UNIT–6 : Cognitive Architecture and Object-Oriented Modelling (PYQ Solution)

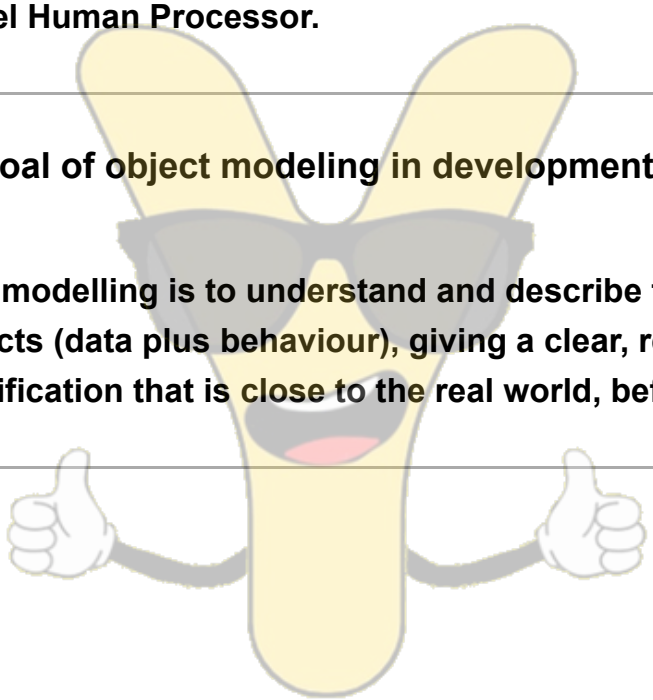
PART–A Answers (≈70 Words Each)

Q1. What is a Cognitive Architecture? (2024)

A cognitive architecture is a computational framework that models the fixed structures of the human mind (memories, processors, learning), so human behaviour on tasks can be simulated and predicted. Examples: SOAR, ACT-R, EPIC and the Model Human Processor.

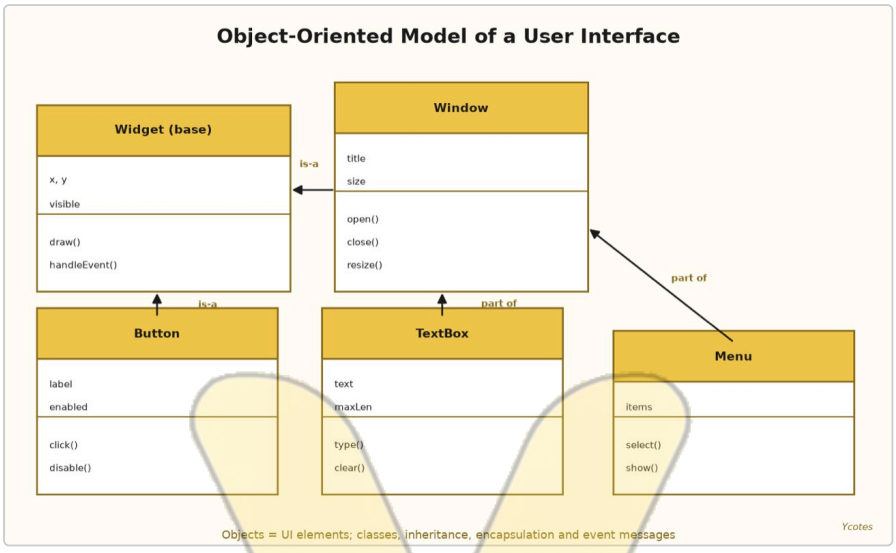
Q2. What is the goal of object modeling in development of any system? (Feb 2024)

The goal of object modelling is to understand and describe the system as a set of interacting objects (data plus behaviour), giving a clear, reusable and maintainable specification that is close to the real world, before coding starts.



PART–B Answers (≈250 Words Each)

Q1. Describe the process of applying OOM to UI design, including the identification of objects, classes, and relationships. (2024)



1. **Identify objects:** take the nouns in the requirements and screens, for example Window, Button, TextBox, Menu, Order, Customer. Both visible UI objects and domain objects are included.
2. **Define classes:** group similar objects into classes and name their **attributes** (label, enabled) and **operations** (click(), draw()).
3. **Identify relationships:**
   - **Inheritance (is-a):** Button and TextBox are kinds of Widget
   - **Association:** Order is placed by Customer
   - **Aggregation (part-of):** a Window contains TextBox, Button and Menu
4. **Model behaviour:** events, messages and states of each object (state diagram, sequence diagram)
5. **Refine:** add details, apply patterns, check against use cases
6. **Implement and evaluate** with a UI toolkit, with prototypes tested by users

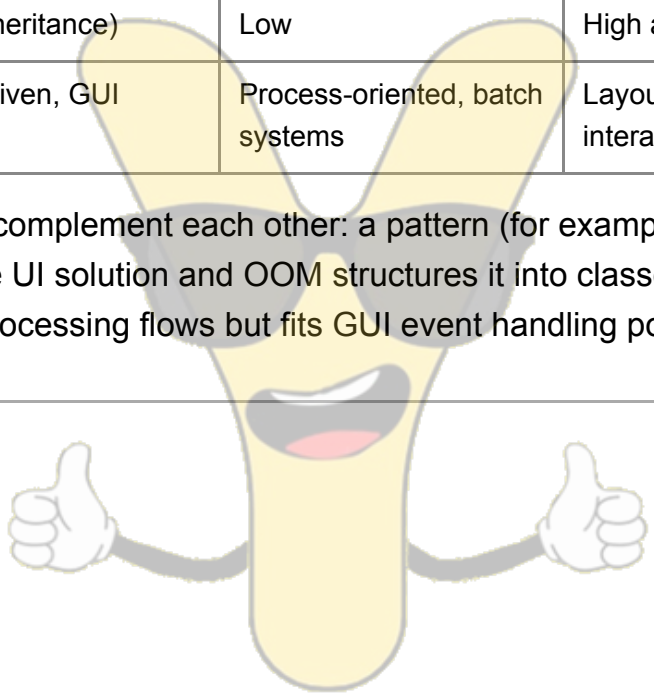
Advantages: UI elements map directly to classes, inheritance gives consistency and reuse, and events fit the message-passing model.

Q2. Compare OOM with other modeling approaches, including structured analysis and design, and human-computer interaction (HCI) design

patterns. (2024)

Point	OOM	Structured analysis and design	HCI design patterns
Focus	Objects (data + behaviour)	Functions and data flow	Proven solutions to common UI problems
Basic unit	Class and object	Process and data store	Pattern (name, problem, solution)
Diagrams	Class, state, use-case, sequence	DFD, structure chart	Pattern language, sketches
Change	Localised, easy	Ripples through functions	Reuse in similar contexts
Reuse	High (inheritance)	Low	High at design level
Best for	Event-driven, GUI systems	Process-oriented, batch systems	Layout, navigation, interaction decisions

OOM and patterns complement each other: a pattern (for example Wizard or Master-Detail) suggests the UI solution and OOM structures it into classes. Structured design suits data-processing flows but fits GUI event handling poorly.



PART–C Answers (≈500 Words Each)

Q1. Write short note on - (a) OOP introduction (b) Object oriented modeling of user interface design (2022)

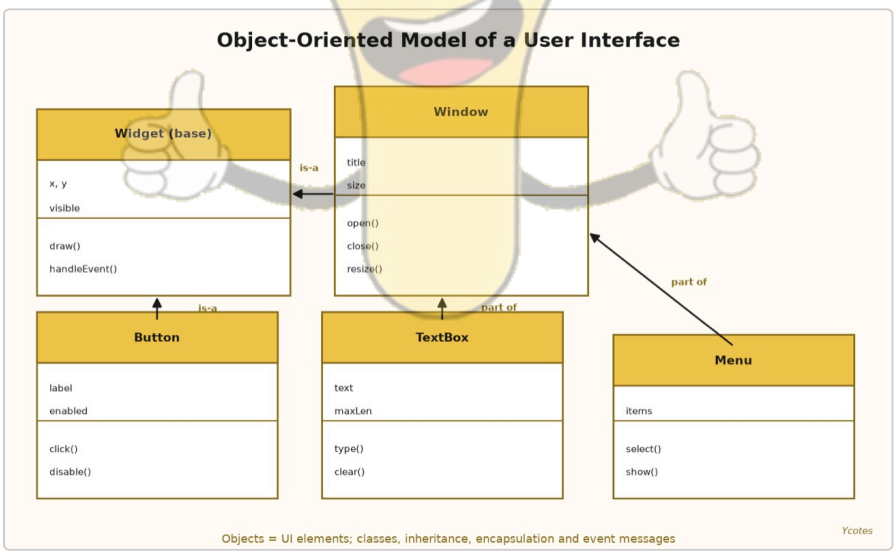
(a) OOP introduction

Object-oriented programming builds software from objects that combine data and behaviour.

- **Class** is a blueprint and **object** is an instance (Button class, OK button)
- **Encapsulation:** data and methods bundled, internal details hidden
- **Abstraction:** only essential features are shown
- **Inheritance:** a class reuses another (Button is a Widget)
- **Polymorphism:** same message, different behaviour (draw() for Button and TextBox)
- **Message passing:** objects communicate through events and method calls

(b) Object-oriented modelling of user interface design

A UI consists of objects (windows, buttons, menus) which respond to events, so OOM fits it naturally.



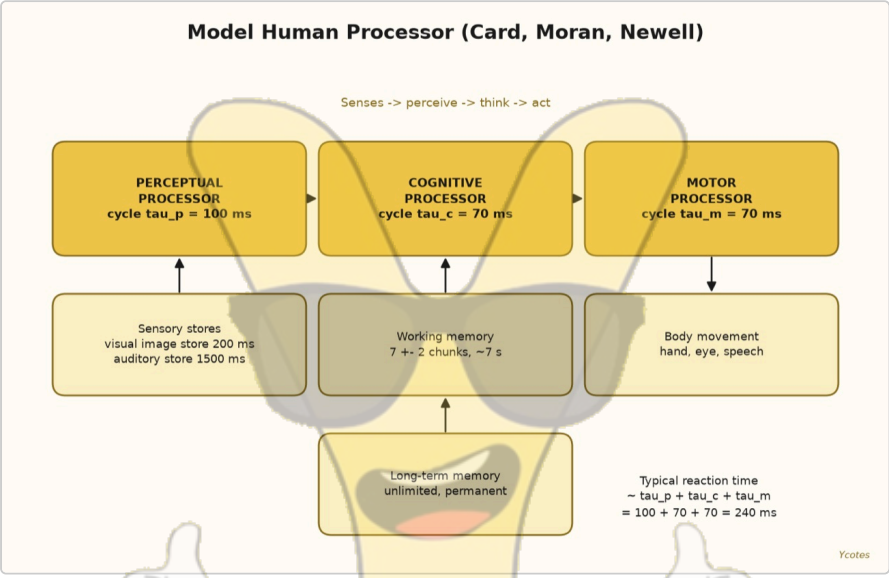
- Identify UI objects and group them into classes with a base class **Widget** (x, y, visible, draw(), handleEvent())
- Use **inheritance** (Button, TextBox, Menu are kinds of Widget) and **aggregation** (Window contains widgets)
- Model events (click, key press) as messages and object states with state diagrams

- Separate the interface objects from the application (domain) objects, as in Model-View-Controller

Benefits: consistency, reuse of components, easy extension and direct mapping to GUI toolkits.

Q2. Describe the MHP model and its components. Explain how MHP is used to model human cognition and behavior. (2024)

The **Model Human Processor** (Card, Moran, Newell 1983) models the human as an information processing system with **three processors and memories**.



Component	Parameter	Role
Perceptual processor	cycle τ_p = 100 ms (50 to 200)	Converts sensory input into stored codes
Cognitive processor	τ_c = 70 ms (25 to 170)	Recognise-act cycle: compares, decides
Motor processor	τ_m = 70 ms (30 to 100)	Carries out movement
Visual image store	decay about 200 ms	Brief sensory buffer
Auditory image store	decay about 1500 ms	Brief sensory buffer
Working memory	7 +- 2 chunks, about 7 s	Current information
Long-term memory	unlimited, permanent	Knowledge and skills

Use to model cognition and behaviour: the time for a task is the sum of the cycles it needs.

- **Simple reaction:** $T = \tau_p + \tau_c + \tau_m = 100 + 70 + 70 = 240$ ms (fast man 105 ms, slow man 470 ms)
- **Matching task** (compare two letters and answer): $\tau_p + 2\tau_c + \tau_m = 310$ ms

It also explains why working memory limits (7 +- 2 chunks) matter, and motor movement time follows Fitts' law. The three versions (Fast, Middle, Slow man) give the range of performance. MHP is the basis of GOMS and KLM.

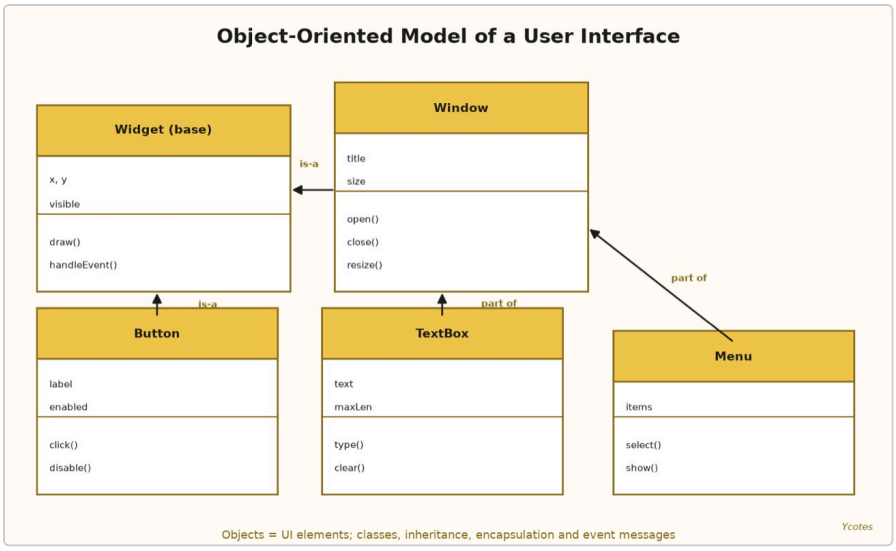
Limitations: simplified, serial view; ignores emotion and social factors.

Q3. What is OOM? Explain the types of model with their purpose in detail. (Feb 2024)

Object-oriented modelling (OOM) represents a system as interacting objects, each with data (attributes) and behaviour (operations), using diagrams to analyse and design before coding.

Types of models (Rumbaugh's OMT and UML):

Model	Shows	Diagram	Purpose
Object (class) model	Static structure: classes, attributes, operations, relationships	Class diagram	Defines what the system consists of
Dynamic model	Behaviour over time: events, states, interactions	State and sequence diagrams	Defines how objects react and interact
Functional model	Data transformation and flow	Data flow diagram	Defines what processing is done
Use-case model	User goals and system functions	Use-case diagram	Captures requirements from the user's view



Example (UI): the object model has Widget, Button, TextBox and Window; the dynamic model shows Button going from Enabled to Pressed on a click event; the functional model shows the data entered in a form being validated and saved; the use-case model has "Place order".

Benefits: realistic, reusable, easy to maintain and consistent through the whole life cycle.

