Department Of Computer Science

بسم الله الرحمن الرحيم

Lecture 08

Systems, Models and System Modeling

Topics covered

- Context Models
- Process Models
- Interaction Models
- Structural Models
- Behavioral models
- Data models

System

- **A system is an organized set of communicating parts**
  - A car, composed of four wheels, a chassis, a body, and an engine, is designed to transport people
  - A payroll system, composed of a computer, printers, software, and the payroll staff, is designed to issue salary checks for employees of a company
- **Parts of a system can in turn be considered as simpler systems called subsystems**

Model

- **A model is a simplification of reality**
- Modeling means constructing an abstraction of a system that focuses on interesting aspects and ignores irrelevant details
- We build models so that we can better understand the system we are developing.
- We build models of complex system because we cannot comprehend such a system in its entirety.

System Modeling

- **System Modelling is the process of developing abstract models of a system each representing a different view or perspective of that system**
- You may develop models of both the existing system and the system to be developed
  - Models of the existing system are used during requirements engineering
    - Help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses
  - Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders

System Perspective

- You may develop different models to represent the system from different perspectives
  - An external perspective, where you model the context or environment of the system.
  - An interaction perspective where you model the interactions between a system and its environment or between the components of a system.
  - A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
  - A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events

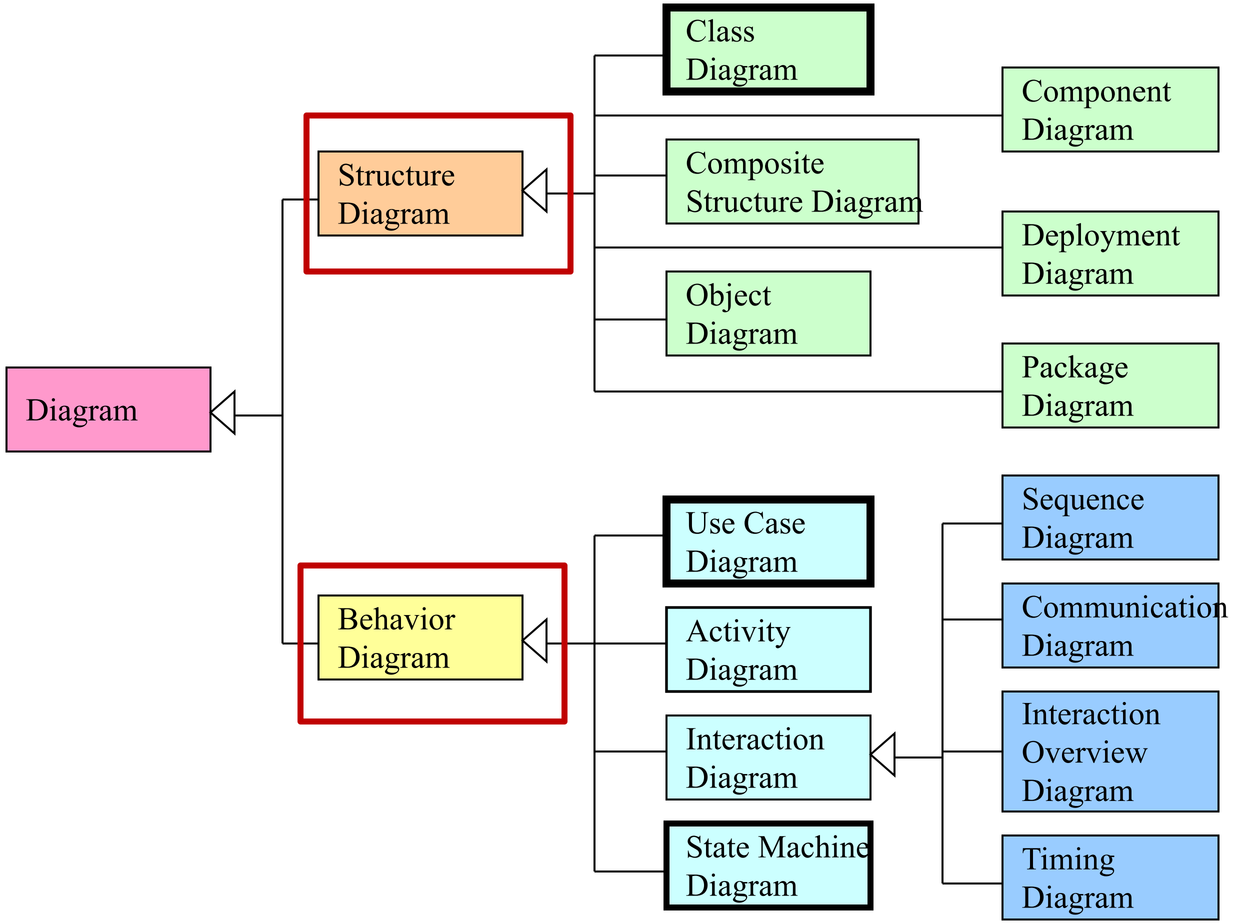
Unified Modeling Language

- UML is a standard language for modeling software systems
  - Serves as a bridge between the requirements specification and the implementation.
  - Provides a means to specify and document the design of a software system.
  - UML is process and programming language independent.
  - UML is particularly suited to object-oriented program development.

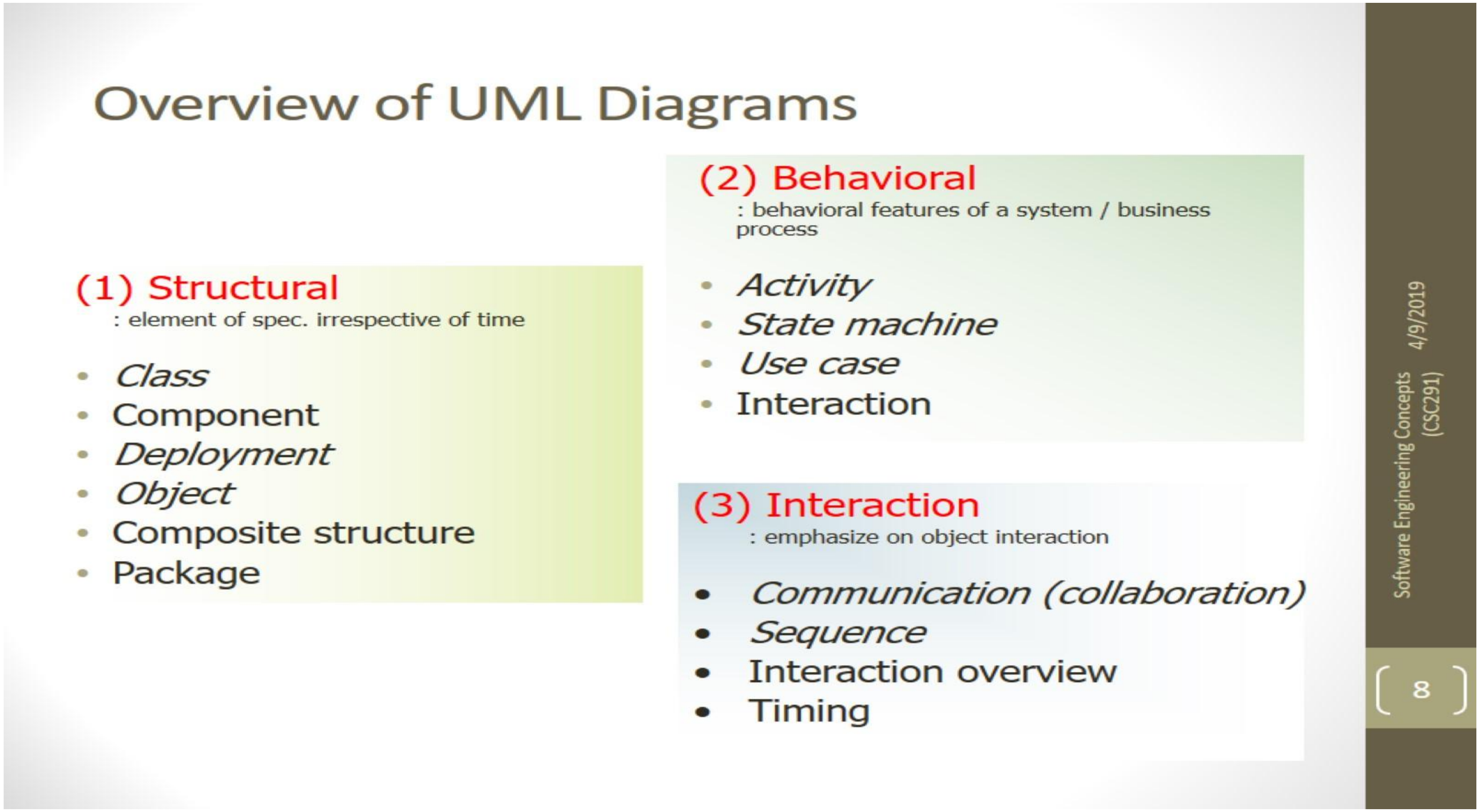
Types of UML Diagrams

- **Use case diagrams**, which show the interactions between a system and its environment.
- **Sequence diagrams**, which show interactions between actors and the system and between system components.
- **Activity diagrams**, which show the activities involved in a process or in data processing .
- **Class diagrams**, which show the object classes in the system and the associations between these classes.
- **State diagrams**, which show how the system reacts to internal and external events

Classification of Diagram Types



Classification of Diagram Types



Overview of UML Diagrams

(1) Structural
: element of spec. irrespective of time

- *Class*
- Component
- *Deployment*
- *Object*
- Composite structure
- Package

(2) Behavioral
: behavioral features of a system / business process

- *Activity*
- *State machine*
- *Use case*
- Interaction

(3) Interaction
: emphasize on object interaction

- *Communication (collaboration)*
- *Sequence*
- Interaction overview
- Timing

Software Engineering Concepts (CSC291) 4/9/2019

8

Use of Graphical Models

- Three ways in which graphical models are common
  - **As a means of** facilitating discussion about an existing or proposed system.
    - Incomplete and incorrect models are OK as their role is to support discussion.
  - **As a way of** documenting an existing system.
    - Models should be an accurate representation of the system but need not be complete.
  - **As a detailed system desc**ription that can be used to generate a system implementation.
    - Models have to be both correct and complete.

Context Models

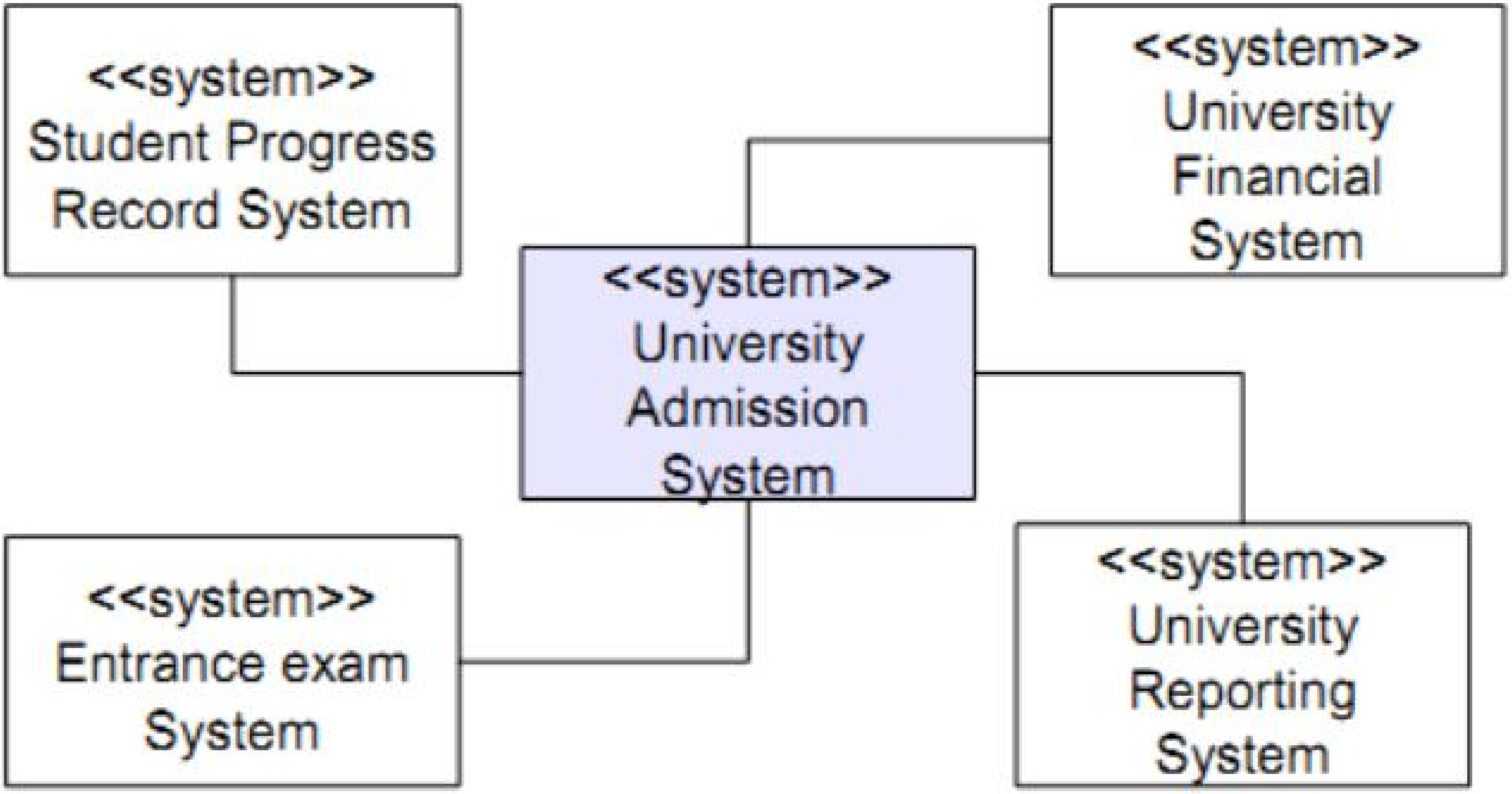
Context Models

- **Context models are used** to illustrate the operational context of a system - they show what lies outside the system boundaries.
- **Social and organizational c**oncerns may affect the decision on where to position system boundaries.
- **Architectural models** show the system and its relationship with other systems.

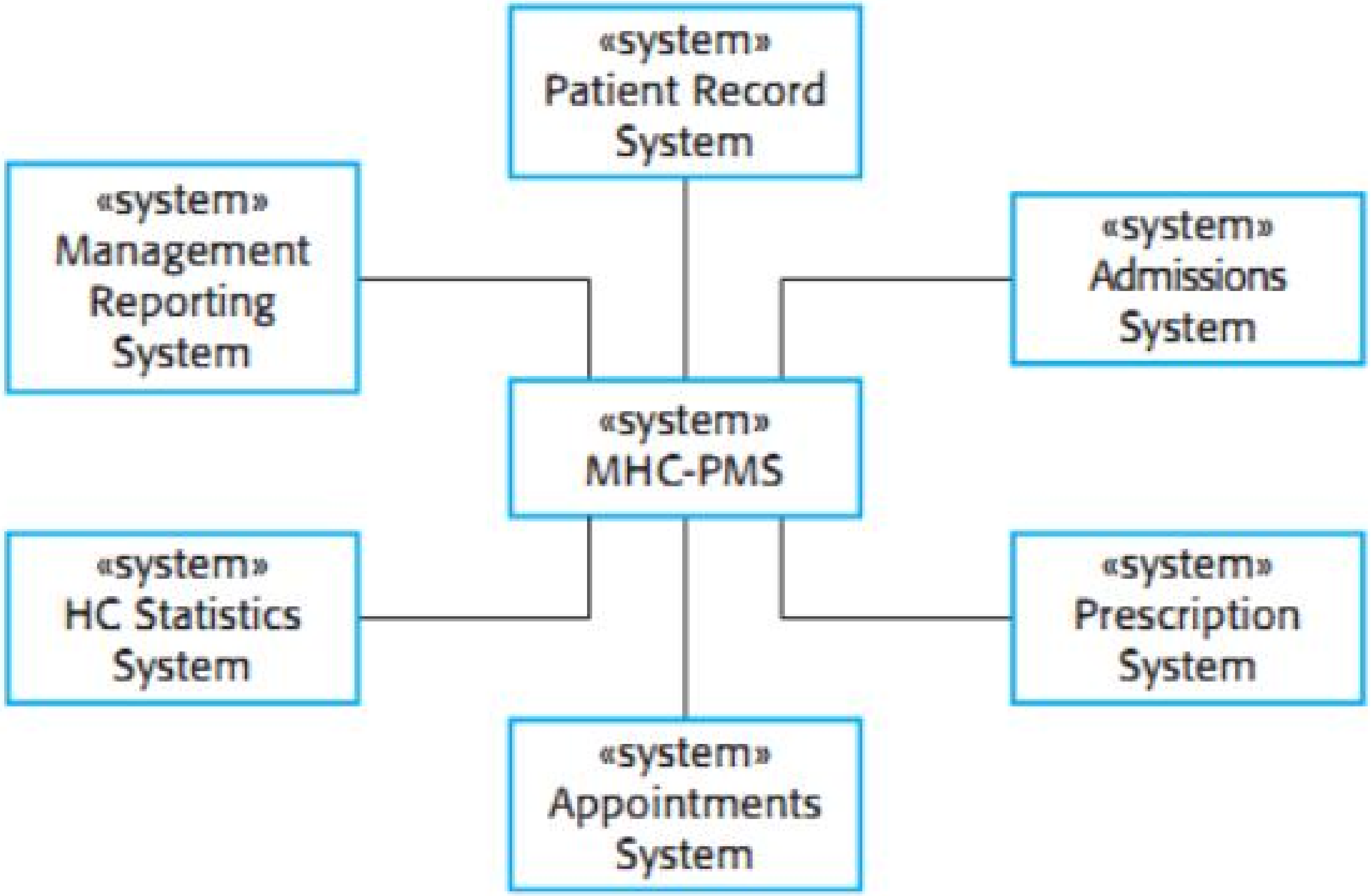
System Boundaries

- System boundaries are established to define what is inside and what is outside the system.
- They show other systems that are used by or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

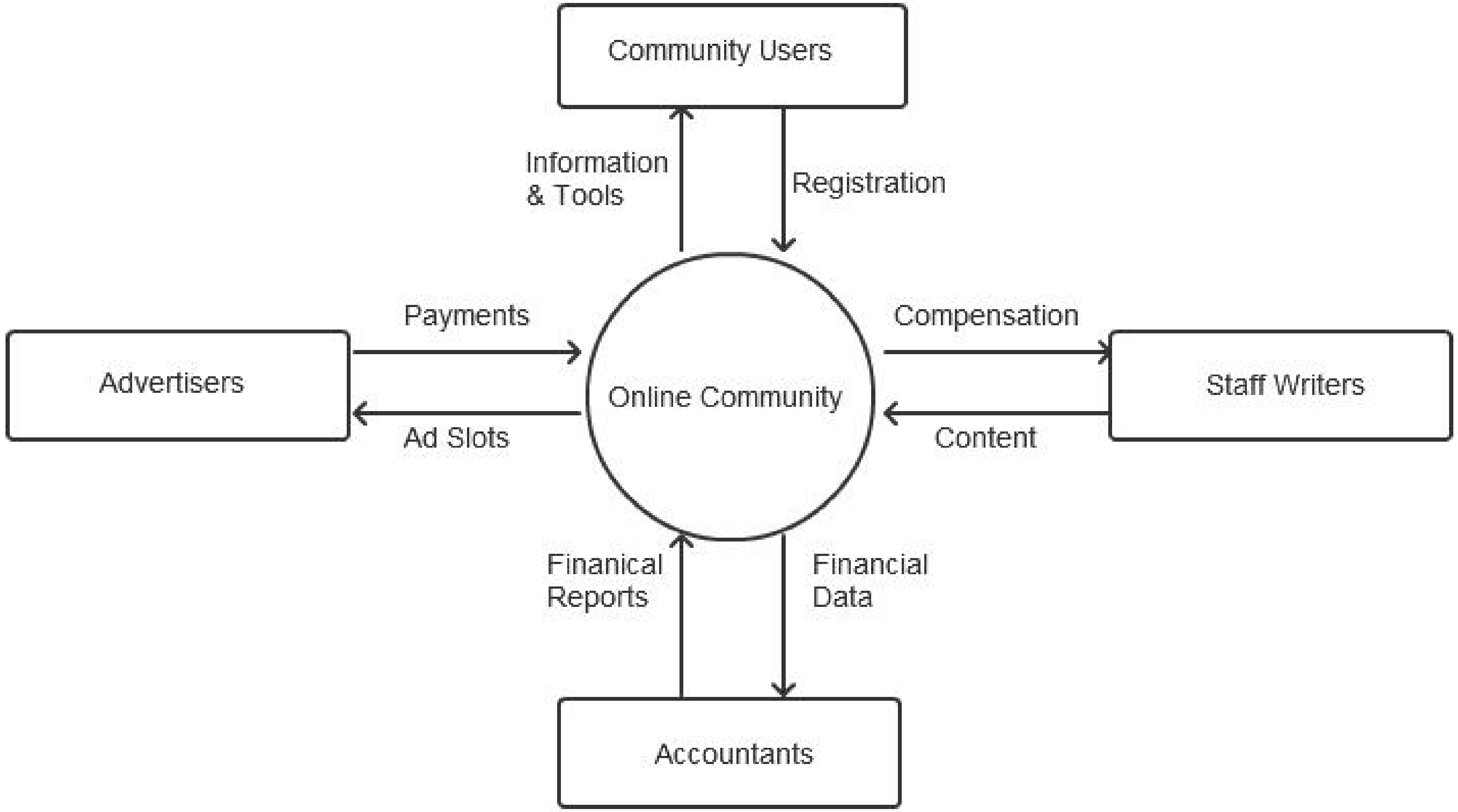
The Context of University Admission System


<<system>> Student Progress Record System
<<system>> University Financial System
<<system>> University Admission System
<<system>> Entrance exam System
<<system>> University Reporting System

The context of the MHC-PMS


«system» Patient Record System
«system» Management Reporting System
«system» Admissions System
«system» MHC-PMS
«system» HC Statistics System
«system» Prescription System
«system» Appointments System

Context diagram of Online System



PROCESS MODEL

Process Model

- **Context models simply show** the other systems in the environment, not how the system being developed is used in that environment.
- **Process models reveal** how the system being developed is used in broader business processes.
- **Process models show** the overall process and the processes that are supported by the system.
- **Data flow models** may be used to show the processes and the flow of information from one process to another.
- UML activity diagrams may be used to define business process models.

ACTIVITY DIAGRAM

UML Activity Diagram Tutorial | Lucidchart

https://www.lucidchart.com/pages/uml-activity-diagram#section_0

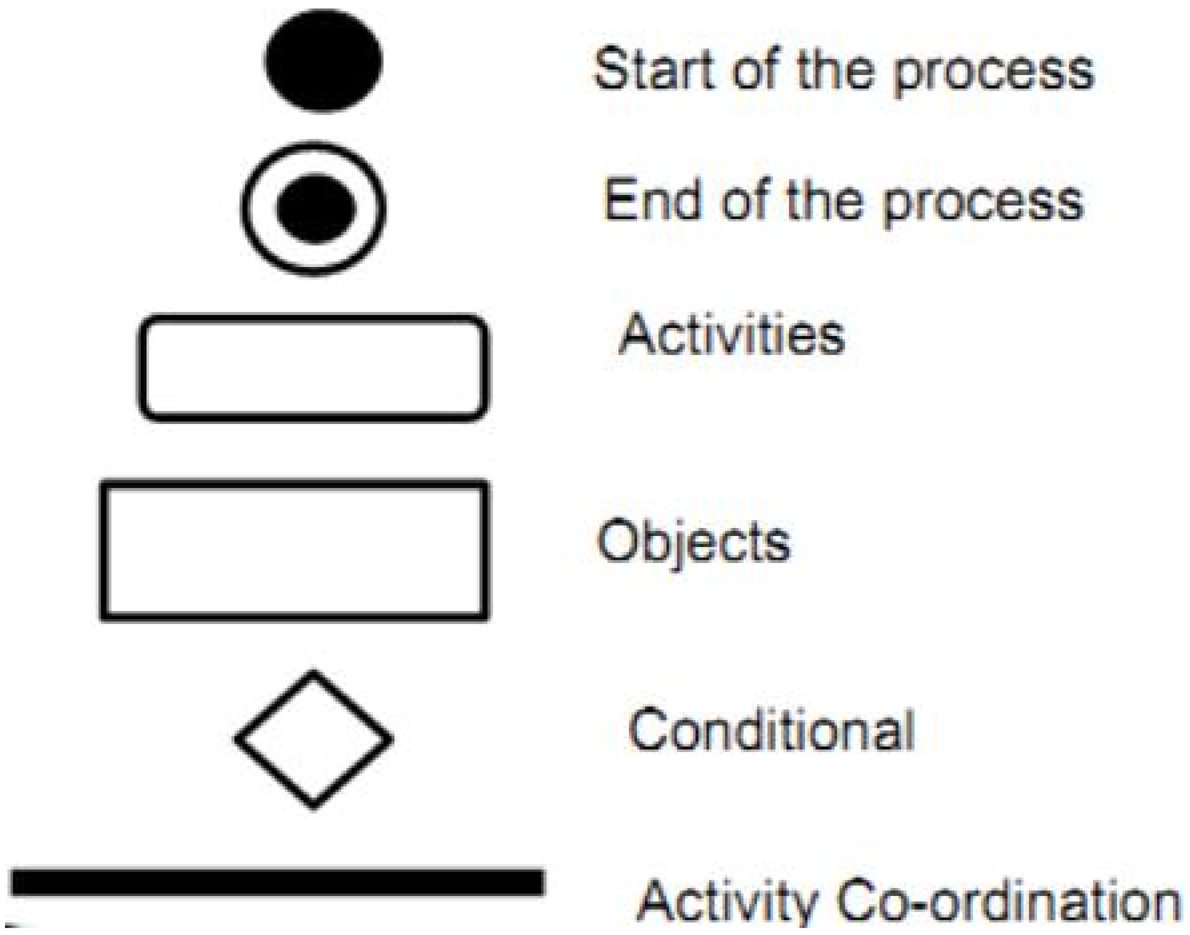
Activity Diagram

- **Graphical representations** of workflows, stepwise activities with support for choice, iteration and concurrency.
- Use to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.
- Activity diagrams are constructed from a limited number of shapes, connected with arrows.
- The most important shape types:
  - **rounded rectangles** represent activities;
  - **diamonds** represent decisions;
  - **bars** represent the start (split) or end (join) of concurrent activities;
  - **a black circle** represents the start (initial state) of the workflow;
  - **an encircled black circle** represents the end (final state).

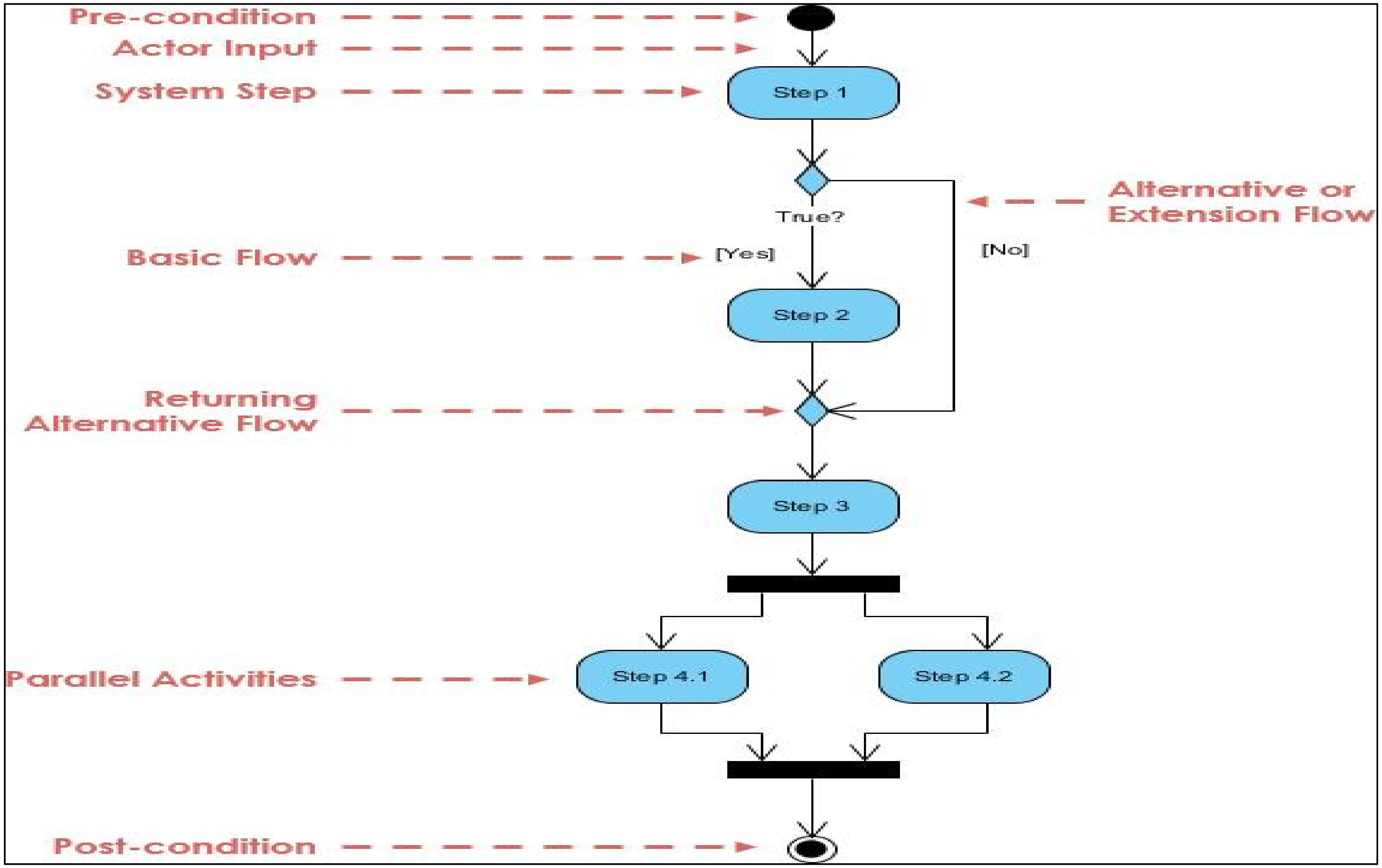
Where to use Activity Diagrams

- Modeling work flow by using activities.
- Modeling business requirements.
- Investigate business requirements at a later stage.
- High level understanding of the system's functionalities.

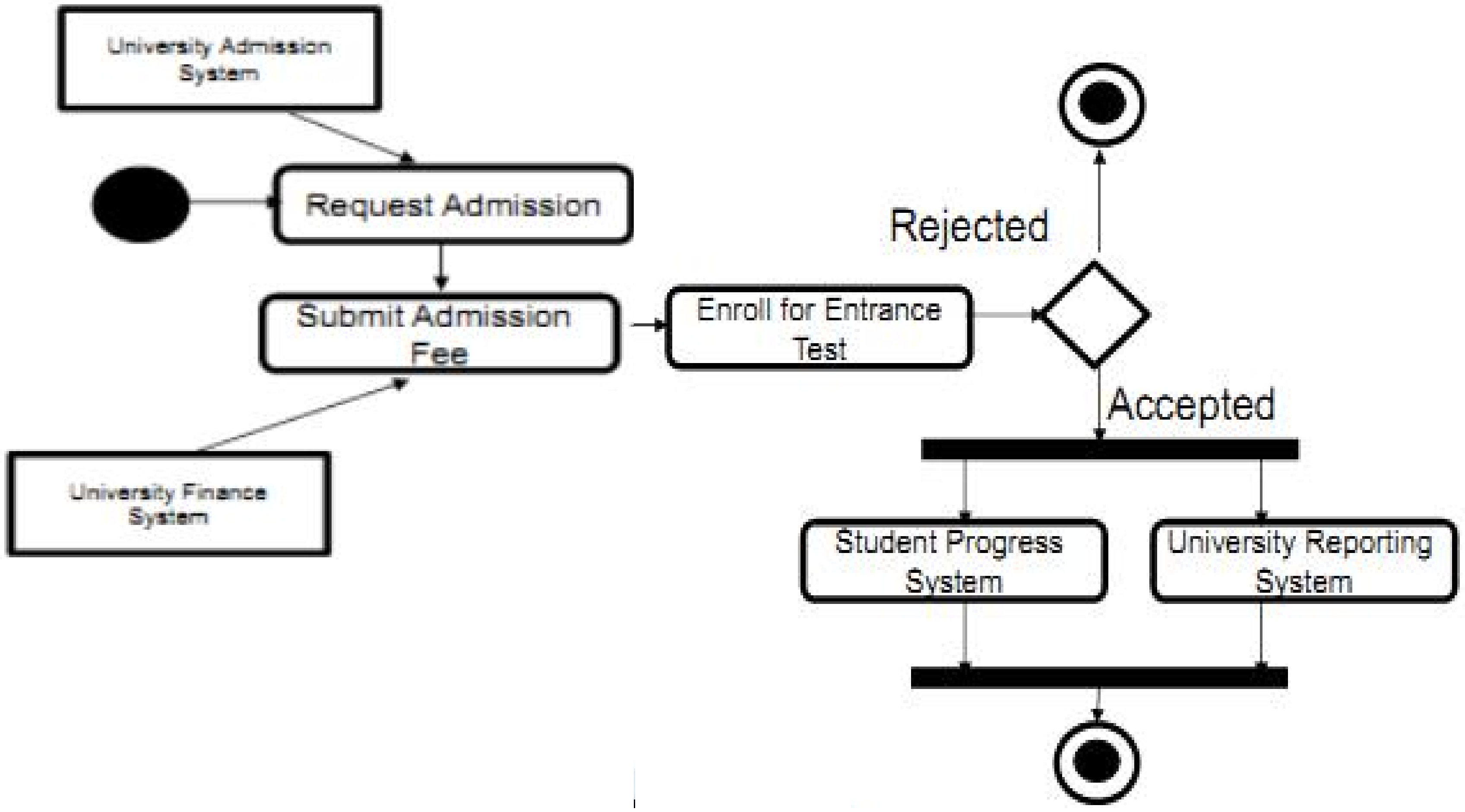
Activity Diagram Notations



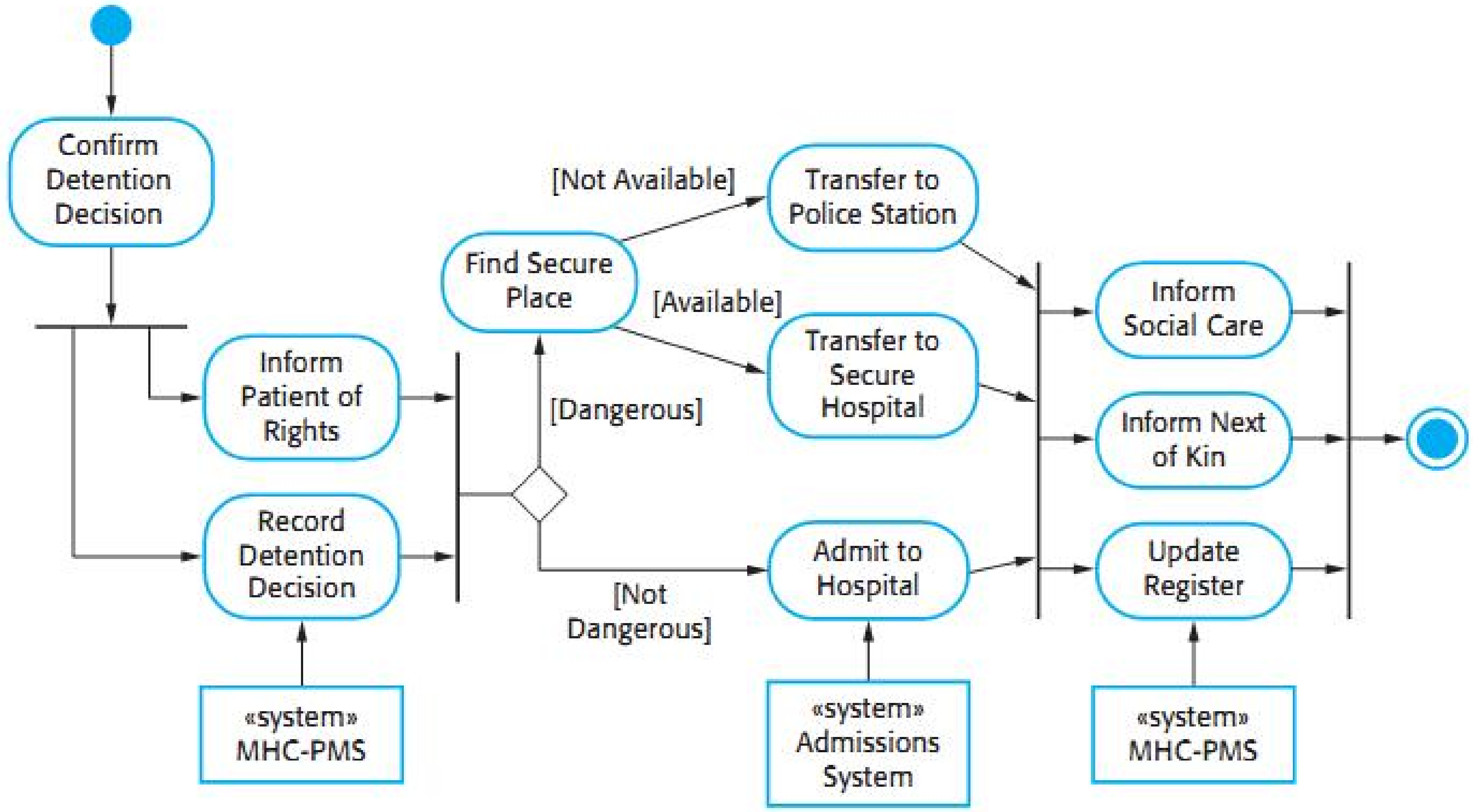
Activity Diagram w.r.t Use Case



University Admission System



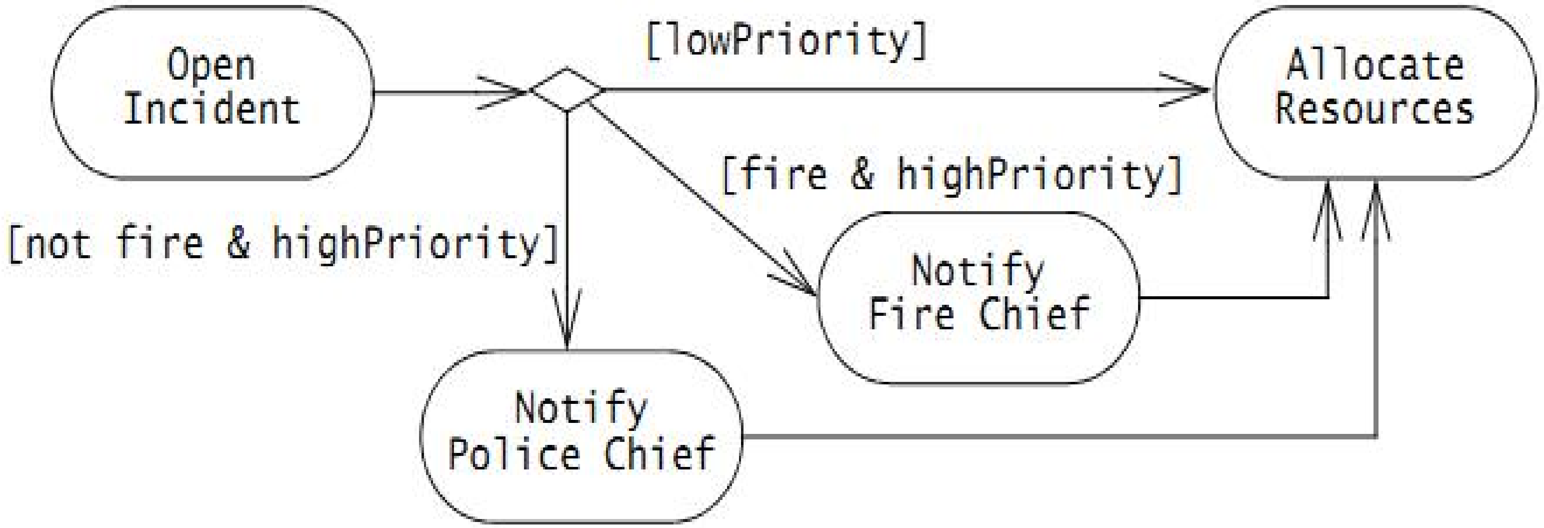
Process model of involuntary detention



Example-1

- Decision in the OpenIncident process. If the Incident is a fire and is of high priority, the Dispatcher notifies the FireChief. If it is a high-priority Incident that is not a fire, the PoliceChief is notified. In all cases, the Dispatcher allocates resources to deal with the Incident.

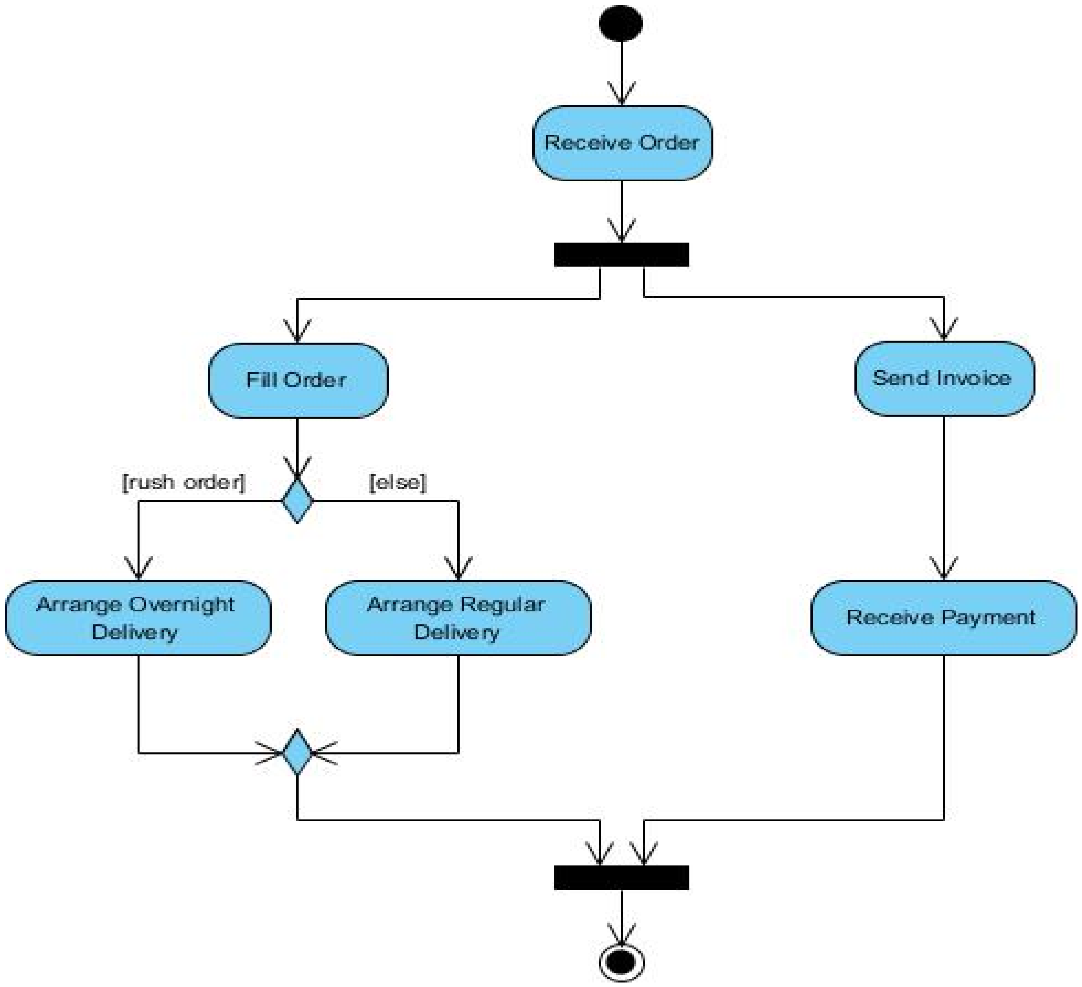
Solution


Open Incident
[lowPriority]
Allocate Resources
[fire & highPriority]
[not fire & highPriority]
Notify Fire Chief
Notify Police Chief

Example-2

- Given the problem description related to the workflow for processing an order, let's model the description in visual representation using an activity diagram:
- **Process Order - Problem Description**
- **Once the order is rece**ived, the activities split into two parallel sets of activities. One side fills and sends the order while the other handles the billing.
- **On the Fill Order si**de, the method of delivery is decided conditionally. Depending on the condition either the **Overnight Delivery activity** or the Regular Delivery activity is performed.
- **Finally the parallel activities** combine to close the order.

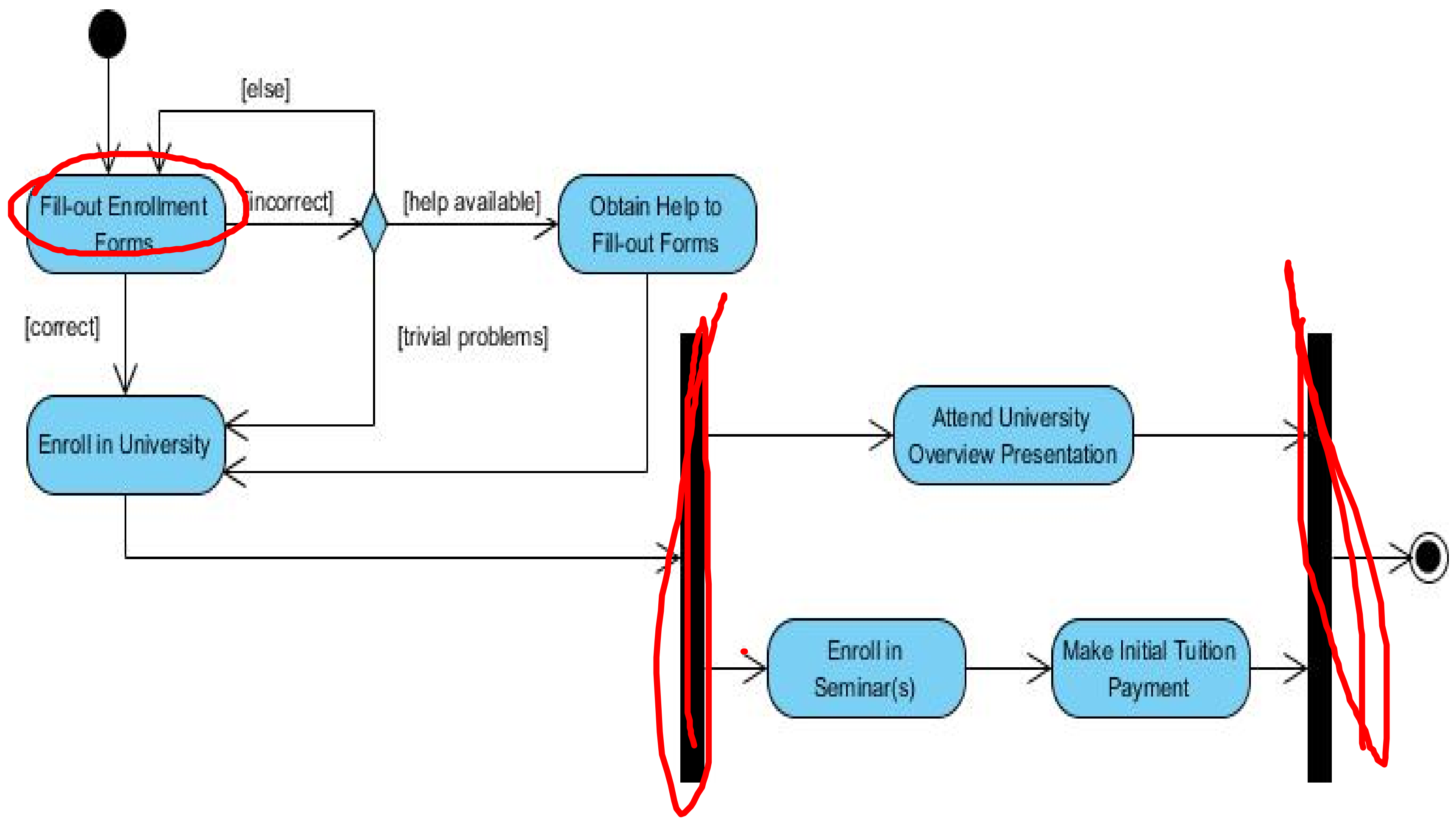
Solution


Receive Order
Fill Order
Send Invoice
[rush order]
[else]
Arrange Overnight Delivery
Arrange Regular Delivery
Receive Payment

Example-3

- **This UML activity diagram example describes a process for student enrollment in a university as follows:**
- An applicant wants to enroll in the university.
- The applicant hands a filled out copy of Enrollment Form.
- The registrar inspects the forms.
- The registrar determines that the forms have been filled out properly.
- The registrar informs student to attend in university overview presentation.
- The registrar helps the student to enroll in seminars
- The registrar asks the student to pay for the initial tuition.

Solution



[else]
Fill-out Enrollment Forms
[incorrect]
[help available]
Obtain Help to Fill-out Forms
[correct]
[trivial problems]
Enroll in University
Attend University Overview Presentation
Enroll in Seminar(s)
Make Initial Tuition Payment

Example-4 (Task)

- Consider the process of ordering a pizza over the phone. Draw an activity diagram representing each step of the process, from the moment you pick up the phone to the point where you start eating the pizza. Do not represent any exceptions. Include activities that others need to perform.

INTERACTION MODELS

Interaction Models

- All Systems involve interaction of some kind.
- **User Interaction**
  - Helps identify user requirements
- **Interaction between system being developed and other systems**
  - Highlights communications requirements and issues
- **Inter component interaction**
  - Helps in identifying whether the structure will deliver the required performance and dependability needs

Approaches to Interaction Modeling

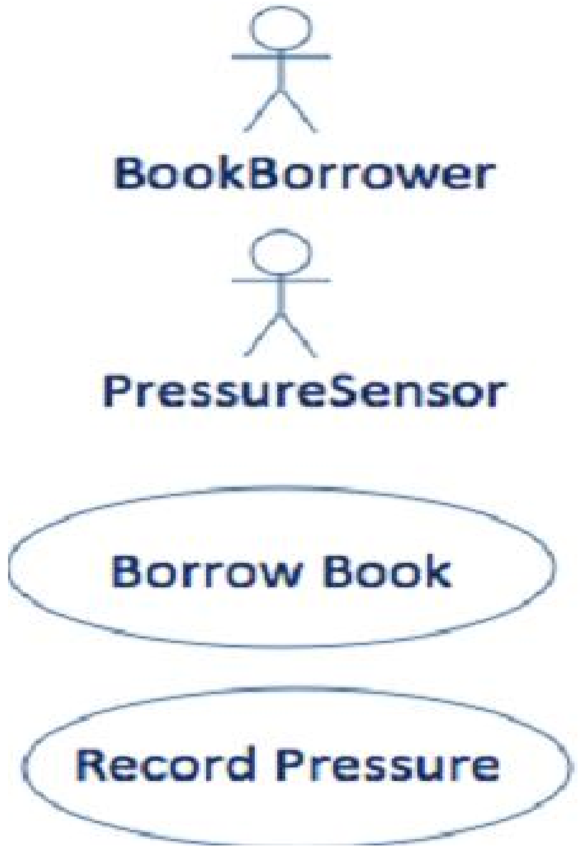
- Two related approaches to interaction modeling are:
- **Use Case Modeling**
  - Interaction between the system and external actors (users or other systems)
- **Sequence Diagrams**
  - Interactions between system components, although external agents may also be included

USE CASE DIAGRAM

Use Case Modeling

- **Use cases** were developed originally to support requirements elicitation and now incorporated into the UML.
- **Each use case** represents a discrete task that involves external interaction with a system.
- **Actors in a use** case may be people or other systems.
- **Represented diagrammatically** to provide an overview of the use case and in a more detailed textual form.

Actors and Use Case Diagram

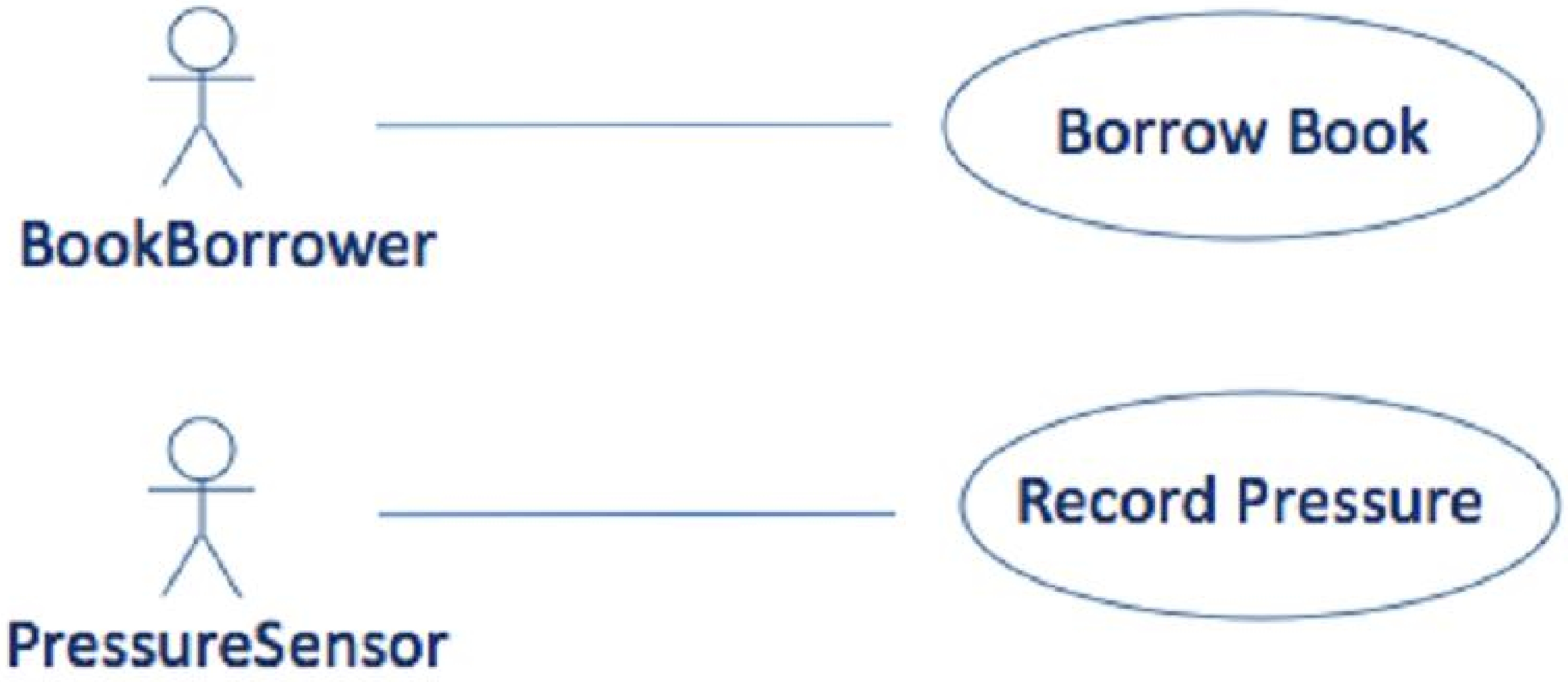


- An **actor** is a user of a system in a particular **role**.

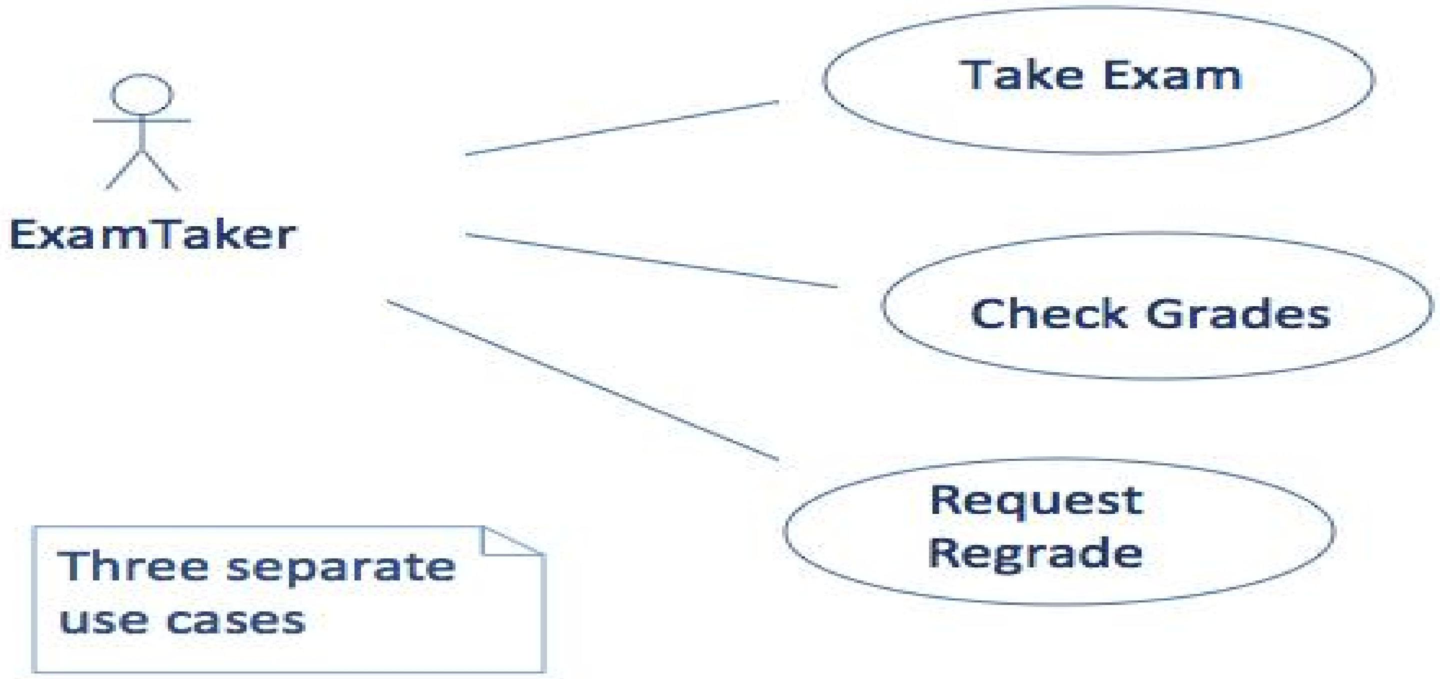
  An actor can be human or an external system.

- A **use case** is a a task that an actor needs to perform with the help of the system.

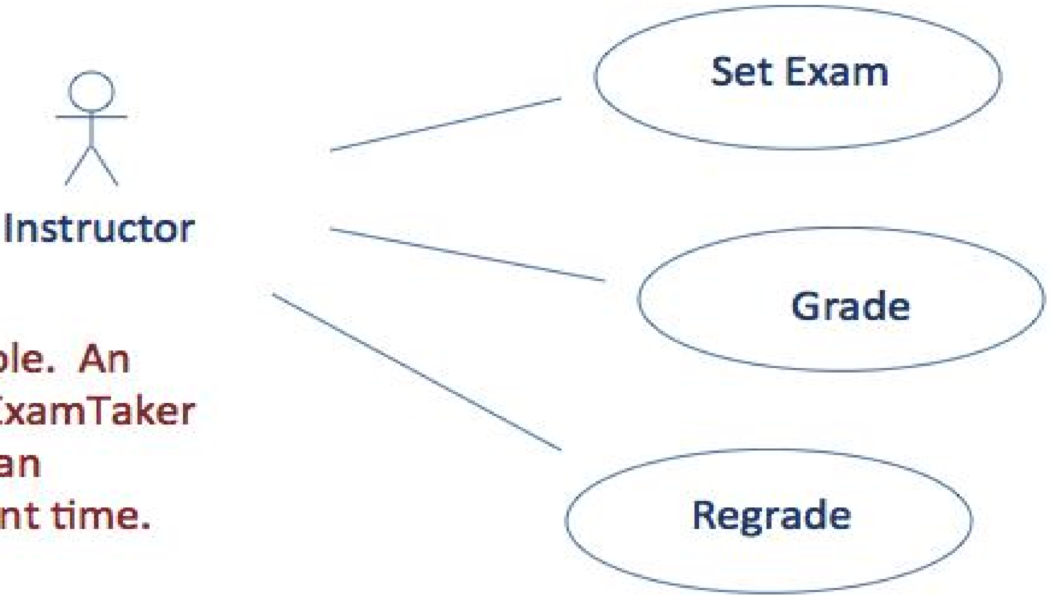
Actors and Use Cases



Use Cases for Exam System

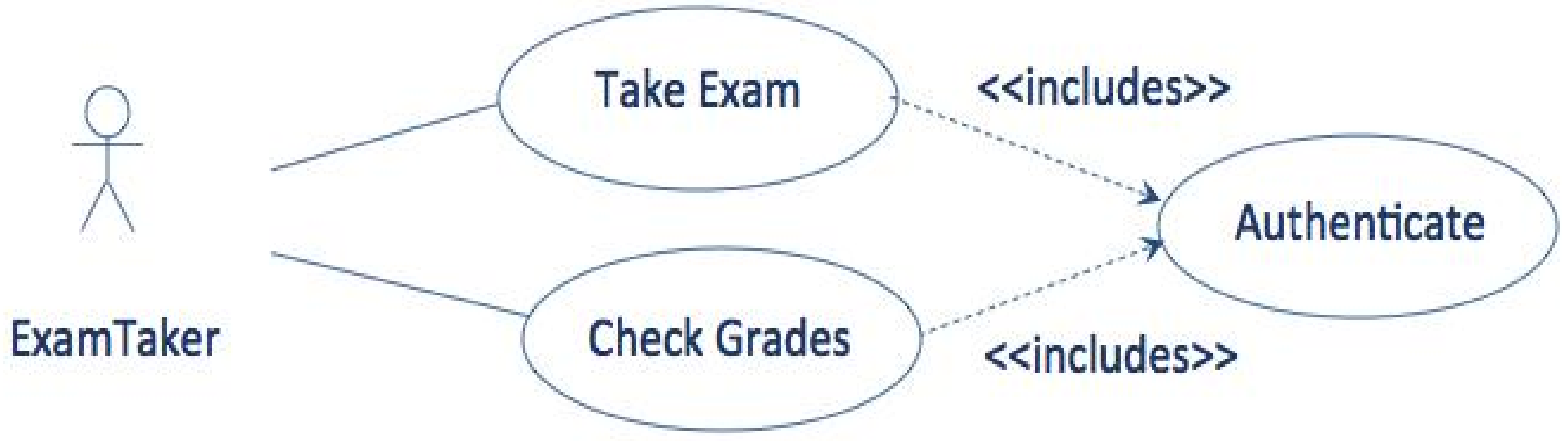


Take Exam Use Case Cont.



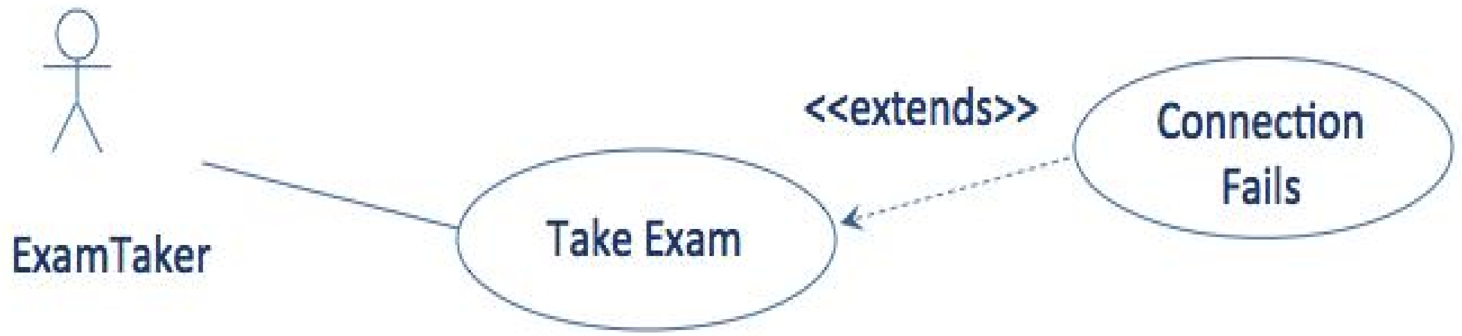
Note that actor is a role. An individual can be an ExamTaker on one occasion and an Instructor at a different time.

Relationship Between Use Cases: <<includes>>



The Authenticate use case may be used in other contexts

Relationship Between Use Cases: <<extends>>



<<includes>>	is used for use cases that are in the flow of events of the main use case.
<<extends>>	is used for exceptional conditions, especially those that can occur at any time.

Class diagram Examples

What is Class Diagram?

Read for details

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

What is Class Diagram?

- In software engineering, **a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.**

Class Diagrams

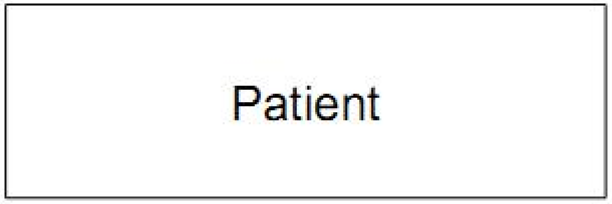
- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An object class can be thought of as a general definition of one kind of system object.
- An association is a link between classes that indicates that there is some relationship between these classes.
- **When you are developing models** during the early stages of the software engineering process,
  - Objects represent something in the real world, such as a patient, a prescription, doctor, etc.
- **As an implementation is developed,**
  - you usually need to define additional implementation objects that are used to provide the required system functionality

Purpose of Class Diagrams

- **Shows Static Structure** of classifiers in a system
- **Diagram provides** a basic notation for other structure diagrams prescribed by UML
- **Helpful for developers** and other team members too
- **Business Analysts** can use class diagrams to model systems from a business perspective

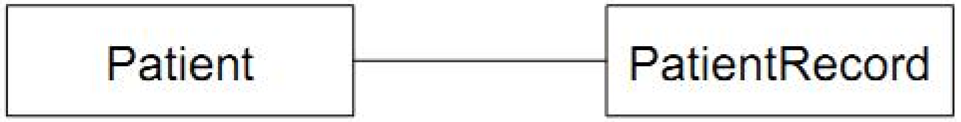
1- Create Class

- What is class: A description of a group of objects all with similar roles in the system Class diagrams in UML can be expressed at different level of detail.
- **At the first stage of developing a model,**
  - Look at the world and identify the essential objects and represent them as class.
- **The simplest way is to write the name of the class in a box**

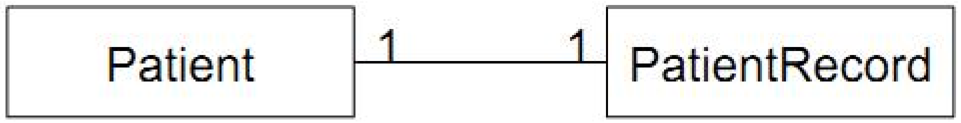


1.1- Class (Simple Association):

- You can also note the existence of an association by drawing a line between the classes

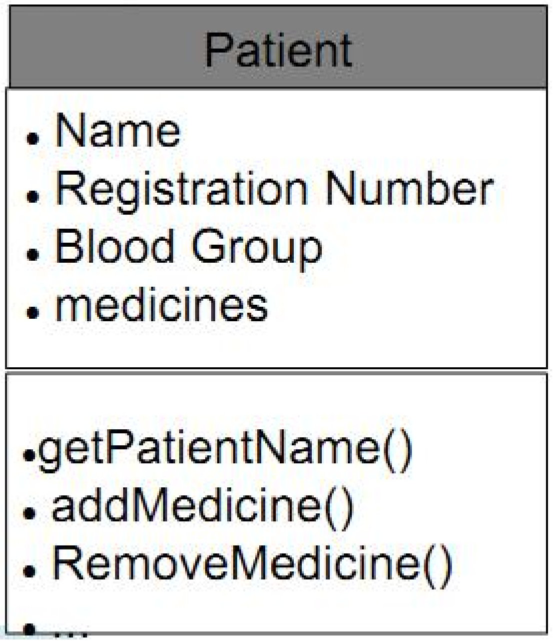


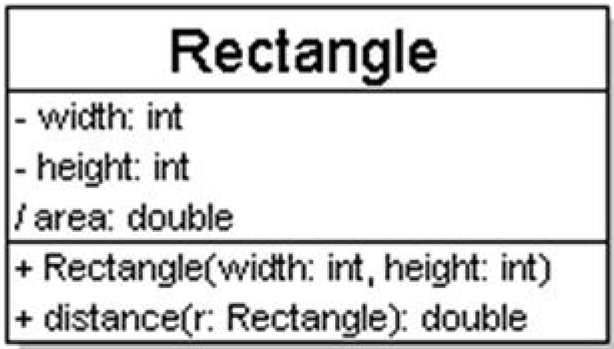
- You can annotate the relationship to show how many objects are involved in the association



1.2 Adding details to Class diagram

- attributes (fields, instance variables)
  - visibility:
  - + public, # protected, - private,~ package (default), / derived
  - underline static attributes




Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

2- Relationships between classes

- **2.2 Generalization**: an inheritance relationship
  - inheritance between classes
  - interface implementation
- **2.3 Association:** a usage relationship
  - Associational Relationship
  - **Association Types:**
    - Aggregation
    - Composition
    - Dependency

2.2 Generalization (inheritance) relationships

- Hierarchies drawn top-down
- arrows point upward to parent
- line/arrow styles indicate whether parent is a(n):
  - class: solid line, black arrow

2.2 Generalization (or Inheritance)

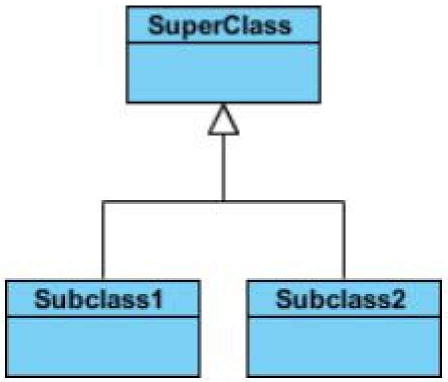
Inheritance (or Generalization):

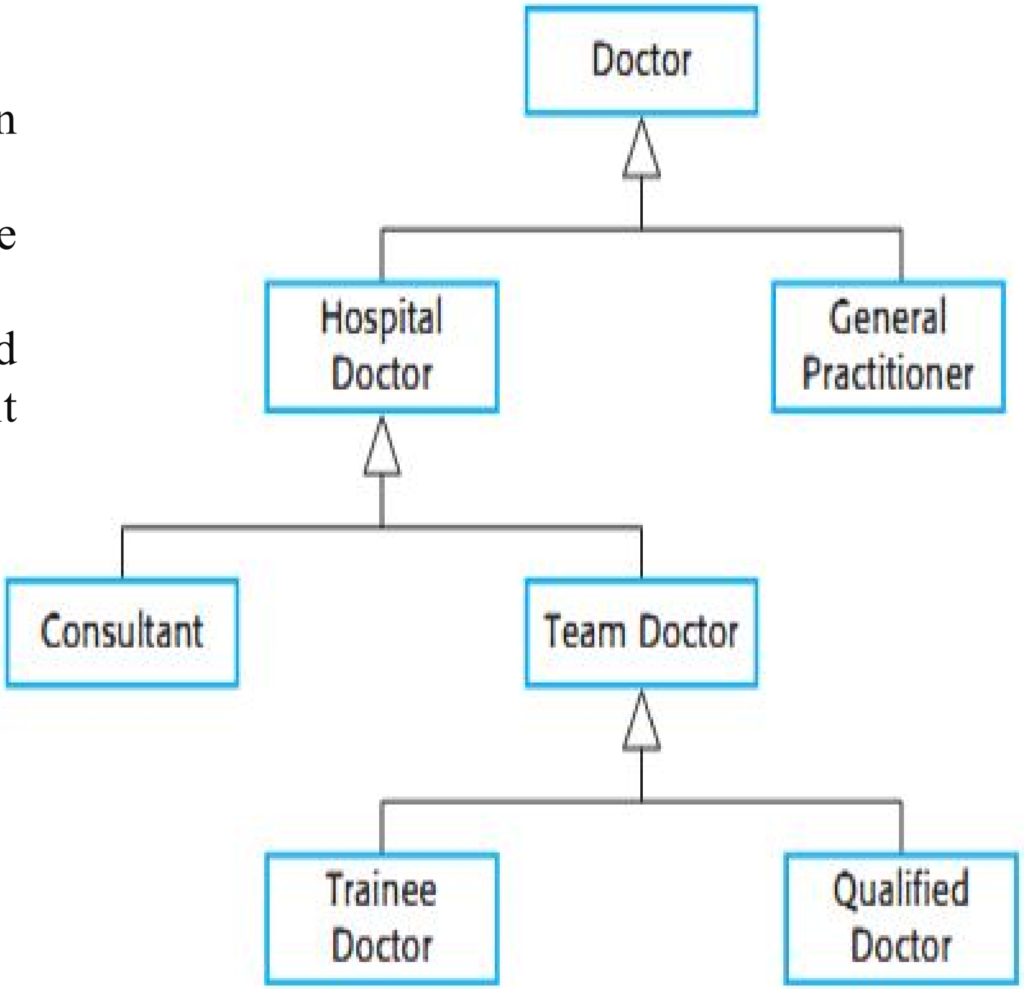
•**Represents an "is-a" relationship.**

•**An abstract class** name is shown in italics.

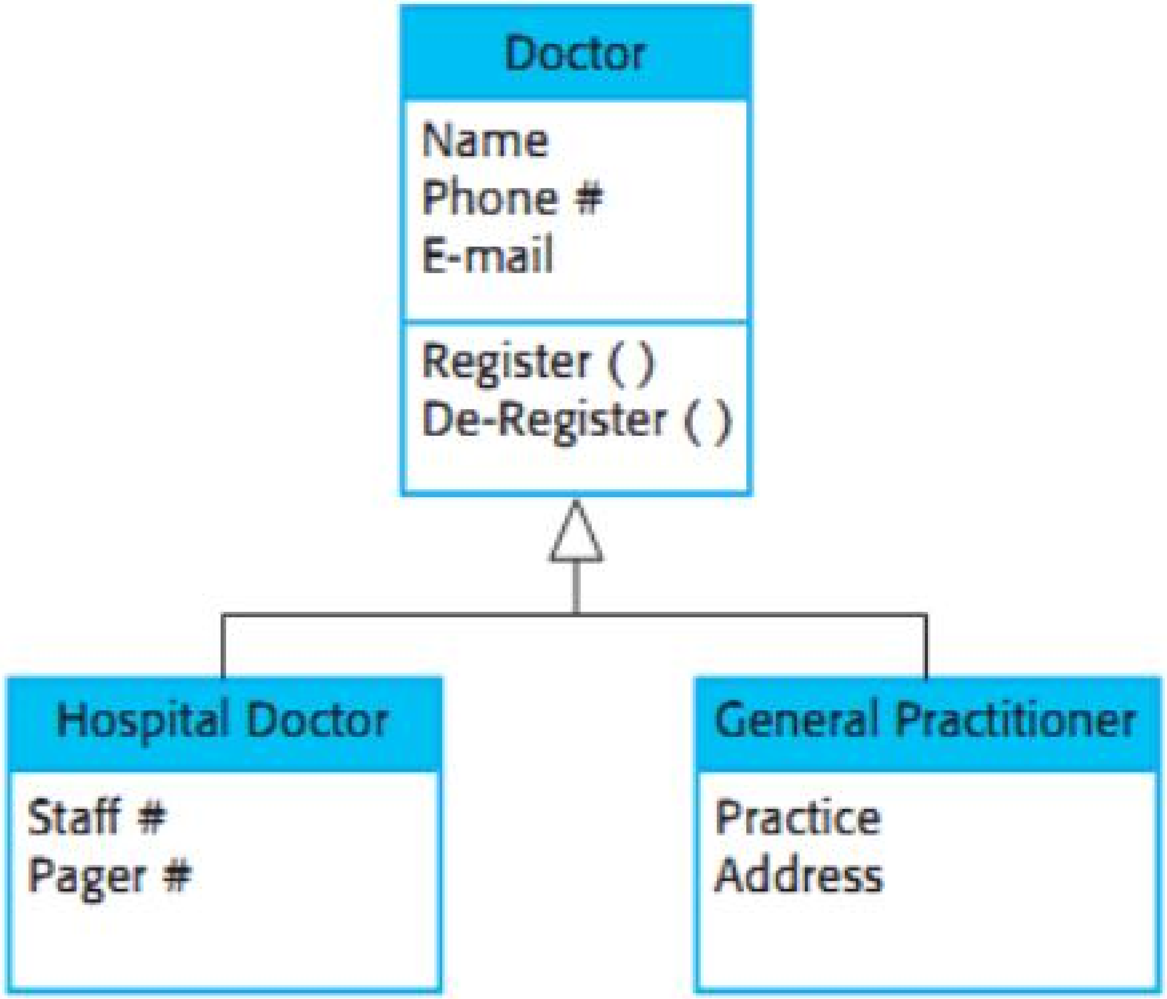
•**SubClass1 and SubCl**ass2 are specializations of Super Class.

•**A solid line** with a hollow arrowhead that point from the child to the parent class





2.3. Generalization (or Inheritance)



2.3.1 Associational relationships

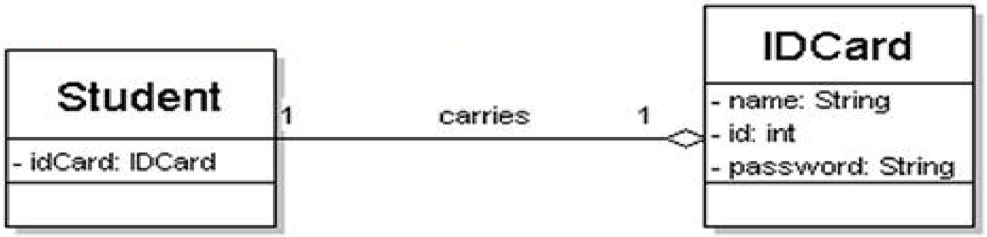
- Associational (usage) relationships
  - multiplicity (how many are used)
    - * ⇒ 0, 1, or more
    - 1 ⇒ 1 exactly
    - 2..4 ⇒ between 2 and 4, inclusive
    - 3..* ⇒ 3 or more (also written as "3..")
  - name (what relationship the objects have)
  - navigability (direction)

2.3.1 Associational relationships

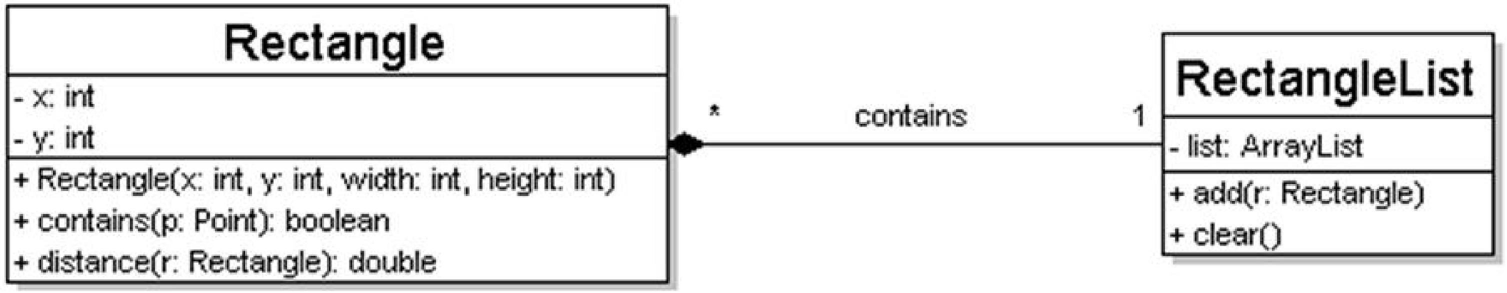
- Associational (usage) relationships
  - multiplicity (how many are used)
    - * ⇒ 0, 1, or more
    - 1 ⇒ 1 exactly
    - 2..4 ⇒ between 2 and 4, inclusive
    - 3..* ⇒ 3 or more (also written as “3..”)
  - name (what relationship the objects have)
  - navigability (direction)

2.3.1 Multiplicity of associations

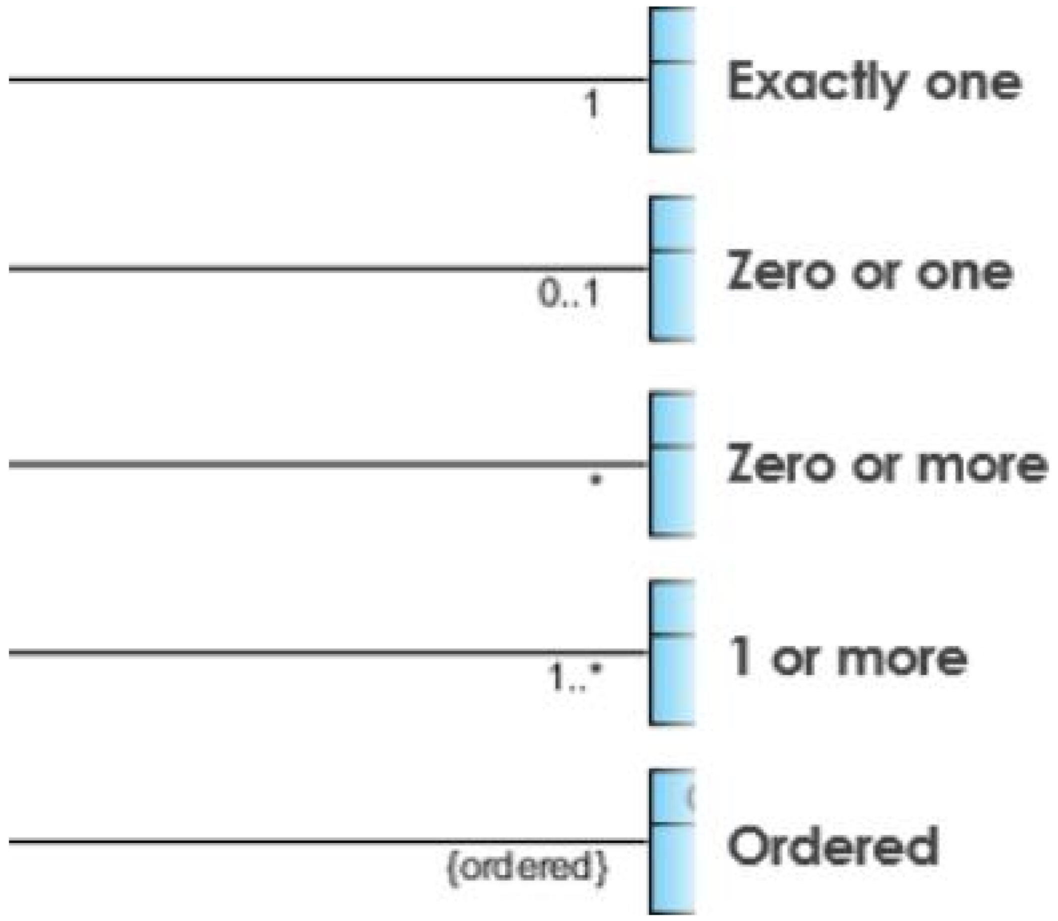
- one-to-one
  - each student must carry exactly one ID card



  - one-to-many
  - one rectangle list can contain many rectangles

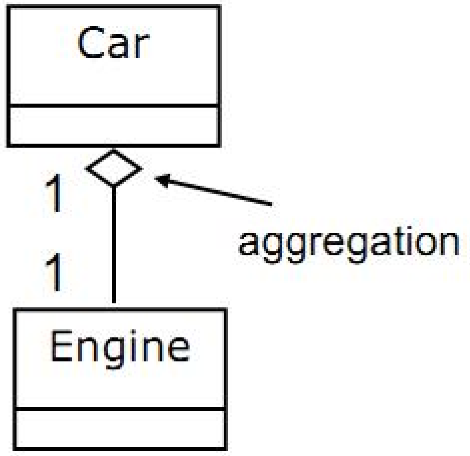

Rectangle
- x: int
- y: int
+ Rectangle(x: int, y: int, width: int, height: int)
+ contains(p: Point): boolean
+ distance(r: Rectangle): double
*
contains
1
RectangleList
- list: ArrayList
+ add(r: Rectangle)
+ clear()

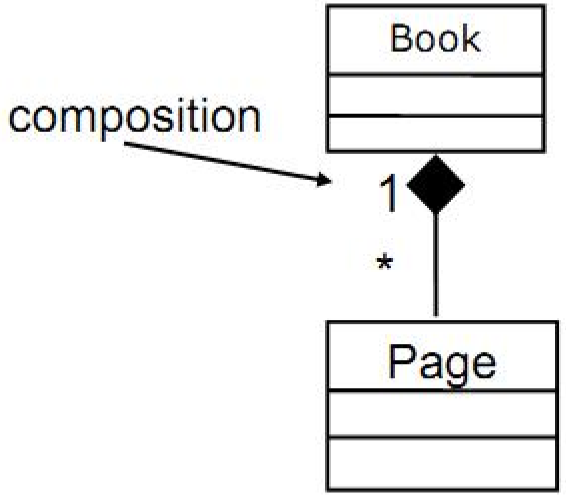
2.3.1 Multiplicity of associations

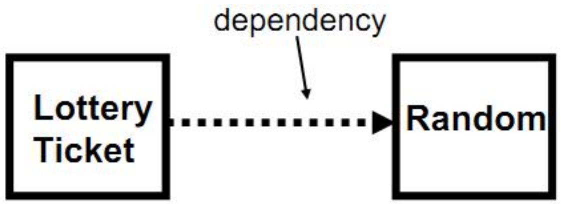


2.3.2 Association types

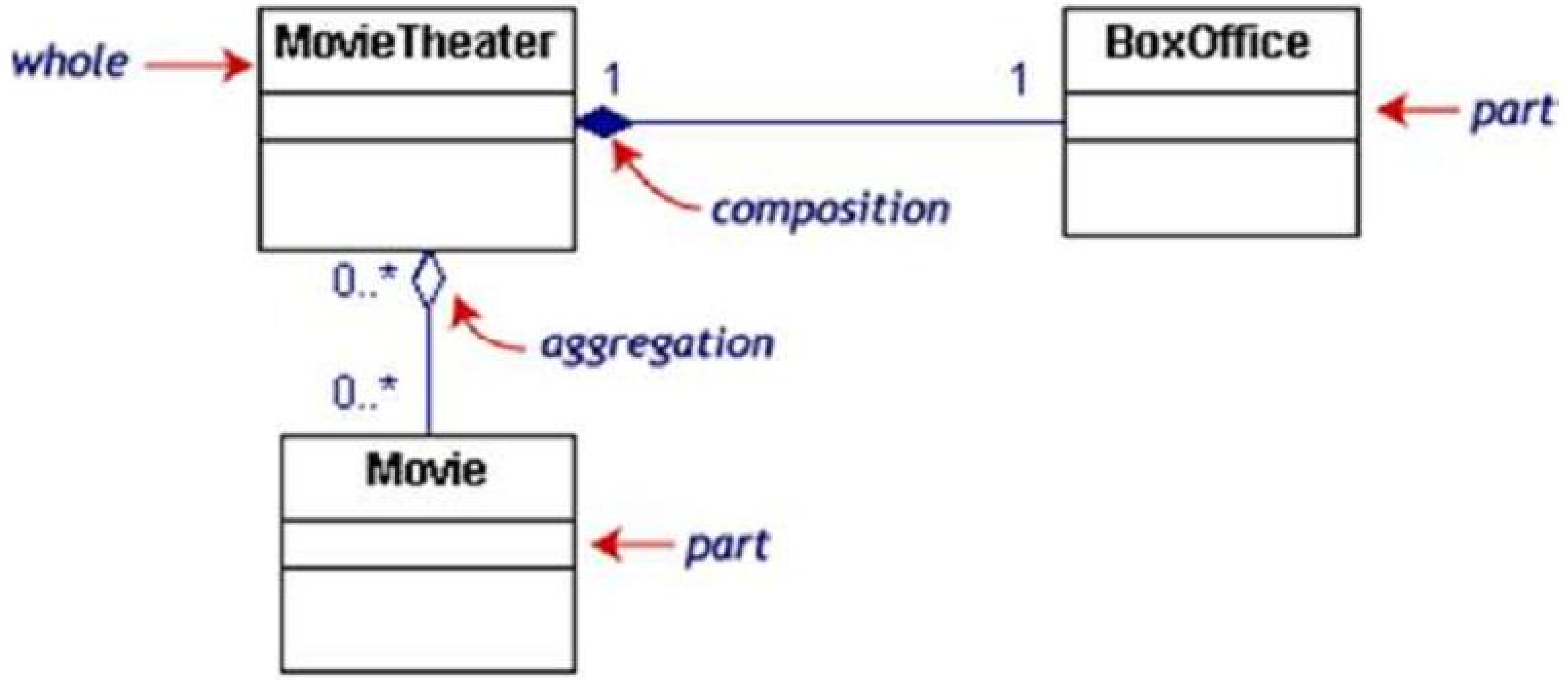
- **Aggregation:** "is part of"
  - symbolized by a clear white diamond
- **Composition:** "is entirely made of"
  - stronger version of aggregation
  - the parts live and die with the whole
  - symbolized by a black diamond
- **Dependency:** "uses temporarily"
  - symbolized by dotted line
  - often is an implementation detail, not an intrinsic part of that object's state







Composition/Aggregation example



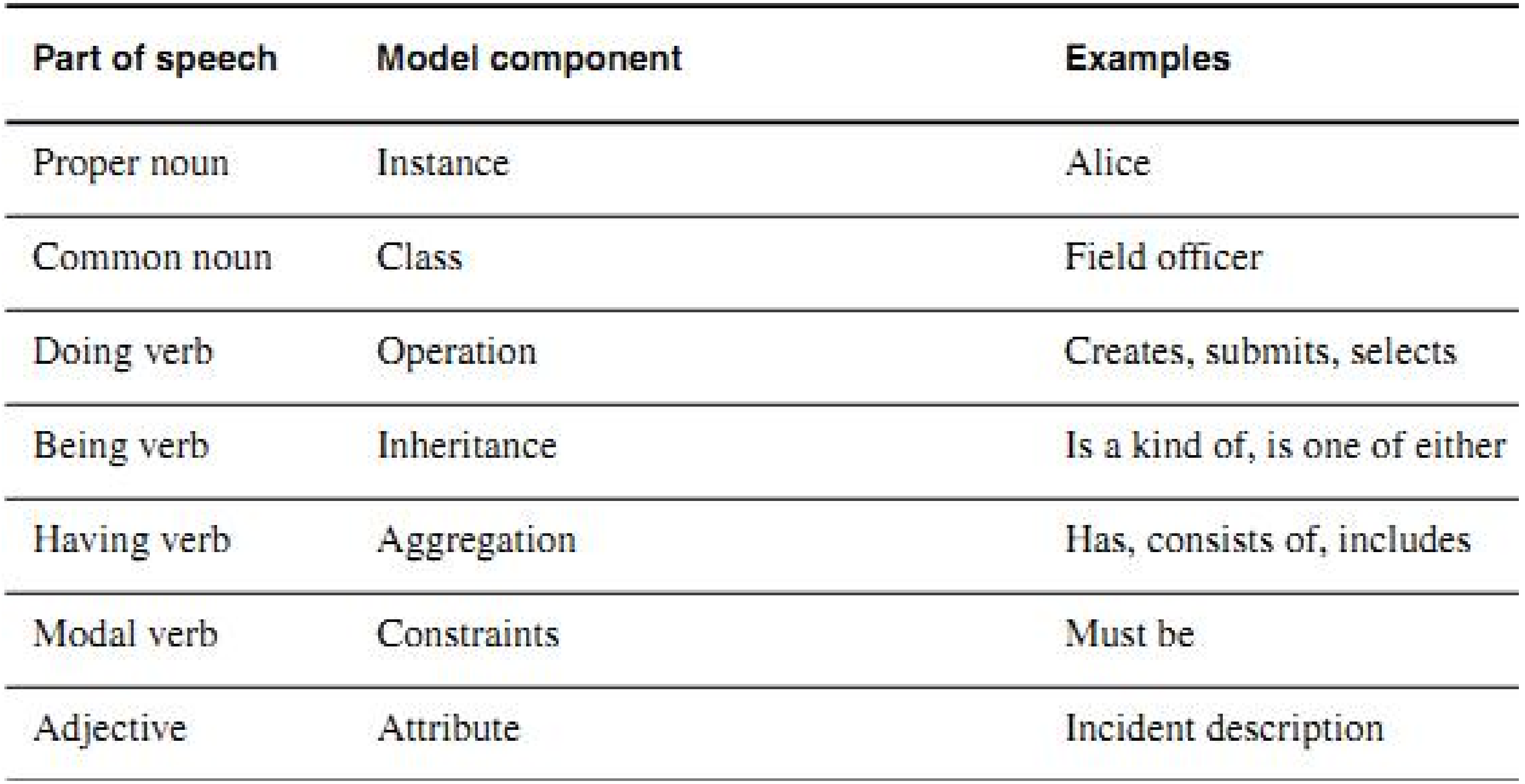
- If the movie theater goes away
  - so does the box office => composition
  - but movies may still exist => aggregation

CLASS DIAGRAM EXAMPLES

UML class diagrams examples - Abstract Factory Design Pattern, Library Management, Online Shopping, Hospital, Digital Imaging in Medicine, Android. (uml-diagrams.org)

https://www.uml-diagrams.org/class-diagrams-examples.html

Identifying Analysis Classes



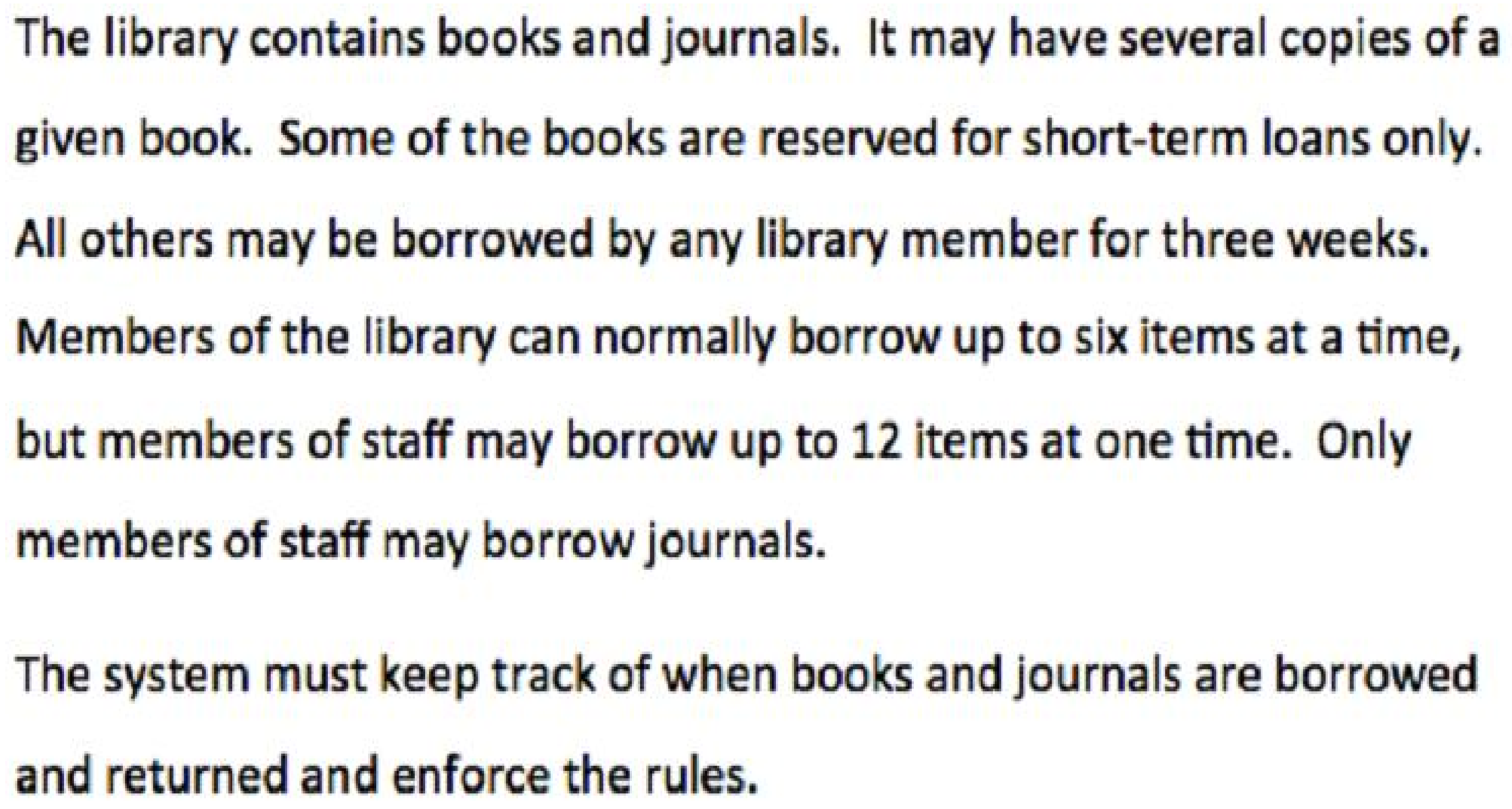
Part of speech	Model component	Examples
Proper noun	Instance	Alice
Common noun	Class	Field officer
Doing verb	Operation	Creates, submits, selects
Being verb	Inheritance	Is a kind of, is one of either
Having verb	Aggregation	Has, consists of, includes
Modal verb	Constraints	Must be
Adjective	Attribute	Incident description

Abbott’s heuristics for mapping parts of speech to model components [Abbott, 1983]

Identifying Classes

- **Classes are found in following forms:**
- **External entities:** The system, people or the device generates the information that is used by the computer based system.
- **Things:** The reports, displays, letter, signal are the part of the information domain or the problem.
- **Occurrences or events:** A property transfer or the completion of a series or robot movements occurs in the context of the system operation.
- **Roles:** The people like manager, engineer, salesperson are interacting with the system.
- **Organizational units:** The division, group, team are suitable for an application.
- **Places:** The manufacturing floor or loading dock from the context of the problem and the overall function of the system.
- **Structures:** The sensors, computers are defined a class of objects or related classes of objects.

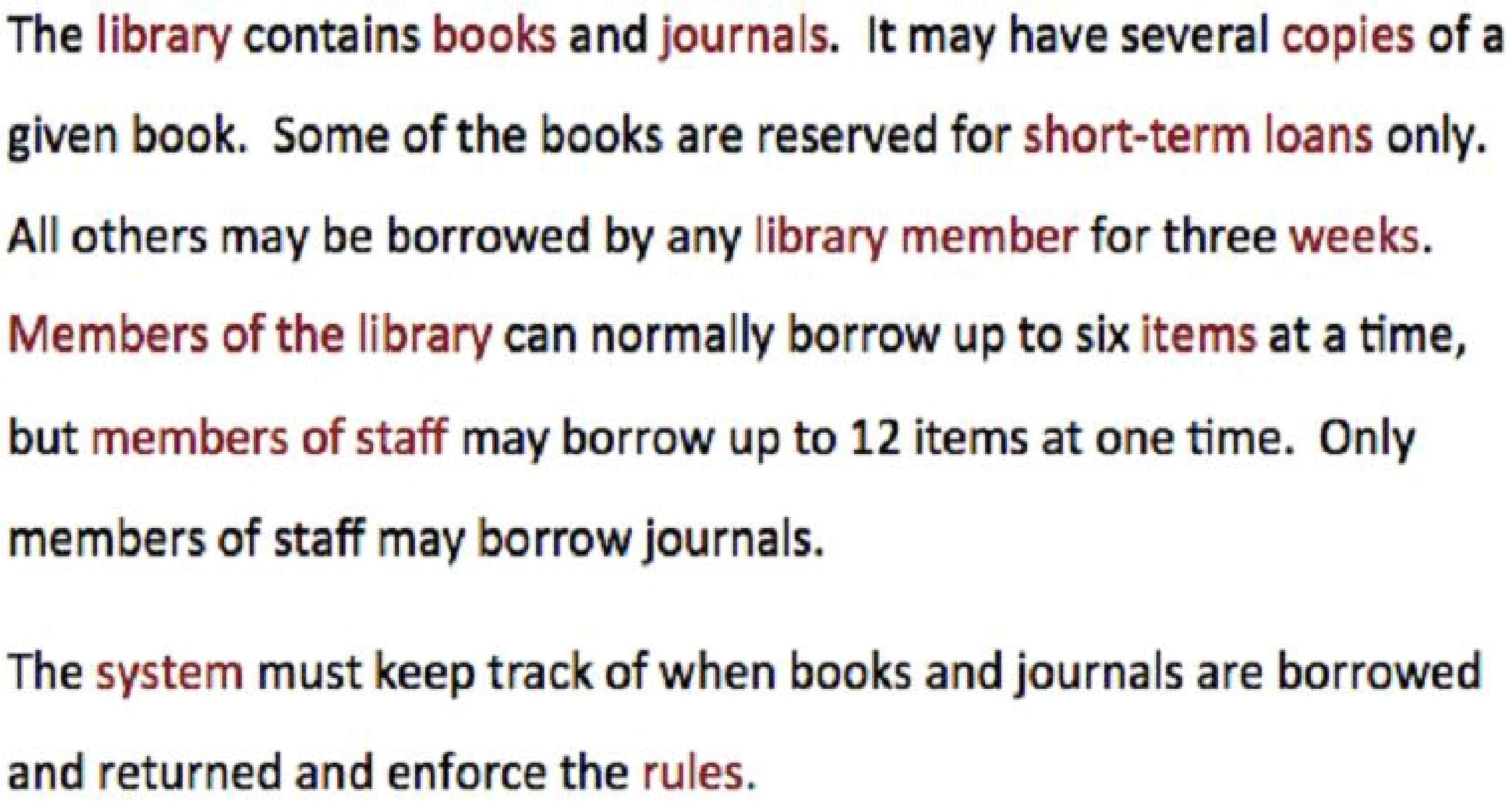
Noun Identification: A Library System



The library contains books and journals. It may have several copies of a given book. Some of the books are reserved for short-term loans only. All others may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

The system must keep track of when books and journals are borrowed and returned and enforce the rules.

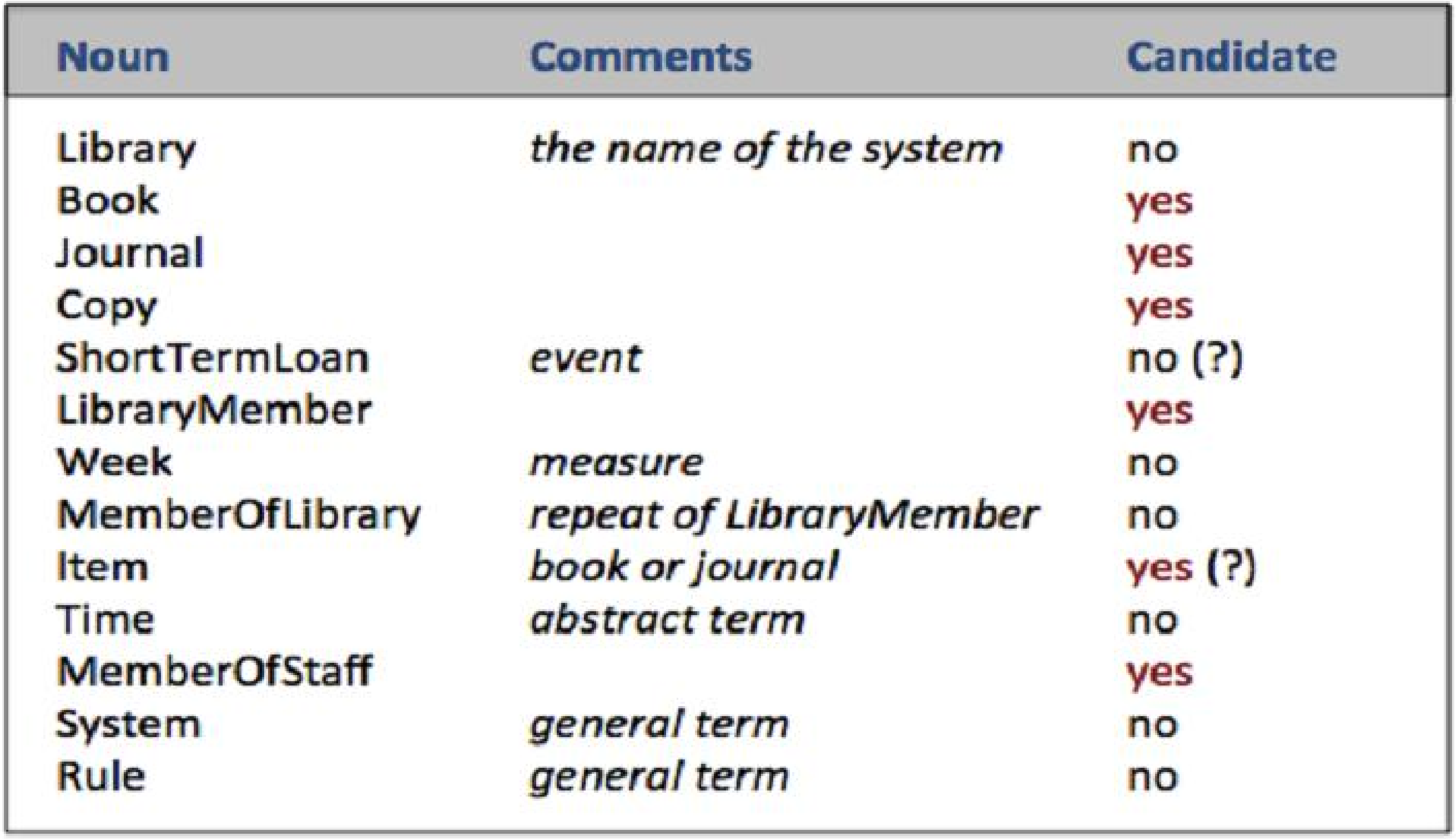
Noun Identification: A Library System



The library contains books and journals. It may have several copies of a given book. Some of the books are reserved for short-term loans only. All others may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

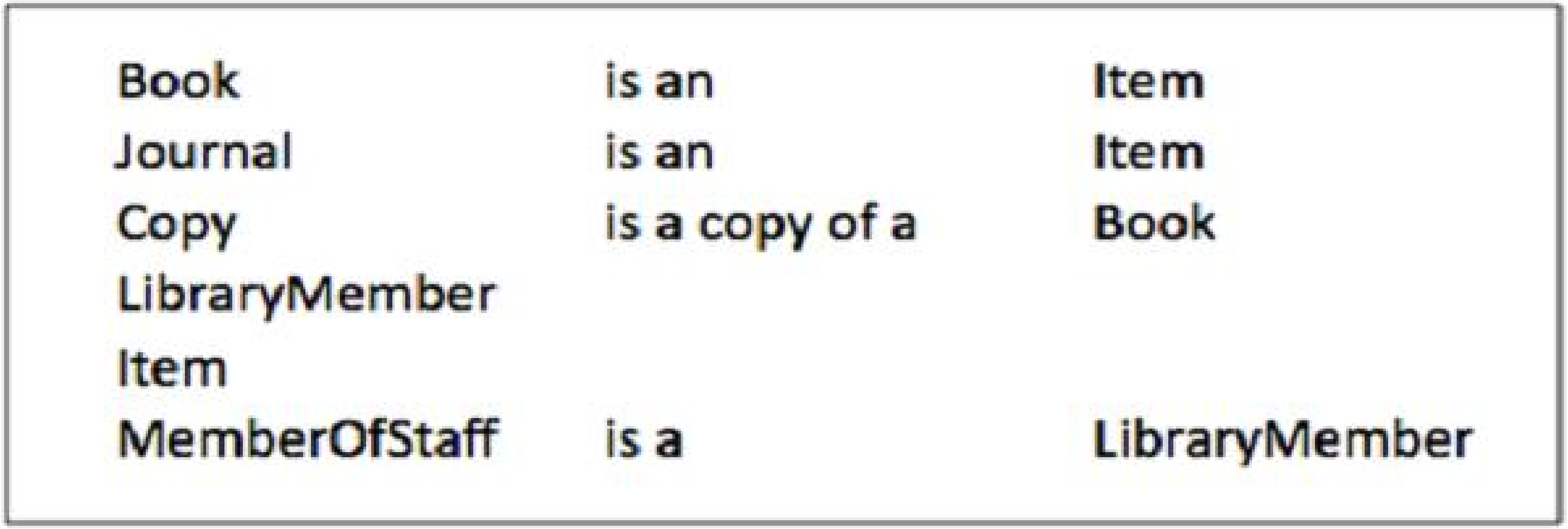
The system must keep track of when books and journals are borrowed and returned and enforce the rules.

Candidate Classes



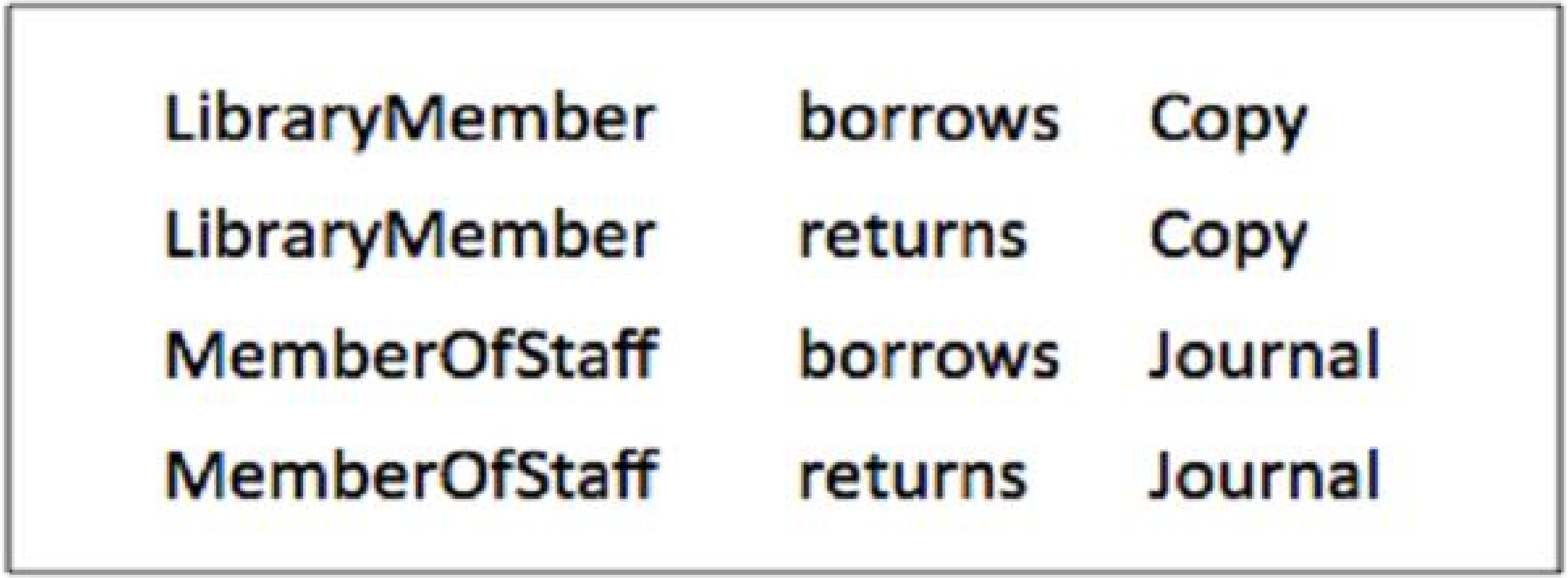
Noun	Comments	Candidate
Library	*the name of the system*	no
Book		yes
Journal		yes
Copy		yes
ShortTermLoan	*event*	no (?)
LibraryMember		yes
Week	*measure*	no
MemberOfLibrary	*repeat of LibraryMember*	no
Item	*book or journal*	yes (?)
Time	*abstract term*	no
MemberOfStaff		yes
System	*general term*	no
Rule	*general term*	no

Relationship between Classes



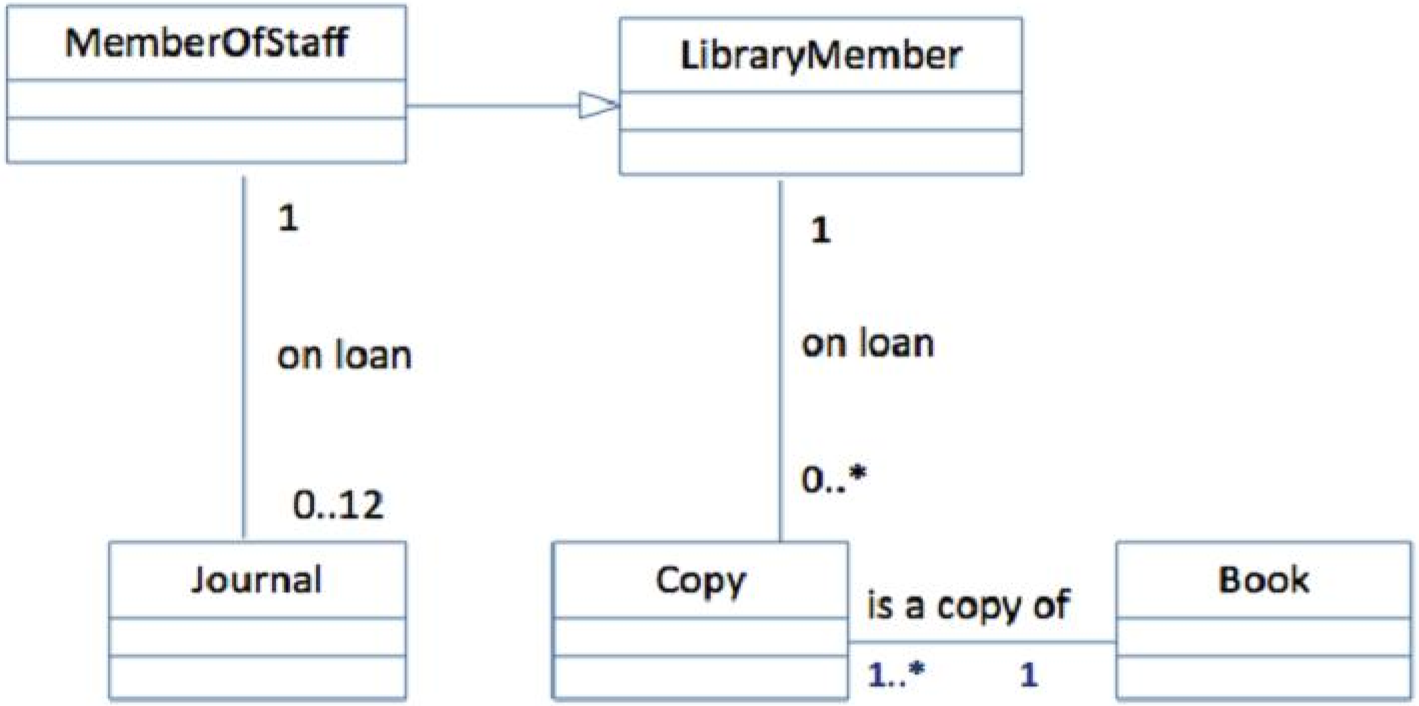
Book	is an	Item
Journal	is an	Item
Copy	is a copy of a	Book
LibraryMember		
Item		
MemberOfStaff	is a	LibraryMember

Methods



LibraryMember	borrows	Copy
LibraryMember	returns	Copy
MemberOfStaff	borrows	Journal
MemberOfStaff	returns	Journal

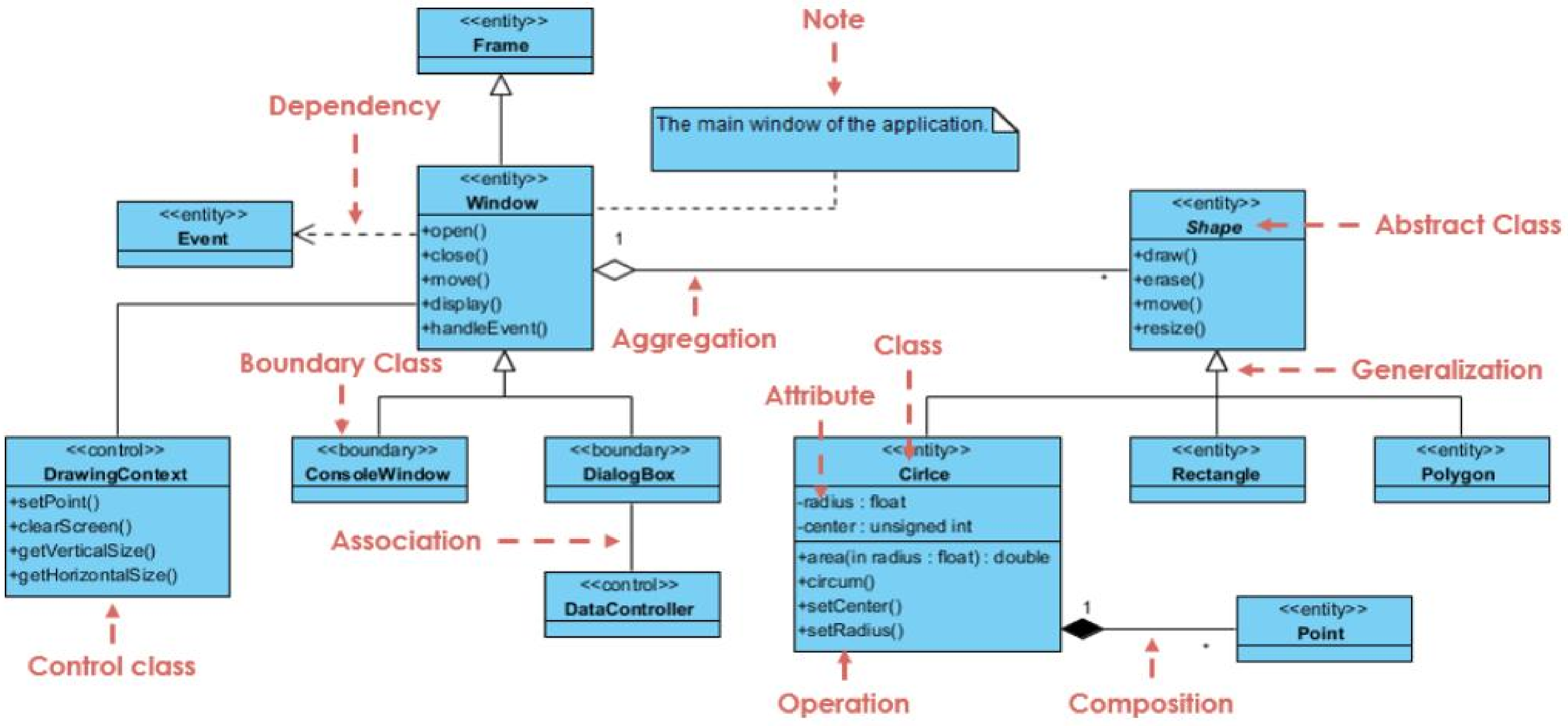
Possible Class Diagram



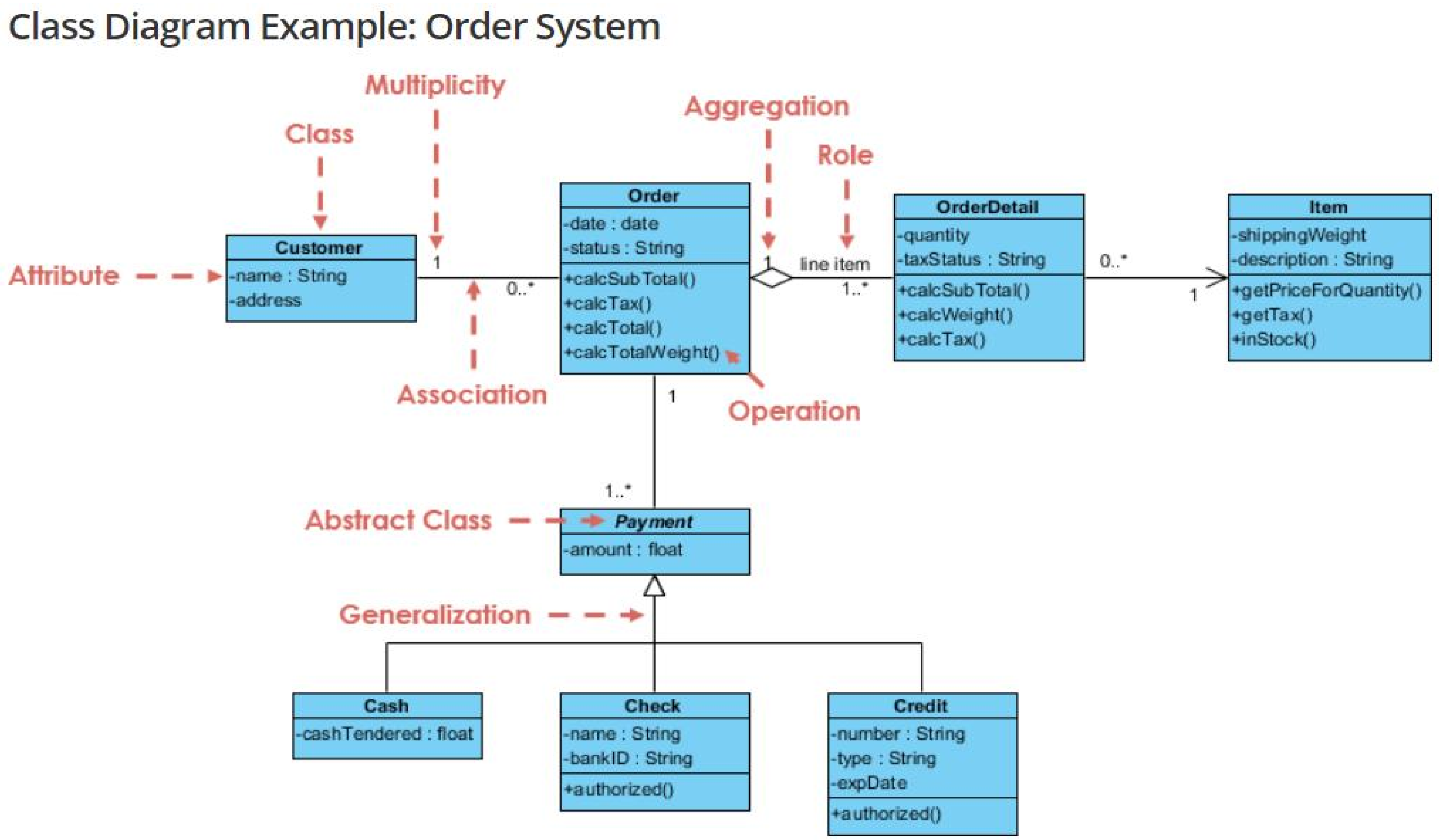
Class Diagram: Example-1

Class Diagram - Diagram Tool Example

A class diagram may also have notes attached to classes or relationships. Notes are shown in grey.



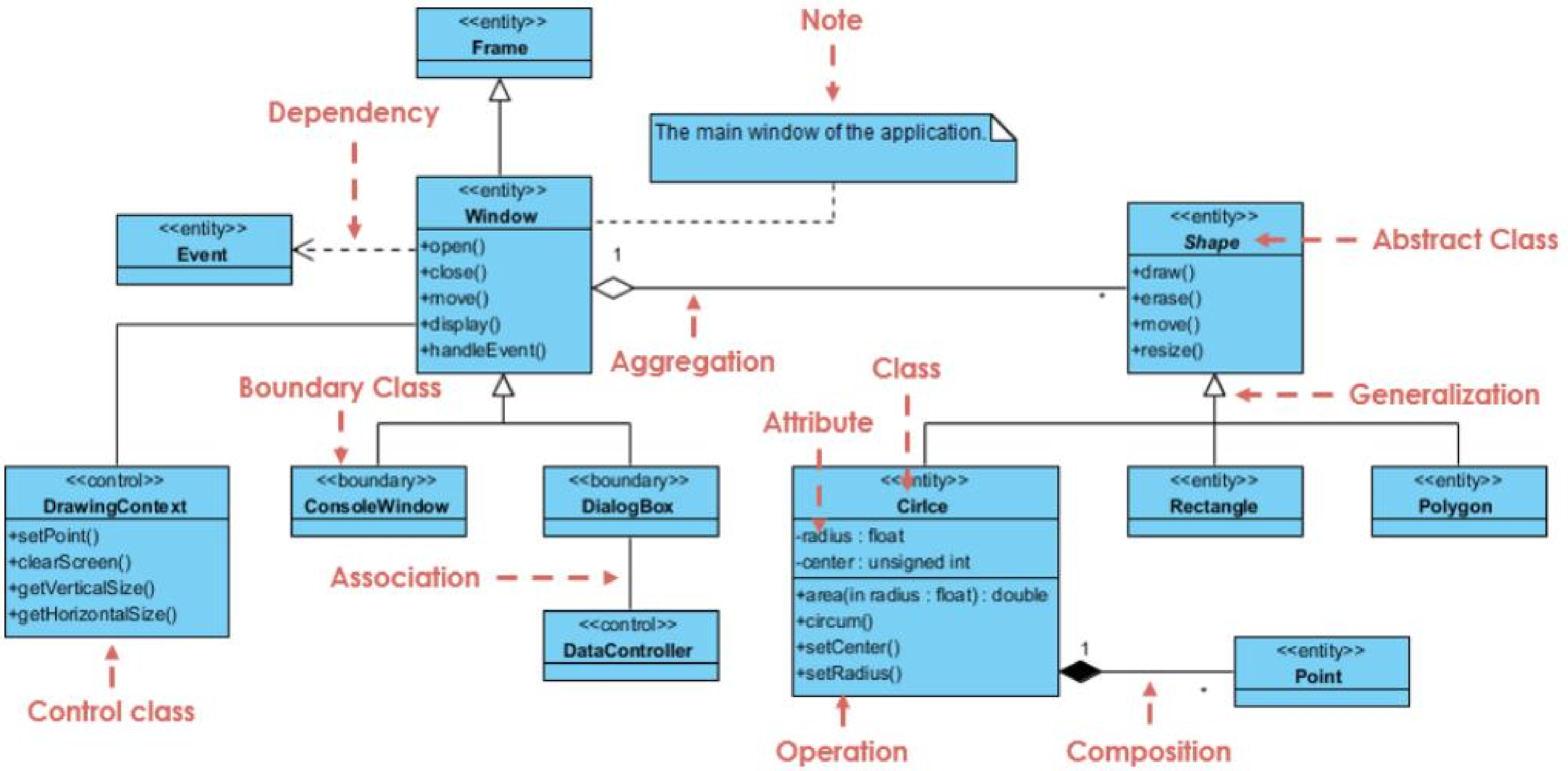
Class Diagram: Example-2



Class Diagram: Example-3

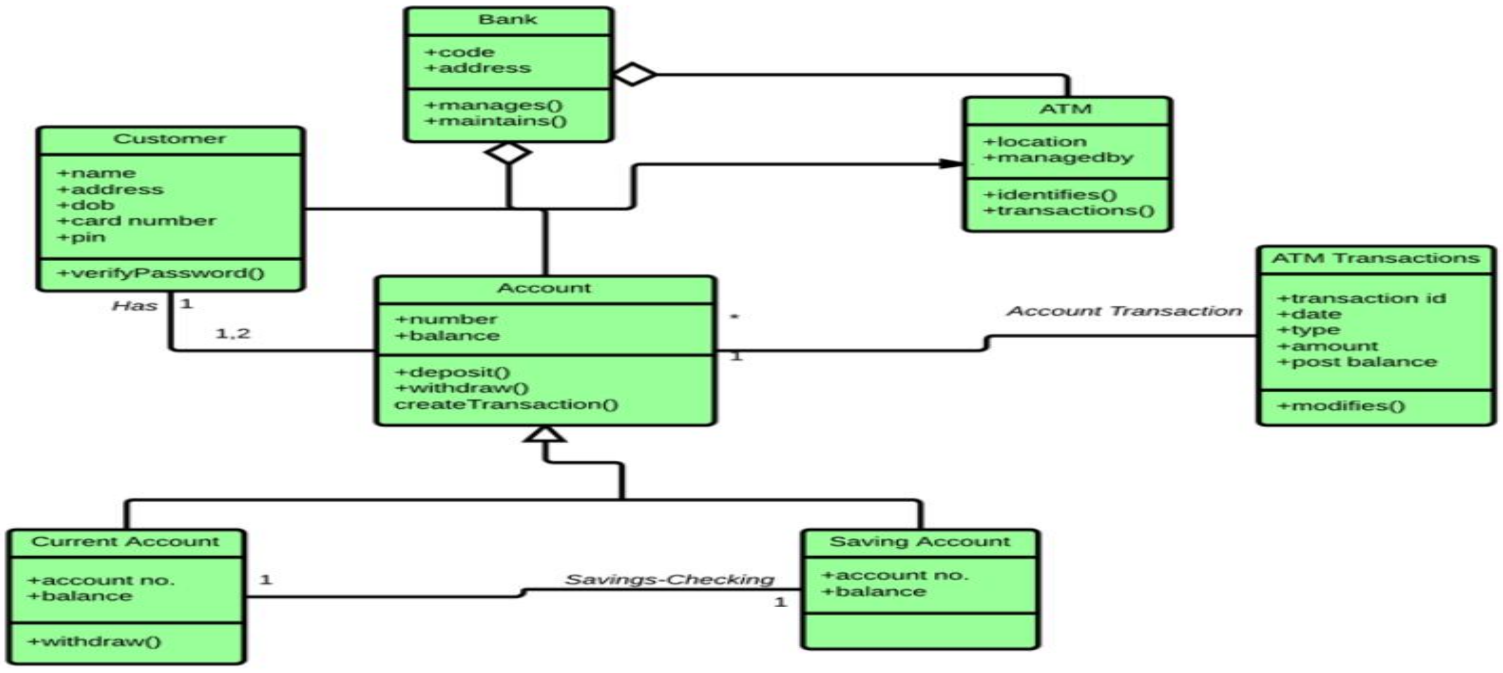
Class Diagram Example: GUI

A class diagram may also have notes attached to classes or relationships.



Class Diagram: Example-4

Creating a class diagram - Example: ATMs system is very simple as customers need to press some buttons to receive cash. However, there are multiple security layers that any ATM system needs to pass. This helps to prevent fraud and provide cash or need details to banking customers.



UML Class Diagram Example

- Class DIAGRAM detail learning with Examples - usefull links
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- https://creately.com/blog/diagrams/class-diagram-tutorial/
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
- https://www.lucidchart.com/pages/uml-class-diagram
- https://www.tutorialspoint.com/uml/uml_class_diagram.htm
- https://www.smartdraw.com/class-diagram/
- https://www.guru99.com/uml-class-diagram.html
- https://www.javatpoint.com/uml-class-diagram
-

Till MidTerm Fall-2022

SEQUENCE DIAGRAM

Sequence Diagram Tutorial - Complete Guide with Examples (creately.com)

https://creately.com/blog/diagrams/sequence-diagram-tutorial/

What is Sequence Diagram?

- Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.
- In simpler words, a sequence diagram shows different parts of a system work in a ‘sequence’ to get something done.

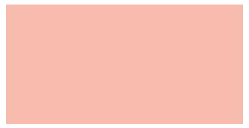
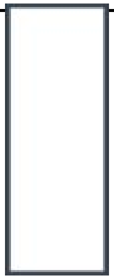
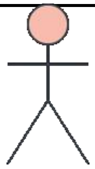
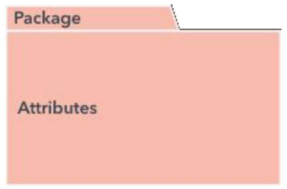
Sequence Diagram

- **A sequence diagram** shows the sequence of interactions that take place during a particular use case.
- **Sequence diagrams** are a kind of interaction diagram, because they describe how—and in what order—a group of objects works together
- **Used by software developers and business people** alike to <u>understand requirements for a new system</u> or <u>to document an existing process</u>

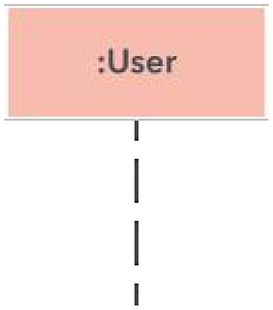
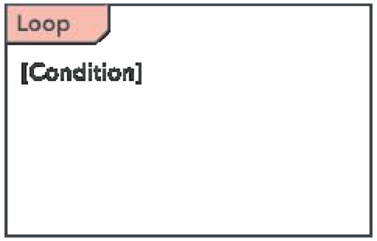
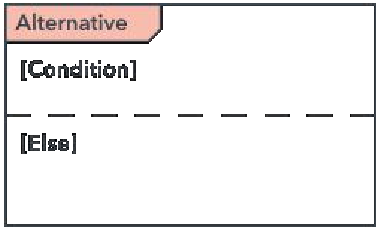
Sequence Diagram Applications

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how tasks are moved between objects or components of a process.
- Plan and understand the detailed functionality of an existing or future scenario.

Sequence Diagram Components

	**Object Symbol** This box shape represents a class, or object, in UML. They demonstrate how an object will behave in the context of the system. Class attributes should not be listed in this shape
	**Activation Box** Symbolized by a rectangle shape, an activation box represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	**Actor Symbol** Represented by a stick figure, actors are entities that are both interactive with and external to the system
	**Package Symbol** Also known as a frame, this is a rectangle shape that is used in UML 2.0 notation to contain interactive elements of the diagram. The shape has a small inner rectangle for labeling the diagram.

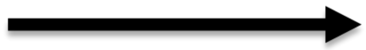
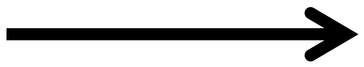
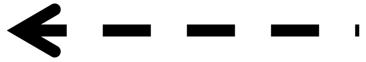
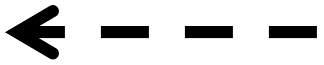
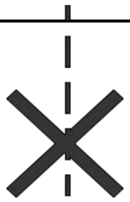
Sequence Diagram Components

	**Lifeline Symbol** A dashed vertical line that represents the passage of time as it extends downward. Along with time, they represent the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.
	**Option Loop Symbol** A rectangle shape with a smaller label within it. This symbol is used to model "if then" scenarios, i.e., a circumstance that will only occur under certain conditions.
	**Alternative Symbol** Used to symbolize a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside

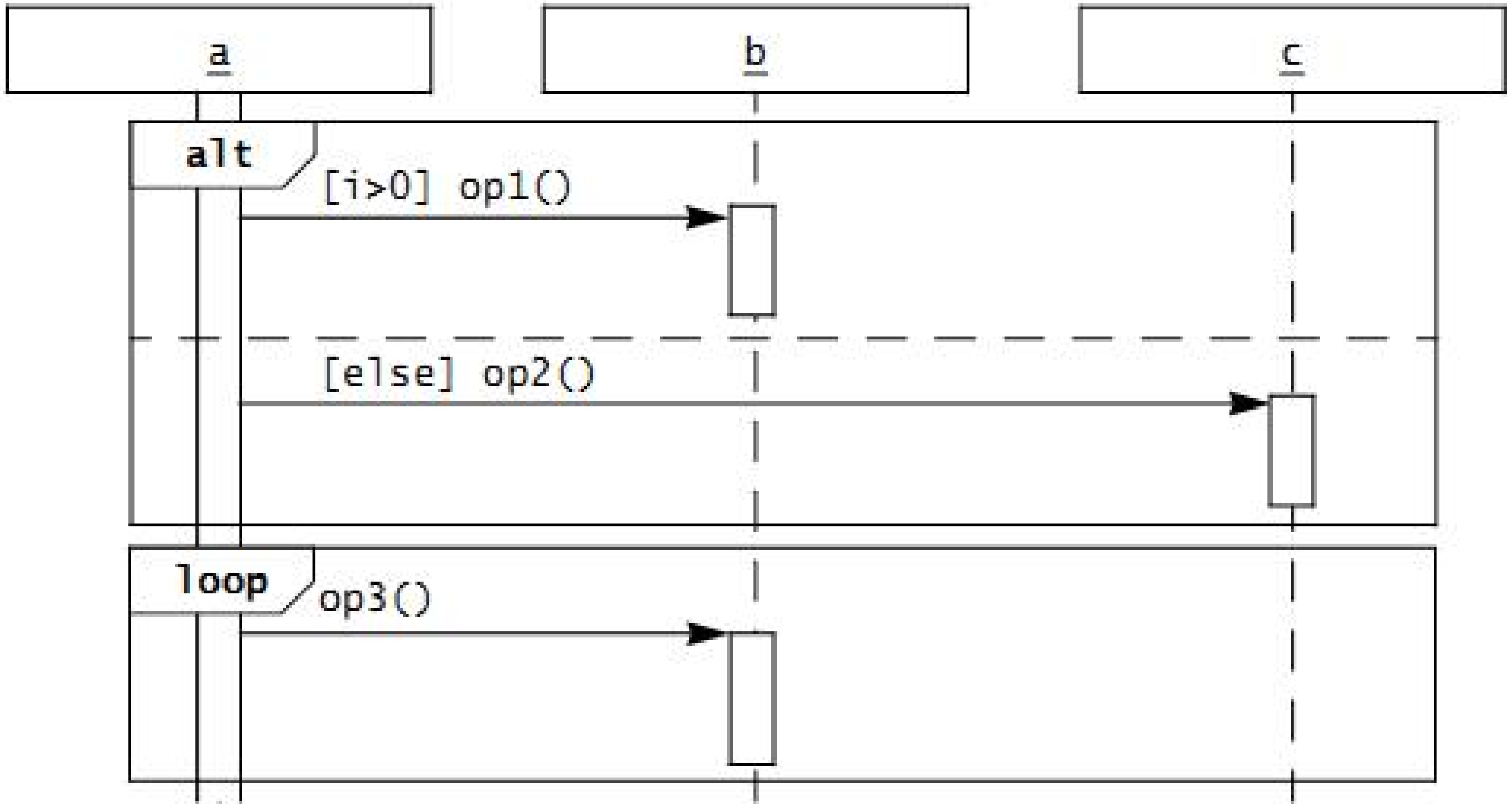
UML Sequence Message Symbols

- Packets of information that are transmitted between objects.
- **They may reflect**
  - Start and execution of an operation, or
  - Sending and reception of a signal.

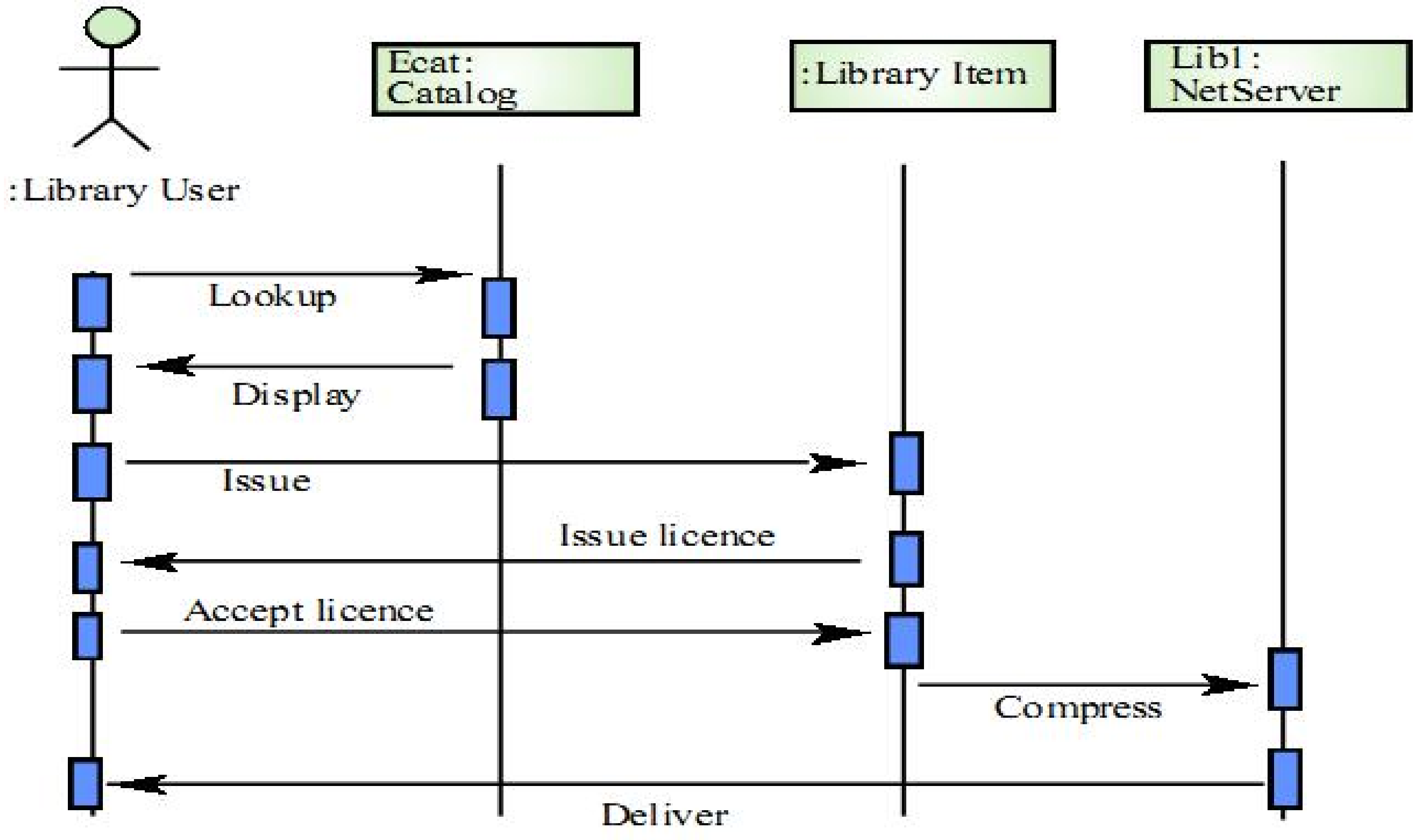
UML Sequence Message Symbols

	**Synchronous Message Symbol** This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	**Asynchronous Message Symbol** Asynchronous messages are those that don't require a response before the sender continues. Only the call should be included in the diagram.
	**Asynchronous Return Message Symbol** Represented by a dashed line with a lined arrowhead.
	**Reply Message Symbol** Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	**Delete Message Symbol** Represented by a solid line with a solid arrowhead, followed by an X symbol. This messages indicates the destruction of an object and is placed in its path on the lifeline

UML Sequence Components

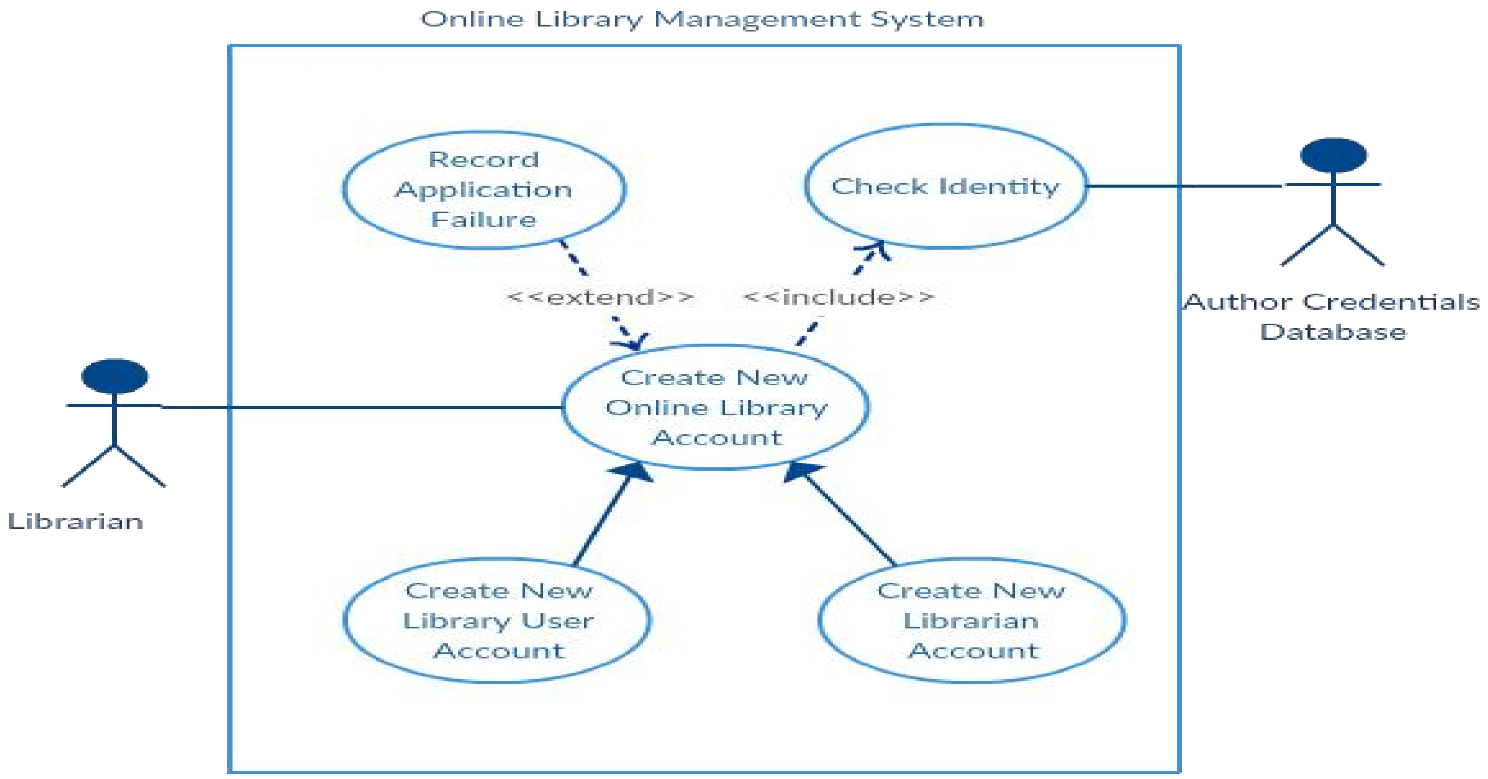


Sequence Diagram: Example-1



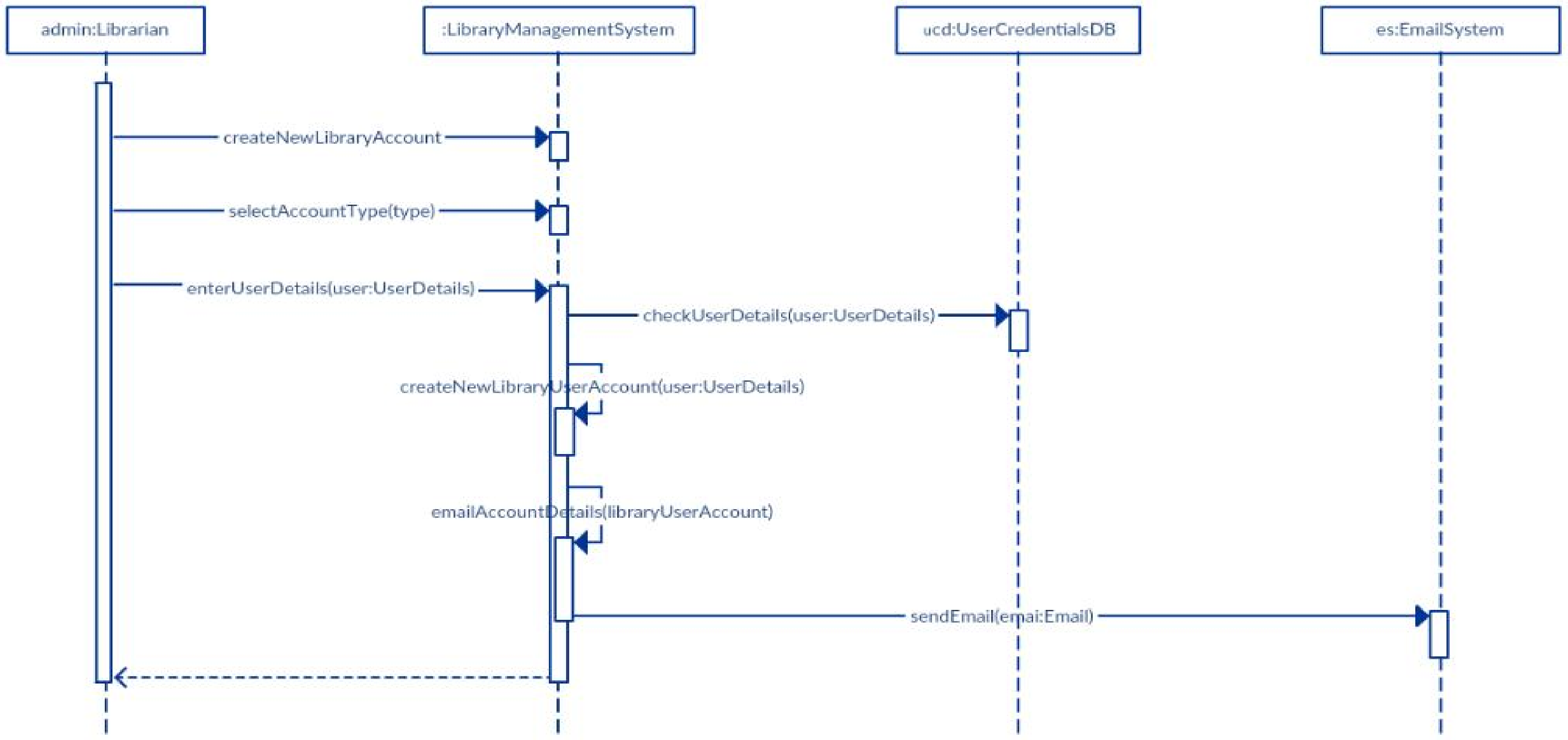
Sequence Diagram: Example-2 Problem

Creating sequence form use case diagram:

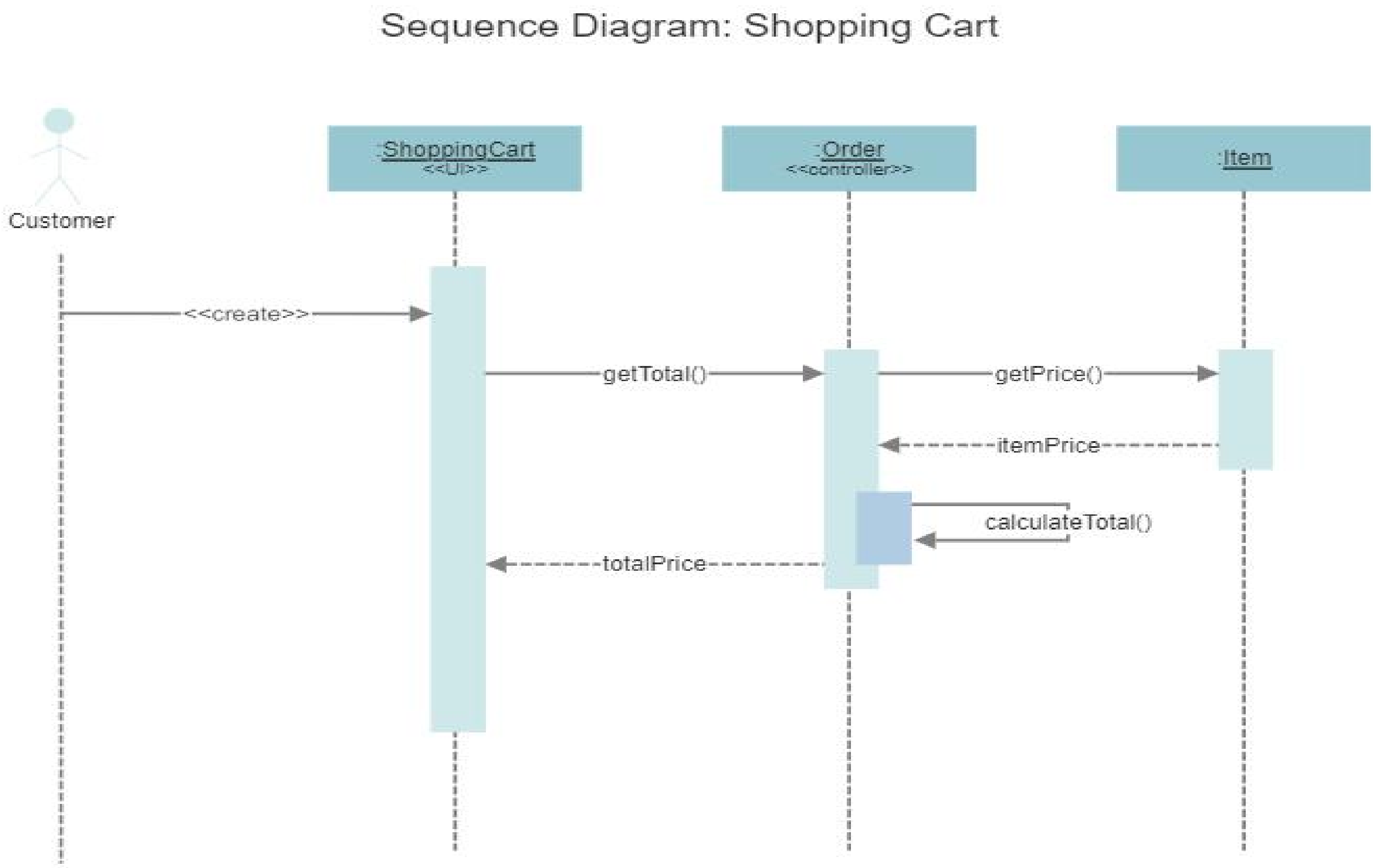

Online Library Management System
Record Application Failure
Check Identity
<<extend>>
<<include>>
Create New Online Library Account
Author Credentials Database
Librarian
Create New Library User Account
Create New Librarian Account

Sequence Diagram: Example-2 Solution

The sequence diagram below shows how the objects in the online library management system interact with each other to perform the function 'Create New Library User Account'.


admin:Librarian
:LibraryManagementSystem
ucd:UserCredentialsDB
es:EmailSystem
createNewLibraryAccount
selectAccountType(type)
enterUserDetails(user:UserDetails)
checkUserDetails(user:UserDetails)
createNewLibraryUserAccount(user:UserDetails)
emailAccountDetails(libraryUserAccount)
sendEmail(emai:Email)

Sequence Diagram: Example-3



Structural Model

Structural Models

- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- **Structural models may be**
  - Static Models—structure of the system
  - Dynamic Models-- structure of the system when executing
- You create structural models of a system when you are discussing and designing the system architecture.

BEHAVIORAL MODELS

Behavioral Models

- **Behavioral models** are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - **Data** Some data arrives that has to be processed by the system.
  - **Events** Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

Event Driven Modeling

- **Event driven modeling shows** how a system responds to external and internal events.
- **It is based on the assumption that a system** has a finite number of states and events may cause a transition from one state to another.
- **This view is particularly appropriate** when reasoning about real-time systems.
  - **For example** a system controlling a valve may move from a state “Valve open” to “Valve close” when a command(stimulus) is received from operator.

STATE DIAGRAM

[What is Class Diagram?](#)
Read for details

State Machine Diagram Tutorial | Lucidchart
https://www.lucidchart.com/pages/uml-state-machine-diagram

State Diagram

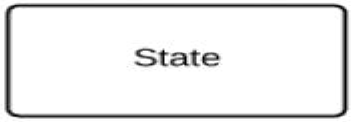
- UML supports event-based modeling using state diagrams, which were based on state charts.
- State diagrams show system states and events that cause transition from one state to another.
- **They DONOT show the flow of data within the system**
- **May include additional** information on the computation carried out in each state.

State Diagram

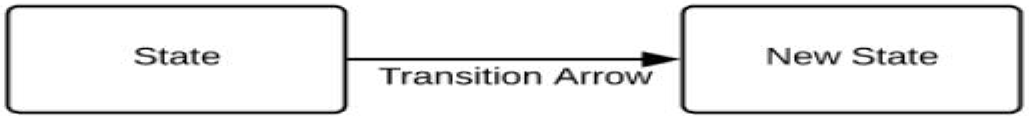
- A state diagram is also called **as state machine or state chart diagrams**, depict the dynamic behavior of the system in response to external and internal events.

State Diagram

- **State diagrams mainly depict states and transitions.**
- States are represented with rectangles with rounded corners that are labeled with the name of the state.



- Transitions are marked with arrows that flow from one state to another, showing how the states change.



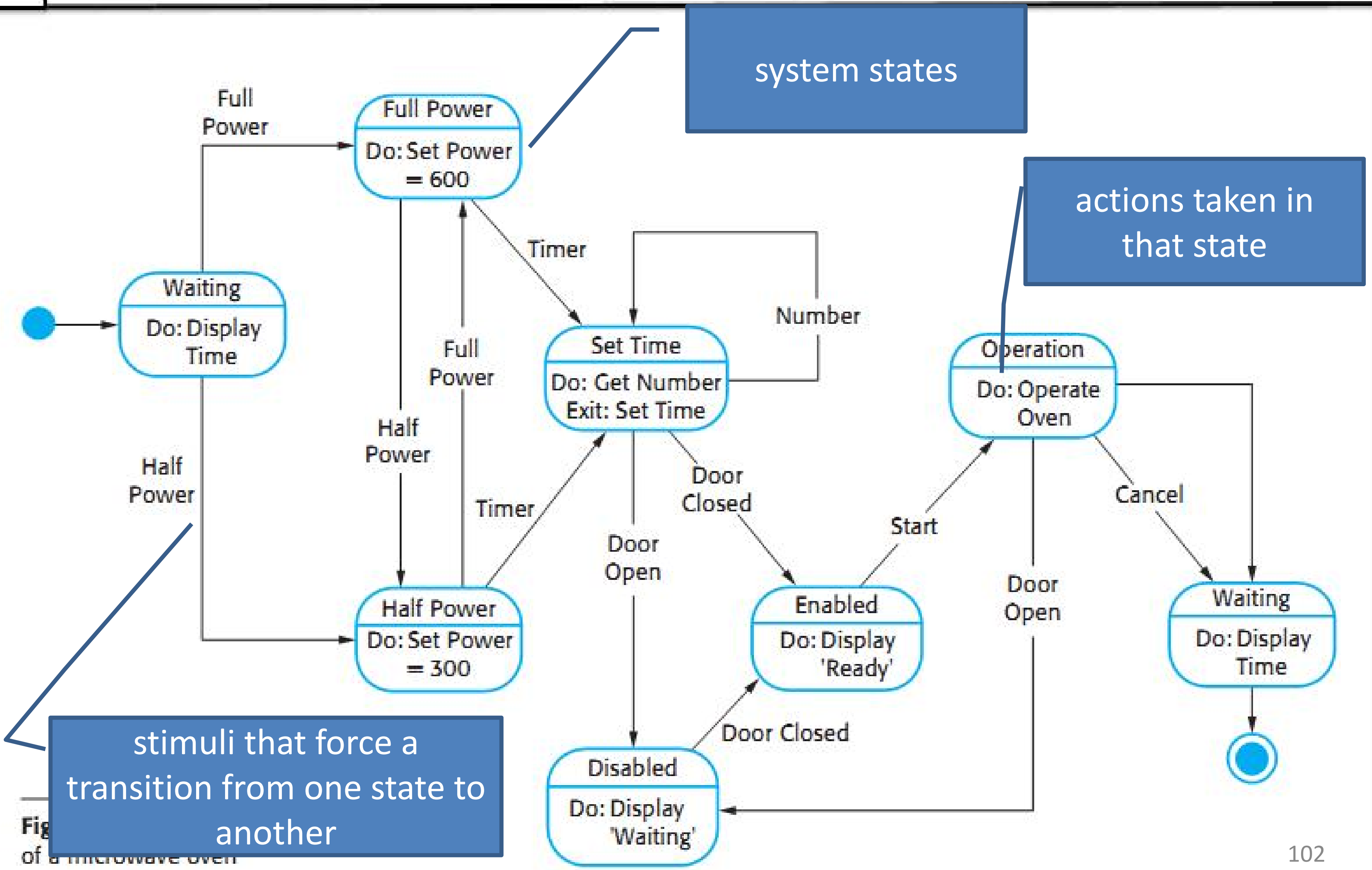
State Diagram

- UML supports event-based modeling using state diagrams, which were based on state charts.
- State diagrams show system states and events that cause transition from one state to another.
- **They DONOT show the flow of data within the system**
- **May include additional** information on the computation carried out in each state.

State Diagram Example

- **Control System for a microwave oven**
  - Has a switch to select full or half power
  - Numeric keypad to enter cooking time
  - Start/stop button
  - Alpha numeric display
- **Sequence:**
  - Select power level (with half or full)
  - Input the cooking time using the numeric keypad
  - Start pressed and the food cooked for the given time

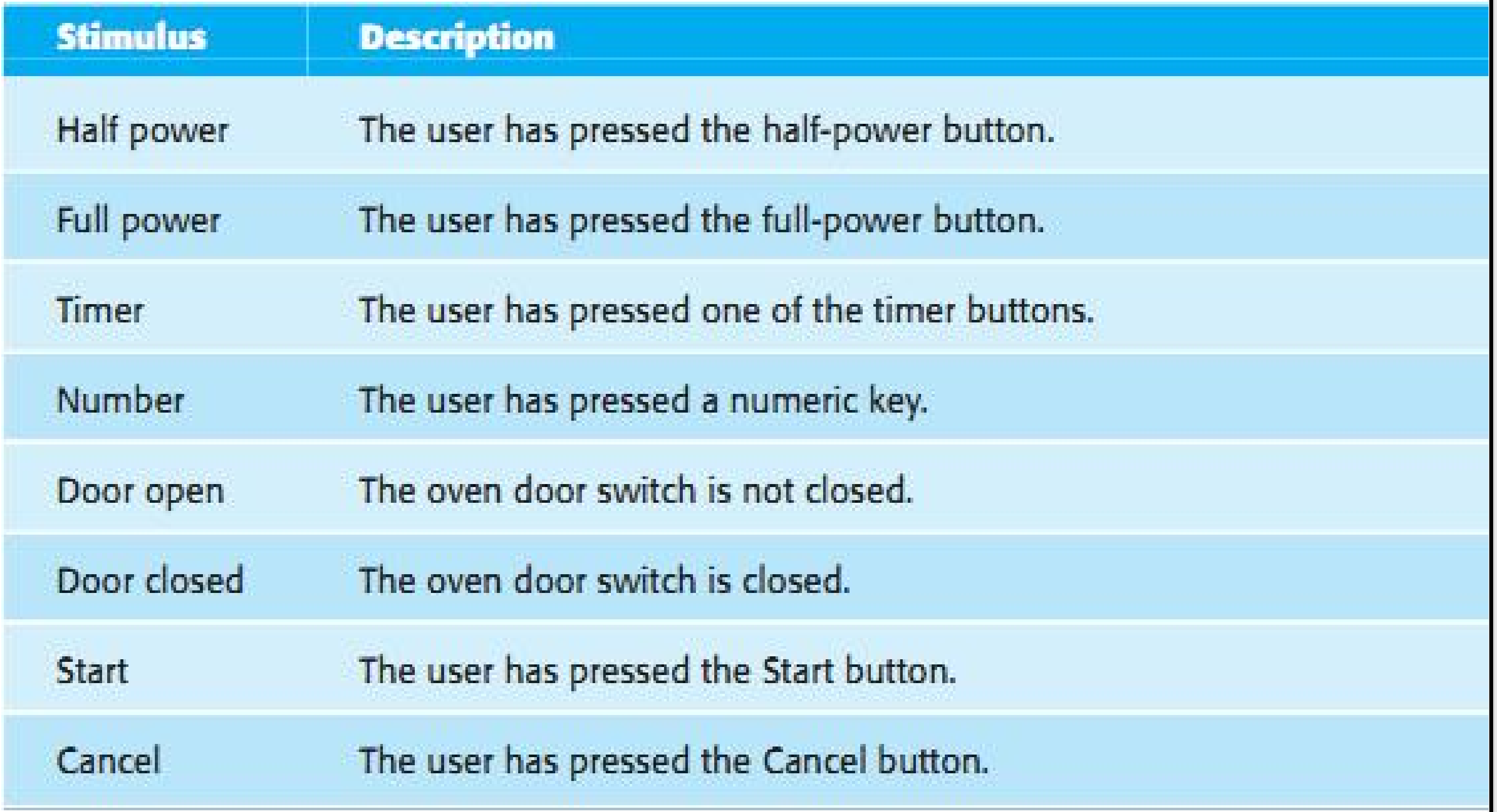
State Diagram Example-1



States for microwave Oven

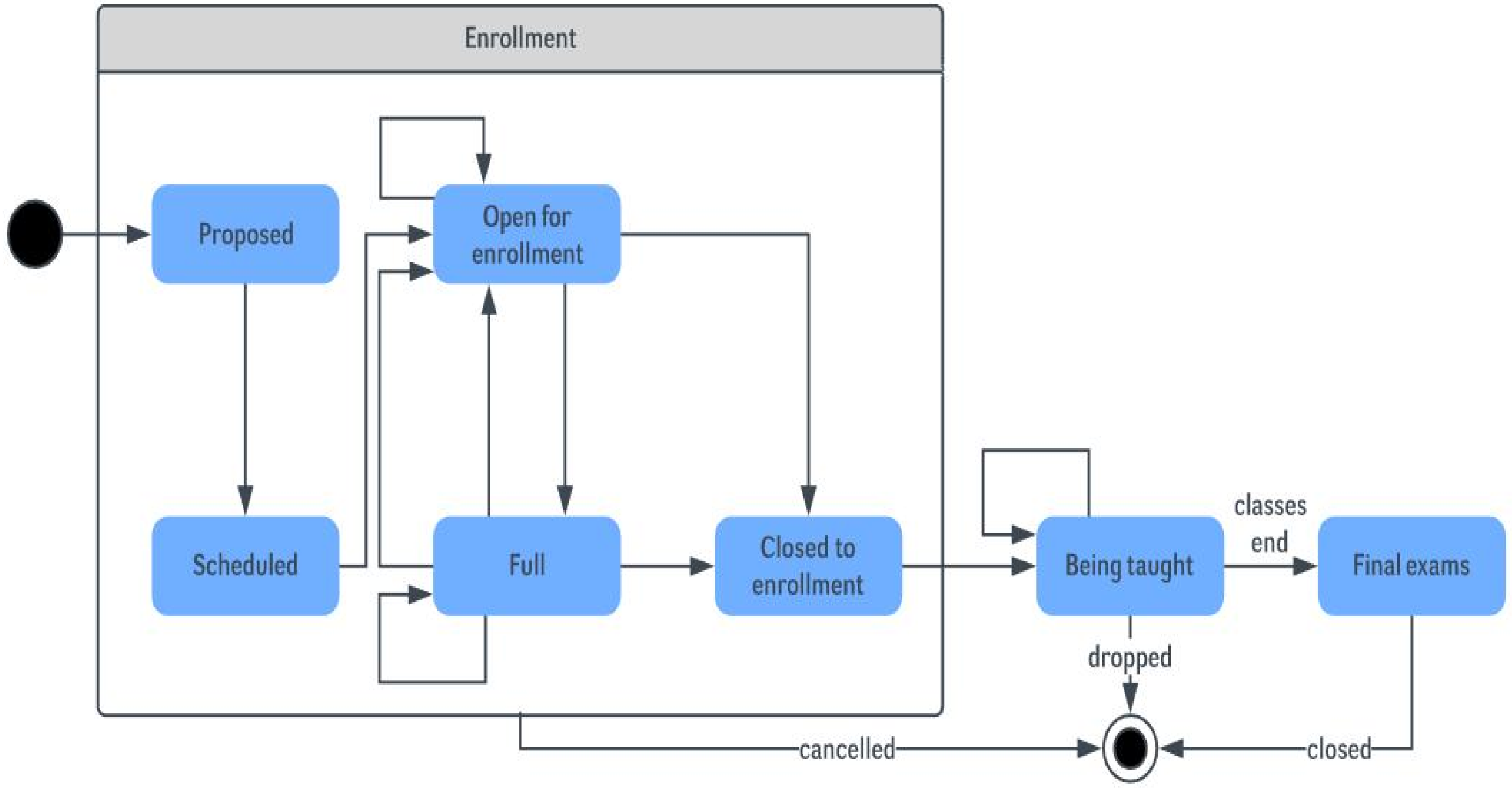
State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

Stimulus for microwave Oven

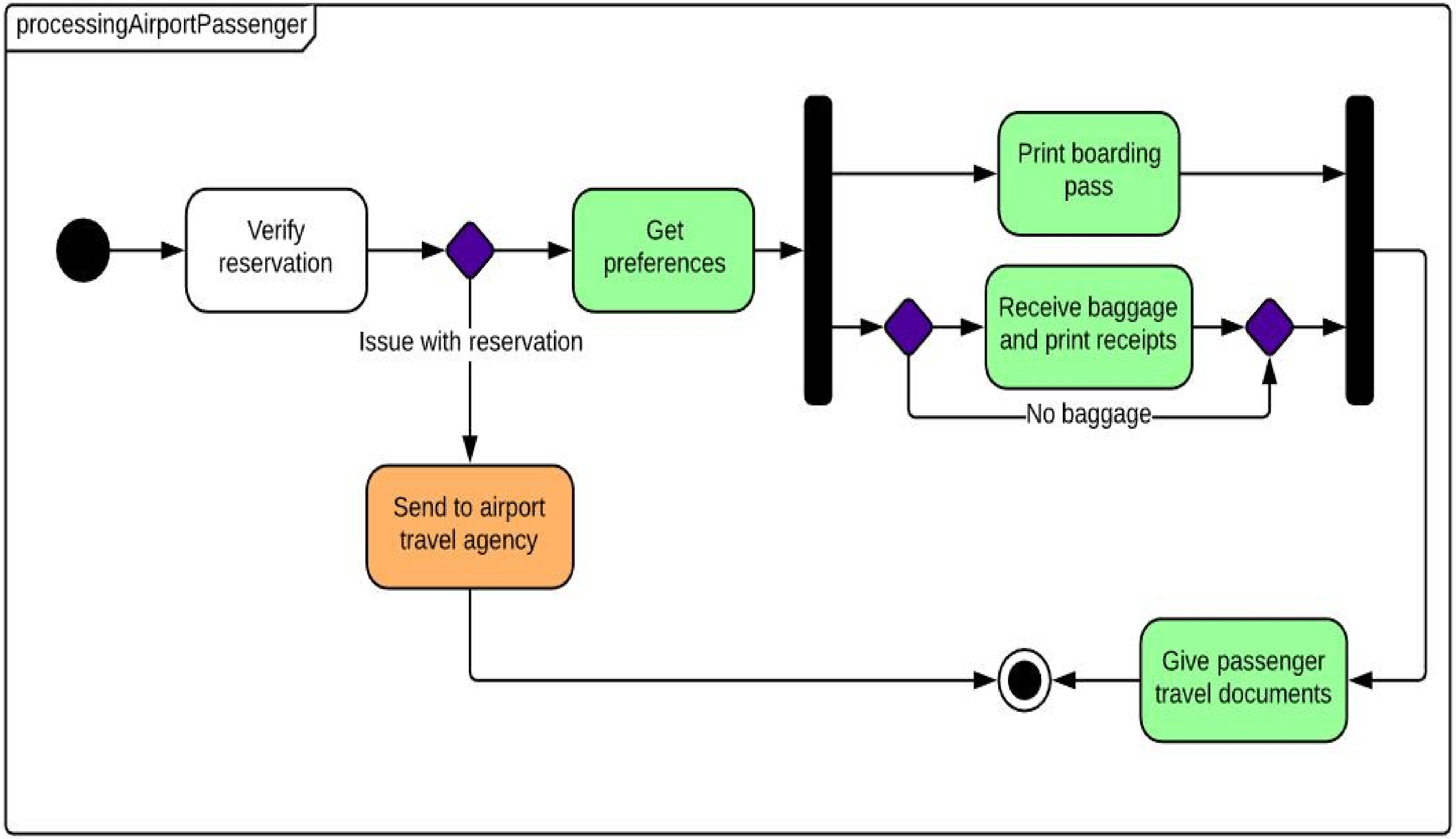


Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

State Diagram Example-2



State Diagram Example-3


processingAirportPassenger
Verify reservation
Issue with reservation
Get preferences
Send to airport travel agency
Print boarding pass
Receive baggage and print receipts
No baggage
Give passenger travel documents

Data Processing Models

DFD

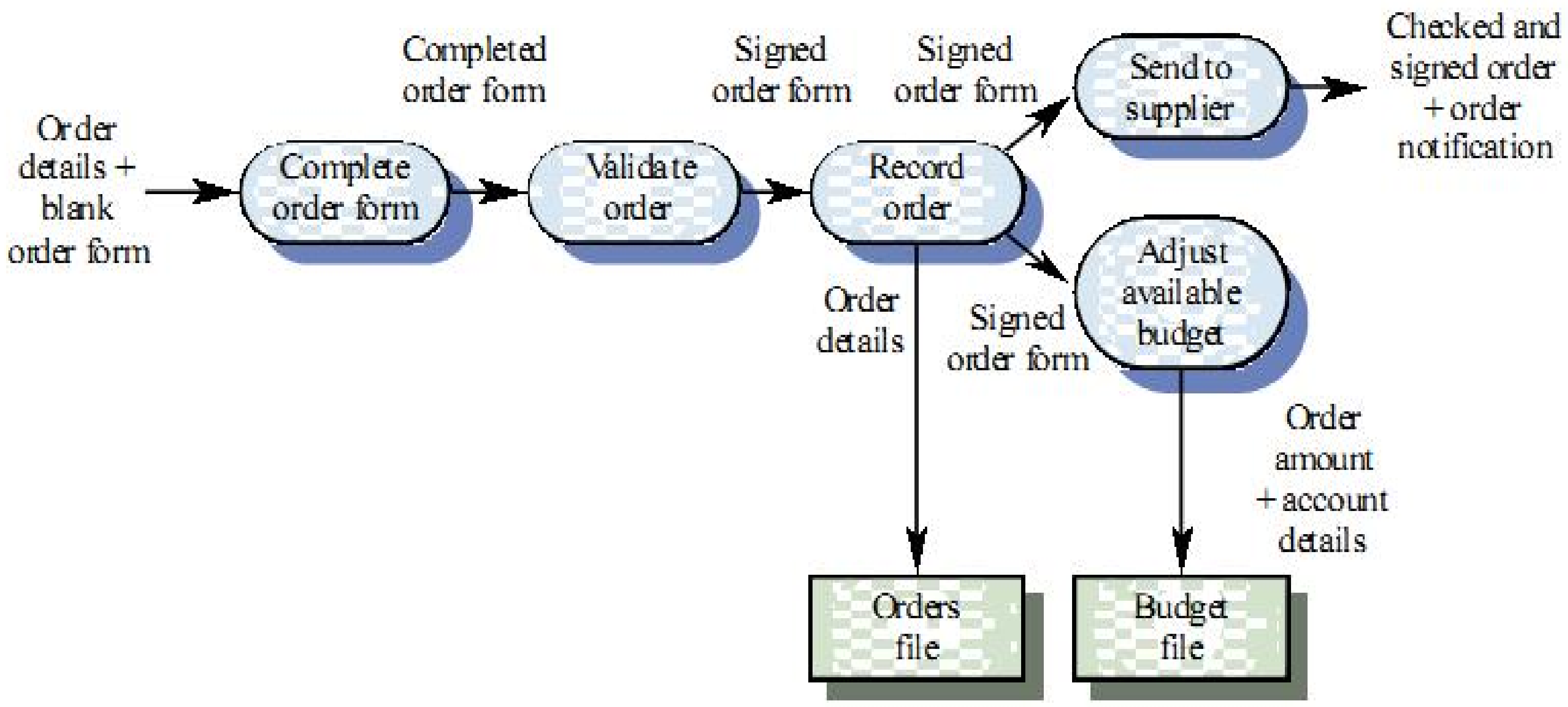
What is a Data Flow Diagram | Lucidchart

HTTPS://WWW.LUCIDCHART.COM/PAGES/DATA-FLOW-DIAGRAM

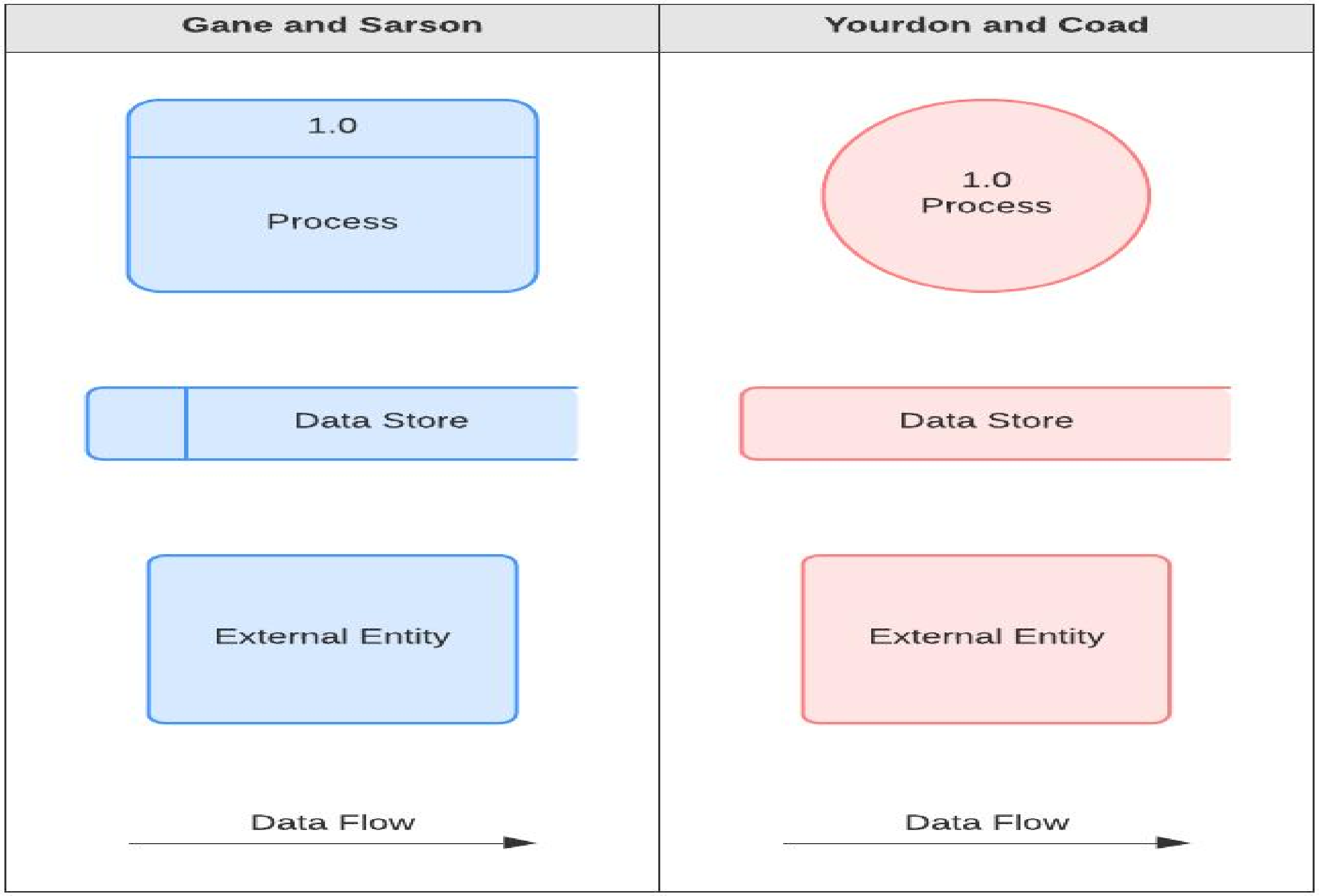
Data Processing Models

- Data flow diagrams (DFDs) are used to model the system's data processing
- **These show the processing steps** as data flows through a system
- **Fundamental part** of many analysis methods
- **Simple and intuitive notation** that customers can understand
- **Show end-to-end** processing of data

Order Processing DFD



DFD


Gane and Sarson
Yourdon and Coad
1.0
Process
1.0
Process
Data Store
Data Store
External Entity
External Entity
Data Flow
Data Flow

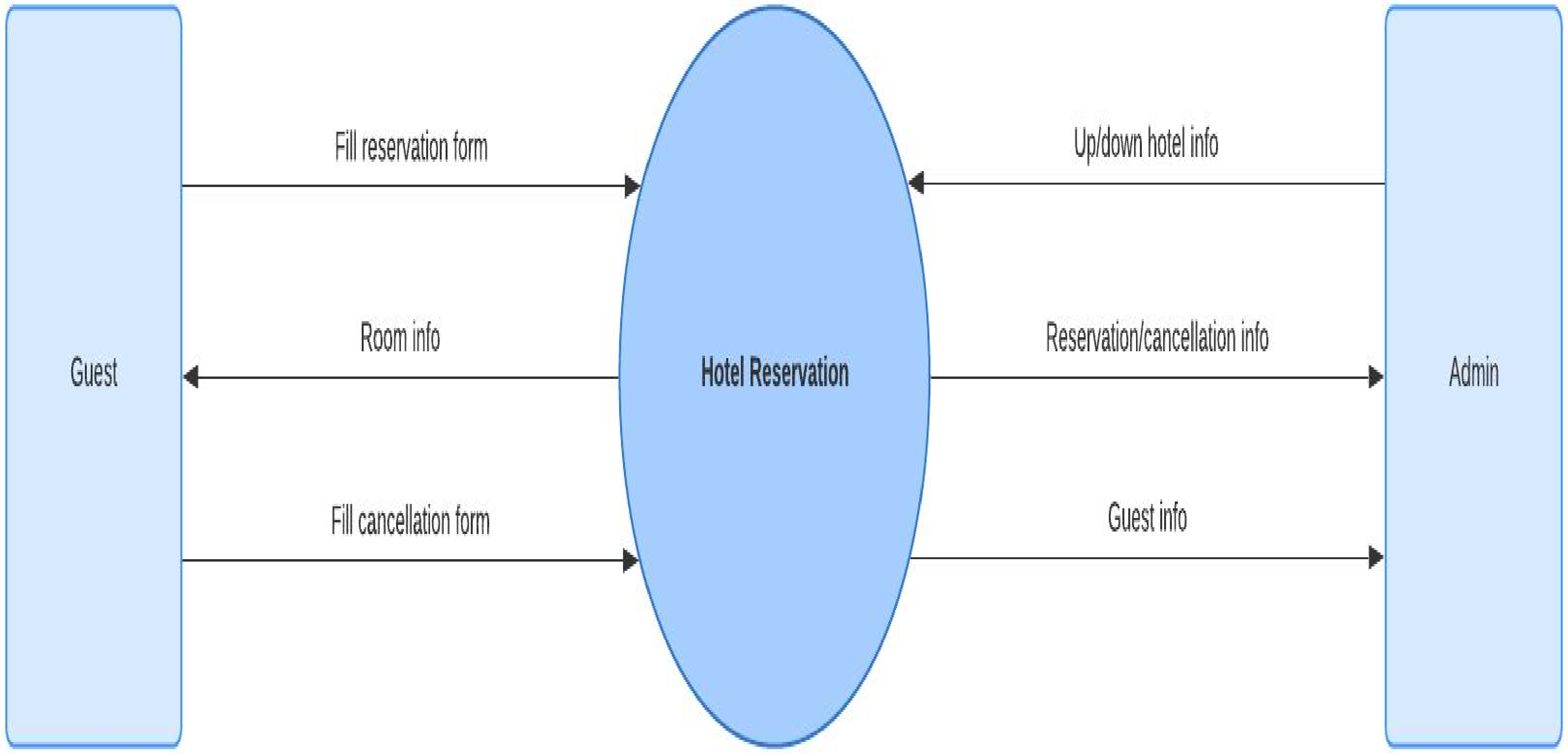
DFD

- **DFD rules and tips**
- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

DFD – level-0 (Context Diagram)

DFD Level 0 is also called a Context Diagram.
It’s a basic overview of the whole system or process being analyzed or modeled.
It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.
It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

DFD – level-0 (Context Diagram)

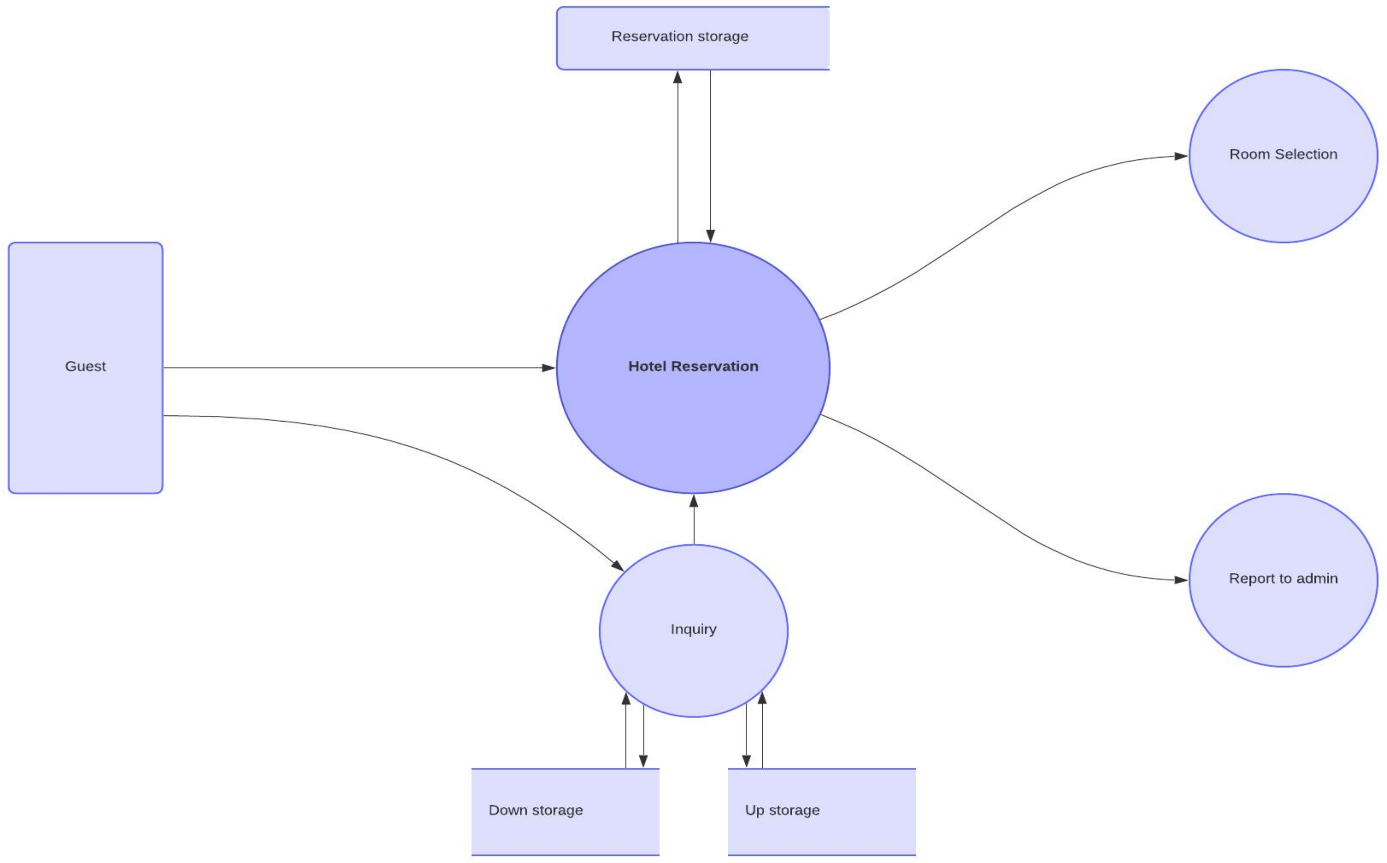


DFD – level-1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram.

You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

DFD – level-1



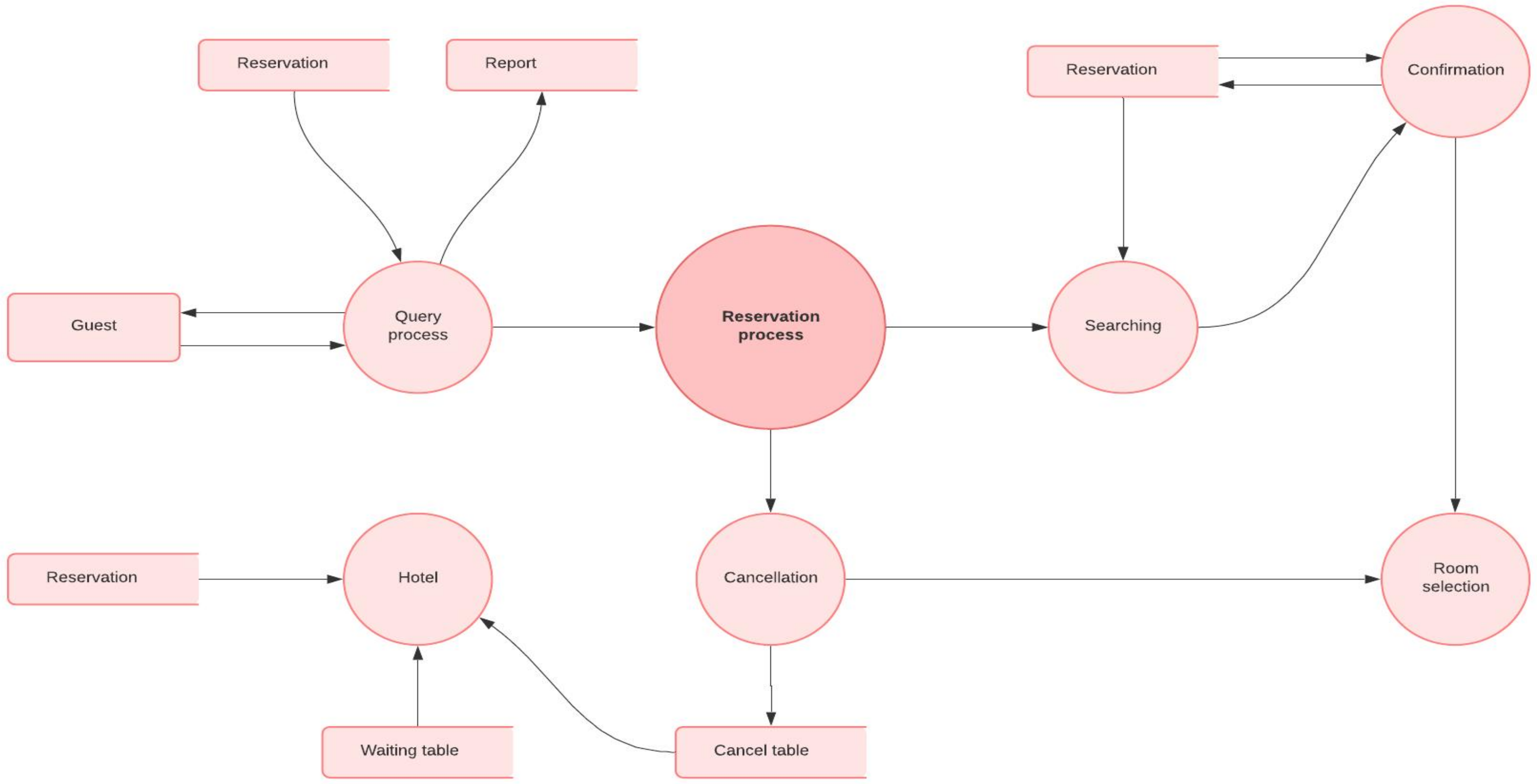
Reservation storage
Room Selection
Guest
Hotel Reservation
Report to admin
Inquiry
Down storage
Up storage

DFD – level-2

DFD Level 2 then goes one step deeper into parts of Level 1.

It may require more text to reach the necessary level of detail about the system’s functioning.

DFD – level-2


Reservation
Report
Reservation
Confirmation
Guest
Query process
Reservation process
Searching
Reservation
Hotel
Cancellation
Room selection
Waiting table
Cancel table

Logical DFD vs. Physical DFD

- These are the two categories of a data flow diagram.
- **A Logical DFD** visualizes the data flow that is essential for a business to operate. It focuses on the business and the information needed, not on how the system works or is proposed to work.
- However, **a Physical DFD** shows how the system is actually implemented now, or how it will be.
- **For example**, in a Logical DFD, the processes would be business activities, while in a Physical DFD, the processes would be programs and manual procedures.

When to use DFD

- **DFDs model the system** from a functional perspective
- **Tracking and documen**ting how the data associated with a process is helpful to develop an overall understanding of the system
- **Data flow diagrams** may also be used in showing the data exchange between a system and other systems in its environment

Thank you