**Annexure- II: Format of Answer Scheme**

** PRESIDENCY UNIVERSITY, BENGALURU**

**School Of Computer Science and Engineering & Information Science**

**End Term Final Examinations, August 2024**

**Extract of Question Distribution [outcome wise & level wise]**

**SOLUTION**

**Summer Term**: August 2024

**Course Code**: CSE 3082

**Course Name**: OBJECT OREINTED ANALYSIS AND DESIGN

**Program**: B.Tech CSE

**Department:** CSE,ISE,CAI,CCS,CEI,CIT,CSG,CST,CDV,CBD

**Branch & Sem**:

**Date**:

**Time**:

**Max Marks**:

**Weightage**:

**Date**: 12/08/2024

**Time**: 09:30 AM to 12:30 PM

**Max Marks**:100

**Weightage**: 50%

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q.NO | C.O.NO  (%age of CO) | Unit/Module Number/Unit  /Module Title | Memory recall type  [Marks allotted]  Bloom’s Levels | Thought provoking type  [Marks allotted]  Bloom’s Levels | Problem Solving type  [Marks allotted] | Total Marks |
|  | K | C | A |  |
| 1 | 1 | 1 | 4 | 6 | 10 | 20 |
| OR | | | | | | |
| 2 | 1 | 1 | 4 | 6 | 10 | 20 |
| 3 | 2 | 2 | 4 | 6 | 10 | 20 |
| OR | | | | | | |
| 4 | 2 | 2 | 4 | 6 | 10 | 20 |
| 5 | 3 | 3 | 4 | 6 | 10 | 20 |
| OR | | | | | | |
| 6 | 3 | 3 | 4 | 6 | 10 | 20 |
| 7 | 4 | 4 | 4 | 6 | 10 | 20 |
|  | | | | | | |
| 8 | 4 | 4 | 4 | 6 | 10 | 20 |
| 9 | 1 | 1 | 4 | 6 | 10 | 20 |
| OR | | | | | | |
| 10 | 2 | 2 | 4 | 6 | 10 | 20 |
|  | Total Marks |  | 20 | 30 | 50 | 100 |

K =Knowledge Level C = Comprehension Level, A = Application Level

Note: While setting all types of questions the general guideline is that about 60%

Of the questions must be such that even a below average students must be able to attempt, About **20% of the questions must be such that above average students** must be able to attempt and finally **20% of the questions must be such that only the bright students** must be able to attempt.

I hereby certify that all the questions are set as per the above guidelines.

**[Name of faculty] Mr.S.Sakthi**

Reviewer’s Comments:

**Annexure- II: Format of Answer Scheme**

****

**PRESIDENCY UNIVERSITY, BENGALURU**

**School Of Computer Science and Engineering & Information Science**

**End Term Final Examinations, August 2024**

**SOLUTION**

**Date**: 12/08/2024

**Time**: 3 Hours

**Max Marks**: 100

**Weightage**: 50%

**Summer Term**: August 2024

**Course Code**: CSE 3082

**Course Name**: OBJECT ORIENTED ANALYSIS AND DESIGN

**Program & Sem**: B.Tech CSE & VI SEM

**Date**:

**Time**:

**Max Marks**:

**Weightage**:

**Part A** (5Q x 20M = 100 Marks)

|  |  |  |  |
| --- | --- | --- | --- |
| Q No. | **Solution** | **Scheme of Marking** | **Max. Time required for each Question** |
| 1A | Describe the types of inheritance. [Remember]  Inheritance  [Inheritance](https://www.javatpoint.com/inheritance-in-java) is a mechanism of driving a new class from an existing class. The existing (old) class is known as base class or super class or parent class. The new class is known as a derived class or sub class or child class. It allows us to use the properties and behavior of one class (parent) in another class (child).  A class whose properties are inherited is known as parent class and a class that inherits the properties of the parent class is known as child class. Thus, it establishes a relationship between parent and child class that is known as parent-child or Is-a relationship.  Suppose, there are two classes named Father and Child and we want to inherit the properties of the Father class in the Child class. We can achieve this by using the extends keyword.  Types of Inheritance   * Single Inheritance * Multi-level Inheritance * Hierarchical Inheritance * Hybrid Inheritance | **4** | **5** |
| 1B | Explain Jacobson Methodology with appropriate examples in detail. [Understand]   * OOSE is developed by Ivar Jacobson in 1992. OOSE is the first object-oriented design methodology that employs use cases in software design. * It covers entire life cycle and stress traceability between different phases * Advantage :-   + Reduction of development time   + Reuse of code   Reuse of analysis and design work  Usecase:   * Use-Case is an interaction between user and the system. * Scenarios for understanding system requirements. * Use-case model captures the goal of the user and the responsibility of the system to its users. * Use Cases are described as :-   + Nonformal text with no clear flow of events.   + Text, easy to read but with a clear flow of events to follow   + Formal style using Pseudo code.   Extends and Uses Relationship:   * Extends – used when you have one use case similar to another use case but does a bit more (Extends the functionality of original use case---sub class). * Uses – Reuses the common behavior in different use case.     **Usecase diagram example** | **6** | **11** |
| 1C | Illustrate object oriented software development life cycle with neat sketch.[Apply]   * Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. * The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. * SDLC is a framework defining tasks performed at each step in the software development process.   Software Development Process   * Transformation 1(analysis) - translates user’s need into system’s requirements & responsibilities. * Transformation 2 (design) - begins with problem statement, ends with detailed design that can be transformed into operational system. * Transformation 3 (implementation) - refines detailed design into system deployment that will satisfy user’s needs.       Waterfall model | **10** | **20** |
| **OR** | | | |
| 2A | Define Polymorphism and Information Hiding. [Remember]  **Polymorphism**  Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.  Like we specified in the previous chapter; [Inheritance](https://www.w3schools.com/cpp/cpp_inheritance.asp) lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.  For example, think of a base class called Animal that has a method called animalSound(). Derived classes of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):  **Information Hiding**  Data hiding is a process of combining data and functions into a single unit. The ideology behind data hiding is to conceal data within a class, to prevent its direct access from outside the class. It helps programmers to create classes with unique data sets and functions, avoiding unnecessary penetration from other program classes. | **4** | **5** |
| 2B | Describe the process and components of unified approach with neat sketch. [Understand]  **Object-oriented software development life cycle consists of**   * + - **Object Oriented Analysis(OOA) - Explanation**     - **Object Oriented Design(OOD) - Explanation**     - **Object Oriented Implementation- Explanation** | **6** | **11** |
| 2C | Illustrate Rambaugh Object Model and Booch methodology with appropriate examples. [Apply]  **Rambaugh Object Model:**  **Object Modeling Techniques (OMT)**   * It was one of the first object oriented methodologies and was introduced by Rumbaugh in 1991. * OMT uses three different models that are combined in a way that is analogous to the older structured methodologies. * A method for analysis, design and implementation by an object oriented technique.   OMT consists of **four phases**, which can be performed iteratively:   1. **Analysis**: The results are objects and dynamic and functional models. 2. **System design**: The results are a structure of the basic architecture of the system along with high-level strategy decisions. 3. **Object design:** This phase produces a design document, consisting of detailed objects static, dynamic, and functional models. 4. **Implementation**: This activity produces reusable, extendible, and robust code.   OMT separates modeling into three different parts:  1. An Object model  2. A Dynamic model  3. A Functional model  **The Object Model**   * It depicts the object classes and their relationships as a class diagram, which represents the static structure of the system. * The object model is represented graphically with an object diagram * The object diagram contains classes interconnected by association lines. * Each class represents a set of individual objects. * The association lines establish relationships among the classes. * Each association line represents a set of links from the objects of one class to the objects of another class.     **The OTM Dynamic Model**   * OMT provides a detailed and comprehensive dynamic model, in addition to letting you depict states, transitions, events, and actions. * The OMT state transition diagram is a network of states and events. * Each state receives one or more events, at which time it makes the transition to the next state. * The next state depends on the current state as well as the events.     **The OMT Functional Model**   * Data Flow Diagram **shows the flow of data between different processes** in a business. * OMT DFD provides simple and intuitive approach for describing business processes. * **Four Primary Symbols used in DFD are :-**  1. Process ->The process is any function being performed; for example, verify Password or PIN in the ATM system 2. Data flow -> The data flow shows the direction of data element movement; for example, PIN code. 3. Data store -> The data store is a location where data are stored; for example, account is a data store in the ATM example. 4. External entity -> An external entity is a source or destination of a data element; for example, the ATM card reader.       **Booch methodology**   * This method starts with class and object diagrams (a discovering activity) in the analysis phase * These diagrams to be refined through various steps * Refinement process continues till the problem domain gets more and more understood following an *evolutionary* approach * Design symbols are to be added when ready to generate code * Usually it represents very final implementation decisions * Booch notation are larger sets and appears to prove beneficial: it is possible to fully document the OO-code   Booch proposes different views to describe an OO system   * 1. Physical model   2. Logical model   3. Static model   4. Dynamic model   **1.Physical model**   * It describes the concrete hardware with respect to the software components of a system.   + Module and process architecture   + Processors   + Devices and communication connections between them   **2.Logical model (i.e., the problem domain)**   * Represented in the class and object structure * In the class diagram one builds up the architecture, or the static model * To deal with complex diagrams   + the notation allows *class categories* to group classes into namespaces, each category being itself a class diagram.   **3 Static Model:**   * + Class diagrams (and the relationships therein) are mostly static     **4 Dynamic Model :**   * + - Object diagrams (and the relationships therein) describe the dynamic behavior of the system     - In this instance relationship means *message exchanges* between objects   **The Booch method consists of the following diagrams**   * Class diagrams * Object diagrams * State transition diagrams * Module diagrams * Process diagrams * Interaction diagrams   **Note: Any two diagram should be drawn**  **Two processes:**   * BOOCH Methodology prescribes 2 different process   1. Macro Development Process  2. Micro Development Process  **1.Macro development process**   * Primary concern – technical management of the system   **Steps involved:**   * **Conceptualization.**   Establish the core requirements and develop a prototype   * **Analysis and development of the model** * Use the class diagram to describe the roles * Responsibilities of objects * Use the object diagram to describe the desired behavior   of the system   * **Design or create the system architecture**   + Use the class diagram to decide what classes exist and how they relate to each other   + The object diagram to decide what mechanisms are used   + The module diagram to map out where each class and object should be declared   + The process diagram to determine to which processor to allocate a process. * **Evolution or implementation**   + Refine the system through much iteration * **Maintenance**   + Make localized changes to the system to add new requirements and eliminate bugs     **Micro development process**   * The micro development process is a description of the **day-to-day activities.** * Steps involved:   + Identify classes and objects   + Identify classes and object semantics   + Identify classes and object relationships   + Identify classes and object interfaces and implementation     **An alarm state transition diagram with Booch notation** | **10** | **20** |
| 3A | Describe why analysis is a difficult activity ? [Remember]   * Analysis is a creative process, that involves understanding problem domain, its associated constraints, and methods to overcome those constraints * Three most common sources of requirements difficulties :-   + 1. Fuzzy Description     2. Incomplete Requirements     3. Unnecessary Features | **4** | **5** |
| 3B | Explain various common class pattern approach and the guidelines for identifying super sub relationship. [Understand]  **Various common class pattern approach:**  **Explanation: Definition and example for the following:**   * Concept Class [Idea or Understanding] * Event Class [points in time to be recorded] * Organization Class [collection of people, resources, groups] * People Class [roles user play to interact with the system] * Places Class [physical locations the system has info about] * Tangible things and devices [physical objects]   **Guidelines for identifying super sub relationship:** | **6** | **11** |
| 3C | A store wants to automate its inventory. It has point-of-sale terminals that can record all of the items and quantities that a customer purchases. Another terminal is also available for the customer service desk to handle returns. It has a similar terminal in the loading dock to handle arriving shipments from suppliers. The meat department and produce department have terminals to enter losses/discounts due to spoilage. Apply **Noun phrase approach** for identifying the objects from the grocery store problem. [Apply] | **10** | **20** |
| **OR** | | | |
| 4A | List the properties of a part of relationships. [Remember]   * **Two major properties of a-part-of relationship are:**   + **Transitivity**   + **Antisymmetry** | **4** | **5** |
| 4B | Explain with an example for class diagram. Summarize the relationship used in class diagram. [Understand]    **Class Diagram (UML) for Airline Reservation System**  Any example ok.  **Relationship used in class diagram with explanation** | **6** | **11** |
| 4C | When the customer inserts the bank or credit card in the ATM’s card reader, the entry action i.e read card is performed by the ATM machine. If the card is not valid then the machine will perform exit action. After the card is being read successfully, the ATM machine will ask for Pin. Then the customer enters the pin and ATM machine then reads pin. If the pin entered is not valid then machine will perform exit action. If the pin entered is valid, then the machine further process towards transaction. After successful transaction, machine undergoes the exit action i.e., eject card that discharges the customer’s card. Sketch the **use case diagram** for the above given scenario. [Apply] | **10** | **20** |
| 5A | Define the following   1. Class visibility 2. Refining attributes and its types [Remember]   **(i)Class visibility**      **(ii)Refining attributes and its types**     * The three basic types of attributes are:   + 1. Single-value attributes.   + 2. Multiplicity or multivalue attributes.   + 3. Reference to another object, or instance connection. | **4** | **5** |
| 5B | Define corollaries. Explain various levels of corollaries. [Understand]   * A *Corollary* is a proposition that follows from an axiom or another proposition that has been proven. | **6** | **11** |
| 5C | Define swimlane. Sketch the **swimlane diagram** for reserving a railway ticket. [Apply]  Swimlane simply describes who is responsible for the activities being performed in the activity diagram and how they are responsible. The activity diagram only represents the activities being performed, but Swimlane describes who does what in a process or activity performed. | **10** | **20** |
| **OR** | | | |
| 6A | Define sequence and collaboration diagram with appropriate examples. [Remember]  A [Sequence diagram](https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/) is an interaction diagram that details about the operation that is carried out. The sequence diagram captures the interaction between the objects in the context of collaboration. Sequence diagrams are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent time.  Example:    Collaboration Diagram represents the interaction of the objects to perform the behavior of a particular use case or a part of use case. The designers use the Sequence diagram and Collaboration Diagrams to define and clarify the roles of the objects that perform a particular flow of events of a use case.  **Example:**  **col**  Similarities Between Sequence and Collaboration Diagram   1. In Unified Modelling Language both the sequence diagram and collaboration diagram are used as interaction diagrams. 2. Both the diagrams details about the behavioral aspects of the system. | **4** | **5** |
| 6B | Define Axiom. Describe various types of axioms. [Understand]  **Axiom definition:**    **Types of axioms:**    **Axiom1**  **Axiom 1:** The independence axiom: Maintain the independence of components. According to axiom 1, each component must satisfy its requirements without affecting other requirements.  Eg. Let us design a refrigerator door which can provide access to food and the energy lost should be minimized when the door is opened and closed. Opening the door should be independent of losing energy.  **Axiom2** | **6** | **11** |
| 6C | Sketch a **state transition diagram and activity diagram** for a  microwave oven for the following scenario:  (a) The oven is initially in idle state with door open when light is turned on.  (b) When the door is closed it is in idle state and light is turned off.  (c) If the button is pressed it moves to initial cooking stage where the timer is set and lights are on and heating starts.  (d) At any moment the door may be opened, the cooking is interrupted, the timer is cleared and heating stops.  (e) Also, while coking another button can be pushed and extended cooking state starts where the timer gets more minutes. At any moment door can be opened here also.  (f) If the timer times out, cooking is complete, heating stops, lights are off, it sounds a beep.  (g) When the door is open, again the oven is in idle state with the door open. [Apply]      **Activity diagram for microoven** | **10** | **20** |
| 7A | Describe the activities of access layer design process. [Remember]  **Activities of access layer design process:**   * The access layer design process consists of these following activities:  1. If methods will be stored in a program then   For every business class identified, *determine if the class has*  *persistent data.*  else  For every business class identified, *mirror the business class*  *package*.   1. ***Define relationships*.** The same rule as applies among business class objects also applies among access classes. 2. ***Simplify classes and relationships*.** The main goal here is to eliminate redundant or unnecessary classes or structures.    1. ***Redundant classes*.** If you have more than one class that provides similar services, simply select one and eliminate the other(s).    2. ***Method classes*.** Revisit the classes that consist of only one or two methods to see if they can be eliminated or combined with existing classes. 3. Iterate and refine. | **4** | **5** |
| 7B | Explain the steps involved in designing view layer classes. [Understand]  **Steps involved in designing view layer classes**   * Design of the view layer classes are divided into the following activities:   – I. Macro Level UI Design Process- Identifying View Layer Objects.   * II. Micro Level UI Design Activities.   a. Design of view layer objects by applying Design Axioms and  Corollaries.  b. Create prototype of the view layer interface.   * III. Usability and User Satisfaction Testing. * IV. Refine and Iterate.   **Macro level design process of view layer classes.**   1. For Every Class Identified    1. 1 Determine If the Class Interacts With Human Actor: If yes, do next step otherwise move to next class.       1. 1.1 Identified the View (Interface) Objects for The Class.   1.1.2 Define Relationships Among the View (Interface) Objects.   1. Iterate and refine.     **Micro level design process of view layer classes.**  For Every Interface Object Identified in the Macro UI Design  Process.  1.1 Apply Micro Level UI Design Rules and Corollaries to  Develop the UI.  2. Iterate and refine | **6** | **11** |
| 7C | Illustrate various database models with suitable example. [Apply]  **Database models**           * The columns of each table are attributes that defines data or value domain for entries in that column. * The rows of each table are tuples representing individual data objects being stored.   A relational table should have only one primary key. | **10** | **20** |
| **OR** | | | |
| 8A | Describe object storage and persistence in detail. [Remember] | **4** | **5** |
| 8B | Explain the features of Object Oriented DataBase System. [Understand] | **6** | **11** |
| 8C | Illustrate macro and micro level design process of view layer classes. [Apply]  **Macro level design process of view layer classes. (4 marks)**   1. For Every Class Identified    1. 1 Determine If the Class Interacts With Human Actor: If yes, do next step otherwise move to next class.       1. 1.1 Identified the View (Interface) Objects for The Class.   1.1.2 Define Relationships Among the View (Interface) Objects.   1. Iterate and refine.     **Micro level design process of view layer classes. (4 marks)**  For Every Interface Object Identified in the Macro UI Design  Process.  1.1 Apply Micro Level UI Design Rules and Corollaries to  Develop the UI.  2. Iterate and refine | **10** | **20** |
| 9A | List the notations used in activity and state diagram. [Remember]    Activity-Diagram-Notations | **4** | **5** |
| 9B | Explain and sketch the **activity diagram** for Library management system. [Understand] | **6** | **11** |
| 9C | The Flight Reservation System is an essential tool for managing flight bookings and reservations. The system offers several features, including the ability to select flights, book flights, cancel bookings, search for flights, view flight details, add flights, edit flights, and cancel flights. The first step in the system is to select a flight. Customers can search for flights based on various criteria, such as departure and arrival locations, dates of travel, and airline preferences. This feature ensures that customers can find a flight that suits their needs and preferences.  Once customers have selected a suitable flight, they can book the flight using the system. The system allows customers to provide their personal and payment information, select the seat they want, and specify any additional services they require, such as baggage handling or in-flight meals. Customers can also specify any special requests or preferences they have, such as a window seat or a vegetarian meal. This feature ensures that customers can customize their flight according to their needs and preferences.  If customers need to cancel their booking, they can do so using the system. The system allows customers to cancel their booking quickly and easily, without the need to contact the airline directly. This feature ensures that customers have flexibility and control over their bookings and can adjust their plans as needed.  Finally, if airlines need to manage their flight schedules, they can add, edit, or cancel flights using the system. This feature ensures that airlines can manage their flight operations efficiently and effectively, and respond quickly to changes in demand or other factors.  Sketch the **Sequence diagram** for the given above scenario. [Apply] | **10** | **20** |
| **OR** | | | |
| 10A | Describe Transaction and concurrency control in brief. [Remember] | **4** | **5** |
| 10B | Explain and sketch the component diagram and deployment diagram for Hospital Management System. [Understand] | **6** | **11** |
| 10C | Explain Quality assurance Tests. Illustrate various testing strategies. [Apply]  **Quality assurance Tests**  Debugging is a process of finding out where something went wrong and correcting the code to eliminate the errors or bugs that cause unexpected results. A software debugging system can provide tools for finding errors in programs and correcting them.  **Kinds of errors:** In general, a software has three types of errors such as below  **Language (syntax) errors** are result of incorrectly constructed code, such as an incorrectly typed keyword or punctuations. They are easiest error to be detected on simple running system  **Run-time errors** are detected on running, when a statemenSSt attempts an operation that is impossible to carry out. Eg. if program tries to access a non-exist file or object, it occurs.  **Logic errors** occur when expected output is not formed. They can detected only by testing the code and analyzing the results performed by intended codes  The **elimination of syntactical bug** is the process of debugging, whereas **detection and elimination of logical bug** is the process of testing.  Quality assurance testing can be divided into two major categories: **error-based testing and scenario-based testing**.  **Error-based testing** techniques search a given class's method for particular clues of interests, then describe how these clues should be tested. E.g: Boundary condition testing.  **Scenario-based testing** also called **usage-based testing,** concentrates on capturing use -cases. Then it traces user's task, performing them with their variants as tests. It can identify interaction bugs. These are more complex tests tend to exercise multiple subsystems in a single test covering higher visibility system interaction bugs.  **Software testing** is one element of a broader topic that is often referred to as verification and validation (V&V).  **Verification** refers to set of activities that ensure that software correctly implements a specific function. **Validation** refers to different set of activities that ensure that software that has been built is traceable to customer requirements.  **Testing Strategies**  The **objective of s/w testing** is to uncover errors. The various testing strategies constitutes:  **Unit Testing-** Black Box testing, White black testing  **Integration Testing** - Top-down testing, Bottom-up testing,  Regression testing  **Validation Testing-** Alpha test, Beta test  **System Testing-** Recovery testing, Security testing, Stress  testing, Performance testing          **Top-down testing:** It assumes that main logic or object interactions and system messages of application need more testing than an individual object's method or supporting logic. This strategy can detect serious flaws early in implementation  **Bottom-Up testing:** It starts with details of system and proceeds to higher levels by a progressive aggregation of details until they collectively fit requirements for system i.e., start with methods and classes then to level up. It makes sense as it checks behavior of piece of code before it is used.  Regression testing: It is activity that helps ensure that changes (due to testing or other reasons) do not introduce unintended behavior or additional errors. The regression test suite contain three different classes of test cases:  ⇒ A representative sample of tests that will exercise all s/w functions  ⇒ Additional tests that focus on s/w functions that are likely to be affected by change  ⇒ Tests that focus on s/w components that have been changed  **Validation Testing:** After integration testing, s/w is completely assembled as a package: interfacing errors have been uncovered and corrected and final series of s/w tests - validating testing - begins. It is nothing but continuous execution on s/w by number of times & users. It is again implemented either by one of two methods: Alpha testing or Beta testing  **Alpha test** is conducted in controlled environment at developer's site by a customer. The s/w is used in setting with developer "looking over shoulder" of user, recording errors & usage problems  **Beta test** is conducted at one or more customer sites by end user(s) of the software. Its live application of s/w where customer records all problems that are encountered during beta testing and reports  **System Testing:** It is a series of different tests whose primary purpose is to fully exercise the computer - based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated and perform allocated functions. The different tests in series are Recovery, security, stress and performance testing | **10** | **20** |