



ABES Engineering College, Ghaziabad

B.Tech Odd/Even Semester Sessional Test-1

Course Code: KCS-054

Course Name: Object Oriented System Design

Maximum Marks: 75

Printed Pages:2

Session: 2022-23

Roll No.

Date of Exam:20/10/ 2022

Time: 2 Hours

Instructions:

1. Attempt All sections.

2. If require any missing data, then choose suitably.

Q.No.	Question	Marks	CO	KL	PI
Section-A					
1	Attempt ALL Parts	(5x2=10)			
a)	What is Difference between Object Oriented Programming and Procedural Oriented Programming.	2	CO1	K1	1.1.2
b)	Explain Building block of UML(Conceptual)	2	CO2	K2	1.1.2
c)	Differentiate between Association and Aggregation.	2	CO2	K2	1.1.2
d)	Discuss Multiplicity in relationships with examples.	2	CO2	K2	1.3.1
e)	Differentiate between Sequence diagram and Activity diagram.	2	CO3	K2	1.4.1
Section-B					
2	Attempt ANY ONE part from the following	(1x5=5)			
a)	Define Modelling. Explain types of modelling and discuss the principle of modelling.	5	CO2	K2	1.3.1
b)	Describe Abstraction, Modularity, Encapsulation in OO development.	5	CO3	K2	1.3.1
3	Attempt ANY ONE part from the following	(1x5=5)			
a)	Why Object-Oriented Programming (OOP) is so important for software industries or in real life? Explain with example. Discuss the pros and cons of object-oriented technology with suitable example.	5	CO2	K2	1.4.1
b)	Draw a Use Case Diagram for Online Shopping.	5	CO2	K3	3.1.2
4	Attempt ANY ONE part from the following	(1x5=5)			
a)	Explain various Relationships in UML. Discuss each with example.	5	CO2	K2	1.4.1
b)	How physical aspect of Object-Oriented System deal through UML? Explain in brief	5	CO2	K2	1.4.1

Section-C					
5	Attempt ANY ONE part from the following	(1x10=10)			
a)	What do we mean by a collaboration diagram? Explain various terms and symbols used in a collaboration diagram. Draw a Collaboration diagram for online book shop.	10	CO2	K3	3.1.2
b)	List the properties of a state chart diagram. Draw a state chart for a coin vending machine present at a railway station.	10	CO2	K3	3.1.2
6	Attempt ANY ONE part from the following	(1x10=10)			
a)	How is an Activity diagram different with Flowchart? Explain properties of Activity diagram and draw the Activity diagram for Hospital Management System.	10	CO2	K3	3.1.2
b)	How is a package diagram useful in UML? Explain its components.	10	CO2	K2	1.4
7	Attempt ANY ONE part from the following	(1x10=10)			
a)	Consider the scenario of ATM Machine, Draw a sequence diagram for ATM machine explain various and objects and messages	10	CO2	K2	3.1.2
b)	Explain the following concepts: i) Object Oriented Analysis ii) Object oriented design, iii) Polymorphism	10	CO3	K2	1.4.1
8	Attempt ANY ONE part from the following	(1x10=10)			
a)	Explain purpose and importance of Use Case Diagram. Explain the properties of use case diagram. Draw Use case diagram for Student Management System.	10	CO2	K3	3.1.2
b)	Draw Class diagram to show the relationship between Department (CSE) and subjects running in a specific Semester. Also define the Object properties and its behaviour in separate object diagram.	10	CO2	K3	3.1.2
9	Attempt ANY ONE part from the following	(1x10=10)			
a)	Define UML? Explain goal of UML. discuss the aspects of UML. Where can we use UML also discuss the advantages and disadvantages	10	CO2	K2	1.4.1
b)	Explain Deployment Diagram with its components. Draw a deployment diagram for Online Railway Reservation.	10	CO2	K3	3.2.3

CO Course Outcomes mapped with respective question

KL Bloom's knowledg Level (K1, K2, K3, K4, K5, K6)

K1- Remember, K2- Understand, K3-Apply, K4- Analyze, K5:

Evaluate, K6- Create

Solution OOSD(KCS-054) ST-1

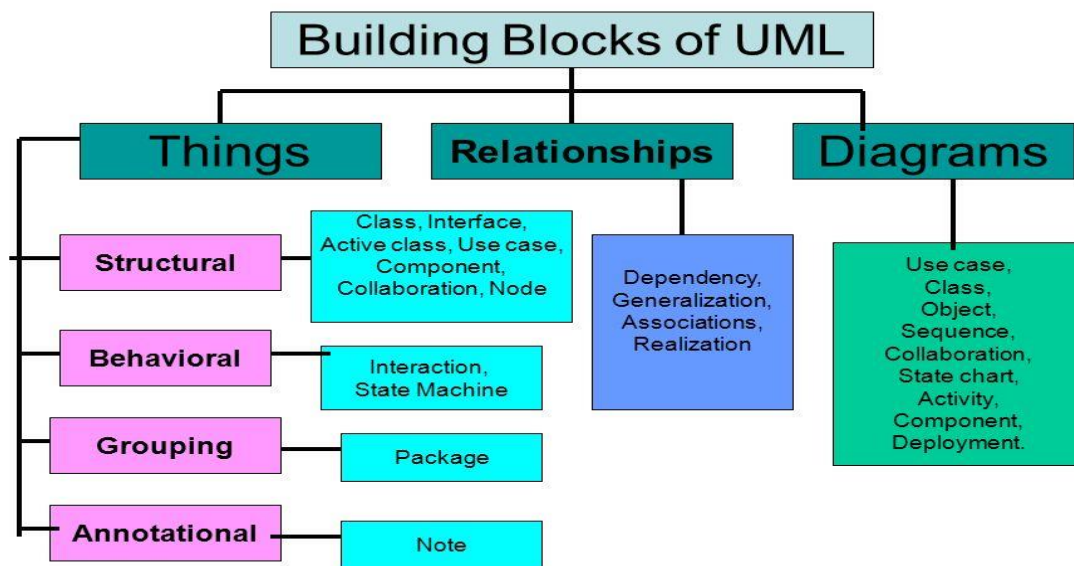
Section A

1(a).

S. N.	Features	Object Oriented Approach (OOP)	Procedure Oriented Approach (POP)
1	Full Form	OOP stands for Object Oriented Programming.	POP stands for Procedural Oriented Programming.
2	Approach	OOP follows bottom-up approach – focused on building blocks and components.	POP follows top-down approach - focused on inputs and outputs
3	Division	A program is divided to objects and their interactions.	A program is divided into functions and they interact.
4	Inheritance supported	YES	NO support
5	Access control	Access control is supported via access modifiers.	No access modifiers are supported.
6	Data Hiding	Encapsulation is used to hide data.	No data hiding present. Data is globally accessible.
7	Example	C++, Java, Python, Ruby, GoLang	C, Pascal

Object Orientation Languages (OOL) is concerned to develop an application based on real time while Procedural Programming Languages (PPL) are more concerned with the processing of procedures and functions. In OOL, more emphasis is given on data rather than procedures, while the programs are divided into Objects and the data is encapsulated (Hidden) from the external environment, providing more security to data which is not applicable or rather possible in PPL. In PPL, it's possible to expose Data and/or variables to the external entities which is STRICTLY restricted IN OOL.






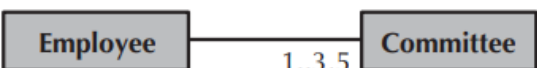
1(b).



1(C).

Association	Aggregation
<p>Association is used to represent the relationship between two elements of a system.</p> <p>The two ends represent two associated elements as shown in the following figure.</p> <p>The multiplicity is also mentioned at the ends (1, *, etc.) to show how many objects are associated.</p>	<p>In Aggregation, parent and child entity maintain Has-A relationship but both can also exist independently.</p> <p>We can use parent and child entity independently.</p> <p>Any modification in the parent entity will not impact the child entity or vice versa.</p> <p>In the UML diagram, aggregation is denoted by an empty diamond, which shows their obvious difference in terms of strength of the relationship.</p>
<p> Person * <i>WorksFor</i> 0..1 Company employee employer Joe Doe Simplex Mary Brown Simplex Jean Smith United Widgets </p>	<p> Company 1 <i>aggregation</i> Department * whole part </p>

1(d). Multiplicity:-When multiple classes share a relationship, a line is drawn and labeled with either the name of the relationship or the roles that the classes play in the relationship

Exactly one	1		A department has one and only one boss.
Zero or more	0..*		An employee has zero to many children.
One or more	1..*		A boss is responsible for one or more employees.
Zero or one	0..1		An employee can be married to zero or one spouse.
Specified range	2..4		An employee can take from two to four vacations each year.
Multiple, disjoint ranges	1..3,5		An employee is a member of one to three or five committees.

1.(e).

Sequence Diagram	Activity Diagram
The Sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	The Activity diagram represents the UML, which is used to model the workflow of a system.
The Sequence diagram shows the message flow from one object to another object.	The Activity diagram shows the message flow from one activity to another.
Sequence diagram is used for the purpose of dynamic modelling.	Activity diagram is used for the purpose of functional modelling.
Sequence diagram is used to describe the behavior of several objects in a single use case	Activity diagrams is used to describe the general sequence of actions for several objects and use cases.
Sequence diagram is mainly used to represent the time order of a process.	Activity diagram is used to represent the execution of the process.

Section-B

2(a).

MODELLING: -

Modelling can help the development team better visualize the plan of their system and allow them to develop more rapidly by helping them build the right thing. The more complex your project, the more likely it is that you will fail or that you will build the wrong thing if you do on modelling at all. It leads to the deployment of good software. It is required to build quality software.

Communicate the desired structure and behavior of our system. To visualize and control the system's architecture. Better understand the system we are building, often exposing opportunities for simplification and reuse. We build models to manage risk."

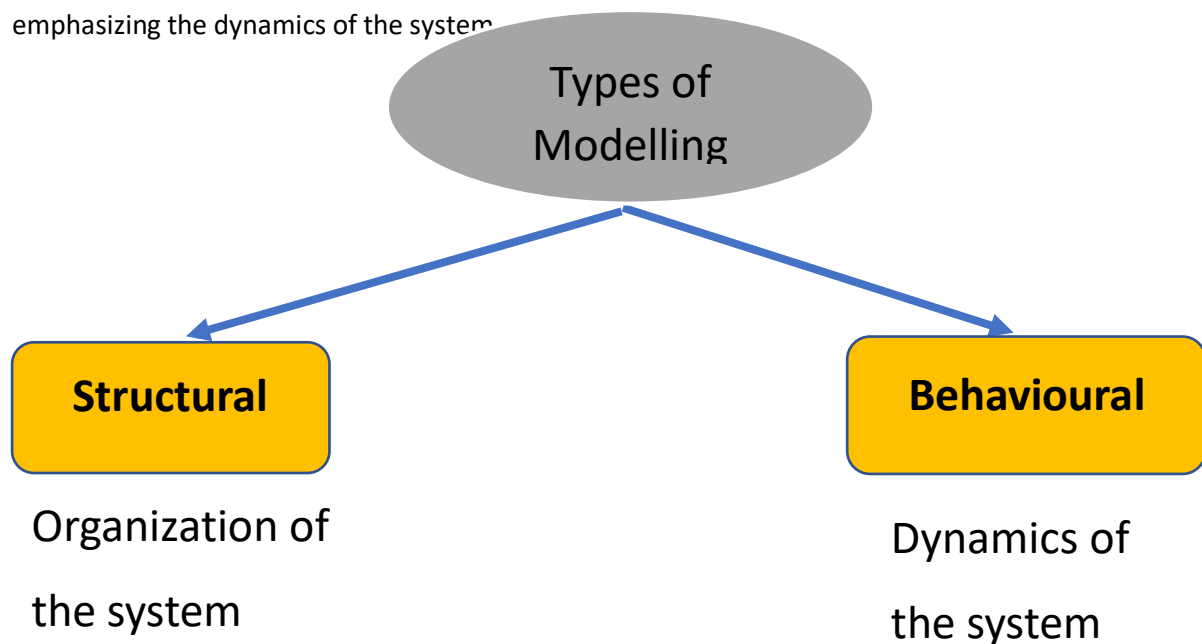
For example: If you want to build a house for your family with same things, is it possible? It requires detailed planning, some sketches etc.

Architecting a house is best achieved by a team. It requires detailed modelling, well-defined processes and powerful tools.



TYPES OF MODELLING:-

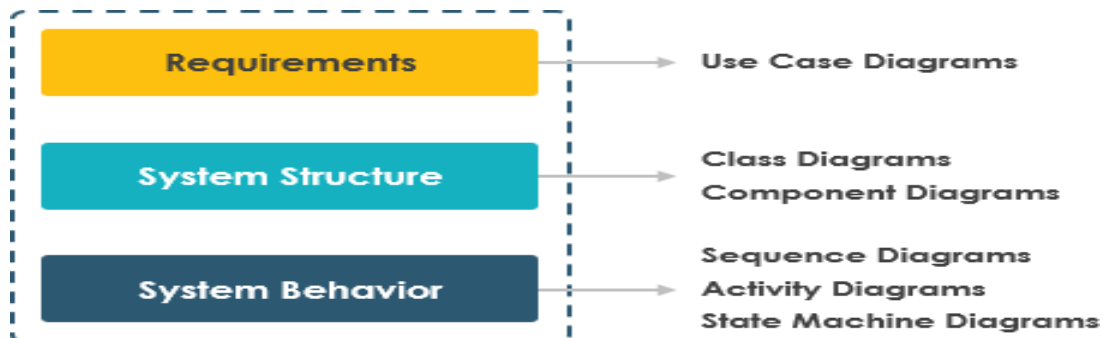
A model may be structural, emphasizing the organization of the system, or it may be behavioral, emphasizing the dynamics of the system



PRINCIPLES OF MODELLING: -

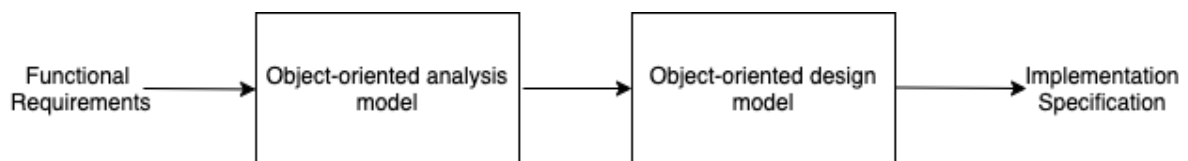
The choice of what models to create. Every model may be expressed at different levels of precision.

The best models are connected to reality No single model is sufficient



- It is the process of preparing and designing what the model's code will actually look like.
- Implemented by using the object-oriented programming model.
- OOM consists three phases: **analysis, design, and implementation.**

Allows for object identification and communication while supporting data abstraction, inheritance and encapsulation.



2(b).

ABSTRACTION: - Abstraction is the concept of object-oriented programming that “shows” only essential attributes and “hides” unnecessary information. The main purpose of abstraction is hiding the unnecessary details from the users. Abstraction is selecting data from a larger pool to show only relevant details of the object to the user. It helps in reducing programming complexity and efforts. It is one of the most important concepts of OOPs.

MODULARITY: - Modularity is a frequently used term in information technology and computer science. Modularity refers to the concept of making multiple modules first and then linking and combining them to form a complete system. Modularity enables re-usability and minimizes duplication. In addition to re-usability, modularity also makes it easier to fix problems as bugs can be traced to specific system modules, thus limiting the scope of detailed error searching. Modular programming is an extensively used concept based on modularity. Modularity is also a feature of object-oriented programming.

ENCAPSULATION: -Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. It is a protective shield that prevents the data from being accessed by the code outside this shield.

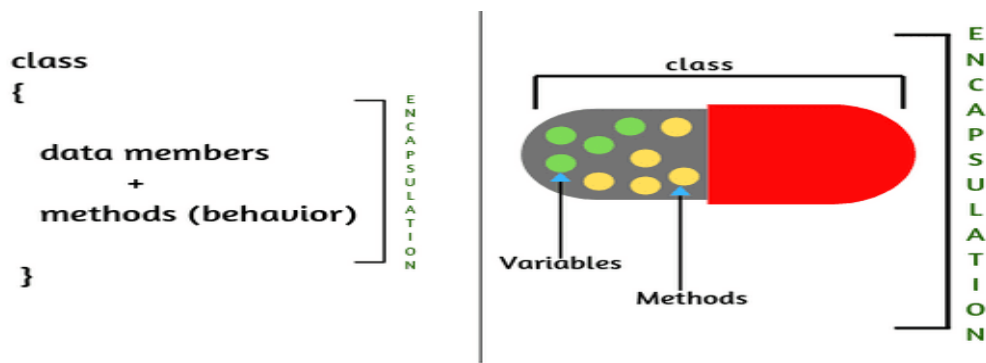


Fig: Encapsulation

Difference between Abstraction and Encapsulation

Abstraction

Abstraction in Object Oriented Programming solves the issues at the design level.

Abstraction in Programming is about hiding unwanted details while showing most essential information.

Data Abstraction in Java allows focussing on what the information object must contain

Encapsulation

Encapsulation solves it implementation level.

Encapsulation means binding the code and data into a single unit.

Encapsulation means hiding the internal details or mechanics of how an object does something for security reasons.

3(a). OOP's is so important in software industries or real life because it deal with.

There are five important concepts which help to deal with real life problem.

- 1)Class
- 2) Object
- 3) Inheritance
- 4)Data Abstraction
- 5) Data Encapsulation
- 6)Polymorphism

Advantages of OOP

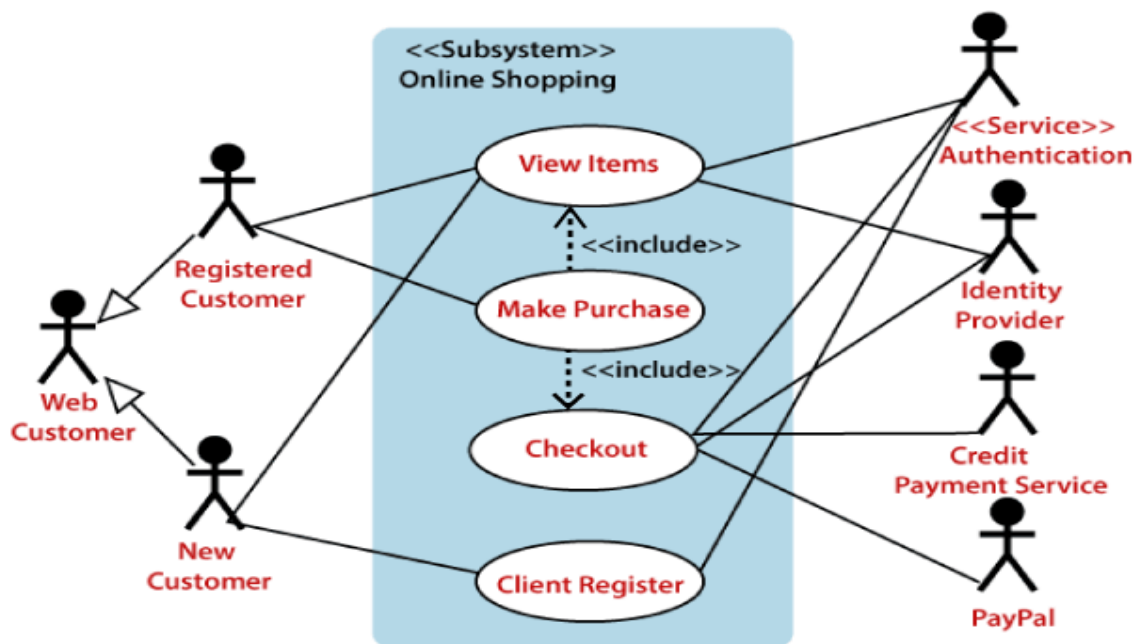
- We can build the programs from standard working modules that communicate with one another, rather than having to start writing the code from scratch which leads to saving of development time and higher productivity,
- OOP language allows to break the program into the bit-sized problems that can be solved easily (one object at a time).
- The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.
- OOP systems can be easily upgraded from small to large systems.
- It is possible that multiple instances of objects co-exist without any interference,
- It is very easy to partition the work in a project based on objects.
- It is possible to map the objects in problem domain to those in the program.
- The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.
- By using inheritance, we can eliminate redundant code and extend the use of existing classes.

- Message passing techniques is used for communication between objects which makes the interface descriptions with external systems much simpler.
- The data-centered design approach enables us to capture more details of model in an implementable form.

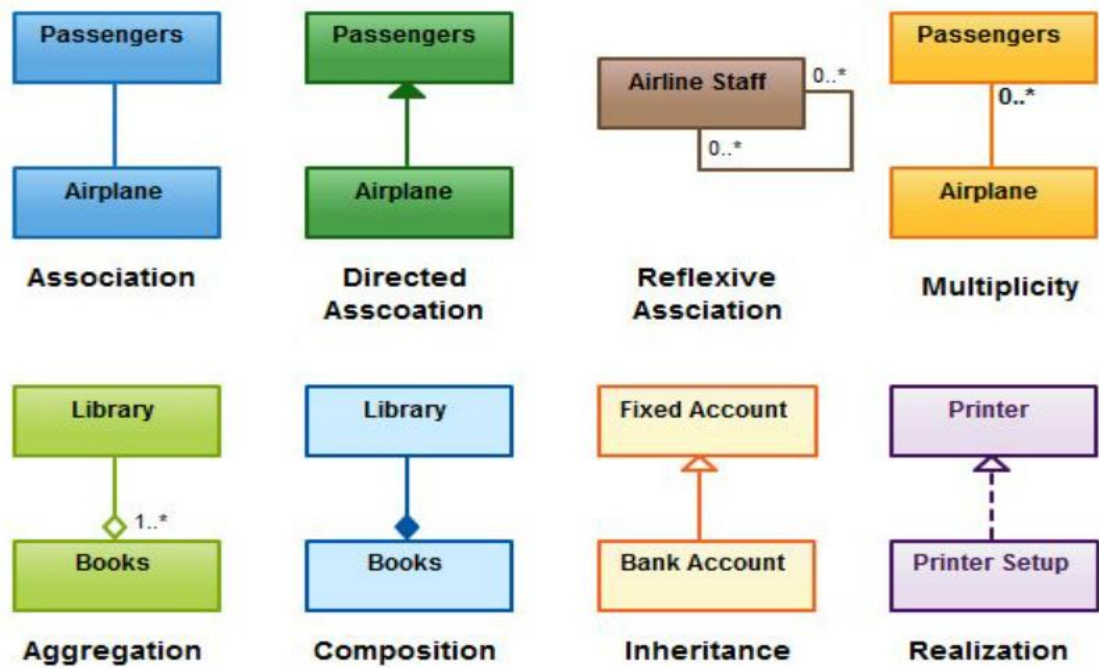
Disadvantages of OOP

- The length of the programmes developed using OOP language is much larger than the procedural approach. Since the programme becomes larger in size, it requires more time to be executed that leads to slower execution of the programme.
- We can not apply OOP everywhere as it is not a universal language. It is applied only when it is required. It is not suitable for all types of problems.
- Programmers need to have brilliant designing skill and programming skill along with proper planning because using OOP is little bit tricky.
- OOPs take time to get used to it. The thought process involved in object-oriented programming may not be natural for some people.
- Everything is treated as object in OOP so before applying it we need to have excellent thinking in terms of objects.

3(b). Use case Diagram for Online Shopping.



4(a). Various relationship.

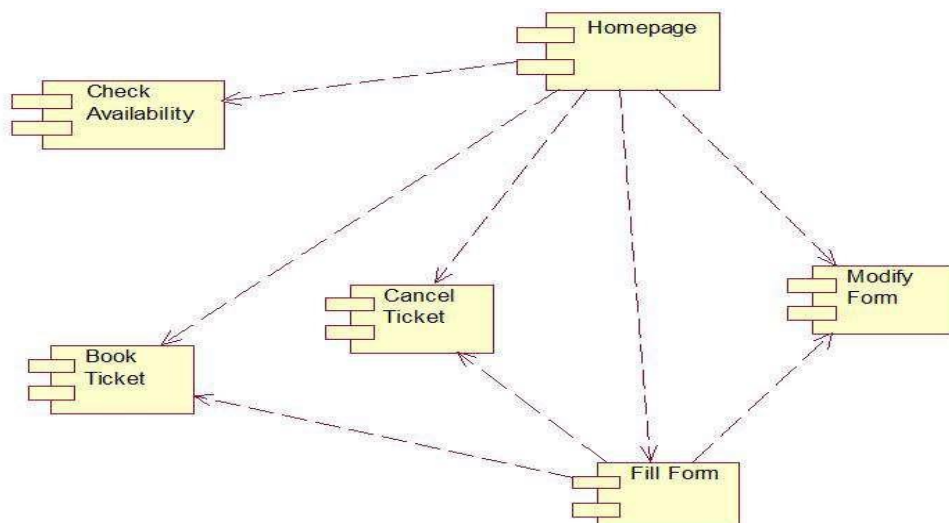


4(b). Component diagrams are one of the two kinds of diagrams found in modeling the physical aspects of object-oriented systems.

Elements of a Component Diagram

- 1.Ports.
- 2.Connector.
- 3.Components
4. interfaces

Online railway reservation form.

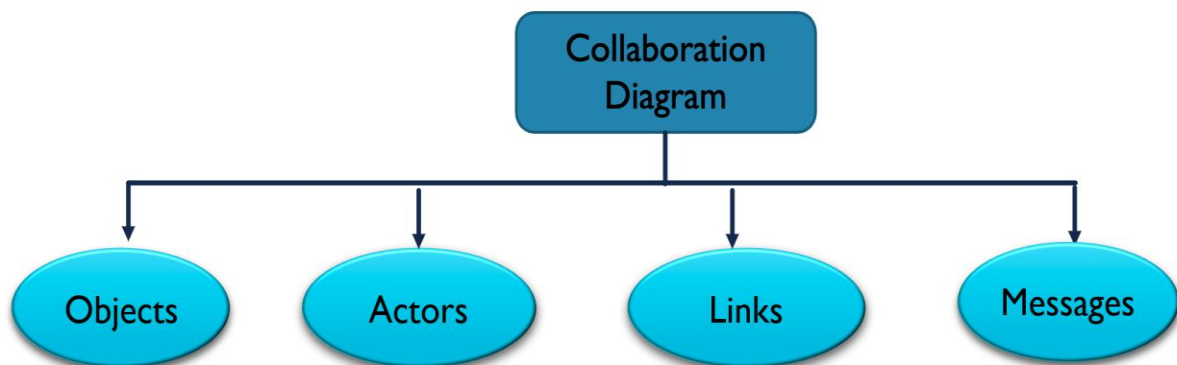


Section-C

5(a). The collaboration diagram is used to show the relationship between the objects in a system.

- It depicts the architecture of the object residing in the system as it is based on object-oriented programming. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system. They tend to be better suited to depicting simpler interactions of smaller numbers of objects. It is difficult to show additional descriptive information such as timing, or other unstructured information that can be easily added to the notes in a sequence diagram.

Following symbols are used in collaboration diagram



1.Objects

Each object is also called a collaborator. An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon:

Object_name : class_name

Each object in the collaboration is named and has its class specified. The objects can be unnamed, but you should name them if you want to discriminate different objects of the same class. In the collaboration diagram, firstly, the object is created, and then its class is specified.

2.Actors

Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction. If you have several actor instances in the same diagram, try keeping them in the periphery of the diagram.

- Each Actor is named and has a role
- One actor will be the initiator of the use case

3.Links

Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram

Links are defined as follows: A link is a relationship among objects across which messages can be sent. In collaboration diagrams, a link is shown as a solid line between two objects. An object interacts with, or navigates to, other objects **through its links to these objects**. A link can be an instance of an association, or it can be anonymous, meaning that its association is unspecified.

Message flows are attached to links, see Messages.

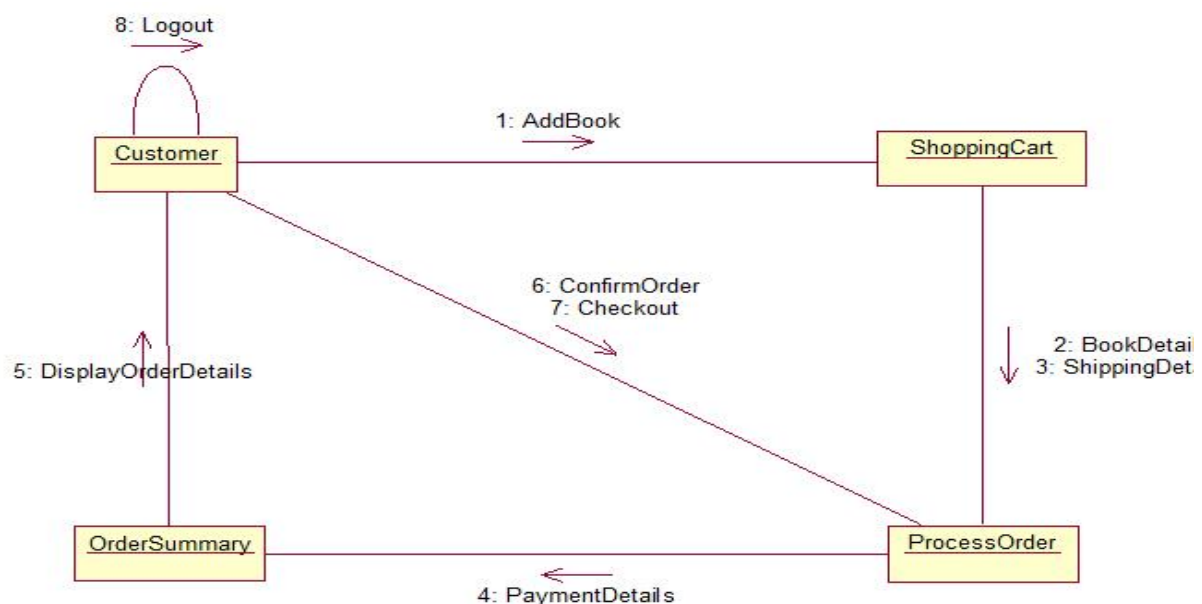
4.Messages

A message is a communication between objects that conveys information with the expectation that activity will ensue.

In collaboration diagrams, a message is shown as a labelled arrow placed near a link.

- The message is directed from sender to receiver
- The receiver must understand the message
- The association must be navigable in that direction.

Collaboration Diagram for ONLINE BOOK SHOP



5(b). A state is an abstraction of the attribute values and links of an object. Sets of values are grouped together into a state according to properties that affect the gross behavior of the object."

State Diagram are used to capture the behavior of a software system. UML State machine diagrams can be used to model the behavior of a class, a subsystem, a package, or even an entire system. It is also called a Statechart or State Transition diagram.

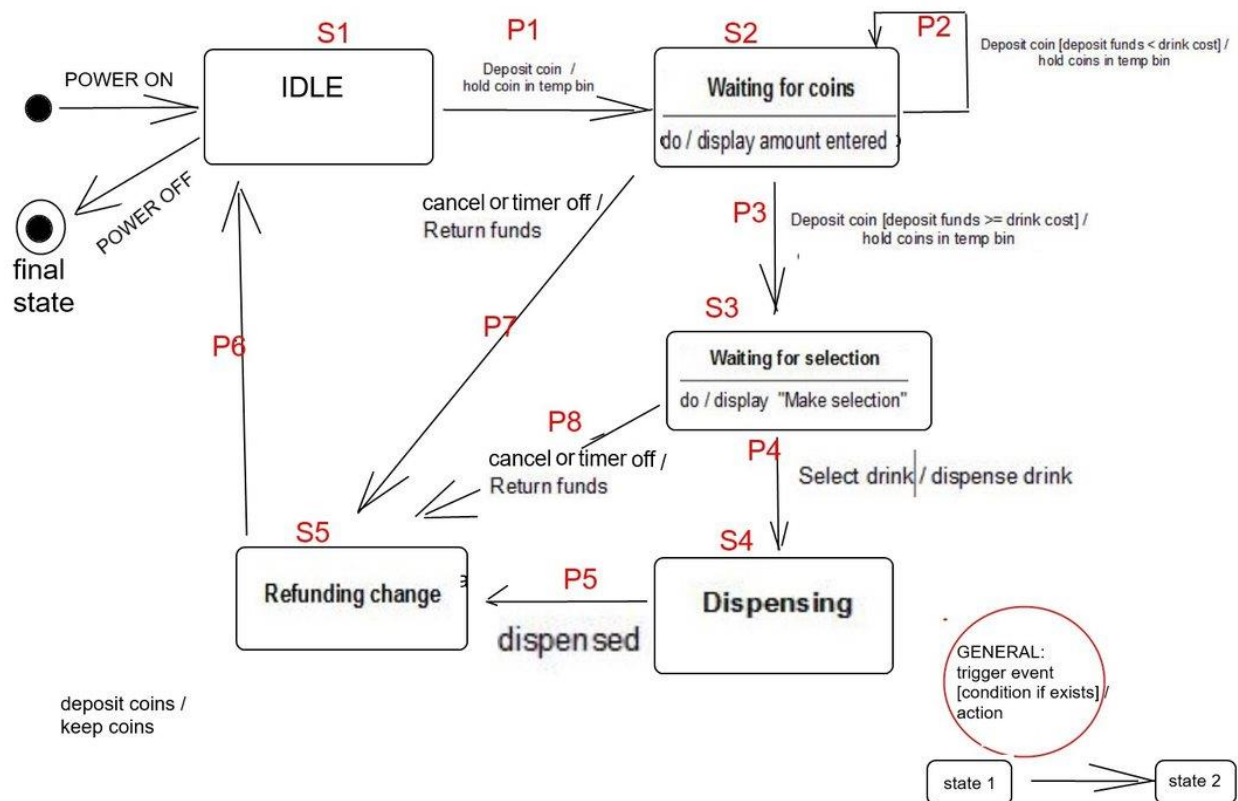
Statechart diagrams provide us an efficient way to model the interactions or communication that occur within the external entities and a system. These diagrams are used to model the event-based

system. A state of an object is controlled with the help of an event. Statechart diagrams are used to describe various states of an entity within the application system.

Properties of state chart diagram

Since it records the dynamic view of a system, it portrays the behavior of a software application. During a lifespan, an object underwent several states, such that the lifespan exist until the program is executing. Each state depicts some useful information about the object. It blueprints an interactive system that response back to either the internal events or the external ones. The execution flow from one state to another is represented by a state machine diagram. It visualizes an object state from its creation to its termination. The main purpose is to depict each state of an individual object. It represents an interactive system and the entities inside the system. It records the dynamic behavior of the system.

State Chart Diagram of Vending Machine



6(a).

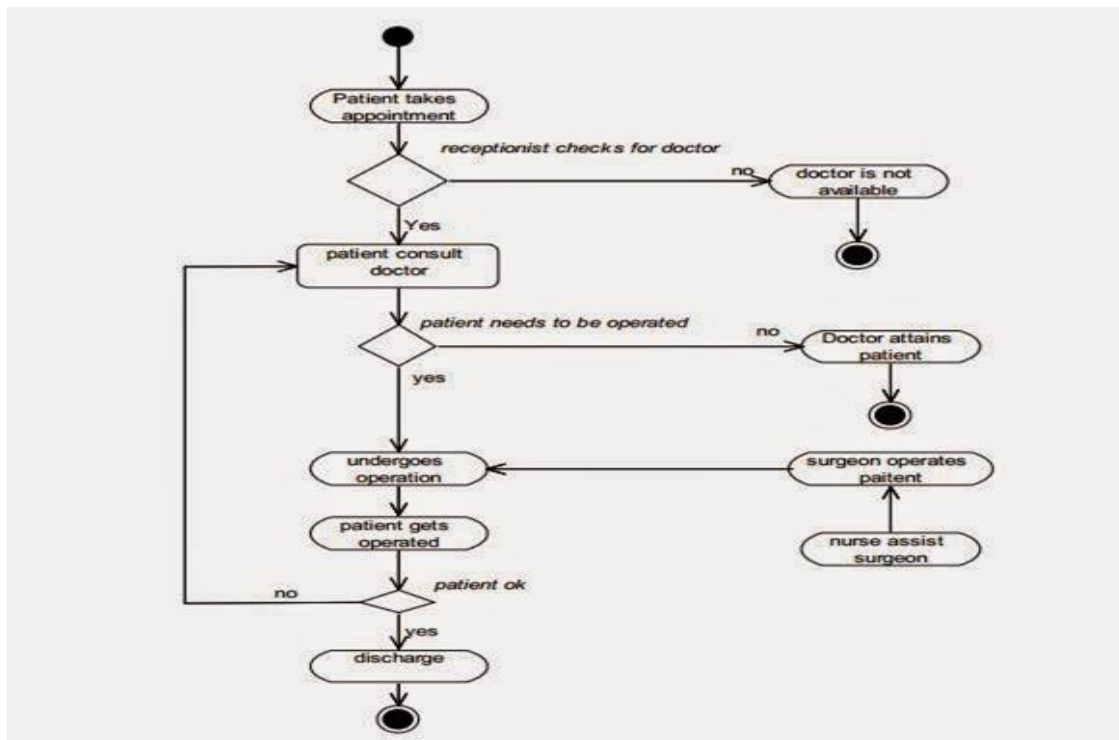
ACTIVITY DIAGRAM	FLOWCHART
A graphical representation of workflows of stepwise activities and actions with support for choice, iteration, and concurrency	A diagrammatic representation that illustrates a solution model to a given problem
Helps to understand the business process or workflow of the system	Helps to analyze and design a program
Associated with UML	Associated with programming

Properties

Activity Diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The basic purpose of activity diagrams is to capture the dynamic behavior of the system.. It is also called object-oriented flowchart. This UML diagram focuses on the execution and flow of the behavior of a system instead of implementation. Activity diagrams consist of activities that are made up of actions that apply to behavioral modeling technology.

An event is created as an activity diagram encompassing a group of nodes associated with edges. To model the behavior of activities, they can be attached to any modeling element. It can model use cases, classes, interfaces, components, and collaborations. It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering. It does not include the message part, which means message flow is not represented in an activity diagram. It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities.

Activity Diagram for Hospital Management System

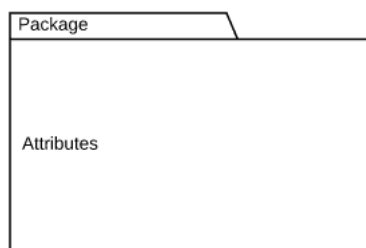


6(b).

- Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of related [UML elements](#), such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram.

The makeup of a package diagram is relatively simple. Each diagram includes only symbols:

1.Package



2.Dependency





Access: Indicates that one package requires assistance from the functions of another package.



Import: Indicates that functionality has been imported from one package to another.

7(a). ATM Use case scenario-

The Automated Teller Machine is a remote unit connected to the bank computer systems. The purpose of the system is to bring regular bank services closer to the customer and increase the working hours to around the clock. It is also important to decrease the amount of bank cashiers. An ATM withdrawal is less expensive for the Bank than a withdrawal from a teller. The ATM system requires that each bank customer has an ATM card and remembers his PIN code. The whole security of this system builds on the PIN code.

Validate User-This use case describes general behaviour for the ATM to validate the Bank Customer. It includes all steps that are the same no matter what kind of transaction the Bank Customer does.

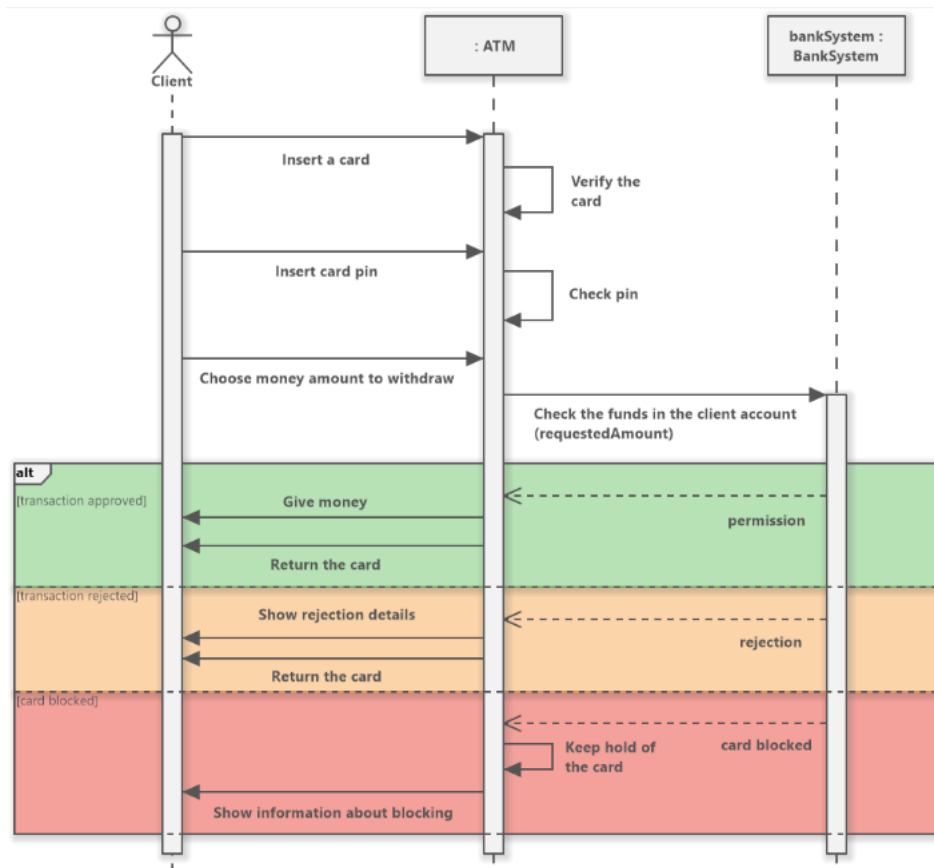
Withdraw Cash-This use case describes how the Bank Customer uses the ATM to withdraw money his/her bank account.

Transfer Fund-This use case describes how the Bank Customer uses the ATM to transfer money between different bank accounts.

Deposit Funds-This use case describes how the Bank Customer deposits money to an account.

Refill Machine-This use case describes how the Maintenance Person refills money, receipt paper a

Sequence diagram for ATM system-



7(b). (i) Object Oriented Analysis

- Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.
- object-oriented approach enables analysts to break a complex system into smaller, more-manageable modules, work on the modules individually, and easily piece the modules back together to form an information system.
- This modularity makes systems development easier to grasp, easier to share among members of a project team, and easier to communicate to users, who are needed to provide requirements and confirm how well the system meets the requirements throughout the systems development process.
- By modularizing systems development, the project team actually is creating reusable pieces that can be plugged into other systems efforts or used as starting points for other projects.

(i) Object Oriented Design-

- By applying object-oriented design, we create software that is resilient to change and written with economy of expression.
- Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design.
- term object-oriented design to refer to any method that leads to an object-oriented decomposition
- There are two important parts to this definition: object-oriented design (1) leads to an object-oriented decomposition and (2) uses different notations to express

different models of the logical (class and object structure) and physical (module and process architecture) design of a system, in addition to the static and dynamic aspects of the system.

(ii) **Polymorphsim-**

- It is one of the significant OOPS concepts. Using Polymorphism, you can have various or multiple forms of objects, variables, or methods. There can be varied implementations of the same method as per the class' need using Polymorphism.
 - The concept of polymorphism offers great scalability and boosts the code's readability. You can define polymorphism to have your unique implementation for the same method within the Parent class.
1. **Compile Time or Static Polymorphism-** With Method Overloading, static polymorphism is achieved in Object-Oriented Programming languages that allow the programmer to implement various methods. The names they use can be the same, but their parameters are different.
 2. **Runtime or Dynamic Polymorphism-**
Run time polymorphism is alternatively called dynamic polymorphism. This is because a method to call is decided at run time but not compiler time. Since both methods will own the same signature and could be called using any one of the class objects, the compile-time compiler doesn't know about this; it can only be identified during runtime.

8(a).

A use case diagram is used to represent the dynamic behaviour of a system. It depicts the high-level functionality of a system and also tells how the user handles a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

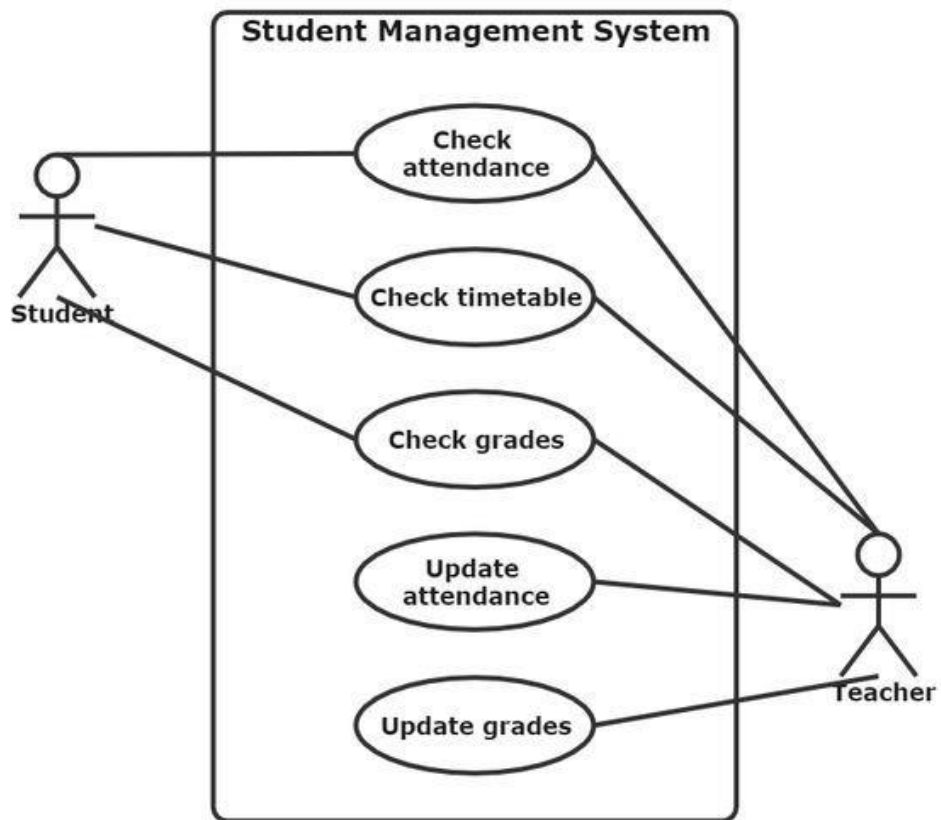
A use case diagram is usually simple. It does not show the detail of the use cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

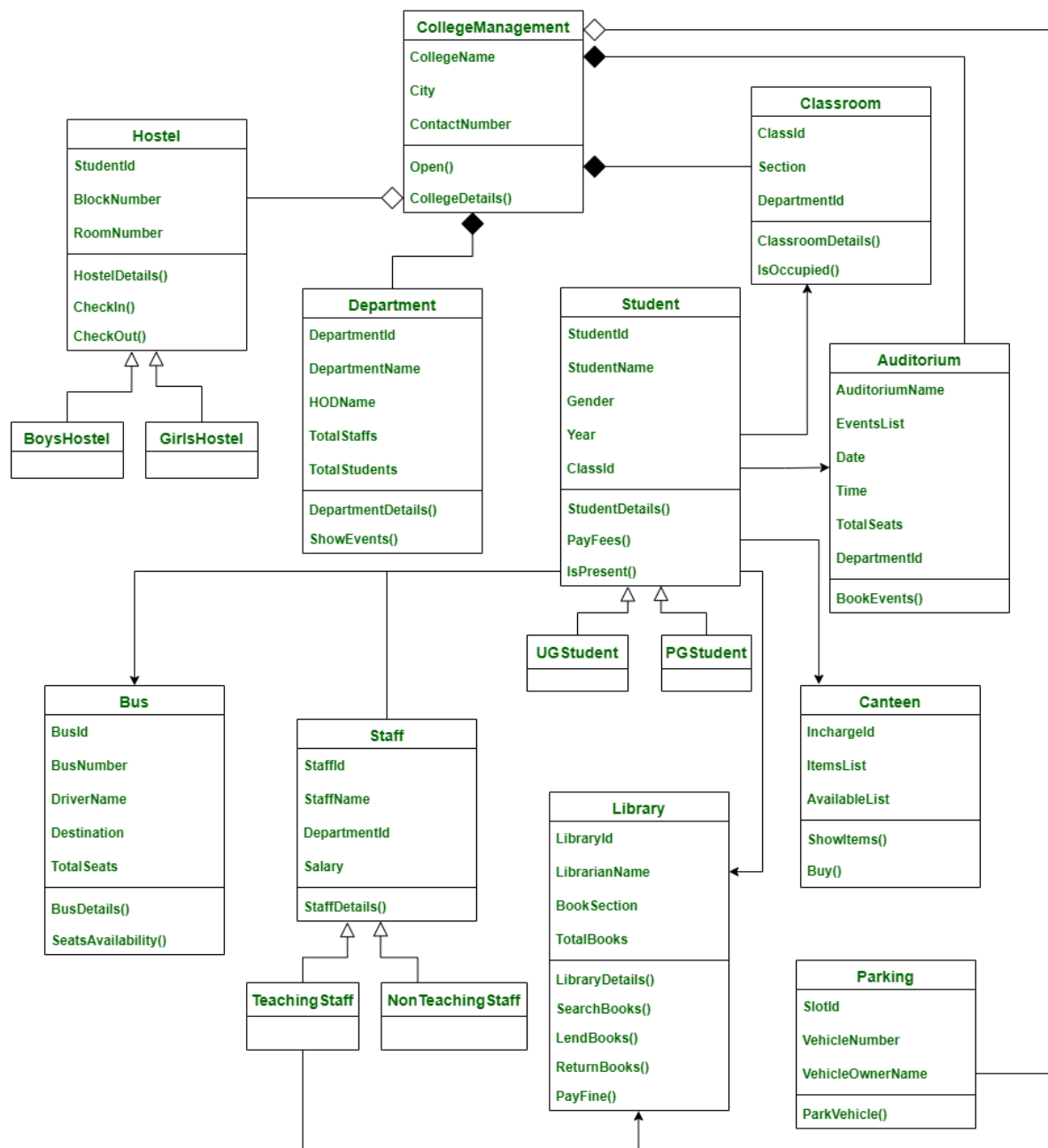
As said, a use case diagram should be simple and contains only a few shapes. If yours contain more than 20 use cases, you are probably misusing use case diagram.

Use Case Diagram for student management system

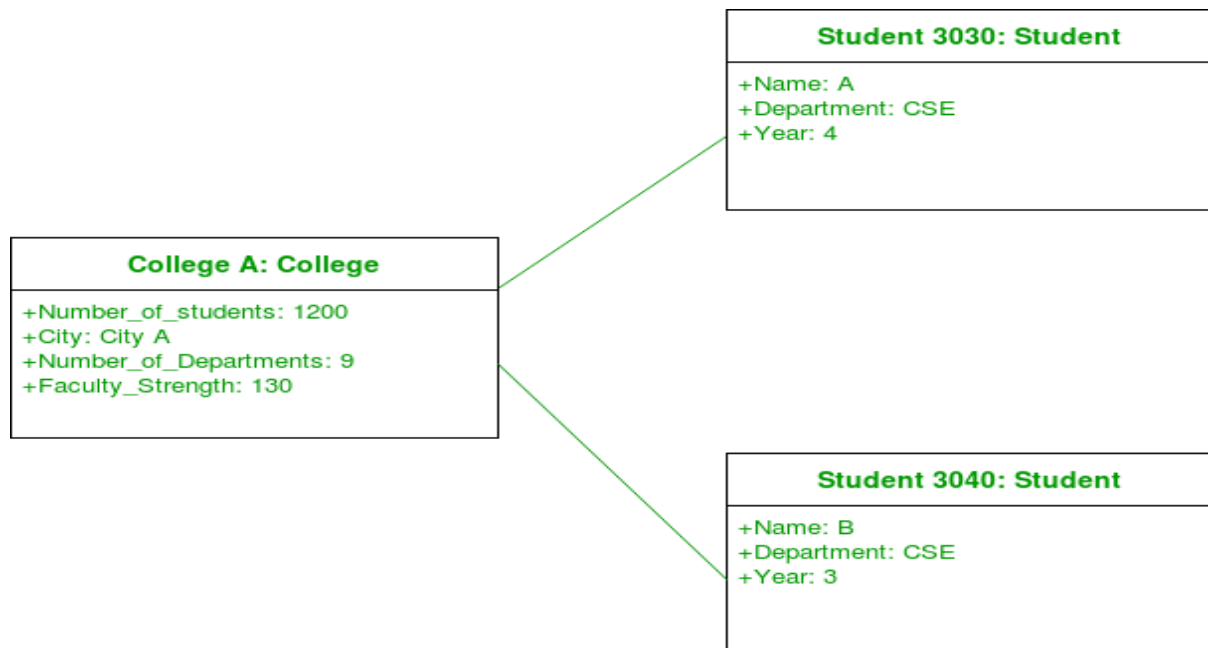
. Also define the Object properties and its behavior in separate object diagram.



8(b). Class Diagram for College Management System.



Object diagram:-



9(a).

UML stands for Unified Modelling Language. UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. OMG is continuously making efforts to create a truly industry standard. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modelling language to visualize, specify, construct, and document software system. UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language, but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design.

Aspects of UML:-

There are three aspects of UML

1. Language:

- It enables us to communicate about a subject which includes the requirements and the system.
- It is difficult to communicate and collaborate for a team to successfully develop a system without a language.

2. Model:

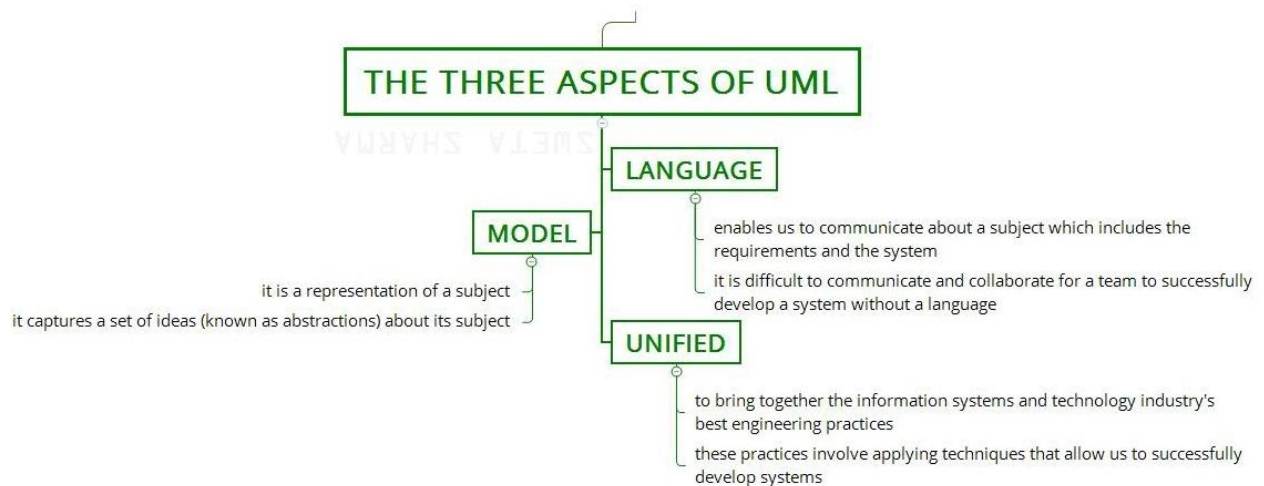
- It is a representation of a subject.
- It captures a set of ideas (known as abstractions) about its subject.

3. Unified:

- It is to bring together the information systems and technology industry's best engineering practices.

These practices involve applying techniques that allow us to successfully develop system

Where Can the UML Be Used



UML can be used

- Enterprise information systems
- Banking and financial services
- Telecommunications
- Transportation
- Defence/aerospace
- Retail
- Medical electronics
- Scientific
- Distributed Web-based service

Advantages of UML

- It has wide industry acceptance in comparison to various modelling language.
- It supports OOAD technology.
- It bridges the communication gap between different entities of system development (i.e, system Analyst, Developer, Client etc.)
- Constructed model are easy to understand, even for non-programmers.

- It is a unified and standardizes modeling language.

Disadvantages of UML

- UML is often criticized as being large and complex.
- It takes a lot of time to keep the diagram reasonable and synchronized with the actual code.
- UML software costs money.
- Complex to learn and takes time to master properly.

9(b).

UML Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

Purpose of Deployment Diagram

The main purpose of the deployment diagram is to represent how software is installed on the hardware component. It depicts in what manner a software interacts with hardware to perform its execution. Both the deployment diagram and the component diagram are closely interrelated to each other as they focus on software and hardware components. The component diagram represents the components of a system, whereas the deployment diagram describes how they are actually deployed on the hardware. The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

Following are the purposes of deployment diagram enlisted below:

- To envision the hardware topology of the system.
- To represent the hardware components on which the software components are installed.
- To describe the processing of nodes at the runtime.
- Symbol and notation of Deployment diagram

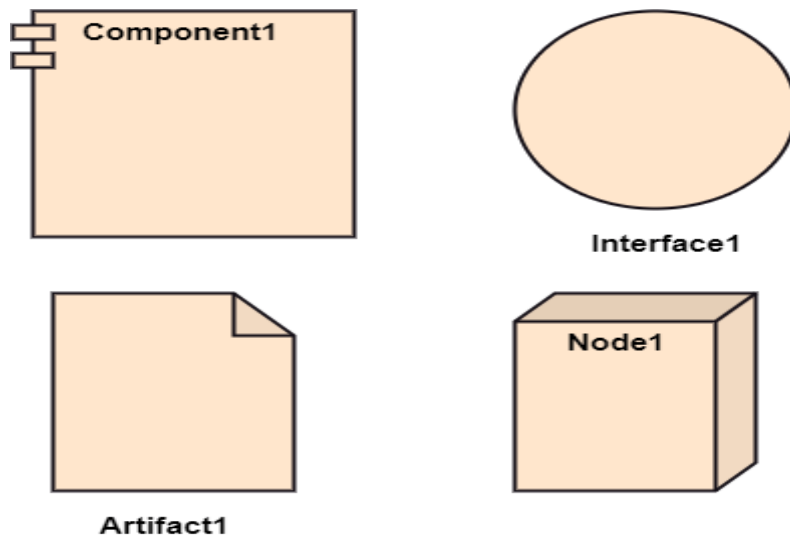
Components of Deployment diagram

1.A component

2.An artifact

3.An interface

4. A node



Deployment diagram for Railway Reservation System

