Name - Piyusha Rajendra Supe. BE Computer - B.
Roll no - 23C0315

(03|10|2025)

# Assignment - OO MD.

Q1] What is design pattern ? Explain different types of design patterns?

→ ◦ It is a reusable solution or template to solve a common problem that occurs repeatedly in software design.
◦ It is not finished code but proven approach that can be adapted to fit specific situations.
◦ They help make software modular, maintainable, and easier to understand.

◦ Types of design pattern -

1. Creational patterns - deal with how objects are created They provide flexible ways of instantiating and managing objects.
   eg - ◦ Singleton - only one instance of a class is Created.
   ◦ Factory method - creates objects without specifying the exact class.
   ◦ Abstract factory - creates families of related objects.

2. Structural patterns - deal with how classes and objects are combined to form larger structures.

- Adapter - makes incompatible interfaces work together
- Bridge - separates abstraction from implementation.
- Composite - treats individual objects dynamically uniformly.
- Decorator - adds responsibilities to objects.

3. Behavioral patterns – deal with how classes and objects interact and communicate.
   - eg – ◦ Strategy - selects an algorithm at runtime.
     ◦ Observer - notifies multiple objects of state changes.
     ◦ Command – encapsulates a request as an object.
     ◦ Iterator – provides a way to traverse a collection.

Q2) State and explain entities involved in design pattern.

→ • Entities in a design pattern are the main components or participants that define how the pattern works.

• Entities include –

1. Client – the object or component that requests a service or uses the design pattern.
2. Context - stores information relevant to the pattern maintains state if needed.
3. Component / interface / Abstract class – defines the common interface definition for objects involved in the pattern.

4. Creator - responsible for creating objects (in creational patterns like factory method.)

5. Decorator / wrapper - adds additional responsibilities to objects dynamically (in structural patterns like decorator).

6. Handler - In patterns like chain of responsibility handles requests or passes them along the chain.

7. Subject / observer - in behavioural patterns, subject maintains a list of observers to notify of changes; observer defines how to respond.

Q3. Explain state design pattern.

→ ◦ The state design pattern is a behavioural design pattern that allows an object to change its behaviour when its internal state changes. It appears as if the object has changed its class.

- ◦ It encapsulates state specific behaviour in separate classes.
- ◦ Context holds a reference to the current state and delegates work to it.

- State interface defines a common contract for all states.
- Concrete state classes implement behaviour for a particular state.

- Eliminates large if-else or switch statements based on state.

- Makes it easy to add new states without modifying existing code
- Commonly used when an object's behaviour must vary dynamically at run time.

Q4) What is communication pattern ? Explain any one pattern in detail?

→
- It defines a standard way in which components, objects or systems exchange information with each other
- It focuses on how messages flow between participants, ensuring consistency and reducing complexity.
- It improves scalability, flexibility and maintainability of software systems.
- Example – Publisher / Subscriber pattern.–
- A messaging pattern where publishers send events/ messages without knowing who receives them.
- Subscribers express interest in certain message types or topics.
- A message broker/event bus manages the distribution of events to all interested subscribers.
- Decoupled senders from receivers, improving modularity
- Allows multiple subscribers to react independently to the same event.
- Commonly used event driven system, notification services, real time updates.

Q5] Write short note on - i) Client dispatcher server.
ii) Publisher subscriber.

→ i) Client dispatcher server.

- An architectural pattern used to manage requests between clients and multiple servers.
- Client sends a request for a service.
- Dispatcher receives the request and determines which server should handle it.
- Server processes the request and returns the result.
- Provides load balancing, routing, and scalability.
- Decouples clients from direct server handling.

ii) Publisher subscriber -

- A communication pattern for decoupling message senders from receivers.
- Publisher sends messages/events without knowing who receives them.
- Subscribers register interest in certain messages or topics.
- A message broker/event bus handles distribution to all subscribers.
- Enables flexible, event driven communication.

Q6] What are different categories of external control? Explain in brief.

→ External control refers to mechanisms outside a program or system that influence how it operates.

- Main categories include –

1. User control - actions taken directly by users (commands, inputs, interface operations) to control system behaviour.

2. Hardware control - external devices or hardware signals (sensors, switches, interrupts) affecting the system's functioning.

3. Software control - other software or external applications controlling or triggering processes (APIs, scripts, middleware).

4. Regulatory environmental control - external rules, standards, or operating conditions (policies, compliance requirements, network conditions) guiding system operations.

– These controls ensure system behaves predictably and meets operational safety.

---

Q7] Discuss how you identify, use cases and actors with respect to use case diagrams?

→ 1. Identifying Actors –

- Look for all entities outside the system that interact with it (people, devices, other systems).
- Actors represent roles, not specific individuals (eg. "Customer", "Admin", "Payment gateway")
- Ask - who uses the system? who provides input? who receives output?

2. Identifying use cases -

- Determine what the actors want to acheive with the system
- Each use case is a goal oriented interaction between an actor and the system (eg login, Place order, generate report)
- Ask: What functions should the system provide to each actor? What tasks must be performed ?

3. Relating actors and use cases in the diagram?

- Draw actors as stick figures outside the system boundary.
- Draw use cases as ovals inside the system boundary.
- Connect each actor to the use cases they participate in with straight lines.

Q8] Explain the tasks involved in design optimization -

→ 1. Analyze design alternatives - compare different design solutions to achieve required functionality.

2. Evaluate performance trade offs - check speed, memory usage, throughput and responsiveness for each design.

3. Minimize complexity - simplify structure, reduce number of components and avoid unnecesary dependancies.

4. Optimize resource usage - ensure efficient use of CPU, memory, storage and network resources.

5. Optimize resource usage - ensure efficient use of CPU, memory, storage and network resources.

6. Improve modularity and reusability - break design into cohesive modules and promote reuse of components.

7. Check scalability and flexibility - ensure design can handle increased load and adapt to future changes.

8. Apply design patterns and best practices - use proven solutions to common problems for efficiency and maintainability.

Q9] Explain the following terms in relation to class design - i) Refactoring.
ii) rectification.

→ i) Refactoring -

- The process of improving existing class structure code without changing its external behaviour.
- Aims to make the class design cleaner, simpler and more maintainable.
- Examples include - renaming methods, extracting methods, removing duplication or improving class heirarchy.
- Helps reduce technical debt and improves readability and reusability.

ii) Reification -

- The process of turning an abstract concept into a concrete class or object in the design.
- Used when an idea or relationship (like "Order", "subscription" or "payment") becomes important enough to model as a class.
- Makes implicit concepts explicit, enabling clearer representation and better management of system behaviour.
- Helps improve clarity, extensibility and consistency in class design.

Q10] Prepare a data dictionary for ATM system scenario Explain each element in brief.

→ ATM data dictionary is as follows -

| Data Element. | Type/format | Explanation. |
|---|---|---|
| ◦ Card-number | Numeric (16 digits). | Unique no. embossed on the ATM card. Used to identify customer's account. |
| ◦ PIN. | Numeric (4 digits). | Secret personal identification number entered by customer Used for authentication. |
| ◦ Account-Type. | Text ("Savings" / "Current") | Indicates type of account connected to the ATM Card. |

Piyusha Supe
23C0315.

Page No. 10.
Date

| | | |
|---|---|---|
| Customer_name | Text (First + Last) | Name of account holder displayed for verification or printed on receipts. |
| Transaction_id. | Alphanumeric. (CTR* 123456). | Unique identifier automatically generated for every transaction. Enables audit trails and tracking. |
| Transaction_Type | Text ("Withdraw", "Deposit", "Enquiry", "Transfer") | Specifies what kind of Transaction the customer is performing so system can follow correct process flow. |
| Transaction_date_time. | DateTime. | The exact date and time when the transaction is performed. Used for logs. |
| Amount | Decimal (10,2) | Money value involved in the transaction. Can be cash withdrawn, deposited. |