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

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. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. Its been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

Principles of modeling are as follows:

1. **"The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped".** This means choose your correct model as per the requirement of problem statement. Wrong model will mislead you, causing to focus on irrelevant issues.

2. **"Every model may be expressed at different levels of precision:"** This means all the user and developers both may visualize a system at different levels of details at different time.

3. **"The best models are connected to reality".** This means that the model must have things that are practically possible. They must satisfy the real word scenarios.

4. **"No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models:"** This means you need to have use case view, design view, process view, implementation view and development view. Each of these views may have structural as well as behavioral aspects. Together these views represent a system.

2.Illustrate Elements of Unified modeling language?

The following topics describe model elements in class diagrams:

- **Classes**
  In UML, a *class* represents an object or a set of objects that share a common structure and behavior. Classes, or instances of classes, are common model elements in UML diagrams.
- **Objects**
  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. For example, a concrete instance of a Customer class represents an actual customer.

A prototypical instance of a Customer class contains data that represents a typical customer.

- **Packages**
  Packages group related model elements of all types, including other packages.
- **Signals**
  In UML models, *signals* are model elements that are independent of the classifiers that handle them. Signals specify one-way, asynchronous communications between active objects.
- **Enumerations**
  In UML models, *enumerations* are model elements in class diagrams that represent user-defined data types. Enumerations contain sets of named identifiers that represent the values of the enumeration. These values are called enumeration literals.
- **Data types**
  In UML diagrams, *data types* are model elements that define data values. You typically use data types to represent primitive types, such as integer or string types, and enumerations, such as user-defined data types.
- **Artifacts**
  In UML models, *artifacts* are model elements that represent the physical entities in a software system. Artifacts represent physical implementation units, such as executable files, libraries, software components, documents, and databases.
- **Relationships in class diagrams**
  In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between model elements.
- **Qualifiers on association ends**
  In UML, *qualifiers* are properties of binary associations and are an optional part of association ends. A qualifier holds a list of association attributes, each with a name and a type. Association attributes model the keys that are used to index a subset of relationship instances.

3. What is a nature of Class and Object?

UML Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

1. It analyses and designs a static view of an application.
2. It describes the major responsibilities of a system.
3. It is a base for component and deployment diagrams.
4. It incorporates forward and reverse engineering.

UML Object Diagram

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant.

Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

Purpose of Object Diagram

The object diagram holds the same purpose as that of a class diagram. The class diagram provides an abstract view which comprises of classes and their relationships, whereas the object diagram represents an instance at a particular point of time.

The object diagram is actually similar to the concrete (actual) system behavior. The main purpose is to depict a static view of a system.

Following are the purposes enlisted below:

- It is used to perform forward and reverse engineering.
- It is used to understand object behavior and their relationships practically.
- It is used to get a static view of a system.
- It is used to represent an instance of a system.

4.Demonstrate the process of identification of Classes and Objects with suitable examples ?

The process of identifying classes and objects in UML involves analyzing the problem domain and identifying the entities and their attributes and behaviors. Here are some steps to follow:

1. Identify the nouns and noun phrases in the problem statement. These represent potential classes.
2. Analyze the relationships between the identified entities. This will help determine the associations between the classes.
3. Identify the attributes and behaviors of the identified classes.
4. Group related classes together to form packages.
5. Create a class diagram to represent the identified classes and their relationships.

Here is an example of this process:

Problem statement: A library wants to develop a system to manage its book inventory.

1. Nouns and noun phrases: Library, book, inventory, borrower, loan.
2. Relationships: A book belongs to the library, a borrower can borrow a book, a loan is associated with a book and a borrower.
3. Attributes and behaviors:

- Library: name, address, phone number, list of books.
- Book: title, author, ISBN, genre, publication date, number of copies available.
- Borrower: name, address, phone number, library card number.
- Loan: due date, return date, late fee.

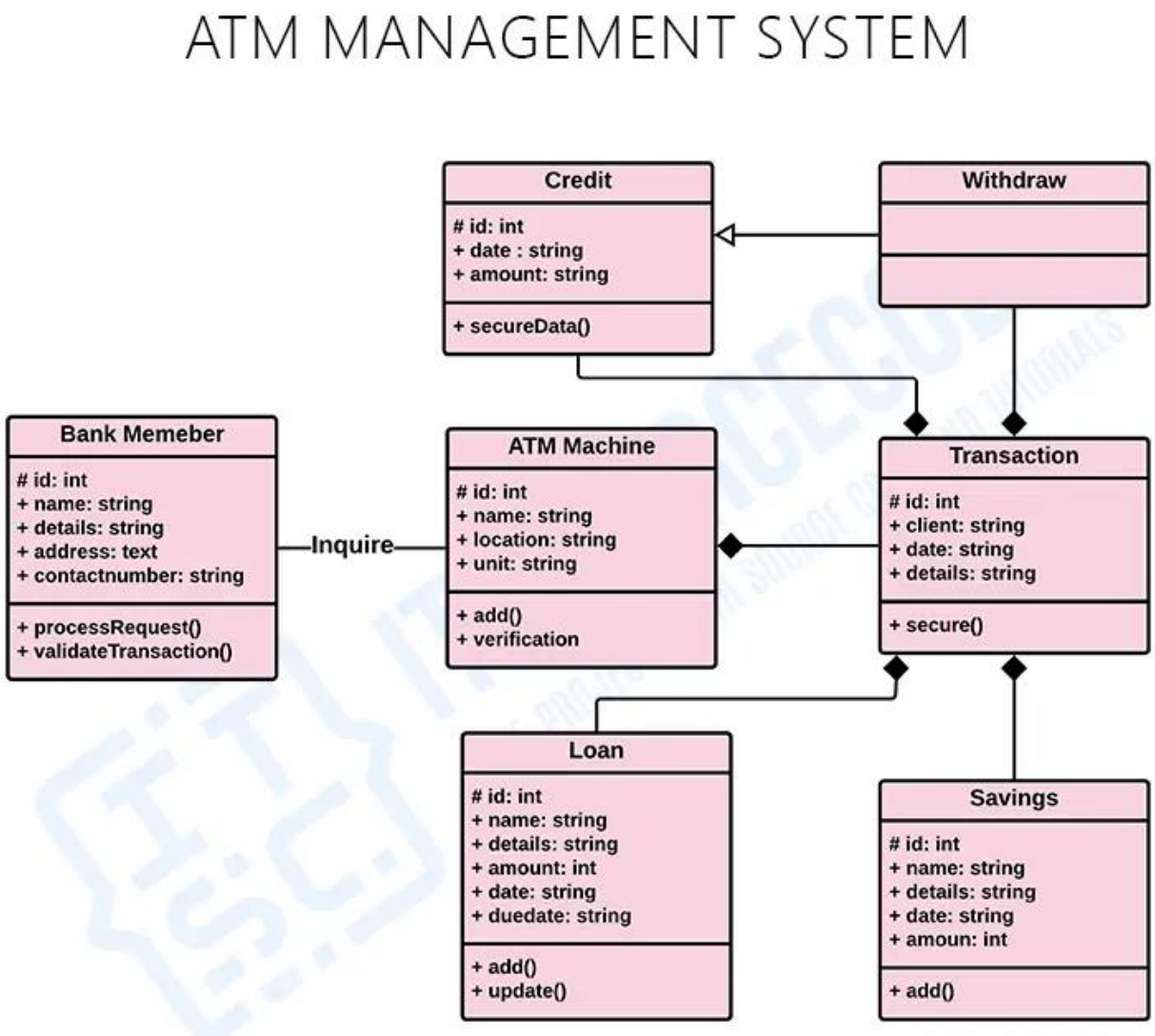
4. Grouping: The classes can be grouped into two packages: Library and Loan.
5. Class diagram: The class diagram for this system would include the Library and Loan packages, with the Book and Borrower classes connected to the Loan class via associations.

5.Draw and explain the class diagram for an ATM application

ATM Management System Class Diagram

This simple class diagram gives you the exact details about the system's class characteristics and methods. It also clarifies the connections of classes in the system.

Here, I will be showing you the sample constructed class diagram provided with its attributes and methods. This is from the simple idea of the ATM machine system's common function.



6.Describe elements of object model? Demonstrate object diagram for Library management system?

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

1. Abstraction

Abstraction is one of the fundamental way in which we as humans cope with complexity. An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects. Thus it provides crisply defined conceptual boundaries, relative to the perspective of the viewer.

We can characterize the behavior of an object by considering the services that it provides to other objects as well as the operations that it may perform upon other objects. A protocol denotes the way in which an object may act or react and thus constitutes the entire static and dynamic outside view of the abstraction.

2. Encapsulation

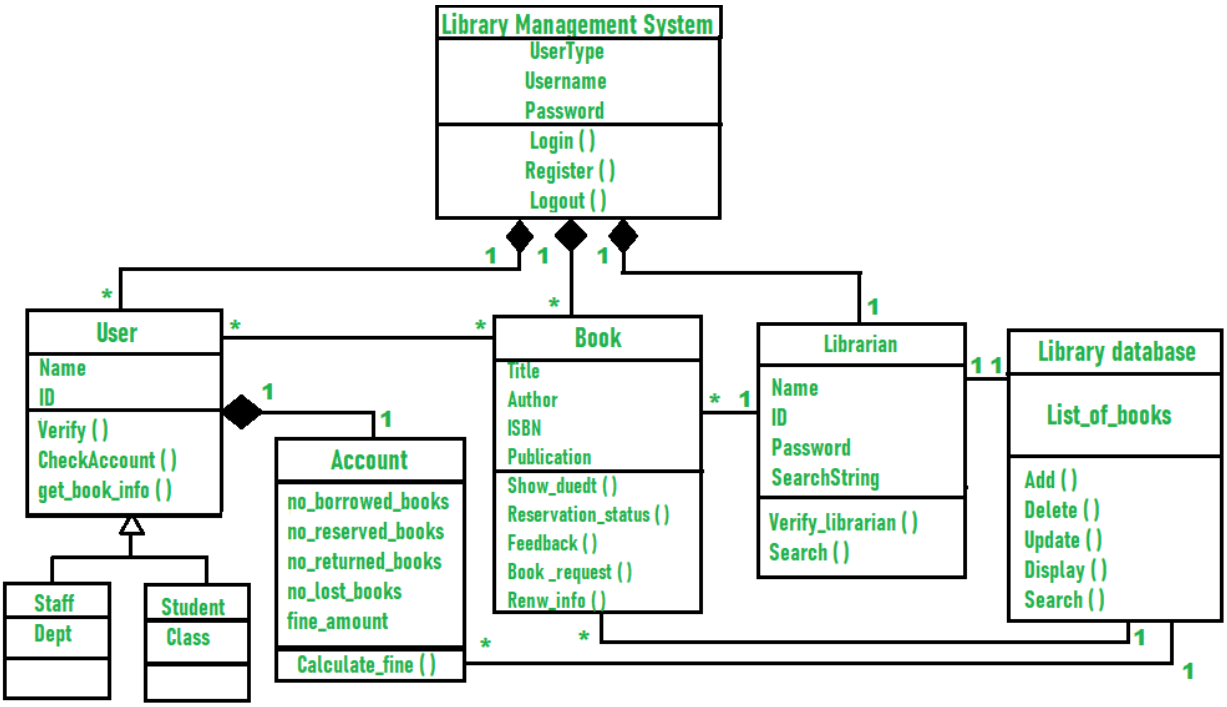
Encapsulation is the process of compartmentalizing the elements of an abstraction that constitutes its structure and behavior. Encapsulation serves to separate the contractual interface of an abstraction and its implementation. Abstraction and encapsulations are complimentary concepts.

3. Modularity

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.

4. Hierarchy

Hierarchy is a ranking of ordering of abstractions. The two most important hierarchies in a complex system are its class structure (‘is a’ or Generalization/specialization) and object structure (‘part of’ or whole part). Inheritance is the most important ‘is a” hierarchy. Basically inheritance defines a relationship among classes wherein one class shares the structure or behavior defined in one or more classes.



CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

7.Explain different phases of Unified Process along with Architectural view?

The Four Phases:

Inception

The primary goal of the **Inception phase** is to establish the case for the viability of the proposed system.

The tasks that a project team performs during Inception include the following:

- Defining the scope of the system (that is, what's in and what's out)
- Outlining a **candidate architecture**, which is made up of initial versions of six different models
- Identifying critical risks and determining when and how the project will address them
- Starting to make the business case that the project is worth doing, based on initial estimates of cost, effort, schedule, and product quality

Elaboration

The primary goal of the **Elaboration phase** is to establish the ability to build the new system given the financial constraints, schedule constraints, and other kinds of constraints that the development project faces.

The tasks that a project team performs during Elaboration include the following:

- Capturing a healthy majority of the remaining functional requirements
- Expanding the candidate architecture into a full **architectural baseline**, which is an internal release of the system focused on describing the architecture
- Addressing significant risks on an ongoing basis
- Finalizing the business case for the project and preparing a project plan that contains sufficient detail to guide the next phase of the project (Construction)

Construction

The primary goal of the **Construction phase** is to build a system capable of operating successfully in beta customer environments.

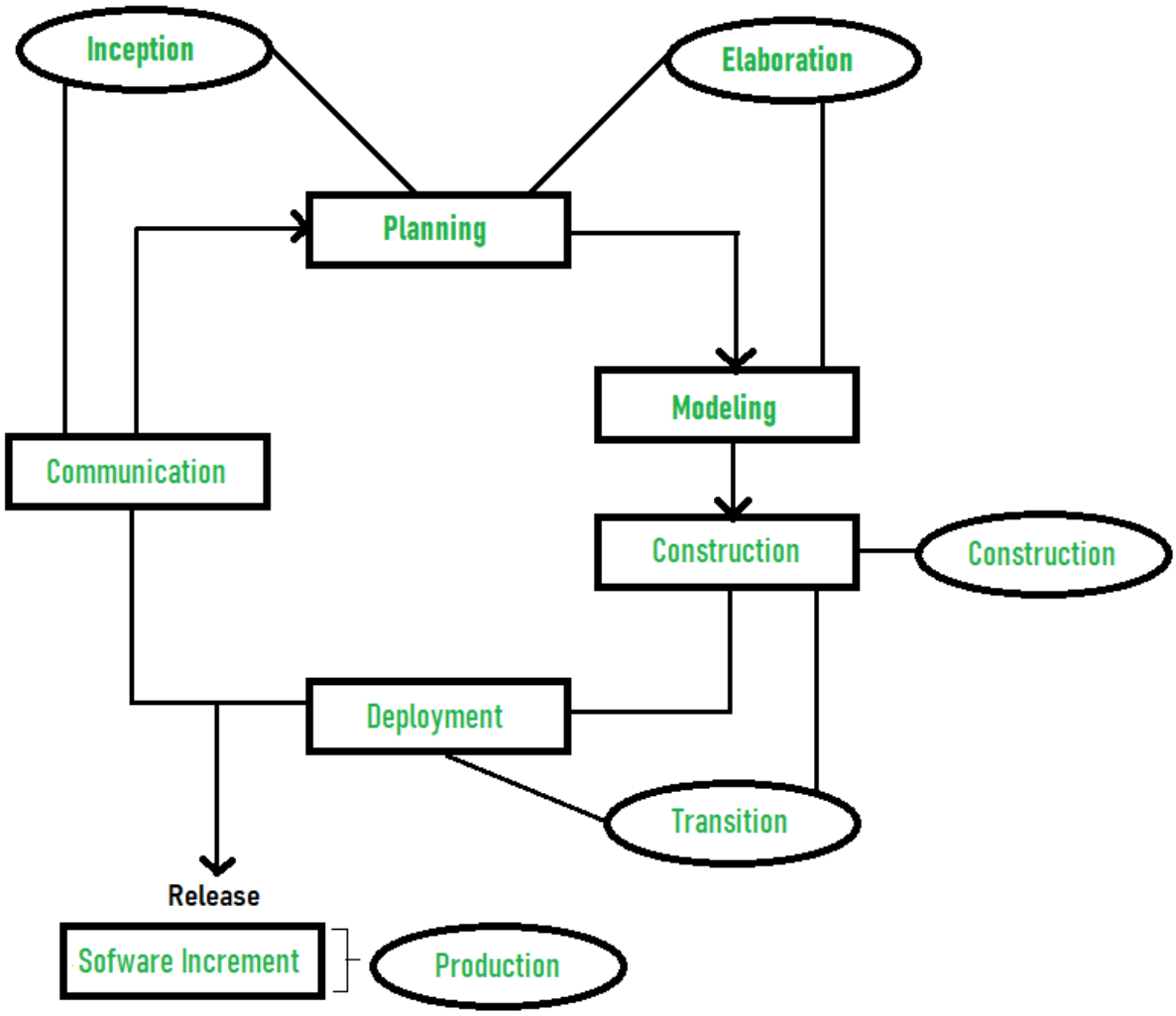
During Construction, the project team performs tasks that involve building the system iteratively and incrementally (see "Iterations and Increments" later in this chapter), making sure that the viability of the system is always evident in executable form.

Transition

The primary goal of the **Transition phase** is to roll out the fully functional system to customers.

During Transition, the project team focuses on correcting defects and modifying the system to correct previously unidentified problems.

The major milestone associated with the Transition phase is called **Product Release**



8.Demonstrate Unified approach of modeling?

Unified Modeling Language (UML) provides a standardized way to model software systems using diagrams and notations. The unified approach of modeling refers to the use of UML as a single, integrated modeling language for all aspects of software development, including requirements analysis, design, implementation, and testing.

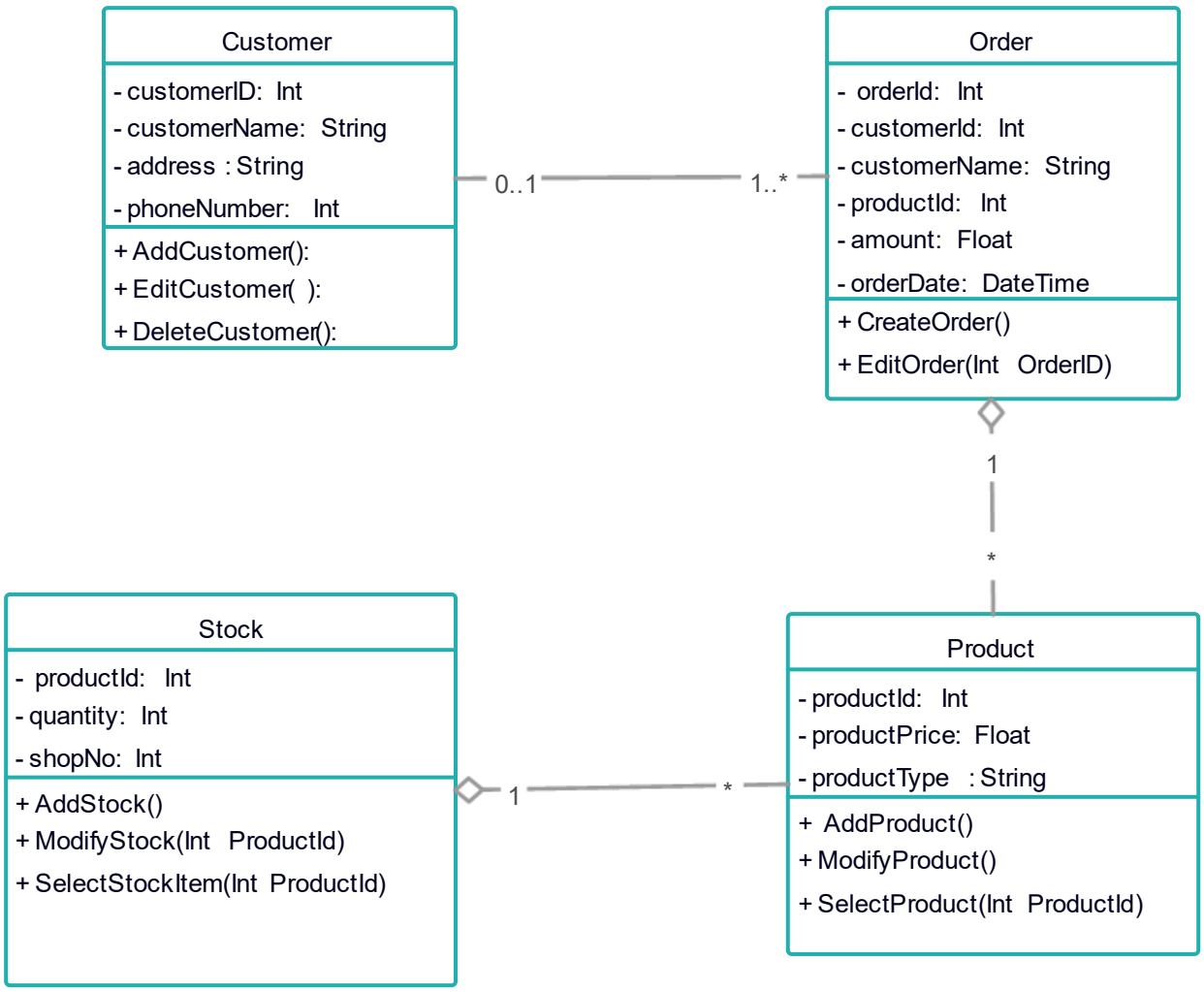
Here's an example of a unified approach to modeling using UML:

1. Requirements Analysis: During the requirements analysis phase, a use case diagram can be used to model the system's functional requirements. Use cases describe the interactions between actors (users, systems, or other entities) and the system.

2. Design: Next, a class diagram can be used to model the system's static structure, including classes, their attributes, and their relationships. Additionally, sequence diagrams can be used to model the system's dynamic behavior and how objects interact with each other.
3. Implementation: During the implementation phase, the class diagram can be used as a blueprint for writing code. The sequence diagrams can be used to ensure that the code correctly implements the system's behavior.
4. Testing: Finally, the use case diagram and sequence diagrams can be used to develop test cases to verify that the system satisfies its requirements.

By using UML throughout the software development process, the unified approach of modeling helps ensure consistency and traceability across all aspects of the system. It also makes it easier for stakeholders to understand the system's design and behavior, which can improve communication and collaboration between developers, testers, and other stakeholders.

9.Analyze Order management system and draw class diagram for order management system?

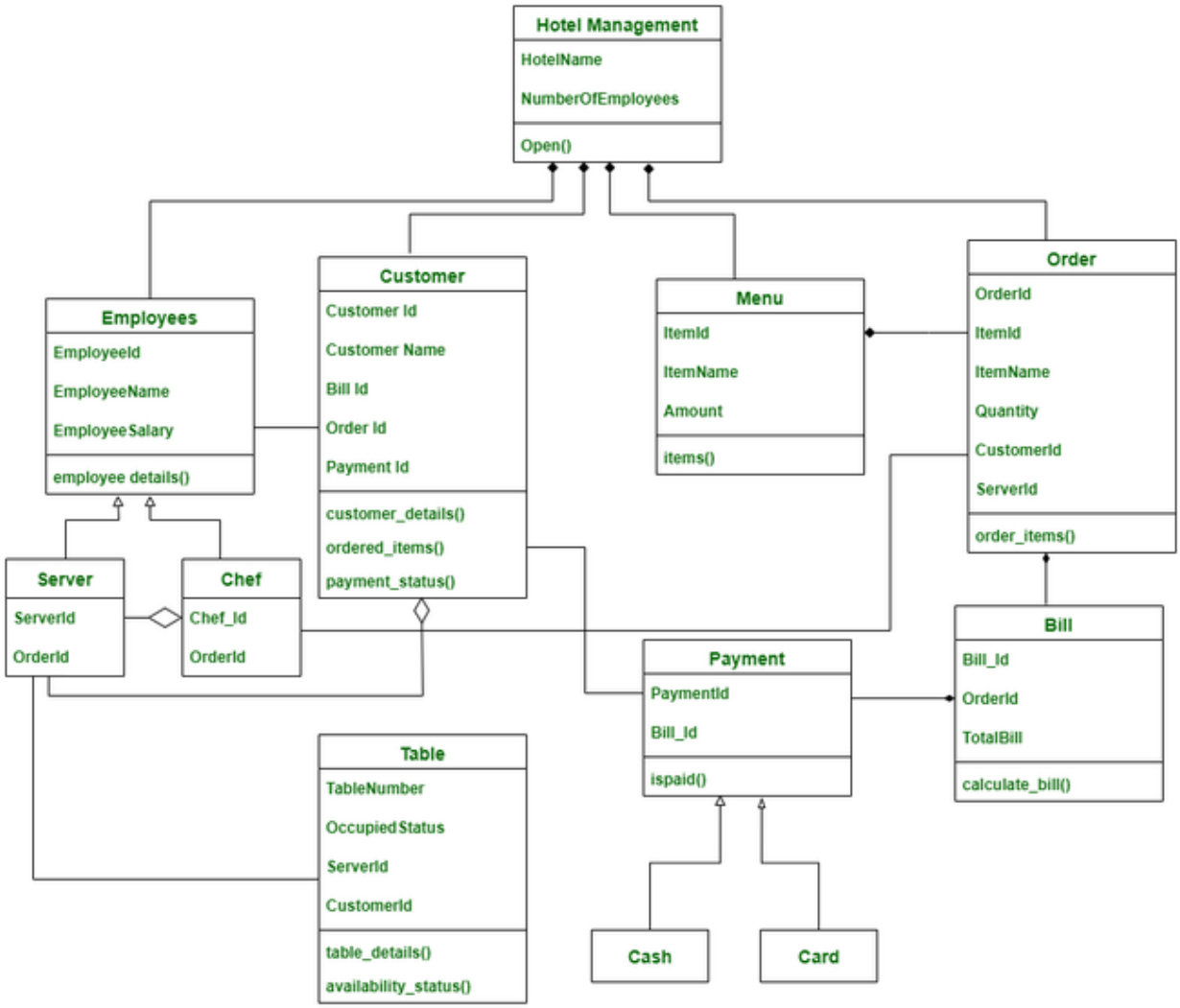


10.Illustrate elements of object model and list out different classes and objects for Hotel management system?

Elements of the Object Model

- Abstraction.
- Encapsulation.
- Modularity.
- Hierarchy.

Refer Question 6



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 :

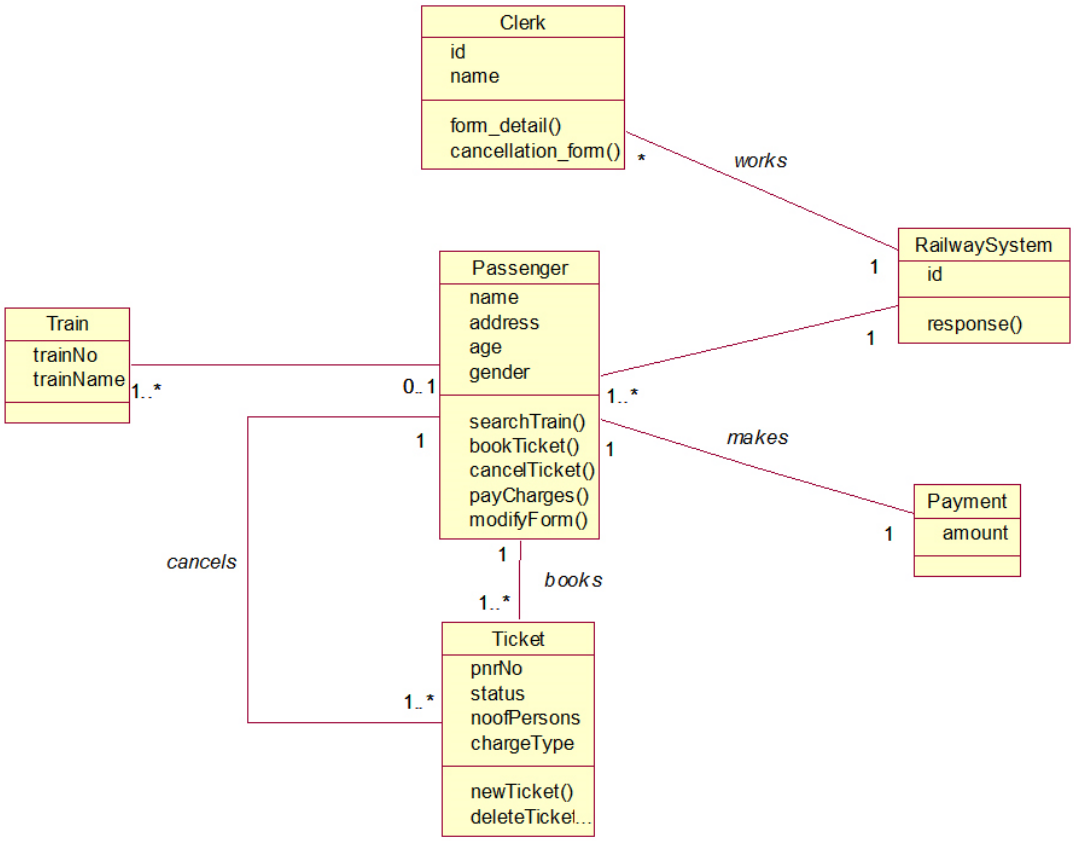
- **Hotel Management** – HotelName, NumberOfEmployees
- **Employees** – EmployeeId, EmployeeName, EmployeeSalary
- **Server** – ServerId, OrderId
- **Chef** – Chef_Id, OrderId
- **Customer** – CustomerId, CustomerName, Bill_Id, OrderId, PaymentId
- **Table** – TableNumber, OccupiedStatus, ServerId, CustomerId
- **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.

Object is an instance of a class in a particular moment in runtime that can have its own state and data values. Likewise a static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships which may be considered a special case of a class diagram or a communication diagram.

The following are the application areas where the object diagrams can be used.

1. To build a prototype of a system.
2. To model complex data structures.
3. To perceive the system from a practical perspective.
4. Reverse engineering.



12.Discuss about principles of modelling?

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.

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.
- Other times, you need to get down to the level of the studs for instance, when there's a tricky pipe run or an unusual structural element.

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:

- You can study electrical plans in isolation
- But you can also see their mapping to the floor plan and perhaps even their interaction with the routing of pipes in the plumbing plan.