

## Semester - 5

# CS & IT

# Object Oriented System Design



- Topic-wise coverage of entire syllabus in Question-Answer form.
- Short Questions (2 Marks)

**Includes solution of following AKTU Question Papers**

2010-11 • 2011-12 • 2012-13 • 2013-14 • 2014-15 • 2015-16

# QUANTUM SERIES

---

*For*

B.Tech Students of Third Year  
of All Engineering Colleges Affiliated to  
**Dr. A.P.J. Abdul Kalam Technical University,**  
**Uttar Pradesh, Lucknow**  
(Formerly Uttar Pradesh Technical University)

## **Object Oriented System Design / Object Oriented Programming**

**By**

**Kanika Dhama**



**QUANTUM PAGE PVT. LTD.**  
**Ghaziabad ■ New Delhi**

**PUBLISHED BY :**            **Apram Singh**  
                                      **Quantum Publications®**  
                                      **(A Unit of Quantum Page Pvt. Ltd.)**  
                                      Plot No. 59/2/7, Site - 4, Industrial Area,  
                                      Sahibabad, Ghaziabad-201 010

**Phone :** 0120 - 4160479

**Email :** [pagequantum@gmail.com](mailto:pagequantum@gmail.com)    **Website:** [www.quantumpage.co.in](http://www.quantumpage.co.in)

**Delhi Office :** 1/6590, East Rohtas Nagar, Shahdara, Delhi-110032

© ALL RIGHTS RESERVED

*No part of this publication may be reproduced or transmitted,  
in any form or by any means, without permission.*

Information contained in this work is derived from sources believed to be reliable. Every effort has been made to ensure accuracy, however neither the publisher nor the authors guarantee the accuracy or completeness of any information published herein, and neither the publisher nor the authors shall be responsible for any errors, omissions, or damages arising out of use of this information.

**Object Oriented System Design (CS/IT : Sem-5)**

**Object Oriented Programming (OE : Sem-6)**

**1<sup>st</sup> Edition : 2020-21**

**Price: Rs. 65/- only**

# CONTENTS

## KCS-054 : OBJECT ORIENTED SYSTEM DESIGN

### UNIT-1 : INTRODUCTION

(1-1 L to 1-22 L)

The meaning of Object Orientation, object identity, Encapsulation, information hiding, polymorphism, generosity, importance of modelling, principles of modelling, object oriented modelling, Introduction to UML, conceptual model of the UML, Architecture.

### UNIT-2 : BASIC STRUCTURAL MODELING

(2-1 L to 2-33 L)

Basic Structural Modeling: Classes, Relationships, common Mechanisms, and diagrams. Class & Object Diagrams: Terms, concepts, modelling techniques for Class & Object Diagrams.

Collaboration Diagrams: Terms, Concepts, depicting a message, polymorphism in collaboration Diagrams, iterated messages, use of self in messages. Sequence Diagrams: Terms, concepts, depicting asynchronous messages with/without priority, call-back mechanism, broadcast messages.

Basic Behavioural Modeling: Use cases, Use case Diagrams, Activity Diagrams, State Machine, Process and thread, Event and signals, Time diagram, interaction diagram, Package diagram. Architectural Modeling: Component, Deployment, Component diagrams and Deployment diagrams.

### UNIT-3 : OBJECT ORIENTED ANALYSIS

(3-1 L to 3-21 L)

Object Oriented Analysis: Object oriented design, Object design, Combining three models, Designing algorithms, design optimization, Implementation of control, Adjustment of inheritance, Object representation, Physical packaging, Documenting design considerations.

Structured analysis and structured design (SA/SD), Jackson Structured Development (JSD). Mapping object oriented concepts using non-object oriented language, Translating classes into data structures, Passing arguments to methods, Implementing inheritance, associations encapsulation.

Object oriented programming style: reusability, extensibility, robustness, programming in the large. Procedural v/s OOP, Object oriented language features. Abstraction and Encapsulation.

### UNIT-4 : C++ BASICS & FUNCTIONS

(4-1 L to 4-21 L)

C++ Basics : Overview, Program structure, namespace, identifiers, variables, constants, enum, operators, typecasting, control structures.

C++ Functions : Simple functions, Call and Return by reference, Inline functions, Macro Vs. Inline functions, Overloading of functions, default arguments, friend functions, virtual functions.

## **UNIT-5 : OBJECTS AND CLASSES**

**(5-1 L to 5-19 L)**

Objects and Classes : Basics of object and class in C++, Private and public members, static data and function members, constructors and their types, destructors, operator overloading, type conversion. Inheritance : Concept of Inheritance, types of inheritance: single, multiple, multilevel, hierarchical, hybrid, protected members, overriding, virtual base class.

Polymorphism : Pointers in C++, Pointers and Objects, this pointer, virtual and pure virtual functions, Implementing polymorphism.

### **SHORT QUESTIONS**

**(SQ-1 L to SQ-15 L)**

### **SOLVED PAPERS (2010-11 TO 2015-16)**

**(SP-1 L to SP-25 L)**

# QUANTUM *Series*

## Related titles in Quantum Series

### For Semester - 5 (Computer Science & Engineering / Information Technology)

- Database Management System
- Design and Analysis of Algorithm
- Compiler Design
- Web Technology

### Departmental Elective-I

- Data Analytics
- Computer Graphics
- Object Oriented System Design

### Departmental Elective-II

- Machine Learning Techniques
- Application of Soft Computing
- Human Computer Interface

### Common Non Credit Course (NC)

- Constitution of India, Law & Engineering
- Indian Tradition, Culture & Society

A comprehensive book to get  
the big picture without spending  
hours over lengthy text books.

**Quantum Series** is the complete one-stop solution for engineering student looking for a simple yet effective guidance system for core engineering subject. Based on the needs of students and catering to the requirements of the syllabi, this series uniquely addresses the way in which concepts are tested through university examinations. The easy to comprehend question answer form adhered to by the books in this series is suitable and recommended for student. The students are able to effortlessly grasp the concepts and ideas discussed in their course books with the help of this series. The solved question papers of previous years act as a additional advantage for students to comprehend the paper pattern, and thus anticipate and prepare for examinations accordingly.

The coherent manner in which the books in this series present new ideas and concepts to students makes this series play an essential role in the preparation for university examinations. The detailed and comprehensive discussions, easy to understand examples, objective questions and ample exercises, all aid the students to understand everything in an all-inclusive manner.

- Topic-wise coverage in Question-Answer form.
- Clears course fundamentals.
- Includes solved University Questions.

- The perfect assistance for scoring good marks.
- Good for brush up before exams.
- Ideal for self-study.



## Quantum Publications®

(A Unit of Quantum Page Pvt. Ltd.)

Plot No. 59/2/7, Site-4, Industrial Area, Sahibabad,  
Ghaziabad, 201010, (U.P.) Phone: 0120-4160479

E-mail: [pagequantum@gmail.com](mailto:pagequantum@gmail.com) Web: [www.quantumpage.co.in](http://www.quantumpage.co.in)



Find us on: [facebook.com/quantumseriesofficial](https://www.facebook.com/quantumseriesofficial)

| Object Oriented System Design (KCS-054)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| Course Outcome ( CO)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Bloom's Knowledge Level (KL)                     |
| <b>At the end of course , the student will be able to:</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                  |
| CO 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Understand the application development and analyze the insights of object oriented programming to implement application                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | K <sub>2</sub> , K <sub>4</sub>                  |
| CO 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Understand, analyze and apply the role of overall modeling concepts (i.e. System, structural)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | K <sub>2</sub> , K <sub>3</sub>                  |
| CO 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Understand, analyze and apply oops concepts (i.e. abstraction, inheritance)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | K <sub>2</sub> , K <sub>3</sub> , K <sub>4</sub> |
| CO 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Understand the basic concepts of C++ to implement the object oriented concepts                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | K <sub>2</sub> , K <sub>3</sub>                  |
| CO 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | To understand the object oriented approach to implement real world problem.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | K <sub>2</sub> , K <sub>3</sub>                  |
| <b>DETAILED SYLLABUS</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | <b>3-0-0</b>                                     |
| Unit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Topic                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Proposed Lecture                                 |
| I                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <b>Introduction:</b> The meaning of Object Orientation, object identity, Encapsulation, information hiding, polymorphism, generosity, importance of modelling, principles of modelling, object oriented modelling, Introduction to UML, conceptual model of the UML, Architecture.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <b>08</b>                                        |
| II                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <b>Basic Structural Modeling:</b> Classes, Relationships, common Mechanisms, and diagrams. Class & Object Diagrams: Terms, concepts, modelling techniques for Class & Object Diagrams.<br><b>Collaboration Diagrams:</b> Terms, Concepts, depicting a message, polymorphism in collaboration Diagrams, iterated messages, use of self in messages. Sequence Diagrams: Terms, concepts, depicting asynchronous messages with/without priority, call-back mechanism, broadcast messages.<br><b>Basic Behavioural Modeling:</b> Use cases, Use case Diagrams, Activity Diagrams, State Machine , Process and thread, Event and signals, Time diagram, interaction diagram, Package diagram.<br><b>Architectural Modeling:</b> Component, Deployment, Component diagrams and Deployment diagrams. | <b>08</b>                                        |
| III                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Object Oriented Analysis: Object oriented design, Object design, Combining three models, Designing algorithms, design optimization, Implementation of control, Adjustment of inheritance, Object representation, Physical packaging, Documenting design considerations.<br><b>Structured analysis and structured design (SA/SD),</b> Jackson Structured Development (JSD). Mapping object oriented concepts using non-object oriented language, Translating classes into data structures, Passing arguments to methods, Implementing inheritance, associations encapsulation.<br><b>Object oriented programming style:</b> reusability, extensibility, robustness, programming in the large. Procedural v/s OOP, Object oriented language features. Abstraction and Encapsulation.            | <b>08</b>                                        |
| IV                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <b>C++ Basics :</b> Overview, Program structure, namespace, identifiers, variables, constants, enum, operators, typecasting, control structures<br><b>C++ Functions :</b> Simple functions, Call and Return by reference, Inline functions, Macro Vs. Inline functions, Overloading of functions, default arguments, friend functions, virtual functions                                                                                                                                                                                                                                                                                                                                                                                                                                      | <b>08</b>                                        |
| V                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <b>Objects and Classes :</b> Basics of object and class in C++, Private and public members, static data and function members, constructors and their types, destructors, operator overloading, type conversion. Inheritance : Concept of Inheritance, types of inheritance: single, multiple, multilevel, hierarchical, hybrid, protected members, overriding, virtual base class<br><b>Polymorphism :</b> Pointers in C++, Pointers and Objects, this pointer, virtual and pure virtual functions, Implementing polymorphism                                                                                                                                                                                                                                                                 | <b>08</b>                                        |
| <b>Text Books</b> <ol style="list-style-type: none"> <li>1. James Rumbaugh et. al, "Object Oriented Modeling and Design", Pearson Education</li> <li>2. Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", Pearson Education</li> <li>3. Object Oriented Programming With C++, E Balagurusamy, McGraw Hill.</li> <li>4. C++ Programming, Black Book, Steven Holzner, dreamtech</li> <li>5. Object Oriented Programming in Turbo C++, Robert Lafore, Galgotia</li> <li>6. Object Oriented Programming with ANSI and Turbo C++, Ashok Kamthane, Pearson</li> <li>7. The Compete Reference C++, Herbert Schlitz, McGraw Hill.</li> </ol> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                  |

# 1

## UNIT

# Introduction

## CONTENTS

- Part-1** : Introduction, The Meaning ..... 1-2L to 1-11L  
of Object Orientation,  
Object Identity, Encapsulation,  
Information Hiding,  
Polymorphism, Generosity
- Part-2** : Importance of Modelling, ..... 1-11L to 1-19L  
Principles of Modelling,  
Object-oriented Modelling
- Part-3** : Introduction of UML, ..... 1-19L to 1-22L  
Conceptual Model of UML,  
Architecture



**PART-1**

*Introduction, The Meaning of Object Orientation, Object Identity, Encapsulation, Information Hiding, Polymorphism, Generosity.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 1.1.** Explain object-oriented approach with its benefits.

**Answer**

1. In the object-oriented approach, the focus is on capturing the structure and behavior of information systems into small modules that combines both data and process.
2. The main aim of Object Oriented Design (OOD) is to improve the quality and productivity of system analysis and design by making it more usable.
3. In analysis phase, OO models are used to fill the gap between problem and solution.
4. It performs well in situation where systems are undergoing continuous design, adaption, and maintenance.
5. It identifies the objects in problem domain, classifying them in terms of data and behavior.
6. Following are the benefits of object-oriented approach :
  - a. It facilitates changes in the system at low cost.
  - b. It promotes the reuse of components.
  - c. It simplifies the problem of integrating components to configure large system.
  - d. It simplifies the design of distributed systems.

**Que 1.2.** Describe the elements of object-oriented system.

**Answer**

Following are the elements of object-oriented system :

**1. Objects :**

- a. An object is something that exists within problem domain and can be identified by data (attribute) or behavior.
- b. All tangible entities (student, patient) and some intangible entities (bank account) are modeled as object.

2. **Attributes :** They describe information about the object.
3. **Behavior :**
  - a. It specifies what the object can do.
  - b. It defines the operation performed on objects.
4. **Class :**
  - a. A class encapsulates the data and its behavior.
  - b. Objects with similar meaning and purpose grouped together as class.
5. **Methods :**
  - a. Methods determine the behavior of a class.
  - b. They are nothing more than an action that an object can perform.
6. **Message :**
  - a. A message is a function or procedure call from one object to another.
  - b. They are information sent to objects to trigger methods.

**Que 1.3.** Describe the features of object-oriented languages ?

**AKTU 2012-13, Marks 05**

**OR**

**Explain the major features of Object-Oriented Programming.**

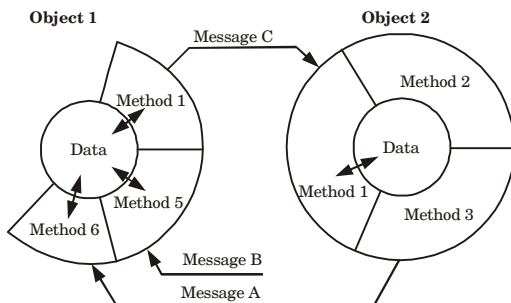
**AKTU 2013-14, Marks 05**

**Answer**

**Features of object-oriented language are :**

**1. Encapsulation :**

- i. Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.



**Fig. 1.3.1.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

- ii. The only way to reach the data is through these particular methods (see Fig. 1.3.1).
- iii. It is the mechanism that binds together code and the data it manipulates.
- iv. This concept is also used to hide the internal representation, or state, of an object from the outside.

## 2. Polymorphism :

- i. Polymorphism means having many forms.
- ii. Polymorphism is the ability of a message to be displayed in more than one form.
- iii. It plays an important role in allowing objects having different internal structure to share the same external interface.

## 3. Inheritance :

- i. Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features.
- ii. Inheritance is mainly used for code reusability.

**Que 1.4.** Describe steps of object-oriented design.

**AKTU 2010-11, Marks 05**

**Answer**

### Steps of object-oriented design :

#### 1. System analysis :

- i. In this stage a statement of the problem is formulated and a model is build. This phase show the important properties associated with the situation.
- ii. The analysis model is a concise, precise abstraction and agreement on how the desired system must be developed.
- iii. The objective is to provide a model that can be understood by any application experts in the area.

#### 2. System design :

- i. At this stage, the complete system architecture is designed.
- ii. In this stage the whole system is divided into subsystems, based on system analysis model and the proposed architecture of the system.

#### 3. Object design :

- i. At this stage, a design model is developed based on the analysis model.
- ii. The object design decides the data structures and algorithms needed to implement each of the classes in the system.

**4. Final implementation :**

- i. At this stage, the final implementation of classes and relationships developed during object design takes place.
- ii. Actual implementation should be done using software engineering practice. This helps to develop a flexible and extensible system.

**Que 1.5. Differentiate between structured approach and object oriented approach.**

**Answer**

| S. No. | Structured approach                                                                                                           | Object oriented approach                                                                                               |
|--------|-------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| 1.     | It works with Top-down approach.                                                                                              | It works with Bottom-up approach.                                                                                      |
| 2.     | Program is divided into number of sub-modules or functions.                                                                   | Program is organized by having number of classes and objects.                                                          |
| 3.     | Function call is used.                                                                                                        | Message passing is used.                                                                                               |
| 4.     | Software reuse is not possible.                                                                                               | Reusability is possible.                                                                                               |
| 5.     | Structured design programming usually left until end phases.                                                                  | Object oriented design programming done concurrently with other phases.                                                |
| 6.     | Structured Design is more suitable for off-shoring.                                                                           | It is suitable for in-house development.                                                                               |
| 7.     | It shows clear transition from design to implementation.                                                                      | Not so clear transition from design to implementation.                                                                 |
| 8.     | It is suitable for real time system, embedded system and projects where objects are not the most useful level of abstraction. | It is suitable for most business applications, game development projects, which are expected to customize or extended. |
| 9.     | DFD & E-R diagram model the data.                                                                                             | Class diagram, sequence diagram, state chart diagram, and use cases all contribute.                                    |
| 10.    | In this, projects can be managed easily due to clearly identifiable phases.                                                   | In this approach, projects can be difficult to manage due to uncertain transitions between phases.                     |

**Que 1.6. Write short notes on : Compare procedural programming with object-oriented programming with examples.**

**AKTU 2012-13, Marks 05**

**OR**

**What is the difference between Procedure Based programming language and Object-Oriented programming language ?**

**AKTU 2013-14, Marks 05**

**OR**

**Write short notes on : Procedural v/s OOP.**

**AKTU 2013-14, Marks 05**

**Answer**

| S.No. | Procedural Oriented Programming                                                           | Object-Oriented Programming                                                            |
|-------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 1.    | In procedural programming, program is divided into small parts called functions.          | In object-oriented programming, program is divided into small parts called objects.    |
| 2.    | Procedural programming follows top down approach.                                         | Object-oriented programming follows bottom up approach.                                |
| 3.    | There is no access specifier in procedural programming.                                   | Object-oriented programming has access specifiers like private, public, protected etc. |
| 4.    | Adding new data and function is not easy.                                                 | Adding new data and function is easy.                                                  |
| 5.    | Procedural programming does not have any proper way for hiding data so it is less secure. | Object-oriented programming provides data hiding so it is more secure.                 |
| 6.    | In procedural programming, overloading is not possible.                                   | Overloading is possible in object-oriented programming.                                |
| 7.    | In procedural programming, function is more important than data.                          | In object-oriented programming, data is more important than function.                  |
| 8.    | Procedural programming is based on unreal world.                                          | Object-oriented programming is based on real world.                                    |
| 9.    | <b>Examples :</b> C, FORTRAN, Pascal, Basic etc.                                          | <b>Examples :</b> C++, Java, Python, C# etc.                                           |

**Que 1.7. What do you understand by object-oriented technology ? Discuss the pros and cons of object-oriented technology with suitable example.**

**AKTU 2011-12, Marks 10**

OR

**What do you mean by object-oriented techniques ? Explain with some examples.**

**AKTU 2012-13, Marks 10****Answer**

1. Object-Oriented Technology (OOT) is an approach to program organization and development that attempts to reduce some of the issues with conventional programming techniques.
2. It is a new way of organizing and developing programs and has nothing to do with any particular programming language.
3. However, not all languages are suitable to implement the object-oriented concepts or implement partial features of object-oriented concepts.

**Pros of object-oriented technology are :**

1. **It allows parallel development :** If we are working with programming teams, then each can work independently of one another once the modular classes have been worked out.
2. **The modular classes are often reusable :** Once the modular classes have been created, they can often be used again in other applications or projects.
3. **The coding is easier to maintain :**
  - a. With OOP, because our coding base has been centralized, it is easier to create a maintainable procedure code.
  - b. That makes it easier to keep our data accessible when it becomes necessary to perform an upgrade.
  - c. This process also improves the security of the programming since high levels of validation are often required.

**Cons of object-oriented technology are :**

1. **It is inefficient :**
  - a. Object-oriented programming tends to use more CPU than alternative options.
  - b. That can make it inefficient choice when there are technical limitations involved due to the size.
2. **It is scalable :**
  - a. If OOP is out of control, then it can create a massive amount of bloated, unnecessary code.
  - b. When that occurs, the overhead rises and that makes it difficult to keep costs down.
3. **It causes duplication :**
  - a. OOP projects tend to be easier to design than implement.
  - b. That is because of the modular classes that are so flexible in their application.

- c. We may be able to get new projects up and running at a greater speed, but that comes at the cost of having projects sometimes feel like they have been cloned.

**Que 1.8. What do you understand by object identity ? Explain**

**with an example.**

**AKTU 2013-14, Marks 05**

**Answer**

1. Object identity is a property of data that is created in the context of an object data model, where an object is assigned a unique internal object identifier, or object ID.
2. The object identifier is used to define associations between objects and to support retrieval and comparison of object-oriented data based on the internal identifier rather than the attribute values of an object.
3. There are many techniques for identifying objects in programming languages.
4. OO languages have built-in mechanisms for identifying objects. There is no need to create explicit object identifier types.
5. For example : In C++ an objects actual memory address serves as a unique identifier and can be obtained by applying the '&' operator to an object or object reference.
6. Object identity can be tested by pointer comparison.

**Que 1.9. Explain encapsulation with example.**

**OR**

**Discuss the concept of encapsulation with suitable example.**

**AKTU 2012-13, Marks 05**

**OR**

**What do you mean by encapsulation ? How does the object-oriented concept of message passing help to encapsulate the implementation of an object, including its data ?**

**AKTU 2010-11, Marks 05**

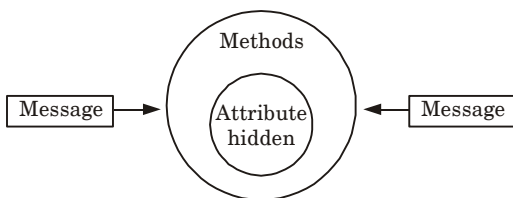
**Answer**

1. Encapsulation consists of separating the external aspects of an object, which are accessible to other objects, from the internal implementation details of the object, which are hidden from other objects.
2. Encapsulation prevents a program from becoming so interdependent that a small change has massive ripple effects.
3. The implementation of an object can be changed without affecting the applications that use it.

4. One may want to change the implementation of an object to improve performance, fix a bug, consolidate code, or for porting.
5. To understand encapsulation, let us consider the object 'Employee'.
6. The attributes of employees say 'salary' is kept hidden inside the object and may be made accessible only through the method meant for the purpose.
7. The method resides within the object.

**For example**, if `getSalary()` is a method of the object 'Employee' to get the salary of an employee, then the salary of an employee can be obtained by no other way but by this method.

8. Other objects can also send messages to the object 'Employee' and get the salary of an employee by the `getSalary()` method.
9. Other objects need not be concerned with the attributes and internal structure of the object.
10. This is shown in Fig. 1.9.1. The figure shows that attributes are hidden inside the object by a method.



**Fig. 1.9.1.** Encapsulation of an object.

**Using message passing to encapsulate the implementation of an object :** Other parts of a system only see an object's interface (services it can perform and operation signatures). Internal details including data are hidden and can only be accessed by a message that contains a valid signature.

**Que 1.10.** Write short note on information hiding.

**Answer**

1. Information hiding is the process of hiding the details of an object or function.
2. The hiding of these details results in an abstraction, which reduces the external complexity and makes the object or function easier to use.
3. In addition, information hiding effectively decouples the calling code from the internal workings of the object or function being called, which makes it possible to change the hidden portions without having to also change the calling code.



4. Encapsulation is a common technique programmers use to implement information hiding.
5. Advantage of information hiding is yielding flexibility, such as allowing a programmer to more readily modify a program.
6. This also may be done by placing source code within modules for easy access in the future, as the program develops and evolves.

**Que 1.11.** What do you mean by polymorphism ? Explain it with an example.

**AKTU 2012-13, Marks 05**

**OR**

What do you mean by polymorphism ? Is this concept only applicable to object-oriented systems ? Explain.

**AKTU 2010-11, Marks 05**

**OR**

Define polymorphism. Is this concept only applicable to object-oriented systems ? Explain.

**AKTU 2014-15, Marks 05**

**Answer**

1. Polymorphism means having many forms.
2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.
4. An operation is a function that may be applied to or by objects in a class.
5. *Open, close, hide, and redisplay* are operations on class *Window*. All objects in a class share the same operations.
6. Each operation has a target object as an implicit argument.
7. The behavior of the operation depends on the class of its target.
8. An object “knows” its class, and hence the right implementation of the operation.
9. The same operation may apply to many different classes. Such an operation is polymorphic; *i.e.*, the same operation takes on different forms in different classes.
10. **For example**, the class *File* may have an operation *print*.
11. Different methods could be implemented to print ASCII files, print binary files, and print digitized picture files.
12. All these methods logically perform the same task. However, each method may be implemented by a different piece of code.

**Applicability of polymorphism :**

1. In programming languages there are two types of polymorphism ad-hoc and universal.
2. There are two kinds of universal polymorphism: parametric and subtyping.
3. Ad-hoc polymorphism is a kind of polymorphism in which polymorphic functions can be applied to arguments of different types.
4. In universal (parametric) polymorphism, the polymorphic functions are written without mention of any specific type.
5. The ad-hoc polymorphism is applicable in both traditional and object-oriented programming environments, whereas universal polymorphism only applies to object-oriented systems.

**PART-2**

*Importance of Modelling, Principles of Modelling,  
Object-oriented Modelling.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 1.12.** What do you mean by modeling ? Discuss several purposes served by models with suitable examples.

**AKTU 2011-12, Marks 05**

**Answer**

1. A model is an abstraction of something for the purpose of understanding it before building it.
2. Since a model leave out non essential detail, it is easier to manipulate them.
3. To build hardware and software systems, the developer needs to :
  - i. Abstract different views of the system.
  - ii. Build models using precise notations.
  - iii. Make sure that the model satisfy the requirements of the system.
  - iv. Add details to transform the model into an implementation.
4. Model serves the following purpose :

**a. Testing a physical entity before building it :**

- i. Simulating a model is cheaper. Also, it provides information that is too inaccessible to be measured from physical model.
- ii. Computer models are usually cheaper than building a complete system and it enable flaws to be corrected early.

**For example :** Scale models of airplane and cars are tested in wind tunnels to improve their aerodynamic.

**b. Communication with customers :**

- i. Software designers build models to show their customers.

**For example :** Demonstration products like mock-ups that imitate some or all of the external behavior of a system.

**c. Visualization :**

- i. Storyboards of movies, television shows, and advertisements allow the writers to see how their idea flows.
- ii. Using models unnecessary segments can be modified before the final development begins.

**d. Reduction of complexity :**

- i. Modeling helps in understanding the systems that are too complex to understand directly.
- ii. Model reduces complexity by leaving out the non essential details.

**Que 1.13.** What are the different models used in object oriented languages ?

OR

Write short note on dynamic modeling and functional modeling.

**AKTU 2011-12, Marks 05**

**Answer**

There are three types of models in object oriented languages are :

**1. Object model :**

- a. The object model identifies the classes in the system and their relationship, as well as their attributes and operations.
- b. It represents the static structure of the system. The object model is represented graphically by a class diagram.

**2. Dynamic model :**

- a. The dynamic model indicates the dynamics of the objects and their changes in state.
- b. The dynamic model captures the functional behavior of the system by exploring the behavior of the objects over time and the flow of control and events among the objects.

**3. Functional model :**

- The functional model is a data flow diagram of the system and describes what the system does, not how it is done.
- A DFD is a network representation of the system to show the functional relationships of the values that are computed by a system.
- Data flow diagrams consist of processes, data flows, actors and data stores.

**Que 1.14. Write short notes on :**

- Data store**
- Actors**
- Control flow**

**AKTU 2015-16, Marks 15****Answer****a. Data store :**

- A data store is a passive object within a data flow diagram that stores data for later access.
- Unlike an actor, a data store does not generate any operations on its own but merely responds to requests to store and access data.
- A data store allows values to be accessed in a different order than they are generated.
- A data store is drawn as a pair of parallel lines containing the name of the store.
- Input arrows indicate information or operations that modify the stored data; this includes adding elements, modifying values, or deleting elements.
- Output arrows indicate information retrieved from the store. This includes retrieving the entire value or some component of it.

**b. Actors :**

- An actor is an active object that drives the data flow graph by producing or consuming values.
- Actors are attached to the inputs and outputs of a data flow graph.
- Examples of actors include the user of a program, a thermostat, and a motor under computer control.
- An actor is drawn as a rectangle to show that it is an object. Arrows between the actor and the diagram are inputs and outputs of the diagram.

**c. Control flow :**

- A data flow diagram shows all possible computation paths for values; it does not show which paths are executed and in what order.

2. This is done by including control flows in the data flow diagram.
3. A control flow is a Boolean value that affects whether a process is evaluated.
4. The control flow is not an input value to the process itself.
5. A control flow is shown by a dotted line from a process producing a Boolean value to the process being controlled.

**Que 1.15. What are the principles of modeling ? What is the importance of modeling ?**

**AKTU 2013-14, Marks 05**

**OR**

**What are the basic principles of modeling ? Explain in detail.**

**AKTU 2014-15, Marks 05**

**Answer**

**Principles of modeling are :**

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 correct model as per the requirement of problem statement.
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 non-trivial system is best approached through a small set of nearly independent models :** This means we 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.

**Importance of modeling :**

Modeling help the development team better to visualize the plan of their system and allow them to develop more rapidly by helping them build the right thing.

**Que 1.16. Define object-oriented modeling (OOM). Describe various steps involved in OOM process. Explain.**

**AKTU 2010-11, Marks 05**

**Answer****Object-oriented modeling :**

1. Object-oriented modeling (OOM) is the construction of objects using a collection of objects that contain stored values of the instance variables found within an object.
2. Object-oriented modeling is an approach to modeling an application that is used at the beginning of the software life cycle when using an object-oriented approach to software development.

**Steps involved in OOM process :****1. System analysis :**

- i. In this stage a statement of the problem is formulated and a model is build. This phase show the important properties associated with the situation.
- ii. The analysis model is a concise, precise abstraction and agreement on how the desired system must be developed.
- iii. The objective is to provide a model that can be understood by any application experts in the area.

**2. System design :**

- i. At this stage, the complete system architecture is designed.
- ii. In this stage the whole system is divided into subsystems, based on system analysis model and the proposed architecture of the system.

**3. Object design :**

- i. At this stage, a design model is developed based on the analysis model.
- ii. The object design decides the data structures and algorithms needed to implement each of the classes in the system.

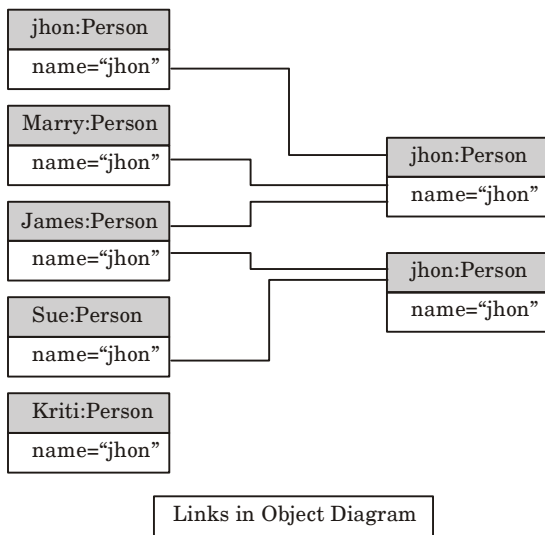
**4. Final implementation :**

- i. At this stage, the final implementation of classes and relationships developed during object design takes place.
- ii. Actual implementation should be done using software engineering practice. This helps to develop a flexible and extensible system.

**Que 1.17. Define link and association. Discuss the role of link and association in object modeling with suitable example.**

**Answer**

1. Link and association in object modeling represent the relation between objects and classes.
2. **Link :** Link defines the relationship between two or more objects and a link is considered as an instance of an association.
3. **Association :** It is a group of links that relates objects from the same classes.
4. **For example :**
  - i. Let us take the two classes Person and Company. Now there is an association relation between these two classes.
  - ii. A person may own stock in zero or more companies.
  - iii. Also it can be related in reverse that a company may have several persons owing its stock.
  - iv. The object diagram below shows the links between the objects of person and company class.

**Fig. 1.17.1.**

- v. The class diagram below shows the association between the person and the company class. Both link and association are represented with a line in UML notation.



Association in class diagram

Fig. 1.17.2.

**Que 1.18.** What do you mean by object modeling technique ?

**Explain. Discuss the various stages of the object modeling techniques with some example.**

**AKTU 2015-16, Marks 10**

**Answer**

**Object modeling technique :**

1. The Object Modeling Technique (OMT) is the methodology that combines the three views of modeling system, *i.e.*, the object model, the dynamic model and the functioned model.
2. These three models separate a system into orthogonal views that can be represented and manipulated with a uniform notation.

**Stages of object modeling technique :**

**1. Analysis :**

- a. Starting from a statement of the problem, the analyst builds a model of the real-world situation showing its important properties.
- b. The analysis model is a concise, precise abstraction of what the desired system must do.
- c. The objects in the model should be application-domain concepts.
- d. A good model can be understood by application experts who are not programmers.
- e. For example, a *Window* class in a workstation windowing system would be described in terms of the attributes and operations visible to a user.

**2. System design :**

- a. The system designer makes high-level decisions about the overall architecture.
- b. During system design, the target system is organized into subsystems based on both the analysis structure and the proposed architecture.



- c. The system designer must decide what performance characteristics to optimize, developing strategy for solving the problem, and making tentative resource allocations.
- d. For example, the system designer might decide that changes to the workstation screen must be fast and smooth and choose an appropriate communications protocol and memory buffering strategy.

### 3. Object design :

- a. The object designer builds a design model based on the analysis model but containing implementation details.
- b. The designer adds details to the design model in accordance with the strategy established during system design.
- c. The focus of object design is the data structures and algorithms needed to implement each class.
- d. The object classes are augmented with computer-domain data structures and algorithms chosen to optimize important performance measures.
- e. For example, the *Window* class operations are now specified in terms of the underlying hardware and operating system.

### 4. Implementation :

- a. The object classes and relationships developed during object design are finally implemented.
- b. During implementation, it is important to follow good software engineering practice so that traceability to the design is straightforward and so that the implemented system remains flexible and extensible.
- c. For example, the *Window* class would be coded in a programming language, using calls to the underlying graphics system on the workstation.

**Que 1.19.** Wire is used in the following applications. For each of the following applications, prepare a list of wire characteristics that are relevant and also explain why each characteristic is important for the application : (1) Designing the filament for a light bulb; (2) Designing the electrical system for an airplane.

**Answer****Designing the filament for a light bulb :**

1. Because the filament of a light bulb operates at a high temperature, resistance to high temperatures is important.
2. Tungsten is generally used because of its high melting point, even though tungsten filaments are brittle.

**Designing the electrical system for an airplane :**

1. Weight is important for wire that is to be used in the electrical system of an airplane, because it affects the total weight of the plane.
2. Toughness of the insulation is important to resist chafing due to vibration.
3. Resistance of the insulation to fire is also important to avoid starting or feeding electrical fires in flight.

**PART-3**

*Introduction of UML, Conceptual Model of UML, Architecture.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 1.20.** What do you mean by UML ? Discuss the conceptual model of UML with the help of an appropriate example.

**AKTU 2011-12, Marks 05**

**OR**

**Give the conceptual model of UML. Use some example to illustrate the model in detail using diagram.**

**AKTU 2012-13, Marks 05**

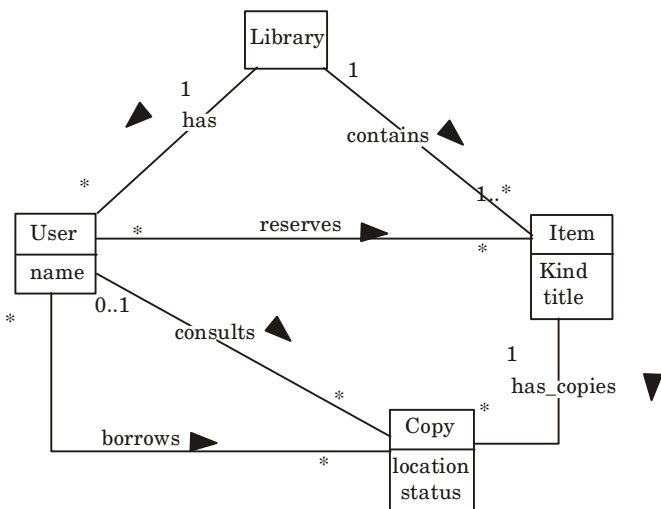
**Answer****UML :**

1. UML stands for Unified Modeling Language.
2. UML is a pictorial language used to make software blueprints.
3. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.

4. It is also used to model non-software systems as well.
5. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams.
6. UML has a direct relation with object-oriented analysis and design.

### Conceptual model of UML :

1. A conceptual model is defined as a model which is made of concepts and their relationships.
2. A conceptual model is the first step before drawing a UML diagram.
3. It helps to understand the entities in the real world and how they interact with each other.
4. The conceptual model of UML has three major elements :
  - a. UML building blocks.
  - b. Rules to connect the building blocks.
  - c. Common mechanisms of UML.
5. A domain model, often referred to as a conceptual model, might be represented by a particular kind of UML class diagram.
6. This model explains the structure of the application domain rather than the application structure itself.
7. It focuses on the domain concepts, rather than on the software entities.
8. Fig. 1.20.1 shows the conceptual model for a library system.



**Fig. 1.20.1.** Conceptual model for a library system.

**Que 1.21. Describe the pros and cons of unified modeling language**

**(UML).**

**AKTU 2012-13, Marks 05**

**Answer**

**Pros of UML :**

1. It has wide industry acceptance in comparison to previous modeling language.
2. It supports OOAD methodology.
3. It bridges the communication gap between different entities of system development (*i.e.*, System Analyst, Developer, Client etc).
4. Constructed models are easy to understand, even for non-programmers.
5. It is a unified and standardize modeling language.

**Cons of UML :**

1. UML is often criticized as being large and complex.
2. It takes a lot of time to keep the diagram reasonable and synchronized with the actual code.
3. You cannot represent every condition in a sequence diagram.
4. UML software costs money.
5. Complex to learn and takes time to master properly.

**Que 1.22. Why UML required ? What are the basic architecture of**

**UML ?**

**AKTU 2014-15, Marks 05**

**Answer**

The UML is required to help system and software developers accomplish the following tasks :

- i. Specification
- ii. Visualization
- iii. Architecture design
- iv. Construction
- v. Simulation and testing
- vi. Documentation

**Basic architecture of UML :**

1. The UML is defined in a circular manner, in which a subset of the language notation and semantics is used to specify the language itself.

2. The UML is defined within a conceptual framework for modeling that consists of four distinct layers or levels of abstraction.
3. This framework is based on the most fundamental UML notation that concepts are depicted as symbols, and relationships among concepts are depicted as paths (lines) connecting symbols. Both of these types of elements may be named.
4. The concepts introduced by the UML are organized around architectural views to define the various diagrams.
5. The UML diagrams are used to understand for conceptualize a problem, solve the problem, and implement or realize the solution.



# 2

## UNIT

# Basic Structural Modeling

## CONTENTS

- Part-1** : Basic Structural Modeling, ..... 2-2L to 2-8L  
Class, Relationships, Common  
Mechanism and Diagram
- Part-2** : Class and Object Diagram, ..... 2-9L to 2-14L  
Terms, Concepts, Modeling  
Techniques for Class and Object Diagram
- Part-3** : Collaboration Diagram, Terms, ..... 2-14L to 2-17L  
Concepts, Depicting A Message,  
Polymorphism in Collaboration  
Diagram, Iterated Message,  
use of Self in Message
- Part-4** : Sequence Diagram, Terms, ..... 2-17L to 2-22L  
Concept, Depicting Asynchronous  
Message With/ Without Priority,  
Callback Mechanism, Broadcast Message
- Part-5** : Basic Behavioural Modeling, ..... 2-23L to 2-31L  
Use Cases, use Case Diagrams,  
Activity Diagram, State Machine,  
Process and Thread, Event and  
Signals, Time Diagram, Interaction  
Diagram, Package Diagram
- Part-6** : Architectural Modeling, ..... 2-31L to 2-33L  
Component, Deployment,  
Component Diagrams and  
Deployment Diagrams

**PART- 1**

*Basic Structural Modeling, Class, Relationships, Common Mechanism and Diagram.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 2.1.** Explain different types of modeling in object oriented system design.

**OR**

What is UML ? Mention the different kinds of modeling diagrams used.

**AKTU 2013-14, Marks 05**

**Answer**

There are three important types of modeling :

**1. Structural Modeling :**

- Structural modeling captures the static features of a system.
- Structural model represents the framework for the system and this framework is the place where all other components exist.
- Hence, the class diagram, component diagram and deployment diagrams are part of structural modeling.
- They all represent the elements and the mechanism to assemble them.
- The structural model never describes the dynamic behavior of the system.

**2. Behavioral Modeling :**

- Behavioral model describes the interaction in the system.
- It represents the interaction among the structural diagrams.
- Behavioral modeling shows the dynamic nature of the system.
- It consists of Activity diagrams, Interaction diagrams, Use case diagrams.

**3. Architectural Modeling :**

- Architectural model represents the overall framework of the system.
- It contains both structural and behavioral elements of the system.

- c. Architectural model can be defined as the blueprint of the entire system.
- d. Package diagram comes under architectural modeling.

**UML :** Refer Q. 1.20, Page 1-19L, Unit-1.

**Que 2.2.** What do you understand by architectural modeling ?

**Explain its various concepts and diagrams with suitable example.**

**AKTU 2010-11, Marks 05**

**OR**

**Write short notes on architectural modeling with suitable example and diagrams.**

**AKTU 2012-13, Marks 05**

**Answer**

**Architectural modeling :**

- 1. Architectural modeling represents the overall framework of the system.
- 2. It contains both structural and behavioral elements of the system.
- 3. Architectural modeling can be defined as the blueprint of the entire system.

**Diagrams used in architectural modeling :**

- 1. The two types of diagrams that give descriptions of the physical information about a system are deployment diagrams and component diagrams.
- 2. Deployment diagrams show the physical relationship between hardware and software in a system.
- 3. Component diagrams show the software components of a system and their relationships.
- 4. These relationships are called dependencies.

**A. Component diagrams :**

- 1. The component diagrams are mainly used to model the static implementation view of a system.
- 2. They represent a high-level packaged view of the code.
- 3. They can be used to model executables, databases and adaptable systems.
- 4. Component diagrams mainly contain the following :

**i. Components :**

- a. A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.
- b. A component is a physical manifestation of an object that has a well-defined interface and a set of implementations for the interface.
- c. A complex system can be built using software components. It enhances re-use in the system and facilitates system evolution.



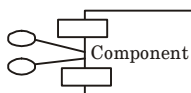


Fig. 2.2.1. Component.

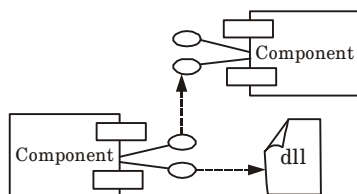


Fig. 2.2.2. Component diagram.

**ii. Interfaces :** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle. An interface possesses the following properties :

- Every interface is identified by a unique name, with an operational pathname.
- Interfaces do not specify any structure or implementation details of the operations. When an interface is represented as a rectangle, it has the same parts as a class. When it is represented as a circle, the display of these parts is suppressed.

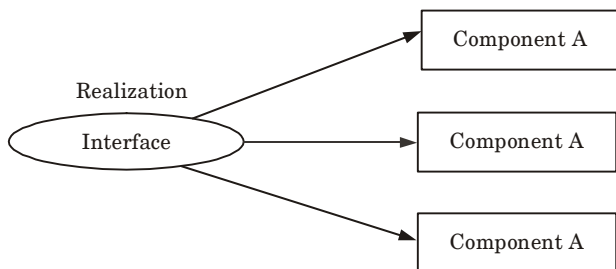
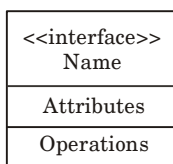


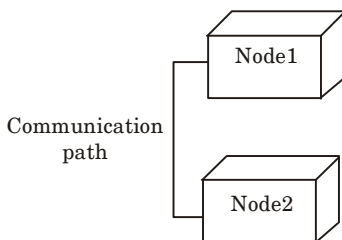
Fig. 2.2.3. Realization of an interface.



Interface name

Fig. 2.2.4. Representation of interfaces.

**B. Deployment diagrams :** They display the configuration of run-time processing elements and the software components, processes and objects. The deployment diagram contains nodes and connections. A node is a piece of hardware in the system. A connection depicts the communication path used by the hardware (Fig. 2.2.5).



**Fig. 2.2.5.** Deployment diagram.

**Que 2.3.** What do you understand by classes in object oriented system design ?

**Answer**

1. The phrase class is use to refer to a group of similar things.
2. A class describes a group of objects with similar properties, common behavior, common relationships to other objects, and common semantics.
3. Objects in a class have the same attributes and behavior pattern.
4. Most objects derive their individuality from differences in their attribute values and relationships to other objects.
5. The objects in a class share a common semantic purpose, above and beyond the requirement of common attributes and behavior.
6. Each object “knows” its class.
7. Object-oriented programming languages can determine an object’s class at run time.
8. An object’s class is an implicit property of the object.

**Que 2.4.** Explain relationship with its different types.

**Answer**

**Relationships :** A model is not complete unless the relationships between elements are described properly. The Relationship gives a proper meaning to a UML model. Following are the different types of relationships used in UML.

**1. Dependency Notation :**

- a. Dependency describes the dependent elements and the direction of dependency.

- b. Dependency is represented by a dotted arrow as shown in the following figure.
- c. The arrow head represents the independent element and the other end represents the dependent element.
- d. Dependency is used to represent the dependency between two elements of a system

## 2. Association Notation :

- a. Association describes how the elements in a UML diagram are associated.
- b. In simple words, it describes how many elements are taking part in an interaction.
- c. Association is represented by a dotted line with (without) arrows on both sides.
- d. The two ends represent two associated elements as shown in the following figure.
- e. The multiplicity is also mentioned at the ends (1, \*, etc.) to show how many objects are associated.

## 3. Generalization Notation :

- a. Generalization describes the inheritance relationship of the object-oriented world.
- b. It is a parent and child relationship.
- c. Generalization is represented by an arrow with a hollow arrow head as shown in the following figure.
- d. One end represents the parent element and the other end represents the child element.

## 4. Extensibility Notation :

- a. All the languages (programming or modeling) have some mechanism to extend its capabilities such as syntax, semantics, etc.
- b. UML also has the following mechanisms to provide extensibility features :
  - i. Stereotypes (Represents new elements)
  - ii. Tagged values (Represents new attributes)
  - iii. Constraints (Represents the boundaries)
- c. Extensibility notations are used to enhance the power of the language.

**Que 2.5.**

**Describe generalization and specialization.**

**AKTU 2010-11, Marks 05**

**OR**

**What do you mean by generalization ? Explain. How is it related with inheritance ?**

**AKTU 2011-12, Marks 05**

**OR**

**Define aggregation and generalization. Explain.**

**AKTU 2012-13, Marks 05**

**Answer**

**Generalization and specialization :** Generalization and specialization represent a hierarchy of relationships between classes, where subclasses inherit from super-classes.

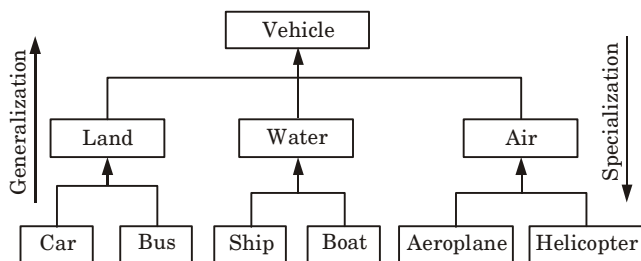
**1. Generalization :**

- In the generalization process, the common characteristics of classes are combined to form a class in a higher level of hierarchy, *i.e.*, subclasses are combined to form a generalized super-class.
- It represents an “is - a - kind - of” relationship. For example, “car is a kind of land vehicle”, or “ship is a kind of water vehicle”.

**2. Specialization :**

- Specialization is the reverse process of generalization. Here, the distinguishing features of groups of objects are used form specialized classes from existing classes.
- It can be said that the subclasses are the specialized versions of the super-class.

The following figure shows an example of generalization and specialization.



**Fig. 2.5.1.**

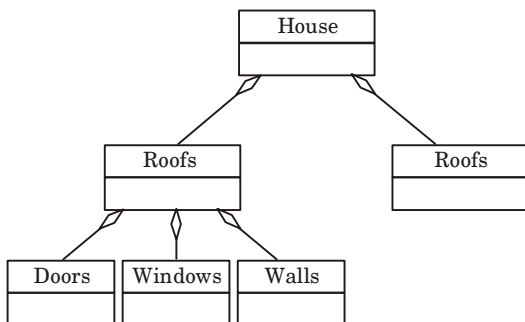
**Relation to inheritance :**

- Developers use the term generalization or inheritance to refer to the same concept of reusing shared attributes and operations that you show in a superclass and reuse in subclasses.
- Generalization refers to the concept of generalizing from specifics (the subclasses) to the generic (the superclass).

3. Inheritance refers to the effect of generalization on the subclasses.

### Aggregation :

1. Aggregation is a stronger form of association. It represents the **has-a** or **part-of** relationship.
2. An aggregation association depicts a complex object that is composed of other objects.
3. For example, we may characterize a house in terms of its roof, floors, foundation, walls, rooms, windows, and so on. A room may, in turn be, composed of walls, ceiling, floor, windows, and doors, as represented in Fig. 2.5.2.



**Fig. 2.5.2.** A house and some of its component.

4. Hence object aggregation helps us describe models of the real world that are composed of other models, as well as those that are composed of still other models.

**Que 2.6.** Categorize the following relationship into generalization, aggregation, or association :

1. A country has a capital city
2. Files contain records.

**AKTU 2012-13, Marks 05**

### Answer

1. **A country has a capital city :** It is an association relationship. A capital city and a country are distinct things so generalization certainly does not apply. You could argue that a capital city is a part of a country and thus they are related by aggregation.
2. **Files contain records :** It is an aggregation relationship. The word “contain” is a clue that the relationship may be aggregation. A record is a part of a file. Some attributes and operations on files propagate to their constituent records.

**PART-2**

*Class and Object Diagram, Terms, Concepts, Modeling Techniques for Class and Object Diagram.*

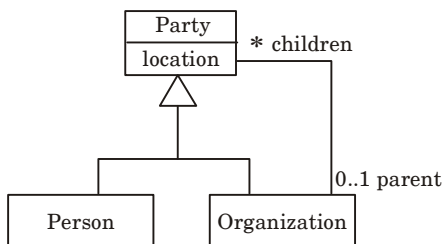
**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 2.7.** Explain class and object diagrams with examples.

**AKTU 2013-14, Marks 05**

**Answer****Class diagram :**

1. Class diagram is a static diagram.
2. It represents the static view of an application.
3. Class diagram is used for visualizing, describing, and documenting different aspects of a system and also for constructing executable code of the software application.
4. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
5. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

**For example :**

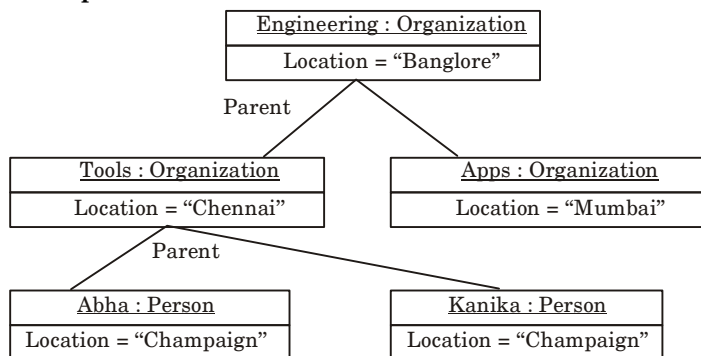
**Fig. 2.7.1.** Class diagram of party composition structure.

**Object diagram :**

1. Object diagrams represent an instance of a class diagram.

- Object diagrams represent the static view of a system but this static view is a snapshot of the system at a particular moment.
- Object diagrams are used to render a set of objects and their relationships as an instance.

**For example :**



**Fig. 2.7.2.** Object diagram showing example instances of party.

**Que 2.8.**

**Differentiate between a class and object with some example. Also prepare a list of objects that you would expect each of the following systems to handle : (1) a program for laying out a newspaper, (2) a catalog store order entry system.**

**AKTU 2011-12, Marks 05**

**Answer**

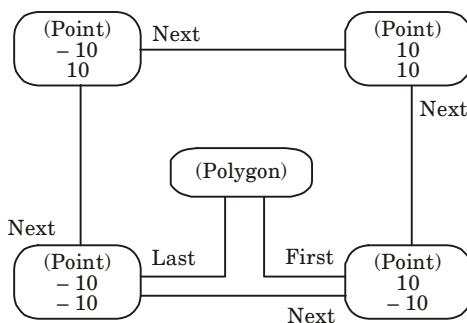
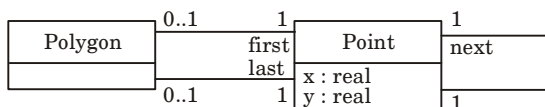
| S. No. | Class                                                               | Object                                                                                      |
|--------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| 1.     | Class is a blueprint or template from which object are created.     | Object is an instance of class.                                                             |
| 2.     | Class is a group of similar objects.                                | Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. |
| 3.     | Class is a logical entity.                                          | Object is physical entity.                                                                  |
| 4.     | Class is declared using class keyword, for example, Class student{} | Object is created through new keyword mainly, for example, Student s1 = new student ();     |
| 5.     | Class is declared once.                                             | Object is created many times as per requirement.                                            |
| 6.     | Class does not allocated memory when it is created.                 | Object allocates memory when it is created.                                                 |

**For example :**

1. A class is a way of grouping objects that share a number of characteristics: attributes (like name, color, height, weight, etc.) and behavior (such as ability to perform jumps, to run, to swim, etc.).
  2. All objects in the class horse will have an attribute named height, for example. That means that all object in that class have a height-the value of the attribute height will be different for each instance of the class (*i.e.*, for each particular horse).
- i. A program for laying out a newspaper :** Classes that you would expect in a program for newspaper layout include Page, Column, Line, Headline, and Paragraph.
- ii. A catalog store order entry system :** For a catalog store order entry system, classes include Customer, Order, Store, and Item.

**Que 2.9.**

**Give the general layout of a class diagram. Also prepare a class diagram for the instance diagram shown in the Fig. 2.9.1. Explain your multiplicity decisions. How does your diagram express the fact that points are in sequence ?**

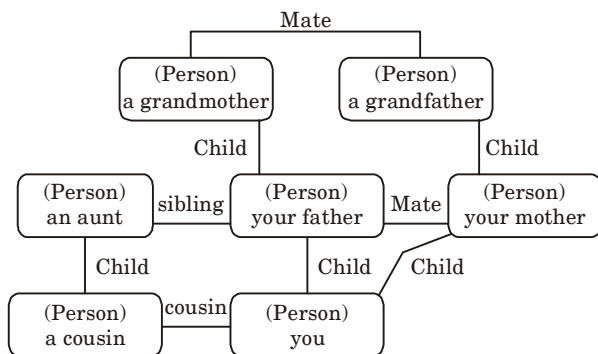
**Fig. 2.9.1.****AKTU 2011-12, Marks 05****Answer****Fig. 2.9.2. General class diagram for polygon and points.**



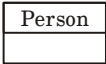
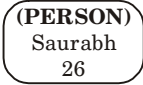
1. Fig. 2.9.2 shows the class diagram.
2. Fig. 2.9.2 permits a degenerate polygon which consists of exactly one point. (The same point is first and last. The point is next to itself).
3. The class diagram also permits a line to be stored as a polygon.
4. Fig. 2.9.2 does not enforce the constraint that the first and last points must be adjacent.
5. In Fig. 2.9.2 the sense of ordering is problematic. A polygon that is traversed in left-to-right order is stored differently than one that is traversed in right-to-left order even though both visually appear the same.
6. There is no constraint that a polygon be closed and that a polygon not cross itself.
7. In general it is difficult to fully capture constraints with class models and we must choose between model complexity and model completeness.

**Que 2.10.**

**What is the difference between a class diagram and an instance diagram ? Discuss the significance of each. Also prepare a class diagram for the following instance diagram as given in Fig. 2.10.1.**

**Fig. 2.10.1.**

**Answer****A. Difference :**

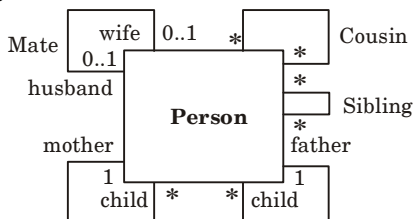
| S. No. | Class diagram                                                                                                                                                                                                                | Instance diagram                                                                                                                                                                                                                                                                                     |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.     | A class diagram is a schema, pattern, or template for describing many possible instances of data.                                                                                                                            | An instance diagram describes how does a particular set of object relates to each other.                                                                                                                                                                                                             |
| 2.     | A class diagram describes the general case in modeling a system.                                                                                                                                                             | An instance diagram describes object instances.                                                                                                                                                                                                                                                      |
| 3.     | A class diagram describes object classes.                                                                                                                                                                                    | An instance diagram is useful for documenting test cases, especially scenarios, and is used to show examples to help to clarify a complex class diagram.                                                                                                                                             |
| 4.     | <p>The OMT symbol for a class is a rectangular box with class name in boldface. A line is drawn between the class name and attributes.</p>  | <p>Figure below shows the OMT representation of instance diagram. The class name in parenthesis is at the top of the object box in boldface and object names are listed in normal font with their attributes.</p>  |

**B. Significance of class diagram :**

1. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
2. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.
3. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

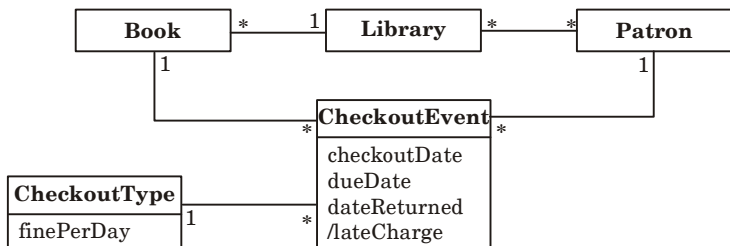
**C. Significance of instance diagram :**

1. An object diagram represents an instance at a particular moment, which is concrete in nature.
2. It means the object diagram is closer to the actual system behaviour.
3. The purpose is to capture the static view of a system at a particular moment.

**D. Class Diagram :****Fig. 2.10.2.** Class diagram for family trees.

**Que 2.11.** Prepare a portion of an object diagram for a library book checkout system that shows the date a book is due and the late charges for an overdue book as derived objects.

AKTU 2011-12, Marks 05

**Answer**

$$\{lateCharge = (dataReturned - dueDate) * CheckoutType.finePerDay\}$$
**Fig. 2.11.1.** Class diagram for library book checkout system.**PART-3**

*Collaboration Diagram, Terms, Concepts, Depicting A Message, Polymorphism in Collaboration Diagram, Iterated Message, use of Self in Message.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 2.12.** What do you mean by a collaboration diagram ? Explain various terms and symbols used in a collaboration diagram. How

**polymorphism is described using a collaboration diagram ? Explain using an example.**

**AKTU 2010-11, Marks 05**

**OR**

**What is a collaboration diagram ? How polymorphism is represented in a collaboration diagram? Explain with an example.**

**AKTU 2011-12, Marks 05**

**Answer**

**Collaboration diagram :**

1. A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).
2. These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.
3. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction.

**Various terms used in collaboration diagram :**

1. **Objects :** Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name; class name.

Object  
name

2. **Actors :** Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.



3. **Links :** Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.

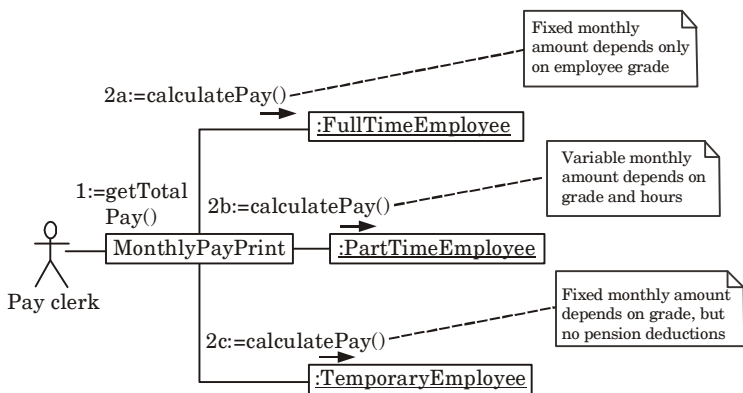
Link

4. **Messages :** Messages between objects are shown as a labeled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

**Polymorphism using collaboration diagram :**

1. Fig. 2.12.1 uses a collaboration diagram to illustrate polymorphism in a business scenario.
2. The diagram assumes that there are different ways of calculating an employee's pay.

- Full-time employees are paid a salary that depends only on his or her grade; part-time staff are paid a salary that depends in a similar way on grade, but must also take into account the number of hours worked; temporary staff differ in that no deductions are made for the company pension scheme, but the salary calculation is otherwise the same as for a full-time employee.
- An object-oriented system to calculate pay for these employees might include a separate class for each type of employee, each able to perform the appropriate pay calculation.
- However, following the principle of polymorphism, the message signature for all calculate pay operations is the same.



**Fig. 2.12.1.** Polymorphism allows a message to achieve the same result even when the mechanism for achieving it differs between different objects.

- Suppose one of the outputs from this system is a print-out showing the total pay for the current month: to assemble the total, a message is sent to each employee object, asking it to calculate its pay.
- Since the message signature is the same in each case, the requesting object (here called MonthlyPayPrint) need not know that the class of each receiving object, still less how each calculation is carried out.

### Que 2.13.

**Explain Polymorphism, Iterated Messages and use of**

**self in message in collaboration diagram. AKTU 2013-14, Marks 05**

### Answer

#### Polymorphism :

- Polymorphism means having many forms.

2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.

**Iterated messages :**

1. Collaboration diagrams use syntax similar to sequence diagrams to indicate that either a message iterates (is run multiple times) or is run conditionally.
2. We can indicate that a particular message iterates by prefixing a message sequence number with an iteration expression.
3. We can simply use an asterisk (\*) to indicate that a message runs more than once, or we can get more specific and show the number of times a message is repeated.
4. To indicate that a message is run conditionally, we can prefix the message sequence number with a conditional clause such as [x = true].
5. This indicates that the message is sent only if the condition is met.
6. The UML leaves the syntax of conditional clauses wide open, so we can create expressions that make sense in the context of our application.

**Use of self in message :**

1. Self represents the ability of an object to send a message to itself.
2. Messages in collaboration diagrams are shown as arrows pointing from the client object to the supplier object.
3. Messages represent a client invoking an operation on a supplier object.
4. Message icons have one or more messages associated with them.
5. Messages are composed of message text prefixed by a sequence number.
6. This sequence number indicates the time-ordering of the message.

**PART-4**

*Sequence Diagram, Terms, Concept, Depicting Asynchronous Message With / Without Priority, Callback Mechanism, Broadcast Message.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 2.14.** What do you mean by sequence diagram? Explain various terms and symbols used in a sequence diagram. Describe

the following using sequence diagram : (i) asynchronous messages with/without priority. (ii) broadcast messages.

AKTU 2011-12, Marks 10

OR

Explain sequence diagrams with example.

AKTU 2013-14, Marks 05

Answer

**Sequence diagram :**

1. Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.
2. They're also called event diagrams.
3. A sequence diagram is a good way to visualize and validate various runtime scenarios.
4. In UML it is shown as a table that shows objects arranged along the X axis and messages along the Y axis.
5. It has a global life line and the focus of control.

**Various terms and symbols used in sequence diagram :**

1. **Class roles or Participants :** Class roles describe the way an object will behave in context.

Object

Fig. 2.14.1.

2. **Activation or Execution Occurrence :** Activation occurrence represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



Fig. 2.14.2.

3. **Messages :** Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

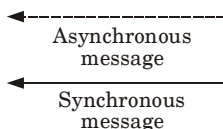


Fig. 2.14.3.

4. **Lifelines :** Lifelines are vertical dashed lines that indicate the object's presence over time.



Fig. 2.14.4

5. **Destroying objects :** Objects can be terminated early using an arrow labeled “<< destroy >>” that points to an X. This object is removed from memory. When that object's lifeline ends, we can place an X at the end of its lifeline to denote a destruction occurrence.
6. **Loops :** A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets.

#### A. Asynchronous messages without priority :

1. An asynchronous message can be implemented on the sequence diagram by using a half arrowhead on the asynchronous message arrow.

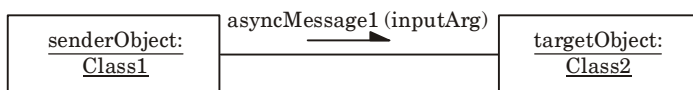


Fig. 2.14.5. A collaboration diagram showing a basic asynchronous message.

2. When an asynchronous message is send, at least to loci of execution are active in the system, because the target begins to execute while the sender remains in execution.
3. Fig. 2.14.5 shows a specific example from a real-time system that authorizes personnel to pass through electronically controlled doors.

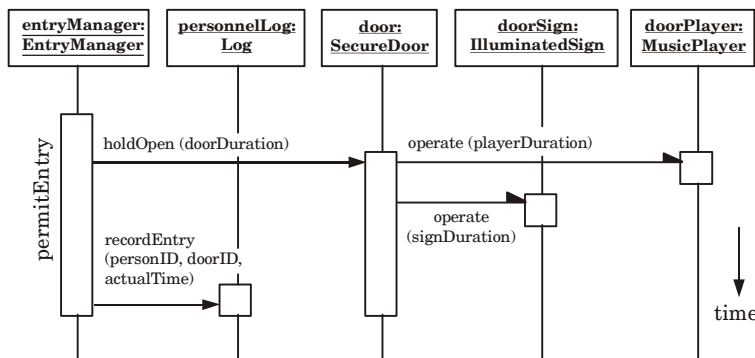


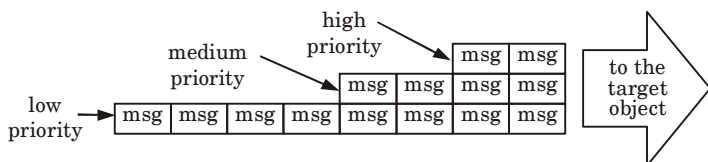
Fig. 2.14.6. A sequence diagram for concurrently executing objects.



4. The employee inserts an ID card into a reader, and if the employee is authorized to enter, the system plays jingly music, displays a greeting and slides the door open.
5. The operation that manages this piece of the application is **permitEntry**.
6. After determining that the employee is allowed through the door **permitEntry** sends the synchronous message **holdOpen** to the object **door**.
7. The operation **holdOpen** then handles the sound, lights, and action associated with opening the door.
8. For the action of opening the door, **holdOpen** sends messages to a hardware driver.
9. For the sound-and-light show, **holdOpen** sends two asynchronous messages, both named **operate** : one to **doorPlayer** and one to **doorSign**.
10. The two operations named **operate** execute concurrently: **door**.

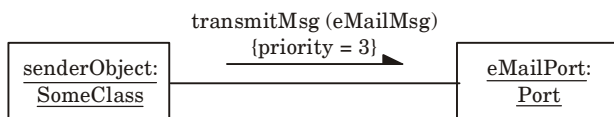
#### B. Asynchronous message with priority :

1. In concurrency the target object receives message from lots of concurrently executing sender objects.
2. Since these messages may arrive faster than the target can process them, they are ushered into the message queue.
3. The object then removes a message from the front of the queue; process the message, and takes the next message from the queue.
4. These messages in a queue are ordered by priority.
5. Fig. 2.14.7 shows a set of parallel queues at the target, each queue with its own priority level.



**Fig. 2.14.7.** Three parallel queues, each with its own priority.

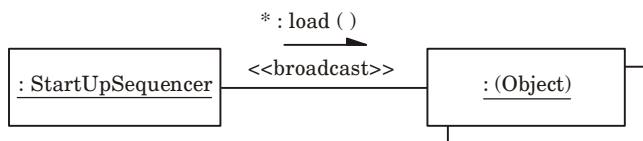
6. Fig. 2.14.8 shows an asynchronous message with its priority level.
7. The property {priority = 3} indicates that the message has a priority of 3.



**Fig. 2.14.8.** An asynchronous message (with priority 3) going to an object with a multiple-priority message queue.

**Broadcast message :**

1. A broadcast message treats every object in the system as a potential target.
2. A copy of the message goes into the queue of every object in the system.
3. An object may broadcast a message in response to some external event that it detects.
4. For example, an object might detect a security compromise and broadcast to all objects the need for a priority system shutdown.
5. Fig. 2.14.9 shows the UML for a broadcast message. Here, a start-up sequencer is getting every object in the system that exists at start-up time to load itself.

**Fig. 2.14.9.** A broadcast message.

**Que 2.15.** Discuss the significance of sequence diagrams. How the following is implemented using sequence diagrams :

- i. Broadcast messages
- ii. Callback mechanism
- iii. Asynchronous messages with/without priority.

AKTU 2012-13, Marks 10

OR

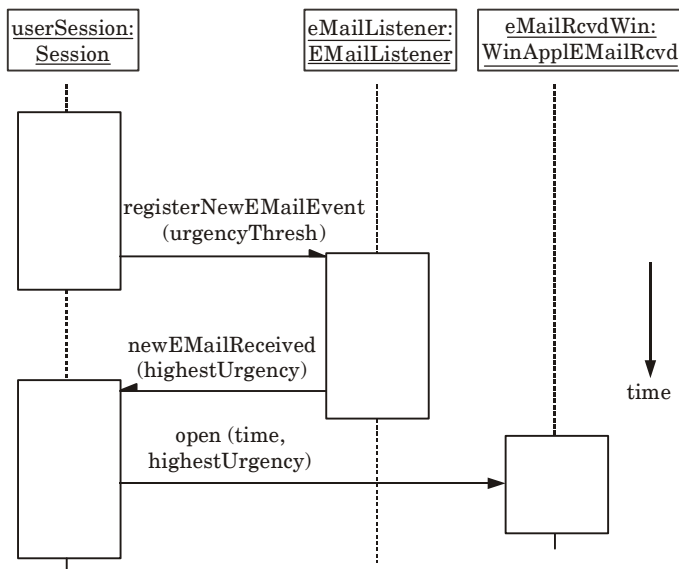
What do you understand by callback mechanisms ?

AKTU 2013-14, Marks 05

**Answer****Significance of sequence diagram :**

- i. Sequence diagrams are one of the important dynamic modeling techniques in the UML.
  - ii. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.
- i. **Broadcast messages :** Refer Q. 2.14, Page 2-17L, Unit-2.
  - ii. **Callback mechanism :**
    1. Here the subscriber object registers an interest in some event via an asynchronous message to the target object.

2. The target object continuous with other activities while it monitors for the occurrence of an event of the registered type.



**Fig. 2.15.1.** Callback mechanism used to detect e-mail.

3. When an event of that type occurs, the target object sends a message back to the subscriber object to notify of the occurrence. This is known as callback mechanism.
4. The sequence diagram in Fig. 2.15.2 shows an example of the callback mechanism, where the event of interest is the arrival of e-mail with an urgency above a certain threshold.
5. The **userSession** object registers its interest in new e-mail with the **eMailListener** object by sending the message **registerNewEMailEvent (urgencyThresh)**.
6. When e-mail that's urgent enough arrives, **eMailListener** calls back **userSession** with **newEMailReceived (highestUrgency)**, whose argument indicates the highest urgency among the messages that actually arrived.

iii. **Asynchronous messages with/without priority** : Refer Q. 2.14, Page 2-17L, Unit-2.

**PART-5**

*Basic Behavioural Modeling, Use Cases, use Case Diagrams, Activity Diagram, State Machine, Process and Thread, Event and Signals, Time Diagram, Interaction Diagram, Package Diagram.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 2.16.** Discuss in brief basic behavioural modeling.

**AKTU 2011-12, Marks 05**

**OR**

**What do you understand by basic behavioural modeling ?**

**AKTU 2013-14, Marks 05**

**Answer**

- i. Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.
- ii. During analysis, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.
- iii. In the design and implementation phases, the detailed design of the operations contained in the object is fully specified.
- iv. There are two types of behavioral models. First, there are behavioral models that are used to represent the underlying details of a business process portrayed by a use case model. In UML, interaction diagrams (sequence and communication) are used for this type of behavioral model.
- v. Second, there is a behavioral model that is used to represent the changes that occur in the underlying data. UML uses behavioral state machines for this.
- vi. During the analysis phase, analysts use behavioral model to capture a basic understanding of the dynamic aspects of the underlying business process.
- vii. Traditionally, behavioral models have been used primarily during the design phase where analysts refine the behavioral models to include implementation details.

**Que 2.17.** Write a short note on use case diagram and time diagram with suitable diagram and their utility in system design.

**AKTU 2010-11, Marks 05**

**Answer**

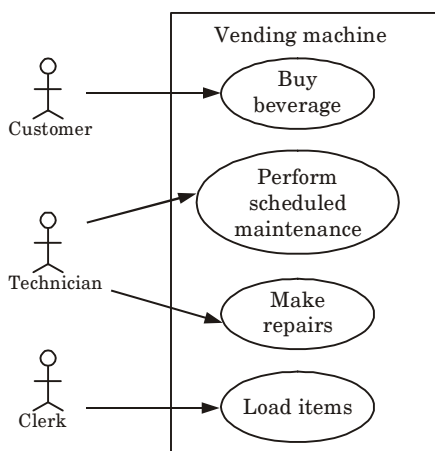
**Use case diagrams :**

1. Use case diagrams consist of actors, use cases and their relationships.
2. The diagram is used to model the system/subsystem of an application.
3. A single use case diagram captures a particular functionality of a system.
4. Hence to model the entire system, a number of use case diagrams are used.

**Utility of use case diagram in system design :**

1. It is used to gather the requirements of a system.
2. It is used to get an outside view of a system.
3. It identifies the external and internal factors influencing the system.
4. It shows the interaction among the requirements of actors.
5. It is used for requirement analysis and high level design.
6. It models the context of a system.
7. It is used in reverse engineering and forward engineering.

**For example :**



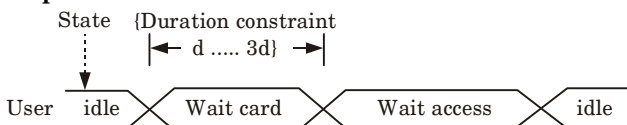
**Fig. 2.17.1.** Use case diagram for a vending machine.

**Time diagram :**

1. Timing diagram is used to show interactions when a primary purpose of the diagram is about time.
2. It focuses on conditions changing within and among lifelines along a linear time axis.
3. Timing diagram is a special form of a sequence diagram.

**Utility of time diagram in system design :**

1. It emphasizes at that particular time when the message has been sent among objects.
2. It explains the time processing of an object in detail.
3. It is employed with distributed and embedded systems.
4. It also explains how an object undergoes changes in its form throughout its lifeline.
5. It depicts a graphical representation of states of a lifeline per unit time.

**For example :****Fig. 2.17.2.**

**Que 2.18.** Define package. Explain the package diagram with

suitable diagram.

**AKTU 2014-15, Marks 05**

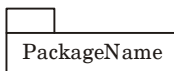
**OR**

**What are package diagrams and why are they used ?**

**AKTU 2013-14, Marks 05**

**Answer**

1. Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages.
2. A package is a group of elements (classes, associations, generalizations, and lesser packages) with a common theme.
3. A package partitions a model, making it easier to understand and manage. Large applications may require several tiers of packages.
4. Packages form a tree with increasing abstraction toward the root, which is the application, the top-level package.
5. As Fig. 2.18.1 shows, the notation for a package is a box with a tab. The purpose of the tab is to suggest the enclosed contents, like a tabbed folder.

**Fig. 2.18.1.** Notation for a package.

- For example, many business systems have a *Customer* package or a *Part* package; *Customer* and *Part* are dominant classes that are important to the business of a corporation and appear in many applications.

**Following are the uses of package diagram :**

- Package diagrams are used to structure high level system elements.
- Packages are used for organizing large system which contains diagrams and documents.
- Package diagram can be used to simplify complex class diagrams; it can group classes into packages.

**Que 2.19.** Write short notes on use case diagram with suitable diagram and their utility in system design.

**AKTU 2014-15, Marks 05**

**Answer**

**Use case diagrams :**

- Use cases describe how a system interacts with external user of a system (i.e., actor).
- Each use case represents a piece of functionality that a system provides to its users.
- Use cases are helpful for capturing informal requirements.
- Use case consists of actors.
- An actor is an object or set of objects that communicates directly with the system but that is not part of the system.
- The various interactions of actors with a system are quantized into use cases.
- A use case is a coherent piece of functionality that a system can provide by interacting with actors.
- For example, a customer actor can buy a beverage from a vending machine. The customer inserts money into the machine, makes a selection, and ultimately receives a beverage. Similarly, a repair technician can perform scheduled maintenance on a vending machine.
- Fig. 2.19.1 summarizes several use cases for a vending machine.

- **Buy a beverage :** The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance :** A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs :** A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items :** A stock clerk adds items into the vending machine to replenish its stock of beverages.

**Fig. 2.19.1.** Use case summaries for a vending machine.

**Utility of use cases diagram in system design :**

1. Use cases identify the functionality of a system and organize it according to the perspective of users.
2. Use cases describe complete transactions and are therefore less likely to omit necessary steps.
3. The main purpose of a system is almost always found in the use cases, with requirements lists supplying additional implementation constraints.

**Que 2.20.** What do you mean by activity diagram ? Explain in detail.

**AKTU 2014-15, Marks 05**

**OR**

What do you mean by activity diagram ? What are the two special states shown in an activity diagram ? Explain with an example.

**AKTU 2010-11, Marks 05**

**Answer**

1. An activity diagram is a flowchart that shows activities performed by a system.
2. The two special states shown in an activity diagram are the Initial State (Start Point) and Final State (End Point).
3. **Initial State or Start Point :** A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.



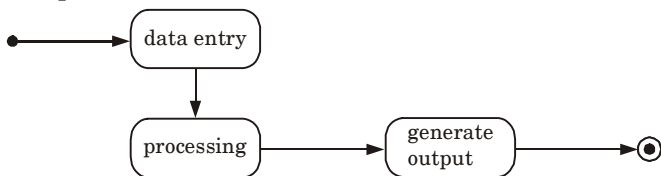
Start Point/Initial State

4. **Final State or End Point :** An arrow pointing to a filled circle nested inside another circle represents the final action state.



Ent Point Symbol

**For example :**



**Fig. 2.20.1.** Notation for activity states and activity-state transitions.

**Que 2.21.** Define state machine ? Draw a state machine diagram

for answering a telephone call.

**AKTU 2014-15, Marks 05**

**Answer**

**State machine :** A state machine diagram models the behaviour of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.



Diagram :

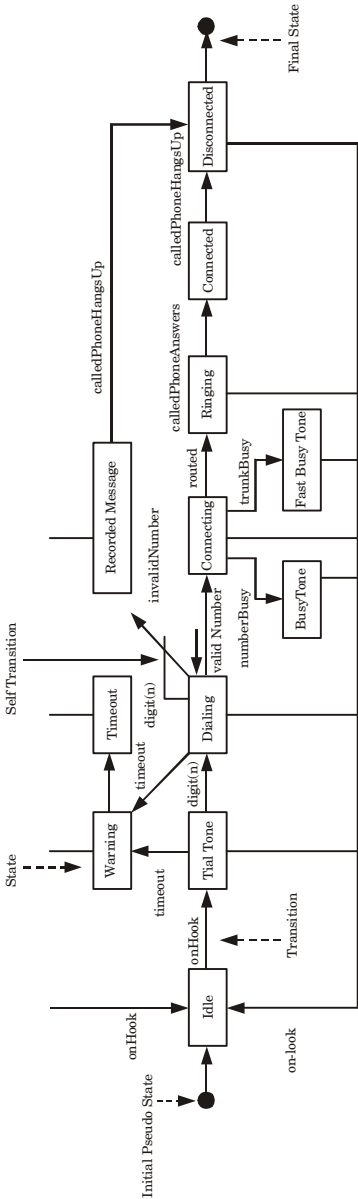


Fig. 2.21.1. State machine diagram for answering a telephone call.

**Que 2.22.** What do you mean by event ? What are the types of event explain with example ?

**AKTU 2014-15, Marks 05**

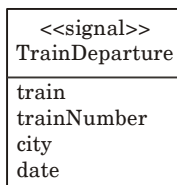
**Answer**

1. An event is something that happens at a point in time, such as user presses right mouse button. An event is a one-way transmission of information from one object to another. An event conveys information from one object to another.
2. An event has no duration. By definition, an event happens instantaneously with regard to time scale of an application.
3. Following are three most common events :

**A. Signal event :**

1. A signal event is the event of sending or receiving a signal.
2. A signal is a one-way transmission of information from one object to another.
3. A signal is a message between objects while a signal event is an occurrence in time.

For example, **TrainDeparture** has attributes train, trainNumber, city, and date. The UML notation is the keyword signal in guillemets (<< >>) above the signal class name in the top section of a box. The bottom section lists the signal attributes.



**Fig. 2.22.1.**

**B. Time event :**

1. A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval.
2. For example, Fig. 2.22.2 shows, the UML notation for an absolute time is the keyword *when* followed by a parenthesized expression involving time. The notation for time interval is the keyword *after* followed by a parenthesized expression that evaluates to a time duration.

- when (date = January 1, 2020)
- after (30 seconds)

**Fig. 2.22.2.**

**C. Change event :**

1. A change event is an event caused by the satisfaction of a boolean expression.
2. The boolean expression is continually tested and whenever the expression changes from false to true, the event occurs.
3. For example, the UML notation for a change event is the keyword *when* followed by a parenthesized boolean expression. Fig. 2.22.3 shows examples of change events.

- when (cabin temperature < heating set point)
- when (cabin temperature > cooling set point)

**Fig. 2.22.3.**

**Que 2.23.** Explain use case with example. How are the diagrams divided ?

**AKTU 2013-14, Marks 05****Answer****Use case diagrams :**

1. Use cases describe how a system interacts with external user of a system (*i.e.*, actor).
2. Each use case represents a piece of functionality that a system provides to its users.
3. Use cases are helpful for capturing informal requirements.
4. Use case consists of actors.
5. An actor is an object or set of objects that communicates directly with the system but that is not part of the system.
6. The various interactions of actors with a system are quantized into use cases.
7. A use case is a coherent piece of functionality that a system can provide by interacting with actors.
8. For example, a customer actor can buy a beverage from a vending machine. The customer inserts money into the machine, makes a selection, and ultimately receives a beverage. Similarly, a repair technician can perform scheduled maintenance on a vending machine.
9. Fig. 2.23.1 summarizes various use cases for a vending machine.

- **Buy a beverage :** The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance :** A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.

- **Make repairs** : A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items** : A stock clerk adds items into the vending machine to replenish its stock of beverages.

**Fig. 2.23.1.** Use case summaries for a vending machine.

**Use case diagrams are divided as :**

1. Use case diagram are divided into three use cases diagrams *i.e.*, Order, SpecialOrder, NormalOrder and one actor which is the customer.
2. The SpecialOrder and NormalOrder use cases are extended from Order use case.
3. Hence, they have extended relationship.
4. The actor Customer lies outside the system as it is an external user of the system.

**PART-6**

*Architectural Modeling, Component, Deployment, Component Diagrams and Deployment Diagrams.*

**Questions-Answers**

**Long Answer Type and Medium Answer Type Questions**

**Que 2.24.** Describe in brief component diagram.

**AKTU 2010-11, Marks 05**

**Answer**

**1. Component diagrams :**

1. The component diagrams are mainly used to model the static implementation view of a system.
2. They represent a high-level packaged view of the code.
3. They can be used to model executables, databases and adaptable systems.
4. Component diagrams mainly contain the following :

**i. Components :**

- a. A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.

- b. A component is a physical manifestation of an object that has a well-defined interface and a set of implementations for the interface.
- c. A complex system can be built using software components. It enhances re-use in the system and facilitates system evolution.

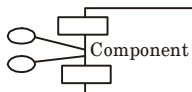


Fig. 2.24.1. Component.

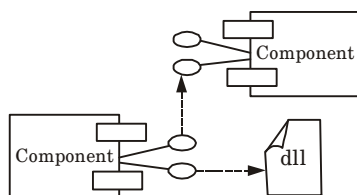


Fig. 2.24.2. Component diagram.

**ii. Interfaces :** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle. An interface possesses the following properties :

- a. Every interface is identified by a unique name, with an operational pathname.
- b. Interfaces do not specify any structure or implementation details of the operations. When an interface is represented as a rectangle, it has the same parts as a class. When it is represented as a circle, the display of these parts is suppressed.

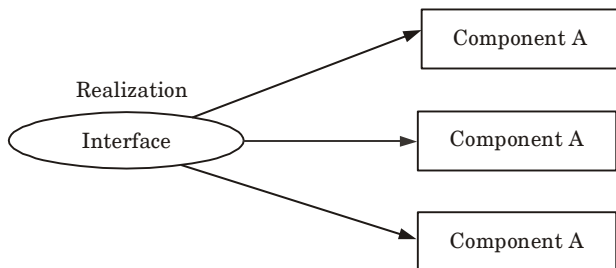
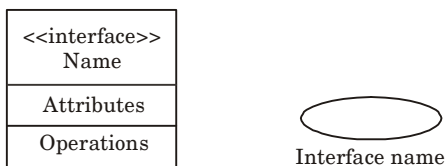


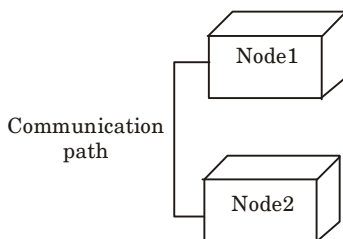
Fig. 2.24.3. Realization of an interface.

**Fig. 2.24.4.** Representation of interfaces.

**Que 2.25.** Explain the deployment diagram. What is the difference between components and nodes ?

**AKTU 2014-15, Marks 05****Answer****Deployment diagram :**

1. They display the configuration of run-time processing elements and the software components, processes and objects.
  2. The deployment diagram contains nodes and connection. A node is a piece of hardware in the system.
  3. A connection depicts the communication path used by the hardware
- Fig. 2.25.1.

**Fig. 2.25.1.** Deployment diagram.

4. Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.
5. Deployment diagrams are used to describe the static deployment view of a system.

**Difference :**

| Node                                                                                           | Component                                                                                                                                                        |
|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Node represents the physical part of the system for instance, server, network and printer etc. | Component represents any part of the system it might be physical aspect such as libraries, file, executables, document, packages etc., which reside on the node. |

# 3

## UNIT

# Object Oriented Analysis

## CONTENTS

- Part-1 :** Object Oriented Analysis, ..... 3-2L to 3-11L  
Object Oriented Design, Object  
Design, Combining Three Models,  
Designing Algorithm, Design  
Optimization, Implementation of  
Control, Adjustment of Inheritance,  
Object Representation, Physical  
Packaging, Documenting  
Design Consideration
- Part-2 :** Structured Analysis and Structured... 3-11L to 3-17L  
Design (SA/SD), Jackson Structured  
Development (JSD), Mapping Object  
Oriented Concepts Using Non-Object  
Oriented Language, Translating  
Classes In to Data Structures,  
Passing Arguments to Method,  
Implementing Inheritance,  
Association Encapsulation
- Part-3 :** Object Oriented Programming, ..... 3-17L to 3-21L  
Reusability, Extensibility,  
Robustness, Programming in the  
Large, Procedural v/s OOP,  
Object Oriented Language Feature,  
Abstraction and Encapsulation

**PART-1**

*Object Oriented Analysis, Object Oriented Design, Object Design, Combining Three Models, Designing Algorithm, Design Optimization, Implementation of Control, Adjustment of Inheritance, Object Representation, Physical Packaging, Documenting Design Consideration.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 3.1.** Write short note on object oriented analysis and object oriented design.

**Answer****Object oriented analysis :**

1. Object-Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.
2. The primary tasks in object-oriented analysis (OOA) are :
  - a. Identifying objects
  - b. Organizing the objects by creating object model diagram.
  - c. Defining the internals of the objects, or object attributes.
  - d. Defining the behavior of the objects, i.e., object actions.
  - e. Describing how the objects interact.
3. The common models used in OOA are use cases and object models.

**Object-Oriented Design :**

1. Object-Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis.
2. In OOD, concepts in the analysis model, which are technology independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.
3. The implementation details include :



- a. Restructuring the class data (if necessary),
- b. Implementation of methods, i.e., internal data structures and algorithms,
- c. Implementation of control, and
- d. Implementation of associations.

**Que 3.2. Write short note on object design.**

**Answer**

1. After the hierarchy of subsystems has been developed, the objects in the system are identified and their details are designed.
2. Here, the designer details out the strategy chosen during the system design.
3. The emphasis shifts from application domain concepts toward computer concepts.
4. The objects identified during analysis are etched out for implementation with an aim to minimize execution time, memory consumption, and overall cost.
5. Object design includes the following phases :
  - a. Object identification
  - b. Object representation, i.e., construction of design models
  - c. Classification of operations
  - d. Algorithm design
  - e. Design of relationships
  - f. Implementation of control for external interactions
  - g. Package classes and associations into modules

**Que 3.3. How can we design an algorithm ? Explain**

**Answer**

1. The operations in the objects are defined using algorithms.
2. An algorithm is a stepwise procedure that solves the problem laid down in an operation. Algorithms focus on how it is to be done.
3. There may be more than one algorithm corresponding to a given operation.
4. Once the alternative algorithms are identified, the optimal algorithm is selected for the given problem domain.
5. The metrics for choosing the optimal algorithm are :
  - a. **Computational Complexity** : Complexity determines the efficiency of an algorithm in terms of computation time and memory requirements.

- b. **Flexibility :** Flexibility determines whether the chosen algorithm can be implemented suitably, without loss of appropriateness in various environments.
- c. **Understandability :** This determines whether the chosen algorithm is easy to understand and implement.

**Que 3.4.** What are object oriented model ? Explain.

**Answer**

Refer Q. 1.13, Page 1-12L, Unit-1.

**Que 3.5.** What are the three models in OMT ? How is the object oriented analysis and design attached with OMT ? Explain with an example.

AKTU 2013-14, Marks 10

**Answer**

Following are the three models in OMT :

**1. Object model :**

- a. Object model encompasses the principles of abstraction, encapsulation, modularity, hierarchy, typing, concurrency and persistence.
- b. Object model emphasizes on the object and class.
- c. Main concepts related with object model are classes and their association with attributes.
- d. Predefined relationships in object model are aggregation and generalization (multiple inheritance).

**2. Dynamic model :**

- a. Dynamic model involves states, events and state diagram (transition diagram) on the model.
- b. Main concepts related with dynamic model are states, transition between states and events to trigger the transitions.
- c. Predefined relationships in object model are aggregation (concurrency) and generalization.

**3. Functional model :**

- a. Functional Model focuses on the how data is flowing, where data is stored and different processes.
- b. Main concepts involved in functional model are data, data flow, data store, process and actors.
- c. Functional model describes the whole processes and actions with the help of data flow diagram (DFD).

**OOAD attachment with OMT :**

1. Object Modeling Technique (OMT) combines the three views of modeling systems.
2. The object model represents the static, structural, “data” aspects of a system.
3. The dynamic model represents the temporal, behavioral, “control” aspects of a system.
4. The functional model represents the transformational, “function” aspects of a system.
5. A typical object oriented software procedure incorporates all three aspects : It uses data structures (object model), it sequences operations in time (dynamic model), and it transforms values (functional model). Each model contains references to entities in other models.
6. For example, operations are attached to objects in the object model but more fully expanded in the functional model.

**Que 3.6.**

**Describe the relation of functional model, object model and dynamic models. What is relationship and difference between OOA (Object oriented analysis) and OOD (Object oriented design) ?**

**AKTU 2014-15, Marks 10**

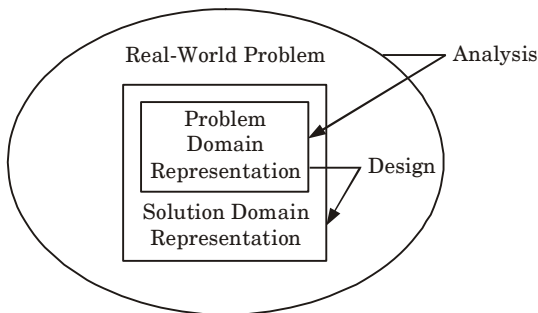
**Answer**

1. The functional model shows what “has to be done” by a system. The leaf process are the operation on objects.
2. The object model shows the “doers” the objects. Each process is implemented by a method on some object.
3. The dynamic model shows the sequences in which the operations are performed. Each sequence is implemented as a sequence, loop or alteration of statements within some method.
4. The processes in the functional model correspond to operations in the object model.
5. Actors are explicit objects in the object models. Data flows to or from actors represent operations on or by the objects.
6. Because actors are self-motivated objects, the functional model is not sufficient to indicate when they act. The dynamic model for an actor object specifies when it acts.

**Relationship between OOA and OOD :**

1. When object orientation is used in analysis as well as design, the boundary between OOA and OOD is blurred. This is particularly true in methods that combine analysis and design.

- One reason for this blurring is the similarity of basic constructs (*i.e.*, objects and classes) that are used in OOA and OOD.



**Fig. 3.6.1.** Relationship between OOA and OOD.

### Difference between OOA and OOD :

- The fundamental difference between OOA and OOD is that OOA models the problem domain, leading to an understanding and specification of the problem, while the OOD models the solution to the problem.
- That is, analysis deals with the problem domain, while design deals with the solution domain.

**Que 3.7.**

**What do you mean by the optimization of design ? Discuss the design optimization with suitable example using diagrams.**

**AKTU 2012-13, Marks 05**

**Answer**

### Optimization of design :

- Design optimization is an engineering design methodology using a mathematical formulation of a design problem to support selection of the optimal design among many alternatives.
- Design optimization involves the following stages :
  - Variables** : Describe the design alternatives.
  - Objective** : Elected functional combination of variables (to be maximized or minimized).
  - Constraints** : Combination of variables expressed as equalities or inequalities that must be satisfied for any acceptable design alternative.
  - Feasibility** : Values for set of variables that satisfies all constraints and minimizes/maximizes objective.

**For example :** Consider the design of a company's employee skills database. Fig. 3.7.1 shows a portion of the analysis class model. The operation `companyfindskill()` returns a set of persons in the company with a given skill. For example, an application might need all the employees who speak Japanese.



**Fig. 3.7.1.**

**Que 3.8.**

**Describe implementation of control in object oriented design.**

**Answer**

1. The object designer may incorporate refinements in the strategy of the state-chart model. In system design, a basic strategy for realizing the dynamic model is made. During object design, this strategy is aptly embellished for appropriate implementation.
2. The approaches for implementation of the dynamic model are :
  - a. **Represent state as a location within a program :**
    - i. This is the traditional procedure-driven approach whereby the location of control defines the program state.
    - ii. A finite state machine can be implemented as a program.
    - iii. A transition forms an input statement, the main control path forms the sequence of instructions, the branches form the conditions, and the backward paths form the loops or iterations.
  - b. **State machine engine :**
    - i. This approach directly represents a state machine through a state machine engine class.
    - ii. This class executes the state machine through a set of transitions and actions provided by the application.
  - c. **Control as concurrent tasks :**
    - i. In this approach, an object is implemented as a task in the programming language or the operating system.
    - ii. Here, an event is implemented as an inter-task call.
    - iii. It preserves inherent concurrency of real objects.

**Que 3.9.**

**What do you mean by object representation ?**

**Answer**

1. Once the classes are identified, they need to be represented using object modeling techniques.
2. This stage essentially involves constructing UML diagrams.
3. There are two types of design models that need to be produced :

- a. **Static Models :** To describe the static structure of a system using class diagrams and object diagrams.
- b. **Dynamic Models :** To describe the dynamic structure of a system and show the interaction between classes using interaction diagrams and state-chart diagrams.

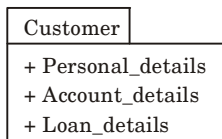
**Que 3.10. Describe physical packaging with example.**

**AKTU 2010-11, Marks 2.5**

**Answer**

**Physical packaging :**

- 1. A component is a physical and replaceable part of the system that conforms and provides the realization of a set of interfaces.
- 2. It represents the physical packaging of elements like classes and interfaces.
- 3. A package is an organized group of elements. A package may contain structural things like classes, components, and other packages in it.
- 4. For example, a package is represented by a tabbed folder.
- 5. A package is generally drawn with only its name. However, it may have additional details about the contents of the package.



**Que 3.11. What are the different aspects of packaging ?**

**Answer**

The different aspects of packaging are :

- 1. **Hiding internal information from outside view :**
  - a. It allows a class to be viewed as a "black box" and permits class implementation to be changed without requiring any clients of the class to modify code.
- 2. **Coherence of elements :**
  - a. An element, such as a class, an operation, or a module, is coherent if it is organized on a consistent plan and all its parts are intrinsically related so that they serve a common goal.
- 3. **Construction of physical modules :** The following guidelines help while constructing physical modules :

- a. Classes in a module should represent similar things or components in the same composite object.
- b. Closely connected classes should be in the same module.
- c. Unconnected or weakly connected classes should be placed in separate modules.
- d. Modules should have good cohesion, i.e., high cooperation among its components.
- e. A module should have low coupling with other modules, i.e., interaction or interdependence between modules should be minimum.

**Que 3.12.** What do you mean by documentation? What are the various considerations in documentation designing? Explain.

AKTU 2011-12, Marks 05

**Answer**

**Documentation :**

1. Documentation is a software development process that records the procedure of making the software.
2. The design decisions need to be documented for any non-trivial software system for transmitting the design to others.
3. Though a secondary product, a good documentation is indispensable, particularly in the following areas :
  - a. In designing software that is being developed by a number of developers.
  - b. In iterative software development strategies.
  - c. In developing subsequent versions of a software project.
  - d. For evaluating a software.
  - e. For finding conditions and areas of testing.
  - f. For maintenance of the software.

**Various Consideration of documentation designing :**

**1. It is a roadmap :**

- a. It allows standardization, and it helps to identify the stages that can be improved.
- b. Process documentation also facilitates the training of new employees.

**2. It is everyone's task :**

- a. The members of an area or project are responsible for documenting their processes.
- b. Every employee knows their own functioning, their strength, and their weakness, so they are the ones better indicates to document their processes.

**3. Make them public :**

- The documentation of the processes must be available to all team and company members.
- Restricting access to documentation creates the false illusion that it's only relevant to a particular group.

**4. Flexible documentation :**

- Companies change, update, improve, so their processes are also subject to constant changes. To improve the effectiveness of the process, incorporate the necessary adjustments to the documentation of the process.
- Document the date of the last update.
- Save a backup copy of the files that document the process.
- Review the documents at least once a year.

**Que 3.13.** What do you mean by multiple inheritances ? Explain it with an example. Can you implement multiple inheritances in Java ?

AKTU 2010-11, Marks 05

**Answer****Multiple inheritance :**

- Multiple inheritance is a feature of object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class.
- When one class extends more than one classes then this is called multiple inheritance.

**For example :** Class C extends class A and B then this type of inheritance is known as multiple inheritance.

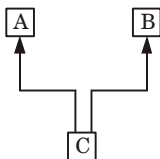


Fig. 3.13.1.

**Multiple inheritance in Java :**

- Java does not allow multiple inheritance, and we cannot extend more than one other class.
- Java doesn't allow multiple inheritance to avoid the ambiguity caused by it.



**Que 3.14.** Describe documenting design considerations. How do

we perform adjustment of inheritance ?

**AKTU 2013-14, Marks 10**

**Answer**

**Documentation design considerations :** Refer Q. 3.12, Page 3-9L, Unit-3.  
**Following kinds of adjustments can be used to increase the chance of inheritance :**

1. Some operations may have fewer arguments than others. The missing arguments can be added but ignored. For example, a draw operation on a monochromatic display does not need a color parameter, but the parameter can be accepted and ignored for consistency with color displays.
2. Similar attributes in different classes may have different names. Give the attributes the same name and move them to the common ancestor class. Then operations that access the attributes will match better.
3. Some operations may have fewer arguments because they are special cases of more general arguments. Implement the special operations by calling the general operation with appropriate parameter values.
4. Opportunities to use inheritance are not always recognized during the analysis phase of development, so it is worthwhile to reexamine the object model looking for commonality between classes.

## **PART-2**

*Structured Analysis and Structured Design(SA/SD), Jackson Structured Development(JSD), Mapping Object Oriented Concepts Using Non-Object Oriented Language, Translating Classes Into Data Structures, Passing Arguments to Method, Implementing Inheritance, Association Encapsulation.*

## **Questions-Answers**

### **Long Answer Type and Medium Answer Type Questions**

**Que 3.15.** Describe the structured analysis and structured design

approach with an example.

**AKTU 2012-13, Marks 05**

**OR**

**Write short notes on : SA/SD.**

**AKTU 2013-14, Marks 05**

OR

**Explain structured analysis and structured design (SA/SD) with example.**

AKTU 2015-16, Marks 10

**Answer****Structured analysis and structure design :**

1. Structured Analysis and Structured Design (SA/SD) is diagrammatic notation which is design to help people understand the system.
2. The basic goal of SA/SD is to improve quality and reduce the risk of system failure.
3. It establishes concrete management specification and documentation.
4. It focuses on solidity, pliability and maintainability of system.
5. The approach of SA/SD is based on the Data Flow Diagram.
6. It is easy to understand SA/SD but it focuses on well defined system boundary whereas JSD approach is too complex and does not have any graphical representation.
7. SA/SD is combined known as SAD and it mainly focuses on following three points :
  - a. System
  - b. Process
  - c. Technology
8. SA/SD involves two phases :
  - a. **Analysis Phase :** It uses Data Flow Diagram, Data Dictionary, State Transition diagram and ER diagram.
  - b. **Design Phase :** It uses Structure Chart and Pseudo Code.
9. **For example :**
  - i. During structured design, data flow diagram processes are grouped into tasks and allocated to operating system processes and CPUs.
  - ii. Data flow diagram processes are converted into programming language functions, and a structure chart is created showing the procedure call tree.

**Que 3.16.**

**Compare the OMT methodology with SA/SD methodology. Explain with suitable example.**

AKTU 2014-15, Marks 10

**Answer**

| S. No. | OMT methodology                                          | SA/SD methodology                                        |
|--------|----------------------------------------------------------|----------------------------------------------------------|
| 1.     | It manages a system around procedures.                   | It manages system around real world objects.             |
| 2.     | Focus more on functional model and less on object model. | Focus more on object model and less on functional model. |
| 3.     | Dominance order is functional, dynamic and object.       | Dominance order is object, dynamic and functional.       |
| 4.     | It is a historical approach.                             | It is advanced approach.                                 |
| 5.     | Reusability of components across projects is less.       | Reusability of components across projects is more.       |
| 6.     | Not easily modifiable and extensible.                    | Easily modifiable and extensible.                        |
| 7.     | Used when functions are more important than data.        | Used when data is more important than functions.         |

**Que 3.17. Write short note on Jackson Structured Development.**

**AKTU 2011-12, Marks 05**

**AKTU 2012-13, Marks 05**

**OR**

**Write short note on JSD.**

**AKTU 2013-14, Marks 05**

**Answer**

- Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
- JSD can start from the stage in a project when there is only a general statement of requirements.
- Following are the phases of JSD :
  - Modeling phase :** In the modeling phase, the designer creates a collection of entity structure diagrams and identifies the entities in the system, the actions they perform, the attributes of the actions and time ordering of the actions in the life of the entities.

- b. Specification phase :** This phase focuses on actually what is to be done. Major goal is to map progress in the real world on progress in the system that models it.
- c. Implementation phase :**
  - i. In the implementation phase JSD determines how to obtain the required functionality.
  - ii. Implementation way of the system is based on transformation of specification into efficient set of processes.

**Que 3.18. How do you map the object-oriented concepts using non-object oriented languages ? Explain with an example.**

**AKTU 2010-11, Marks 05**

**OR**

**How object-oriented concept can be implemented using non-object-oriented language? Explain with an example.**

**AKTU 2011-12, Marks 05**

**Answer**

Implementing an object-oriented concept in a non-object oriented language requires the following steps :

**1. Translate classes into data structures :**

- i. Each class is implemented as a single contiguous block of attributes. Each attribute contains variable. Now an object has state and identity and is subject to side effects.
- ii. A variable that identifies an object must therefore be implemented as a sharable reference.

**2. Pass arguments to methods :**

- i. Every method has at least one argument. In a non-object-oriented language, the argument must be made explicit.
- ii. Methods can contain additional objects as arguments. In passing an object as an argument to a method, a reference to the object must be passed if the value of the object can be updated within the method.

**3. Allocate storage for objects :**

- i. Objects can be allocated statically, dynamically or on a stack.
- ii. Most temporary and intermediate objects are implemented as stack-based variables.
- iii. Dynamically allocated objects are used when there number is not known at compile time.
- iv. A general object can be implemented as a data structure allocated on request at run time from a heap.

4. **Implement inheritance in data structures :** Following ways are use to implement data structures for inheritance in a non-object-oriented language :
  - i. Avoid it.
  - ii. Flatten the class hierarchy.
  - iii. Break out separate objects.
5. **Implement method resolution :** Method resolution is one of the main features of an object-oriented language that is lacking in a non-object-oriented language. Method resolution can be implemented in following ways :
  - i. Avoid it.
  - ii. Resolve methods at compile time.
  - iii. Resolve methods at run time.
6. **Implement associations :** Implementing associations in a non-object-oriented language can be done by :
  - i. Mapping them into pointers.
  - ii. Implementing them directly as association container objects.
7. **Deal with concurrency :**
  - i. Most languages do not explicitly support concurrency.
  - ii. Concurrency is usually needed only when more than one external event occurs, and the behaviour of the program depends on their timing.
8. **Encapsulate internal details of classes :**
  - i. Object-oriented languages provide constructs to encapsulate implementation.
  - ii. Some of this encapsulation is lost when object-oriented concept is translated into a non-object-oriented language, but we can still take advantage of the encapsulation facilities provided by the language.

**Que 3.19. Write short note on translating object oriented design into an implementation.**

**AKTU 2010-11, Marks 05**

**AKTU 2014-15, Marks 05**

### **Answer**

- i. It is easy to implement an object-oriented design with an object-oriented language since language constructs are similar to design constructs.
- ii. The following steps are required to implement an object oriented design in an object-oriented language :

**1. Class definitions :**

- i. The first step in implementing an object-oriented design is to declare object classes. Each attribute and operation in an object diagram must be declared as part of its corresponding class.
- ii. Assign data types to attributes. Declare attributes and operations as either public or private.

**2. Creating objects :**

- i. Object-oriented languages create new objects in following two ways :
  - a. Class operation applied to a class object creates a new object of the class.
  - b. Using special operations that create new objects.
- ii. When a new object is created, the language allocates storage for its attribute values and assigns it a unique object ID.

**3. Calling operations :**

- i. In most object-oriented languages, each operation has at least one implicit argument, the target object, indicated with a special syntax.
- ii. Operations may or may not have additional arguments.

**4. Using Inheritance :**

- i. To implement inheritance object-oriented languages use different mechanisms.
- ii. There are three independent dimensions for classifying inheritance mechanisms :
  - a. Static or dynamic
  - b. Implicit or explicit
  - c. Per object or per group.
- iii. Many of the popular languages are static, implicit and per group.

**5. Implementing associations :**

- i. There are two approaches to implement associations : buried pointers and distinct association objects.
- ii. If the language does not explicitly support association objects then buried pointers are easy to implement.
- iii. An association can also be implemented as a distinct container object.

**Que 3.20. Describe passing arguments to method with example.**

**AKTU 2012-13, Marks 05**

**Answer**

1. There are different ways in which parameter data can be passed into and out of methods and functions.

2. Let us assume that a function B() is called from another function A(). In this case A is called the “caller function” and B is called the “called function or callee function”.
3. The arguments which A sends to B are called actual arguments and the parameters of B are called formal arguments.
4. Following are the types of parameters :
  - a. **Formal parameter** : A variable and its type as they appear in the prototype of the function or method.  
**Syntax** : function\_name(datatype variable\_name) :
  - b. **Actual parameter** : The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.  
**Syntax** : func\_name(variable name(s));

**Que 3.21.** Describe implementation of inheritance with example.

AKTU 2012-13, Marks 05

**Answer**

1. Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship.
2. Object-oriented languages provide strong support for implementation of inheritance.
3. Inheritance has become synonymous with code reuse within the object-oriented programming community.
4. After modeling a system, the developer looks at the resulting classes and tries to group similar classes together and reuse common code.
5. Often code is available from past work (such as a class library) which the developer can reuse and modify to get the precise desired behavior.

### PART-3

*Object Oriented Programming, Reusability, Extensibility, Robustness, Programming in the Large, Procedural v/s OOP, Object Oriented Language Feature, Abstraction and Encapsulation.*

### Questions-Answers

**Long Answer Type and Medium Answer Type Questions**

**Que 3.22.** Describe the various features of object-oriented languages. Also compare any two object-oriented languages.

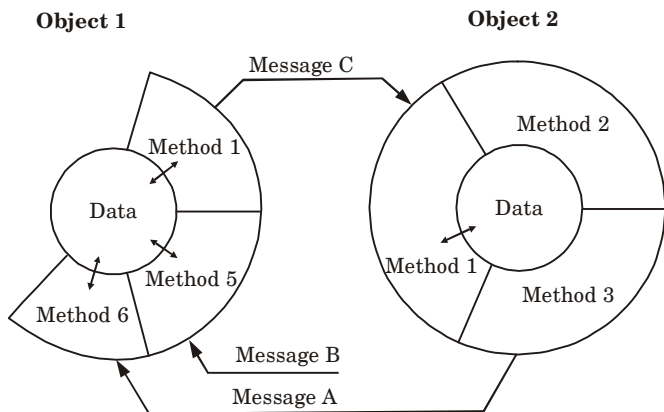
**AKTU 2010-11, Marks 05**

**Answer**

**Features of object-oriented language are :**

**1. Encapsulation :**

- Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.
- The only way to reach the data is through these particular methods (see Fig. 3.22.1).
- It is the mechanism that binds together code and the data it manipulates.
- This concept is also used to hide the internal representation, or state, of an object from the outside.



**Fig. 3.22.1.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

**2. Polymorphism :**

- Polymorphism means having many forms.
- Polymorphism is the ability of a message to be displayed in more than one form.
- It plays an important role in allowing objects having different internal structure to share the same external interface.

**3. Inheritance :**

- Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features.
- Inheritance is mainly used for code reusability.



**Difference :**

| S. No. | C++                                   | Java                                                                                                      |
|--------|---------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 1.     | C++ is a platform dependent language. | Java is platform-independent language.                                                                    |
| 2.     | C++ is a compiled language.           | Java is a compiled as well as an interpreted language.                                                    |
| 3.     | C++ code is not portable.             | Java, translates the code into byte code. This byte code is portable and can be executed on any platform. |
| 4.     | Memory management in C++ is manual.   | In Java the memory management is automatic.                                                               |

**Que 3.23.** Explain each of the following with in reference to object oriented programming style with an example :

- i. Reusability
- ii. Robustness
- iii. Extensibility
- iv. Abstraction

**AKTU 2011-12, Marks 10**

**OR**

Write short notes on the following reusability and robustness.

**AKTU 2010-11, Marks 05**

**Answer****i. Reusability :**

1. Reusability is a segment of source code that can be used again to add new functionalities with slight or no modification.
2. Reusable software reduces design, coding and testing cost. Also, reducing the amount of code simplifier understanding. In object-oriented languages the possibility of code reuse is greatly enhanced.
3. There are two kind of reuse :
  - i. Sharing of newly-written code within a project.
  - ii. Reuse of previously-written code on new project.
4. **Rules for reusability :** Following rules must be kept in mind :
  - i. Keep methods coherent.
  - ii. Keep methods small.
  - iii. Keep method consistent.

- iv. There should be separate policy and implementation.
- v. Provide uniform average.
- vi. Broaden the method as much as possible.
- vii. Avoid global information.
- viii. Avoid modes.

## ii. Robustness :

- 1. A method is robust if it does not fail even if it receives improper parameters.
- 2. Robustness against internal bugs may be traded off against efficiency.
- 3. Robustness against user errors should never be sacrificed.
- 4. Rules for robustness :
  - a. **Protect against errors** : Software should protect itself against incorrect user input. Incorrect user input should never cause a crash.
  - b. **Optimize after the program runs** : Don't optimize a program until you get it working.
  - c. **Validate arguments** : External operations, those available to users of the class, must rigorously check their arguments to prevent failure. Don't include arguments that cannot be validated.
  - d. **Avoid predefined limits** : When possible use dynamic memory allocation to create data structures that do not have predefined limits.
  - e. **Instrument the program for debugging and performance monitoring** : You should instrument your code for debugging, statistics, and performance. The level of debugging that you must build into your code depends on the programming environment presented by the language.

## iii. Extensibility :

- 1. Extensibility is a software engineering and systems design principle that provides for future growth.
- 2. Extensibility is a measure of the ability to extend a system and the level of effort required to implement the extension.
- 3. Extensions can be through the addition of new functionality or through modification of existing functionality.
- 4. The principle provides for enhancements without impairing existing system functions.

**For example** : Object-oriented application frameworks which achieve extensibility typically by using inheritance and dynamic binding.

## iv. Abstraction :

- 1. Abstraction is the selective examination of certain aspects of a problem.
- 2. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant.

3. Abstraction must always be for some purpose, because the purpose determines what is and is not important.
4. Many different abstractions of the same thing are possible, depending on the purpose for which they are made.

**Que 3.24. Compare Abstraction and Encapsulation.**

**AKTU 2015-16, Marks 05**

**Answer**

| S. No. | Abstraction                                                           | Encapsulation                                                                                                                  |
|--------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 1.     | Abstraction is the process or method of gaining the information.      | Encapsulation is the process or method to contain the information.                                                             |
| 2.     | In abstraction, problems are solved at the design or interface level. | In encapsulation, problems are solved at the implementation level.                                                             |
| 3.     | Abstraction is the method of hiding the unwanted information.         | Encapsulation is a method to hide the data in a single entity or unit along with a method to protect information from outside. |
| 4.     | We can implement abstraction using abstract class and interfaces.     | Encapsulation is implemented using by access modifier <i>i.e.</i> private, protected and public.                               |
| 5.     | The objects that help to perform abstraction are encapsulated.        | Objects that result in encapsulation need not be abstracted.                                                                   |



# 4

## UNIT

# C++ Basics & Functions

---

## CONTENTS

---

**Part-1** : C++ Basics, Overview, ..... 4-2L to 4-10L  
Program Structure, Namespace,  
Identifiers, Variables,  
Constants, Enum, Operators,  
Typecasting, Control Structures

**Part-2** : C++ Function, Simple Function, ..... 4-10L to 4-21L  
Call and Return by Reference,  
Inline Function Macro vs Inline  
Functions, Overloading of  
Functions, Default Arguments,  
Friend Functions, Virtual Function

**PART- 1**

*C++ Basics, Overview, Program Structure, Namespace, Identifiers, Variables, Constants, Enum, Operators, Typecasting, Control Structures.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 4.1.** Write short note on C++ and its applications.

**Answer**

1. C++ is an object-oriented programming language developed by Bjarne Stroustrup.
2. C++ is a cross-platform language that can be used to create high-performance applications.
2. C++ is a versatile language for handling very large programs.
3. It is suitable for variety of programming task including development of editors, compilers, and different complex real-life application systems.

**Applications of C++ :**

1. Since C++ allows us to create hierarchy-related objects, we can build special object oriented libraries which can be used later by many programmers.
2. While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine-level details.
3. C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add to the existing structure of an object.

**Que 4.2.** Explain the structure of C++ program.

**Answer**

1. Fig. 4.2.1 shows the structure of C++ program that contain four sections.
2. The class declarations are placed in a header file and the definitions of member functions go into another file.
3. This approach enables the programmer to separate the abstract specification of the interface (class definition) from the implementation details (member functions definition).

4. The main program that uses the class is placed in a third file which “includes” the previous two files as well as any other files required.

|                              |
|------------------------------|
| Include files                |
| Class declaration            |
| Member functions definitions |
| Main function program        |

**Fig. 4.2.1.** Structure of a C++ program.

**Que 4.3.** Describe briefly the term namespace, identifiers, variables constants, enum.

**Answer**

**A. Namespace :**

1. Namespace defines a scope for the identifiers that are used in a program.
2. For using the identifiers defined in the namespace scope we must include the using directive :  
using namespace std;
3. Here, std is the namespace where standard class libraries are defined.
4. This directive will bring all the identifiers defined in std to the current global scope.

**B. Identifiers :**

1. Identifiers refer to the names of variables, functions, arrays, classes, etc., which are created by the programmer.
2. They are the fundamental requirement of any language.
3. Each language has its own rules for naming these identifiers. Following are the rules for C++ :
  - a. Only alphabetic characters, digits and underscores are permitted.
  - b. The name cannot start with a digit.
  - c. Uppercase and lowercase letters are distinct.
  - d. A declared keyword cannot be used as a variable name.

**C. Constants :**

1. Constants refer to fixed values that do not change during the execution of a program.
2. They include integers, characters, floating point numbers and strings.
3. Literal constant do not have memory locations.

**4. Examples :**

```
123           // decimal integer
12.34         // floating point integer
037           // octal integer
0X2           // hexadecimal integer
"C++"         // string constant
'A'           // character constant
L'ab'         // wide-character constant
```

**D. Enumerated :**

1. An enum specifier defines the set of all names that will be permissible values of the type. These permissible values are called members.
2. The enum type `days_of_week` has seven members : Sun, Mon, Tue, and so on, up to Sat.
3. Enumerated means that all the values are listed.
4. This is unlike the specification of an int, for example, which is given in terms of a range of possible values.
5. In an enum you must give a specific name to every possible value.

**E. Variables :**

1. A variable is a symbolic name that can be given a variety of values.
2. Variables are stored in particular places in the computer's memory.
3. When a variable is given a value, that value is actually placed in the memory space occupied by the variable.
4. Most popular languages use the same general variable types, such as integers, floating-point numbers, and characters.

**Que 4.4.****Write a C++ program to calculate the value of sin (x) and****cos (x).****Answer**

```
#include<iostream>
#include<math.h>
using namespace std;
//calculate value of sin
void cal_sin(float n) {
    float acc = 0.0001, denominator, sinx, sinval;
    n = n * (3.142 / 180.0); //convert in radian
    float temp = n;
    sinx = n;
```

```
    sinval = sin(n);
int i = 1;
do {
    denominator = 2 * i * (2 * i + 1);
    temp = -temp * n * n / denominator;
    sinx = sinx + temp;
    i = i + 1;
} while (acc <= fabs(sinval - sinx));
cout<<sinx;
}

//calculate value of cos
void cal_cos(float n) {
    float acc = 0.0001, temp, denominator, cosx, cosval;
    n = n * (3.142 / 180.0); //convert in radian
    temp = 1;
    cosx = temp;
    cosval = cos(n);
int i = 1;
do {
    denominator = 2 * i * (2 * i - 1);
    temp = -temp * n * n / denominator;
    cosx = cosx + temp;
    i = i + 1;
} while (acc <= fabs(cosval - cosx));
cout<< cosx;
}

int main() {
    float n = 30;
    cout<<"value of Sin is : "; cal_sin(n);
    cout<<"\n";
    n=60;
    cout<<"value of Cos is : ";
    cal_cos(n);
    return 0;
}
```

**Que 4.5.****What are the different types of operators used in C++ ?**



**Answer****1. Memory Management Operators :**

- a. C++ defines two unary operators **new** and **delete** that perform the task of allocating and freeing the memory in a easier way.
- b. Since these operators manipulate memory on the free store, they are also known as free store operators.
- c. The **new** operator can be used to create objects of any type. It takes the following general form :

pointer-variable = new data-type;

- d. When a data object is no longer needed, it is destroyed using the **delete** operator.

**2. Manipulators :**

- a. Manipulators are operators that are used to format the data display.
- b. The most commonly used manipulators are **endl** and **setw**.
- c. The **endl** manipulator, when used in an output statement, causes a linefeed to be inserted.
- d. It has the same effect as using the newline character “\n”.

**3. Type cast operator :**

- a. C++ permits explicit type conversion of variables or expressions using the type cast operator.
- b. The following two versions are equivalent :  
(type-name) expression // C notation  
type-name (expression) // C++ notation
- c. A type-name behaves as if it is a function for converting values to a designated type.
- d. The function-call notation usually leads to simplest expressions.
- e. However, it can be used only if the type is an identifier. For example,  
`p = int * (q);`  
is illegal. In such cases, we must use C type notation.  
`p = (int *) q;`

**Que 4.6. Explain typecasting in C++.****Answer****Typecasting :**

- 1. Converting an expression of a given type into another type is known as typecasting.
- 2. Following are the types of typecasting :

**a. Implicit conversion :**

- i. Implicit conversions do not require any operator.
- ii. They are automatically performed when a value is copied to a compatible type.
- iii. For example :  
short a = 2000;  
int b;  
b = a;

**b. Explicit conversion :**

- i. Many conversions, especially those that imply a different interpretation of the value, require an explicit conversion.
- ii. For example :  
short a = 2000;  
int b;  
b = (int) a; // C like cast notation  
b = int (a); // functional notation

**c. Dynamic cast :**

- i. `dynamic_cast` can be used only with pointers and references to objects.
- ii. Its purpose is to ensure that the result of the type conversion is a valid complete object of the requested class.

**d. Static cast :**

- i. `static_cast` can perform conversions between pointers to related classes, not only from the derived class to its base, but also from a base class to its derived.

**e. reinterpret\_cast :**

- i. `reinterpret_cast` converts any pointer type to any other pointer type, even of unrelated classes.
- ii. The operation result is a simple binary copy of the value from one pointer to the other.

- f. const\_cast :** This type of casting manipulates the constness of an object, either to be set or to be removed.

**g. typeid :**

- i. `typeid` allows to check the type of an expression :  
`typeid (expression)`
- ii. This operator returns a reference to a constant object of type `type_info` that is defined in the standard header file `<typeinfo>`.

**Que 4.7.****Describe various control statements used in C++.**

**Answer**

1. **The if statement :** The if statement is implemented in two forms :

- a. Simple if statement
- b. if...else statement

2. **The switch statement :**

- a. This is a multiple-branching statement where, based on a condition, the control is transferred to one of the many possible points.
- b. This is implemented as follows :

```
switch (expression)
```

```
{  
    case1 :  
    {  
        action1;  
    }  
    case2 :  
    {  
        action2;  
    }  
    case3 :  
    {  
        action3;  
    }  
    default :  
    {  
        action4;  
    }  
}  
action5;
```

3. **The do-while statement :**

- a. The do-while is an exit-controlled loop. Based on a condition, the control is transferred back to a particular point in the program.
- b. The syntax is as follows :

```
do  
{  
    action;  
}
```

```
while (condition is true);  
action2;
```

**4. The while statement :**

- a. This is an entry-controlled loop structure.
- b. The syntax is as follows :

```
while (condition is true)  
{  
    action1;  
}  
action2;
```

**5. The for statement :**

- a. The for is an entry-controlled loop and is used when an action is to be repeated for a predetermined number of times.
- b. The syntax is as follows :

```
for (initial value; test; increment)  
{  
    action1;  
}  
action2;
```

**Que 4.8.**

**Write a program in C++ to count display the frequency of vowels in a given sentence of at least 35 characters long.**

**Answer**

```
#include <iostream>  
using namespace std;  
int main()  
{  
    char line[35];  
    int vowels;  
    vowels = 0;  
    cout<< "Enter a line of string: ";  
    cin.getline(line, 35);  
    for(int i = 0; line[i]!='\0'; ++i)  
{  
        if(line[i]=='a' || line[i]=='e' || line[i]=='i' ||  
           line[i]=='o' || line[i]=='u' || line[i]=='A' ||  
           line[i]=='E' || line[i]=='T' || line[i]=='O' ||  
           line[i]=='U')
```

```
{
    ++vowels;
}

cout<< "Vowels: " <<vowels<<endl;
return 0;
}
```

**Que 4.9.** Differentiate between identifiers and keywords.

**Answer**

| S. No. | Identifiers                                                                             | Keywords                                                                |
|--------|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| 1.     | Identifiers are the names defined by the programmer to the basic elements of a program. | Keywords are the reserved words whose meaning is known by the compiler. |
| 2.     | It is used to identify the name of the variable.                                        | It is used to specify the type of entity.                               |
| 3.     | It can consist of letters, digits, and underscore.                                      | It contains only letters.                                               |
| 4.     | It can use both lowercase and uppercase letters.                                        | It uses only lowercase letters.                                         |
| 5.     | No special character can be used except the underscore.                                 | It cannot contain any special character.                                |
| 6.     | The starting letter of identifiers can be lowercase, uppercase or underscore.           | It can be started only with the lowercase letter.                       |
| 7.     | It can be classified as internal and external identifiers.                              | It cannot be further classified.                                        |
| 8.     | Examples are test, result, sum, power, etc.                                             | Examples are 'for', 'if', 'else', 'break', etc.                         |

## PART-2

*C++ Function, Simple Function, Call and Return by Reference, Inline Function Macro vs Inline Functions, Overloading of Functions, Default Arguments, Friend Functions, Virtual Function.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 4.10.** Explain the prototype of main() function.

**Answer**

1. The main() returns a value of type int to the operating system.
2. C++ explicitly defines main() as matching one of the following prototypes :  

```
int main();  
int main(int argc, char * argv[]);
```
3. The functions that have a return value should use the return statement for termination.
4. Hence the main() function in C++ is defined as follows :  

```
int main()  
{  
    .....  
    .....  
    return 0;  
}
```
5. Since the return type of functions is int by default, the keyword int in the main() header is optional.
6. Most C++ compilers will generate an error or warning if there is no return statement.

**Que 4.11.** Describe call by reference and return by reference with example.

**Answer**

1. **Call by reference :**
  - a. Provision of the reference variables in C++ permits us to pass parameters to the functions by reference.
  - b. When we pass arguments by reference, the 'formal' arguments in the called function become aliases to the 'actual' arguments in the calling function.

- c. This means that when the function is working with its own arguments, it is actually working on the original data.
- d. For example, consider the following function :

```
void swap(int &a, int &b)    // a and b are reference variables
{
    int t = a;              // Dynamic initialization
    a = b;
    b = t;
}
```

Now, if m and n are two integer variables, then the function call  
swap(m, n);

will exchange the values of m and n using their aliases (reference variables) a and b.

## 2. Return by reference :

- a. A function can also return a reference.
- b. For example, consider the following function :

```
int & max(int &x, int &y)
{
    if (x > y)
        return x;
    else
        return y;
}
```

- c. Since the return type of max() is int &, the function returns reference to x or y (and not the values).
- d. Then a function call such as max(a, b) will yield a reference to either a or b depending on their values.
- e. This means that this function call can appear on the left-hand side of an assignment statement.
- f. This means the statement  
max(a, b) = -1;  
is legal and assigns -1 to a if it is larger, otherwise -1 to b.

**Que 4.12. Differentiate between call by value and call by reference.**

**Answer**

| S. No. | Call by value                                                                                                                                           | Call by reference                                                                                                                                        |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.     | A copy of the value is passed into the function.                                                                                                        | An address of value is passed into the function.                                                                                                         |
| 2.     | Changes made inside the function are limited to the function only. The values of the actual parameters do not change by changing the formal parameters. | Changes made inside the function validate outside of the function also. The values of the actual parameters do change by changing the formal parameters. |
| 3.     | Actual and formal arguments are created at the different memory location.                                                                               | Actual and formal arguments are created at the same memory location.                                                                                     |

**Que 4.13. Explain inline function.****Answer**

1. Inline function is a function that is expanded in line when it is called.
2. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.
3. This substitution is performed by the C++ compiler at compile time. Inline function may increase efficiency if it is small.
4. The syntax for defining the function inline is :  
inline return-type function-name(parameters)  
{  
    // function code  
}

**Que 4.14. Write a program in C++ to calculate the sum of the digits of a given positive integer number. For example, if the given number is 12345 then the program should display : 15.****Answer**

```
#include<iostream>
using namespace std;
int main()
{
    int val, num, sum = 0;
```



```
cout << "Enter the number : ";
cin >> val;
num = val;
while (num != 0)
{
    sum = sum + num % 10;
    num = num / 10;
}
cout << "The sum of the digits of "
    << val << " is " << sum;
}
```

**Que 4.15.** Write a program to perform addition, subtraction, multiplication and division using inline function.

**Answer**

```
#include<iostream>
using namespace std;
class operation
{
    int a, b, add, sub, mul;
    float div;
public :
    void get();
    void sum();
    void difference();
    void product();
    void division();
};
inline void operation :: get()
{
    cout << "Enter first value:";
    cin >> a;
    cout << "Enter second value:";
    cin >> b;
}
inline void operation :: sum()
```

```
{
    add = a + b;
    cout << "Addition of two numbers : " << a + b << "\n";
}

inline void operation :: difference()
{
    sub = a - b;
    cout << "Difference of two numbers : " << a - b << "\n";
}

inline void operation :: product()
{
    mul = a * b;
    cout << "Product of two numbers: " << a * b << "\n";
}

inline void operation :: division()
{
    div = a / b;
    cout << "Division of two numbers: " << a / b << "\n";
}

int main()
{
    cout << "Program using inline function\n";
    operation s;
    s.get();
    s.sum();
    s.difference();
    s.product();
    s.division();
    return 0;
}
```

**Que 4.16.** What are the advantages and disadvantages of inline function ?

**Answer**

**A. Advantages :**

1. Function call overhead doesn't occur.

2. It also saves the overhead of push/pop variables on the stack when function is called.
3. It also saves overhead of a return call from a function.
4. When you inline a function, you may enable compiler to perform context specific optimization on the body of function. Such optimizations are not possible for normal function calls. Other optimizations can be obtained by considering the flows of calling context and the called context.
5. Inline function may be useful (if it is small) for embedded systems because inline can yield less code than the function call preamble and return.

**B. Disadvantages :**

1. The added variables from the inlined function consume additional registers.
2. If you use too many inline functions then the size of the binary executable file will be large, because of the duplication of same code.
3. Too much inlining can also reduce your instruction cache hit rate, thus reducing the speed of instruction fetch from that of cache memory to that of primary memory.
4. Inline function may increase compile time overhead if someone changes the code inside the inline function then all the calling location has to be recompiled because compiler would require to replace all the code once again to reflect the changes, otherwise it will continue with old functionality.
5. Inline functions may not be useful for many embedded systems. Because in embedded systems code size is more important than speed.

**Que 4.17. Differentiate between inline and macro function.**

**Answer**

| S. No. | Inline                                                              | Macro                                                                                                       |
|--------|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| 1.     | An inline function is defined by the inline keyword.                | Whereas the macros are defined by the are define keyword.                                                   |
| 2.     | Through inline function, the class's data members can be accessed.  | Whereas macro can't access the class's data members.                                                        |
| 3.     | In the case of inline function, the program can be easily debugged. | Whereas in the case of macros, the program can't be easily debugged.                                        |
| 4.     | In the case of inline, the arguments are evaluated only once.       | Whereas in the case of macro, the arguments are evaluated every time whenever macro is used in the program. |

|    |                                                                                                   |                                                                                  |
|----|---------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|
| 5. | In C++, inline may be defined either inside the class or outside the class.                       | Whereas the macro is all the time defined at the beginning of the program.       |
| 6. | In C++, inside the class, the short length functions are automatically made the inline functions. | While the macro is specifically defined.                                         |
| 7. | Inline is not as widely used as macros.                                                           | While the macro is widely used.                                                  |
| 8. | Inline is not used in competitive programming.                                                    | While the macro is very much used in competitive programming.                    |
| 9. | Inline function is terminated by the curly brace at the end.                                      | While the macro is not terminated by any symbol, it is terminated by a new line. |

**Que 4.18.** What do you mean by overloading of a function ?

**Answer**

1. C++ allows you to specify more than one definition for a function name or an operator in the same scope, which is called function overloading.
2. An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and different definition (implementation).
3. When you call an overloaded function or operator, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions.
4. The process of selecting the most appropriate overloaded function or operator is called overload resolution.

**Que 4.19.** Write a program of function overloading when the number of arguments are different.

**Answer**

```
#include<iostream>
using namespace std;
class Cal {
    public:
```

```
static int add(int a,int b){
    return a + b;
}

static int add(int a, int b, int c)
{
    return a + b + c;
}

};

int main(void) {
    Cal C;                                // class object declaration.
    cout << C.add(10, 20) << endl;
    cout << C.add(12, 20, 23);
    return 0;
}
```

**Que 4.20.** Write a program to print the length of a rectangle using friends function.

**Answer**

```
#include<iostream>
using namespace std;
class Box
{
    private:
    int length;
    public:
    Box(): length(0) { }
    friend int printLength(Box); //friend function
};

int printLength(Box b)
{
    b.length += 10;
    return b.length;
}

int main()
{
    Box b;
```

```
cout<<"Length of box:"<< printLength(b)<<endl;
return 0;
```

```
}
```

**Que 4.21.** Write a program in C++ to display the longest word in a given sentence.

**Answer**

```
#include<iostream>
using namespace std;
//function to find longest word
int word_length(string str) {
    int len = str.length();
    int temp = 0;
    int newlen = 0;
    for (int i = 0; i < len; i++) {
        if (str[i] != ' ')
            newlen++;
        else {
            temp = max(temp, newlen);
            newlen = 0;
        }
    }
    return max(temp, newlen);
}

int main() {
    string str = "Quantum Series is the best for clearing exam";
    cout << "maximum length of a word is : " << word_length(str);
    return 0;
}
```

**Que 4.22.** Explain default argument.

**Answer**

A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the caller of the function doesn't provide a value for the argument with a default value.

```
#include<iostream>
```

```
using namespace std;
// A function with default arguments, it can be called with
// 2 arguments or 3 arguments or 4 arguments.
int sum(int x, int y, int z=0, int w=0)
{
    return (x + y + z + w);
}
```

**Que 4.23. Explain friend function with example.**

**Answer**

1. If a function is defined as a friend function in C++, then the protected and private data of a class can be accessed using the function.
2. By using the keyword friend compiler knows the given function is a friend function.
3. For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend.
4. For example,

```
#include <iostream>
using namespace std;
class Box
{
    private:
        int length;
    public:
        Box(): length(0) { }
        friend int printLength(Box); //friend function
};
```

**Que 4.24. What are the characteristics of friend function ?**

**Answer**

Characteristics of friend function are :

1. The function is not in the scope of the class to which it has been declared as a friend.
2. It cannot be called using the object as it is not in the scope of that class.
3. It can be invoked like a normal function without using the object.
4. It cannot access the member names directly and has to use an object name and dot membership operator with the member name.

5. It can be declared either in the private or the public part.

**Que 4.25. Describe virtual function.**

**Answer**

1. A virtual function is a member function which is declared within a base class and is redefined (Overridden) by a derived class.
2. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.
3. Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
4. They are mainly used to achieve Runtime polymorphism
5. Functions are declared with a virtual keyword in base class.
6. The resolving of function call is done at Run-time.

**Que 4.26. What are the rules used for a virtual function ?**

**Answer**

1. Virtual functions cannot be static and also cannot be a friend function of another class.
2. Virtual functions should be accessed using pointer or reference of base class type to achieve run time polymorphism.
3. The prototype of virtual functions should be same in base as well as derived class.
4. They are always defined in base class and overridden in derived class. It is not mandatory for derived class to override (or re-define the virtual function), in that case base class version of function is used.
5. A class may have virtual destructor but it cannot have a virtual constructor.





# 5

## UNIT

# Object and Classes

## CONTENTS

- Part-1** : Objects and Classes, ..... 5-2L to 5-6L  
Basics of Object and Class  
in C++, Private and  
Public Members, Static  
Data and Function Members
- Part-2** : Constructors and their ..... 5-6L to 5-10L  
Types, Destructors, Operators  
Overloading, Type Conversion
- Part-3** : Inheritance : Concept of ..... 5-10L to 5-15L  
Inheritance, Types of  
Inheritance : Single,  
Multiple, Multilevel,  
Hierarchical, Hybrid,  
Protected Members,  
Overriding, Virtual Base Class
- Part-4** : Polymorphism, Pointers in ..... 5-15L to 5-19L  
C++, Pointers and Objects,  
this Pointer, Virtual and Pure  
Virtual Function, Implementing  
Polymorphism

**PART- 1**

*Objects and Classes, Basics of Object and Class in C++, Private and Public Members, Static Data and Function Members.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 5.1.** Write short note on object and classes.

**Answer****A. Object :**

1. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.
2. When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

**B. Classes :**

1. A class is a way to bind the data and its associated functions together. It allows the data (and functions) to be hidden, if necessary, from external use.
2. When defining a class, we are creating a new abstract data type that can be treated like any other built-in data type.
3. The class declaration describes the type and scope of its members.
4. The class function definitions describe how the class functions are implemented.
5. The general form of a class declaration is :

```
class class_name
{
    private :
        variable declarations;
        function declarations;
    public :
        variable declarations;
        function declarations;
};
```

6. The class declaration is similar to a struct declaration. The keyword `class` specifies that what follows is an abstract data of type `class_name`.
7. The body of a class is enclosed within braces and terminated by a semicolon.
8. The class body contains the declaration of variables and functions. These functions and variables are collectively called class members.
9. The class members that have been declared as `private` can be accessed only from within the class.
10. While the public members can be accessed from outside the class also.
11. The variables declared inside the class are known as data members and the functions are known as member functions.

**Que 5.2.** Design a class using C++ to create a singly linked list.

**Answer**

```
#include<iostream>
using namespace std;
struct node
{
    int data;
    node *next;
};
class list
{
    private:
    node *head, *tail;
    public:
    list()
    {
        head = NULL;
        tail = NULL;
    }
    void createnode(int value)
    {
        node *temp = new node;
        temp->data = value;
        temp->next = NULL;
        if(head == NULL)
```

```
        {
            head = temp;
            tail = temp;
            temp = NULL;
        }
    else
    {
        tail -> next = temp;
        tail = temp;
    }
}

};

int main()
{
    list obj;
    obj.createnode(25);
    obj.createnode(50);
    obj.createnode(90);
    obj.createnode(40);
    cout<<"\n-----\n";
    cout<<"-----Displaying All nodes-----";
    cout<<"\n-----\n";
    system("pause");
    return 0;
}
```

**Que 5.3.** Explain private member function.

**Answer**

1. A private member function can only be called by another function that is a member of its class.
2. Even an object cannot invoke a private function using the dot operator.
3. Consider a class as defined below :

```
class sample
```

```
{
```

```
    in m;
```

```
    void read(void);    //    private member function
```

```
public :  
    void update(void);  
    void write(void);  
};
```

4. If s1 is an object of sample, then

```
s1.read();          // won't work; objects cannot access  
                   // private members
```

is illegal.

5. However, the function **read()** can be called by the function **update()** to update the value of **m**.

```
void sample :: update(void)  
{  
    read();          // simple call; no object used  
}
```

**Que 5.4.** Differentiate between public and private member function.

**Answer**

| S. No. | Public                                                                                                                                                | Private                                                                                                      |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| 1.     | All the class members declared under public will be available to everyone.                                                                            | The class members declared as private can be accessed only by the functions inside the class.                |
| 2.     | The data members and member functions declared public can be accessed by other classes too.                                                           | Only the member functions or the friend functions are allowed to access the private data members of a class. |
| 3.     | The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class. | They are not allowed to be accessed directly by any object or function outside the class.                    |

**Que 5.5.** Explain static data and static function member.

**Answer**

**Static data :**

1. Static data is data that does not change after being recorded. It is a fixed data set.

- Experts contrast static data with dynamic data, where dynamic data may change after it is recorded, and has to be continually updated.

**Member function :**

- A member function of a class is a function that has its definition or its prototype within the class definition like any other variable.
- It operates on any object of the class of which it is a member, and has access to all the members of a class for that object.
- Member functions can be defined within the class definition or separately using scope resolution operator.
- Defining a member function within the class definition declares the function inline, even if you do not use the inline specifier.

**PART-2**

*Constructors and their Types, Destructors, Operators  
Overloading, Type Conversion.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 5.6.** Define constructor. How constructor is different from normal member function.

**Answer****Constructor :**

- A constructor is a member function of a class which initializes objects of a class.
- In C++, Constructor is automatically called when object (instance of class) create. It is special member function of the class.

A constructor is different from normal functions in following ways :

- Constructor has same name as the class itself.
- Constructors don't have return type.
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

**Que 5.7.** What are the types of constructor ?

**Answer**

Different types of constructor are :

**1. Default constructors :**

- Default constructor is the constructor which doesn't take any argument. It has no parameters.
- Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly.

**2. Parameterized constructors :**

- It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.
- To create a parameterized constructor, simply add parameters to it the way you would to any other function.
- When you define the constructor's body, use the parameters to initialize the object.

**3. Copy constructor :**

- A copy constructor is a member function which initializes an object using another object of the same class.
- Whenever we define one or more non-default constructors (with parameters) for a class, a default constructor (without parameters) should also be explicitly defined as the compiler will not provide a default constructor in this case.
- However, it is not necessary but it's considered to be the best practice to always define a default constructor.

**Que 5.8. Define destructor with its properties.**

**Answer****Destructor :**

- Destructor is a member function which destructs or deletes an object.
- Following is the Syntax of destructor :

~constructor-name();

**Properties of Destructor :**

- Destructor function is automatically invoked when the objects are destroyed.
- It cannot be declared static or const.
- The destructor does not have arguments.
- It has no return type not even void.
- An object of a class with a Destructor cannot become a member of the union.

6. A destructor should be declared in the public section of the class.
7. The programmer cannot access the address of destructor.

**Que 5.9. When we call a destructor ? How it is different from normal member function. Can there be more than one destructor in a class ?**

**Answer**

**We call a destructor when :**

1. A destructor function is called automatically when the object goes out of scope :
  - a. The function ends
  - b. The program ends
  - c. A block containing local variables ends
  - d. A delete operator is called
2. Destructors are different from a normal member function :
  - a. Destructors have same name as the class preceded by a tilde (~).
  - b. Destructors don't take any argument and don't return anything.

No, there can only one destructor in a class with classname preceded by ~, no parameters and no return type.

**Que 5.10. Describe operator overloading.**

**Answer**

1. Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type.
2. Operator overloading is used to overload or redefines most of the operators available in C++.
3. It is used to perform the operation on the user-defined data type.
4. For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.
5. The advantage of Operators overloading is to perform different operations on the same operand.
6. Syntax of Operator Overloading

```
return_type class_name :: operator op(argument_list)
{
    // body of the function.
}
```



**Que 5.11. What are the rules of operator overloading ?**

**Answer**

**Rules for Operator Overloading :**

1. Existing operators can only be overloaded, but the new operators cannot be overloaded.
2. The overloaded operator contains at least one operand of the user-defined data type.
3. We cannot use friend function to overload certain operators. However, the member function can be used to overload those operators.
4. When unary operators are overloaded through a member function take no explicit arguments, but, if they are overloaded by a friend function, takes one argument.
5. When binary operators are overloaded through a member function takes one explicit argument, and if they are overloaded through a friend function takes two explicit arguments.

**Que 5.12. Explain type conversion with its type.**

**Answer**

A type cast is basically a conversion from one type to another. There are two types of type conversion :

1. **Implicit Type Conversion (automatic type conversion) :**
  - a. Done by the compiler on its own, without any external trigger from the user.
  - b. Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid lose of data.
  - c. All the data types of the variables are upgraded to the data type of the variable with largest data type.
  - d. It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long is implicitly converted to float).
2. **Explicit Type Conversion :**
  - a. This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.
  - b. In C++, it can be done by two ways :
    - i. Converting by assignment :
      - a. This is done by explicitly defining the required type in front of the expression in parenthesis. This can be also considered as forceful casting.

- b. Following is the syntax :
- (type) expression
- where type indicates the data type to which the final result is converted.
- ii. Conversion using Cast operator :
- Inheritance : concept of inheritance, types of inheritance: single, multiple, multilevel, hierarchical , hybrid, protected members, overriding, virtual base class
- a. A cast operator is a unary operator which forces one data type to be converted into another data type.
- b. C++ supports four types of casting :
- Static Cast
  - Dynamic Cast
  - Const Cast
  - Reinterpret Cast

**PART-3**

*Inheritance : Concept of Inheritance , Types of Inheritance : Single, Multiple, Multilevel, Hierarchical , Hybrid, Protected Members, Overriding, Virtual Base Class.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 5.13.** Define inheritance. What are the modes of inheritance ?

**Answer****Inheritance :**

1. The capability of a class to derive properties and characteristics from another class is called Inheritance.
2. Inheritance is one of the most important features of Object Oriented Programming.
3. The class that inherits properties from another class is called Sub class or Derived class.

4. The class whose properties are inherited by sub class is called Base Class or Super class.

### **Modes of inheritance :**

#### **1. Public mode :**

- If we derive a sub class from a public base class.
- Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.

#### **2. Protected mode :**

- If we derive a sub class from a Protected base class.
- Then both public member and protected members of the base class will become protected in derived class.

#### **3. Private mode :**

- If we derive a sub class from a Private base class.
- Then both public member and protected members of the base class will become Private in derived class.

**Que 5.14.** What are the types of inheritance ? Explain.

**Answer**

### **Different types of inheritance :**

#### **1. Single inheritance :**

- In single inheritance, a class is allowed to inherit from only one class. *i.e.*, one sub class is inherited by one base class only.
- Following is the syntax :

```
class subclass_name : access_mode base_class
{
    //body of subclass
};
```

#### **2. Multiple inheritance :**

- Multiple inheritance is a feature of C++ where a class can inherit from more than one classes. *i.e* one sub class is inherited from more than one base classes.
- Following is the syntax :

```
class subclass_name : access_mode base_class1, access_mode
base_class2, ....
{
    //body of subclass
};
```

3. **Multilevel inheritance :** In this type of inheritance, a derived class is created from another derived class.
4. **Hierarchical inheritance :** In this type of inheritance, more than one sub class is inherited from a single base class, i.e., more than one derived class is created from a single base class.
5. **Hybrid (Virtual) inheritance :**
  - a. Hybrid inheritance is implemented by combining more than one type of inheritance.
  - b. For example : Combining Hierarchical inheritance and Multiple Inheritance.

**Que 5.15. What are protected member ? What is difference between public and protected access modifier.**

**Answer**

1. Protected access modifier is similar to private access modifier in the sense that it can't be accessed outside of it's class unless with the help of friend class, the difference is that the class members declared as Protected can be accessed by any subclass (derived class) of that class as well.

| S. No. | Public                                                                                      | Protected                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.     | All the class members declared under public will be available to everyone.                  | Protected access modifier is similar to that of private access modifiers.                                                                         |
| 2.     | The data members and member functions declared public can be accessed by other classes too. | The class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass (derived class) of that class. |

**Que 5.16. Explain function overriding.**

**Answer**

1. If derived class defines same function as defined in its base class, it is known as function overriding in C++.
2. It is used to achieve runtime polymorphism.
3. It enables you to provide specific implementation of the function which is already provided by its base class.
4. For example, we are overriding the eat() function.  

```
#include <iostream>
using namespace std;
```

```
class Animal {
    public:
    void eat(){
        cout << "Eating...";
    }
};

class Dog: public Animal
{
    public:
    void eat()
    {
        cout << "Eating bread...";
    }
};

int main(void) {
    Dog d = Dog();
    d.eat();
    return 0;
}
```

**Que 5.17.** Write a program in C++ to read in two matrices from the keyboard and compute their sum.

**Answer**

```
#include<iostream>
using namespace std;
int main()
{
    int row, col, m1[10][10], m2[10][10], sum[10][10];
    cout << "Enter the number of rows(should be >1 and <10): ";
    cin >> row;
    cout << "Enter the number of column(should be >1 and <10): ";
    cin >> col;
    cout << "Enter the elements of first 1st matrix: ";
    for (int i = 0; i < row; i++ ) {
        for (int j = 0; j < col; j++ ) {
            cin >> m1[i][j];
```

```
}  
}  
  
cout << "Enter the elements of first 1st matrix: ";  
for (int i = 0; i < row; i++ ) {  
    for (int j = 0; j < col; j++ ) {  
        cin >> m2[i][j];  
    }  
}  
  
cout << "Output: ";  
for (int i = 0; i < row; i++ ) {  
    for (int j = 0; j < col; j++ ) {  
        sum[i][j] = m1[i][j] + m2[i][j];  
        cout << sum[i][j] << " ";  
    }  
}  
  
return 0;  
}
```

**Que 5.18.** What is virtual base class ?

**Answer**

1. Virtual base classes are used in virtual inheritance in a way of preventing multiple “instances” of a given class appearing in an inheritance hierarchy when using multiple inheritances.
2. Following is the Syntax for Virtual Base Classes :

Syntax 1:

```
class B : virtual public A  
{  
};
```

Syntax 2 :

```
class C : public virtual A  
{  
};
```

3. For example,  

```
#include <iostream>  
using namespace std;  
class A {
```

```
public:
    void show()
    {
        cout << "Hello from A \n";
    }
};
class B : public A {
};
class C : public A {
};
class D : public B, public C {
};
int main()
{
    D object;
    object.show();
}
```

**PART-4**

*Polymorphism, Pointers in C++, Pointers and Objects, this Pointer, Virtual and Pure Virtual Function, Implementing Polymorphism.*

**Questions-Answers****Long Answer Type and Medium Answer Type Questions**

**Que 5.19.** Explain the term polymorphism with its types.

**Answer**

**Polymorphism :** Refer Q. 1.3, Page 1-3L, Unit-1.

**Different types of polymorphism :**

1. **Compile time polymorphism :** This type of polymorphism is achieved by function overloading or operator overloading.

**a. Function overloading :**

- i. When there are multiple functions with same name but different parameters then these functions are said to be overloaded.
- ii. Functions can be overloaded by change in number of arguments or/and change in type of arguments.

**b. Operator overloading :** C++ also provides option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings.

**2. Runtime polymorphism :** This type of polymorphism is achieved by function overriding :

**a. Function overriding :**

- i. It occurs when a derived class has a definition for one of the member functions of the base class.
- ii. That base function is said to be overridden.

**Que 5.20. Write a program to swap two numbers using pointers.**

**Answer**

```
#include<iostream.h>
#include<conio.h>
void main() {
    clrscr();
    int *a,*b,*temp;
    cout << "Enter value of a and b :";
    cin >> *a >> *b;
    temp = a;
    a = b;
    b = temp;
    cout << "After swaping na =" << *a << "nb =" << *b;
    getch();
}
```

**Que 5.21. What is a pointer ? How to use a pointer.**

**Answer**

**Pointer :**

1. Pointers are symbolic representation of addresses.
2. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures.



3. Its general declaration in C++ has the format :

`datatype *var_name;`

`int *ptr; //ptr can point to an address`

**Use a pointer :**

1. Define a pointer variable
2. Assigning the address of a variable to a pointer using unary operator (&) which returns the address of that variable.
3. Accessing the value stored in the address using unary operator (\*) which returns the value of the variable located at the address specified by its operand.

**Que 5.22. Describe briefly about the term object.**

**Answer**

1. Objects are the basic run-time entities in an object-oriented system.
2. In the object-oriented programming paradigm object can be a combination of variables, functions, and data structures.
3. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.
4. Programming problem is analyzed in terms of objects and the nature of communication between them.
5. Program objects should be chosen such that they match closely with the real-world objects.
6. Objects can interact without having to know details of each other's data or code.

**Que 5.23. Explain 'this pointer' with the help of example.**

**Answer**

1. In C++ programming, this is a keyword that refers to the current instance of the class.
2. There can be three main usage of this keyword in C++ :
  - a. It can be used to pass current object as a parameter to another method.
  - b. It can be used to refer current class instance variable.
  - c. It can be used to declare indexers.
3. For example,

```
#include <iostream>
using namespace std;
class Employee {
```

```
public:
    int id; //data member (also instance variable)
    string name; //data member(also instance variable)
    float salary;
    Employee(int id, string name, float salary)
    {
        this->id = id;
        this->name = name;
        this->salary = salary;
    }
    void display()
    {
        cout << id << " " << name << " " << salary << endl;
    }
};

int main(void) {
    Employee e1 = Employee(101, "Sonu", 890000); //creating an object
    of Employee
    Employee e2 = Employee(102, "Nakul", 59000); //creating an object
    of Employee
    e1.display();
    e2.display();
    return 0;
}
```

**Que 5.24.** Differentiate between virtual and pure virtual function.

**Answer**

| S. No | Virtual function                                                                               | Pure virtual function                                                                                                                                                                             |
|-------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | A virtual function is a member function of base class which can be redefined by derived class. | A pure virtual function is a member function of base class whose only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract. |
| 2.    | Classes having virtual functions are not abstract.                                             | Base class containing pure virtual function becomes abstract.                                                                                                                                     |

|    |                                                                                                         |                                                                                                                                                                |
|----|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3. | Syntax :<br>virtual<func_type><br><func_name>()<br>{<br>// code<br>}                                    | Syntax :<br>virtual<func_type><func_name>()<br>= 0;                                                                                                            |
| 4. | Definition is given in base class.                                                                      | No definition is given in base class.                                                                                                                          |
| 5. | Base class having virtual function can be instantiated i.e., its object can be made.                    | Base class having pure virtual function becomes abstract i.e. it cannot be instantiated.                                                                       |
| 6. | If derived class does not redefine virtual function of base class, then it does not affect compilation. | If derived class does not redefine virtual function of base class, then no compilation error but derived class also becomes abstract just like the base class. |
| 7. | All derived class may or may not redefine virtual function of base class.                               | All derived class must redefine pure virtual function of base class otherwise derived class also becomes abstract just like base class.                        |





## Introduction (2 Marks Questions)

### 1.1. What are the benefits of object-oriented approaches ?

**Ans.** Following are the benefits of object-oriented approach :

- It facilitates changes in the system at low cost.
- It promotes the reuse of components.
- It simplifies the problem of integrating components to configure large system.
- It simplifies the design of distributed systems.

### 1.2. What are the elements of object-oriented system ?

**Ans.** Following are the elements of object-oriented system:

- Object
- Attributes
- Behavior
- Class
- Methods
- Message

### 1.3. What are the features of object-oriented system ?

**Ans.** Following are the features of object-oriented system:

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

### 1.4. Define the term encapsulation and abstraction.

**Ans. Encapsulation :** Encapsulation is a process of information hiding. It allows improvement or modification of methods used by objects without affecting other parts of a system.

**Abstraction :** It is a process of taking or selecting necessary method and attributes to specify the object.

### 1.5. What do you mean by inheritance and polymorphism ?

**Ans.** **Inheritance :** Inheritance is a feature that allows to create sub-classes from an existing class by inheriting the attributes and/or operations of existing classes. **Polymorphism :** Polymorphism is the ability to take on many different forms. It applies to both objects and operations.

**1.6. Define object-oriented technology.**

**Ans.** Object-Oriented Technology (OOT) is an approach to program organization and development that attempts to reduce issues with conventional programming techniques.

**1.7. What are the advantages of object-oriented technology ?**

**Ans.** Advantages of object-oriented technology are :

1. It allows parallel development.
2. The modular classes are often reusable .
3. The coding is easier to maintain .

**1.8. What are the disadvantages of object-oriented technology ?**

**Ans.** Disadvantages of object-oriented technology are :

1. It is inefficient.
2. It is scalable.
3. It causes duplication.

**1.9. Define object identity.**

**Ans.** Object identity is a property of data that is created in the context of an object data model, where an object is assigned a unique internal object identifier, or object ID.

**1.10. What do you mean by information hiding ?**

**Ans.** Information hiding is the process of hiding the details of an object or function. The hiding of these details results in an abstraction, which reduces the external complexity and makes the object or function easier to use.

**1.11. Which technique is used to implement information hiding.**

**Ans.** Encapsulation is a technique programmers use to implement information hiding.

**1.12. What are the models of object-oriented languages ?**

**Ans.** Following are the models of object-oriented languages:

1. Object model
2. Dynamic model
3. Functional model

**1.13. Define the term actor.**

**Ans.** An actor is an active object that drives the data flow graph by producing or consuming values. Actors are attached to the inputs and outputs of a data flow graph..

**1.14. What is object-oriented modeling?**

**Ans.** Object-oriented modeling (OOM) is the construction of objects using a collection of objects that contain stored values of the instance variables found within an object.

**1.15. What are the steps involved in object-oriented modeling?**

**Ans.** Following are the steps involved in object-oriented modeling:

1. System analysis
2. System design
3. Object design
4. Final implementation

**1.16. Define the term link and association.**

**Ans.** Link : Link defines the relationship between two or more objects and a link is considered as an instance of an association.  
Association : It is a group of links that relates objects from the same classes.

**1.17. What do you mean by UML ?**

**OR**

**What is unified markup language ?**

**AKTU 2015-16, Marks 02**

**Ans.** UML stands for Unified Modeling Language. It is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.



# 2

## UNIT

# Basic Structural Modeling (2 Marks Questions)

**2.1. What are different types of structural modeling in object oriented system design?**

**Ans.** Following are the types of structural modeling in object oriented system design :

1. Structural modeling
2. Behavioural modeling
3. Architectural modeling

**2.2. Define structural modeling.**

**Ans.** Structural modeling captures the static features of a system. Structural model represents the framework for the system and this framework is the place where all other components exist.

**2.3. Define Behavioral Model.**

**Ans.** Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system.

**2.4. What does an architectural model represent ?**

**Ans.** Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system.

**2.5. Name the diagrams used in architectural modeling.**

**Ans.** Diagrams that give descriptions of the physical information about a system in architectural modeling are deployment diagrams and component diagrams.

**2.6. Define deployment diagrams and component diagram.**

**Ans.** Deployment diagrams : Deployment diagrams show the physical relationship between hardware and software in a system.  
Component diagrams : Component diagrams show the software components of a system and their relationships.

**2.7. What is a component ?**

**Ans.** A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.

## **2.8. Define the term interface.**

**Ans.** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle.

## **2.9. What are the different types of notation used in UML.**

**Ans.** Following are the different types of notation used in UML:

1. Dependency Notation
2. Association notation
3. Generalization notation
4. Extensibility notation

## **2.10. Define association and generalization notation.**

**Ans.** Association Notation : Association describes how the elements in a UML diagram are associated. it describes how many elements are taking part in an interaction.

Generalization notation : Generalization describes the inheritance relationship of the object-oriented world. It is a parent and child relationship.

## **2.10. What are the mechanism that provide extensibility feature ?**

**Ans.** Following mechanisms to provide extensibility features : i. Stereotypes (Represents new elements) ii. Tagged values (Represents new attributes) iii. Constraints (Represents the boundaries).

## **2.11. Which notation is used to enhance the power of a language.**

**Ans.** Extensibility notation is used to enhance the power of a language.

## **2.12. Define class diagram.**

**Ans.** Class diagram is a static diagram that represents the static view of an application. Class diagram is used for visualizing, describing, and documenting different aspects of a system and also for constructing executable code of the software application.

## **2.13. Define object diagram.**

**Ans.** Object diagrams represent an instance of a class diagram. Object diagrams represent the static view of a system but this static view is a snapshot of the system at a particular moment.

## **2.14. What is a collaboration diagram ?**

**Ans.** A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).



**2.15. What are the various terms used in collaboration diagram ?**

**Ans.** Various terms used in collaboration diagram are:

1. Object
2. Actor
3. Link
4. Message

**2.16. Define sequence diagram .**

**Ans.** Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. It visualizes and validates various runtime scenarios.

**2.17. What are package diagrams ?**

**Ans.** Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages.

**2.18. Define activity diagram.**

**Ans.** An activity diagram is a flowchart that shows activities performed by a system. The two special states shown in an activity diagram are the initial state.

**2.19. What is activity diagram ? Explain with suitable example.**

**AKTU 2015-16, Marks 02**

**Ans.**

1. An activity diagram is a flowchart that shows activities performed by a system.
2. The two special states shown in an activity diagram are the Initial State (Start Point) and Final State (End Point).
3. **Initial State or Start Point :** A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.



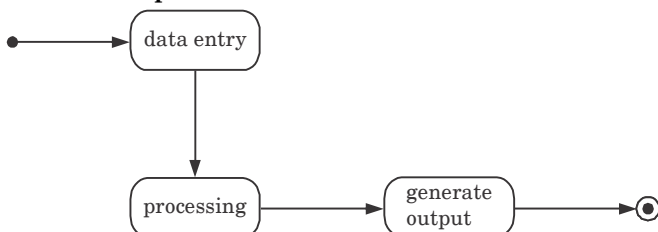
Start Point/Initial State

4. **Final State or End Point :** An arrow pointing to a filled circle nested inside another circle represents the final action state.



Ent Point Symbol

**For example :**



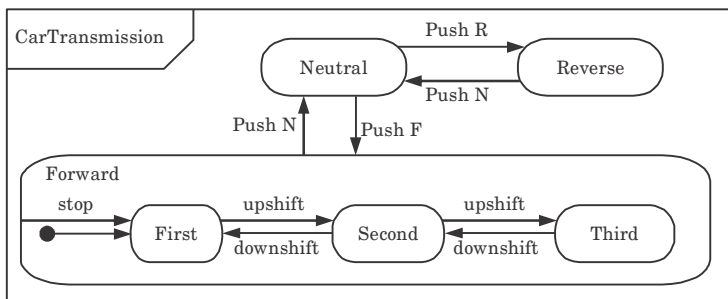
**Fig. 1.** Notation for activity states and activity-state transitions.

**2.20. Discuss the synchronization of concurrent activities.****AKTU 2015-16, Marks 02****Ans.**

1. Sometimes one object must perform two (or more) activities concurrently.
2. The object does not synchronize the internal steps of the activities but must complete both activities before it can progress to its next state.
3. For example, a cash dispensing machine dispenses cash and returns the user's card at the end of a transaction. The machine must not reset itself until the user takes both the cash and the card, but the user may take them in either order or even simultaneously.

**2.21. What is nested state diagram ? Explain with suitable example.****AKTU 2015-16, Marks 02****Ans.**

1. A nested state diagram is used to model the complex system as the regular state diagram is inadequate in describing the large and complex problem.
2. The nested state diagram is the concept of advanced state modeling.

**For example :****Fig. 2. Nested stage diagram of car transmission.**

1. There are three states inside CarTransmission state diagram *i.e.*, reverse, neutral and forward.
2. Among these three states, the forward state has three nested states *i.e.* First, Second and Third.
3. At any nested state of Forward gear composite state, selecting *N* would transit the corresponding state to the neutral state.
4. Being in a neutral state, selecting *F* would transit you to the forward state.

5. But here by default, the First nested state in the Forward contour is the initial state and the control is in the First state.
6. Here the Forward is just an abstract state.
7. The event Stop is shared by all the three nested state.

**2.22. What do you mean by candidate keys in object modeling ?**

**AKTU 2015-16, Marks 02**

**Ans.**

1. A super key with no redundant attribute is known as candidate key.
2. Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is that the candidate key should not have any redundant attributes.



# 3

## UNIT

# Object Oriented Analysis (2 Marks Questions)

### 3.1. What is an object-oriented analysis ?

**Ans.** Object-Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.

### 3.2. What are the primary task in object-oriented analysis ?

**Ans.** The primary tasks in object-oriented analysis (OOA) are :

- Identifying objects
- Organizing the objects by creating object model diagram.
- Defining the internals of the objects, or object attributes.
- Defining the behavior of the objects, *i.e.*, object actions.
- Describing how the objects interact.

### 3.3. What are the different phases of object-oriented design?

**Ans.** Object design includes the following phases :

- Object identification.
- Object representation, *i.e.*, construction of design models.
- Classification of operations.
- Algorithm design.
- Design of relationships.
- Implementation of control for external interactions.

### 3.4. What is design optimization ?

**Ans.** Design optimization is an engineering design methodology using a mathematical formulation of a design problem to support selection of the optimal design among many alternatives.

### 3.5. Differentiate between early and late binding with an example.

**Ans.**

| S. No. | Early binding                                                             | Late binding                                                          |
|--------|---------------------------------------------------------------------------|-----------------------------------------------------------------------|
| 1.     | It is a compile-time process.                                             | It is a run-time process.                                             |
| 2.     | The method definition and method call are linked during the compile time. | The method definition and method call are linked during the run time. |

**3.6. What are the stages used in design optimization ?****Ans.** Design optimization involves the following stages :

- Variables
- Objective
- Constraints
- Feasibility

**3.7. What is a documentation ?****Ans.** Documentation is a software development process that records the procedure of making the software. The design decisions need to be documented for any non-trivial software system for transmitting the design to others.**3.8. What is structured analysis and design phase ?****Ans.** Structured Analysis and Structured Design (SA/SD) is diagrammatic notation which is design to help people understand the system. The basic goal of SA/SD is to improve quality and reduce the risk of system failure.**3.9. What are the phases of SA/SD ?****Ans.** SA/SD involves two phases :

- Analysis Phase : It uses Data Flow Diagram, Data Dictionary, State Transition diagram and ER diagram.
- Design Phase : It uses Structure Chart and Pseudo Code.

**3.10. Define jackson system development.****Ans.** Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.**3.11. What are the phases of JSD ?****Ans.** Following are the phases of JSD :

- Modeling phase
- Specification phase
- Implementation phase

**3.12. What are the independent dimensions used for classifying inheritance ?**

**Ans.** There are three independent dimensions for classifying inheritance mechanisms :

1. Static or dynamic
2. Implicit or explicit
3. Per object or per group.

**3.13. Define formal and actual parameters.**

**Ans.** Formal parameter : A variable and its type as they appear in the prototype of the function or method.

Actual parameter : The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.



# 4

## UNIT

# C++ Basics & Functions (2 Marks Questions)

### 4.1. Define the term namespace and identifiers.

**Ans. Namespace :** Namespace defines a scope for the identifiers that are used in a program.

**Identifiers :** Identifiers refer to the names of variables, functions, arrays, classes, etc. created by the programmer.

### 4.2. What is a constants ?

**Ans.** Constants refer to fixed values that do not change during the execution of a program. They include integers, characters, floating point numbers and strings.

### 4.3. Define the term enumerated and variables.

**Ans. Enumerated :** An enum specifier defines the set of all names that will be permissible values of the type. These permissible values are called members.

**Variables :** A variable is a symbolic name that can be given a variety of values. Variables are stored in particular places in the computer's memory.

### 4.4. What are the different operators used in C++ ?

**Ans.** Different operators used in C++ are :

1. Memory management operator
2. Manipulators
3. Typecast operators

### 4.5. What do you mean by typecast?

**Ans.** Converting an expression of a given type into another type is known as type-casting.

### 4.6. What are the different types of typecast ?

**Ans.** Different types of typecast are :

1. Implicit typecast
2. Explicit typecast
3. Dynamic cast

**4.7. Define switch statement.**

**Ans.** This is a multiple-branching statement where, based on a condition, the control is transferred to one of the many possible points.

**4.8. Define do-while loop.**

**Ans.** The do-while is an exit-controlled loop. Based on a condition, the control is transferred back to a particular point in the program.

**4.9. Define while and for statement.**

**Ans. The while statement :** This is also a loop structure, but is an entry-controlled one.

**The for statement :** The for is an entry-entrolled loop and is used when an action is to be repeated for a predetermined number of times.

**4.10. What do you mean by keywords ?**

**Ans.** Keywords are the reserved words whose meaning is known by the compiler. It is used to specify the type of entity. It contains only letters. It uses only lowercase letters. It cannot contain any special character.

**4.11. What is an inline function ?**

**Ans.** Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.

**4.12. What is a default argument?**

**Ans.** A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the caller of the function doesn't provide a value for the argument with a default value.

**4.13. Define virtual function.**

**Ans.** A virtual function is a member function which is declared within a base class and is re-defined (Overriden) by a derived class.





# 5

## UNIT

# Object and Classes (2 Marks Questions)

---

### 5.1. Define the term object and classes.

**Ans.** Object: An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. Classes : A class is a way to bind the data and its associated functions together. It allows the data (and functions) to be hidden, if necessary, from external use.

### 5.2. What is a member function ?

**Ans.** A member function of a class is a function that has its definition or its prototype within the class definition like any other variable.

### 5.3. Define constructor.

**Ans.** A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class.

### 5.4. What are the different types of constructors ?

**Ans.** Different types of constructor are :

1. Default constructor
2. Parameterized constructor
3. Copy constructor

### 5.5. Define default constructor and copy constructor.

**Ans.** **Default constructor :** Default constructor is the constructor which doesn't take any argument. It has no parameters.

**Copy constructor :** A copy constructor is a member function which initializes an object using another object of the same class.

### 5.6. What is a destructor function ?

**Ans.** Destructor is a member function which destructs or deletes an object. Following is the Syntax of destructor : constructor-name();

### 5.7. Define operator overloading.

**Ans.** Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. Operator overloading is used to overload or redefines most of the operators available in C++.

### 5.8. What is inheritance ?

**Ans.** The capability of a class to derive properties and characteristics from another class is called Inheritance.

### 5.9. What are the types of inheritance ?

**Ans.** Types of inheritance are :

1. Single inheritance
2. Multiple inheritance
3. Multilevel inheritance
4. Hybrid inheritance

### 5.10. What are the modes of inheritance ?

**Ans.** Following are the modes of inheritance :

1. Public modes
2. Private modes
3. Protected modes

### 5.11. What is the use of virtual base classes ?

**Ans.** Virtual base classes are used in virtual inheritance in a way of preventing multiple "instances" of a given class appearing in an inheritance hierarchy when using multiple inheritances.

### 5.12. What are the types of polymorphism ?

**Ans.** There are two types of polymorphism:

1. Compile-time polymorphism
2. Run-time polymorphism

### 5.13. What is a pointer ?

**Ans.** Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures.

### 5.14. Define pure virtual function.

**Ans.** A pure virtual function is a member function of base class whose only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract.



**B.Tech.**  
**(SEM. V) ODD SEMESTER THEORY**  
**EXAMINATION, 2010-11**  
**OBJECT ORIENTED TECHNIQUES**

---

**Time : 3 Hours****Max. Marks : 100**

---

**Note :** Attempt all questions.

1. Attempt any **two** parts : (10 × 2 = 20)
  - a. i. What do you mean by encapsulation ? How does the object-oriented concept of message passing help to encapsulate the implementation of an object, including its data ?
  - ii. What do you mean by polymorphism ? Is this concept only applicable to object-oriented systems ? Explain.
  - b. i. What do you mean by activity diagram ? What are the two special states shown in an activity diagram ? Explain with an example.
  - ii. Define an abstract class. When is it required to create an abstract class ? Explain it with an example.
  - c. i. Define object-oriented modeling (OOM). Describe various steps involved in OOM process. Explain.
  - ii. What do you mean by multiple inheritances ? Explain it with an example. Can you implement multiple inheritances in Java ?
2. Answer any **two** parts : (10 × 2 = 20)
  - a. Design an AUTOMOBILE base class in Java. Define its all methods and data structures. Through inheritance mechanism, create one class namely CAR. Implement its data structures and important methods. Observe the following while designing the classes :
    - i. Clearly indicate the private and public classes.
    - ii. Design constructors in each class and explain its purpose.
  - iii. Identify data structures and methods which can be inherited (Make suitable assumption where required).

- b. What do you mean by a collaboration diagram ? Explain various terms and symbols used in a collaboration diagram. How polymorphism is described using a collaboration diagram ? Explain using an example.
- c.i. What do you understand by architectural modeling ? Explain its various concepts and diagrams with suitable example.
- ii. Write a short note on use case diagram and time diagram with suitable diagram and their utility in system design.
3. Answer any two parts : (10 × 2 = 20)
- a. Describe the following with example :
- i. Steps of object-oriented design
- ii. Generalization and specialization
- iii. Modeling association as a class
- iv. Physical packaging.
- b.i. How do you map the object-oriented concepts using non-object oriented languages ? Explain with an example.
- ii. Describe the various features of object-oriented languages. Also compare any two object-oriented languages.
- c. Write short notes on the following :
- i. Reusability and robustness.
- ii. Translating object oriented design into an implementation.
4. Answer any two parts : (10 × 2 = 20)
- a.i. Describe the main features of Java. Also discuss the features that make Java different from C ++.
- ii. What do you mean by multithreading ? Does it have an impact on the performance of Java ? Explain.
- b.i. Write a program in Java to calculate the sum of the digits of a given positive integer number. For example, if the given number is 12345 then the program should display : 15.
- ii. Design a class using Java to represent a student record having the following attributes and methods :
- i. Attributes are as follows : Enroll\_no, Name, Father\_name, Branch, Year\_of\_admission, Student\_semester, Student\_address, Student\_status.

- ii. The methods are as follows : to assign the initial values to all attributes, to admit a new student, display the list of students admitted in a given year.
- c. Write short notes on the following :
  - i. Enterprise Java Beans.
  - ii. Abstract methods and classes.
- 5. Answer any two parts : (10 × 2 = 20)
  - a. What do you understand by ODBC ? Why is it required ? How is it implemented using Java ? Explain with an example.
  - b. i. Define Applet. Also discuss Applet life cycle using a diagram and explain its various states.
  - ii. Write a short note on the utility of Java as Internet programming language.
  - c. Write short notes on the following with suitable examples :
    - i. Java Servlets
    - ii. Exception handling techniques



## SOLUTION OF PAPER (2010-11)

**Note :** Attempt **all** questions.

1. Attempt any **two** parts :

(10 × 2 = 20)

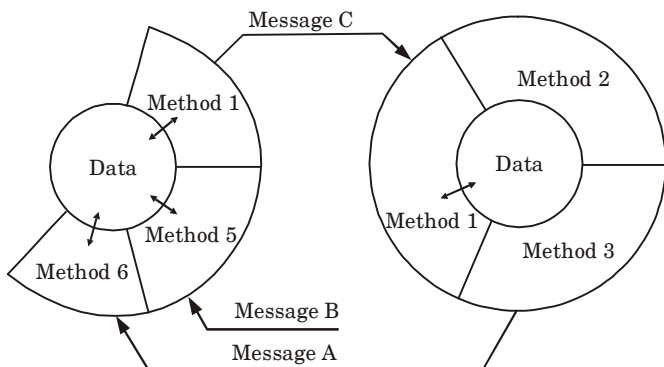
**a. i. What do you mean by encapsulation ? How does the object-oriented concept of message passing help to encapsulate the implementation of an object, including its data ?**

**Ans. Encapsulation :**

1. Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.
2. The only way to reach the data is through these particular methods (Fig. 1).
3. It is the mechanism that binds together code and the data it manipulates.
4. This concept is also used to hide the internal representation, or state, of an object from the outside.

**Object 1**

**Object 2**



**Fig. 1.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

**Using message passing to encapsulate the implementation of an object :** Other parts of a system only see an object's interface (services it can perform and operation signatures). Internal details including data are hidden and can only be accessed by a message that contains a valid signature.

**ii. What do you mean by polymorphism ? Is this concept only applicable to object-oriented systems ? Explain.**

**Ans. Polymorphism :**

1. Polymorphism means having many forms.
2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.

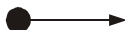
**Applicability of polymorphism :**

1. In programming languages there are two types of polymorphism ad-hoc and universal.
2. There are two kinds of universal polymorphism: parametric and subtyping.
3. Ad-hoc polymorphism is a kind of polymorphism in which polymorphic functions can be applied to arguments of different types.
4. In universal (parametric) polymorphism, the polymorphic functions are written without mention of any specific type.
5. The ad-hoc polymorphism is applicable in both traditional and object-oriented programming environments, whereas universal polymorphism only applies to object-oriented systems.

**b. i. What do you mean by activity diagram ? What are the two special states shown in an activity diagram ? Explain with an example.**

**Ans.**

1. An activity diagram is a flowchart that shows activities performed by a system.
2. The two special states shown in an activity diagram are the Initial State (Start Point) and Final State (End Point).
3. **Initial State or Start Point :** A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.

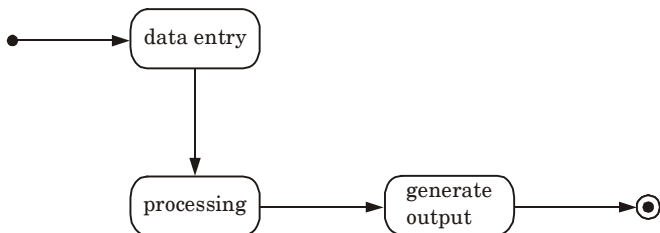


Start Point/Initial State

4. **Final State or End Point :** An arrow pointing to a filled circle nested inside another circle represents the final action state.



Ent Point Symbol

**For example :**

**Fig. 2.** Notation for activity states and activity-state transitions.

**ii. Define an abstract class. When is it required to create an abstract class ? Explain it with an example.**

**Ans. Abstract class :**

1. An abstract class is a class that contains at least one abstract method.
2. An abstract method is a method that is declared, but not instantiated.

**When to create an abstract class :**

1. An abstract class is created when we are using the inheritance concept since it provides a common base class implementation to derived classes.
2. An abstract class is used when we want to declare non-public members.
3. If we want to add new methods in the future, then an abstract class is used.
4. If we want to create multiple versions of our component, we create an abstract class.
5. If we want to provide common, implemented functionality among all implementations of our component, we use an abstract class.

**For example :**

```
//abstract parent class
abstract class animal{
//abstract method
public abstract void sound();
}

//Dog class extends animal class
public class dog extends animal{
public void sound(){
System.out.println("Woof")
}

public static void main(String args[]){
animal obj = new dog();
obj.sound();
}
}
```

**c. i. Define object-oriented modeling (OOM). Describe various steps involved in OOM process. Explain.**



**Ans. Object-oriented modeling :**

1. Object-oriented modeling (OOM) is the construction of objects using a collection of objects that contain stored values of the instance variables found within an object.
2. Object-oriented modeling is an approach to modeling an application that is used at the beginning of the software life cycle when using an object-oriented approach to software development.

**Steps involved in OOM process :****1. System analysis :**

- i. In this stage a statement of the problem is formulated and a model is build. This phase show the important properties associated with the situation.
- ii. The analysis model is a concise, precise abstraction and agreement on how the desired system must be developed.
- iii. The objective is to provide a model that can be understood by any application experts in the area.

**2. System design :**

- i. At this stage, the complete system architecture is designed.
- ii. In this stage the whole system is divided into subsystems, based on system analysis model and the proposed architecture of the system.

**3. Object design :**

- i. At this stage, a design model is developed based on the analysis model.
- ii. The object design decides the data structures and algorithms needed to implement each of the classes in the system.

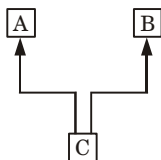
**4. Final implementation :**

- i. At this stage, the final implementation of classes and relationships developed during object design takes place.
- ii. Actual implementation should be done using software engineering practice. This helps to develop a flexible and extensible system.

**ii. What do you mean by multiple inheritances ? Explain it with an example. Can you implement multiple inheritances in Java ?****Ans. Multiple inheritance :**

1. Multiple inheritance is a feature of object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class.
2. When one class extends more than one classes then this is called multiple inheritance.

**For example :** Class C extends class A and B then this type of inheritance is known as multiple inheritance.



### Multiple inheritance in java :

1. Java does not allow multiple inheritance, and we cannot extend more than one other class.
2. Java doesn't allow multiple inheritance to avoid the ambiguity caused by it.

2. Answer any two parts :

(10 × 2 = 20)

a. **Design an AUTOMOBILE base class in Java. Define its all methods and data structures. Through inheritance mechanism, create one class namely CAR. Implement its data structures and important methods. Observe the following while designing the classes :**

- i. Clearly indicate the private and public classes.
- ii. Design constructors in each class and explain its purpose.
- iii. Identify data structures and methods which can be inherited (Make suitable assumption where required).

**Ans.** This question is out of syllabus from session 2020-21.

b. **What do you mean by a collaboration diagram ? Explain various terms and symbols used in a collaboration diagram. How polymorphism is described using a collaboration diagram ? Explain using an example.**

**Ans.** **Collaboration diagram :**

1. A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).
2. These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.
3. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction.

### Various terms used in collaboration diagram :

1. **Objects :** Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name; class name.

Object  
name

2. **Actors :** Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.



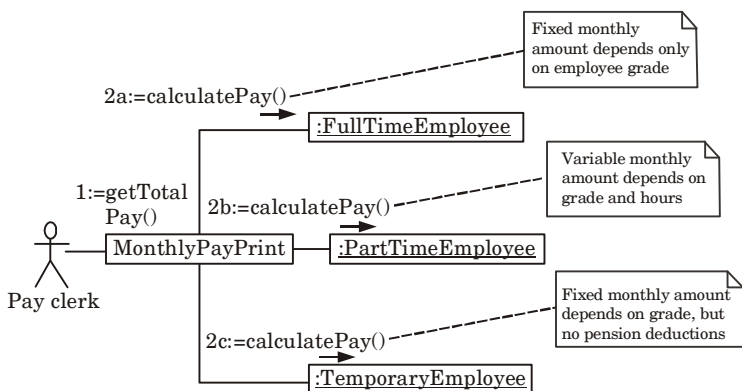
3. **Links :** Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.

Link

4. **Messages :** Messages between objects are shown as a labeled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

**Polymorphism using collaboration diagram :**

- Fig. 3 uses a collaboration diagram to illustrate polymorphism in a business scenario.
- The diagram assumes that there are different ways of calculating an employee's pay.
- Full-time employees are paid a salary that depends only on his or her grade; part-time staff are paid a salary that depends in a similar way on grade, but must also take into account the number of hours worked; temporary staff differ in that no deductions are made for the company pension scheme, but the salary calculation is otherwise the same as for a full-time employee.
- An object-oriented system to calculate pay for these employees might include a separate class for each type of employee, each able to perform the appropriate pay calculation.
- However, following the principle of polymorphism, the message signature for all calculate pay operations is the same.



**Fig. 3.** Polymorphism allows a message to achieve the same result even when the mechanism for achieving it differs between different objects.

6. Suppose one of the outputs from this system is a print-out showing the total pay for the current month: to assemble the total, a message is sent to each employee object, asking it to calculate its pay.
7. Since the message signature is the same in each case, the requesting object (here called MonthlyPayPrint) need not know that the class of each receiving object, still less how each calculation is carried out.

**c.i. What do you understand by architectural modeling ? Explain its various concepts and diagrams with suitable example.**

**Ans. Architectural modeling :**

1. Architectural modeling represents the overall framework of the system.
2. It contains both structural and behavioral elements of the system.
3. Architectural modeling can be defined as the blueprint of the entire system.

**Diagrams used in architectural modeling :**

1. The two types of diagrams that give descriptions of the physical information about a system are deployment diagrams and component diagrams.
2. Deployment diagrams show the physical relationship between hardware and software in a system.
3. Component diagrams show the software components of a system and their relationships.
4. These relationships are called dependencies.

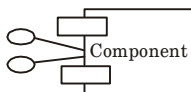
**A. Component diagrams :**

1. The component diagrams are mainly used to model the static implementation view of a system.
2. They represent a high-level packaged view of the code.
3. They can be used to model executables, databases and adaptable systems.
4. Component diagrams mainly contain the following :

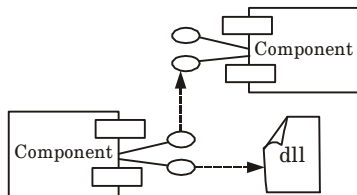
**i. Components :**

- a. A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.
- b. A component is a physical manifestation of an object that has a well-defined interface and a set of implementations for the interface.

- c. A complex system can be built using software components. It enhances re-use in the system and facilitates system evolution.

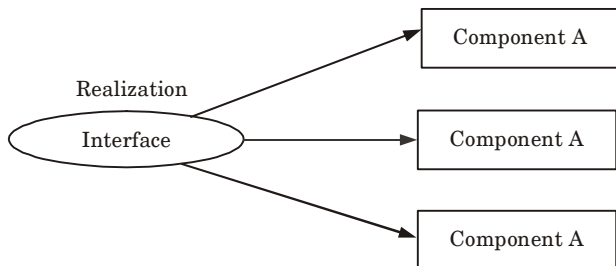


**Fig. 4.** Component.

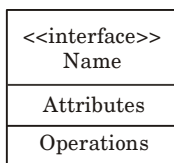


**Fig. 5.** Component diagram.

- ii. **Interfaces :** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle. An interface possesses the following properties :
- Every interface is identified by a unique name, with an operational pathname.
  - Interfaces do not specify any structure or implementation details of the operations. When an interface is represented as a rectangle, it has the same parts as a class. When it is represented as a circle, the display of these parts is suppressed.



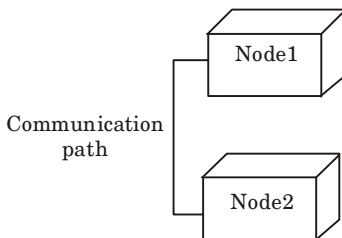
**Fig. 6.** Realization of an interface.



Interface name

**Fig. 7.** Representation of interfaces.

**B. Deployment diagrams :** They display the configuration of run-time processing elements and the software components, processes and objects. The deployment diagram contains nodes and connections. A node is a piece of hardware in the system. A connection depicts the communication path used by the hardware (Fig. 8).



**Fig. 8.** Deployment diagram.

**ii. Write a short note on use case diagram and time diagram with suitable diagram and their utility in system design.**

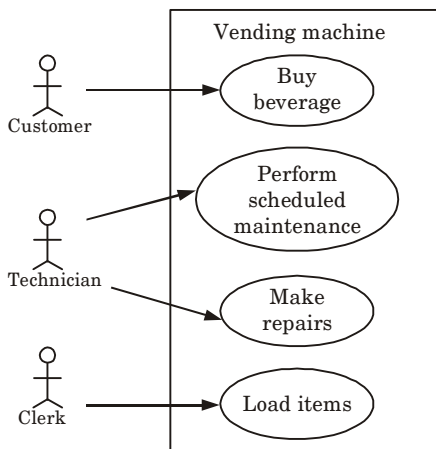
**Ans. Use case diagrams :**

1. Use case diagrams consist of actors, use cases and their relationships.
2. The diagram is used to model the system/subsystem of an application.
3. A single use case diagram captures a particular functionality of a system.
4. Hence to model the entire system, a number of use case diagrams are used.

**Utility of use case diagram in system design :**

1. It is used to gather the requirements of a system.
2. It is used to get an outside view of a system.
3. It identifies the external and internal factors influencing the system.
4. It shows the interaction among the requirements of actors.
5. It is used for requirement analysis and high level design.
6. It models the context of a system.
7. It is used in reverse engineering and forward engineering.

**For example :**



**Fig. 9.** Use case diagram for a vending machine.

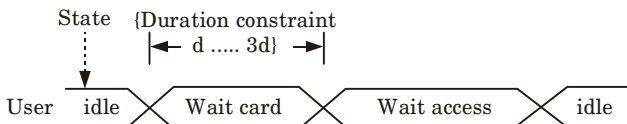
**Time diagram :**

1. Timing diagram is used to show interactions when a primary purpose of the diagram is about time.
2. It focuses on conditions changing within and among lifelines along a linear time axis.
3. Timing diagram is a special form of a sequence diagram.

**Utility of time diagram in system design :**

1. It emphasizes at that particular time when the message has been sent among objects.
2. It explains the time processing of an object in detail.
3. It is employed with distributed and embedded systems.
4. It also explains how an object undergoes changes in its form throughout its lifeline.
5. It depicts a graphical representation of states of a lifeline per unit time.

**For example :**



**Fig. 10.**

**3. Answer any two parts :**

**(10 × 2 = 20)**

**a. Describe the following with example :**

**i. Steps of object-oriented design**

**ii. Generalization and specialization**

iii. **Modeling association as a class**

iv. **Physical packaging.**

**Ans.**

i. **Steps of object-oriented design :**

1. **System analysis :**

- i. In this stage a statement of the problem is formulated and a model is build. This phase show the important properties associated with the situation.
- ii. The analysis model is a concise, precise abstraction and agreement on how the desired system must be developed.
- iii. The objective is to provide a model that can be understood by any application experts in the area.

2. **System design :**

- i. At this stage, the complete system architecture is designed.
- ii. In this stage the whole system is divided into subsystems, based on system analysis model and the proposed architecture of the system.

3. **Object design :**

- i. At this stage, a design model is developed based on the analysis model.
- ii. The object design decides the data structures and algorithms needed to implement each of the classes in the system.

4. **Final implementation :**

- i. At this stage, the final implementation of classes and relationships developed during object design takes place.
- ii. Actual implementation should be done using software engineering practice. This helps to develop a flexible and extensible system.

ii. **Generalization and specialization :** Generalization and specialization represent a hierarchy of relationships between classes, where subclasses inherit from super-classes.

1. **Generalization :**

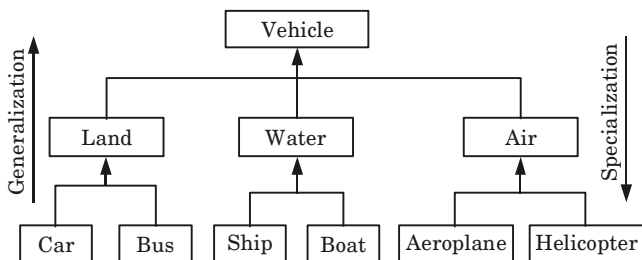
- i. In the generalization process, the common characteristics of classes are combined to form a class in a higher level of hierarchy, *i.e.*, subclasses are combined to form a generalized super-class.
- ii. It represents an “is - a - kind - of” relationship. For example, “car is a kind of land vehicle”, or “ship is a kind of water vehicle”.



## 2. Specialization :

- Specialization is the reverse process of generalization. Here, the distinguishing features of groups of objects are used form specialized classes from existing classes.
- It can be said that the subclasses are the specialized versions of the super-class.

The following figure shows an example of generalization and specialization.



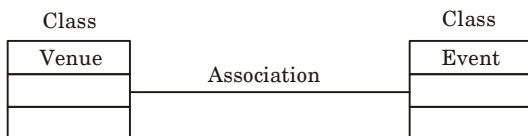
**Fig. 11.**

### iii. Modeling association as a class :

- The purpose of an association is to establish the reasons why two classes of objects need to know about one another and the rules that govern the relationship.
- Modeling an association begins by identifying the participating classes.

#### A. Binary association notation :

- In a class, a binary association documents the rules that govern one relationship between two classes of object. The association is rule that explains what is allowed.
- Fig. 12 shows an association connecting venue class and event class.



**Fig. 12.**

- The venue class defines what a venue object is and what it can do. The event class defines what an event is and what it can do.

#### B. Association name :

- The naming of classes, attributes, and operations is very important. The usual way to name an association is with a verb or verb phrase.

2. The Fig. 13 shows the name association modeled with two different names using verb “hosts” and a verb phrase “is hosted by”.

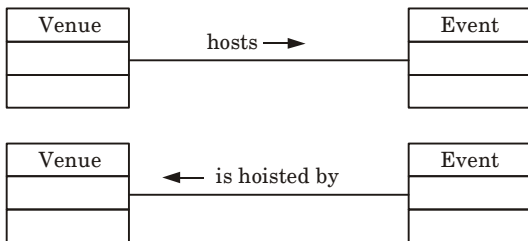


Fig. 13.

- C. Reflexive association :** A reflexive association is a very common phase to use role names. As in following example.

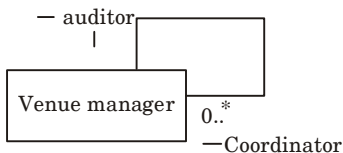


Fig. 14.

- D. Derived association :**

1. A derived association is much like a derived attribute in that it is not really needed, i.e., the same information could be figured out by looking at other associations.
2. For example Fig. 15 represents a portion of the theatre system.

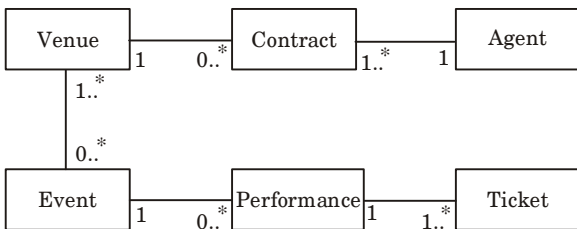


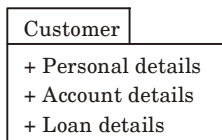
Fig. 15.

3. The venue hosts an event that is made up of a number of performances for which it sells tickets.
4. In order to find out how many tickets an agent sold the application would need to traverse all of these associations, filtering out unwanted object along the way.

#### iv. Physical packaging :

1. A component is a physical and replaceable part of the system that conforms and provides the realization of a set of interfaces.

2. It represents the physical packaging of elements like classes and interfaces.
3. A package is an organized group of elements. A package may contain structural things like classes, components, and other packages in it.
4. For example, a package is represented by a tabbed folder.
5. A package is generally drawn with only its name. However, it may have additional details about the contents of the package.



**b. i. How do you map the object-oriented concepts using non-object oriented languages ? Explain with an example.**

**Ans.** Implementing an object-oriented concept in a non-object oriented language requires the following steps :

**1. Translate classes into data structures :**

- i. Each class is implemented as a single contiguous block of attributes. Each attribute contains variable. Now an object has state and identity and is subject to side effects.
- ii. A variable that identifies an object must therefore be implemented as a sharable reference.

**2. Pass arguments to methods :**

- i. Every method has at least one argument. In a non-object-oriented language, the argument must be made explicit.
- ii. Methods can contain additional objects as arguments. In passing an object as an argument to a method, a reference to the object must be passed if the value of the object can be updated within the method.

**3. Allocate storage for objects :**

- i. Objects can be allocated statically, dynamically or on a stack.
- ii. Most temporary and intermediate objects are implemented as stack-based variables.
- iii. Dynamically allocated objects are used when their number is not known at compile time.
- iv. A general object can be implemented as a data structure allocated on request at run time from a heap.

**4. Implement inheritance in data structures :** Following ways are used to implement data structures for inheritance in a non-object-oriented language :

- i. Avoid it.
- ii. Flatten the class hierarchy.
- iii. Break out separate objects.

**5. Implement method resolution :** Method resolution is one of the main features of an object-oriented language that is lacking in a non-object-oriented language. Method resolution can be implemented in following ways :

- i. Avoid it.
- ii. Resolve methods at compile time.
- iii. Resolve methods at run time.

**6. Implement associations :** Implementing associations in a non-object-oriented language can be done by :

- i. Mapping them into pointers.
- ii. Implementing them directly as association container objects.

**7. Deal with concurrency :**

- i. Most languages do not explicitly support concurrency.
- ii. Concurrency is usually needed only when more than one external event occurs, and the behaviour of the program depends on their timing.

**8. Encapsulate internal details of classes :**

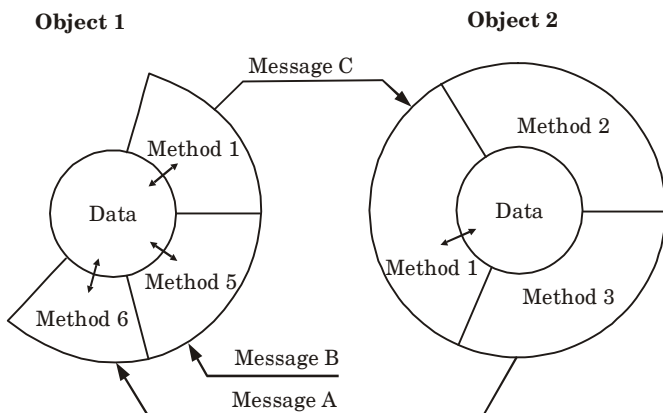
- i. Object-oriented languages provide constructs to encapsulate implementation.
- ii. Some of this encapsulation is lost when object-oriented concept is translated into a non-object-oriented language, but we can still take advantage of the encapsulation facilities provided by the language.

**ii. Describe the various features of object-oriented languages. Also compare any two object-oriented languages.**

**Ans.** Features of object-oriented language are :

**1. Encapsulation :**

- i. Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.
- ii. The only way to reach the data is through these particular methods (see Fig. 16).
- iii. It is the mechanism that binds together code and the data it manipulates.
- iv. This concept is also used to hide the internal representation, or state, of an object from the outside.



**Fig. 16.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

## 2. Polymorphism :

- Polymorphism means having many forms.
- Polymorphism is the ability of a message to be displayed in more than one form.
- It plays an important role in allowing objects having different internal structure to share the same external interface.

## 3. Inheritance :

- Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features.
- Inheritance is mainly used for code reusability.

### Difference :

| S. No. | C++                                   | Java                                                                                                      |
|--------|---------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 1.     | C++ is a platform dependent language. | Java is platform-independent language.                                                                    |
| 2.     | C++ is a compiled language.           | Java is a compiled as well as an interpreted language.                                                    |
| 3.     | C++ code is not portable.             | Java, translates the code into byte code. This byte code is portable and can be executed on any platform. |
| 4.     | Memory management in C++ is manual.   | In Java the memory management is automatic.                                                               |

**c. Write short notes on the following :****i. Reusability and robustness.****ii. Translating object oriented design into an implementation.****Ans.****i. Reusability :**

1. Reusability is a segment of source code that can be used again to add new functionalities with slight or no modification.
2. Reusable software reduces design, coding and testing cost. Also, reducing the amount of code simplifies understanding. In object-oriented languages the possibility of code reuse is greatly enhanced.
3. There are two kind of reuse :
  - i. Sharing of newly-written code within a project.
  - ii. Reuse of previously-written code on new project.
4. **Rules for reusability :** Following rules must be kept in mind :
  - i. Keep methods coherent.
  - ii. Keep methods small.
  - iii. Keep methods consistent.
  - iv. There should be separate policy and implementation.
  - v. Provide uniform coverage.
  - vi. Broaden the method as much as possible.
  - vii. Avoid global information.
  - viii. Avoid modes.

**ii. Robustness :**

1. A method is robust if it does not fail even if it receives improper parameters.
2. Robustness against internal bugs may be traded off against efficiency.
3. Robustness against user errors should never be sacrificed.
4. Rules for robustness :
  - i. **Protect against errors :** Software should protect itself against incorrect user input. Incorrect user input should never cause a crash.
  - ii. **Optimize after the program runs :** Don't optimize a program until you get it working.
  - iii. **Validate arguments :** External operations, those available to users of the class, must rigorously check their arguments to prevent failure. Don't include arguments that cannot be validated.

- iv. **Avoid predefined limits :** When possible use dynamic memory allocation to create data structures that do not have predefined limits.
  - v. **Instrument the program for debugging and performance monitoring :** You should instrument your code for debugging, statistics and performance. The level of debugging that you must build into your code depends on the programming environment presented by the language.
- ii. **Translating object-oriented design into an implementation :**
- i. It is easy to implement an object-oriented design with an object-oriented language since language constructs are similar to design constructs.
  - ii. The following steps are required to implement an object oriented design in an object-oriented language :
    - 1. **Class definitions :**
      - i. The first step in implementing an object-oriented design is to declare object classes. Each attribute and operation in an object diagram must be declared as part of its corresponding class.
      - ii. Assign data types to attributes. Declare attributes and operations as either public or private.
    - 2. **Creating objects :**
      - i. Object-oriented languages create new objects in following two ways :
        - a. Class operation applied to a class object creates a new object of the class.
        - b. Using special operations that create new objects.
      - ii. When a new object is created, the language allocate storage for its attribute values and assigns it a unique object ID.
    - 3. **Calling operations :**
      - i. In most object-oriented languages, each operation has at least one implicit argument, the target object, indicated with a special syntax.
      - ii. Operations may or may not have additional arguments.
    - 4. **Using Inheritance :**
      - i. To implement inheritance object-oriented languages use different mechanisms.

- ii. There are three independent dimensions for classifying inheritance mechanisms :
  - a. Static or dynamic
  - b. Implicit or explicit
  - c. Per object or per group.
- iii. Many of the popular languages are static, implicit and per group.

**5. Implementing associations :**

- i. There are two approaches to implement associations : buried pointers and distinct association objects.
- ii. If the language does not explicitly support association objects then buried pointers are easy to implement.
- iii. An association can also be implemented as a distinct container object.

4. Answer any **two** parts :

**(10 × 2 = 20)**

**a. i. Describe the main features of Java. Also discuss the features that make Java different from C ++.**

**Ans.** This question is out of syllabus from session 2020-21.

**ii. What do you mean by multithreading ? Does it have an impact on the performance of Java ? Explain.**

**Ans.** This question is out of syllabus from session 2020-21.

**b. i. Write a program in Java to calculate the sum of the digits of a given positive integer number. For example, if the given number is 12345 then the program should display : 15.**

**Ans.** This question is out of syllabus from session 2020-21.

**ii. Design a class using Java to represent a student record having the following attributes and methods :**

**i. Attributes are as follows : Enroll\_no, Name, Father\_name, Branch, Year\_of\_admission, Student\_semester, Student\_address, Student\_status.**

**ii. The methods are as follows : to assign the initial values to all attributes, to admit a new student, display the list of students admitted in a given year.**

**Ans.** This question is out of syllabus from session 2020-21.

**c. Write short notes on the following :**

**i. Enterprise Java Beans.**

**ii. Abstract methods and classes.**

**Ans.** This question is out of syllabus from session 2020-21.



**5. Answer any two parts : (10 × 2 = 20)**

**a. What do you understand by ODBC ? Why is it required ? How is it implemented using Java ? Explain with an example.**

**Ans.** This question is out of syllabus from session 2020-21.

**b. i. Define Applet. Also discuss Applet life cycle using a diagram and explain its various states.**

**Ans.** This question is out of syllabus from session 2020-21.

**ii. Write a short note on the utility of Java as Internet programming language.**

**Ans.** This question is out of syllabus from session 2020-21.

**c. Write short notes on the following with suitable examples :**

**i. Java Servlets**

**ii. Exception handling techniques**

**Ans.** This question is out of syllabus from session 2020-21.



**B.Tech.**  
**(SEM. V) ODD SEMESTER THEORY**  
**EXAMINATION, 2011-12**  
**OBJECT ORIENTED TECHNIQUES**

---

**Time : 3 Hours****Max. Marks : 100**

---

**Note :** Attempt all questions.

1. Attempt any *two* parts : (2 × 10 = 20)
    - a. i. What do you understand by object-oriented technology ? Discuss the pros and cons of object-oriented technology with suitable example.
    - ii. Differentiate between a class and object with some example. Also prepare a list of objects that you would expect each of the following systems to handle : (1) a program for laying out a newspaper, (2) a catalog store order entry system.
  - b. i. What do you mean by modeling ? Discuss several purposes served by models with suitable examples.
  - ii. What do you mean by generalization ? Explain. How is it related with inheritance ?
  - c. i. What do you mean by UML? Discuss the conceptual model of UML with the help of an appropriate example.
  - ii. Wire is used in the following applications. For each of the following applications, prepare a list of wire characteristics that are relevant and also explain why each characteristic is important for the application : (1) Designing the filament for a light bulb; (2) Designing the electrical system for an airplane.
2. Answer any *two* parts : (10 × 2 = 20)
    - a. i. Give the general layout of a class diagram. Also prepare a class diagram for the instance diagram shown in the Fig. 3. Explain your multiplicity decisions. How does your diagram express the fact that points are in sequence ?

- ii. What is a collaboration diagram ? How polymorphism is represented in a collaboration diagram? Explain with an example.
- b. What do you mean by sequence diagram? Explain various terms and symbols used in a sequence diagram. Describe the following using sequence diagram : (i) asynchronous messages with/without priority. (ii) broadcast messages.
- c. i. Discuss in brief the following terms : (1) Component diagrams. (2) Basic behavioural modeling.
- ii. Prepare a portion of an object diagram for a library book checkout system that shows the date a book is due and the late charges for an overdue book as derived objects.
- 3. Answer any two parts : (10 × 2 = 20)
  - a. Explain each of the following with in reference to object oriented programming style with an example :
    - i. Reusability
    - ii. Robustness
    - iii. Extensibility
    - iv. Abstraction
  - b. i. How object-oriented concept can be implemented using non-object-oriented language? Explain with an example.
  - ii. What do you mean by documentation? What are the various considerations in documentation designing? Explain.
  - c. Write short notes on the following :
    - i. Jackson Structured Development (JSD).
    - ii. Dynamic modeling and Functional modeling.
- 4. Answer any two parts : (10 × 2 = 20)
  - a. i. Why Java is known as a platform independent language? Discuss the advantages and disadvantages of a platform independent language. Also give various data types in Java.
  - ii. How polymorphism is handled in Java ? Explain with some suitable example using Java programming language.
  - b. i. Write a program in Java to count display the frequency of vowels in a given sentence of a at least 35 characters long.

- ii. **Design class using Java to represent a student record having the following attributes and methods :** (i) **Attributes of the student Institute are as follows :** Student\_ID, Student\_Name, Student\_Address, Birth\_Date, Course, Enrollment\_Year; (ii) **The methods are as follows:** to assign the initial values to all attributes, to add a new student record, display the list of students for a given year of enrollment and course.
  
- c. **Write short notes on the following giving their significance and with suitable example using Java in brief :**
  - i. **Enterprise Java Beans**
  - ii. **Java API's**
  
- 5. **Answer any two parts :** (10 × 2 = 20)
  - a. i. **What do you mean by Applets ? How Applets differ from the applications ? Explain with an example using Java.**
  
  - ii. **Write a short note on Java Swing with suitable example.**
  
  - b. **Write short notes on the following with an example using Java :** (i) JAR files (ii) Packages (iii) Multithreading (iv) Interface.
  
  - c. **What do you mean by JDBC? What is its significance ? How database connectivity is done using Java ? Discuss it with suitable example.**



**SOLUTION OF PAPER (2011-12)**

**Note :** Attempt **all** questions.

**1. Attempt any two parts :**

**(2 × 10 = 20)**

- a. i. What do you understand by object-oriented technology ? Discuss the pros and cons of object-oriented technology with suitable example.**

**Ans.**

1. Object-Oriented Technology (OOT) is an approach to program organization and development that attempts to reduce some of the issues with conventional programming techniques.
2. It is a new way of organizing and developing programs and has nothing to do with any particular programming language.
3. However, not all languages are suitable to implement the object-oriented concepts or implement partial features of object-oriented concepts.

**Pros of object-oriented technology are :**

1. **It allows parallel development :** If we are working with programming teams, then each can work independently of one another once the modular classes have been worked out.
2. **The modular classes are often reusable :** Once the modular classes have been created, they can often be used again in other applications or projects.
3. **The coding is easier to maintain :**
  - a. With OOP, because our coding base has been centralized, it is easier to create a maintainable procedure code.
  - b. That makes it easier to keep our data accessible when it becomes necessary to perform an upgrade.
  - c. This process also improves the security of the programming since high levels of validation are often required.

**Cons of object-oriented technology are :**

1. **It is inefficient :**
  - a. Object-oriented programming tends to use more CPU than alternative options.
  - b. That can make it inefficient choice when there are technical limitations involved due to the size.
2. **It is scalable :**
  - a. If OOP is out of control, then it can create a massive amount of bloated, unnecessary code.
  - b. When that occurs, the overhead rises and that makes it difficult to keep costs down.

**3. It causes duplication :**

- OOP projects tend to be easier to design than implement.
- That is because of the modular classes that are so flexible in their application.
- We may be able to get new projects up and running at a greater speed, but that comes at the cost of having projects sometimes feel like they have been cloned.

**ii. Differentiate between a class and object with some example. Also prepare a list of objects that you would expect each of the following systems to handle : (1) a program for laying out a newspaper, (2) a catalog store order entry system.****Ans.**

| S. No. | Class                                                                            | Object                                                                                               |
|--------|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| 1.     | Class is a blueprint or template from which object are created.                  | Object is an instance of class.                                                                      |
| 2.     | Class is a group of similar objects.                                             | Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.          |
| 3.     | Class is a logical entity.                                                       | Object is physical entity.                                                                           |
| 4.     | Class is declared using class keyword, for example, <code>Class student{}</code> | Object is created through new keyword mainly, for example, <code>Student s1 = new student ();</code> |
| 5.     | Class is declared once.                                                          | Object is created many times as per requirement.                                                     |
| 6.     | Class does not allocated memory when it is created.                              | Object allocates memory when it is created.                                                          |

**For example :**

- A class is a way of grouping objects that share a number of characteristics: attributes (like name, color, height, weight, etc.) and behavior (such as ability to perform jumps, to run, to swim, etc.).
  - All objects in the class horse will have an attribute named height, for example. That means that all object in that class have a height- the value of the attribute height will be different for each instance of the class (*i.e.*, for each particular horse).
- i. A program for laying out a newspaper :** Classes that you would expect in a program for newspaper layout include Page, Column, Line, Headline, and Paragraph.

**ii. A catalog store order entry system :** For a catalog store order entry system, classes include Customer, Order, Store, and Item.

**b. i. What do you mean by modeling ? Discuss several purposes served by models with suitable examples.**

**Ans.**

1. A model is an abstraction of something for the purpose of understanding it before building it.
2. Since a model leave out non essential detail, it is easier to manipulate them.
3. To build hardware and software systems, the developer needs to :
  - i. Abstract different views of the system.
  - ii. Build models using precise notations.
  - iii. Make sure that the model satisfy the requirements of the system.
  - iv. Add details to transform the model into an implementation.
4. Model serves the following purpose :

**a. Testing a physical entity before building it :**

- i. Simulating a model is cheaper. Also, it provides information that is too inaccessible to be measured from physical model.
- ii. Computer models are usually cheaper than building a complete system and it enable flaws to be corrected early.

**For example :** Scale models of airplane and cars are tested in wind tunnels to improve their aerodynamic.

**b. Communication with customers :**

- i. Software designers build models to show their customers.

**For example :** Demonstration products like mock-ups that imitate some or all of the external behavior of a system.

**c. Visualization :**

- i. Storyboards of movies, television shows, and advertisements allow the writers to see how their idea flows.
- ii. Using models unnecessary segments can be modified before the final development begins.

**d. Reduction of complexity :**

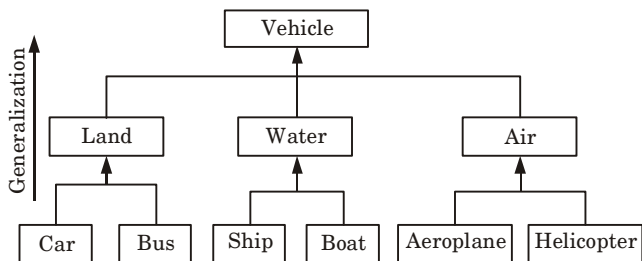
- i. Modeling helps in understanding the systems that are too complex to understand directly.
- ii. Model reduces complexity by leaving out the non essential details.

**ii. What do you mean by generalization ? Explain. How is it related with inheritance ?**

**Ans.**

**1. Generalization :**

- i. In the generalization process, the common characteristics of classes are combined to form a class in a higher level of hierarchy, *i.e.*, subclasses are combined to form a generalized super-class.
- ii. It represents an “is - a - kind - of” relationship. For example, “car is a kind of land vehicle”, or “ship is a kind of water vehicle”.



**Fig. 1.**

**Relation to inheritance :**

1. Developers use the term generalization or inheritance to refer to the same concept of reusing shared attributes and operations that you show in a superclass and reuse in subclasses.
2. Generalization refers to the concept of generalizing from specifics (the subclasses) to the generic (the superclass).
3. Inheritance refers to the effect of generalization on the subclasses.

**c. i. What do you mean by UML? Discuss the conceptual model of UML with the help of an appropriate example.**

**Ans. UML :**

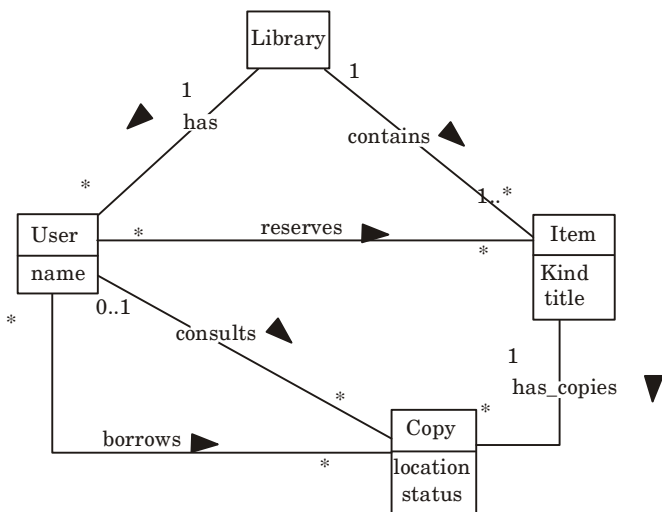
1. UML stands for Unified Modeling Language.
2. UML is a pictorial language used to make software blueprints.
3. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
4. It is also used to model non-software systems as well.
5. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams.
6. UML has a direct relation with object-oriented analysis and design.

**Conceptual model of UML :**

1. A conceptual model is defined as a model which is made of concepts and their relationships.



2. A conceptual model is the first step before drawing a UML diagram.
3. It helps to understand the entities in the real world and how they interact with each other.
4. The conceptual model of UML has three major elements :
  - a. UML building blocks.
  - b. Rules to connect the building blocks.
  - c. Common mechanisms of UML.
5. A domain model, often referred to as a conceptual model, might be represented by a particular kind of UML class diagram.
6. This model explains the structure of the application domain rather than the application structure itself.
7. It focuses on the domain concepts, rather than on the software entities.
8. Fig. 2 shows the conceptual model for a library system.



**Fig. 2.** Conceptual model for a library system.

- ii. Wire is used in the following applications. For each of the following applications, prepare a list of wire characteristics that are relevant and also explain why each characteristic is important for the application : (1) Designing the filament for a light bulb; (2) Designing the electrical system for an airplane.

**Ans. Designing the filament for a light bulb :**

1. Because the filament of a light bulb operates at a high temperature, resistance to high temperatures is important.
2. Tungsten is generally used because of its high melting point, even though tungsten filaments are brittle.

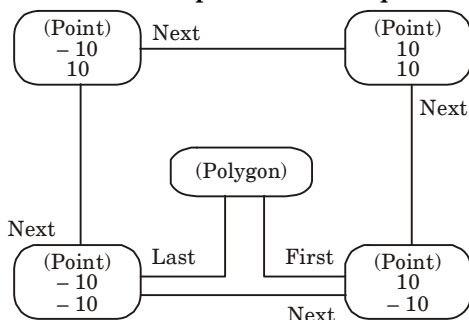
**Designing the electrical system for an airplane :**

1. Weight is important for wire that is to be used in the electrical system of an airplane, because it affects the total weight of the plane.
2. Toughness of the insulation is important to resist chafing due to vibration.
3. Resistance of the insulation to fire is also important to avoid starting or feeding electrical fires in flight.

2. Answer any **two** parts :

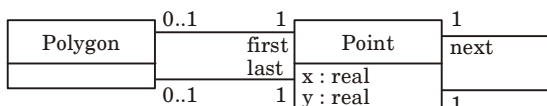
(10 × 2 = 20)

- a. i. **Give the general layout of a class diagram. Also prepare a class diagram for the instance diagram shown in the Fig. 3. Explain your multiplicity decisions. How does your diagram express the fact that points are in sequence ?**



**Fig. 3.**

**Ans.**



**Fig. 4.** General class diagram for polygon and points.

1. Fig. 4 shows the class diagram.
2. Fig. 4 permits a degenerate polygon which consists of exactly one point. (The same point is first and last. The point is next to itself).
3. The class diagram also permits a line to be stored as a polygon.
4. Fig. 4 does not enforce the constraint that the first and last points must be adjacent.

5. In Fig. 4 the sense of ordering is problematic. A polygon that is traversed in left-to-right order is stored differently than one that is traversed in right-to-left order even though both visually appear the same.
6. There is no constraint that a polygon be closed and that a polygon not cross itself.
7. In general it is difficult to fully capture constraints with class models and we must choose between model complexity and model completeness.

**ii. What is a collaboration diagram ? How polymorphism is represented in a collaboration diagram? Explain with an example.**

**Ans. Collaboration diagram :**

1. A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).
2. These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.
3. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction.

**Various terms used in collaboration diagram :**

1. **Objects :** Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name; class name.



2. **Actors :** Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.



3. **Links :** Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.

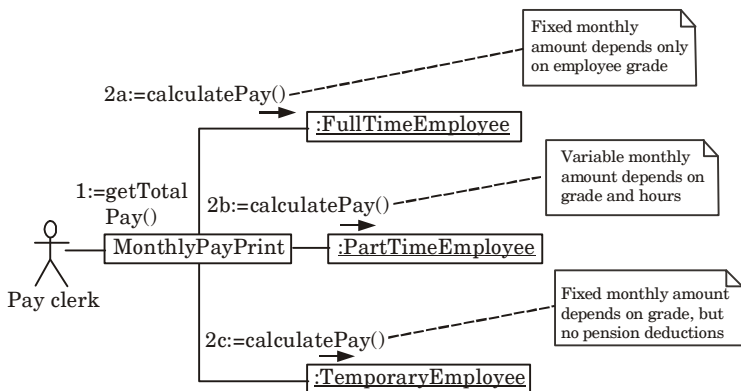


4. **Messages :** Messages between objects are shown as a labeled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

**Polymorphism using collaboration diagram :**

1. Fig. 5 uses a collaboration diagram to illustrate polymorphism in a business scenario.

- The diagram assumes that there are different ways of calculating an employee's pay.
- Full-time employees are paid a salary that depends only on his or her grade; part-time staff are paid a salary that depends in a similar way on grade, but must also take into account the number of hours worked; temporary staff differ in that no deductions are made for the company pension scheme, but the salary calculation is otherwise the same as for a full-time employee.
- An object-oriented system to calculate pay for these employees might include a separate class for each type of employee, each able to perform the appropriate pay calculation.
- However, following the principle of polymorphism, the message signature for all calculate pay operations is the same.



**Fig. 5.** Polymorphism allows a message to achieve the same result even when the mechanism for achieving it differs between different objects.

- Suppose one of the outputs from this system is a print-out showing the total pay for the current month: to assemble the total, a message is sent to each employee object, asking it to calculate its pay.
- Since the message signature is the same in each case, the requesting object (here called MonthlyPayPrint) need not know that the class of each receiving object, still less how each calculation is carried out.

**b. What do you mean by sequence diagram? Explain various terms and symbols used in a sequence diagram. Describe the following using sequence diagram : (i) asynchronous messages with/without priority. (ii) broadcast messages.**

**Ans. Sequence diagram :**

- Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

2. They're also called event diagrams.
3. A sequence diagram is a good way to visualize and validate various runtime scenarios.
4. In UML it is shown as a table that shows objects arranged along the X axis and messages along the Y axis.
5. It has a global life line and the focus of control.

### Various terms and symbols used in sequence diagram :


1. **Class roles or Participants :** Class roles describe the way an object will behave in context.



Object

Fig. 6.

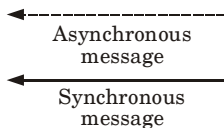
2. **Activation or Execution Occurrence :** Activation occurrence represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



Activation  
occurrence

Fig. 7.

3. **Messages :** Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



Asynchronous  
message

Synchronous  
message

Fig. 8.

4. **Lifelines :** Lifelines are vertical dashed lines that indicate the object's presence over time.



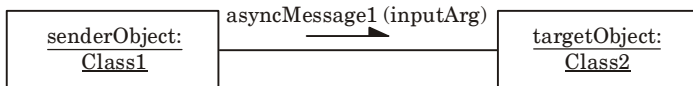
Lifeline

Fig. 9.

5. **Destroying objects :** Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, we can place an X at the end of its lifeline to denote a destruction occurrence.
6. **Loops :** A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets.

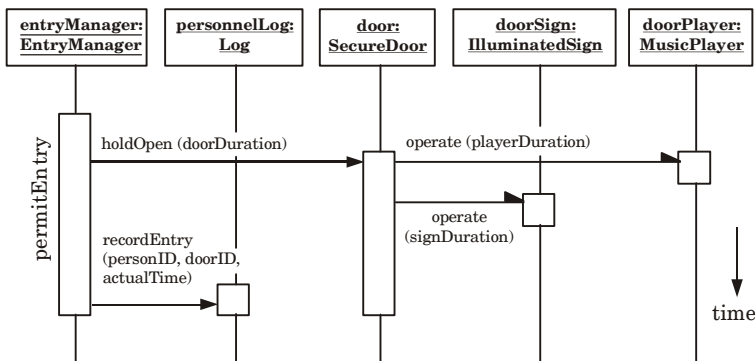
### A. Asynchronous messages without priority :

1. An asynchronous message can be implemented on the sequence diagram by using a half arrowhead on the asynchronous message arrow.



**Fig. 10.** A collaboration diagram showing a basic asynchronous message.

2. When an asynchronous message is send, at least to loci of execution are active in the system, because the target begins to execute while the sender remains in execution.
3. Fig. 11 shows a specific example from a real-time system that authorizes personnel to pass through electronically controlled doors.
4. The employee inserts an ID card into a reader, and if the employee is authorized to enter, the system plays jingly music, displays a greeting and slides the door open.



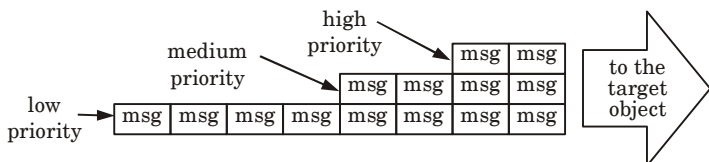
**Fig. 11.** A sequence diagram for concurrently executing objects.

5. The operation that manages this piece of the application is **permitEntry**.
6. After determining that the employee is allowed through the door **permitEntry** sends the synchronous message **holdOpen** to the object **door**.
7. The operation **holdOpen** then handles the sound, lights, and action associated with opening the door.
8. For the action of opening the door, **holdOpen** sends messages to a hardware driver.
9. For the sound-and-light show, **holdOpen** sends two asynchronous messages, both named **operate** : one to **doorPlayer** and one to **doorSign**.

10. The two operations named **operate** execute concurrently: **door**.

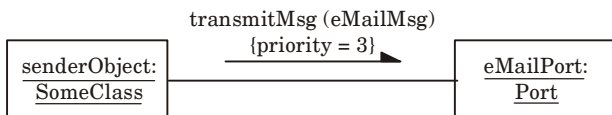
### B. Asynchronous message with priority :

1. In concurrency the target object receives message from lots of concurrently executing sender objects.
2. Since these messages may arrive faster than the target can process them, they are ushered into the message queue.
3. The object then removes a message from the front of the queue; process the message, and takes the next message from the queue.
4. These messages in a queue are ordered by priority.
5. Fig. 12 shows a set of parallel queues at the target, each queue with its own priority level.



**Fig. 12.** Three parallel queues, each with its own priority.

6. Fig. 13 shows an asynchronous message with its priority level.
7. The property {priority = 3} indicates that the message has a priority of 3.



**Fig. 13.** An asynchronous message (with priority 3) going to an object with a multiple-priority message queue.

### Broadcast message :

1. A broadcast message treats every object in the system as a potential target.
2. A copy of the message goes into the queue of every object in the system.
3. An object may broadcast a message in response to some external event that it detects.
4. For example, an object might detect a security compromise and broadcast to all objects the need for a priority system shutdown.
5. Fig. 14 shows the UML for a broadcast message. Here, a start-up sequencer is getting every object in the system that exists at start-up time to load itself.

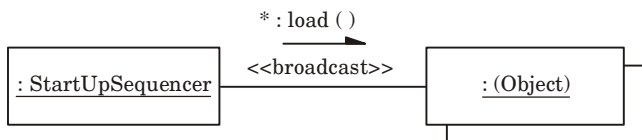


Fig. 14. A broadcast message.

**c.i. Discuss in brief the following terms : (1) Component diagrams. (2) Basic behavioural modeling.**

**Ans.**

**1. Component diagrams :**

1. The component diagrams are mainly used to model the static implementation view of a system.
2. They represent a high-level packaged view of the code.
3. They can be used to model executables, databases and adaptable systems.
4. Component diagrams mainly contain the following :

**i. Components :**

- a. A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.
- b. A component is a physical manifestation of an object that has a well-defined interface and a set of implementations for the interface.
- c. A complex system can be built using software components. It enhances re-use in the system and facilitates system evolution.

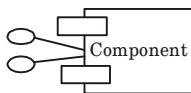


Fig. 15. Component.

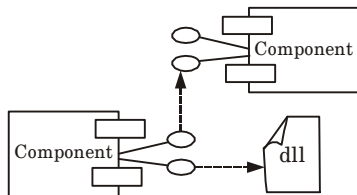
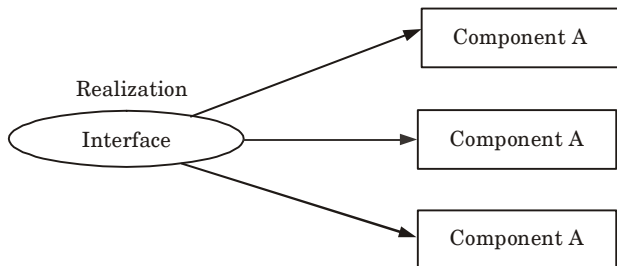


Fig. 16. Component diagram.

- ii. Interfaces :** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle. An interface possesses the following properties :



- a. Every interface is identified by a unique name, with an operational pathname.
- b. Interfaces do not specify any structure or implementation details of the operations. When an interface is represented as a rectangle, it has the same parts as a class. When it is represented as a circle, the display of these parts is suppressed.



**Fig. 17.** Relations of an interface.



**Fig. 18.** Representation of interfaces.

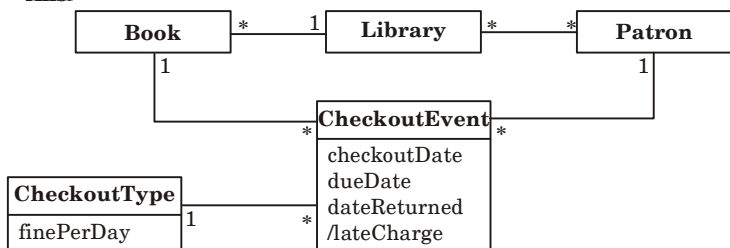
## 2. Basic behavioural modeling:

- i. Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.
- ii. During analysis, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.
- iii. In the design and implementation phases, the detailed design of the operations contained in the object is fully specified.
- iv. There are two types of behavioral models. First, there are behavioral models that are used to represent the underlying details of a business process portrayed by a use case model. In UML, interaction diagrams (sequence and communication) are used for this type of behavioral model.
- v. Second, there is a behavioral model that is used to represent the changes that occur in the underlying data. UML uses behavioral state machines for this.

- vi. During the analysis phase, analysts use behavioral model to capture a basic understanding of the dynamic aspects of the underlying business process.
- vii. Traditionally, behavioral models have been used primarily during the design phase where analysts refine the behavioral models to include implementation details.

ii. Prepare a portion of an object diagram for a library book checkout system that shows the date a book is due and the late charges for an overdue book as derived objects.

**Ans.**



{lateCharge = (dataReturned - dueDate) \* CheckoutType.finePerDay}

**Fig. 19.** Class diagram for library book checkout system.

3. Answer any two parts :

(10 × 2 = 20)

a. Explain each of the following with in reference to object oriented programming style with an example :

- i. Reusability
- ii. Robustness
- iii. Extensibility
- iv. Abstraction

**Ans.**

i. **Reusability :**

1. Reusability is a segment of source code that can be used again to add new functionalities with slight or no modification.
2. Reusable software reduces design, coding and testing cost. Also, reducing the amount of code simplifier understanding. In object-oriented languages the possibility of code reuse is greatly enhanced.
3. There are two kind of reuse :
  - i. Sharing of newly-written code within a project.
  - ii. Reuse of previously-written code on new project.
4. **Rules for reusability :** Following rules must be kept in mind :
  - i. Keep methods coherent.
  - ii. Keep methods small.

- iii. Keep method consistent.
- iv. There should be separate policy and implementation.
- v. Provide uniform average.
- vi. Broaden the method as much as possible.
- vii. Avoid global information.
- viii. Avoid modes.

## ii. Robustness :

1. A method is robust if it does not fail even if it receives improper parameters.
2. Robustness against internal bugs may be traded off against efficiency.
3. Robustness against user errors should never be sacrificed.
4. Rules for robustness :
  - a. **Protect against errors :** Software should protect itself against incorrect user input. Incorrect user input should never cause a crash.
  - b. **Optimize after the program runs :** Don't optimize a program until you get it working.
  - c. **Validate arguments :** External operations, those available to users of the class, must rigorously check their arguments to prevent failure. Don't include arguments that cannot be validated.
  - d. **Avoid predefined limits :** When possible use dynamic memory allocation to create data structures that do not have predefined limits.
  - e. **Instrument the program for debugging and performance monitoring :** You should instrument your code for debugging, statistics, and performance. The level of debugging that you must build into your code depends on the programming environment presented by the language.

## iii. Extensibility :

1. Extensibility is a software engineering and systems design principle that provides for future growth.
2. Extensibility is a measure of the ability to extend a system and the level of effort required to implement the extension.
3. Extensions can be through the addition of new functionality or through modification of existing functionality.
4. The principle provides for enhancements without impairing existing system functions.

**For example :** Object-oriented application frameworks which achieve extensibility typically by using inheritance and dynamic binding.

#### iv. Abstraction :

1. Abstraction is the selective examination of certain aspects of a problem.
2. The goal of abstraction is to isolate those aspects that are important for some purpose and suppress those aspects that are unimportant.
3. Abstraction must always be for some purpose, because the purpose determines what is and is not important.
4. Many different abstractions of the same thing are possible, depending on the purpose for which they are made.

#### b. i. How object-oriented concept can be implemented using non-object-oriented language? Explain with an example.

**Ans.** Implementing an object-oriented concept in a non-object oriented language requires the following steps :

##### 1. Translate classes into data structures :

- i. Each class is implemented as a single contiguous block of attributes. Each attribute contains variable. Now an object has state and identity and is subject to side effects.
- ii. A variable that identifier an object must therefore be implemented as a sharable reference.

##### 2. Pass arguments to methods :

- i. Every method has at least one argument. In a non-object-oriented language, the argument must be made explicit.
- ii. Methods can contain additional object as arguments. In passing an object as an argument to a method, a reference to the object must be passed if the value of the object can be updated within the method.

##### 3. Allocate storage for object :

- i. Objects can be allocated statically, dynamically or on a stack.
- ii. Most temporary and intermediate objects are implemented as stack-based variables.
- iii. Dynamically allocated objects are used when there number is not known at compile time.
- iv. A general object can be implemented as a data structure allocated on request at run time from a heap.

##### 4. Implement inheritance in data structures : Following ways are use to implement data structures for inheritance in a non-object-oriented language :

- i. Avoid it.
- ii. Flatten the class hierarchy.
- iii. Break out separate objects.

**5. Implement method resolution :** Method resolution is one of the main features of an object-oriented language that lacking in a non-object-oriented language. Method resolution can be implemented in following ways :

- i. Avoid it.
- ii. Resolve methods at compile time.
- iii. Resolve methods at run time.

**6. Implement associations :** Implementing associations in a non-object-oriented language can be done by :

- i. Mapping them into pointers.
- ii. Implementing them directly as association container objects.

**7. Deal with concurrency :**

- i. Most languages do not explicitly support concurrency.
- ii. Concurrency is usually needed only when more than one external event occurs, and the behaviour of the program depends on their timing.

**8. Encapsulate internal details of classes :**

- i. Object-oriented languages provide constructs to encapsulate implementation.
- ii. Some of this encapsulation is lost when object-oriented concept is translated into a non-object-oriented language, but we can still take advantage of the encapsulation facilities provided by the language.

**ii. What do you mean by documentation? What are the various considerations in documentation designing? Explain.**

**Ans. Documentation :**

1. Documentation is a software development process that records the procedure of making the software.
2. The design decisions need to be documented for any non-trivial software system for transmitting the design to others.
3. Though a secondary product, a good documentation is indispensable, particularly in the following areas :
  - a. In designing software that is being developed by a number of developers.
  - b. In iterative software development strategies.
  - c. In developing subsequent versions of a software project.
  - d. For evaluating a software.

- e. For finding conditions and areas of testing.
- f. For maintenance of the software.

**Various Consideration of documentation designing :**

**1. It is a roadmap :**

- a. It allows standardization, and it helps to identify the stages that can be improved.
- b. Process documentation also facilitates the training of new employees.

**2. It is everyone's task :**

- a. The members of an area or project are responsible for documenting their processes.
- b. Every employee knows their own functioning, their strength, and their weakness, so they are the ones better indicates to document their processes.

**3. Make them public :**

- a. The documentation of the processes must be available to all team and company members.
- b. Restricting access to documentation creates the false illusion that it's only relevant to a particular group.

**4. Flexible documentation :**

- a. Companies change, update, improve, so their processes are also subject to constant changes. To improve the effectiveness of the process, incorporate the necessary adjustments to the documentation of the process.
- b. Document the date of the last update.
- c. Save a backup copy of the files that document the process.
- d. Review the documents at least once a year.

**c. Write short notes on the following :**

- i. **Jackson Structured Development (JSD).**
- ii. **Dynamic modeling and Functional modeling.**

**Ans. Jackson structured development :**

- 1. Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
- 2. JSD can start from the stage in a project when there is only a general statement of requirements.
- 3. Following are the phases of JSD :

- a. **Modeling phase :** In the modeling phase, the designer creates a collection of entity structure diagrams and identifies the entities in the system, the actions they perform, the attributes of the actions and time ordering of the actions in the life of the entities.
- b. **Specification phase :** This phase focuses on actually what is to be done. Major goal is to map progress in the real world on progress in the system that models it.
- c. **Implementation phase :**
  - i. In the implementation phase JSD determines how to obtain the required functionality.
  - ii. Implementation way of the system is based on transformation of specification into efficient set of processes.

**Dynamic modeling :**

- 1. The dynamic model represents the time-dependent aspects of a system.
- 2. It is concerned with the temporal changes in the states of the objects in a system.
- 3. It is used to specify and implement the control aspect of the system.
- 4. Dynamic model is represented graphically with the help of state diagrams.
- 5. The dynamic model consists of multiple state diagrams and shows the pattern of activity for an entire system.

**Functional modeling :**

- 1. Functional modeling gives the process perspective of the object-oriented analysis model and an overview of what the system is supposed to do.
- 2. It defines the function of the internal processes in the system with the aid of Data Flow Diagrams (DFDs).
- 3. It depicts the functional derivation of the data values without indicating how they are derived when they are computed, or why they need to be computed.
- 4. Functional modeling is represented through a hierarchy of DFDs.
- 5. The DFD is a graphical representation of a system that shows the input of the system, the processing upon the inputs, the outputs of the system as well as the internal data stores.

4. Answer any **two** parts : (10 × 2 = 20)

- a. i. **Why Java is known as a platform independent language? Discuss the advantages and disadvantages of a platform independent language. Also give various data types in Java.**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. How polymorphism is handled in Java ? Explain with some suitable example using Java programming language.**

**Ans.** This question is out of syllabus from session 2020-21.

- b. i. Write a program in Java to count display the frequency of vowels in a given sentence of a at least 35 characters long.**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. Design class using Java to represent a student record having the following attributes and methods : (i) Attributes of the student Institute are as follows : Student\_ID, Student\_Name, Student\_Address, Birth\_Date, Course, Enrollment\_Year; (ii) The methods are as follows: to assign the initial values to all attributes, to add a new student record, display the list of students for a given year of enrollment and course.**

**Ans.** This question is out of syllabus from session 2020-21.

- c. Write short notes on the following giving their significance and with suitable example using Java in brief :**

**i. Enterprise Java Beans**

**ii. Java API's**

**Ans.** This question is out of syllabus from session 2020-21.

**5. Answer any two parts : (10 × 2 = 20)**

- a. i. What do you mean by Applets ? How Applets differ from the applications ? Explain with an example using Java.**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. Write a short note on Java Swing with suitable example.**

**Ans.** This question is out of syllabus from session 2020-21.

- b. Write short notes on the following with an example using Java : (i) JAR files (ii) Packages (iii) Multithreading (iv) Interface.**

**Ans.** This question is out of syllabus from session 2020-21.

- c. What do you mean by JDBC ? What is its significance ? How database connectivity is done using Java ? Discuss it with suitable example.**

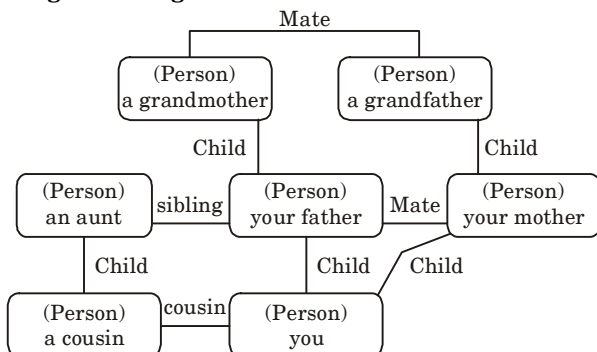
**Ans.** This question is out of syllabus from session 2020-21.





**B. Tech.****(SEM. V) ODD SEMESTER THEORY  
EXAMINATION, 2012-13  
OBJECT ORIENTED TECHNIQUES****Time : 3 Hours****Max. Marks : 100****Note :** 1. Attempt **all** questions.

2. Make suitable assumptions if required.

1. Answer any **two** parts :**(2 × 10 = 20)****a. i. What do you mean by object-oriented techniques ? Explain with some examples.****ii. Discuss the concept of encapsulation with suitable example.****iii. Describe the pros and cons of unified modeling language (UML).****b. i. Give the conceptual model of UML. Use some example to illustrate the model in detail using diagram.****ii. Define link and association. Discuss the role of link and association in object modeling with suitable example.****c. i. Define aggregation and generalization. Explain.****ii. What do you mean by polymorphism ? Explain it with an example.**2. Answer any **two** parts :**(2 × 10 = 20)****a. What is the difference between a class diagram and an instance diagram ? Discuss the significance of each. Also prepare a class diagram for the following instance diagram as given in Fig. 1.****Fig. 1.**

- b. Discuss the significance of sequence diagrams. How the following is implemented using sequence diagrams :**
  - i. Broadcast messages**
  - ii. Callback mechanism**
  - iii. Asynchronous messages with/without priority.**
- c. i. Write short notes on architectural modeling with suitable example and diagrams.**
- ii. Categorize the following relationship into generalization, aggregation, or association :**
  - 1. A country has a capital city**
  - 2. Files contain records.**
- 3. Answer any two parts : (2 × 10 = 20)**
  - a. Describe the following with example :**
    - i. Passing arguments to methods.**
  - ii. Features of object oriented languages.**
- iii. Implementation of inheritance.**
- b. i. What do you mean by the optimization of design ? Discuss the design optimization with suitable example using diagrams.**
- ii. Describe the structured analysis and structured design approach with an example.**
- c. Write short notes on the following :**
  - i. Compare procedural programming with object-oriented programming with examples.**
- ii. Write a short note on Jackson Structured Development (JSD).**
- 4. Answer any two parts : (2 × 10 = 20)**
  - a. What is the significance of data types in a programming language ? Describe the various data types used in Java. Also compare C++ and Java.**
- b. i. Write a program in java to display the longest word in a given sentence of at least having 9 words.**

- ii. **Design a class using Java to create a singly linked list. Then also write the methods for adding a node to the linked list in the beginning and also to search a given node.**
- c. **Write in short on the following with suitable example in Java and explaining their significance :**
  - i. **Session beans and Entity beans.**
  - ii. **Jave APIs**
- 5. **Answer any two parts : (2 × 10 = 20)**
  - a. **What do you understand by the database connectivity ? Give the connectivity model. Also discus JDBC in detail with an example.**
  - b. **Write applets to draw the following figures with proper syntax :**
    - i. **A triangle inside another triangle.**
    - ii. **A square inside another square.**
  - c. **Write short notes on the following with suitable examples giving their significance in application development :**
    - i. **AWT v/s Swing.**
    - ii. **Multithreading in Java.**



**SOLUTION OF PAPER (2012-13)**

**Note :** 1. Attempt **all** questions.  
2. Make suitable assumptions if required.

1. Answer any **two** parts : (2 × 10 = 20)

a. i. **What do you mean by object-oriented techniques ? Explain with some examples.**

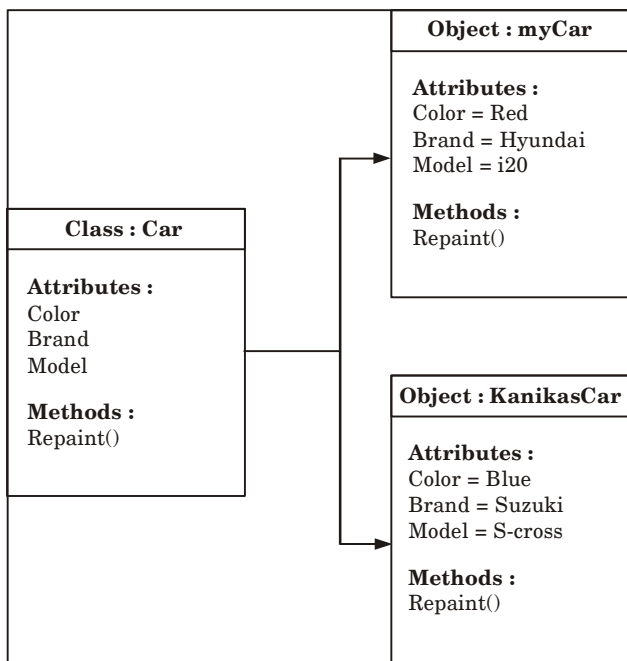
**Ans.**

1. Object-Oriented Techniques (OOT) is an approach to program organization and development that attempts to reduce some of the issues with conventional programming techniques.
2. It is a new way of organizing and developing programs and has nothing to do with any particular programming language.
3. However, not all languages are suitable to implement the object-oriented concepts or implement partial features of object-oriented concepts.
4. Object oriented programming paradigm relies on the concept of classes and objects.
5. A class is an abstract blueprint used to create more specific, concrete objects.
6. Classes often represent broad categories, like **Car** that share attributes. These classes define what attributes an instance of this type will have, like **color**.
7. Classes can also contain functions, called **methods** available only to objects of that type.

For example, our **Car** class may have a method **repaint** that changes the **color** attribute of our car. This function is only helpful to objects of type **Car**, so we declare it within the **Car** class thus making it a method.

8. Class templates are used as a blueprint to create individual **objects**. These represent specific examples of the abstract class, like **myCar**.
9. Each object can have unique values to the properties defined in the class.

For example, say we created a class, **Car**, to contain all the properties a car must have, **color**, **brand**, and **model**. We then create an instance of a **Car** type object, **myCar** to represent my specific car.



**Fig. 1.** Class blueprint being used to create two car type objects, myCar and KanikasCar.

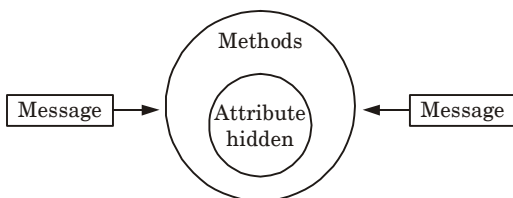
**ii. Discuss the concept of encapsulation with suitable example.**

**Ans.**

1. Encapsulation consists of separating the external aspects of an object, which are accessible to other objects, from the internal implementation details of the object, which are hidden from other objects.
2. Encapsulation prevents a program from becoming so interdependent that a small change has massive ripple effects.
3. The implementation of an object can be changed without affecting the applications that use it.
4. One may want to change the implementation of an object to improve performance, fix a bug, consolidate code, or for porting.
5. To understand encapsulation, let us consider the object 'Employee'.
6. The attributes of employees say 'salary' is kept hidden inside the object and may be made accessible only through the method meant for the purpose.
7. The method resides within the object.

For example, if `getSalary()` is a method of the object 'Employee' to get the salary of an employee, then the salary of an employee can be obtained by no other way but by this method.

8. Other objects can also send messages to the object 'Employee' and get the salary of an employee by the `getSalary()` method.
9. Other objects need not be concerned with the attributes and internal structure of the object.
10. This is shown in Fig. 2. The figure shows that attributes are hidden inside the object by a method.



**Fig. 2.** Encapsulation of an object.

### iii. Describe the pros and cons of unified modeling language (UML).

**Ans. Pros of UML :**

1. It has wide industry acceptance in comparison to previous modeling language.
2. It supports OOAD methodology.
3. It bridges the communication gap between different entities of system development (*i.e.* System Analyst, Developer, Client etc).
4. Constructed models are easy to understand, even for non-programmers.
5. It is a unified and standardize modeling language.

**Cons of UML :**

1. UML is often criticized as being large and complex.
2. It takes a lot of time to keep the diagram reasonable and synchronized with the actual code.
3. You cannot represent every condition in a sequence diagram.
4. UML software costs money.
5. Complex to learn and takes time to master properly.

### b. i. Give the conceptual model of UML. Use some example to illustrate the model in detail using diagram.

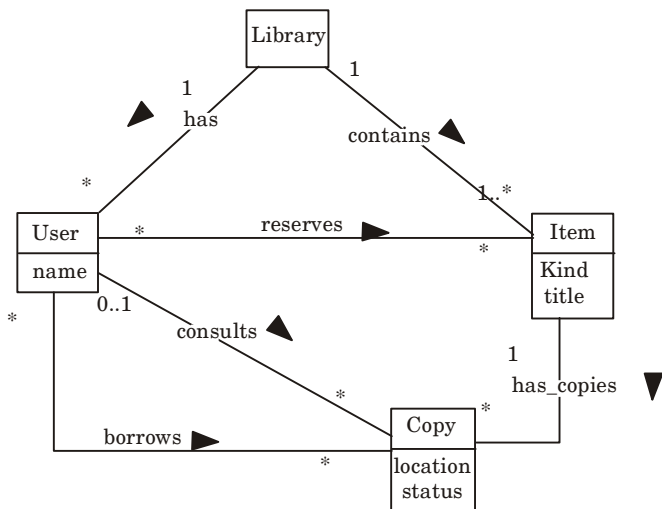
**Ans. UML :**

1. UML stands for Unified Modeling Language.
2. UML is a pictorial language used to make software blueprints.

**Conceptual model of UML :**

1. A conceptual model is defined as a model which is made of concepts and their relationships.

2. A conceptual model is the first step before drawing a UML diagram.
3. It helps to understand the entities in the real world and how they interact with each other.
4. The conceptual model of UML has three major elements :
  - a. UML building blocks.
  - b. Rules to connect the building blocks.
  - c. Common mechanisms of UML.
5. A domain model, often referred to as a conceptual model, might be represented by a particular kind of UML class diagram.
6. This model explains the structure of the application domain rather than the application structure itself.
7. It focuses on the domain concepts, rather than on the software entities.
8. Fig. 3 shows the conceptual model for a library system.



**Fig. 3.** Conceptual model for a library system.

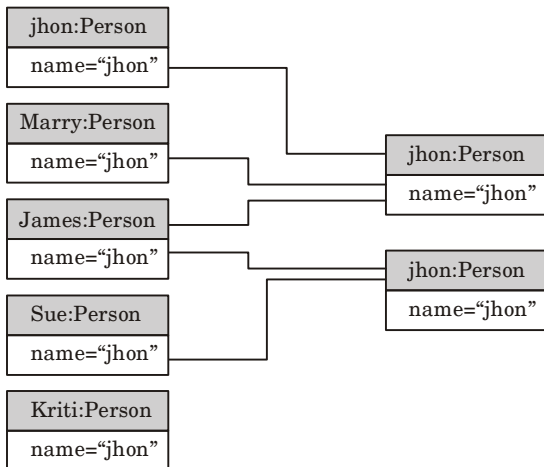
- ii. **Define link and association. Discuss the role of link and association in object modeling with suitable example.**

**Ans.**

1. Link and association in object modeling represent the relation between objects and classes.
2. **Link :** Link defines the relationship between two or more objects and a link is considered as an instance of an association.
3. **Association :** It is a group of links that relates objects from the same classes.

**4. For example :**

- Let us take the two classes Person and Company. Now there is an association relation between these two classes.
- A person may own stock in zero or more companies.
- Also it can be related in reverse that a company may have several persons owing its stock.
- The object diagram below shows the links between the objects of person and company class.



Links in Object Diagram

**Fig. 4.**

- The class diagram below shows the association between the person and the company class. Both link and association are represented with a line in UML notation.



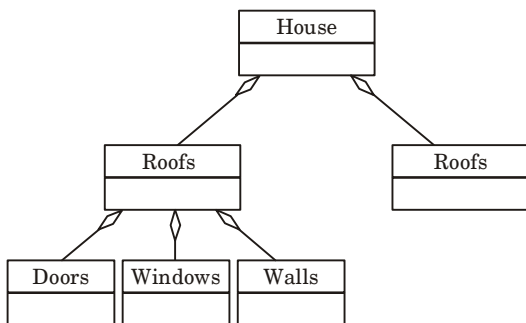
Association in class diagram

**Fig. 5.****c. i. Define aggregation and generalization. Explain.****Ans.****A. Aggregation :**

- Aggregation is a stronger form of association. It represents the **has-a** or **part-of** relationship.
- An aggregation association depicts a complex object that is composed of other objects.



3. For example, we may characterize a house in terms of its roof, floors, foundation, walls, rooms, windows, and so on. A room may, in turn be, composed of walls, ceiling, floor, windows, and doors, as represented in Fig. 6.

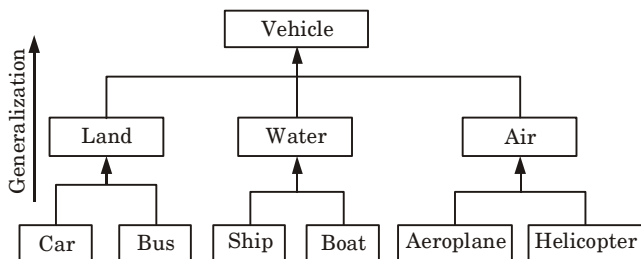


**Fig. 6.** A house and some of its component.

4. Hence object aggregation helps us describe models of the real world that are composed of other models, as well as those that are composed of still other models.

### B. Generalization :

- In the generalization process, the common characteristics of classes are combined to form a class in a higher level of hierarchy, *i.e.*, subclasses are combined to form a generalized super-class.
- It represents an “is – a – kind – of” relationship. For example, “car is a kind of land vehicle”, or “ship is a kind of water vehicle”.



**Fig. 7.**

- ii. What do you mean by polymorphism ? Explain it with an example.

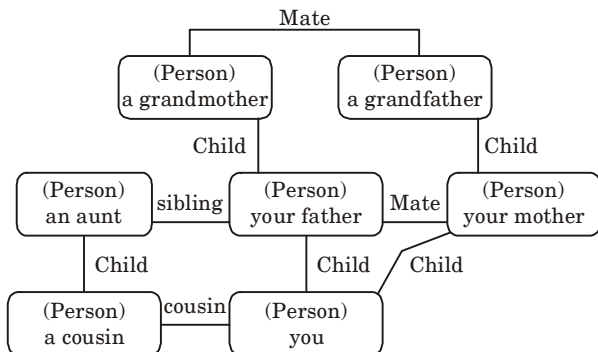
**Ans. Polymorphism :**

1. Polymorphism means having many forms.
2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.
4. An operation is a function that may be applied to or by objects in a class.
5. *Open, close, hide, and redisplay* are operations on class *Window*. All objects in a class share the same operations.
6. Each operation has a target object as an implicit argument.
7. The behavior of the operation depends on the class of its target.
8. An object “knows” its class, and hence the right implementation of the operation.
9. The same operation may apply to many different classes. Such an operation is polymorphic; *i.e.*, the same operation takes on different forms in different classes.
10. For example, the class *File* may have an operation *print*.
11. Different methods could be implemented to print ASCII files, print binary files, and print digitized picture files.
12. All these methods logically perform the same task. However, each method may be implemented by a different piece of code.

2. Answer any **two** parts :

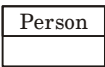

(2 × 10 = 20)

- a. **What is the difference between a class diagram and an instance diagram ? Discuss the significance of each. Also prepare a class diagram for the following instance diagram as given in Fig. 8.**



**Fig. 8.**

**Ans.****A. Difference :**

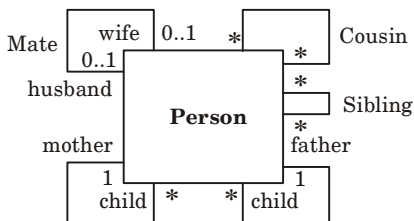
| S. No. | Class diagram                                                                                                                                                                                                                | Instance diagram                                                                                                                                                                                                                                                                                     |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.     | A class diagram is a schema, pattern, or template for describing many possible instances of data.                                                                                                                            | An instance diagram describes how does a particular set of object relates to each other.                                                                                                                                                                                                             |
| 2.     | A class diagram describes the general case in modeling a system.                                                                                                                                                             | An instance diagram describes object instances.                                                                                                                                                                                                                                                      |
| 3.     | A class diagram describes object classes.                                                                                                                                                                                    | An instance diagram is useful for documenting test cases, especially scenarios, and is used to show examples to help to clarify a complex class diagram.                                                                                                                                             |
| 4.     | <p>The OMT symbol for a class is a rectangular box with class name in boldface. A line is drawn between the class name and attributes.</p>  | <p>Figure below shows the OMT representation of instance diagram. The class name in parenthesis is at the top of the object box in boldface and object names are listed in normal font with their attributes.</p>  |

**B. Significance of class diagram :**

1. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
2. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.
3. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

**C. Significance of instance diagram :**

1. An object diagram represents an instance at a particular moment, which is concrete in nature.
2. It means the object diagram is closer to the actual system behaviour.
3. The purpose is to capture the static view of a system at a particular moment.

**D. Class Diagram :****Fig. 9.** Class diagram for family trees.

**b. Discuss the significance of sequence diagrams. How the following is implemented using sequence diagrams :**

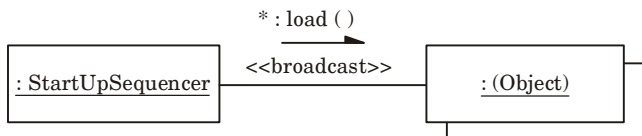
- i. Broadcast messages
- ii. Callback mechanism
- iii. Asynchronous messages with/without priority.

**Ans. Significance of sequence diagram :**

- i. Sequence diagrams are one of the important dynamic modeling techniques in the UML.
- ii. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

**i. Broadcast messages :**

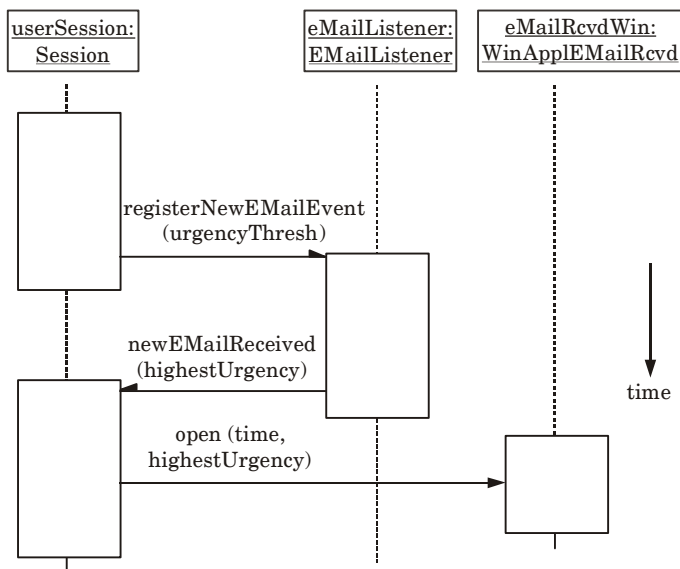
1. A broadcast message treats every object in the system as a potential target.
2. A copy of the message goes into the queue of every object in the system.
3. An object may broadcast a message in response to some external event that it detects.
4. For example, an object might detect a security compromise and broadcast to all objects the need for a priority system shutdown.
5. Fig. 10 shows the UML for a broadcast message. Here, a start-up sequencer is getting every object in the system that exists at start-up time to load itself.

**Fig. 10.** A broadcast message.

**ii. Callback mechanism :**

1. Here the subscriber object registers an interest in some event via an asynchronous message to the target object.
2. The target object continues with other activities while it monitors for the occurrence of an event of the registered type.

3. When an event of that type occurs, the target object sends a message back to the subscriber object to notify of the occurrence. This is known as callback mechanism.
4. The sequence diagram in Fig. 11 shows an example of the callback mechanism, where the event of interest is the arrival of e-mail with an urgency above a certain threshold.



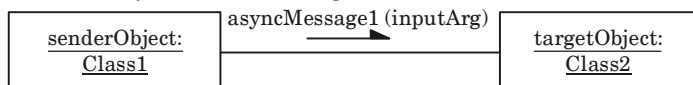
**Fig. 11.** Callback mechanism used to detect e-mail.

5. The **userSession** object registers its interest in new e-mail with the **eMailListener** object by sending the message **registerNewEMailEvent (urgencyThresh)**.
6. When e-mail that's urgent enough arrives, **eMailListener** calls back **userSession** with **newEMailReceived (highestUrgency)**, whose argument indicates the highest urgency among the messages that actually arrived.

### iii.

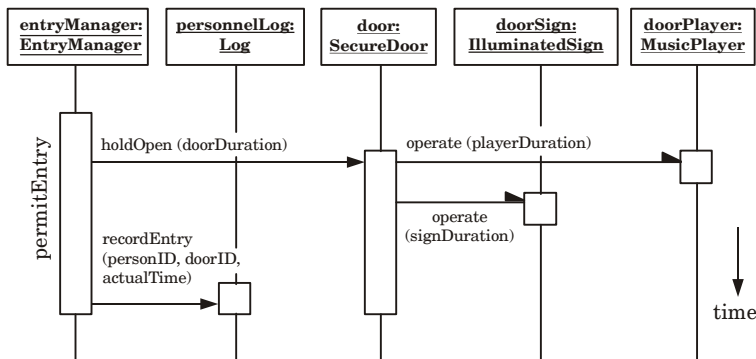
#### A. Asynchronous messages without priority :

1. An asynchronous message can be implemented on the sequence diagram by using a half arrowhead on the asynchronous message arrow.



**Fig. 12.** A collaboration diagram showing a basic asynchronous message.

2. When an asynchronous message is send, at least to loci of execution are active in the system, because the target begins to execute while the sender remains in execution.
3. Fig. 13 shows a specific example from a real-time system that authorizes personnel to pass through electronically controlled doors.
4. The employee inserts an ID card into a reader, and if the employee is authorized to enter, the system plays jingly music, displays a greeting and slides the door open.



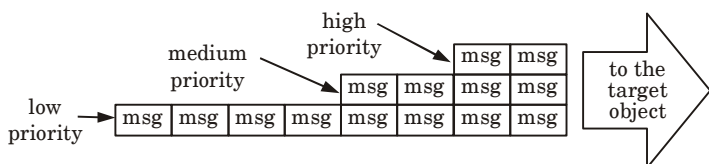
**Fig. 13.** A sequence diagram for concurrently executing objects.

5. The operation that manages this piece of the application is **permitEntry**.
6. After determining that the employee is allowed through the door **permitEntry** sends the synchronous message **holdOpen** to the object **door**.
7. The operation **holdOpen** then handles the sound, lights, and action associated with opening the door.
8. For the action of opening the door, **holdOpen** sends messages to a hardware driver.
9. For the sound-and-light show, **holdOpen** sends two asynchronous messages, both named **operate** : one to **doorPlayer** and one to **doorSign**.
10. The two operations named **operate** execute concurrently: **door**.

#### **B. Asynchronous message with priority :**

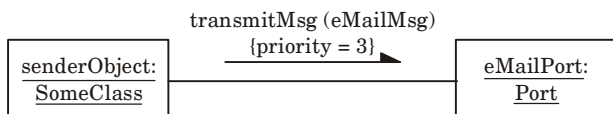
1. In concurrency the target object receives message from lots of concurrently executing sender objects.
2. Since these messages may arrive faster than the target can process them, they are ushered into the message queue.

3. The object then removes a message from the front of the queue; process the message, and takes the next message from the queue.
4. These messages in a queue are ordered by priority.
5. Fig. 14 shows a set of parallel queues at the target, each queue with its own priority level.



**Fig. 14.** Three parallel queues, each with its own priority.

6. Fig. 15 shows an asynchronous message with its priority level.
7. The property {priority = 3} indicates that the message has a priority of 3.



**Fig. 15.** An asynchronous message (with priority 3) going to an object with a multiple-priority message queue.

### c. i. Write short notes on architectural modeling with suitable example and diagrams.

#### **Ans.** Architectural modeling :

1. Architectural modeling represents the overall framework of the system.
2. It contains both structural and behavioral elements of the system.
3. Architectural modeling can be defined as the blueprint of the entire system.

#### **Diagrams used in architectural modeling :**

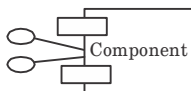
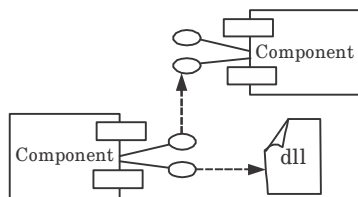
1. The two types of diagrams that give descriptions of the physical information about a system are deployment diagrams and component diagrams.
2. Deployment diagrams show the physical relationship between hardware and software in a system.
3. Component diagrams show the software components of a system and their relationships.
4. These relationships are called dependencies.

**A. Component diagrams :**

1. The component diagrams are mainly used to model the static implementation view of a system.
2. They represent a high-level packaged view of the code.
3. They can be used to model executables, databases and adaptable systems.
4. Component diagrams mainly contain the following :

**i. Components :**

- a. A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.
- b. A component is a physical manifestation of an object that has a well-defined interface and a set of implementations for the interface.
- c. A complex system can be built using software components. It enhances re-use in the system and facilitates system evolution.

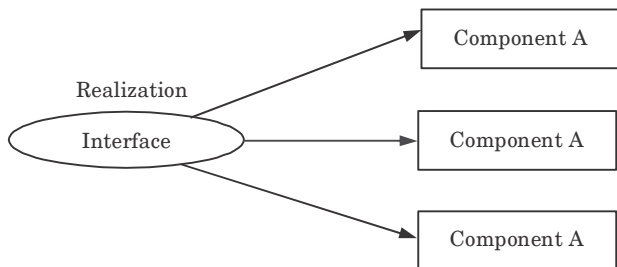
**Fig. 16.** Component.**Fig. 17.** Component diagram.

- ii. Interfaces :** An interface is a collection of operations that are used to specify a service of a class or a component. It is represented by a circle. An interface possesses the following properties :

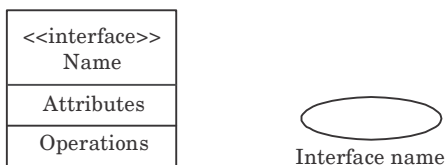
- a. Every interface is identified by a unique name, with an operational pathname.
- b. Interfaces do not specify any structure or implementation details of the operations. When an interface is represented as a rectangle, it has the same parts as a class. When it is



represented as a circle, the display of these parts is suppressed.

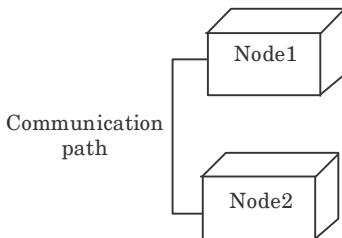


**Fig. 18.** Realization of an interface.



**Fig. 19.** Representation of interfaces.

**B. Deployment diagrams :** They display the configuration of run-time processing elements and the software components, processes and objects. The deployment diagram contains nodes and connections. A node is a piece of hardware in the system. A connection depicts the communication path used by the hardware Fig. 20.



**Fig. 20.** Deployment diagram.

ii. Categorize the following relationship into generalization, aggregation, or association :

1. A country has a capital city
2. Files contain records.

**Ans.**

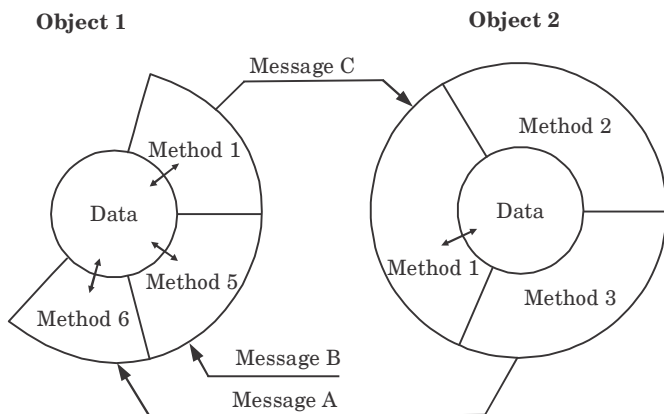
1. **A country has a capital city :** It is an association relationship. A capital city and a country are distinct things so generalization certainly does not apply. You could argue that a capital city is a part of a country and thus they are related by aggregation.
2. **Files contain records :** It is an aggregation relationship. The word “contain” is a clue that the relationship may be aggregation. A record is a part of a file. Some attributes and operations on files propagate to their constituent records.
3. **Answer any two parts :** (2 × 10 = 20)
  - a. **Describe the following with example :**
    - i. **Passing arguments to methods.**

**Ans.**

1. There are different ways in which parameter data can be passed into and out of methods and functions.
2. Let us assume that a function B() is called from another function A(). In this case A is called the “caller function” and B is called the “called function or callee function”.
3. The arguments which A sends to B are called actual arguments and the parameters of B are called formal arguments.
4. Following are the types of parameters:
  - a. **Formal parameter :** A variable and its type as they appear in the prototype of the function or method.  
**Syntax :** function\_name(datatype variable\_name) :
  - b. **Actual parameter :** The variable or expression corresponding to a formal parameter that appears in the function or method call in the calling environment.  
**Syntax :** func\_name(variable name(s));

**ii. Features of object oriented languages.****Ans. Features of object-oriented language are :****1. Encapsulation :**

- i. Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.
- ii. The only way to reach the data is through these particular methods (see Fig. 21).
- iii. It is the mechanism that binds together code and the data it manipulates.
- iv. This concept is also used to hide the internal representation, or state, of an object from the outside.



**Fig. 21.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

## 2. Polymorphism :

- Polymorphism means having many forms.
- Polymorphism is the ability of a message to be displayed in more than one form.
- It plays an important role in allowing objects having different internal structure to share the same external interface.

## 3. Inheritance :

- Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features.
- Inheritance is mainly used for code reusability.

### iii. Implementation of inheritance.

**Ans.**

- Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship.
- Object-oriented languages provide strong support for implementation of inheritance.
- Inheritance has become synonymous with code reuse within the object-oriented programming community.
- After modeling a system, the developer looks at the resulting classes and tries to group similar classes together and reuse common code.
- Often code is available from past work (such as a class library) which the developer can reuse and modify to get the precise desired behavior.

**b. i. What do you mean by the optimization of design ? Discuss the design optimization with suitable example using diagrams.**

**Ans.** **Optimization of design :**

1. Design optimization is an engineering design methodology using a mathematical formulation of a design problem to support selection of the optimal design among many alternatives.
2. Design optimization involves the following stages :
  - a. **Variables** : Describe the design alternatives.
  - b. **Objective** : Elected functional combination of variables (to be maximized or minimized).
  - c. **Constraints** : Combination of variables expressed as equalities or inequalities that must be satisfied for any acceptable design alternative.
  - d. **Feasibility** : Values for set of variables that satisfies all constraints and minimizes/maximizes objective.

**For example :** Consider the design of a company's employee skills database. Fig. 22 shows a portion of the analysis class model. The operation `companyfindskill()` returns a set of persons in the company with a given skill. For example, an applications might need all the employees who speak Japanese.



**Fig. 22.**

**ii. Describe the structured analysis and structured design approach with an example.**

**Ans.** **Structured analysis and structure design :**

1. Structured Analysis and Structured Design (SA/SD) is diagrammatic notation which is design to help people understand the system.
2. The basic goal of SA/SD is to improve quality and reduce the risk of system failure.
3. It establishes concrete management specification and documentation.
4. It focuses on solidity, pliability and maintainability of system.
5. The approach of SA/SD is based on the Data Flow Diagram.
6. It is easy to understand SA/SD but it focuses on well defined system boundary whereas JSD approach is too complex and does not have any graphical representation.
7. SA/SD is combined known as SAD and it mainly focuses on following three points :

- a. System
  - b. Process
  - c. Technology
8. SA/SD involves two phases :
- a. Analysis Phase :** It uses Data Flow Diagram, Data Dictionary, State Transition diagram and ER diagram.
  - b. Design Phase :** It uses Structure Chart and Pseudo Code.
- c. Write short notes on the following :**
- i. Compare procedural programming with object-oriented programming with examples.**

**Ans.**

| S.No. | Procedural Oriented Programming                                                           | Object-Oriented Programming                                                            |
|-------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 1.    | In procedural programming, program is divided into small parts called functions.          | In object-oriented programming, program is divided into small parts called objects.    |
| 2.    | Procedural programming follows top down approach.                                         | Object-oriented programming follows bottom up approach.                                |
| 3.    | There is no access specifier in procedural programming.                                   | Object-oriented programming has access specifiers like private, public, protected etc. |
| 4.    | Adding new data and function is not easy.                                                 | Adding new data and function is easy.                                                  |
| 5.    | Procedural programming does not have any proper way for hiding data so it is less secure. | Object-oriented programming provides data hiding so it is more secure.                 |
| 6.    | In procedural programming, overloading is not possible.                                   | Overloading is possible in object-oriented programming.                                |
| 7.    | In procedural programming, function is more important than data.                          | In object-oriented programming, data is more important than function.                  |
| 8.    | Procedural programming is based on unreal world.                                          | Object-oriented programming is based on real world.                                    |
| 9.    | <b>Examples :</b> C, FORTRAN, Pascal, Basic etc.                                          | <b>Examples :</b> C++, Java, Python, C# etc.                                           |

- ii. Write a short note on Jackson Structured Development (JSD).**

**Ans. Jackson structured development :**

1. Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
2. JSD can start from the stage in a project when there is only a general statement of requirements.
3. Following are the phases of JSD :
  - a. **Modeling phase :** In the modeling phase, the designer creates a collection of entity structure diagrams and identifies the entities in the system, the actions they perform, the attributes of the actions and time ordering of the actions in the life of the entities.
  - b. **Specification phase :** This phase focuses on actually what is to be done. Major goal is to map progress in the real world on progress in the system that models it.
  - c. **Implementation phase :**
    - i. In the implementation phase JSD determines how to obtain the required functionality.
    - ii. Implementation way of the system is based on transformation of specification into efficient set of processes.

4. Answer any **two** parts : (2 × 10 = 20)

- a. **What is the significance of data types in a programming language ? Describe the various data types used in Java. Also compare C++ and Java.**

**Ans.** This question is out of syllabus from session 2020-21.

- b.i. **Write a program in java to display the longest word in a given sentence of at least having 9 words.**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. **Design a class using Java to create a singly linked list. Then also write the methods for adding a node to the linked list in the beginning and also to search a given node.**

**Ans.** This question is out of syllabus from session 2020-21.

- c. **Write in short on the following with suitable example in Java and explaining their significance :**

- i. **Session beans and Entity beans.**
- ii. **Jave APIs**

**Ans.** This question is out of syllabus from session 2020-21.

**5. Answer any two parts : (2 × 10 = 20)**

- a. What do you understand by the database connectivity ? Give the connectivity model. Also discuss JDBC in detail with an example.**

**Ans.** This question is out of syllabus from session 2020-21.

- b. Write applets to draw the following figures with proper syntax :**

- i. A triangle inside another triangle.**
- ii. A square inside another square.**

**Ans.** This question is out of syllabus from session 2020-21.

- c. Write short notes on the following with suitable examples giving their significance in application development :**

- i. AWT v/s Swing.**
- ii. Multithreading in Java.**

**Ans.** This question is out of syllabus from session 2020-21.



**B. Tech.****(SEM. V) ODD SEMESTER THEORY  
EXAMINATION, 2013-14  
OBJECT ORIENTED TECHNIQUES****Time : 3 Hours****Max. Marks : 100****Note :** Attempt all questions.

1. Attempt any **four** questions : (5 × 4 = 20)
  - a. Explain the major features of Object-Oriented Programming.
  - b. What is the difference between Procedure Based programming language and Object-Oriented programming language ?
  - c. What do you understand by object identity ? Explain with an example.
  - d. What are the principles of modeling ? What is the importance of modeling ?
  - e. What is UML ? Mention the different kinds of modeling diagrams used.
  - f. Explain use case with example. How are the diagrams divided ?
2. Attempt any **four** parts : (5 × 4 = 20)
  - a. Explain class and object diagrams with examples.
  - b. Explain Polymorphism, Iterated Messages and use of self in message in collaboration diagram.
  - c. Explain sequence diagrams with example.
  - d. What do you understand by callback mechanisms ?
  - e. What do you understand by basic behavioural modeling ?
  - f. What are package diagrams and why are they used ?



3. Attempt any **two** parts : (10 × 2 = 20)
- a. What are the three models in OMT ? How is the object oriented analysis and design attached with OMT ? Explain with an example.
- b. Describe documenting design considerations. How do we perform adjustment of inheritance ?
- c. Write short notes on :
- i. SA/SD and JSD
- ii. Procedural v/s OOP.
4. Attempt any **two** questions : (10 × 2 = 20)
- a. What is Abstraction ? Explain Abstract method and Abstract class. Write a Java program for Employee class where salary is an abstract method with full implementation.
- b. Explain Multithreading in Java. What is the effect in program when we use multithreading in a program ? Write a Java program for  $z = \sin(x) + \cos(y)$  using multithreading.
- c. Write short notes on :
- i. Event Handling
- ii. EJB
5. Attempt any **two** questions : (10 × 2 = 20)
- a. What is Applet ? Explain life cycle of an applet with methods declaration. Write a Java program which shows a House.
- b. Explain Swing. What is the role of layout managers in swing ? Write a Java program which shows swing application.
- c. Write short notes on :
- i. JDBC ii. Servlet



**SOLUTION OF PAPER (2013-14)**

**Note :** Attempt **all** questions.

1. Attempt any **four** questions :

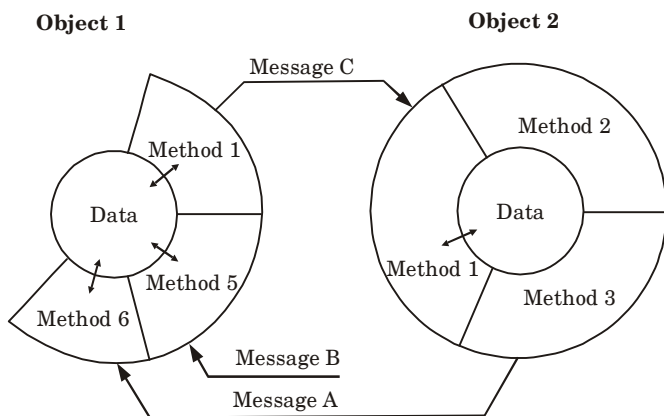
(5 × 4 = 20)

a. **Explain the major features of Object-Oriented Programming.**

**Ans.** Features of object-oriented language are :

1. **Encapsulation :**

- Encapsulation means that data are encapsulated inside an inviolable shell along with the methods required to use it.
- The only way to reach the data is through these particular methods (see Fig. 1).
- It is the mechanism that binds together code and the data it manipulates.
- This concept is also used to hide the internal representation, or state, of an object from the outside.



**Fig. 1.** Message passing between objects with encapsulation, activating methods that can use or modify the data within their object.

2. **Polymorphism :**

- Polymorphism means having many forms.
- Polymorphism is the ability of a message to be displayed in more than one form.

- iii. It plays an important role in allowing objects having different internal structure to share the same external interface.

### 3. Inheritance :

- a. Inheritance is the ability to create classes that share the attributes and methods of existing classes, but with more specific features.
- b. Inheritance is mainly used for code reusability.

### b. What is the difference between Procedure Based programming language and Object-Oriented programming language ?

**Ans.**

| S.No. | Procedural Oriented Programming                                                           | Object-Oriented Programming                                                            |
|-------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 1.    | In procedural programming, program is divided into small parts called functions.          | In object-oriented programming, program is divided into small parts called objects.    |
| 2.    | Procedural programming follows top down approach.                                         | Object-oriented programming follows bottom up approach.                                |
| 3.    | There is no access specifier in procedural programming.                                   | Object-oriented programming has access specifiers like private, public, protected etc. |
| 4.    | Adding new data and function is not easy.                                                 | Adding new data and function is easy.                                                  |
| 5.    | Procedural programming does not have any proper way for hiding data so it is less secure. | Object-oriented programming provides data hiding so it is more secure.                 |
| 6.    | In procedural programming, overloading is not possible.                                   | Overloading is possible in object-oriented programming.                                |
| 7.    | In procedural programming, function is more important than data.                          | In object-oriented programming, data is more important than function.                  |
| 8.    | Procedural programming is based on unreal world.                                          | Object-oriented programming is based on real world.                                    |
| 9.    | <b>Examples :</b> C, FORTRAN, Pascal, Basic etc.                                          | <b>Examples :</b> C++, Java, Python, C# etc.                                           |

- c. **What do you understand by object identity ? Explain with an example.**

**Ans. Object identity :**

1. Object identity is a property of data that is created in the context of an object data model, where an object is assigned a unique internal object identifier, or object ID.
2. The object identifier is used to define associations between objects and to support retrieval and comparison of object-oriented data based on the internal identifier rather than the attribute values of an object.
3. There are many techniques for identifying objects in programming languages.
4. OO languages have built-in mechanisms for identifying objects. There is no need to create explicit object identifier types.
5. For example : In C++ an objects actual memory address serves as a unique identifier and can be obtained by applying the '&' operator to an object or object reference.
6. Object identity can be tested by pointer comparison.

- d. **What are the principles of modeling ? What is the importance of modeling ?**

**Ans. Principles of modeling are :**

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 correct model as per the requirement of problem statement.
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 non-trivial system is best approached through a small set of nearly independent models :** This means we 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.

**Importance of modeling :**

Modeling help the development team better to visualize the plan of their system and allow them to develop more rapidly by helping them build the right thing.

- e. **What is UML ? Mention the different kinds of modeling diagrams used.**

**Ans. UML :**

1. UML stands for Unified Modeling Language.
  2. UML is a pictorial language used to make software blueprints.
  3. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
  4. It is also used to model non-software systems as well.
  5. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams.
  6. UML has a direct relation with object-oriented analysis and design.
- Different kinds of modeling diagrams used are :

**A. Structural modeling diagrams :**

1. **Class diagram :** Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.
2. **Component diagram :** A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components.
3. **Deployment diagram :** A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration.
4. **Object diagram :** Object diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects but they use real-world examples.
5. **Package diagram :** A package diagram shows the dependencies between different packages in a system.

**B. Behavioral modeling diagrams :**

1. **Use case diagram :** Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.
2. **Activity diagram :** Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system.
3. **State machine diagram :** State machine diagrams are very useful to describe the behavior of objects that act differently according to the state they are in at the moment.

4. **Sequence diagram** : Sequence diagrams show how objects interact with each other and the order in which those interactions occur.
5. **Communication diagram** : Communication diagrams are similar to sequence diagrams, but the focus is on messages passed between objects.

f. **Explain use case with example. How are the diagrams divided?**

**Ans.** Use case diagrams :

1. Use cases describe how a system interacts with external user of a system (*i.e.*, actor).
2. Each use case represents a piece of functionality that a system provides to its users.
3. Use cases are helpful for capturing informal requirements.
4. Use case consists of actors.
5. An actor is an object or set of objects that communicates directly with the system but that is not part of the system.
6. The various interactions of actors with a system are quantized into use cases.
7. A use case is a coherent piece of functionality that a system can provide by interacting with actors.
8. For example, a customer actor can buy a beverage from a vending machine. The customer inserts money into the machine, makes a selection, and ultimately receives a beverage. Similarly, a repair technician can perform scheduled maintenance on a vending machine.
9. Fig. 2 summarizes various use cases for a vending machine.

- **Buy a beverage** : The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance** : A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs** : A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items** : A stock clerk adds items into the vending machine to replenish its stock of beverages.

**Fig. 2.** Use case summaries for a vending machine.

**Use case diagrams are divided as :**

1. Use case diagram are divided into three use cases diagrams *i.e.*, Order, SpecialOrder, NormalOrder and one actor which is the customer.
2. The SpecialOrder and NormalOrder use cases are extended from Order use case.
3. Hence, they have extended relationship.
4. The actor Customer lies outside the system as it is an external user of the system.

2. Attempt any **four** parts :

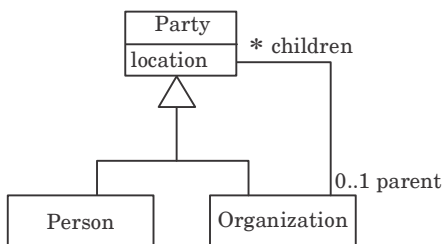
(5 × 4 = 20)

a. **Explain class and object diagrams with examples.**

**Ans. Class diagram :**

1. Class diagram is a static diagram.
2. It represents the static view of an application.
3. Class diagram is used for visualizing, describing, and documenting different aspects of a system and also for constructing executable code of the software application.
4. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
5. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

**For example :**

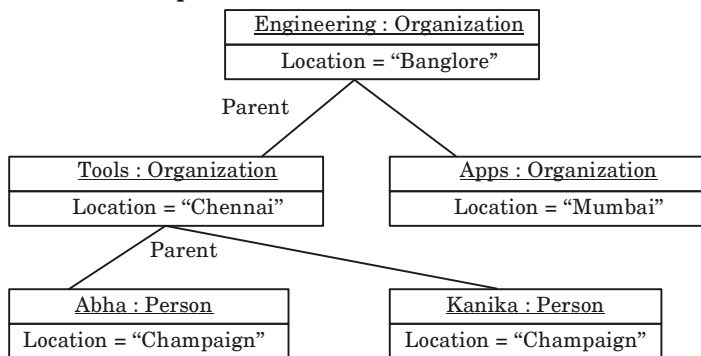


**Fig. 3.** Class diagram of party composition structure.

**Object diagram :**

1. Object diagrams represent an instance of a class diagram.
2. Object diagrams represent the static view of a system but this static view is a snapshot of the system at a particular moment.
3. Object diagrams are used to render a set of objects and their relationships as an instance.

**For example :**



**Fig. 4.** Object diagram showing example instances of party.

**b. Explain Polymorphism, Iterated Messages and use of self in message in collaboration diagram.**

**Ans. Polymorphism :**

1. Polymorphism means having many forms.
2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.

**Iterated messages :**

1. Collaboration diagrams use syntax similar to sequence diagrams to indicate that either a message iterates (is run multiple times) or is run conditionally.
2. We can indicate that a particular message iterates by prefixing a message sequence number with an iteration expression.
3. We can simply use an asterisk (\*) to indicate that a message runs more than once, or we can get more specific and show the number of times a message is repeated.
4. To indicate that a message is run conditionally, we can prefix the message sequence number with a conditional clause such as [x = true].
5. This indicates that the message is sent only if the condition is met.
6. The UML leaves the syntax of conditional clauses wide open, so we can create expressions that make sense in the context of our application.

**Use of self in message :**

1. Self represents the ability of an object to send a message to itself.
2. Messages in collaboration diagrams are shown as arrows pointing from the client object to the supplier object.



3. Messages represent a client invoking an operation on a supplier object.
4. Message icons have one or more messages associated with them.
5. Messages are composed of message text prefixed by a sequence number.
6. This sequence number indicates the time-ordering of the message.

**c. Explain sequence diagrams with example.**

**Ans. Sequence diagram :**

1. Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.
2. They're also called event diagrams.
3. A sequence diagram is a good way to visualize and validate various runtime scenarios.
4. In UML it is shown as a table that shows objects arranged along the X axis and messages along the Y axis.
5. It has a global life line and the focus of control.

**Various terms and symbols used in sequence diagram :**


1. **Class roles or Participants :** Class roles describe the way an object will behave in context.



Object

**Fig. 5.**


2. **Activation or Execution Occurrence :** Activation occurrence represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



Activation  
occurrence

**Fig. 6.**

3. **Messages :** Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



Asynchronous  
message

Synchronous  
message

**Fig. 7.**

4. **Lifelines :** Lifelines are vertical dashed lines that indicate the object's presence over time.

Lifeline

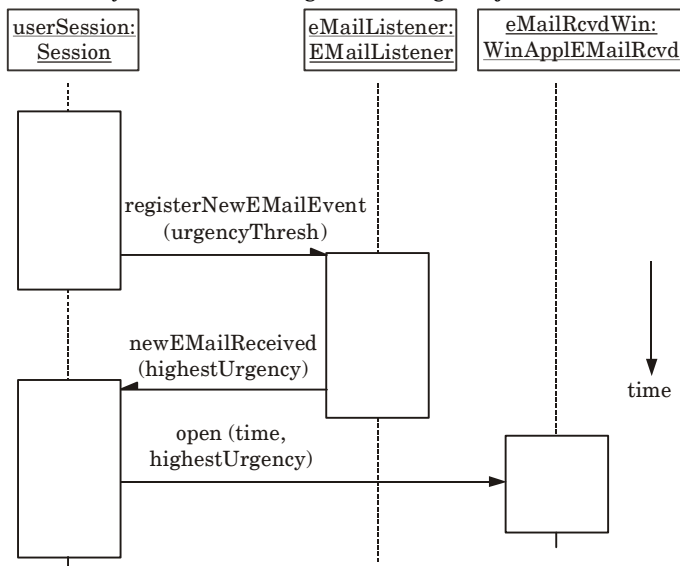
**Fig. 8.**

- 5. Destroying objects :** Objects can be terminated early using an arrow labeled “<< destroy >>” that points to an X. This object is removed from memory. When that object's lifeline ends, we can place an X at the end of its lifeline to denote a destruction occurrence.
- 6. Loops :** A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets.

**d. What do you understand by callback mechanisms ?**

**Ans. Callback mechanism :**

1. Here the subscriber object registers an interest in some event via an asynchronous message to the target object.



**Fig. 9.** Callback mechanism used to detect e-mail.

2. The target object continues with other activities while it monitors for the occurrence of an event of the registered type.
3. When an event of that type occurs, the target object sends a message back to the subscriber object to notify of the occurrence. This is known as callback mechanism.

4. The sequence diagram in Fig. 9 shows an example of the callback mechanism, where the event of interest is the arrival of e-mail with an urgency above a certain threshold.
5. The **userSession** object registers its interest in new e-mail with the **emailListener** object by sending the message **registerNewEMailEvent (urgencyThresh)**.
6. When e-mail that's urgent enough arrives, **eMailListener** calls back **userSession** with **newEMailReceived (highestUrgency)**, whose argument indicates the highest urgency among the messages that actually arrived.

**e. What do you understand by basic behavioural modeling ?**

**Ans.**

- i. Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.
- ii. During analysis, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.
- iii. In the design and implementation phases, the detailed design of the operations contained in the object is fully specified.
- iv. There are two types of behavioral models. First, there are behavioral models that are used to represent the underlying details of a business process portrayed by a use case model. In UML, interaction diagrams (sequence and communication) are used for this type of behavioral model.
- v. Second, there is a behavioral model that is used to represent the changes that occur in the underlying data. UML uses behavioral state machines for this.
- vi. During the analysis phase, analysts use behavioral model to capture a basic understanding of the dynamic aspects of the underlying business process.
- vii. Traditionally, behavioral models have been used primarily during the design phase where analysts refine the behavioral models to include implementation details.

**f. What are package diagrams and why are they used ?**

**Ans.**

1. Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages.
2. A package is a grouping of related UML elements, such as diagrams, documents, classes, or even other packages.

3. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram.
4. Package diagrams are used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

**Following are the uses of package diagram :**

1. Package diagrams are used to structure high level system elements.
2. Packages are used for organizing large system which contains diagrams and documents.
3. Package diagram can be used to simplify complex class diagrams; it can group classes into packages.

3. Attempt any **two** parts : (10 × 2 = 20)

a. **What are the three models in OMT ? How is the object oriented analysis and design attached with OMT ? Explain with an example.**

**Ans.** Following are the three models in OMT :

**1. Object model :**

- a. Object model encompasses the principles of abstraction, encapsulation, modularity, hierarchy, typing, concurrency and persistence.
- b. Object model emphasizes on the object and class.
- c. Main concepts related with object model are classes and their association with attributes.
- d. Predefined relationships in object model are aggregation and generalization (multiple inheritance).

**2. Dynamic model :**

- a. Dynamic model involves states, events and state diagram (transition diagram) on the model.
- b. Main concepts related with dynamic model are states, transition between states and events to trigger the transitions.
- c. Predefined relationships in object model are aggregation (concurrency) and generalization.

**3. Functional model :**

- a. Functional Model focuses on the how data is flowing, where data is stored and different processes.
- b. Main concepts involved in functional model are data, data flow, data store, process and actors.

- c. Functional model describes the whole processes and actions with the help of data flow diagram (DFD).

**OOAD attachment with OMT :**

1. Object Modeling Technique (OMT) combines the three views of modeling systems.
2. The object model represents the static, structural, “data” aspects of a system.
3. The dynamic model represents the temporal, behavioral, “control” aspects of a system.
4. The functional model represents the transformational, “function” aspects of a system.
5. A typical object oriented software procedure incorporates all three aspects : It uses data structures (object model), it sequences operations in time (dynamic model), and it transforms values (functional model). Each model contains references to entities in other models.
6. For example, operations are attached to objects in the object model but more fully expanded in the functional model.

**b. Describe documenting design considerations. How do we perform adjustment of inheritance ?**

**Ans.** Documentation is a software development process that records the procedure of making the software.

**Various considerations of documentation designing are :**

**1. It is a roadmap :**

- a. It allows standardization, and it helps to identify the stages that can be improved.
- b. Process documentation also facilitates the training of new employees.

**2. It is everyone's task :**

- a. The members of an area or project are responsible for documenting their processes.
- b. Every employee knows their own functioning, their strength, and their weakness, so they are the ones better indicates to document their processes.

**3. Make them public :**

- a. The documentation of the processes must be available to all team and company members.

- b. Restricting access to documentation creates the false illusion that it's only relevant to a particular group.

#### 4. Flexible documentation :

- a. Companies change, update, improve, so their processes are also subject to constant changes. To improve the effectiveness of the process, incorporate the necessary adjustments to the documentation of the process.
- b. Document the date of the last update.
- c. Save a backup copy of the files that document the process.
- d. Review the documents at least once a year.

Following kinds of adjustments can be used to increase the chance of inheritance :

1. Some operations may have fewer arguments than others. The missing arguments can be added but ignored. For example, a draw operation on a monochromatic display does not need a color parameter, but the parameter can be accepted and ignored for consistency with color displays.
2. Similar attributes in different classes may have different names. Give the attributes the same name and move them to the common ancestor class. Then operations that access the attributes will match better.
3. Some operations may have fewer arguments because they are special cases of more general arguments. Implement the special operations by calling the general operation with appropriate parameter values.
4. Opportunities to use inheritance are not always recognized during the analysis phase of development, so it is worthwhile to reexamine the object model looking for commonality between classes.

#### c. Write short notes on :

##### i. SA/SD and JSD

**Ans.**

##### i. SA/SD :

1. Structured Analysis and Structured Design (SA/SD) is diagrammatic notation which is design to help people understand the system.
2. The basic goal of SA/SD is to improve quality and reduce the risk of system failure.
3. It establishes concrete management specification and documentation.

4. It focuses on solidity, pliability and maintainability of system.
5. The approach of SA/SD is based on the Data Flow Diagram.
6. It is easy to understand SA/SD but it focuses on well defined system boundary whereas JSD approach is too complex and does not have any graphical representation.
7. SA/SD is combined known as SAD and it mainly focuses on following three points :
  - a. System
  - b. Process
  - c. Technology
8. SA/SD involves two phases :
  - a. **Analysis Phase :** It uses Data Flow Diagram, Data Dictionary, State Transition diagram and ER diagram.
  - b. **Design Phase :** It uses Structure Chart and Pseudo Code.

## ii. Jackson structured development :

1. Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
2. JSD can start from the stage in a project when there is only a general statement of requirements.
3. Following are the phases of JSD :
  - a. **Modeling phase :** In the modeling phase, the designer creates a collection of entity structure diagrams and identifies the entities in the system, the actions they perform, the attributes of the actions and time ordering of the actions in the life of the entities.
  - b. **Specification phase :** This phase focuses on actually what is to be done. Major goal is to map progress in the real world on progress in the system that models it.
  - c. **Implementation phase :**
    - i. In the implementation phase JSD determines how to obtain the required functionality.
    - ii. Implementation way of the system is based on transformation of specification into efficient set of processes.

## ii. Procedural v/s OOP.

**Ans.**

| S.No. | Procedural Oriented Programming                                                           | Object-Oriented Programming                                                            |
|-------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 1.    | In procedural programming, program is divided into small parts called functions.          | In object-oriented programming, program is divided into small parts called objects.    |
| 2.    | Procedural programming follows top down approach.                                         | Object-oriented programming follows bottom up approach.                                |
| 3.    | There is no access specifier in procedural programming.                                   | Object-oriented programming has access specifiers like private, public, protected etc. |
| 4.    | Adding new data and function is not easy.                                                 | Adding new data and function is easy.                                                  |
| 5.    | Procedural programming does not have any proper way for hiding data so it is less secure. | Object-oriented programming provides data hiding so it is more secure.                 |
| 6.    | In procedural programming, overloading is not possible.                                   | Overloading is possible in object-oriented programming.                                |
| 7.    | In procedural programming, function is more important than data.                          | In object-oriented programming, data is more important than function.                  |
| 8.    | Procedural programming is based on unreal world.                                          | Object-oriented programming is based on real world.                                    |
| 9.    | <b>Examples :</b> C, FORTRAN, Pascal, Basic etc.                                          | <b>Examples :</b> C++, Java, Python, C# etc.                                           |

4. Attempt any **two** questions : (10 × 2 = 20)

a. **What is Abstraction ? Explain Abstract method and Abstract class. Write a Java program for Employee class where salary is an abstract method with full implementation.**

**Ans.** This question is out of syllabus from session 2020-21.

b. **Explain Multithreading in Java. What is the effect in program when we use multithreading in a program ? Write a Java program for  $z = \sin(x) + \cos(y)$  using multithreading.**

**Ans.** This question is out of syllabus from session 2020-21.

c. **Write short notes on :**

i. **Event Handling**

ii. **EJB**



**Ans.** This question is out of syllabus from session 2020-21.

**5.** Attempt any **two** questions : **(10 × 2 = 20)**

**a.** **What is Applet ? Explain life cycle of an applet with methods declaration. Write a Java program which shows a House.**

**Ans.** This question is out of syllabus from session 2020-21.

**b.** **Explain Swing. What is the role of layout managers in swing ? Write a Java program which shows swing application.**

**Ans.** This question is out of syllabus from session 2020-21.

**c.** **Write short notes on :**

**i.** **JDBC**

**ii.** **Servlet**

**Ans.** This question is out of syllabus from session 2020-21.



**B. Tech.**  
**(SEM. V) ODD SEMESTER THEORY**  
**EXAMINATION, 2014-15**  
**OBJECT ORIENTED TECHNIQUES**

---

**Time : 3 Hours****Max. Marks : 100**

---

**Note :** Attempt **All** questions.

1. Attempt any **four** parts : (5 × 4 = 20)
  - a. **Define polymorphism. Is this concept only applicable to object-oriented systems ? Explain.**
  - b. **What is the difference between object-based and object oriented programming language ?**
  - c. **What are the basic principles of modelling ? Explain in detail.**
  - d. **Why UML required ? What are the basic architecture of UML ?**
  - e. **“Object oriented programs are easy to maintain” Justify.**
  - f. **Explain generosity in java with suitable example.**
2. Answer any **four** parts : (5 × 4 = 20)
  - a. **Explain the deployment diagram. What is the difference between components and nodes ?**
  - b. **Define package. Explain the package diagram with suitable diagram.**
  - c. **Write short notes on use case diagram with suitable diagram and their utility in system design.**
  - d. **What do you mean by activity diagram ? Explain in detail.**
  - e. **Define state machine ? Draw a state machine diagram for answering a telephone call.**
  - f. **What do you mean by event ? What are the types of event explain with example ?**

**3. Answer any two parts : (10 × 2 = 20)**

**a. Compare the OMT methodology with SA/SD methodology. Explain with suitable example.**

**b. Write example short notes on the following :**

**i. Translating object oriented design into an implementation.**

**ii. Jackson Structured Development.**

**c. Describe the relation of functional model, object model and dynamic models. What is relationship and difference between OOA (Object oriented analysis) and OOD (Object oriented design) ?**

**4. Answer any two parts : (10 × 2 = 20)**

**a. i. What do you mean by pure object-oriented language ? Is Java pure object oriented language ? If yes How ?**

**ii. What is concept of constructor explain with example ? Why it is needed ?**

**b. i. What do you mean by multithreading ? Does it have an impact in the performance of Java ? Explain.**

**ii. Write a program in Java to read in two matrices from the keyboard and compute their sum. Overload to String ( ) method to display the result matrix in row and column form.**

**c. Write down the difference between the following :**

**i. Abstract class and Interface**

**ii. Mouse Listener and Mouse Motion Listener**

**5. Answer any two parts : (10 × 2 = 20)**

**a. Write short notes on the following with suitable examples :**

**i. Applet and its life cycle. ii. Java Swings.**

**b. What do you understand by JDBC/ODBC bridge / Why it is required ? How it is implemented in Java ? Explain with an example.**

**c. Write short notes on any two of following :**

**i. Java Servlets ii. AWT**

**iii. Exception Handling Techniques**

**iv. Enterprise Java Beans.**



## SOLUTION OF PAPER (2014-15)

**Note :** Attempt **All** questions.

1. Attempt any **four** parts : (5 × 4 = 20)

a. **Define polymorphism. Is this concept only applicable to object-oriented systems ? Explain.**

**Ans. Polymorphism :**

1. Polymorphism means having many forms.
2. Polymorphism is the ability of a message to be displayed in more than one form.
3. It plays an important role in allowing objects having different internal structure to share the same external interface.

**Applicability of polymorphism :**

1. In programming languages there are two types of polymorphism ad-hoc and universal.
2. There are two kinds of universal polymorphism: parametric and subtyping.
3. Ad-hoc polymorphism is a kind of polymorphism in which polymorphic functions can be applied to arguments of different types.
4. In universal (parametric) polymorphism, the polymorphic functions are written without mention of any specific type.
5. The ad-hoc polymorphism is applicable in both traditional and object-oriented programming environments, whereas universal polymorphism only applies to object-oriented systems.

b. **What is the difference between object-based and object oriented programming language ?**

**Ans.**

| Basis               | Object Based Language                                                                                                                      | Object-Oriented Language                                                           |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Support of features | Object Based Language does not support all the features of Oops.                                                                           | Object-Oriented Language supports all the features of Oops.                        |
| Sample              | Visual Basic is an Object based Programming Language because we can use class and Object but can not inherit one class from another class. | Java is an Object-Oriented Languages because it supports all the concepts of Oops. |
| Example             | Javascript, VB                                                                                                                             | C#, Java, VB.                                                                      |

- c. **What are the basic principles of modelling ? Explain in detail.**

**Ans.** Principle of modeling are :

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 correct model as per the requirement of problem statement.
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 non-trivial system is best approached through a small set of nearly independent models :** This means we 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.

**Importance of modeling :**

Modeling help the development team better to visualize the plan of their system and allow them to develop more rapidly by helping them build the right thing.

- d. **Why UML required ? What are the basic architecture of UML ?**

**Ans.** The UML is required to help system and software developers accomplish the following tasks :

- i. Specification
- ii. Visualization
- iii. Architecture design
- iv. Construction
- v. Simulation and testing
- vi. Documentation

**Basic architecture of UML :**

1. The UML is defined in a circular manner, in which a subset of the language notation and semantics is used to specify the language itself.
2. The UML is defined within a conceptual framework for modeling that consists of four distinct layers or levels of abstraction.

3. This framework is based on the most fundamental UML notation that concepts are depicted as symbols, and relationships among concepts are depicted as paths (lines) connecting symbols. Both of these types of elements may be named.
4. The concepts introduced by the UML are organized around architectural views to define the various diagrams.
5. The UML diagrams are used to understand for conceptualize a problem, solve the problem, and implement or realize the solution.

**e. "Object oriented programs are easy to maintain" Justify.**

**Ans.** The main benefits that object-oriented programming (OOP) bring are :

1. OOP is a well-defined paradigm, so you are using a shared vocabulary and set of programming idioms with other programmers, which improves communication.
2. OOP offers modularity, since each class is a separate namespace that helps organize the code into independent subprograms.
3. OOP offers encapsulation, meaning you can restrict access to a class's data and methods, which reduces the number of ways that state can change in your program, thus simplifying it.
4. OOP offers inheritance and polymorphism, which enables to customize the behavior of a parent class and use behavior that is specific to each subclass without having to check the type each time.
5. OOP provides a well-defined interface for unit testing, which helps during software maintenance since unit tests can be rerun after refactoring the code to determine if problems have occurred.
6. Due to the above benefits we can say that, "Object oriented programs are easy to maintain".

**f. Explain generosity in java with suitable example.**

**Ans.** This question is out of syllabus from session 2020-21.

**2. Answer any four parts :**

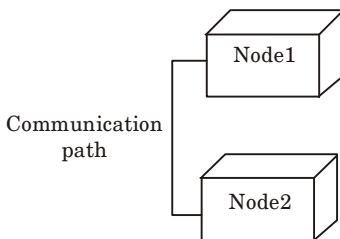
**(5 × 4 = 20)**

**a. Explain the deployment diagram. What is the difference between components and nodes ?**

**Ans.** **Deployment diagram :**

1. They display the configuration of run-time processing elements and the software components, processes and objects.
2. The deployment diagram contains nodes and connection. A node is a piece of hardware in the system.

3. A connection depicts the communication path used by the hardware Fig. 1.



**Fig. 1.** Deployment diagram.

4. Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.
5. Deployment diagrams are used to describe the static deployment view of a system.

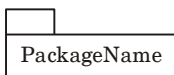
**Difference :**

| Node                                                                                          | Component                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Node represent the physical part of the system for instance, server, network and printer etc. | Component represent any part of the system it might be physical aspect such as libraries, file, executables, document, packages etc., that reside on the node. |

- b. Define package. Explain the package diagram with suitable diagram.**

**Ans. Packages :**

1. A package is a group of elements (classes, associations, generalizations, and lesser packages) with a common theme.
2. A package partitions a model, making it easier to understand and manage. Large applications may require several tiers of packages.
3. Packages form a tree with increasing abstraction toward the root, which is the application, the top-level package.
4. As Fig. 2 shows, the notation for a package is a box with a tab. The purpose of the tab is to suggest the enclosed contents, like a tabbed folder.



**Fig. 2.** Notation for a package.

5. For example, many business systems have a *Customer* package or a *Part* package; *Customer* and *Part* are dominant classes that are important to the business of a corporation and appear in many applications.
- c. **Write short notes on use case diagram with suitable diagram and their utility in system design.**

**Ans. Use case diagrams :**

1. Use cases describe how a system interacts with external user of a system (*i.e.*, actor).
2. Each use case represents a piece of functionality that a system provides to its users.
3. Use cases are helpful for capturing informal requirements.
4. Use case consist of actors.
5. An actor is an object or set of objects that communicates directly with the system but that is not part of the system.
6. The various interactions of actors with a system are quantized into use cases.
7. A use case is a coherent piece of functionality that a system can provide by interacting with actors.
8. For example, a customer actor can buy a beverage from a vending machine. The customer inserts money into the machine, makes a selection, and ultimately receives a beverage. Similarly, a repair technician can perform scheduled maintenance on a vending machine.
9. Fig. 3 summarizes several use cases for a vending machine.

- **Buy a beverage :** The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance :** A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs :** A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items :** A stock clerk adds items into the vending machine to replenish its stock of beverages.

**Fig. 3.** Use case summaries for a vending machine.

**Utility of use cases diagram in system design :**

1. Use cases identify the functionality of a system and organize it according to the perspective of users.



2. Use cases describe complete transactions and are therefore less likely to omit necessary steps.
3. The main purpose of a system is almost always found in the use cases, with requirements lists supplying additional implementation constraints.

**d. What do you mean by activity diagram ? Explain in detail.**

**Ans.**

1. An activity diagram is a flowchart that shows activities performed by a system.
2. The two special states shown in an activity diagram are the Initial State (Start Point) and Final State (End Point).
3. **Initial State or Start Point :** A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.



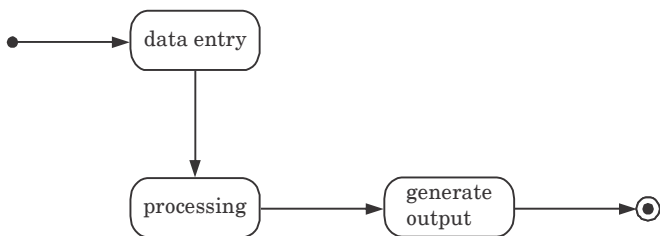
Start Point/Initial State

4. **Final State or End Point :** An arrow pointing to a filled circle nested inside another circle represents the final action state.



Ent Point Symbol

**For example :**



**Fig. 4.** Notation for activity states and activity-state transitions.

**e. Define state machine ? Draw a state machine diagram for answering a telephone call.**

**Ans.** **State machine :** A state machine diagram models the behaviour of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.

Diagram :

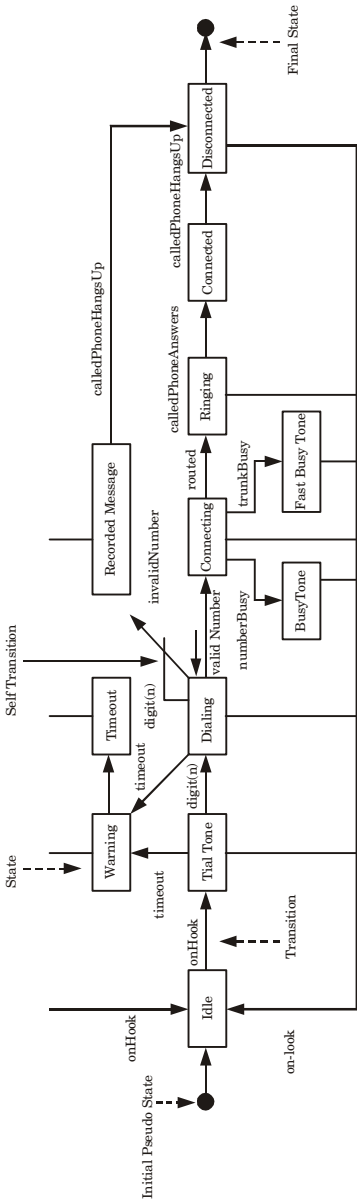


Fig. 5. State machine diagram for answering a telephone call.

**f. What do you mean by event ? What are the types of event explain with example ?**

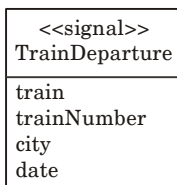
**Ans.**

1. An event is something that happens at a point in time, such as user presses right mouse button. An event is a one-way transmission of information from one object to another. An event conveys information from one object to another.
2. An event has no duration. By definition, an event happens instantaneously with regard to time scale of an application.
3. Following are three most common events :

**A. Signal event :**

1. A signal event is the event of sending or receiving a signal.
2. A signal is a one-way transmission of information from one object to another.
3. A signal is a message between objects while a signal event is an occurrence in time.

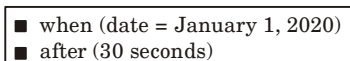
For example, **TrainDeparture** has attributes train, trainNumber, city, and date. The UML notation is the keyword signal in guillemets (<<signal>>) above the signal class name in the top section of a box. The bottom section lists the signal attributes.



**Fig. 6.**

**B. Time event :**

1. A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval.
2. For example, Fig. 7 shows, the UML notation for an absolute time is the keyword *when* followed by a parenthesized expression involving time. The notation for time interval is the keyword *after* followed by a parenthesized expression that evaluates to a time duration.



**Fig. 7.**

**C. Change event :**

1. A change event is an event caused by the satisfaction of a boolean expression.

2. The boolean expression is continually tested and whenever the expression changes from false to true, the event occurs.
3. For example, the UML notation for a change event is the keyword *when* followed by a parenthesized boolean expression. Fig. 8 shows examples of change events.

- when (cabin temperature < heating set point)
- when (cabin temperature > cooling set point)

Fig. 8.

3. Answer any **two** parts :

(10 × 2= 20)

- a. **Compare the OMT methodology with SA/SD methodology. Explain with suitable example.**

**Ans.**

| S. No. | OMT methodology                                          | SA/SD methodology                                        |
|--------|----------------------------------------------------------|----------------------------------------------------------|
| 1.     | It manages a system around procedures.                   | It manages system around real world objects.             |
| 2.     | Focus more on functional model and less on object model. | Focus more on object model and less on functional model. |
| 3.     | Dominance order is functional, dynamic and object.       | Dominance order is object, dynamic and functional.       |
| 4.     | It is a historical approach.                             | It is advanced approach.                                 |
| 5.     | Reusability of components across projects is less.       | Reusability of components across projects is more.       |
| 6.     | Not easily modifiable and extensible.                    | Easily modifiable and extensible.                        |
| 7.     | Used when functions are more important than data.        | Used when data is more important than functions.         |

**b. Write example short notes on the following :**

- i. **Translating object oriented design into an implementation.**

**Ans.**

- i. It is easy to implement an object-oriented design with an object-oriented language since language constructs are similar to design constructs.
- ii. The following steps are required to implement an object oriented design in an object-oriented language :

**1. Class definitions :**

- i. The first step in implementing an object-oriented design is to declare object classes. Each attribute and operation in an object diagram must be declared as part of its corresponding class.
- ii. Assign data types to attributes. Declare attributes and operations as either public or private.

**2. Creating objects :**

- i. Object-oriented languages create new objects in following two ways :
  - a. Class operation applied to a class object creates a new object of the class.
  - b. Using special operations that create new objects.
- ii. When a new object is created, the language allocates storage for its attribute values and assigns it a unique object ID.

**3. Calling operations :**

- i. In most object-oriented languages, each operation has at least one implicit argument, the target object, indicated with a special syntax.
- ii. Operations may or may not have additional arguments.

**4. Using Inheritance :**

- i. To implement inheritance object-oriented languages use different mechanisms.
- ii. There are three independent dimensions for classifying inheritance mechanisms :
  - a. Static or dynamic
  - b. Implicit or explicit
  - c. Per object or per group.
- iii. Many of the popular languages are static, implicit and per group.

**5. Implementing associations :**

- i. There are two approaches to implement associations : Buried pointers and distinct association objects.
- ii. If the language does not explicitly support association objects then buried pointers are easy to implement.
- iii. An association can also be implemented as a distinct container object.

**ii. Jackson Structured Development.**

**Ans.**

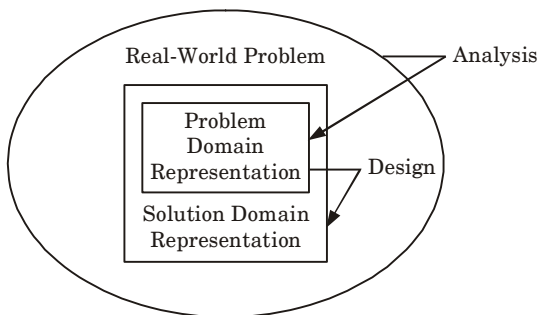
1. Jackson System Development (JSD) is a method of system development that covers the software life cycle either directly or by providing a framework into which more specialized techniques can fit.
2. JSD can start from the stage in a project when there is only a general statement of requirements.
3. Following are the phases of JSD :
  - a. **Modeling phase :** In the modeling phase, the designer creates a collection of entity structure diagrams and identifies the entities in the system, the actions they perform, the attributes of the actions and time ordering of the actions in the life of the entities.
  - b. **Specification phase :** This phase focuses on actually what is to be done. Major goal is to map progress in the real world on progress in the system that models it.
  - c. **Implementation phase :**
    - i. In the implementation phase JSD determines how to obtain the required functionality.
    - ii. Implementation way of the system is based on transformation of specification into efficient set of processes.
- c. **Describe the relation of functional model, object model and dynamic models. What is relationship and difference between OOA (Object oriented analysis) and OOD (Object oriented design) ?**

**Ans.**

1. The functional model shows what “has to be done” by a system. The leaf process are the operation on objects.
2. The object model shows the “doers” the objects. Each process is implemented by a method on some object.
3. The dynamic model shows the sequences in which the operations are performed. Each sequence is implemented as a sequence, loop or alteration of statements within some method.
4. The processes in the functional model correspond to operations in the object model.
5. Actors are explicit objects in the object models. Data flows to or from actors represent operations on or by the objects.
6. Because actors are self-motivated objects, the functional model is not sufficient to indicate when they act. The dynamic model for an actor object specifies when it acts.

**Relationship between OOA and OOD :**

1. When object orientation is used in analysis as well as design, the boundary between OOA and OOD is blurred. This is particularly true in methods that combine analysis and design.
2. One reason for this blurring is the similarity of basic constructs (*i.e.*, objects and classes) that are used in OOA and OOD.



**Fig. 9.** Relationship between OOA and OOD.

**Difference between OOA and OOD :**

1. The fundamental difference between OOA and OOD is that OOA models the problem domain, leading to an understanding and specification of the problem, while the OOD models the solution to the problem.
2. That is, analysis deals with the problem domain, while design deals with the solution domain.

4. Answer any **two** parts :

(10 × 2 = 20)

- a. i. What do you mean by pure object-oriented language ? Is Java pure object oriented language ? It yes How ?**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. What is concept of constructor explain with example ? Why it is needed ?**

**Ans.** This question is out of syllabus from session 2020-21.

- b. i. What do you mean by multithreading ? Does it have an impact in the performance of Java ? Explain.**

**Ans.** This question is out of syllabus from session 2020-21.

- ii. Write a program in Java to read in two matrices from the keyboard and compute their sum. Overload to String ( ) method to display the result matrix in row and column form.**

- Ans.** This question is out of syllabus from session 2020-21.
- c. Write down the difference between the following :**
- Abstract class and Interface**
  - Mouse Listener and Mouse Motion Listener**

**Ans.** This question is out of syllabus from session 2020-21.

- 5. Answer any two parts : (10 × 2 = 20)**
- a. Write short notes on the following with suitable examples :**
- Applet and its life cycle.**
  - Java Swings.**

**Ans.** This question is out of syllabus from session 2020-21.

- b. What do you understand by JDBC/ODBC bridge / Why it is required ? How it is implemented in Java ? Explain with an example.**

**Ans.** This question is out of syllabus from session 2020-21.

- c. Write short notes on any two of following :**
- Java Servlets**
  - AWT**
  - Exception Handling Techniques**
  - Enterprise Java Beans.**

**Ans.** This question is out of syllabus from session 2020-21.





**B. Tech.**  
**(SEM. V) ODD SEMESTER THEORY**  
**EXAMINATION, 2015-16**  
**OBJECT ORIENTED TECHNIQUES**

**Time : 3 Hours****Max. Marks : 100**

**Section-A**

1. Attempt **all** parts. All parts carry equal marks. Write answer of each part in short (2 × 10 = 20)
- a. **What is nested state diagram ? Explain with suitable example.**
- b. **Differentiate between early and late binding with an example.**
- c. **Differentiate between overriding and overloading of function in java.**
- d. **What is unified markup language ?**
- e. **What is activity diagram ? Explain with suitable example.**
- f. **Discuss the synchronization of concurrent activities.**
- g. **Explain package bundling in java.**
- h. **What do you mean by candidate keys in object modeling ?**
- i. **What is inheritance in java ?**

**Section-B**

Attempt any **five** questions from this section. (10 × 5 = 50)

2. **Compare architectural modeling and behavioral modeling with justification.**
3. **What is open data base connectivity (ODBC) ? How it is used in java data base connectivity ?**
4. **Explain the Dynamic Model with an example.**
5. **What do you mean by object modeling technique ? Explain. Discuss the various stages of the object modeling techniques with some example.**

6. What do you mean by Scenarios ? Prepare an event trace for a phone call.
7. Why Java does not support multiple inheritances ? Justify.
8. Explain structured analysis and structured design (SA/SD) with example.
9. What is multi threading ? How it is achieved in java.

### Section-C

Attempt any **two** questions from this section.

(15 × 2 = 30)

**10. Write short notes on :**

- a. AWT in Java
- b. Applet in java
- c. JDBC in Java

**11. Write short notes on :**

- a. Data store
- b. Actors
- c. Control flow

**12. Compare each in detail :**

- a. Applet and Application
- b. Abstraction and Encapsulation.

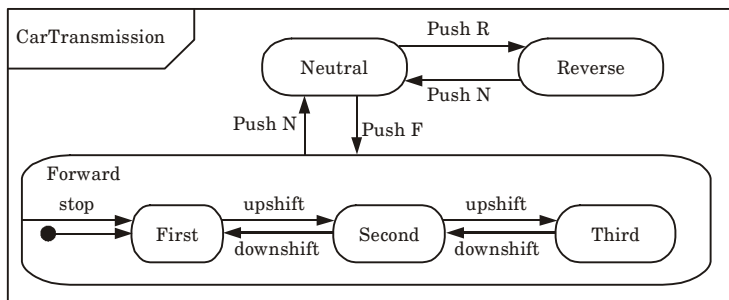


**SOLUTION OF PAPER (2015-16)****Section-A**

1. Attempt **all** parts. All parts carry equal marks. Write answer of each part in short **(2 × 10 = 20)**
- a. **What is nested state diagram ? Explain with suitable example.**

**Ans.**

1. A nested state diagram is used to model the complex system as the regular state diagram is inadequate in describing the large and complex problem.
2. The nested state diagram is the concept of advanced state modeling.

**For example :****Fig. 1.** Nested stage diagram of car transmission.

1. There are three states inside CarTransmission state diagram i.e., reverse, neutral and forward.
2. Among these three states, the forward state has three nested states i.e. First, Second and Third.
3. At any nested state of Forward gear composite state, selecting N would transit the corresponding state to the neutral state.
4. Being in a neutral state, selecting F would transit you to the forward state.
5. But here by default, the First nested state in the Forward contour is the initial state and the control is in the First state.
6. Here the Forward is just an abstract state.
7. The event Stop is shared by all the three nested state.

- b. Differentiate between early and late binding with an example.**

**Ans.**

| S. No. | Early binding                                                             | Late binding                                                          |
|--------|---------------------------------------------------------------------------|-----------------------------------------------------------------------|
| 1.     | It is a compile-time process.                                             | It is a run-time process.                                             |
| 2.     | The method definition and method call are linked during the compile time. | The method definition and method call are linked during the run time. |

- c. Differentiate between overriding and overloading of function in java.**

**Ans.** This question is out of syllabus from session 2020-21.

- d. What is unified markup language ?**

**Ans. UML :**

1. UML stands for Unified Modeling Language.
2. UML is a pictorial language used to make software blueprints.
3. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.

- e. What is activity diagram ? Explain with suitable example.**

**Ans.**

1. An activity diagram is a flowchart that shows activities performed by a system.
2. The two special states shown in an activity diagram are the Initial State (Start Point) and Final State (End Point).
3. **Initial State or Start Point :** A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.



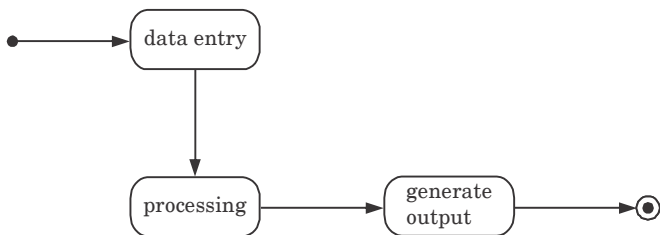
Start Point/Initial State

4. **Final State or End Point :** An arrow pointing to a filled circle nested inside another circle represents the final action state.



Ent Point Symbol

**For example :**



**Fig. 4.** Notation for activity states and activity-state transitions.

**f. Discuss the synchronization of concurrent activities.**

**Ans.**

1. Sometimes one object must perform two (or more) activities concurrently.
2. The object does not synchronize the internal steps of the activities but must complete both activities before it can progress to its next state.
3. For example, a cash dispensing machine dispenses cash and returns the user's card at the end of a transaction. The machine must not reset itself until the user takes both the cash and the card, but the user may take them in either order or even simultaneously.

**g. Explain package bundling in java.**

**Ans.** This question is out of syllabus from session 2020-21.

**h. What do you mean by candidate keys in object modeling ?**

**Ans.**

1. A super key with no redundant attribute is known as candidate key.
2. Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is that the candidate key should not have any redundant attributes.

**i. What is inheritance in java ?**

**Ans.** This question is out of syllabus from session 2020-21.

## Section-B

Attempt any **five** questions from this section. **(10 × 5 = 50)**

2. **Compare architectural modeling and behavioral modeling with justification.**

**Ans.**

| S. No. | Architectural Modeling                                                        | Behavioral Modeling                                                 |
|--------|-------------------------------------------------------------------------------|---------------------------------------------------------------------|
| 1.     | It represents the overall framework of the system.                            | It describes the interaction within the system.                     |
| 2.     | The structural and the behaviour elements of the system are represented here. | The interaction among the structural diagrams is represented here.  |
| 3.     | It is the blue print for the entire system.                                   | It shows the dynamic nature of the system.                          |
| 4.     | It uses package diagram.                                                      | It uses activity diagrams, interaction diagrams, use case diagrams. |

**3. What is open data base connectivity (ODBC) ? How it is used in java data base connectivity ?**

**Ans.** This question is out of syllabus from session 2020-21.

**4. Explain the Dynamic Model with an example.**

**Ans.** **Dynamic modeling :**

- The dynamic model represents the time-dependent aspects of a system.
- It is concerned with the temporal changes in the states of the objects in a system.
- It is used to specify and implement the control aspect of the system.
- Dynamic model is represented graphically with the help of state diagrams.
- The dynamic model consists of multiple state diagrams and shows the pattern of activity for an entire system.
- For example :
  - Fig. 1 shows a simple dynamic model of programmable thermostat.
  - This device controls a furnace and air conditioner according to time-dependent attributes.
  - While running, the thermostat operates the furnace or air conditioner to keep the current temperature equal to the target temperature.
  - The target temperature is taken from a table of program values supplied by the user.

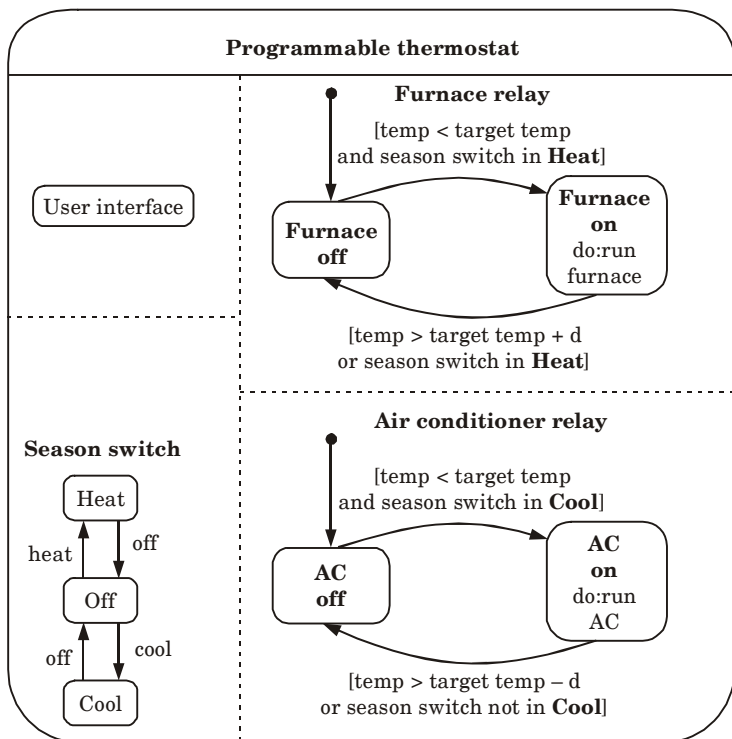


Fig. 1.

**5. What do you mean by object modeling technique ? Explain. Discuss the various stages of the object modeling techniques with some example.**

**Ans. Object modeling technique :**

1. The Object Modeling Technique (OMT) is the methodology that combines the three views of modeling system, i.e., the object model, the dynamic model and the functioned model.
2. These three models separate a system into orthogonal views that can be represented and manipulated with a uniform notation.

**Stages of object modeling technique :**

**1. Analysis :**

- a. Starting from a statement of the problem, the analyst builds a model of the real-world situation showing its important properties.

- b. The analysis model is a concise, precise abstraction of what the desired system must do.
- c. The objects in the model should be application-domain concepts.
- d. A good model can be understood by application experts who are not programmers.
- e. For example, a *Window* class in a workstation windowing system would be described in terms of the attributes and operations visible to a user.

## 2. System design :

- a. The system designer makes high-level decisions about the overall architecture.
- b. During system design, the target system is organized into subsystems based on both the analysis structure and the proposed architecture.
- c. The system designer must decide what performance characteristics to optimize, developing strategy for solving the problem, and making tentative resource allocations.
- d. For example, the system designer might decide that changes to the workstation screen must be fast and smooth and choose an appropriate communications protocol and memory buffering strategy.

## 3. Object design :

- a. The object designer builds a design model based on the analysis model but containing implementation details.
- b. The designer adds details to the design model in accordance with the strategy established during system design.
- c. The focus of object design is the data structures and algorithms needed to implement each class.
- d. The object classes are augmented with computer-domain data structures and algorithms chosen to optimize important performance measures.
- e. For example, the *Window* class operations are now specified in terms of the underlying hardware and operating system.

## 4. Implementation :

- a. The object classes and relationships developed during object design are finally implemented.
- b. During implementation, it is important to follow good software engineering practice so that traceability to the design is



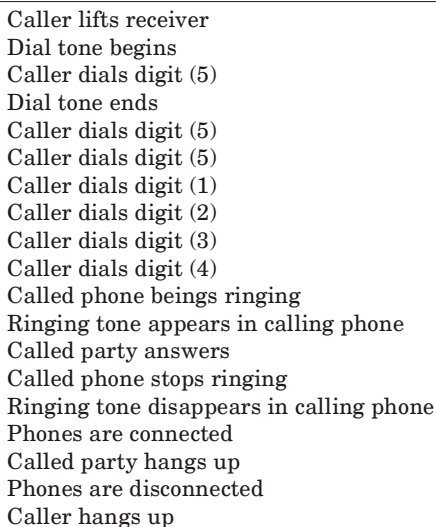
straight-forward and so that the implemented system remains flexible and extensible.

- c. For example, the *Window* class would be coded in a programming language, using calls to the underlying graphics system on the workstation.

**6. What do you mean by Scenarios ? Prepare an event trace for a phone call.**

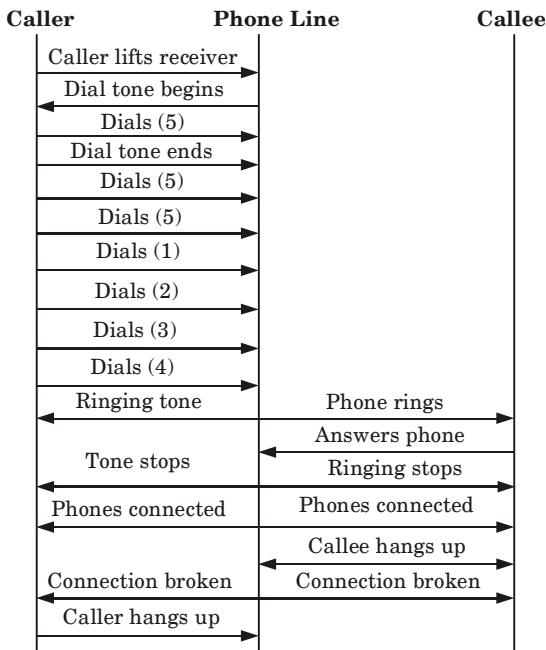
**Ans. Scenarios :**

1. A scenario is a sequence of actions that takes place within the system.
2. The starting point is a stimulus from an external source and the end point is the logical termination of a particular sequence.
3. The event trace for a phone call is depicted in Fig. 2.
4. Each of the three objects, Caller, Phone Line, and Callee, is shown with a vertical line.
5. Events between the objects are shown with horizontal lines connecting the objects.



Caller lifts receiver  
Dial tone begins  
Caller dials digit (5)  
Dial tone ends  
Caller dials digit (5)  
Caller dials digit (5)  
Caller dials digit (1)  
Caller dials digit (2)  
Caller dials digit (3)  
Caller dials digit (4)  
Called phone beings ringing  
Ringing tone appears in calling phone  
Called party answers  
Called phone stops ringing  
Ringing tone disappears in calling phone  
Phones are connected  
Called party hangs up  
Phones are disconnected  
Caller hangs up

**Fig. 2.** Scenario for a phone call.



**Fig. 3.** Event trace for a phone call.

**7. Why Java does not support multiple inheritances ? Justify.**

**Ans.** This question is out of syllabus from session 2020-21.

**8. Explain structured analysis and structured design (SA/SD) with example.**

**Ans.** **Structured analysis and structure design :**

1. Structured Analysis and Structured Design (SA/SD) is diagrammatic notation which is design to help people understand the system.
2. The basic goal of SA/SD is to improve quality and reduce the risk of system failure.
3. It establishes concrete management specification and documentation.
4. It focuses on solidity, pliability and maintainability of system.
5. The approach of SA/SD is based on the Data Flow Diagram.

6. It is easy to understand SA/SD but it focuses on well defined system boundary whereas JSD approach is too complex and does not have any graphical representation.
7. SA/SD is combined known as SAD and it mainly focuses on following three points :
  - a. System
  - b. Process
  - c. Technology
8. SA/SD involves two phases :
  - a. **Analysis Phase** : It uses Data Flow Diagram, Data Dictionary, State Transition diagram and ER diagram.
  - b. **Design Phase** : It uses Structure Chart and Pseudo Code.
9. **For example :**
  - i. During structured design, data flow diagram processes are grouped into tasks and allocated to operating system processes and CPUs.
  - ii. Data flow diagram processes are converted into programming language functions, and a structure chart is created showing the procedure call tree.

**9. What is multi threading ? How it is achieved in java.**

**Ans.** This question is out of syllabus from session 2020-21.

**Section-C**

Attempt any **two** questions from this section.

**(15 × 2 = 30)**

**10. Write short notes on :**

**a. AWT in Java**

**Ans.** This question is out of syllabus from session 2020-21.

**b. Applet in java**

**Ans.** This question is out of syllabus from session 2020-21.

**c. JDBC in Java**

**Ans.** This question is out of syllabus from session 2020-21.

**11. Write short notes on :**

**a. Data store**

**Ans.**

1. A data store is a passive object within a data flow diagram that stores data for later access.

2. Unlike an actor, a data store does not generate any operations on its own but merely responds to requests to store and access data.
3. A data store allows values to be accessed in a different order than they are generated.
4. A data store is drawn as a pair of parallel lines containing the name of the store.
5. Input arrows indicate information or operations that modify the stored data; this includes adding elements, modifying values, or deleting elements.
6. Output arrows indicate information retrieved from the store. This includes retrieving the entire value or some component of it.

**b. Actors****Ans.**

1. An actor is an active object that drives the data flow graph by producing or consuming values.
2. Actors are attached to the inputs and outputs of a data flow graph.
3. Examples of actors include the user of a program, a thermostat, and a motor under computer control.
4. An actor is drawn as a rectangle to show that it is an object. Arrows between the actor and the diagram are inputs and outputs of the diagram.

**c. Control flow****Ans.**

1. A data flow diagram shows all possible computation paths for values; it does not show which paths are executed and in what order.
2. This is done by including control flows in the data flow diagram.
3. A control flow is a Boolean value that affects whether a process is evaluated.
4. The control flow is not an input value to the process itself.
5. A control flow is shown by a dotted line from a process producing a Boolean value to the process being controlled.

**12. Compare each in detail :****a. Applet and Application****Ans.** This question is out of syllabus from session 2020-21.

**b. Abstraction and Encapsulation.****Ans.**

| S. No. | Abstraction                                                           | Encapsulation                                                                                                                  |
|--------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 1.     | Abstraction is the process or method of gaining the information.      | Encapsulation is the process or method to contain the information.                                                             |
| 2.     | In abstraction, problems are solved at the design or interface level. | In encapsulation, problems are solved at the implementation level.                                                             |
| 3.     | Abstraction is the method of hiding the unwanted information.         | Encapsulation is a method to hide the data in a single entity or unit along with a method to protect information from outside. |
| 4.     | We can implement abstraction using abstract class and interfaces.     | Encapsulation is implemented using by access modifier <i>i.e.</i> private, protected and public.                               |
| 5.     | The objects that help to perform abstraction are encapsulated.        | Objects that result in encapsulation need not be abstracted.                                                                   |

