UML Diagrams - Quick Reference Guide

1. Class Diagram

Description

Shows the static structure of a system by displaying classes, their attributes, methods, and relationships between classes. It's the blueprint of the system's architecture.

When to Use

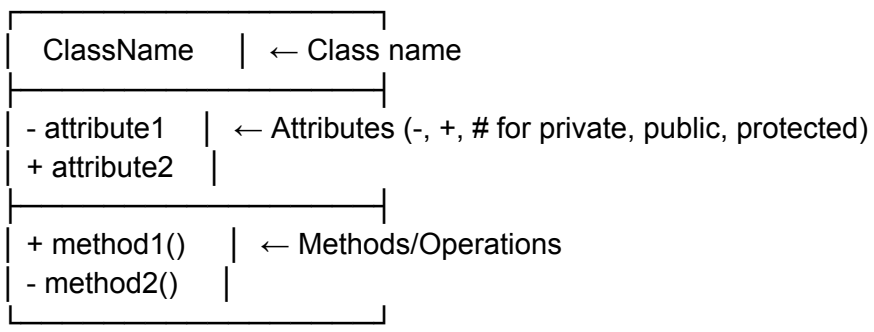
- Designing system architecture
- Showing database structure
- Documenting object-oriented designs

How to Draw

Step 1: Identify all classes (nouns in requirements)

- Example: Library System → Book, Member, Librarian, Transaction

Step 2: For each class, draw a rectangle with 3 sections:



Step 3: Draw relationships:

- **Association** (solid line): Classes are related
- **Inheritance** (arrow with hollow triangle): "is-a" relationship
- **Aggregation** (hollow diamond): "has-a" relationship (weak)
- **Composition** (filled diamond): "owns-a" relationship (strong)
- **Dependency** (dashed arrow): One class uses another

Step 4: Add multiplicity (1, *, 0..1, 1..*)

Example Steps for "Online Shopping":

1. Classes: Customer, Order, Product, Payment, Cart
2. Customer "has many" Orders (1 to *)
3. Order "contains many" Products (* to *)
4. Order "requires one" Payment (1 to 1)
5. Customer "has one" Cart (1 to 1)

---

2. Object Diagram

Description

Shows a snapshot of objects (instances of classes) and their relationships at a specific moment in time. It's like a photograph of the system's state.

When to Use

- Demonstrating real examples of class relationships
- Testing class diagram accuracy
- Showing system state at a specific time

How to Draw

Step 1: Identify specific object instances from scenario

- Example: "John's order #123 for 2 books"

Step 2: Draw rectangles with 2 sections:

```
┌──────────────────────────────┐
│ objectName:Class             │ ← Underlined object name and class
├──────────────────────────────┤
│ attribute1 = value1          │ ← Actual values, not types
│ attribute2 = value2          │
└──────────────────────────────┘
```

Step 3: Connect objects with solid lines showing actual relationships

Step 4: Label the links if needed

Example Steps for "Library on Jan 15, 2025":

1. Objects: john:Member, book1:Book, loan1:Transaction
2. john:Member with values: memberID=101, name="John Doe"
3. book1:Book with values: ISBN="123", title="Java Guide", status="borrowed"
4. loan1:Transaction with values: loanID=501, date="2025-01-15"
5. Connect: john borrowed book1 through loan1

---

3. Sequence Diagram

Description

Shows how objects interact with each other over time through messages. It focuses on the time ordering of messages exchanged between objects.

When to Use

- Documenting specific use case flows
- Showing method calls between objects
- Understanding message flow and timing

How to Draw

Step 1: Identify actors and objects involved in the scenario

- Example: "User login process" → User, LoginUI, AuthController, Database

Step 2: Draw participants as boxes at the top (left to right)

Step 3: Draw vertical dashed lines (lifelines) below each participant

Step 4: Draw horizontal arrows (messages) between lifelines:

- **Solid arrow** → synchronous call (waits for response)
- **Dashed arrow** → return message
- **Filled arrowhead** → synchronous
- **Stick arrowhead** → asynchronous

Step 5: Draw activation boxes (rectangles on lifelines) when object is active

Step 6: Number messages in sequence (optional)

Example Steps for "ATM Withdrawal":

1. Participants: Customer, ATM, Bank, Database
2. Customer → ATM: Insert card
3. ATM → Bank: Verify card
4. Bank → Database: Check account
5. Database → Bank: Account details (dashed return)
6. Bank → ATM: Card valid (dashed return)
7. ATM → Customer: Request PIN
8. Customer → ATM: Enter PIN
9. Continue until cash dispensed

---

4. Activity Diagram

Description

Shows the workflow of a process or algorithm with sequential and parallel activities. It's like a flowchart but more powerful, showing concurrent flows.

When to Use

- Modeling business processes
- Showing algorithm logic
- Documenting workflows with decision points and parallel tasks

How to Draw

Step 1: Identify start and end points

- Use filled circle (●) for start
- Use filled circle with border (⊙) for end

Step 2: List all activities (action boxes with rounded corners)

Step 3: Add decision points (diamonds) for conditions

Step 4: Connect with arrows showing flow

Step 5: Add swim lanes if multiple actors involved (vertical columns)

Step 6: Use fork/join bars for parallel activities:

- **Fork** (thick horizontal bar): Split into parallel flows
- **Join** (thick horizontal bar): Merge parallel flows

Example Steps for "Online Order Processing":

1. Start (●)
2. Activity: Receive Order
3. Decision: ◇ Payment Valid?
4. If Yes → Activity: Process Payment
5. If No → Activity: Notify Customer → End
6. Fork: Parallel activities
    - Update Inventory
    - Generate Invoice
    - Notify Warehouse
7. Join: Wait for all parallel tasks
8. Activity: Ship Order
9. End (◉)

With Swim Lanes:

- Customer lane: Place Order, Receive Confirmation
- System lane: Validate, Process
- Warehouse lane: Pack, Ship

---

5. Use Case Diagram

Description

Shows the functional requirements of a system from the user's perspective. It displays actors (users) and use cases (functionalities) they can perform.

When to Use

- Gathering requirements
- Showing system scope and boundaries
- Communicating with stakeholders (non-technical)

How to Draw

Step 1: Identify actors (stick figures outside system)

- Who uses the system? User, Admin, External System

Step 2: Draw system boundary (large rectangle)

Step 3: Identify use cases (ovals inside boundary)

- What can actors do? Login, Create Account, View Report

Step 4: Connect actors to use cases with solid lines

Step 5: Add relationships between use cases:

- **<<include>>** (dashed arrow): One use case always includes another
  - Example: "Place Order" includes "Validate Payment"
- **<<extend>>** (dashed arrow): One use case optionally extends another
  - Example: "Login" extends to "Reset Password"
- **Generalization** (solid arrow with hollow triangle): Inheritance between actors or use cases

Example Steps for "Banking System":

1. Actors: Customer, Bank Manager, ATM System
2. System boundary: "Online Banking"
3. Use cases:
   - Customer: Login, Check Balance, Transfer Money, Pay Bills
   - Manager: Generate Reports, Manage Accounts
   - ATM System: Withdraw Cash
4. Relationships:
   - Transfer Money <<include>> Verify Account
   - Login <<extend>> Forgot Password

---

6. State Machine Diagram

Description

Shows the different states an object can be in and how it transitions between states based on events. It models the lifecycle of an object.

When to Use

- Modeling object behavior over time
- Showing status workflows
- Designing state-dependent systems (order processing, user lifecycle)

How to Draw

Step 1: Identify all possible states of the object

- Example: Order → New, Processing, Shipped, Delivered, Cancelled

Step 2: Draw start state (filled circle ●)

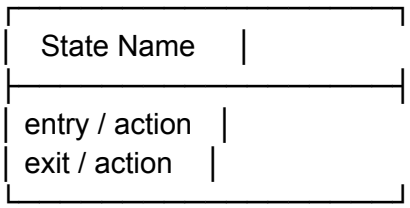
Step 3: Draw end state (filled circle with border ⊙)

Step 4: Draw state boxes (rounded rectangles) with state names

Step 5: Add transitions (arrows between states) with labels:

- Format: `event [condition] / action`
- Example: `Submit Order [payment valid] / send confirmation`

Step 6: Add entry/exit actions inside states if needed:



Example Steps for "Student Registration":

1. Start (●)
2. State: Unregistered
3. Transition: Submit Application → State: Pending Review
4. Transition: Approve → State: Registered
5. Transition: Reject → State: Rejected
6. From Registered:
    - Transition: Start Semester → State: Active
    - Transition: Drop Out → State: Inactive
7. From Active:
    - Transition: Graduate → State: Graduated → End (⊙)
    - Transition: Fail → State: Suspended
8. From Suspended: Can return to Active or go to Inactive

Key Elements:

- **Self-transition**: Arrow that loops back to same state
- **Guard conditions**: [balance > 0] on transitions
- **Actions**: / sendEmail after transition

Quick Selection Guide

Need to show...	Use this diagram
System structure and classes	Class Diagram
Real examples with actual data	Object Diagram
Message flow between objects	Sequence Diagram
Business process or workflow	Activity Diagram
System features and users	Use Case Diagram
Object lifecycle and status changes	State Machine Diagram

General Tips for All Diagrams

1. **Keep it simple** - Don't include everything, focus on what's relevant
2. **Use clear names** - Verbs for actions, nouns for objects
3. **Be consistent** - Use same naming conventions throughout
4. **Add notes** - Clarify complex parts with text annotations
5. **Review** - Check if non-technical people can understand (for use case)
6. **Iterate** - Start simple, add details gradually
7. **Use tools** - Draw.io, Lucidchart, PlantUML, Mermaid for clean diagrams