1.Define UML. And explain the principles of modeling?

Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis.

Principles of modeling:

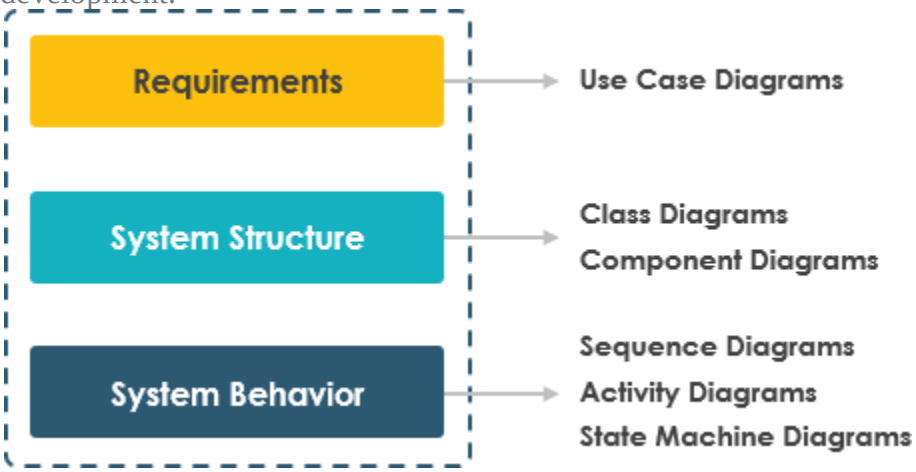
Principles of UML Modeling

1. The choice of model is important

The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped. We need to choose your models well.

- The right models will highlight the most critical development problems.
- Wrong models will mislead you, causing you to focus on irrelevant issues.

For Example: We can use different types of diagrams for different phases in software development.



2. Every model may be expressed at different levels of precision

For Example,

- If you are building a high rise, sometimes you need a 30,000-foot view for instance, to help your investors visualize its look and feel.

3. The best models are connected to reality

All models simplify reality and a good model reflects important key characteristics.

4. No single model is sufficient

Every non-trivial system is best approached through a small set of nearly independent models. Create models that can be built and studied separately, but are still interrelated. In the case of a building:

2.)**Illustrate elements of UML.**

1. **Structural elements:**
- **Class:** A class is a blueprint for creating objects that have the same attributes and behaviors. It represents a group of objects with similar properties and behaviors.
- **Object:** An object is an instance of a class. It represents a specific instance of the class with its own set of attribute values.
- **Interface:** An interface defines a set of methods that a class must implement. It specifies the behavior of the class without revealing its implementation details.
- **Package:** A package is a collection of related classes, interfaces, and other packages. It provides a way to organize and manage the elements of the system.
- **Component:** A component is a modular, self-contained unit of the system that provides a specific functionality. It can be reused across different systems and applications.
- **Deployment:** A deployment diagram shows the physical deployment of the system components on hardware devices.

2. **Behavioral elements:**
- **Use Case:** A use case represents a specific functionality of the system from the user's perspective. It describes the interactions between the user and the system.
- **Activity**: An activity represents a set of actions or tasks performed by the system to achieve a specific goal. It shows the flow of activities and the conditions that trigger them.
- **Sequence:** A sequence diagram shows the interactions between the system components and the order in which they occur. It shows the message exchanges between objects.
- **Stat**e: A state diagram shows the different states of an object and the transitions between them. It represents the lifecycle of an object.
- **Collaboration**: A collaboration diagram shows the interactions between the system components and their relationships. It shows how the components work together to achieve a specific goal.
- **Timing**: A timing diagram shows the timing constraints and behavior of the system components. It shows the timing relationships between the objects.

These are some of the key elements of UML. By using these elements to create different types of diagrams, we can develop a comprehensive model of the system that captures its structure, behavior, and interactions. This allows us to better understand the system, communicate its design to others, and identify potential problems or areas for improvement.

3)Nature of class and object.

A class represents a collection of objects having same characteristic properties that exhibit common behavior. It gives the blueprint or description of the objects that can be created from it. Creation of an object as a member of a class is called instantiation. Thus, object is an instance of a class.

In UML models, objects are model elements that represent instances of a class or of classes.

The constituents of a class are −

- A set of attributes for the objects that are to be instantiated from the class. Generally, different objects of a class have some difference in the values of the attributes. Attributes are often referred as class data.

- A set of operations that portray the behavior of the objects of the class. Operations are also referred as functions or methods.

Example

Let us consider a simple class, Circle, that represents the geometrical figure circle in a two–dimensional space. The attributes of this class can be identified as follows −

- x–coord, to denote x–coordinate of the center
- y–coord, to denote y–coordinate of the center
- a, to denote the radius of the circle

Some of its operations can be defined as follows −

- findArea(), method to calculate area
- findCircumference(), method to calculate circumference
- scale(), method to increase or decrease the radius

Object

In UML models, objects are **model elements that represent instances of a class or of classes**. You can add objects to your model to represent concrete and prototypical instances. A concrete instance represents an actual person or thing in the real world.. Each object has −

- Identity that distinguishes it from other objects in the system.
- State that determines the characteristic properties of an object as well as the values of the properties that the object holds.
- Behavior that represents externally visible activities performed by an object in terms of changes in its state.

Objects can be modelled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

4.Demonstrate the process of identification of classes and objects with suitable examples

The process of identifying classes and objects is a crucial step in the object-oriented software development process. It involves identifying the nouns in the problem domain and grouping them into classes based on their common attributes and behaviors. Here is a step-by-step process for identifying classes and objects with suitable examples:

1. **Identify the problem domain**: The first step is to identify the problem domain, which is the area of the system that we want to model. For example, if we are building a library management system, the problem domain would be the library system.
2. **Identify the nouns:** The next step is to identify the nouns in the problem domain. Nouns are typically the objects in the system. For example, in a library management system, the nouns might include books, patrons, librarians, and borrowing.
3. **Group the nouns into classes**: The next step is to group the nouns into classes based on their common attributes and behaviors. For example, we can group books into a Book class, which would have attributes such as title, author, and ISBN, and behaviors such as checking in and out.

4. **Identify the relationships between the classes**: The next step is to identify the relationships between the classes. For example, a patron might have a relationship with a Book class if they borrow a book from the library.
5. **Refine the classes:** The final step is to refine the classes by adding additional attributes and behaviors as needed. For example, we might add a due date attribute to the Book class to track when the book is due back to the library.

Here is an example of identifying classes and objects for a library management system:

1. **Problem domain**: Library management system
2. **Nouns:** Books, patrons, librarians, borrowing
3. **Classes:**

- Book: Attributes: title, author, ISBN; Behaviors: check in, check out, reserve
- Patron: Attributes: name, address, library card number; Behaviors: borrow, return, renew
- Librarian: Attributes: name, employee ID; Behaviors: check out, check in, add book, remove book
- Borrowing: Attributes: date borrowed, due date; Behaviors: none

4. **Relationships:**

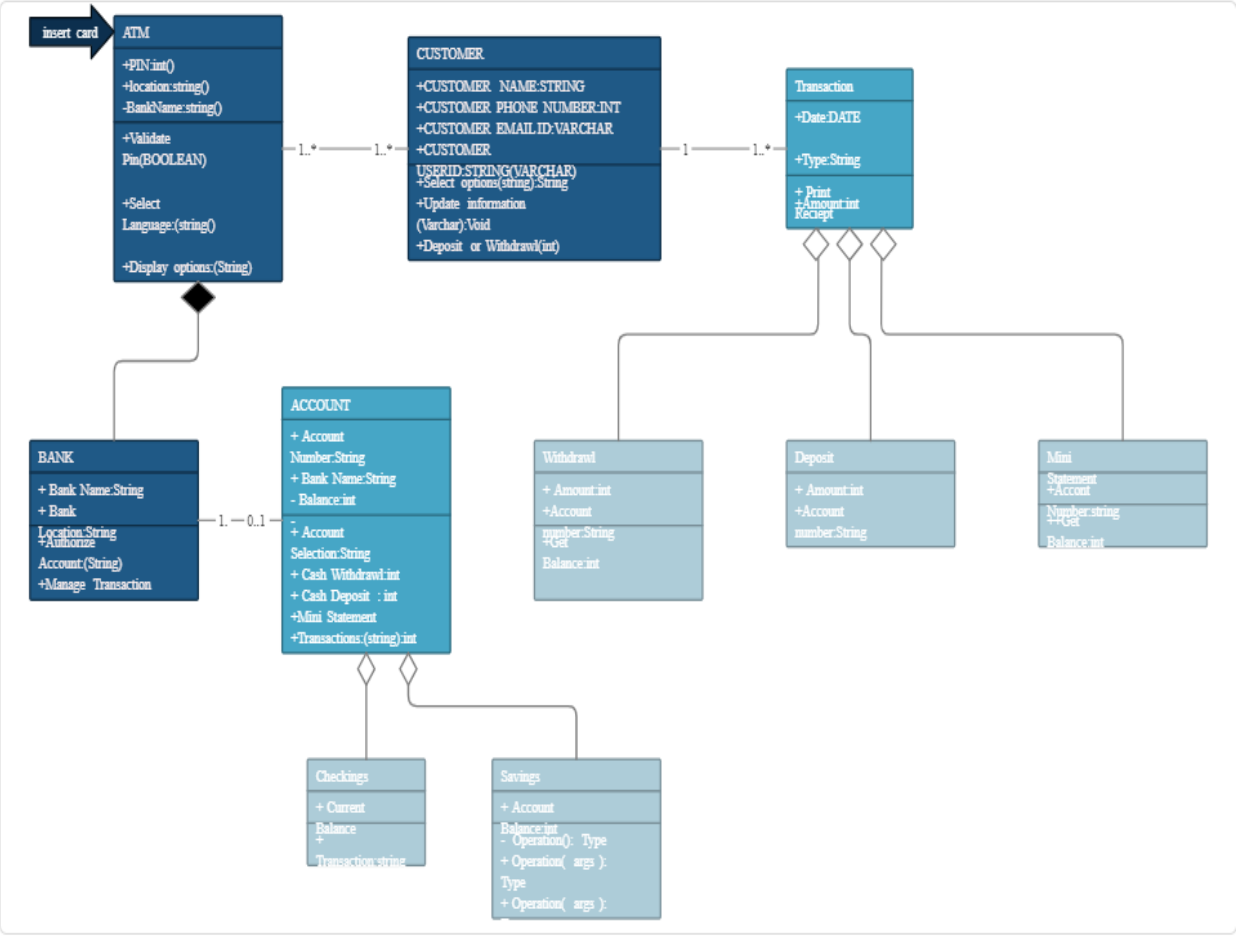
- A patron has a relationship with the Book class when they borrow a book.
- A librarian has a relationship with the Book class when they check out or check in a book.
- A patron has a relationship with the Borrowing class when they borrow a book.

5. **Refinement:**

- Add a genre attribute to the Book class to categorize books by genre.
- Add a late fee behavior to the Borrowing class to calculate the late fee for overdue books.

By following this process, we can identify the classes and objects needed to build a library management system. This allows us to better understand the system, communicate its design to others, and develop a more efficient and effective software solution.

5.Draw and explain class diagram for ATM application.



This ATM Class Diagram provides an effective visual representation of how an automated teller machine operates.

A ATM Class Diagram contains common classes, relationships, associations and components of an ATM, including the customer, card, terminal and bank.

This diagram shows how the different components interact within the system to enable customers to withdraw and deposit money.

The ATM Class Diagram also highlights the role of ATMs in financial transactions, such as transferring funds from one account to another, bill payments and banking operations.

This diagram is essential for understanding the functionality of ATMs and how they are used in financial services.

6.Elements of object model.

Elements of the Object Model

The four major elements of the object model (the conceptual framework of an object-oriented thing) are —

1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy

and the three minor elements are —

1. Typing
2. Concurrency
3. Persistence.

The major elements are essential whereas the minor ones are useful but not essential. Now we try to elaborately define major elements and minor elements.

Major Elements

The major elements are listed below.

Encapsulation: Encapsulation is the practice of hiding the internal details of an object and exposing only what is necessary for other objects to interact with it. Encapsulation is achieved through the use of access modifiers such as public, private, and protected.

Abstraction in object-oriented analysis and design (OOAD) is the process of identifying the essential characteristics of an object, and ignoring or hiding the details that are not relevant to the problem being solved. Abstraction allows us to focus on the most important aspects of an object, and ignore the rest.

In object-oriented analysis and design (OOAD), a **hierarchy r**efers to a tree-like structure of classes or objects, where each level represents a specialization or generalization of the levels below it. The hierarchy is also known as a class hierarchy or an inheritance hierarchy.

Modularity in object-oriented analysis and design (OOAD) refers to the practice of breaking down a system or problem domain into smaller, more manageable units called modules or components. Each module is designed to perform a specific function or set of related functions.

Minor Elements

The minor elements are listed below.

Typing

Concepts of typing derive primarily from theories of abstract data types. A type is a precise characterization of structural or behavioral which a collection of entities all share. Typing is the enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways.

Concurrency

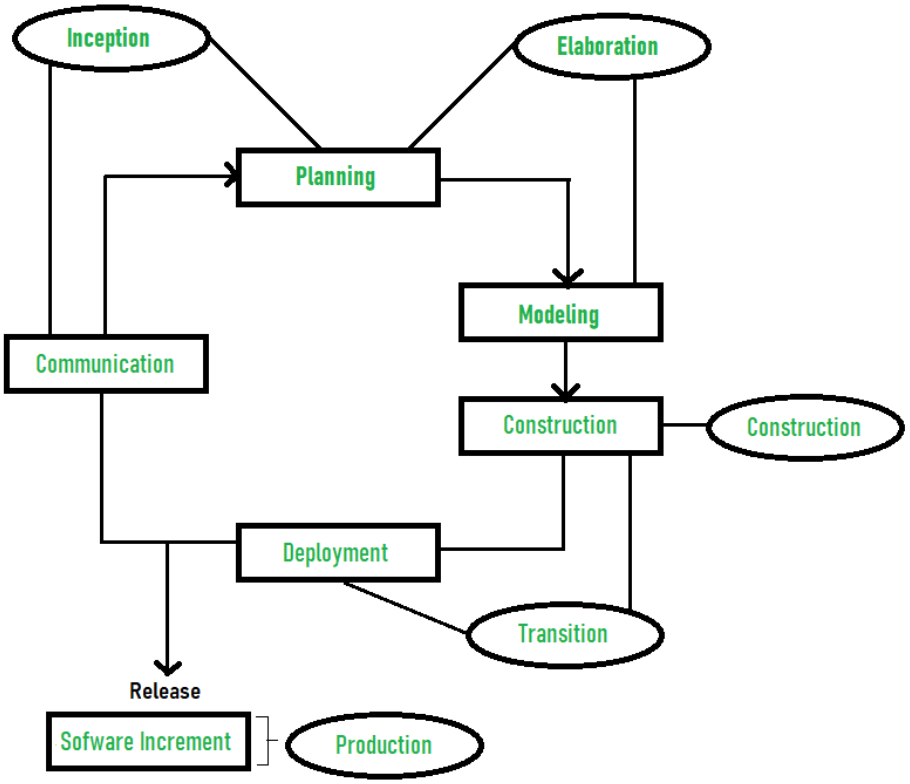
It is the property that distinguishes an active object from one that is not active.

Persistence

Is the property of an object through which its existence transcends time (i.e. the object continues to exist after its creator ceases to exist) and/or space (i.e. the object's location moves from the address space in which it was created).

Object diagram for library management system.

7.Phases of UP: There is total of five phases of the life cycle of



RUP:

Inception –

Communication and planning are the main ones.

Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.

Customers' requirements are identified and then it becomes easy to make a plan for the project.

Elaboration –

Planning and modeling are the main ones.

A detailed evaluation and development plan is carried out and diminishes the risks.

Revise or redefine the use-case model (approx. 80%), business case, and risks.

Construction –

The project is developed and completed.

System or source code is created and then testing is done.

Coding takes place.

Transition –

The final project is released to the public.

Transit the project from development into production.

Update project documentation.

Beta testing is conducted.

Defects are removed from the project based on feedback from the public.

Production –

The final phase of the model.

The project is maintained and updated accordingly.

8.Unified approach of modeling.

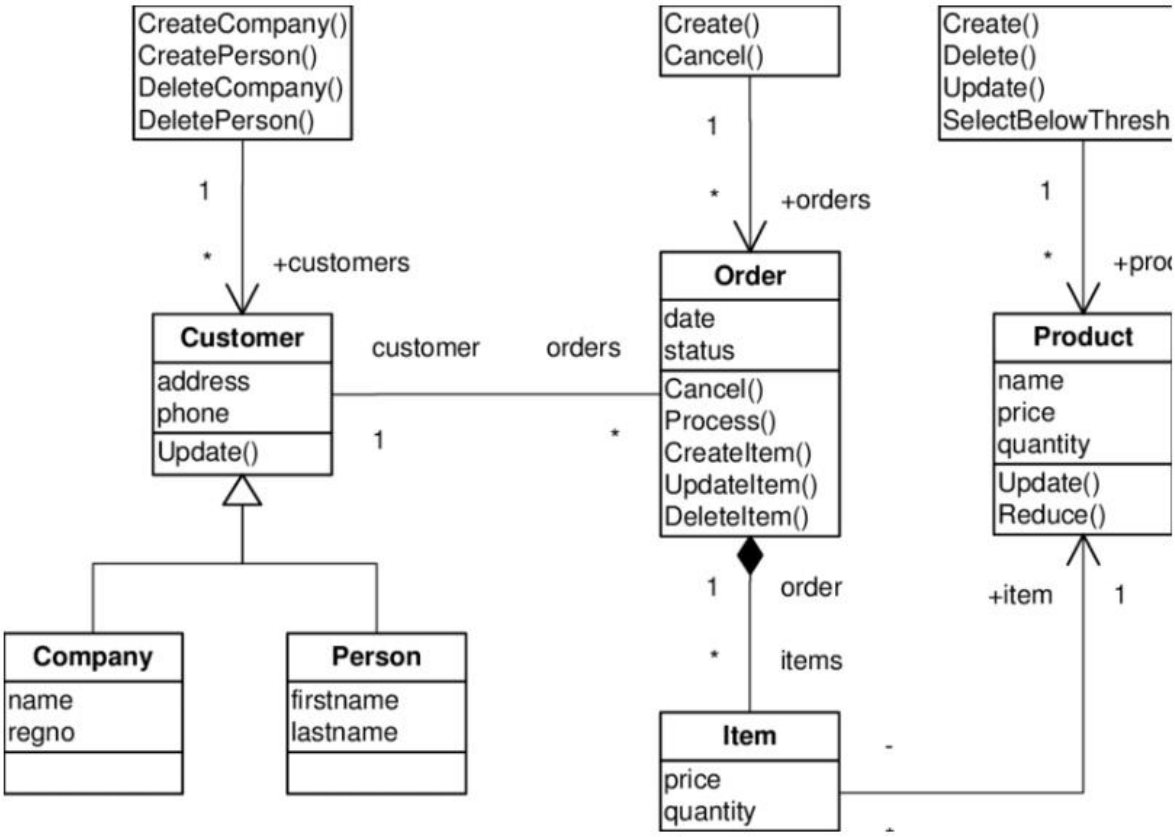
Unified Modeling Language (UML) is a standardized visual modeling language used for designing and documenting software systems. It provides a set of diagrams and notations to describe the structure, behavior, and interactions of the system components. The unified approach of modeling involves using UML to develop a comprehensive model that represents all aspects of the system.

Here is an example of the unified approach of modeling for an e-commerce website:

1. **Use Case Diagram**: The use case diagram describes the system's functionality from the user's perspective. It shows the interactions between the user and the system and defines the use cases for the system. In our example, the use case diagram might include use cases such as "search for product," "add to cart," "checkout," and "pay for order."
2. **Class Diagram:** The class diagram describes the structure of the system by defining the classes, their attributes, and their relationships. In our example, the class diagram might include classes such as "product," "customer," "order," and "payment."
3. **Sequence Diagram**: The sequence diagram describes the interactions between the system components and shows the order in which they occur. In our example, the sequence diagram might show the steps involved in a customer placing an order, including searching for a product, adding it to their cart, and checking out.
4. **Activity Diagram:** The activity diagram describes the behavior of the system by showing the flow of activities and the conditions that trigger them. In our example, the activity diagram might show the steps involved in processing an order, such as verifying payment, updating inventory, and notifying the customer.
5. **State Diagram**: The state diagram describes the lifecycle of an object and the transitions between its different states. In our example, the state diagram might show the different states of an order, such as "pending," "shipped," and "delivered."
6. **Component Diagram**: The component diagram describes the components of the system and their dependencies. In our example, the component diagram might show the different modules of the e-commerce website, such as the front-end user interface, the back-end database, and the payment gateway.

By using UML to create these different types of diagrams, we can develop a comprehensive model of the e-commerce website that captures all aspects of the system. This allows us to better understand the system, communicate its design to others, and identify potential problems or areas for improvement.

9.class diagram for order management system.



1: A UML class diagram of an order management system The class diagram contains the main classes of the problem: Product, Customer, Company, Person, Order and Item. It also contains the classes Products,

10.List out the classes and objects for hotel management system

Class :

The classes used in this system are,

Hotel Management : This class depicts the entire hotel and says whether the hotel is opened or closed.

Employees : It contains the details of the Employee. There are two kinds of employees, Server and the chef. This employee class is the parent class of two subclass – Server and Chef

Server : It contains the details of the server, the table to which they are assigned, the order which is currently serving, etc.

Chef : It contains the details of the chef working on a particular order.

Customer : It contains the details of the customer.

Table : It contains the table details like table number and the server who are assigned to that table.

Menu : Menu contains all the food items available in the restaurant, their availability, prize, etc.

Order : Order depicts the order associated with a particular table and the customer.

Bill : Bill is calculated using the order and menu.

Payment : This class is for doing payment. The payment can be done in two ways either cash or card. So payment is the parent class and cash and card are subclasses.

Cash : Payment can be done by cash

Card : Payment can be done by card or online

Attributes :

Hotel Management – HotelName, NumberOfEmployees

Employees – EmployeeId, EmployeeName, EmployeeSalary

Server – ServerId, OrderId

Chef – Chef_Id, OrderId

Customer – CustomerId, CustomerName, Bill_Id, OrderId, PaymentId

Table – TableNumber, OccupiedStatus, ServerId, CustomerId

Menu – ItemId, ItemName, Amount

Order – OrderId, ItemId, ItemName, Quantity, CustomerId, ServerId

Bill – Bill_Id, OrderId, TotalBill

Payment – PaymentId, Bill_Id

11)Define an object and mention common uses of objects.Draw the object diagram for online reservation system.

An object is a self-contained entity that has its own state and behavior. Objects can represent anything from physical objects, such as a car or a book, to abstract concepts, such as an order or a customer. Objects are a fundamental building block of object-oriented programming, which is a programming paradigm that focuses on creating software systems composed of objects.

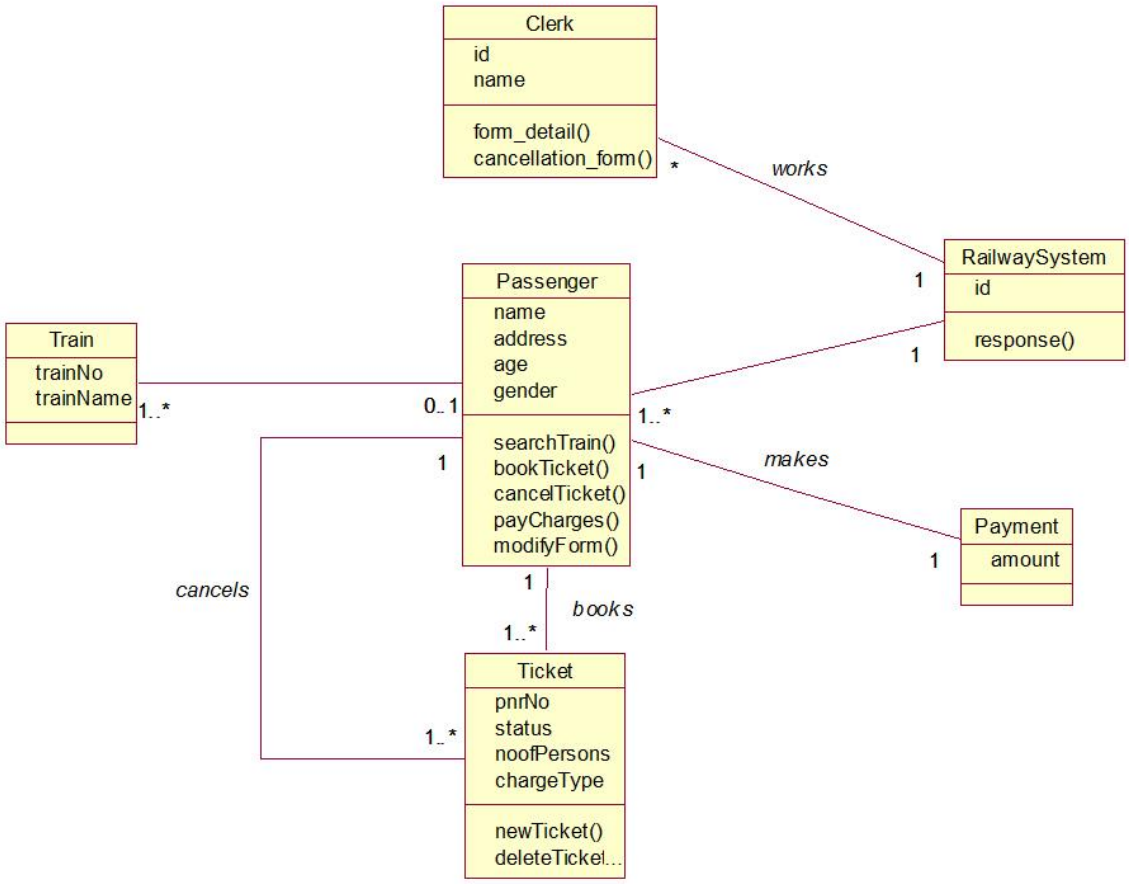
Common uses of objects include:

Modeling real-world objects: Objects can be used to model real-world objects, such as cars, houses, and people, in software systems.

Representing abstract concepts: Objects can also be used to represent abstract concepts, such as orders, invoices, and transactions.

Encapsulating data and behavior: Objects can encapsulate data and behavior, allowing developers to hide implementation details and protect the integrity of the system.

Improving code organization and reuse: Objects can improve code organization and reuse by breaking down complex systems into smaller, more manageable components.



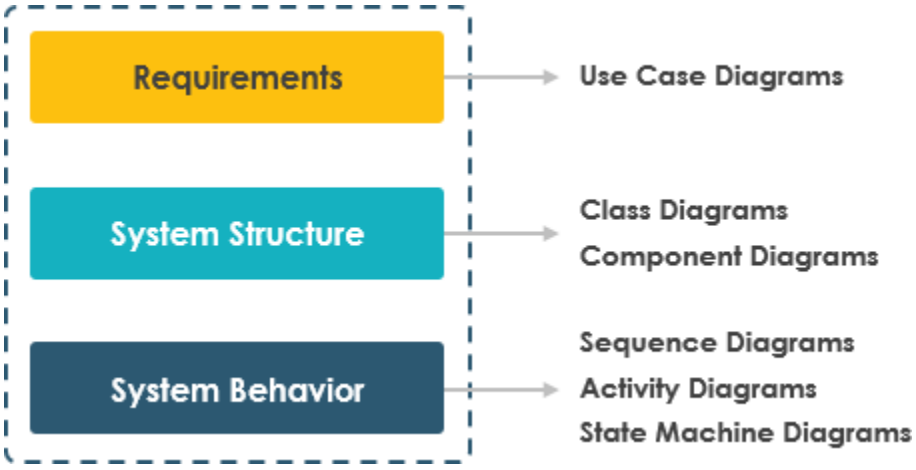
12. Principles of UML Modeling

1. The choice of model is important

The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped. We need to choose your models well.

- The right models will highlight the most critical development problems.
- Wrong models will mislead you, causing you to focus on irrelevant issues.

For Example: We can use different types of diagrams for different phases in software development.



2. Every model may be expressed at different levels of precision

For Example,

- If you are building a high rise, sometimes you need a 30,000-foot view for instance, to help your investors visualize its look and feel.

3. The best models are connected to reality

All models simplify reality and a good model reflects important key characteristics.

4. No single model is sufficient

Every non-trivial system is best approached through a small set of nearly independent models. Create models that can be built and studied separately, but are still interrelated. In the case of a building: