*Structural UML diagrams

Structural UML diagrams on the other hand focus on depicting the concepts involved in a system and how they relate to each other. There are also 7 types of structural UML diagrams:

1. Class Diagram
2. Component Diagram
3. Deployment Diagram
4. Object Diagram
5. Package Diagram
6. Profile Diagram
7. Composite Structure Diagram

*Behavioral UML diagrams

Behavioral UML diagrams provide a standard way to visualize the design and behavior of a system. Under them are 7 other types of diagrams which are:

1. Activity diagrams
2. State machine diagrams
3. Sequence diagrams
4. Communication diagrams
5. Interaction overview diagrams
6. Timing diagrams
7. Use case diagrams

*E-R Diagram

The Flight Database

The flight database stores details about an airline’s fleet, flights, and seat bookings. Again, it’s a hugely simplified version of what a real airline would use, but the principles are the same.

Consider the following requirements list:

- The airline has one or more airplanes.
- An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- Each flight is carried out by a single airplane.
- A passenger has given names, a surname, and a unique email address.
- A passenger can book a seat on a flight.

The ER diagram derived from our requirements is shown in Figure below:

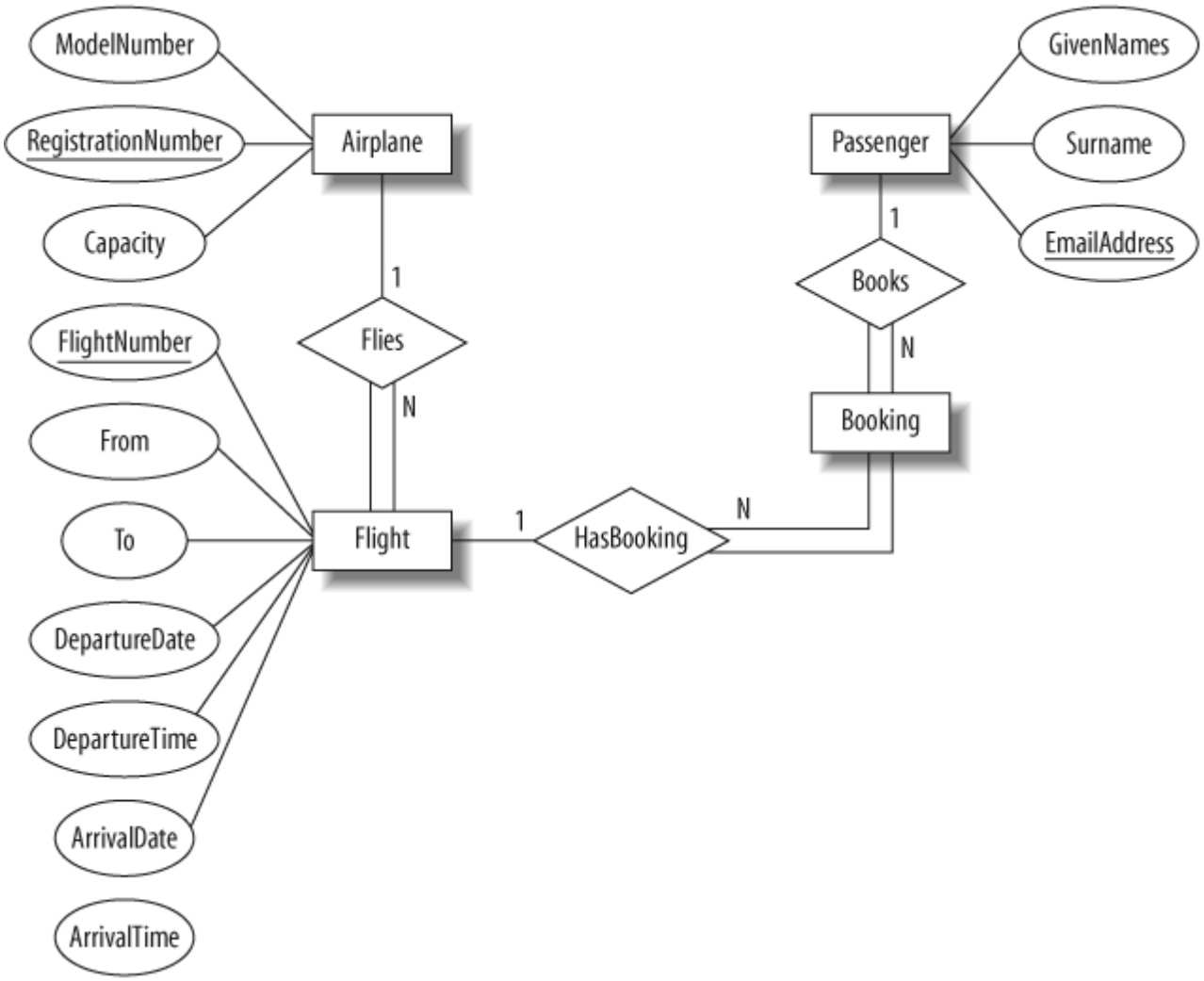


Figure:. The ER diagram of the flight database

- An `Airplane` is uniquely identified by its `RegistrationNumber`, so we use this as the primary key.
- A `Flight` is uniquely identified by its `FlightNumber`, so we use the flight number as the primary key. The departure and destination airports are captured in the `From` and `To` attributes, and we have separate attributes for the departure and arrival date and time.
- Because no two passengers will share an email address, we can use the `EmailAddress` as the primary key for the `Passenger` entity.
- An airplane can be involved in any number of flights, while each flight uses exactly one airplane, so the `Flies` relationship between the `Airplane` and `Flight` relationships has cardinality 1:N; because a flight cannot exist without an airplane, the `Flight` entity participates totally in this relationship.
- A passenger can book any number of flights, while a flight can be booked by any number of passengers.
- As discussed earlier in Intermediate Entities," we could specify an M:N `Books` relationship between the `Passenger` and `Flight` relationship, but considering the issue more carefully shows that there is a hidden entity here: the booking itself. We capture this by creating the intermediate entity `Booking` and 1:N relationships between it and the `Passenger` and `Flight` entities. Identifying such entities allows us to get a better picture of the requirements. Note that even if we didn't notice this hidden entity, it would come out as part of the ER-to-tables mapping process we'll describe next in Using the Entity Relationship Model."

*The University Database

The `university` database stores details about university students, courses, the semester a student took a particular course (and his mark and grade if

he completed it), and what degree program each student is enrolled in. The database is a long way from one that'd be suitable for a large tertiary institution, but it does illustrate relationships that are interesting to query, and it's easy to relate to when you're learning SQL. We explain the requirements next and discuss their shortcomings at the end of this section.

Consider the following requirements list:

- The university offers one or more programs.
- A program is made up of one or more courses.
- A student must enroll in a program.
- A student takes the courses that are part of her program.
- A program has a name, a program identifier, the total credit points required to graduate, and the year it commenced.
- A course has a name, a course identifier, a credit point value, and the year it commenced.
- Students have one or more given names, a surname, a student identifier, a date of birth, and the year they first enrolled. We can treat all given names as a single object—for example, "John Paul."
- When a student takes a course, the year and semester he attempted it are recorded. When he finishes the course, a grade (such as A or B) and a mark (such as 60 percent) are recorded.
- Each course in a program is sequenced into a year (for example, year 1) and a semester (for example, semester 1).

The ER diagram derived from our requirements is shown below .Although it is compact, the diagram uses some advanced features, including relationships that have attributes and two many-to-many relationships.

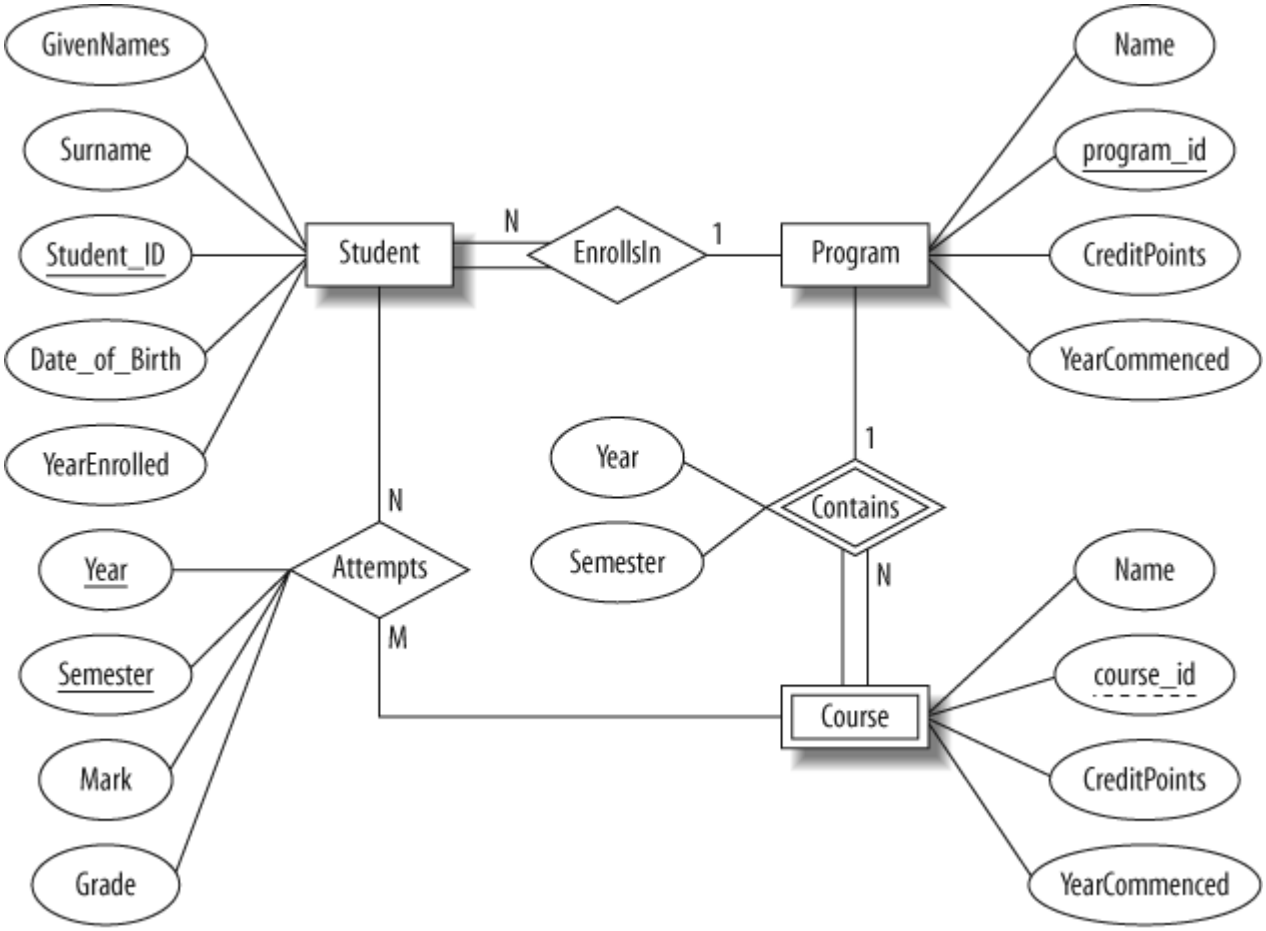


Figure . The ER diagram of the university database

In our design:

- `Student` is a strong entity, with an identifier, `student_id`, created to be the primary key used to distinguish between students (remember, we could have several students with the same name).
- `Program` is a strong entity, with the identifier `program_id` as the primary key used to distinguish between programs.
- Each student must be enrolled in a program, so the `Student` entity participates totally in the many-to-one `EnrollsIn` relationship with `Program`. A program can exist without having any enrolled students, so it participates partially in this relationship.

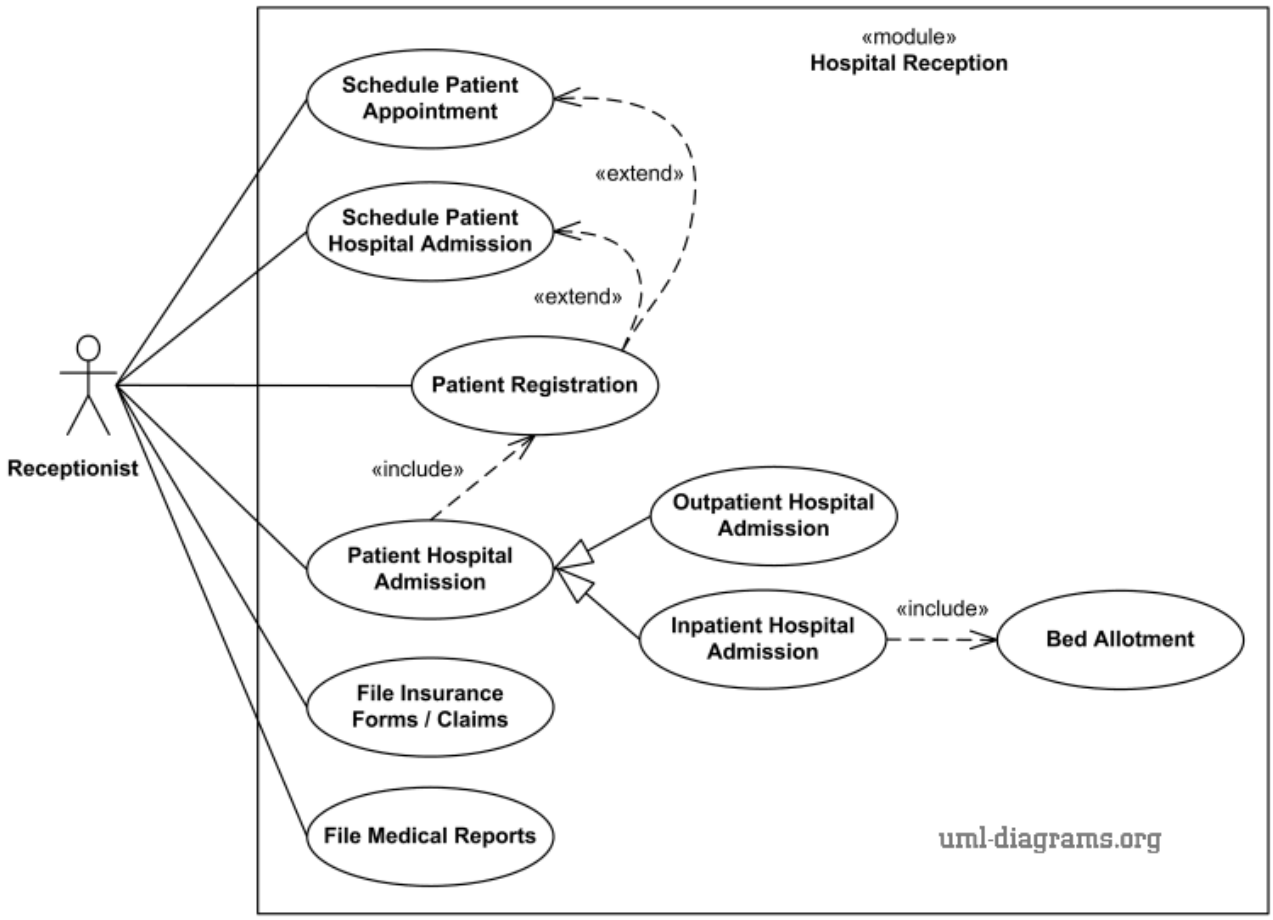
- A `Course` has meaning only in the context of a `Program`, so it's a weak entity, with `course_id` as a weak key. This means that a `Course` is uniquely identified using its `course_id` and the `program_id` of its owning program.
- As a weak entity, `Course` participates totally in the many-to-one identifying relationship with its owning `Program`. This relationship has `Year` and `Semester` attributes that identify its sequence position.
- `Student` and `Course` are related through the many-to-many `Attempts` relationships; a course can exist without a student, and a student can be enrolled without attempting any courses, so the participation is not total.
- When a student attempts a course, there are attributes to capture the `Year` and `Semester`, and the `Mark` and `Grade`.

*USE CASE DIAGRAM

Purpose: *Describe major services (functionality) provided by a hospital's reception.*
Summary*: This UML use case diagram example shows actor and use cases for a hospital's reception.* ***Hospital Reception*** *subsystem or module supports some of the many job duties of a hospital receptionist. Receptionist schedules patient's appointment and admission to the hospital, collects information from the patient by phone and/or upon patient's arrival to the hospital.*
For the patient that will stay in the hospital ("inpatient") she or he should have a bed allotted in a ward. Receptionists might also receive patient's payments, record them in a database and provide receipts, file insurance claims and medical reports.

Hospital Management System is a large system including several subsystems or modules providing variety of functions. UML use case diagram example below shows actor and use cases for a hospital's reception.
Purpose: *Describe major services (functionality) provided by a hospital's reception.*
Hospital Reception subsystem or module supports some of the many job duties of hospital receptionist. Receptionist schedules patient's appointments and admission to the hospital, collects information from patient upon patient's arrival and/or by phone. For the patient that will stay in the hospital ("inpatient") she or he should have a bed allotted in a

ward. Receptionists might also receive patient's payments, record them in a database and provide receipts, file insurance claims and medical reports.

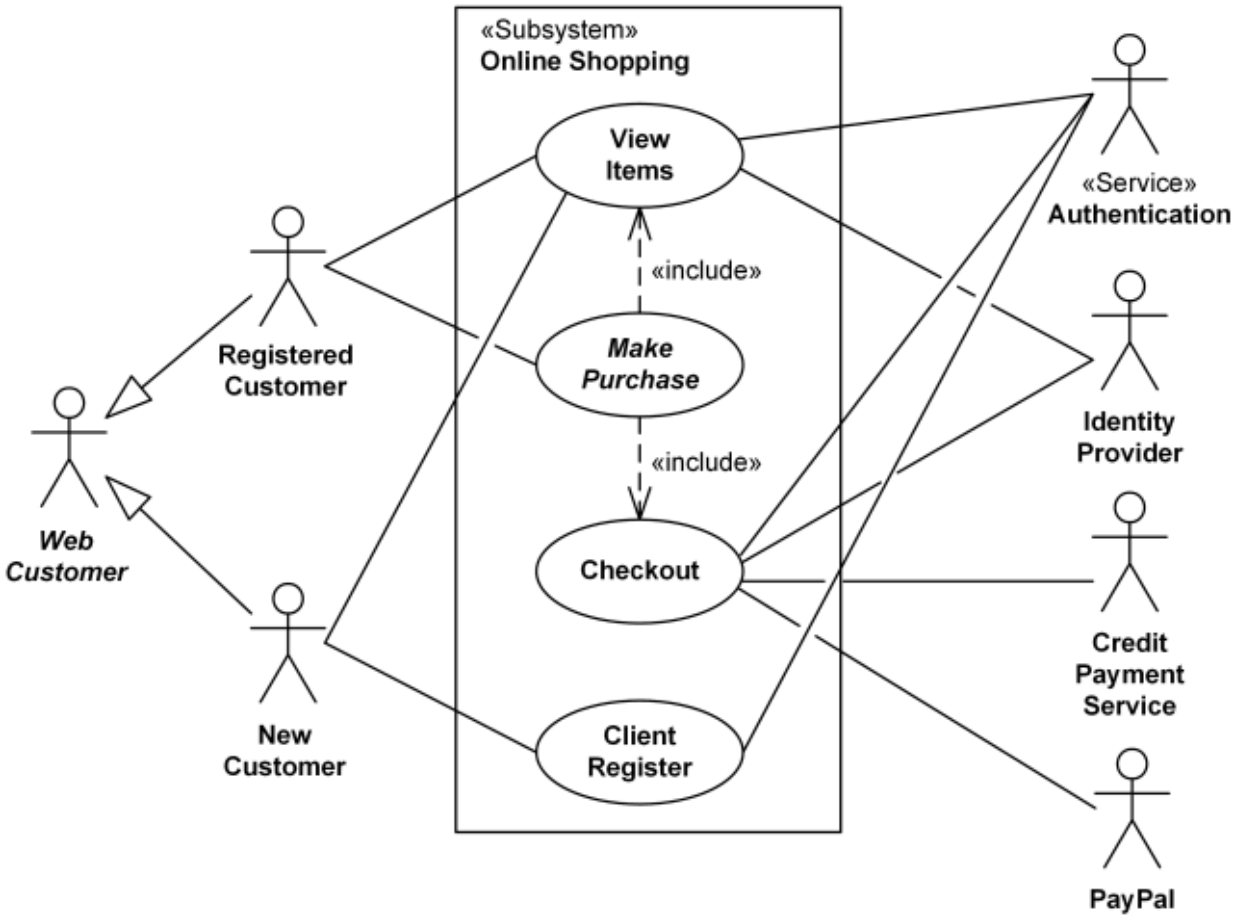


An example of use case diagram for Hospital Reception.

*Online Shopping

UML Use Case Diagram Example

Web Customer **actor** uses some web site to make purchases online. Top level **use cases** are **View Items**, **Make Purchase** and **Client Register**. View Items use case could be used by customer as top level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site, for example to get some coupons or be invited to private sales. Note, that **Checkout** use case is **included use case** not available by itself - checkout is part of making purchase.



Online shopping UML use case diagram example - top level use cases.

View Items use case is **extended** by several optional use cases - customer may search for items, browse catalog, view items recommended for him/her, add items to shopping cart or wish list. All these use cases are extending use cases because they provide some optional functions allowing customer to find item.

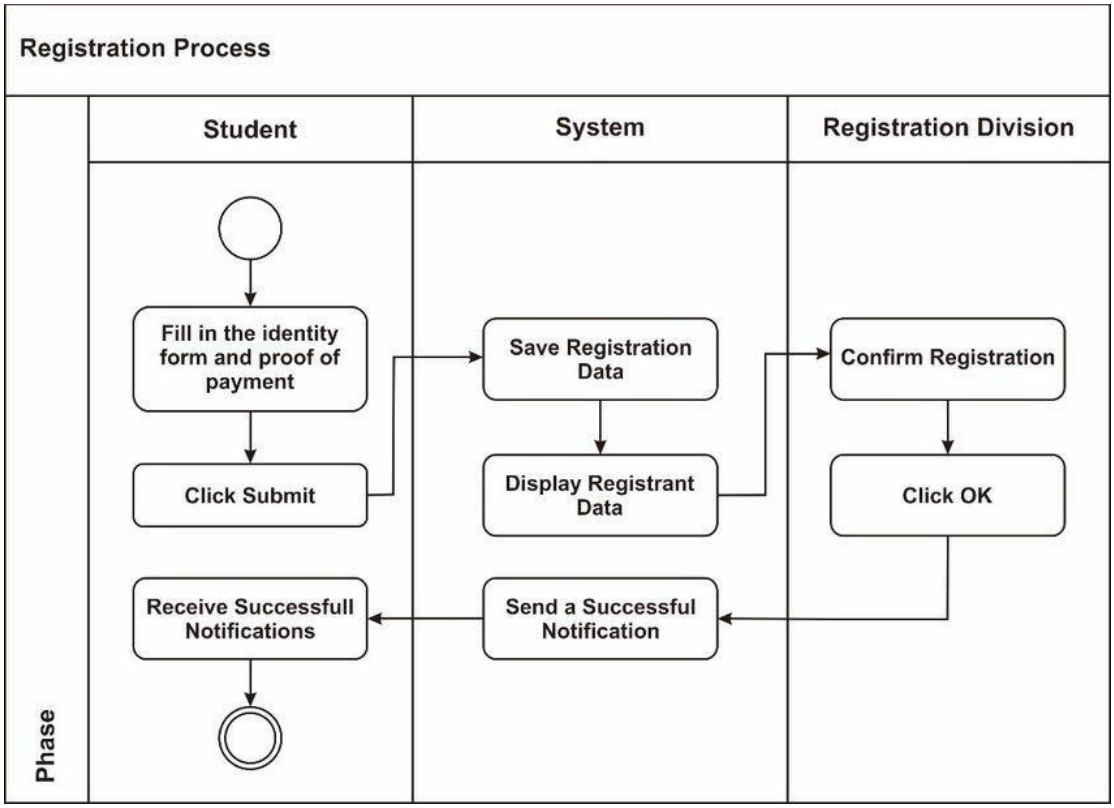
Customer Authentication use case is **included** in **View Recommended Items** and **Add to Wish List** because both require the customer to be authenticated. At the same time, item could be added to the shopping cart without user authentication.

Online shopping UML use case diagram example - view items use case.

*Activity Diagram

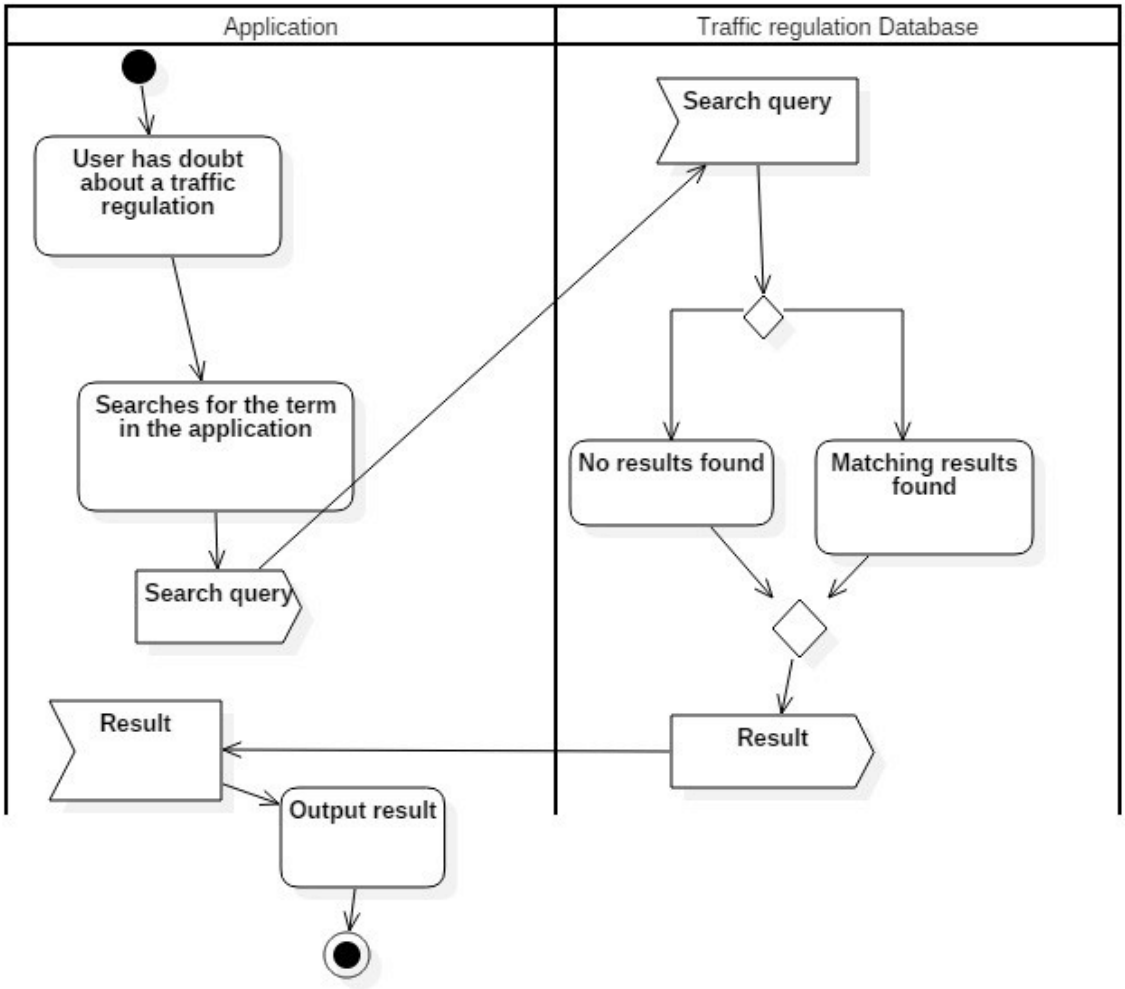
1.Student registration process activity diagram

This activity diagram shows the series of actions performed by the student, the student registration system, and the registration division to complete the student registration process.



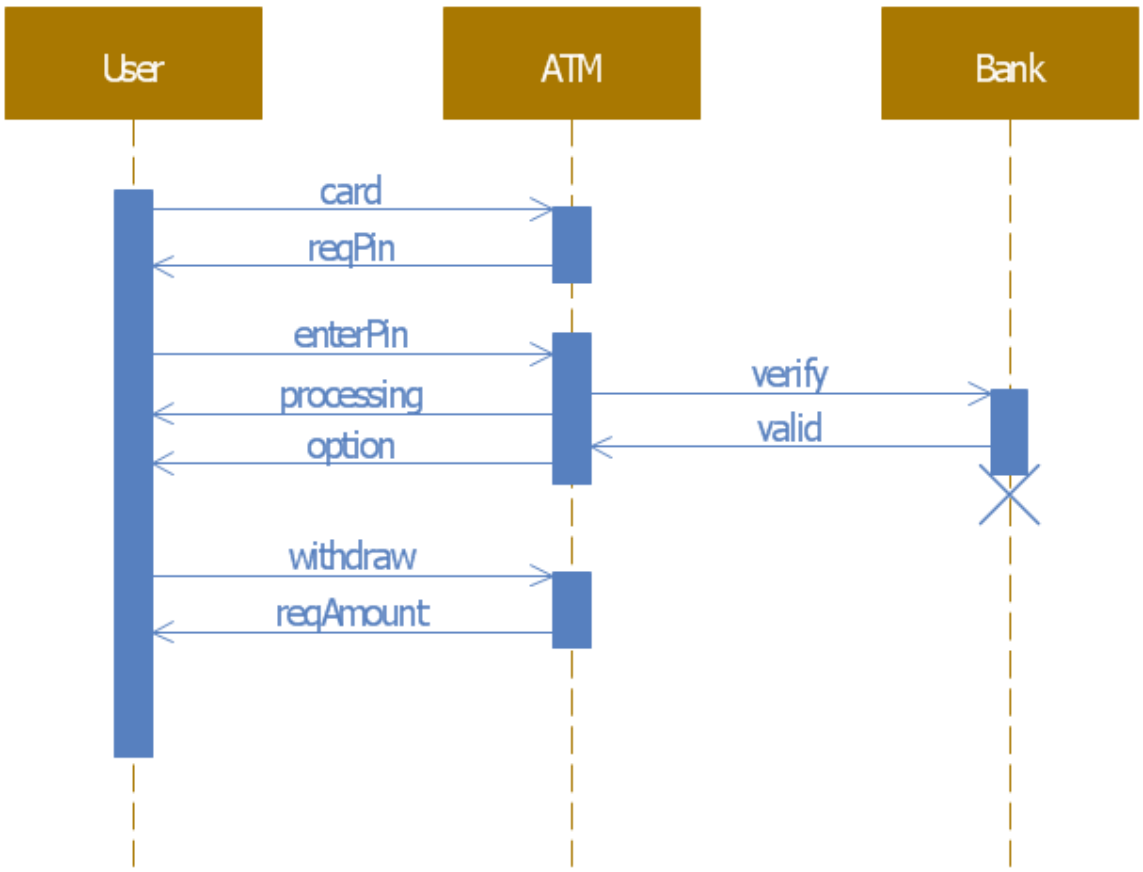
Traffic regulation search query process activity diagram

This activity diagram depicts the process of running a search query in a traffic regulation database. It features the sent and received signals as well as the object node to indicate that an object in the form of a search result is created from the last step in the process.

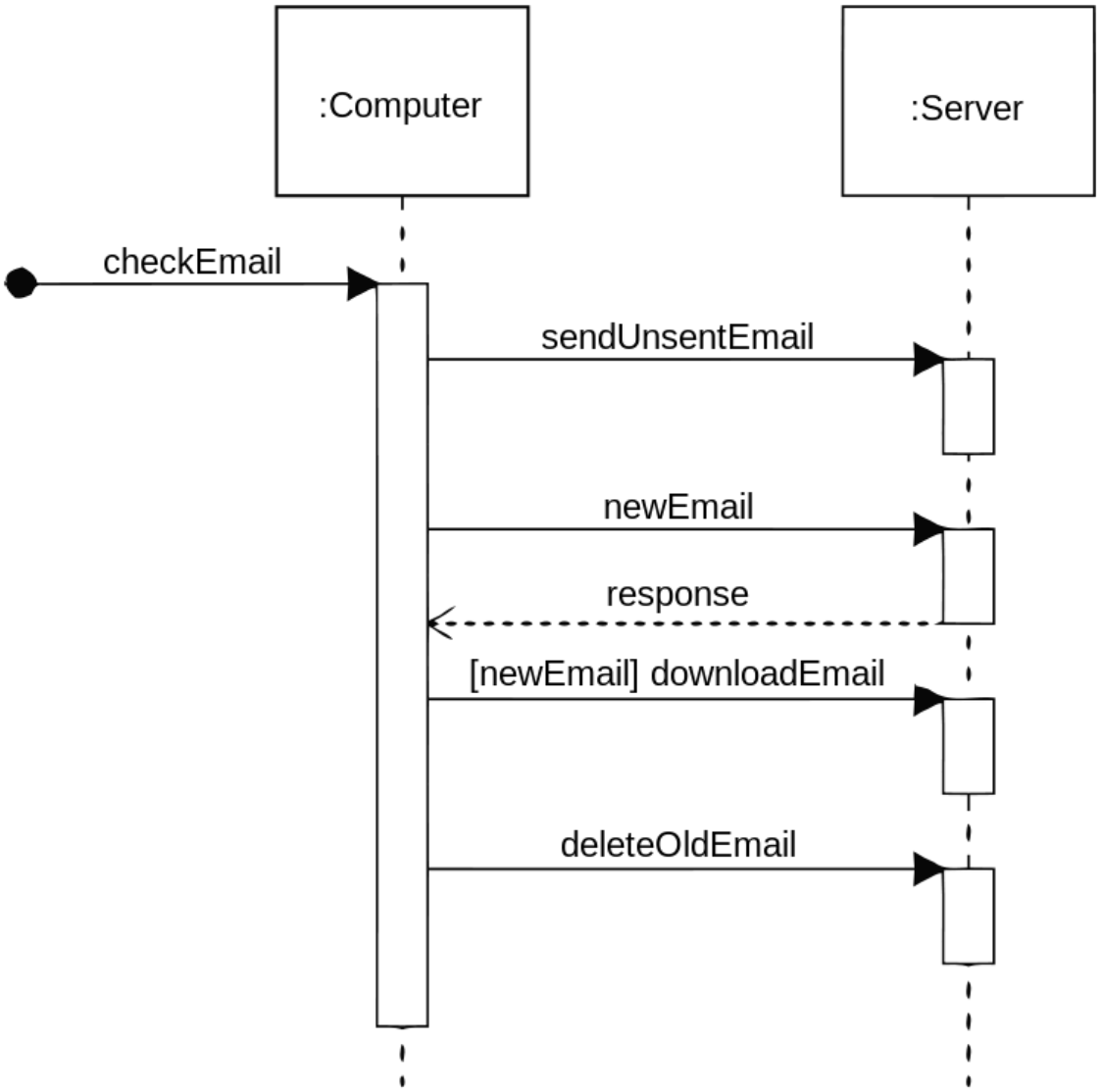

Application
Traffic regulation Database
User has doubt about a traffic regulation
Searches for the term in the application
Search query
Result
Output result
Search query
No results found
Matching results found
Result

*SEQUENCE DIAGRAM

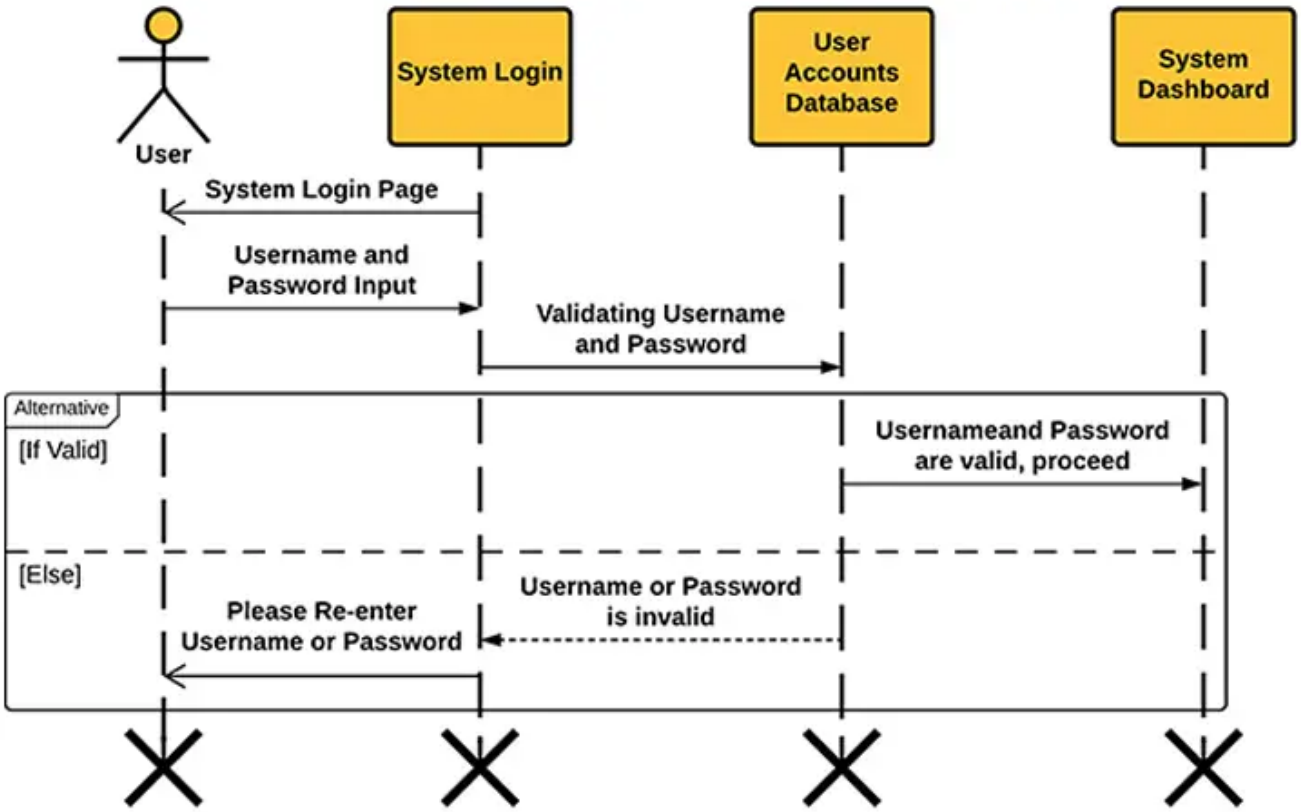
ATM



To check Email



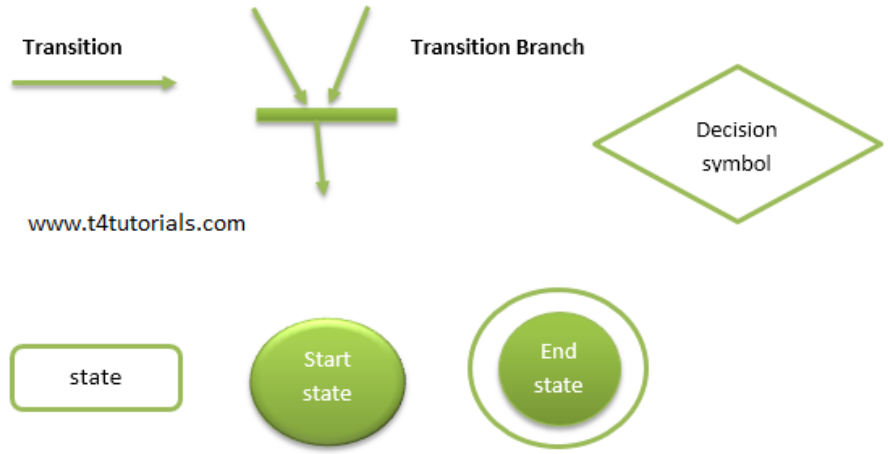
Login Page Sequence Diagram



https://itsourcecode.com/uml/sequence-diagram-for-login-system-uml-diagrams/

*State Transition diagram

Symbols used in State Transition Diagram



erminologies of State Transition Diagram

State

In the state transition diagram, An object always remains in some state. Further, the state of the object may change after an event occur.

Event

Any activity that may trigger a state transition or can change the state.

Guard

In the state transition diagram, a guard is a boolean expression. Suppose if the guard is true, then it enables an event to trigger a transition.

Transition

The change of state within an object is represented with a transition. It is possible that an object changes its state when the transition occurs.

Action

One or more actions are taken by an object when the object changes a state.

Examples of State Transition Diagram

Case Study:

You need to develop a web-based application in such a way that users can search other users, and after getting search complete, the user can send the friend request to other users. If the request is accepted, then both users are added to the friend list of each other. If one user does not

accept the friend request. The second user can send another friend request. The user can also block each other.

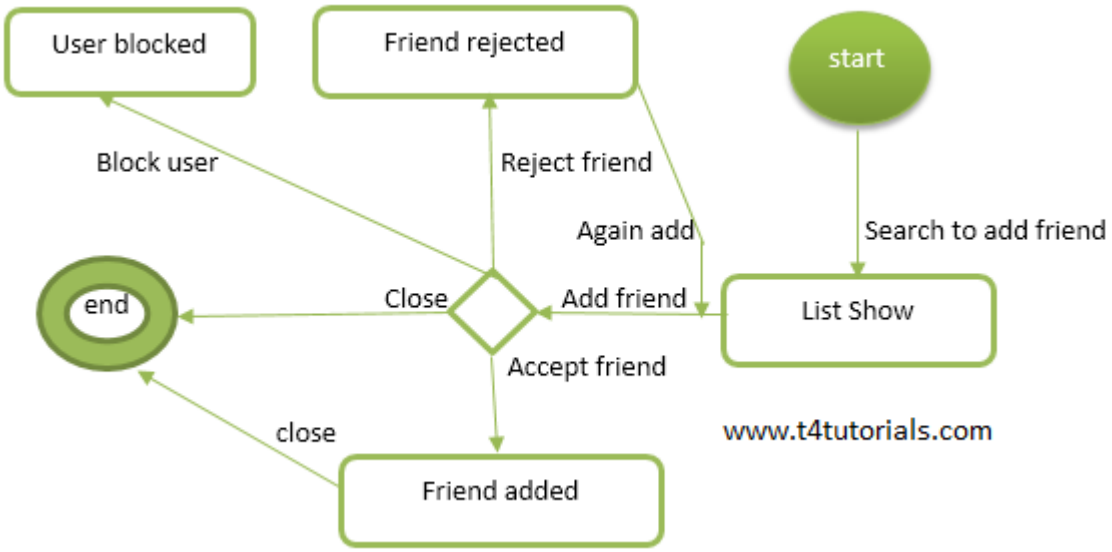
Solution:

1. First of all, identify the object that you will create during the development of classes in oop
2. Identify the actions or events
3. Identify the possible states for an object
4. Draw the diagram.

Object: friends

Events or actions: Search to add a friend, add a friend, accept a friend, reject a friend, again add, block user and close.

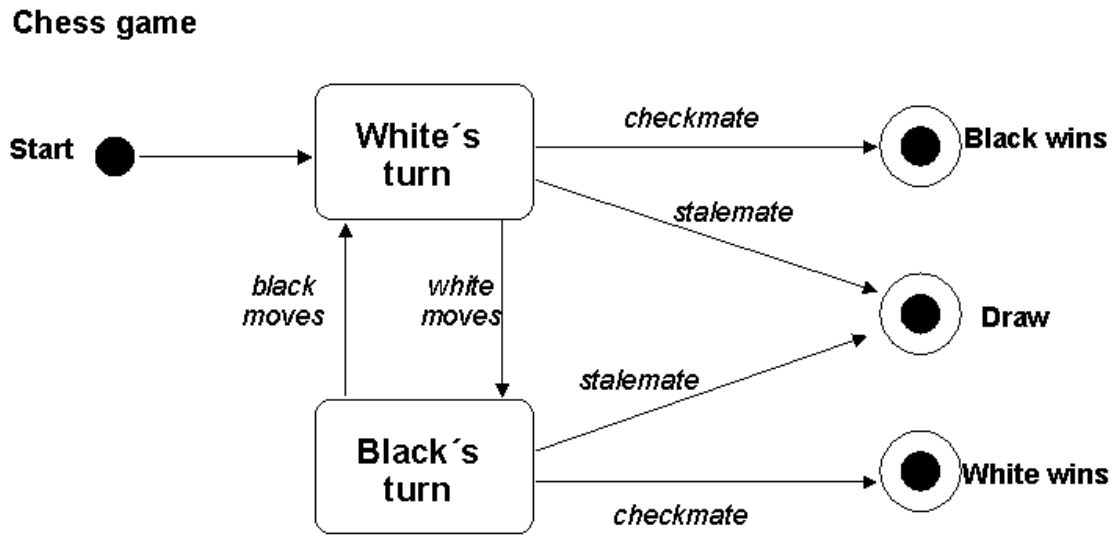
States: Start, the friend added, friend rejected, user blocked and end.



Source : https://t4tutorials.com/what-is-state-transition-diagram-software-engineering/

*Chess Game:

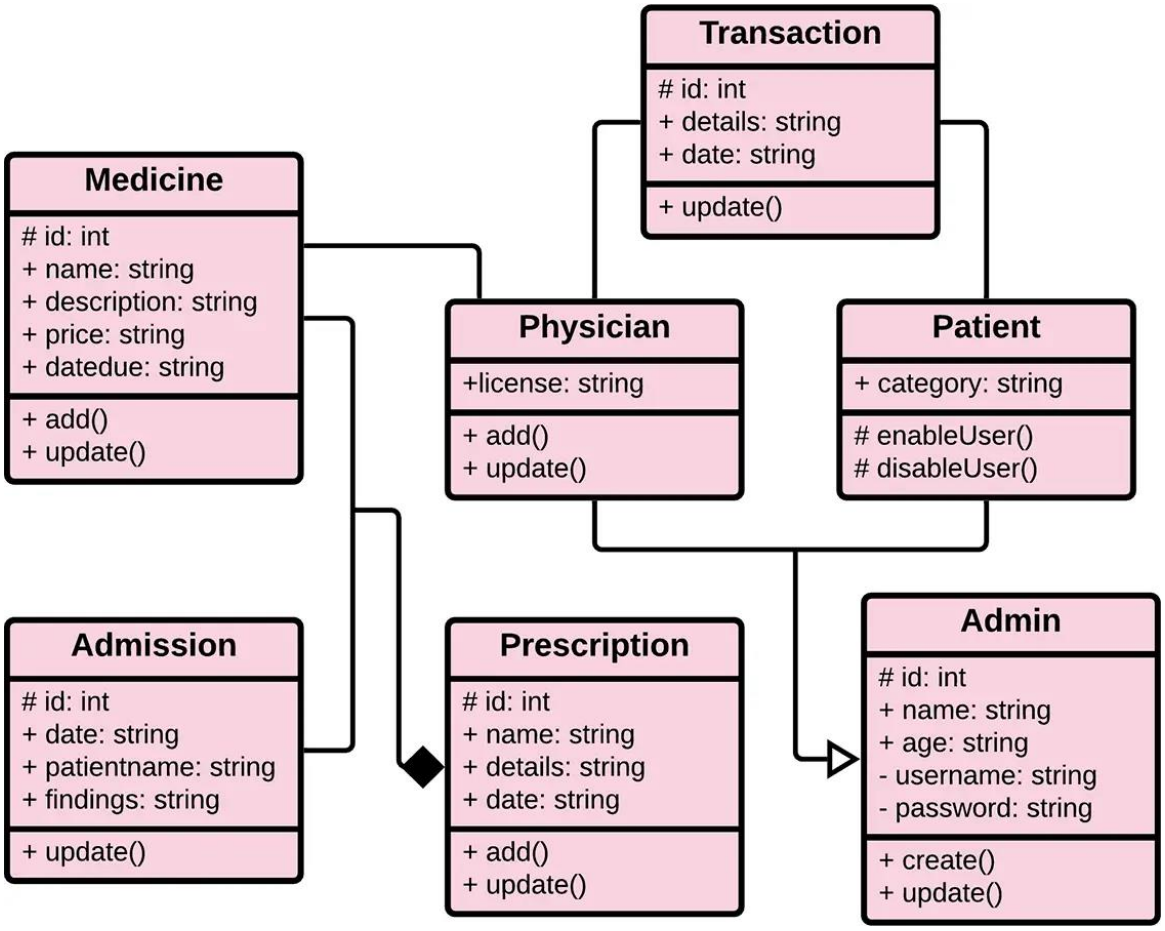
UML State Diagram - example



Source:
http://users.csc.calpoly.edu/~jdalbey/SWE/Design/STDeg1.gif

*Class Diagram

Hospital Management system

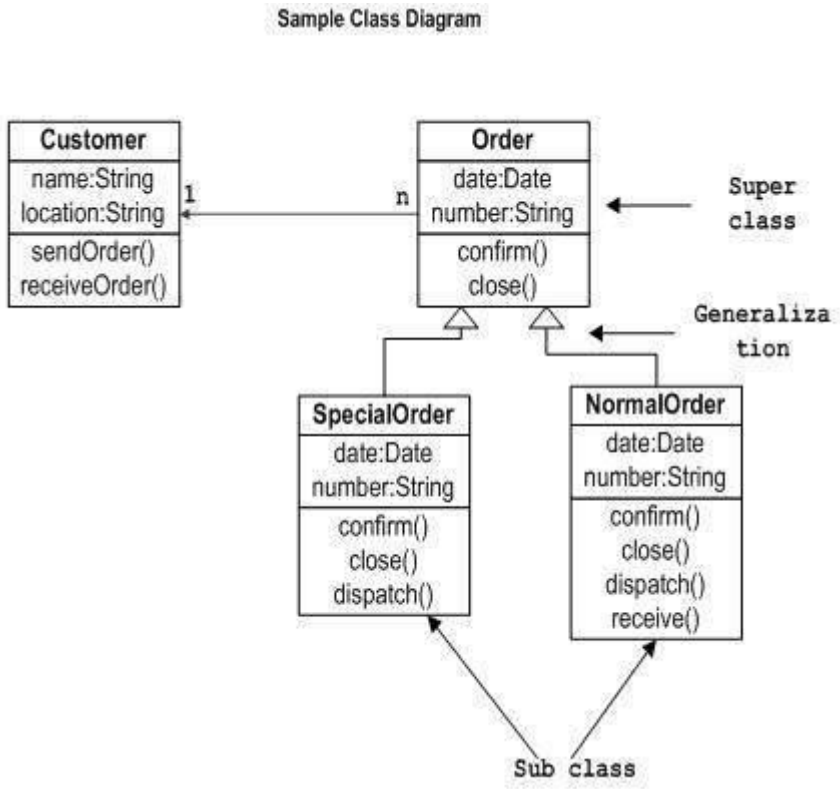


Source: https://itsourcecode.com/uml/hospital-management-system-class-diagram-uml/

*Order system

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

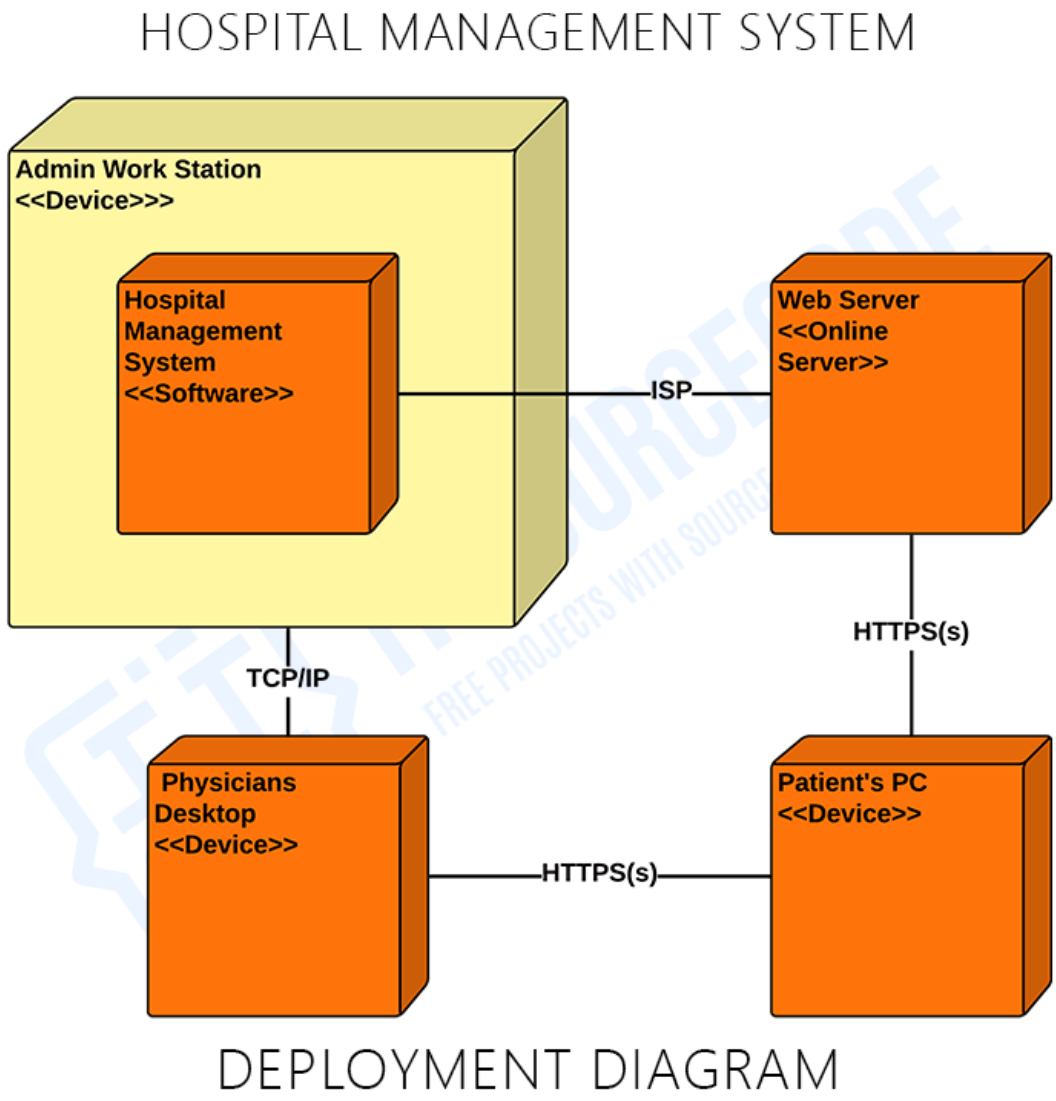
- First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
- Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().



Source:

https://www.tutorialspoint.com/uml/uml_class_diagram.htm

*Deployment Diagram for Online Hospital Management System



Source : https://itsourcecode.com/wp-content/uploads/2022/03/Deployment-Diagram-of-Hospital-Management-System-in-UML.png

*Note: 1)All diagram are outsourced from various site available from Google search engine.

2) The Design shown above is used only for academic purpose for references only and not utilized for any commercial purpose.