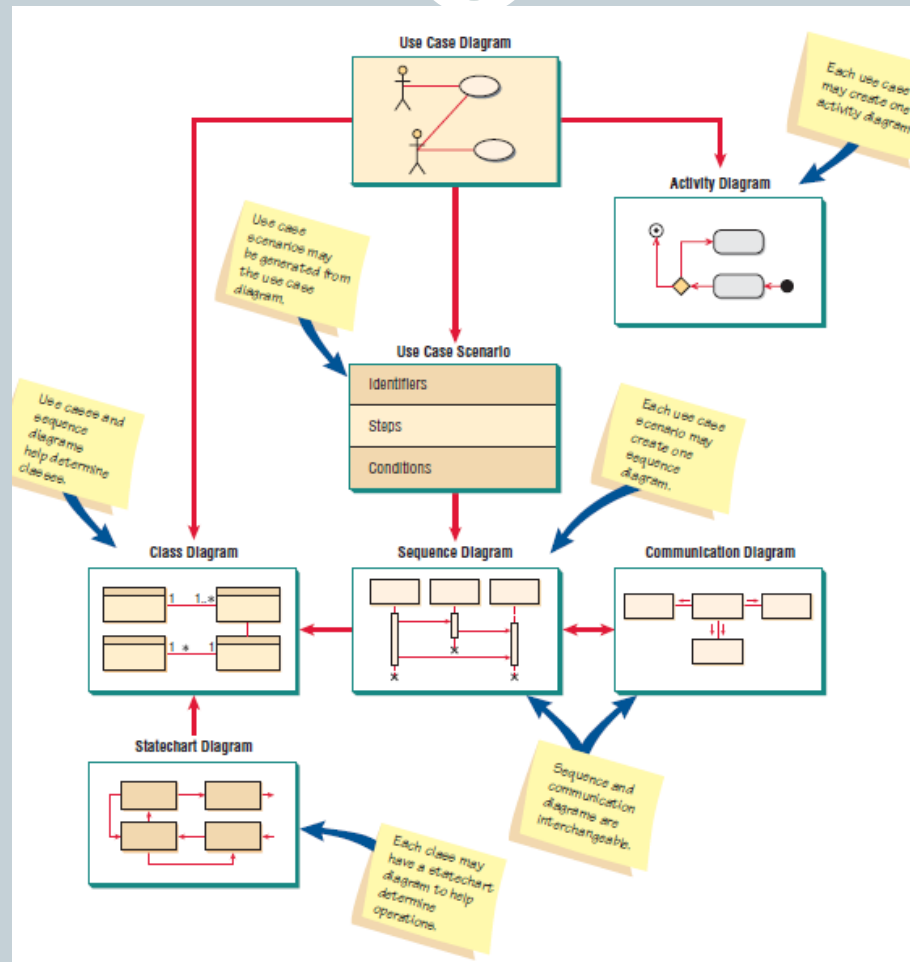
Classes

- Objects are typically part of a group of similar items called classes
- The idea behind classes is to have a reference point and describe a specific object in terms of its similarities to or differences from members of its own class
- Reusability: means you can be more efficient, because you do not have to start at the beginning to describe an object every time it is needed for software development.
- Inheritance

Steps in the UML development process



UML diagrams



Use case modeling



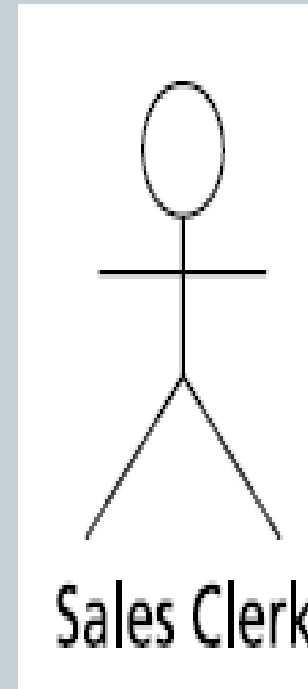
- A use case model shows a view of the system from the user perspective, thus describing what a system does without describing how the system does it.
- A use case provides developers with a view of what the users want.
- It is free of technical or implementation details.
- The use case model is based on the interactions and relationships of individual use cases.

Use case symbols



Actors

- are similar to external entities; they exist outside of the system.
- The term *actor* refers to a *particular* role of a user of the system.
- For example, an actor may be an employee, but also may be a customer at the company store.



Use case symbols

Use case


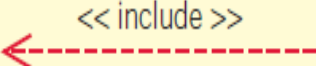
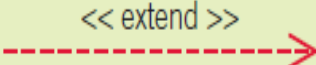

- Describes a set of activities of a system from the point of view of its actors, which lead to a **perceptible outcome for the actors**



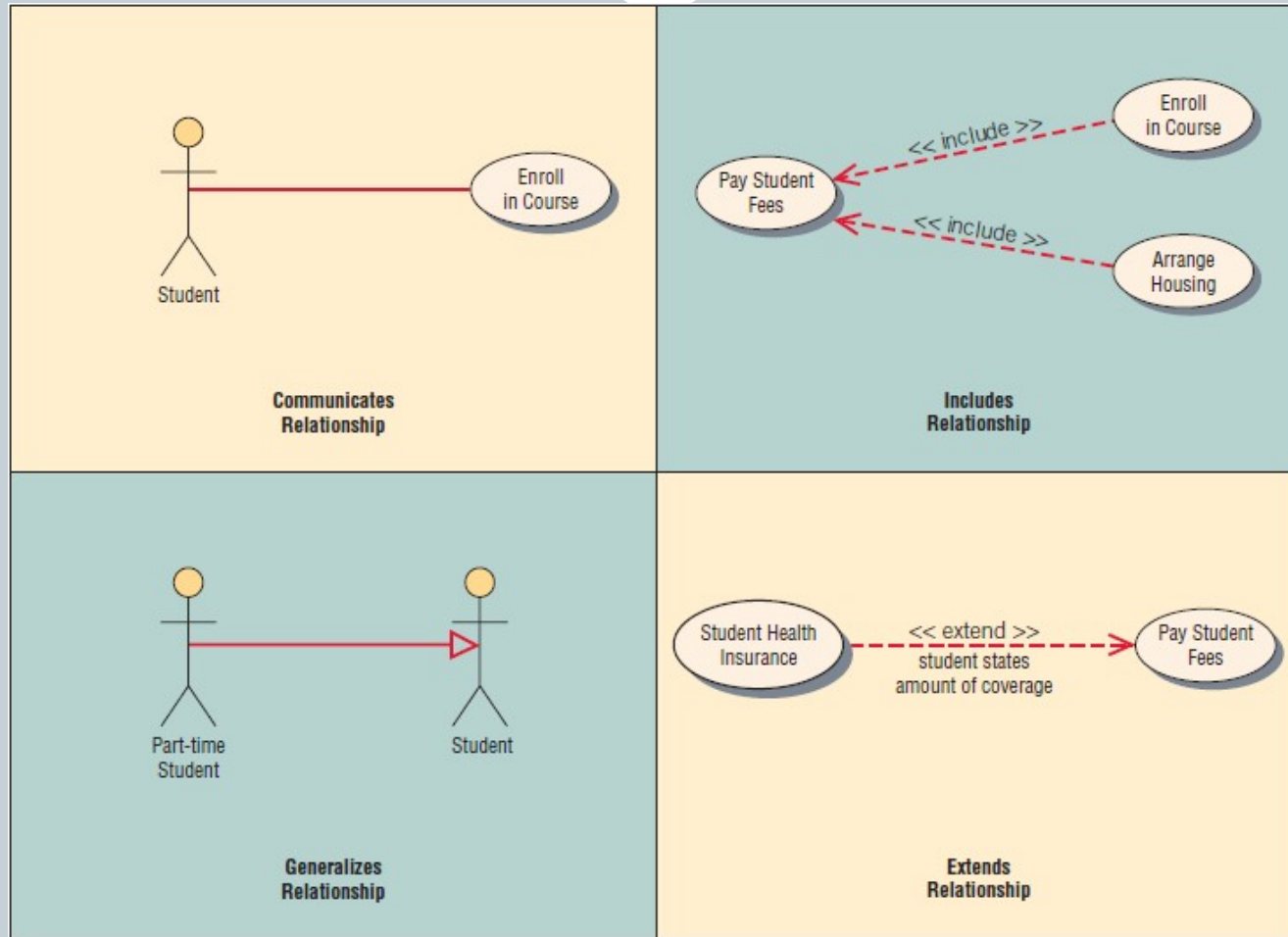
Use case symbols



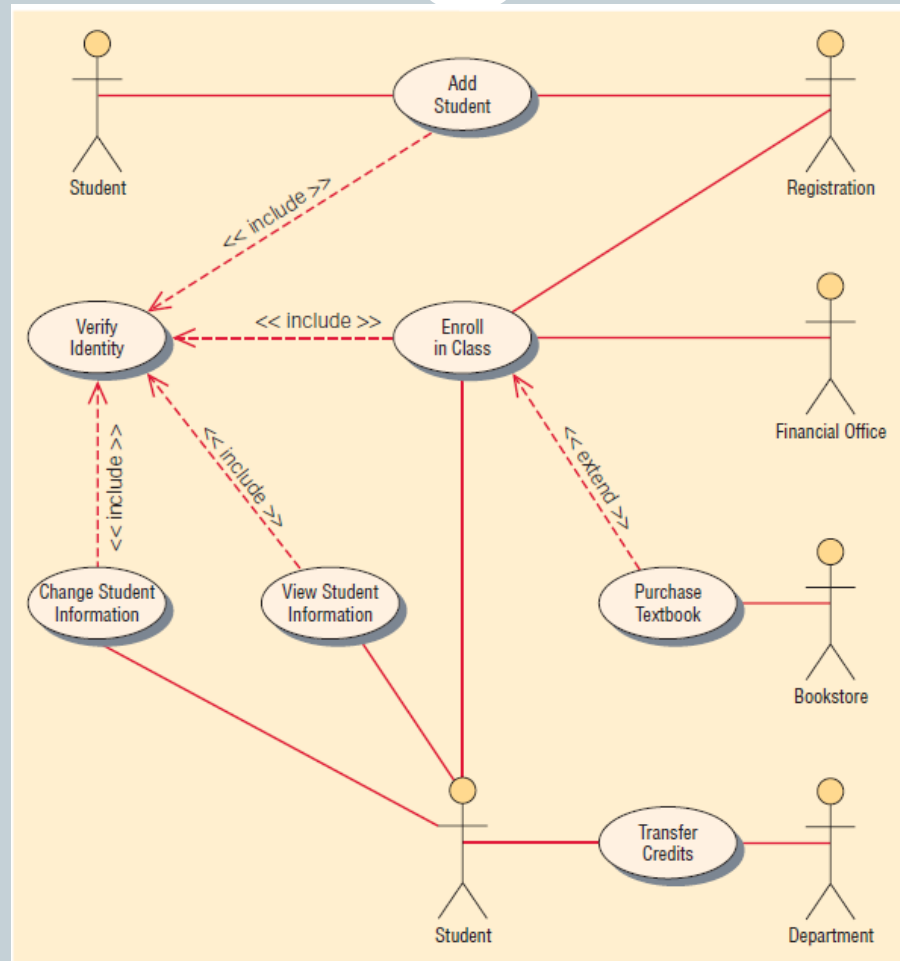
- Use case relationships

Relationship	Symbol	Meaning
Communicates		An actor is connected to a use case using a line with no arrowheads.
Includes		A use case contains a behavior that is common to more than one other use case. The arrow points to the common use case.
Extends		A different use case handles exceptions from the basic use case. The arrow points from the extended to the basic use case.
Generalizes		One UML “thing” is more general than another “thing.” The arrow points to the general “thing.”

Use case relationships



Example of the use case diagram



scenario



A library loans system allows members of the library to borrow books for a four week period. When a member wishes to borrow a book, it is taken to the librarian who records the loan. On returning the book, the member takes it to the librarian who records that it has been returned. There are occasions where a member would like to borrow a book that is currently on loan to another member. In this case the book can be reserved, so that when it is returned the member is informed and can come to the library and collect the book. In order for the system to function correctly there needs to be up-to-date records of members and books.

A use case scenario



- This scenario is a verbal articulation of exceptions to the main behavior described by the primary use case.
- Describing individual use cases in detailed textual format.
- The most important piece of information included in a written use case is the flow of events that describes the main success scenario of actions required for successful completion of the use case

A use case scenario



Use Case	<i>Use case identifier and reference number and modification history</i>
Description	<i>Goal to be achieved by use case and sources for requirement</i>
Actors	<i>List of actors involved in use case</i>
Assumptions	<i>Conditions that must be true for use case to terminate successfully</i>
Steps	<i>Interactions between actors and system that are necessary to achieve goal</i>
Variations (optional)	<i>Any variations in the steps of a use case</i>
Non-Functional (optional)	<i>List of non-functional requirements that the use case must meet.</i>
Issues	<i>List of issues that remain to be resolved</i>

A use case scenario



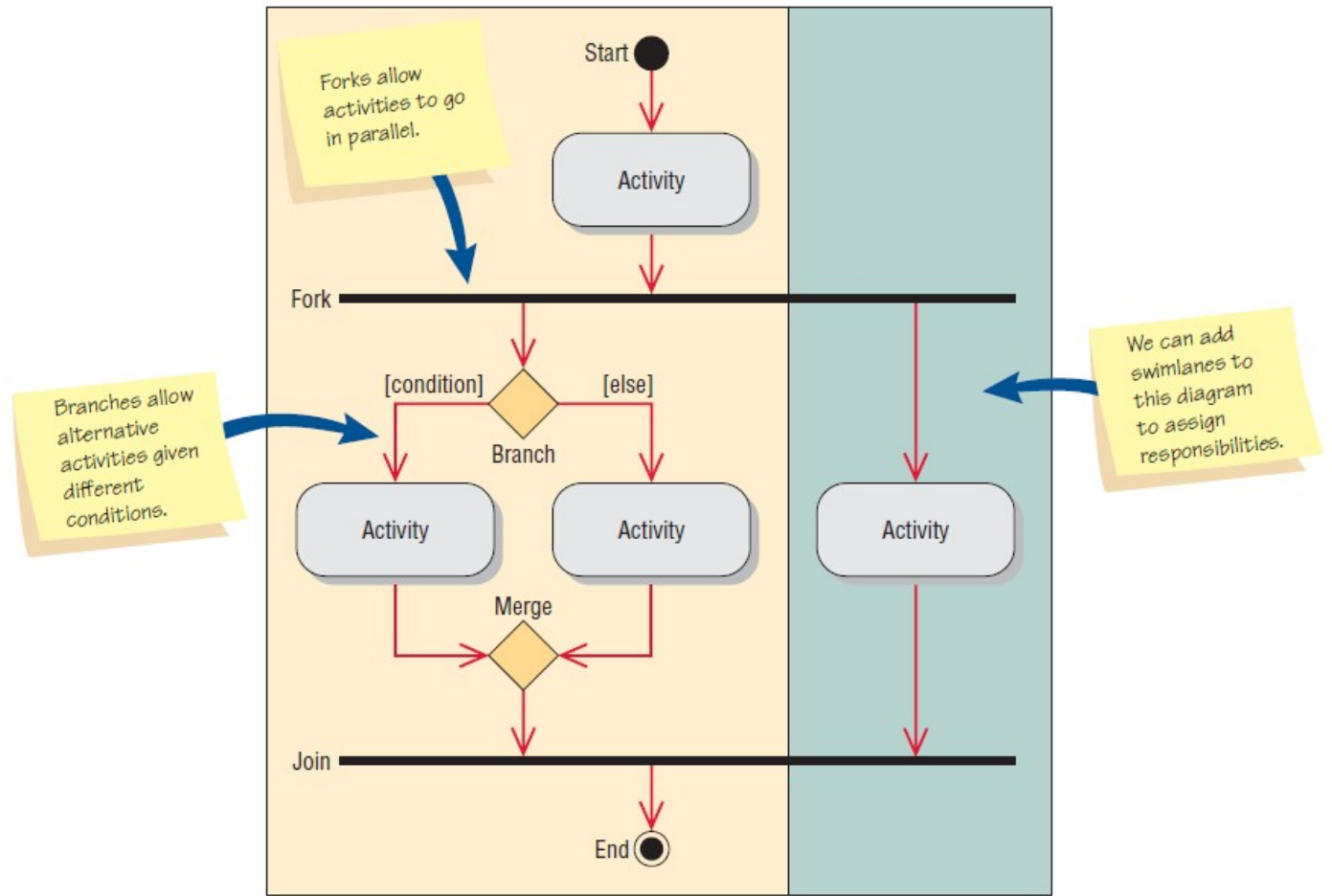
Use Case 1	Login
Description	The system should be able to control different access levels by capturing username and password of the user and granting a user appropriate access and privileges to resources.
Actors	Minimal user, depositor, Editor, and RIMS administrator
Assumptions	User has been registered with the system.
Scenarios	<ol style="list-style-type: none">1) User clicks login button2) System displays login screen3) User provides username and password, and clicks login button4) System verify the username and password5) System grants the appropriate access and privileges to its resources.6) Main user area screen is presented.
Extensions	<ol style="list-style-type: none">3a) Unable to login due to invalid username or password, notify the user including error message to re-login3b) Unable to login if account does not exist; notify the use to contact the system administrator for new account creation

Activity diagram

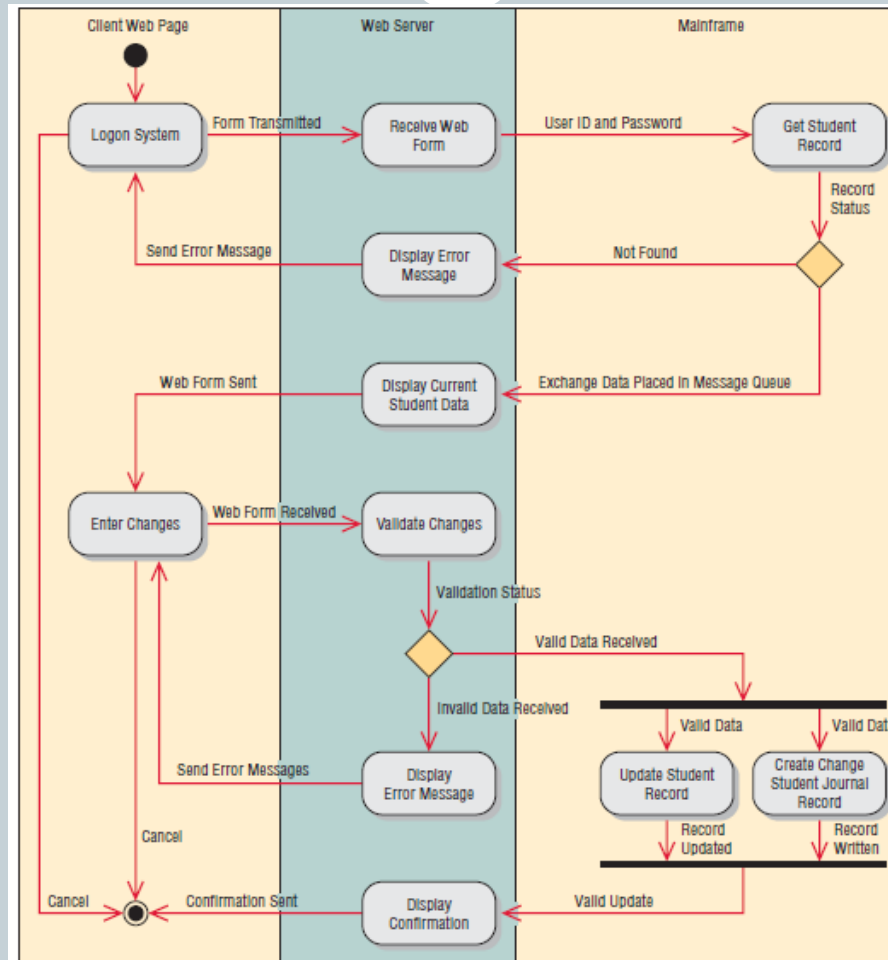


- Activity diagrams show the sequence of activities in a process, including sequential and parallel activities, and decisions that are made.
- In cases in which use cases are present, activity diagrams can model a specific use case at a more detailed level.
- However, activity diagrams can be used independently of use cases for modeling a business-level function.
- Logic Modeling

Activity diagram



Activity diagram for change student information use case



Behavioral modeling



- The key purpose of behavioral modeling is to describe how objects collaborate in each of the use case.
- The behavioral model show the internal view of the business process that the use case describe.
- *Sequence diagram and communication diagram* shows interaction that take place between object in the use case.
- *State diagram shows* how objects in the system change over the time
- Logic Modeling

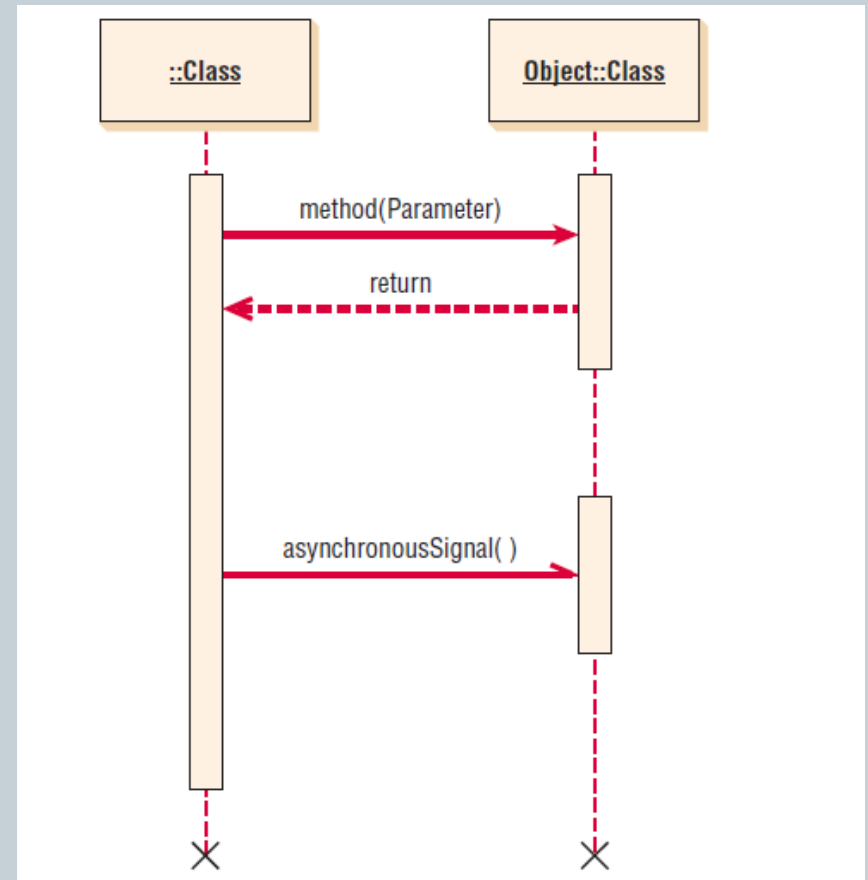
Sequence diagrams



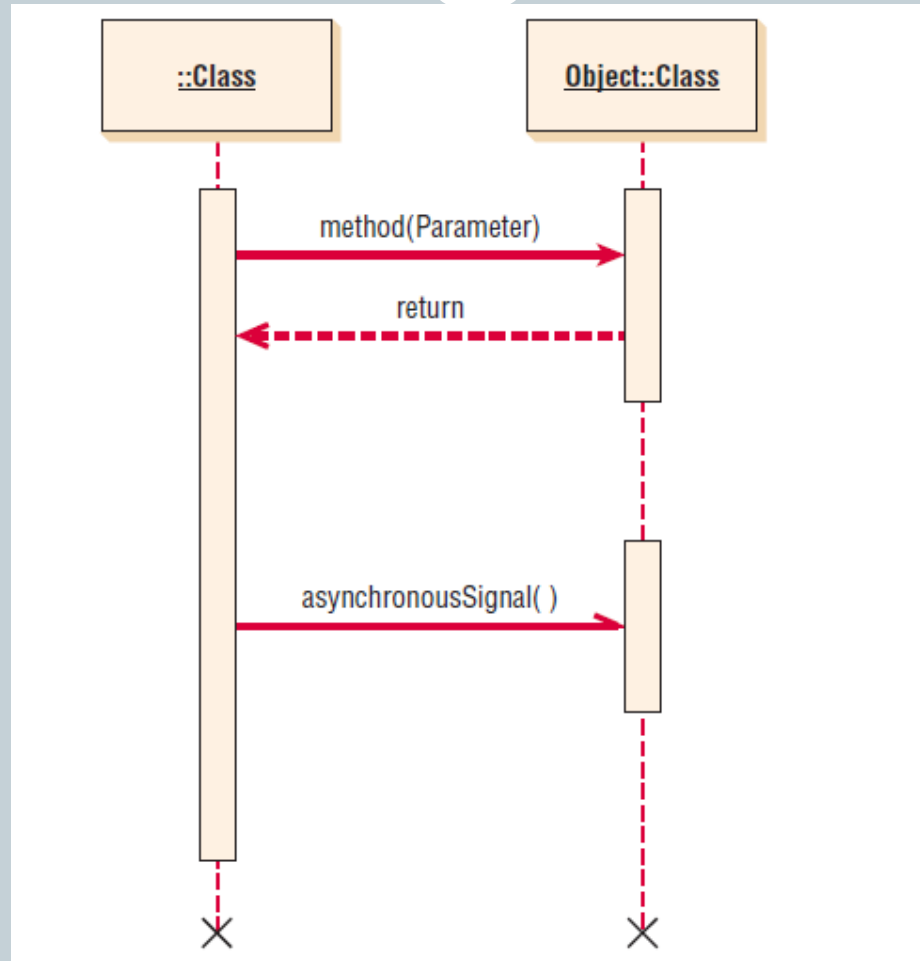
- Show the objects that collaborate in a single use case and the message that passes between them. Sequence diagrams emphasize **the time ordering of messages**.
- **objectName:** A name with a colon after it represents an object.
- **:class A** colon with a name after it represents a class.
- **objectName: class A** name, followed by a colon and another name, represents an object in a class

Sequence diagrams

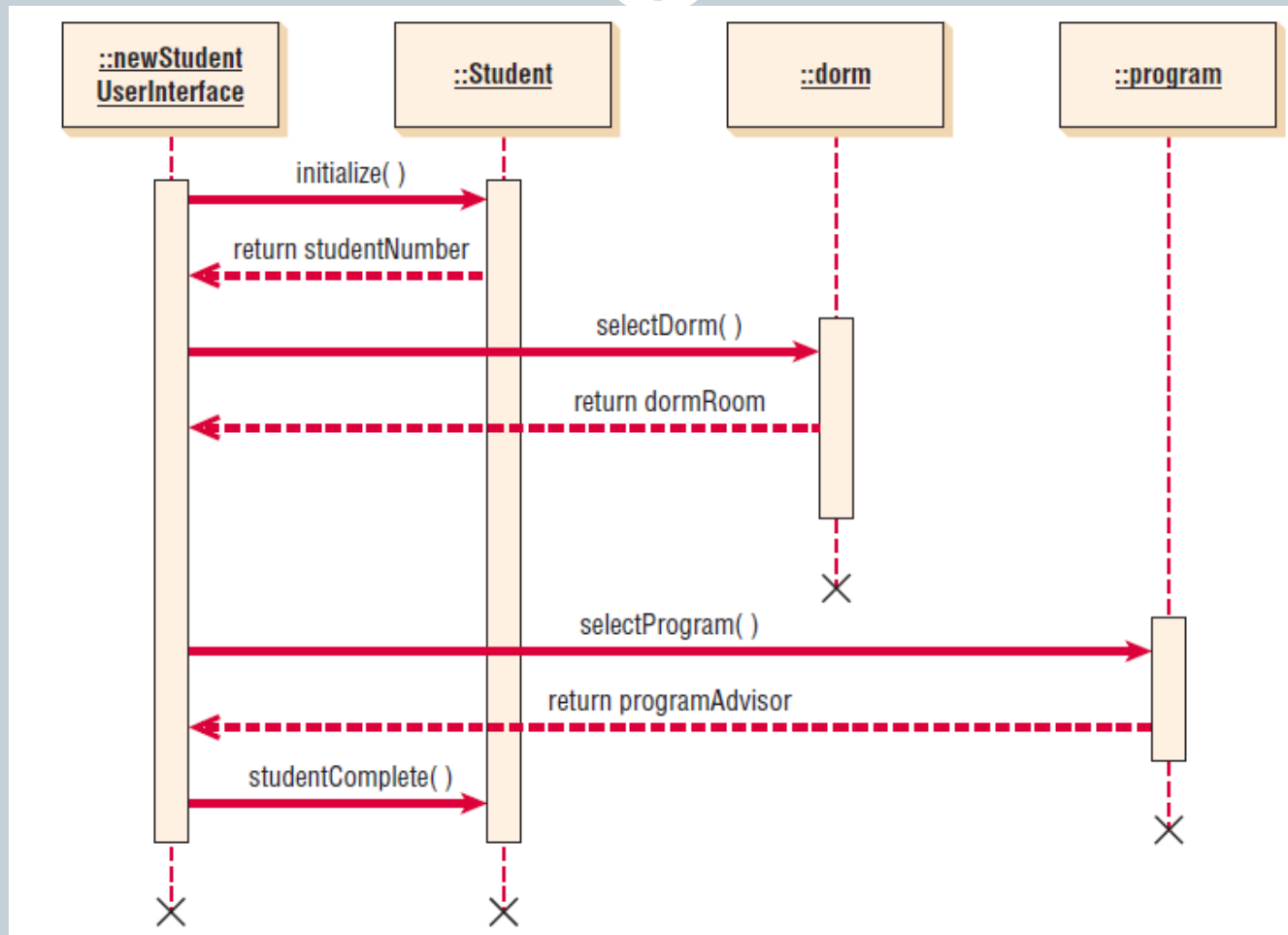
- 1 **Actor**—the initiating actor of the use case is shown with the use case actor symbol.
- 2 **System**—the box indicates the system as a “black box” or as a whole. The colon (:) is standard sequence diagram notation to indicate a running “instance” of the system.
- 3 **Lifelines**—the dashed vertical lines extending downward from the actor and system symbols, which indicate the life of the sequence.
- 4 **Activation bars**—the bars that are set over the lifelines indicate the period of time when the participant is active in the interaction. Some methodologists leave them off the system sequence diagram, but we have included them to be consistent with the full sequence diagram.
- 5 **Input messages**—horizontal arrows from the actor to the system indicate the message inputs. The UML convention for messages is to begin the first word with a lowercase letter and append additional words with an initial uppercase letter and no space. In parentheses include any parameters that you know at this point, following the same naming convention and separating individual



Sequence diagrams



A sequence diagram for student admission use case



Class diagrams



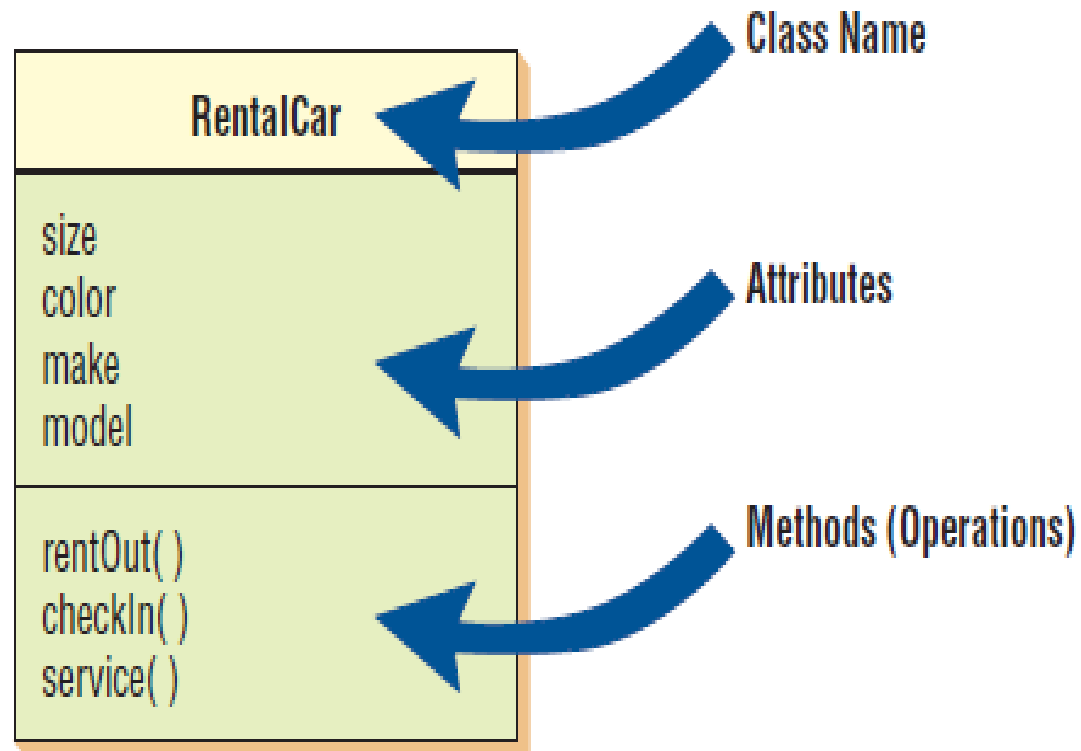
- Object-oriented methodologies work to discover classes, attributes, methods, and relationships between classes.
- Because programming occurs at the class level, defining classes is one of the most important object-oriented analysis tasks.
- Class diagrams show the static features of the system and do not represent any particular processing.
- A class diagram also shows the nature of the relationships between classes.
- Data modeling

Class diagram



- Classes are represented by a rectangle on a class diagram.
- The rectangle include only the class name, **attributes and methods**.
- Attributes are what the class knows about characteristics of the objects, and methods (also called operations) are what the class knows about how to do things.
- Methods are small sections of code that work with the attributes.

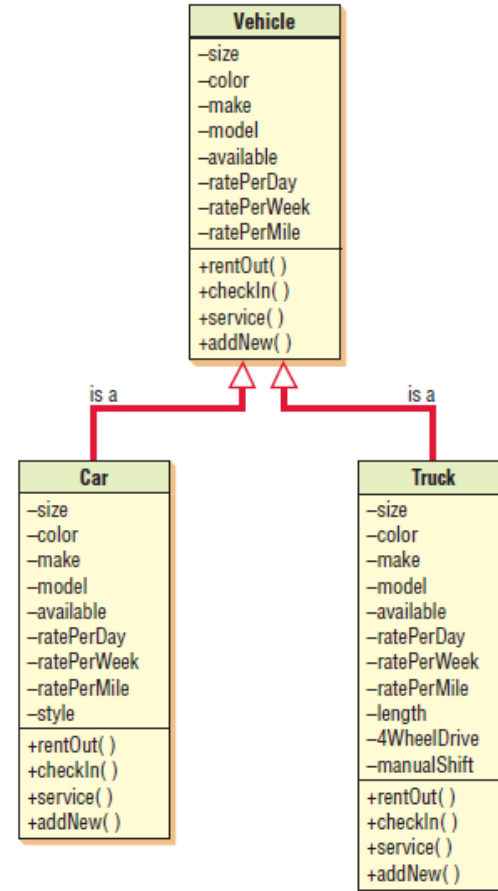
A class



Class relationships

Generalization

- Allow the analyst to create classes that inherit attributes and operation from other classes.
- It is a *kind-of* relationship

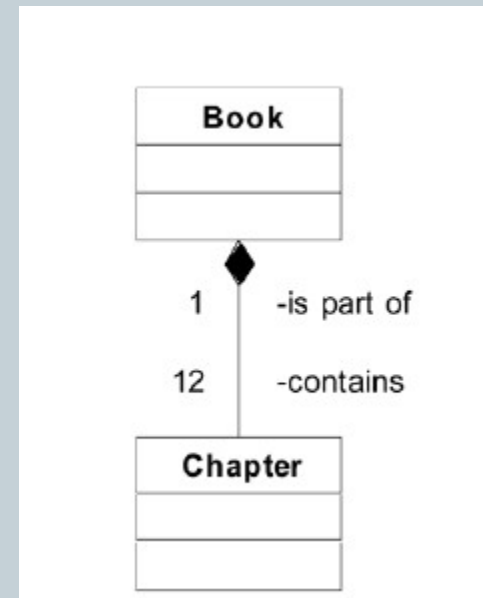


Class relationships

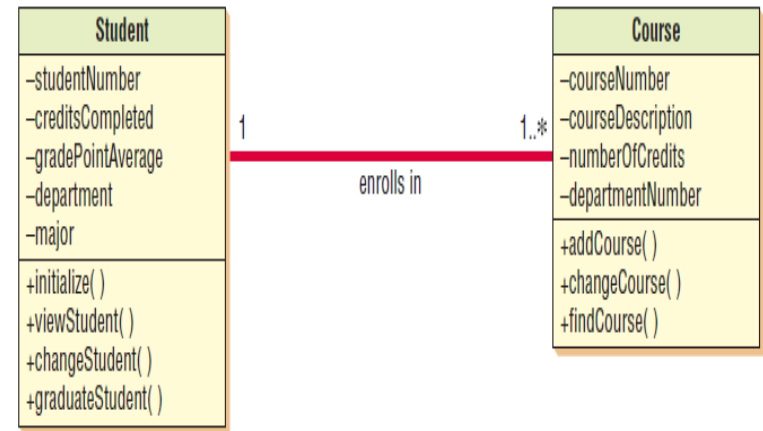
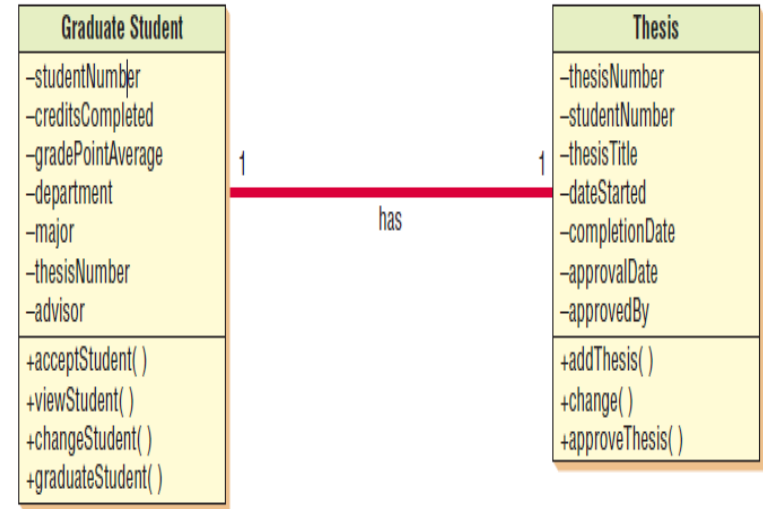
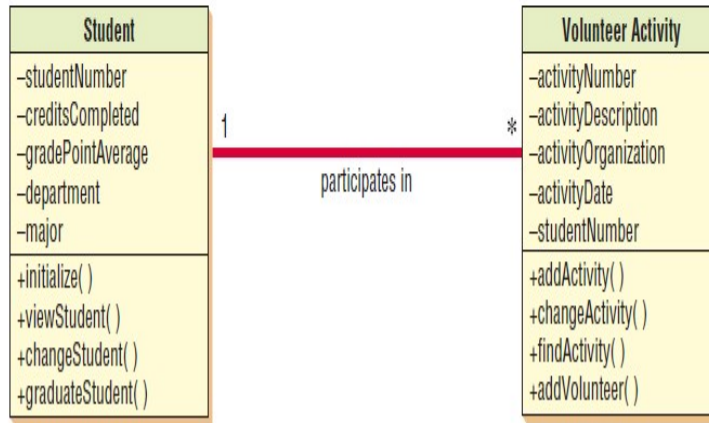
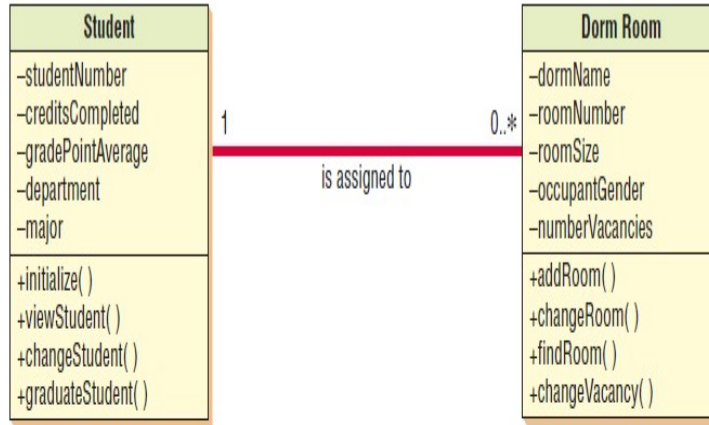


Aggregation

- *Is-a-part-of* or *has parts* relationships
- Aggregations are bidirectional relationship and has reverse side to them known decomposition.
- Decomposition is the process of discovering the constituent parts of the class



Class relationships: Association

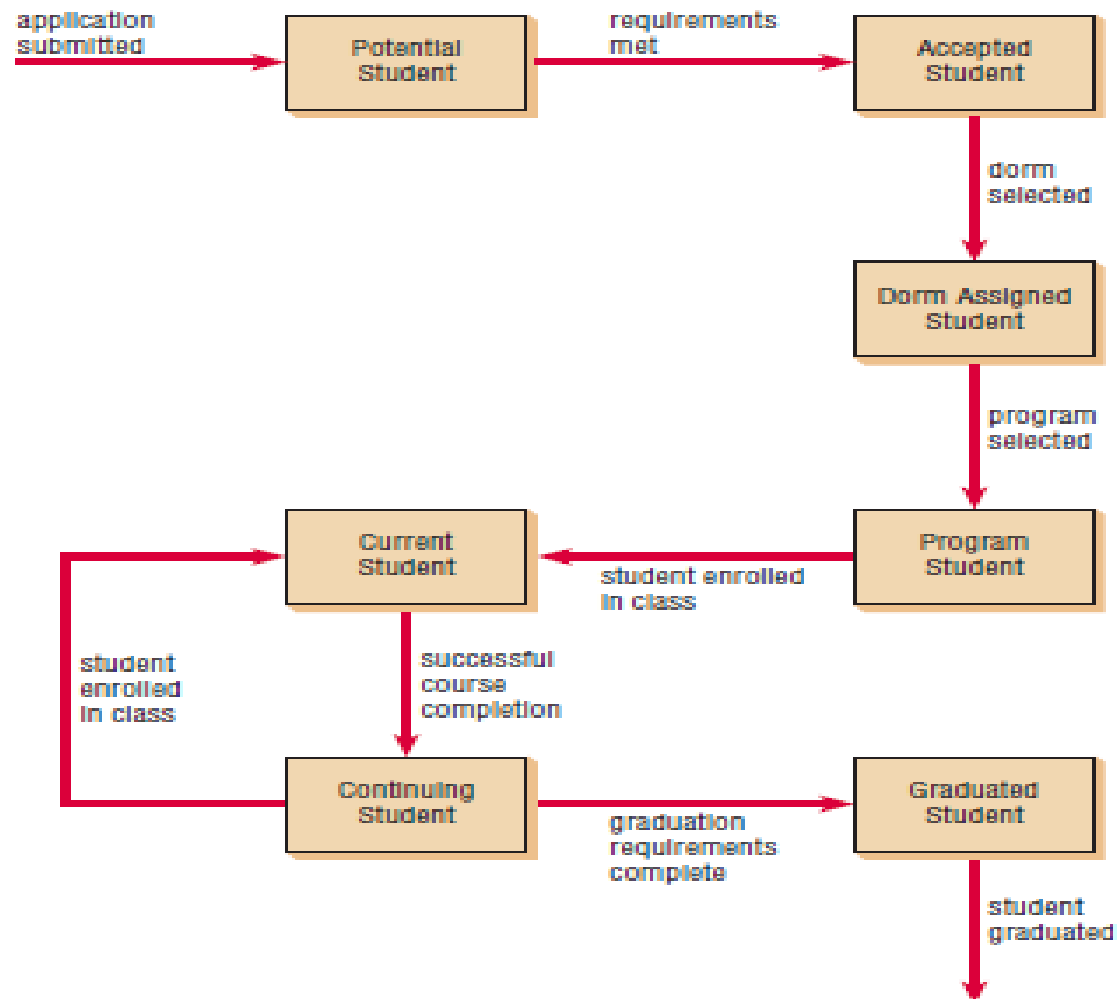


State diagrams

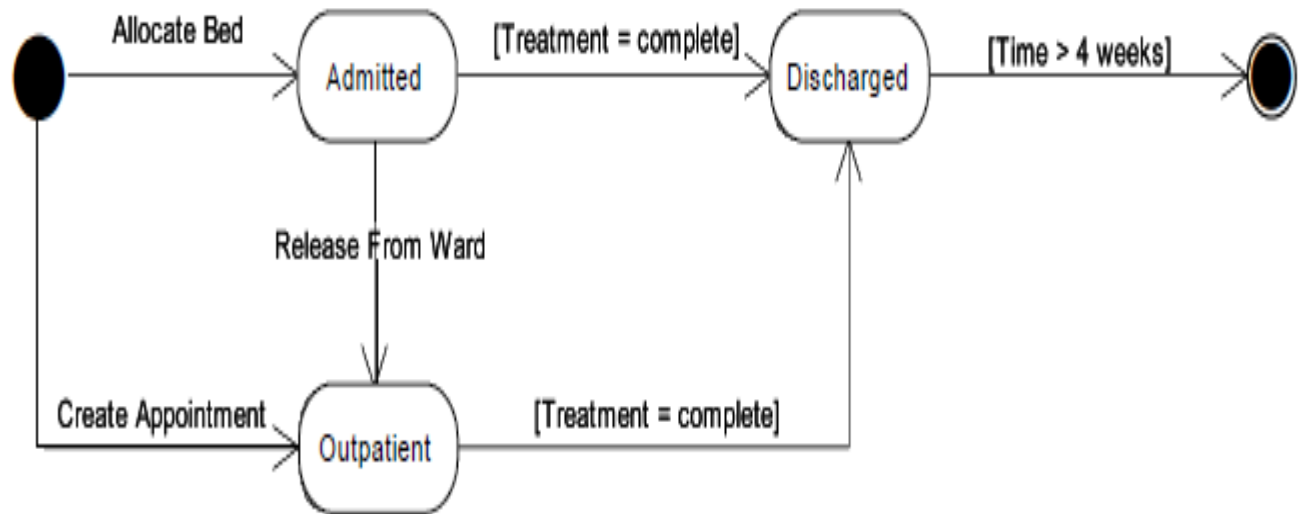


- It shows how the state of the single object changes through out its life as it respond to event.
- Not all object need to be depicted by state diagrams but they are useful for more complex objects that change state as the system operate.
- This diagram should be created when it is important to understand the dynamic change of the state for a single class and how they evolve over a time
- A state machine is not therefore related to a particular use-case but to a particular class
- Logic Modeling

Student enrolling at a university



State diagram example conti....



State diagrams



- State-chart diagrams are not created for all classes. They are created when:
 - I. A class has a complex life cycle.
 - II. An instance of a class may update its attributes in a number of ways through the life cycle.
 - III. A class has an operational life cycle.
 - IV. Two classes depend on each other.
 - V. The object's current behavior depends on what happened previously.